UNIVERSIDAD DE OVIEDO

ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Aplicación móvil para el registro de entrada en locales

DIRECTOR: Miguel Riesco Albizu

AUTOR: Raúl Pérez Molinero

Yo, Raúl Pérez Molinero, declaro que este documento basado en la plantilla oficial para trabajos de fin de grado de la Escuela de Ingeniería Informática ha sido realizado de manera íntegra por mí. Además, declaro que se trata de una obra original, en la que las fuentes utilizadas para obtener información han sido citadas a lo largo del documento y se encuentran presentes en la bibliografía del mismo.

Agradecimientos

Como nunca sé que decir en estas situaciones, voy a ser breve.

Quiero agradecer en primer lugar a mis padres Emilio y Charo, y a mi hermana Marta, todo el apoyo que me han dado, no solo durante la realización de este trabajo, sino durante toda mi vida.

A todos los profesores que he tenido durante mi etapa universitaria, muchas gracias. En especial, como no puede ser de otro modo, muchísimas gracias por todo Albizu, tanto por tu papel como profesor en las asignaturas de Sistemas Operativos y Seguridad de Sistemas Informáticos, como por supuesto por el apoyo y los consejos que me has dado durante la realización de este proyecto.

Por último, quiero acabar acordándome de todos aquellos que han estado ahí compartiendo estos momentos conmigo, ya sea desde dentro de la facultad o viendo los toros desde la barrera. Muchas gracias, Rubén, Riesgo, Isaac, Iyán, Jony, Kevin, Iván, Jorge, Miguel, Álvaro, Luis y, por supuesto, muchas gracias a todos los Cyberpapis.

Resumen

Este proyecto busca desarrollar un sistema de rastreo de contagios de COVID-19 producidos en el interior de locales mediante una aplicación móvil basada en el escaneo de códigos QR asignados a cada establecimiento, con el fin de registrar de forma anónima las visitas de los usuarios a dichos locales y poder notificar los brotes que se produzcan en ellos. El producto final desarrollado en este proyecto pretende servir de prototipo para que en un futuro se puedan implementar sistemas de esta índole a nivel nacional con el fin de controlar la expansión de futuras enfermedades cuya propagación se realice por medio del contacto estrecho entre infectados o por aerosoles.

Con la privacidad de los usuarios como foco del proyecto, el rastreo se llevará a cabo sin tratar datos personales, y todos los datos que recopilen las aplicaciones móviles relativos a las visitas a locales se almacenarán internamente en los dispositivos. Estos dispositivos móviles serán los encargados de, periódicamente, realizar llamadas por medio de una API REST al servidor para obtener posibles alertas, que será procesadas en el dispositivo con el fin de obtener si dicha alerta afecta al usuario o no. Además, el sistema de rastreo incorporará una aplicación web desde la cual se podrán realizar actividades como dar de alta un local o buscar si un local determinado usa el sistema de rastreo. Por último, con el objetivo de mantener un grado de efectividad lo más alto posible con independencia de las mutaciones del virus que puedan aparecer, las variables utilizadas para determinar periodos de infectividad, periodos de aparición de síntomas, etcétera, podrán ser modificadas por un administrador en tiempo real sin necesidad de lanzar una nueva versión del sistema.

Para llevar a cabo este proyecto se ha seguido una metodología ágil, lo cual permite obtener pronto un producto funcional que pueda ser puesto en producción para comenzar cuanto antes con el rastreo, además de agilizar la toma de decisiones y las respuestas a los cambios que puedan surgir a medida que el proyecto y el virus van evolucionando. Para acompañar esta metodología ágil, el servidor y la página web se han desarrollado con Node.JS y los *frameworks* Express y React.JS respectivamente. En el caso de la aplicación web, con el objetivo de que esté disponible para los sistemas operativos móviles más importantes, se decidió implementar con Xamarin.Forms, combinando así la compatibilidad multiplataforma y un rendimiento similar al de las aplicaciones nativas.

Palabras Clave

COVID-19, Xamarin.Forms, Node.JS, Código QR, Rastreo, Metodología Ágil.

Abstract

The aim of this project is to develop a COVID-19 tracking system for those infections produced inside places such as restaurants, cinemas, supermarkets, residencies. By scanning QR codes associated to each place, the mobile application will register anonymously the visits made by the user. This way, the system will be able to notify the user about new outbreaks that may happen inside those places. The final product developed in this Project aims to serve as a prototype of a future system that can be implanted in the whole country to control the expansion of future diseases whose propagation is done via direct contact between humans or via aerosols.

Privacy in this project is key. Therefore, the tracking Will be carried on without the need of acquiring any kind of personal data. Moreover, visits registered by the application will be stored internally in the user's smartphone or tablet. These devices are responsible for making calls to the server via a REST API to obtain new alerts. These alerts are all processed locally by the application in order to find out if any of them affects the user. In addition to the mobile app and the server, this system incorporates a webpage from which the users will be able to register their establishments and to search if a specific place is using the tracking system. Finally, to maintain the effectiveness of the tracking system no matter the virus mutations, the administrator of the system will be able to use a control panel existing in the webpage to update in real time every single variable used to calculate infective periods, the average time that symptoms take to appear, and so on, without the need of having to redeploy the system or to launch a new version of the app.

To carry out this project, an agile methodology has been followed. This choice speeds up the process of having a first functional product that can be put into production to start tracking down the virus as soon as possible. Moreover, due to the nature of the project, changes are going to appear, and so it is important to follow a working methodology that helps dealing with them in an agile and efficient way. To join the agile approach, the server and the webpage have been developed with Node.JS and the frameworks Express and React.JS respectively. The mobile application, however, in order for it to be available for the most important mobile operating systems of the market while still having almost the same performance as a native app, has been developed with the .NET library Xamarin.Forms.

Keywords

COVID-19, Xamarin.Forms, Node.JS, QR Code, Tracking, Agile Methodology.

Índice General

Índice de Figuras

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

La figura del rastreador surge como consecuencia de la gran capacidad de propagación del COVID-19 y a su difícil control. La función de estos rastreadores consiste en, una vez que se ha detectado un caso positivo del virus, ponerse en contacto con dicha persona para averiguar con que gente se ha relacionado en los últimos días y así poder detectar y aislar a los nuevos positivos lo antes posible, minimizando el riesgo de que estos posibles contagiados puedan seguir propagando el virus. Además, las personas que hayan sido contagiadas, al detectarse dicha situación incluso antes de manifestar síntomas, podrán someterse a tratamiento desde un primer momento y minimizar así los posibles síntomas graves que puedan llegar a desarrollar.

Debido a la enorme cantidad de casos nuevos que se detectan todos los días, rápidamente se vio la necesidad de complementar la figura del rastreador con algún sistema que permitiera realizar dicho rastreo por medios informáticos, agilizando y optimizando las tareas manuales que conlleva rastrear un brote. Como consecuencia de esto, en España surge Radar Covid, una aplicación móvil ideada para detectar posibles contagios por vía del contacto estrecho entre un positivo y la gente con la que se relaciona. Sin embargo, debido a su funcionamiento y al momento de tensión política y social vivida en España, esta aplicación produjo mucho rechazo en gran parte de la población. Al no ser obligatoria, la gran mayoría decidió no usarla y, por tanto, su utilidad fue y sigue siendo residual.

En los últimos meses, a medida que las restricciones impuestas por el gobierno han ido relajándose, los principales focos de contagio del virus han pasado a ser los locales de ocio y restauración, así como residencias o centros comerciales. Dado que en estos lugares los contagios se producen entre gente desconocida, y a un rastreador le es prácticamente imposible averiguar con que gente has mantenido contacto en este tipo de establecimientos, surge la necesidad de un sistema que permita identificar a estas personas con el fin de notificarles su posible exposición al virus.

Este sistema, gracias a una aplicación móvil, permitirá relacionar a todos las personas que coincidan durante un determinado tiempo en un mismo local. Sin embargo, a la hora de llevar esto a cabo, es fundamental que se respete la privacidad de los usuarios y no se transmitan datos personales, con el fin de ganar la confianza de la población y fomentar el uso de esta aplicación.

Con estas premisas claras, el funcionamiento del sistema planteado es muy simple. Los clientes de un local escanearán un código QR en el momento de la entrada y salida de dicho local. AL hacer esto, se registrará en su propio dispositivo, de manera local y sin almacenar ningún dato personal, que ha visitado dicho establecimiento. De manera periódica, la aplicación buscará nuevas alertas que se hayan podido producir en los locales visitados por el usuario,

procesándolas en el dispositivo móvil en busca de coincidencias espaciotemporales. Dichas alertas las crearán los propios usuarios también mediante el uso de la aplicación móvil, notificando al sistema su positivo, y enviando los locales y las horas a las que se visitaron, pero sin que en ningún momento se transfiera la identidad o algún dato que permita identificar al usuario que notificó dicho contagio.

1.2 Resumen de Todos los Aspectos

En este apartado se indica brevemente el contenido de cada uno de los capítulos posteriores que forman este documento.

- Capítulo 2. Introducción: En este capítulo se describen la justificación del proyecto y sus objetivos principales. A continuación, se explica la situación sanitaria actual y como se realizan las labores de rastreo, a la vez que se analizan otros sistemas de rastreo que se encuentran actualmente en uso.
- Capítulo 3. Aspectos Legales: Este capítulo repasa la normativa vigente en cuestión de políticas de privacidad y tratamiento de datos, y propone las medidas que se tomarán en el sistema para cumplir con dichas directrices.
- Capítulo 4. Aspectos Teóricos: Aquí se explican los fundamentos teóricos en los que se basa el proyecto, desde un análisis del COVID-19 hasta las metodologías y tecnologías usadas.
- Capítulo 5. Planificación del Proyecto y Resumen del Presupuesto: En este quinto capítulo se describen la planificación y el cálculo del presupuesto realizados previo al desarrollo del proyecto.
- Capítulo 6. Seguimiento y evolución del proyecto: Este capítulo describe como se llevó a cabo el seguimiento durante el proyecto y muestra la evolución del mismo con el paso del tiempo.
- Capítulo 7. Análisis: En este capítulo se detallan las labores de análisis realizadas durante el desarrollo del proyecto y se muestran los productos obtenidos tras dichas tareas de análisis.
- Capítulo 8. Diseño del Sistema: El octavo capítulo de este documento contiene la descripción del diseño del sistema elaborado. En el se describen su arquitectura, sus módulos, las interfaces de usuario presentes en él y el plan de pruebas a llevar a cabo.
- Capítulo 9. Implementación del Sistema: Este se trata del capítulo más técnico del documento. En él se describen las tecnologías y herramientas utilizadas para el desarrollo del sistema de rastreo, se explican los principales problemas encontrados durante dicho desarrollo y se detalla la composición de las clases más complejas e importantes del sistema
- Capítulo 10. Desarrollo de las pruebas: Este capítulo muestra los resultados de la ejecución del plan de pruebas especificado en el capítulo 8.
- Capítulo 11. Manuales del sistema: En este capítulo se recopilan los manuales de instalación y ejecución del sistema junto con las guías de usuario y un manual de ayuda para futuros programadores que quieran ampliar el sistema.
- Capítulo 12. Conclusiones y ampliaciones: Este capítulo contiene las conclusiones obtenidas tras la realización del proyecto, y recopila una serie de ampliaciones propuestas para mejorar el sistema desarrollado.

- Capítulo 13. Presupuesto: Aquí se explica de manera más detallada el cálculo de presupuestos realizado al inicio del proyecto.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

A finales de diciembre del año 2019, un brote de una neumonía desconocida originado en la ciudad de Wuhan, China, se comenzó a propagar por todo el mundo. A comienzos de enero el virus ya se había propagado fuera del país asiático, y el 11 de marzo de 2020, la OMS (Organización Mundial de la Salud) declaró el COVID-19 como pandemia mundial. Desde esa fecha hasta la actualidad, el virus ha estado presente en todos los ámbitos de nuestra sociedad. Tras superar unos primeros meses muy duros, en los que la labor del personal sanitario e investigador, así como las severas medidas adoptadas por los gobiernos de cada país, fueron claves, se lograron paliar los efectos adversos de la pandemia y reducir tanto los contagios como las muertes en la mayor parte de los países afectados.

A medida que la pandemia se debilitaba, las medidas adoptadas por los gobiernos se relajaron. En España, el 21 de junio de 2020 finalizaba el Estado de Alarma, poniendo fin a las limitaciones de movilidad de los ciudadanos. Habiendo relajado gran parte de las medidas de prevención, tanto las restricciones de movilidad como las que afectaban a la interacción entre personas, el papel del ciudadano a la hora de contener la transmisión del virus y de prevenir contagios se hizo fundamental.

Durante todo este tiempo se desarrollaron numerosos estudios científicos en laboratorios de todo el mundo, cuyo objetivo común era analizar los motivos y vías de transmisión del SARS-CoV-2, así como comprobar que medidas son efectivas para prevenir los contagios. Una de las principales conclusiones alcanzadas en diversos ensayos es que la transmisión del virus en locales cerrados con poca o nula ventilación aumenta drásticamente con respecto a los espacios abiertos. El aumento de la probabilidad de contagio, sumado a la interacción con un mayor número de personas ajenas a un mismo núcleo de convivencia, convierten a lugares como cines, supermercados, restaurantes, escuelas o gimnasios, en los sitios con más peligrosos de cara a una posible transmisión del virus. Además, dado que son lugares que no suelen llevar un registro de las personas que acuden y las franjas horarias en las que lo hacen, el rastreo de posibles contagios se vuelve muy complicado.

El rastreo de posibles contagios es clave de cara a un rápido aislamiento y seguimiento de los positivos en coronavirus. Con un rastreo eficaz se logra disminuir el número de personas que entran en contacto con un positivo, así como detectar antes dichos positivos y por tanto comenzar antes con un tratamiento en caso de que fuera necesario.

De la necesidad de agilizar el rastreo y prevenir contagios en lugares cerrados surge este proyecto. La idea principal era conseguir crear un método de registro eficaz al que los rastreadores de cada comunidad autónoma puedan acudir cuando se reporta un caso positivo. De la misma manera, dándole a la población una herramienta con la que puedan estar constantemente informados sobre si han podido estar en contacto con algún positivo dentro de un ambiente que favorezca la transmisión del virus, se consigue que aquellos afectados sean conscientes de ello y tomen las medidas correspondientes.

A la idea original se le suma el concepto de la privacidad de los datos de actividad de los usuarios. Uno de los puntos más importantes de este proyecto es que el usuario sea el dueño de sus datos hasta el momento que sean requeridos para las labores de rastreo. Mediante un tratamiento de datos descentralizado, la actividad de los usuarios no será visible en ningún caso para los locales a los que acude. El sistema almacenará en el servidor central únicamente los registros de los infectados que se hayan producido dentro de los plazos marcados por las autoridades sanitarias en los que exista posibilidad de contagio.

El tercer punto clave del proyecto es el método elegido para llevar el registro de la actividad de los usuarios. Teniendo en cuenta las vías de contagio del virus, así como la necesidad de que se realice de una manera ágil y cómoda para el usuario, las tecnologías *contactless* emergieron como principales candidatas. Dentro de este segmento, las más comunes son los códigos QR (sustituyen a los códigos de barras tradicionales), los sistemas NFC (Near Field Communication) y la identificación por radio RFID (Radio Frequency Identification).

Juntando estas tres características, en este proyecto se pretende desarrollar una aplicación móvil multiplataforma, accesible para el mayor número de usuarios posible, que mediante tecnologías *contactless*, registre la actividad de dichos usuarios en los distintos locales a los que acudan. Esta información, con su debida política de privacidad y tratamiento de datos acorde a la normativa vigente, almacenará en el dispositivo móvil la actividad del usuario, y se comunicará con un servidor centralizado con el fin de obtener alertas sobre nuevos positivos que puedan haber coincidido en una misma franja espacial y temporal con el usuario. A su vez, mediante una aplicación web, los propietarios de los respectivos locales (restaurantes, cines, gimnasios, etc.) podrán darlos de alta en el sistema para que se puedan realizar las labores de rastreo.

2.2 Objetivos del Proyecto

A continuación, se describen los objetivos principales que se persiguen con la realización de este proyecto:

- Diseñar un sistema de registro de entradas en locales que facilite las labores de rastreo de contagios del virus COVID-19 que se puedan producir en el interior de los mismos.
- Adaptar un sistema de rastreo de coronavirus que se ha demostrado efectivo en otros países a la sociedad y las políticas de privacidad europeas.
- Proporcionar al administrador del sistema un mecanismo para poder modificar los parámetros que determinan los posibles contagios que se puedan producir, con el fin de mejorar la efectividad del sistema a medida que se realizan avances en la comprensión del virus.
- Desarrollar un sistema de rastreo que pueda ser utilizado no sólo durante la actual pandemia de coronavirus, sino en un futuro para rastrear todos aquellos virus cuyas vías de contagio sean el contacto directo y los aerosoles.
- Proporcionar un mecanismo de control de aforo para locales.

2.3 Estudio de la Situación Actual

En junio de 2020 surgió en España la figura del rastreador. Formado tanto por voluntarios como por personal cualificado o incluso por miembros de las fuerzas de seguridad del estado, el cuerpo de rastreadores tiene como principal función, dado un nuevo positivo de coronavirus, ponerse en contacto con dicha persona para averiguar quiénes son sus contactos estrechos y la gente con la que ha estado recientemente. De esta manera se busca realizar un seguimiento del avance del virus y se trata de prevenir que posibles nuevos infectados sigan propagando el virus.

El método de rastreo actual es bastante precario. En el momento que una persona da positivo en una prueba de COVID, un rastreador se pondrá en contacto por vía telefónica con dicha persona. El rastreador preguntará al contagiado por sus contactos estrechos. Se denomina contacto estrecho a todo aquel que se haya relacionado con el infectado a una distancia menor que la distancia de seguridad (en España se considera que la distancia de seguridad son 2 metros) durante al menos 15 minutos en los 2 días previos a la aparición de los síntomas. Para los asintomáticos, el periodo es de 2 días antes del diagnóstico. Tras obtener esta información, el rastreador llamará por teléfono uno a uno a dichos contactos estrechos para informarles de la situación y agendarles la realización de una prueba PCR.

Pese a que este método es eficaz para obtener las personas con las que más se relacionó el infectado, posibles contagiados como los camareros de un restaurante en el que estuviera cenando o una persona que coincidiera con él en un gimnasio nunca serían contactados por los rastreadores, pese a existir una gran probabilidad de contagio por aerosoles. Para tratar de solventar este problema, la Secretaría de Estado de Digitalización e Inteligencia Artificial del Gobierno de España diseñó y dirigió el desarrollo de la aplicación móvil Radar COVID.

2.3.1 Radar COVID

Esta aplicación móvil disponible tanto para Android como para iOS fue diseñada por la Secretaría de Estado de Digitalización e Inteligencia Artificial del Gobierno de España con el fin de automatizar las labores de rastreo. Esta herramienta no se planteó como sustituta de los rastreadores, sino que su objetivo es el de liberar de carga a un cuerpo que en las fases más duras de la pandemia se ve totalmente sobrepasado en muchas comunidades autónomas. Radar COVID se utiliza como herramienta de rastreo en 12 países de la Unión Europea.

El funcionamiento de Radar COVID se basa en el uso de la tecnología Bluetooth, que consiste en un enlace entre dispositivos por radiofrecuencia en la banda ISM de los 2,4GHz. En un estudio realizado por el Gobierno de España en el año 2011, el 90,1% de los encuestados disponían de esta tecnología en sus dispositivos móviles. En 2018, la cantidad de dispositivos portátiles en el mundo que incorporaban tecnología Bluetooth era de más de 2.000 millones. Una vez instalada Radar COVID en el dispositivo y activado el Bluetooth, la aplicación se comunicará con otros dispositivos en un rango cercano que también tengan instalada la aplicación y el Bluetooth activo, almacenando en cada uno de los terminales la franja temporal en la que coincidieron. Si uno de los dos dispositivos reportase un contagio, la aplicación revisa las comunicaciones con otros dispositivos y lanzaría una alerta a aquellos que cumplieran los

requisitos de espacio y tiempo en contacto necesarios para ser considerados contactos estrechos.

Pese al gran número de dispositivos con Bluetooth, las recomendaciones del Gobierno de España y aun habiendo publicado el código íntegro de la aplicación en GitHub con el fin de dar trasparencia a Radar COVID, fruto del enfrentado clima político español y de la desconfianza generada por el desconocimiento, tan solo una pequeña parte de la población instaló Radar COVID en sus dispositivos (6.985.178 descargas), lo que llevó a que la aplicación detectase menos de 50.000 casos de coronavirus. Esto supone, a febrero de 2021, que tan solo el 2,06% de los casos de coronavirus han sido detectados gracias a Radar COVID. Estos datos contrastan mucho con los provenientes de otros países como por ejemplo Alemania, donde la aplicación Warn App, cuyo funcionamiento es muy similar al de Radar COVID, usando Bluetooth para establecer comunicaciones entre dispositivos cercanos, contaba ya en noviembre con más de 22,8 millones de descargas.

En este proyecto se pretende desarrollar un sistema que, gracias a su tratamiento de datos y un funcionamiento más transparente para el usuario, sea más atractivo y genere menos desconfianzas a la población, obteniendo de esta manera una mayor cantidad de usuarios y siendo por tanto una herramienta más útil para detectar posibles contagios. Además, gracias al uso de tecnologías que no requieren la comunicación directa entre dispositivos móviles, el nuevo sistema será capaz de determinar como contacto estrecho a una persona que no necesariamente lleve consigo su dispositivo en el momento del contacto, como puede ser el caso de un dependiente de supermercado o de alguien que comparta horario de gimnasio con un infectado.

2.3.2 Otras Alternativas

Habiendo mencionado Radar COVID y Warn App, dos aplicaciones cuya base tecnológica y el tratamiento de datos que realizan son similares, cabe pensar que esta ha sido la medida adoptada por la mayoría de los países para facilitar las labores de rastreo. Sin embargo, dado que la mayoría de los países más desarrollados cuentan con su propia aplicación de rastreo, existen muchas otras que usan tecnologías distintas al rastreo por Bluetooth, y por supuesto que manejan los datos personales y la privacidad del usuario de formas distintas.

La eficacia que han tenido las aplicaciones de rastreo en cada país depende en mayor medida de la aceptación que han tenido en el seno de la sociedad y de la rigidez que cada gobierno ha tenido a la hora de implantar como medida obligatoria su instalación o simplemente recomendando a la población que haga uso de ella. En países como Corea del Sur, la labor de estas aplicaciones fue vital para ganar la lucha al virus. Sin embargo, países europeos como España, Francia o Italia han demostrado que, si un porcentaje importante de la población no acepta su uso, la utilidad de estas aplicaciones es mínima.

