Aplicación asimétrica móvil de asistencia para familias con personas afectadas de Alzheimer

Ricardo Soto Estévez

Trabajo Fin de Grado

Director: Dr. Miguel Sánchez Santillán

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

6 de febrero de 2022

Declaración responsable

El alumno: Ricardo Soto Estévez
Con DNI: [redacted]
Y UO: [redacted]

DECLARA
Que esta obra es completamente original y se han citado debidamente las fuentes utilizadas durante la realización de esta.
Y para que conste, lo firma en Uviéu, a 6 de febrero de 2022

Firmado:

Agradecimientos

Gracias mamá. **Gracias Álvaro**. Esos cafés de media tarde me daban la vida mientras hacía este trabajo.

Gracias An, por ser mi mayor apoyo día tras día, minuto tras minuto, segundo tras segundo.

Muchas gracias también a **mis tíos**, por la segunda oportunidad que me trajo aquí.

Gracias a **Miguel** por tantas dudas resueltas.

Gracias **Círculo** por tantas horas de clase, estudio y de rajar juntos entre risas.

Gracias **Manitos** por el pedazo de hogar que me dabais en cada llamada.

Gracias a los **Mendrugos** por todas esas tardes de juntarnos y recordar anécdotas.

Gracias a los **profesores** que me han hecho amar esta especialidad.

Y gracias Brandon, Eiichiro, Queen, Team Cherry, Hideaki, Mobius Digital, Elton y Hidetaka, por todas esas obras que han definido quien soy y me han aportado tantos refugios de felicidad.

Y sobre todo, y por encima de todas las cosas, **gracias papá**. Llego tarde, lo sé y lo lamento en el alma. Este trabajo existe por y para ti, no has escapado a mi memoria en ninguno de los días que he trabajado en esto, si hay una voluntad que me ha ayudado a seguir es la que tú me inculcaste. No llegaste a verme llegar aquí, pero esto va por ti, que este proyecto pueda ayudar a familias que vivimos lo mismo que nosotros. **Te echo de menos, va por ti**.

Gracias

Resumen

La **enfermedad de Alzheimer** es una patología que exige mucho del paciente de la misma y de su familia y entorno cercano para ser afrontada. Ayudar al damnificado es clave y el impulso de la colaboración que lo logre es la meta de este proyecto.

All For One es un sistema compuesto de una aplicación móvil y un servidor en la nube comunicados por WebSockets y por una API REST. La aplicación está desarrollada en Kotlin y el servidor usa Node y se aloja en Azure. La interfaz HTTP está servida utilizando Express y la WebSocket se maneja con Socket.IO.

Este sistema persigue facilitar la colaboración de sus pacientes y cuidadores con funcionalidades como chats en tiempo real, geolocalización o gestión de tareas. De esta forma se busca reducir en la medida de lo posible las dificultades planteadas por la enfermedad.

Palabras clave

All For One, asistencia, enfermedad de Alzheimer, paciente, cuidador, vínculo, aplicación móvil, geolocalización, tareas

Abstract

Alzheimer's disease is a very demanding pathology for all: the patient, their whole family and their close environment. Helping the affected is key, and boosting the collaboration needed to achieve this is the goal of this project.

All For One is a system composed by a mobile application and a cloud server communicating through WbeSockets and a REST API. The application is developed with Kotlin and the server uses Node and is being hosted in Azure. The HTTP interface is served using Express and the WebSocket one is managed with Socket.IO.

This system wants to make easy the collaboration between its patients and keepers with features such as real-time chats, geolocation and task management. This way we plan to reduce as much as possible the difficulties thrown by this pathology.

Keywords

All For One, assistance, Alzheimer's Disease, patient, keeper, bond, mobile application, geolocation, tasks

Índice de contenido

Índice de contenido **9**

Índice de figuras **17**

Índice de tablas **20**

Índice de fragmentos **30**

I Presentación **32**

1 Memoria del proyecto **33**
1.1 Resumen de la motivación, objetivos y alcance del proyecto 33
1.2 Estructura del documento 34

2 Introducción **37**
2.1 Justificación del proyecto 37
2.2 Objetivos del proyecto 39
2.3 Estudio de la situación actual 40
2.3.1 Evaluación de alternativas 42

3 Aspectos teóricos **46**
3.1 API REST 46
3.2 WebSocket 47
3.3 NoSQL 47

II Planificación **49**

4 Requisitos iniciales **50**

5 Evaluación de alternativas **53**
5.1 Desarrollo de la aplicación móvil en Android 53
5.1.1 Desarrollo nativo en Java 53
5.1.2 Desarrollo nativo en Kotlin 53
5.1.3 Desarrollo con el framework React Native 53
5.2 Desarrollo de la API 54
5.2.1 Spring Framework 54
5.2.2 Node.js con Express y Socket.io 54

5.2.3 Micronaut . . . 55
5.3 Tecnología de geolocalización . . . 55
5.3.1 OsmDroid . . . 55
5.3.2 Google Maps for Android . . . 56
5.4 Sistema de gestión de bases de datos . . . 56
5.4.1 MongoDB . . . 56
5.4.2 Firebase . . . 57
5.5 Nube para el despliegue del servidor . . . 57
5.5.1 Amazon Web Services . . . 57
5.5.2 Microsoft Azure . . . 57
5.5.3 Heroku . . . 58

6 Análisis de riesgos 59
6.1 Descripción de riesgos detectados . . . 59
6.1.1 Incapacidad de obtención de colaboración con asociaciones . . . 59
6.1.2 Inexperiencia del equipo de proyecto . . . 59
6.1.3 Carencia de fondos . . . 60
6.1.4 Falta de formación en tecnologías clave . . . 60
6.2 Análisis de los riesgos . . . 61
6.3 Respuesta a los riesgos . . . 61

7 Descripción de la solución propuesta 62
7.1 Aplicación móvil . . . 62
7.1.1 Autenticación . . . 63
7.1.2 Geolocalización . . . 63
7.1.3 Comunicación con la API . . . 63
7.1.4 Vinculación de usuarios . . . 63
7.2 API . . . 64
7.2.1 Autenticación . . . 64
7.2.2 Persistencia de datos . . . 64
7.3 Despliegue . . . 65

8 Planificación temporal 66

9 Resumen del presupuesto 71

III Análisis 72

10 Requisitos del sistema 73
10.1 Identificación de actores del sistema . . . 73
10.1.1 Paciente . . . 73
10.1.2 Cuidador . . . 73
10.1.3 Administración . . . 73
10.1.4 Identificación de relación de actores . . . 73
10.2 Identificación de requisitos . . . 74
10.2.1 RSU. Requisitos de sesión de usuario . . . 74
10.2.2 RGU. Requisitos de gestión de usuarios . . . 75

10.2.3 RGT. Requisitos de gestión de tareas . . . 77
10.2.4 RSV. Requisitos de servicio de mensajería . . . 79
10.2.5 RSG. Requisitos del sistema de geolocalización . . . 81
10.2.6 RSN: Requisitos del sistema de notificaciones . . . 83
10.2.7 RNF: Requisitos no funcionales . . . 83
10.2.8 Valores de requisitos . . . 84

11 Identificación de subsistemas 85
11.1 Aplicación móvil . . . 85
11.1.1 Actividades . . . 85
11.1.2 Modelo . . . 86
11.1.3 Repositorios . . . 86
11.1.4 Servicios . . . 86
11.2 API . . . 86
11.2.1 Bases de datos . . . 86
11.2.2 Manejador HTTP REST . . . 87
11.2.3 Manejador WebSocket . . . 87
11.3 Comunicación entre subsistemas . . . 87
11.3.1 Subsistemas internos de la aplicación móvil . . . 87
11.3.2 Subsistemas internos de la API . . . 88
11.3.3 Subsistemas API y aplicación móvil . . . 89

12 Especificación de casos de uso 90
12.1 CU1. Registro . . . 90
12.2 CU2. Iniciar sesión . . . 90
12.3 CU3. Vincularse con otro usuario . . . 91
12.4 CU4. Compartir ubicación . . . 93
12.5 CU5. Gestionar tareas . . . 94
12.6 CU6. Comunicarse por mensaje instantáneo . . . 96
12.7 CU7. Consultar información de asociados . . . 97
12.8 CU8. Consultar notificaciones . . . 98
12.9 CU9. Cerrar sesión . . . 99

13 Análisis de Interfaz de Usuario 100
13.1 Pantalla de inicio de sesión . . . 100
13.2 Pantalla de creación de perfil . . . 101
13.3 Pantalla principal . . . 101
13.4 Pantalla de vínculos . . . 102
13.5 Pantalla del feed . . . 102
13.6 Pantalla de geolocalización . . . 103
13.7 Pantalla de gestión de tareas . . . 104
13.8 Pantalla de notificaciones . . . 104
13.9 Mapa de navegación . . . 104

14 Diagrama de clases 105
14.1 Clases de la aplicación . . . 105
14.1.1 UI . . . 106
14.1.2 Repositories . . . 109

14.1.3 API . . . 112
14.1.4 VO . . . 114
14.2 Clases de la API . . . 117
14.2.1 Root . . . 117
14.2.2 Routes . . . 118
14.2.3 Sockets . . . 119
14.2.4 Services . . . 120
14.2.5 Models . . . 122

15 Especificación del plan de pruebas 125
15.1 Pruebas unitarias . . . 125
15.2 Pruebas de integración . . . 125
15.3 Pruebas de sistema . . . 126
15.4 Pruebas de usabilidad . . . 126

16 Especificación del plan de despliegue 127
16.1 Despliegue de la API . . . 127
16.2 Aplicación . . . 127

IV Diseño 128

17 Arquitectura del sistema 129
17.1 Diagrama de paquetes . . . 129
17.1.1 API . . . 129
17.1.2 Aplicación móvil . . . 130
17.2 Diagrama de despliegue . . . 130
17.3 Diagrama de componentes . . . 132

18 Diseño de clases 133
18.1 Clases de la API . . . 133
18.1.1 Root . . . 134
18.1.2 Routes . . . 134
18.1.3 Sockets . . . 136
18.1.4 Services . . . 138
18.1.5 Models . . . 140
18.1.6 Shared . . . 143
18.2 Clases de la aplicación móvil . . . 145
18.2.1 UI . . . 146
18.2.2 Repositories . . . 156
18.2.3 API . . . 161
18.2.4 VO . . . 163

19 Modelo de datos 168
19.1 Mensaje . . . 168
19.1.1 BaseMessage . . . 169
19.1.2 TaskMessage . . . 169
19.1.3 TextMessage . . . 170

19.2 Notificación . . . 170
19.3 Sesión . . . 171
19.4 Usuario . . . 171
19.4.1 Address . . . 172

20 Interfaces de comunicación **173**
20.1 API REST . . . 173
20.1.1 Autenticación . . . 173
20.1.2 Feed . . . 175
20.1.3 Tareas . . . 176
20.1.4 Usuarios . . . 179
20.2 API del WebSocket . . . 182
20.2.1 Feed . . . 183
20.2.2 Localización . . . 184
20.2.3 Notificación . . . 185

21 Interacción y estados **188**
21.1 Inicio de sesión de usuarios existentes y nuevos . . . 188
21.2 Vinculación . . . 191
21.2.1 Creación de vínculos . . . 191
21.3 Ubicación . . . 192
21.4 Tareas y mensajes . . . 194
21.4.1 Envío de un mensaje . . . 194
21.4.2 Creación de nueva tarea . . . 194
21.4.3 Estado de una tarea . . . 195
21.5 Notificaciones . . . 196
21.5.1 Envío de notificaciones . . . 196

22 Interfaz de usuario **198**
22.1 Guía de estilo . . . 198
22.1.1 Tema de color . . . 198
22.1.2 Icono de la aplicación . . . 199
22.2 Logo de la aplicación . . . 200
22.3 Launch . . . 200
22.4 Set Up . . . 201
22.5 Main . . . 202
22.6 Feed . . . 203
22.7 Tasks . . . 204
22.8 Location . . . 205
22.9 Bonds . . . 205
22.10 Scanner . . . 206
22.11 Notifications . . . 206
22.12 Settings . . . 207
22.13 Mapa de navegación . . . 207

23 Especificación técnica del plan de pruebas **209**
23.1 Pruebas unitarias . . . 209
23.1.1 Aplicación móvil . . . 209

23.1.2 Modelo de vista de lanzamiento . . . 212
23.1.3 API . . . 223
23.2 Pruebas de integración . . . 234
23.2.1 Aplicación móvil . . . 234
23.2.2 API . . . 242
23.3 Pruebas de usabilidad y accesibilidad . . . 252

V Implementación 256

24 Tecnologías 257
24.1 Lenguajes y entornos . . . 257
24.1.1 LaTeX . . . 257
24.1.2 Kotlin . . . 257
24.1.3 Node . . . 257
24.1.4 TypeScript . . . 258
24.2 Librerías . . . 258
24.2.1 Bibliotecas Android . . . 258
24.2.2 Paquetes de Node . . . 263
24.3 Sistemas operativos . . . 269
24.3.1 Android . . . 269
24.3.2 Fedora Workstation . . . 270
24.3.3 Windows . . . 271

25 Herramientas de desarrollo 272
25.1 Entornos de desarrollo . . . 272
25.1.1 Android Studio . . . 272
25.1.2 Overleaf . . . 272
25.1.3 Vim . . . 272
25.1.4 Visual Studio Code . . . 273
25.2 Herramientas . . . 273
25.2.1 Git . . . 273
25.2.2 GitHub Actions . . . 273
25.2.3 Insomnia . . . 274
25.2.4 Mendeley . . . 274
25.2.5 PlantUML . . . 274
25.2.6 ProjectLibre . . . 275
25.3 Referencias . . . 275
25.3.1 Guías para desarrolladores de Android . . . 275
25.3.2 Guía de estilo Material Design . . . 275
25.4 Sitios web . . . 276
25.4.1 Color Tool . . . 276
25.4.2 Diagrams.net . . . 276
25.4.3 GitHub . . . 276
25.4.4 Portal de Azure . . . 276
25.4.5 Portal de MongoDB Atlas . . . 277

26 Obstáculos 278

26.1 Comunicación cifrada . . . 278
26.2 Ejecución en dispositivos móviles antes del despliegue . . . 278
26.3 Despliegue del servidor . . . 279

27 Desarrollo de pruebas 280
27.1 Pruebas unitarias . . . 280
27.1.1 API . . . 280
27.1.2 Aplicación móvil . . . 282
27.2 Pruebas de integración . . . 284
27.2.1 API . . . 284
27.2.2 Aplicación móvil . . . 286
27.3 Pruebas usabilidad y accesibilidad . . . 287

VI Manuales 288

28 Manual de instalación 289

29 Manual de usuario 290
29.1 Inicio de sesión y registro . . . 290
29.2 Cierre de sesión . . . 291
29.3 Vinculación . . . 292
29.3.1 Paciente - Generar código de vinculación . . . 292
29.3.2 Cuidador - Escanear código de vinculación . . . 293
29.3.3 Consultar vínculos . . . 293
29.3.4 Eliminar vínculos . . . 294
29.4 Geolocalización . . . 295
29.5 Enviar mensajes . . . 296
29.6 Gestionar tareas . . . 297
29.6.1 Crear tarea . . . 297
29.6.2 Marcar y desmarcar tareas . . . 299
29.6.3 Eliminar tareas . . . 299
29.7 Consultar notificaciones . . . 299
29.8 Editar datos . . . 300

30 Manual de despliegue 301
30.1 Creación del servicio . . . 301
30.2 Configuración de la aplicación . . . 302
30.3 Configuración de MongoDB Atlas . . . 303
30.4 Integración continua . . . 303

31 Manual de desarrollador 304
31.1 Despliegue en local . . . 304
31.1.1 Despliegue de la API . . . 304
31.1.2 Despliegue de la aplicación . . . 305
31.2 Lanzamientos de las pruebas . . . 305

VII Conclusión 307

32 Estado del sistema 308

33 Conclusiones personales 309

34 Ampliaciones 311
34.1 Avenencias legales . . . 311
34.2 Edición de tareas y mensajes . . . 311
34.3 Filtros y búsqueda . . . 311
34.4 Imágenes de usuario . . . 312
34.5 Parametrización de la REST API . . . 312

VIII Anexos 313

35 Presupuesto 314

36 Licencia 321

37 Modelado de datos 339
37.1 Esquemas de la base de datos . . . 339
37.1.1 MessageSchema . . . 339
37.1.2 TaskMessageSchema . . . 340
37.1.3 TextMessageSchema . . . 340
37.1.4 NotificacionSchema . . . 340
37.1.5 SessionSchema . . . 341
37.1.6 UserSchema . . . 342
37.2 Tipos auxiliares y DTOs . . . 343
37.2.1 Message . . . 343
37.2.2 TaskMinDto . . . 343
37.2.3 TaskDto . . . 344
37.2.4 NotificationDto . . . 344
37.2.5 SessionDto . . . 344
37.2.6 UserMinDto . . . 345
37.2.7 UserPublicDto . . . 345
37.2.8 UserDto . . . 345

38 Acción de despliegue 346

39 Documentación adicional 348

Glosario 349

Siglas 350

Bibliografía 351

Índice de figuras

2.1 Mapa de Life365 . . . 41
2.2 Mapa de GeoZilla . . . 41
2.3 Icono de Elderly Care . . . 42
2.4 Capturas de pantalla de la página web Elderly Care . . . 43
2.5 Icono de Tú y Nosotros . . . 44
2.6 Ficha de contacto de Tú y Nosotros . . . 45
2.7 Juego de memoria de Tú y Nosotros . . . 45

7.1 Reparto de mercado de Android (nov 20 - nov 21) . . . 62

8.1 Diagrama de Gantt del proyecto . . . 70

11.1 Arquitectura y comunicación de la aplicación móvil . . . 88
11.2 Arquitectura y comunicación de la API . . . 89
11.3 Comunicación entre la aplicación y la API . . . 89

12.1 Diagrama de casos de uso generales . . . 91

13.1 Diseño inicial de LaunchActivity . . . 100
13.2 Diseño inicial de SetUpActivity . . . 100
13.3 Diseño inicial de MainActivity . . . 101
13.4 Diseño inicial de BondsActivity . . . 101
13.5 Diseño inicial de FeedActivity . . . 102
13.6 Diseño inicial de LocationActivity . . . 102
13.7 Diseño inicial de TasksActivity . . . 103
13.8 Diseño inicial de NotificationDialog . . . 103
13.9 Propuesta de mapa de navegación . . . 104

14.1 Diagrama de clases de la Aplicación . . . 105
14.2 Diagrama de clases de la API . . . 117

17.1 Diagrama de paquetes de la API . . . 129
17.2 Diagrama de paquetes de la aplicación móvil . . . 130
17.3 Diagrama de despliegue del sistema . . . 131
17.4 Diagrama de componentes del sistema . . . 131

18.1 Primera parte del diagrama de clases de la API . . . 133
18.2 Segunda parte del diagrama de clases de la API . . . 133
18.3 Tercera parte del diagrama de clases de la API . . . 133
18.4 Primera parte del diagrama de clases de la aplicación móvil . . . 145
18.5 Segunda parte del diagrama de clases de la aplicación móvil . . . 145

19.1 Diagrama del modelo de datos 168

21.1 Diagrama de actividades del inicio de sesión 189
21.2 Diagrama de secuencia del inicio de sesión 190
21.3 Diagrama de secuencia de registro 190
21.4 Diagrama de actividades de la vinculación de usuarios 191
21.5 Diagrama de secuencia de la vinculación de usuarios 192
21.6 Diagrama de actividades de la difusión de localizaciones 193
21.7 Diagrama de secuencia de la difusión de localizaciones 193
21.8 Diagrama de secuencia del envío de un mensaje 194
21.9 Diagrama de secuencia de la creación de una tarea 195
21.10 Diagrama de estado de una tarea 196
21.11 Diagrama de secuencia del envío de notificaciones 197

22.1 Icono de Paciente 199
22.2 Icono de Cuidador 199
22.3 Icono de AllForOne 199
22.4 Logo de la aplicación 200
22.5 Diseño de la pantalla de lanzamiento 200
22.6 Diseño de la pantalla de configuración de nombre 201
22.7 Diseño de la pantalla de elección de rol 201
22.8 Diseño de la pantalla de introducción de datos 201
22.9 Diseño de la pantalla de confirmación del perfil 201
22.10 Diseño de la pantalla principal 202
22.11 Diseño de la pantalla de feed 203
22.12 Diseño de la pantalla de tareas 204
22.13 Diseño de la pantalla de crear tarea 204
22.14 Diseño de la pantalla de geolocalización 205
22.15 Diseño de la pantalla de vínculos 205
22.16 Diseño de la pantalla principal sin vínculo 206
22.17 Diseño de la pantalla de escáner 206
22.18 Diseño de la pantalla de notificaciones 207
22.19 Diseño de la pantalla de ajustes 207
22.20 Mapa de navegación 208

24.1 Logo de LaTeX 257
24.2 Logo de Kotlin 257
24.3 Logo de Node.js 258
24.4 Logo de TypeScript 258
24.5 Logo de Material 260
24.6 Logo de Play Services 261
24.7 Logo de Google Maps 262
24.8 Logo de ESLint 264
24.9 Logo de Jest 265
24.10 Escritorio de Fedora Workstation 270

25.1 Logo de Vim 272
25.2 Logo de Visual Studio Code 273

25.3 Logo de GitHub 276
25.4 Logo de MongoDB Atlas 277

28.1 Descarga de la APK desde el repositorio 289

29.1 Guía de inicio de sesión. 290
29.2 Guía de configuración del perfil 291
29.3 Guía de generación de código QR de vinculación 292
29.4 Guía de escaneo del código QR de vinculación 293
29.5 Guía de consulta de vínculos 294
29.6 Guía de eliminación de vínculos 294
29.7 Guía de uso de geolocalización 295
29.8 Guía de envío de mensajes 296
29.9 Guía de acceso a Tareas 297
29.10 Guía de creación de tareas a través de Tareas 298
29.11 Guía de creación de tareas a través de Feed 298
29.12 Guía de consulta de notificaciones 299
29.13 Guía de edición del perfil 300

Índice de tablas

6.1 Probabilidades y impacto de los riesgos 61

8.1 Planificación temporal global 66
8.2 Planificación temporal del arranque de proyecto 67
8.3 Planificación temporal de la planificación del proyecto 67
8.4 Planificación temporal del análisis del sistema 67
8.5 Planificación temporal del diseño del sistema 68
8.6 Planificación temporal de la implementación del sistema 68
8.7 Planificación temporal de la elaboración de manuales 69
8.8 Planificación temporal de la clausura del proyecto 69

9.1 Resumen de presupuesto 71

10.1 Valores especificados en requisitos 84

12.1 Especificación del CU1. Registro 90
12.2 Especificación del CU2. Iniciar sesión 90
12.3 Especificación del CU3.1. Vincular un Cuidador 91
12.4 Especificación del CU3.2. Vincular un Paciente 92
12.5 Especificación del CU3.3. Desvincularse 92
12.6 Especificación del CU4.1. Compartir ubicación del usuario 93
12.7 Especificación del CU4.2. Ver ubicaciones de usuarios asociados . . . 93
12.8 Especificación del CU5.1. Listar tareas 94
12.9 Especificación del CU5.2. Crear tarea 94
12.10 Especificación del CU5.3. Marcar tarea como hecha 95
12.11 Especificación del CU5.4. Desmarcar tarea hecha 95
12.12 Especificación del CU5.5. Eliminar tarea 96
12.13 Especificación del CU6.1. Ver mensajes 96
12.14 Especificación del CU6.2. Enviar mensajes 97
12.15 Especificación del CU7.1. Consultar Cuidadores asociados 97
12.16 Especificación del CU7.2. Consultar Paciente asociado 98
12.17 Especificación del CU8.1 Consultar notificaciones no leídas 98
12.18 Especificación del CU8.2 Marcar notificaciones como leídas 99
12.19 Especificación del CU9. Cerrar sesión 99

14.1 Especificación de la clase LaunchActivity 106
14.2 Especificación de la clase LaunchViewModel 106
14.3 Especificación de la clase SetUpActivity 106
14.4 Especificación de la clase SetUpViewModel 106
14.5 Especificación de la clase MainActivity 107

14.6 Especificación de la clase MainViewModel . . . 107
14.7 Especificación de la clase BondsActivity . . . 107
14.8 Especificación de la clase BondsViewModel . . . 108
14.9 Especificación de la clase FeedActivity . . . 108
14.10 Especificación de la clase FeedViewModel . . . 108
14.11 Especificación de la clase LocationActivity . . . 108
14.12 Especificación de la clase LocationViewModel . . . 109
14.13 Especificación de la clase TasksActivity . . . 109
14.14 Especificación de la clase TasksViewModel . . . 109
14.15 Especificación de la clase RepositoryFactory . . . 109
14.16 Especificación de la clase FeedRepository . . . 110
14.17 Especificación de la clase GlobalRoomRepository . . . 110
14.18 Especificación de la clase LocationRepository . . . 110
14.19 Especificación de la clase NotificationRepository . . . 111
14.20 Especificación de la clase SessionRepository . . . 111
14.21 Especificación de la clase TasksRepository . . . 111
14.22 Especificación de la clase UserRepository . . . 112
14.23 Especificación de la clase ApiFactory . . . 112
14.24 Especificación de la clase SocketManager . . . 112
14.25 Especificación de la clase AuthService . . . 112
14.26 Especificación de la clase FeedService . . . 113
14.27 Especificación de la clase TaskService . . . 113
14.28 Especificación de la clase UserService . . . 113
14.29 Especificación de la clase Message de la aplicación . . . 114
14.30 Especificación de la clase Task de la aplicación . . . 114
14.31 Especificación de la clase Session de la aplicación . . . 114
14.32 Especificación de la clase Notification de la aplicación . . . 115
14.33 Especificación de la clase Action de la aplicación . . . 115
14.34 Especificación de la clase User de la aplicación . . . 116
14.35 Especificación de la clase Role de la aplicación . . . 116
14.36 Especificación de la clase Address de la aplicación . . . 116
14.37 Especificación de la clase Server . . . 117
14.38 Especificación de la clase App . . . 118
14.39 Especificación de la clase Mongo . . . 118
14.40 Especificación de la clase BaseRouter . . . 118
14.41 Especificación de la clase AuthRouter . . . 118
14.42 Especificación de la clase FeedRouter . . . 118
14.43 Especificación de la clase TaskRouter . . . 119
14.44 Especificación de la clase UserRouter . . . 119
14.45 Especificación de la clase BaseSocket . . . 119
14.46 Especificación de la clase FeedSocket . . . 119
14.47 Especificación de la clase GlobalSocket . . . 120
14.48 Especificación de la clase LocationSocket . . . 120
14.49 Especificación de la clase FeedService de la API . . . 120
14.50 Especificación de la clase NotificationService . . . 120
14.51 Especificación de la clase SessionService . . . 121
14.52 Especificación de la clase TaskService de la API . . . 121

14.53 Especificación de la clase UserService de la API 121
14.54 Especificación de la clase Action de la API 122
14.55 Especificación de la clase Address de la API 122
14.56 Especificación de la clase Notification de la API 122
14.57 Especificación de la clase Role de la API 123
14.58 Especificación de la clase Session de la API 123
14.59 Especificación de la clase Task de la API 123
14.60 Especificación de la clase TextMessage de la API 124
14.61 Especificación de la clase User de la API 124

16.1 Características del Plan B1 127

18.1 Documentación de la entidad Server 134
18.2 Documentación de la entidad App 134
18.3 Documentación de la entidad Mongo 134
18.4 Documentación de la entidad io 134
18.5 Documentación de la entidad baseRouter 134
18.6 Documentación de la entidad authRouter 135
18.7 Documentación de la entidad feedRouter 135
18.8 Documentación de la entidad notificationRouter 135
18.9 Documentación de la entidad taskRouter 135
18.10 Documentación de la entidad userRouter 136
18.11 Documentación de la entidad bondsRouter 136
18.12 Documentación de la entidad ioRoot 136
18.13 Documentación del enumerado RootEvent 136
18.14 Documentación de la entidad feedSocket 137
18.15 Documentación del enumerado FeedEvent de la API 137
18.16 Documentación de la entidad globalSocket 137
18.17 Documentación del enumerado GlobalEvent de la API 137
18.18 Documentación de la entidad locationSocket 138
18.19 Documentación del enumerado LocationEvent de la API 138
18.20 Documentación de la clase FeedService de la API 138
18.21 Documentación de la clase NotificationService 138
18.22 Documentación de la clase SessionService 139
18.23 Documentación de la clase TaskService de la API 139
18.24 Documentación de la clase TokenService 139
18.25 Documentación de la clase UserService de la API 140
18.26 Documentación del enumerado Action de la API 140
18.27 Documentación de la interfaz Address de la API 140
18.28 Documentación del enumerado MessageType de la API 141
18.29 Documentación del esquema de Message de la API 141
18.30 Documentación del esquema de Notification de la API 141
18.31 Documentación del enumerado Role de la API 142
18.32 Documentación del esquema de Session de la API 142
18.33 Documentación del esquema de TaskMessage de la API 142
18.34 Documentación del esquema de TextMessage de la API 142
18.35 Documentación del esquema de User de la API 143
18.36 Documentación de la entidad Logger 143

18.37 Documentación de la entidad ErrorHandler 143
18.38 Documentación de la clase HttpError 144
18.39 Documentación del enumerado Severity de la API 144
18.40 Documentación del enumerado Environment de la API 144
18.41 Documentación de la interfaz ExtendedViewModel 146
18.42 Documentación de la interfaz WithModel 146
18.43 Documentación de la clase ExtendedViewModelFactory 146
18.44 Documentación de la clase PrivateViewModel 146
18.45 Documentación de la clase PrivateActivity 147
18.46 Documentación de la clase LaunchActivity 147
18.47 Documentación de la clase LaunchViewModel 147
18.48 Documentación de la clase SetUpActivity 148
18.49 Documentación de la clase SetUpViewModel 148
18.50 Documentación de la clase NameSelectionFragment 148
18.51 Documentación de la clase RoleSelectionFragment 149
18.52 Documentación de la clase ContactFillFragment 149
18.53 Documentación de la clase SetUpConfirmationFragment 149
18.54 Documentación de la clase MainActivity 149
18.55 Documentación de la clase MainViewModel 150
18.56 Documentación de la clase NotificationsDialog 150
18.57 Documentación de la clase SettingsDialog 150
18.58 Documentación de la clase NotificationsManager 151
18.59 Documentación de la clase SettingsManager 151
18.60 Documentación de la clase QRScannerActivity 151
18.61 Documentación de la clase BondsActivity 152
18.62 Documentación de la clase BondsViewModel 152
18.63 Documentación de la clase FeedActivity 152
18.64 Documentación de la clase FeedViewModel 153
18.65 Documentación de la clase LocationActivity 154
18.66 Documentación de la clase LocationViewModel 154
18.67 Documentación de la clase MarkerManager 155
18.68 Documentación de la clase TasksActivity 155
18.69 Documentación de la clase TasksViewModel 155
18.70 Documentación de la clase CreateTaskDialog 156
18.71 Documentación de la entidad RepositoryContext 156
18.72 Documentación de la interfaz SocketRoomRepository 156
18.73 Documentación de la clase BaseSocketRepository 156
18.74 Documentación de la interfaz FeedRepository 157
18.75 Documentación de la implementación de FeedRepository 157
18.76 Documentación del enumerado FeedEvent de la aplicación 157
18.77 Documentación de la interfaz GlobalRoomRepository 157
18.78 Documentación de la implementación de GlobalRoomRepository . . . 158
18.79 Documentación del enumerado GlobalEvent de la aplicación 158
18.80 Documentación de la interfaz LocationRepository 158
18.81 Documentación de la implementación de LocationRepository 158
18.82 Documentación del enumerado LocationEvent de la aplicación 158
18.83 Documentación de la interfaz NotificationRepository 159

18.84 Documentación de la implementación de NotificationRepository . . . 159
18.85 Documentación de la interfaz SessionRepository 159
18.86 Documentación de la implementación de SessionRepository 159
18.87 Documentación de la interfaz TasksRepository 160
18.88 Documentación de la implementación de TasksRepository 160
18.89 Documentación de la interfaz UserRepository 160
18.90 Documentación de la implementación de UserRepository 160
18.91 Documentación de la entidad ApiFactory 161
18.92 Documentación de la entidad SocketManager 161
18.93 Documentación de la clase ApiErrorException 161
18.94 Documentación de la interfaz AuthService 161
18.95 Documentación de la interfaz FeedService 162
18.96 Documentación de la interfaz TaskService 162
18.97 Documentación de la interfaz UserService 162
18.98 Documentación de la clase Message de la aplicación 163
18.99 Documentación de la clase TaskMessage de la aplicación 163
18.100 Documentación de la clase TextMessage de la aplicación 163
18.101 Documentación del enumerado MessageType de la aplicación 163
18.102 Documentación de la clase Notification de la aplicación 164
18.103 Documentación del enumerado Action de la aplicación 164
18.104 Documentación del enumerado Channel de la aplicación 165
18.105 Documentación de la clase Session de la aplicación 165
18.106 Documentación de la clase Task de la aplicación 165
18.107 Documentación de la clase User de la aplicación 166
18.108 Documentación del enumerado Role de la aplicación 166
18.109 Documentación de la clase Address de la aplicación 166
18.110 Documentación de la clase UserMarker de la aplicación 167

20.1 Documentación del endpoint de inicio de sesión 173
20.2 Documentación del endpoint de cierre de sesión 174
20.3 Documentación del endpoint de refresco de sesión 174
20.4 Documentación del endpoint de recuperación de mensajes 175
20.5 Documentación del endpoint de recuperar notificaciones 175
20.6 Documentación del endpoint de marca una notificación como leída . . 176
20.7 Documentación del endpoint de marcar todas las notificaciones como leídas 176
20.8 Documentación del endpoint de crear una tarea 176
20.9 Documentación del endpoint de recuperar tareas relevantes 177
20.10 Documentación del endpoint de eliminar una tarea 177
20.11 Documentación del endpoint de marcar una tarea como hecha 178
20.12 Documentación del endpoint de marcar una tarea como no hecha . . . 178
20.13 Documentación del endpoint de actualización de información de usuarios 179
20.14 Documentación del endpoint de generación de tokens de vinculación . 179
20.15 Documentación del endpoint de establecimiento de vínculos 180
20.16 Documentación del endpoint de eliminación de vínculos 180
20.17 Documentación del endpoint de actualización de información de usuarios 181
20.18 Documentación del endpoint de recuperación de información de asociados 181

20.19 Documentación del evento Subscribe de la sala Global 182
20.20 Documentación del evento Subscription de la sala Global 182
20.21 Documentación del evento Join de la sala Feed 183
20.22 Documentación del evento Leave de la sala Feed 183
20.23 Documentación del evento Send de la sala Feed 183
20.24 Documentación del evento New de la sala Feed 184
20.25 Documentación del evento Delete de la sala Feed 184
20.26 Documentación del evento Share de la sala Location 184
20.27 Documentación del evento Stop de la sala Location 185
20.28 Documentación del evento Update de la sala Location 185
20.29 Documentación del evento Notify Bond Created de la sala Global . . 185
20.30 Documentación del evento Notify Bond Deleted de la sala Global . . . 186
20.31 Documentación del evento Notify Task Deleted de la sala Global . . . 186
20.32 Documentación del evento Notify Task Done de la sala Global 186
20.33 Documentación del evento Notify Task Undone de la sala Global . . . 186
20.34 Documentación del evento Notify Location Sharing Start de la sala Global 187
20.35 Documentación del evento Notify Location Sharing Stop de la sala Global 187

22.1 Paleta de colores de la aplicación 199

23.1 Prueba unitaria de la aplicación: Actualizar datos del usuario 209
23.2 Prueba unitaria de la aplicación: Cerrar la sesión de usuario 209
23.3 Prueba unitaria de la aplicación: Eliminar el vínculo con el Paciente vinculado 209
23.4 Prueba unitaria de la aplicación: Añadir un marcador 209
23.5 Prueba unitaria de la aplicación: Comprobar la existencia de un marcador 210
23.6 Prueba unitaria de la aplicación: Eliminar un marcador 210
23.7 Prueba unitaria de la aplicación: Actualizar un marcador 210
23.8 Prueba unitaria de la aplicación: Añadir una notificación 210
23.9 Prueba unitaria de la aplicación: Añadir un conjunto de notificaciones 210
23.10 Prueba unitaria de la aplicación: Limpiar lista de notificaciones 211
23.11 Prueba unitaria de la aplicación: Cargar lista de notificaciones 211
23.12 Prueba unitaria de la aplicación: Eliminar una notificación 211
23.13 Prueba unitaria de la aplicación: Eliminar todas las notificaciones . . 211
23.14 Prueba unitaria de la aplicación: Enviar confirmación de datos de usuario 211
23.15 Prueba unitaria de la aplicación: Gestionar resultado de inicio de sesión en Google 212
23.16 Prueba unitaria de la aplicación: Gestionar resultado de inicio de sesión en Google erróneo 212
23.17 Prueba unitaria de la aplicación: Añadir un nuevo marcador 212
23.18 Prueba unitaria de la aplicación: Recibir una actualización de posición de marcador no existente 212
23.19 Prueba unitaria de la aplicación: Recibir una actualización de posición de marcador ya existente 213
23.20 Prueba unitaria de la aplicación: Obtener una página de mensajes . . 213
23.21 Prueba unitaria de la aplicación: Enviar una tarea por el feed 213
23.22 Prueba unitaria de la aplicación: Enviar un mensaje por el feed 213

23.23 Prueba unitaria de la aplicación: Recibir una actualización de tarea . 214
23.24 Prueba unitaria de la aplicación: Recibir una eliminación de tarea . . 214
23.25 Prueba unitaria de la aplicación: Recibir un nuevo mensaje 214
23.26 Prueba unitaria de la aplicación: Confirmar creación de una tarea . . 214
23.27 Prueba unitaria de la aplicación: Eliminación una tarea 214
23.28 Prueba unitaria de la aplicación: Eliminación una tarea inválida . . . 215
23.29 Prueba unitaria de la aplicación: Marcar tarea como hecha/no hecha . 215
23.30 Prueba unitaria de la aplicación: Obtener la lista de tareas 215
23.31 Prueba unitaria de la aplicación: Eliminar un vínculo 215
23.32 Prueba unitaria de la aplicación: Obtener un código QR de vinculación 215
23.33 Prueba unitaria de la aplicación: Obtener los vínculos 216
23.34 Prueba unitaria de la aplicación: Actualizar usuario 216
23.35 Prueba unitaria de la aplicación: Enviar código de vinculación 216
23.36 Prueba unitaria de la aplicación: Obtener lista de notificaciones . . . 216
23.37 Prueba unitaria de la aplicación: Obtener Paciente vinculado 216
23.38 Prueba unitaria de la aplicación: Envío de mensaje 217
23.39 Prueba unitaria de la aplicación: Recuperación de mensajes 217
23.40 Prueba unitaria de la aplicación: Recuperación de notificaciones . . . 217
23.41 Prueba unitaria de la aplicación: Marcado de notificación como leída . 217
23.42 Prueba unitaria de la aplicación: Marcado de todas las notificaciones como leídas 217
23.43 Prueba unitaria de la aplicación: Inicio de sesión 218
23.44 Prueba unitaria de la aplicación: Cierre de sesión 218
23.45 Prueba unitaria de la aplicación: Refresco de sesión 218
23.46 Prueba unitaria de la aplicación: Recuperación de tareas 218
23.47 Prueba unitaria de la aplicación: Guardado de una tarea 218
23.48 Prueba unitaria de la aplicación: Eliminación de una tarea 219
23.49 Prueba unitaria de la aplicación: Actualización de una tarea 219
23.50 Prueba unitaria de la aplicación: Actualización de un usuario 219
23.51 Prueba unitaria de la aplicación: Eliminar vinculación 219
23.52 Prueba unitaria de la aplicación: Enviar código de vinculación 219
23.53 Prueba unitaria de la aplicación: Recuperación de vínculos 220
23.54 Prueba unitaria de la aplicación: Recuperación del Paciente vinculado 220
23.55 Prueba unitaria de la aplicación: Solicitar código de vinculación . . . 220
23.56 Prueba unitaria de la aplicación: Obtención de un servicio 220
23.57 Prueba unitaria de la aplicación: Procesar respuesta de inicio de sesión 220
23.58 Prueba unitaria de la aplicación: Procesar respuesta de inicio de sesión 221
23.59 Prueba unitaria de la aplicación: Procesar respuesta de recuperación de notificaciones 221
23.60 Prueba unitaria de la aplicación: Procesar respuesta de recuperación de mensajes 221
23.61 Prueba unitaria de la aplicación: Procesar respuesta de recuperación de tareas 221
23.62 Prueba unitaria de la aplicación: Procesar respuesta de publicación de tareas 222
23.63 Prueba unitaria de la aplicación: Procesar respuesta de actualización de usuario 222

23.64 Prueba unitaria de la aplicación: Procesar respuesta de recuperación de Paciente vinculado . . . 222
23.65 Prueba unitaria de la aplicación: Procesar respuesta de recuperación de vínculos . . . 222
23.66 Prueba unitaria de la aplicación: Procesar respuesta de recuperación de código de vinculación . . . 223
23.67 Prueba unitaria de la API: Encabezado de autenticación correcto . . . 223
23.68 Prueba unitaria de la API: Encabezado de autenticación faltante . . . 223
23.69 Prueba unitaria de la API: Encabezado de autenticación mal formateado 223
23.70 Prueba unitaria de la API: Encabezado de autenticación con token inválido . . . 224
23.71 Prueba unitaria de la API: Manejo correcto de errores conocidos . . . 224
23.72 Prueba unitaria de la API: Manejo correcto de errores desconocidos . 224
23.73 Prueba unitaria de la API: Creación de un mensaje de texto . . . 224
23.74 Prueba unitaria de la API: Creación de un mensaje de texto inválido 224
23.75 Prueba unitaria de la API: Recuperación de mensajes . . . 225
23.76 Prueba unitaria de la API: Recuperación de mensajes de una página concreta . . . 225
23.77 Prueba unitaria de la API: Creación de una notificación . . . 225
23.78 Prueba unitaria de la API: Marcado de una notificación con más interesados como leída . . . 225
23.79 Prueba unitaria de la API: Marcado de una notificación sin más interesados como leída . . . 226
23.80 Prueba unitaria de la API: Marcado de notificaciones como leídas . . 226
23.81 Prueba unitaria de la API: Recuperación de notificaciones . . . 226
23.82 Prueba unitaria de la API: Recuperación de notificaciones con edad especificada . . . 226
23.83 Prueba unitaria de la API: Inicio de sesión . . . 227
23.84 Prueba unitaria de la API: Cierre de sesión . . . 227
23.85 Prueba unitaria de la API: Comprobación de sesión abierta . . . 227
23.86 Prueba unitaria de la API: Comprobación de sesión cerrada . . . 227
23.87 Prueba unitaria de la API: Comprobación de tuplas de sesión existentes 227
23.88 Prueba unitaria de la API: Comprobación de tuplas de sesión inexistentes 228
23.89 Prueba unitaria de la API: Comprobación de tuplas de sesión inválidas 228
23.90 Prueba unitaria de la API: Creación de una tarea . . . 228
23.91 Prueba unitaria de la API: Creación de una tarea inválida . . . 228
23.92 Prueba unitaria de la API: Eliminación de una tarea . . . 228
23.93 Prueba unitaria de la API: Actualización de una tarea . . . 229
23.94 Prueba unitaria de la API: Actualización de una tarea no existente . . 229
23.95 Prueba unitaria de la API: Recuperación de tareas relevantes . . . 229
23.96 Prueba unitaria de la API: Recuperación de tareas relevantes de edad especificada . . . 229
23.97 Prueba unitaria de la API: Comprobación de la sala de una tarea . . 229
23.98 Prueba unitaria de la API: Generación de tokens de sesión . . . 230
23.99 Prueba unitaria de la API: Refresco de tokens de sesión . . . 230
23.100 Prueba unitaria de la API: Refresco de tokens de sesión inválida . . . 230
23.101 Prueba unitaria de la API: Generación de tokens de vinculación . . . 230

23.102 Prueba unitaria de la API: Decodificación de tokens de vinculación . 230
23.103 Prueba unitaria de la API: Decodificación de tokens de vinculación inválidos . . . 231
23.104 Prueba unitaria de la API: Comprobación de tuplas de tokens 231
23.105 Prueba unitaria de la API: Recuperación de un usuario no existente por su ID de Google . . . 231
23.106 Prueba unitaria de la API: Recuperación de un usuario existente por su ID de Google . . . 231
23.107 Prueba unitaria de la API: Actualización de un usuario 231
23.108 Prueba unitaria de la API: Creación de vínculo 232
23.109 Prueba unitaria de la API: Creación de vínculo inválida 232
23.110 Prueba unitaria de la API: Eliminación de vínculo 232
23.111 Prueba unitaria de la API: Recuperación de Paciente vinculado de un Cuidador . . . 232
23.112 Prueba unitaria de la API: Recuperación de Paciente vinculado de un Cuidador no vinculado . . . 233
23.113 Prueba unitaria de la API: Recuperación de Paciente vinculado inválida 233
23.114 Prueba unitaria de la API: Recuperación de vínculos 233
23.115 Prueba unitaria de la API: Recuperación de rol de un usuario 233
23.116 Prueba de integración de la aplicación: Registro de un Paciente en la aplicación . . . 234
23.117 Prueba de integración de la aplicación: Registro de un Cuidador en la aplicación . . . 235
23.118 Prueba de integración de la aplicación: Inicio de sesión en la aplicación 235
23.119 Prueba de integración de la aplicación: Mostrar código de vinculación en la aplicación . . . 236
23.120 Prueba de integración de la aplicación: Desplegar escáner en la aplicación 236
23.121 Prueba de integración de la aplicación: Eliminación de vínculo de un Paciente . . . 236
23.122 Prueba de integración de la aplicación: Eliminación de vínculo de un Cuidador . . . 237
23.123 Prueba de integración de la aplicación: Compartir ubicación en la aplicación . . . 237
23.124 Prueba de integración de la aplicación: Gestionar las tareas desde Tareas 238
23.125 Prueba de integración de la aplicación: Gestionar las tareas desde el Feed . . . 239
23.126 Prueba de integración de la aplicación: Enviar y recibir mensajes en la aplicación . . . 240
23.127 Prueba de integración de la aplicación: Gestionar notificaciones en la aplicación . . . 241
23.128 Prueba de integración de la aplicación: Consultar los vínculos en la aplicación . . . 241
23.129 Prueba de integración de la aplicación: Consultar paciente en la aplicación . . . 242
23.130 Prueba de integración de la API: Registro de usuario correcto 242
23.131 Prueba de integración de la API: Inicio de sesión correcto 242
23.132 Prueba de integración de la API: Inicio de sesión incorrecto 242

23.133 Prueba de integración de la API: Cierre de sesión correcto 243
23.134 Prueba de integración de la API: Cierre de sesión incorrecto 243
23.135 Prueba de integración de la API: Refresco de sesión correcto 243
23.136 Prueba de integración de la API: Refresco de sesión incorrecto 243
23.137 Prueba de integración de la API: Conexión a la sala de localización . 243
23.138 Prueba de integración de la API: Desconexión de la sala de localización 244
23.139 Prueba de integración de la API: Actualización de ubicación 244
23.140 Prueba de integración de la API: Recuperación de mensajes de Pacientes 244
23.141 Prueba de integración de la API: Recuperación de mensajes de Cuidadores correcta 244
23.142 Prueba de integración de la API: Recuperación de páginas de mensajes concreta 244
23.143 Prueba de integración de la API: Recuperación de mensajes con página incorrecta 245
23.144 Prueba de integración de la API: Recuperación de mensajes con usuario incompleto 245
23.145 Prueba de integración de la API: Recuperación de mensajes de Cuidador incorrecta 245
23.146 Prueba de integración de la API: Conexión a la sala de mensajería . . 245
23.147 Prueba de integración de la API: Desconexión de la sala de mensajería 245
23.148 Prueba de integración de la API: Envío de un mensaje de texto . . . 246
23.149 Prueba de integración de la API: Envío de una tarea 246
23.150 Prueba de integración de la API: Marcar una notificación válida como leída 246
23.151 Prueba de integración de la API: Marcar una notificación inválida como leída 246
23.152 Prueba de integración de la API: Marcar todas las notificaciones como leídas 247
23.153 Prueba de integración de la API: Recuperación de notificaciones no leídas por defecto 247
23.154 Prueba de integración de la API: Recuperación de notificaciones no leídas de edad especificada 247
23.155 Prueba de integración de la API: Creación de una tarea 247
23.156 Prueba de integración de la API: Creación de una tarea inválida . . . 247
23.157 Prueba de integración de la API: Recuperación de notificaciones no leídas de edad especificada 248
23.158 Prueba de integración de la API: Eliminación de una tarea inválida . 248
23.159 Prueba de integración de la API: Marcar tarea como hecha 248
23.160 Prueba de integración de la API: Marcar tarea como no hecha 248
23.161 Prueba de integración de la API: Marcar tarea hecha como hecha . . 248
23.162 Prueba de integración de la API: Marcar tarea no hecha como no hecha 249
23.163 Prueba de integración de la API: Recuperación de tareas de un Paciente 249
23.164 Prueba de integración de la API: Recuperación de tareas de un Cuidador 249
23.165 Prueba de integración de la API: Recuperación de tareas con edad máxima 249
23.166 Prueba de integración de la API: Actualización de usuario 249
23.167 Prueba de integración de la API: Actualización de usuario inválida . . 250

23.168 Prueba de integración de la API: Creación de vínculo 250
23.169 Prueba de integración de la API: Creación de vínculo inválida 250
23.170 Prueba de integración de la API: Eliminación de vínculo válida 250
23.171 Prueba de integración de la API: Generación de código de vinculación 250
23.172 Prueba de integración de la API: Recuperación de vínculos de un Paciente 251
23.173 Prueba de integración de la API: Recuperación de vínculos de un Cuidador 251
23.174 Prueba de integración de la API: Recuperación de vínculos de un usuario incompleto 251
23.175 Prueba de integración de la API: Recuperación de Paciente de un Cuidador vinculado 251
23.176 Prueba de integración de la API: Recuperación de Paciente de un Cuidador no vinculado 251
23.177 Prueba de integración de la API: Recuperación de Paciente inválida . 252

27.1 Primera parte de los resultados de las pruebas unitarias de la API . . 280
27.2 Segunda parte de los resultados de las pruebas unitarias de la API . . 281
27.3 Primera parte de los resultados de las pruebas unitarias de la aplicación 282
27.4 Segunda parte de los resultados de las pruebas unitarias de la aplicación 283
27.5 Tercera parte de los resultados de las pruebas unitarias de la aplicación 284
27.6 Primera parte de los resultados de las pruebas de integración de la API 284
27.7 Segunda parte de los resultados de las pruebas de integración de la API 285
27.8 Tercera parte de los resultados de las pruebas de integración de la API 286
27.9 Resultados de las pruebas de integración de la aplicación 286

30.1 Variables de entorno del servicio 302

35.1 Presupuesto de costes expandido 314
35.2 Partida 1. Planificación 315
35.3 Partida 2. Análisis 316
35.4 Partida 3. Diseño 316
35.5 Partida 4. Construcción 318
35.6 Partida 5. Formación 318
35.7 Coste del personal 318
35.8 Costes indirectos 319
35.9 Medios de producción 319
35.10 Facturación del equipo 319
35.11 Resumen de coste-facturación 320

Índice de fragmentos

37.1 Enumerado de MessageType 339
37.2 Esquema de Message 339
37.3 Esquema de TaskMessage 340
37.4 Esquema de TextMessage 340
37.5 Enumerado de Action 340
37.6 Esquema de Notification 341
37.7 Esquema de Session 341
37.8 Enumerado de Role 342
37.9 Esquema de Address 342
37.10Esquema de User 342
37.11Tipo de Message 343
37.12DTO mínimo de Task 343
37.13DTO de Task 344
37.14DTO de Notification 344
37.15DTO de Session 344
37.16DTO mínimo de User 345
37.17DTO público de User 345
37.18DTO de User 345
38.1 Archivo de despliegue 346

Parte I

Presentación

1. Memoria del proyecto

1.1. Resumen de la motivación, objetivos y alcance del proyecto

La enfermedad de Alzheimer es un trastorno neurodegenerativo que reduce grave y progresivamente las capacidades cognitivas de sus pacientes. Su duración media tras el diagnóstico es de diez años, a lo largo de los cuales se va produciendo un deterioro en la memoria, la independencia, los trastornos conductuales y las capacidades físicas del afectado.

El diagnóstico de esta enfermedad es también el inicio de un cambio repentino en el entorno del diagnosticado. Actualmente no existe ningún tratamiento que detenga la enfermedad, aunque algunos fármacos contribuyan a retrasar el avance de la misma. Aún así, la mayor parte de los paliativos de la misma son de tipo **conductual** e involucran no solo al paciente sino también a **su entorno cercano**.

El objetivo de este proyecto es el **desarrollo de un sistema software** que pueda facilitar algunos de los aspectos en los que familiares, amigos y demás allegados del paciente son clave para servirle de apoyo en las etapas tempranas de la enfermedad.

El sistema a desarrollar será una **aplicación móvil** con funciones de contacto, gestión de tareas, geolocalización y mensajería. Todas estas funciones de la aplicación estarán respaldadas por **un servidor** desplegado en la nube que se encargará de la lógica subyacente y de la persistencia de los datos y la comunicación de estos entre usuarios. Dicho servidor ofrecerá dos interfaces de comunicación: una basada en WebSockets para los envíos de comunicaciones más inmediatas de los usuarios con otros usuarios y con el sistema; y otra HTTP con una API REST para el resto de operaciones.

Para poder crear el sistema deseado será necesario utilizar **diversas tecnologías** como los anteriormente mencionados WebSockets, los sistemas de mapas y geolocalización, las corrutinas, las bases de datos NoSQL o los servicios en la nube. La mayor parte de estas son nuevas para el equipo de desarrollo y de esta forma representan un obstáculo a la vez que una oportunidad de aprendizaje.

La creación del sistema se basará en la planificación, análisis y diseño presentados en este documento. Documento que junto al sistema descrito en las líneas previas conforman este proyecto de desarrollo de una **aplicación asimétrica móvil de asistencia para familias con personas afectadas de Alzheimer**.

1.2. Estructura del documento

El documento que nos ocupa consta de los siguientes secciones y capítulos:

1. **Presentación**
   a) **Memoria del proyecto**. Breve introducción al documento.
   b) **Introducción**. Presentación general del proyecto que se desarrollará en este documento.
   c) **Aspectos teóricos**. Explicación de conceptos de la especialidad importantes para la comprensión del proyecto.
2. **Planificación**
   a) **Requisitos iniciales**. Lista inicial de requisitos del sistema.
   b) **Evaluación de alternativas**. Análisis de opciones tecnológicas para el desarrollo del proyecto.
   c) **Análisis de riesgos**. Perspectiva de riesgos del desarrollo y plan de actuación.
   d) **Descripción de la solución propuesta**. Resumen del sistema a desarrollar.
   e) **Planificación temporal**. Propuesta de calendario para la realización del proyecto.
   f) **Resumen del presupuesto**. Puntos clave del presupuesto estimado.
3. **Análisis**
   a) **Requisitos del sistema**. Listado definitivo de requisitos planeados para el sistema.
   b) **Identificación de subsistemas**. Definición de los diferentes subsistemas que conformarán *AllForOne*.
   c) **Análisis de Interfaz de Usuario**. Propuesta de pantallas y de navegación para la aplicación.
   d) **Diagrama de clases**. Diseño previo de las clases que se estimará que tenga el sistema.
   e) **Especificación del plan de pruebas**. Plan de desarrollo y realización de pruebas del sistema.
   f) **Especificación del plan de despliegue**. Plan de despliegue del servidor.

4. **Diseño**
    a) **Arquitectura del sistema**. Propuesta de arquitectura para el sistema.
    b) **Diseño de clases**. Diseño completo del sistema de clases y componentes.
    c) **Modelo de datos**. Definición de las entidades manejadas por las bases de datos.
    d) **Interfaces de comunicación**. Definición de las API de comunicación entre aplicación y servidor.
    e) **Interacción y estados**. Explicación de las interacciones, procedimientos y estados relacionados con los casos de uso a implementar.
    f) **Interfaz de usuario**. Diseño definitivo de las pantallas de la aplicación.
    g) **Especificación técnica del plan de pruebas**. Recolección de los casos de prueba anticipados para el desarrollo.
5. **Implementación**
    a) **Tecnologías**. Enumeración y descripción de las tecnologías usadas en el desarrollo.
    b) **Herramientas de desarrollo**. Enumeración y descripción de las herramientas usadas para el desarrollo.
    c) **Obstáculos**. Descripción de los obstáculos encontrados a lo largo del desarrollo.
    d) **Desarrollo de pruebas**. Resultados y conclusiones del apartado de pruebas del proyecto.
6. **Manuales**
    a) **Manual de instalación**. Guía de instalación de la aplicación.
    b) **Manual de usuario**. Guía de uso de la aplicación.
    c) **Manual de despliegue**. Guía de procedimiento para el despliegue del servidor.
    d) **Manual de desarrollador**. Conjunto de guías de ayuda para la continuación del desarrollo.
7. **Conclusión**
    a) **Estado del sistema**. Descripción del estado del sistema en el momento de la entrega.
    b) **Conclusiones personales**. Experiencia personal del desarrollo y el proceso que condujo al mismo.

c) **Ampliaciones**. Lista de posibles ampliaciones planeadas o propuestas para el sistema.

8. **Anexos**

a) **Presupuesto**. Versión detallada del presupuesto.

b) **Licencia**. Licencia de uso del sistema desarrollado.

c) **Modelado de datos**. Esquemas de las entidades modeladas.

d) **Acción de despliegue**. Fichero de despliegue del servidor.

e) **Documentación adicional**. Listado de documentación adicional entregada junto a este documento.

2. Introducción

2.1. Justificación del proyecto

En el tiempo que se tarda en leer esta primera oración completa un nuevo caso de Alzheimer ha sido diagnosticado. **Siete segundos**. Si extendiésemos los casos de Alzheimer anuales del mundo equitativamente en el tiempo que dura dicho año tendríamos un nuevo diagnóstico cada siete segundos.

Según el estudio *El cuidador en España. Contexto actual y perspectivas de futuro. Propuestas de intervención.*[1] de CEAFA en 2016, en España existen más de 1,2 millones de personas afectadas por Alzheimer, lo que equivale a que uno de cada cuatro hogares tenga relación directa con un paciente de esta enfermedad.

La motivación de este proyecto parte de esta situación. La aparición de esta enfermedad neurodegenerativa implica un cambio radical en todos esos miles de hogares que son tocados por ella. Las familias y el entorno cercano deben adaptarse para poder proporcionar toda la ayuda necesaria para el paciente. La comunicación es la piedra básica sobre la que es necesario construir los cimientos que permitan capear lo mejor posible este inesperado temporal. Descubrir esta demencia implica la necesidad de crear una coordinación, no solo con el paciente, sino con el resto de cuidadores.

Como se comentó en el Capítulo 1, el mal de Alzheimer tiene un diagnóstico terminal y las principales indicaciones para combatir la enfermedad pasan por actividades, conductas o actitudes que paciente y allegados deben llevar a cabo. Es habitual que se inste a formar parte de las diversas y numerosas asociaciones dedicadas a esta enfermedad que se encuentran repartidas por todo el territorio español. La lucha contra el Alzheimer de las familias afectadas a día de hoy no es pasiva ni individual, **es un esfuerzo colectivo activo** de todo el entorno para paliar los estragos que deja tras de sí el trastorno.

En la etapa más temprana de la enfermedad, la persona afectada aún se puede **desenvolver de forma independiente**. Es habitual que una persona recién diagnosticada continúe trabajando o relacionándose de forma muy próxima a lo que era habitual en su vida por un tiempo. Sin embargo, experimentará una serie de dificultades que se irán haciendo más patentes y habituales con el paso del tiempo.

La incapacidad de encontrar una palabra o nombre correcto. Olvidar inmediatamente conocimientos recién adquiridos y extraviar objetos. Ser incapaces de planificar y, posteriormente, de llevar a cabo lo planeado. Sufrir cambios de humor o pérdidas de ánimo. Son algunas de las situaciones que los afectados por esta enfermedad vivirán **cada vez más a menudo** según pase el tiempo.

Posteriormente llega la etapa media, **la más prolongada en el tiempo**, a lo largo de la cual el enfermo va perdiendo su autonomía y se vuelve muy necesario otorgarle máxima atención. En este periodo el enfermo comienza a sufrir síntomas más inhabilitantes como son: desorientarse y perderse incluso en entornos conocidos, olvidar datos personales propios, no reconocer su propio rostro, desórdenes del sueño, desubicación temporal, pérdida de capacidad de uso de herramientas, problemas para controlar las necesidades básicas y la higiene o la incapacidad total de realizar cosas planificadas como tomar su medicación.

Está en la mano de sus personas cercanas el ayudarles a paliar y minimizar las dificultades que estas situaciones pueden generar. A medida que la demencia avanza, el afectado es cada vez menos capaz de afrontar dichas dificultades y más vital se vuelve que su entorno le ayude a llevar a cabo todas esas acciones que la enfermedad le impide llevar a cabo con normalidad.

Facilitar todo esto es el objetivo principal de **All for One**, este proyecto. El nombre proviene del lema que *Los Tres Mosqueteros* de Alejandro Dumas proclamaban con sus roperas cruzadas:

> *Unus pro omnibus, omnes pro uno*
> - Alexandre Dumas, *1844*

Uno para todos, todos para uno. La segunda mitad del lema evoca precisamente el ánimo que este sistema persigue. La colaboración y dedicación por y para alguien apreciado. Un esfuerzo de todos los seres queridos por esa persona víctima de la peor de las suertes.

La propuesta de proyecto es una **aplicación móvil** que ofrezca herramientas para facilitar, a los aquejados de Alzheimer y a su entorno, la larga y dura travesía a lo largo de esas etapas tempranas de la enfermedad. *All for One* busca ser la ropera que permita a todos blandirse en duelo contra el mal que persigue a uno.

2.2. Objetivos del proyecto

El objetivo del proyecto es completar el desarrollo de un prototipo de aplicación que ofrezca a sus usuarios, pacientes y cuidadores, herramientas para facilitar en la medida de lo posible los siguientes síntomas de las etapas temprana y media de la enfermedad:

1. **Olvido de datos personales**. El paciente podrá guardar sus datos elementales como la dirección o el número de teléfono para tenerlos siempre a mano en caso de necesidad. Los datos de sus cuidadores también se le ofrecerán para que también puedan ser de ayuda en dichos momentos.

2. **Desorientación**. En caso de que el paciente se pierda, se podrá utilizar un servicio para compartir la ubicación a través del que sus cuidadores podrán localizarlo.

3. **Olvido de tareas y responsabilidades**. Pacientes y cuidadores tendrán a su disponibilidad una lista de tareas compartida en la que todos podrán crear y gestionar tareas, favoreciendo que no sean olvidadas y puedan ser llevadas a cabo.

Todo esto se desarrollará con un enfoque centrado en potenciar y facilitar la cooperación de todos los involucrados con un paciente. Esto implica añadir una vía de comunicación compartida y un canal de notificaciones para favorecer que todos los cuidadores estén siempre al tanto de las novedades del paciente.

Por último, un objetivo importante del proyecto es ofrecer un producto que sea de utilidad para todas las personas que deban interaccionar con esta enfermedad, no solo directa, sino también indirectamente. Esto quiere decir que otro de los objetivos de este proyecto es que la aplicación pueda ser un punto de valor para el entorno tecnológico enfocado en el Alzheimer.

No existe un interés comercial en el desarrollo de este sistema y el enfoque económico y presupuestario del mismo siempre irán enfocado en el afán survivabilista y de coexistencia con el resto de aplicaciones de índole similar, por ello, en el resto del documento nunca se hablará de mercado, sino de ámbito. Para sumar aún más sobre este objetivo, *All for One* tiene como objetivo ser un proyecto de **código abierto** y así lo ha sido desde el día uno.

2.3. Estudio de la situación actual

Por suerte, hoy en día existe ya un considerable número de aplicaciones móviles que giran alrededor del mismo eje que lo hace *All for One*, el cuidado y la ayuda de pacientes de Alzheimer. Es de destacar que el Centro de Referencia Estatal de Atención a Personas con Enfermedad de Alzheimer y otras Demencias, abreviado como **CREA** o CRE Alzheimer, dispone en su página web de un listado muy completo[1] (aunque no debidamente actualizado) de las aplicaciones directamente enfocadas en este tema o que lateralmente pueden ser de utilidad en el mismo, categorizándolas según sean de utilidad para cuidadores, pacientes o profesionales; e indicando información de las mismas como una breve descripción o los idiomas en los que se encuentra disponible.

A continuación se ofrece una selección de las aplicaciones activas más destacables y el cometido para que el son de utilidad a la hora de lidiar con la enfermedad.

- **YoTeCuido**. Información detallada y categorizada para pacientes y cuidadores.
- **Pharma Alzheimer**. Divulgación e información acerca de la enfermedad.
- **The Dementia-Friendly-Home**. Información y consejos para adaptar un hogar para que viva una persona con demencia.
- **Cuidador de Alzheimer**. Comunidad digital para cuidadores.
- **Alert Cops**. Aplicación de la Secretaría de Seguridad que permitiría al paciente contactar rápidamente con los cuerpos de seguridad en caso de necesidad.
- **The Brainy App**. Ejercitación mental para el paciente.
- **Stimulus**. Aplicación para estimulación cognitiva con registro de las sesiones y posibilidad de seguimiento de las mismas por parte de familiares o profesionales.
- **Imentia**. Otra aplicación de estimulación mental con capacidades multiusuario y de personalización de las sesiones.
- **Recordatorio de Medicación**. Aplicación especializada en alarmas y recordatorios para evitar el olvido de consumo de los fármacos.

Todas las aplicaciones anteriores pertenecen al mismo ámbito que el sistema a desarrollar, pero tienen enfoques o utilidades distintas. Ninguna de ellas está enfocada en ofrecer perfiles especializados para pacientes y cuidadores, ni ofrecen las mismas

[1] https://crealzheimer.imserso.es/crealzheimer_01/recursos/apps/index.htm

herramientas que pretende ofrecer *All for One*. Son complementos idóneos para los usuarios objetivo del sistema, pero sin superposición con la que nos ocupa.

Por otro lado, en el mercado móvil sí se puede encontrar otro buen número de aplicaciones transversales que ofrecen funcionalidades similares a las que persigue *All for One*, sin pertenecer al mismo ámbito de forma directa. Por ejemplo, aplicaciones de tareas enfocadas a equipos como **Asana** pueden ofrecer la gestión de recordatorios y tareas cooperativa que se intenta ofrecer en el sistema.

En el aspecto de la geolocalización personal es donde se puede encontrar un mayor número de alternativas como **Life360** (Figura 2.1), **Geozilla** (Figura 2.2), **SmartPanics** o **Safe365**. Todas estas aplicaciones ofrecen un servicio de localización personal de gran nivel con funcionamiento intuitivo y eficaz. Las dos primeras cuentan con un uso muy extendido con varios cientos de miles de descargas debido a su versatilidad para adoptarse a muchas situaciones en las que una funcionalidad de este estilo pueda ser de importancia.

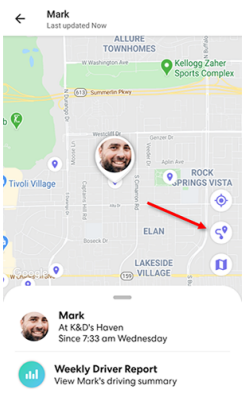


Figura 2.1: Mapa de Life365

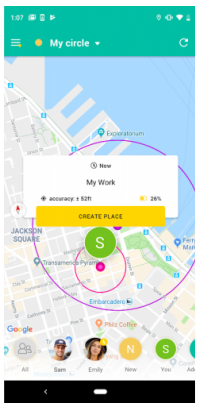


Figura 2.2: Mapa de GeoZilla

Aún con todo, estas aplicaciones transversales tienen el inconveniente de la carencia del enfoque en la enfermedad. Los gestores de tareas grupales tienden a tener un enfoque profesional para la gestión de equipos de trabajo y con ello tienen un alcance excesivo para el tipo de recordatorios que se deben manejar, complicando su uso para estos menesteres. Las aplicaciones de geolocalización, por otro lado, aunque no están centradas en esta misma enfermedad, sí están diseñadas con necesidades similares en mente y por ello ofrecen una experiencia óptima para el uso que se daría en este ámbito. Lo mismo se puede decir de las aplicaciones de mensajería, que con su diseño generalista también son idóneas para utilizar en los casos que nos ocupan.

Respecto a estos conjuntos de aplicaciones más dedicadas, el aporte que pretende ofrecer *All for One* radica en la comunión de todas estas funcionalidades en un único

sistema, evitando configuraciones extra o varias instalaciones para poder contar con todas. Permitiendo a su vez, facilidad en el uso interrelacionado de estas funcionalidades sin excesivos pasos extra y con un canal único de notificaciones común.

2.3.1. Evaluación de alternativas

Dejando a un lado las aplicaciones que ofrecen coincidencias parciales o moderadas podemos pasar a un estudio enfocado en la evaluación de las aplicación con mayores coincidencias y con similutes más directas. Se han seleccionado tres de estas aplicaciones para ofrecer una revisión más dedicada de la que extraer conclusiones que aporten a la hora de diseñar y desarrollar **All for One**.

2.3.1.1. Trusted Elderly Care


Figura 2.3: Icono de Elderly Care

Trusted Elderly Care es una aplicación desarrollada por **LEVSTONE** disponible en la PlayStore[2] de Android. La aplicación está enfocada en la monitorización de familiares, especialmente ancianos, y aunque se ofrece de forma gratuita necesita un pago de seis euros para crear los grupos que se usan para el monitoreo, sin embargo, una única cuenta que lo haya adquirido ya permite crear un grupo para todos los familiares. Además de eso, la compañía acepta donaciones en su página web[3].

Como se ha dicho, la aplicación gira en torno a la **creación de grupos** a los que se deben unir los usuarios para obtener información del resto de usuarios. Entre las funciones que ofrece la aplicación están: elegir un emoji que represente el estado de ánimo, sincronizar la aplicación con sensores y obtener sus lecturas, consultar un mapa con las localizaciones de los usuarios, activar un modo pánico, acceder a los contactos del grupo o crear y consultar tareas entre otras cosas.

La mayoría de estas acciones y otros sucesos relacionados con el usuario como el nivel de carga o el último momento de uso del teléfono móvil **son notificados al resto de usuarios** del grupo. Sin embargo, el sistema de notificaciones no es instantáneo, la aplicación se debe refrescar para obtener los nuevos cambios en las pantallas o las notificaciones. La aplicación se refresca según el tiempo que el usuario establece en ajustes o usando manualmente el botón de refresco.

La aplicación permite personalización en bastantes apartados y amplio nivel de

[2] https://play.google.com/store/apps/details?id=com.levstone.mobility.trustedelderlycare
[3] https://levstone.com/health-family/trusted-elderly-care/

Figura 2.4: Capturas de pantalla de la página web Elderly Care

detalle y opciones en el uso de sus herramientas. Las tareas que permite crear cuentan con un amplia variedad de campos como establecer la hora y lugar de comienzo y final, el tipo de tarea o los usuarios relacionados con la misma, así como la opción de añadir imágenes o enlaces. En la función de localización ofrece la opción de guardar localizaciones que luego pueden servir, además de para mostrarlas en el mapa, para crear avisos rápidos de estar dirigiéndose hacia el lugar o para ponerlo como lugar donde se llevará a cabo una tarea.

El uso de la aplicación es difícil. La interfaz es ofuscada, está llena de información y no la presenta con iconos o convenciones habituales. Entre otras cosas, los filtros de la ventana de notificaciones están en diferentes botones, nombrados como Usuario o Lugar, que no ofrecen ninguna información de su utilidad. Sin embargo, el mayor problema radica en la navegación, es abstracta y enrevesada, dificultando el acceso al gran número de funciones. Esto es un gran problema de cara a una aplicación pensada para ser usada por gente de avanzada edad, que tienen menos manejo con las tecnologías además de peor visión o memoria entre otras cosas.

Virtudes

- Herramienta de tareas muy completa e informativa.
- La capacidad de guardar lugares en el mapa.
- La vinculación de sensores biométricos.

Inconvenientes

- Elevado consumo de batería por el monitoreo de recursos.
- Navegación ofuscada y enrevesada.
- Necesidad de refresco manual para la recepción de nueva información.
- Interfaz sobrecargada y poco intuitiva.

2.3.1.2. Tú y nosotros

Tú y nosotros (o como aparece en la Play Store[4], **Alzheimer App TyN**) es una aplicación desarrollada por A. Calvo. Su principal función es almacenar una ficha con los datos del paciente de Alzheimer y de uno de sus cuidadores (Figura 2.6) para que el paciente siempre tenga esa información a mano. Su otra función principal es permitir la creación de recordatorios. Además, **tiene dos juegos** para el ejercicio mental y un botón para realizar llamadas de emergencia. Como muchas otras aplicaciones también contiene información y enlaces de interés.

Los recordatorios que se pueden crear están compuestos por un nombre que lo define y por una hora a la que se deba recordar. Cuando llega dicha hora **una notificación es enviada al usuario** con el aviso, de forma que favorece que la pueda llevar a cabo. Además de eso, la notificación se puede marcar como repetible, lo cual hará que esta se notifique diariamente. El usuario también tiene disponible una lista desde la que consultar y eliminar sus recordatorios guardados.

Los juegos ofrecidos por la aplicación son: **Memoria** (Figura 2.7), un juego de parejas con un tablero 4x4 para recordar y juntar parejas de imágenes de conceptos comunes como utensilios, animales o vehículos; y **Reconoce**, juego en el que se sirven al jugador dos columnas, una con imágenes del mismo tipo que Memoria y otra con los nombres de estos, el jugador debe relacionar cada objeto o animal con el sustantivo que lo nombra.

Figura 2.5: Icono de Tú y Nosotros

La aplicación es sencilla de utilizar y muy clara. Sin embargo, **no cuenta con perfiles ni con interconexión** entre aplicaciones. Es una aplicación con ejecución y persistencia local que no permite la interacción entre cuidadores y pacientes, de hecho, es una aplicación destinada únicamente a los dispositivos de los clientes. Si un cuidador

[4] https://play.google.com/store/apps/details?id=com.tuynosotros.alzheimertyn

quisiese incluir su información o añadir recordatorios al paciente debería hacerlo desde el móvil del mismo.

Virtudes

- Manejo muy sencillo y apto para cualquier usuario.
- Aviso por notificación de los recordatorios.
- Ficha con información útil para los pacientes.

Inconvenientes

- No permite ayudar al paciente de forma remota.
- Los recordatorios permiten poca información u opciones.
- Visualmente le falta claridad en algunos apartados.

Figura 2.6: Ficha de contacto de Tú y Nosotros

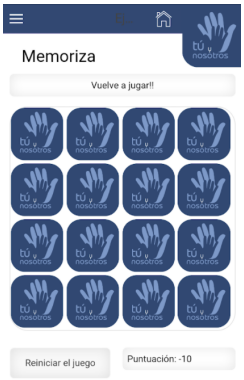


Figura 2.7: Juego de memoria de Tú y Nosotros

3. Aspectos teóricos

En este capítulo se presentarán definiciones de una serie de conceptos de la especialidad que son clave en la comprensión del proyecto que nos aborda.

3.1. API REST

En el año 2000, Roy Fielding publicó *Architectural Styles and the Design of Network-based Software Architectures*[2], disertación en la que presentó al mundo la transferencia de estado representacional o REST.

REST es una propuesta de arquitectura para aplicaciones web basada en la comunicación de representaciones del estado en las peticiones siguiendo las especificaciones HTTP. En una petición REST un cliente hace una petición al servidor con una serie de encabezados y parámetros que serán usados en el extremo solicitado para procesarla y responder a la petición con una representación del recurso solicitado y una nueva serie de encabezados con información acerca de la petición como el código de estado o el formato de la respuesta.

Por otro lado, una API es una interfaz de programación de aplicaciones utilizada en la definición, diseño e integración de software para especificar los requisitos en las dos partes de la comunicación entre los sistemas software que la emplean para garantizar el éxito de la transferencia.

Una API REST es, por tanto, una serie de definiciones y protocolos que se adapta a la arquitectura REST, permitiendo una interacción con la que los clientes puedan transferir y recibir recursos de un servidor por medio de peticiones HTTP sin estado, esto es que cada una es independiente de todas las demás. La comunicación entre un cliente y un servidor con una API REST se realiza por medio de extremos o endpoints.

Los recursos de una API REST deben, a su vez, seguir una interfaz de comunicación estándar que facilite la comunicación total con el servicio a la vez que permite el manejo y gestión del recurso a partir de la representación transferida en la comunicación.

3.2. WebSocket

El protocolo WebSocket es una interfaz de comunicación a través de un único canal TCP. Este protocolo ofrece una comunicación en tiempo real a través de una API que permite que los extremos de la comunicación (cliente y servidor) puedan enviarse mensajes sin una petición previa del receptor. WebSocket fue estandarizado en 2011 con el *RFC 6455*[3] por el IETF.

La comunicación de los WebSocket se realiza a través de URI con un nuevo esquema que comienza por `ws:` o `wss:`, según sean para transmisiones no encriptadas o encriptadas respectivamente, equivalentes a las similares de HTTP. Dichos enlaces son utilizados para establecer un canal de comunicación, cerrarlo o para utilizarlo para enviar un mensaje.

Las API de comunicación por medio de WebSocket lo aplican por medio de un patrón de suscripción, asignando a los identificadores de los diferentes tipos de mensaje un punto de escucha con una función o servicio a ejecutar cuando se reciba un nuevo mensaje. El envío de mensajes de un cliente a un servidor es elemental, pero la comunicación servidor-cliente puede ser única o de difusión, enviando el mismo mensaje a una serie de clientes conectados.

3.3. NoSQL

El término NoSQL (del inglés *Not only SQL*) engloba una serie de tecnologías de bases de datos que se apartan del clásico modelo de gestión de bases de datos relacionales cuyo principal representante es SQL. Las características comunes son que permiten la consulta sin necesidad de emplear SQL, están diseñadas para favorecer la escalabilidad sacrificando la consistencia o atomicidad y apuestan por la flexibilidad.

Existen muchos tipos de sistemas de gestión NoSQL siendo los más resaltables:

1. **Bases de datos documentales**, basadas en el almacenamiento de documentos asociados a una clave única. La gran fortaleza de estas bases de datos radica en la capacidad de guardar y recuperar grandes volúmenes de datos con facilidad y gran flexibilidad. Los documentos más habituales son de formato JSON (conjunto de pares clave-valor, clave-matriz o clave-documento). Un ejemplo de este tipo de bases de datos es MongoDB[1] (Apartado 5.4.1).

[1] https://www.mongodb.com/

2. **Bases de datos de grafos**, enfocados en las relaciones entre entidades. Su mayor ventaja radica en la rápida navegación y agregación de datos que basan con sus relaciones, mucho más ágil que los *join* de SQL. La base de datos de grafos más popular es Neo4J[2].

3. **Almacenes de clave-valor**. La bases de datos más atómicas. Almacenan pares de clave y valor sin mayor estructura o relación, lo que las hace muy veloces para la recuperación de información básica conocida y un complemento habitual de otras bases de datos para proporcionar una caché de acceso veloz a datos habitualmente solicitados. Redis[3], por ejemplo, es una base de datos clave-valor que funciona en memoria.

4. **Bases de datos orientadas a columnas**. En las bases de datos SQL, que almacenan los datos en filas, se favorece el acceso a las propiedades completas de una entidad. En una base de datos columnar, la disposición favorece el acceso a los conjuntos de propiedades de las entidades, haciéndolas destacar en la evaluación de grandes volúmenes de datos, sacrificando rendimiento en las transacciones. Una base de datos de este estilo es Apache Cassandra[4].

Otros tipos de bases de datos NoSQL son, por ejemplo, las bases de datos orientadas a objetos o las bases de datos multivalor entre otras.

[2] https://neo4j.com/
[3] https://redis.io/
[4] https://cassandra.apache.org/

Parte II

Planificación

4. Requisitos iniciales

En la planificación inicial del sistema se han definido unos primeros requisitos generales a considerar para usar como base en todo el tramo de planificación.

RI1. El sistema ofrecerá una aplicación móvil.

- RI1.1. La aplicación estará disponible en Android.
- RI1.2. La aplicación permitirá el inicio de sesión en la aplicación con una cuenta de Google.
- RI1.3. La aplicación permitirá cerrar la sesión.
- RI1.4. Los usuarios podrán cubrir su perfil con datos personales.
- RI1.5. Los usuarios recién creados podrán elegir un rol.
  - RI1.5.1. Paciente.
  - RI1.5.2. Cuidador.
- RI1.6. Los Pacientes y Cuidadores podrán vincularse.
  - RI1.6.1. Un Paciente podrá tener varios Cuidadores.
  - RI1.6.2. Un Cuidador podrá tener sólo un Paciente.
- RI1.7. Los Pacientes y Cuidadores podrán desvincularse.
- RI1.8. La aplicación ofrecerá una serie de herramientas a los usuarios registrados y vinculados.
  - RI1.8.1. Los Pacientes podrán ver su ficha personal.
  - RI1.8.2. Los Cuidadores podrán ver la ficha personal de su Paciente vinculado.
  - RI1.8.3. Los Pacientes y sus Cuidadores podrán comunicarse en una sala de chat compartida.
  - RI1.8.4. Los Pacientes y sus Cuidadores podrán gestionar tareas para el Paciente.
    - RI1.8.4.1. Los Pacientes y sus Cuidadores podrán crear una nueva tarea.
    - RI1.8.4.2. Los Pacientes y sus Cuidadores podrán consultar las tareas existentes.
    - RI1.8.4.3. Los Pacientes y sus Cuidadores podrán marcar una tarea como hecha.
    - RI1.8.4.4. Los Pacientes y sus Cuidadores podrán marcar una tarea como no hecha.
    - RI1.8.4.5. Los Pacientes podrán eliminar cualquier tarea.
    - RI1.8.4.6. Lo usuario de tipo Cuidador podrán eliminar las tareas que han creado ellos mismos.

RI1.8.5. Los Pacientes y sus Cuidadores podrán compartir su ubicación.

RI1.8.6. Los Pacientes y sus Cuidadores podrán consultar en un mapa la ubicación compartida por los demás usuarios relacionados.

RI1.8.7. Los Pacientes y sus Cuidadores podrán consultar el listado de Cuidadores del paciente.

RI1.8.7.1. Los Pacientes y sus Cuidadores podrán consultar los datos de los Cuidadores.

RI1.8.8. Los Pacientes y sus Cuidadores serán notificados de las acciones clave de los usuarios relacionados.

RI1.8.9. Los Pacientes y sus Cuidadores podrán consultar sus notificaciones pendientes.

RI2. El sistema contará con un servidor desplegado en la nube para manejar la lógica.

RI3. El servidor del sistema ofrecerá una API REST.

RI3.1. La API ofrecerá endpoints para la gestión de sesión de usuarios.

RI3.1.1. La API ofrecerá un endpoint para tramitar el inicio de sesión.

RI3.1.2. La API ofrecerá un endpoint para refrescar la sesión.

RI3.1.3. La API ofrecerá un endpoint para cerrar la sesión.

RI3.2. La API ofrecerá endpoints para la gestión de información de usuarios.

RI3.2.1. La API ofrecerá un endpoint para actualizar los datos de un usuario.

RI3.2.2. La API ofrecerá un endpoint para recuperar la información de un Paciente vinculado.

RI3.2.3. La API ofrecerá un endpoint para recuperar la información de los Cuidadores de un Paciente.

RI3.3. La API ofrecerá endpoints para la gestión de los vínculos de usuarios.

RI3.3.1. La API ofrecerá un endpoint para crear un código de vinculación.

RI3.3.2. La API ofrecerá un endpoint para crear un vínculo entre un Paciente y un Cuidador.

RI3.3.3. La API ofrecerá un endpoint para eliminar un vínculo entre un Paciente y un Cuidador.

RI3.4. La API ofrecerá endpoints para la gestión de tareas.

RI3.4.1. La API ofrecerá un endpoint para crear una tarea.

RI3.4.2. La API ofrecerá un endpoint para listar las tareas.

RI3.4.3. La API ofrecerá un endpoint para eliminar una tarea.

RI3.4.4. La API ofrecerá un endpoint para marcar una tarea como completada.

RI3.4.5. La API ofrecerá un endpoint para marcar una tarea como no completada.

RI3.5. La API ofrecerá endpoints para la gestión de mensajes.

RI3.5.1. La API ofrecerá un endpoint para recuperar los mensajes más recientes.

RI3.6. La API ofrecerá endpoints para la gestión de notificaciones.

RI3.6.1. La API ofrecerá un endpoint para recuperar las notificaciones no leídas.

RI3.6.2. La API ofrecerá un endpoint para marcar una notificación como leída.

RI3.6.3. La API ofrecerá un endpoint para marcar todas las notificaciones como leídas.

RI4. El servidor del sistema ofrecerá un servicio de WebSocket para comunicación con la aplicación.

RI4.1. El sistema permitirá establecer la conexión desde la aplicación.

RI4.2. El sistema gestionará las desconexiones.

RI4.3. El sistema gestionará eventos de una sala global.

RI4.3.1. El sistema escuchará un evento de subscripción a la sala global.

RI4.4. El sistema gestionará eventos de una sala de localización.

RI4.4.1. El sistema escuchará un evento de subscripción a la sala de localización.

RI4.4.2. El sistema escuchará un evento de actualización de localización.

RI4.4.3. El sistema escuchará un evento de desconexión de la sala de localización.

RI4.5. El sistema gestionará eventos de una sala de mensajería.

RI4.5.1. El sistema escuchará un evento de subscripción a la sala de mensajería.

RI4.5.2. El sistema escuchará un evento de envío de mensajes.

RI4.5.3. El sistema escuchará un evento de desconexión de la sala de mensajería.

RI4.6. El sistema gestionará eventos de una sala de notificaciones.

RI4.6.1. El sistema escuchará un evento de subscripción a la sala de notificaciones.

RI4.6.2. El sistema escuchará un evento de comunicación de notificaciones.

RI4.6.3. El sistema escuchará un evento de desconexión de la sala de notificaciones.

5. Evaluación de alternativas

5.1. Desarrollo de la aplicación móvil en Android

5.1.1. Desarrollo nativo en Java

Java es el lenguaje principal utilizado a lo largo de todo el grado y en base a eso mismo es también el lenguaje con el que el equipo de desarrollo cuenta con **más experiencia y soltura**. Es también el primer lenguaje que fue seleccionado por Google para el desarrollo nativo en Android.

Esta opción ha sido largamente considerada puesto que, además, conforma el punto de conocimiento del equipo acerca del desarrollo de aplicaciones móviles por la asignatura[1] del grado centrada en esto. A pesar de esto, la siguiente alternativa, Kotlin, ofrece todas las capacidades de Java más algunos añadidos y mejoras que se han considerado suficientes para **desechar esta opción** respecto a la siguiente.

5.1.2. Desarrollo nativo en Kotlin

Kotlin es un lenguaje, desarrollado por **JetBrains** en 2011, de tipado estático que funciona sobre la Java Virtual Machine y cuya filosofía de creación es la mejora y extensión de Java manteniendo toda una interoperabilidad total con código Java. En el año 2017 fue nombrado por Google como **lenguaje oficial de Android**[4], y en 2019 lo declararon como el lenguaje preferido para dicho sistema operativo[5], sustituyendo a Java.

Los desarrolladores de Kotlin son los mismos que están detrás del **Android Studio**, el IDE por antonomasia para el desarrollo en Android, y esto permite que dicho entorno facilite en gran medida el paso de Java a Kotlin con auto-conversión de código entre otras cosas. Esta facilidad, más la oficialidad de Kotlin como lenguaje por excelencia de Android y el interés por parte del equipo de desarrollo de aprender nuevas herramientas en auge han auspiciado **la selección de esta alternativa** para construír la aplicación.

5.1.3. Desarrollo con el framework React Native

La otra opción que se barajó para el desarrollo de la aplicación fue la opción de utilizar algún **framework multiplataforma** que permitiese también el desarrollo de la

[1] SDM: Software para Dispositivos Móviles

aplicación para iOS o que, al menos, ofreciese otro lenguaje o sistema para el desarrollo. Uno de los frameworks más extendidos de este tipo es **React Native**.

React Native permite el desarrollo de la aplicación utilizando Javascript y React[2], que luego es compilado a código nativo de Android. El equipo de desarrollo cuenta con cierta experiencia en el uso del mismo, en base a la cual también se sabe que el uso de dichos frameworks no es óptimo y provoca muchos problemas derivados de trabajar con ellos. Puesto que iOs no es un requisito para la aplicación (e inviable para el equipo de desarrollo por carencia de dispositivos) se ha **descartado esta opción**.

5.2. Desarrollo de la API

5.2.1. Spring Framework

Spring es un framework de código abierto para el desarrollo de aplicaciones ejecutables sobre la JVM, admitiendo desarrollo en **Java y Kotlin**. A día de hoy es la plataforma base de la mayoría de aplicaciones empresariales desarrolladas en Java. Ofrece herramientas de sencilla ejecución como el despliegue sobre un servidor Apache o inyección de dependencias facilitando el desarrollo. Por defecto, usa JUnit como librería de pruebas.

Es una de las dos tecnologías con las que se ha aprendido a desarrollar aplicaciones web con controladores API REST en el grado[3], por lo que el equipo de desarrollo cuenta con experiencia en su uso. Aún con todo, y a pesar de admitir el desarrollo en el mismo lenguaje en que se desarrollará la aplicación móvil se ha decidido **descartar esta opción** pues las tecnologías y herramientas que ofrece para el desarrollo con WebSockets es más complicada que las librerías ofrecidas por otras alternativas.

5.2.2. Node.js con Express y Socket.io

Node.js es un entorno multiplataforma para el desarrollo de servidores y aplicaciones basado en el **Javascript**, aunque acepta otros lenguajes compilables a Javascript como **Typescript**. El desarrollo en Node está basado en un amplio entorno de librerías. La librería más extendida para la creación de API RESTs es **Express**, que será la considerada en esta alternativa. Para el trabajo con WebSockets se considerará **Socket.io** y para las pruebas se elegirá **Jest**, ambas son librerías conocidas por la facilidad de

[2]Librería de Javascript para el desarrollo de interfaces de usuario desarrollada por Facebook
[3]En la asignatura de Sistemas Distribuidos e Internet (SDI)

uso.

Node es la otra tecnología con la que se ha aprendido a trabajar en el grado[4], por lo que, de nuevo, el equipo de desarrollo cuenta con experiencia en este campo. Debido a la agilidad que ofrece y la sencillez de manejo de la librería de WebSockets (así como el que esta cuenta con una librería para implementar el cliente en Android) se **ha estimado elegir esta alternativa**. La API se desarrollará en Node.js utilizando Typescript y con Express, Socket.io y Jest como librerías de enrutamiento REST, de WebSocket y de testing, respectivamente.

5.2.3. Micronaut

Micronaut es también un framework open-source para el desarrollo de aplicaciones sobre la JVM que admite desarrollo en **Java, Groovy y Kotlin**. Micronaut ofrece una revisión sobre frameworks más antiguos como Spring haciendo énfasis en un diseño orientado a la nube u optimizaciones varias, por ejemplo, haciendo la inyección de dependencias con funcionamiento bajo demanda para reducir el peso y el tiempo de despliegue o testing (basado también en JUnit).

Una de las ventajas de Micronaut, su modernidad, es también uno de sus mayores defectos pues carece de la amplia documentación o ejemplos que sí tenían las otras dos alternativas gracias a su longevidad y amplio uso. Es por este motivo por el que el equipo de desarrollo, aún teniendo cierta experiencia con el mismo, ha decidido **descartar Micronaut** para reducir en lo posible los bloqueos derivados del uso del framework.

5.3. Tecnología de geolocalización

5.3.1. OsmDroid

OpenStreetMap es un proyecto colaborativo de mapeado global y de software libre que construye sus mapeados gracias a la aportación de los dispositivos GPS y a los añadidos manuales de sus contribuyentes. OpenStreetMap ofrece una API gratuita para la consulta y acceso a los datos albergados y a su mapa digital.

La librería más completa para trabajar con OpenStreetMap en Android es **OsmDroid**, que logra un reemplazamiento casi completo de la primera versión de la API

[4]También en la asignatura de Sistemas Distribuidos e Internet (SDI)

de mapas de Android con su código abierto. Sin embargo, esta librería aún no tiene el pulido de sus hermanas web y cuenta con menos documentación o ejemplos que la librería oficial de Google. Por lo que **no se considera** su uso contra esta.

5.3.2. Google Maps for Android

Google Maps es la API más extendida y usada del mundo para la implementación de mapas, sistemas de geolocalización o de navegación, entre otras cosas. Su precio varía según la plataforma donde se use y según las herramientas del mismo o el número de llamadas que se hagan a la API. Sin embargo, en el momento del desarrollo del proyecto **es gratuita para las aplicaciones de Android** con sus servicios básicos.

El único servicio que está planeado para la aplicación es el de la visualización del mapa en las coordenadas especificadas y este se encuentra incluido en el paquete gratuito. Además, la empresa desarrolladora de Android y de la API son la misma y de ello deriva que la librería que la emplea sea la más completa y funcional del sistema operativo. Por estas razones se decidió **optar por estar alternativa**.

5.4. Sistema de gestión de bases de datos

De cara a la selección de un sistema de gestión de bases de datos, debido al tipo de información que se manejarán en los chats, se seleccionó una **base de datos documental** como la mejor opción para el desarrollo de esta aplicación, dos gestores de este tipo fueron considerados:

5.4.1. MongoDB

MongoDB es la base de datos por excelencia en el mundo de las bases de datos documentales e incluso lidera ampliamente el conjunto de todos los sistemas de bases de datos NoSQL[6]. MongoDB es un proyecto de código abierto y es compatible tanto con la nube privada, mediante el despliegue de la base de datos en los servidores del sistema que la vaya a utilizar; como con la nube pública, por medio del servicio **MongoDB Atlas**.

MongoDB Atlas ofrece niveles del servicio gratuitos y, además, un crédito de 500$ para estudiantes. Los requisitos iniciales del sistema son compatibles con los rangos que permiten estas dos opciones, sumado a su extendido uso, a la experiencia del equipo de desarrollo y facilidad de implementación en Node.js, se ha **elegido esta alternativa**.

5.4.2. Firebase

Firebase es una plataforma para el desarrollo móvil y web de Google que ofrece una serie de herramientas como un sistema de autenticación o una base de datos documental entre otras cosas. Está ampliamente documentada e implementada en gran número de aplicaciones Android.

Su sistema de gestión de bases de datos es documental en formato JSON y se llama **Realtime Database**. Sin embargo, y a pesar de que su coste sería gratuito para los propósitos iniciales de *All for One*, la serie de pasos que requiere para su uso en el sistema son un punto negativo respecto a la otra alternativa considerada y en base a esto, **no será la opción empleada**.

5.5. Nube para el despliegue del servidor

5.5.1. Amazon Web Services

Amazon Web Services (o AWS) es el servicio líder en cloud en el mundo[7]. Derivado de esto se extiende una gran cantidad de documentación acerca de su uso y un servicio que se puede prever fiable. Además, el equipo de desarrollo ya ha desplegado anteriormente aplicaciones en este servicio. Lamentablemente, su coste es un problema para este desarrollo carente de inversión, por lo que se **ha decidido buscar otras alternativas**.

5.5.2. Microsoft Azure

Tras AWS, Azure de Microsoft es el siguiente servicio con mayor cuota de mercado[7], esto se traduce en unas virtudes similares a los nombrados en el punto anterior. Aunque el equipo de desarrollo no ha llevado a cabo ningún despliegue anterior sí ha acudido a un curso de formación[5] relacionado con el mismo, por lo que tiene un conocimiento mucho mayor que con cualquier otro proveedor. Además, los estudiantes pueden recibir un crédito de 100$ que puede ser suficiente para el despliegue necesario en este proyecto. La suma de todo esto ha convertido a Azure **en la opción elegida**.

[5] **Fundamentos de Microsoft Azure**, impartido por la Universidad de Oviedo en julio de 2021

5.5.3. Heroku

Puesto que el coste económico es el factor determinante en la decisión del proveedor de una nube para el despliegue del servidor, también se tuvo en amplia consideración otra alternativa, a pesar de carecer de total experiencia con ella. Heroku carece del mercado de las dos otras alternativas, pero ofrece un servicio de hosting para aplicaciones gratuito que habría sido suficiente para este proyecto. Aún con este punto positivo se decidió **optar por otra alternativa** más popular actualmente y que ofreciese un aprendizaje más útil de cara a la futura e inminente carrera profesional.

6. Análisis de riesgos

A continuación se presentan los cuatro riesgos del proyecto detectados por el equipo de desarrollo. La descripción, análisis y respuesta al riesgo se han realizado siguiendo las indicaciones del PMBOK[8].

6.1. Descripción de riesgos detectados

6.1.1. Incapacidad de obtención de colaboración con asociaciones

- **Identificador**: Rsk-1
- **Categoría**: Externo
- **Subcategoría**: Subcontratistas y proveedores

Este riesgo parte de experiencia previa al comienzo del proyecto, basada en contactos preliminares que se intentaron antes de proponer la idea del mismo.

El posible riesgo radicaría en no poder contar con el apoyo o colaboración de alguna asociación de pacientes de Alzheimer que pudiese aportar una opinión experta o consejos de utilidad para el diseño de la aplicación. Es un riesgo concretamente por la **falta de formación específica** en este área del equipo de desarrollo, algo que podría perjudicar la calidad del producto entregado.

Además, la alianza con una asociación de este estilo podría ser necesaria de cara a la realización de pruebas de usabilidad del sistema generado con usuarios objetivo. En caso de no obtenerla, el proyecto tendría que ser entregado **sin una prueba de campo adecuada** a los objetivos y pretensiones iniciales.

6.1.2. Inexperiencia del equipo de proyecto

- **Identificador**: Rsk-2
- **Categoría**: Dirección de Proyectos
- **Subcategoría**: Estimación y Planificación

Actualmente, la única experiencia del equipo de proyecto en la dirección y gestión

de proyectos es el de la simulación realizada en la asignatura de DPPI[1]. No existe una **experiencia práctica real** y esto puede manifestarse en forma de cálculos erróneos en la planificación del proyecto.

El riesgo radica en la posibilidad de aparición de estos errores. Como podrían ser, por ejemplo, **malas estimaciones** del tiempo necesario para completar tareas. Las desviaciones de esto desembocarían en grandes problemas de la planificación y también afectaría, en cascada, al cálculo del presupuesto o al alcance y calidad conseguibles por el sistema.

6.1.3. Carencia de fondos

- **Identificador**: Rsk-3
- **Categoría**: Organizativo
- **Subcategoría**: Financiación

El riesgo en sí mismo no es la carencia de fondos, puesto que eso ya es una característica conocida y definida de este proyecto. Sin embargo, sigue existiendo el riesgo de que dicha carencia de fondos sea problemática de cara a la consecución de la calidad deseada en el proyecto. La imposibilidad de costearse un servicio de alojamiento o de pagar ciertas API o servicios de pago pueden suponer **complicaciones no previstas** para el proyecto inicialmente.

6.1.4. Falta de formación en tecnologías clave

- **Identificador**: Rsk-4
- **Categoría**: Técnico
- **Subcategoría**: Tecnología

En el desarrollo se utilizarán varias tecnologías con las que no se cuenta con experiencia de desarrollo previo. Algunas de estas **son clave** para el funcionamiento del sistema. Este hecho supone un riesgo al poder provocar impedimentos o retrasos en la generación de los componentes que trabajen con dichas tecnologías. Así como en errores de funcionamiento o concepto derivados del desconocimiento de las capacidades completas de las tecnologías a manejar.

[1]Dirección y Planificación de Proyectos Informáticos

6.2. Análisis de los riesgos

Riesgo	Probabilidad	Impacto				Impacto
		Presupuesto	Planificación	Alcance	Calidad	
Colaboración con asociaciones	Alta	Mínima	Medio	Alto	Alta	0.39
Inexperiencia del equipo	Media	Alto	Crítico	Medio	Bajo	0.45
Carencia de fondos	Media	Crítico	Bajo	Medio	Medio	0.45
Formación en tecnologías	Media	Bajo	Alto	Bajo	Alto	0.28

Tabla 6.1: Probabilidades y impacto de los riesgos

En base a este análisis del impacto global de los riesgos detectados se ha resaltado la inexperiencia del equipo de proyecto y la carencia de fondos como los riesgos de **mayor prioridad** a paliar, pues son aquellos que pueden afectar en mayor medida al éxito del proyecto.

6.3. Respuesta a los riesgos

Para los distintos riesgos detectados se han propuesto las siguientes respuestas:

- **Inexperiencia del equipo de proyecto**. Puesto que no existe la posibilidad de contratar una opinión más formada ni de adquirir la experiencia necesaria para paliar los posibles errores. Se ha decidido **mitigar** el riesgo. Se tendrá en cuenta desde el principio la posibilidad de fallar en el desarrollo de la planificación. Se tomarán medidas cautelares al respecto como dejar margen de tiempo prudencial antes de la fecha límite o guardar tiempo de vacaciones por si es necesario una dedicación extra para compensar los errores.

- **Carencia de fondos**. En respuesta a este riesgo se ha decidido buscar la mayor **mitigación** posible por medio de la aceptación del uso de tecnologías de pago cuando sea necesario y existan de beneficios de estudiante que los puedan costear.

- **Incapacidad de obtención de colaboración con asociaciones**. Ante la situación actual con la pandemia y la imposibilidad de retrasar el desarrollo en esperas o aras de conseguir la ansiada colaboración se ha decidido **asumir** la posibilidad de no lograr dicho apoyo.

- **Falta de formación en tecnologías clave**. Para afrontar este problema se ha decidido intentar **eliminar** el riesgo por medio de llevar a cabo formación específica en las tecnologías que se consideren oportunas. La idea es adquirir suficiente bagaje antes de comenzar el desarrollo para reducir al mínimo los riesgos de la inexperiencia del equipo desarrollador.

7. Descripción de la solución propuesta

7.1. Aplicación móvil

La aplicación móvil a desarrollar será una aplicación nativa de Android desarrollada con la última versión de Kotlin estable al momento de comienzo del desarrollo, **Kotlin 1.5.0**. El código Kotlin será compilado con la JVM 1.8 como destino. El SDK de Android objetivo de la aplicación será el SDK 31, kit de desarrollo de la versión más reciente del sistema operativo, Android 12. Sin embargo, el SDK mínimo soportado será el de **Android 8.1 Oreo** (SDK 26), este abanico equivale a ofrecer un sistema compatible con más del 80 % de dispositivos Android actualmente en el mercado[9].

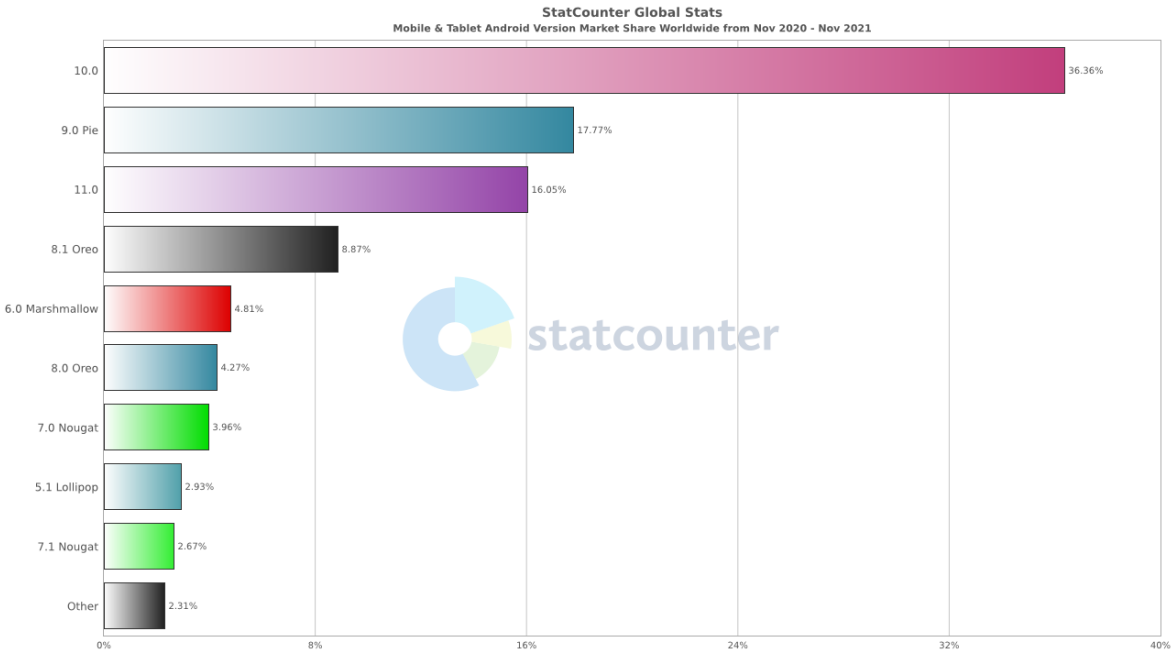


Figura 7.1: Reparto de mercado de Android (nov 20 - nov 21)

Dicho desarrollo se realizará utilizando como herramienta principal el IDE Android Studio. La arquitectura de la aplicación será una **arquitectura MVVM**, modelo de arquitectura recomendado actualmente para el desarrollo de aplicaciones Android[10]. Las vistas se realizarán siguiendo las recomendaciones de **Material Design** (Apartado 25.3.2). Las pruebas del sistema se llevarán a cabo con **JUnit 4** para las pruebas unitarias de componentes y con **Espresso** para las pruebas de comportamiento de las vistas.

7.1.1. Autenticación

La autenticación de los usuarios en la aplicación se llevará a cabo siguiendo el protocolo **OAuth 2.0**[11]. Dicha autenticación será lograda con apoyo en la API de autenticación de Google, de esta forma el usuario podrá iniciar sesión con la cuenta Google que ya tiene vinculada al dispositivo o cualquier otra que desee usar. En una primera versión del sistema no se valora añadir otros métodos o vías de autenticación.

7.1.2. Geolocalización

Para la función de geolocalización se hará uso de los dispositivos GPS montados en los terminales de los usuarios. La obtención de la localización y su muestra en un mapa se llevarán a cabo con la **el kit de desarrollo de mapas de Google** (Apartado 24.2.1.10) para Android. La localización del usuario se comunicará al servidor por vía del WebSocket cuando comience a compartirla, por la misma vía se obtendrá también la localización del resto de usuarios vinculados.

7.1.3. Comunicación con la API

La comunicación por red con la API se llevará a cabo haciendo uso de la librería **Retrofit 2** (Apartado 24.2.1.11) de Square Open Source para la realización de peticiones REST. La comunicación por medio de los WebSockets se llevará a cabo con el **cliente para Android de Socket.io** (Apartado 24.2.1.12). El intercambio de datos en ambos protocolos hará uso de Gson[1], la librería de Google para la conversión de objetos Java a JSON y viceversa. En los tests la simulación de la comunicación de la API se realizará con el MockWebServer (Apartado 24.2.1.8) de Square Open Source.

7.1.4. Vinculación de usuarios

Para la realización de una vinculación de usuarios fiable a la vez que cómoda se realizará por medio del escaneo de códigos QR. Para gestionar la creación y escaneo de los mismos se utilizará **Zxing**[2], librería de Google para la codificación y decodificación de códigos de barras, códigos QR y similares.

[1] https://github.com/google/gson
[2] https://github.com/zxing/zxing

7.2. API

El servidor con la API y la lógica de negocio del sistema será desarrollada utilizando la última versión estable de Node.js, **Node v16**[3]. Para fomentar la creación de código más mantenible y resistente a errores se usará como lenguaje de programación **Typescript** (Apartado 24.1.4). El código Typescript será compilado a EcmaScript 6[12] con el módulo CommonJS. Para las pruebas se usará la librería **Jest** (Apartado 24.2.2.9), compatibilizada con Typescript por medio de TSJest (Apartado 24.2.2.17).

El enrutamiento y el manejo de las peticiones HTTP de la API REST será desarrollado y gestionado con la librería **Express** (Apartado 24.2.2.5). Sus pruebas serán manejadas con la librería SuperTest (Apartado 24.2.2.16). Por otro lado, los WebSockets serán manejados por **Socket.io** (Apartado 24.2.2.15).

7.2.1. Autenticación

La autenticación en el servidor para el uso de la API REST será implementado por medio de Bearer tokens[13]. Se manejarán dos tipos de token. El inicio de sesión se llevará a cabo por medio de **tokens de autenticación de Google** servidos por la app y que se verificarán por medio de Google Auth Library (Apartado 24.2.2.6). Una vez iniciada la sesión se proporcionarán tokens de autenticación y refresco para la realización de peticiones privadas a la API REST, estos tokens serán **Json Web Token**[14]. La librería de NPM[4] de JWT será la que se usará para la creación, validación y verificación de estos tokens.

7.2.2. Persistencia de datos

La persistencia de datos se realizará con una base de datos documental de MongoDB alojada en la nube en remoto de **MongoDB Atlas**[5]. Para el manejo de la base de datos se usará la librería Mongoose (Apartado 24.2.2.13). Para utilizarla con todas las fortalezas de Typescript se usará la librería **Typegoose** (Apartado 24.2.2.21). De cara a la ejecución de tests con persistencia se usará la librería de MongoDB Memory Server (Apartado 24.2.2.12) para replicar la base de datos en memoria.

[3] https://nodejs.org/en/about/releases
[4] Gestor de paquetes de Node.js
[5] https://www.mongodb.com/atlas/database

7.3. Despliegue

La API del sistema será desplegada en la nube haciendo uso del servicio **Microsoft Azure**. Puesto que los requisitos de uso del sistema en este proyecto serán mínimos se usará una **AppService de nivel B1** financiada por el crédito para estudiantes. Para el manejo del despliegue se creará una pipeline de despliegue continuo con GitHub Actions (Apartado 25.2.2) que actualice el sistema desplegado cuando se actualice el código de la rama principal del sistema.

8. Planificación temporal

De cara a la realización de una planificación temporal se ha determinado un calendario de trabajo para el equipo de desarrollo implicado, en el proyecto que nos ocupa este equipo cuenta únicamente con un desarrollador. El desarrollador estará empleado en un trabajo ajeno al proyecto a jornada completa durante todo el transcurso del mismo. Los horarios serán diseñados en torno a dicha situación. El calendario resultante es el siguiente:

- Lunes a viernes: de 19:00 a 20:00
- Sábados: de 10:00 a 14:00
- Domingos: de 10:00 a 13:00

En total se establecerá un horario de trabajo semanal de 12 horas. Los descansos necesarios son incluidos en dichas jornadas. No se tendrán en cuenta festivos pero sí habrá días de parones por exceso de horas de trabajo acumuladas entre proyecto y empleo del desarrollador. Tanto la formación necesaria para el desarrollo del sistema como el tiempo dedicado al desarrollo de pruebas estará incluida en el tiempo de implementación de la características relacionadas, pues son labores paralelas al desarrollo de las mismas.

El inicio de desarrollo del proyecto ha sido establecido en el día de **1 de mayo de 2021**. Su finalización estimada tras el cálculo estimado será el día **15 de enero de 2022**. En resumen, el equipo de desarrollo se espera que dedique un máximo de doce horas semanales a lo largo de treinta y cinco semanas. La planificación temporal global es la ilustrada en el Cuadro 8.1, una más detallada se puede encontrar en los cuadros posteriores.

Código	Hito	Duración (h)	Predecesores	Desglose
1	Arranque del proyecto	7		8.2
2	Planificación	31	1	8.3
3	Análisis	36	2	8.4
4	Diseño	46	3	8.5
5	Implementación	278	4	8.6
6	Elaboración de manuales	9		8.7
7	Clausura del proyecto	13	5, 6	8.8

Tabla 8.1: Planificación temporal global

Código	Tarea	Duración (h)	Predecesoras
1	*Arranque del proyecto*	7	
1.1	Justificación	2	
1.2	Objetivos	1	
1.3	Estudio de la situación actual	4	

Tabla 8.2: Planificación temporal del arranque de proyecto

Código	Tarea	Duración (h)	Predecesoras
2	*Planificación*	31	1
2.1	Requisitos iniciales	2	
2.2	Evaluación de alternativas	8	2.1
2.3	Definición de la solución	3	2.2
2.4	Planificación temporal	8	2.3
2.5	Elaboración de presupuesto	10	2.4

Tabla 8.3: Planificación temporal de la planificación del proyecto

Código	Tarea	Duración (h)	Predecesoras
3	*Análisis*	36	2
3.1	Identificación de requisitos	12	
3.2	Identificación de subsistemas	4	
3.3	Especificación de casos de uso	5	
3.4	Bocetado de interfaces de usuario	4	3.3
3.6	Propuesta de clases del sistema	8	3.1, 3.2, 3.4
3.7	Especificación del plan de pruebas	1	3.2
3.8	Definición de plan de despliegue	2	3.2

Tabla 8.4: Planificación temporal del análisis del sistema

Código	Tarea	Duración (h)	Predecesoras
4	*Diseño*	46	3
4.1	Definición de la arquitectura	10	
4.2	Diseño de clases	20	4.2
4.3	Especificación del modelo de datos	2	4.3
4.4	Especificación de las interfaces de comunicación	4	4.4
4.5	Diseño de interfaces de usuario	10	4.4

Tabla 8.5: Planificación temporal del diseño del sistema

Código	Tarea	Duración (h)	Predecesoras
5	*Implementación*	278	4
5.1	Prototipado	6	
5.2	REST API	53	
5.2.1	Autenticación	6	
5.2.2	Usuarios	15	5.2.1
5.2.3	Feed	22	5.2.2
5.2.4	Tareas	10	5.2.2
5.3	WebSocket API	39	
5.3.1	Global	8	5.2.1
5.3.2	Mensajería	17	5.2.3, 5.2.4
5.3.3	Localización	2	5.2.1
5.3.4	Notificación	12	5.2.3
5.4	Despliegue del servidor	14	5.2, 5.3
5.5	Aplicación móvil	166	
5.5.1	Inicio y cierre de sesión	11	5.2.1
5.5.2	Pantalla principal	9	5.5.1
5.5.3	Vinculación	20	5.22
5.5.4	Listado de vínculos	15	5.22
5.5.5	Geolocalización	29	5.3.3
5.5.6	Feed	37	5.2.3, 5.2.4, 5.3.2
5.5.7	Gestión de tareas	25	5.2.4
5.5.8	Notificaciones	20	5.2.3, 5.3.4

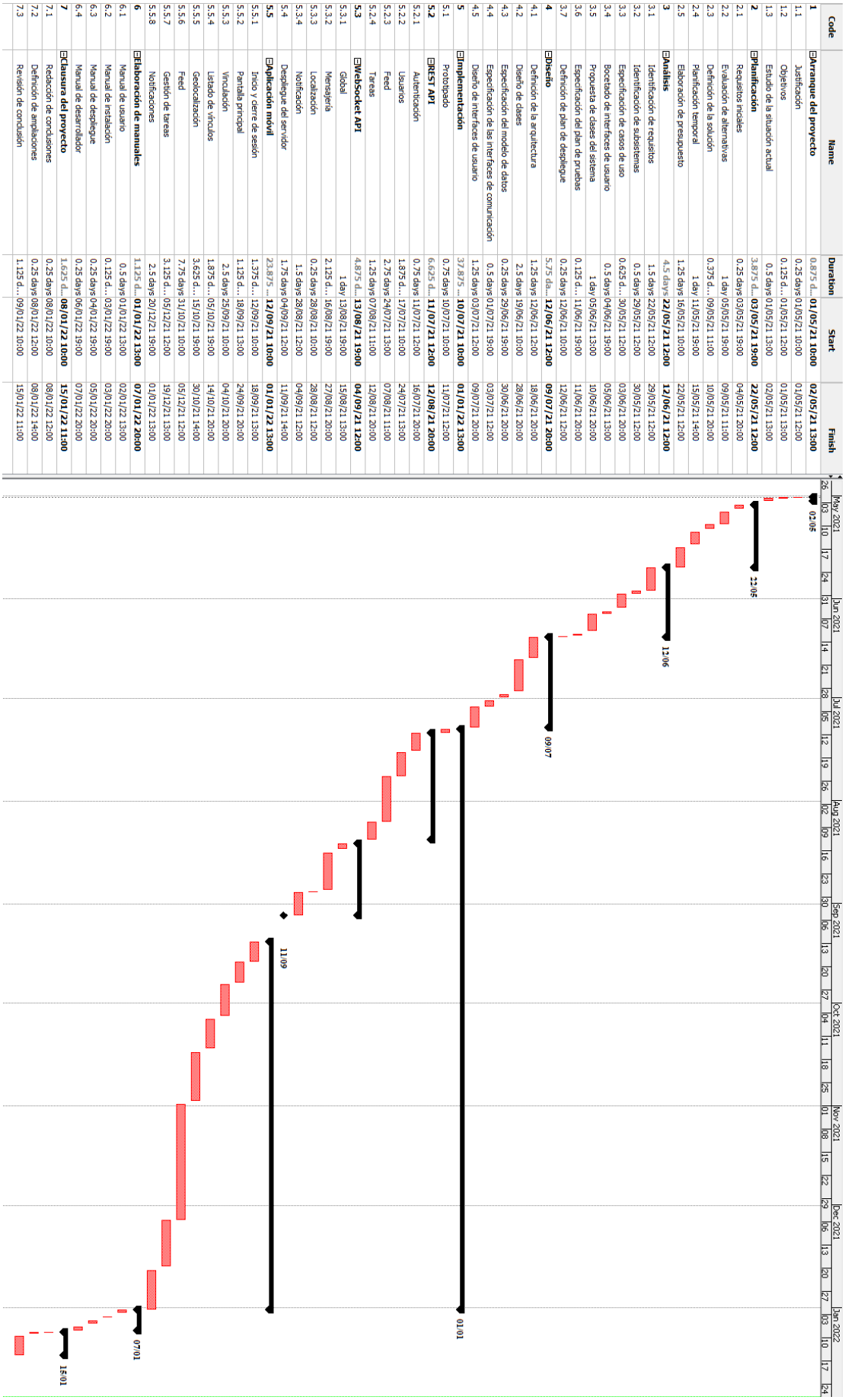
Tabla 8.6: Planificación temporal de la implementación del sistema

Código	Tarea	Duración (h)	Predecesoras
6	*Elaboración de manuales*	9	
6.1	Manual de usuario	4	5.5
6.2	Manual de instalación	1	5.5
6.3	Manual de despliegue	2	5.4
6.4	Manual de desarrollador	2	5

Tabla 8.7: Planificación temporal de la elaboración de manuales

Código	Tarea	Duración (h)	Predecesoras
7	*Clausura del proyecto*	13	6
7.1	Redacción de conclusiones	2	
7.2	Definición de ampliaciones	2	
7.3	Revisión de conclusión	9	7.1, 7.2

Tabla 8.8: Planificación temporal de la clausura del proyecto



Code	Name	Duration	Start	Finish
1	**Arranque del proyecto**	0.875 d...	**01/05/21 10:00**	**02/05/21 13:00**
1.1	Justificación	0.25 days	01/05/21 10:00	01/05/21 12:00
1.2	Objetivos	0.125 d...	01/05/21 12:00	01/05/21 13:00
1.3	Estudio de la situación actual	0.5 days	01/05/21 13:00	02/05/21 13:00
2	**Planificación**	3.875 d...	**03/05/21 19:00**	**22/05/21 12:00**
2.1	Requisitos iniciales	0.25 days	03/05/21 19:00	04/05/21 20:00
2.2	Evaluación de alternativas	1 day	05/05/21 19:00	09/05/21 11:00
2.3	Definición de la solución	0.375 d...	09/05/21 11:00	10/05/21 20:00
2.4	Planificación temporal	1 day	11/05/21 19:00	15/05/21 14:00
2.5	Elaboración de presupuesto	1.25 days	16/05/21 10:00	22/05/21 12:00
3	**Análisis**	4.5 days	**22/05/21 12:00**	**12/06/21 12:00**
3.1	Identificación de requisitos	1.5 days	22/05/21 12:00	29/05/21 12:00
3.2	Identificación de subsistemas	0.5 days	29/05/21 12:00	30/05/21 12:00
3.3	Especificación de casos de uso	0.625 d...	30/05/21 12:00	03/06/21 20:00
3.4	Bocetado de interfaces de usuario	0.5 days	04/06/21 19:00	05/06/21 13:00
3.5	Propuesta de clases del sistema	1 day	05/06/21 13:00	10/06/21 20:00
3.6	Especificación del plan de pruebas	0.125 d...	11/06/21 19:00	11/06/21 20:00
3.7	Definición de plan de despliegue	0.25 days	12/06/21 10:00	12/06/21 12:00
4	**Diseño**	5.75 da...	**12/06/21 12:00**	**09/07/21 20:00**
4.1	Definición de la arquitectura	1.25 days	12/06/21 12:00	18/06/21 20:00
4.2	Diseño de clases	2.5 days	19/06/21 10:00	28/06/21 20:00
4.3	Especificación del modelo de datos	0.25 days	29/06/21 19:00	30/06/21 20:00
4.4	Especificación de las interfaces de comunicación	0.5 days	01/07/21 19:00	03/07/21 12:00
4.5	Diseño de interfaces de usuario	1.25 days	03/07/21 12:00	09/07/21 20:00
5	**Implementación**	37.875 ...	**10/07/21 10:00**	**01/01/22 13:00**
5.1	Prototipado	0.75 days	10/07/21 10:00	11/07/21 12:00
5.2	**REST API**	6.625 d...	**11/07/21 12:00**	**12/08/21 20:00**
5.2.1	Autenticación	0.75 days	11/07/21 12:00	16/07/21 20:00
5.2.2	Usuarios	1.875 d...	17/07/21 10:00	24/07/21 13:00
5.2.3	Feed	2.75 days	24/07/21 13:00	07/08/21 11:00
5.2.4	Tareas	1.25 days	07/08/21 11:00	12/08/21 20:00
5.3	**WebSocket API**	4.875 d...	**13/08/21 19:00**	**04/09/21 12:00**
5.3.1	Global	1 day	13/08/21 19:00	15/08/21 13:00
5.3.2	Mensajería	2.125 d...	16/08/21 19:00	27/08/21 20:00
5.3.3	Localización	0.25 days	28/08/21 10:00	28/08/21 12:00
5.3.4	Notificación	1.5 days	28/08/21 12:00	04/09/21 12:00
5.4	Despliegue del servidor	1.75 days	04/09/21 12:00	11/09/21 14:00
5.5	**Aplicación móvil**	23.875 ...	**12/09/21 10:00**	**01/01/22 13:00**
5.5.1	Inicio y cierre de sesión	1.375 d...	12/09/21 10:00	18/09/21 13:00
5.5.2	Pantalla principal	1.125 d...	18/09/21 13:00	24/09/21 20:00
5.5.3	Vinculación	2.5 days	25/09/21 10:00	04/10/21 20:00
5.5.4	Listado de vínculos	1.875 d...	05/10/21 19:00	14/10/21 20:00
5.5.5	Geolocalización	3.625 d...	15/10/21 19:00	30/10/21 14:00
5.5.6	Feed	7.75 days	31/10/21 10:00	05/12/21 12:00
5.5.7	Gestión de tareas	3.125 d...	05/12/21 12:00	19/12/21 13:00
5.5.8	Notificaciones	2.5 days	20/12/21 19:00	01/01/22 13:00
6	**Elaboración de manuales**	1.125 d...	**01/01/22 13:00**	**07/01/22 20:00**
6.1	Manual de usuario	0.5 days	01/01/22 13:00	02/01/22 13:00
6.2	Manual de instalación	0.125 d...	03/01/22 19:00	03/01/22 20:00
6.3	Manual de despliegue	0.25 days	04/01/22 19:00	05/01/22 20:00
6.4	Manual de desarrollador	0.25 days	06/01/22 19:00	07/01/22 20:00
7	**Clausura del proyecto**	1.625 d...	**08/01/22 10:00**	**15/01/22 11:00**
7.1	Redacción de conclusiones	0.25 days	08/01/22 10:00	08/01/22 12:00
7.2	Definición de ampliaciones	0.25 days	08/01/22 12:00	08/01/22 14:00
7.3	Revisión de conclusión	1.125 d...	09/01/22 10:00	15/01/22 11:00

Figura 8.1: Diagrama de Gantt del proyecto

9. Resumen del presupuesto

El desarrollo de este proyecto **carece de perspectiva comercial o de negocio**. No existe ningún porcentaje de beneficio aplicado sobre el coste del proyecto, no se aplica el impuesto de valor añadido y el margen entre el coste total y la facturación se ha reducido a un escueto 1.89 %.

El equipo de desarrollo cuenta con **un único miembro** y en base a ello se ha considerado el desarrollo como un proyecto *freelance* sin declaración de autónomo. A pesar de existir un único miembro encargado del desarrollo, los costes de las diferentes partidas y tareas se realizaron teniendo en cuenta el cargo que se personará en cada caso concreto en vez de utilizar un único salario y rol general. Los salarios utilizados están basados en las medias encontradas en el portal web Glassdoor[1], con una reducción compensatoria de la inexperiencia del desarrollador.

De cara al cálculo de costes indirectos y de medios de producción del equipo de desarrollo se tomaron en cuenta: el cálculo estimado de consumo eléctrico del portátil usado en el desarrollo, una única licencia de pago[2] puesto que el resto de softwares utilizados son gratuitos, el coste del despliegue del servidor de prueba durante el desarrollo y el coste por hora en base a la amortización de los teléfonos y ordenador empleados durante el desarrollo y que ya se poseían de antes.

A continuación se ofrece el resumen del presupuesto. El desglose detallado de este y sus partidas se puede encontrar en el Capítulo 35 Presupuesto y en la hoja de cálculo adjunta a este documento y que se indica en el Capítulo 39. Cada partida listada en este resumen contiene también la referencia de su partida detallada en dicho anexo.

Código	Partida	Desglose	Importe
1	Planificación	Cuadro 35.2	1 149.00€
2	Análisis	Cuadro 35.3	1 174.00€
3	Diseño	Cuadro 35.4	1 500.00€
4	Construcción	Cuadro 35.5	6 957.00€
5	Formación	Cuadro 35.6	197.00€
	TOTAL		**10 977.00€**

Tabla 9.1: Resumen de presupuesto

El coste total del proyecto es de **10 977.00€**

[1] https://www.glassdoor.es/member/home/index.htm
[2] Licencia de Office 365 para la elaboración del presupuesto con Microsoft Excel

Parte III

Análisis

10. Requisitos del sistema

10.1. Identificación de actores del sistema

10.1.1. Paciente

El **Paciente** o **Patient** es la figura base alrededor de la que orbita el sistema. Los Pacientes son los usuarios de la aplicación que serán auxiliados por los Cuidadores. Un paciente podrá tener hasta un máximo de seis Cuidadores.

10.1.2. Cuidador

Los usuarios de tipo **Cuidador** o **Keeper** serán aquellos que usen la aplicación con ánimo de ayudar a un Paciente. Los Cuidadores sólo podrán tener un Paciente vinculado, pero estarán en contacto con los demás Cuidadores de dicho Paciente.

10.1.3. Administración

Debido a las funciones y carácter cerrado de la aplicación se ha estimado que una figura o rol de administración dentro del sistema sería innecesario. No es necesario ninguna clase de control ni gestión o comprobación a gran escala que exija una moderación interna. Todos los conflictos que puedan surgir por parte de los usuarios podrían ser resueltos por medio de modificaciones directas a la base de datos, que ya cuenta con su propio portal de administración, haciendo innecesario el desarrollo de ningún sistema dedicado para las labores de mantenimiento y administración.

10.1.4. Identificación de relación de actores

Entre los actores relevantes del sistema existen dos tipos de relación que serán nombradas en el documento:

- **Vínculo**. Relación entre un Paciente y un Cuidador que han sido enlazados. Los usuarios vinculados de un Paciente son sus Cuidadores. El único usuario vinculado de un Cuidador es su Paciente.

- **Asociación**. Relación entre Cuidadores del mismo Paciente, súper relación de Vínculo. Los usuarios asociados de un Paciente son sus Cuidadores. Los usuarios asociados de un Cuidador son el resto de Cuidadores del paciente y el propio Paciente.

10.2. Identificación de requisitos

10.2.1. RSU. Requisitos de sesión de usuario

RSU 1. Los usuarios podrán iniciar sesión en la aplicación mediante autenticación con su cuenta de Google.

RSU 2. Los inicios de sesión serán validados con una petición a la API REST con el token obtenido de la validación con Google.

RSU 2.1. Si el inicio de sesión es inválido el usuario será notificado del error y se le permitirá reintentarlo.

RSU 2.2. Si el inicio de sesión es válido y el usuario ya está registrado, se creará la sesión del usuario.

RSU 2.2.1. La API REST creará y almacenará la sesión del usuario. Dicha sesión estará compuesta por:

RSU 2.2.1.1. Token de sesión para realizar las peticiones. Tendrá un tiempo de validez de TIEMPO_VALIDEZ_TOKEN_SESIÓN.

RSU 2.2.1.2. Token de refresco de sesión para mantener la sesión activa y obtener un nuevo par de tokens. Tendrá un tiempo de validez de TIEMPO_VALIDEZ_TOKEN_REFRESCO.

RSU 2.2.2. La sesión pasará a ser inválida cuando caduque su RSU 2.2.1.2..

RSU 2.2.3. La API REST enviará a la aplicación la información de sesión del usuario.

RSU 2.2.3.1. Token de sesión (RSU 2.2.1.1.).

RSU 2.2.3.2. Token de refresco de sesión (RSU 2.2.1.2.).

RSU 2.2.3.3. Instante de caducidad de la sesión.

RSU 2.2.4. El usuario será redirigido a la pantalla principal de la aplicación.

RSU 2.3. Si el inicio de sesión es válido y el usuario es nuevo se le redirigirá a la actividad de creación de usuarios.

RSU 2.3.1. El usuario deberá indicar un nombre válido.

RSU 2.3.2. El usuario deberá seleccionar un rol entre los posibles:

RSU 2.3.2.1. Paciente

RSU 2.3.2.2. Cuidador

RSU 2.3.3. El usuario podrá añadir información adicional a su perfil.

- Teléfono de contacto
- Teléfono de contacto alternativo

- Dirección de correo electrónico
- Dirección física. Conformada por dos campos para componer la dirección postal, un campo para la localidad y un campo para la región.

RSU 2.3.4. La aplicación pedirá confirmación al usuario de los datos insertados.

RSU 2.3.4.1. Si el usuario confirma los datos insertados, los datos serán enviados a la API REST para su almacenamiento y validación.

RSU 2.3.4.2. Si el usuario no es válido, la API REST responderá con un mensaje de error.

RSU 2.3.4.3. Si el usuario es válido, la API REST responderá a la petición con un mensaje de éxito y el perfil completo del usuario.

RSU 2.3.4.4. Tras la creación exitosa del perfil, el usuario será dado de alta siguiendo RSU 2.2..

RSU 2.4. La sesión podrá ser renovada con una solicitud a la API REST con el token de refresco de sesión mientras este siga siendo válido.

RSU 2.4.1. Refrescar una sesión generará una tupla como la de RSU 2.2.1. y sustituirá la almacenada anteriormente.

RSU 2.4.2. La API REST responderá con la información de la sesión de usuario como en RSU 2.2.3..

RSU 2.4.3. Al refrescar una sesión los tokens anteriores quedarán invalidados.

RSU 3. Los usuarios con la sesión iniciada podrán cerrar sesión en la aplicación.

RSU 3.1. La aplicación enviará la petición de cierre de sesión a la API REST con el token de autenticación.

RSU 3.2. La API REST eliminará la sesión almacenada y confirmará la acción a la aplicación.

RSU 3.3. Si el cierre de sesión es exitoso el usuario será redirigido a la pantalla de inicio de sesión.

10.2.2. RGU. Requisitos de gestión de usuarios

RGU 1. Los usuarios podrán ver la información del Paciente vinculado en la pantalla principal.

RGU 1.1. La información a mostrar será suministrada por la API REST.

RGU 1.2. En el caso de Pacientes se les mostrará su propia información.

RGU 1.3. La información mostrada será:

- Nombre.
- Teléfono de contacto.
- Teléfono de contacto alternativo.
- Dirección de correo electrónico.
- Dirección postal.

RGU 1.4. Los datos de contacto de la tarjeta proporcionarán atajos para usarlos:

RGU 1.4.1. Los números de teléfono desplegarán el dial del teléfono con el número marcado.

RGU 1.4.2. El correo electrónico abrirá una aplicación de correo electrónico con un nuevo correo preparado para ser enviado a la dirección.

RGU 1.4.3. La dirección postal desplegará Google Maps con la dirección ya buscada.

RGU 2. Los usuarios podrán actualizar su información.

RGU 2.1. El usuario actualizará los campos que prefiera de:

- Nombre.
- Teléfono de contacto.
- Teléfono de contacto alternativo.
- Dirección de correo electrónico.
- Dirección postal.

RGU 2.2. Cuando el usuario confirme la actualización, los datos serán enviados a la API REST para validación.

RGU 2.3. La API REST confirmará los cambios efectuados y responderá con la información completa del usuario.

RGU 2.4. La aplicación se actualizará con los nuevos datos del usuario.

RGU 2.5. Una notificación será enviada a los usuarios vinculados acerca de la actualización de los datos del usuario.

RGU 3. La aplicación permitirá a los usuarios establecer vínculos con otros usuarios.

RGU 3.1. Los Pacientes podrán vincular MAX_VINCULOS_PACIENTE usuarios de tipo Cuidador.

RGU 3.2. Los Cuidadores podrán vincular MAX_VINCULOS_CUIDADOR Pacientes.

RGU 3.3. Los Pacientes podrán generar un código QR para vincularse.

RGU 3.3.1. El código QR representará un token provisto por la API REST.

RGU 3.3.2. El token caducará en TIEMPO_CADUCIDAD_TOKEN_VINCULACIÓN.

RGU 3.4. Los Cuidadores podrán escanear el código QR de un Paciente para completar el vínculo.

RGU 3.4.1. El código leído será enviado a la API REST.

RGU 3.4.2. La API REST validará el token y que el vínculo sea válido.

RGU 3.4.2.1. Si el vínculo no es válido, una respuesta de error será emitida

RGU 3.4.2.2. Si el vínculo es válido, se creará el vínculo entre ambos usuarios.

RGU 3.5. La API REST emitirá una notificación de la creación del vínculo al resto de usuarios asociados.

RGU 4. Los usuarios con vínculos activos podrán eliminar sus vínculos actuales a través de la aplicación.

RGU 4.1. La petición se realizará con una llamada a la API REST.

RGU 4.2. La API REST validará la petición y enviará una respuesta de confirmación.

RGU 4.3. La API REST emitirá una notificación de la eliminación del vínculo al resto de usuarios asociados antes de la eliminación del vínculo.

RGU 5. La aplicación permitirá listar en la aplicación los Cuidadores relacionados de los usuarios.

RGU 5.1. La aplicación listará a los Pacientes sus Cuidadores vinculados.

RGU 5.2. La aplicación listará a los Cuidadores el resto de Cuidadores vinculados por su Paciente.

RGU 5.3. El listado de Cuidadores será proporcionado por una petición a la API REST.

RGU 5.4. Con cada Cuidador listado se mostrará su información de contacto:

- Teléfono de contacto.
- Teléfono de contacto alternativo.
- Dirección de correo electrónico.
- Dirección postal.

RGU 5.5. Desde el listado, la aplicación permitirá a los usuarios los mismos atajos de funcionalidad que RGU 1.4..

10.2.3. RGT. Requisitos de gestión de tareas

RGT 1. Los Pacientes y sus Cuidadores podrán crear una nueva tarea desde la aplicación.

RGT 1.1. Las tareas contarán con la siguiente información:

RGT 1.1.1. Un identificador único, obligatorio y generado en el proceso de persistencia de la tarea.

RGT 1.1.2. Título, obligatorio.

RGT 1.1.3. Creador, obligatorio y rellenado automáticamente por la aplicación.

RGT 1.1.4. Descripción, opcional.

RGT 1.1.5. Instante de creación, obligatorio y rellenado automáticamente por la aplicación.

RGT 1.1.6. Instante de la última actualización, obligatorio y por defecto será creada con el mismo valor que el instante de creación.

RGT 1.1.7. Estado de la tarea. Puede ser completa o incompleta. Obligatorio, por defecto será creada como incompleta.

RGT 1.2. Las tareas creadas en la aplicación se enviarán a la API REST para su validación y persistencia.

RGT 1.2.1. Si la tarea es inválida, la API REST emitirá un mensaje de error y no persistirá la tarea.

RGT 1.2.2. Si la tarea es válida, la API REST responderá con un mensaje de éxito con la información completa de la tarea creada.

RGT 1.3. La creación de una tarea la añadirá a la lista de tareas del Paciente involucrado.

RGT 1.4. El sistema emitirá una notificación de la tarea creada a los usuarios asociados.

RGT 2. Los Pacientes y sus Cuidadores podrán consultar las tareas existentes en la aplicación.

RGT 2.1. La lista de tareas será suministrada por la API REST. Se listarán todas las tareas del paciente que se consideren relevantes. Una tarea es relevante si cumple alguna de las siguientes condiciones:

RGT 2.1.1. Si la tarea está incompleta.

RGT 2.1.2. Si la última actualización de la tarea entra dentro del margen de días especificado por el usuario.

RGT 2.2. Las tareas listadas mostrarán el título, el creador, la descripción, el estado y las fechas de creación y actualización.

RGT 2.3. Las tareas completas e incompletas serán visualmente diferenciables.

RGT 2.4. Por defecto, las tareas aparecerán listadas de más recientes a más antiguas.

RGT 2.5. Las tareas se podrán gestionar desde el listado.

RGT 3. Los Pacientes y sus Cuidadores podrán marcar una tarea del Paciente como hecha desde la aplicación.

RGT 3.1. El usuario deberá confirmar que quiere marcar la tarea como hecha.

RGT 3.2. La petición de actualización se hará a la API REST, que la validará.

RGT 3.2.1. Si la petición no es válida, la API REST emitirá un mensaje de error.

RGT 3.2.2. Si la petición es válida, la API REST emitirá una respuesta de confirmación.

RGT 3.3. La tarea se actualizará visualmente para reflejar su nuevo estado.

RGT 3.4. La tarea se seguirá mostrando aunque pierda relevancia (véase RGT 2.1.) hasta que la lista sea refrescada. Permitiendo deshacer la acción.

RGT 3.5. El sistema notificará el cambio en el estado a los usuarios asociados.

RGT 4. Los Pacientes y sus Cuidadores podrán marcar una tarea como no hecha desde la aplicación.

RGT 4.1. El usuario deberá confirmar que quiere desmarcar la tarea como hecha.

RGT 4.2. La petición de actualización se hará a la API REST, que la validará.

RGT 4.2.1. Si la petición no es válida, la API REST emitirá un mensaje de error.

RGT 4.2.2. Si la petición es válida, la API REST emitirá una respuesta de confirmación.

RGT 4.3. La tarea se actualizará visualmente para reflejar su nuevo estado.

RGT 4.4. El sistema notificará el cambio en el estado a los usuarios asociados.

RGT 5. Los usuarios podrán eliminar tareas en la aplicación.

RGT 5.1. Los Pacientes podrán eliminar cualquiera de sus tareas.

RGT 5.2. Los Cuidadores sólo podrán eliminar las tareas creadas por ellos.

RGT 5.3. El usuario deberá confirmar que quiere eliminar la tarea.

RGT 5.4. La petición de eliminación se hará a la API REST, que la validará.

RGT 5.4.1. Si la petición no es válida, la API REST emitirá un mensaje de error.

RGT 5.4.2. Si la petición es válida, la API REST emitirá una respuesta de confirmación.

RGT 5.5. El sistema notificará a los usuarios vinculados de la eliminación de la tarea.

10.2.4. RSV. Requisitos de servicio de mensajería

RSV 1. La aplicación ofrecerá una sala de mensajería a los usuarios vinculados.

RSV 2. La aplicación se conectará a un WebSocket de la API de la sala.

RSV 3. Los participantes de la sala de mensajería serán el Paciente y todos sus Cuidadores vinculados.

RSV 4. Los mensajes de la sala pueden ser de tres tipos:

- RSV 4.1. Mensajes de texto.
- RSV 4.2. Tareas.
- RSV 4.3. Notificaciones.

RSV 5. La sala de mensajería mostrará, a los usuarios que se conecten, los mensajes más recientes.