A continuación, se explica en términos generales el funcionamiento de las aplicaciones usadas en China y Corea del Sur, que son dos ejemplos de países que han logrado erradicar casi por completo el virus gracias a un uso intensivo y eficaz de este tipo de sistemas de rastreo.

2.3.2.1 China: Geolocalización y control estricto de la población

Desde febrero de 2020, el gobierno chino estableció como una de sus medidas principales para controlar el avance de la pandemia un sistema mediante el cual a cada ciudadano se le asigna un color en función del grado de contacto que ha tenido con el virus. Pese a que cada gobierno regional o local puede tener su propia aplicación, el código de colores es común para todo el país. Un ciudadano que no haya estado expuesto al virus tendrá color verde, y podrá moverse libremente por la ciudad, pudiendo acceder a todo tipo de establecimientos escaneando un código QR. Sin embargo, si al ciudadano se le ha asignado el color naranja o el rojo, deberá guardar una cuarentena de 7 o 14 días respectivamente. Además de para acceder a la mayoría de los establecimientos, en la gran mayoría de trabajos se requiere del color verde para permitir al empleado ir a trabajar.

Pese a que cada región puede usar una aplicación distinta, el núcleo de la funcionalidad de todas ellas es el mismo. Tomaré como ejemplo la utilizada en Pekín, Health Kit. Esta aplicación pide al usuario su nombre, su número de teléfono, su documento de identificación personal y su foto y, mediante el uso de geolocalización, determina si el dispositivo ha estado en contacto con algún infectado. Todo el tratamiento de datos se realiza de manera centralizada por las autoridades chinas, y los datos obtenidos por la aplicación se usan en conjunto con los suministrados por las autoridades sanitarias y de transporte.

En China existe un gran control sobre la población y, por lo general, el debate sobre la privacidad del usuario no está a la orden del día, menos aún en una situación de emergencia sanitaria. Esto se debe a que las libertades individuales en el país asiático están muy restringidas y la población está obligada a facilitar este tipo de información al estado. Desde algunos países europeos se ha criticado mucho que las autoridades chinas recojan información personal y sensible sin tener en cuenta ninguna protección de datos. Sin embargo, este método de rastreo tan intrusivo y el control tan estricto ejercido sobre la población ha demostrado ser muy eficaz a la hora de contener el avance de la pandemia. Un año después de la implantación de este sistema, China no ha registrado ningún caso local de coronavirus durante una semana.

2.3.2.2 Corea del Sur: Geolocalización y códigos QR

En Corea del Sur, un país más liberal que China, el gobierno desarrolló un sistema de aplicaciones menos intrusivo que el chino. Estas aplicaciones son de descarga voluntaria tal y como ocurre en Europa. El gobierno de Corea hace uso de los registros de pagos con tarjetas, sistemas de videovigilancia y los rastreos realizados por GPS que realizan estas aplicaciones para determinar los contactos del usuario con infectados.

Durante la primera ola de la pandemia, Corea del Sur fue uno de los países pioneros a la hora de establecer el uso de estas aplicaciones, y también fue uno de los primeros que logró acabar parcialmente con el virus. Sin embargo, tras un brote ocurrido en junio en una discoteca de Seul, el gobierno coreano implantó el uso de una nueva aplicación que sería obligatoria para acceder a locales como bares o gimnasios. Esta aplicación genera un código QR personal para cada usuario, que será escaneado a la entrada de cada establecimiento. Mediante este escaneo se obtienen los datos del usuario con el fin de poder rastrearlo en caso de que se

detecte un positivo por coronavirus. Estos datos personales son almacenados de manera centralizada por el Ministerio de Salud surcoreano durante cuatro semanas.

Al igual que en China, la cultura social existente en Corea del Sur con respecto a la privacidad y el tratamiento de datos no suele cuestionar las decisiones del gobierno, confiando en todo momento en que las medidas tomadas son las más adecuadas, aunque esto suponga aportar a las autoridades información personal y que se ejerza una vigilancia constante sobre los movimientos de la población. El uso de estas aplicaciones ha resultado muy eficaz a la hora de prevenir nuevos contagios, y la pandemia en ambos países está casi controlada.

2.3.3 Integración con el sistema de rastreo actual

En España, el ciudadano tiene la libertad de decidir si quiere instalar en su dispositivo móvil alguna aplicación de rastreo, como es el caso de Radar COVID, o si prefiere no hacerlo. Dado que el gobierno en ningún caso se plantea regular su uso, para tratar de maximizar el número de personas que dispongan de la aplicación instalada, se ha de incentivar a la población para que la usen. Para ello no solo basta con ser un sistema totalmente transparente en cuanto a su funcionamiento, sino que dicho funcionamiento ha de ser fácilmente comprensible por gente con escasos conocimientos técnicos de informática y movilidad. Este es uno de los puntos en los que, desde mi punto de vista, Radar COVID fracasó, puesto que la comunicación entre dispositivos por medio del Bluetooth no es algo que el usuario pueda apreciar, como si ocurre en el sistema que se plantea en este proyecto, dónde el propio usuario es el que ha de escanear el código QR presente en el local al que va a acceder.

El segundo punto importante para incentivar el uso de la aplicación recae de la mano de los locales. En este proyecto se va a desarrollar una aplicación web que permita a los locales darse de alta en el sistema y obtener sus respectivos códigos QR. Desde una perspectiva económica, un local que quiera mantener un número alto de clientes ya sea un gimnasio, un bar o cualquier otro tipo de establecimiento, tendrá que estar abierto el mayor tiempo posible. Debido a la situación actual de pandemia, cuando aumenta la incidencia del virus en un territorio, la primera medida que toman los responsables políticos es siempre la de cerrar establecimientos en los que se pueda juntar mucha gente, con el fin de evitar la propagación del virus entre personas sin ninguna relación aparente. Si un local está dado de alta en este sistema estará contribuyendo de manera proactiva a rebajar la incidencia del virus gracias a posibilitar el rastreo, y por tanto previniendo que las administraciones públicas correspondientes cierren su negocio hasta que los efectos de la pandemia se vuelvan a relajar. Desde un punto de vista social, dar el local de alta en este sistema es aún más importante. Un establecimiento registrado que cuente con su propio código QR puede establecer como condición indispensable escanear el código QR con la aplicación de rastreo a la hora de acceder, prohibiendo la entrada a todo aquel que se niegue a hacerlo. De este modo se consigue que un mayor porcentaje de la población se instale la aplicación para no ser privados de poder disfrutar de estos establecimientos, sin que en ningún momento se esté obligando directamente al ciudadano a tenerla si va en contra de su voluntad.

De cara a la integración con el sistema actual de rastreo en España, este proyecto en ningún caso se plantea como sustituto de Radar COVID o de cualquier otra aplicación de rastreo que pueda crear el gobierno en un futuro. Al igual que Radar COVID, para que este sistema sea

funcional se ha de integrar dentro del protocolo de actuación sanitario frente al coronavirus. Para ello, el sistema contará con la funcionalidad que permitirá al usuario notificar de su resultado positivo en una prueba del mismo modo que se hace actualmente en Radar COVID. Para realizar esto de manera segura, evitando posibles reportes falsos, el sistema pedirá al usuario que introduzca un código numérico, personal e intransferible, de un solo uso que le habrá sido proporcionado por el personal sanitario a la hora de notificarle el resultado de la prueba. El sistema comprobará la validez del código y notificará el positivo al servidor para que se procedan a realizar las alertas correspondientes. Finalmente, en los teléfonos de contacto destinados únicamente al coronavirus, al igual que existe una opción para notificar alertas recibidas mediante la aplicación Radar COVID, las administraciones autonómicas deberían de habilitar una sección para notificar las alertas recibidas por este sistema con el fin de solicitar una prueba PCR.

Capítulo 3. Aspectos Legales

Este capítulo está destinado al estudio de la Ley Orgánica de Protección de Datos vigente en la normativa española, así como el Reglamento General de Protección de Datos (RGPD) aprobado en 2016 por el Parlamento Europeo y aplicado a partir de 2018, sobre el que se sustentan las leyes nacionales de los distintos países miembros de la Unión Europea.

Actualmente, en España, la Ley Orgánica 3/2018, 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales, regula como se debe realizar el tratamiento de los datos personales en los sistemas informáticos. Esta ley adapta al ordenamiento jurídico español el reglamento de la Unión Europea en lo que respecta a esta materia. Además, garantiza que se ejerza el derecho fundamental de las personas a la protección de datos personales, tal y como se recoge en el artículo 18.4 de la Constitución Española.

3.1 Principios de protección de datos

A continuación, se explican los principios de protección de datos que recoge la ley española que pueden ser aplicables en este proyecto. Se comentan también como se puede desarrollar el sistema para cumplir con todos ellos.

3.1.1 Principio de exactitud de datos

El primer principio de protección de datos que contempla esta ley es el principio de exactitud. Conforme a esto, los datos personales han de ser exactos y, en caso de ser necesario, actualizables. Para lograr esto, todo sistema que recopile datos personales ha de dar al usuario la posibilidad de suprimir o rectificar los datos que ha introducido, con el fin de corregirlos. Si se adoptan las medidas necesarias para lograr esto, el responsable del sistema no será responsable en caso de que los datos personales sean inexactos.

En el sistema a desarrollar, dado que los únicos datos personales que se pretenden almacenar son las entradas y salidas de los usuarios en los locales correspondientes, para cumplir con este principio se ha de dar la posibilidad al usuario de que, una vez registrada su entrada, pueda rectificar. Implementar este mecanismo es tan intuitivo como el proceso real de entrar y salir de un local. El usuario, una vez haya escaneado el código QR de entrada al local, únicamente tendrá que escanear el código QR de salida correspondiente para indicar al sistema que ya no se encuentra en dicho establecimiento.

3.1.2 Deber de confidencialidad

Los responsables y encargados del tratamiento de los datos personales están sujetos al deber de confidencialidad al que se refiere el artículo 5.1.f) del Reglamento (UE) 2016/679. Así mismo, esta obligación es complementaria a los deberes de secreto profesional. Dicho artículo obliga a que los datos personales sean tratados de tal manera que se garantice una seguridad

adecuada de los mismos, incluida la protección contra el tratamiento no autorizado o ilícito y contra su pérdida, destrucción o daño accidental.

En este proyecto, dado que el tratamiento de datos se realiza, en gran medida, de manera descentralizada, se tomarán medidas en la aplicación móvil cliente para que las bases de datos locales sean seguras. Así mismo, se han tomar las debidas medidas de seguridad para que las comunicaciones entre el servidor central y dichas aplicaciones sean seguras. También se garantizará que las comunicaciones entre la aplicación web de registro de locales y el servidor sean seguras, así como que la información almacenada en dicho servidor esté debidamente protegida.

3.1.3 Tratamiento basado en el consentimiento del afectado

En este principio se entiende como consentimiento del afectado toda manifestación voluntaria, específica, informada e inequívoca por la que este afecta, ya sea mediante una declaración o una acción afirmativa clara, el tratamiento de los datos personales que le conciernen.

Para cumplir con este principio, en el momento que un usuario acceda por primera vez al sistema, este le informará acerca de qué datos recopila y cómo los va a usar. Así mismo, el usuario ha de confirmar al sistema que es mayor de catorce años, puesto que, en caso negativo, el tratamiento de sus datos no podría fundamentarse en su consentimiento. Si el usuario acepta dicho tratamiento de datos, se lo indicará explícitamente al programa, y podrá usar la aplicación. Por el contrario, si el usuario no consiente dicho tratamiento, no podrá usarla y el sistema no almacenará ningún dato personal aun teniendo la aplicación instalada en su dispositivo.

3.1.4 Tratamiento de datos por obligación legal, interés público o ejercicio de poderes públicos

Tal y como dice la ley en estudio, El tratamiento de datos personales solo podrá considerarse fundado en el cumplimiento de una misión realizada en interés público o en el ejercicio de poderes públicos conferidos al responsable, en los términos previstos en el artículo 6.1 e) del Reglamento (UE) 2016/679, cuando derive de una competencia atribuida por una norma con rango de ley.

En dicho artículo del reglamento europeo se sostiene que el tratamiento es lícito en caso de que sea necesario para cumplir una misión realizada en interés público o en el ejercicio de poderes públicos conferidos al responsable del tratamiento. Para este proyecto, pese a que se aplicaría la primera condición, puesto que el sistema se desarrolla con un interés público, dichas competencias en materia de tratamiento de datos no derivan de una norma con rango de ley, por lo que este principio no es aplicable a este proyecto.

3.2 Derechos de las personas

En este apartado se recogen los derechos en materia de protección de datos que tenemos todos los ciudadanos españoles y se explica cómo el sistema planteado en este proyecto garantizará el cumplimiento de aquellos que puedan verse implicados durante el uso del sistema propuesto. Es de vital importancia remarcar que los derechos y libertades consagrados en la Constitución y en los Tratados y Convenios Internacionales en que España sea parte son plenamente aplicables en Internet, y por ello el sistema que se pretende desarrollar en este proyecto no ha de ser una excepción en ningún caso.

3.2.1 Transparencia e información al afectado

El usuario afectado por el tratamiento de sus datos personales debe tener acceso a la información básica del responsable de dicho tratamiento. Se considera información básica a la identidad del responsable del tratamiento y de su representante (de existir), la finalidad del tratamiento y la posibilidad de ejercer los derechos establecidos en los artículos 15 a 22 del Reglamento (UE) 2016/679. Para garantizar que se cumple este derecho, el sistema informará a sus usuarios de todo esto en el momento de acceder por primera vez, y podrán consultar esta información en cualquier momento desde la aplicación web.

3.2.2 Derecho de acceso

De acuerdo con lo establecido en el artículo 15 del Reglamento (UE) 2016/679, el usuario debe tener acceso a, además de lo mencionado en el apartado previo, las categorías de datos personales que se traten, los destinatarios a los que se les comunicarán y el plazo previsto de conservación de los datos. Todos estos aspectos serán redactados y comunicados al usuario cuando acceda por primera vez al sistema, así como podrá consultarlos en cualquier momento desde la aplicación web.

3.2.3 Derecho de rectificación

Al ejercer el derecho de rectificación reconocido en el artículo 16 del Reglamento (UE) 2016/679, el afectado deberá indicar en su solicitud a qué datos se refiere y la corrección que haya de realizarse. Tal y como se mencionó previamente en el apartado 3.1.1, la corrección de los datos de acceso a los locales se podrá realizar en cualquier momento, de la misma manera que una persona puede entrar y salir por una puerta.

3.2.4 Derecho de supresión

El usuario tiene el comúnmente denominado, “derecho al olvido”. En cualquier momento, el usuario tiene derecho a obtener por parte del responsable del tratamiento la supresión de todos sus datos personales en caso de que estos ya no sean necesarios para el fin por el que fueron recogidos. Este derecho estará ampliamente garantizado por el sistema propuesto en este proyecto, puesto que, al almacenar los datos localmente en los dispositivos, bastará con

que el usuario desinstale la aplicación y elimine sus datos para que estos desaparezcan por completo del sistema.

3.2.5 Derecho de oposición

El usuario tendrá derecho a oponerse en cualquier momento, por motivos relacionados con su situación particular, a que datos personales que le conciernan sean tratados por el sistema. El responsable del tratamiento dejará de tratar los datos personales, salvo que acredite motivos legítimos imperiosos para el tratamiento que prevalezcan sobre los intereses, los derechos y las libertades del interesado, o para la formulación, el ejercicio o la defensa de reclamaciones. En lo que respecta a este proyecto, en el momento que un usuario se oponga al tratamiento de sus datos por el motivo que fuere, tan solo ha de no acudir a establecimientos que requieran que registre su entrada en ellos. Así mismo, a la hora de notificar un resultado positivo de una prueba PCR, si el usuario se opone a realizar dicha acción, no hay ningún factor externo que le obligue a introducir dichos datos en su aplicación móvil. Por lo tanto, el derecho a la oposición está totalmente garantizado.

Capítulo 4. Aspectos Teóricos

En esta sección se describen los conceptos, herramientas y tecnologías usadas en el proyecto y sus principales características.

4.1 COVID-19

El COVID-19 (acrónimo del inglés *coronavirus disease 2019*) o coronavirus, es una enfermedad infecciosa y altamente contagiosa provocada por el virus SARS-CoV-2. Entre sus síntomas más comunes se incluyen la fiebre, tos, fatiga, sensación de falta de aire, náuseas y dolor de cabeza. En los casos más graves provoca neumonía, dificultad para respirar, pulso irregular y convulsiones, llegando a producir la muerte a un 2,52% de los infectados, según los datos recogidos por el CSSE (Center for Systems Science and Engineering) de la Universidad Johns Hopkins.

La principal vía de propagación de la enfermedad es el contacto directo entre personas. El virus está presente en las gotículas (pequeñas partículas líquidas provenientes del sudor, saliva y mucosidad) que desprenden los infectados al hablar, toser o estornudar. Estas partículas pueden permanecer sobre los objetos y superficies con los que ha estado en contacto el contagiado, pudiendo infectar de esta manera a todo aquel que entre en contacto con dichos objetos. Manteniendo una distancia de seguridad entre personas (la OMS recomienda que dicha distancia sea de al menos 1 metro) se previene que las personas sanas inhalen directamente las gotículas que salen de las infectadas. Gracias a un lavado frecuente de manos con agua y jabón o con desinfectantes de base alcohólica se consiguen reducir también las probabilidades de contagio por el contacto con superficies infectadas.

A medida que se fueron realizando estudios científicos, se descubrió que el SARS-CoV-2 también puede estar presente en aerosoles. Los aerosoles son partículas, tanto sólidas como líquidas, de tamaño muy pequeño (entre 0,002µm y 100 µm) que pueden permanecer suspendidas en un gas durante un tiempo variable. Esto implica que el virus puede permanecer en el aire y puede desplazarse en corrientes por largas distancias, pudiendo infectar a un mayor número de personas sin necesidad de que exista un contacto directo entre el infectado y el resto de las personas presentes en el mismo lugar. En zonas con ventilación nula o escasa, el riesgo de contagio por aerosoles es mucho más alto que en espacios bien ventilados o al aire libre.

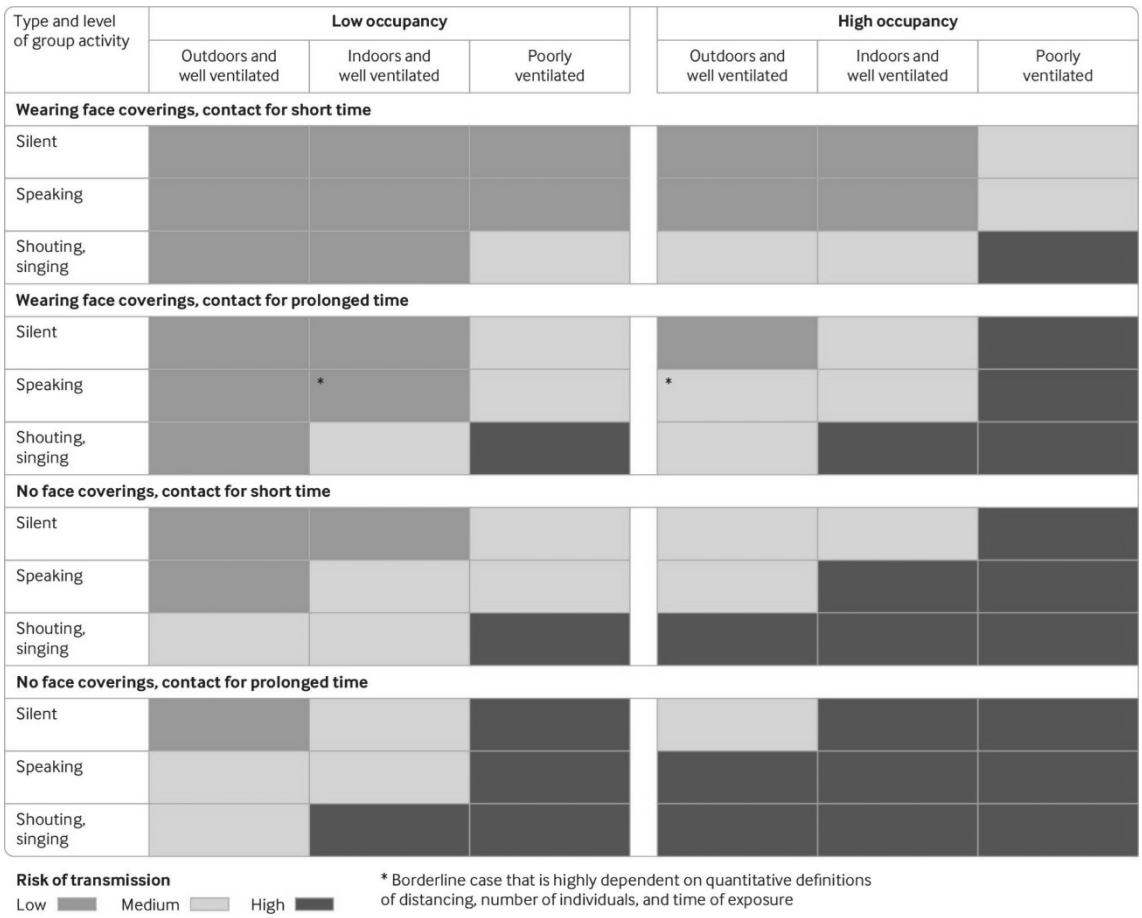

Figura 4.1 Niveles de riesgo de transmisión del SARS-CoV-2

En la Figura 1.1 se muestra una tabla planteada a modo de conclusión del estudio realizado por miembros de *The BMJ*, en el que se analiza el riesgo de transmisión del virus SARS-CoV-2 en diferentes ambientes. En la tabla se muestran desde un punto de vista cualitativo los niveles relativos de riesgo de contagio en los distintos ambientes cuando existe una persona asintomática infectada en ellos.

En base a los estudios realizados hasta la fecha, la transmisión del virus se iniciaría entre dos y tres días antes de que se manifiesten los síntomas por primera vez, y haría pico al inicio de la clínica. En los siguientes días, la transmisión descendería de manera progresiva, hasta que, pasada una semana, la probabilidad de contagio ya sería muy baja. Esta tendencia, demostrada en un estudio de *Nature Medicine*, coincide también con las observaciones realizadas a casos asintomáticos. Estos datos coinciden también con los estudios de evolución de la carga viral en pacientes con coronavirus. El periodo medio de incubación del virus de acuerdo con estos estudios es de 5,2 días. El momento de mayor carga viral coincide con las horas previas a la aparición de los primeros síntomas. El descenso de la carga viral con el paso del tiempo es progresivo, hasta alcanzar su límite de detección en una media de 21 días. Ese es, por tanto, el tiempo medio que tardaría un positivo por coronavirus en dar negativo en una prueba PCR.