- RSV 5.1. Los mensajes al abrir la sala serán servidos por la API REST.
- RSV 5.2. La API REST servirá los CANTIDAD_GRUPO_MENSAJES mensajes más recientes.
- RSV 5.3. Los participantes podrán cargar mensajes más antiguos.
- RSV 5.4. Al solicitar mensajes más antiguos la API REST servirá los CANTIDAD_GRUPO_MENSAJE mensajes más recientes siguientes.
- RSV 5.5. La aplicación mostrará los nuevos mensajes entrantes en tiempo real.
- RSV 5.6. Los mensajes entrantes serán servidos a través del WebSocket.
- RSV 5.7. Los mensajes salientes y entrantes serán visualmente diferentes.
- RSV 5.8. Los tipos de mensajes será visualmente diferentes.
- RSV 5.9. Los mensajes serán agrupados por fecha de emisión.

RSV 6. Los participantes de una sala de mensajería podrán enviar mensajes de texto al resto de pacientes.

- RSV 6.1. El mensaje saliente se enviará a través del WebSocket.
- RSV 6.2. La API persistirá el mensaje y lo reenviará por el WebSocket al resto de participantes.
- RSV 6.3. El mensaje saliente se añadirá a la lista de mensajes mostrados.

RSV 7. Los participantes de una sala de mensajería podrán gestionar tareas a través de la sala.

- RSV 7.1. Las tareas serán tratadas como mensajes.
- RSV 7.2. Los participantes podrán crear una tarea que se enviará por la sala de mensajería igual que en RSV 6..

RSV 7.3. Las tareas creadas de esta forma cumplirán las mismas pautas que en RGT 1.

RSV 7.4. Las tareas mostradas en la sala de mensajería podrán ser marcadas como hechas como en RGT 3.

RSV 7.5. Las tareas mostradas en la sala de mensajería podrán ser desmarcadas como hechas como en RGT 4.

RSV 7.6. Las tareas mostradas en la sala de mensajería podrán ser eliminadas como en RGT 5.

RSV 7.6.1. Las tareas eliminadas mientras el usuario esté conectado serán eliminadas de la lista.

RSV 7.6.2. La eliminación de una tarea por cualquier medio será comunicada a los usuarios conectados a la sala con una notificación.

RSV 7.7. Las acciones sobre las tareas de la sala serán enviadas a la API a través del WebSocket.

RSV 7.8. La API ejecutará la petición y enviará un mensaje al resto de participantes de la sala que actualice la tarea.

RSV 8. Los participantes serán desconectados del WebSocket de la sala al abandonarla.

RSV 9. La sala de mensajería mostrará notificaciones de la conexión y desconexión de usuarios de la sala al resto de usuarios conectados.

10.2.5. RSG. Requisitos del sistema de geolocalización

RSG 1. La aplicación servirá un mapa con un servicio de geolocalización mutua entre usuarios conectados.

RSG 2. La aplicación utilizará la API de Google Maps.

RSG 3. A través de la aplicación los usuarios podrán compartir su ubicación.

RSG 3.1. La aplicación solicitará permiso al usuario para utilizar el GPS integrado en el teléfono móvil.

RSG 3.1.1. Si el usuario deniega acceso al GPS, un mensaje de error será mostrado y la función no se llevará a cabo.

RSG 3.1.2. Si el usuario permite el acceso, se empezará a compartir la ubicación y se mostrará el mapa.

RSG 3.2. La aplicación se conectará a un WebSocket compartido con el resto de usuarios relacionados.

RSG 3.3. La ubicación comenzará a compartirse automáticamente al iniciar el servicio de geolocalización.

RSG 3.4. La ubicación del usuario será enviada a la API través del WebSocket en intervalos regulares de INTERVALO_COMPARTIR_UBICACIÓN.

RSG 3.5. La posición actual del usuario se mostrará en un mapa.

RSG 3.6. Cada actualización de la ubicación del usuario modificará su posición visual en el mapa.

RSG 3.7. Cuando un usuario empiece a compartir su ubicación, una notificación será enviada al resto de usuarios conectados.

RSG 4. Los usuarios compartiendo ubicación podrán dejar de hacerlo abandonando el mapa del servicio.

RSG 4.1. Si un usuario abandona el servicio, la aplicación se desconectará del WebSocket.

RSG 4.2. Cuando un usuario se desconecta del servicio, una notificación será enviada al resto de usuarios conectados.

RSG 4.3. La notificación de un usuario dejando de compartir su ubicación eliminará la notificación previa del comienzo de la función.

RSG 5. Los usuarios conectados al servicio podrán ver la ubicación de otros usuarios conectados.

RSG 5.1. Las ubicaciones ajenas se recibirán en tiempo real.

RSG 5.2. Las ubicaciones del resto de usuarios se recibirán a través del WebSocket.

RSG 5.3. La aplicación mostrará un aviso en pantalla cuando la lista de usuarios conectados al servicio cambie.

RSG 5.3.1. Cuando un usuario entre al servicio.

RSG 5.3.2. Cuando un usuario abandone el servicio.

RSG 5.4. Las ubicaciones del resto de usuarios se mostrará con un marcador diferenciado.

RSG 5.4.1. El marcador contendrá la información del usuario respectivo.

RSG 5.4.2. La actualización de las ubicaciones desplazará el marcador asociado al usuario a la nueva ubicación.

RSG 5.4.3. Cuando un usuario se desconecte, su marcador será retirado del mapa.

RSG 6. Los usuarios podrán desplazarse por el mapa.

RSG 7. Los usuarios podrán hacer acercar o alejar la vista en el mapa.

RSG 8. Los usuarios podrán centrar el mapa sobre su ubicación.

10.2.6. RSN: Requisitos del sistema de notificaciones

RSN 1. La aplicación notificará a los usuarios acerca de diversos sucesos de interés.

- RSN 1.1. Un nuevo vínculo genera un nuevo usuario asociado (véase RGU 3.5.)
- RSN 1.2. Un usuario asociado ha actualizado sus datos personales (véase RGU 2.5.)
- RSN 1.3. Un usuario asociado ha creado una nueva tarea (véase RGT 1.4.).
- RSN 1.4. Un usuario asociado ha eliminado una tarea (véase RGT 5.5.).
- RSN 1.5. Un usuario asociado ha marcado una tarea como hecha (véase RGT 3.5.).
- RSN 1.6. Un usuario asociado ha desmarcado una tarea como hecha (véase RGT 4.4.).
- RSN 1.7. Un usuario asociado ha comenzado a compartir su ubicación (véase RSG 3.7.).
- RSN 1.8. Un usuario asociado ha dejado de compartir su ubicación (véase RSG 4.2.).

RSN 2. La aplicación se unirá a un WebSocket para recibir las notificaciones.

RSN 3. Las notificaciones pueden ser pendientes o no, dependiendo de si el usuario las ha leído.

RSN 4. La aplicación mostrará la cantidad de notificaciones pendientes del usuario.

RSN 5. El usuario podrá ver en la aplicación sus notificaciones pendientes:

- RSN 5.1. La lista de notificaciones será servida por la API REST.
- RSN 5.2. Las notificaciones aparecerán agrupadas por fecha de emisión.
- RSN 5.3. Las notificaciones podrán ser marcadas como leídas individualmente o todas a la vez.
- RSN 5.4. Las notificaciones referentes a funciones de la aplicación podrán ser usadas para navegar a dicha función.

10.2.7. RNF: Requisitos no funcionales

RNF 1. La aplicación estará disponible en Android 8.1 y versiones posteriores del sistema operativo.

RNF 2. La aplicación estará disponible en los siguientes idiomas:

- Inglés
- Español
- Gallego

RNF 3. La API debe estar alojada en la nube.

- RNF 3.1. El sistema de la nube de alojamiento debe ser Microsoft Azure.
- RNF 3.2. La disponibilidad de la API debe ser superior al 99 %. Esto corresponde a una disponibilidad de 8672 horas y 24 minutos por año.
- RNF 3.3. La velocidad de respuesta a peticiones de la API no debe exceder los TIEMPO_MAXIMO_RESPUESTA_API.

RNF 4. La base de datos debe estar alojada en la nube.

- RNF 4.1. El sistema de base de datos en la nube debe ser MongoDB Atlas.
- RNF 4.2. La disponibilidad de la base de datos debe ser superior al 99 %. Esto corresponde a una disponibilidad de 8672 horas y 24 minutos por año.

RNF 5. El sistema conectará con las siguientes APIs de terceros:

- RNF 5.1. Google OAuth 2.0.
- RNF 5.2. Google Maps for Android.

RNF 6. Los usuarios únicamente necesitarán un conocimiento mínimo en el manejo de móviles inteligentes.

10.2.8. Valores de requisitos

ID	Valor
CANTIDAD_GRUPO_MENSAJES	30
INTERVALO_COMPARTIR_UBICACIÓN	5 segundos
MAX_VINCULOS_CUIDADOR	1
MAX_VINCULOS_PACIENTE	6
TIEMPO_MAXIMO_RESPUESTA_API	10 segundos
TIEMPO_VALIDEZ_TOKEN_SESIÓN	1 hora
TIEMPO_VALIDEZ_TOKEN_REFRESCO	24 horas
TIEMPO_VALIDEZ_TOKEN_VINCULACION	1 minuto

Tabla 10.1: Valores especificados en requisitos

11. Identificación de subsistemas

En base a los requisitos elaborados y al plan de diseño y arquitectura sopesados para el sistema se han identificado los siguientes subsistemas internos a los dos subsistemas mayores del sistema: la aplicación móvil y la API. Así como la comunicación entre dichos subsistemas y sus subsistemas internos.

11.1. Aplicación móvil

De cara a seguir las principales indicaciones en el desarrollo de sistemas Android, se ha decidido utilizar la arquitectura MVVM[10] y, por consiguiente, su misma especificación de subsistemas.

11.1.1. Actividades

Las actividades conforman los mayores componentes de una aplicación Android pues gestionan las pantallas mostradas al usuario y toda lógica de estados y funcionalidad ofrecida por dicha pantalla. Con la aplicación de la arquitectura MVVM estos componentes se componen a su vez de dos subsistemas interrelacionados.

11.1.1.1. Modelos de vista

Los modelos de vista suponen la novedad introducida por la arquitectura a utilizar y son módulos encargados de extraer la lógica de datos de las actividades para separarlos de la vista. El modelo de vista contiene la información que se mostrará en pantalla o que sea necesaria para la ejecución de la aplicación, y de la misma forma está a cargo de la obtención de esta, gestionando la comunicación con subsistemas inferiores.

11.1.1.2. Vistas

Las vistas son el sistema en control de la lógica de dibujado, animación y manejo de los elementos de la interfaz de usuario. Recuperan la información contenida en sus modelos de vista y la muestran por pantalla. Están compuestas por archivos de diseño de la interfaz y por archivos de código para las gestiones más complejas de la misma.

11.1.2. Modelo

La representación de las entidades de datos a manejar en la aplicación. Son las entidades que se usarán en la comunicación entre subsistemas y carecerán de más responsabilidades que la de manejarse a sí mismas.

11.1.3. Repositorios

Capa de comunicación entre las actividades y los servicios o bases de datos. Tendrán la responsabilidad de comunicarse con los módulos necesarios para obtener los datos solicitados por las actividades y de la transformación de los mismos en los modelos que serán usados en la aplicación. También gestionarán las peticiones en la dirección contraria, solicitando a los servicios acciones sobre entidades proporcionadas por las actividades.

11.1.4. Servicios

Los servicios son la capa más profunda de la aplicación y son los encargados de la comunicación directa con las bases de datos o interfaces web. En el primer planteamiento de la aplicación no se ha estimado la presencia de una base de datos local por lo que los únicos servicios serán todos web. Sin embargo, dentro de los servicios web podremos distinguir dos tipos: unos enfocados en la comunicación HTTP y otros en la comunicación por medio del protocolo WebSocket.

11.2. API

Para la API se ha distinguido un total de tres subsistemas mayores relacionados mutuamente en un esquema triangular que invalida el planteamiento de arquitecturas por capas.

11.2.1. Bases de datos

El subsistema a cargo de gestionar la persistencia y recuperación de datos estará compuesto por dos partes interrelacionadas:

11.2.1.1. Modelo

Los modelos serán la representación de las entidades almacenadas en la base de datos y que serán recuperadas por los dos subsistemas superiores. Carecen de toda

lógica que no implique su autogestión y serán siempre modificados, creados, leídos o eliminados por los servicios.

11.2.1.2. Servicios

Sistema de gestión del Modelo de la base de datos. Módulos encargados de manejar las consultas y operaciones de la base de datos con toda la lógica necesaria para aseverar la consistencia y restricciones necesarias en los datos. Son el único sistema con capacidad de manejo sobre el Modelo.

11.2.2. Manejador HTTP REST

Módulo a cargo de recibir y gestionar las peticiones HTTP para la API REST. Contiene la lógica de las operaciones a realizar para cumplir la petición, de todas las comprobaciones necesarias y del manejo de error de las mismas en caso de que surjan errores o sean incorrectas. Realizan las operaciones concernientes a los datos por medio de los servicios, quedando en la lógica de este manejador únicamente la gestión del flujo de operaciones.

11.2.3. Manejador WebSocket

Similar al manejador HTTP REST pero con la diferencia de que este subsistema escucha los mensajes recibidos por medio de la interfaz WebSocket. Contiene la lógica para gestionar los eventos notificados a través del mismo y para responder a estos de vuelta o por retransmisión a través del socket. Como el otro manejador, todas sus operaciones son únicamente llamadas sucesivas a los servicios necesarios.

11.3. Comunicación entre subsistemas

Para satisfacer el funcionamiento total de los subsistemas de la aplicación se ha definido la siguiente comunicación que garantice la integridad de las arquitecturas.

11.3.1. Subsistemas internos de la aplicación móvil

La comunicación de los subsistemas de la aplicación seguirá también la arquitectura MVVM y como esta define la comunicación entre sus subsistemas. El diagrama 11.1 que lo representa está basado en el mismo diagrama del artículo de referencia de Google[10].

Dentro de la aplicación la comunicación entre subsistemas será vertical y unidirec-

cional. Las vistas solo serán accedidas por la propia interacción del usuario y a su vez serán las únicas que se comuniquen con sus modelos de vista. Esto se hará por medio de llamadas para la ejecución de operaciones o mediante el patrón observador sobre los datos (usando objetos LiveData[15] de Android) gestionados y almacenados por el modelo de vista.

El modelo de vista por su parte estará a cargo de proveer cualquier necesidad de la vista y para toda la lógica que implique una comunicación externa o la persistencia o recuperación de datos se servirá de los repositorios. Los repositorios llamarán a los servicios necesarios para que tramiten la petición y recuperar la información necesaria o llevar a cabo la operación encomendada desde el modelo de vista contactando con la API.

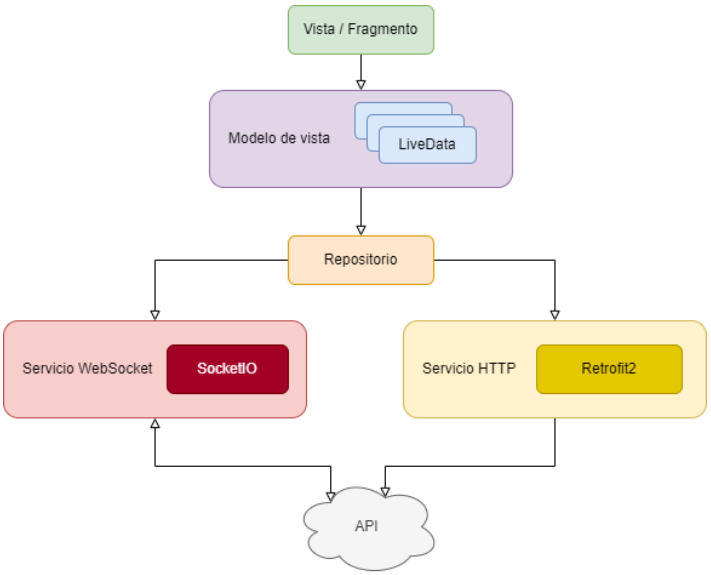


Figura 11.1: Arquitectura y comunicación de la aplicación móvil

11.3.2. Subsistemas internos de la API

La API puede recibir comunicaciones por dos vías diferentes. Por peticiones HTTP a través de su API REST y por medio del socket a través de su interfaz WebSocket. A su vez, peticiones HTTP pueden requerir el envío de eventos a través del socket, por lo que es necesario que exista comunicación entre estos dos subsistemas. Dicha comunicación será llevada a cabo por medio de la referencia recíproca al manejador raíz de ambos subsistemas, de forma que se puedan realizar llamadas mutuas. A su vez, ambos subsistemas requieren la comunicación con la base de datos que realizarán a partir de los servicios.

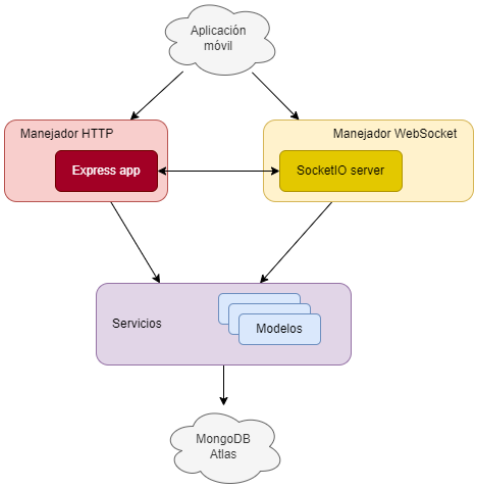


Figura 11.2: Arquitectura y comunicación de la API

11.3.3. Subsistemas API y aplicación móvil

Finalmente, de cara a que el sistema funcione se ha definido también la comunicación entre los dos mayores componentes. La comunicación entre la aplicación y la API tendrá dos vías distintas. Por un lado estarán las peticiones HTTP realizadas a través de la librería **Retrofit 2** (Apartado 24.2.1.11) que la API gestionará y responderá a través de los endpoint servidos por medio de su API REST construída sobre el framework Express (Apartado 24.2.2.5). Y por otro lado, al iniciar la aplicación se establecerá un canal TCP entre la aplicación y la API utilizando Socket.IO (Apartado 24.2.2.15) en ambos extremos de esta comunicación.

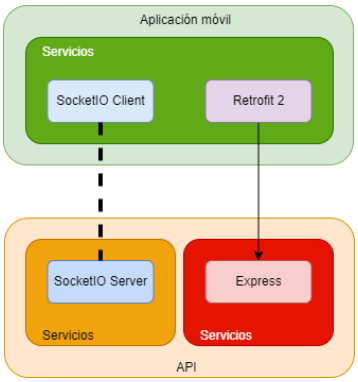


Figura 11.3: Comunicación entre la aplicación y la API

12. Especificación de casos de uso

12.1. CU1. Registro

Caso de Uso 1. Registro	
Requisitos	RSU 2.3.
Precondiciones	La cuenta de Google del usuario no debe estar registrada
Actores	Usuarios no registrados
Descripción	Un usuario nuevo se creará un perfil del sistema e iniciará sesión
Secuencia normal	El usuario abrirá la aplicación por primera vez Seleccionará el botón de *Iniciar Sesión* Seleccionará su cuenta de Google Introducirá un nombre para identificarse Elegirá el rol que desee Incluirá su información de contacto adicional Confirmará los datos introducidos El usuario será dirigido a la pantalla principal con un nuevo perfil
Postcondiciones	-
Excepciones	Cuenta de Google ya usada: Se iniciará sesión normalmente

Tabla 12.1: Especificación del CU1. Registro

12.2. CU2. Iniciar sesión

Caso de Uso 2. Iniciar sesión	
Requisitos	RSU 2.2.
Precondiciones	El usuario debe disponer de un perfil en el sistema
Actores	Pacientes y Cuidadores
Descripción	El usuario iniciará sesión con su perfil en el sistema
Secuencia normal	El usuario abrirá la aplicación Seleccionará el botón de *Iniciar Sesión* Seleccionará su cuenta de Google El usuario será dirigido a la pantalla principal con su perfil
Postcondiciones	-
Excepciones	-

Tabla 12.2: Especificación del CU2. Iniciar sesión

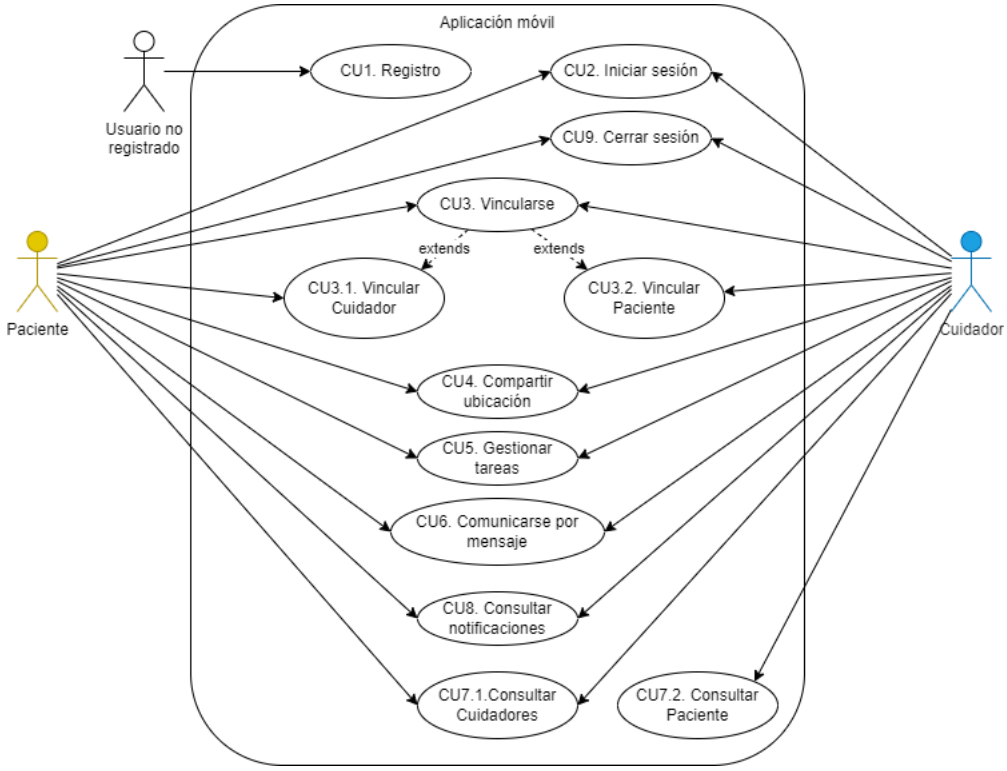


Figura 12.1: Diagrama de casos de uso generales

12.3. CU3. Vincularse con otro usuario

Caso de Uso 3.1. Vincular un Cuidador	
Requisitos	RGU 3.3.
Precondiciones	El usuario debe haber iniciado sesión El Cuidador a vincular debe estar cerca y preparado
Actores	Pacientes
Descripción	El usuario generará un código para vincularse con un Cuidador
Secuencia normal	El usuario seleccionará la función *Forjar vínculo* Un código QR se mostrará en su pantalla El Cuidador lo escaneará Se creará el vínculo entre los usuarios y se notificará a los usuarios asociados
Postcondiciones	-
Excepciones	El Paciente no puede vincular más Cuidadores: No se forjará el vínculo

Tabla 12.3: Especificación del CU3.1. Vincular un Cuidador

Caso de Uso 3.2. Vincular un Paciente	
Requisitos	RGU 3.4.
Precondiciones	El usuario debe haber iniciado sesión El usuario no debe estar ya vinculado El Paciente a vincular debe estar cerca y con su código preparado
Actores	Cuidadores
Descripción	El usuario leerá un código para vincularse con un Paciente
Secuencia normal	El usuario seleccionará la función *Forjar vínculo* La aplicación solicitará permisos para utilizar la cámara El usuario aceptará la petición El usuario escaneará el código del Paciente Se creará el vínculo entre los usuarios y se notificará a los asociados
Postcondiciones	-
Excepciones	No se dan los permisos: La función no se iniciará El Paciente no puede vincular más Cuidadores: No se forjará el vínculo

Tabla 12.4: Especificación del CU3.2. Vincular un Paciente

Caso de Uso 3.3. Desvincularse	
Requisitos	RGU 4.
Precondiciones	El usuario debe haber iniciado sesión El usuario debe estar vinculado
Actores	Pacientes y Cuidadores
Descripción	El usuario eliminará uno de sus vínculos
Secuencia normal	El usuario seleccionará la función *Eliminar vínculo* El usuario seleccionará el vínculo a eliminar La aplicación le pedirá confirmación Se eliminará el vínculo entre los usuarios y se notificará a los usuarios asociados
Postcondiciones	-
Excepciones	-

Tabla 12.5: Especificación del CU3.3. Desvincularse

12.4. CU4. Compartir ubicación

Caso de Uso 4.1. Compartir ubicación del usuario	
Requisitos	RSG 3.
Precondiciones	El usuario debe haber iniciado sesión El usuario debe estar vinculado
Actores	Pacientes y Cuidadores
Descripción	El usuario enviará su ubicación actual a los usuarios asociados
Secuencia normal	El usuario seleccionará la función *Compartir ubicación* La aplicación solicitará permisos para acceder a la ubicación El usuario aceptará la petición Se desplegará un mapa mostrando la ubicación del usuario Empezará a compartirse la ubicación del usuario y se notificará a los asociados
Postcondiciones	-
Excepciones	No se dan los permisos: La función no se iniciará

Tabla 12.6: Especificación del CU4.1. Compartir ubicación del usuario

Caso de Uso 4.2. Ver ubicaciones de usuarios asociados	
Requisitos	RSG 5.
Precondiciones	El usuario debe haber iniciado sesión El usuario debe estar vinculado Los usuarios asociados deben estar compartiendo su ubicación
Actores	Pacientes y Cuidadores
Descripción	La aplicación mostrará al usuario la ubicación de sus usuarios asociados
Secuencia normal	El usuario seleccionará la función *Compartir ubicación* El usuario empezará a compartir su ubicación (véase 12.6) Se desplegará un mapa mostrando la ubicación de los usuarios conectados
Postcondiciones	Si el usuario deja de compartir su ubicación no podrá ver las del resto
Excepciones	-

Tabla 12.7: Especificación del CU4.2. Ver ubicaciones de usuarios asociados

12.5. CU5. Gestionar tareas

Caso de Uso 5.1. Listar tareas	
Requisitos	RGT 2.
Precondiciones	El usuario debe haber iniciado sesión Si es Cuidador debe estar vinculado
Actores	Pacientes y Cuidadores
Descripción	El usuario consultará la lista de tareas relevantes que tiene asociadas
Secuencia normal	El usuario seleccionará la pantalla *Tareas* o *Feed* La aplicación recuperarás las tareas relevantes asociadas al usuario Las tareas recuperadas se mostrarán en la pantalla
Postcondiciones	-
Excepciones	-

Tabla 12.8: Especificación del CU5.1. Listar tareas

Caso de Uso 5.2. Crear tarea	
Requisitos	RGT 1.
Precondiciones	El usuario debe haber iniciado sesión Si es Cuidador debe estar vinculado
Actores	Pacientes y Cuidadores
Descripción	El usuario creará una tarea que se compartirá con los demás usuarios asociados
Secuencia normal	El usuario seleccionará la pantalla *Tareas* o *Feed* El usuario seleccionará la función *Crear tarea* El usuario rellenará los campos obligatorios El usuario rellenará los campos opcionales que desee El usuario confirmará la creación de la tarea El sistema creará la tarea y la notificará a los demás usuarios asociados
Postcondiciones	-
Excepciones	-

Tabla 12.9: Especificación del CU5.2. Crear tarea

Caso de Uso 5.3. Marcar tarea como hecha	
Requisitos	RGT 3.
Precondiciones	El usuario debe haber iniciado sesión Si es Cuidador debe estar vinculado Debe existir alguna tarea no hecha asociada al usuario
Actores	Pacientes y Cuidadores
Descripción	El usuario marcará una tarea asociada como hecha
Secuencia normal	El usuario listará sus tareas, véase CU5.1 (12.8) El usuario seleccionará la tarea no hecha que desee El usuario marcará la opción de marcarla hecha La aplicación pedirá confirmación al usuario El usuario confirmará la acción Se modificará la tarea y se notificará a los usuarios asociados
Postcondiciones	-
Excepciones	-

Tabla 12.10: Especificación del CU5.3. Marcar tarea como hecha

Caso de Uso 5.4. Desmarcar tarea hecha	
Requisitos	RGT 4.
Precondiciones	El usuario debe haber iniciado sesión Si es Cuidador debe estar vinculado Debe existir alguna tarea hecha asociada al usuario
Actores	Pacientes y Cuidadores
Descripción	El usuario desmarcará una tarea marcada como hecha
Secuencia normal	El usuario listará sus tareas, véase CU5.1 (12.8) El usuario seleccionará la tarea hecha que desee El usuario marcará la opción de marcarla como no hecha La aplicación pedirá confirmación al usuario El usuario confirmará la acción Se modificará la tarea y se notificará a los usuarios asociados
Postcondiciones	-
Excepciones	-

Tabla 12.11: Especificación del CU5.4. Desmarcar tarea hecha

Caso de Uso 5.5. Eliminar tarea	
Requisitos	RGT 5.
Precondiciones	El usuario debe haber iniciado sesión Si es Cuidador debe estar vinculado Debe existir alguna tarea asociada al usuario
Actores	Pacientes y Cuidadores
Descripción	El usuario eliminará una tarea asociada
Secuencia normal	El usuario listará sus tareas, véase CU5.1 (12.8) El usuario seleccionará la tarea que desee eliminar El usuario elegirá la opción de *Eliminar* la tarea La aplicación pedirá confirmación al usuario El usuario confirmará la acción Se eliminará la tarea y se notificará a los asociados
Postcondiciones	-
Excepciones	El usuario no es el Paciente o el creador de la tarea: Se le comunicará que es una acción que no puede realizar y no se llevará a cabo

Tabla 12.12: Especificación del CU5.5. Eliminar tarea

12.6. CU6. Comunicarse por mensaje instantáneo

Caso de Uso 6.1. Ver mensajes	
Requisitos	RSV 5.
Precondiciones	El usuario debe haber iniciado sesión El usuario debe estar vinculado
Actores	Pacientes y Cuidadores
Descripción	El usuario verá sus mensajes y tareas asociadas
Secuencia normal	El usuario abrirá la pantalla *Feed* La aplicación mostrará los mensajes y tareas asociados al usuario más recientes La aplicación mostrará los nuevos mensajes que vayan enviado los usuarios asociados
Postcondiciones	-
Excepciones	-

Tabla 12.13: Especificación del CU6.1. Ver mensajes

Caso de Uso 6.2. Enviar mensajes	
Requisitos	RSV 6.
Precondiciones	El usuario debe haber iniciado sesión El usuario debe estar vinculado
Actores	Pacientes y Cuidadores
Descripción	El usuario verá sus mensajes asociados
Secuencia normal	El usuario abrirá la pantalla *Feed* El usuario redactará el mensaje que quiera enviar El usuario pulsará el botón de *Enviar mensaje* El mensaje se listará en la pantalla del usuario El mensaje se enviará al resto de usuarios asociados
Postcondiciones	-
Excepciones	-

Tabla 12.14: Especificación del CU6.2. Enviar mensajes

12.7. CU7. Consultar información de asociados

Caso de Uso 7.1. Consultar Cuidadores asociados	
Requisitos	RGU 5.
Precondiciones	El usuario debe haber iniciado sesión El usuario debe estar vinculado
Actores	Pacientes y Cuidadores
Descripción	El usuario listará sus Cuidadores asociados y la información de estos
Secuencia normal	El usuario abrirá la pantalla *Vínculos* La aplicación obtendrá el listado de los Cuidadores asociados al usuario Los cuidadores asociados se mostrarán en pantalla El usuario desplegará la información del Cuidador que desee para ver su información de contacto completa
Postcondiciones	-
Excepciones	-

Tabla 12.15: Especificación del CU7.1. Consultar Cuidadores asociados

Caso de Uso 7.2. Consultar Paciente asociado	
Requisitos	RGU 1.
Precondiciones	El usuario debe haber iniciado sesión El usuario debe estar vinculado
Actores	Cuidadores
Descripción	El usuario consultará la información de su paciente asociado
Secuencia normal	El usuario abrirá la aplicación La aplicación recuperará la información del Paciente Se mostrará en la pantalla principal la información del Paciente El usuario desplegará la información del Paciente para ver su información de contacto completa
Postcondiciones	-
Excepciones	-

Tabla 12.16: Especificación del CU7.2. Consultar Paciente asociado

12.8. CU8. Consultar notificaciones

Caso de Uso 8.1 Consultar notificaciones no leídas	
Requisitos	RSN 5.
Precondiciones	El usuario debe haber iniciado sesión El usuario debe estar vinculado
Actores	Pacientes y Cuidadores
Descripción	El usuario consultará las notificaciones no leídas
Secuencia normal	El usuario abrirá la pantalla *Notificaciones* La aplicación recuperará las notificaciones no leídas del usuario Las notificaciones recuperadas se mostrarán en la pantalla agrupadas
Postcondiciones	-
Excepciones	-

Tabla 12.17: Especificación del CU8.1 Consultar notificaciones no leídas

Caso de Uso 8.2 Marcar notificaciones como leídas	
Requisitos	RSN 5.3.
Precondiciones	El usuario debe haber iniciado sesión El usuario debe estar vinculado El usuario debe tener notificaciones pendientes no leídas
Actores	Pacientes y Cuidadores
Descripción	El usuario marcará uno o todas sus notificaciones pendientes como leídas
Secuencia normal	El usuario listará sus aplicaciones, véase CU8.1 (12.17) El usuario usará el botón *Marcar como leída* de una notificación o *Marcar todas como leídas* La notificación se marcará como leídas
Postcondiciones	-
Excepciones	-

Tabla 12.18: Especificación del CU8.2 Marcar notificaciones como leídas

12.9. CU9. Cerrar sesión

Caso de Uso 9. Cerrar sesión	
Requisitos	RSU 3.
Precondiciones	El usuario debe haber iniciado sesión
Actores	Pacientes y Cuidadores
Descripción	El usuario seleccionará la función *Cerrar sesión*
Secuencia normal	Se cerrará la sesión y se devolverá al usuario a la pantalla de *Iniciar sesión*
Postcondiciones	-
Excepciones	-

Tabla 12.19: Especificación del CU9. Cerrar sesión

13. Análisis de Interfaz de Usuario

La única interfaz de usuario del sistema serán las actividades y diálogos presentes en la aplicación móvil, así pues todas las pantallas identificadas han sido concebidas para un funcionamiento y navegación móvil.

13.1. Pantalla de inicio de sesión

La pantalla de inicio de sesión será la pantalla de entrada a la aplicación para todos los usuarios sin sesión iniciada. Su diseño es el ilustrado en Figura 13.1. Puesto que de primeras sólo se permitirá el inicio de sesión con Google, únicamente se dispondrá un botón para el inicio de sesión que abrirá el selector de cuentas de Google. El inicio sesión puede dirigir al usuario a Pantalla de creación de perfil si es un usuario nuevo o a Pantalla principal para usuarios con perfil ya existente.

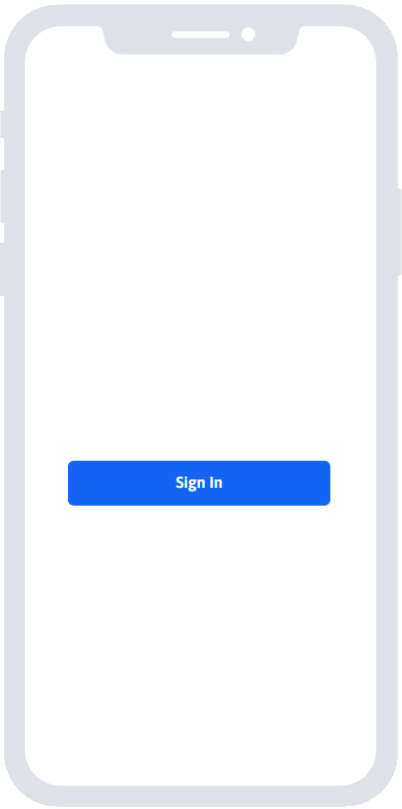


Figura 13.1: Diseño inicial de LaunchActivity

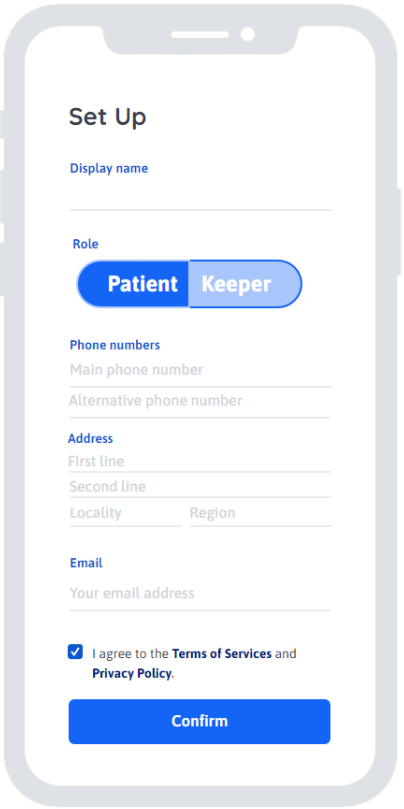


Figura 13.2: Diseño inicial de SetUpActivity

13.2. Pantalla de creación de perfil

En esta pantalla se completará el perfil de un usuario recién registrado, en ella se deben disponer campos de texto para que el usuario pueda introducir su nombre visible (RSU 2.3.1.) y su información adicional (RSU 2.3.3.), además de un selector que le permita elegir entre uno de los dos roles posibles (RSU 2.3.2.). Por último, existirá un botón para confirmar los datos introducidos y completar la creación del perfil, lo cuál redirigirá al usuario hacia Pantalla principal. El diseño previo de esta pantalla es el mostrado en Figura 13.2.

13.3. Pantalla principal

La pantalla principal es el punto de navegación básico de toda la aplicación para los usuarios autenticados. A través de esta pantalla se puede acceder a Pantalla de vínculos, Pantalla del feed, Pantalla de geolocalización, Pantalla de gestión de tareas y Pantalla de notificaciones. Su diseño es Figura 13.3.

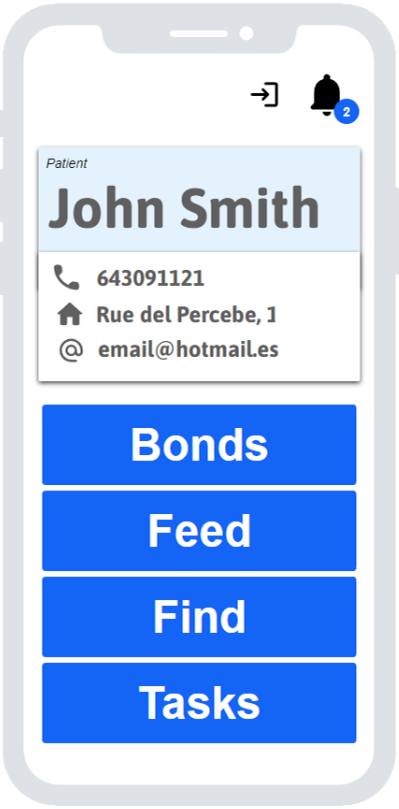


Figura 13.3: Diseño inicial de MainActivity

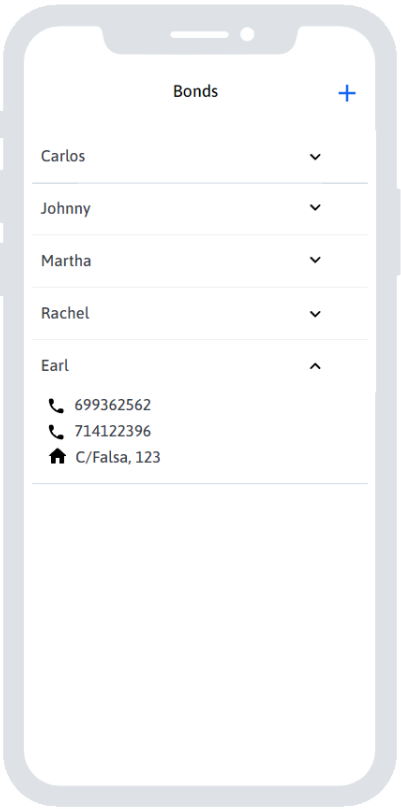


Figura 13.4: Diseño inicial de BondsActivity

13.4. Pantalla de vínculos

En la pantalla de vínculos se listan los usuarios asociados. Cada uno de estos usuarios listados podrá ser desplegado para consultar su información de contacto. Además de esta información también habrá un botón para crear un nuevo vínculo, que en el caso de los Pacientes mostrará un código QR de vinculación y en el de los Cuidadores abrirá la cámara para escanearlo. La Figura 13.4 muestra su primer diseño.

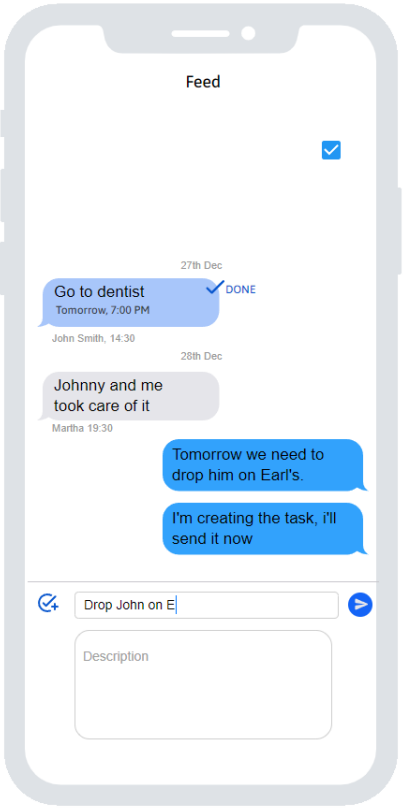


Figura 13.5: Diseño inicial de FeedActivity

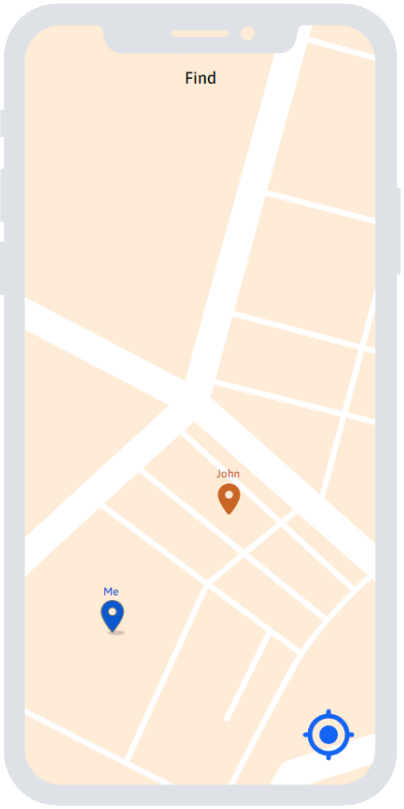


Figura 13.6: Diseño inicial de LocationActivity

13.5. Pantalla del feed

La pantalla que contiene el chat entre usuarios asociados. Contendrá las funciones habituales de un chat como la lista de mensajes, diferenciando los enviados y los recibidos; fechas y horas de envío de mensajes; o un campo de texto y un botón para enviar los mensajes. Aparte de eso también dispondrá de un botón para cambiar de modo al de creación de tareas, desplegando un área de texto para la descripción. Dichas tareas también tendrán un aspecto diferenciado de los mensajes y mostrarán visualmente su

estado con una caja de verificación que también podrá ser pulsada para modificarlo. Este chat puede verse en la Figura 13.5.

13.6. Pantalla de geolocalización

El servicio de geolocalización de los usuarios se servirá por medio de una actividad compuesta de un mapa decorado con los marcadores de la ubicación de los distintos usuarios que se encuentren compartiendo su ubicación en esos momentos y que se irá actualizando a medida que dichos usuarios se desplacen. Aparte de esto también habrá un botón para centrar el mapa sobre la posición del usuario y permitir que siempre pueda recuperar la referencia. Una muestra de todo eso se ofrece en Figura 13.6.

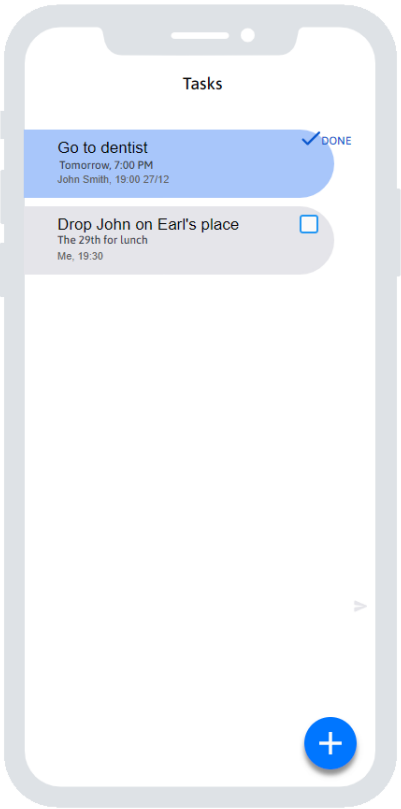


Figura 13.7: Diseño inicial de TasksActivity

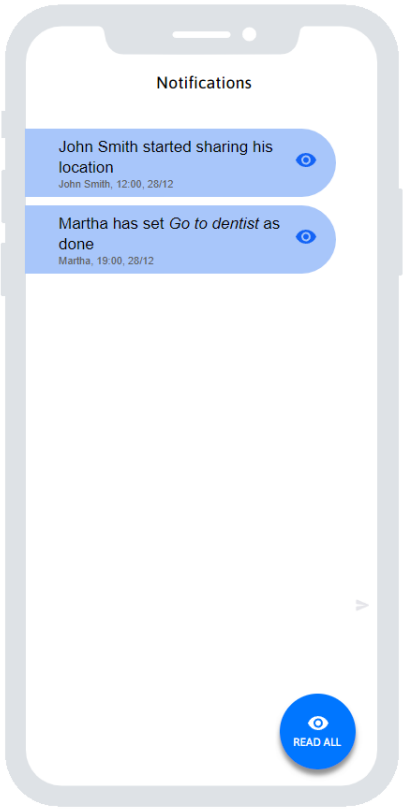


Figura 13.8: Diseño inicial de NotificationDialog

13.7. Pantalla de gestión de tareas

La última actividad ofrecerá un listado de las tareas relevantes del usuario con toda la información importante (RGT 2.2.) de las mismas, así como una representación visual del estado de estas, estado que podrá modificarse con una caja de verificación. Aparte de esto, contará también con un botón para crear notificaciones que desplegará un creador de tareas similar al presente en el diseño de Diseño inicial de FeedActivity. La vista preliminar de esta actividad es la Figura 13.7.

13.8. Pantalla de notificaciones

Este pantalla no será una actividad sino un diálogo sobre la Diseño inicial de MainActivity que será desplegado al utilizar el botón con el icono de la campana. Esta actividad listará todas las notificaciones no leídas por el usuario y cada una de ellas contará con un botón para marcarla como leída. Algo que también podrá hacerse con el botón de marcar todas como leídas, Figura 13.8.

13.9. Mapa de navegación

El mapa de navegación de las pantallas anteriores es el ilustrado en Figura 13.9. Aquellas pantallas con fondo azul son pantalla privadas que requieren autenticación para acceder.

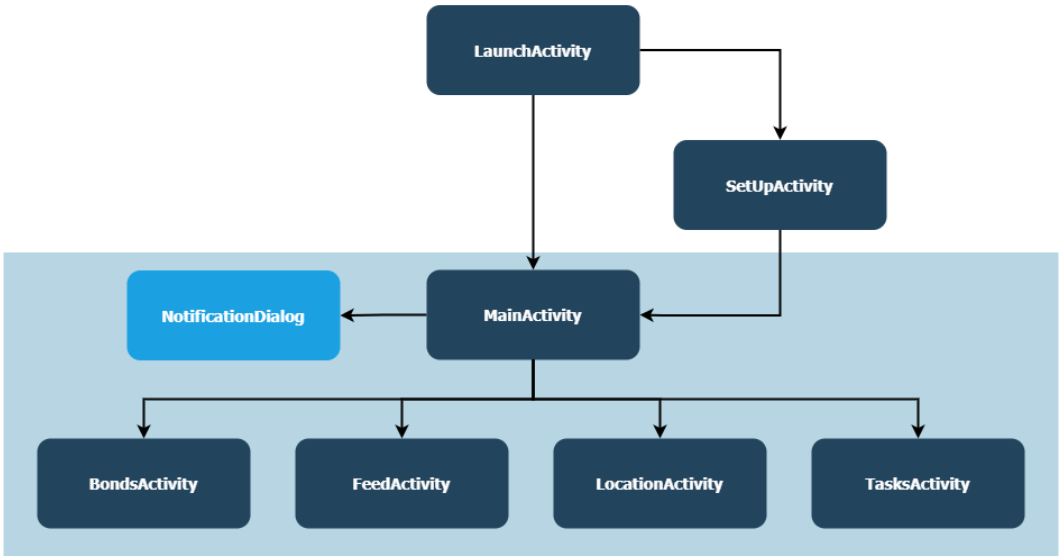


Figura 13.9: Propuesta de mapa de navegación

14. Diagrama de clases

En base a los análisis de las secciones anteriores se ha definido una versión preliminar del diseño y arquitectura orientativos del sistema.

14.1. Clases de la aplicación

El conjunto de clases general de la aplicación a desarrollar se encuentra resumido en el diagrama de la Figura 14.1. Las clases de color rojo representan librerías externas clave de la aplicación.

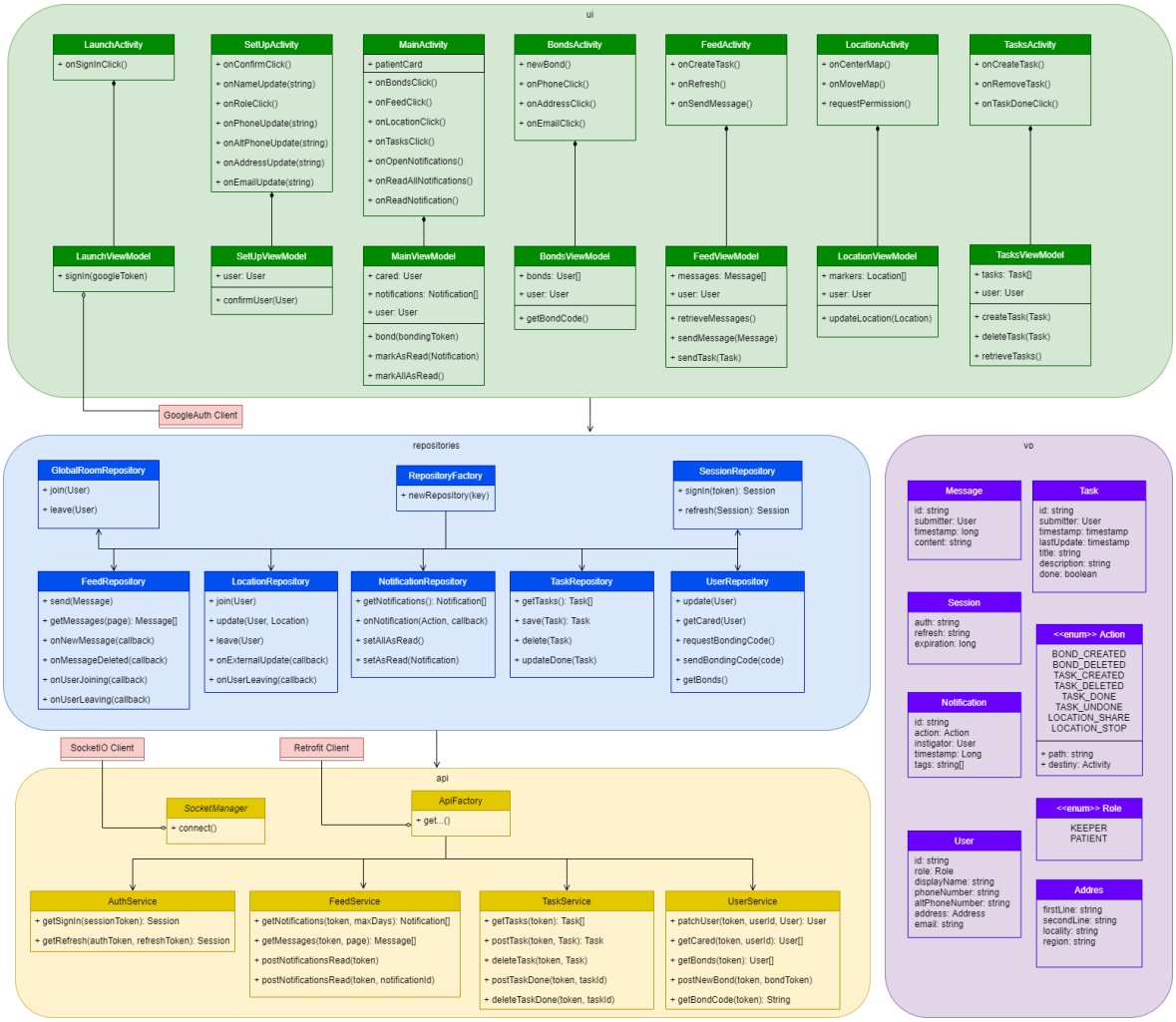


Figura 14.1: Diagrama de clases de la Aplicación

14.1.1. UI

14.1.1.1. Launch

LaunchActivity	
Descripción	Actividad de inicio de la aplicación con el inicio de sesión
Funciones	
onSignClick	Arranca el proceso de inicio de sesión

Tabla 14.1: Especificación de la clase LaunchActivity

LaunchViewModel	
Descripción	Modelo de la vista de LaunchActivity
Funciones	
signIn	Solicita el inicio de sesión a la API

Tabla 14.2: Especificación de la clase LaunchViewModel

14.1.1.2. Set up

SetUpActivity	
Descripción	Actividad de configuración de la cuenta de usuario en la creación de esta
Funciones	
onConfirmClick	Confirma la creación del usuario
onNameUpdate	Actualiza el nombre de usuario
onRoleClick	Selecciona un rol
onPhoneUpdate	Actualiza el número de teléfono principal
onAltPhoneUpdate	Actualiza el número de teléfono alternativo
onAddressUpdate	Actualiza la dirección postal
onEmailUpdate	Actualiza la dirección electrónica

Tabla 14.3: Especificación de la clase SetUpActivity

SetUpViewModel	
Descripción	Modelo de la vista de SetUpActivity
Propiedades	
user	Información del usuario
Funciones	
confirmUser	Envía los datos del usuario para confirmarlo

Tabla 14.4: Especificación de la clase SetUpViewModel

14.1.1.3. Main

MainActivity	
Descripción	Actividad principal de la aplicación desde la que acceder a todas las funciones con la sesión iniciada
Propiedades	
patientCard	Tarjeta con la información del Paciente vinculado (14.7)
Funciones	
onBondsClick	Dirige al usuario a la actividad de vínculos (14.7)
onFeedClick	Dirige al usuario a la actividad del feed (14.9)
onLocationClick	Dirige al usuario a la actividad de la geolocalización (14.9)
onTasksClick	Dirige al usuario a la actividad de gestión de tareas (14.13)
onOpenNotifications	Despliega las notificaciones de usuario
onReadAllNotifications	Marca todas las notificaciones como leídas
onReadNotification	Marca una notificación como leída

Tabla 14.5: Especificación de la clase MainActivity

MainViewModel	
Descripción	Modelo de la vista MainActivity
Propiedades	
cared	Información del paciente vinculado
notifications	Lista de notificaciones no leídas del usuario
user	Información del usuario
Funciones	
bond	Envía una petición para crear un vínculo con otro usuario
markAsRead	Envía una petición para marcar una notificación como leída
markAsReadAll	Envía una petición para marcar todas las notificación como leída

Tabla 14.6: Especificación de la clase MainViewModel

14.1.1.4. Bonds

BondsActivity	
Descripción	Actividad para la gestión de los vínculos
Funciones	
newBond	Dirige al usuario a la actividad de vínculos
onPhoneClick	Abre la aplicación de teléfono con el número pulsado
onAddresClick	Abre la aplicación de mapas con la dirección pulsada
onEmailClick	Abre la aplicación de correo electrónico con el email pulsado

Tabla 14.7: Especificación de la clase BondsActivity

BondsViewModel	
Descripción	Modelo de la vista BondsActivity
Propiedades	
bonds	Lista con la información de los usuarios asociados
user	Información del usuario
Funciones	
getBondCode	Envía una petición para obtener un token de vinculación

Tabla 14.8: Especificación de la clase BondsViewModel

14.1.1.5. Feed

FeedActivity	
Descripción	Actividad de la aplicación con el chat entre usuarios asociados
Funciones	
onCreateTask	Envía el mensaje como tarea
onRefresh	Actualiza la lista de mensajes
onSendMessage	Envía el mensaje

Tabla 14.9: Especificación de la clase FeedActivity

FeedViewModel	
Descripción	Modelo de la vista FeedActivity
Propiedades	
messages	Lista con los mensajes del Feed
user	Información del usuario
Funciones	
retrieveMessage	Envía una petición para recuperar los mensajes
sendMessage	Envía el mensaje al resto de usuarios
sendTask	Envía la tarea al resto de usuarios

Tabla 14.10: Especificación de la clase FeedViewModel

14.1.1.6. Location

LocationActivity	
Descripción	Actividad con el mapa de la geolocalización
Funciones	
onCenterMap	Centra el mapa en el usuario
onMoveMap	Mueve el mapa
requestPermission	Solicita el permiso del usuario para usar la geolocalización

Tabla 14.11: Especificación de la clase LocationActivity

LocationViewModel	
Descripción	Modelo de la vista LocationActivity
Propiedades	
markers	Lista de localizaciones del resto de usuarios
user	Información del usuario
Funciones	
updateLocation	Envía la localización al resto de usuarios

Tabla 14.12: Especificación de la clase LocationViewModel

14.1.1.7. Tasks

TasksActivity	
Descripción	Actividad para la gestión de tareas
Funciones	
onCreateTask	Confirma la creación de una tarea
onRemoveTask	Elimina una tarea
onTaskDoneClick	Cambia el estado de una tarea

Tabla 14.13: Especificación de la clase TasksActivity

TasksViewModel	
Descripción	Modelo de la vista TasksActivity
Propiedades	
tasks	Lista de tareas relevantes del usuario
user	Información del usuario
Funciones	
createTask	Envía una nueva tarea para crear
deleteTask	Envía una petición para eliminar una tarea
retrieveTasks	Recupera la lista de tareas relevantes del usuario

Tabla 14.14: Especificación de la clase TasksViewModel

14.1.2. Repositories

14.1.2.1. RepositoryFactory

RepositoryFactory	
Descripción	Factoría de repositorios
Funciones	
newRepository	Genera y devuelve el repositorio solicitado

Tabla 14.15: Especificación de la clase RepositoryFactory

14.1.2.2. FeedRepository

FeedRepository	
Descripción	Repositorio para la gestión de las operaciones relacionadas con el feed
Funciones	
send	Envía un mensaje
getMessages	Recupera la lista de mensajes
onNewMessage	Suscribe una acción a la entrada de un nuevo mensaje
onMessageDeleted	Suscribe una acción a la eliminación de un mensaje
onUserJoining	Suscribe una acción a la conexión de un usuario a la sala del feed
onUserLeaving	Suscribe una acción a la desconexión de un usuario a la sala del feed

Tabla 14.16: Especificación de la clase FeedRepository

14.1.2.3. GlobalRoomRepository

GlobalRoomRepository	
Descripción	Repositorio para la gestión de la conexión con la sala Global
Funciones	
join	Conecta al usuario a la sala Global
leave	Desconecta al usuario de la sala Global

Tabla 14.17: Especificación de la clase GlobalRoomRepository

14.1.2.4. LocationRepository

LocationRepository	
Descripción	Repositorio para operaciones relacionadas con la geolocalización
Funciones	
join	Suscribe al usuario a la sala de geolocalización
update	Actualiza la localización del usuario
leave	Desconecta al usuario de la sala de geolocalización
onExternalUpdate	Suscribe una acción a la llegada de actualizaciones
onUserLeaving	Suscribe una acción a la desconexión de un usuario

Tabla 14.18: Especificación de la clase LocationRepository

14.1.2.5. NotificationRepository

NotificationRepository	
Descripción	Repositorio para operaciones relacionadas con las notificaciones
Funciones	
getNotifications	Recupera la lista de notificaciones
onNotification	Suscribe una acción a la llegada de una nueva notificación
setAllAsRead	Marca todas las notificaciones como leídas
setAsRead	Marca una notificación como leída

Tabla 14.19: Especificación de la clase NotificationRepository

14.1.2.6. SessionRepository

SessionRepository	
Descripción	Repositorio para operaciones relacionadas con la sesión
Funciones	
signIn	Inicia la sesión del usuario
refresh	Refresca la sesión del usuario

Tabla 14.20: Especificación de la clase SessionRepository

14.1.2.7. TasksRepository

TasksRepository	
Descripción	Repositorio para operaciones relacionadas con las tareas
Funciones	
delete	Elimina una tarea
getTasks	Recupera las tareas
save	Guarda una tarea
update	Actualiza una tarea

Tabla 14.21: Especificación de la clase TasksRepository

14.1.2.8. UserRepository

UserRepository	
Descripción	Repositorio para operaciones relacionadas con los usuarios
Funciones	
update	Actualiza los datos del usuario
getCared	Recupera los datos del Paciente vinculado
requestBondingCode	Solicita un token de vinculación
sendBondingCode	Elimina un token de vinculación
getBonds	Recupera los vínculos del usuario

Tabla 14.22: Especificación de la clase UserRepository

14.1.3. API

14.1.3.1. ApiFactory

ApiFactory	
Descripción	Factoría de servicios de la API
Funciones	
get	Crea y regresa el servicio solicitado

Tabla 14.23: Especificación de la clase ApiFactory

14.1.3.2. SocketManager

SocketManager	
Descripción	Clase a cargo de la gestión del socket
Funciones	
connect	Conecta al usuario a la API WebSocket

Tabla 14.24: Especificación de la clase SocketManager

14.1.3.3. AuthService

AuthService	
Descripción	Servicio para la realización de peticiones al endpoint /auth
Funciones	
getSignIn	Realiza una petición a GET /auth/signin
getRefresh	Realiza una petición a GET /auth/refresh

Tabla 14.25: Especificación de la clase AuthService

14.1.3.4. FeedService

FeedService	
Descripción	Servicio para la realización de peticiones al endpoint /feed
Funciones	
getNotifications	Realiza una petición a GET /feed/notification
getMessages	Realiza una petición a GET /feed/message
postNotificationsRead	Realiza una petición a POST /feed/notification/read

Tabla 14.26: Especificación de la clase FeedService

14.1.3.5. TaskService

TaskService	
Descripción	Servicio para la realización de peticiones al endpoint /task
Funciones	
getTasks	Realiza una petición a GET /task
postTask	Realiza una petición a POST /task
deleteTask	Realiza una petición a DELETE /task
postTaskDone	Realiza una petición a POST /task/done
deleteTaskDone	Realiza una petición a DELETE /task/done

Tabla 14.27: Especificación de la clase TaskService

14.1.3.6. UserService

UserService	
Descripción	Servicio para la realización de peticiones al endpoint /user
Funciones	
patchUser	Realiza una petición a PATCH /user
getCared	Realiza una petición a GET /user/bond/cared
getBonds	Realiza una petición a GET /user/bond
postNewBond	Realiza una petición a POST /user/bond
getBondCode	Realiza una petición a GET /user/bond

Tabla 14.28: Especificación de la clase UserService

14.1.4. VO

14.1.4.1. Message

Message	
Descripción	Objeto representante de un mensaje
Propiedades	
id	Identidad del mensaje
submitter	Usuario que envió el mensaje
timestamp	Instante de creación del mensaje
content	Contenido del mensaje

Tabla 14.29: Especificación de la clase Message de la aplicación

14.1.4.2. Task

Task	
Descripción	Objeto representante de una tarea
Propiedades	
id	ID
submitter	Usuario que envió la tarea
timestamp	Instante de creación del mensaje
lastUpdate	Instante de última actualización del mensaje
title	Título
description	Descripción
done	Estado: hecha o no hecha

Tabla 14.30: Especificación de la clase Task de la aplicación

14.1.4.3. Session

Session	
Descripción	Objeto representante de una sesión
Propiedades	
auth	Token de autenticación
refresh	Token de refresco
expiration	Instante de expiración de la sesión

Tabla 14.31: Especificación de la clase Session de la aplicación

14.1.4.4. Notification

Notification	
Descripción	Objeto representante de una notificación
Propiedades	
id	ID
action	Acción notificada
instigator	Usuario que realizó la acción
timestamp	Instante de creación de la notificación
tags	Etiquetas de la notificación

Tabla 14.32: Especificación de la clase Notification de la aplicación

Action	
Descripción	Lista de acciones notificables
Valores	
BOND CREATED	Creación de un vínculo
BOND DELETED	Eliminación de un vínculo
TASK CREATED	Creación de una tarea
TASK DELETED	Eliminación de una tarea
TASK DONE	Actualización de una tarea a hecha
TASK UNDONE	Actualización de una tarea a no hecha
LOCATION SHARE	Ubicación empezando a ser compartida
LOCATION STOP	Ubicación dejando de ser compartida
Propiedades	
path	Ruta de la notificación
destiny	Actividad destino si la tiene

Tabla 14.33: Especificación de la clase Action de la aplicación

14.1.4.5. User

User	
Descripción	Objeto representante de un usuario
Propiedades	
id	ID
role	Rol
displayName	Nombre visible
phoneNumber	Número de teléfono
altPhoneNumber	Número alternativo de teléfono
address	Dirección postal
email	Dirección electrónica

Tabla 14.34: Especificación de la clase User de la aplicación

Role	
Descripción	Lista de posibles roles de los usuarios
Valores	
PATIENT	Pacientes
KEEPER	Cuidadores

Tabla 14.35: Especificación de la clase Role de la aplicación

Address	
Descripción	Objeto representante de una dirección postal
Propiedades	
firstLine	Primera línea
secondLine	Segunda línea
locality	Localidad
region	Región

Tabla 14.36: Especificación de la clase Address de la aplicación

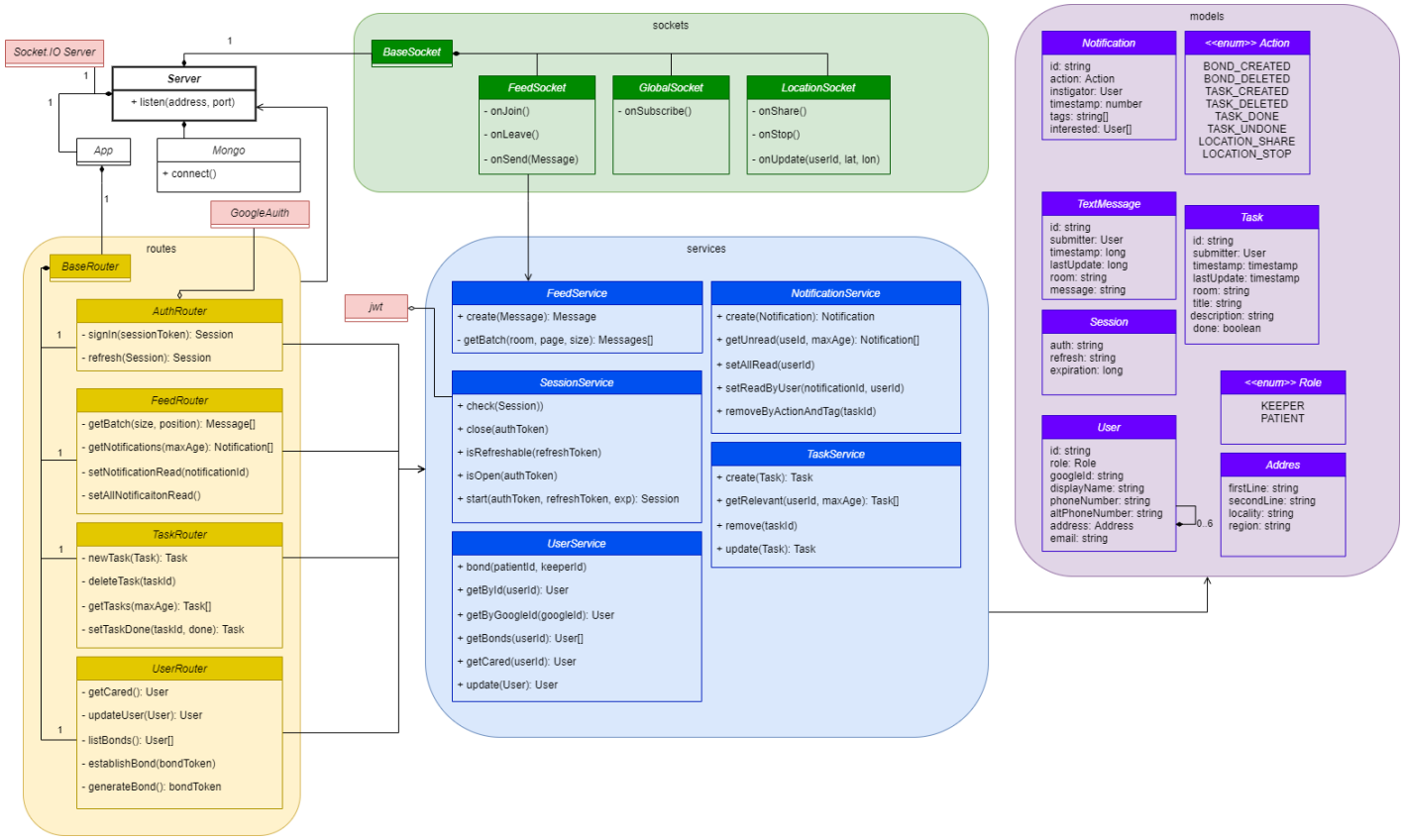


Figura 14.2: Diagrama de clases de la API

14.2. Clases de la API

El conjunto de clases general de la API a desarrollar se encuentra resumido en el diagrama de la 14.2. En el caso de la API los distinto módulos representarán no clases sino paquetes de funciones servidos por sus diferentes índices. Salvo en el caso del paquete de modelos que estará conformado por clases. Los módulos en rojo corresponden a librerías externas ya identificadas.