Por las características de la enfermedad, la probabilidad de que una persona infectada por coronavirus no desarrolle síntomas es bastante alta. Esto dificulta las labores de prevención y rastreo, ya que una persona asintomática puede no saber que está contagiada, y por ende no

tomar las medidas de aislamiento oportunas. Independientemente de si el enfermo desarrolla o no algún síntoma, el virus está presente en su organismo y por tanto tiene la capacidad de contagiar a otras personas por cualquiera de las vías mencionadas anteriormente. Actualmente, en base a los estudios realizados hasta la fecha, alrededor de un 40% de los infectados por COVID-19 son asintomáticos.

La duración de la enfermedad aumenta proporcionalmente a la gravedad de esta. En los casos más leves la fiebre se suele calmar en menos de una semana, aunque otros síntomas como la tos o el dolor de cabeza pueden persistir durante más tiempo. En los casos más graves, transcurrido un periodo de tiempo cercano a los diez días después de la infección, comienzan a aparecer los síntomas más preocupantes. Una persona en estado grave puede tardar en curarse por completo entre dos y ocho semanas, aunque existe la posibilidad de que permanezca con secuelas de la enfermedad durante muchos años o incluso toda su vida.

4.2 Tecnologías, lenguajes de programación y frameworks de desarrollo

4.2.1 Tecnologías *contactless*

Las tecnologías *contactless* (tecnologías sin contacto), son aquellas que permiten que dos sistemas que se encuentran en un rango corto de distancia se comuniquen entre sí. En la actualidad, las tecnologías de este tipo más comunes son los escáneres de códigos QR y de barras, la tecnología NFC (Near-Field Communication) y el RFID (Radio Frequency Identification). Con la aparición de la pandemia del COVID-19, el miedo al contagio ha impulsado el crecimiento de estas tecnologías que evitan el contacto directo entre las personas y objetos como cajeros automáticos o las cartas de los restaurantes.

De cara a la realización de este proyecto, se eligió el escaneo de códigos QR como la tecnología principal sobre la que se basaría el método de registro de la actividad del usuario. El motivo principal de su elección por encima de las otras alternativas fue su simpleza y su coste económico.

Respecto a este segundo punto, para escanear etiquetas RFID hacen falta dispositivos especializados, y tanto su coste como el de las propias etiquetas es significativamente superior al de imprimir un código QR que pueda ser escaneado con la cámara presente en casi cualquier dispositivo móvil.

En cuanto al NFC, si bien diseñar las etiquetas NFC es más complejo que codificar información en un código QR, el hecho de que sea una tecnología más moderna hace que no todo el mundo cuente con un dispositivo capaz de comunicarse mediante este sistema. Puesto que con este proyecto lo que se busca es que el mayor porcentaje de población posible pueda usar el sistema, implementarlo en base a esta tecnología sería ir en contra de ese objetivo.

4.2.1.1 Códigos QR

Los códigos QR (del inglés *Quick Response*) son códigos de barras bidimensionales. Surgen como evolución de los códigos de barras tradicionales en el año 1994, creados por Denso Wave, una compañía japonesa subsidiaria de Toyota. A diferencia de otros formatos de códigos bidimensionales, sus derechos de patente no se ejercen, por lo que son de carácter Open Source (Código Libre).

Pese a que su uso se popularizó mucho alrededor del año 2013 gracias a la creciente venta de smartphones, nunca llegaron a tener una importancia destacable en cuanto a las funciones que desempeñaban, lo que hizo que dicha popularidad se fuese diluyendo con el paso del tiempo. Hasta la llegada del coronavirus, su uso más allá de los almacenes era residual.

Un código QR es una matriz bidimensional con módulos de dos colores que presentan mucho contraste entre sí (generalmente blanco y negro). Según la cantidad de estos módulos que haya, los códigos QR se clasifican en versiones que van desde la 1 (matriz de 21 x 21 módulos) hasta la versión 10 (177 x 177 módulos). Cuanta más cantidad de módulos haya en el código, o lo que es lo mismo, a mayor versión, más cantidad de información es capaz de almacenar. Además de estos módulos, los códigos QR cuentan con tres recuadros en las dos esquinas superiores y en la inferior izquierda, cuya función es detectar la posición del código con respecto al escáner, y con otro recuadro más pequeño situado cerca de la esquina inferior derecha que detecta el alineamiento del lector con respecto al código.

Actualmente, las cámaras digitales integradas en los smartphones son capaces de escanear códigos QR, lo que los convierte en una herramienta muy accesible para la mayor parte de la población, y con la que la interacción es tan simple como la de enfocarlo con dicha cámara y dejar que el dispositivo haga el resto.

4.2.2 Desarrollo móvil multiplataforma

El principal objetivo de este proyecto es crear un sistema de rastreo que sea accesible y usable para el mayor número de personas posible. Por este motivo, a la hora de desarrollar la aplicación móvil, es necesario que esté disponible tanto para los principales sistemas operativos del mercado. Según datos de *StatCounter Global Stats*, a enero de 2021, Android cuenta con una cuota de mercado global del 71,93% y iOS con un 27,47%. Entre los dos suman el 99,4% de los dispositivos móviles del mundo. En España, el porcentaje de dispositivos iOS es algo menos, un 19,73%, mientras que el de dispositivos Android asciende al 79,85%. En ambos casos, tanto a nivel global como nacional, la tendencia del mercado es a mantenerse constante, tal y como viene sucediendo los últimos años. Habiendo estudiado estos datos, se descartó el desarrollo nativo, dado que esto significaría crear al menos dos aplicaciones distintas si se intentase cubrir casi el 100% del espectro.

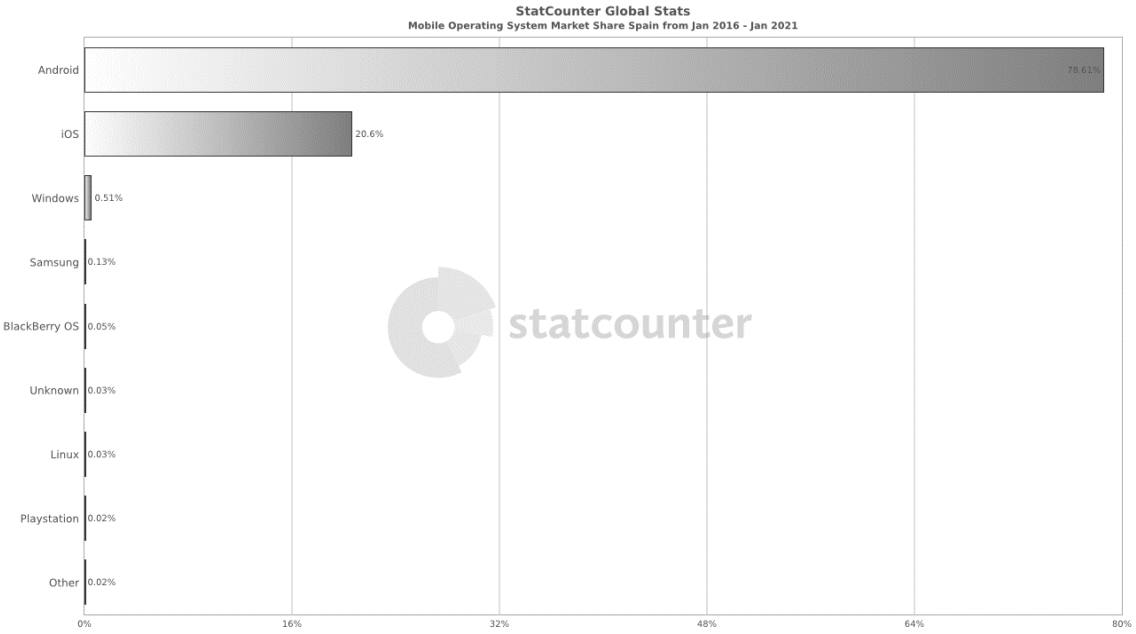

Figura 4.2. Cuota de mercado de los sistemas operativos móviles

4.2.2.1 Xamarin.Forms

Xamarin.Forms es un marco de interfaz de usuario de código abierto del *framework* Xamarin, desarrollado por Microsoft, que extiende la plataforma .NET con herramientas y librerías destinadas a la creación de aplicaciones para dispositivos móviles. Xamarin.Forms permite desarrollar aplicaciones móviles para Android, iOS y Windows mediante una única base de código en C# común para las tres plataformas.

En sus inicios, Xamarin permitía desarrollar aplicaciones móviles multiplataforma manteniendo una base de código común, pero teniendo que implementar las interfaces de usuario específicas para cada plataforma. Xamarin.Forms, mediante interfaces de usuario escritas en XAML con código subyacente en C#, que se traducen en tiempo de ejecución a controles nativos para cada plataforma, permite a los desarrolladores crear aplicaciones multiplataforma que conserven tanto la apariencia como el funcionamiento nativo, sin necesidad de desarrollar por separado dichas interfaces.

La API de Xamarin.Forms puede implementarse en XAML y en C#, y proporciona mecanismos de enlace a datos que pueden ser usados para implementar patrones de arquitectura móvil como el MVVM (Model-View-ViewModel).

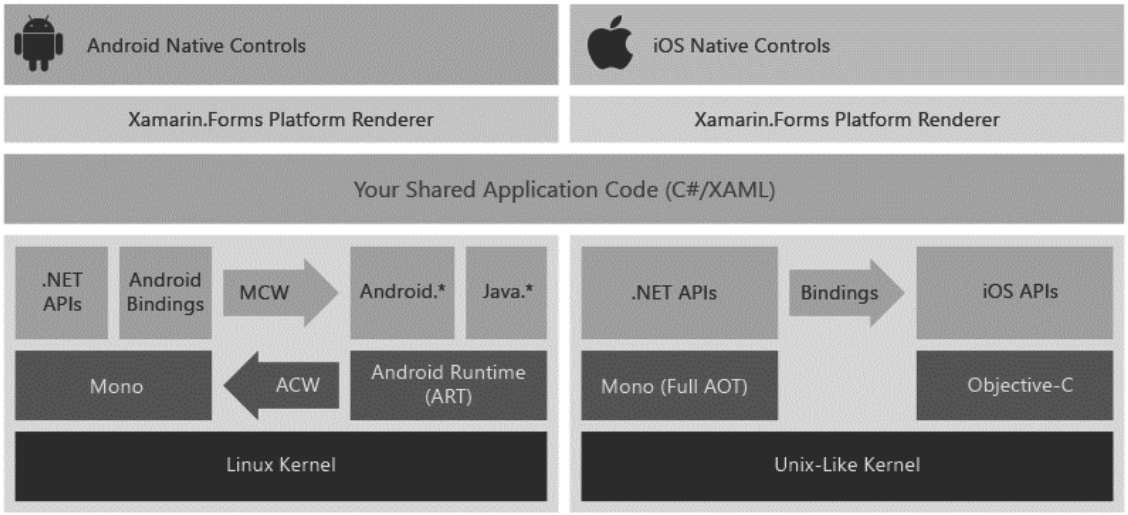

Figura 4.3. Arquitectura de Xamarin.Forms

4.3 Metodologías ágiles

El principal objetivo de un enfoque ágil aplicado al desarrollo de software es distribuir de forma permanente sistemas de software en funcionamiento diseñados con iteraciones rápidas. Con este enfoque se trata de proporcionar en poco tiempo piezas pequeñas de software en funcionamiento con el fin de mejorar la satisfacción del cliente y de obtener con la mayor brevedad posible un producto funcional.

La metodología ágil como se conoce actualmente surge en el año 2001, ideada por una comunidad de desarrolladores que, tras reunirse y formar la "alianza ágil", deciden redactar el Manifiesto para el Desarrollo Ágil de Software. En dicho manifiesto se establecen cuatro características que los desarrolladores han de priorizar por encima de otras cuestiones. Estas características a priorizar son las siguientes:

- Las personas y las interacciones antes que los procesos y las herramientas.
- El software en funcionamiento antes que la documentación exhaustiva.
- La colaboración con el cliente antes que la negociación contractual.
- La respuesta ante el cambio antes que el apego a un plan establecido.

Para este proyecto se ha decidido aplicar un enfoque ágil debido a las características del producto de software a desarrollar. En primer lugar, el sistema de rastreo en el que consiste este proyecto se plantea como una respuesta de inmediata aplicación a uno de los grandes problemas de la crisis del coronavirus, la dificultad de las labores de rastreo de nuevos contagios. Una rápida capacidad de respuesta ante las dificultades es una de las claves del éxito, y es por este motivo que con este proyecto se busca contar con un sistema de software funcional desde lo antes posible. Además, tal y como se ha ido descubriendo a lo largo de los meses que está durando la pandemia, el SARS-CoV-2 tiene una gran capacidad de mutación, lo que posibilita que puedan existir cambios en su forma de transmisión y replicación. Por este motivo, es necesario que en este proyecto exista una gran capacidad de respuesta ante los cambios en el entorno que se puedan producir. Por último, considero que, en un proyecto destinado a proteger la salud pública y el bienestar de un país, la colaboración total entre

todas las personas encargadas de dar respuesta a esta crisis es fundamental. Por ello es necesario que exista una colaboración total entre la comunidad científica, el cliente y el equipo de desarrollo encargado de llevar a cabo este sistema.

4.4 Integración y distribución continuas (CI/CD)

La CI/CD (siglas en inglés de *Continuous Integration and Continuous Delivery*) es un método para distribuir productos de software con frecuencia mediante el uso de la automatización en las etapas de desarrollo. En esta práctica se automatiza y controla permanentemente todo el ciclo de vida del software, desde las etapas de integración y pruebas hasta la distribución e implementación.

La integración continua consiste en automatizar los procesos de compilación y ejecución de pruebas con el fin de que los cambios realizados al modificar e introducir código nuevo durante el desarrollo del software se prueben y combinen de manera periódica en un repositorio compartido. Esto soluciona el problema de que, al desarrollar varias partes del software en paralelo, puedan ocurrir conflictos que no se detecten hasta muy avanzado dicho desarrollo. Además, también se consigue que el estado de la rama principal del proyecto pase siempre todas las pruebas y esté en un estado válido.

La distribución o implementación continua es el segundo concepto en el que se basa este método de trabajo. De manera similar a la integración continua, el objetivo de la distribución continua es automatizar la liberación de los cambios implementados en el software desde el repositorio en el que se encuentran hasta el entorno de producción una vez han sido probados. La principal ventaja de la distribución continua es liberar de carga a los equipos de operaciones, evitando que realicen tareas manuales de despliegue que suelen retrasar la distribución del software.

Capítulo 5. Planificación del Proyecto y Resumen de Presupuestos

5.1 Planificación

5.1.1 Etapa inicial

En el apartado 4.3 se indica que el desarrollo de este proyecto se va a realizar siguiendo una metodología de desarrollo ágil de software. Sin embargo, previo al desarrollo, han de realizarse una serie de tareas que son necesarias para poder llevar a cabo dicho desarrollo. A este conjunto de tareas lo denominaremos "etapa inicial".

Dicha etapa inicial comenzará el día 4 de febrero de 2021 y se pretende que finalice el día 28 del mismo mes, obteniendo así un plazo de casi un mes para realizar todas las tareas de preparación del proyecto necesarias. La naturaleza de estas tareas es variada, desde actividades de estudio y análisis hasta tareas más técnicas como por ejemplo la configuración del entorno de pruebas del proyecto. A continuación, se muestran la estructura de desglose de trabajo (EDT) de esta etapa y un diagrama de Gantt que representa dicha EDT.

Número de esquema	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Estudio y análisis previo**	**12 días**	**jue 04/02/21**	**vie 19/02/21**	
1.1	Estudio de la situación actual	5 días	jue 04/02/21	lun 08/02/21	
1.2	Antecedentes	3 días	jue 11/02/21	sáb 13/02/21	2
1.3	Alternativas y competencia	2 días	dom 14/02/21	lun 15/02/21	3
1.4	Estudio de integración en el ecosistema actual	2 días	jue 18/02/21	vie 19/02/21	4
2	**Aspectos teóricos**	**17 días**	**sáb 06/02/21**	**dom 28/02/21**	
2.1	Estudio de tecnologías	5 días	sáb 06/02/21	vie 12/02/21	
2.2	Estudio de la normativa vigente	1 día	dom 28/02/21	dom 28/02/21	5;7
3	**Preparación de la infraestructura**	**12 días**	**sáb 13/02/21**	**dom 28/02/21**	
3.1	Creación de los repositorios GIT	1 día	sáb 13/02/21	sáb 13/02/21	7
3.2	Creación de la base de datos central	1 día	dom 14/02/21	dom 14/02/21	10
3.3	Configuración de los entornos de desarrollo	1 día	lun 15/02/21	lun 15/02/21	11
3.4	Preparación de los entornos de compilación y despliegue	4 días	jue 18/02/21	dom 21/02/21	12
3.5	Integración con el sistema de CI/CD	5 días	lun 22/02/21	dom 28/02/21	13
3.6	Configuración del entorno de pruebas	5 días	lun 22/02/21	dom 28/02/21	13

Figura 5.1. EDT de la etapa inicial

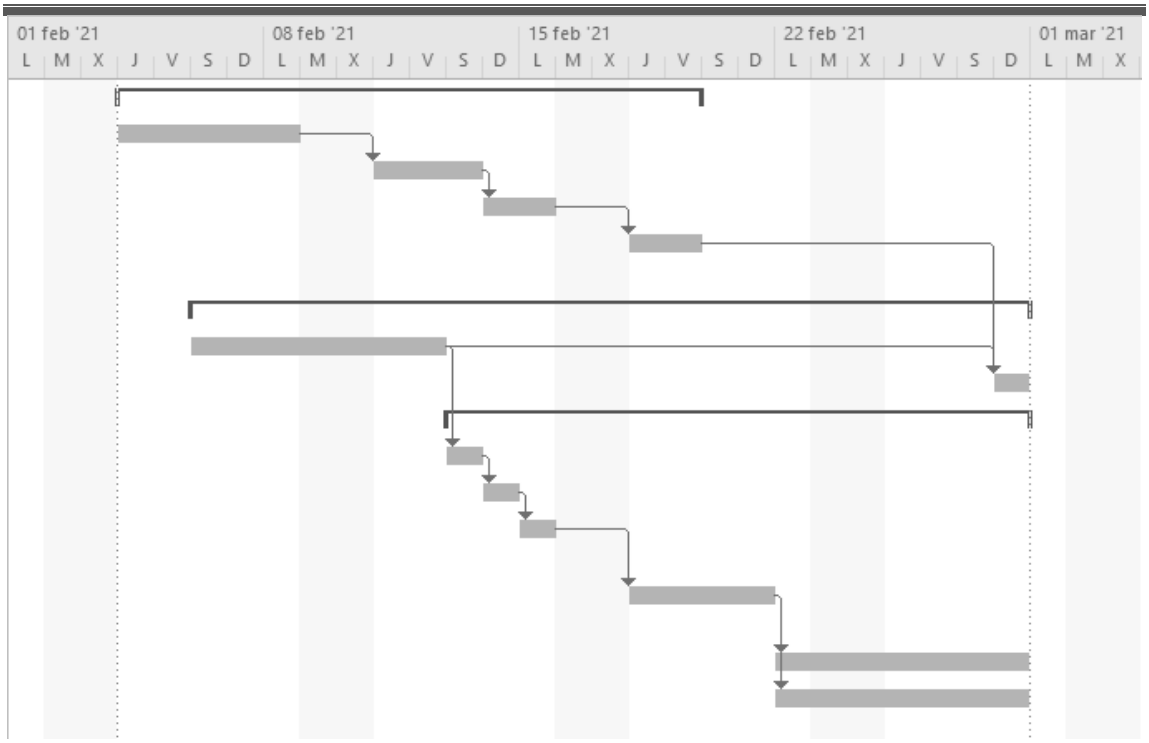

Figura 5.2. Diagrama de Gantt de la etapa inicial

Como se puede apreciar en el diagrama, los martes y miércoles se ha decido que sean los días no laborales durante todo lo que dure el proyecto. Esto se debe únicamente a cuestiones de disponibilidad del equipo de desarrollo.

5.1.2 Desarrollo del proyecto

Tal y como se viene explicando a lo largo de este documento, para el desarrollo de este proyecto se ha decido usar una metodología ágil. De entre todas las metodologías existentes enmarcadas dentro de un enfoque ágil, se ha optado por adaptar Scrumban a las necesidades y particularidades de este proyecto concreto. Scrumban nace de una fusión entre la metodología ágil Scrum y la metodología de desarrollo flexible Kanban, consistiendo en una metodología de trabajo iterativa en la que se hace uso de un tablero Kanban para gestionar las historias de usuario o tareas a realizar en cada una de las iteraciones.

Por su naturaleza iterativa, carece de sentido realizar una planificación de la fase de desarrollo del proyecto por medio de una EDT (Estructura de Desglose de Trabajo) o un diagrama de Gantt, como se suele llevar a cabo cuando se siguen metodologías de desarrollo más tradicionales. De esta manera, en lugar de estimar la duración y fechas de realización de cada conjunto y subconjunto de tareas, se han de planificar la duración de las iteraciones, las reuniones con el cliente y la forma de estructurar las tareas a realizar. A continuación, se describen las características mencionadas anteriormente enmarcadas dentro de la metodología de trabajo Scrumban:

- Fechas de comienzo y fin del desarrollo: El comienzo del desarrollo de este proyecto y, por tanto, el comienzo de la primera iteración tendrá lugar el 1 de marzo de 2021. Pese a que, por su característica de respuesta a una crisis sanitaria, no es posible

determinar una fecha de cierre del proyecto, se ha decidido marcar como objetivo que el día 13 de junio de 2021 se disponga un producto que cumpla con el objetivo descrito en el apartado 2.2 de este documento.

- Planificación de reuniones: Dado que el equipo que va a desarrollar este proyecto se compone de un único integrante, no será necesario realizar reuniones diarias. Al final de cada iteración, se realizará una reunión con el cliente con el fin de presentar los avances realizados durante dicho ciclo y hacer una retrospectiva de cómo fue la iteración. Después, se realizará una reunión con formato de entrevista en la que se obtendrán nuevos requisitos del sistema.
- Formato de las iteraciones: Para la realización de este proyecto se llevarán a cabo iteraciones de entre 3 y 4 semanas de duración. La duración concreta de cada iteración se decidirá en base de la carga de trabajo que se pretende realizar en ella. Cada semana de trabajo estará compuesta por 5 días laborales y 2 días de descanso.
- Tablero Kanban: Para seguir la metodología Scrumban es preciso contar con un tablero en el que se han de organizar las tareas a realizar en cada iteración con el fin de conocer su estado actual y su evolución. La configuración de las columnas de estos tableros puede variar según las características del proyecto a realizar. En este caso, las tareas se organizarán en las siguientes 4 columnas:
    - **New:** En esta columna se han de colocar aquellas tareas que aún no se han empezado, pero que están planeadas para ser realizadas a lo largo de la iteración actual.
    - **Active:** Esta segunda columna contendrá todas aquellas tareas que están siendo realizadas.
    - **Resolved:** Una vez que el desarrollo de una historia de usuario o tarea se da por finalizado, se traslada a esta columna. Las tareas aquí colocadas serán aquellas que se encuentren en fase de pruebas.
    - **Closed:** Cuando todas las pruebas necesarias para comprobar el correcto funcionamiento de las novedades introducidas al proyecto con la realización de una tarea se han completado de manera satisfactoria, dicha tarea o historia de usuario se da por cerrada y se mueve a esta columna. Al finalizar la iteración, el equipo de desarrollo podrá reabrir una tarea cerrada en caso de que el cliente no esté conforme con su resultado. También se pueden reabrir tareas a lo largo de las iteraciones si se encuentran nuevos fallos o bugs.

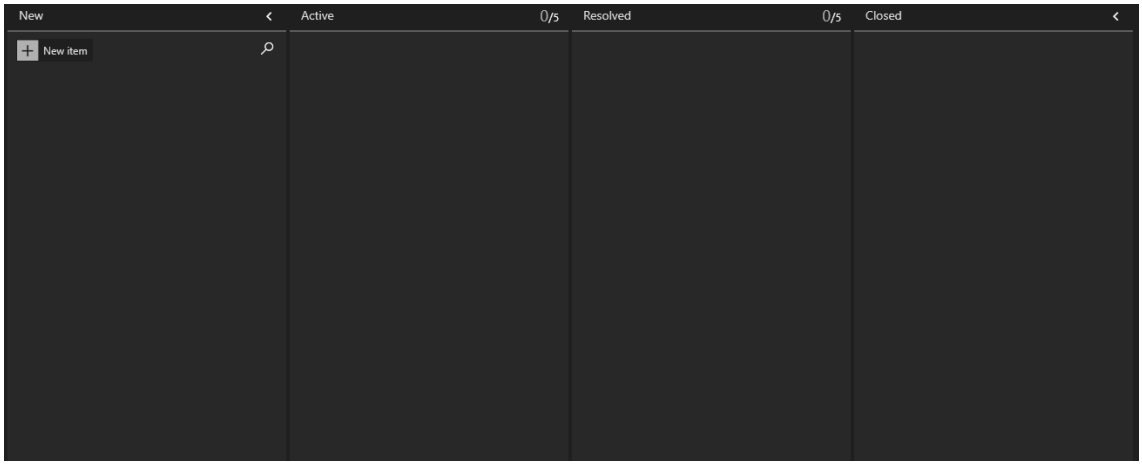

Figura 5.3. Tablero Kanban del proyecto

5.2 Resumen del Presupuesto

Debido al uso de una metodología ágil iterativa para el desarrollo de este proyecto, no es posible realizar un desglose de tareas a través del cual obtener un presupuesto de costes, puesto que dichas tareas irán apareciendo y evolucionando sobre la marcha. Es por este motivo, que en su lugar se ha trazado una hoja de ruta, con el fin de acotar los tiempos de desarrollo y establecer un plan de lanzamiento de versiones, a través del cual se puedan estimar los costes del proyecto hasta una serie de hitos. Tal y como se describe en la planificación, existirán dos etapas dentro del desarrollo del proyecto, la etapa inicial y la etapa de desarrollo. La etapa inicial durará un mes, mientras que la de desarrollo tendrá una duración de 3 meses y medio, tal y como se indica en la planificación. Tras estas dos, de cara a realizar un presupuesto, es necesario añadir una fase de implantación y mantenimiento en la que se realizarán las ampliaciones necesarias en caso de que se decida poner en marcha este sistema de rastreo. Para esta última etapa, se calculará el presupuesto para una duración de un año, con el fin de poder conocer el coste anual estimado de mantener activo este sistema de rastreo. A continuación, se muestra el resumen del presupuesto calculado más en detalle en el capítulo 13.

Partida	Coste (€)
Etapa inicial	4.089,72
Desarrollo del primer prototipo del proyecto	14.041,37
Ampliación y mantenimiento del sistema	49.200,64
TOTAL	**67.331,73 €**

Figura 5.4. Resumen del presupuesto

Capítulo 6. Seguimiento y evolución del proyecto

Uno de los pilares fundamentales de los enfoques ágiles es el seguimiento constante del estado del proyecto. Dado que, por las características de la metodología, no se puede realizar un estudio del alcance del proyecto, es crucial que tanto el equipo de desarrollo como el cliente o cualquier otro stakeholder pueda estar al corriente del rumbo del proyecto, su estado actual, objetivos a corto plazo, etcétera. A continuación, se procede a describir cómo se ha gestionado el seguimiento del proyecto.

6.1 Backlogs de producto y de iteración

El primero de los elementos usados para monitorizar el estado del proyecto es el backlog (o pila) de producto. Un backlog o pila de producto es una lista priorizada de historias de usuario (definidas en el capítulo 7 del análisis) en la que se encuentran todas aquellas nuevas funcionalidades o cambios que se han detectado y se pretenden llevar a cabo en algún momento del desarrollo. Analizándolo, tanto los stakeholders como el equipo de desarrollo pueden hacerse una idea del alcance global del proyecto planificado hasta la fecha.

El backlog de iteración (*sprint backlog* en inglés) es también una lista priorizada, salvo que, en esta ocasión, en lugar de tratarse de una lista de historias de usuario, lo que contiene son tareas. Estas tareas son actividades de bajo nivel, pensadas para ser manejadas por el equipo de desarrollo, no por el cliente u otros stakeholders. Cada historia de usuario puede tener un conjunto de tareas asociadas a ella, y serán necesario completarlas para cerrar dicha historia. Este backlog, por tanto, contendrá las tareas asociadas a las historias de usuario que han sido planificadas para la iteración actual, y dará al equipo de desarrollo una visión acerca del trabajo restante planificado para dicha iteración.

6.2 Tableros Kanban

Tal y como indica la metodología Scrumban, para gestionar el desarrollo del proyecto se hará uso de un tablero Kanban con las características descritas en el apartado 5.1.2. En este tablero se colocan las historias de usuario existentes en el backlog, y organizándose por columnas en función de su estado. El objetivo de este tablero es que todos los miembros del equipo de desarrollo sepan cuál es la dedicación actual de cada desarrollador, así como conocer el estado actual de las historias de usuario.

Además del tablero Kanban del proyecto, existirá un tablero Kanban específico para cada iteración, en el que se organizarán las tareas correspondientes a las historias de usuario planificadas para dicha iteración. Al igual que el backlog de iteración, este tablero está enfocado únicamente al uso por parte del equipo de desarrollo.

6.3 Diagramas de quemado y evolución del proyecto

Los diagramas de quemado son representaciones gráficas del trabajo que queda por hacer en el proyecto, medido, en este caso, en horas de trabajo. Se trata de diagramas bidimensionales, en los que el eje vertical representa el trabajo por hacer, y el eje horizontal el tiempo. Cada vez que una historia de usuario se cierra, la carga de trabajo pendiente disminuye, originando así diagramas en forma de escalera. Al igual que en la figura 5.2, en los diagramas de quemado se ven representados los días no laborales (martes y miércoles), con el fin de explicar por qué el proyecto no avanza esos días.

En este proyecto se ha usado un diagrama de quemado por cada iteración, pretendiendo de esta manera poder visualizar la frecuencia con la que se cerraban las historias de usuario en las diferentes iteraciones. En las retrospectivas realizadas al final de cada iteración, gracias a poder analizar estos diagramas de quemado, se sacan conclusiones sobre el flujo de trabajo que pueden resultar útiles en el futuro.

A continuación, se muestran los diagramas de quemado de las 4 iteraciones realizadas hasta la fecha.

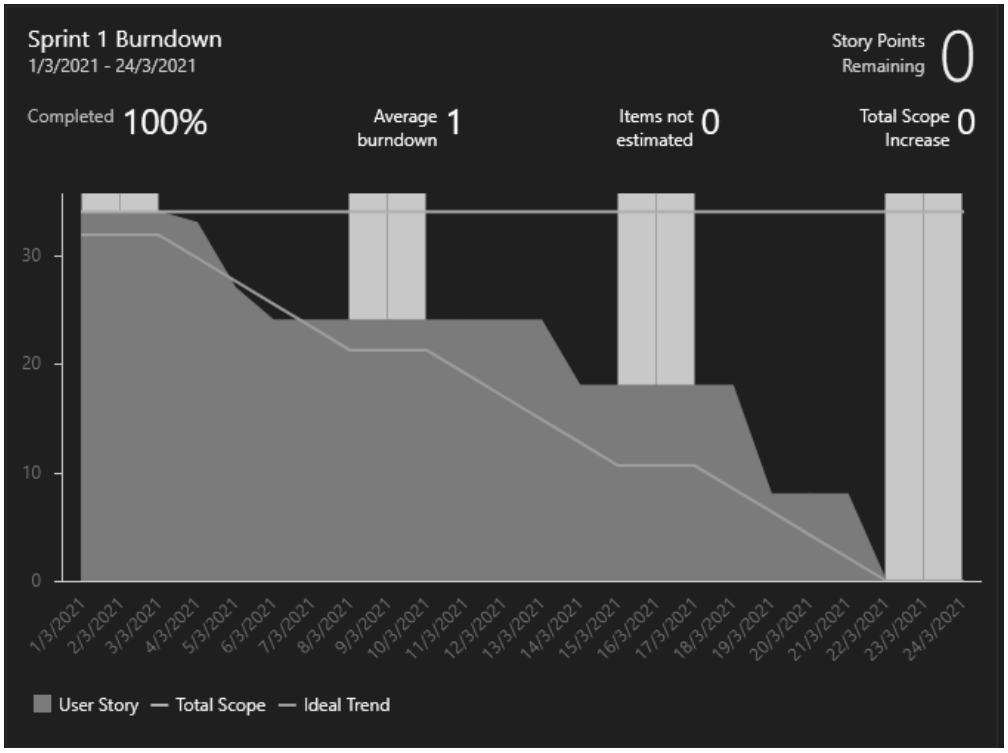

Figura 6.1. Diagrama de quemado de la primera iteración

Tal y como se aprecia en este primer diagrama, en la primera iteración del desarrollo se cerraron todas las historias de usuario planificadas para ella. Una observación que importante que se realizó en la retrospectiva fue que, se comenzó el proyecto con mucha fuerza los primeros días, pero después se frenó el ritmo en el segundo cuarto de la iteración. En las siguientes iteraciones se trató de llevar un ritmo de trabajo más constante.

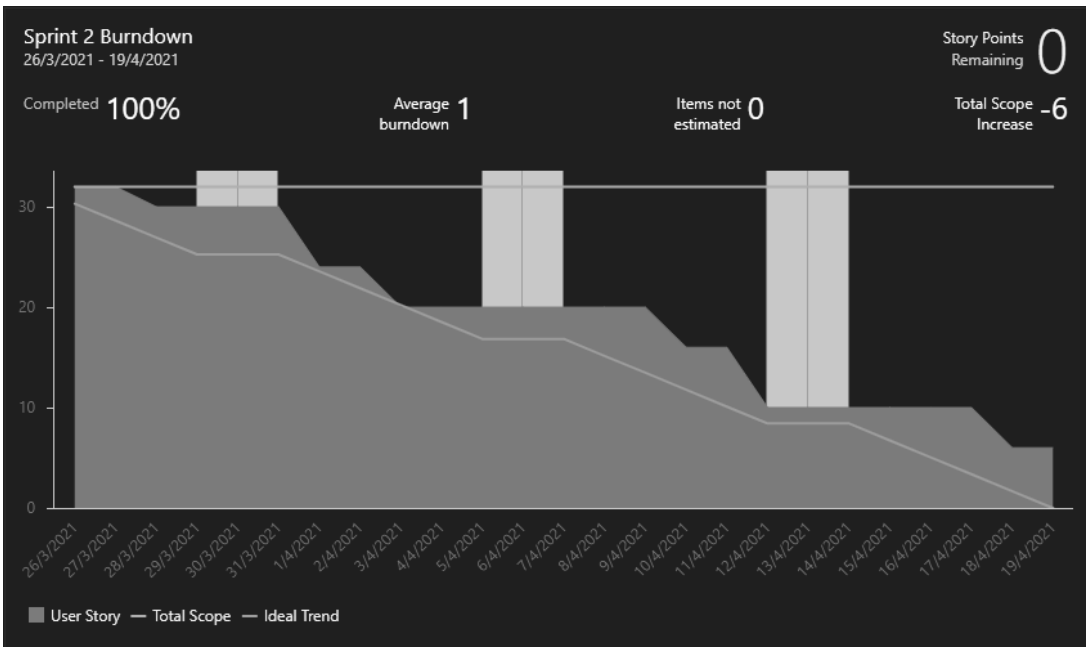

Figura 6.2. Diagrama de quemado de la segunda iteración

En esta segunda iteración, que tuvo lugar entre los días 26 de marzo y 19 de abril, fue en la que se implementó el grueso del proyecto. La principal funcionalidad nueva desarrollada fue la sincronización de alertas de posible contacto con casos positivos. La implementación de esta sincronización, tanto manual como automática, no fue sencilla debido a la gran diferencia de comportamiento entre los sistemas operativos Android y iOS, lo cual produjo que la duración estimada para algunas historias de usuario no se correspondiera con la real, llegando a ser bastante más de lo planificado. Por este motivo, además, se puede apreciar en el diagrama de quemado de la segunda iteración que no se consiguieron cerrar todas las historias planificadas para dicha iteración.

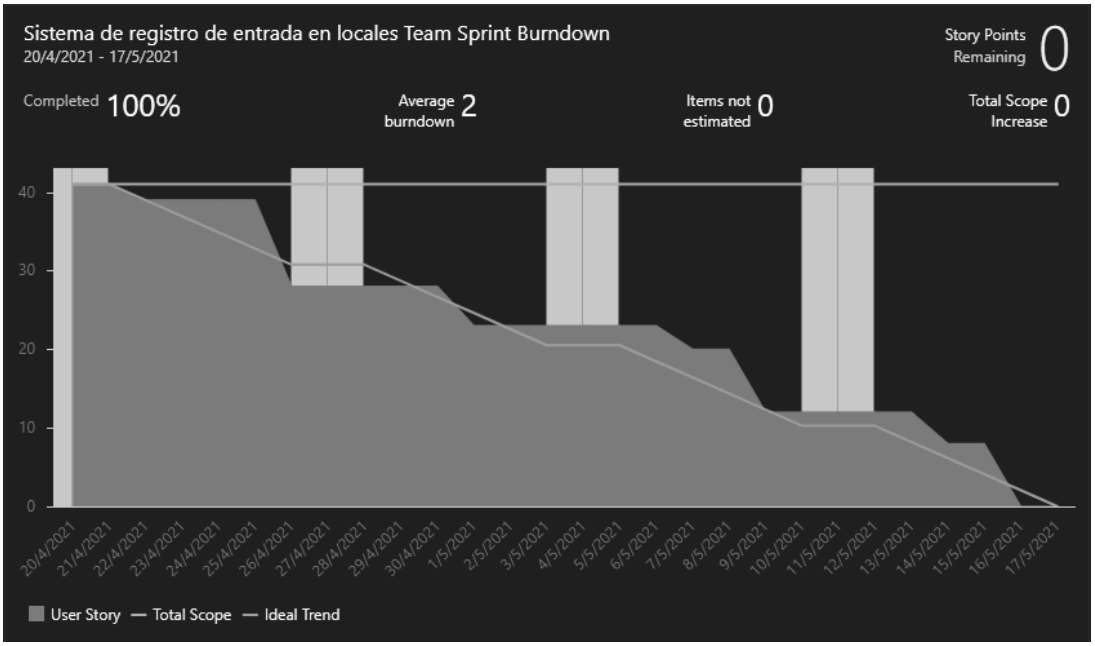

Figura 6.3. Diagrama de quemado de la tercera iteración

El primer objetivo de la tercera iteración del proyecto fue cerrar la historia de usuario empezada en la anterior iteración. Nuevamente, se puede apreciar como esto requirió de una semana de trabajo, pero se pudo completar. De ahí en adelante, se puede observar como el proyecto avanzó de manera constante, sin sufrir retrasos o parones muy grandes.

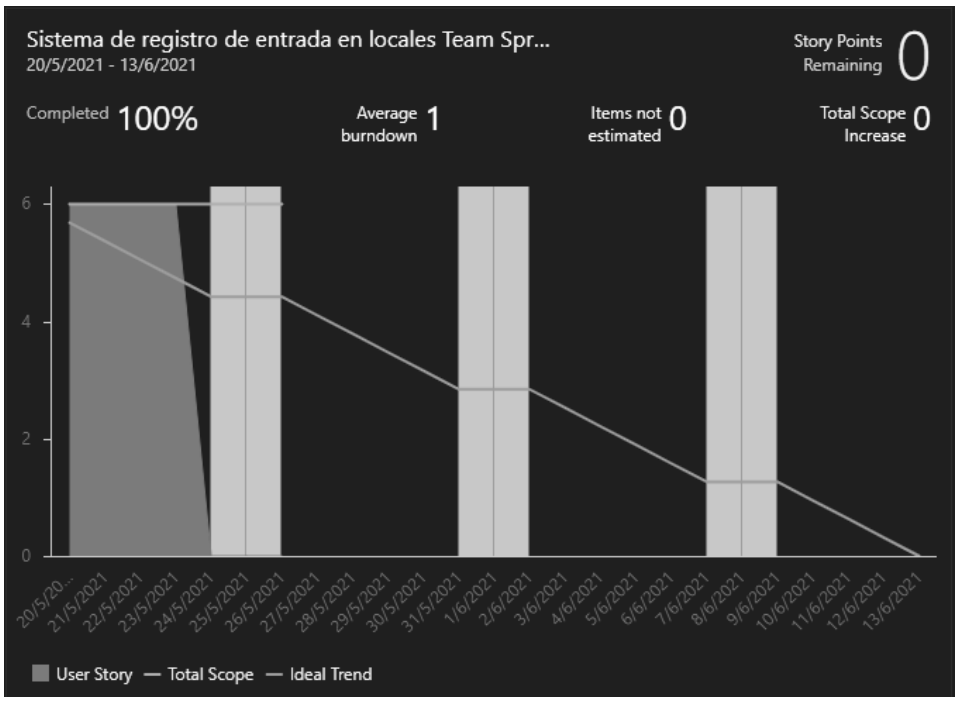

Figura 6.2. Diagrama de quemado de la cuarta iteración

Por último, en la cuarta iteración, únicamente se quiso añadir una pequeña funcionalidad extra al sistema para dar por finalizado el producto a entregar el día 13 de junio, alcanzando de esta manera el hito propuesto en la planificación del proyecto.

Capítulo 7. Análisis

7.1 Determinación del Alcance del Sistema

Tal y como se describe en el capítulo 2.2, el objetivo de este proyecto es construir un sistema de rastreo de contagios de coronavirus producidos por el contacto estrecho en el interior de locales cerrados, demostrando que es posible crear e implementar sistemas de rastreo eficaces si se siguen los pasos adecuados.

La intención de este proyecto es lograr un sistema plenamente funcional, cuya única limitación sea la integración con los sistemas sanitarios actuales. De esta manera, se logrará un producto que pueda ser presentado a modo de prototipo ante las autoridades competentes, con el fin de que la implantación de un sistema con estas características se vea como una posibilidad real tanto a corto como a largo plazo.

Para lograr lo propuesto anteriormente, el sistema debe contar con una aplicación web que ha de permitir a los dueños de los locales registrar sus establecimientos, asignándoles un código QR único que puedan colocar en las entradas y salidas de sus locales. Los usuarios podrán buscar si existe un local concreto en el sistema, con el fin de tenerlo en cuenta a la hora de decidir si ir o no. Además, también se ha de entregar a la población una aplicación móvil que les permita escanear dichos códigos y notificar los resultados de una prueba en caso de que el resultado sea positivo. Por último, es imprescindible que la aplicación sirva como mecanismo para alertar al usuario en caso de que este pueda haber sido infectado, con el fin de que contacte lo antes posible con las autoridades médicas y se tomen las medidas de prevención necesarias para evitar seguir con la propagación del virus.

Puesto que la naturaleza de este proyecto es de carácter público, es crucial que la política de tratamiento de datos sea estricta. Uno de los pilares más importantes de este proyecto es el tratamiento mínimo de información personal, limitándose a recopilar datos de acceso y salida de locales, junto con resultados positivos de pruebas contra el coronavirus. Los registros de visitas a locales, en cualquier caso, han de ser almacenados de manera local con el objetivo de que el usuario sea el único dueño de los mismos, hasta el momento en el que reporte un resultado positivo. Será entonces cuando dichos datos, de manera totalmente anónima, serán enviados al servidor con el fin de poder realizar las labores de rastreo pertinentes. Al mismo tiempo, todos los potenciales usuarios del sistema han de conocer y poder consultar en cualquier momento la información referida al tratamiento de datos, detallando en ella cómo se usan dichos datos, sean o no objeto de la ley de protección de datos. De esta manera se ha de lograr una total confianza en el tratamiento de datos del sistema por parte de la población, lo que lo diferenciará de sus homólogos en el mercado.

Por último, es de vital importancia que las interfaces de usuario de ambas aplicaciones (web y móvil) sean intuitivas y fácilmente usables. Esto es importante de cara a la recepción que el sistema pueda tener por parte de la población, dado que se necesita que todo tipo de personas puedan aprender a usarlas fácilmente sin que se conviertan en un incordio para su día a día.

7.2 Requisitos del Sistema

7.2.1 Obtención de los Requisitos del Sistema

Al contrario que en las metodologías de desarrollo de software tradicionales, en los métodos ágiles el proceso de obtención de requisitos se realiza de manera continua a lo largo de todo el proyecto. De esta manera, los requisitos se identifican y detallan generalmente poco antes de ser desarrollados, propiciando así que estos puedan variar y evolucionar durante el desarrollo del sistema.