14.2.1. Root

Server	
Descripción	Módulo principal y global del sistema. Inicializa todos los sistemas de la API en su lanzamiento desde el punto de entrada de la aplicación
Funciones	
listen	Despliega el servidor en la direcciones y puerto especificados

Tabla 14.37: Especificación de la clase Server

App	
Descripción	Implementación del servidor Express, punto de entrada de las consultas REST

Tabla 14.38: Especificación de la clase App

Mongo	
Descripción	Módulo a cargo de la lógica relativa a la conexión de Mongoose con la base de datos
Funciones	
connect	Crea la conexión con la base de datos remota

Tabla 14.39: Especificación de la clase Mongo

14.2.2. Routes

BaseRouter	
Descripción	Manejador raíz encargado de redireccionar las peticiones REST al manejador encargado de cada una según la ruta

Tabla 14.40: Especificación de la clase BaseRouter

AuthRouter	
Descripción	Manejador de las peticiones de /auth
Funciones	
signIn	Crea una sesión para el usuario
signIn	Refresca la sesión del usuario

Tabla 14.41: Especificación de la clase AuthRouter

FeedRouter	
Descripción	Manejador de las peticiones de /feed
Funciones	
getBatch	Devuelve el conjunto de mensajes solicitado por el usuario
getNotifications	Devuelve las notificaciones pendientes del usuario
setNotificationRead	Marcar una notificación como leída
setAllNotificationsRead	Marca todas las notificaciones del usuario como leídas

Tabla 14.42: Especificación de la clase FeedRouter

TaskRouter	
Descripción	Manejador de las peticiones de /task
Funciones	
newTask	Crea una nueva tarea
deleteTask	Elimina la tarea especificada
getTasks	Retorna las tareas relevantes relacionadas con el usuario
setTaskDone	Marca una tarea como hecha/no hecha

Tabla 14.43: Especificación de la clase TaskRouter

UserRouter	
Descripción	Manejador de las peticiones de /user
Funciones	
getCared	Devuelve paciente vinculado con el usuario
updateUser	Actualiza la información del usuario
listBonds	Devuelve los vínculos del usuario
establishBond	Crea un vínculo entre usuarios
generateBond	Genera un código de vinculación

Tabla 14.44: Especificación de la clase UserRouter

14.2.3. Sockets

BaseSocket	
Descripción	Manejador raíz encargado de redireccionar los eventos del WebSocket al manejador encargado de cada una según la sala

Tabla 14.45: Especificación de la clase BaseSocket

FeedSocket	
Descripción	Manejador de la sala del Feed
Funciones	
onJoin	Gestiona la entrada de un usuario en la sala
onLeave	Gestione el abandono de la sala de un usuario
onSendMessage	Maneja el envío de un mensaje

Tabla 14.46: Especificación de la clase FeedSocket

GlobalSocket	
Descripción	Manejador de la sala global
Funciones	
onSubscribe	Gestiona la suscripción de un usuario a la sala

Tabla 14.47: Especificación de la clase GlobalSocket

LocationSocket	
Descripción	Manejador de la sala de la geolocalización
Funciones	
onShare	Gestiona la entrada de un usuario en la sala
onStop	Gestione el abandono de la sala de un usuario
onUpdate	Maneja el envío de una localización

Tabla 14.48: Especificación de la clase LocationSocket

14.2.4. Services

FeedService	
Descripción	Servicio a cargo de los mensajes del feed
Funciones	
create	Crea y persiste un nuevo mensaje
getBatch	Devuelve la lista de mensajes especificada

Tabla 14.49: Especificación de la clase FeedService de la API

NotificationService	
Descripción	Servicio a cargo de las notificaciones
Funciones	
create	Crea y persiste una nueva notificación
getUnread	Devuelve la lista de notificaciones no leídas de un usuario
setAllRead	Marca todas las notificaciones de un usuario como leídas
setReadByUser	Marca una notificación como leída
removeByActionAndTag	Elimina las notificaciones que tienen la misma acción y etiquetas

Tabla 14.50: Especificación de la clase NotificationService

SessionService	
Descripción	Servicio a cargo de las sesiones de usuario
Funciones	
check	Comprueba la validez de una sesión
close	Cierra la sesión especificada
isRefreshable	Comprueba si una sesión se puede refrescar
isOpen	Comprueba si una sesión está activa
start	Crea una nueva sesión de usuario

Tabla 14.51: Especificación de la clase SessionService

TaskService	
Descripción	Servicio a cargo de las tareas
Funciones	
create	Crea y persiste una nueva tarea
getRelevant	Devuelve la lista de tareas relevantes de un usuario
remove	Elimina la tarea especificada
update	Actualiza una tarea

Tabla 14.52: Especificación de la clase TaskService de la API

UserService	
Descripción	Servicio a cargo de los datos de usuario
Funciones	
bond	Crea un vínculo entre usuarios
getById	Retorna el usuario con la identidad especificada
getByGoogleId	Retorna el usuario con la identidad de Google especificada
getBonds	Devuelve los vínculos de un usuario
getCared	Devuelve el paciente vinculado de un usuario
update	Actualiza los datos de un usuario

Tabla 14.53: Especificación de la clase UserService de la API

14.2.5. Models

Action	
Descripción	Lista de posibles acciones notificables
Valores	
BOND CREATED	Creación de un vínculo
BOND DELETED	Eliminación de un vínculo
TASK CREATED	Creación de una tarea
TASK DELETED	Eliminación de una tarea
TASK DONE	Actualización de una tarea a hecha
TASK UNDONE	Actualización de una tarea a no hecha
LOCATION SHARE	Ubicación empezando a ser compartida
LOCATION STOP	Ubicación dejando de ser compartida

Tabla 14.54: Especificación de la clase Action de la API

Address	
Descripción	Abstracción de direcciones postales
Propiedades	
firstLine	Primera línea (por ejemplo, calle y número)
secondLine	Segunda línea (por ejemplo, piso y puerta)
locality	Localidad
region	Región (por ejemplo, provincia o estado)

Tabla 14.55: Especificación de la clase Address de la API

Notification	
Descripción	Entidad representante de las notificaciones de la aplicación
Propiedades	
id	ID
action	Acción a notificar
instigator	Autor de la acción
timestamp	Instante de realización de la acción
tags	Etiquetas con información extra de la notificación
interested	Lista de usuarios receptores de la notificación

Tabla 14.56: Especificación de la clase Notification de la API

Role	
Descripción	Lista de posibles roles de los usuarios
Valores	
PATIENT	Pacientes
KEEPER	Cuidadores

Tabla 14.57: Especificación de la clase Role de la API

Session	
Descripción	Entidad representante de una sesión de usuario
Propiedades	
auth	Token de autenticación de la sesión
refresh	Token de refresco de la sesión
expiration	Instante de expiración de la sesión

Tabla 14.58: Especificación de la clase Session de la API

Task	
Descripción	Entidad representante de las notificaciones de la aplicación
Propiedades	
id	ID
submitter	Autor de la tarea
timestamp	Instante de envío de la tarea
lastUpdate	Última actualización de la tarea
room	Sala de envío de la tarea
title	Título
description	Descripción
done	Estado: hecha/no hecha

Tabla 14.59: Especificación de la clase Task de la API

TextMessage	
Descripción	Entidad representante de los mensajes de texto del Feed
Propiedades	
id	ID
submitter	Autor del mensaje
timestamp	Instante de envío del mensaje
lastUpdate	Última actualización del mensaje
room	Sala de envío del mensaje
message	Mensaje enviado

Tabla 14.60: Especificación de la clase TextMessage de la API

User	
Descripción	Entidad representante de los usuarios de la aplicación
Propiedades	
id	ID
role	Rol
googleId	Identidad de Google
displayName	Nombre para mostrar al resto de usuarios
phoneNumber	Teléfono principal
altPhoneNumber	Teléfono alternativo
address	Dirección postal
email	Dirección electrónica
bonds	Lista de usuarios vinculados

Tabla 14.61: Especificación de la clase User de la API

15. Especificación del plan de pruebas

15.1. Pruebas unitarias

Todas las funciones desarrolladas en la API, a excepción de las funciones de manejo de endpoints y de eventos del WebSocket serán probadas con pruebas unitarias de forma exhaustiva antes de que dicha función pueda ser aceptada y añadida a la rama principal del desarrollo. Las pruebas unitarias de la API se llevarán a cabo con la librería **Jest** (Apartado 24.2.2.9) apoyada en **ts-jest** (Apartado 24.2.2.17) para dotarla de compatibilidad con Typescript.

Todos los componentes que se relacionen con otros recibirán imitaciones (o mocks) en vez de los componentes originales para garantizar su funcionamiento correcto de forma aislada. Para este proceso se usará la librería **MongoDB In-Memory Server** (Apartado 24.2.2.12) de cara a poder hacer una imitación de la base de datos con la que probar los componentes que se comuniquen con ella de forma aislada. El resto de imitaciones se creará con la utilidad de Jest.

En la aplicación móvil se realizarán pruebas unitarias de todas las clases con la única excepción de las actividades al ser componentes muy acoplados y que tienen una pertenencia más cercana al ámbito de Pruebas de integración. El resto de clases contarán con sus respectivas pruebas unitarias siguiendo la misma estrategia de imitación de cualquier componente acoplado. La librería de pruebas será **JUnit 4** (Apartado 24.2.1.5) apoyada en **Mockk** (Apartado 24.2.1.7) y **Mock Web Server** (Apartado 24.2.1.8) para la creación de todas las imitaciones necesarias, la primera para los componentes de la aplicación y la segunda para la replicar la comunicación con la API.

15.2. Pruebas de integración

Las pruebas de integración de la API se irán creando según se complete la funcionalidad de los distintos endpoints y eventos del WebSocket intentando replicar interacciones completas con simulaciones de las peticiones que se esperan recibir de la aplicación. La simulación de las peticiones HTTP se realizará con la librería **SuperTest** (24.2.2.16), las de los eventos WebSocket con el cliente suministrado en la librería de SocketIO (Apartado 24.2.2.15). En la realización de todas estas pruebas se seguirá

imitando la base de datos con el mismo mock de las pruebas unitarias. De ser posible dentro del tiempo de desarrollo se crearía otra colección en la base de datos en la nube y se realizarían en dicho entorno de pruebas.

Las pruebas de integración de la aplicación se corresponderán con los casos de uso presentados en el Capítulo 12 y se llevarán a cabo con la librería **Espresso** (Apartado 24.2.1.4) de Android. Igual que en el caso de las pruebas unitarias se simularán las comunicaciones con la API por medio de la misma librería de imitación. La opción óptima pasaría por el despliegue de una API de pruebas con la que realizar las comunicaciones, sin embargo, la logística y las limitaciones de este desarrollo deja esa posibilidad fuera del alcance.

15.3. Pruebas de sistema

Las pruebas de sistema se empezarán a realizar en la etapa final del desarrollo y una vez la API haya sido desplegada en la nube como se indicará en el Apartado 16.1. Se realizarán con la aplicación instalada en móviles con Android de diferentes marcas y versiones del sistema operativo y contra la API desplegada. Se probarán manualmente todos los casos de uso del Capítulo 12 y otras posibilidades que se consideren oportunas. Para probar las funciones de comunicación se realizarán pruebas con hasta tres usuarios interconectados.

15.4. Pruebas de usabilidad

Se desconoce si este plan de pruebas podrá ser ejecutado en el proyecto actual por diversas imposibilidades en la obtención de colaboración por parte de asociaciones especializadas. En caso de conseguir la colaboración de alguna asociación de enfermos con Alzheimer se crearán dos cuestionarios con una serie de preguntas acerca de las funciones y la usabilidad de la aplicación. Uno de los cuestionarios irá dirigido a pacientes de Alzheimer y el otro a los cuidadores de los mismos.

16. Especificación del plan de despliegue

16.1. Despliegue de la API

La API será desplegada en la nube de Azure. El despliegue de esta se llevará a cabo por medio de la integración continua a través de la herramienta GitHub Actions (Apartado 25.2.2). El sistema se desplegará en un entorno AppService de Node.js con el **plan de tarifa B1** (Cuadro 16.1) es un plan de desarrollo y pruebas que se considera suficiente para el alcance del proyecto. En caso de realizar un lanzamiento oficial habría que escalar este despliegue a un plan de tarifas de producción

Plan de tarifa B1	
Equivalente de proceso de serie A	
Total de ACU	100
Memoria	1,75 GB
Almacenamiento	10GB
Escala manual	Hasta 3 instancias
Coste estimado	11.08€/mes

Tabla 16.1: Características del Plan B1

16.2. Aplicación

Cae fuera del alcance del proyecto realizar una publicación de la aplicación en la PlayStore o algún otro servicio de publicación. En un principio se estudiaba dicha publicación de cara a la obtención de experiencia por parte del equipo desarrollador en ese proceso, pero ante el hecho de que los componentes de dicho equipo ya han realizado la publicación de una aplicación se ha terminado descartando. El único lanzamiento público de la aplicación será enfocado al enfoque abierto y colaboracional del proyecto y será **la publicación de la APK** como *release* del repositorio en GitHub. Esto se llevará a cabo máximo una semana antes de la entrega del proyecto para valoración.

Parte IV

Diseño

17. Arquitectura del sistema

17.1. Diagrama de paquetes

17.1.1. API

El diagrama de paquetes de la API es la Figura 17.1. Esta API se cimentará sobre el **Server**, que será el punto de inicio del sistema. Esta entidad iniciará los funciones de sus tres componentes principales: **App**, con la API REST desarrollada en Express (Apartado 24.2.2.5); **Mongo**, con el cliente Mongoose (Apartado 24.2.2.13) de la base de datos; y **socketio** (Apartado 24.2.2.15), con la lógica de la API WebSocket.

Todo el funcionamiento de la API REST se haya en el paquete **routers**, el cuál está dividido a su vez en paquetes relativos a los endpoints ofrecidos, paquetes que contienen sus enrutadores respectivos. Por otro lado, el funcionamiento de la API WebSocket está en el paquete **sockets** y tiene una distribución similar pero con manejadores de eventos en vez de enrutadores.

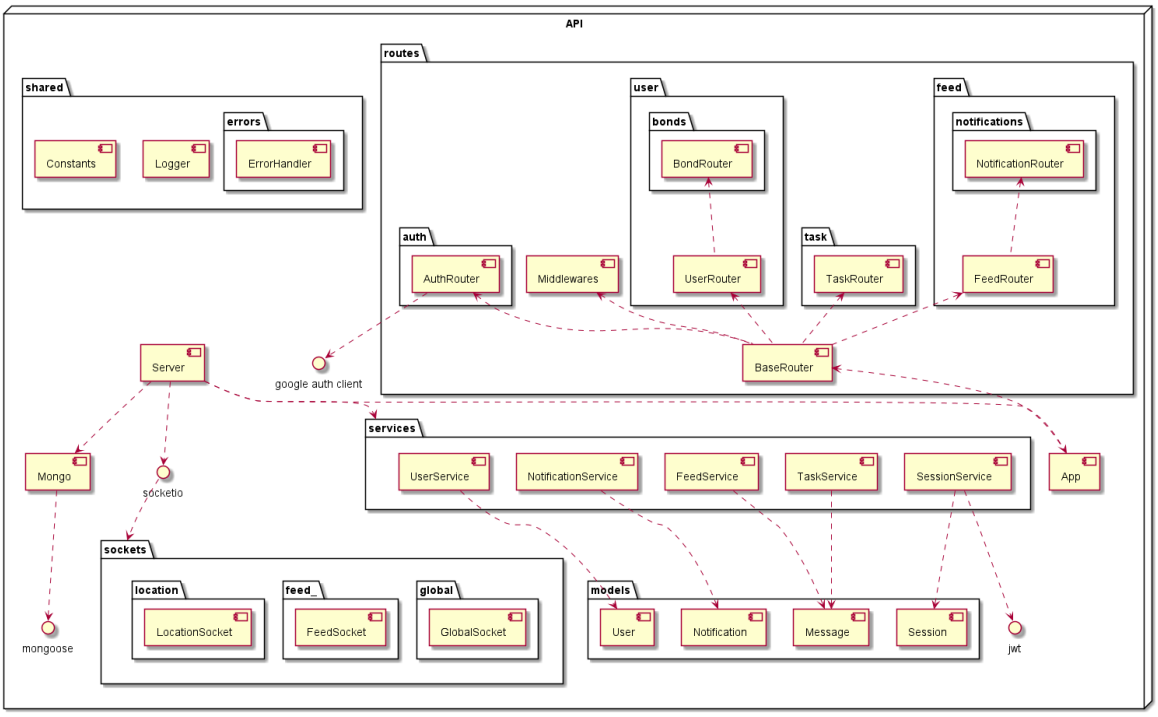


Figura 17.1: Diagrama de paquetes de la API

Los paquetes **services** y **models** contienen toda la lógica de datos del sistema y son accedidos por las entidades de los dos paquetes mencionados con anterioridad. Las

entidades del modelo de datos (ver Capítulo 19) están en el paquete **models** mientras que los servicios que gestionan la persistencia y recuperación de dichas entidades están en **services**. Por último, en el paquete **shared** se encuentran funciones, datos y entidades comunes a todos los componentes del sistema como los errores, las cadenas de texto o el registro del sistema.

17.1.2. Aplicación móvil

Como se puede ver en la Figura 17.2, la aplicación móvil tiene un diagrama de paquetes que sigue la arquitectura de capas del Model-View-View Model que ya planteamos en la fase de análisis en la Apartado 11.1. La parte clave es el paquete de **ui**, que se encuentra dividido a su vez en paquetes relativos a las distintas pantallas que tendrá la aplicación móvil, conteniendo la actividad, el modelo de la vista y cualquier otra pantalla, diálogo o fragmento auxiliar de la misma.

La interfaz de usuario se comunicará con los los repositorios del paquete **repositories** por medio de los objetos de valor (o *value objects*) que se definirán en el paquete **vo**. Por último, la capa más profunda de la aplicación es aquella que se conectará con las API, lo que da nombre al paquete en el que se albergan **api**. Este paquete expondrá una interfaz para el manejo del WebSocket y una factoría para los servicios del paquete **services**.

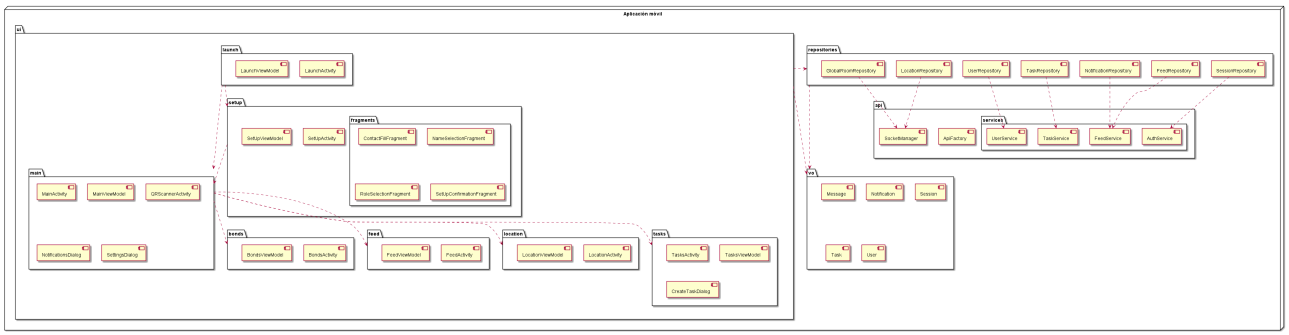


Figura 17.2: Diagrama de paquetes de la aplicación móvil

17.2. Diagrama de despliegue

La aplicación móvil se instalará en los dispositivos Android de los usuarios por medio de la APK o el Android Application Bundle de la aplicación. La API se desplegará en una instancia AppService de Microsoft Azure como la que se especificó en el Apartado 16.1. La base de datos estará hospedada en el servicio en la nube de MongoDB, MongoDB Atlas. Las aplicaciones de los clientes se comunicarán con la API por me-

dio de peticiones HTTP REST o de eventos a través del WebSocket. La API, por su parte, se comunicará por red con la base de datos. Esto todo puede visualizarse en la Figura 17.3.

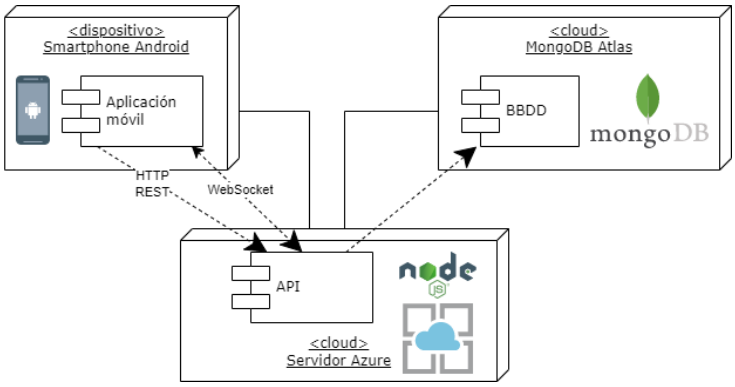


Figura 17.3: Diagrama de despliegue del sistema

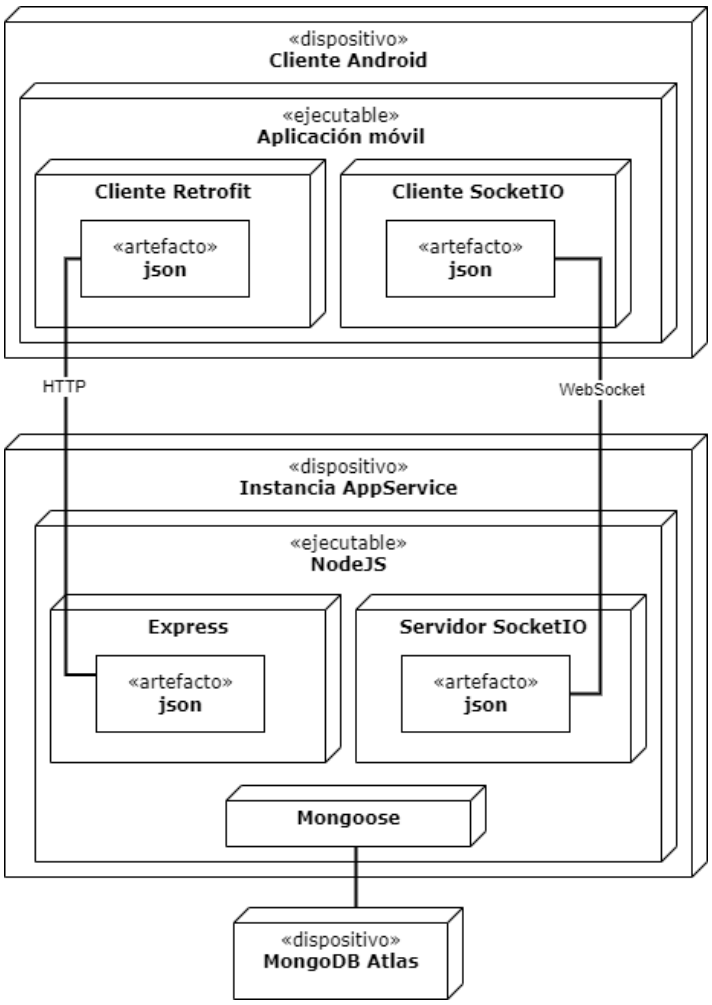


Figura 17.4: Diagrama de componentes del sistema

17.3. Diagrama de componentes

El sistema estará compuesto por tres componentes principales análogos a las entidades de la sección anterior. Por un lado estará el **cliente**, que será el dispositivo del usuario y sobre el que se ejecutará la aplicación móvil; por otro lado estará la **API** ejecutada sobre una instancia de AppService; y por último, la **base de datos** alojada en MongoDB Atlas. En la aplicación habrá un cliente HTTP y otro cliente Socket.io (Apartado 24.2.2.15) que se comunicarán con los componentes Express (Apartado 24.2.2.5) y servidor Socket.io de la API, respectivamente. La comunicación definitiva entre la API y la base de datos se llevará a cabo con el componente Mongoose (Apartado 24.2.2.13). Véase Figura 17.4.

18. Diseño de clases

18.1. Clases de la API

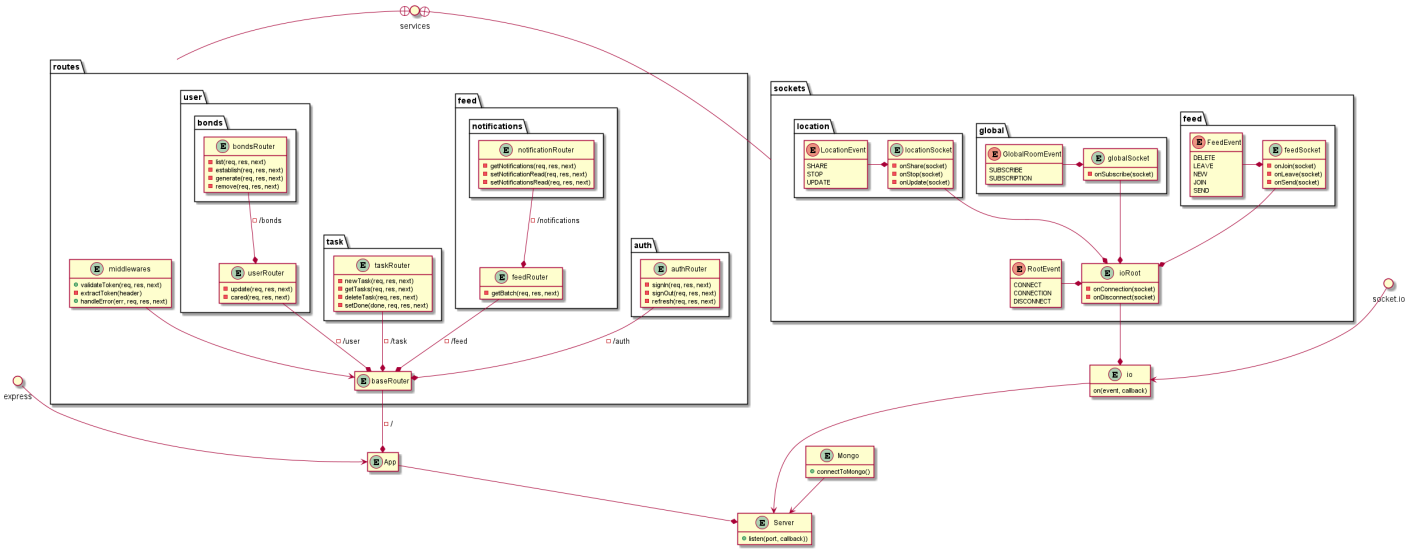


Figura 18.1: Primera parte del diagrama de clases de la API

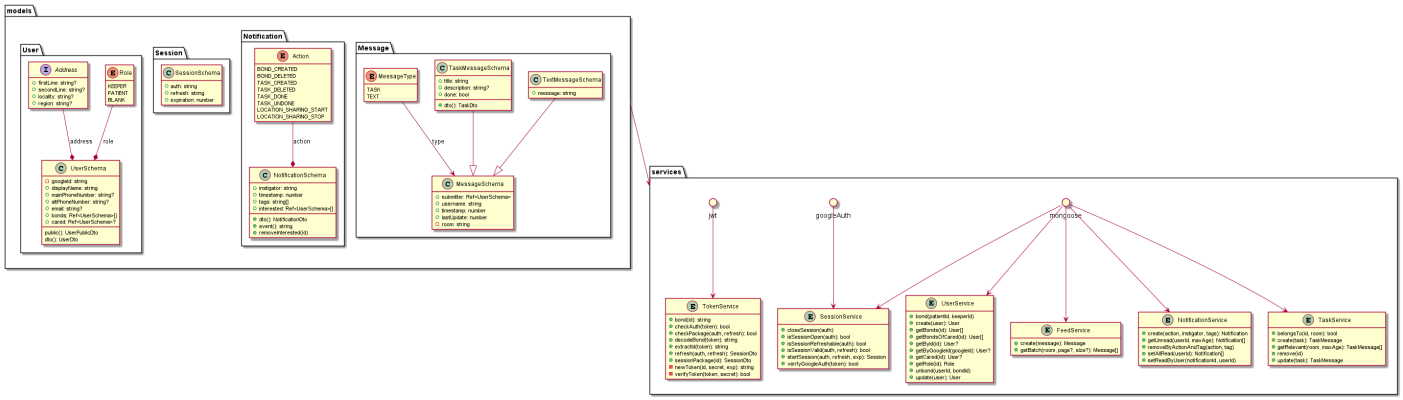


Figura 18.2: Segunda parte del diagrama de clases de la API

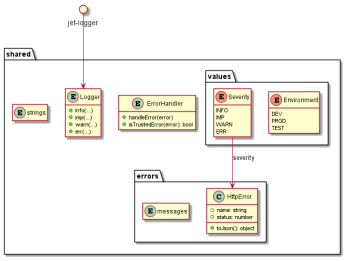


Figura 18.3: Tercera parte del diagrama de clases de la API

18.1.1. Root

Server	
Descripción	Módulo principal del sistema. Inicializa todos los sistemas de la API en su lanzamiento desde el punto de entrada de la aplicación
Funciones	
listen	Despliega el servidor en la direcciones y puerto especificados

Tabla 18.1: Documentación de la entidad Server

App	
Descripción	Implementación del servidor Express, punto de entrada REST

Tabla 18.2: Documentación de la entidad App

Mongo	
Descripción	Módulo a cargo de la lógica relativa a la conexión de Mongoose con la base de datos
Funciones	
connectToMongo	Crea la conexión con la base de datos remota

Tabla 18.3: Documentación de la entidad Mongo

io	
Descripción	Implementación del servidor SocketIO, punto de entrada de los eventos WebSocket
Funciones	
on	Define un evento y la función a realizar en el mismo

Tabla 18.4: Documentación de la entidad io

18.1.2. Routes

baseRouter	
Descripción	Manejador raíz encargado de redireccionar las peticiones REST al manejador encargado de cada una según la ruta

Tabla 18.5: Documentación de la entidad baseRouter

authRouter	
Descripción	Manejador de las peticiones de /auth
Funciones	
signIn	Crea una sesión para el usuario
signOut	Cierra la sesión de un usuario
refresh	Refresca la sesión del usuario

Tabla 18.6: Documentación de la entidad authRouter

feedRouter	
Descripción	Manejador de las peticiones de /feed
Funciones	
getBatch	Devuelve el conjunto de mensajes solicitado por el usuario

Tabla 18.7: Documentación de la entidad feedRouter

notificationRouter	
Descripción	Manejador de las peticiones de /feed/notifications
Funciones	
getNotifications	Devuelve las notificaciones pendientes del usuario
setNotificationRead	Marcar una notificación como leída
setNotificationsRead	Marca todas las notificaciones del usuario como leídas

Tabla 18.8: Documentación de la entidad notificationRouter

taskRouter	
Descripción	Manejador de las peticiones de /task
Funciones	
newTask	Crea una nueva tarea
deleteTask	Elimina la tarea especificada
getTasks	Retorna las tareas relevantes relacionadas con el usuario
setTaskDone	Marca una tarea como hecha/no hecha

Tabla 18.9: Documentación de la entidad taskRouter

userRouter	
Descripción	Manejador de las peticiones de /user
Funciones	
cared	Devuelve la información del paciente vinculado con el usuario
update	Actualiza la información del usuario

Tabla 18.10: Documentación de la entidad userRouter

bondsRouter	
Descripción	Manejador de las peticiones de /user/bonds
Funciones	
list	Devuelve los vínculos del usuario
establish	Crea un vínculo entre usuarios
generate	Genera un código de vinculación
remove	Elimina un vínculo entre usuarios

Tabla 18.11: Documentación de la entidad bondsRouter

18.1.3. Sockets

ioRoot	
Descripción	Manejador raíz encargado de los eventos de conexión y desconexión y de gestionar el registro de manejadores

Tabla 18.12: Documentación de la entidad ioRoot

RootEvent	
Descripción	Eventos de conexión al socket
Valores	
CONNECT	Intento de conexión al socket
CONNECTION	Conexión exitosa al socket
DISCONNECT	Desconexión del socket

Tabla 18.13: Documentación del enumerado RootEvent

feedSocket	
Descripción	Manejador de la sala del Feed
Funciones	
onJoin	Gestiona la entrada de un usuario en la sala
onLeave	Gestione el abandono de la sala de un usuario
onSend	Maneja el envío de un mensaje

Tabla 18.14: Documentación de la entidad feedSocket

FeedEvent	
Descripción	Eventos de la sala del feed
Valores	
DELETE	Eliminación de un mensaje
LEAVE	Abandono de la sala
NEW	Nuevo usuario en la sala
JOIN	Unión a la sala
SEND	Envío de un nuevo mensaje

Tabla 18.15: Documentación del enumerado FeedEvent de la API

globalSocket	
Descripción	Manejador de la sala Global
Funciones	
onSubscribe	Gestiona la suscripción de un usuario a la sala

Tabla 18.16: Documentación de la entidad globalSocket

GlobalEvent	
Descripción	Eventos de la sala global
Valores	
SUBSCRIBE	Unión a la sala
SUBSCRIPTION	Nuevo usuario en la sala

Tabla 18.17: Documentación del enumerado GlobalEvent de la API

locationSocket	
Descripción	Manejador de la sala de la geolocalización
Funciones	
onShare	Gestiona la entrada de un usuario en la sala
onStop	Gestione el abandono de la sala de un usuario
onUpdate	Maneja el envío de una localización

Tabla 18.18: Documentación de la entidad locationSocket

LocationEvent	
Descripción	Eventos de la sala Location
Valores	
SHARE	Unión a la sala
STOP	Abandono de la sala
UPDATE	Envío de una nueva ubicación

Tabla 18.19: Documentación del enumerado LocationEvent de la API

18.1.4. Services

FeedService	
Descripción	Servicio a cargo de los mensajes del feed
Funciones	
create	Crea y persiste un nuevo mensaje
getBatch	Devuelve la lista de mensajes especificada

Tabla 18.20: Documentación de la clase FeedService de la API

NotificationService	
Descripción	Servicio a cargo de las notificaciones
Funciones	
create	Crea y persiste una nueva notificación
getUnread	Devuelve la lista de notificaciones no leídas de un usuario
removeByActionAndTag	Elimina las notificaciones que tienen la misma acción y etiquetas
setAllRead	Marca todas las notificaciones de un usuario como leídas
setReadByUser	Marca una notificación como leída

Tabla 18.21: Documentación de la clase NotificationService

SessionService	
Descripción	Servicio a cargo de las sesiones de usuario
Funciones	
closeSession	Cierra la sesión especificada
isSessionOpen	Comprueba si una sesión está activa
isSessionRefreshable	Comprueba si una sesión se puede refrescar
isSessionValid	Comprueba si una sesión es válida
startSession	Crea una nueva sesión de usuario
verifyGoogleAuth	Verifica un token de autenticación de Google

Tabla 18.22: Documentación de la clase SessionService

TaskService	
Descripción	Servicio a cargo de las tareas
Funciones	
belongsTo	Comprueba si la tarea pertenece a un usuario
create	Crea y persiste una nueva tarea
getRelevant	Devuelve la lista de tareas relevantes de un usuario
remove	Elimina la tarea especificada
update	Actualiza una tarea

Tabla 18.23: Documentación de la clase TaskService de la API

TokenService	
Descripción	Servicio para proveer y gestionar tokens
Funciones	
bond	Genera un token de vinculación
checkAuth	Comprueba la validez de un token de autenticación
checkPackage	Comprueba la validez de una dupla de tokens de sesión
decodeBond	Descodifica un token de vinculación
extractId	Devuelve la ID de usuario almacenada en un token
newToken	Produce un nuevo token según la ID de usuario, el secreto y el tiempo de expiración
refresh	Genera una nueva tupla de tokens de sesión a partir una válida
session	Genera una tupla de tokens de sesión nueva
verifyToken	Verifica la validez de un token

Tabla 18.24: Documentación de la clase TokenService

UserService	
Descripción	Servicio a cargo de los datos de usuario
Funciones	
bond	Crea un vínculo entre usuarios
create	Crea un nuevo usuario
getBonds	Devuelve los vínculos de un usuario
getBondsOfCared	Devuelve los vínculos del paciente vinculado
getById	Retorna el usuario con la identidad especificada
getByGoogleId	Retorna el usuario con la identidad de Google especificada
getCared	Devuelve el paciente vinculado de un usuario
getRole	Devuelve el rol de un usuario
unbond	Elimina el vínculo entre dos usuarios
update	Actualiza los datos de un usuario

Tabla 18.25: Documentación de la clase UserService de la API

18.1.5. Models

Action	
Descripción	Accciones notificables
Valores	
BOND_CREATED	Creación de un vínculo
BOND_DELETED	Eliminación de un vínculo
TASK_CREATED	Creación de una tarea
TASK_DELETED	Eliminación de una tarea
TASK_DONE	Actualización de una tarea a hecha
TASK_UNDONE	Actualización de una tarea a no hecha
LOCATION_SHARING_START	Ubicación empezando a ser compartida
LOCATION_SHARING_STOP	Ubicación dejando de ser compartida

Tabla 18.26: Documentación del enumerado Action de la API

Address	
Descripción	Abstracción de direcciones postales
Propiedades	
firstLine	Primera línea (por ejemplo, calle y número)
secondLine	Segunda línea (por ejemplo, piso y puerta)
locality	Localidad
region	Región (por ejemplo, provincia o estado)

Tabla 18.27: Documentación de la interfaz Address de la API

MessageType	
Descripción	Tipos de mensajes
Valores	
TASK	Tarea
TEXT	Mensaje de texto

Tabla 18.28: Documentación del enumerado MessageType de la API

MessageSchema	
Descripción	Esquema base de los mensajes
Propiedades	
submitter	Referencia a la entidad del autor
username	Nombre del autor
timestamp	Instante de envío
lastUpdate	Última actualización
room	Sala de envío

Tabla 18.29: Documentación del esquema de Message de la API

NotificationSchema	
Descripción	Esquema de las notificaciones de la aplicación
Propiedades	
action	Acción a notificar
instigator	Autor de la acción
timestamp	Instante de realización de la acción
tags	Etiquetas con información extra de la notificación
interested	Lista de usuarios receptores de la notificación
Funciones	
dto	Devuelve un NotificationDto de la notificación
event	Devuelve el código de evento del WebSocket
removeInterested	Elimina al usuario especificado de la lista de interesados

Tabla 18.30: Documentación del esquema de Notification de la API

Role	
Descripción	Roles de los usuarios
Valores	
PATIENT	Pacientes
KEEPER	Cuidadores
BLANK	Sin rol

Tabla 18.31: Documentación del enumerado Role de la API

SessionSchema	
Descripción	Esquema de una sesión de usuario
Propiedades	
auth	Token de autenticación de la sesión
refresh	Token de refresco de la sesión
expiration	Instante de expiración de la sesión

Tabla 18.32: Documentación del esquema de Session de la API

TaskMessageSchema	
Descripción	Esquema de una tarea
Propiedades	
title	Título
description	Descripción
done	Estado: hecha/no hecha
Funciones	
dto	Devuelve un TaskDto de la tarea

Tabla 18.33: Documentación del esquema de TaskMessage de la API

TextMessageSchema	
Descripción	Esquema de los mensajes de texto del Feed
Propiedades	
message	Mensaje enviado

Tabla 18.34: Documentación del esquema de TextMessage de la API

UserSchema	
Descripción	Esquema de los usuarios de la aplicación
Propiedades	
role	Rol
googleId	Identidad de Google
displayName	Nombre para mostrar al resto de usuarios
mainPhoneNumber	Teléfono principal
altPhoneNumber	Teléfono alternativo
address	Dirección postal
email	Dirección electrónica
bonds	Lista de usuarios vinculados de los pacientes
cared	Usuario vinculado de los cuidadores

Tabla 18.35: Documentación del esquema de User de la API

18.1.6. Shared

Logger	
Descripción	Encapsulación de la lógica del registro
Funciones	
info	Registra un mensaje de información
imp	Registra un mensaje importante
warn	Registra un mensaje de advertencia
err	Registra un mensaje de error

Tabla 18.36: Documentación de la entidad Logger

ErrorHandler	
Descripción	Manejador de errores
Funciones	
handleError	Gestiona y procesa un error
isTrustedError	Comprueba si es un error conocido

Tabla 18.37: Documentación de la entidad ErrorHandler

HttpError	
Descripción	Errores con respuesta HTTP definida
Propiedades	
name	Nombre
severity	Severidad
status	Código de estado HTTP
Funciones	
toJson	Devuelve un JSON para enviar el error como respuesta

Tabla 18.38: Documentación de la clase HttpError

Severity	
Descripción	Severidad
Valores	
INFO	Información
IMP	Información importante
WARN	Advertencia
ERROR	Error severo

Tabla 18.39: Documentación del enumerado Severity de la API

Environment	
Descripción	Entornos de ejecución
Valores	
DEV	Desarrollo
PROD	Producción
TEST	Pruebas

Tabla 18.40: Documentación del enumerado Environment de la API

18.2. Clases de la aplicación móvil

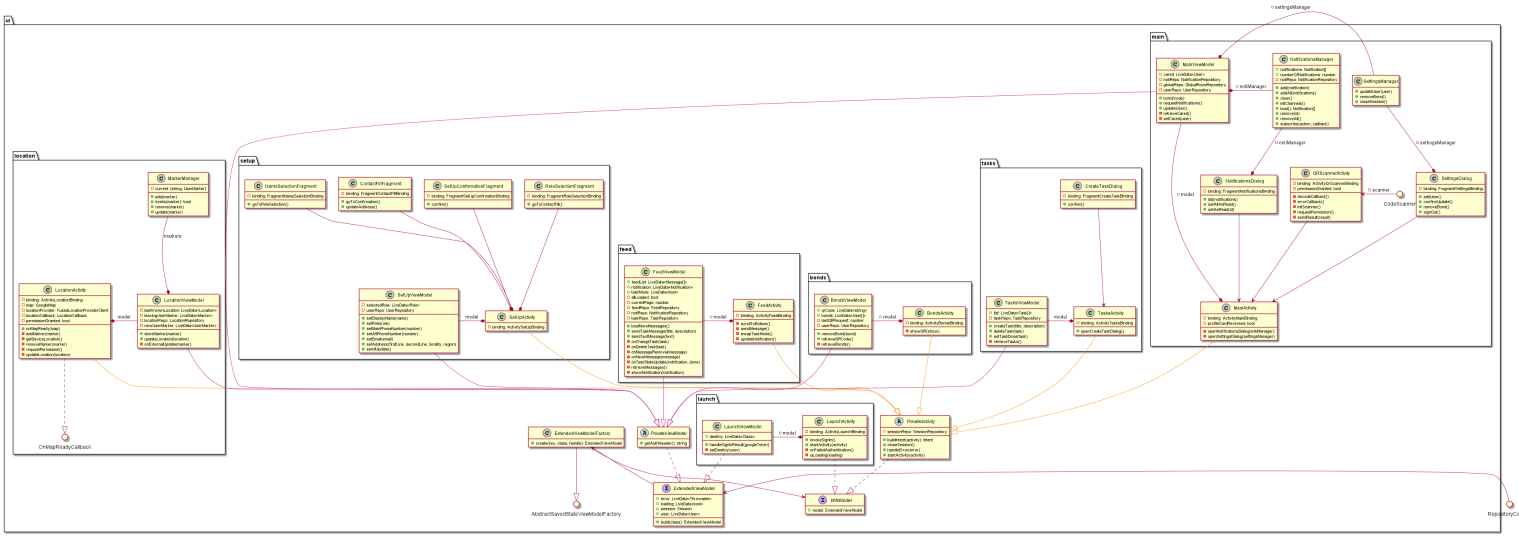


Figura 18.4: Primera parte del diagrama de clases de la aplicación móvil

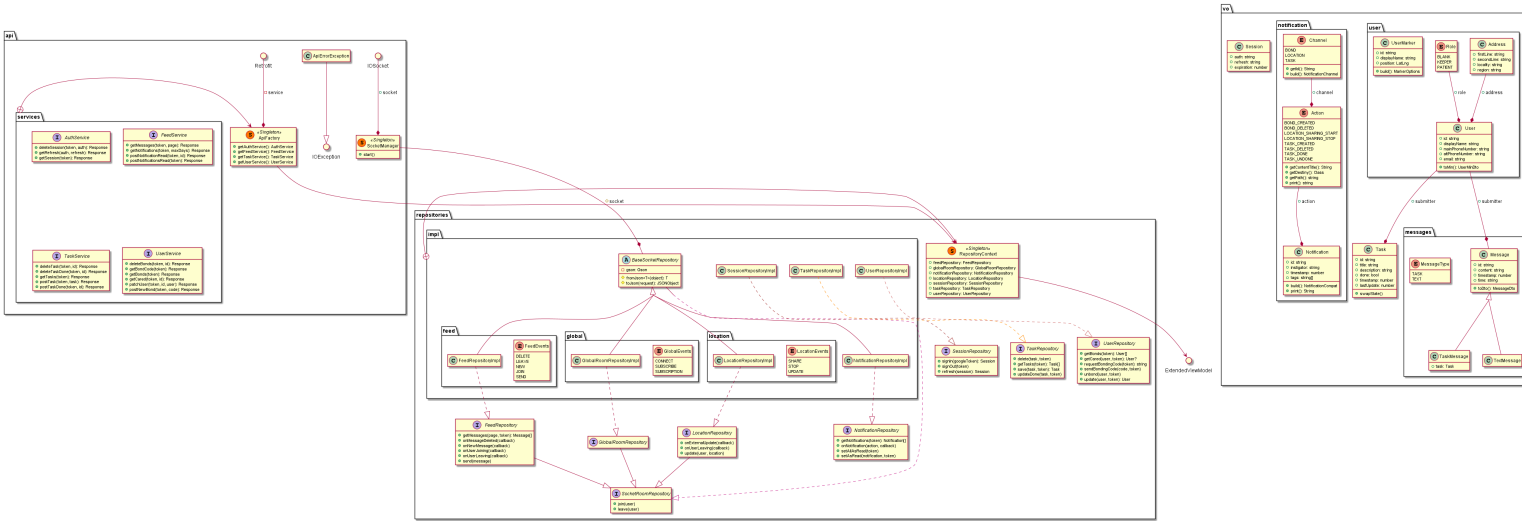


Figura 18.5: Segunda parte del diagrama de clases de la aplicación móvil

18.2.1. UI

ExtendedViewModel	
Descripción	Interfaz común para los ViewModel a emplear en la aplicación
Propiedades	
error	Contenedor del último error ocurrido
loading	Contenedor del estado de carga de la pantalla
session	Sesión actual
user	Contenedor de los datos del usuario
Funciones	
build	Constructor para la creación en ExtendedViewModelFactory (18.43)

Tabla 18.41: Documentación de la interfaz ExtendedViewModel

WithModel	
Descripción	Interfaz para clases que implementen un ExtendedViewModel (18.41)
Propiedades	
model	ExtendedViewModel implementado

Tabla 18.42: Documentación de la interfaz WithModel

ExtendedViewModelFactory	
Descripción	Factoría de clases que implementan ExtendedViewModel (18.41)
Funciones	
create	Crea el ExtendedViewModel especificado

Tabla 18.43: Documentación de la clase ExtendedViewModelFactory

PrivateViewModel	
Descripción	Implementación base de las clases ExtendedViewModel
Funciones	
getAuthHeader	Obtiene y devuelve el encabezado de autenticación para las peticiones REST

Tabla 18.44: Documentación de la clase PrivateViewModel

PrivateActivity	
Descripción	Implementación base de las actividades que implementen WithModel
Propiedades	
sessionRepo	Repositorio para operaciones relacionadas con la sesión
Funciones	
buildIntent	Construye una intención de navegación a otra actividad con la información necesaria
closeSession	Elimina los datos de sesión y retorna a LaunchActivity (18.46)
handleError	Implementación por defecto para manejo de errores
startActivity	Inicia la actividad especificada

Tabla 18.45: Documentación de la clase PrivateActivity

18.2.1.1. Launch

LaunchActivity	
Descripción	Actividad de inicio de la aplicación con el inicio de sesión
Propiedades	
binding	Vista vinculada
model	Modelo de vista
Funciones	
invokeSign	Inicia el proceso de inicio de sesión
startActivity	Inicia la siguiente actividad
onFailedAuthentication	Gestiona errores de autenticación
uiLoading	Cambia el estado de carga de la interfaz

Tabla 18.46: Documentación de la clase LaunchActivity

LaunchViewModel	
Descripción	Modelo de la vista de LaunchActivity
Propiedades	
destiny	Contenedor para la siguiente actividad a la que navegar
Funciones	
handleSignInResult	Gestiona el resultado del intento de inicio de sesión con Google
setDestiny	Modifica el destino de navegación

Tabla 18.47: Documentación de la clase LaunchViewModel

18.2.1.2. Set up

SetUpActivity	
Descripción	Actividad de configuración de la cuenta de usuario en la creación de esta
Propiedades	
binding	Vista vinculada
model	Modelo de vista

Tabla 18.48: Documentación de la clase SetUpActivity

SetUpViewModel	
Descripción	Modelo de la vista de SetUpActivity
Propiedades	
selectedRole	Contenedor del rol seleccionado
userRepo	Repositorio para gestionar los datos de usuario
Funciones	
setDisplayName	Establece el nombre
setRole	Establece el rol
setMainPhoneNumber	Establece el teléfono principal
setAltPhoneNumber	Establece el teléfono secundario
setEmail	Establece el email
setAddress	Establece la dirección postal
sendUpdate	Envía los datos del usuario para confirmarlo

Tabla 18.49: Documentación de la clase SetUpViewModel

NameSelectionFragment	
Descripción	Fragmento de selección del nombre del usuario
Propiedades	
binding	Vista vinculada
Funciones	
goToRoleSelection	Navega a RoleSelectionFragment (18.51)

Tabla 18.50: Documentación de la clase NameSelectionFragment

RoleSelectionFragment	
Descripción	Fragmento de selección del rol del usuario
Propiedades	
binding	Vista vinculada
Funciones	
goToContactFill	Navega a ContactFillFragment (18.52)

Tabla 18.51: Documentación de la clase RoleSelectionFragment

ContactFillFragment	
Descripción	Fragmento de especificación de datos de contacto del usuario
Propiedades	
binding	Vista vinculada
Funciones	
goToConfirmation	Navega a SetUpConfirmationFragment (18.53)
updateAddress	Actualiza la dirección del usuario

Tabla 18.52: Documentación de la clase ContactFillFragment

SetUpConfirmationFragment	
Descripción	Fragmento de confirmación de los datos introducidos
Propiedades	
binding	Vista vinculada
Funciones	
confirm	Envía la confirmación de datos

Tabla 18.53: Documentación de la clase SetUpConfirmationFragment

18.2.1.3. Main

MainActivity	
Descripción	Actividad principal de la aplicación
Propiedades	
binding	Vista vinculada
model	Modelo vinculado
openNotificationsDialog	Despliega el NotificationsDialog (18.56)
openSettingssDialog	Despliega el SettingsDialog (18.57)

Tabla 18.54: Documentación de la clase MainActivity

MainViewModel	
Descripción	Modelo de la vista MainActivity
Propiedades	
cared	Contenedor de la información del paciente vinculado
notiManager	Gestor de notificaciones
notiRepo	Repositorio para las operaciones de notificaciones
globalRepo	Repositorio para las operaciones de la habitación global
userRepo	Repositorio para las operaciones de datos del usuario
Funciones	
bond	Envía una petición para crear un vínculo con otro usuario
requestNotifications	Recupera las notificaciones no leídas
updateUser	Actualiza la información del usuario
retrieveCared	Recupera la información del paciente vinculado
setCared	Establece la información del paciente vinculado

Tabla 18.55: Documentación de la clase MainViewModel

NotificationsDialog	
Descripción	Diálogo de gestión de las notificaciones
Propiedades	
binding	Vista vinculada
cared	Información del paciente vinculado
notiManager	Gestor de notificaciones
Funciones	
list	Lista las notificaciones no leídas
setAllAsRead	Marca todas las notificaciones como leídas
setAsRead	Marca una notificación como leída

Tabla 18.56: Documentación de la clase NotificationsDialog

SettingsDialog	
Descripción	Diálogo de gestión de las notificaciones
Propiedades	
binding	Vista vinculada
Funciones	
editUser	Abre la opción de edición del usuario
cconfirmUpdate	Confirma la actualización de datos del usuario
removeBond	Elimina el vínculo
signOut	Invoca el cierre de sesión

Tabla 18.57: Documentación de la clase SettingsDialog

NotificationsManager	
Descripción	Gestor de las notificaciones
Propiedades	
notifications	Lista de notificaciones no leídas
numberOfNotifications	Número de notificaciones no leídas
notiRepo	Repositorio para las operaciones de notificaciones
Funciones	
add	Añade una notificación a la lista
addAll	Añade un conjunto de notificaciones a la lista
clear	Vacía la lista de notificaciones
initChannels	Inicia los canales de notificación
load	Carga las notificaciones no leídas
remove	Sustrae una notificación de la lista
removeAll	Sustrae un conjunto de notificaciones de la lista
subscribe	Subscribe el WebSocket a los eventos de notificaciones

Tabla 18.58: Documentación de la clase NotificationsManager

SettingsManager	
Descripción	Gestor de las opciones
Funciones	
updateUser	Actualiza los datos del usuario con los nuevos
removeBond	Elimina el vínculo
closeSession	Cierra la sesión actual

Tabla 18.59: Documentación de la clase SettingsManager

QRScannerActivity	
Descripción	Escáner para la lectura del código QR
Propiedades	
binding	Vista vinculada
permissionGranted	Estado de los permisos para el uso de la cámara
Funciones	
decodeCallback	Maneja el resultado de la descodificación
errorCallback	Gestiona errores en la lectura
initScanner	Inicializa el escáner
requestPermission	Solicita el permiso del usuario para usar la cámara
sendResult	Envía el resultado de la lectura a la actividad invocada

Tabla 18.60: Documentación de la clase QRScannerActivity

18.2.1.4. Bonds

BondsActivity	
Descripción	Actividad para la gestión de los vínculos
Propiedades	
binding	Vista vinculada
model	Modelo de vista
Funciones	
showQr	Muestra un QR de vinculación

Tabla 18.61: Documentación de la clase BondsActivity

BondsViewModel	
Descripción	Modelo de la vista BondsActivity
Propiedades	
bonds	Lista de usuarios vinculados
lastQRRequest	Instante de la última petición de código QR
qrCode	Código QR para mostrar
userRepo	Repositorio para las operaciones de datos del usuario
Funciones	
removeBond	Elimina el vínculo
retrieveBonds	Recupera la información de los usuarios asociados
retrieveQRCode	Solicita un código QR para vinculación

Tabla 18.62: Documentación de la clase BondsViewModel

18.2.1.5. Feed

FeedActivity	
Descripción	Actividad de la aplicación con el chat entre usuarios asociados
Propiedades	
binding	Vista vinculada
model	Modelo de vista
Funciones	
scrollToBottom	Mueve el feed hasta el final
sendMessage	Envía un mensaje
swapTaskMode	Cambia al modo de envío de tareas
updateNotification	Actualiza la notificación en pantalla

Tabla 18.63: Documentación de la clase FeedActivity

FeedViewModel	
Descripción	Modelo de la vista FeedActivity
Propiedades	
allLoaded	Bandera de indicación de que todos los mensajes fueron cargados
currentPage	Última página de mensajes cargada
feedList	Contenedor de los mensajes cargados
feedRepo	Repositorio para las operaciones del feed
notification	Contenedor de la última notificación a mostrar
notiRepo	Repositorio para las operaciones de notificaciones
taskMode	Contenedor del estado del modo de creación de tareas
taskRepo	Repositorio para las operaciones de las tareas
Funciones	
loadMoreMessages	Carga una nueva páginas de mensajes
onChangeTask	Maneja la modificación de una tarea
onDeleteTask	Maneja la eliminación de una tarea
onMessageRemoval	Maneja la eliminación de un mensaje
onNewMessage	Maneja la entrada de nuevos mensajes
onTaskStateUpdate	Maneja el cambio de estado de una tarea
retrieveMessages	Recupera los mensajes
sendTaskMessage	Envía una tarea
sendTextMessage	Envía un mensaje de texto
showNotification	Muestra una notificación

Tabla 18.64: Documentación de la clase FeedViewModel

18.2.1.6. Location

LocationActivity	
Descripción	Actividad con el mapa de la geolocalización
Propiedades	
binding	Vista vinculada
map	Mapa de Google
locationProvider	Cliente proveedor de la localización del usuario
locationCallback	Llamada para actualizaciones de la localización
permissionGranted	Estado de los permisos de localización
modelo	Modelo de vista
Funciones	
addMarker	Añade un nuevo marcador al mapa
getDeviceLocation	Recupera la localización del usuario
onMapReady	Gestiona la carga del mapa
removeMarker	Elimina un marcador
requestPermission	Solicita el permiso del usuario para usar la geolocalización
updateLocation	Actualiza un marcador

Tabla 18.65: Documentación de la clase LocationActivity

LocationViewModel	
Descripción	Modelo de la vista LocationActivity
Propiedades	
markers	Gestor de marcadores
leavingUser	Contenedor de un marcador a eliminar
locationRepo	Repositorio para operaciones relacionadas con la localización
newUserMarker	Contenedor de un nuevo marcador
Funciones	
onExternalUpdate	Gestiona una actualización de ubicación de terceros
storeMarker	Almacena un nuevo marcador
updateLocation	Envía la localización al resto de usuarios

Tabla 18.66: Documentación de la clase LocationViewModel

MarkerManager	
Descripción	Gestiona los marcadores de usuario a mostrar en el mapa
Propiedades	
current	Mapa actual de usuarios y marcadores a mostrar
Funciones	
add	Añade un nuevo marcador
exists	Comprueba si existe un marcador
remove	Elimina un marcador
update	Actualiza un marcador

Tabla 18.67: Documentación de la clase MarkerManager

18.2.1.7. Tasks

TasksActivity	
Descripción	Actividad para la gestión de tareas
Propiedades	
binding	Vista vinculada
model	Modelo de vista
Funciones	
openCreateTask	Despliega CreateTaskDialog (18.70)

Tabla 18.68: Documentación de la clase TasksActivity

TasksViewModel	
Descripción	Modelo de la vista TasksActivity
Propiedades	
list	Lista de tareas relevantes del usuario
taskRepo	Repositorio para las operaciones con las tareas
Funciones	
createTask	Envía una nueva tarea para crear
deleteTask	Envía una petición para eliminar una tarea
setTaskDone	Cambia el estado de una tarea
retrieveTasks	Recupera la lista de tareas relevantes del usuario

Tabla 18.69: Documentación de la clase TasksViewModel

CreateTaskDialog	
Descripción	Actividad para la gestión de tareas
Propiedades	
binding	Vista vinculada
Funciones	
confirm	Confirma la tarea creada

Tabla 18.70: Documentación de la clase CreateTaskDialog

18.2.2. Repositories

RepositoryContext	
Descripción	Inyector de repositorios para los modelos de vista
Propiedades	
feedRepository	Proporciona un FeedRepository (18.74)
globalRoomRepository	Proporciona un GlobalRoomRepository (18.77)
notificationRepository	Proporciona un NotificationRepository (18.83)
locationRepository	Proporciona un LocationRepository (18.80)
sessionRepository	Proporciona un SessionRepository (18.85)
taskRepository	Proporciona un TaskRepository (18.87)
userRepository	Proporciona un UserRepository (18.89)

Tabla 18.71: Documentación de la entidad RepositoryContext

SocketRoomRepository	
Descripción	Interfaz de repositorios con lógica WebSocket
Funciones	
join	Gestiona la unión a una sala
leave	Gestione el abandono de una sala

Tabla 18.72: Documentación de la interfaz SocketRoomRepository

BaseSocketRepository	
Descripción	Implementación base para los repositorios con WebSockets
Propiedades	
gson	Conversor entre objetos JSON y instancias del código
Funciones	
fromJson	Instancia un objeto generado a partir de un JSON
toJson	Transforma un objeto en un JSON

Tabla 18.73: Documentación de la clase BaseSocketRepository

18.2.2.1. FeedRepository

FeedRepository	
Descripción	Interfaz para la gestión de las operaciones relacionadas con el feed
Funciones	
getMessages	Recupera la lista de mensajes
onMessageDeleted	Suscribe una acción a la eliminación de un mensaje
onNewMessage	Suscribe una acción a la entrada de un nuevo mensaje
onUserJoining	Suscribe una acción a la conexión de un usuario a la sala del feed
onUserLeaving	Suscribe una acción a la desconexión de un usuario a la sala del feed
send	Envía un mensaje

Tabla 18.74: Documentación de la interfaz FeedRepository

FeedRepositoryImpl	
Descripción	Implementación de la interfaz FeedRepository

Tabla 18.75: Documentación de la implementación de FeedRepository

FeedEvent	
Descripción	Eventos de la sala del feed
Valores	
DELETE	Eliminación de un mensaje
LEAVE	Abandono de la sala
NEW	Nuevo usuario en la sala
JOIN	Unión a la sala
SEND	Envío de un nuevo mensaje

Tabla 18.76: Documentación del enumerado FeedEvent de la aplicación

18.2.2.2. GlobalRoomRepository

GlobalRoomRepository	
Descripción	Interfaz para la gestión de la conexión con la sala global

Tabla 18.77: Documentación de la interfaz GlobalRoomRepository

GlobalRoomRepositoryImpl	
Descripción	Implementación de la interfaz GlobalRoomRepository

Tabla 18.78: Documentación de la implementación de GlobalRoomRepository

GlobalEvent	
Descripción	Eventos de la sala global
Valores	
CONNECT	Conexión del WebSocket
SUBSCRIBE	Unión a la sala
SUBSCRIPTION	Nuevo usuario en la sala

Tabla 18.79: Documentación del enumerado GlobalEvent de la aplicación

18.2.2.3. LocationRepository

LocationRepository	
Descripción	Interfaz para operaciones relacionadas con la geolocalización
Funciones	
onExternalUpdate	Suscribe una acción a la llegada de actualizaciones
onUserLeaving	Suscribe una acción a la desconexión de un usuario de la sala
update	Actualiza la localización del usuario

Tabla 18.80: Documentación de la interfaz LocationRepository

LocationRepositoryImpl	
Descripción	Implementación de la interfaz LocationRepository

Tabla 18.81: Documentación de la implementación de LocationRepository

LocationEvent	
Descripción	Eventos de la sala Location
Valores	
SHARE	Unión a la sala
STOP	Abandono de la sala
UPDATE	Envío de una nueva ubicación

Tabla 18.82: Documentación del enumerado LocationEvent de la aplicación

18.2.2.4. NotificationRepository

NotificationRepository	
Descripción	Interfaz para operaciones relacionadas con las notificaciones
Funciones	
getNotifications	Recupera la lista de notificaciones
onNotification	Suscribe una acción a la llegada de una nueva notificación
setAllAsRead	Marca todas las notificaciones como leídas
setAsRead	Marca una notificación como leída

Tabla 18.83: Documentación de la interfaz NotificationRepository

NotificationRepositoryImpl	
Descripción	Implementación de la interfaz NotificationRepository

Tabla 18.84: Documentación de la implementación de NotificationRepository

18.2.2.5. SessionRepository

SessionRepository	
Descripción	Interfaz para operaciones relacionadas con la sesión
Funciones	
signIn	Inicia la sesión del usuario
signOut	Cierra la sesión del usuario
refresh	Refresca la sesión del usuario

Tabla 18.85: Documentación de la interfaz SessionRepository

SessionRepositoryImpl	
Descripción	Implementación de la interfaz SessionRepository

Tabla 18.86: Documentación de la implementación de SessionRepository

18.2.2.6. TasksRepository

TasksRepository	
Descripción	Interfaz para operaciones relacionadas con las tareas
Funciones	
delete	Elimina una tarea
getTasks	Recupera las tareas
save	Guarda una tarea
update	Actualiza una tarea

Tabla 18.87: Documentación de la interfaz TasksRepository

TasksRepositoryImpl	
Descripción	Implementación de la interfaz TasksRepository

Tabla 18.88: Documentación de la implementación de TasksRepository

18.2.2.7. UserRepository

UserRepository	
Descripción	Interfaz para operaciones relacionadas con los usuarios
Funciones	
getBonds	Recupera los vínculos del usuario
getCared	Recupera los datos del Paciente vinculado
requestBondingCode	Solicita un token de vinculación
sendBondingCode	Elimina un token de vinculación
unbond	Elimina el vínculo con el usuario
update	Actualiza los datos del usuario

Tabla 18.89: Documentación de la interfaz UserRepository

UserRepositoryImpl	
Descripción	Implementación de la interfaz UserRepository

Tabla 18.90: Documentación de la implementación de UserRepository

18.2.3. API

18.2.3.1. ApiFactory

ApiFactory	
Descripción	Factoría de servicios de la API
Funciones	
getAuthService	Crea y regresa un AuthService (18.94)
getFeedService	Crea y regresa un FeedService (18.95)
getTaskService	Crea y regresa un TaskService (18.96)
getUserService	Crea y regresa un UserService (18.97)

Tabla 18.91: Documentación de la entidad ApiFactory

18.2.3.2. SocketManager

SocketManager	
Descripción	Clase a cargo de la gestión del socket
Funciones	
start	Conecta al usuario a la API WebSocket

Tabla 18.92: Documentación de la entidad SocketManager

ApiErrorException	
Descripción	Excepción en comunicaciones con la API

Tabla 18.93: Documentación de la clase ApiErrorException

18.2.3.3. AuthService

AuthService	
Descripción	Interfaz de comunicaciones con el endpoint /auth
Funciones	
deleteSession	Realiza una petición a DELETE /auth/session
getRefresh	Realiza una petición a GET /auth/refresh
getSession	Realiza una petición a GET /auth/session

Tabla 18.94: Documentación de la interfaz AuthService

18.2.3.4. FeedService

FeedService	
Descripción	Interfaz de comunicaciones con el endpoint /feed
Funciones	
getMessages	Realiza una petición a GET /feed/message
getNotifications	Realiza una petición a GET /feed/notification
postNotificationRead	Realiza una petición a POST /feed/notification/:id/read
postNotificationsRead	Realiza una petición a POST /feed/notification/read

Tabla 18.95: Documentación de la interfaz FeedService

18.2.3.5. TaskService

TaskService	
Descripción	Interfaz de comunicaciones con el endpoint /task
Funciones	
deleteTask	Realiza una petición a DELETE /task
deleteTaskDone	Realiza una petición a DELETE /task/done
getTasks	Realiza una petición a GET /task
postTask	Realiza una petición a POST /task
postTaskDone	Realiza una petición a POST /task/done

Tabla 18.96: Documentación de la interfaz TaskService

18.2.3.6. UserService

UserService	
Descripción	Interfaz de comunicaciones con el endpoint /user
Funciones	
deleteBonds	Realiza una petición a DELETE /user/bond
getBondCode	Realiza una petición a GET /user/bond
getBonds	Realiza una petición a GET /user/bond
getCared	Realiza una petición a GET /user/bond/cared
patchUser	Realiza una petición a PATCH /user
postNewBond	Realiza una petición a POST /user/bond

Tabla 18.97: Documentación de la interfaz UserService

18.2.4. VO

18.2.4.1. Message

Message	
Descripción	Objeto representante de un mensaje
Propiedades	
id	ID
content	Contenido del mensaje
submitter	Usuario que envió el mensaje
time	Representación textual del *timestamp*
timestamp	Instante de creación del mensaje
Funciones	
toDto	Genera un MessageDto representativo del mensaje

Tabla 18.98: Documentación de la clase Message de la aplicación

TaskMessage	
Descripción	Adaptación de una tarea como mensaje
Propiedades	
task	Tarea decorada

Tabla 18.99: Documentación de la clase TaskMessage de la aplicación

TextMessage	
Descripción	Objeto representante de un mensaje de texto

Tabla 18.100: Documentación de la clase TextMessage de la aplicación

MessageType	
Descripción	Tipos de mensajes
Valores	
TASK	Tarea
TEXT	Mensaje de texto

Tabla 18.101: Documentación del enumerado MessageType de la aplicación

18.2.4.2. Notification

Notification	
Descripción	Objeto representante de una notificación
Propiedades	
id	ID
action	Acción notificada
instigator	Usuario que realizó la acción
timestamp	Instante de creación de la notificación
tags	Etiquetas de la notificación
Funciones	
build	Genera una notificación de Android con la información de la notificación
print	Genera un mensaje representativo de la notificación

Tabla 18.102: Documentación de la clase Notification de la aplicación

Action	
Descripción	Acciones notificables
Valores	
BOND CREATED	Creación de un vínculo
BOND DELETED	Eliminación de un vínculo
TASK CREATED	Creación de una tarea
TASK DELETED	Eliminación de una tarea
TASK DONE	Actualización de una tarea a hecha
TASK UNDONE	Actualización de una tarea a no hecha
LOCATION SHARE	Ubicación empezando a ser compartida
LOCATION STOP	Ubicación dejando de ser compartida
Funciones	
getContentTitle	Retorna el título de la notificación
getDestiny	Retorna la clase destino de la notificación si la tiene
getPath	Retorna la ruta del evento de la notificación
print	Genera un mensaje representativo de la acción

Tabla 18.103: Documentación del enumerado Action de la aplicación

Channel	
Descripción	Canales de notificación
Valores	
BOND	Canal de notificaciones de vínculos
LOCATION	Canal de notificaciones de geolocalización
TASK	Canal de notificaciones de tareas
Funciones	
getId	Retorna la ID del canal
build	Genera un canal de notificaciones de Android con la información del canal

Tabla 18.104: Documentación del enumerado Channel de la aplicación

18.2.4.3. Session

Session	
Descripción	Objeto representante de una sesión
Propiedades	
auth	Token de autenticación
refresh	Token de refresco
expiration	Instante de expiración de la sesión

Tabla 18.105: Documentación de la clase Session de la aplicación

18.2.4.4. Task

Task	
Descripción	Objeto representante de una tarea
Propiedades	
id	ID
submitter	Usuario que envió la tarea
title	Título
description	Descripción
done	Estado: hecha o no hecha
timestamp	Instante de creación del mensaje
lastUpdate	Instante de última actualización del mensaje
Funciones	
swapState	Cambia el estado de *done*

Tabla 18.106: Documentación de la clase Task de la aplicación

18.2.4.5. User

User	
Descripción	Objeto representante de un usuario
Propiedades	
id	ID
role	Rol
displayName	Nombre visible
mainPhoneNumber	Número de teléfono principal
altPhoneNumber	Número alternativo de teléfono
address	Dirección postal
email	Dirección electrónica
Funciones	
toMin	Genera un UserMinDto de la entidad

Tabla 18.107: Documentación de la clase User de la aplicación

Role	
Descripción	Roles de los usuarios
Valores	
BLANK	Sin rol
KEEPER	Cuidadores
PATIENT	Pacientes

Tabla 18.108: Documentación del enumerado Role de la aplicación

Address	
Descripción	Objeto representante de una dirección postal
Propiedades	
firstLine	Primera línea
secondLine	Segunda línea
locality	Localidad
region	Región

Tabla 18.109: Documentación de la clase Address de la aplicación

UserMarker	
Descripción	Objeto representante de un marcador de mapa de un usuario
Propiedades	
id	ID del usuario
displayName	Nombre del usuario
position	Coordenadas del marcador
Funciones	
build	Construye un marcador de Google Maps de la entidad

Tabla 18.110: Documentación de la clase UserMarker de la aplicación

19. Modelo de datos

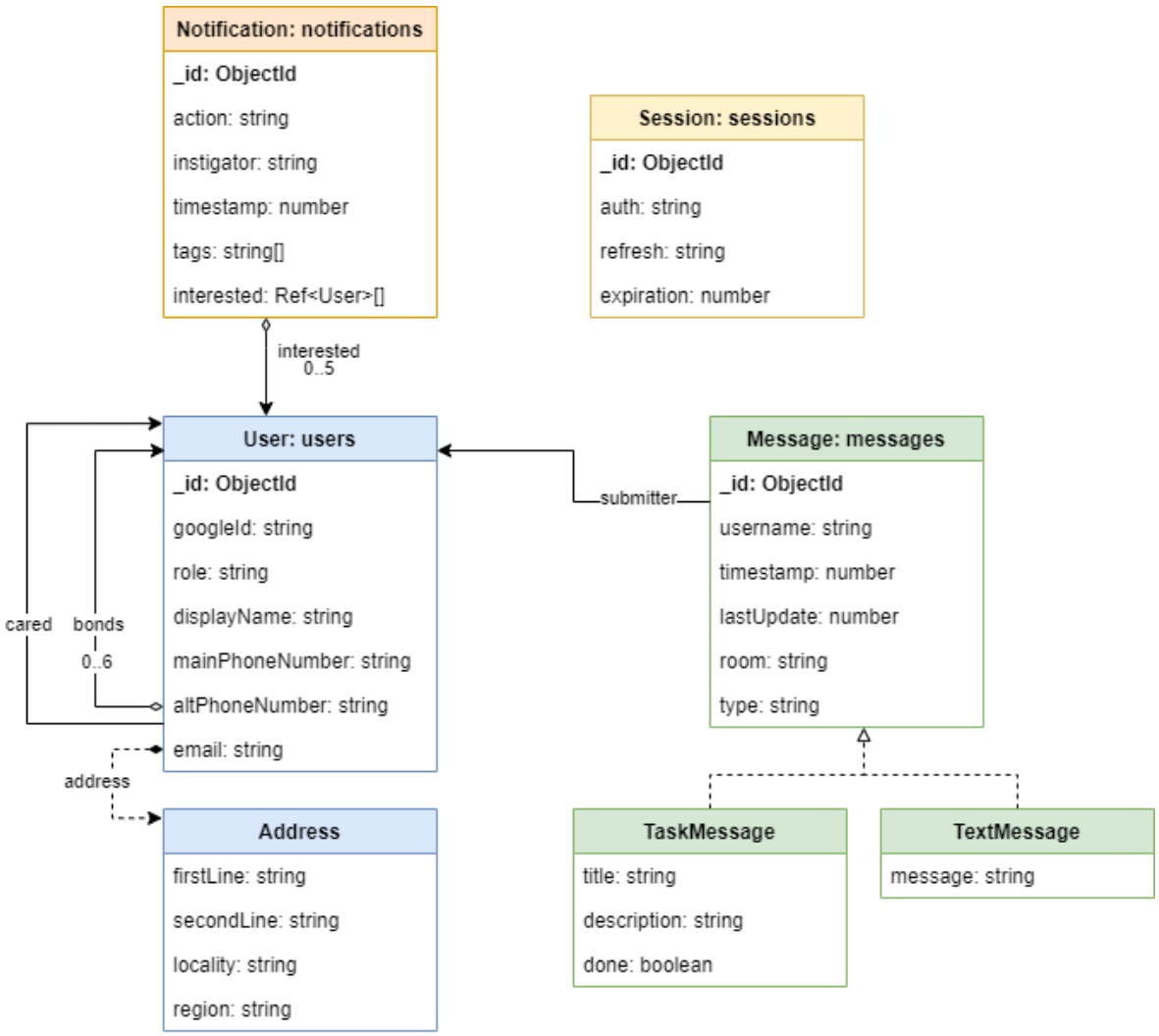


Figura 19.1: Diagrama del modelo de datos

19.1. Mensaje

Dentro de los mensaje se aúnan dos entidades de la aplicación: los **mensajes de texto** que se envían a través del Feed y las **tareas**, que también pueden ser enviadas a través del mismo. Todos estos mensajes se almacenan en la colección *messages*. Aunque se puede poner en duda la elección de tratar las tareas como mensajes, esta decisión de diseño parte de un aprovechamiento de la flexibilidad de las bases de datos documentales como la que se emplea en el sistema.

Las consultas para obtener las entidades del Feed en caso de utilizar colecciones

diferentes para los mensajes de texto y las tareas habría implicado la necesidad de realizar *joins*, que aunque es una operación soportada por MongoDB en sus últimas versiones, es un claro indicativo de un modelo de datos mal diseñado para bases de datos no relacionales como la nuestra. Es por eso que tanto tareas como mensajes son **incluidas en la misma colección** y tratadas ambas como mensajes al nivel de lógica de la API.

19.1.1. BaseMessage

De cara a favorecer una compatibilidad lo mayor posible entre los dos tipos de entidades que se guardarán en la colección se define una primera entidad con campos comunes a ambas y que serán necesarios en las consultas a la colección. Tanto los mensajes de texto como las tareas contarán con todas las propiedades de esta clase base. Véase Apartado 37.1.1.

- **id**: *string*. Obligatorio. Identificador único del mensaje. Generada automáticamente por MongoDB al persistir la entidad por primera vez.
- **submitter**: *User*. Obligatorio. Referencia al usuario creador del mensaje.
- **username**: *string*. Obligatorio. Nombre público del autor en el momento de la creación. Cacheado en este documento para evitar la necesidad de hacer *joins* con la colección de usuarios para acceder a este dato y optimizar las consultas con este tipo de base de datos.
- **timestamp**: *number*. Obligatorio. Instante de creación del mensaje.
- **lastUpdate**: *number*. Obligatorio. Instante de la última actualización del mensaje, si no se ha actualizado nunca desde su creación su valor será el mismo que el de **timestamp**.
- **room**: *string*. Obligatorio. Identificador de la sala de mensajería a través de la que fue enviado el mensaje.
- **type**: *string*. Obligatorio. Tipo de mensaje, puede ser: Text o Task.

19.1.2. TaskMessage

Representa las tareas creadas por los usuarios, tanto en la vista de Tareas como a través del Feed. Hereda de BaseMessage y su tipo es **Task**. Véase Apartado 37.1.2. Además de las propiedades de BaseMessage cuenta con las siguientes:

- **title**: *string*. Obligatorio. Título de la tarea.
- **description**: *string*. Opcional. Descripción de la tarea.
- **done**: *boolean*. Obligatorio. Estado de la tarea, a `true` es hecha y `false` es no hecha.

19.1.3. TextMessage

Representa los mensajes de texto simples enviados a través del Feed. Hereda de BaseMessage y su tipo es **Text**. Véase Apartado 37.1.3. Además de las propiedades de BaseMessage tiene la siguiente propiedad:

- **message**: *string*. Obligatorio. Cuerpo del mensaje.

19.2. Notificación

Las notificaciones reflejan una **acción de interés** para una serie de usuarios. Dicha acción se almacena en la entidad y es una de una enumeración conocida. Como algunas de estas acciones pueden servirse más de una vez, para aportar más contexto cuentan un campo para añadir etiquetas que podrán ser almacenadas y leídas según fuese necesario con cada tipo de acción.

Por otro lado, las notificaciones son **entidades compartidas** por una serie de usuarios. Los usuarios a notificar son almacenados en la entidad para su recuperación por parte de estos, una vez que un usuario marque como leída la notificación su ID será eliminada de esa lista de usuarios, de forma que la entidad pase a ir dirigida únicamente a aquellos usuarios que aún no la leyesen. Cuando una notificación ha sido leída por todos sus usuarios asociados será eliminada de la base de datos.

Las notificaciones se almacenan en la colección *notifications*. El esquema definitivo se puede ver en el Apartado 37.1.4. Sus propiedades son:

- **id**: *string*. Obligatorio. Identificador único de la notificación. Generada automáticamente por MongoDB al persistir la entidad por primera vez.
- **action**: *Action*. Obligatorio. Acción notificada, puede ser una de las listadas en Cuadro 14.54.

- **instigator**: *string*. Obligatorio. Nombre del usuario que provocó la acción a notificar.
- **timestamp**: *number*. Obligatorio. Instante en el que ocurrió la acción.
- **tags**: *string array*. Lista de cadenas de texto con información adicional de la notificación. Por defecto existirá como lista vacía.
- **interested**: *User array*. Lista de referencias a los usuarios que son notificados y que aún no han leído la notificación.

19.3. Sesión

De cara a gestionar las sesiones y validar e invalidar los tokens de forma correcta, se ha creado también una colección para el almacenamiento de estas llamada *sessions* (véase Apartado 37.1.5). Las propiedades almacenadas son:

- **id**: *string*. Obligatorio. Identificador único de la sesión. Generada automáticamente por MongoDB al persistir la entidad por primera vez.
- **auth**: *string*. Obligatorio. Es el token activo de la sesión.
- **refresh**: *string*. Obligatorio y único. Es el token que permite refrescar la sesión de usuario sin necesidad de volver a autenticarse.
- **expiration**: *number*. Obligatorio. Instante de expiración de la sesión, equivalente al tiempo de validez del token de autenticación. Almacenado para limpiar sesiones caducadas de la base de datos.

19.4. Usuario

Los usuarios son las entidades clave de la aplicación. Su esquema en la base de datos (véase Apartado 37.1.6) contiene toda la información de cada usuario, así como sus relaciones con otros usuarios. No existe una entidad diferente para Pacientes y Cuidadores. Las diferencias se gestionarán en el servicio, por lo que la entidad contiene tanto la lista de vínculos de los Pacientes como el campo para almacenar el vínculo de un Cuidador. Tiene las siguientes propiedades:

- **id**: *string*. Obligatorio. Identificador único del usuario. Generada automáticamente por MongoDB al persistir la entidad por primera vez.

- **googleId**: *string*. Obligatorio y único. Token de usuario de la cuenta de Google del usuario. Almacenada para autenticar al usuario con su cuenta de Google, no es recuperada de la base de datos al no tener ningún otro uso.
- **role**: *Role*. Obligatorio. Rol del usuario, puede ser cualquiera de los listados en Cuadro 14.57.
- **displayName**: *number*. Opcional. Nombre visible del usuario.
- **mainPhoneNumber**: *string*. Opcional. Número de teléfono principal del usuario.
- **altPhoneNumber**: *string*. Opcional. Número de teléfono alternativo del usuario.
- **address**: *Address*. Opcional. Dirección postal del usuario (véase Apartado 19.4.1).
- **email**: *string*. Opcional. Dirección electrónica del usuario.
- **bonds**: *User array*. Opcional. Lista de referencias a los usuarios vinculados con un Paciente.
- **cared**: *User*. Opcional. Referencia al Paciente vinculado de un Cuidador.

19.4.1. Address

Representación de las direcciones postales como objetos del sistema, compuesta por las siguientes propiedades:

- **firstLine**: *string*. Opcional. Primera línea de la dirección para la información principal.
- **secondLine**: *string*. Opcional. Segunda línea de la dirección para la información principal.
- **locality**: *string*. Opcional. Localidad.
- **region**: *string*. Opcional. Región.

20. Interfaces de comunicación

20.1. API REST

A continuación se describen los endpoints de la API REST del sistema y toda la información de su funcionamiento. El único endpoint público de la interfaz es el de inicio de sesión (20.1), el token de autenticación provisto en las llamadas exitosas a dicho endpoint será el que se deba proveer en el encabezado de autenticación siguiendo el **RFC 6750**[13] para poder acceder a las funcionalidades del resto de funciones de la interfaz.

Cualquier petición a los endpoints privados puede provocar los siguientes errores de autenticación aparte de los inherentes de cada función:

- 400 BAD REQUEST si falta el Authorization header o está mal formateado.
- 401 UNAUTHORIZED si el token es inválido o está expirado.

20.1.1. Autenticación

Inicio de sesión	
Descripción	Inicia la sesión del usuario en el sistema con su token de autenticación en Google y responde con los tokens de la sesión y la información del usuario. Si el usuario no existe, lo crea.
Petición	
URL	/auth/session/:token
Método	GET
Parámetros URL	**token**: *string*. Token de sesión obtenido tras la autenticación exitosa en el servicio de *Google Auth*
Respuesta	
Código	200 OK
Content-type	application-json
Cuerpo	**session**: *object*. DTO de la sesión iniciada (37.2.5) **user**: *object*. Contiene toda la información del usuario en un UserDto (37.2.8)
Errores	400 BAD REQUEST si el token proporcionado no pasa la validación de Google.

Tabla 20.1: Documentación del endpoint de inicio de sesión

Cierre de sesión	
Descripción	Cierre la sesión del usuario en el sistema con su token de autenticación.
Petición	
URL	/auth/session/:token
Método	DELETE
Encabezados	**Authorization**: Token de sesión según RFC6750.
Parámetros URL	**token**: *string*. Token de autenticación de la sesión a cerrar
Respuesta	
Código	204 NO CONTENT
Errores	403 FORBIDDEN si el token proporcionado no pertenece al usuario realizando la petición.

Tabla 20.2: Documentación del endpoint de cierre de sesión

Refresco de sesión	
Descripción	Renueva la sesión de usuario y devuelve los nuevos tokens de sesión
Petición	
URL	/auth/refresh/:token
Método	GET
Encabezados	**Authorization**: Token de sesión según RFC6750
Parámetros URL	**token**: *string*. Token de refresco de la sesión a renovar
Respuesta	
Código	200 OK
Content-type	application-json
Cuerpo	**session**: *object*. DTO de la sesión refrescada (37.2.5)
Errores	400 BAD REQUEST si no existe la sesión abierta a la que los tokens están asignados 401 UNAUTHORIZED si los tokens proporcionados en la petición no son válidos 401 UNAUTHORIZED si el token de refresco está expirado 401 UNAUTHORIZED si el token de refresco es inválido.

Tabla 20.3: Documentación del endpoint de refresco de sesión

20.1.2. Feed

Recuperar grupo de mensajes	
Descripción	Devuelve el grupo de mensajes especificado por el usuario. Cada grupo contiene hasta 25 mensajes. Los mensajes son devueltos en orden de más reciente a más antiguo.
Petición	
URL	/feed/messages
Método	GET
Encabezados	**Authorization**: Token de sesión según RFC6750.
Parámetros consulta	**page**: *number*. *Opcional*. Página de mensajes a recuperar, por defecto: 1.
Respuesta	
Código	200 OK
Content-type	application-json
Cuerpo	**messages**: *object array*. Lista de mensajes, los mensajes son de tipo Message (37.2.1).
Errores	400 BAD REQUEST si el usuario no es paciente ni cuidador. 400 BAD REQUEST si lo solicita un cuidador no vinculado.

Tabla 20.4: Documentación del endpoint de recuperación de mensajes

20.1.2.1. Notificaciones

Recuperar notificaciones	
Descripción	Devuelve las notificaciones no leídas del usuario.
Petición	
URL	/feed/notifications
Método	GET
Encabezados	**Authorization**: Token de sesión según RFC6750.
Parámetros consulta	**maxDays**: *number*. *Opcional*. Número máximo de días de antigüedad de las notificaciones devueltas. Por defecto, siete días.
Respuesta	
Código	200 OK
Content-type	application-json
Cuerpo	**notifications**: *object array*. Lista de NotificationDto (37.2.4).

Tabla 20.5: Documentación del endpoint de recuperar notificaciones

Marcar notificación como leída	
Descripción	Marca la notificación especificada como leída por el usuario.
Petición	
URL	/feed/notifications/:id/read
Método	POST
Encabezados	**Authorization**: Token de sesión según RFC6750.
Parámetros URL	**id**: *string*. ID de la notificación.
Respuesta	
Código	204 NO CONTENT

Tabla 20.6: Documentación del endpoint de marca una notificación como leída

Marcar todas las notificaciones como leídas	
Descripción	Marca todas las notificaciones de un usuario como leídas.
Petición	
URL	/auth/notifications/read
Método	POST
Encabezados	**Authorization**: Token de sesión según RFC6750.
Respuesta	
Código	204 NO CONTENT

Tabla 20.7: Documentación del endpoint de marcar todas las notificaciones como leídas

20.1.3. Tareas

Crear tarea	
Descripción	Persiste y completa la tarea enviada.
Petición	
URL	/tasks
Método	POST
Encabezados	**Authorization**: Token de sesión según RFC6750.
Cuerpo	*object*. Tarea a persistir con las propiedades de TaskMinDto (37.2.2).
Respuesta	
Código	201 CREATED
Content-type	application-json
Cuerpo	*object*. DTO de la tarea creada con las propiedades de TaskDto (37.2.3).
Errores	400 BAD REQUEST si el usuario es un Cuidador no vinculado.

Tabla 20.8: Documentación del endpoint de crear una tarea

Recuperar tareas relevantes	
Descripción	Devuelve la lista de tareas relevantes del usuario. Una tarea relevante es aquella incompleta o que ha sido actualizada alguna vez en el margen de días máximo especificado en la petición.
Petición	
URL	/tasks
Método	GET
Encabezados	**Authorization**: Token de sesión según RFC6750.
Parámetros consulta	**maxDays**: *number*. *Opcional*. Número de días máximo desde la última actualización de las tareas completadas para considerarlas relevantes. Por defecto, 3 días.
Respuesta	
Código	200 OK
Content-type	application-json
Cuerpo	**tasks**: *object array*. Lista de tareas recuperadas de tipo TaskDto (37.2.3).
Errores	400 BAD REQUEST si el usuario es un Cuidador no vinculado

Tabla 20.9: Documentación del endpoint de recuperar tareas relevantes

Eliminar tarea	
Descripción	Elimina la tarea especificada.
Petición	
URL	/tasks/:id
Método	DELETE
Encabezados	**Authorization**: Token de sesión según RFC6750.
Parámetros URL	**id**: *string*. Identificador único de la tarea a eliminar.
Respuesta	
Código	204 NO CONTENT
Errores	403 FORBIDDEN si el usuario no tiene permisos para eliminar la tarea

Tabla 20.10: Documentación del endpoint de eliminar una tarea

Marcar tarea como hecha	
Descripción	Marca la tarea especificada como hecha.
Petición	
URL	/tasks/:id/done
Método	POST
Encabezados	**Authorization**: Token de sesión según RFC6750.
Parámetros URL	**id**: *string*. Identificador único de la tarea a actualizar.
Respuesta	
Código	204 NO CONTENT
Errores	403 FORBIDDEN si el usuario no tiene permisos para actualizar la tarea

Tabla 20.11: Documentación del endpoint de marcar una tarea como hecha

Marcar tarea como no hecha	
Descripción	Marca la tarea especificada como no hecha.
Petición	
URL	/tasks/:id/done
Método	DELETE
Encabezados	**Authorization**: Token de sesión según RFC6750.
Parámetros URL	**id**: *string*. Identificador único de la tarea a actualizar.
Respuesta	
Código	204 NO CONTENT
Errores	403 FORBIDDEN si el usuario no tiene permisos para actualizar la tarea

Tabla 20.12: Documentación del endpoint de marcar una tarea como no hecha

20.1.4. Usuarios

Actualizar usuario	
Descripción	Actualiza la información de un usuario.
Petición	
URL	/user/:id
Método	PATCH
Encabezados	**Authorization**: Token de sesión según RFC6750.
Parámetros URL	**id**: *string*. Identificador único del usuario.
Cuerpo	*object*. Propiedades a actualizar del usuario en un DTO de tipo UserPublicDto (37.2.7).
Respuesta	
Código	201 CREATED
Content-type	application-json
Cuerpo	*object*. Información del usuario de tipo UserDto (37.2.8)
Errores	403 FORBIDDEN si el solicitante no es el usuario que intenta actualizar.

Tabla 20.13: Documentación del endpoint de actualización de información de usuarios

Generar token de vinculación	
Descripción	Devuelve un token de vinculación válido del usuario.
Petición	
URL	/user/bonds/token
Método	GET
Encabezados	**Authorization**: Token de sesión según RFC6750.
Respuesta	
Código	200 OK
Content-type	application-json
Cuerpo	**code**: *string*. Token de vinculación.

Tabla 20.14: Documentación del endpoint de generación de tokens de vinculación

Establecer vínculos	
Descripción	Valida el token de vinculación enviado y crea el vínculo entre los dos usuarios si es correcto.
Petición	
URL	/user/bonds/token
Método	POST
Encabezados	**Authorization**: Token de sesión según RFC6750.
Parámetros URL	**code**: *string*. Token de vinculación.
Respuesta	
Código	200 OK
Content-type	application-json
Cuerpo	**message**: *string*. Mensaje de confirmación de la acción.
Errores	400 BAD REQUEST si el token ha expirado. 400 BAD REQUEST si el Paciente ya tiene el máximo de vínculos 400 BAD REQUEST si el Cuidador ya está vinculado 403 FORBIDDEN si el usuario es un Paciente

Tabla 20.15: Documentación del endpoint de establecimiento de vínculos

Eliminar vínculos	
Descripción	Elimina el vínculo con el usuario especificado.
Petición	
URL	/user/bonds/:id
Método	DELETE
Encabezados	**Authorization**: Token de sesión según RFC6750.
Cuerpo	**id**: *string*. ID del usuario a desvincular.
Respuesta	
Código	204 NO CONTENT

Tabla 20.16: Documentación del endpoint de eliminación de vínculos

Recuperar Paciente vinculado	
Descripción	Recupera la información del Paciente vinculado al usuario.
Petición	
URL	/user/:id/cared
Método	GET
Encabezados	**Authorization**: Token de sesión según RFC6750.
Parámetros URL	**id**: *string*. Identificador único del usuario.
Respuesta	
Código	200 OK
Content-type	application-json
Cuerpo	**cared**: *object*. Información del Paciente vinculado en un DTO de tipo UserDto (37.2.8).
Errores	400 BAD REQUEST si el usuario no es un Cuidador. 403 FORBIDDEN si el solicitante no es el usuario del que pretende obtener el Paciente vinculado.

Tabla 20.17: Documentación del endpoint de actualización de información de usuarios

Recuperar información de asociados	
Descripción	Devuelve la información de contacto de los usuarios asociados con el solicitante.
Petición	
URL	/user/bonds
Método	GET
Encabezados	**Authorization**: Token de sesión según RFC6750.
Respuesta	
Código	200 OK
Content-type	application-json
Cuerpo	**bonds**: *object array*. Lista de información de contacto de los asociados del usuario. Enviados en DTOs de tipo UserPublicDto (37.2.7)
Errores	400 BAD REQUEST si el usuario no es Paciente ni Cuidador 403 FORBIDDEN si el solicitante no es el usuario que intenta actualizar.

Tabla 20.18: Documentación del endpoint de recuperación de información de asociados

20.2. API del WebSocket

El otro tipo de comunicación ofrecido por la API es la que se produce a través del WebSocket. Esta comunicación se produce de forma **no bidireccional** por medio de envío de mensajes de texto con una clave que indica el evento que invoca. Las peticiones de un cliente (mensajes de cliente o entrada) serán enviados al servidor y no se quedarán esperando una respuesta, y si esta se produce será por medio de otro evento enviado por el servidor (mensaje de servidor o salida). Si una solicitud de un cliente tiene algún error, el mensaje será ignorado. Algunos mensaje de cliente pueden ser remitidos por el servidor con la misma clave de evento, estos mensajes constituyen un nuevo tipo de comunicación llamado **retransmisión**.

Todo cliente que se suscriba al socket debe unirse a una habitación global compartida entre todos los usuarios asociados. Es a través de esta sala que se gestiona la unión al resto de salas. La desconexión de esta sala supondría la desconexión del resto de salas. La sala global cuenta con dos eventos ilustrados en Cuadro 20.19 y Cuadro 20.20.

GLOBAL_SUBSCRIBE	
Descripción	Suscribe a un cliente a una sala global.
Clave	global:subscribe
Tipo	Cliente
Mensaje de entrada	*string*. ID del usuario solicitando unirse a la sala.
Emisión	GLOBAL_SUBSCRIPTION (20.20) a los usuarios suscritos a la sala
Errores	Si el cliente no es Paciente ni Cuidador Si el cliente es un Cuidador no vinculado

Tabla 20.19: Documentación del evento Subscribe de la sala Global

GLOBAL_SUBSCRIPTION	
Descripción	Notificación a los suscriptores de una sala acerca de la suscripción de un nuevo cliente.
Clave	global:subscription
Tipo	Servidor
Emisión	**user**: *string*. ID del nuevo suscriptor. **roomId**: *string*. ID de la sala.

Tabla 20.20: Documentación del evento Subscription de la sala Global

20.2.1. Feed

FEED_JOIN	
Descripción	Suscribe a un cliente a una sala de mensajería.
Clave	feed:join
Tipo	Retransmisión
Mensaje de entrada	*object*. DTO con la información del usuario, del tipo UserMinDto (Apartado 37.2.6).
Emisión	FEED_JOIN a los usuarios suscritos a la sala.
Errores	Si el cliente no está conectado a una sala global .

Tabla 20.21: Documentación del evento Join de la sala Feed

FEED_LEAVE	
Descripción	Da de baja a un cliente de su sala de mensajería.
Clave	feed:leave
Tipo	Retransmisión
Mensaje de entrada	*object*. DTO con la información del usuario, del tipo UserMinDto (Apartado 37.2.6).
Emisión	FEED_LEAVE a los usuarios suscritos a la sala
Errores	Si el cliente no está conectado a una sala de mensajería.

Tabla 20.22: Documentación del evento Leave de la sala Feed

FEED_SEND	
Descripción	Envía un mensaje a través de la sala.
Clave	feed:send
Tipo	Cliente
Mensaje de entrada	*object*. DTO del mensaje, del tipo Message (Apartado 37.2.1).
Emisión	FEED_NEW a los usuarios suscritos a la sala
Errores	Si el cliente no está conectado a una sala de mensajería

Tabla 20.23: Documentación del evento Send de la sala Feed

FEED_NEW	
Descripción	Nuevo mensaje enviado a través de la sala.
Clave	feed:new
Tipo	Servidor
Emisión	*object*. DTO del mensaje, del tipo Message (Apartado 37.2.1).

Tabla 20.24: Documentación del evento New de la sala Feed

FEED_DELETE	
Descripción	Elimina un mensaje eviado a través de la sala.
Clave	feed:delete
Tipo	Servidor
Emisión	*object*. DTO del mensaje, del tipo Message (Apartado 37.2.1).

Tabla 20.25: Documentación del evento Delete de la sala Feed

20.2.2. Localización

LOCATION_SHARE	
Descripción	Suscribe a un cliente a una sala de geolocalización.
Clave	location:share
Tipo	Cliente
Mensaje de entrada	*object*. DTO con la información del usuario, del tipo UserMinDto (Apartado 37.2.6).
Emisión	NOTIFY_LOCATION_SHARING_START (20.34) a los usuarios suscritos a la sala
Errores	Si el cliente no está conectado a una sala global

Tabla 20.26: Documentación del evento Share de la sala Location

LOCATION_STOP	
Descripción	Da de baja a un cliente de su sala de localización.
Clave	location:stop
Tipo	Retransmisión
Mensaje de entrada	*object*. DTO con la información del usuario, del tipo UserMinDto (Apartado 37.2.6).
Emisión	LOCATION_STOP a los usuarios suscritos a la sala NOTIFY_LOCATION_SHARING_STOP (20.35) a los usuarios asociados conectados
Errores	Si el cliente no está conectado a una sala de localización

Tabla 20.27: Documentación del evento Stop de la sala Location

LOCATION_UPDATE	
Descripción	Comparte la ubicación de un usuario.
Clave	location:update
Tipo	Retransmisión
Mensaje de entrada	**id**: *string*. ID del usuario. **displayName**: *string*. Nombre público del usuario. **position**: *object*. Compuesto por **latitute**:*number* y **longitude**:*number*. La posición a compartir.
Emisión	LOCATION_UPDATE al resto de usuarios suscritos a la sala
Errores	Si el cliente no está conectado a una sala de localización

Tabla 20.28: Documentación del evento Update de la sala Location

20.2.3. Notificación

Los eventos de notificaciones no cuentan con su propia sala sino que son emitidas a través de la sala global y son todos de tipo servidor pues son mensajes generados por el usuario a raíz de acciones transversales que son comunicadas a los usuarios relevantes.

NOTIFY_BOND_CREATED	
Descripción	Notifica la creación de un vínculo.
Clave	notify:bond_created
Tipo	Servidor
Emisión	*object*. DTO de la notificación, del tipo NotificationDto (Apartado 37.2.4).

Tabla 20.29: Documentación del evento Notify Bond Created de la sala Global

NOTIFY_BOND_DELETED	
Descripción	Notifica la eliminación de un vínculo.
Clave	notify:bond_deleted
Tipo	Servidor
Emisión	*object*. DTO de la notificación, del tipo NotificationDto (Apartado 37.2.4).

Tabla 20.30: Documentación del evento Notify Bond Deleted de la sala Global

NOTIFY_TASK_DELETED	
Descripción	Notifica la eliminación de una tarea.
Clave	notify:task_deleted
Tipo	Servidor
Emisión	*object*. DTO de la notificación, del tipo NotificationDto (Apartado 37.2.4).

Tabla 20.31: Documentación del evento Notify Task Deleted de la sala Global

NOTIFY_TASK_DONE	
Descripción	Notifica el cambio de una tarea a hecha.
Clave	notify:task_done
Tipo	Servidor
Emisión	*object*. DTO de la notificación, del tipo NotificationDto (Apartado 37.2.4).

Tabla 20.32: Documentación del evento Notify Task Done de la sala Global

NOTIFY_TASK_UNDONE	
Descripción	Notifica el cambio de una tarea a no hecha.
Clave	notify:task_undone
Tipo	Servidor
Emisión	*object*. DTO de la notificación, del tipo NotificationDto (Apartado 37.2.4).

Tabla 20.33: Documentación del evento Notify Task Undone de la sala Global

NOTIFY_LOCATION_SHARING_START	
Descripción	Notifica que un usuario asociado ha comenzado a compartir su ubicación.
Clave	notify:location_sharing_start
Tipo	Servidor
Emisión	*object*. DTO de la notificación, del tipo NotificationDto (Apartado 37.2.4).

Tabla 20.34: Documentación del evento Notify Location Sharing Start de la sala Global

NOTIFY_LOCATION_SHARING_STOP	
Descripción	Notifica que un usuario asociado ha dejado de compartir su ubicación.
Clave	notify:location_sharing_stop
Tipo	Servidor
Emisión	*object*. DTO de la notificación, del tipo NotificationDto (Apartado 37.2.4).

Tabla 20.35: Documentación del evento Notify Location Sharing Stop de la sala Global

21. Interacción y estados

21.1. Inicio de sesión de usuarios existentes y nuevos

El inicio de sesión se comienza desde la pantalla de lanzamiento de la aplicación utilizando el botón destinado para tal fin. El proceso varía si el usuario es un usuario nuevo o si ya está registrado en el sistema y cuenta con un perfil completo. El flujo de actividades para el inicio de sesión global con ambos casos es el ilustrado en la Figura 21.1. Estos dos tipos de inicio de sesión se corresponden con los casos de uso **CU1. Registro** y **CU2. Iniciar sesión**.

El **inicio de sesión** estándar puede verse en detalle en el diagrama de secuencia de la Figura 21.2. El proceso comienza con el usuario solicitando el inicio de sesión en la aplicación móvil, tras lo que se le solicitará una cuenta de Google para realizarlo. Una vez que se verifique la cuenta con la librería de autenticación de Google se enviará el token de la sesión de Google a la API por el endpoint correspondiente. La API usará dicho token para obtener la ID única de Google del usuario, con la que consultará a la base de datos para recuperar el perfil de usuario almacenado. Con dicha información generará una nueva sesión que persistirá en la base de datos antes de enviarla como respuesta a la aplicación junto a los datos del usuario. Tras recibir la respuesta, la aplicación almacenará esa sesión y el usuario, y avanzará hasta la pantalla principal con la **sesión iniciada**.

El **registro de nuevos usuarios** comienza igual que el inicio de sesión hasta la obtención de la ID de la cuenta de Google del usuario por parte de la API. Tras esto, la consulta a la base de datos para recuperar el perfil no retornará ningún usuario, indicando que **dicho usuario no existe**. En ese momento, la API creará un nuevo usuario con la ID de Google que persistirá antes de crear una sesión que enviar junto a dicho usuario incompleto. Al recibir un *User* incompleto la aplicación dirigirá al usuario a la **SetUpActivity**. En dicha pantalla el usuario introducirá sus datos restantes, completando su perfil. Una vez confirme los datos enviados, la aplicación realizará la petición de parcheo a la API, que persistirá dichos datos y devolverá el nuevo *User* actualizado. El final de esta secuencia será como el del **inicio de sesión**, con el almacenamiento de dicho *User* y el avance hasta la pantalla principal con la sesión iniciada en el nuevo perfil.

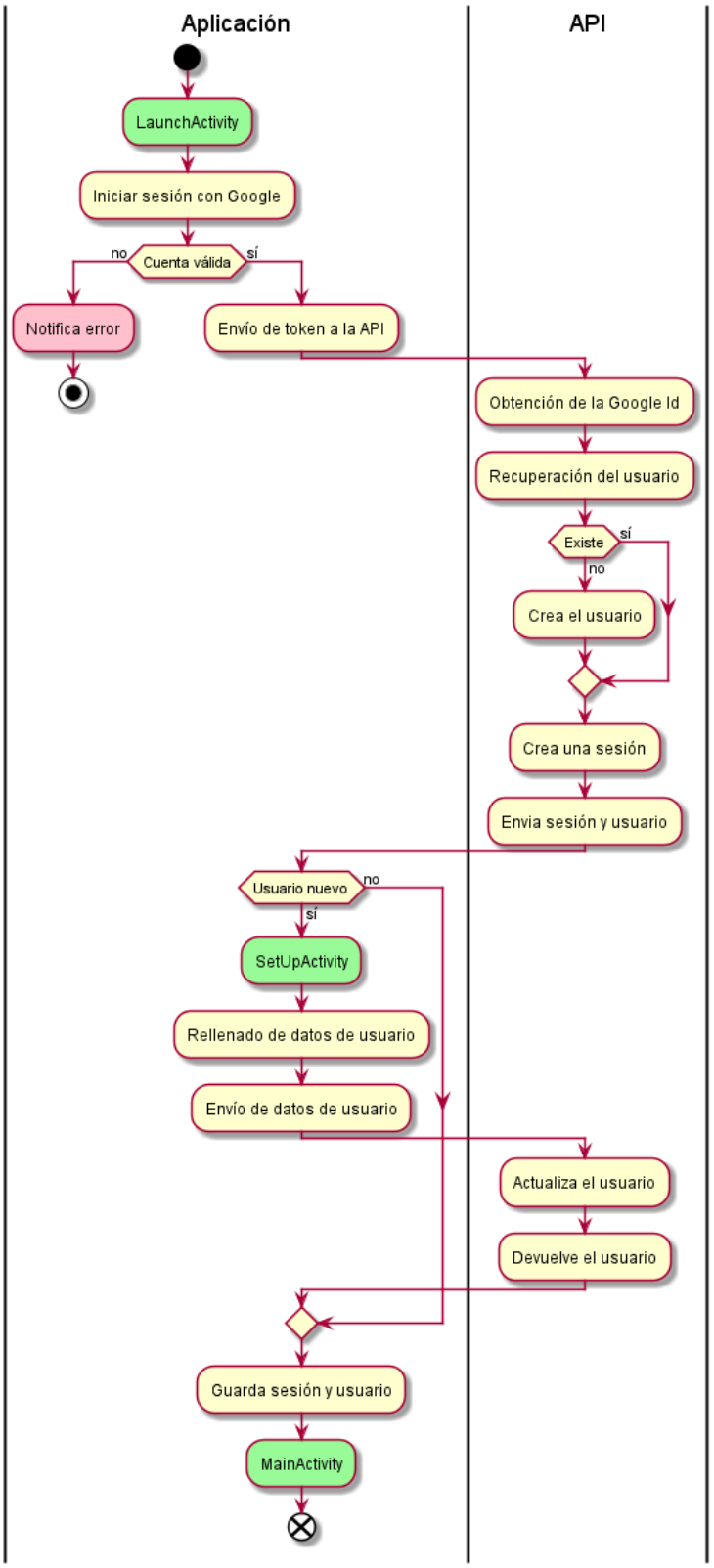


Figura 21.1: Diagrama de actividades del inicio de sesión

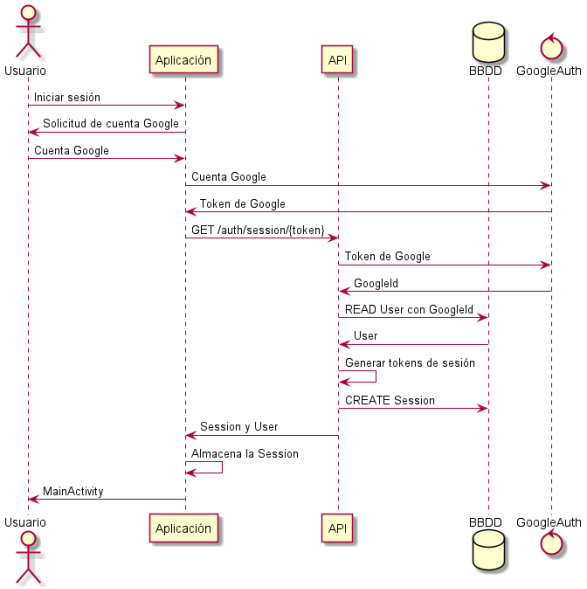


Figura 21.2: Diagrama de secuencia del inicio de sesión

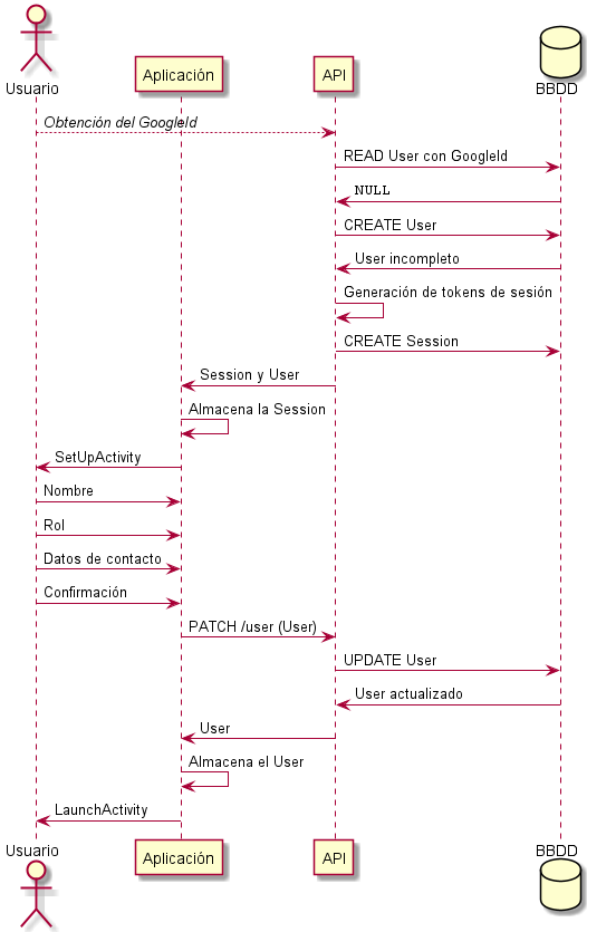


Figura 21.3: Diagrama de secuencia de registro

21.2. Vinculación

21.2.1. Creación de vínculos

La vinculación de usuarios (relativa al CU3. Vincularse con otro usuario) es una función que requiere la participación activa de **dos actores**, un Paciente y un Cuidador. El proceso paralelo puede verse en la Figura 21.4, mientras que la secuencia pormenorizada de los procesos y comunicaciones entre componentes se muestra en la Figura 21.5.

Este proceso se puede comprender como tres subprocesos: la generación del token o código de vinculación, el escaneo de este y la validación del vínculo. El primero de ellos es el realizado por el Paciente al elegir la opción de **Añadir vínculo** en su pantalla de vínculos. Al hacerlo, su aplicación móvil emitirá una petición de token al respectivo endpoint (Cuadro 20.14) de la API, que lo generará en base a los datos de dicho paciente y lo enviará como respuesta. Una vez que la aplicación móvil lo reciba, lo convertirá en un **código QR** y lo mostrará en la pantalla del Paciente. Estos códigos tendrá un tiempo de caducidad que también se enviará junto al código, si llega a caducar, la aplicación solicitará otro código y lo mostrará en la pantalla.

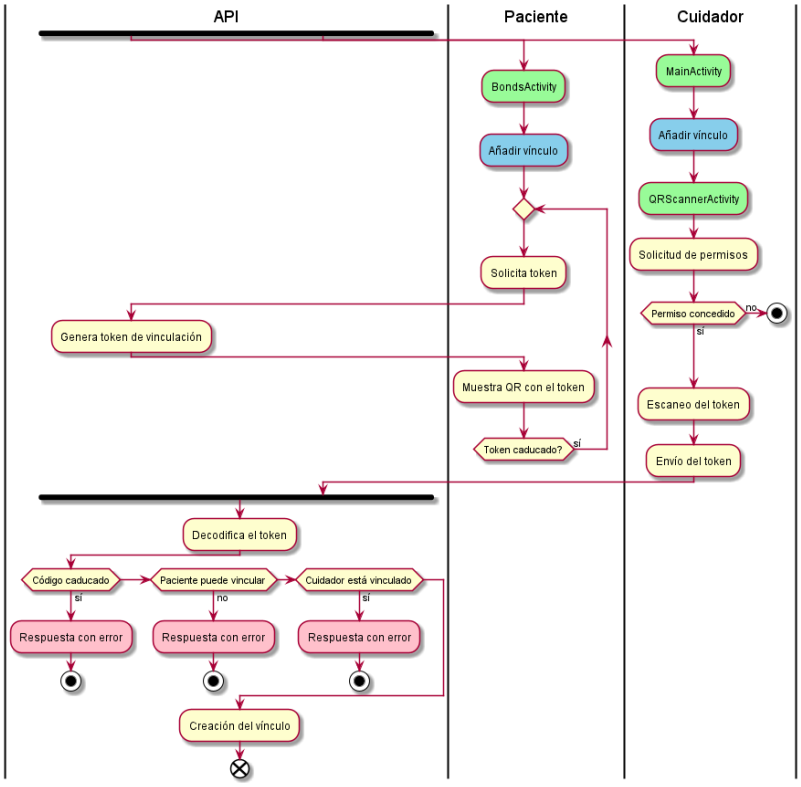


Figura 21.4: Diagrama de actividades de la vinculación de usuarios

El Cuidador también iniciará su proceso por medio de una opción de **Añadir vínculo** situado en su pantalla principal. Al usarla la aplicación se dirigirá a la actividad del escáner y, previa obtención de requisitos de usuario, desde ella podrá enfocar el código QR mostrada en la aplicación del Paciente. La aplicación del Cuidador escaneará e **interpretará el código QR** obteniendo el token que enviará a la API con una petición al endpoint del Cuadro 20.15.

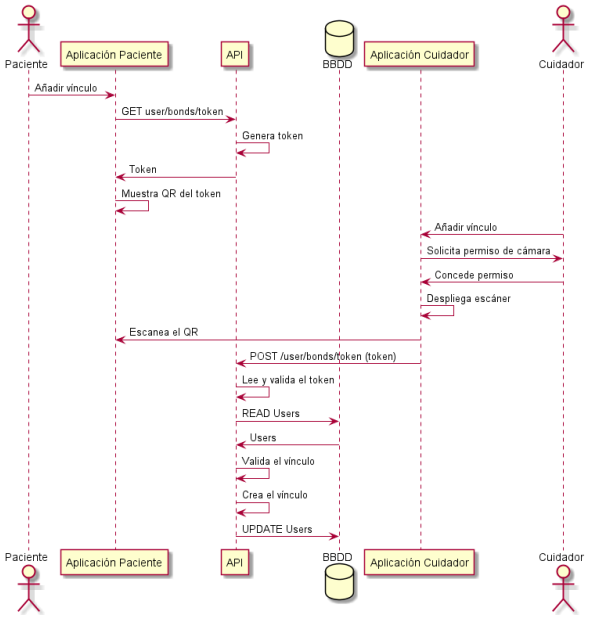


Figura 21.5: Diagrama de secuencia de la vinculación de usuarios

Finalmente, una vez la API reciba el token del paciente comenzará su **lectura y validación**. Comprobará que el token aún no haya caducado, si es el caso, lo leerá y obtendrá de él la ID del Paciente a vincular. Después solicitará la información de los usuarios para **comprobar si el vínculo es posible**. Si ese Paciente aún puede vincular más Cuidadores y el Cuidador no está vinculado, entonces la API creará el vínculo añadiendo a cada uno de los actores a la lista de vínculos del otro, tras lo que actualizará los datos de ambos en la base de datos. De esta forma **el vínculo se habrá creado**.

21.3. Ubicación

El **CU4. Compartir ubicación** trata acerca del envío de la ubicación actual del usuario al resto de sus usuarios asociados conectados a la misma sala de geolocalización. Un usuario se une a esta sala al acceder a la actividad de geolocalización, al hacerlo se envía un evento de suscripción a la WebSocket API, que lo incluye en dicha sala.

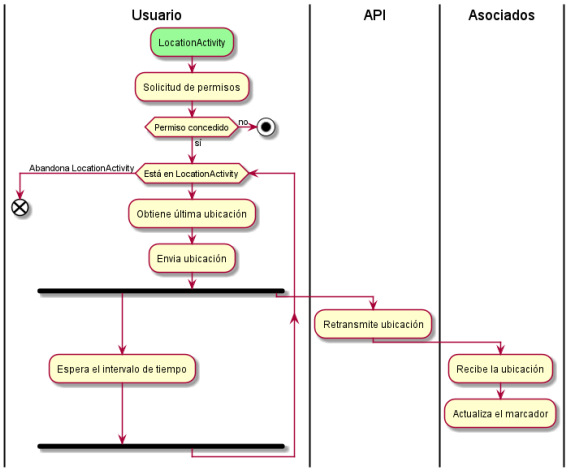


Figura 21.6: Diagrama de actividades de la difusión de localizaciones

Para poder compartir la ubicación del usuario es necesario que este **conceda permisos** a la aplicación para poder usar los servicios de su dispositivo que la proporcionan. Una vez que lo haga, la aplicación mostrará al usuario una pantalla con un mapa y su localización además de la del resto de usuarios conectados, si hay alguno. En ese momento también se iniciará una tarea de fondo que de forma repetida tras un intervalo de tiempo **solicitará la última ubicación** del usuario y procederá a enviarla por el WebSocket a la API.

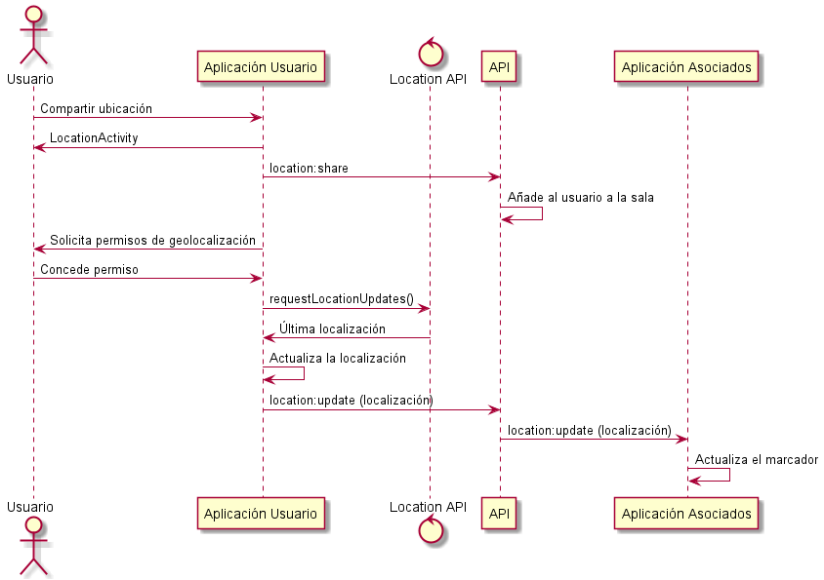


Figura 21.7: Diagrama de secuencia de la difusión de localizaciones

La API recibirá las ubicaciones de los usuario a través del evento de actualización (Cuadro 20.28), mismo evento que usará para **retransmitir esa ubicación** al resto de

usuario conectados a dicha sala (pero no al emisor). Los usuarios asociados que estén compartiendo su ubicación recibirán la ubicación del usuario a través de dicho evento y actualizarán el marcador del mismo acorde a la nueva localización.

21.4. Tareas y mensajes

21.4.1. Envío de un mensaje

El envío de un mensaje comprende todo el recorrido desde que un usuario crea y envía un mensaje desde su pantalla de feed hasta que este llega al resto de usuarios asociados, se corresponde con el CU6. Comunicarse por mensaje instantáneo y se puede ver en la Figura 21.8.

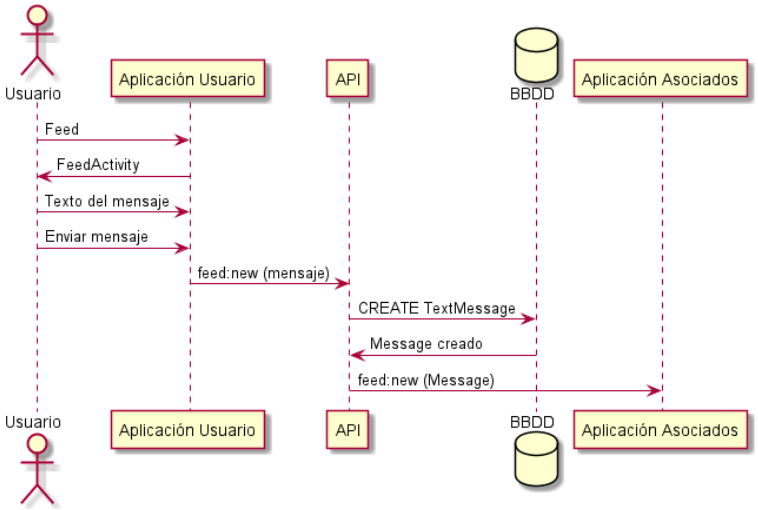


Figura 21.8: Diagrama de secuencia del envío de un mensaje

La secuencia de envío de mensajes arranca cuando un usuario redacta un mensaje y utiliza el botón de enviar de la actividad del feed. Al hacerlo, la aplicación envía el mensaje a través del socket según el evento del Cuadro 20.23. Cuando la API recibe dicho mensaje lo persiste en la base de datos y lo reenvía por medio del mismo evento al resto de usuarios asociados, que lo mostrará en pantalla si están conectados a la sala de feed.

21.4.2. Creación de nueva tarea

Una nueva tarea (CU5. Gestionar tareas) se puede crear de dos maneras distintas. Una por medio del envío de la misma a través del feed, esta vía sigue un proceso similar

al envío de mensajes, únicamente cambiando el tipo de entidad que se persiste, véase pues la sección anterior (21.4.1 Envío de un mensaje). La otra es desde la pantalla de tareas por medio de la función de crear una nueva tarea.

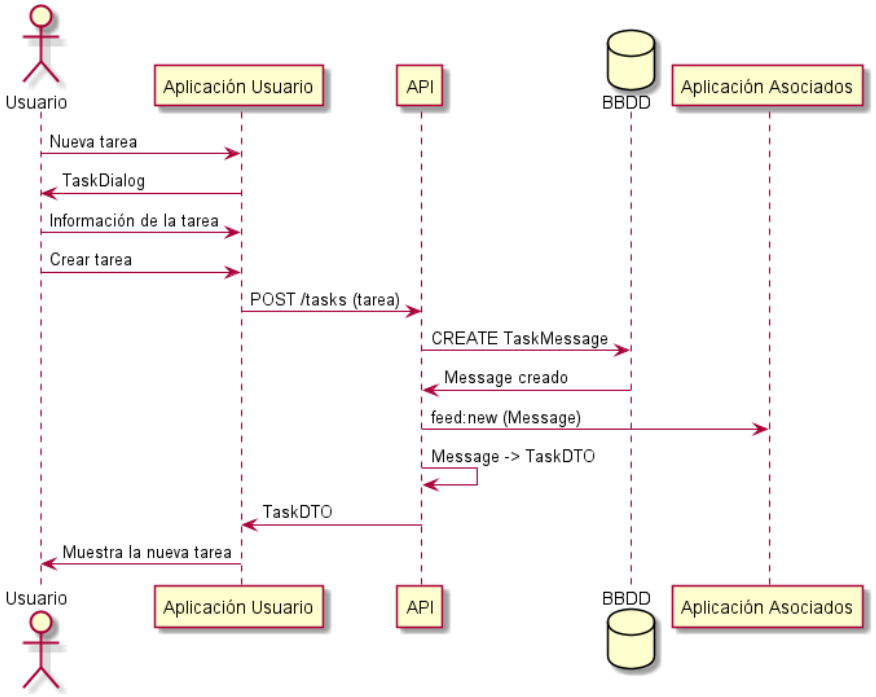


Figura 21.9: Diagrama de secuencia de la creación de una tarea

En el diálogo desplegado por dicha opción de la pantalla de tareas el usuario rellenará los datos de la tarea y una vez confirme su creación, la aplicación móvil enviará dicha tarea a la API por medio del endpoint del Cuadro 20.8. Una vez la API reciba dicha tarea la enviará a la base de datos para persistirla y completar su creación con una ID (ver 21.4.3 Estado de una tarea). Finalmente, la tarea obtenida tras dicha creación será enviada a través de dos canales: primero como mensaje por WebSocket a la sala del feed (Cuadro 20.23) y, posteriormente, como respuesta a la petición HTTP como TaskDto.

21.4.3. Estado de una tarea

Las cuatro acciones que se pueden realizar sobre una tarea (crearla, marcarla como hecha, marcarla como no hecha y eliminarla) derivan en los estados que se pueden ver en la Figura 21.10. Primero, cuando un usuario crea una nueva tarea desde su aplicación esta existe inicialmente como **Nueva** y contiene los datos de un TaskMinDto.

Esta tarea es enviada por la aplicación por medio de cualquiera de los dos sistemas

de creación (WebSocket o HTTP). Entonces es recibida por la API y persistida en la base de datos, momento en el que recibe una ID y se convierte en **Creada**. A diferencia de una tarea Nueva, que siempre existe como *No hecha*, una tarea Creada puede tener dos estados: **Hecha**, en caso de que haya sido completada, y *No hecha*, si aún se debe realizar. La transición entre estos dos estados pasa por peticiones a los endpoints respectivos (Cuadro 20.11 Documentación del endpoint de marcar una tarea como hecha y Cuadro 20.12 Documentación del endpoint de marcar una tarea como no hecha) con la ID de la tarea.

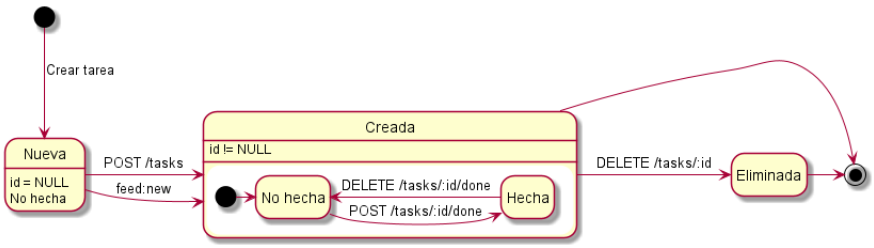


Figura 21.10: Diagrama de estado de una tarea

Finalmente, el último estado que puede tomar una tarea es el de **Eliminada**, para ello la tarea debía estar ya Creada y una petición de eliminación (véase Documentación del endpoint de eliminar una tarea) debe haber sido enviada a la API. Tanto Creada como Eliminada son estados finales de la tarea, el único estado inicial es el de Nueva.

21.5. Notificaciones

21.5.1. Envío de notificaciones

Cuando la API procesa alguna de las acciones listadas en el punto RSN 1. genera una notificación. En la creación de esta notificación la API busca en la base de datos los usuarios asociados y los añade como interesados a la entidad de la notificación antes de almacenarla en la base de datos de forma que pueda ser recuperada más adelante.

La notificación completa y devuelva por la base de datos es finalmente enviada por medio del WebSocket y el evento respectivo a la acción a notificar a los usuarios asociados, una vez recibida por las diferentes aplicaciones la muestran a sus usuarios por alguno de los medios dispuestos para tal hecho.

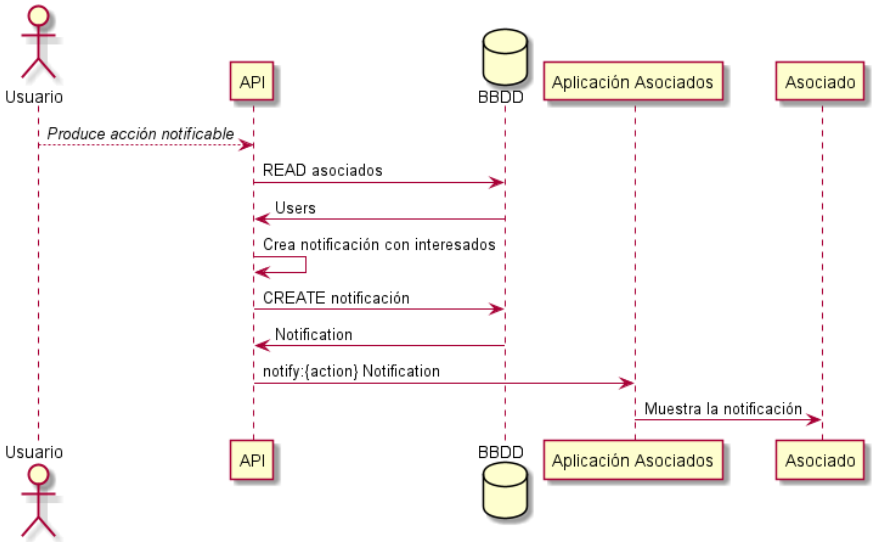


Figura 21.11: Diagrama de secuencia del envío de notificaciones

22. Interfaz de usuario

22.1. Guía de estilo

A la hora de llevar a cabo el diseño de las interfaces de usuario se realizó una definición del estilo gráfico que tendría la aplicación y que se debería seguir para conseguir consistencia entre todas las pantallas de la misma. La base de la guía de estilo son **las directrices de Material Design** (véase Apartado 25.3.2). Sobre esas bases se decidieron también los siguientes puntos:

- El color principal de la aplicación será el *dodger blue* (`#29b6f6`).
- El color secundario de la aplicación será su complementario, el *selective yellow* (`#ffb300`).
- La aplicación ofrecerá temas claro y oscuro. Por defecto se mostrará acorde al tema del sistema operativo del usuario.
- Se utilizarán tarjetas para mostrar la información personal de los usuarios.
  - El nombre del usuario es la información mínima a mostrar inicialmente en la tarjeta.
  - La información adicional de contacto estará en el anverso de la tarjeta o en una extensión desplegable, se usará la alternativa que más convenga a cada situación.
  - Las dos alternativas se usarán con botones situados en el lado derecho de la tarjeta.
- Todos los botones de navegación ofrecerán un icono y un texto indicando el destino.
- Los elementos de la interfaz como los botones o tarjetas tendrán bordes redondeados.
- Los errores se comunicarán por medio de *toasts* al usuario.
- Todos los elementos utilizarán colores definidos en el tema de la aplicación.

22.1.1. Tema de color

El tema de color fue generado con la herramienta de que Material ofrece para tal empresa, **Color Tool** (Apartado 25.4.1), a partir de los dos colores que fueron

seleccionados como colores base de la aplicación. La paleta de colores resultante fue la siguiente:

Color	RGB
Primario	`#29b6f6`
Primario claro	`#73e8ff`
Primario oscuro	`#0086c3`
Secundario	`#ffb300`
Secundario claro	`#ffe54c`
Secundario oscuro	`#c68400`
Texto primario	`#000000`
Texto secundario	`#212121`
Superficie	`#eaf4f6`

Tabla 22.1: Paleta de colores de la aplicación

22.1.2. Icono de la aplicación

El icono de la aplicación (Figura 22.3) está basado en el lema principal y nombre de la aplicación, el *"Todos para uno y uno para todos"*, que ya se comentó en el Apartado 2.1, y en el concepto de vínculos entre pacientes y cuidadores.

El icono está compuesto por otros dos iconos que representan a su vez a los dos tipos de usuarios de la aplicación: los **Pacientes** (Figura 22.1) y los **Cuidadores** (Figura 22.2). Los primeros son el punto central alrededor del que gira este sistema y su icono representa esto, un entidad central que recibe la ayuda exterior de sus cuidadores. Los Cuidadores por otro lado se representan como esas entidades exteriores que orbitan alrededor del Cuidador y que al hacerlo establecen vínculos entre sí, pues la base de esta aplicación radica también la cooperación entre Cuidadores. Los dos iconos superpuestos representan el entramado que se teje en *All For One*, una cadena de socorro mutuo entre Pacientes y Cuidadores.

Figura 22.1: Icono de Paciente

Figura 22.2: Icono de Cuidador

Figura 22.3: Icono de AllForOne

22.2. Logo de la aplicación

Además del icono también se creó un logo que permita identificar la aplicación de una forma más descriptiva y representativa, pues estará compuesto del nombre de la aplicación. La base de su diseño es el propio nombre de la aplicación y, por tanto, el lema de la misma. El logo (Figura 22.4) es un **All** escrito con los colores del tema, los cuales distinguen también un **4** sobre la letra *A* y un **1** bajo la letra *L* del final. Esto nos deja: **ALL-4-1**, lo cuál leído en inglés tiene una lectura similar al nombre de la aplicación: **All For One.**

Figura 22.4: Logo de la aplicación

22.3. Launch

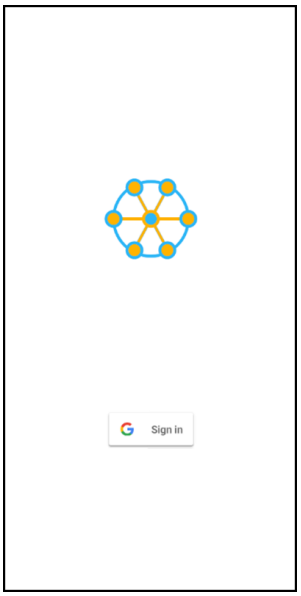


Figura 22.5: Diseño de la pantalla de lanzamiento

La pantalla de lanzamiento (Figura 22.5) ofrece una única funcionalidad, **el inicio de sesión** con la cuenta de Google del usuario., por ello la pantalla contará únicamente con un botón para llevar a cabo esa acción. El botón en cuestión debe ser el servido por la propia biblioteca de autenticación de Google para ofrecer familiaridad al usuario y transmitir rápidamente que el inicio de sesión usa ese servicio. Aparte de dicho botón se dispondrá el icono de la aplicación para que el usuario sepa qué aplicación está empleando por si se inicia o accede a ella de forma indirecta. Hay margen para otros posibles añadidos como podrían ser la marca de registro o la versión de la aplicación.

22.4. Set Up

Figura 22.6: Diseño de la pantalla de configuración de nombre

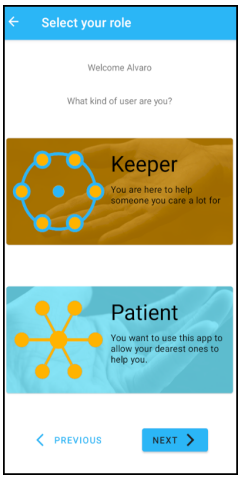


Figura 22.7: Diseño de la pantalla de elección de rol

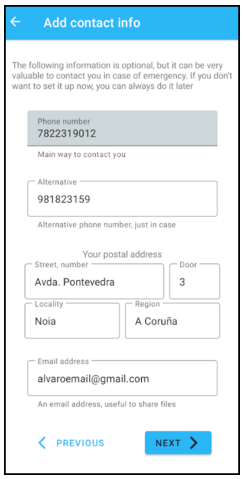


Figura 22.8: Diseño de la pantalla de introducción de datos

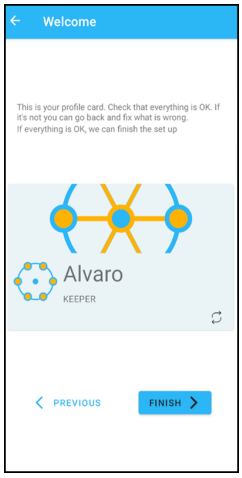


Figura 22.9: Diseño de la pantalla de confirmación del perfil

La pantalla de configuración del perfil será la pantalla a la que accedan los nuevos usuarios que intenten iniciar sesión por primera vez. Esta pantalla estará compuesta por cuatro distintas, una para cada etapa del proceso con tendrán botones para avanzar o retroceder entre ellas.

La primera de estas pantallas es la de **configuración del nombre** (Figura 22.6), donde habrá un campo de texto en el que el usuario podrá introducir el nombre que lo represente. A esta pantalla le sigue la de **selección del rol** (Figura 22.7), que ofrecerá dos tarjetas representando las opciones.

Una vez seleccionado el rol, el usuario progresará a otra pantalla (Figura 22.8) donde podrá rellenar sus **datos de contacto** en los distintos campos de texto que se le dispondrán para tal acción. Tras completar todos estos pasos la única pantalla restante será la de **confirmación del perfil** (Figura 22.9). En esta última pantalla se mostrará la tarjeta de perfil del usuario con los datos que ha introducido de forma que pueda ver todo de nuevo de forma sencilla antes de terminar el registro.

22.5. Main

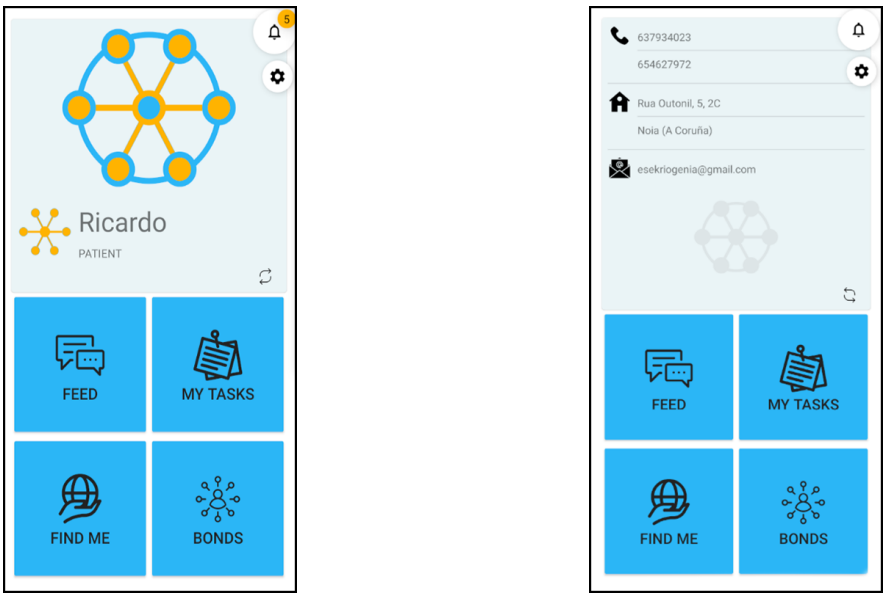


Figura 22.10: Diseño de la pantalla principal

La pantalla principal es el punto de navegación principal de la aplicación y da acceso al resto de funciones de la aplicación. Al ser el también el punto de entrada tras el inicio de sesión es un lugar idóneo para situar **los datos del Paciente**, de forma que el acceso a los mismos requiera el menor esfuerzo posible de memoria, facilitando el acceso a ellos a los usuarios con los síntomas de la enfermedad. La tarjeta es voltaeable por medio del botón situado en su esquina inferior derecha.

Las cuatro grandes funciones principales de la aplicación (el feed de comunicación, la gestión de tareas, la geolocalización y la consulta de vínculos) serán accesibles a través de los **cuatros grandes botones** situados bajo la tarjeta del Paciente. Al ser las funciones principales cuenta con un acceso más resaltable que los demás puntos de navegación de la pantalla.

Esos otros dos puntos de navegación son el acceso a las notificaciones y a los ajustes.

Los botones a los mismos no cuentan con el relleno del color principal y están situados en la esquina superior derecha, lugar habitual de localización para estas funciones. El acceso a las notificaciones contará además con una chapa indicando **el número de notificaciones** no leídas que tiene el usuario, indicando así si es necesario desplegar dicha pantalla.

22.6. Feed

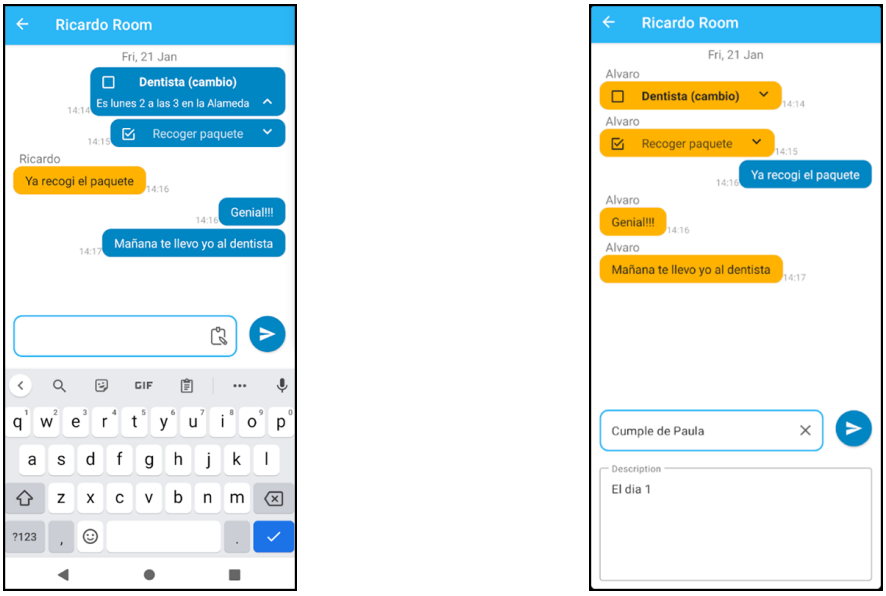


Figura 22.11: Diseño de la pantalla de feed

El diseño de la pantalla del feed (Apartado 22.6) es similar a los diseños habituales de **pantallas de chat**. En el lado derecho y en azul se mostrarán los mensajes usados por el propio usuario y a la izquierda se listarán los mensajes enviados por los usuarios asociados con el nombre del usuario que lo envió. Como en los diseños habituales, en la parte inferior se ofrecerá un campo de texto para escribir y enviar el mensaje con el botón de su derecha.