Por la característica iterativa de la metodología Scrumban, en este proyecto el análisis de requisitos se realiza en las reuniones periódicas programadas al comienzo de cada iteración. En ellas se obtienen requisitos generalmente de muy alto nivel, que posteriormente son refinados y representados como historias de usuario. Una vez creadas estas historias de usuario, se estima la cantidad de tiempo necesaria para completar su desarrollo. A la hora de estimar el tiempo se tiene en cuenta el diseño, el desarrollo y la ejecución de pruebas. Al final de cada iteración, en la reunión de demostración y retrospectiva, se realiza una validación de dichas historias de usuario. Dado que no existe un proceso formal de gestión de requisitos, lo cambios que puedan surgir en ellos a lo largo del desarrollo se incluyen en los criterios de aceptación de la historia de usuario correspondiente con el fin de tratarlos en la siguiente iteración o más avanzada la misma en caso de no estar cerrada dicha historia.

7.2.2 Historias de usuario

Una historia de usuario es una explicación general e informal de una función de software escrita desde la perspectiva del usuario final. Su propósito es articular cómo proporcionará una función de software valor al cliente. Mediante el uso de un lenguaje poco técnico, aportan información al equipo de desarrollo acerca de qué se debe hacer, cómo se debe hacer, por qué se debe hacer y quién se ve beneficiado por su desarrollo.

Cada historia de usuario está formada por un título, una descripción y una serie de criterios de aceptación. El título debe indicar qué actor se ve beneficiado por el desarrollo de la historia, en qué consiste lo que se pide desarrollar y el motivo por el que se ha de llevar a cabo. Estas tres cualidades se describirán usando el siguiente formato:

```
Cómo (actor) quiero (funcionalidad) para (motivo del desarrollo)
```

Los criterios de aceptación que acompañan al título de la historia de usuario describen siempre un contexto, un evento y la respuesta o consecuencia esperada del sistema. Aunque los criterios de aceptación pueden ser muy extensos, se mantiene el uso de un lenguaje de negocio poco o nada técnico.

En este apartado se describen las historias de usuario identificadas a lo largo del desarrollo del sistema. Cada historia de usuario está representada por una tarjeta que contiene los tres elementos indicados anteriormente, su título, una descripción y los criterios de aceptación

correspondientes, junto con un identificador único (ID) y el tiempo que se estimó que iba a ser necesario para su realización, medido en puntos de historia (SP, del inglés *Story Points*). Estos puntos de historia no se corresponden necesariamente con horas, sino que estiman el esfuerzo necesario para completar una historia de usuario.

ID: 1	**Como usuario estándar quiero que la aplicación móvil tenga un lector QR para poder leer los códigos QR de los locales a los que acudo**
SP: 3	
La aplicación móvil ha de presentar una pantalla destinada a la lectura de códigos QR. En dicha pantalla tiene que haber un lector que se active cuando el usuario así lo decida, que sea capaz de decodificar códigos QR mediante el uso de la cámara del dispositivo móvil. La aplicación debe solicitar permiso al usuario para poder usar la cámara.	
• El lector QR se debe poder activar y desactivar pulsando un botón. • La aplicación deberá mostrar un mensaje de error cuando se escaneé un código cuya información no se corresponda con un local del sistema.	

ID: 2	**Como usuario estándar quiero que la aplicación móvil guarde el estado de que estoy dentro de un local para poder cerrarla y abrirla solo cuándo quiera marchar.**
SP: 3	
La aplicación móvil ha de mantener de manera persistente el estado de si el usuario está o no dentro de un local, de tal manera que se haga diferencia entre escanear un QR para entrar o para salir. Dicho estado debe verse reflejado en la pantalla para que el usuario esté informado en todo momento.	
• La aplicación mostrará al usuario la imagen de una puerta abierta o cerrada en función de si se encuentra dentro o fuera del local. • Como parte del feedback de estado que recibirá el usuario, el botón de activación del escáner tendrá un color distinto según el estado del usuario.	

ID: 3	**Como dueño de un local quiero poder registrarlo mediante la aplicación web para poder colaborar con el rastreo de casos de coronavirus.**
SP: 6	
La aplicación web tendrá una pantalla de registro de locales en la que se presentará un formulario para que los dueños de los locales lo rellenen y registren así su establecimiento en el sistema. Todos los campos del formulario deben validar su contenido.	
• El formulario solicitará el nombre, la dirección y el aforo máximo del local. • En caso de que los datos de algún campo sean incorrectos, se ha de mostrar el error en dicho campo. • El campo del aforo solamente debe aceptar entradas numéricas positivas.	

• El sistema asignará un identificador único al local.

ID: 4	**Como dueño de un local quiero que el sistema genere un código QR único que lo identifique para que mis clientes puedan escanearlo al entrar y salir de él.**
SP: 3	
La aplicación web, una vez se ha validado que todos los campos del formulario de registro de locales son correctos, generará un código QR que contenga el identificador del local creado.	
• El QR debe codificar el identificador único del local, que se trata de una cadena alfanumérica. • La resolución del QR debe permitir que sea rápidamente detectado por la cámara de un dispositivo móvil.	

ID: 5	**Como dueño de un local quiero que la aplicación web me muestre el código QR que se ha generado para poder descargarlo.**
SP: 5	
Tras completar de manera satisfactoria el registro del nuevo local, la aplicación web mostrará el código QR que se ha generado y dará la opción al usuario de descargarlo a su equipo en formato imagen.	
• Tras registrar el local, el usuario será dirigido a una nueva pantalla. • En esta pantalla se mostrará el código QR junto con dos botones, uno para volver atrás y otro para descargarlo. • El identificador del local irá encriptado en la URL. • En caso de querer forzar el sistema introduciendo una URL con un parámetro erróneo, la pantalla indicará que existe un error. • El código QR se descargará en formato PNG.	

ID: 6	**Como usuario estándar quiero que se almacenen en mi dispositivo los registros de entrada y salida de locales para que se me avise en caso de haber contactado con un positivo por coronavirus.**
SP: 8	

La aplicación móvil deberá almacenar de manera persistente en el tiempo las fechas exactas de entrada y salida de los locales a los que acude el usuario. Únicamente se deben guardar dichas fechas, acompañadas del identificador del local correspondiente, sin que se almacene ningún tipo de datos personales o de ubicación.

• Se deberá utilizar una base de datos local en el dispositivo móvil para almacenar los datos de visitas a locales.
• Cada visita tendrá los siguientes campos:
  ○ Identificador del local
  ○ Fecha y hora de entrada
  ○ Fecha y hora de salida
• Cuando se escanea por primera vez el código de un local, se creará una nueva visita con el identificador y la fecha y hora de entrada.
• Cuando se escanee de nuevo el QR para salir, se completará dicha visita añadiendo la fecha y hora de salida.
• En el caso posible (pero poco probable) de que un usuario se olvide de escanear el QR para salir, se determinará como fecha y hora de salida aquella en la que el usuario acceda al próximo local, dado que no es posible dictaminar con exactitud la hora estimada de salida de un local del usuario sin realizar otro tipo de tratamiento de datos más intrusivo.

ID: 7	**Como dueño de un local quiero que no se puedan registrar locales duplicados para que no exista confusión sobre cuál es el local real.**
SP: 2	
El sistema deberá comprobar que no se registran locales duplicados. Para ello se permitirá que existan varios locales con el mismo nombre, pero se prohibirá registrar un local en una dirección que ya esté en uso.	
• En caso de que se intente registrar un local con una dirección que ya esté en uso, la aplicación web mostrará un mensaje de error.	

ID: 8	**Como usuario estándar quiero que la aplicación móvil tenga un menú inferior para poder navegar por ella.**
SP: 3	
La aplicación móvil contará con un menú de navegación situado en la parte inferior de la pantalla para que los usuarios puedan navegar por sus distintas opciones. Dicho menú seguirá el estándar *Material* para que sea fácilmente usable por el usuario.	
• El menú debe estar en la parte inferior de la pantalla. • Se creará un botón para la pantalla del lector QR. • Cada botón del menú tendrá un texto corto y un icono que representen la funcionalidad de la pantalla a la que se accede a través de él.	

ID: 9 **SP: 3**	**Como usuario estándar quiero poder comunicar mi resultado positivo en una prueba al sistema para que los usuarios que hayan coincidido conmigo sean notificados.**
La aplicación móvil tendrá una pantalla destinada a reportar un resultado positivo obtenido en una prueba de COVID-19. Para reportar dicho positivo, el usuario indicará el código de rastreo que le habrán dado las autoridades sanitarias a la hora de notificarle su resultado, junto con la fecha en la que manifestó los síntomas por primera vez. En caso de que el usuario sea asintomático, en su lugar indicará la fecha en la que se realizó la prueba.	
• Se habilitará un campo numérico para que el usuario introduzca el código de rastreo. • Un código de rastreo tiene 12 dígitos, por lo que la aplicación validará que el introducido cumpla con dicha longitud. • Si el código es incorrecto, se mostrará un mensaje de error. • Se habilitará un campo para introducir la fecha de manifestación de los síntomas, cuyo valor por defecto será la fecha actual. • Al tocar sobre dicho campo, se abrirá un calendario en el que el usuario podrá seleccionar el día que manifestó los síntomas o se realizó la prueba. • La aplicación validará que la fecha seleccionada no sea posterior a la actual. • En caso de que la fecha sea posterior a la actual, se mostrará un mensaje de error. • Si las dos entradas son correctas, se mostrará un mensaje al usuario que indique que se notificó el positivo correctamente.	

ID: 10 **SP: 2**	**Como usuario estándar quiero que se produzca una alerta cada vez que se notifica un nuevo positivo para que se pueda informar a todos los posibles afectados.**
Cuando se reporta un nuevo positivo, la aplicación móvil enviará al servidor los datos necesarios para crear una nueva alerta. Además de enviarla al servidor, la alerta se almacenará en el dispositivo móvil.	

• Cada alerta contendrá la siguiente información:
  ○ Fecha de notificación de la alerta
  ○ Fecha de inicio de los síntomas o fecha de realización de la prueba
  ○ Lista de visitas que se produjeron dentro de los plazos de infectividad del virus. Cada visita contendrá la siguiente información:
  ▪ Identificador del local
  ▪ Fecha y hora de entrada
  ▪ Fecha y hora de salida	

ID: 11 **SP: 3**	**Como administrador del sistema quiero poder modificar los periodos de infectividad y las variables que determinan si un contacto es estrecho para que el rastreo se ajuste a los criterios científicos que existan en cada momento.**

La aplicación web contará con una sección para el administrador desde la que se podrán modificar los valores de las variables de las que depende el contagio del virus. Esta sección tendrá formato de panel, con un campo numérico para cada variable existente acompañado de un botón que permita actualizar su valor. El valor de estas variables debe estar presente en el servidor para que la aplicación móvil tenga acceso a él.

• Por cada variable, el panel tendrá los siguientes elementos:
  ○ Breve descripción de la variable
  ○ Campo numérico editable con su valor actual
  ○ Botón para actualizar el valor de la variable en el servidor
• A esta sección solo podrá acceder el administrador del sistema tras haberse autenticado previamente.
• Si se intenta acceder al panel sin estar autenticado, el usuario será redirigido a la página de inicio de sesión.

ID: 12	**Como administrador del sistema quiero poder autenticarme en la aplicación web para acceder al panel de variables.**
SP: 3	
La aplicación web contará con una página de inicio de sesión para que el administrador se pueda identificar. Tras identificarse correctamente se generará un token de sesión que caducará a los 5 minutos, que será imprescindible para acceder al panel y para actualizar los valores de las variables.	
• La página de inicio de sesión tendrá un formulario con dos campos: usuario y contraseña. • Si alguno de los campos es incorrecto, se marcarán ambos como inválidos para mayor seguridad. • Si el inicio de sesión es correcto, se generará un token que identificará al administrador y le permitirá acceder al panel.	

ID: 13	**Como usuario estándar quiero que la aplicación móvil únicamente me notifique de aquellos positivos con los que pueda haber tenido contacto para que no me lleguen más notificaciones de las necesarias.**
SP: 3	
La aplicación móvil utilizará un algoritmo de filtrado de alertas para decidir cuáles afectan o no al usuario. Para ello se utilizarán las variables de contagio establecidas por los organismos sanitarios que se encuentran almacenadas en el servidor.	

• El filtrado de alertas se hará en base a los siguientes criterios:
  ○ La alerta no se encuentra en el dispositivo, lo que indicaría que la produjo el mismo usuario.
  ○ Coincidencia en el mismo local del usuario y el positivo.
  ○ Dicha coincidencia se produce en el periodo de infectividad del usuario contagiado.
  ○ Coinciden durante, al menos, el tiempo mínimo necesario para que se considere un contacto estrecho.	

ID: 14	**Como usuario estándar quiero poder ver una lista con las alertas que me afecten para poder tomar las precauciones necesarias cuando aparezca alguna alerta nueva.**
SP: 3	
La aplicación móvil tendrá una pantalla que ha de contener una lista de alertas o un mensaje indicando que no hay alertas que le afecten al usuario si dicha lista está vacía. Esta nueva pantalla tiene que ser accesible desde el menú de navegación inferior de la aplicación. Las alertas que se mostrarán en esta lista son aquellas que han sido filtradas previamente.	

• Se creará una pantalla nueva y un botón en el menú desde el que acceder a ella.
• Esta pantalla tendrá el siguiente aspecto:
  o En la parte superior habrá un título indicando su función.
  o En la parte inferior habrá una sección que mostrará el periodo de sincronización de nuevas alertas.
  o El resto de la pantalla estará ocupado por la lista de alertas.
• Cada alerta estará representada por una tarjeta que muestre la fecha y hora de posible contacto y el lugar en el que se produjo.
• Cuando la lista esté vacía, se mostrará un mensaje indicando que no hay alertas que afecten al usuario.	

ID: 15	**Como usuario estándar quiero poder decidir cada cuanto tiempo la aplicación móvil va a sincronizarse con el servidor en búsqueda de nuevas alertas para poder optimizar la batería y el uso de datos móviles en función de mis intereses.**
SP: 2	
En la pantalla que se muestran las alertas habrá una sección destinada a mostrar y configurar el periodo de sincronización. Dicha sección mostrará el periodo seleccionado y tendrá un botón para cambiarlo. Al pulsar en él se abrirá una ventana flotante con las distintas frecuencias para que el usuario escoja una. Esta funcionalidad solo estará disponible en los dispositivos Android, ya que el sistema operativo iOS realizará la sincronización cuando lo crea conveniente.	
• Se ha de comprobar que tipo de dispositivo está ejecutando la aplicación con el fin de mostrar u ocultar la selección de frecuencia de sincronización. • Al pulsar el botón para cambiar la frecuencia se mostrará un “popup”. • El “popup” tendrá una lista de frecuencias seleccionables de una en una. • El “popup” también tendrá un botón para confirmar la selección.	

ID: 16	**Como usuario estándar quiero poder sincronizar la aplicación móvil manualmente para recibir nuevas alertas en el momento que quiera.**
SP: 2	

La lista de alertas se ha de poder actualizar manualmente, sin necesidad de esperar a la sincronización automática. Se habilitará el mecanismo "pull-to-refresh" para realizar dicha sincronización y actualizar la lista en caso de que apareciera alguna alerta nueva. También se realizará la sincronización automáticamente cada vez que se abre la aplicación.
• Al desplazar hacia arriba la lista estando ya en el primer elemento se activará la sincronización. • La hora de esta sincronización servirá para calcular la próxima sincronización automática según el periodo seleccionado.

ID: 17	**Como usuario estándar quiero que se me explique qué son las alertas para conocer cómo funciona el sistema y en qué medida me afectan.**
SP: 1	
Se habilitará una vista de información en la pantalla del listado de alertas de la aplicación móvil en la que se explicará al usuario qué son y cómo le afectan para garantizar su conocimiento y tranquilidad en caso de que reciba alguna.	
• En la parte superior derecha de la pantalla se mostrará un icono de información. • Al pulsar en él se mostrará una ventana flotante con una breve explicación de qué son las alertas y cómo ha de actuar el usuario si recibe alguna. • Esta ventana tendrá un botón para cerrarla.	

ID: 18	**Como usuario estándar quiero poder saber qué locales usan el sistema de rastreo para tomarlo en cuenta a la hora de decidir ir a donde ir en situación de pandemia.**
SP: 3	
Se ha de crear una pantalla en la aplicación web que cuente con una barra de búsqueda. En esta pantalla el usuario podrá introducir el nombre de un local, y la aplicación le mostrará todos los locales presentes en el sistema con ese nombre.	

• Esta nueva pantalla tendrá una barra de búsqueda y un botón para realizarla.
• Por cada local cuyo nombre coincida con el introducido se mostrará una tarjeta con los siguientes datos:
  ○ Nombre del local
  ○ Dirección
  ○ Aforo máximo
  ○ Código QR del local	

ID: 19	**Como usuario estándar quiero que la aplicación web tenga un menú superior para poder navegar por todas sus opciones cómodamente.**
SP: 1	
Se habilitará un menú de navegación en la parte superior de las pantallas de la aplicación web que son accesibles para todo tipo de usuarios que permita desplazarse de una a otra en cualquier momento.	
• Cada botón del menú estará formado por un título y un icono que describan la pantalla a la que se accede.	

ID: 20	**Como usuario estándar quiero que la aplicación web se adapte a todo tipo de pantallas para poder usarla desde cualquier dispositivo, en especial dispositivos móviles.**
SP: 2	
Todas las pantallas y funcionalidades de la aplicación web se tienen que adaptar de manera dinámica al tamaño de la pantalla.	

ID: 21	**Como administrador del sistema quiero poder recargar la pantalla del panel de administración sin tener que volver a identificarme.**
SP: 3	
El token de sesión generado al identificarse como administrador se debe conservar durante toda la navegación por la aplicación para que, si no se cumplen los 5 minutos, no sea necesario volver a identificarse para acceder al panel de variables.	

• El token de sesión se conservará al realizar las siguientes acciones:
  ○ Recargar el panel de alertas
  ○ Navegar a otra pantalla de la aplicación	

ID: 22	**Como usuario estándar quiero que la aplicación móvil muestre una notificación cada vez que se realiza la sincronización para conocer sus resultados.**
SP: 4	
Cada vez que se realice una sincronización de alertas, la aplicación móvil mostrará una notificación al usuario, indicando si se han encontrado o no nuevas alertas que le afecten.	

• Cada notificación estará compuesta por los siguientes elementos:
  ○ Nombre de la aplicación.
  ○ Icono de la aplicación.
  ○ Mensaje indicando el resultado de la sincronización.	

ID: 23	**Como usuario estándar quiero que la aplicación móvil Android se sincronice automáticamente en busca de nuevas alertas conforme al periodo que he seleccionado para que no sea necesario que actualice manualmente la lista.**
SP: 8	
En los dispositivos Android se realizará una sincronización automática con la frecuencia seleccionada por el usuario. Esta sincronización automática ha de llevarse a cabo con la aplicación en primer y segundo plano, sin importar el estado del dispositivo. En caso de que el usuario no quiera que se realice la sincronización automática, podrá cancelarla cerrando por completo la aplicación, para que no se hagan procesamientos en segundo plano si el usuario no está conforme con ello.	
• La sincronización automática no debe influir en los tiempos de respuesta de la aplicación. • Aún con la aplicación cerrada, si esta se encuentra en segundo plano, la sincronización se hará igualmente. • Sólo no se producirá la sincronización si el usuario mata por completo la aplicación. • La sincronización automática se hará independientemente del estado de bloqueo del dispositivo.	

ID: 24	**Como usuario estándar quiero que la aplicación móvil iOS se sincronice automáticamente en busca de nuevas alertas para que no sea necesario que actualice manualmente la lista.**
SP: 4	
En los dispositivos iOS se realizará una sincronización automática con la frecuencia determinada por el sistema operativo. Esta sincronización automática ha de llevarse a cabo con la aplicación en primer y segundo plano, sin importar el estado del dispositivo. En caso de que el usuario no quiera que se realice la sincronización automática, podrá cancelarla cerrando por completo la aplicación, para que no se hagan procesamientos en segundo plano si el usuario no está conforme con ello.	

- La sincronización automática no debe influir en los tiempos de respuesta de la aplicación.
- Aún con la aplicación cerrada, si esta se encuentra en segundo plano, la sincronización se hará igualmente.
- Sólo no se producirá la sincronización si el usuario mata por completo la aplicación.
- La sincronización automática se hará independientemente del estado de bloqueo del dispositivo.

ID: 25	**Como dueño de producto quiero poder mostrar el funcionamiento del prototipo del sistema sin necesidad de haberlo integrado con los sistemas informáticos sanitarios para poder realizar demostraciones de su funcionamiento previas a su implantación final.**
SP: 5	

Como parte del prototipo inicial se incluirá un sistema de validación de alertas independiente al real. Este sistema consistirá en una sección en la aplicación web desde la que poder confirmar o eliminar las alertas creadas. Hasta que una alerta no esté validada, se descartará cuando los dispositivos sincronicen en busca de nuevas alertas.

- Puesto que se trata de algo provisional, la pantalla de validación de alertas no se incluirá en el menú de la aplicación web.
- En la pantalla de validación de alertas se mostrará una tarjeta por cada alerta nueva con su código de rastreo y la fecha de creación de la misma.
- En cada tarjeta habrá también dos botones, uno para eliminarla y otro para validarla.
- Las alertas eliminadas se borrarán del sistema.
- Solamente las alertas validadas podrán ser notificadas a los usuarios afectados.

ID: 26	**Como usuario estándar quiero que la aplicación móvil tenga un icono personalizado para poder identificarla entre el resto de las aplicaciones instaladas.**
SP: 1	

Se ha de diseñar y configurar un icono de aplicación personalizado que represente la funcionalidad y el objetivo de la aplicación.

- El icono de la aplicación tendrá, al menos, una puerta o entrada y un virus.
- Se usarán los mismos colores que en el resto de la aplicación para mantener la consistencia visual.

ID: 27	**Como usuario estándar quiero que la aplicación móvil tenga una pantalla de bienvenida para ver algo que no sea una pantalla en blanco mientras la aplicación está iniciándose.**
SP: 1	

Se creará una pantalla de bienvenida que se mostrará al iniciar la aplicación mientras se carga todo lo necesario.
• La pantalla de bienvenida será del color principal de la aplicación. • En el centro se mostrará el icono de la aplicación. • Se cerrará automáticamente cuando se complete la inicialización de la aplicación.