Todos los mensajes mostrarán la hora de envío de los mismos y se agruparán por fecha de envío. Las tareas se mostrarán como mensajes de texto pero incluyendo además una casilla representando su estado y una flecha que permita desplegar la descripción de la tarea. Las tareas ya completas se verán con un color de texto más apagado para que sean fácilmente distinguibles a primera vista.

Una diferencia que tendrá esta pantalla de feed respecto a otros chats será el **modo crear tarea**. Este se mostrará usando el botón con forma de portafolios del interior

del campo de texto. Al usarlo el icono cambiará a una X que sirva para cerrarlo y una campo de texto mayor se abrirá en la parte inferior de la pantalla para poder introducir la descripción. El envío de estas será con el mismo botón que los mensajes de texto.

22.7. Tasks

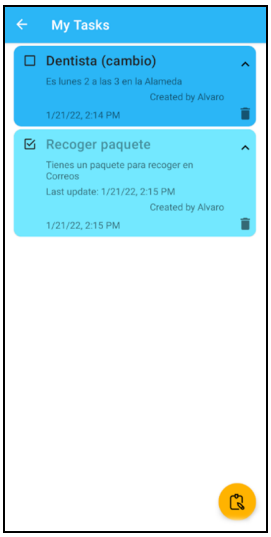


Figura 22.12: Diseño de la pantalla de tareas

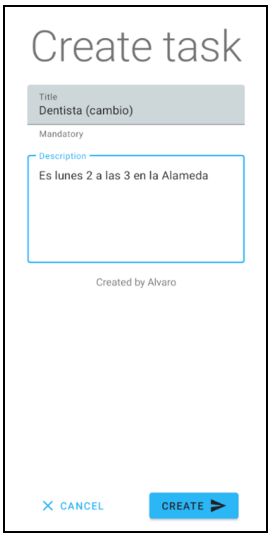


Figura 22.13: Diseño de la pantalla de crear tarea

La pantalla de tareas (Figura 22.12) lista las tareas en tarjetas con el título, una casilla con el estado de la tarea, el autor de la misma y la fecha de creación de la tarea. Más información como la descripción o el último momento de actualización se mostrarán al desplegar los datos de la tarea con la flecha de la tarjeta para tal uso. Las tareas ya completas tendrán color y texto más claro, por lo que a simple vista **podrán diferenciarse los dos tipos de tarea**. La casilla del estado será también un botón que se pueda usar para cambiar el estado de la tarea, además de este botón habrá otro para eliminarlas.

Finalmente, la pantalla también tendrá un botón para desplegar la pantalla de **creación de tareas** (Figura 22.13). En dicho diálogo habrá un campo de texto para introducir el título y otro para rellenar la descripción. La creación de la tarea se podrá confirmar o cancelar con los botones que este diálogo tendrá en la parte inferior.

22.8. Location

Figura 22.14: Diseño de la pantalla de geolocalización

La pantalla de geolocalización (Figura 22.14) es una **actividad con un mapa** que ocupa toda la actividad similar a muchos otros servicios. Los colores del mapa de Google Maps se ofrecerán modificados para que vayan más acordes a los colores de la aplicación. En esa pantalla se mostrará la posición del usuario con un círculo azul con un cono visión. Marcadores de distintos colores representarán las posiciones del resto de usuarios asociados conectados. Pulsar en esos marcadores mostrará el nombre del usuario al que pertenece.

22.9. Bonds

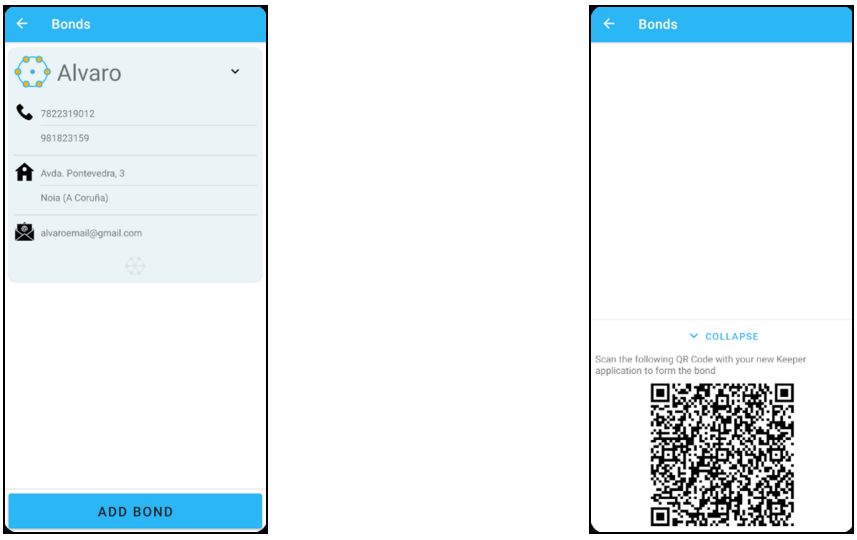


Figura 22.15: Diseño de la pantalla de vínculos

La función de acceso a los vínculos se servirá en una pantalla (Apartado 22.9) que listará los vínculos. Cada uno de estos vínculos **será una tarjeta** con el nombre del usuario que podrá ser desplegada para mostrar los datos de contacto. Los pacientes, además, tendrán un botón en la parte inferior que les permitirá desplegar un nuevo código QR con un token de vinculación como se puede ver en Apartado 22.9.

22.10. Scanner

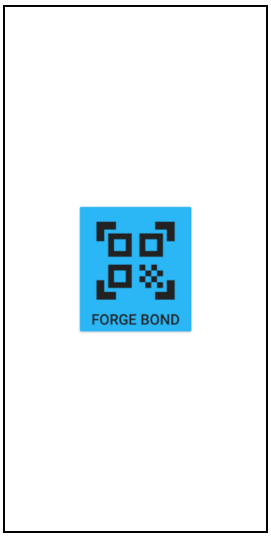


Figura 22.16: Diseño de la pantalla principal sin vínculo

Figura 22.17: Diseño de la pantalla de escáner

Para completar los vínculos se debe escanear el código QR, lo que se hace en la pantalla de escáner (Figura 22.17) a la que se podrán acceder los **Cuidadores no vinculados** a través de su actividad principal, que se mostrará como se ve en Figura 22.16, con únicamente un botón para abrir el escáner. El susodicho escáner será una pantalla ocupada por completo por la cámara en la que estará oscurecida la sección en la que no podrá enfocarse el código.

22.11. Notifications

Las notificaciones se desplegarán y mostrarán en una pantalla como la mostrada en Figura 22.18. Dicha pantalla listará las acciones a notificar agrupadas según la fecha en la que sucedieron y en un orden de más recientes a más antiguas. Cada notificará contará con un botón con el icono de un ojo para marcar esa notificación como vista.

Aparte de ese botón habrá otro en la parte superior para **marcar todas como vistas**. Por último, las notificaciones relacionadas con alguna pantalla o acceso que se deba visitar contarán con un botón para navegar hasta dicho lugar de la aplicación.

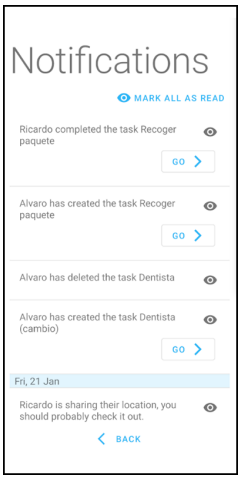


Figura 22.18: Diseño de la pantalla de notificaciones

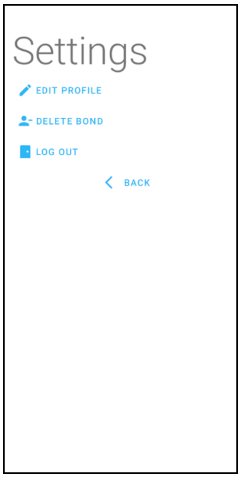


Figura 22.19: Diseño de la pantalla de ajustes

22.12. Settings

Las opciones, al igual que las notificaciones, será un diálogo a pantalla completa que se abrirá desde la pantalla principal. Será como la Figura 22.19. En esta pantalla se listarán las posibles opciones del usuario. La opción de **editar usuario** desplegará campos de texto de las diferentes propiedades del usuario para que este pueda editarlas, lo cuál confirmará con un botón que también se desplegará con la misma opción. El resto de opciones serán también botones, y habrá uno más para cerrar el diálogo y retornar a la pantalla principal.

22.13. Mapa de navegación

El mapa de navegación definitivo con las pantallas recién mostradas es el ilustrado en Figura 22.20. Las pantallas con fondo azul requieren inicio de sesión y las que tienen fondo verde son exclusivas de los Cuidadores no vinculados.

La navegación comienza siempre en LaunchActivity. Si el usuario no está registrado avanzará a SetUpActivity a través de sus cuatro fragmentos en orden. Si está registrado o si completa el registro a través de SetUpActivity avanzará a MainActivity, desde

donde se navegarán al resto de actividades y a los diálogos de notificaciones y ajustes.

Se ha intentando reducir los pasos de navegación al mínimo y por ello prácticamente todas las funcionalidades son accesibles con una única navegación desde la pantalla principal. A excepción del diálogo de crear tarea, que debe accederse desde TasksActivity.

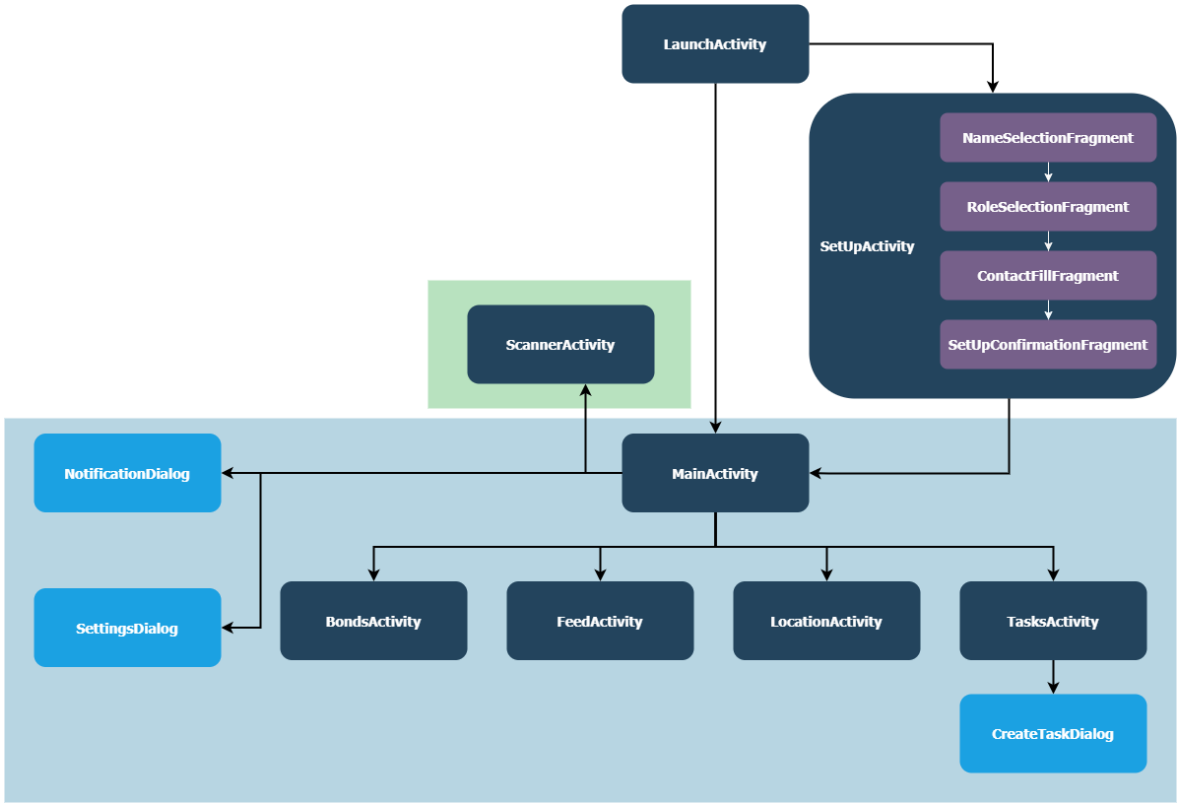


Figura 22.20: Mapa de navegación

23. Especificación técnica del plan de pruebas

23.1. Pruebas unitarias

23.1.1. Aplicación móvil

23.1.1.1. Gestor de ajustes

Actualizar datos del usuario	
Descripción	Se ejecuta la actualización de datos del usuario
Resultado esperado	Llamada a la actualización de los datos en el modelo de vista

Tabla 23.1: Prueba unitaria de la aplicación: Actualizar datos del usuario

Cerrar la sesión de usuario	
Descripción	Se ejecuta la función de cierre de sesión
Resultado esperado	Llamada al cierre de sesión en repositorio y el modelo de vista

Tabla 23.2: Prueba unitaria de la aplicación: Cerrar la sesión de usuario

Eliminar el vínculo con el Paciente vinculado	
Descripción	Se ejecuta la función de borrado de vínculos
Resultado esperado	Llamada al borrado de vínculo en repositorio y el modelo de vista

Tabla 23.3: Prueba unitaria de la aplicación: Eliminar el vínculo con el Paciente vinculado

23.1.1.2. Gestor de marcadores

Añadir un marcador	
Descripción	Se añade un marcador al gestor
Entrada	Un marcador de usuario
Resultado esperado	El marcador es añadido a la lista de marcadores

Tabla 23.4: Prueba unitaria de la aplicación: Añadir un marcador

Comprobar la existencia de un marcador	
Descripción	Se comprueba si un marcador existe ya en el gestor
Entrada	Un marcador de usuario existente Un marcador de usuario no existente
Resultado esperado	Verdadero o falso según el marcador exista o no

Tabla 23.5: Prueba unitaria de la aplicación: Comprobar la existencia de un marcador

Eliminar un marcador	
Descripción	Se intenta eliminar un marcador del gestor
Entrada	Un marcador de usuario existente Un marcador de usuario no existente
Resultado esperado	El marcador es eliminado si existe e ignorado si no

Tabla 23.6: Prueba unitaria de la aplicación: Eliminar un marcador

Actualizar un marcador	
Descripción	Se intenta actualizar un marcador del gestor
Entrada	Un marcador de usuario existente Un marcador de usuario no existente
Resultado esperado	El marcador es actualizado si existe e ignorado si no

Tabla 23.7: Prueba unitaria de la aplicación: Actualizar un marcador

23.1.1.3. Gestor de notificaciones

Añadir una notificación	
Descripción	Se añade una notificación al gestor
Entrada	Una notificación
Resultado esperado	La notificación es añadida a la lista de notificaciones

Tabla 23.8: Prueba unitaria de la aplicación: Añadir una notificación

Añadir un conjunto de notificaciones	
Descripción	Se añade un conjunto de notificaciones al gestor
Entrada	Un conjunto de notificaciones
Resultado esperado	Las notificaciones son añadidas a la lista de notificaciones

Tabla 23.9: Prueba unitaria de la aplicación: Añadir un conjunto de notificaciones

Limpiar lista de notificaciones	
Descripción	Se limpia la lista de notificaciones del gestor
Entrada	Lista de notificaciones con notificaciones Lista de notificaciones vacía
Resultado esperado	La lista de notificaciones queda vacía

Tabla 23.10: Prueba unitaria de la aplicación: Limpiar lista de notificaciones

Cargar lista de notificaciones	
Descripción	Se llama a la carga de notificaciones
Resultado esperado	Se llama al repositorio y las notificaciones obtenidas son añadidas a la lista

Tabla 23.11: Prueba unitaria de la aplicación: Cargar lista de notificaciones

Eliminar una notificación	
Descripción	Se elimina una notificación
Entrada	Notificación existente Notificación no existente
Resultado esperado	Si existe se llama al repositorio para leer la notificación y se borra de la lista

Tabla 23.12: Prueba unitaria de la aplicación: Eliminar una notificación

Eliminar todas las notificaciones	
Descripción	Se elimina todas las notificaciones
Entrada	Lista de notificaciones con notificaciones Lista de notificaciones vacía
Resultado esperado	Se llama al repositorio para leer todas las notificaciones existentes y la lista queda vacía

Tabla 23.13: Prueba unitaria de la aplicación: Eliminar todas las notificaciones

23.1.1.4. Modelo de vista de configuración

Enviar confirmación de datos de usuario	
Descripción	Se llama al envío de la confirmación de datos del usuario
Entrada	Un objeto Usuario
Resultado esperado	Se llama al repositorio para la actualización y se invoca el cambio a la pantalla principal

Tabla 23.14: Prueba unitaria de la aplicación: Enviar confirmación de datos de usuario

23.1.2. Modelo de vista de lanzamiento

Gestionar resultado de inicio de sesión en Google	
Descripción	Se ejecuta la llamada que gestiona el inicio de sesión a partir del inicio de sesión con Google
Entrada	Token de sesión de Google
Resultado esperado	Se llama al repositorio para inicar sesión y se invoca el cambio a la pantalla principal

Tabla 23.15: Prueba unitaria de la aplicación: Gestionar resultado de inicio de sesión en Google

Gestionar resultado de inicio de sesión en Google erróneo	
Descripción	Se maneja un intento inicio de sesión con Google fallido
Entrada	Token de sesión de Google inválida
Resultado esperado	Lanzamiento de mensaje de error

Tabla 23.16: Prueba unitaria de la aplicación: Gestionar resultado de inicio de sesión en Google erróneo

23.1.2.1. Modelo de vista de localización

Añadir un nuevo marcador	
Descripción	Se añade un nuevo marcador al mapa
Entrada	Nueva localización
Resultado esperado	Se crea un marcador, se añade al gestor y se invoca una actualización de vista

Tabla 23.17: Prueba unitaria de la aplicación: Añadir un nuevo marcador

Recibir una actualización de posición de marcador no existente	
Descripción	Se gestiona la llegada de una actualización de posición de un marcador no conocido
Entrada	Nueva localización
Resultado esperado	Se crea un marcador, se añade al gestor y se invoca una actualización de vista

Tabla 23.18: Prueba unitaria de la aplicación: Recibir una actualización de posición de marcador no existente

Recibir una actualización de posición de marcador ya existente	
Descripción	Se gestiona la llegada de una actualización de posición de un marcador ya existente
Entrada	Localización actualizada
Resultado esperado	Se actualiza el marcador respectivo y se invoca una actualización de vista

Tabla 23.19: Prueba unitaria de la aplicación: Recibir una actualización de posición de marcador ya existente

23.1.2.2. Modelo de vista de mensajería

Obtener una página de mensajes	
Descripción	Se gestiona la petición de una nueva página de mensajes
Entrada	Primera página Página más avanzada Página inválida
Resultado esperado	Se llama al repositorio y se actualiza la lista de mensajes

Tabla 23.20: Prueba unitaria de la aplicación: Obtener una página de mensajes

Enviar una tarea por el feed	
Descripción	Se envía una tarea a través del feed
Entrada	Una tarea
Resultado esperado	Se llama al repositorio para el envío

Tabla 23.21: Prueba unitaria de la aplicación: Enviar una tarea por el feed

Enviar un mensaje por el feed	
Descripción	Se envía un mensaje a través del feed
Entrada	Un mensaje
Resultado esperado	Se llama al repositorio para el envío

Tabla 23.22: Prueba unitaria de la aplicación: Enviar un mensaje por el feed

Recibir una actualización de tarea	
Descripción	Se gestiona la llegada de una actualización de una tarea
Entrada	Tarea existente actualizada Tarea no existente actualizada
Resultado esperado	Se actualiza la tarea si existe y se invoca una actualización de vista

Tabla 23.23: Prueba unitaria de la aplicación: Recibir una actualización de tarea

Recibir una eliminación de tarea	
Descripción	Se gestiona la llegada de una actualización de posición de un marcador ya existente
Entrada	Localización actualizada
Resultado esperado	Se actualiza el marcador respectivo

Tabla 23.24: Prueba unitaria de la aplicación: Recibir una eliminación de tarea

Recibir un nuevo mensaje	
Descripción	Se gestiona la llegada de una actualización de posición de un marcador ya existente
Entrada	Localización actualizada
Resultado esperado	Se actualiza el marcador respectivo

Tabla 23.25: Prueba unitaria de la aplicación: Recibir un nuevo mensaje

23.1.2.3. Modelo de vista de tareas

Confirmar creación de una tarea	
Descripción	Se confirma la creación de una tarea
Entrada	Propiedades una tarea
Resultado esperado	Se llama al repositorio, se añade a la lista de tareas y se invoca una actualización de vista

Tabla 23.26: Prueba unitaria de la aplicación: Confirmar creación de una tarea

Eliminación una tarea	
Descripción	Se elimina una de las tareas listadas por un usuario capaz de ello
Entrada	Tarea a eliminar
Resultado esperado	Se llama al repositorio, se elimina de la lista y se invoca una actualización de vista

Tabla 23.27: Prueba unitaria de la aplicación: Eliminación una tarea

Eliminación una tarea inválida	
Descripción	Se elimina una de las tareas listadas por un usuario que no puede hacerlo
Entrada	Tarea a eliminar
Resultado esperado	Mensaje de error

Tabla 23.28: Prueba unitaria de la aplicación: Eliminación una tarea inválida

Marcar tarea como hecha/no hecha	
Descripción	Se marca una de las tareas listadas por un usuario como hecha o no hecha
Entrada	Tarea hecha Tarea no hecha
Resultado esperado	Se llama al repositorio, se modifica la tarea y se invoca una actualización de vista

Tabla 23.29: Prueba unitaria de la aplicación: Marcar tarea como hecha/no hecha

Obtener la lista de tareas	
Descripción	Se gestiona la petición de la lista de tareas
Resultado esperado	Se llama al repositorio, se almacena la lista y se invoca una actualización de vista

Tabla 23.30: Prueba unitaria de la aplicación: Obtener la lista de tareas

23.1.2.4. Modelo de vista de vínculos

Eliminar un vínculo	
Descripción	Se gestiona la eliminación de un vínculo
Resultado esperado	Se llama al repositorio Se elimina de la lista y se invoca una actualización de vista

Tabla 23.31: Prueba unitaria de la aplicación: Eliminar un vínculo

Obtener un código QR de vinculación	
Descripción	Se gestiona la petición de un código QR de vinculación
Resultado esperado	Se llama al repositorio, se transforma el código a QR, se almacena y se invoca una actualización de vista

Tabla 23.32: Prueba unitaria de la aplicación: Obtener un código QR de vinculación

Obtener los vínculos	
Descripción	Se gestiona la petición de la lista de vínculos
Resultado esperado	Se llama al repositorio, se almacena la lista y se invoca una actualización de vista

Tabla 23.33: Prueba unitaria de la aplicación: Obtener los vínculos

23.1.2.5. Modelo de vista principal

Actualizar usuario	
Descripción	Se gestiona la petición de actualización de datos del usuario
Resultado esperado	Se llama al repositorio, se actualizan los datos y se invoca una actualización de vista

Tabla 23.34: Prueba unitaria de la aplicación: Actualizar usuario

Enviar código de vinculación	
Descripción	Se envía un código de vinculación para completar un vínculo
Entrada	Código
Resultado esperado	Se llama al repositorio y se lanza una petición de obtención de Paciente vinculado

Tabla 23.35: Prueba unitaria de la aplicación: Enviar código de vinculación

Obtener lista de notificaciones	
Descripción	Se gestiona la petición de la lista de notificaciones
Resultado esperado	Se llama al repositorio, se almacena la lista y se invoca una actualización de vista

Tabla 23.36: Prueba unitaria de la aplicación: Obtener lista de notificaciones

Obtener Paciente vinculado	
Descripción	Se gestiona la petición de los datos del Paciente vinculado
Resultado esperado	Se llama al repositorio, se almacenan los datos y se invoca una actualización de vista

Tabla 23.37: Prueba unitaria de la aplicación: Obtener Paciente vinculado

23.1.2.6. Repositorio de mensajes

Envío de mensaje	
Descripción	Se envía un mensaje por el socket
Entrada	Mensaje de texto
Resultado esperado	Se invoca al cliente con el evento y el mensaje

Tabla 23.38: Prueba unitaria de la aplicación: Envío de mensaje

Recuperación de mensajes	
Descripción	Se envía una petición de recuperación de la lista de mensajes
Entrada	Token de sesión
Resultado esperado	Se llama al servicio y se convierte la respuesta en una lista de mensajes

Tabla 23.39: Prueba unitaria de la aplicación: Recuperación de mensajes

23.1.2.7. Repositorio de notificaciones

Recuperación de notificaciones	
Descripción	Se envía una petición de recuperación de la lista de notificaciones
Entrada	Token de sesión
Resultado esperado	Se llama al servicio y se convierte la respuesta en una lista de notificaciones

Tabla 23.40: Prueba unitaria de la aplicación: Recuperación de notificaciones

Marcado de notificación como leída	
Descripción	Se manda una petición de marcado de una notificación como leída
Entrada	Notificación y token de sesión
Resultado esperado	Se llama al servicio

Tabla 23.41: Prueba unitaria de la aplicación: Marcado de notificación como leída

Marcado de todas las notificaciones como leídas	
Descripción	Se manda una petición de marcado de todas las notificaciones como leída
Entrada	Token de sesión
Resultado esperado	Se llama al servicio

Tabla 23.42: Prueba unitaria de la aplicación: Marcado de todas las notificaciones como leídas

23.1.2.8. Repositorio de sesión

Inicio de sesión	
Descripción	Se manda una petición de inicio de sesión
Entrada	Token de Google
Resultado esperado	Se llama al servicio y se convierte la respuesta en una sesión

Tabla 23.43: Prueba unitaria de la aplicación: Inicio de sesión

Cierre de sesión	
Descripción	Se manda una petición de cierre de sesión
Entrada	Token de sesión
Resultado esperado	Se llama al servicio

Tabla 23.44: Prueba unitaria de la aplicación: Cierre de sesión

Refresco de sesión	
Descripción	Se manda una petición de marcado de refresco de sesión
Entrada	Token de sesión y de refresco
Resultado esperado	Se llama al servicio y se convierte la respuesta en una sesión

Tabla 23.45: Prueba unitaria de la aplicación: Refresco de sesión

23.1.2.9. Repositorio de tareas

Recuperación de tareas	
Descripción	Se envía una petición de recuperación de la lista de tareas
Entrada	Token de sesión
Resultado esperado	Se llama al servicio y se convierte la respuesta en una lista de tareas

Tabla 23.46: Prueba unitaria de la aplicación: Recuperación de tareas

Guardado de una tarea	
Descripción	Se manda una petición de publicación de una tarea
Entrada	Token de sesión y una tarea
Resultado esperado	Se llama al servicio y se convierte la respuesta en una tarea

Tabla 23.47: Prueba unitaria de la aplicación: Guardado de una tarea

Eliminación de una tarea	
Descripción	Se manda una petición de eliminación de una tarea
Entrada	Token de sesión y una tarea
Resultado esperado	Se llama al servicio

Tabla 23.48: Prueba unitaria de la aplicación: Eliminación de una tarea

Actualización de una tarea	
Descripción	Se manda una petición de actualización de una tarea
Entrada	Token de sesión y una tarea
Resultado esperado	Se llama al servicio y se convierte la respuesta en una tarea

Tabla 23.49: Prueba unitaria de la aplicación: Actualización de una tarea

23.1.2.10. Repositorio de usuario

Actualización de un usuario	
Descripción	Se manda una petición de actualización de un usuario
Entrada	Token de sesión y un usuario
Resultado esperado	Se llama al servicio y se convierte la respuesta en un usuario

Tabla 23.50: Prueba unitaria de la aplicación: Actualización de un usuario

Eliminar vinculación	
Descripción	Se manda una petición de eliminación de vínculo con otro usuario
Entrada	Token de sesión y un usuario
Resultado esperado	Se llama al servicio

Tabla 23.51: Prueba unitaria de la aplicación: Eliminar vinculación

Enviar código de vinculación	
Descripción	Se manda una petición con un código de vinculación
Entrada	Token de sesión y el código
Resultado esperado	Se llama al servicio

Tabla 23.52: Prueba unitaria de la aplicación: Enviar código de vinculación

Recuperación de vínculos	
Descripción	Se envía una petición de recuperación de la lista de vínculos
Entrada	Token de sesión
Resultado esperado	Se llama al servicio y se convierte la respuesta en una lista de usuarios

Tabla 23.53: Prueba unitaria de la aplicación: Recuperación de vínculos

Recuperación del Paciente vinculado	
Descripción	Se envía una petición de recuperación del Paciente vinculado
Entrada	Token de sesión
Resultado esperado	Se llama al servicio y se convierte la respuesta en un usuario

Tabla 23.54: Prueba unitaria de la aplicación: Recuperación del Paciente vinculado

Solicitar código de vinculación	
Descripción	Se manda una petición de código de vinculación
Entrada	Token de sesión
Resultado esperado	Se llama al servicio y se convierte la respuesta en una cadena de texto

Tabla 23.55: Prueba unitaria de la aplicación: Solicitar código de vinculación

23.1.2.11. Factoría de servicios

Obtención de un servicio	
Descripción	Se crea un servicio del tipo especificado
Entrada	Diferentes tipos de servicios
Resultado esperado	Se obtiene el servicio esperado

Tabla 23.56: Prueba unitaria de la aplicación: Obtención de un servicio

23.1.2.12. Servicio de autenticación

Procesar respuesta de inicio de sesión	
Descripción	Se envía una petición de inicio de sesión y se procesa la respuesta
Entrada	Petición válida Petición inválida
Resultado esperado	Respuesta del servidor convertida en el objeto correspondiente

Tabla 23.57: Prueba unitaria de la aplicación: Procesar respuesta de inicio de sesión

Procesar respuesta de refresco de sesión	
Descripción	Se envía una petición de refresco de sesión y se procesa la respuesta
Entrada	Petición válida Petición inválida
Resultado esperado	Respuesta del servidor convertida en el objeto correspondiente

Tabla 23.58: Prueba unitaria de la aplicación: Procesar respuesta de inicio de sesión

23.1.2.13. Servicio de mensajería

Procesar respuesta de recuperación de notificaciones	
Descripción	Se envía una petición de recuperación de notificaciones y se procesa la respuesta
Entrada	Petición válida Petición inválida
Resultado esperado	Respuesta del servidor convertida en el objeto correspondiente

Tabla 23.59: Prueba unitaria de la aplicación: Procesar respuesta de recuperación de notificaciones

Procesar respuesta de recuperación de mensajes	
Descripción	Se envía una petición de recuperación de mensajes y se procesa la respuesta
Entrada	Petición válida Petición inválida
Resultado esperado	Respuesta del servidor convertida en el objeto correspondiente

Tabla 23.60: Prueba unitaria de la aplicación: Procesar respuesta de recuperación de mensajes

23.1.2.14. Servicio de tareas

Procesar respuesta de recuperación de tareas	
Descripción	Se envía una petición de recuperación de tareas y se procesa la respuesta
Entrada	Petición válida Petición inválida
Resultado esperado	Respuesta del servidor convertida en el objeto correspondiente

Tabla 23.61: Prueba unitaria de la aplicación: Procesar respuesta de recuperación de tareas

Procesar respuesta de publicación de tareas	
Descripción	Se envía una petición de publicación de tareas y se procesa la respuesta
Entrada	Petición válida Petición inválida
Resultado esperado	Respuesta del servidor convertida en el objeto correspondiente

Tabla 23.62: Prueba unitaria de la aplicación: Procesar respuesta de publicación de tareas

23.1.2.15. Servicio de usuarios

Procesar respuesta de actualización de usuario	
Descripción	Se envía una petición de actualización de usuario y se procesa la respuesta
Entrada	Petición válida Petición inválida
Resultado esperado	Respuesta del servidor convertida en el objeto correspondiente

Tabla 23.63: Prueba unitaria de la aplicación: Procesar respuesta de actualización de usuario

Procesar respuesta de recuperación de Paciente vinculado	
Descripción	Se envía una petición de recuperación de Paciente vinculado y se procesa la respuesta
Entrada	Petición válida Petición inválida
Resultado esperado	Respuesta del servidor convertida en el objeto correspondiente

Tabla 23.64: Prueba unitaria de la aplicación: Procesar respuesta de recuperación de Paciente vinculado

Procesar respuesta de recuperación de vínculos	
Descripción	Se envía una petición de recuperación de vínculos y se procesa la respuesta
Entrada	Petición válida Petición inválida
Resultado esperado	Respuesta del servidor convertida en el objeto correspondiente

Tabla 23.65: Prueba unitaria de la aplicación: Procesar respuesta de recuperación de vínculos

Procesar respuesta de recuperación de código de vinculación	
Descripción	Se envía una petición de código de vinculación y se procesa la respuesta
Entrada	Petición válida Petición inválida
Resultado esperado	Respuesta del servidor convertida en el objeto correspondiente

Tabla 23.66: Prueba unitaria de la aplicación: Procesar respuesta de recuperación de código de vinculación

23.1.3. API

23.1.3.1. Middleware de autenticación

Encabezado de autenticación correcto	
Descripción	Se lanza una petición privada con un encabezado de autenticación con formato y token correctos
Entrada	Petición con encabezado .Authorization: Bearer valid"
Resultado esperado	Extrae el token y llama a la siguiente función

Tabla 23.67: Prueba unitaria de la API: Encabezado de autenticación correcto

Encabezado de autenticación faltante	
Descripción	Se lanza una petición privada sin encabezado de autenticación
Entrada	Petición sin encabezado de autorización
Resultado esperado	Error Bad Request

Tabla 23.68: Prueba unitaria de la API: Encabezado de autenticación faltante

Encabezado de autenticación mal formateado	
Descripción	Se lanza una petición privada con encabezado de autenticación con formato erróneo
Entrada	Petición con encabezado sin Bearer Petición con encabezado con Bearer mal escrito
Resultado esperado	Error Bad Request

Tabla 23.69: Prueba unitaria de la API: Encabezado de autenticación mal formateado

Encabezado de autenticación con token inválido	
Descripción	Se lanza una petición privada con encabezado de autenticación con formato erróneo
Entrada	Petición con encabezado "Authorization: Bearer invalid"
Resultado esperado	Error Unauthorized

Tabla 23.70: Prueba unitaria de la API: Encabezado de autenticación con token inválido

23.1.3.2. Middleware de manejo de errores

Manejo correcto de errores conocidos	
Descripción	Se lanza un error HTTP esperando la respuesta correspondiente
Entrada	HttpError con diferentes estados
Resultado esperado	Respuesta de error con el estado del error lanzado

Tabla 23.71: Prueba unitaria de la API: Manejo correcto de errores conocidos

Manejo correcto de errores desconocidos	
Descripción	Se lanza un Error esperando una respuesta de error interno
Entrada	Error
Resultado esperado	Respuesta con estado 500: Internal Server Error

Tabla 23.72: Prueba unitaria de la API: Manejo correcto de errores desconocidos

23.1.3.3. Servicio de mensajes

Creación de un mensaje de texto	
Descripción	Se persiste un mensaje de texto válido en la base de datos
Entrada	Mensaje de texto válido
Resultado esperado	Mensaje persistido en la base de datos Retorno del mensaje creado

Tabla 23.73: Prueba unitaria de la API: Creación de un mensaje de texto

Creación de un mensaje de texto inválido	
Descripción	Se persiste un mensaje de texto inválido en la base de datos
Entrada	Mensaje de texto sin alguna propiedad obligatoria
Resultado esperado	HttpError con BadRequest

Tabla 23.74: Prueba unitaria de la API: Creación de un mensaje de texto inválido

Recuperación de mensajes	
Descripción	Se recupera la primera página de mensajes de una habitación
Entrada	Habitación de los mensajes
Resultado esperado	Lista de mensajes esperados

Tabla 23.75: Prueba unitaria de la API: Recuperación de mensajes

Recuperación de mensajes de una página concreta	
Descripción	Se recupera una página de mensajes de una habitación distinta de la primera
Entrada	Habitación de los mensajes y página a recuperar
Resultado esperado	Lista de mensajes esperados

Tabla 23.76: Prueba unitaria de la API: Recuperación de mensajes de una página concreta

23.1.3.4. Servicio de notificaciones

Creación de una notificación	
Descripción	Se crea y persiste una notificación en la base de datos
Entrada	Acción e ID del autor
Resultado esperado	Notificación persistida en la base de datos Retorno de la notificación creada

Tabla 23.77: Prueba unitaria de la API: Creación de una notificación

Marcado de una notificación con más interesados como leída	
Descripción	Se marcan una notificación con más interesados como leída por un usuario
Entrada	ID de la notificación y del usuario
Resultado esperado	La notificación es marcada como leída por el usuario

Tabla 23.78: Prueba unitaria de la API: Marcado de una notificación con más interesados como leída

Marcado de una notificación sin más interesados como leída	
Descripción	Se marcan una notificación sin más interesados como leída por el usuario
Entrada	ID de la notificación y del usuario
Resultado esperado	La notificación es eliminada de la base de datos

Tabla 23.79: Prueba unitaria de la API: Marcado de una notificación sin más interesados como leída

Marcado de todas las notificaciones como leídas	
Descripción	Se marcan las notificaciones de un usuario como leídas
Entrada	ID del usuario
Resultado esperado	Todos las notificaciones del usuario marcadas como leídas

Tabla 23.80: Prueba unitaria de la API: Marcado de notificaciones como leídas

Recuperación de notificaciones	
Descripción	Se recupera las notificaciones no leídas por defecto
Entrada	ID del usuario
Resultado esperado	Lista de notificaciones esperadas

Tabla 23.81: Prueba unitaria de la API: Recuperación de notificaciones

Recuperación de notificaciones con edad especificada	
Descripción	Se recupera las notificaciones no leídas con una edad máxima especificada
Entrada	ID del usuario y edad máxima de la notificación
Resultado esperado	Lista de notificaciones esperadas

Tabla 23.82: Prueba unitaria de la API: Recuperación de notificaciones con edad especificada

23.1.3.5. Servicios de sesiones

Inicio de sesión	
Descripción	Se crea y persiste una nueva sesión en la base de datos
Entrada	Expiración y los tokens de autenticación y refresco
Resultado esperado	Sesión persistida en la base de datos Retorno de la sesión creada

Tabla 23.83: Prueba unitaria de la API: Inicio de sesión

Cierre de sesión	
Descripción	Se elimina una sesión en la base de datos
Entrada	Token de autenticación de la sesión
Resultado esperado	Sesión eliminada en la base de datos

Tabla 23.84: Prueba unitaria de la API: Cierre de sesión

Comprobación de sesión abierta	
Descripción	Se comprueba que una sesión activa lo está
Entrada	Token de autenticación de la sesión
Resultado esperado	Verdadero

Tabla 23.85: Prueba unitaria de la API: Comprobación de sesión abierta

Comprobación de sesión cerrada	
Descripción	Se comprueba que una sesión inactiva lo está
Entrada	Token de autenticación de una sesión inexistente
Resultado esperado	Falso

Tabla 23.86: Prueba unitaria de la API: Comprobación de sesión cerrada

Comprobación de tuplas de sesión existentes	
Descripción	Se comprueba que una tupla de tokens de sesión son válidos
Entrada	Tokens de autenticación y refresco de una sesión existente
Resultado esperado	Verdadero

Tabla 23.87: Prueba unitaria de la API: Comprobación de tuplas de sesión existentes

Comprobación de tuplas de sesión inexistentes	
Descripción	Se comprueba que una tupla de tokens de sesión relacionados e inactivos son inválidos
Entrada	Tokens de autenticación y refresco de una sesión inexistente
Resultado esperado	Falso

Tabla 23.88: Prueba unitaria de la API: Comprobación de tuplas de sesión inexistentes

Comprobación de tuplas de sesión inválidas	
Descripción	Se comprueba que una tupla de tokens de sesión no relacionados son inválidos
Entrada	Tokens de autenticación y refresco no relacionados
Resultado esperado	Falso

Tabla 23.89: Prueba unitaria de la API: Comprobación de tuplas de sesión inválidas

23.1.3.6. Servicio de tareas

Creación de una tarea	
Descripción	Se persiste una tarea válida en la base de datos
Entrada	Tarea válida
Resultado esperado	Tarea persistida en la base de datos Retorno de la tarea creada

Tabla 23.90: Prueba unitaria de la API: Creación de una tarea

Creación de una tarea inválida	
Descripción	Se persiste una tarea inválida en la base de datos
Entrada	Tarea sin alguna propiedad obligatoria
Resultado esperado	HttpError con BadRequest

Tabla 23.91: Prueba unitaria de la API: Creación de una tarea inválida

Eliminación de una tarea	
Descripción	Se elimina una tarea en la base de datos
Entrada	ID de la tarea
Resultado esperado	Tarea eliminada de la base de datos

Tabla 23.92: Prueba unitaria de la API: Eliminación de una tarea

Actualización de una tarea	
Descripción	Se actualiza una tarea con datos válidos en la base de datos
Entrada	Tarea válida
Resultado esperado	Tarea actualizada en la base de datos Retorno de la tarea creada

Tabla 23.93: Prueba unitaria de la API: Actualización de una tarea

Actualización de una tarea no existente	
Descripción	Se actualiza una tarea no existente la base de datos
Entrada	Tarea inválida
Resultado esperado	Error fatal

Tabla 23.94: Prueba unitaria de la API: Actualización de una tarea no existente

Recuperación de tareas relevantes	
Descripción	Se recuperan las tareas relevantes sin más especificación
Entrada	Habitación de las tareas a recuperar
Resultado esperado	Lista de tareas esperadas

Tabla 23.95: Prueba unitaria de la API: Recuperación de tareas relevantes

Recuperación de tareas relevantes de edad especificada	
Descripción	Se recuperan las tareas relevantes con una edad máxima especificada
Entrada	Habitación de las tareas a recuperar y edad máxima
Resultado esperado	Lista de tareas esperadas

Tabla 23.96: Prueba unitaria de la API: Recuperación de tareas relevantes de edad especificada

Comprobación de la sala de una tarea	
Descripción	Se comprueba si una tarea pertenece a una sala específica
Entrada	ID de la tarea y sala de la tarea ID de la tarea y sala errónea
Resultado esperado	Verdadero o falso según sea correcto que pertenezca a la sala

Tabla 23.97: Prueba unitaria de la API: Comprobación de la sala de una tarea

23.1.3.7. Servicio de tokens

Generación de tokens de sesión	
Descripción	Se crean los tokens de una sesión
Entrada	Clave del token
Resultado esperado	Tokens de sesión de la clave proporcionada

Tabla 23.98: Prueba unitaria de la API: Generación de tokens de sesión

Refresco de tokens de sesión	
Descripción	Se refresco los tokens de una sesión válida
Entrada	Token de autenticación y de refresco
Resultado esperado	Nueva tupla de sesión

Tabla 23.99: Prueba unitaria de la API: Refresco de tokens de sesión

Refresco de tokens de sesión inválida	
Descripción	Se refresco los tokens de una sesión inválida
Entrada	Token de refresco inactivo Token de refresco expirado Token de refresco incorrecto
Resultado esperado	Error

Tabla 23.100: Prueba unitaria de la API: Refresco de tokens de sesión inválida

Generación de tokens de vinculación	
Descripción	Se crean un token de vinculación
Entrada	ID del usuario
Resultado esperado	Token de vinculación del usuario

Tabla 23.101: Prueba unitaria de la API: Generación de tokens de vinculación

Decodificación de tokens de vinculación	
Descripción	Se decodifica un token de vinculación
Entrada	Token de vinculación
Resultado esperado	ID del usuario

Tabla 23.102: Prueba unitaria de la API: Decodificación de tokens de vinculación

Decodificación de tokens de vinculación inválidos	
Descripción	Se decodifica un token de vinculación inválidos
Entrada	Token de vinculación expirado Token de vinculación incorrecto
Resultado esperado	Error

Tabla 23.103: Prueba unitaria de la API: Decodificación de tokens de vinculación inválidos

Comprobación de tuplas de tokens	
Descripción	Se comprueba si una tupla de dos tokens es válida
Entrada	Tokens a comprobar
Resultado esperado	Verdadero y falso según sea válidos

Tabla 23.104: Prueba unitaria de la API: Comprobación de tuplas de tokens

23.1.3.8. Servicio de usuarios

Recuperación de un usuario no existente por su ID de Google	
Descripción	Se recupera un usuario no existente con su ID de Google
Entrada	ID de Google de un usuario no existente
Resultado esperado	Usuario nuevo y asociado a la ID

Tabla 23.105: Prueba unitaria de la API: Recuperación de un usuario no existente por su ID de Google

Recuperación de un usuario existente por su ID de Google	
Descripción	Se recupera un usuario con su ID de Google
Entrada	ID de Google de un usuario
Resultado esperado	Usuario asociado a la ID

Tabla 23.106: Prueba unitaria de la API: Recuperación de un usuario existente por su ID de Google

Actualización de un usuario	
Descripción	Se actualiza la información de un usuario
Entrada	Datos de usuario válidos
Resultado esperado	Actualización del usuario en la base de datos Retorno del usuario actualizado

Tabla 23.107: Prueba unitaria de la API: Actualización de un usuario

Creación de vínculo	
Descripción	Se crea un vínculo entre dos usuarios
Entrada	ID del Paciente y el Cuidador a vincular
Resultado esperado	Creación del vínculo entre los usuarios

Tabla 23.108: Prueba unitaria de la API: Creación de vínculo

Creación de vínculo inválida	
Descripción	Se intenta crear un vínculo inválido
Entrada	ID de dos Pacientes ID de dos Cuidadores ID de usuario sin rol ID de Paciente con el máximo de vínculos ID de Cuidador ya vinculado
Resultado esperado	Error con el mensaje tocante

Tabla 23.109: Prueba unitaria de la API: Creación de vínculo inválida

Eliminación de vínculo	
Descripción	Se elimina un vínculo entre dos usuarios
Entrada	ID de un Paciente y un Cuidador vinculados
Resultado esperado	Eliminación del vínculo

Tabla 23.110: Prueba unitaria de la API: Eliminación de vínculo

Recuperación de Paciente vinculado de un Cuidador	
Descripción	Se recupera el Paciente vinculado de un Cuidador
Entrada	ID de un Cuidador vinculado
Resultado esperado	Paciente vinculado del Cuidador

Tabla 23.111: Prueba unitaria de la API: Recuperación de Paciente vinculado de un Cuidador

Recuperación de Paciente vinculado de un Cuidador no vinculado	
Descripción	Se intenta recuperar el Paciente vinculado de un Cuidador no vinculado
Entrada	ID de un Cuidador no vinculado
Resultado esperado	NULL

Tabla 23.112: Prueba unitaria de la API: Recuperación de Paciente vinculado de un Cuidador no vinculado

Recuperación de Paciente vinculado inválida	
Descripción	Se intenta recuperar el Paciente vinculado de un no Cuidador
Entrada	ID de un usuario no Cuidador
Resultado esperado	Error

Tabla 23.113: Prueba unitaria de la API: Recuperación de Paciente vinculado inválida

Recuperación de vínculos	
Descripción	Se recuperan los vínculos de un usuario
Entrada	ID de un usuario ID de un usuario no vinculado
Resultado esperado	Lista de usuarios asociados al usuario

Tabla 23.114: Prueba unitaria de la API: Recuperación de vínculos

Recuperación de rol de un usuario	
Descripción	Se recupera el rol de un usuario
Entrada	ID de un usuario
Resultado esperado	Rol del usuario

Tabla 23.115: Prueba unitaria de la API: Recuperación de rol de un usuario

23.2. Pruebas de integración

23.2.1. Aplicación móvil

23.2.1.1. Registro

Registro de un Paciente en la aplicación	
Descripción	Realizar el proceso de registro de una cuenta nueva en la aplicación
Caso de uso	CU1. Registro
Entrada	Pulsar en *Iniciar sesión* Selección de cuenta no registrada Introducir nombre Pulsar en *Avanzar* Seleccionar rol Paciente Pulsar en *Avanzar* Rellenar datos de contacto Pulsar en *Avanzar* Aserción de datos Pulsar en *Finalizar* Aserción de la tarjeta de Paciente
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.116: Prueba de integración de la aplicación: Registro de un Paciente en la aplicación

Registro de un Cuidador en la aplicación	
Descripción	Realizar el proceso de registro de una cuenta nueva en la aplicación
Caso de uso	CU1. Registro
Entrada	Pulsar en *Iniciar sesión* Selección de cuenta no registrada Introducir nombre Pulsar en *Avanzar* Seleccionar rol Cuidador Pulsar en *Avanzar* Rellenar datos de contacto Pulsar en *Avanzar* Aserción de datos Pulsar en *Finalizar* Aserción de presencia del botón *Forjar vínculo*
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.117: Prueba de integración de la aplicación: Registro de un Cuidador en la aplicación

23.2.1.2. Inicio y cierre de sesión

Inicio de sesión en la aplicación	
Descripción	Realizar el proceso de inicio de sesión en la aplicación
Caso de uso	Especificación del CU2. Iniciar sesión
Entrada	Pulsar en *Iniciar sesión* Selección de cuenta de Paciente ya registrada Aserción de la tarjeta de Paciente Pulsar en *Ajustes* Pulsar en *Cerrar sesión* Aserción de la presencia del botón *Iniciar sesión*
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.118: Prueba de integración de la aplicación: Inicio de sesión en la aplicación

23.2.1.3. Vinculación

Mostrar código de vinculación en la aplicación	
Descripción	Mostrar el código de vinculación de un Paciente en la aplicación
Caso de uso	Especificación del CU3.2. Vincular un Paciente
Entrada	Iniciar sesión con un Paciente no vinculado Navegar a *Vínculos* Pulsar en *Añadir vínculo* Aserción de presencia de código QR en pantalla
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.119: Prueba de integración de la aplicación: Mostrar código de vinculación en la aplicación

Desplegar escáner en la aplicación	
Descripción	Abrir el escáner en la aplicación
Caso de uso	Especificación del CU3.1. Vincular un Cuidador
Entrada	Iniciar sesión con un Cuidador no vinculado Pulsar en *Forjar vínculo* Pulsar en *Conceder permiso* Aserción de apertura del escáner
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.120: Prueba de integración de la aplicación: Desplegar escáner en la aplicación

Eliminación de vínculo de un Paciente	
Descripción	Eliminar un vínculo activo de un Paciente
Caso de uso	Especificación del CU3.3. Desvincularse
Entrada	Iniciar sesión con un Paciente vinculado Pulsar en *Vínculos* Mantener pulsado en un vínculo Confirmar la eliminación Aserción de desaparición del vínculo Cambiar a cuenta vinculada Aserción de desaparición del vínculo
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.121: Prueba de integración de la aplicación: Eliminación de vínculo de un Paciente

Eliminación de vínculo de un Cuidador	
Descripción	Eliminar un vínculo activo de un Cuidador
Caso de uso	Especificación del CU3.3. Desvincularse
Entrada	Iniciar sesión con un Cuidador vinculado Pulsar en *Ajustes* Pulsar en *Eliminar vínculo* Confirmar la eliminación Aserción de desaparición del vínculo Cambiar a cuenta vinculada Pulsar en *Vínculos* Aserción de desaparición del vínculo
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.122: Prueba de integración de la aplicación: Eliminación de vínculo de un Cuidador

23.2.1.4. Compartir ubicación

Compartir ubicación en la aplicación	
Descripción	Compartir la ubicación de un usuario en la aplicación
Caso de uso	Especificación del CU4.1. Compartir ubicación del usuario
Entrada	Iniciar sesión con un usuario vinculado Navegar a *Localización* Pulsar en *Conceder permiso* Aserción de la localización Desplazar GPS Aserción de la localización
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.123: Prueba de integración de la aplicación: Compartir ubicación en la aplicación

23.2.1.5. Gestionar tareas

Gestionar las tareas desde Tareas	
Descripción	Probar todas las opciones de gestión de tareas en la aplicación
Caso de uso	Especificación del CU5.1. Listar tareas, Especificación del CU5.2. Crear tarea, Especificación del CU5.3. Marcar tarea como hecha, Especificación del CU5.4. Desmarcar tarea hecha, Especificación del CU5.5. Eliminar tarea
Entrada	Iniciar sesión con un usuario vinculado Navegar a *Tareas* Pulsar en *Crear tarea* Introducir título y descripción Pulsar en *Confirmar* Aserción de la presencia de la tarea Cambiar a cuenta vinculada Navegar a *Tareas* Aserción de la presencia de la tarea Marcar tarea como hecha Aserción del estado de la tarea Cambiar a cuenta inicial Navegar a *Tareas* Aserción del estado de la tarea Marcar tarea como no hecha Aserción del estado de la tarea Pulsar en el botón de eliminar tarea Confirmar eliminación Aserción de la ausencia de la tarea Cambiar a cuenta inicial Navegar a *Tareas* Aserción de la ausencia de la tarea
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.124: Prueba de integración de la aplicación: Gestionar las tareas desde Tareas

Gestionar las tareas desde el Feed	
Descripción	Probar todas las opciones de gestión de tareas desde el Feed de la aplicación
Caso de uso	Especificación del CU5.1. Listar tareas, Especificación del CU5.2. Crear tarea, Especificación del CU5.3. Marcar tarea como hecha, Especificación del CU5.4. Desmarcar tarea hecha, Especificación del CU5.5. Eliminar tarea
Entrada	Iniciar sesión con un usuario vinculado Navegar a *Feed* Pulsar en *Modo tarea* Introducir título y descripción Pulsar en *Enviar* Aserción del listado de la tarea Cambiar a cuenta vinculada Navegar a *Feed* Aserción de la presencia de la tarea Marcar tarea como hecha Aserción del estado de la tarea Cambiar a cuenta inicial Navegar a *Feed* Aserción del estado de la tarea Marcar tarea como no hecha Aserción del estado de la tarea Mantener presionada la tarea Confirmar eliminación Aserción de la ausencia de la tarea Cambiar a cuenta inicial Navegar a *Feed* Aserción de la ausencia de la tarea
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.125: Prueba de integración de la aplicación: Gestionar las tareas desde el Feed

23.2.1.6. Envío y recepción de mensaje

Enviar y recibir mensajes en la aplicación	
Descripción	Probar el sistema de mensajería de la aplicación
Caso de uso	Especificación del CU6.2. Enviar mensajes
Entrada	Iniciar sesión con un usuario vinculado Navegar a *Feed* Introducir un mensaje Pulsar en *Enviar* Aserción del listado del mensaje Cambiar a cuenta vinculada Navegar a *Feed* Aserción de la del mensaje Introducir un mensaje Pulsar en *Enviar* Aserción del listado del mensaje Cambiar a cuenta vinculada Navegar a *Feed* Aserción de la del mensaje
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.126: Prueba de integración de la aplicación: Enviar y recibir mensajes en la aplicación

23.2.1.7. Gestionar notificaciones

Gestionar notificaciones en la aplicación	
Descripción	Probar las notificaciones de la aplicación
Caso de uso	Especificación del CU8.1 Consultar notificaciones no leídas
Entrada	Iniciar sesión con un usuario vinculado Crear tres tareas Cambiar a cuenta vinculada Aserción del número de notificaciones Abrir *Notificaciones* Aserción de las notificaciones Marcar una como leída Aserción de la ausencia de la notificación Cerrar *Notificaciones* Aserción del número de notificaciones Abrir *Notificaciones* Marcar todas como leídas Aserción de la ausencia de las notificaciones Cerrar *Notificaciones* Aserción del número de notificaciones
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.127: Prueba de integración de la aplicación: Gestionar notificaciones en la aplicación

23.2.1.8. Consultar vínculos

Consultar los vínculos en la aplicación	
Descripción	Consultar los vínculos en la aplicación
Caso de uso	Especificación del CU7.1. Consultar Cuidadores asociados
Entrada	Iniciar sesión con un Paciente con dos vínculos Abrir *Vínculos* Aserción de los vínculos (dos) Cambiar a cuenta vinculada Abrir *Vínculos* Aserción de los vínculos (uno)
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.128: Prueba de integración de la aplicación: Consultar los vínculos en la aplicación

23.2.1.9. Consultar paciente

Consultar paciente en la aplicación	
Descripción	Realizar el proceso de inicio de sesión en la aplicación
Caso de uso	Especificación del CU7.2. Consultar Paciente asociado
Entrada	Iniciar sesión con Cuidador vinculado Aserción de la tarjeta de Paciente
Resultado esperado	Debe completarse satisfactoriamente sin errores

Tabla 23.129: Prueba de integración de la aplicación: Consultar paciente en la aplicación

23.2.2. API

23.2.2.1. Autenticación

Registro de usuario correcto	
Descripción	Petición a GET /auth/session correcta de un usuario nuevo
Entrada	Petición con googleId no registrado
Resultado esperado	Respuesta con un usuario nuevo y una sesión nueva

Tabla 23.130: Prueba de integración de la API: Registro de usuario correcto

Inicio de sesión correcto	
Descripción	Petición a GET /auth/session correcta de un usuario ya existente
Entrada	Petición con googleId registrado
Resultado esperado	Respuesta con el usuario y una sesión nueva

Tabla 23.131: Prueba de integración de la API: Inicio de sesión correcto

Inicio de sesión incorrecto	
Descripción	Petición a GET /auth/session incorrecta
Entrada	Petición sin token
Resultado esperado	Respuesta con estado 404: Not Found

Tabla 23.132: Prueba de integración de la API: Inicio de sesión incorrecto

Cierre de sesión correcto	
Descripción	Petición a DELETE /auth/session de un usuario autenticado con su token de sesión
Entrada	Petición con token de autenticación del propio usuario
Resultado esperado	Eliminación de la sesión en la base de datos Respuesta sin contenido

Tabla 23.133: Prueba de integración de la API: Cierre de sesión correcto

Cierre de sesión incorrecto	
Descripción	Petición a DELETE /auth/session de un usuario con un token de sesión diferente al suyo
Entrada	Petición con token de autenticación de otro usuario
Resultado esperado	Respuesta con estado 403: Forbidden

Tabla 23.134: Prueba de integración de la API: Cierre de sesión incorrecto

Refresco de sesión correcto	
Descripción	Petición a GET /auth/refresh correcta
Entrada	Petición con tokens válidos
Resultado esperado	Respuesta con con una nueva sesión

Tabla 23.135: Prueba de integración de la API: Refresco de sesión correcto

Refresco de sesión incorrecto	
Descripción	Petición a GET /auth/refresh incorrecta
Entrada	Petición con tokens inválidos
Resultado esperado	Respuesta con estado 401: Unauthorized

Tabla 23.136: Prueba de integración de la API: Refresco de sesión incorrecto

23.2.2.2. Localización

Conexión a la sala de localización	
Descripción	Evento location:share de un usuario
Entrada	ID y nombre del usuario
Resultado esperado	Unión satisfactoria a la sala de localización

Tabla 23.137: Prueba de integración de la API: Conexión a la sala de localización

Desconexión de la sala de localización	
Descripción	Evento location:stop de un usuario conectado
Entrada	ID y nombre del usuario
Resultado esperado	Abandono satisfactorio de la sala de localización

Tabla 23.138: Prueba de integración de la API: Desconexión de la sala de localización

Actualización de ubicación	
Descripción	Evento location:update de un usuario
Entrada	Ubicación actualizada
Resultado esperado	Reenvío de la ubicación

Tabla 23.139: Prueba de integración de la API: Actualización de ubicación

23.2.2.3. Mensajes

Recuperación de mensajes de Pacientes	
Descripción	Petición a GET /feed/messages correcta de un Paciente
Entrada	Petición sin página y usuario Paciente
Resultado esperado	Respuesta con los mensajes esperados

Tabla 23.140: Prueba de integración de la API: Recuperación de mensajes de Pacientes

Recuperación de mensajes de Cuidadores correcta	
Descripción	Petición a GET /feed/messages correcta de un Cuidador
Entrada	Petición sin página y usuario Cuidador
Resultado esperado	Respuesta con los mensajes esperados

Tabla 23.141: Prueba de integración de la API: Recuperación de mensajes de Cuidadores correcta

Recuperación de páginas de mensajes concreta	
Descripción	Petición a GET /feed/messages correcta con una página específica
Entrada	Petición con página y usuario Paciente
Resultado esperado	Respuesta con los mensajes esperados

Tabla 23.142: Prueba de integración de la API: Recuperación de páginas de mensajes concreta

Recuperación de mensajes con página incorrecta	
Descripción	Petición a GET /feed/messages con una página inválida
Entrada	Petición con página -1 y usuario Paciente
Resultado esperado	Respuesta con los mensajes de la primera página

Tabla 23.143: Prueba de integración de la API: Recuperación de mensajes con página incorrecta

Recuperación de mensajes con usuario incompleto	
Descripción	Petición a GET /feed/messages de un usuario sin rol
Entrada	Petición sin página y usuario Blank
Resultado esperado	Respuesta con estado 404: Bad Request

Tabla 23.144: Prueba de integración de la API: Recuperación de mensajes con usuario incompleto

Recuperación de mensajes de Cuidador incorrecta	
Descripción	Petición a GET /feed/messages de un Cuidador sin vínculo
Entrada	Petición sin página y usuario Cuidador no vinculado
Resultado esperado	Respuesta con estado 404: Bad Request

Tabla 23.145: Prueba de integración de la API: Recuperación de mensajes de Cuidador incorrecta

Conexión a la sala de mensajería	
Descripción	Evento feed:join de un usuario
Entrada	ID y nombre del usuario
Resultado esperado	Unión satisfactoria a la sala de mensajería

Tabla 23.146: Prueba de integración de la API: Conexión a la sala de mensajería

Desconexión de la sala de mensajería	
Descripción	Evento feed:leave de un usuario conectado
Entrada	ID y nombre del usuario
Resultado esperado	Abandono satisfactorio de la sala de mensajería

Tabla 23.147: Prueba de integración de la API: Desconexión de la sala de mensajería

Envío de un mensaje de texto	
Descripción	Evento feed:send de un mensaje de texto
Entrada	Mensaje enviado
Resultado esperado	Almacenamiento del mensaje en la base de datos Reenvío del mensaje

Tabla 23.148: Prueba de integración de la API: Envío de un mensaje de texto

Envío de una tarea	
Descripción	Evento feed:send de una tarea
Entrada	Tarea enviada
Resultado esperado	Almacenamiento de la tarea en la base de datos Reenvío de la tarea

Tabla 23.149: Prueba de integración de la API: Envío de una tarea

23.2.2.4. Notificaciones

Marcar una notificación válida como leída	
Descripción	Petición a POST /feed/notifications/:/read con una notificación válida
Entrada	Petición con notificación no leída
Resultado esperado	La notificación marcados como leída por el usuario Respuesta sin contenido

Tabla 23.150: Prueba de integración de la API: Marcar una notificación válida como leída

Marcar una notificación inválida como leída	
Descripción	Petición a POST /feed/notifications/:/read con una notificación inválida
Entrada	Petición con notificación ya leída Petición sin notificación
Resultado esperado	Respuesta con estado 404: Bad Request

Tabla 23.151: Prueba de integración de la API: Marcar una notificación inválida como leída

Marcar todas las notificaciones como leídas	
Descripción	Petición a POST /feed/notifications/read de un usuario
Entrada	Petición válida
Resultado esperado	Todas las notificaciones del usuario marcados como leídas Respuesta sin contenido

Tabla 23.152: Prueba de integración de la API: Marcar todas las notificaciones como leídas

Recuperación de notificaciones no leídas por defecto	
Descripción	Petición a GET /feed/notifications de un usuario
Entrada	Petición válida sin edad máxima
Resultado esperado	Respuesta con las notificaciones esperadas

Tabla 23.153: Prueba de integración de la API: Recuperación de notificaciones no leídas por defecto

Recuperación de notificaciones no leídas de edad especificada	
Descripción	Petición a GET /feed/notifications de un usuario
Entrada	Petición válida con edad máxima especificada
Resultado esperado	Respuesta con las notificaciones esperadas

Tabla 23.154: Prueba de integración de la API: Recuperación de notificaciones no leídas de edad especificada

23.2.2.5. Tareas

Creación de una tarea	
Descripción	Petición a POST /task con una tarea válida
Entrada	Petición con tarea válida
Resultado esperado	Respuesta con la tarea creada

Tabla 23.155: Prueba de integración de la API: Creación de una tarea

Creación de una tarea inválida	
Descripción	Petición a POST /task con una tarea inválida
Entrada	Petición con tarea sin campo obligatorio
Resultado esperado	Respuesta con estado 404: Bad Request

Tabla 23.156: Prueba de integración de la API: Creación de una tarea inválida

Eliminación de una tarea válida	
Descripción	Petición a DELETE /task de una tarea válida
Entrada	Petición con tarea válida
Resultado esperado	Respuesta con las notificaciones esperadas

Tabla 23.157: Prueba de integración de la API: Recuperación de notificaciones no leídas de edad especificada

Eliminación de una tarea inválida	
Descripción	Petición a DELETE /task de una tarea inválida
Entrada	Petición con tarea no eliminable por el usuario Petición sin tarea
Resultado esperado	Respuesta con estado 404: Bad Request

Tabla 23.158: Prueba de integración de la API: Eliminación de una tarea inválida

Marcar tarea como hecha	
Descripción	Petición a POST /task/read de una tarea no hecha
Entrada	Petición con tarea no hecha válida
Resultado esperado	La tarea se marca como hecha
	Respuesta sin contenido

Tabla 23.159: Prueba de integración de la API: Marcar tarea como hecha

Marcar tarea como no hecha	
Descripción	Petición a DELETE /task/read de una tarea hecha
Entrada	Petición con tarea hecha válida
Resultado esperado	La tarea se marca como no hecha
	Respuesta sin contenido

Tabla 23.160: Prueba de integración de la API: Marcar tarea como no hecha

Marcar tarea hecha como hecha	
Descripción	Petición a POST /task/read de una tarea hecha
Entrada	Petición con tarea hecha válida
Resultado esperado	La tarea se marca como hecha
	Respuesta sin contenido

Tabla 23.161: Prueba de integración de la API: Marcar tarea hecha como hecha

Marcar tarea no hecha como no hecha	
Descripción	Petición a DELETE /task/read de una tarea no hecha
Entrada	Petición con tarea no hecha válida
Resultado esperado	La tarea se marca como no hecha
	Respuesta sin contenido

Tabla 23.162: Prueba de integración de la API: Marcar tarea no hecha como no hecha

Recuperación de tareas de un Paciente	
Descripción	Petición a GET /task de un Paciente
Entrada	Petición sin edad máximo con usuario Paciente
Resultado esperado	Respuesta con las tareas relevantes del usuario

Tabla 23.163: Prueba de integración de la API: Recuperación de tareas de un Paciente

Recuperación de tareas de un Cuidador	
Descripción	Petición a GET /task de un Cuidador
Entrada	Petición sin edad máxima con usuario Cuidador
Resultado esperado	Respuesta con las tareas relevantes del usuario

Tabla 23.164: Prueba de integración de la API: Recuperación de tareas de un Cuidador

Recuperación de tareas con edad máxima	
Descripción	Petición a GET /task de un Cuidador especificando edad máxima
Entrada	Petición con edad máxima y usuario Cuidador
Resultado esperado	Respuesta con las tareas esperadas

Tabla 23.165: Prueba de integración de la API: Recuperación de tareas con edad máxima

23.2.2.6. Usuarios

Actualización de usuario	
Descripción	Petición a PATCH /user/ de un usuario válido
Entrada	Petición con actualización válida
Resultado esperado	Respuesta con usuario actualizado

Tabla 23.166: Prueba de integración de la API: Actualización de usuario

Actualización de usuario inválida	
Descripción	Petición a PATCH /user/ inválida
Entrada	Petición con ID diferente a la del usuario
Resultado esperado	Respuesta con estado 401: Unauthorized

Tabla 23.167: Prueba de integración de la API: Actualización de usuario inválida

Creación de vínculo	
Descripción	Petición a POST /user/bond válida
Entrada	Petición con token de vinculación correcto
Resultado esperado	Vínculo creado entre los dos usuario Respuesta de éxito

Tabla 23.168: Prueba de integración de la API: Creación de vínculo

Creación de vínculo inválida	
Descripción	Petición a POST /user/bond inválida
Entrada	Petición con token de vinculación incorrecto
Resultado esperado	Respuesta con estado 401: Unauthorized

Tabla 23.169: Prueba de integración de la API: Creación de vínculo inválida

Eliminación de vínculo válida	
Descripción	Petición a DELETE /user/bond válida
Entrada	Petición con ID de usuario vinculado
Resultado esperado	Respuesta con estado 204: No Content

Tabla 23.170: Prueba de integración de la API: Eliminación de vínculo válida

Generación de código de vinculación	
Descripción	Petición a GET /user/bond válida
Entrada	Petición válida
Resultado esperado	Respuesta con el código de vinculación del usuario

Tabla 23.171: Prueba de integración de la API: Generación de código de vinculación

Recuperación de vínculos de un Paciente	
Descripción	Petición a GET /user/bonds de un Paciente vinculado
Entrada	Petición con Paciente vinculado
Resultado esperado	Respuesta con datos de los vínculos del Paciente

Tabla 23.172: Prueba de integración de la API: Recuperación de vínculos de un Paciente

Recuperación de vínculos de un Cuidador	
Descripción	Petición a GET /user/bonds de un Cuidador vinculado
Entrada	Petición con Cuidador vinculado
Resultado esperado	Respuesta con datos de los vínculos del Cuidador

Tabla 23.173: Prueba de integración de la API: Recuperación de vínculos de un Cuidador

Recuperación de vínculos de un usuario incompleto	
Descripción	Petición a GET /user/bonds de un usuario sin rol
Entrada	Petición con usuario Blank
Resultado esperado	Respuesta con estado 404: Bad Request

Tabla 23.174: Prueba de integración de la API: Recuperación de vínculos de un usuario incompleto

Recuperación de Paciente de un Cuidador vinculado	
Descripción	Petición a GET /user/cared de un Cuidador vinculado
Entrada	Petición con Cuidador vinculado
Resultado esperado	Respuesta con datos del Paciente vinculado

Tabla 23.175: Prueba de integración de la API: Recuperación de Paciente de un Cuidador vinculado

Recuperación de Paciente de un Cuidador no vinculado	
Descripción	Petición a GET /user/cared de un Cuidador no vinculado
Entrada	Petición con Cuidador no vinculado
Resultado esperado	Respuesta sin ningún Paciente

Tabla 23.176: Prueba de integración de la API: Recuperación de Paciente de un Cuidador no vinculado

Recuperación de Paciente inválida	
Descripción	Petición a GET /user/cared inválida
Entrada	Petición con un Paciente Petición con un usuario sin rol
Resultado esperado	Respuesta con estado 401: Unauthorized

Tabla 23.177: Prueba de integración de la API: Recuperación de Paciente inválida

23.3. Pruebas de usabilidad y accesibilidad

Para la realización de estas pruebas se contactarán con familias afectadas por Alzheimer que puedan probar la aplicación de forma real y dar su veredicto acerca de ella por medio de cuestionario. Posteriormente, se analizarán las preguntas del cuestionario y se realizarán las mejores necesarias en función de la información recabada.