ID: 28	**Como usuario estándar quiero que la navegación hacia atrás por la aplicación sea la esperada para que no cerrarla sin pretenderlo.**
SP: 2	
• Si hay alguna ventana flotante visible, el botón de atrás la cerrará. • Cuando no haya ninguna, desde las pantallas de reportar un positivo y la lista de alertas se navegará a la pantalla inicial, es decir, la del escáner de códigos QR. • Si la pantalla actual es la del escáner, se cerrará la aplicación. • La pantalla de bienvenida nunca se ha de mostrar al navegar hacia atrás.	

ID: 29	**Como usuario estándar necesito que se me informe la primera vez que abro la aplicación sobre la política y el tratamiento de datos que se hace en ella para conocer su funcionamiento que se respete mi privacidad y mis derechos digitales.**
SP: 4	
Cuando se inicie la aplicación por primera vez en un dispositivo se tiene que mostrar una ventana con los términos y condiciones de uso de la aplicación, explicándole al usuario que datos se tratan y cómo, con el fin de garantizar su conocimiento y fomentar que confíe en la aplicación. Además, se solicitará su consentimiento explícito para permitirle usar la aplicación.	
• Al abrir por primera vez la aplicación se mostrará una ventana flotante que contendrá los términos y condiciones. • Dicha ventana permanecerá en primer plano hasta que el usuario los acepte pulsando un botón situado al final del texto de los términos y condiciones. • Una vez aceptados, se mostrará la aplicación con normalidad. • No se realizará la sincronización al abrir la aplicación si es la primera vez que se abre en el dispositivo. • Una vez aceptados, los términos y condiciones no se volverán a mostrar nunca salvo que se desinstale y se vuelva a instalar la aplicación.	

ID: 30	**Como usuario estándar necesito que exista una página en la aplicación web donde poder consultar la política y el tratamiento de datos que se hace en ella para conocer su funcionamiento que se respete mi privacidad y mis derechos digitales.**
SP: 4	

Se habilitará una página en la aplicación web, accesible desde cualquier otra sección, en la que estará descrito el tratamiento de datos que se realiza en el sistema siguiendo la normativa vigente (LOPD/RGPD).
• La página de explicación de la política de privacidad será accesible desde toda la aplicación web. • Para acceder a ella se habilitará un enlace en el pie de página de todas las pantallas de la aplicación web. • Esta sección debe contener todos los apartados que sean requeridos por la LOPD/RGPD referidos al tratamiento de datos personales. • También se describirá el conjunto del tratamiento de los datos que se hace en la aplicación web, aunque no se trate de datos personales.

ID: 31	**Como dueño de un local quiero que el sistema lleve un registro de la ocupación en tiempo real del local para que no se supere el aforo máximo.**
SP: 3	
El sistema mantendrá un registro en tiempo real de la cantidad de personas que hay en un local con el fin de controlar el aforo en todo momento.	
• El campo de ocupación de un local tiene que actualizarse en tiempo real cada vez que una persona entra o abandona dicho local. • Cada vez que una persona escanea el código de un local para entrar, se sumará uno a la ocupación. • Cuando una persona escanea el código de un local para salir, se restará uno a la ocupación.	

ID: 32	**Como dueño del local quiero que el sistema niegue la entrada a nuevas personas si el aforo ya está completo para prevenir que la mayor concentración de gente provoque contagios.**
SP: 3	
La aplicación móvil prohibirá la entrada a un local a una persona si en el momento de escanear su código QR, la ocupación del local es igual a su aforo máximo.	
• Al escanear el código de un local, si la ocupación es igual que el aforo máximo, no se registrará la entrada. • La aplicación mostrará un mensaje al usuario diciéndole que el local está lleno.	

7.2.3 Actores del Sistema

A continuación, se describen los actores identificados en el sistema, comentando brevemente por qué aparecen y que particularidades tienen.

7.2.3.1 Usuario estándar

Un usuario estándar es todo aquel ciudadano residente en la ciudad, región o país en el que se haya implantado el uso de este sistema de rastreo. Todos los ciudadanos que dispongan de acceso a internet tendrán acceso a la aplicación web y serán por tanto usuarios potenciales. Además, todos los que dispongan de un dispositivo móvil o tableta con sistema operativo Android o iOS podrán descargarse la aplicación móvil.

Como usuario estándar, cualquier ciudadano con la aplicación móvil instalada podrá registrar sus entradas y salidas de locales, así como reportar su positivo por COVID-19. Todos recibirán también alertas de posibles contactos en caso de que los hayan tenido. Además, a través de la aplicación web podrán buscar si un local utiliza este sistema de rastreo y consultar su aforo máximo.

7.2.3.2 Dueño de un local

Como cualquier otro ciudadano, los dueños de los locales podrán usar la aplicación móvil para registrar sus visitas a su local o a cualquier otro, para registrar que han dado positivo en una prueba y para estar al tanto de posibles contactos que hayan sufrido con algún infectado.

En cuanto a la aplicación web, además del buscador de locales, los dueños de locales podrán registrar su establecimiento en el sistema. En caso de que el sistema de rastreo se ponga en marcha, será necesario que el formulario de registro valide que, efectivamente, el local lo está registrando el dueño o en su defecto un delegado nombrado por él. Para más información acerca de este último punto se puede consultar el capítulo 12, apartado 2, de este documento.

7.2.3.3 Administrador del sistema

Además de todas las funcionalidades asociadas a los usuarios estándar, el administrador del sistema tendrá acceso a un panel de control habilitado en la aplicación web desde el que podrá consultar y modificar el valor de las variables de infección del virus según los descubrimientos científicos y los datos probados hasta la fecha.

7.2.3.4 Dueño del producto

Pese a que en una futura versión de este sistema este actor desaparecerá, hasta que no se integre este proyecto con los sistemas informáticos sanitarios actuales, el dueño del producto contará con una sección en la aplicación web destinada a la validación de nuevos positivos. El objetivo de esta funcionalidad es meramente demostrativo y estará eliminada en el momento de la puesta en producción del sistema.

7.3 Identificación de los Subsistemas en la Fase de Análisis

En esta sección se enumeran y describen brevemente los subsistemas que se han identificado al comienzo del desarrollo del proyecto, tras mantener las primeras conversaciones con el dueño del producto y comenzar a elaborar historias de usuario.

7.3.1 Base de datos central

Pese a que los datos de los usuarios se almacenarán localmente en sus dispositivos, será necesaria la creación de una base de datos central en la que se almacenarán los datos de los locales que se registren en el sistema, las nuevas alertas que se produzcan y las variables de contagio.

7.3.2 API REST

A modo de enlace de las aplicaciones con la base de datos se encuentra un servidor *backend*. Este servidor proporcionará una API REST cuyo objetivo será comunicar las aplicaciones con la base de datos, realizando ciertas operaciones en el servidor y liberando de carga a los dispositivos clientes.

7.3.3 Aplicación móvil

Como parte imprescindible del sistema de rastreo se encuentra la aplicación móvil. Se desarrollará una aplicación móvil multiplataforma que, mediante la API REST, se comunique con el resto del sistema. Esta aplicación contará también con un módulo de base de datos local.

7.3.4 Aplicación web

Para complementar a la aplicación móvil, se creará una aplicación web para gestionar el registro de nuevos locales. Además, contará con una sección de administración para el control de los valores de las variables de contagio. Esta aplicación delegará todas las labores de procesamiento en el servidor, por lo que hará uso también de la API REST para comunicarse con el resto del sistema de manera ligera.

7.4 Análisis de Interfaces de Usuario

En este apartado se describen de manera detallada las interfaces de usuario bocetadas tras analizar las historias de usuario correspondientes. A continuación, se recopilan dichos bocetos dibujados durante el transcurso de las iteraciones, previos al diseño y la maquetación de los mismos en sus respectivas plataformas y lenguajes.

7.4.1 Interfaces de usuario de la aplicación web

7.4.1.1 Formulario de registro de locales

Figura 7.1. Boceto del formulario de registro de locales

Esta pantalla pretendía ser la pantalla inicial de la aplicación web. Cuenta con una cabecera con el nombre del sistema y con un formulario con 3 campos: nombre, dirección y aforo, y un botón para aceptar el registro. En caso de que algún campo sea incorrecto, se resaltará en color rojo. La cabecera en todas las pantallas será táctil, llevando al usuario a la página inicial (esta) cuando pulse sobre el título.

En este boceto inicial no se contempla el menú de navegación que fue introducido posteriormente.

7.4.1.2 Pantalla de búsqueda de locales

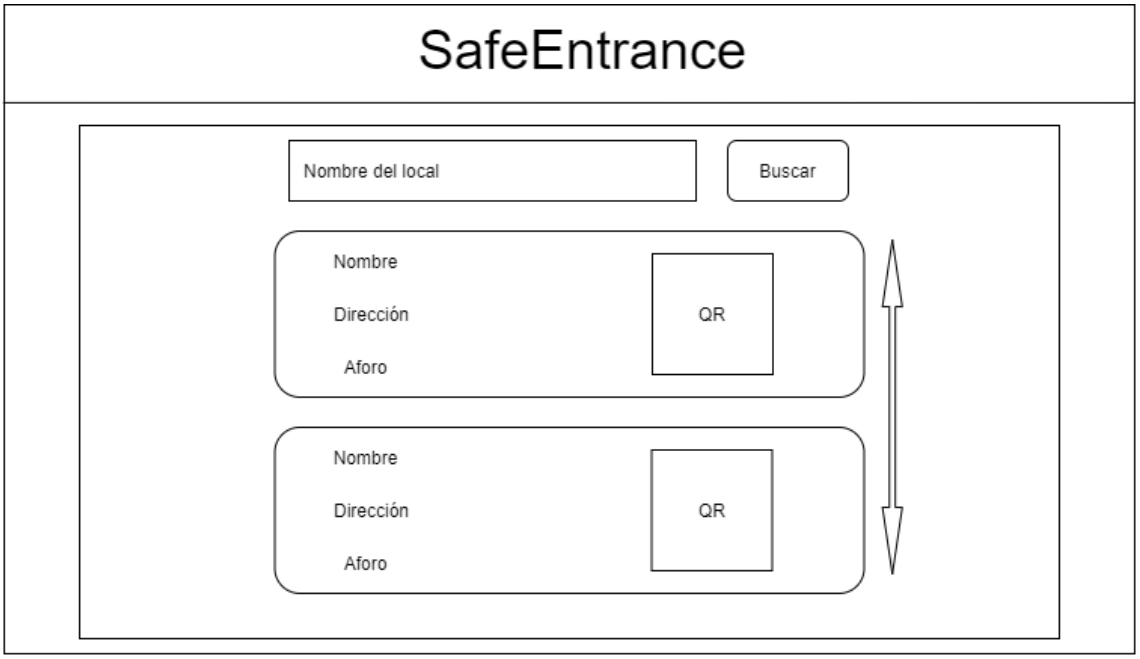

Figura 7.2. Boceto de la pantalla de búsqueda de locales

Al igual que en el formulario de registro de locales, no se contempla en el boceto el menú de navegación. Nuevamente, se puede apreciar la cabecera con el título en la parte superior, y en este caso la parte principal de la pantalla tendrá dos secciones. La primera consiste en un campo de introducción de texto para el nombre del local junto con un botón de buscar. La segunda es una lista desplazable de tarjetas. Cada tarjeta contiene el nombre, la dirección y el aforo en formato texto de un local, junto con su código QR en la parte derecha. La lista será desplazable verticalmente.

7.4.1.3 Menú de navegación

Figura 7.3. Boceto del menú de navegación de la aplicación web

Con este boceto se contempla la incorporación de un menú de navegación en forma de barra, situado justo debajo de la cabecera. Este menú tendría una pestaña por cada pantalla de la aplicación, y cada una de esas pestañas estaría compuesta por un título y un icono.

7.4.1.4 Pantalla de descarga del código QR

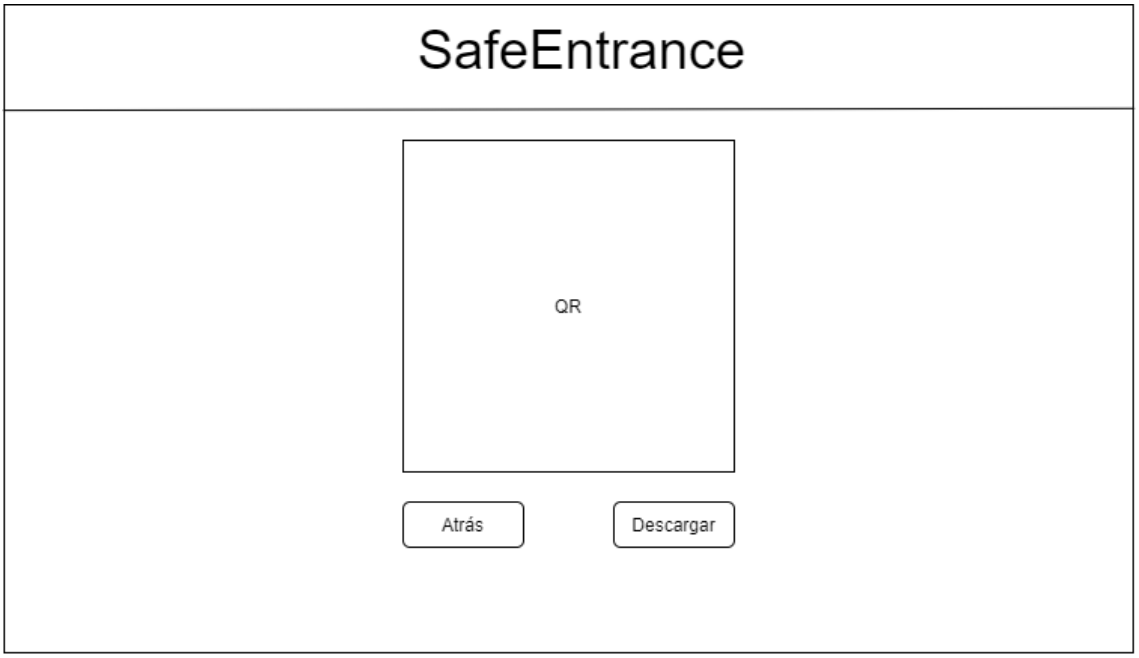

Figura 7.4. Boceto de la pantalla de generación y descarga del QR

Una vez registrado el local, en esta pantalla se mostraría en grande el código QR generado. Debajo de él habrá dos botones, uno para volver al inicio y otro para descargar el código. Como en todas las pantallas de la aplicación, se busca mantener una estética material minimalista que sea sencilla e intuitiva de usar.

En caso de que se trate de acceder a esta pantalla de manera incorrecta o haya algún error al generar el código QR se mostrará la pantalla indicada a continuación, en la que el código será sustituido por un mensaje de error.

Figura 7.5. Boceto de la pantalla de generación de QR incorrecto

7.4.1.5 Formulario de autenticación del administrador

Figura 7.6. Boceto del formulario de inicio de sesión como administrador

Esta pantalla es un formulario de inicio de sesión estándar, con sus campos de usuario y contraseña y un botón para aceptar y entrar. En caso de que los datos no sean correctos, los campos se resaltarán en color rojo y se le mostrará un mensaje al usuario.

7.4.1.6 Panel de control de variables de contagio

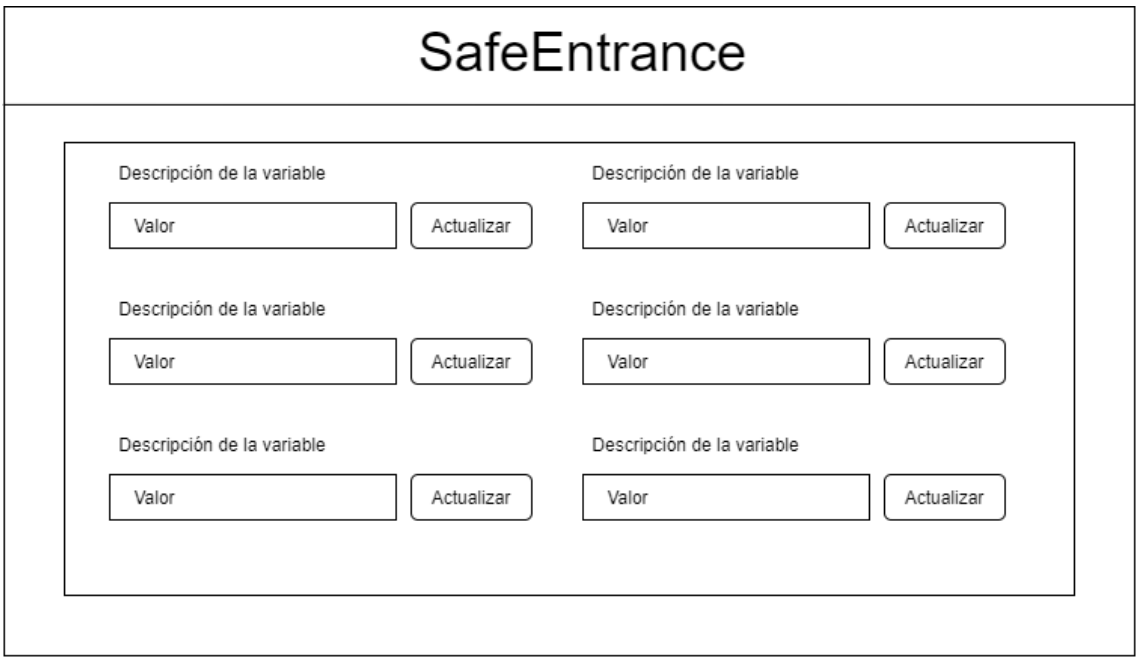

Figura 7.7. Boceto del panel de control de variables de contagio

Esta pantalla tendrá formato de panel y contendrá, para cada variable de contagio presente en el sistema, una breve descripción de la misma, una entrada numérica que mostrará su valor y se podrá editar y un botón para actualizar el valor de la variable en el servidor. En caso de que haya muchas variables, el panel será desplazable verticalmente.

7.4.1.7 Lista de validación de nuevos positivos

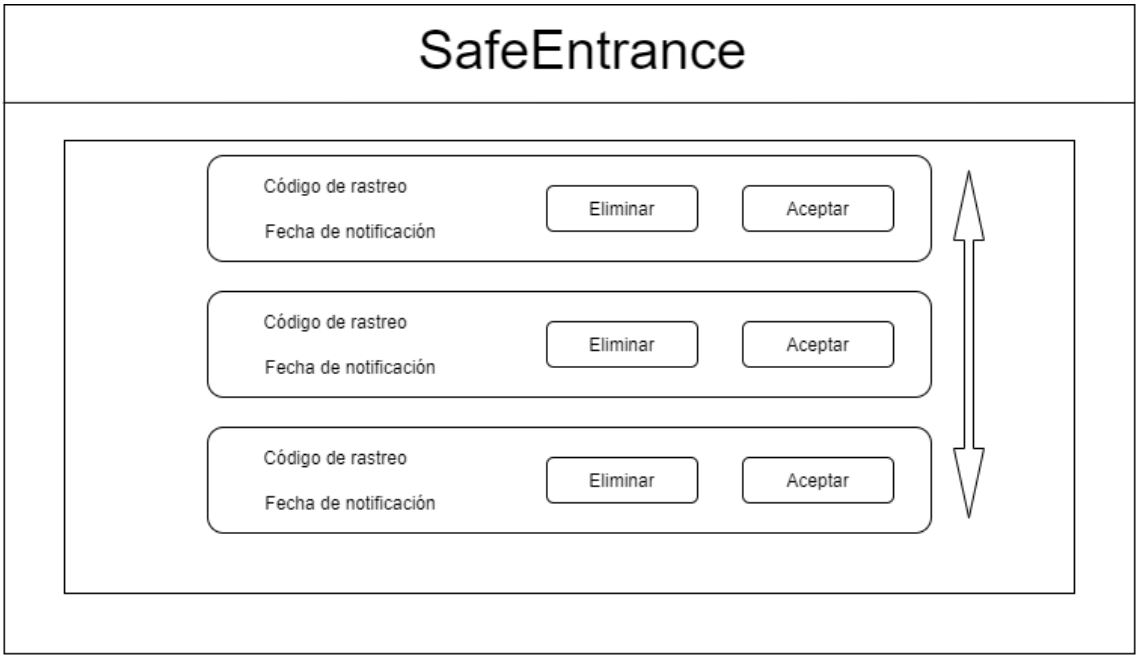

Figura 7.8. Boceto de la lista de nuevos positivos notificados

Esta vista contará con una lista de tarjetas desplazable verticalmente. Cada tarjeta representará un nuevo positivo notificado al sistema, y mostrará el código de rastreo correspondiente y la fecha en la que se notificó. Además, cada tarjeta contará con dos botones. El primero eliminará la alerta del sistema, y el segundo la confirmará. En cualquiera de los dos casos, tanto si se elimina como si se valida la alerta, la tarjeta desaparecerá de la lista.

7.4.1.8 Pantalla de aviso legal y política de privacidad

Figura 7.9. Boceto de la pantalla de aviso legal

En esta pantalla tan solo se mostrará el aviso legal y la explicación referida al tratamiento y la política de datos de la aplicación. En este boceto se incorpora además el pie de página con el enlace a esta vista. Dicho pie de página estará presente en todas las pantallas de la aplicación.

7.4.2 Interfaces de usuario de la aplicación móvil

7.4.2.1 Pantalla del lector de códigos QR

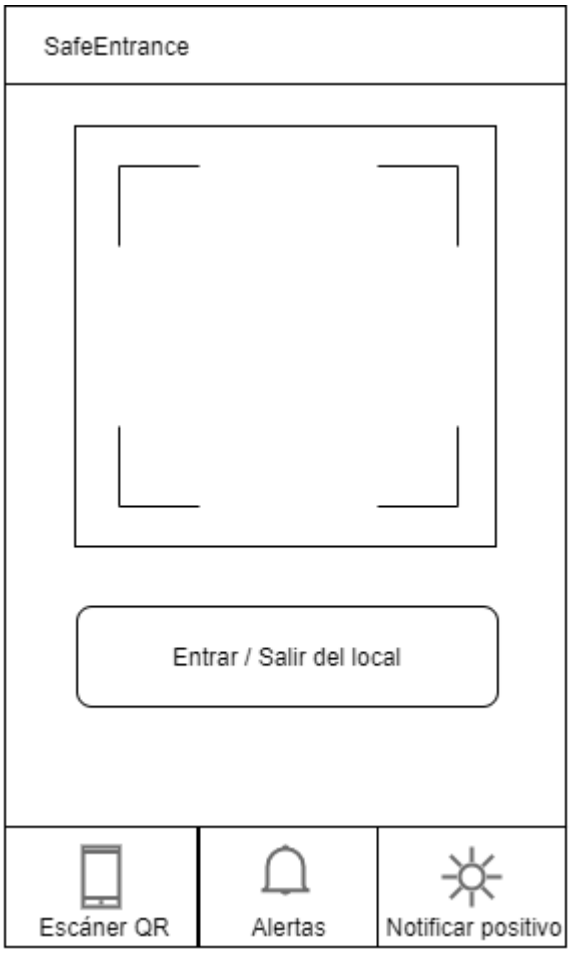

Figura 7.10. Boceto de la pantalla para escanear códigos QR

Esta pantalla será la primera que vea el usuario al abrir la aplicación. En ella habrá un lector de códigos QR junto con un botón que servirá para activar y desactivar dicho lector. El botón le indicará al usuario si va a entrar o a salir de un local al escanear el código.