Existirán dos grupos de control dentro de estas pruebas. Por un lado, tendremos los usuarios que padecen la enfermedad (Pacientes); y por otro lado, los usuarios que se vincularán con estos, los Cuidadores. Debido a los ratios de los dos grupos de población se estima obtener una muestra mayor de Cuidadores que de Pacientes, sin embargo, la opinión de los Pacientes será tomada más en cuenta pues son los usuarios principales.

Las preguntas del cuestionario podrán ser de diferentes tipos: **selección única**, preguntas que requieren la selección de una de las opciones elegidas; **rellenar**, preguntas que solicitan una respuesta redactada, el tamaño puede variar; **rangos**, se response selección un único punto del rango establecido, por ejemplo, un rango de cinco niveles desde *Nada satisfecho* a *Muy satisfecho*; o **elección múltiple**, donde el usuario podrá seleccionar cuantas opciones quiera entre las ofrecidas. Además, las preguntas podrán ser o no opcionales.

Pregunta 1: ¿Eres Paciente o Cuidador?
Tipo: Selección única
Opciones: Paciente, Cuidador
Obligatoria

Pregunta 2: ¿Usas habitualmente aplicaciones móviles?
Tipo: Rango, cinco opciones
Opciones: Desde Nunca a Muy habitualmente
Obligatoria

Pregunta 3: Si eres Paciente, ¿con cuántos Cuidadores te has vinculado?
Tipo: Rango, siete opciones
Opciones: De 0 a 6
Opcional

Pregunta 4: ¿Has tenido dificultades para crear tu cuenta?
Tipo: Selección única
Opciones: Sí, No
Obligatoria

Pregunta 5: En caso afirmativo, ¿cuáles?
Tipo: Rellenar, línea
Opcional

Pregunta 6: ¿Has tenido dificultades para vincularte?
Tipo: Selección única
Opciones: Sí, No
Obligatoria

Pregunta 7: En caso afirmativo, ¿cuáles?
Tipo: Rellenar, línea
Opcional

Pregunta 8: ¿Qué datos de contacto has añadido a tu perfil?
Tipo: Selección múltiple
Opciones: Teléfono principal, Teléfono secundario, Dirección postal, Email **Obligatoria**

Pregunta 9: ¿Echas en falta algún dato de contacto que crees que podría ser útil?
Tipo: Rellenar, línea
Opcional

Pregunta 10: ¿Has contactado a algún otro usuarios a través de los datos de su tarjeta?
Tipo: Selección única
Opciones: Sí, No
Obligatoria

Pregunta 11: En caso negativo, ¿sabías que se podía?
Tipo: Selección única
Opciones: Sí, No
Opcional

Pregunta 12: ¿Qué funciones de la aplicación has utilizado?
Tipo: Selección múltiple
Opciones: Vínculos, Feed, Tareas, Localización, Notificaciones, Ninguna **Obligatoria**

Pregunta 13: ¿Qué funciones añadirías a la aplicación?
Tipo: Rellenar, área de texto
Opcional

Pregunta 14: Valora la utilidad de las siguientes funciones.
Tipo: Rango, cinco opciones
Opciones: Desde Nada a Mucho, y Ninguno
Cuestiones: Vínculos, Feed, Tareas, Localización, Notificaciones **Obligatoria**

Pregunta 15: Valora la facilidad de uso de las siguientes funciones.
Tipo: Rango, cinco opciones
Opciones: Desde Nada a Mucho, y Ninguno
Cuestiones: Contacto de vínculos, Envío de mensajes, Creación de tareas, Cambio de estado de tareas, Eliminación de tareas, Compartir ubicación
Obligatoria

Pregunta 16: Valora la legibilidad de las siguientes pantallas.
Tipo: Rango, cinco opciones
Opciones: Desde Nada a Mucho, y Ninguno
Cuestiones: Inicio, Principal, Vínculos, Feed, Tareas, Localización, Notificaciones, Ajustes
Obligatoria

Pregunta 17: ¿Te ha resultado útil la aplicación?
Tipo: Selección única
Opciones: Sí, No
Obligatoria

Pregunta 18: Selecciona tu grado de satisfacción.
Tipo: Rango, diez opciones
Opciones: De 1 a 10
Obligatoria

Pregunta 19: Opiniones adicionales
Tipo: Rellenar, área de texto
Opcional

Parte V

Implementación

24. Tecnologías

24.1. Lenguajes y entornos

En el desarrollo se han usado una serie de lenguajes y entornos con los que se ha desarrollado el proyecto entero, entendiéndose por esto la aplicación móvil, la API y este documento.

24.1.1. LaTeX

Figura 24.1: Logo de LaTeX

LaTeX es un sistema para la **elaboración de documentos** por medio de texto plano a diferencia del texto formateado que se produce en otros softwares de creación de documentos. Es un software gratuito y el estándar de facto para la comunicación y publicación de documentos científicos a día de hoy. LaTeX ha sido empleado en la **redacción de este documento**.

Para más información: https://www.latex-project.org/

24.1.2. Kotlin

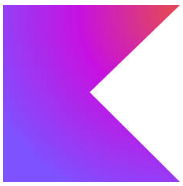


Figura 24.2: Logo de Kotlin

Kotlin es un lenguaje de programación desarrollado por **JetBrains** con aportaciones comunitarias. Es un lenguaje de tipado estático para programación de propósito general multiplataforma.

Entre sus principales características se encuentran la interoperabilidad total con Java y la JVM (y por tanto con Spring o Micronaut), la inferencia de tipos, su capacidad de compilación a código Javascript o que, desde el año 2019, Kotlin es el lenguaje de desarrollo elegido por Google como **primera opción para Android**[5]. Siguiendo esto, Kotlin ha sido empleado en este proyecto en el **desarrollo de la aplicación móvil**.

Para más información: https://kotlinlang.org/

24.1.3. Node

Desarrollado y lanzado por **Ryan Dahl** en 2009, Node.js es un entorno multiplataforma de back-end de Javascript. Es de código abierto y a día es mantenido por la **OpenJS Foundation**, la misma fundación que mantiene jQuery o webpack entre

otros proyectos.

Este entorno permite a los desarrolladores crear aplicaciones en el lado del servidor con un lenguaje inicialmente concebido para ser client-side. De esta forma Node.js permite el **desarrollo de aplicaciones web íntegramente en Javascript**. Tiene una arquitectura basada en eventos con entradas y salidas asíncronas. A día de hoy se encuentra en la versión 17, teniendo Node.js 16 como su versión de largo soporte activa actualmente.

Figura 24.3: Logo de Node.js

Para más información: https://nodejs.org/en/

24.1.4. TypeScript

TypeScript es un superset de Javascript desarrollado y mantenido por **Microsoft**. Su sintaxis contiene todo lo que contiene Javascript con el añadido de tipado estático opcional. El código de TypeScript se **compila a código Javascript**, que es el código finalmente ejecutado. Su premisa nace de dotar de mayor seguridad y validación a los desarrollos con Javascript.

Figura 24.4: Logo de TypeScript

Al cimentarse sobre Javascript, TypeScript es también un lenguaje multiparadigama que permite la programación funcional, imperativa y orientada a objetos. Es compatible con todas las librerías de Javascript aunque no estén escritas en TypeScript, permitiendo incluso dotarlas de las capacidades de este lenguaje por medio del uso de **archivos de definición de tipos** que añade esos tipos sobre código Javascript, sea ajeno o propio. En los últimos años, TypeScript ha sido el lenguaje con mayor crecimiento en uso como demuestra su presencia en GitHub[16].

Para más información: https://www.typescriptlang.org/

24.2. Librerías

24.2.1. Bibliotecas Android

24.2.1.1. AndroidX

AndroidX es un conjunto de bibliotecas desarrollada por **Google** con una serie de herramientas de gran utilidad para el desarrollo en su sistema operativo. Contiene

una serie de dependencias que anteriormente formaban parte del paquete `android` pero que terminaron por ser extraídas del mismo para ser servidas como bibliotecas opcionales bajo el espacio de nombres `androidx`. En este proyecto se han usado varias dependencias de AndroidX:

- **Android KTX**, conjunto de extensiones del lenguaje Kotlin para Android como las corrutinas, utilizadas en el proyecto para la implementación del paralelismo.
- **Appcompat**, proporciona compatibilidad con las nuevas API a versiones anteriores.
- **ConstraintLayout**, otorga un nuevo tipo de layout para las vistas basado en restricciones posicionales. Ha sido el layout más usado en los diseños de las vistas.
- **Lifecycle**. Bibliotecas relacionadas con los ciclos de vida de Android, necesarios en el desarrollo para la programación en paralelo con corrutinas.
  - **Lifecycle KTX**, facilita la programación con ciclos de vida con Kotlin.
  - **LiveData KTX**, permite usar los LiveData de Android con las corrutinas de Kotlin.
  - **ViewModel KTX**, optimiza la programación con ciclos de vida de los ViewModel de Android en Kotlin.
- **Navigation**. Bibliotecas para la navegación entre actividades y fragmentos en Android que se utilizó en la aplicación.
  - **Fragment KTX**, navegación en Kotlin para Fragments de Android.
  - **UI KTX**, navegación en Kotlin para actividades de Android.

Para más información: https://developer.android.com/reference/androidx/packages

24.2.1.2. CodeScanner

Escáner de códigos para Android basado en la librería **Zxing** (biblioteca también instalada para satisfacer la dependencia) de Google. Proporciona componentes y funciones para crear escáneres de código con diversas características ya implementadas como el enfocado automático, el cambio entre vista de retrato y panorámico o entre las cámaras trasera y delantera. Soporta gran variedad de códigos como matrices de datos o códigos de barra, entre otros. Es un proyecto *open source* creado por **Yuriy Budiyev** que cuenta con más de una decena de contribuyentes. Fue utilizado en el desarrollo de la aplicación para la implementación del escáner QR.

Para más información: https://github.com/yuriy-budiyev/code-scanner

24.2.1.3. Coroutine Test

Dependencia para pruebas. Proporciona utilidades para la realización eficiente de pruebas con componentes que usen las corrutinas de Kotlin.

Para más información: https://kotlin.github.io/kotlinx.coroutines/kotlinx-coroutines-test/

24.2.1.4. Espresso

Dependencia para pruebas. Espresso es una librería para la realización de pruebas de interfaces de usuario para Android. Ha sido utilizada en las pruebas de integración de la aplicación móvil.

Para más información: https://developer.android.com/training/testing/espresso

24.2.1.5. JUnit 4

Dependencia para pruebas. El framework de desarrollo de pruebas por antonomasia en Java. Como todas las bibliotecas de Java es compatible con Kotlin y ha sido utilizada en el desarrollo para la realización de las pruebas unitarias de la aplicación móvil. Actualmente existe también JUnit 5, que además es complemente retrocompatible con JUnit 3 y JUnit 4. Sin embargo, la experiencia del equipo de desarrollo con dicha biblioteca es menor y por ello se optó por utilizar JUnit 4. Aún con todo, el cambio a JUnit 5 es sencillo y podría hacerse la actualización con facilidad.

Para más información: https://github.com/junit-team/junit4

24.2.1.6. Material

Véase Apartado 25.3.2. Entre la serie de herramientas que ofrece Material se encuentra una biblioteca de componentes para Android que cumple todas sus guías de estilo. En la aplicación se han seguido las guías de Material para el desarrollo de las interfaces y para facilitar esta tarea se ha utilizado dicha biblioteca para los componentes de las mismas.

Figura 24.5: Logo de Material

Para más información: https://material.io/design

24.2.1.7. Mockk

Dependencia para pruebas. Biblioteca de mocks para Kotlin, utilidad para la que fue empleada durante el desarrollo en las pruebas unitarias de la aplicación móvil.

Para más información: https://mockk.io/

24.2.1.8. MockWebServer

Dependencia para pruebas. La aplicación hace uso de peticiones a la API que deben ser simuladas durante la ejecución de las pruebas. Para ello se ha usado MockWebServer, proporcionada por la misma compañía que desarrolla *Retrofit 2* (24.2.1.11), que imita la comunicación con un servidor web de forma que se puedan especificar las respuestas a las distintas peticiones y así emular el comportamiento esperado sin necesidad de una comunicación real con la API.

Para más información: https://github.com/square/okhttp/tree/master/mockwebserver

24.2.1.9. NetworkResponseAdapter

A la hora de realizar comunicaciones por red se deben realizar una serie de comprobaciones y transformaciones de las respuestas recibidas que produce una gran duplicidad del código, lo cual deriva en la necesidad de implementar un sistema de manejo de estas respuestas. Esta biblioteca desarrollada por Kshitij Chauhan con la colaboración de otros usuarios proporciona un adaptador basado en corrutinas que permite la implementación sencilla de código para la conversión y manejo de las respuestas recibidas de estas comunicaciones.

Para más información: https://github.com/haroldadmin/NetworkResponseAdapter.

24.2.1.10. Play Services

Figura 24.6: Logo de Play Services

Google ofrece una serie de APIs de desarrollo bajo el espacio de nombres de Play Services. En la aplicaicón han sido utilizadas las siguientes:

Auth

La biblioteca para la implementación de autenticaciones según el flujo de OAuth 2.0[13] de Google. Ha sido utilizada para implementar con facilidad la lógica de auten-

ticación sin necesidad de verificar las cuentas al poder confianzar en la verificación de Google. Entre otras cosas los componentes permite la validación de las sesiones con un servidor back-end como ha sido el caso en este desarrollo.

Para más información: https://developers.google.com/identity/sign-in/android/start-integrating

Maps for Android

El kit de desarrollo de Maps para Android ha sido utilizado para agregar un mapa de dicho servicio a la aplicación móvil para utilizar en la función de la geolocalización de usuarios.

Figura 24.7: Logo de Google Maps

Para más información: https://developers.google.com/maps/documentation/android-sdk/overview

Location

El último SDK de PlayServices utilizado en la aplicación va en consonancia con el anterior en el servicio de geolocalización de usuarios. A través de esta API se obtiene en la aplicación móvil la ubicación espacial del dispositivo que posteriormente se enviará al resto de usuarios asociados.

Para más información: https://developers.google.com/maps/documentation/android-sdk/location

24.2.1.11. Retrofit 2

Cliente HTTP con seguridad de tipos para Android de Square que funciona sobre el OkHttpClient de la misma compañía. Ofrece una API de muy sencillo manejo que genera toda la lógica para la comunicación con servidores HTTP a partir de la declaración de interfaces por parte del desarrollador, que únicamente deben especificar el método de la petición, la ruta de misma y el tipo de objeto que será recibido como respuesta. Además, permite la especificación de adaptadores o conversores que gestionen las comunicaciones y generan respuestas que proporcionen directamente una implementación de la clase especificada como tipo de retorno. Ha sido la base de toda la lógica de comunicación con la API REST.

Para más información: https://square.github.io/retrofit/

24.2.1.12. Socket.IO

Cliente para Android de Socket.IO (véase Apartado 24.2.2.15) que permite la especificación y realización de comunicaciones por medio de WebSockets con la WebSocket API.

Para más información: https://socket.io/blog/native-socket-io-and-android/

24.2.2. Paquetes de Node

24.2.2.1. arr-union

Desarrollado por Jon Schlinkert, proporciona facilidades para generar uniones de arrays.

Para más información: https://www.npmjs.com/package/arr-union

24.2.2.2. command-line-args

Librería para la gestión e implementación de la lectura y manejo de los argumentos en línea de comandos empleado en la aplicación para establecer un comando de lanzamiento para especificar uno de los entornos de ejecución de la aplicación.

Para más información: https://www.npmjs.com/package/command-line-args

24.2.2.3. dotenv

Módulo sin dependencias enfocado en la carga de variables de entorno a través de ficheros de extensión `.env` proporcionando acceso a las mismas desde cualquier punto de la aplicación. De esta forma permite la modificación sencilla de las mismas y la carga selectiva según entornos definidos. En la API se ha utilizado para definir entornos de desarrollo, pruebas y producción con valores distintos para definir características como el puerto de despliegue, el tiempo de caducidad de los diferentes tokens o las direcciones de conexión.

Para más información: https://www.npmjs.com/package/dotenv

24.2.2.4. ESLint

Paquete de desarrollo. ESLint es una herramienta de análisis estático de código para la identificación de patrones problemáticos en archivos de código Javascript y

TypeScript. Funciona entorno a una serie de reglas configurables que responden a diferentes tipos y permiten cubrir problemas de calidad y estilo. Junto a esta librería se instalaron también los siguientes paquetes: **eslint-import-resolver-typescript**, para resolver la importación de paquetes con aliases de rutas de TypeScript; **eslint-plugin-import**, para añadir soporte para la sintaxis de importación y exportación de módulos de EcmaScript6[12]; **eslint-plugin-node**, para proporcionar reglas adicionales para el framework de Node.js; **@typescript-eslint/eslint-plugin**, para incluir reglas para bases de código TypeScript; y **@typescript-eslint/parser**, para permitir el análisis del código TypeScript.

Figura 24.8: Logo de ESLint

Durante el desarrollo se especificó la necesidad de pasar de forma limpia el escaneo de ESLint con el archivo de configuración del proyecto antes de que nuevo código se juntase con el código anterior, garantizando que el código principal siempre estuviese libre de estos errores. En un inicio se planteaba añadir esta comprobación al proceso de despliegue de la aplicación pero al final se desestimó por tema de tiempo y por el tamaño del equipo de desarrollo.

Para más información: https://eslint.org/

24.2.2.5. Express

Entorno de trabajo de desarrollo de aplicaciones web para Node. Es un proyecto gratuito y de código libre bajo la licencia MIT. El énfasis de su diseño apunta a la construcción de aplicaciones web dinámicas y APIs. Su extensivo uso dentro del ecosistema de Node.js lo convierte en el framework de facto estándar de esta plataforma. Ha sido usado en el sistema para toda la infraestructura de la API REST.

Entre las características que ofrece se encuentra un sistema de enrutamiento robusto y funcional con apoyo en el uso de middleware; una serie de herramientas de ayuda para la comunicación HTTP como un sistema de caché entre otras cosas; y énfasis en ofrecer alto rendimiento. Está presente en la mayoría de pilas de desarrollo Javascript como son MEAN[1], MERN[2] o MEVN[3].

Para más información: https://expressjs.com/

[1]MongoDB-Express-Angular-Node
[2]MongoDB-Express-React-Node
[3]MongoDB-Express-Vue-Node

24.2.2.6. Google Auth Library

Biblioteca de cliente oficial de Google para la autorización y autenticación con las API de Google y OAuth2.0[11]. Empleada en la API para la validación de las autenticaciones con Google realizadas en la aplicación móvil.

Para más información: https://cloud.google.com/nodejs/docs/reference/google-auth-library/latest

24.2.2.7. Helmet

Middleware para Express que ayuda a aumentar la seguridad de los endpoints de la misma añadiendo automáticamente una serie de cabeceras a las comunicaciones HTTP. En el sistema se utiliza en la versión de producción.

Para más información: https://helmetjs.github.io/

24.2.2.8. http-status-codes

Ofrece los códigos de estado HTTP como constantes para una implementación más sencilla y léxica de estos.

Para más información: https://www.npmjs.com/package/http-status-codes

24.2.2.9. Jest

Paquete para pruebas. Desarrollado y mantenido por Facebook, Jest un entorno de pruebas para Javascript con enfoque en la simplicidad y el soporte de aplicaciones web complejas. Permite el desarrollo de pruebas de todos los niveles sin requerir un gran número de configuraciones como las otras alternativas de pruebas para Javascript. Ha sido el entorno utilizado para probar la Application Programming Interface (API) a todos los niveles.

Figura 24.9: Logo de Jest

Para más información: https://jestjs.io/

24.2.2.10. Jet-Logger

Jet-Logger es la herramienta de registro utilizada a lo largo de toda la API para llevar control del flujo, llamadas o errores en este subsistema. Se puede configurar por medio de variables de entorno o por código, en el desarrollo se empleó la primera

opción. Permite especificar como salida la consola, un archivo o algún otro mecanismo personalizado y las funciones de registro que expone son `info` (información), `imp` (importante), `warn` (advertencia) y `err` (error).

Para más información: https://www.npmjs.com/package/jet-logger

24.2.2.11. jsonwebtoken

Ofrece una interfaz para la implementación de JSON Web Tokens[14] en el sistema. En síntesis, ofrece funciones para firmar un nuevo token, para verificarlo o para descodificarlo. Dichos tokens pueden ser personalizados especificando el algoritmo de codificación, el tiempo de expiración y demás campos del estándar. Al firmar y descodificar estos tokens se usa una clave privada que en el caso de nuestro sistema se define en las variables de entorno.

Para más información: https://jwt.io/libraries

24.2.2.12. MongoDB In-Memory Server

Paquete para pruebas que despliega un servidor en memoria real de MongoDB de forma programática desde el Node. En el desarrollo permitió la realización de pruebas con consultas u operaciones de MongoDB desde un entorno controlado, manipulable e independiente, ahorrando la necesidad de desplegar un entorno o clúster para test.

Para más información: https://github.com/nodkz/mongodb-memory-server

24.2.2.13. Mongoose

Mongoose es la librería por excelencia de Javascript para la gestión de bases de datos de MongoDB. Mongoose proporciona una solución basada en esquemas que permite especificar y dotar de tipos a las entidades de la base de datos, además de otras funciones como la validación automática de las mismas o funciones para la construcción de peticiones válidas entre otras cosas.

Para más información: https://mongoosejs.com/

24.2.2.14. morgan

Proporciona un middleware para registrar las peticiones HTTP recibidas en la aplicación.

Para más información: https://www.npmjs.com/package/morgan

24.2.2.15. Socket.IO

Socket.IO es una biblioteca de Javascript para aplicación web en tiempo real que permite establecer comunicaciones bidireccionales en tiempo real entre clientes web y servidores. Está compuesta por dos partes: primero, la librería para Node que proporciona la lógica para el despliegue en el lado del servidor y sobre la que se cimenta la WebSocket API del sistema; y el segundo, la librería para el lado de los clientes que funciona sobre navegadores o sobre entornos de prueba, como fue el caso en este desarrollo, en el cuál se empleó para realizar los test de integración de la WebSocket API.

Para más información: https://socket.io/

24.2.2.16. SuperTest

Paquete para pruebas con la motivación de proporcionar una abstracción de alto nivel para realizar pruebas con comunicaciones HTTP sin eliminar la opción de realizar pruebas con APIs de bajo nivel. En el desarrollo fue la base de todas las pruebas de integración la REST API.

Para más información: https://www.npmjs.com/package/supertest

24.2.2.17. ts-jest

Paquete para pruebas que ha permitido producir pruebas con Jest utilizando TypeScript.

Para más información: https://kulshekhar.github.io/ts-jest/

24.2.2.18. ts-node-dev

Paquete para el entorno de desarrollo. Ofrece una utilidad para el reinicio rápido y automático del servidor al detectar cambios en los archivos de desarrollo. Es similar a otros paquetes que ofrece utilidades similares como **nodemon**, pero con ventajas para el desarrollo en TypeScript al compartir el proceso de compilación entre reinicios, lo que se traduce en una mejora de velocidad para los reinicios respecto a estas alternativas.

Para más información: https://www.npmjs.com/package/ts-node-dev

24.2.2.19. tsconfig-paths

Paquete para el entorno de desarrollo. TypeScript permite especificar mapeados de rutas para crear alias que reduzcan la navegación de las importaciones de módulos. Sin embargo, estos alias no se traducen correctamente en las ejecuciones con `node` o, como es el caso de este desarrollo, `ts-node`. Este paquete resuelve ese problema y permite dichos lanzamientos con los alias declarados.

Para más información: https://www.npmjs.com/package/tsconfig-paths

24.2.2.20. tscpaths

Librería que ofrece una funcionalidad similar y complementaria a la anterior. Este paquete resuelve los mismos alias reemplazándolos por las rutas absolutas a las que equivalen en tiempo de compilación evitando problemas al ejecutar la versión compilada en Javascript, que será la versión de producción desplegada en el servidor.

Para más información: https://www.npmjs.com/package/tscpaths

24.2.2.21. Typegoose

Aunque Mongoose (24.2.2.13) funciona perfectamente con TypeScript *out-of-the-box*, su implementación no aprovecha como debería las capacacidades del superset y las herramientas que proporciona. Un desarrollo con Mongoose y TypeScript requiere definir el modelo de Mongoose y una interfaz TypeScript aparte para poder analizar su tipo en la compilación. Esto implica que cualquier cambio al modelo requiere también modificar la interfaz, lo que puede provocar errores de consistencia.

Typegoose existe para solucionar el problema, proporcionando la capacidad de decorar interfaces TypeScript de forma que genere los esquemas y modelos de datos a partir de estos, requiriendo con ello una única declaración de la entidad que garantiza la consistencia entre el esquema almacenado y los tipos empleados en el código. Typegoose ha sido por tanto la librería de diseño de las entidades de la base de datos en la API.

Para más información: https://typegoose.github.io/typegoose/

24.2.2.22. TypeScript

Véase Apartado 24.1.4. El uso de TypeScript en un proyecto requiere la instalación de esta librería en el mismo. Además de este paquete se instalaron también como paquetes de desarrollo una serie de librerías para proporcionar las interfaces y tipos TypeScript a paquetes Javascript utilizados en el proyecto que carecían de estas declaraciones. Estas librerías son creadas y publicadas por usuarios en un repositorio ofrecido por Microsoft llamado DefinitelyTyped[4]. Los tipos de dicha base de código son publicados en NPM bajo el espacio de nombres `@types`.

Las bibliotecas de los que se instalaron dependencias de tipos fueron: command-line-args, Express, Jest, jsonwebtoken, morgan, Node y SuperTest

Para más información: https://www.typescriptlang.org/

24.2.2.23. Yarn

Gestor de paquetes desarrollado por **Facebook** para entornos. Node. Es una alternativa a NPM, el gestor por defecto de Node.js que ya viene instalado con el mismo. Ha sido el gestor de dependencias elegido en el desarrollo de la API. La principal ventaja de Yarn y que ha motivado su uso en el proyecto es la mejora de velocidad que ofrece respecto a NPM[17], sobre todo a la hora de hacer instalaciones cacheadas (donde llega a ofrecer una mejora del 200 %) o reinstalaciones (de 28 segundos a menos de 2). De cara a la funcionalidad ambos son muy similares con pequeñas diferencias pero características casi idénticas.

Para más información: https://yarnpkg.com/

24.3. Sistemas operativos

24.3.1. Android

Android es un sistema operativo móvil creado por **Google** y cimentado sobre el kernel de Linux y otros softwares de código libre. Está diseñado principalmente para dispositivos con pantallas táctiles como smartphones o tablets, aunque a día de hoy se encuentra ya extendido a todo tipo de dispositivos como en televisiones inteligentes con AndroidTV o en automóviles con AndroidAuto. Ha sido un sistema operativo de

[4] https://github.com/DefinitelyTyped/DefinitelyTyped

obligado uso en este desarrollo al ser *All For One* un sistema dirigido a dispositivos que lo usan. Se utilizó a lo largo de todo el desarrollo para probar la aplicación móvil desarrollada en diversos entornos y versiones.

Las versiones 8 y 9 fueron empleadas a través de los emuladores de Android Studio, el hardware replicado con los mismos fue el del teléfono móvil Google Pixel 4. La versión 10 fue empleada en un teléfono real de la marca Xiaomi MiA2 tanto por medio de conexión con el IDE como por medio de la instalación de la APK. Por último, la versión más actual en el momento de desarrollo, **Android 11**, fue utilizado en el terminal habitual del desarrollador, un Xiaomi Redmi Note 10 Pro de menos de un año de antigüedad. Como en el caso del smartphone anterior, fue ejecutado con la emulación de la aplicación con el IDE y por medio de la instalación de la APK.

Para más información: https://www.android.com/

24.3.2. Fedora Workstation

Fedora Linux es una distribución Linux desarrollada por la comunidad bajo un proyecto del mismo nombre que el sistema operativo. Su principal patrocinador es la empresa **Red Hat**. Desde el lanzamiento de su versión 30, Fedora cuenta con cinco ediciones disponibles de su sistema operativo con énfasis en diferentes ámbitos: **Workstation**, para ordenadores personales; **Server**, para servidores; **CoreOS**, para la computación en la nube; **Silverblue**, para flujos de trabajo basados en contenedores; y **IoT**, para dispositivos del internet de las cosas.

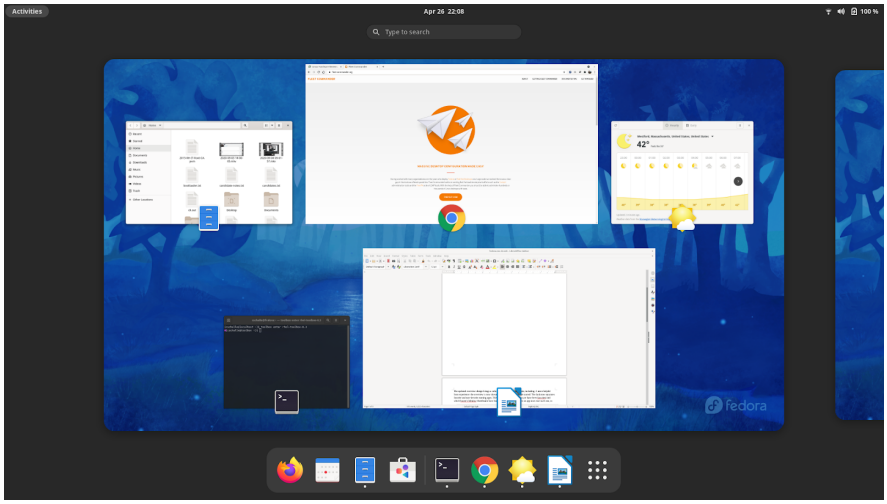


Figura 24.10: Escritorio de Fedora Workstation

En este desarrollo se utilizó **Fedora Workstation 34** en un portátil Lenovo Ideapad 3 de última generación. Workstation utiliza GNOME 40 como escritorio y es un sistema operativo creado con el desarrollo de software en mente, lo que lo convirtió en una gran herramienta para la etapa de implementación, facilitando y agilizando el desarrollo. Fue el sistema operativo principal de desarrollo.

Para más información: https://getfedora.org/en/workstation/

24.3.3. Windows

El último sistema operativo utilizado a lo largo del proceso de implementación por parte del equipo de desarrollo fue **Windows 10**, el sistema operativo lanzado por **Microsoft** en el año 2015. Fue lanzado de forma gratuita para los usuarios de sus versiones anteriores[5] por medio de actualizaciones y de la Windows Store. Actualmente se encuentra dentro de los diez años de soporte garantizados para cualquier sistema operativo de la familia Windows. En marzo de 2020 alcanzó el millardo de usuarios[18], dos años después de superar a Windows 7 en usuarios. A día de hoy ya ha sida lanzado una versión más reciente, **Windows 11**.

Esta plataforma de software se utilizó sobre un PC de configuración personal cuando por motivos de desplazamiento no se podía utilizar el entorno habitual. Para más información: https://www.microsoft.com/es-es/windows/windows-10-specifications

24.3.3.1. WSL2

De cara a ofrecer la mayor consistencia posible en los entornos de desarrollo, durante la creación del sistema con el sistema operativo Windows se hizo uso de la herramienta **Windows Subsystem for Linux** con una imagen de **Ubuntu 20.04 LTS**. Esta herramienta proporciona una capa de compatibilidad con sistemas operativos Linux ejecutada nativamente sobre Windows. De esta forma el desarrollo se pudo llevar a cabo sobre una base UNIX en las dos plataformas y sistemas operativos empleados durante la creación del sistema.

Para más información: https://docs.microsoft.com/en-us/windows/wsl/about

[5] Windows 7, 8 y 8.1

25. Herramientas de desarrollo

25.1. Entornos de desarrollo

25.1.1. Android Studio

IDE oficial para el sistema operativo **Android**. Está construido sobre la plataforma **IntelliJ IDEA** de JetBrains con un enfoque específico para el desarrollo en Android. Entre otras herramientas que ofrece para este objetivo se encuentran: emuladores del sistema operativo, plantillas de actividades Android o un editor de interfaces de usuario del sistema operativo entre otras cosas. Fue la plataforma utilizada durante todo el desarrollo de la aplicación móvil.

Para más información: https://developer.android.com/studio

25.1.2. Overleaf

Editor colaborativo en la nube de **LaTeX** que permite redactar, editar y publicar documentos de diversas índoles con dicha tecnología. Tiene herramientas de revisión de texto, de historial de cambios y de chat entre colaboradores, entre otras cosas. También ofrece una serie de plantillas colgadas por usuarios o por editoriales oficiales, con las cuáles también cuenta con pasarelas para el envío de directo de enlaces o documentos para solicitar la publicación. Fue la plataforma utilizada en gran parte de la redacción de este documento.

Para más información: https://www.overleaf.com/

25.1.3. Vim

Vim es un editor de texto en pantalla gratis y de código libre para sistemas **Unix** creado por **Bram Moolenaar** y lanzado en 1991. Está basado en el editor **vi** con añadidos y funcionalidades extra. Ha sido un herramienta de apoyo en el desarrollo de todos los subsistemas desde la línea de comandos de la plataforma Fedora Workstation.

Figura 25.1: Logo de Vim

Para más información: https://www.vim.org/

25.1.4. Visual Studio Code

Visual Studio Code, también conocido como **VSCode** es un editor de código desarrollador por **Microsoft** para sistemas Windows, Linux y macOS. En el año 2021 fue elegido como el entorno de desarrollo favorito de los usuarios de StackOverflow[19]. Su código está disponible en GitHub de forma pública bajo una licencia MIT. Busca ser un IDE ligero con las herramientas básicas, como depuración, resaltado de sintaxis, refactorización de código o conclusión de código, que pueda ser ampliamente expandido con extensiones disponibles en un *marketplace*; de esta forma es un entorno compatible con prácticamente cualquier lenguaje y tecnología existente.

Figura 25.2: Logo de Visual Studio Code

Es por este motivo que este editor fue ampliamente utilizado durante el desarrollo completo del sistema. VSCode fue la plataforma principal de implementación de la API (TypeScript), de gran parte de la redacción de este documento (LaTeX) y de alguna sección de la aplicación móvil (Kotlin). Además, se utilizó también como interfaz de comunicación con el servidor desplegado en Azure.

Para más información: https://code.visualstudio.com/

25.2. Herramientas

25.2.1. Git

Git es un software gratuito y libre de control de versiones desarrollado por **Linus Torvalds**. Los software de control de versiones permiten monitorizar los cambios realizados a una serie de ficheros y son una de las herramientas principales de los desarrollos de software colaborativos. Se ha utilizado en conjunción de con GitHub (Apartado 25.4.3) para la publicación y gestión de los cambios en el desarrollo del sistema.

Para más información: https://git-scm.com/

25.2.2. GitHub Actions

GitHub Actions es una plataforma de **integración continua y entrega continua** (CI/CD por sus siglas en inglés) que permite la automatización de la compilación, ejecución de pruebas y el despliegue de sistemas. Permite la creación de flujos de

trabajo que construyen y comprueban el código en cada solicitud y el despliegue en producción del resultado de las mezclas de este. Ha sido utilizado en el proyecto para automatizar el despliegue de la API en la nube.

Para más información: https://github.com/features/actions

25.2.3. Insomnia

Aplicación de escritorio multiplataforma enfocada en el diseño y prueba de APIs HTTP. Insomnia ofrece una interfaz de usuario simple para la creación de peticiones personalizadas y fácilmente editables con muchas características avanzadas como facilidades para la autenticación, generación de código o declaración de variables de entorno. Fue utilizada para probar maunalmente la API REST sin necesidad de utilizar la aplicación móvil o de construir una batería de pruebas de integración.

Para más información: https://insomnia.rest/

25.2.4. Mendeley

Herramienta para la gestión de referencias bibliográficas. Permite la importación de documentos, creación de citas y anotaciones, guardado de referencias y su acceso desde cualquier lugar a través de la nube. Está disponible en diversidad de formas, como por ejemplo, de aplicación de escritorio o de extensión de navegador. Ha sido empleada en el proyecto para la creación de la biblioteca de referencias que se usaron en este documento.

Para más información: https://www.mendeley.com/

25.2.5. PlantUML

Proyecto de código abierto para el dibujado de diagramas UML usando descripciones de texto simples y de lectura sencilla. Permite crear diagramas de secuencia, de casos de uso, de clases, de objetos, de actividades, de componentes, de despliegue y de estados. Las imágenes producidas se pueden generar en formatos PNG, SVG y LaTeX. Es compatible con multitud de tecnologías y entornos. En el desarrollo se ha utilizado con el *plugin* de Android Studio para la generación de parte de los diagramas de esta documentación.

Para más información: https://plantuml.com/es/

25.2.6. ProjectLibre

ProjectLibre es un software de gestión de proyectos gratuito y de código abierto. Actualmente está disponible únicamente como aplicación de escritorio, pero ya cuenta con una versión en la nube en camino. El objetivo de este software es el de ofrecer una alternativa gratuita y libre a Microsoft Project. En este desarrollo fue utilizado precisamente con ese fin, con el de reemplazar dicho software con una opción menos privativa con la que generar y gestionar la planificación del proyecto.

Para más información: https://www.projectlibre.com/

25.3. Referencias

25.3.1. Guías para desarrolladores de Android

A la hora de desarrollar la aplicación móvil el principal punto de referencia fue la guía para desarrolladores de Android que ofrece Google. Entre otras cosas ofrece laboratorios de código, cursos, vídeos y ejemplos de multitud de aspectos del desarrollo en este sistema operativo. Esta documentación ha sido de ayuda en el proyecto en aspectos como el aprendizaje de la arquitectura MVVM, del *data-binding* de las vistas o del uso de las corrutinas de Kotlin.

Para más información: https://developer.android.com/guide

25.3.2. Guía de estilo Material Design

Material es un conjunto de guías, componentes y herramientas para favorecer la implementación de interfaces de usuario que sigan las mejores prácticas posibles de diseño. Soportada por una comunidad de código libre, Material es una línea de colaboración entre diseñadores y desarrolladores que facilita la creación de interfaces de usuario impecables. Sus guías son muy completas y visuales, con explicaciones detalladas de los razonamientos detrás de los consejos y ejemplos de aplicaciones que los aplican.

Para más información: https://material.io/design

25.4. Sitios web

25.4.1. Color Tool

Herramienta ofrecida por Material Design (Apartado 25.3.2) para la generación de paletas de colores de una aplicación. El usuario selecciona los colores principales que quiere utilizar y Color Tool le ofrecerá los colores que deben completar su tema siguiendo las directrices de Material Design. También tiene información y muestras del nivel de accesibilidad de la paleta creada. El tema de colores de la aplicaicón fue creada con esta herramienta.

Para más información: https://material.io/resources/color/

25.4.2. Diagrams.net

Antiguamente conocida como **Draw.io**. Es un software de creación de diagramas en líneas. Permite crear gráficos de flujo, diagramas UML y o diagramas de red entre otros muchos tipos. Permite importar y exportar los diagramas en varios almacenamientos en la nube y el trabajo colaborativo. Todos los diagramas de este documento que no fueron creadas con PlantUML lo fueron con la versión web de este software.

Para más información: https://app.diagrams.net/

25.4.3. GitHub

GitHub es un sitio web comprado por **Microsoft** en 2018 por 7,5 millardos de dólares. Es un proveedor de alojamiento en Internet para desarrollos de software y versiones de control que empleen Git. Ofrece toda la funcional de gestión de código y del control de versiones que ofrece Git con una serie de características propias adicionales como GitHub Actions. Ha sido el portal en el que se ha alojado todo el código del proyecto durante su desarrollo.

Figura 25.3: Logo de GitHub

Para más información: https://github.com/

25.4.4. Portal de Azure

Portal dedicado a la gestión de la nube Azure. Permite crear, administrar y supervisar todas las aplicaciones alojadas en el servicio de la nube de **Microsoft**. Aunque existen alternativas para esto como la **Azure CLI** o Visual Studio Code, en este de-

sarrollo ha sido la vía alternativa para la gestión del servidor en el que se desplegó la API.

Para más información: https://azure.microsoft.com/es-es/features/azure-portal/

25.4.5. Portal de MongoDB Atlas

Sitio web ofrecido por MongoDB para al gestión e interacción con su servicio de bases de datos en la nube, MongoDB Atlas. Desde este portal se pueden los diferentes clústers, establecer los requisitos y accesos o acceder a los datos y modificarlos entre otras cosas. Ha sido la vía de gestión de la base de datos durante el desarrollo.

Para más información: https://www.mongodb.com/atlas/database

Figura 25.4: Logo de MongoDB Atlas

26. Obstáculos

Durante la creación del sistema se encontraron una serie de dificultades que supusieron un obstáculo en la implementación y ralentizaron el desarrollo. Aquí se exponen las más importantes.

26.1. Comunicación cifrada

Una de las primeras funciones agregadas al sistema fue el establecimiento de la comunicación entre la API y la aplicación móvil. En esa primera implementación se utilizó un prototipo de cada subsistema que realizase la comunicación más básica posible tanto a través del WebSocket como por medio de HTTP. Esos primeros intercambios resultaron en errores debido a la **falta de encriptación** de las respuestas de la API.

El siguiente paso en el intento de sobrepasar este problema fue el de lanzar Express como servidor HTTPS con certificados generados en la máquina de ejecución local de dicho servidor. De cara a permitir esta comunicación dichos certificados debían ser también copiados en la aplicación móvil para añadirlos a una lista blanca de entidades confiables. Tras preparar todo, la comunicación siguió sin ser posible pues los certificados generados **no contaban con una fuente de confianza** y Android los rechazaba.

Finalmente, y viendo que generar un servidor HTTPS con certificados emitidos por fuentes confiables escapaba al alcance del proyecto se decidió **modificar la configuración de red** de la aplicación y permitir el tráfico de texto plano. Para reducir el peligro de esta decisión al mínimo se realizó dicha permisión únicamente a las direcciones IP locales y del servidor a desplegar. Tras este cambio, que se puede ver en el fichero `network_security_config.xml` se pudo realizar envío de información entre ambos subsistemas.

26.2. Ejecución en dispositivos móviles antes del despliegue

A lo largo del desarrollo era posible realizar la comunicación entre la API y la aplicación móvil durante su ejecución en la misma máquina local por medio de la especificación de la IP `http://10.0.2.2` para conseguir que la aplicación pudiese encontrar al servidor, pues es la dirección que representa a la máquina local sobre la que se ejecuta

el emulador.

Sin embargo, esta dirección existe sólo como enrutamiento en los emuladores de **Android Studio**, pero no en los dispositivos móviles en los que se pueda ejecutar la aplicación, aunque sea a través de la conexión con el IDE. Es por esto, que a lo largo del desarrollo no fue posible probar la aplicación en un dispositivo móvil real hasta que se llevó a cabo el despliegue del servidor en la nube.

26.3. Despliegue del servidor

El despliegue del servidor en la nube no constó de un obstáculo sino que lo fue en sí mismo. Ante la falta de experiencia particular en despliegues de este tipo o en ingeniería DevOps la tarea resultó de **gran dificultad** y consumió una cantidad de tiempo mucho mayor a la esperada inicialmente. Dilatada especialmente por el propio tiempo que llevaba cada despliegue.

De cara a poder lograr este despliegue en la nube fue necesario realizar una serie de cambios en el código del servidor que permitiesen que la compilación, ejecución y lanzamiento fuesen fructíferos. Se encontraron problemas en cada uno de estos pasos, los más resaltables fueron:

- A la hora de compilar se generaron rutas entre ficheros erróneas en la generación del código Javascript a partir de los alias de TypeScript, esto llevó a la necesidad de instalar librerías adicionales que cubriesen este error.
- La ejecución de los archivos compilados también falló puesto que la acción de despliegue generada por Azure intentaba ejecutar el archivo equivocado.
- Problemas al encontrar variables de entorno.
- Incapacidad de conexión con la base de datos por el listado blanco de direcciones IP.

Finalmente, el despliegue fue posible con todos los pasos indicados en el Manual de despliegue (Capítulo 30), pero esta tarea consumió más tiempo del esperado en la Planificación temporal (Capítulo 8).

27. Desarrollo de pruebas

A continuación se ofrece el resultado de las pruebas propuestas en el Capítulo 23. La tabla lista la referencia del caso de prueba en dicha sección, el título del caso de prueba y el resultado de los mismos en el momento de entrega de este documento. Este resultado puede ser **PASA**, si la prueba ha sido implementada y se completa con éxito; **FALLA**, si la prueba ha sido implementada pero no ofrece el resultado esperado; y **NO**, para las pruebas que no han llegado a ser implementadas en el sistema.

27.1. Pruebas unitarias

27.1.1. API

Referencia	Caso	Resultado
23.67	Encabezado de autenticación correcto	PASA
23.68	Encabezado de autenticación faltante	PASA
23.69	Encabezado de autenticación mal formateado	PASA
23.70	Encabezado de autenticación con token inválido	PASA
23.71	Manejo correcto de errores conocidos	PASA
23.72	Manejo correcto de errores desconocidos	PASA
23.73	Creación de un mensaje de texto	PASA
23.74	Creación de un mensaje de texto inválido	PASA
23.75	Recuperación de mensajes	PASA
23.76	Recuperación de mensajes de una página concreta	PASA
23.77	Creación de una notificación	PASA
23.78	Marcado de una notificación con más interesados como leída	PASA
23.79	Marcado de una notificación sin más interesados como leída	PASA
23.80	Marcado de todas las notificaciones como leídas	PASA
23.81	Recuperación de notificaciones	PASA
23.82	Recuperación de notificaciones con edad especificada	PASA

Tabla 27.1: Primera parte de los resultados de las pruebas unitarias de la API

Referencia	Caso	Resultado
23.83	Inicio de sesión	PASA
23.84	Cierre de sesión	PASA
23.85	Comprobación de sesión abierta	PASA
23.86	Comprobación de sesión cerrada	PASA
23.87	Comprobación de tuplas de sesión existentes	PASA
23.88	Comprobación de tuplas de sesión inexistentes	PASA
23.89	Comprobación de tuplas de sesión inválidas	PASA
23.90	Creación de una tarea	PASA
23.91	Creación de una tarea inválida	PASA
23.92	Eliminación de una tarea	PASA
23.93	Actualización de una tarea	PASA
23.94	Actualización de una tarea no existente	PASA
23.95	Recuperación de tareas relevantes	PASA
23.96	Recuperación de tareas relevantes de edad especificada	PASA
23.97	Comprobación de la sala de una tarea	PASA
23.98	Generación de tokens de sesión	PASA
23.99	Refresco de tokens de sesión	PASA
23.100	Refresco de tokens de sesión inválida	PASA
23.101	Generación de tokens de vinculación	PASA
23.102	Decodificación de tokens de vinculación	PASA
23.103	Decodificación de tokens de vinculación inválidos	PASA
23.104	Comprobación de tuplas de tokens	PASA
23.105	Recuperación de un usuario no existente por su ID de Google	PASA
23.106	Recuperación de un usuario existente por su ID de Google	PASA
23.107	Actualización de un usuario	PASA
23.108	Creación de vínculo	PASA
23.109	Creación de vínculo inválida	PASA
23.110	Eliminación de vínculos inválida	PASA
23.111	Recuperación de Paciente vinculado de un Cuidador	PASA
23.112	Recuperación de Paciente vinculado de un Cuidador no vinculado	PASA
23.113	Recuperación de Paciente vinculado inválida	PASA
23.114	Recuperación de vínculos	PASA
23.115	Recuperación de rol de un usuario	PASA

Tabla 27.2: Segunda parte de los resultados de las pruebas unitarias de la API

Como se puede comprobar las pruebas unitarias de la API no han podido ofrecer un resultado más satisfactorio. Todos los casos de prueba han sido implementados y sus resultados son exitosos. Además, se ha conseguido una cobertura de código superior al **90 %** en líneas, condiciones, métodos y clases.

27.1.2. Aplicación móvil

Referencia	Caso	Resultado
23.1	Actualizar datos del usuario	NO
23.2	Cerrar la sesión de usuario	NO
23.3	Eliminar el vínculo con el Paciente vinculado	NO
23.4	Añadir un marcador	PASA
23.5	Comprobar la existencia de un marcador	PASA
23.6	Eliminar un marcador	PASA
23.7	Actualizar un marcador	PASA
23.8	Añadir una notificación	NO
23.9	Añadir un conjunto de notificaciones	NO
23.10	Limpiar lista de notificaciones	NO
23.11	Cargar lista de notificaciones	NO
23.12	Eliminar una notificación	NO
23.13	Eliminar todas las notificaciones	NO
23.14	Enviar confirmación de datos de usuario	PASA
23.15	Gestionar resultado de inicio de sesión en Google	PASA
23.16	Gestionar resultado de inicio de sesión en Google erróneo	PASA
23.17	Añadir un nuevo marcador	PASA
23.18	Recibir una actualización de posición de marcador no existente	NO
23.19	Recibir una actualización de posición de marcador ya existente	NO
23.20	Obtener una página de mensajes	PASA
23.21	Enviar una tarea por el feed	NO
23.22	Enviar un mensaje por el feed	NO
23.23	Recibir una actualización de tarea	NO
23.24	Recibir una eliminación de tarea	NO
23.25	Recibir un nuevo mensaje	NO

Tabla 27.3: Primera parte de los resultados de las pruebas unitarias de la aplicación

Referencia	Caso	Resultado
23.26	Confirmar creación de una tarea	PASA
23.27	Eliminación una tarea	PASA
23.28	Eliminación una tarea inválida	PASA
23.29	Marcar tarea como hecha/no hecha	PASA
23.30	Obtener la lista de tareas	PASA
23.31	Eliminar un vínculo	NO
23.32	Obtener un código QR de vinculación	NO
23.33	Obtener los vínculos	PASA
23.34	Actualizar usuario	NO
23.35	Enviar código de vinculación	PASA
23.36	Obtener lista de notificaciones	NO
23.37	Obtener Paciente vinculado	PASA
23.38	Envío de mensaje	NO
23.39	Recuperación de mensajes	PASA
23.40	Recuperación de notificaciones	PASA
23.41	Marcado de notificación como leída	PASA
23.42	Marcado de todas las notificaciones como leídas	PASA
23.43	Inicio de sesión	PASA
23.44	Cierre de sesión	PASA
23.45	Refresco de sesión	PASA
23.46	Recuperación de tareas	PASA
23.47	Guardado de una tarea	PASA
23.48	Eliminación de una tarea	PASA
23.49	Actualización de una tarea	PASA
23.50	Actualización de un usuario	PASA
23.51	Eliminar vinculación	PASA
23.52	Enviar código de vinculación	PASA
23.53	Recuperación de vínculos	PASA
23.54	Recuperación del Paciente vinculado	PASA
23.55	Solicitar código de vinculación	PASA
23.56	Obtención de un servicio	PASA
23.57	Procesar respuesta de inicio de sesión	PASA
23.58	Procesar respuesta de refresco de sesión	PASA
23.59	Procesar respuesta de recuperación de notificaciones	PASA
23.60	Procesar respuesta de recuperación de mensajes	PASA

Tabla 27.4: Segunda parte de los resultados de las pruebas unitarias de la aplicación

Referencia	Caso	Resultado
23.61	Procesar respuesta de recuperación de tareas	PASA
23.62	Procesar respuesta de publicación de tareas	PASA
23.63	Procesar respuesta de actualización de usuario	PASA
23.64	Procesar respuesta de recuperación de Paciente vinculado	PASA
23.65	Procesar respuesta de recuperación de vínculos	PASA
23.66	Procesar respuesta de recuperación de código de vinculación	PASA

Tabla 27.5: Tercera parte de los resultados de las pruebas unitarias de la aplicación

En el desarrollo de pruebas de la aplicación no fueron posibles realizar todas las pruebas que estaban planeadas por limitaciones de tiempo. Dentro de las que sí pudieron implementadas se consiguió un éxito completo. Estas pruebas necesitan mayor trabajo.

27.2. Pruebas de integración

27.2.1. API

Referencia	Caso	Resultado
23.130	Registro de usuario correcto	PASA
23.131	Inicio de sesión correcto	PASA
23.132	Inicio de sesión incorrecto	PASA
23.133	Cierre de sesión correcto	PASA
23.134	Cierre de sesión incorrecto	PASA
23.135	Refresco de sesión correcto	PASA
23.136	Refresco de sesión incorrecto	PASA
23.137	Conexión a la sala de localización	PASA
23.138	Desconexión de la sala de localización	PASA
23.139	Actualización de ubicación	PASA

Tabla 27.6: Primera parte de los resultados de las pruebas de integración de la API

Referencia	Caso	Resultado
23.140	Recuperación de mensajes de Pacientes	PASA
23.141	Recuperación de mensajes de Cuidadores correcta	PASA
23.142	Recuperación de páginas de mensajes concreta	PASA
23.143	Recuperación de mensajes con página incorrecta	PASA
23.144	Recuperación de mensajes con usuario incompleto	PASA
23.145	Recuperación de mensajes de Cuidador incorrecta	PASA
23.146	Conexión a la sala de mensajería	PASA
23.147	Desconexión de la sala de mensajería	PASA
23.148	Envío de un mensaje de texto	PASA
23.149	Envío de una tarea	PASA
23.150	Marcar una notificación válida como leída	PASA
23.151	Marcar una notificación inválida como leída	PASA
23.152	Marcar todas las notificaciones como leídas	PASA
23.153	Recuperación de notificaciones no leídas por defecto	PASA
23.154	Recuperación de notificaciones no leídas de edad especificada	PASA
23.155	Creación de una tarea	PASA
23.156	Creación de una tarea inválida	PASA
23.157	Eliminación de una tarea válida	PASA
23.158	Petición a DELETE /task de una tarea válida	PASA
23.159	Marcar tarea como hecha	PASA
23.160	Marcar tarea como no hecha	PASA
23.161	Marcar tarea hecha como hecha	PASA
23.162	Marcar tarea no hecha como no hecha	PASA
23.163	Recuperación de tareas de un Paciente	PASA
23.164	Recuperación de tareas de un Cuidador	PASA
23.165	Recuperación de tareas con edad máxima	PASA

Tabla 27.7: Segunda parte de los resultados de las pruebas de integración de la API

Referencia	Caso	Resultado
23.166	Actualización de usuario	PASA
23.167	Actualización de usuario inválida	PASA
23.168	Creación de vínculo	PASA
23.169	Creación de vínculo inválida	PASA
23.170	Eliminación de vínculo válida	PASA
23.171	Generación de código de vinculación	PASA
23.172	Recuperación de vínculos de un Paciente	PASA
23.173	Recuperación de vínculos de un Cuidador	PASA
23.174	Recuperación de vínculos de un usuario incompleto	PASA
23.175	Recuperación de Paciente de un Cuidador vinculado	PASA
23.176	Recuperación de Paciente de un Cuidador no vinculado	PASA
23.177	Recuperación de Paciente inválida	PASA

Tabla 27.8: Tercera parte de los resultados de las pruebas de integración de la API

27.2.2. Aplicación móvil

Referencia	Caso	Resultado
23.116	Registro de un Paciente en la aplicación	NO
23.117	Registro de un Cuidador en la aplicación	NO
23.118	Inicio de sesión en la aplicación	NO
23.119	Mostrar código de vinculación en la aplicación	NO
23.120	Desplegar escáner en la aplicación	NO
23.121	Eliminación de vínculo de un Paciente	NO
23.122	Eliminación de vínculo de un Cuidador	NO
23.123	Compartir ubicación en la aplicación	NO
23.124	Gestionar las tareas desde Tareas	NO
23.125	Gestionar las tareas desde el Feed	NO
23.126	Enviar y recibir mensajes en la aplicación	NO
23.127	Gestionar notificaciones en la aplicación	NO
23.128	Consultar los vínculos en la aplicación	NO
23.129	Consultar paciente en la aplicación	NO

Tabla 27.9: Resultados de las pruebas de integración de la aplicación

No se han podido llevar a cabo pruebas de integración en la aplicación. El equipo de desarrollo no ha sido capaz de lanzar la aplicación con Espresso y conseguir realizar el inicio de sesión a través de la autenticación de Google por problemas con las cuentas de usuario. Ha sido el mayor error del proyecto.

27.3. Pruebas usabilidad y accesibilidad

Por desgracia, **no fue posible** llevar a cabo estas pruebas. Todas las asociaciones contactadas rechazaron poder prestar ayuda y sin ellas, y con la situación de pandemia que ha existido durante el desarrollo, ha sido imposible contactar con usuarios potenciales de la aplicación para que respondiese a nuestro formulario de Apartado 23.3.

Parte VI

Manuales

28. Manual de instalación

Puesto que la publicación de la aplicación en un mercado de aplicaciones no entra dentro del alcance del desarrollo, la única instalación posible es por medio de la APK. Este fichero se puede obtener entre los archivos adjuntos a este documento o se puede descargar en el apartado **Releases**[1] del repositorio público de GitHub:

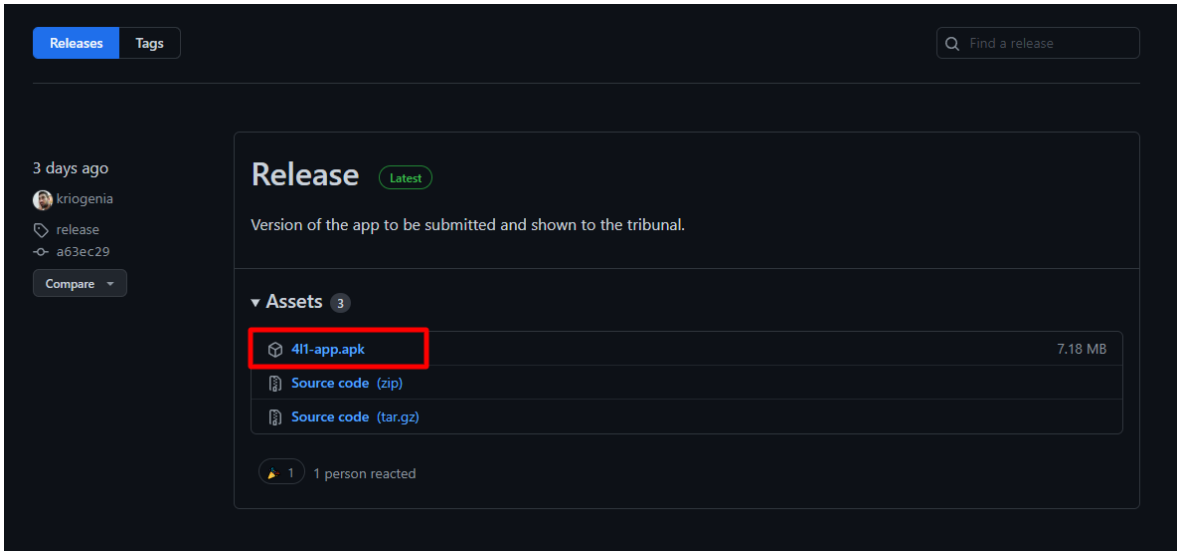


Figura 28.1: Descarga de la APK desde el repositorio

Una vez se tenga el archivo `.apk` este deberá ser transferido al dispositivo móvil en el que se quiera instalar. También es posible descargar la APK desde el propio *smartphone* en el link anterior. Al no ser una aplicación descargada desde un *marketplace* certificado será necesario habilitar la instalación de aplicaciones de orígenes desconocidos en los **Ajustes** del sistema[20].

Tras activar dicho permiso el fichero podrá ser abierto desde el gestor de ficheros del sistema operativo. Si Android pregunta si estás seguro de realizar la instalación, confirma que sí. Cuando la instalación finalice la aplicación ya estará lista para ser usada. *Véase Capítulo 29 Manual de usuario.*

[1] https://github.com/kriogenia/AllForOne-App/releases

29. Manual de usuario

29.1. Inicio de sesión y registro

A la hora de abrir la aplicación la primera pantalla que se muestra es la ilustrada en la Figura 29.1. Esta pantalla consta de un solo botón, pulsarlo desplegará un *pop-up* solicitando una cuenta de Google del dispositivo. Debes seleccionar la cuenta con la que se quiere iniciar la sesión, sea para una nueva cuenta o para una ya existente. Es posible que se solicite la conformidad del usuario para compartir la información de la cuenta con la aplicación, si no se concede esta no podrá funcionar.

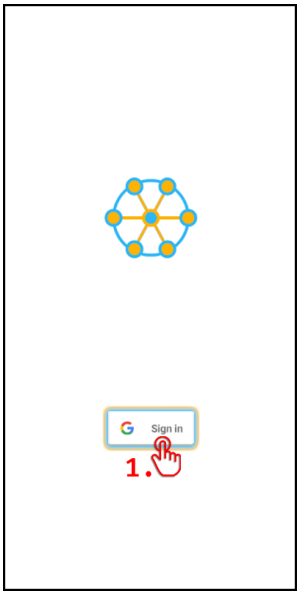


Figura 29.1: Guía de inicio de sesión.

Si la cuenta ya existe, la aplicación te redirigirá a la pantalla principal, desde la cuál se podrán usar las funciones de la aplicación que se pueden consultar en el resto de secciones de este manual. Si la cuenta introducida es nueva en el sistema, la aplicación mostrará una pantalla como la primera que se muestra en la Figura 29.2.

En esta pantalla debes introducir un nombre para identificarte de cara a los usuarios con los que te vincularás. No tiene que ser único en la aplicación ni es necesario que sea tu nombre real, si eres conocido por algún sobrenombre concreto entre la gente con la que compartirás funciones entonces ese apodo será un buena elección. Si entre los usuarios que os vayáis vincular va a haber alguien más que comparta tu mismo nombre, intentad elegir uno que os distinga. Una vez hayas introducido un nombre pulsa en el botón de **Siguiente** y avanzarás a la pantalla de selección de rol.

En esta aplicación existen dos roles: el de **Paciente** y el de **Cuidador**. Todas las funciones giran en torno al paciente. Selecciona aquel rol que represente la clase de papel que tomarás mientras usas la aplicación. Una vez lo hayas hecho vuelve a pulsar en **Siguiente** para rellenar el resto de datos de tu perfil.

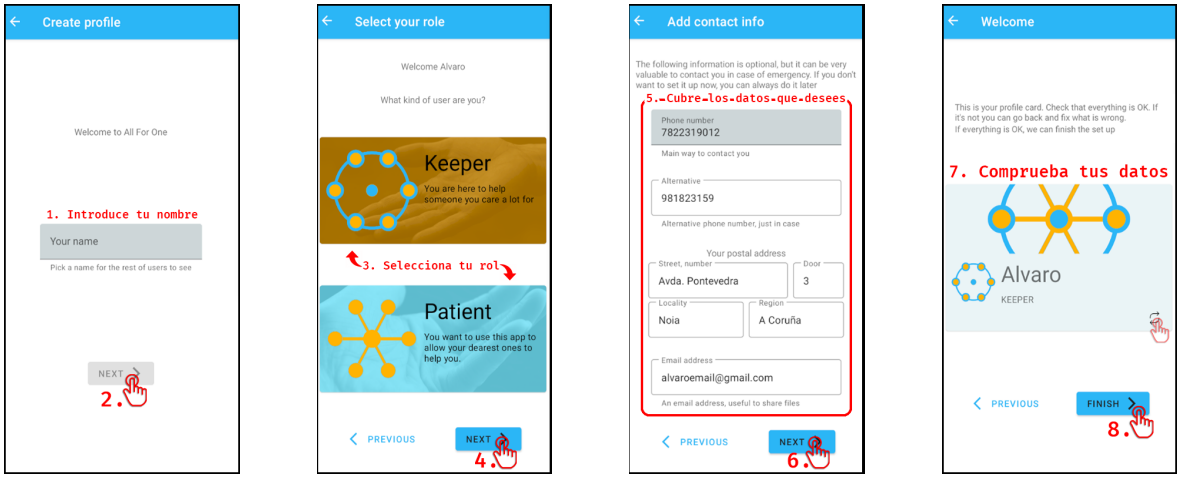


Figura 29.2: Guía de configuración del perfil

Puedes completar tu perfil con datos adicionales que exponer a los usuarios con los que te vincules para facilitar que se pongan en contacto contigo. Todos estos campos son opcionales, rellenarlos o dejarlos en blanco queda a tu discreción. La lista de datos que puedes rellenar son: **teléfonos de contacto**, puedes añadir uno principal y uno secundario, por si el principal no está operativo y urge contactar contigo; tu **tu dirección postal**, para facilitar encuentros en persona si fuese necesario, no tiene que ser tu dirección personal, puede ser la del lugar de trabajo; y **tu dirección electrónica** o email, de utilidad si algún usuario te quiere enviar algún archivo, por ejemplo. Ten en cuenta que todos estos datos sólo podrán ser vistos por tus usuarios asociados. Cuando hayas acabado avanza con el botón **Siguiente**.

Finalmente, para terminar la configuración del perfil solamente debes revisar tu información personal en la tarjeta que se te muestra. Por delante se mostrará tu nombre y tu rol, si le das la vuelta con el botón de las flechas podrás ver el resto de tus datos. Si estás conforme con ellos, pulsa en **Finalizar** para terminar y avanzar a la aplicación.

29.2. Cierre de sesión

El cierre de sesión es sencillo y similar al de muchas otras aplicaciones. Sólo es necesario acceder a **Ajustes** y seleccionar la opción de **Cerrar sesión**.

29.3. Vinculación

Para realizar la vinculación entre dos usuarios el primer requisito es que ambos sean de roles distintos, esto es, uno debe ser Paciente y el otro debe ser Cuidador. Además, el Cuidador no puede estar vinculado y el Paciente debe tener como máximo cinco vínculos activos. Si todo esto se cumple, podréis vincularos.

La vinculación consta de dos pasos: primero el Paciente tendrá que generar un código QR de vinculación y, una vez lo tenga en pantalla, el Cuidador deberá escanearlo. Esto requiere que ambos usuarios estén juntos. Estas dos acciones se explican en detalle a continuación.

29.3.1. Paciente - Generar código de vinculación

El código de vinculación lo generan los pacientes accediendo a la pantalla de **Vínculos** como se indica en el primer paso de la Figura 29.3. Al final de esa pantalla habrá un botón **Añadir vínculo** que al presionarlo desplegará un código QR válido para escanear y crear el vínculo. Este código es válido durante un minuto.

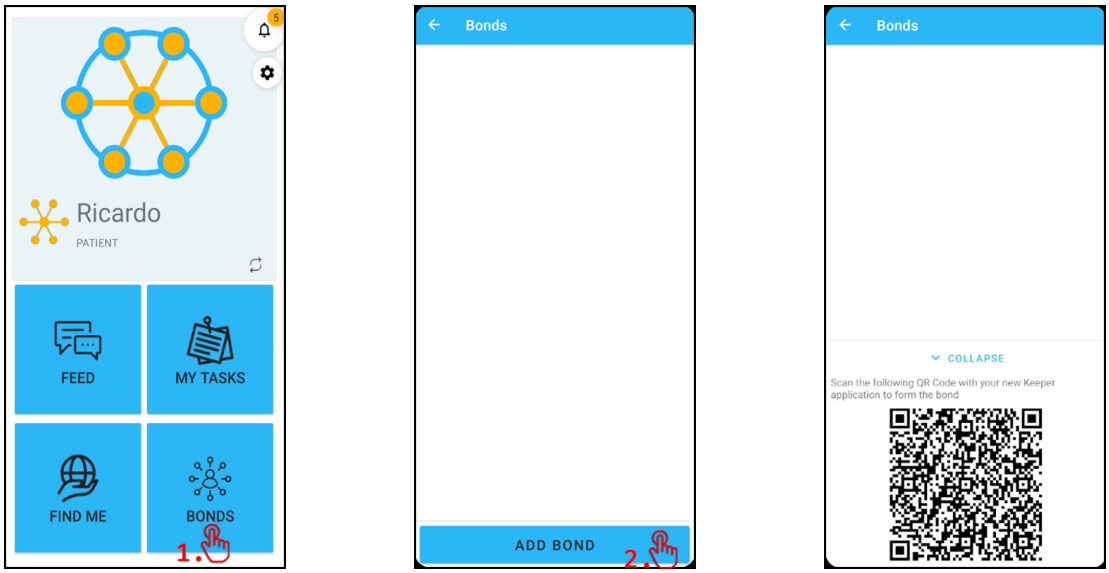


Figura 29.3: Guía de generación de código QR de vinculación

29.3.2. Cuidador - Escanear código de vinculación

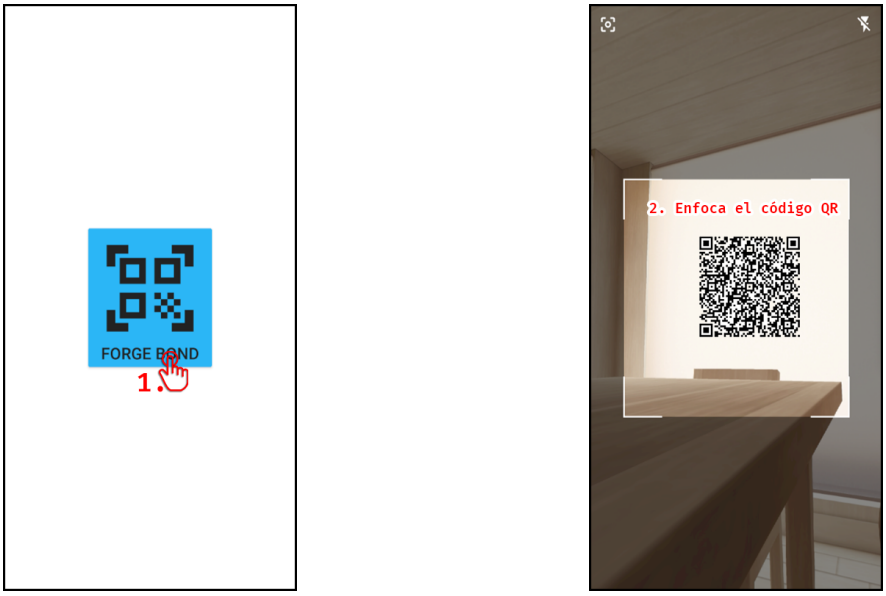


Figura 29.4: Guía de escaneo del código QR de vinculación

Cuando te registras como cuidador sólo ves la pantalla que se muestra en la primera parte de Figura 29.4. Ese único botón despliega la **cámara** con un escáner para enfocar el código QR de la aplicación del paciente. Primero se te pedirá permiso para usar la cámara, si no lo concedes no será posible establecer el vínculo de otra manera. El centro de la cámara (la parte no oscurecida) indica donde deberías centrar el código para facilitar la lectura. Una vez que lo consigas escanear, **el vínculo se creará** y el Cuidador será dirigido a una pantalla principal similar a la de los pacientes con una tarjeta con los datos del Paciente.