Además de la pantalla, en este boceto se incluye también la barra inferior de navegación. En ella hay tres botones, uno para cada funcionalidad de la aplicación, y cada botón tendrá un título y un icono.

7.4.2.2 Pantalla para reportar positivos

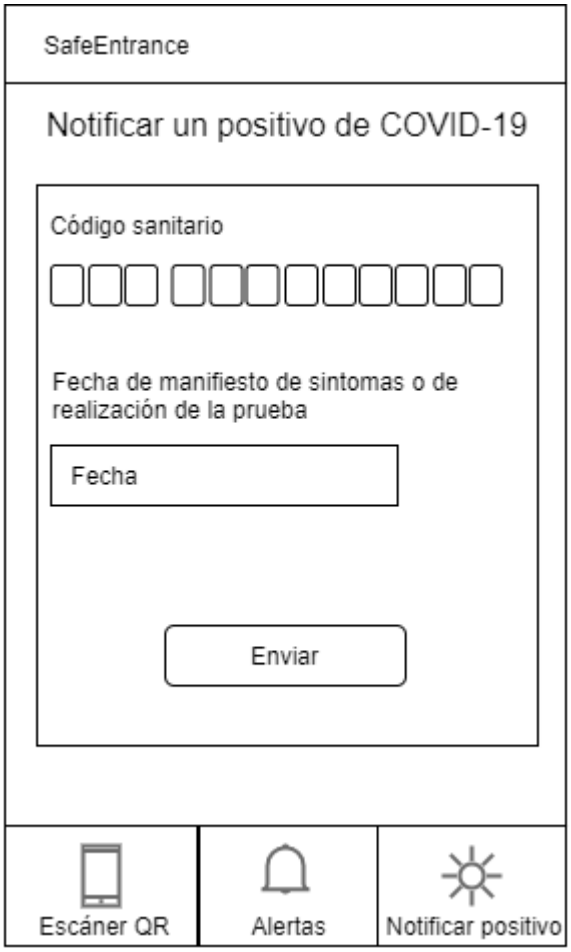

Figura 7.11. Boceto del formulario para notificar positivos

En esta pantalla el usuario encontrará un formulario simple, con dos entradas, que servirá para notificar su positivo por coronavirus. El primero de los campos, dividido en 12 pequeños recuadros, está destinado a que se introduzca el código de rastreo que le enviaron las autoridades sanitarias. El segundo campo es la entrada de la fecha en que manifestó los síntomas por primera vez o, en caso de ser asintomático, la fecha de realización de la prueba. Al pulsar sobre este campo se abrirá un calendario en el que el usuario podrá seleccionar la fecha en cuestión.

7.4.2.3 Listado de posibles contactos con positivos

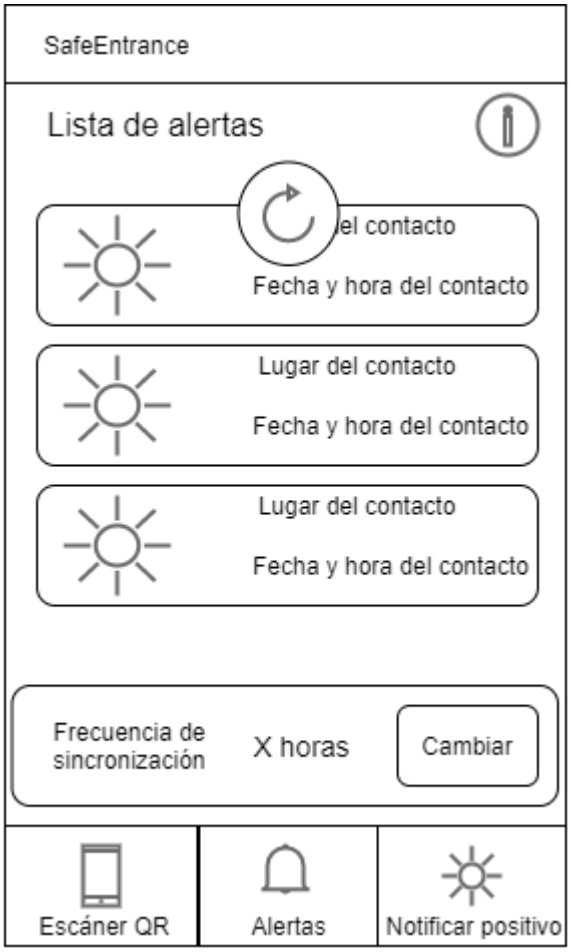

Figura 7.12. Boceto de la pantalla de lista de posibles contactos

Esta pantalla es la más compleja de la aplicación móvil. En la parte superior cuenta con el título de la sección y un botón de información. Al pulsar en él se abrirá una ventana flotante con explicación sobre qué son las alertas. En el centro de la pantalla se encuentra la lista de posibles contactos (alertas). Cada alerta estará representada con una tarjeta que tendrá en la parte izquierda el icono de un virus y en la parte derecha el nombre del local donde se produjo y la hora a la que se produjo. El icono de recargar circular situado sobre la lista representa el icono que se muestra al realizar la acción “pull-to-refresh.”

Por último, en la parte inferior, sobre la barra de navegación, se encuentra la sección de cambiar la frecuencia de sincronización automática de la aplicación. En ella se muestra la frecuencia seleccionada y un botón para cambiarla. Al pulsar dicho botón se abrirá una ventana flotante con las distintas opciones de frecuencia para que el usuario elija una. A continuación, se muestra el boceto de dicha ventana flotante, compuesta por una lista de frecuencias y un botón para aceptar la selección.

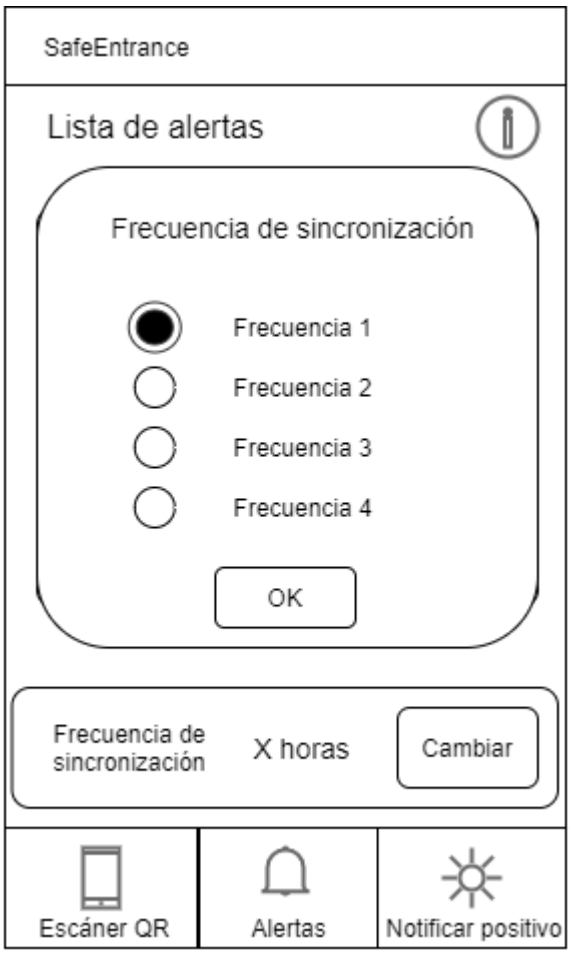

Figura 7.13. Boceto de la ventana de selección de frecuencia

7.4.3 Diagramas de Navegabilidad

7.4.3.1 Diagrama de navegabilidad de la aplicación móvil

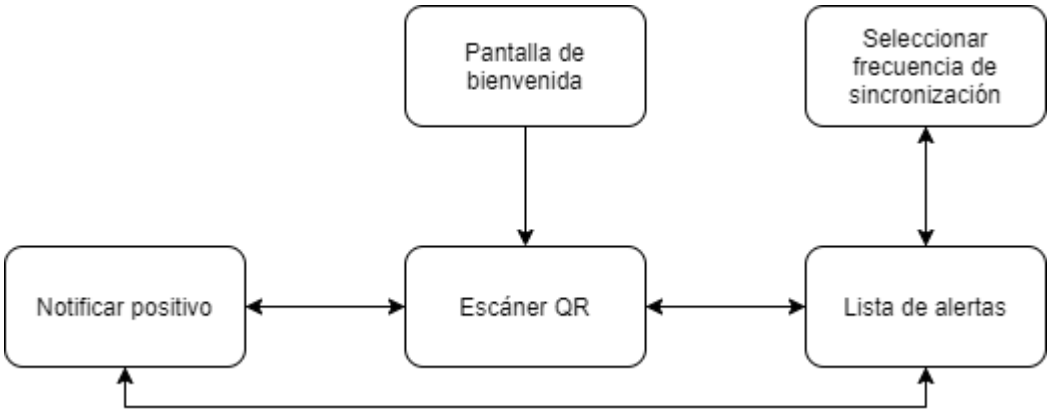

Figura 7.14. Mapa de navegación de la aplicación móvil

7.4.3.2 Diagrama de navegabilidad de la aplicación web

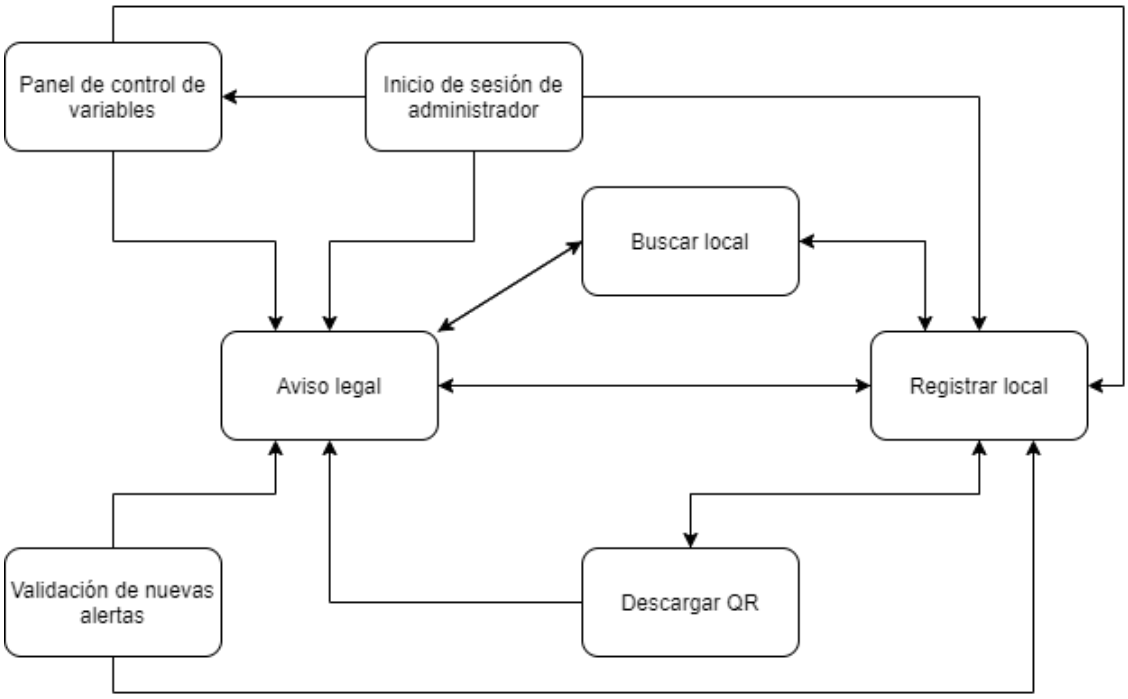

Figura 7.15. Mapa de navegación de la aplicación web

7.5 Especificación del Plan de Pruebas

En esta sección se describe en que consistirá el plan de pruebas del sistema y sus funciones, así como los mecanismos que se utilizarán para detectar errores y corregirlos a lo largo de todas las iteraciones del desarrollo.

Las pruebas contemplarán aspectos tanto de funcionalidad de la aplicación como de aspectos de los usuarios o clientes de la misma.

Se contemplarán tres tipos de pruebas:

- **Pruebas Unitarias**: Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código, o en este caso una clase individual que cumple con una función concreta. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Para este sistema, debido a la identificación de los cuatro subsistemas identificados en el apartado 7.3 y a la arquitectura que se pretende diseñar, se realizarán pruebas unitarias contra los servicios REST. Estas pruebas se automatizarán para ser ejecutadas como paso previo al despliegue del servidor mediante el sistema de CI/CD Azure Pipelines. Para la ejecución de estas pruebas se hará uso de un entorno de pruebas específico, que permita modificar la base de datos sin dañar los datos del sistema en producción.
- **Pruebas de aceptación del sistema**: Las pruebas de aceptación del sistema consisten en pruebas de interfaz y pruebas de integración del sistema construido completo, que permiten probar el conjunto de todo el sistema y que sus relaciones con otros sistemas que necesite son correctas, verificando así que todas sus especificaciones funcionales y técnicas se cumplen. Estas pruebas son cruciales para validar correctamente las historias de usuario. Previo paso para marcarlas como cerradas, el equipo de desarrollo realizará un conjunto extenso de pruebas para comprobar que los cambios introducidos funcionan correctamente antes de desplegarlos. En estas pruebas se debe verificar la correcta integración de los cambios en los tres subsistemas: servidor, aplicación web y aplicación móvil. La ejecución de algunas de estas pruebas se automatizará en la medida de lo posible mediante la herramienta TestProject en base a los criterios de aceptación definidos en las historias de usuario. En las reuniones de demostración realizadas al final de cada iteración se mostrará la ejecución y el resultado de estas pruebas automatizadas.
- **Pruebas de Usabilidad**: Este tipo de pruebas determinan la satisfacción del cliente con el producto final. Por este motivo, se realizarán pruebas de usabilidad y accesibilidad para la aplicación web y la aplicación móvil. La realización de dichas pruebas se hará mediante actividades guiadas con usuarios finales. Tras completar estas actividades, se entregará un cuestionario con preguntas cortas al usuario con el objetivo de evaluar el grado de usabilidad de ambas aplicaciones.

Capítulo 8. Diseño del Sistema

8.1 Arquitectura del Sistema

El sistema de rastreo estará compuesto por cuatro bloques de software claramente diferenciados. A continuación, se muestra un diagrama en el que se pueden apreciar dichos módulos y las relaciones que existen entre ellos.

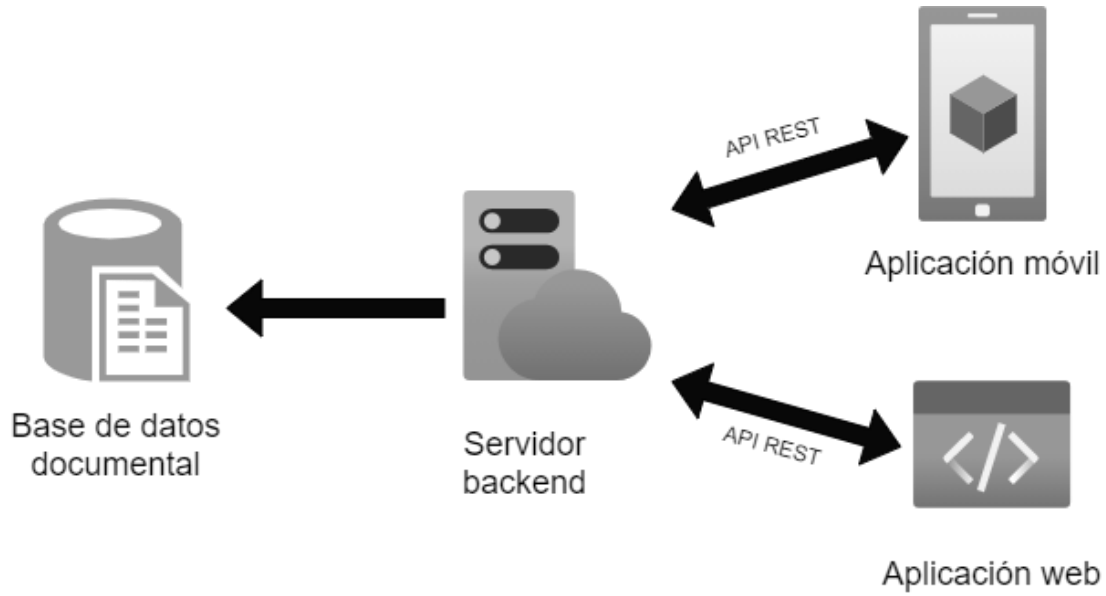

Figura 8.1. Arquitectura general del sistema

8.1.1 Arquitectura REST

REST (del inglés *Representational State Transfer*) es un estilo de arquitectura de software destinada a sistemas distribuidos cuya principal característica es que los datos y las funcionalidades se consideran recursos, los cuales son accesibles utilizando identificadores de recursos uniformes (URIs). En esta arquitectura, el servidor y las aplicaciones cliente intercambian representaciones de recursos mediante el uso de una interfaz y protocolo estandarizados (API). Estos recursos se envían y reciben representados en formato JSON, lo que simplifica mucho las comunicaciones.

8.1.2 Diagramas de Paquetes

Tras haber visto un esquema general de la arquitectura del sistema de rastreo, en este apartado se describen la arquitectura y estructura de paquetes interna de cada uno de los módulos de software que lo componen.

8.1.2.1 Servidor backend

Con el fin de favorecer el desarrollo ágil de este componente, se optó por seguir una arquitectura modular. Este patrón de arquitectura consiste en dividir las responsabilidades en diversos módulos que se pueden comunicar entre sí. La arquitectura modular está enfocada a desarrollos ágiles en entornos muy dinámicos, siendo ambas las características más importantes del desarrollo de este proyecto. El objetivo de aplicar esta arquitectura al servidor backend es lograr una base de código que permita realizar cambios de forma rápida y efectiva, puesto que es muy probable que a lo largo del desarrollo o en el futuro surjan cambios que necesiten de una rápida respuesta por parte del sistema.

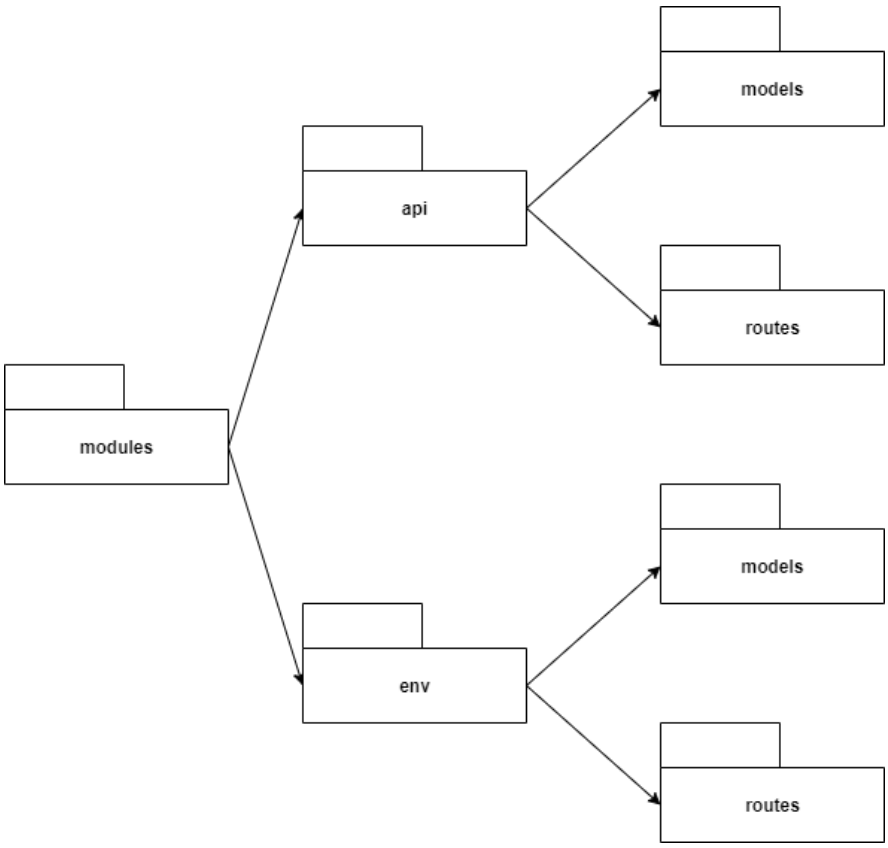

Figura 8.1. Estructura de paquetes del servidor

En el paquete raíz, junto con los archivos de configuración del servidor, se encuentra el paquete **modules**. Este paquete contiene todos los módulos del servidor, divididos en dos, **api** y **env**. El paquete **env** contiene los recursos relacionados con las variables de entorno (*environment* en inglés), y el paquete **api** el resto de los recursos del sistema de rastreo.

Dentro de cada uno de estos dos paquetes se encuentra el paquete **models**. Es contendrá los esquemas de las distintas entidades modeladas en el sistema. A cada uno de estos esquemas le corresponderá una colección en la base de datos.

Al mismo nivel que los paquetes **models** se encuentran los paquetes **routes**. En ellos se definen los recursos y métodos de la API REST. Cada uno de estos recursos o métodos será accesible desde una ruta única. Dichas rutas se agrupan en módulos según la entidad con la que trabajen, por ejemplo, los métodos para crear y obtener alertas pertenecerán al mismo

módulo, mientras que el método para añadir un local a la base de datos pertenecerá a otro distinto.

8.1.2.2 Aplicación móvil

La estructura de paquetes de la aplicación móvil se ha diseñado con la tecnología Xamarin.Forms y la arquitectura MVVM (Model-View-View Model) en mente. En el capítulo de implementación se explica en detalle cómo se ha implementado dicho patrón y sus ventajas.