29.3.3. Consultar vínculos

Si eres un usuario ya vinculado puedes consultar tus vínculos en la pantalla de **Vínculos** a la que se accede desde la pantalla principal como se indica en Figura 29.5. En esta página aparecerá una tarjeta por cada vínculo directo si eres un Paciente o por cada vínculo de tu Paciente si eres un cuidador. Cada una de estas tarjetas lleva el nombre de uno de estos y usando la flecha de la misma se puede desplegar el resto de la información del usuario.

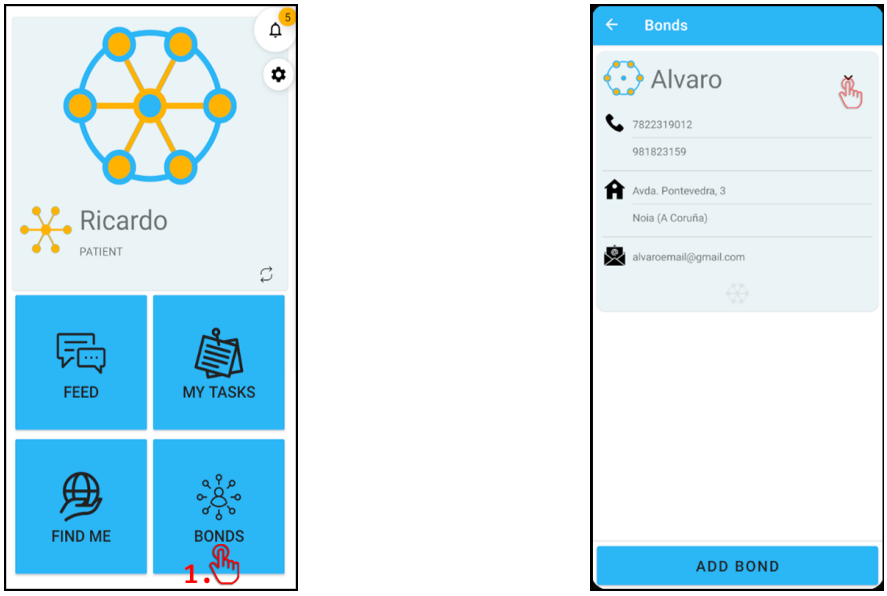


Figura 29.5: Guía de consulta de vínculos

29.3.4. Eliminar vínculos

Los vínculos de los usuarios se eliminan desde lugares diferentes (mostrados en Figura 29.6) seas un Paciente o un Cuidador. Los Pacientes podéis eliminar los vínculos desde la pantalla de **Vínculos** manteniendo pulsado sobre el vínculo que se quiera eliminar. Los Cuidadores en cambio lo podéis hacer desde **Ajustes**, con la opción **Eliminar vínculo**. En ambos casos se pedirá confirmación antes de llevarlo a cabo.

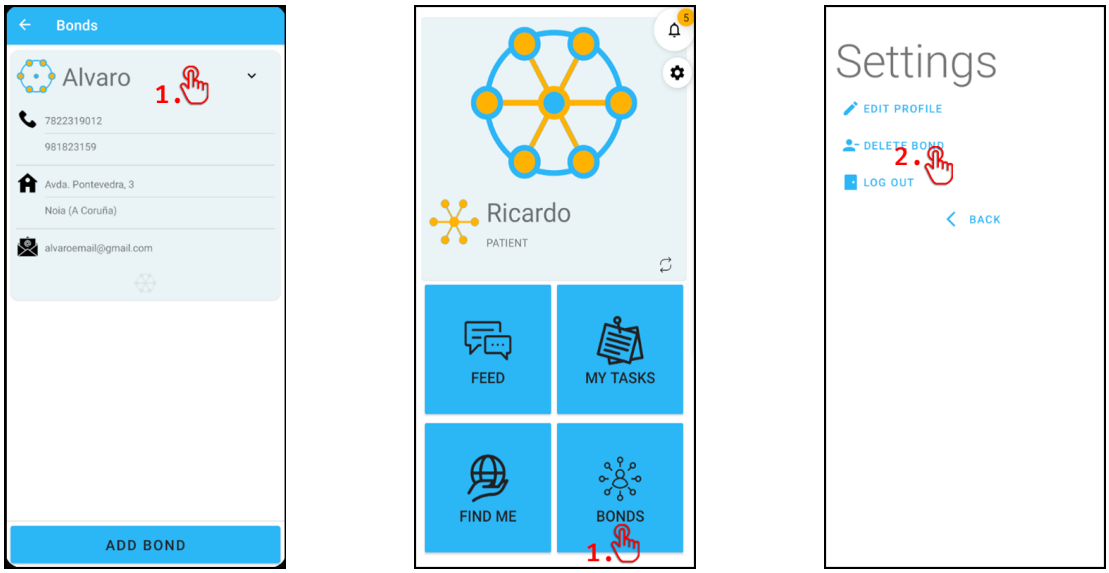


Figura 29.6: Guía de eliminación de vínculos

29.4. Geolocalización

La geolocalización se encuentra disponible en la pantalla principal con el botón con el icono del globo como se indica en Figura 29.7. Esto despliega un mapa a pantalla completa que se maneja de forma similar a cualquier otro mapa basado en **Google Maps**. Se te solicitará acceder al permiso a tu ubicación, si quieres compartirla con tus usuarios vinculados, concédela.

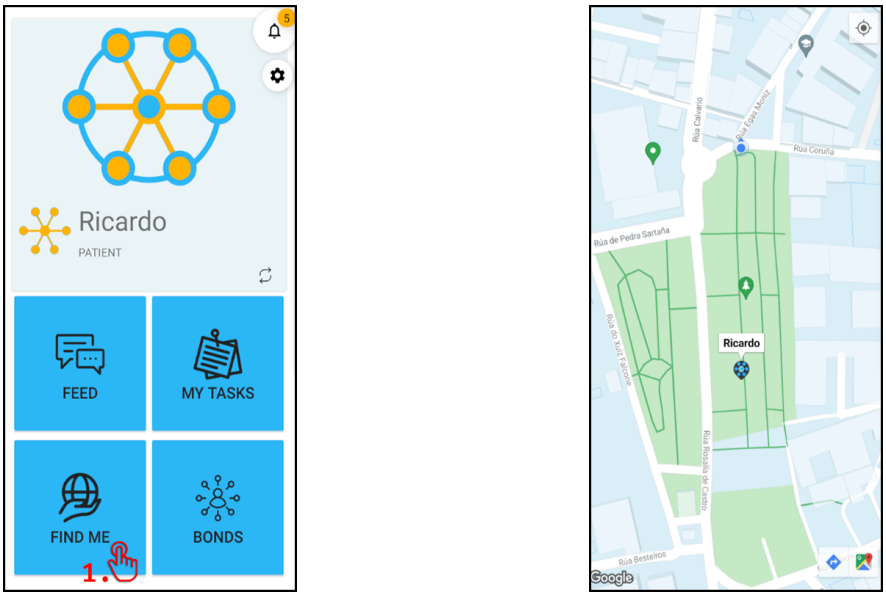


Figura 29.7: Guía de uso de geolocalización

Cuando accedes a la página de geolocalización y comienzas a compartir tu ubicación, **una notificación será enviada** al resto de usuarios. De esta forma, si un Paciente comienza a compartir su ubicación todos sus Cuidadores lo sabrán, y si un Cuidador lo hace tanto Pacientes como Cuidadores vinculados también serán avisados de que este ha comenzado una búsqueda.

El manejo del mapa sigue los controles habituales. El mapa se puede mover deslizándose, se puede hacer *zoom in* y *zoom out* con el gesto de juntar o separar los dedos respectivamente. El botón de la esquina superior derecha centra el mapa en la posición del usuario, al usarlo el mapa seguirá al usuario cuando se actualice su ubicación salvo que lo desplace manualmente.

Por último, siempre que otro usuario acceda a esta misma pantalla y empiece a compartir su ubicación **se mostrará un marcador** de un color único representando a dicho usuario. Pulsando en los marcadores de los usuarios se **mostrará el nombre** del usuario al que pertenece.

29.5. Enviar mensajes

La función de enviar mensajes está en el **Feed**. Su uso es igual que el de cualquier otro chat. Pulsando el campo de texto de la parte inferior se desplegará el teclado y se podrá escribir el mensaje que se desee enviar. El botón azul a la derecha de dicho campo sirve para enviarlo.

Esta sala de chat estará integrada por el Paciente y por todos sus Cuidadores. Los mensajes se envían en tiempo real, pero también se podrán ver todos los mensajes que se hayan enviado mientras no se está conectado. Subiendo hacia arriba se puede acceder a mensajes más antiguos y al llegar al límite de los cargados se cargarán los siguientes hasta que no haya ninguno más.

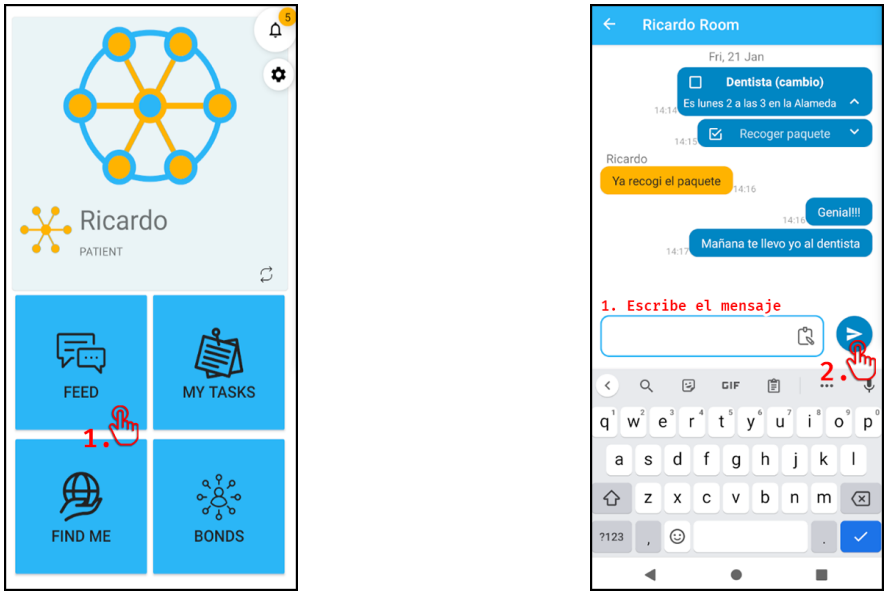


Figura 29.8: Guía de envío de mensajes

29.6. Gestionar tareas

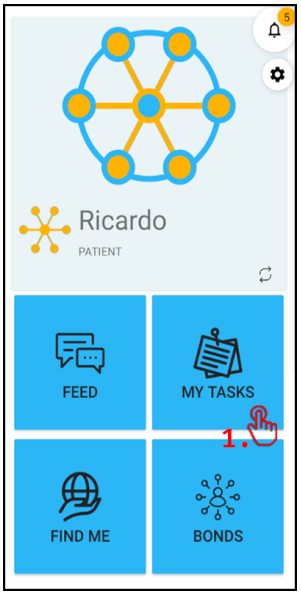


Figura 29.9: Guía de acceso a Tareas

Las tareas se pueden crear, marcar o desmarcar como hechas y eliminar. Todas estas acciones se pueden llevar a cabo desde la pantalla de **Tareas** (Figura 29.9) y desde la pantalla de **Feed** (véase Apartado 29.5).

29.6.1. Crear tarea

Para crear una tarea desde **Tareas** se debe pulsar el botón flotante de la esquina inferior derecha con el icono del portapapeles (véase Figura 29.10). Esto desplegará una ventana con el título de **Crear tarea** y dos campos de texto. Introduce el **título** de la tarea en el primero, el título será la información más básica de la tarea, procura que sea representativo y conciso.

El otro campo de texto, de mayor tamaño, es para la **descripción**. La descripción es opcional, pero sirve de mucha utilidad para añadir detalles importantes de la tarea, de forma que el Paciente o los Cuidadores que la vayan a llevar a cabo tengan todas las facilidades posibles para hacerlo. Aprovecha este campo para introducir localizaciones, horas o nombres.

Una vez estés conforme con lo que has introducido, pulsa en **Crear** para finalizar la creación de la tarea y volver a la pantalla de Tareas donde la nueva tarea aparecerá listada. También puedes usar **Cancelar** si te arrepentiste y no quieres crear una tarea.

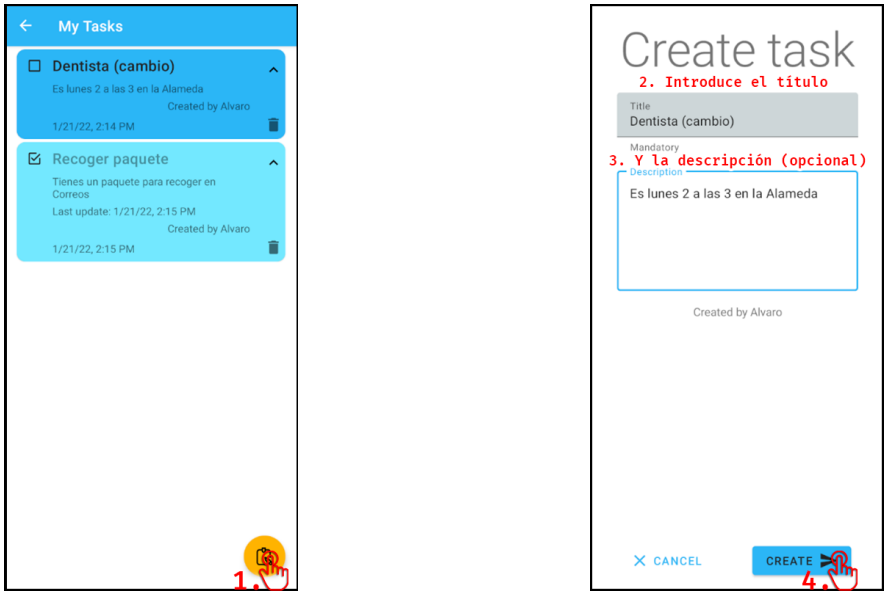


Figura 29.10: Guía de creación de tareas a través de Tareas

El otro sistema para crear una tarea es a través del **Feed**. En el mismo encontrarás un icono de portapapeles dentro del campo de texto, pulsarlo desplegará el **modo tarea** (Figura 29.11). En este modo aparecerá un área de texto adicional. Ahora el campo de texto de enviar mensajes sirve para introducir el **título** y la nueva área para la **descripción**. Sigue las mismas indicaciones que antes y cuando estés conforme envía la tarea con el mismo botón de **enviar mensajes**. La tarea aparecerá en tu chat y el de tus usuarios asociados. Si, en cambio, quieres salir del modo tarea puedes usar la equis que sustituyó al icono del portapapeles.

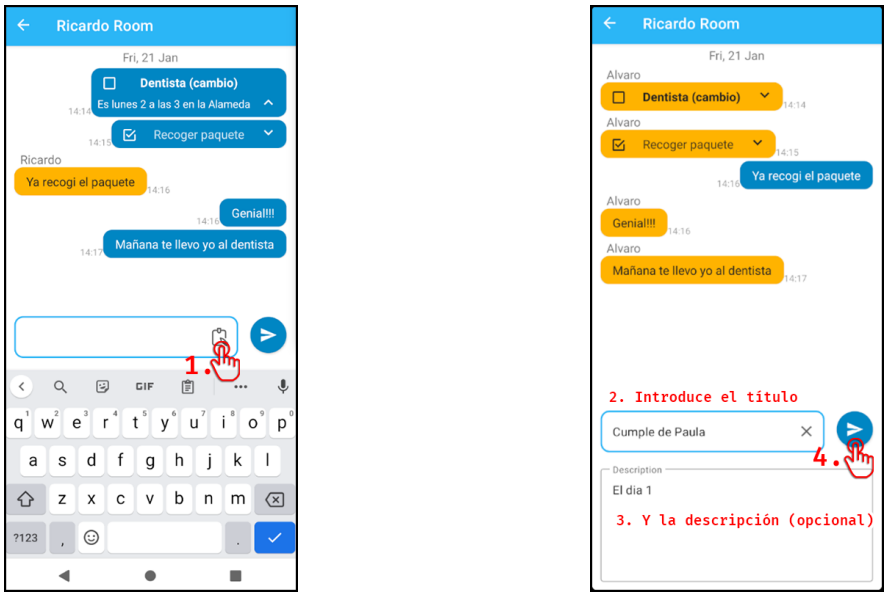


Figura 29.11: Guía de creación de tareas a través de Feed

29.6.2. Marcar y desmarcar tareas

Tanto desde **Tareas** como desde el **Feed** se pueden modificar las tareas de la misma forma. Para lo primero se debe pulsar en la casilla de *check* de la tarea. Si está vacía la tarea no está hecha y pulsarla la marcará como hecha y lo notificará a los usuarios interesados. Si por el contrario la casilla tiene un *tick* (y se ve de un color más pálido) entonces estará marcada como hecha, pulsar la casilla en este caso logrará el efecto contrario, marcará la tarea como no hecha. Esto es útil si se marca por error o si una tarea que se creía hecha en realidad no lo estaba.

29.6.3. Eliminar tareas

La eliminación de tareas es distinta si se lleva a cabo desde el **Feed** y desde **Tareas**. En Tareas podrás encontrar un **icono de papelera**, al pulsar se te preguntará si la quieres eliminar, una vez confirmes se borrará de tu lista de tareas y de todas. Para eliminar una tarea en el feed debes localizarla y **mantener pulsado encima**. La larga pulsación desplegará el mismo diálogo de confirmación y aceptarlo eliminará la tarea. Sin embargo, esta acción comparte el mismo requisito en ambos lados, sólo puedes eliminar una tarea si eres **el Paciente** o si eres **el creador** de la misma

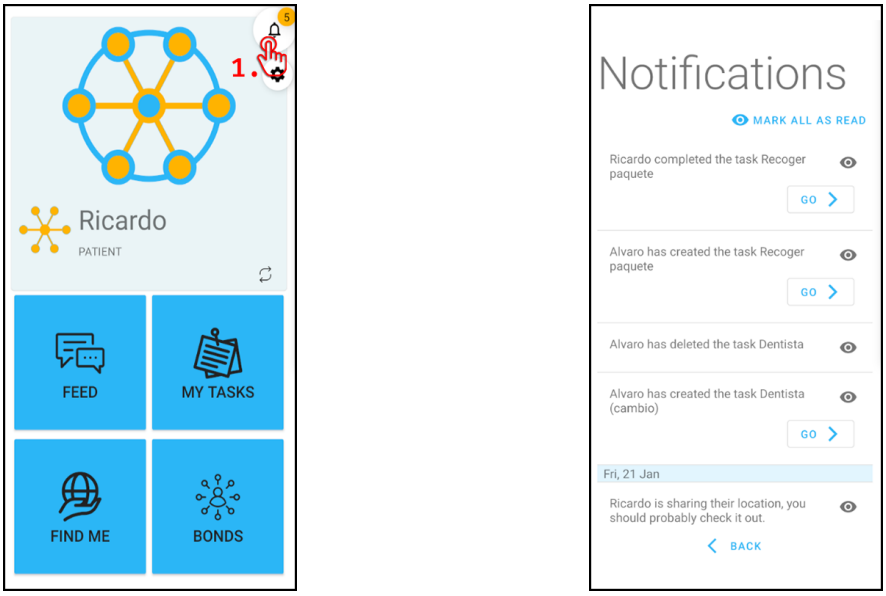


Figura 29.12: Guía de consulta de notificaciones

29.7. Consultar notificaciones

En la parte superior de la pantalla principal hay un icono con forma de **campana**. En ocasiones este icono puede tener un número encima, si es el caso, este número indica

la cantidad de notificaciones pendientes del usuario. Pulsar en dicho botón flotante (Figura 29.12) desplegará una pantalla con la lista de notificaciones aún no leídas.

Algunas de estas notificaciones tendrán un botón **Ir**. Estas notificaciones están relacionadas con otras pantallas y con dicho botón podrás acceder rápidamente al destino deseado. Por ejemplo, cuando un usuario vinculado empieza a compartir su ubicación recibirás una notificación indicándotelo y el botón *Ir* de la misma te llevará directamente a la pantalla del mapa.

Las notificaciones pueden marcarse como leídas haciendo uso del icono con forma de **ojo** de su derecha. Al hacerlo esa notificación desaparecerá para siempre. También puedes agilizar esto usando la función **Marcar todas como leídas** de la parte superior. Esto señalará todas las notificaciones actuales como leídas y vaciará la pantalla completa.

29.8. Editar datos

Desde **Ajustes**, pantalla a la que se accede desde el botón con forma de **engranaje** de la pantalla principal, puedes actualizar tus datos de usuario. Para ello debes seguir los pasos de la Figura 29.13. Primero despliega los campos de edición pulsando en **Editar perfil**. Una vez se desplieguen, modifica aquellos campos que quieras cambiar. Cuando estés satisfecho con los datos, completa la acción pulsando en **Confirmar**. En cualquier momento puedes usar **Cancelar** para cerrar los campos y abortar la operación.

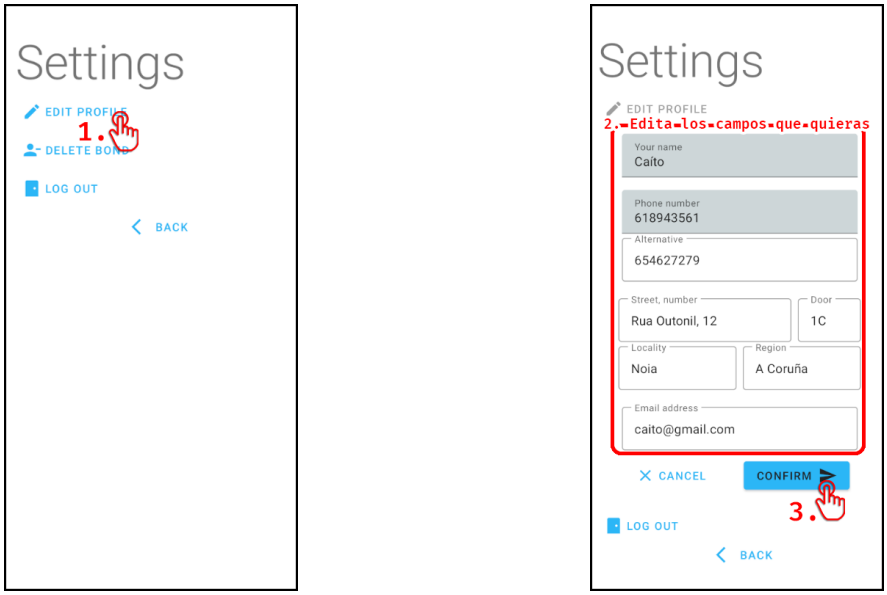


Figura 29.13: Guía de edición del perfil

30. Manual de despliegue

30.1. Creación del servicio

De cara a la realización del despliegue por primera vez es necesario acceder al **Portal de Azure**[1]. En dicho portal se seleccionará la opción de crear un recurso de tipo **App Services**. La primera pestaña, **Datos básicos**, permite seleccionar las características más elementales del servicio a desplegar. A continuación se describen los campos a rellenar.

- **Suscripción**, grupo de facturación de la cuenta que se utilizará para gestionar el pago de este recurso. En el caso de este proyecto será la suscripción **Azure para estudiantes** obtenida con los beneficios proporcionados por la cuenta corporativa de la universidad.
    - **Grupo de recursos**, colección de recursos de una suscripción que comparten ciclo de vida, permisos y directivas. Si no existe ninguno, se puede crear en este paso. Se usará **4l1**, nombre en clave del proyecto.
- **Nombre**, nombre que se desea dar al recurso. Emplearemos el mismo que el repositorio del servicio: **4l1service**.
- **Publicar**, forma de publicación del sistema en el servicio. En el caso de este despliegue será por medio de **Código**.
- **Pila de entorno en tiempo de ejecución**, entorno tecnológico en el que funcionará el sistema, en nuestro caso **Node 16 LTS**.
- **Sistema operativo**, sobre el que se ejecutará el sistema. Al haber sido desarrollado en **Linux**, se usará ese entorno para replicar lo máximo posible el entorno de desarrollo.
- **Región**, área de servidores en la que emplazar el sistema, por precios y cercanía se ha seleccionado **West Europe**.
- **Plan de App Service**, plan de coste y características deseado. Al no realizarse un despliegue completo, un plan de pruebas como es **Básico B1** es suficiente.

Tras completar esta primera sección se puede avanzar a **Implementación**, en la cuál se ha de seleccionar la opción de **Habilitar la Integración continua**. Haciéndolo se desplegará una nueva sección en la que se podrá iniciar sesión con **GitHub** y

[1] https://portal.azure.com/

seleccionar la organización, el repositorio y la rama que se desplegará el servicio. Este despliegue se hará sobre el repositorio del servicio (**4l1-service**) y su rama **main**. Automáticamente se creará un archivo con la configuración del flujo de trabajo de **GitHub Actions**. Este archivo es modificable, pero en el caso de este despliegue no es necesario realizarle modificaciones. Dicho archivo se puede ver en el Capítulo 38 Acción de despliegue.

El resto de secciones no será necesario modificarlas para este despliegue. En la última, **Revisar y crear**, se mostrará un resumen de todo lo que hemos seleccionado y podremos confirmar la creación del recurso. El propio servicio de Azure y GitHub Actions crearán el flujo que desplegará el código en el servidor. Sin embargo aún no será operativo.

30.2. Configuración de la aplicación

En el lanzamiento en local de la API durante el desarrollo se utilizan archivos de tipo `.env` para definir las variables de entorno del sistema. En este despliegue no se utilizarán ficheros, sino que se definirán dentro de la opción de **Configuración** de nuestro AppService recién creado. Deben definirse las variables de entorno listadas en el Cuadro 30.1.

Nombre	**Valor**
AUTH_TOKEN_EXPIRATION_TIME	15m
AUTH_TOKEN_SECRET	*<generar clave entrópica>*
BOND_TOKEN_EXPIRATION_TIME	1m
BOND_TOKEN_SECRET	*<generar clave entrópica>*
GOOGLE_CLIENT_ID	*<token de cliente de Google API>*
GOOGLE_SERVER_ID	*<token de servidor de Google API>*
JET_LOGGER_FORMAT	LINE
JET_LOGGER_MODE	FILE
JET_LOGGER_TIMESTAMP	TRUE
MONGO_URL	*<url conexión a MongoDB Atlas>*
NODE_ENV	production
REFRESH_TOKEN_EXPIRATION_TIME	1d
REFRESH_TOKEN_SECRET	*<generar clave entrópica>*

Tabla 30.1: Variables de entorno del servicio

30.3. Configuración de MongoDB Atlas

De cara a finalizar el despliegue es necesario conectar la aplicación con el clúster que proporcionará la base de datos. Para ello se creará una base de datos desde la opción **Create** del portal de Cloud MongoDB[2]. Se seleccionará un clúster de tipo **Dedicated** en algún servidor europeo sobre **Azure**. Se seleccionará un como **Cluster Tier** el **M0**, gratuito y suficiente para este despliegue. También es posible aumentar a otros sirviéndonos del crédito de estudiante. Se selecciona un nombre y finalmente se crea el clúster.

Una vez el clúster esté desplegado será posible obtener la URL de conexión que se debe introducir como variable de entorno en el servicio. Sin embargo, eso no será suficiente para que el servidor pueda comunicarse con la base de datos. El clúster está protegido con una lista blanca de direcciones IP que se puede configurar en **Network Access**. Todas las direcciones IP de salida que lista el AppService en **Propiedades** deben ser añadidas a esta lista blanca. Una vez se haga y la actualización se procese, el despliegue será completo.

30.4. Integración continua

Gracias a la configuración de **GitHub Actions** en la creación del servicio todo lo necesario para la integración continua del sistema está ya creado. De cara a integrar cambios en la versión desplegada del sistema sólo es necesario lanzar una *pull request* a la rama **main** del repositorio de la API y unir el código. Esto lanzará de forma automática un flujo de trabajo que gestionará y llevará a cabo la actualización del código del servicio y reinicio.

[2] https://cloud.mongodb.com/

31. Manual de desarrollador

31.1. Despliegue en local

31.1.1. Despliegue de la API

De cara a realizar un despliegue y ejecución del sistema en local es necesario realizar una serie de configuraciones de cara a permitir el funcionamiento y la comunicación entre los diferentes subsistemas.

Primero, es necesario establecer las **variables de entorno** que se van a utilizar en la API y que se tratan en el manual de despliegue, en la Apartado 30.2. Estas variables pueden establecerse en el sistema operativo que se estén usando, pero es más recomendable crear ficheros `.env` para tal empresa. Al hacerlo se pueden modificar con facilidad y se pueden tener ficheros distintos según el entorno que se quiera replicar.

Estos ficheros se deben crear en la ruta `src/pre-start/env/` de la base de código de la API utilizando como nombre la palabra que se quiera utilizar para invocar dicho entorno. Se recomienda usar **development** como nombre para el entorno que se vaya a usar habitualmente y **production** para uno que emule el entorno de producción. El repositorio cuenta con un fichero de entorno creado para los tests llamado `test.env` que puede servir de ejemplo. Los pares clave-valor que se deben definir en el fichero son los indicados en Cuadro 30.1 además de **PORT** y **HOST**.

Una vez se hayan preparado las variables de entorno es necesario añadir la IP local a la lista blanca del clúster de MongoDB que se vaya a utilizar, de forma similar a como se hizo en la Apartado 30.3 con las direcciones del servidor. Esto permitirá a nuestro servidor local establecer la conexión con el clúster y realizar operaciones con la base de datos.

Al finalizar estas preparaciones, la API estará preparada para ser desplegada en local con la dirección IP y puerto indicados en las variables de entorno destinadas a tal uso. Se recomienda usar `PORT=3000` y `HOST=127.0.0.1`. Para lanzar la API se puede utilizar el comando `yarn run dev` desde la raíz del proyecto para lanzarlo con las variables de **development** en un entorno autoreiniciable. Usando `yarn run prod` se compilará el código con las variables de **production** y se ejecutará el código compilado como ocurrirá en el servidor. En el `package.json` se puede consultar a qué comandos equivalen estos y crear nuevos a partir de ellos.

31.1.2. Despliegue de la aplicación

La aplicación hace uso de servicios de la API de Google que en desarrollo requieren añadir una *hash* local a una lista blanca. Esto se lleva a cabo en la pestaña **Credentials** del proyecto. Se puede encontrar toda la información del procedimiento a realizar en el siguiente enlace:

https://developers.google.com/android/guides/client-auth

Tras añadir la clave a la lista blanca la aplicación lanzada en local podrá comunicarse con los servicios de Google necesarios por la aplicación, falta establecer su comunicación con el servidor local. Para hacer esto se debe añadir la IP del servidor al archivo `build.gradle` de la carpeta **app** (no al de la raíz). En ese fichero se debe añadir la siguiente línea en **android.buildTypes.release** y **android.buildTypes.debug**, según el entorno que se vaya a usar.

```
buildConfigField("String", "SERVER_IP", "http://10.0.2.2:3000")
```

Atención, la IP de la máquina local no es 127.0.0.1 para este caso, pues dicha IP corresponderá con el emulador en el que se lance la aplicación, la dirección que representa a la máquina que ejecutará el emulador es `http://10.0.2.2`. En caso de querer hacer un lanzamiento local de la aplicación que conecte con el servidor desplegado debe sustituirse la dirección por la del servidor, tal que:

```
buildConfigField("String", "SERVER_IP", "https://4l1service.azurewebsites.net"
```

Cuando todas estas configuraciones estén listas y el servidor desplegado (esto es cuando hayan emitido el mensaje informativo indicándolo) ya se podrá lanzar la aplicación con el emulador de Android Studio y el sistema debería funcionar correctamente.

31.2. Lanzamientos de las pruebas

Para el lanzamiento de pruebas de la API en local es únicamente necesario tener el archivo `test.env` nombrado en 31.1.1 Despliegue de la API en el lugar especificado. Para lanzar el conjunto completo de pruebas se puede usar los comandos `yarn run test` o `yarn jest`. En caso de querer lanzar únicamente las pruebas de un fichero se puede especificar el nombre o la ruta tras el comando de pruebas, por ejemplo: `yarn jest TokenService.ts`.

La API también cuenta con un sistema de revisión de código que puede ser ejecutado con el comando `yarn run lint`.

Para lanzar las pruebas de la aplicación móvil sólo hay que seleccionar la clase, el paquete o la prueba que se desee ejecutar y lanzarla con las propias herramientas de Android Studio. Una forma de lanzar todas las pruebas de la aplicación es situarse encima del paquete **test/java** en el árbol de ficheros o en el paquete de test deseado si se usa la vista de proyecto; abrir el menú contextual con el click derecho o el sistema preferido; y elegir la opción **Ejecutar tests en ...**.

Parte VII

Conclusión

32. Estado del sistema

Al momento de la entrega de este documento y del sistema para la presentación del proyecto, *AllForOne* no se encuentra en un estado de poder ser publicado o utilizado por el público general.

Si bien las funcionalidades clave planeadas para el sistema se encuentran desarrolladas y probadas, el sistema **sigue necesitando iteraciones y revisiones** que terminen de pulir estas y de agregar la seguridad o fiabilidad esperadas de un producto disponible públicamente. Por poner un ejemplo que fue nombrado en el Capítulo 26, las comunicaciones entre el servidor y la aplicación móvil no se encuentran a día de hoy cifradas y esta característica debería ser un requisito mínimo a alcanzar antes de dar el sistema por concluido.

De igual manera, una característica planeada que no se ha podido llevar a cabo por temas de tiempo y prioridades y que se plantea en el Apartado 34.1 es el desarrollo de todo el entramado necesario para garantizar los derechos y leyes relativos a los sistemas digitales o a los datos de los usuarios. No es siquiera planteable el lanzar una aplicación que registra datos de contacto de un paciente de Alzheimer sin cubrir las debidas protecciones de datos críticos como esos.

Otro aspecto necesario de cara a completar el sistema es el escalado del servidor a una instancia más preparada para un despliegue en producción. Actualmente el servidor se encuentra en una instancia de **tipo B**, dirigidas para pruebas; por lo que sería necesario plantear la mejora a una cuota de **tipo A**.

Por lo demás, y aunque algunas de las ampliaciones que se propondrán en Capítulo 34 deberían ser desarrolladas antes de alcanzar una versión `1.0` del sistema, el sistema se encuentra en un estado de completitud suficientemente satisfactorio para su uso por público limitado o grupos de control y, por tanto, suficiente de cara a esta entrega.

33. Conclusiones personales

Ha sido un largo camino, y me gustaría poder hacer énfasis en el adjetivo de dicha afirmación. Ni siquiera ha sido mi primer intento en esa ingeniería, cuatro años antes de entrar a la Escuela ya me había aventurado en la facultad de informática de A Coruña en el mismo grado. El día que pisé por primera vez aquel lugar a casi trescientos kilómetros hace más de nueve años ya tenía claro cuál sería mi proyecto de fin de estudios.

Aquí estoy por fin, redactando las conclusiones de este documento acerca del desarrollo de ese sistema para el apoyo a pacientes y familias afectadas por la terrible enfermedad de Alzheimer que siempre quise hacer. Lo que he creado dista mucho de parecerse mínimamente a aquella primera idea que tuve entonces. No se parece siquiera a ninguna de las iteraciones del concepto que pasaron por mi cabeza a lo largo de todos estos años. Hubo una época en la que incluso me planteaba la construcción de un dispositivo destinado a este fin.

Sin embargo, estoy muy contento de lo que ha salido aquí. Creo que es la mejor versión de todo lo que he pensado, y esto es porque se construye sobre la experiencia y sobre los conocimientos adquiridos en estos cuatro años de carrera. Es fácil soñar con características de tu aplicación ideal, pero cuando le aplicas la pintura de realidad y lo consigues plasmar en una función realizable con la aplicación de las buenas prácticas y estándares de calidad esperables, lo que resulta es incluso más atractivo. Porque es patente que todo este sistema se construye sobre los aprendizajes de este grado.

En la elaboración de este proyecto trabajé sobre cosas en las que ya tenía experiencia, pero también he descubierto nuevas tecnologías y sistemas en los que mi conocimiento era nulo. **Socket.io** y los **WebSocket** son una tecnología con un muy buen funcionamiento y, sobre todo, con una curva de entrada muy halagüeña con la que se me hizo muy sencillo desarrollar las comunicaciones que necesitaba en muy poco tiempo. En similar manera descubrí el lenguaje **Kotlin** en profundidad, encontrándome con una herramienta de gran utilidad que aún con el proceso de aprendizaje redujo ampliamente mis tiempos de creación de código respecto a hacer lo mismo en Java.

Con otras muchas cosas ya tenía cierta experiencia, como con **Express**, **MongoDB** o **Android**. Para la base de cada uno de mis subsistemas decidí abogar por algún sistema que se me hubiese enseñado en la carrera y lo cierto es que fue la mejor decisión posible. En el proceso gané agilidad, pero no perdí la obtención de nuevos

conocimientos, porque siempre hay más que aprender y con estas tecnologías también mejoré las bases que se me habían proporcionado en la carrera con buenas prácticas recabadas en diferentes documentaciones o artículos.

Aún con todo, y siguiendo la base de que la creación de este proyecto es sinónima con el aprendizaje y el crecimiento, donde más progresé como ingeniero informático con este desarrollo es en todos los pasos anteriores a ponerse a crear código. Cada diagrama que dibujaba en mi cuaderno de notas, cada requisito que me apuntaba en la primera lista que hice o cada clase que ya tenía perfectamente concebida antes de ponerme delante del IDE fue un gran progreso, ayuda y aprendizaje. Fue la demostración más palpable de que saldré de aquí como **ingeniero de software** y no únicamente como una persona capaz de hablar un idioma que entienda un sistema informático.

He disfrutado cada minuto de este desarrollo. Se me ha hecho una tarea titánica que me llegó a sobrepasar, pero lo he disfrutado. Me he frustrado, me he agotado, he aprendido y me he alegrado. No es la aplicación con la que soñaba cuando pisaba por primera vez la educación universitaria, pero sí consigue todos los objetivos que quería y entrega un producto mucho mejor de lo que podría haber esperado. Y todo esto lo he hecho yo, con mis conocimientos, averiguaciones y mis propias manos. Y no puedo estar más **orgulloso**.

34. Ampliaciones

34.1. Avenencias legales

Una de las tareas pendientes a la conclusión de este desarrollo de cara a tener la aplicación lista para un lanzamiento público sería la adecuación del sistema con las diferentes obligaciones legales y morales. Por ejemplo, a día de hoy los diferentes datos de los usuario están siendo almacenados sin las medidas necesarias como podrían ser la seudonimización y no existe la infraestructura necesaria detrás del sistema de la base de datos para garantizar o denegar el acceso a los datos según las autorizaciones del personal.

Aunque muchas de estas cosas ya se presumían fuera del alcance del proyecto otras han quedado fuera por limitaciones de tiempo aunque existía un plan de implementarlas. Uno de estos casos sería añadir a la última pantalla de confirmación del registro del usuario un aviso legal que el usuario pudiese leer y debiese aceptar antes de terminar de crearse el perfil. Desgraciadamente no ha podido llevarse a cabo.

34.2. Edición de tareas y mensajes

El proyecto entregado únicamente permite cambiar el estado de una tarea, pero no su contenido, ni el de los mensajes de texto enviado a través del feed. Esto obliga a que en caso de crear una tarea erróneamente o de que una sufra un cambio como podría ser el retraso de una cita médica, lo único que se pueda hacer para corregir la información errónea sea eliminar la tarea y volver a crearla.

Permitir editar el título o la descripción de la tarea solucionaría este problema, para llevar esto a cabo un nuevo endpoint podría ser incluido en la API de tareas y mensajes que gestione esa actualización y envíe la debida notificación.

34.3. Filtros y búsqueda

Una característica que siempre mejora mucho la usabilidad de gran número de aplicaciones es la inclusión de filtros y buscadores en funciones que incluyan listados. De esta forma se facilita a un usuario el encontrar un ítem concreto que conocen o ignoran aquellos que no le interesan. Esto podría ser implementado en el feed para la

búsqueda de un mensaje concreto que coincida con la búsqueda o en la lista de tareas para filtrar fuera aquellas ya completas o creadas por otros usuarios.

34.4. Imágenes de usuario

La función planeada que más ha costado dejar fuera de este proyecto por las constricciones temporales fue la implementación de imágenes como parte de los perfiles de usuario. La idea inicial era añadir la opción de subir la imagen o poder sacarla desde la pantalla de introducción del nombre del usuario. Más adelante, esta imagen sería mostrada en las tarjetas de perfil de los usuarios.

Esta ampliación es especialmente útil de cara a fomentar el recuerdo de los rostros de los allegados del paciente, combatiendo una de las degeneraciones más relevantes del avance de la enfermedad en su etapa intermedia. Otra utilidad podría ser la de contar con una foto del paciente si se extravía y se debe buscar de forma que se pueda preguntar a viandantes, o al revés si el paciente extraviado busca a la persona con la que estaba.

34.5. Parametrización de la REST API

Actualmente, algunos de los endpoint de la API REST aceptan algunos parámetros, como es el caso de la recuperación de notificaciones que permite especificar la edad máxima de las notificaciones a recuperar o el de recuperación de tareas que también acepta dicho parámetro. Estos parámetros podrían después ser convertidos en opciones de personalización de los usuarios en la aplicación, lo que supondría una mejor de calidad de uso.

Sin embargo, la parametrización alcanzada en el momento de entrega de este proyecto no es tan amplia como se habría deseado. El extremo de recuperación de páginas de mensajes sólo permite especificar la página a recuperar, pero no el tamaño de la página, de forma que un usuario no tiene la oportunidad de cargar grupos mayores de mensajes aunque tenga la capacidad de procesamiento necsaria para permitirse esa mejora. Otros posibles parámetros que podrían ser de utilidad son aquellos que permitirían añadir filtros, como el tipo de mensaje o el autor, que permitirían implementar fácilmente nuevas características.

Parte VIII

Anexos

35. Presupuesto

Código	Ítem	Partida	Importe	Total
1		**Planificación**		1 149.00€
	1	Justificación y objetivos	111.00€	
	2	Estudio de la situación actual	148.00€	
	3	Requisitos iniciales	74.00€	
	4	Evaluación de alternativas	248.00€	
	5	Definición de la solución	54.00€	
	6	Planificación temporal	144.00€	
	7	Elaboración de presupuesto	370.00€	
2		**Análisis**		1 174.00€
	1	Identificación de requisitos	399.00€	
	2	Identificación de subsistemas	137.00€	
	3	Especificación de casos de uso	165.00€	
	4	Bocetado de interfaces de usuario	88.00€	
	5	Propuesta de clases del sistema	280.00€	
	6	Especificación del plan de pruebas	35.00€	
	7	Definición del plan de despliegue	70.00€	
3		**Diseño**		1 500.00€
	1	Definición de la arquitectura	350.00€	
	2	Diseño de clases	700.00€	
	3	Especificación del modelo de datos	70.00€	
	4	Especificación de las interfaces de comunicación	140.00€	
	5	Diseño de interfaces de usuario	240.00€	
4		**Construcción**		6 957.00€
	1	REST API	1 431.00€	
	2	WebSocket API	1 053.00€	
	3	Despliegue del servidor	336.00€	
	4	Aplicación móvil	3 984.00€	
5		**Formación**		197.00€
	1	Elaboración de manuales de usuario	66.00€	
	2	Elaboración de manual de instalación	24.00€	
	3	Elaboración de manual de despliegue	56.00€	
	4	Elaboración de manual de desarrollador	51.00€	
		TOTAL		**10 977.00€**

Tabla 35.1: Presupuesto de costes expandido

I1	I2	I3	Descripción	Horas	Precio	Subtotal	Total
1			**Arranque del proyecto**				259.00€
	1		Justificación y objetivos			111.00€	
		1	*Jefe de Proyecto*	3	37.00€		
	2		Estudio de la situación actual			148.00€	
		1	*Jefe de Proyecto*	4	37.00€		
2			**Planificación del sistema**				890.00€
	1		Requisitos iniciales			74.00€	
		1	*Jefe de Proyecto*	2	37.00€		
	2		Evaluación de alternativas			248.00€	
		1	*Analista de Sistemas*	5	32.00€		
		1	*Consultor*	1	18.00€		
		1	*Arquitecto Software*	2	35.00€		
	3		Definición de la solución			54.00€	
		1	*Arquitecto Software*	3	35.00€		
	4		Planificación temporal			144.00€	
		1	*Jefe de Proyecto*	8	37.00€		
	5		Elaboración de presupuesto			370.00€	
		1	*Jefe de Proyecto*	10	37.00€		
			TOTAL				**1 149.00€**

Tabla 35.2: Partida 1. Planificación

I1	I2	I3	Descripción	Horas	Precio	Subtotal	Total
1			**Análisis**				1 174.00€
	1		Identificación de requisitos			399.00€	
		1	*Analista de Sistemas*	9	32.00€		
		2	*Jefe de Proyecto*	3	37.00€		
	2		Identificación de subsistemas			137.00€	
		1	*Arquitecto Software*	3	35.00€		
		2	*Analista de Sistemas*	1	32.00€		
	3		Especificación de casos de uso			165.00€	
		1	*Analista de Sistemas*	4	32.00€		
		2	*Jefe de Proyecto*	1	37.00€		
	4		Bocetado de interfaces de usuario			88.00€	
		1	*Diseñador de interfaces*	4	29.00€		
	5		Propuesta de clases del sistema			280.00€	
		1	*Arquitecto Software*	8	35.00€		
	6		Especificación del plan de pruebas			35.00€	
		1	*Arquitecto Software*	1	35.00€		
	7		Definición del plan de despliegue			70.00€	
		1	*Arquitecto Software*	2	35.00€		
			TOTAL				**1 174.00€**

Tabla 35.3: Partida 2. Análisis

I1	I2	I3	Descripción	Horas	Precio	Subtotal	Total
1			**Diseño**				1 500.00€
	1		Definición de la arquitectura			350.00€	
		1	*Arquitecto Software*	10	35.00€		
	2		Diseño de clases			700.00€	
		1	*Arquitecto Software*	20	35.00€		
	3		Especificación del modelo de datos			70.00€	
		1	*Arquitecto Software*	2	35.00€		
	4		Espec. de las interfaces de comunicación			140.00€	
		1	*Arquitecto Software*	4	35.00€		
	5		Diseño de interfaces de usuario			240.00€	
		1	*Diseñador de interfaces*	10	29.00€		
			TOTAL				**1 500.00€**

Tabla 35.4: Partida 3. Diseño

I1	I2	I3	Descripción	Horas	Precio	Subtotal	Total
1			**Preparación del Desarrollo**				153.00€
	1		Prototipado			153.00€	
		1	*Desarrollador Back-End*	3	27.00€		
		2	*Desarrollador Móvil*	3	31.00€		
2			REST API				1 431.00€
	1		Autenticación			162.00€	
		1	*Desarrollador Back-End*	6	27.00€		
	2		Usuarios			405.00€	
		1	*Desarrollador Back-End*	15	27.00€		
	3		Feed			594.00€	
		1	*Desarrollador Back-End*	22	27.00€		
	4		Tareas			270.00€	
		1	*Desarrollador Back-End*	10	27.00€		
3			WebSocket API				1 053.00€
	1		Global			216.00€	
		1	*Desarrollador Back-End*	8	27.00€		
	2		Mensajería			459.00€	
		1	*Desarrollador Back-End*	17	27.00€		
	3		Localización			54.00€	
		1	*Desarrollador Back-End*	2	27.00€		
	4		Notificación			324.00€	
		1	*Desarrollador Back-End*	12	27.00€		
4			Despliegue del servidor				336.00€
	1		Pipeline de despliegue			336.00€	
		1	*Desarrollador DevOps*	12	28.00€		
5			Aplicación móvil				3 984.00€
	1		Inicio y cierre de sesión			264.00€	
		1	*Desarrollador Móvil*	11	24.00€		
	2		Pantalla principal			216.00€	
		1	*Desarrollador Móvil*	9	24.00€		
	3		Vinculación			480.00€	
		1	*Desarrollador Móvil*	20	24.00€		
	4		Listado de vinculos			360.00€	
		1	*Desarrollador Móvil*	15	24.00€		
	5		Geolocalización			696.00€	
		1	*Desarrollador Móvil*	29	24.00€		

	6		Feed			888.00€
		1	*Desarrollador Móvil*	37	24.00€	
	7		Gestión de tareas			600.00€
		1	*Desarrollador Móvil*	25	24.00€	
	8		Notificaciones			480.00€
		1	*Desarrollador Móvil*	20	24.00€	
			TOTAL			**6 957.00€**

Tabla 35.5: Partida 4. Construcción

I1	I2	I3	Descripción	Horas	Precio	Subtotal	Total
1			**Formación**				197.00€
	1		Elaboración de manuales de usuario			66.00€	
		1	*Escritor de Contenido*	3	14.00€		
		2	*Desarrollador Móvil*	1	24.00€		
	2		Elaboración de manual de instalación			24.00€	
		1	*Desarrollador Móvil*	1	24.00€		
	3		Elaboración de manual de despliegue			56.00€	
		1	*Desarrollador DevOps*	2	28.00€		
	4		Elaboración de manual de desarrollador			51.00€	
		1	*Desarrollador Móvil*	1	24.00€		
		2	*Desarrollador Back-End*	1	27.00€		
			TOTAL				**197.00€**

Tabla 35.6: Partida 5. Formación

Personal	Sueldo Bruto Año	Coste Salarial Año
Arquitecto de Software	40 000.00€	53 333.33€
Analista de Sistemas	35 000.00€	46 666.67€
Jefe de Proyecto	41 500.00€	55 333.33€
Consultor	25 000.00€	33 333.33€
Escritor de Contenidos	22 320.00€	29 760.00€
Desarrollador DevOps	32 500.00€	43 333.33€
Desarrollador Back-End	31 000.00€	41 333.33€
Desarrollador Móvil	29 500.00€	39 333.33€
Diseñador de Interfaces	27 200.00€	36 266.67€
TOTAL		**378 693.33€**

Tabla 35.7: Coste del personal

Servicio	Coste Mes	Coste Hora	Coste Proyecto
Pequeño utilaje y herramientas	5.00€	0.03€	12.55€
Consumo de electricidad	40.00€	0.24€	100.40€
Cuota de Internet	35.00€	0.21€	87.85€
TOTAL			**200.80€**

Tabla 35.8: Costes indirectos

Equipo/Licencia	Tipo	Uds.	Amortización	Coste Hora	Coste Proyecto
Xiaomi Redmi Note 10 Pro	Amortización	1	50.00€	0.02€	10.46€
Azure App Service B1	Suscripción	1	132.96€	0.07€	27.81€
Xiaomi Mi A2	Amortización	1	35.00€	0.01€	3.34€
IdeaPad 3 Gen 6	Amortización	1	120.00€	0.06€	25.10€
Microsoft 365 Estándar	Suscripción	1	58.80€	0.03€	12.30€
TOTAL					**79.01€**

Tabla 35.9: Medios de producción

Personal	Precio/hora	Facturación
Arquitecto de Software	35.00€	52 719.00€
Analista de Sistemas	32.00€	51 404.80€
Jefe de Proyecto	37.00€	74 296.00€
Consultor	18.00€	28 915.20€
Escritor de Contenidos	14.00€	16 687.20€
Desarrollador DevOps	28.00€	44 979.20€
Desarrollador Back-End	27.00€	40 662.00€
Desarrollador Móvil	24.00€	40 963.20€
Diseñador de Interfaces	22.00€	35 340.80€
TOTAL		**386 138.40€**

Tabla 35.10: Facturación del equipo

Concepto	Importe
Total de costes directos	293 398.67€
Total de costes indirectos	85 574.47€
Suma de costes directos e indirectos	378 973.14€
Beneficio deseado	0.00€
Coste total	378 973.14€
Facturación posible del equipo	386 138.40€
Margen entre coste total y facturación	1.89 %

Tabla 35.11: Resumen de coste-facturación

36. Licencia

Todo el sistema, tanto la API como la aplicación, se encuentran dispuestas públicamente con una licencia **GNU Affero General Public License v3.0**. El contenido de la misma se ofrece a continuación.

GNU AFFERO GENERAL PUBLIC LICENSE

Version 3, 19 November 2007

Copyright (C) 2007 Free Software Foundation, Inc. https://fsf.org

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU Affero General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this

License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official

standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of

copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your

work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This

alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions.

Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for

any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is

reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if

the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting

any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING

WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

```
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.

    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If your software can interact with users remotely through a computer
network, you should also make sure that it provides a way for users to
get its source.  For example, if your program is a web application, its
interface could display a "Source" link that leads users to an archive
of the code.  There are many ways you could offer source, and different
solutions will be better for different programs; see section 13 for the
specific requirements.

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU AGPL, see
<https://www.gnu.org/licenses/>.
```

37. Modelado de datos

En este anexo se proporciona el modelado de las entidades de datos definidos en Capítulo 19, tanto en la API como en la aplicación; así como otros tipos o entidades de datos que han sido referidas en secciones como Capítulo 20.

37.1. Esquemas de la base de datos

La base de datos ha sido modelada con la API en mente al ser el sistema que se encargará de la gestión de la misma. Puesto que la base de datos será MongoDB y el lenguaje de la API será TypeScript se ha decidido usar la librería Typegoose (Apartado 24.2.2.21).

En esta librería se definen las entidades que se almacenarán en la base de datos por medio de esquemas que luego son convertidos a objetos y modelos con los que se llevarán a cabo las operaciones. La definición de los esquemas se lleva a cabo por medio de decoradores de Typescript a través de los que se anotan las características de la entidad y de sus propiedades y que se usarán por parte de la librería para la validación de los datos y la realización de las operaciones. Los esquemas modelados son los siguientes:

37.1.1. MessageSchema

```
1 export enum MessageType {
2   Task = "task",
3   Text = "text"
4 }
```

Fragmento 37.1: Enumerado de MessageType

```
1 @modelOptions({ schemaOptions: { collection: "messages" } })
2 export class MessageSchema {
3
4   @prop({ required: true, ref: () => UserSchema })
5   public submitter!: Ref<UserSchema>;
6
7   @prop({ required: true }) // Cached
8   public username!: string;
9
10  @prop({ required: true })
11  public timestamp!: number;
12
```

```
13    @prop({ required: true })
14    public lastUpdate!: number;
15
16    @prop({ required: true, select: false })
17    public room!: string;
18
19    @prop({ required: true })
20    public type!: MessageType
21
22  }
```

Fragmento 37.2: Esquema de Message

37.1.2. TaskMessageSchema

```
1  export class TaskMessageSchema extends MessageSchema {
2
3    @prop({ required: true })
4    public title!: string;
5
6    @prop()
7    public description?: string;
8
9    @prop({ required: true })
10   public done!: boolean;
11
12 }
```

Fragmento 37.3: Esquema de TaskMessage

37.1.3. TextMessageSchema

```
1  export class TextMessageSchema extends MessageSchema {
2
3    @prop({ required: true })
4    public message!: string;
5
6  }
```

Fragmento 37.4: Esquema de TextMessage

37.1.4. NotificacionSchema

```
1  export enum Action {
2    BOND_CREATED = "bond_created",
3    BOND_DELETED = "bond_deleted",
4    TASK_CREATED = "task_created",
```

```
5   TASK_DELETED = "task_deleted",
6   TASK_DONE = "task_done",
7   TASK_UNDONE = "task_undone",
8   LOCATION_SHARING_START = "location_sharing_start",
9   LOCATION_SHARING_STOP = "location_sharing_stop"
10 }
```

Fragmento 37.5: Enumerado de Action

```
1  @modelOptions({ schemaOptions: { collection: "notifications" }})
2  export class NotificationSchema {
3
4    @prop({ required: true })
5    public action!: Action;
6
7    @prop({ required: true })
8    public instigator!: string;
9
10   @prop({ required: true })
11   public timestamp!: number;
12
13   @prop({ type: String, default: [] })
14   public tags?: mongoose.Types.Array<string>;
15
16   @prop({ ref: () => UserSchema })
17   public interested!: Ref<UserSchema>[];
18
19 }
```

Fragmento 37.6: Esquema de Notification

37.1.5. SessionSchema

```
1  @modelOptions({ schemaOptions: { collection: "sessions" } })
2  export class SessionSchema {
3
4    @prop({ required: true, unique: true })
5    public refresh!: string;
6
7    @prop({ required: true })
8    public auth!: string;
9
10   @prop({ required: true })
11   public expiration!: number;
12
13 }
```

Fragmento 37.7: Esquema de Session

37.1.6. UserSchema

```
export enum Role {
  Keeper = "keeper",
  Patient = "patient",
  Blank = "blank"
}
```

Fragmento 37.8: Enumerado de Role

```
export interface Address {
  firstLine?: string,
  secondLine?: string,
  locality?: string,
  region?: string
}
```

Fragmento 37.9: Esquema de Address

```
export class UserSchema {

  @prop({ required: true, unique: true, select: false })
  public googleId?: string;

  @prop({ required: true, enum: Role })
  public role!: Role;

  @prop()
  public displayName?: string;

  @prop()
  public mainPhoneNumber?: string;

  @prop()
  public altPhoneNumber?: string;

  @prop()
  public address?: Address;

  @prop()
  public email?: string;

  @prop({ ref: () => UserSchema })
  public bonds?: mongoose.Types.Array<Ref<UserSchema>>;

  @prop({ ref: () => UserSchema })
  public cared?: Ref<UserSchema>;

```

```
30 }
```

Fragmento 37.10: Esquema de User

37.2. Tipos auxiliares y DTOs

De cara a las comunicaciones entre la API y la aplicación también existen los siguientes tipos o DTOs representando súperconjuntos de las entidades o subconjuntos de las propiedades de dichas entidades. En este documento se han referenciado las siguientes:

37.2.1. Message

Conjunto unión de las entidades TaskMessage (Apartado 19.1.2) y TextMessage (Apartado 19.1.3). Un elemento de tipo Message puede ser tanto una tarea como un mensaje de texto. También incluye a las representaciones parciales de estas entidades como TaskMinDto.

```
1 export type Message = Partial<TextMessage | TaskMessage>;
```

Fragmento 37.11: Tipo de Message

37.2.2. TaskMinDto

Representación mínima de una tarea. Contiene únicamente la información imprescindible para la creación y persistencia de una tarea. Será la información proporcionada por la aplicación a la API cuando se cree una nueva tarea. El resto de campos obligatorios serán suministrados por la propia API o la base de datos en el momento de la persistencia.

Los datos del autor de la tarea son agregados bajo un mismo objeto del tipo UserMinDto (Apartado 37.2.6). El esquema definitivo de este es el siguiente:

```
1 export interface TaskMinDto {
2   title: string,
3   description?: string
4   submitter: UserMinDto,
5   done: boolean,
6   timestamp: number,
7   lastUpdate: number
8 }
```

Fragmento 37.12: DTO mínimo de Task

37.2.3. TaskDto

Objeto representativo de una tarea completa. Contiene todos los campos relevantes para gestionar y mostrar una tarea desde el cliente. Se ignora el campo **room** al ser necesario únicamente para la gestión interna de la API. Este objeto es una ampliación de TaskMinDto con la única adición de los campos de identidad y tipo. Por tanto, la representación del autor es también un objeto UserMinDto (Apartado 37.2.6).

```
1 export interface TaskDto extends TaskMinDto {
2   _id: string,
3   type: MessageType
4 }
```

Fragmento 37.13: DTO de Task

37.2.4. NotificationDto

Objeto representativo de una notificación con la información necesaria para su gestión por parte de los usuarios individuales de la aplicación. No cuenta con el campo **interested** pues sólo es necesario para conocer los usuarios relacionados con la entidad en la recuperación de los datos.

```
1 export interface NotificationDto {
2   _id: string,
3   action: Action,
4   instigator: string,
5   timestamp: number,
6   tags?: string[]
7 }
```

Fragmento 37.14: DTO de Notification

37.2.5. SessionDto

Objeto representativo de la sesión con los dos tokens necesarios para manejar la sesión y el instante de caducidad de la sesión. Contiene todos los datos del modelo (Apartado 19.3) y el esquema (Apartado 37.1.5).

```
1 export interface SessionDto {
2   auth: string,
3   refresh: string,
4   expiration: number
5 }
```

Fragmento 37.15: DTO de Session

37.2.6. UserMinDto

Objeto mínimo representativo de un usuario para acompañar al resto de entidades en las comunicaciones para encapsular los datos del usuario. Únicamente contiene la **id** y el **displayName**.

```
1 export interface UserMinDto {
2   _id: string,
3   displayName: string
4 }
```

Fragmento 37.16: DTO mínimo de User

37.2.7. UserPublicDto

Objeto compuesto únicamente con las propiedades de un usuario que pueden ser compartidos con sus usuarios. Es una ampliación respecto a UserMinDto y un subconjunto de las propiedades de UserDto, respecto al cual no cuenta con la **id** de la entidad.

```
1 export interface UserPublicDto {
2   role: Role,
3   displayName: string,
4   mainPhoneNumber?: string,
5   altPhoneNumber?: string,
6   address?: Address,
7   email?: string,
8 }
```

Fragmento 37.17: DTO público de User

37.2.8. UserDto

Objeto representativo de un usuario con la información necesaria para el funcionamiento de la aplicación móvil. Carece de las relaciones con los otros usuarios y de la **googleId** pues son únicamente necesarios para responder a las peticiones por parte de la API. Es, por tanto, una versión ampliada de UserPublicDto.

```
1 export interface UserDto extends UserPublicDto {
2   _id: string
3 }
```

Fragmento 37.18: DTO de User

38. Acción de despliegue

A continuación se muestra el archivo utilizado para gestionar el despliegue de la API en la nube de Azure por medio de GitHub Actions.

```
1  # Docs for the Azure Web Apps Deploy action: https://github.com/Azure/
       webapps-deploy
2  # More GitHub Actions for Azure: https://github.com/Azure/actions
3
4  name: Build and deploy Node.js app to Azure Web App - 4l1service
5
6  on:
7    push:
8      branches:
9        - main
10   workflow_dispatch:
11
12 jobs:
13   build:
14     runs-on: ubuntu-latest
15
16     steps:
17       - uses: actions/checkout@v2
18
19       - name: Set up Node.js version
20         uses: actions/setup-node@v1
21         with:
22           node-version: '14.x'
23
24       - name: npm install, build, and test
25         run: |
26           npm install
27           npm run build
28           npm run test
29       - name: Upload artifact for deployment job
30         uses: actions/upload-artifact@v2
31         with:
32           name: node-app
33           path: .
34
35   deploy:
36     runs-on: ubuntu-latest
37     needs: build
38     environment:
39       name: 'Production'
```

```
40        url: ${{ steps.deploy-to-webapp.outputs.webapp-url }}
41
42      steps:
43        - name: Download artifact from build job
44          uses: actions/download-artifact@v2
45          with:
46            name: node-app
47
48        - name: 'Deploy to Azure Web App'
49          id: deploy-to-webapp
50          uses: azure/webapps-deploy@v2
51          with:
52            app-name: '4l1service'
53            slot-name: 'Production'
54            publish-profile: ${{ secrets.
                  AZUREAPPSERVICE_PUBLISHPROFILE_AF3E5424A8E840ABBE914263143E6F56
                   }}
55            package: .
```

Fragmento 38.1: Archivo de despliegue

39. Documentación adicional

Adjuntos a este documento se entregan también la siguiente documentación adicional:

- Archivo comprimido `Binarios` con los binarios de ejecución del proyecto:
  - Archivo `app-release.apk` para la instalación de la aplicación en un dispositivo Android.
- Archivo comprimido `Código` con el código fuente del proyecto:
  - Fichero `4l1-app` con todo el código fuente de la aplicación móvil.
  - Fichero `4l1-doc` con todo el código fuente en LaTeX de este documento.
  - Fichero `4l1-service` con todo el código fuente de la API.
- Archivo comprimido `Adicional` con el resto de documentación complementaria del proyecto:
  - Archivo `lib.bib` con la bibliografía completa utilizada en el documento.
  - Archivo `Planificación.pod` con la planificación del proyecto creada con ProjectLibre.
  - Archivo `Presupuesto.xlsx` con el presupuesto completo del proyecto creado con Microsoft Excel.
  - Fichero `Diagramas` con todos los diagramas del proyecto. El código de los diagramas creados con PlantUML también se adjuntan en esta carpeta.
  - Fichero `Pantallas` con los mock-ups y los diseños definitivos de las diferentes pantallas de la aplicación.

Glosario

endpoint Extremo o punto de escucha de una comunicación definido en una API REST a través del cual se realizan y responden las peticiones de los clientes.. 46, 51, 52, 89, 125, 129, 173, 188, 191, 192, 195, 196, 265, 311, 312

join Operación de consultas a base de datos que obtiene datos de más de una tabla y las devuelve en una nueva vista generada combinando campos o columnas de las diferentes tablas.. 48, 169

middleware Software que sirve de puente entre módulos de un sistema o entre este y otras entidades como las bases de datos o el sistema operativo.. 264–266

mock Imitación de un componente de software que ofrece las mismas funciones que el componente al que replica para poder probar características sin utilizar el componente original. Por lo general los mocks ofrecen la capacidad de especificar el código a ejecutar cuando se llama a funciones concretas o también sistemas para monitorizar los usos que ha recibido el mock para garantizar que ha sido invocado de la forma que se espera invocar al componente original.. 125, 126, 261

seudonimización Reemplazamiento de campos de información personal dentro de un registro de datos por uno o más identificadores artificiales o pseudónimos.. 311

token Un token es el resultado de un proceso de sustitución de datos sensibles por un equivalente no sensible, sin valor extrínseco o explotable. Una aplicación habitual de estos tokens es para la representación de los datos de una cuenta o sesión en forma de una cadena de texto codificada.. 74, 75, 171–173, 188, 191, 192, 206, 263, 266, 344

uri Un identificador uniforme de recurso, o URI según sus siglas en inglés, es una cadena de caracteres breve que contiene el nombre o dirección que identifica o dirige a un objeto de la web. 47

Siglas

AAB Android Application Bundle. 130

API Application Programming Interface. 33, 35, 46, 47, 51, 52, 54–56, 60, 63–65, 74–89, 125–127, 129–132, 169, 188, 191–196, 257, 261–265, 267–269, 273, 274, 277, 278, 282, 302–306, 311, 312, 339, 343–346, 348

APK Android Package. 130, 270, 289

CEAFA Confederación Española de Alzheimer. 37

DTO Data Transfer Object. 173, 174, 176, 179, 181, 183–187, 343

GPS Global Positioning System. 55, 63, 81

HTTP Hyper Text Transfer Protocol. 33, 46, 47, 64, 86–89, 125, 131, 132, 195, 196, 262, 264–267, 274, 278

HTTPS Hyper Text Transfer Protocol Secure. 278

IDE Integrated Development Environment. 53, 62, 270, 272, 273, 279, 310

IETF Internet Engineering Task Force. 47

JSON JavaScript Object Notation. 47, 57, 63, 266

JVM Java Virtual Machine. 53–55, 62, 257

JWT Json Web Token. 64

MVVM Model-View-View Model. 62, 85, 87, 130, 275

NPM Node Package Manager. 64, 269

REST Representational State Transfer. 33, 46, 51, 54, 55, 63, 64, 74–80, 83, 87–89, 129, 131, 262, 264, 267, 274, 312

SDK Software Development Kit. 62, 262

SQL Structured Query Language. 47, 48

TCP Transmission Control Protocol. 47, 89

Bibliografía

[1] C. E. de Asociaciones de Familiares de Personas con Alzheimer y otras Demencias y Fundación Sanitas., "El cuidador en españa. contexto actual y perspectivas de futuro. propuestas de intervención.," 2016.

[2] R. T. Fielding, *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.

[3] I. Fette and A. Melnikov, "The websocket protocol," rfc-6455, RFC Editor, 12 2011.

[4] P. Miller, "Google is adding kotlin as an official programming language for android development," 5 2017.

[5] F. Lardinois, "Kotlin is now google's preferred language for android app development," 5 2019.

[6] DB-Engines.com, "Db-engines ranking - popularity ranking of database management systems," 12 2021.

[7] E. Jones, "Cloud market share – a look at the cloud ecosystem in 2022," *Kinsta*, 9 2021.

[8] P. M. Institute, *A Guide to the Project Management Body of Knowledge: PMBOK Guide*. Project Management Institute, 5 ed., 2013.

[9] statcounter GlobalStats, "Mobile & tablet android version market share worldwide," 10 2021.

[10] Google, "Guide to app architecture," *Android Developers*, 12 2021.

[11] D. Hardt, "The oauth 2.0 authorization framework," rfc-6749, RFC Editor, 10 2012.

[12] E. International, *Standard ECMA-262 - ECMAScript Language Specification*. ECMA International, 5.1 ed., 6 2011.

[13] M. Jones and D. Hardt, "The oauth 2.0 authorization framework: Bearer token usage," rfc-6750, RFC Editor, 10 2012.

[14] M. Jones, J. Bradley, and N. Sakimura, "Json web token (jwt)," rfc-7519, RFC Editor, 5 2015.

[15] Google, "Livedata overview," *Android Developers*, 11 2021.

[16] GitHub, "The 2021 state of the octoverse," 2021.

[17] J. Holmgren, "Yarn 1 vs yarn 2 vs npm," *Infinite Red*, 6 2020.

[18] T. Warren, "Microsoft hits its goal of 1 billion devices running windows 10," 3 2020.

[19] StackOverflow, "2021 developer profile - integrated development environment," 2021.

[20] Cosmos, "Cómo instalar aplicaciones en apk en un móvil android," 10 2020.