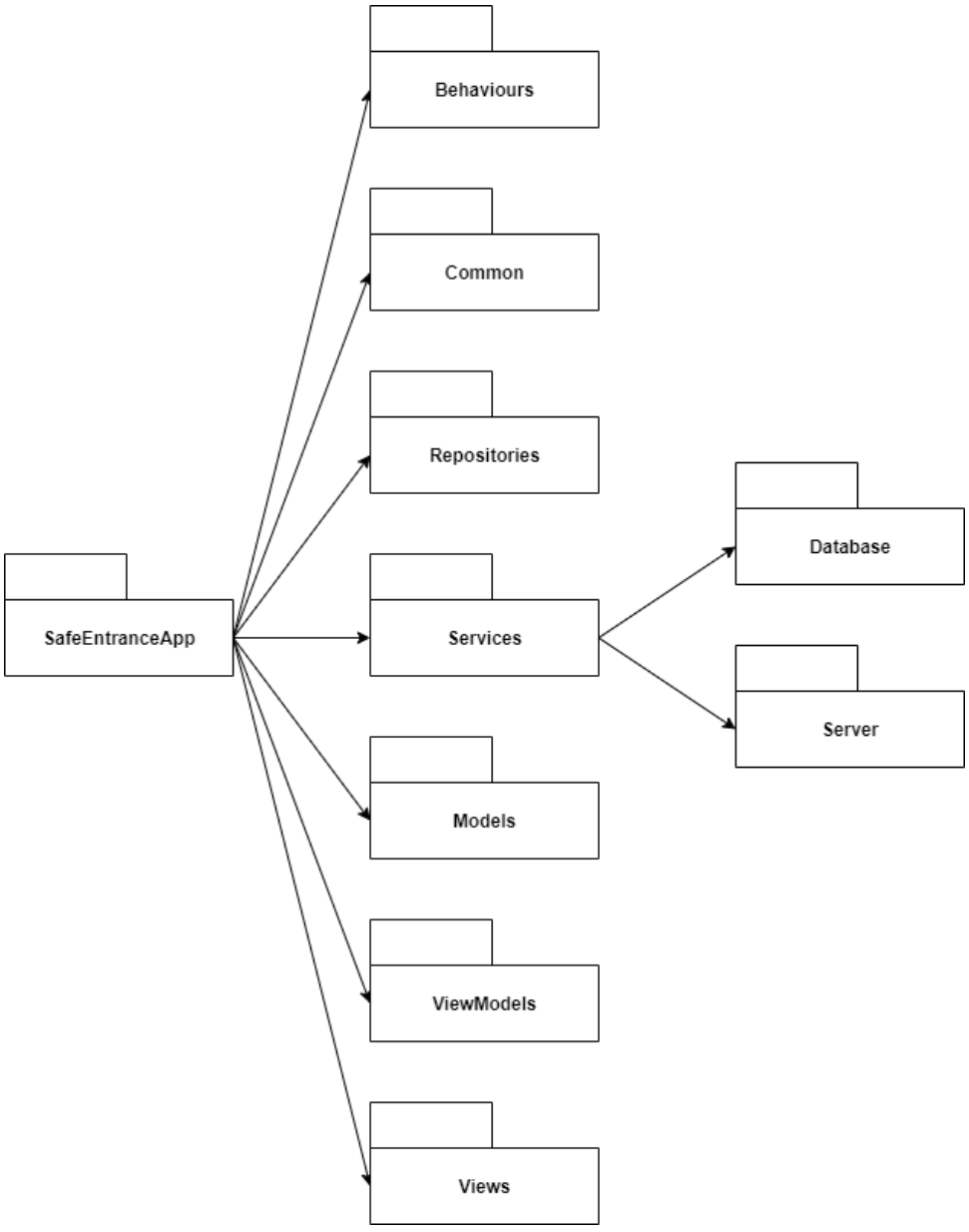

Figura 8.2. Estructura MVVM de la aplicación móvil

A continuación, se describe el objetivo de cada paquete presente en el proyecto de la aplicación móvil.

8.1.2.2.1 Behaviors

Este primer paquete contendrá clases que definirán comportamientos personalizados para los controles de la interfaz de usuario que proporciona por defecto Xamarin.Forms.

8.1.2.2.2 Common

A modo de cajón de sastre, este paquete contendrá todas aquellas clases que no encajen dentro de ninguno de los demás paquetes. Ficheros de constantes o procesadores de texto plano son ejemplos de clases que irán ubicadas en este paquete.

8.1.2.2.3 Repositories

Este tercer paquete contendrá las clases de acceso a las distintas tablas presentes en la base de datos local del dispositivo móvil.

8.1.2.2.4 Services

En la aplicación móvil existirán dos tipos de servicios, los que obtendrán datos a través de peticiones a la API REST y los que los obtendrán de los repositorios locales. Para cada uno de estos tipos existe un paquete específico.

8.1.2.2.5 Models

Este paquete contendrá las clases que definirán las entidades y modelos de dominio de la aplicación. Es la primera capa del patrón MVVM, y no debe depender de ningún otro paquete.

8.1.2.2.6 ViewModels

Este paquete es la capa intermedia del patrón MVVM. En él se encuentran las clases que contendrán la lógica de presentación de cada una de las vistas de la aplicación. Es imprescindible que estas clases estén totalmente desacopladas de las vistas.

8.1.2.2.7 Views

El último paquete contendrá las vistas de la aplicación, es decir, las interfaces de usuario.

8.1.2.3 Aplicación web

La configuración de paquetes de la aplicación web es bastante simple. En el paquete raíz existirán dos paquetes, **public** y **src**. El primero contendrá los archivos de recursos que se utilizarán en la página web, generalmente imágenes, mientras que el según tendrá dentro el código de la aplicación. Además de los paquetes descritos a continuación, el paquete **src** contendrá las vistas raíz de la página.

8.1.2.3.1 Components

Este paquete es el principal de la aplicación. En él se localizan todos los componentes que se usarán para crear las distintas vistas de la aplicación web. Dentro de él, cada componente tendrá su propio paquete con archivos JavaScript y CSS.

8.1.2.3.2 Services

Por último, el paquete este paquete contendrá las funciones o clases necesarias para comunicar la aplicación web con el servidor mediante la API REST.

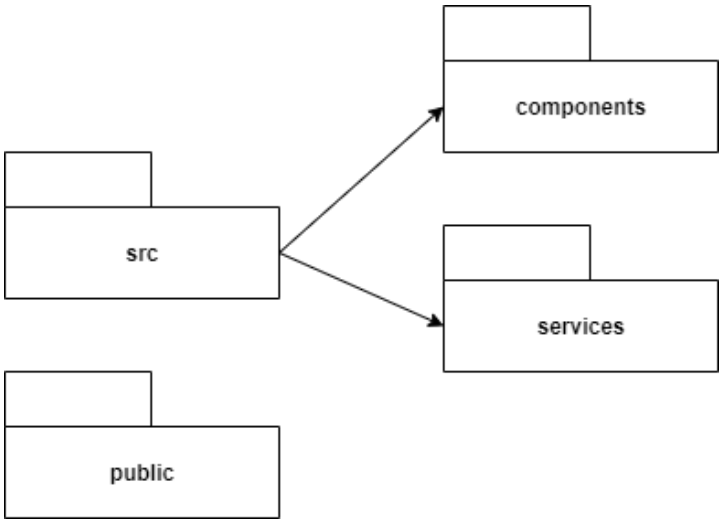

Figura 8.3. Estructura de paquetes de la aplicación web

8.2 Diseño de Clases y Módulos

Una de las consecuencias de llevar a cabo un desarrollo ágil es la falta de un diseño general de la estructura de clases del sistema antes de comenzar el desarrollo. Debido a que los requisitos, en forma de historias de usuario surgen sobre la marcha durante las iteraciones del desarrollo, el modelo y la estructura de clases se va creando progresivamente, y puede llegar a sufrir muchos cambios con respecto a su versión inicial. Por lo tanto, los diagramas que se mostrarán en esta sección no son el punto de partida del desarrollo, sino que representan su estado actual.

A continuación, se divide el diseño de clases en tres apartados, cada uno correspondiente a uno de los subsistemas de software que componen el sistema total.

8.2.1 Módulos del servidor

Como se explicó en el apartado 8.1.2.1, el servidor cuenta con una arquitectura modular que no sigue el paradigma de orientación a objetos tradicional. Como consecuencia de esto, en lugar de clases en el servidor habrá módulos, que actuarán como nodos de un mismo sistema. El módulo llamado **index** es la raíz de la estructura. Este módulo contiene la instancia del servidor, y su función será configurarlo e inicializar y enlazar él el resto de los módulos.

A continuación, por tanto, se muestra el contenido de cada uno de estos módulos, indicando sus atributos, operaciones y un breve resumen de sus responsabilidades.

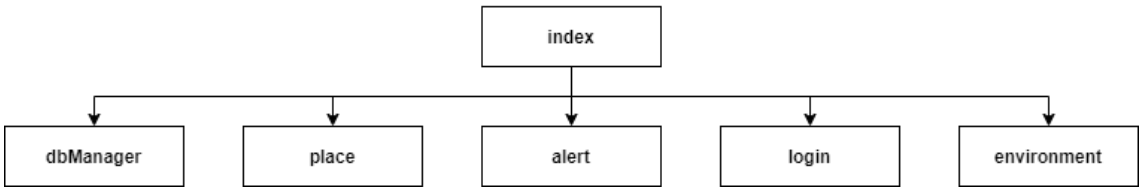

Figura 8.4. Estructura de módulos del servidor

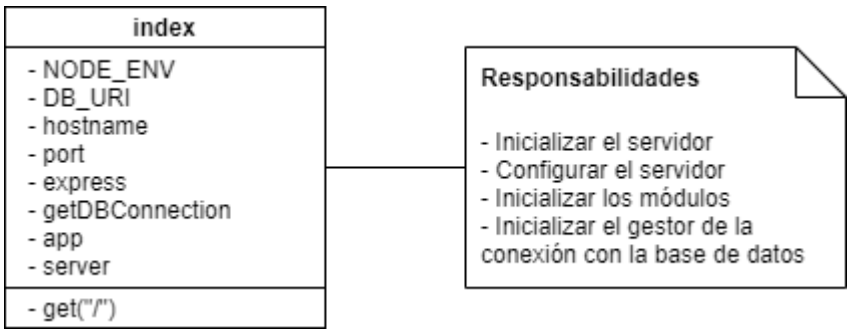

Figura 8.5. Módulo raíz del servidor

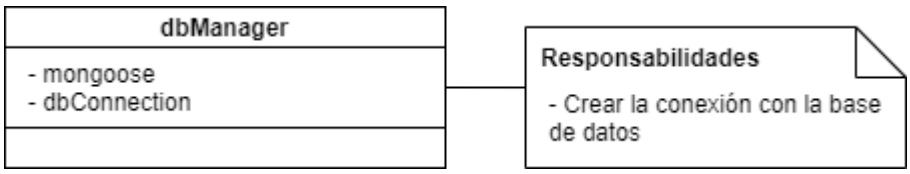

Figura 8.6. Módulo de conexión con la base de datos

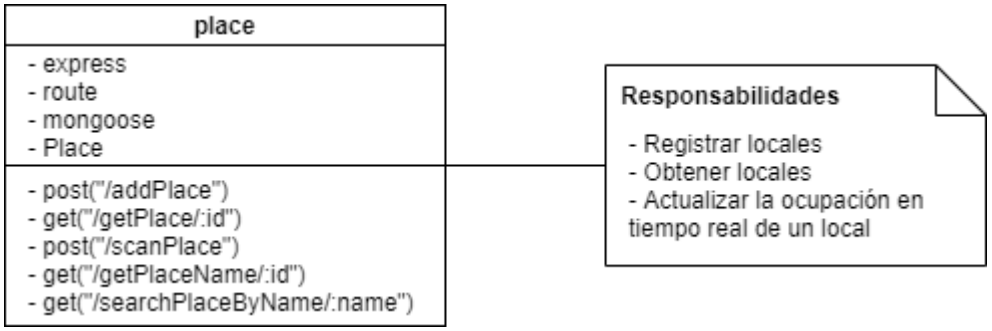

Figura 8.7. Módulo con las rutas asociadas a la gestión de locales

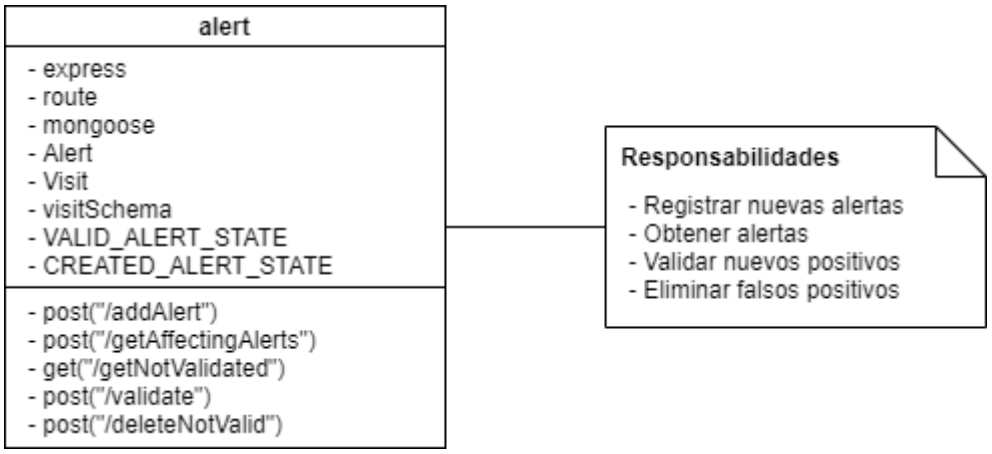

Figura 8.8. Módulo con las rutas asociadas a la gestión de alertas

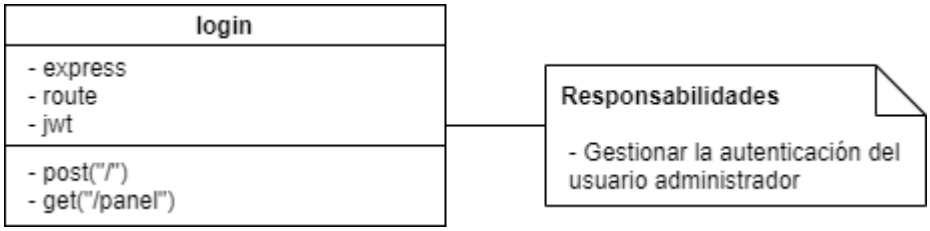

Figura 8.9. Módulo con las rutas asociadas a la autenticación del administrador

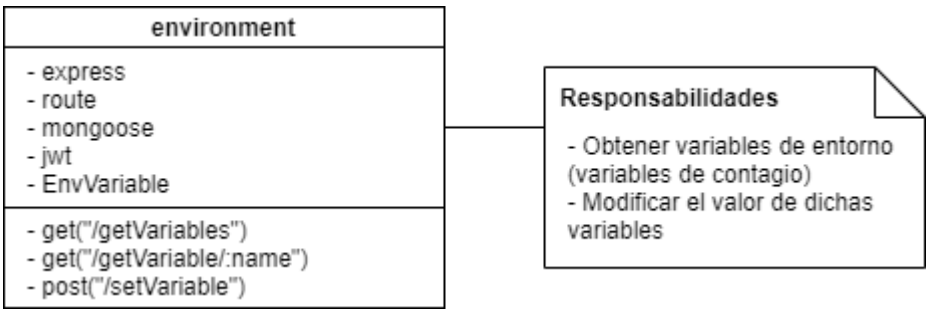

Figura 8.10. Módulo con las rutas asociadas a la gestión de variables de entorno

A continuación, la figura 8.11 muestra el modelo de datos del servidor.

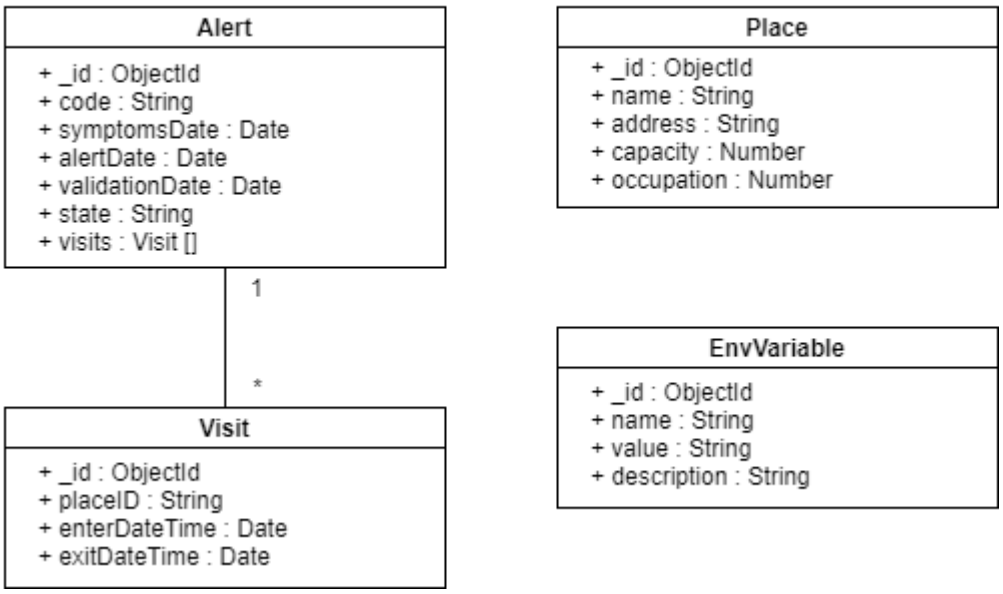

Figura 8.11. Diagrama de clases del modelo del servidor

8.2.2 Diagrama de clases de la aplicación web

La aplicación web se trata únicamente de una interfaz de usuario que interactúa con el servidor a través de la API REST. Esta interfaz, por lo tanto, carecerá de lógica de negocio y únicamente contará con vistas y controles que permitan al usuario interactuar con el sistema.

A continuación, se muestra la jerarquía de estos componentes y como están organizados según las relaciones de composición y agregación existentes entre ellos.

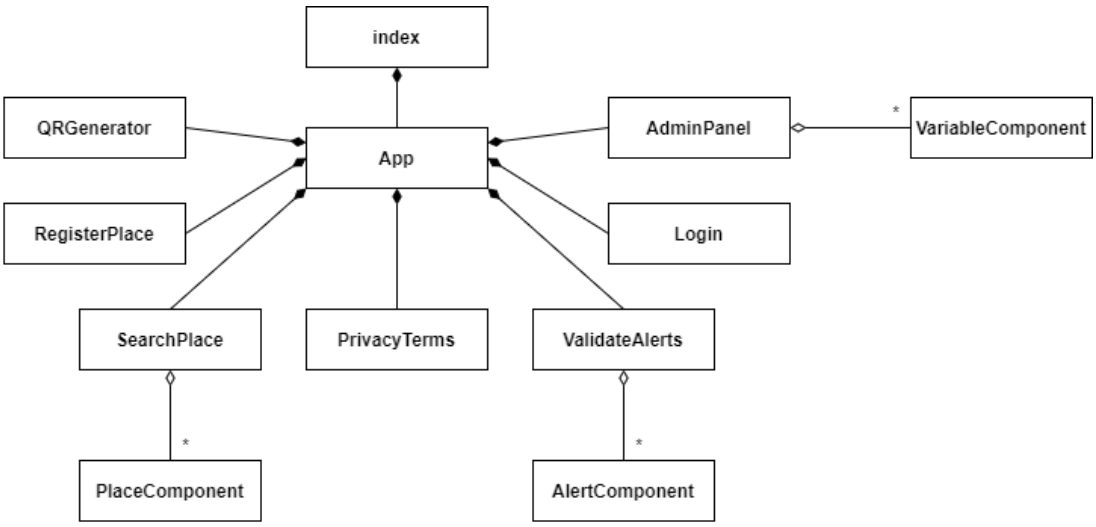

Figura 8.12. Diagrama de componentes de los componentes de la aplicación web

8.2.3 Clases de la aplicación móvil

En este apartado se describe la estructura de clases de la aplicación móvil. Tan sólo se mostrarán los diagramas de aquellos paquetes que son importantes de cara a entender el diseño de la aplicación, es decir, los modelos, vistas, vistas modelo y los paquetes de acceso a datos.

8.2.3.1 Modelos

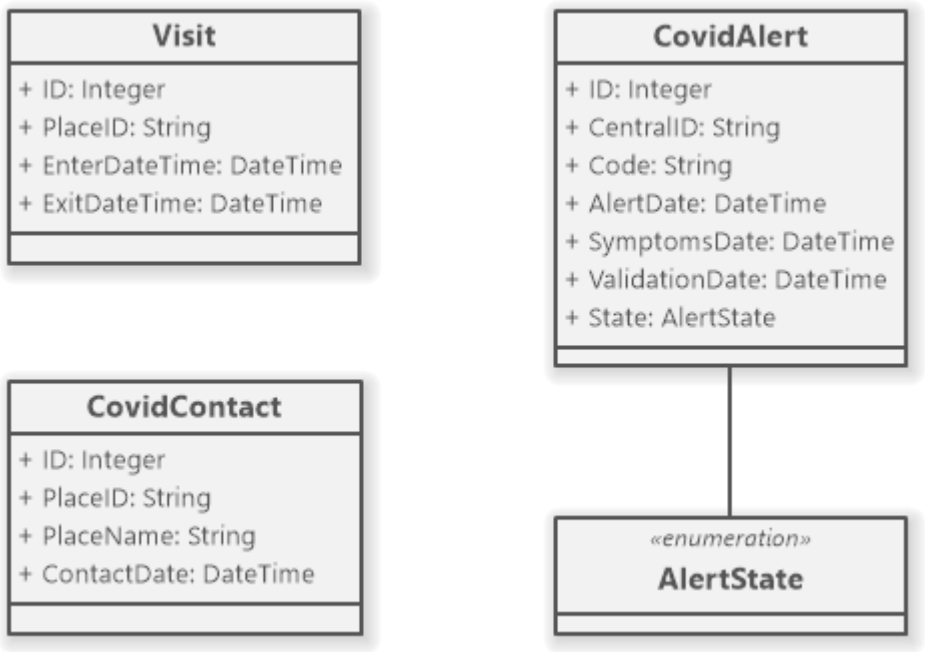

Figura 8.13. Diagrama de clases del modelo de la aplicación móvil

8.2.3.2 Vistas y vistas modelo

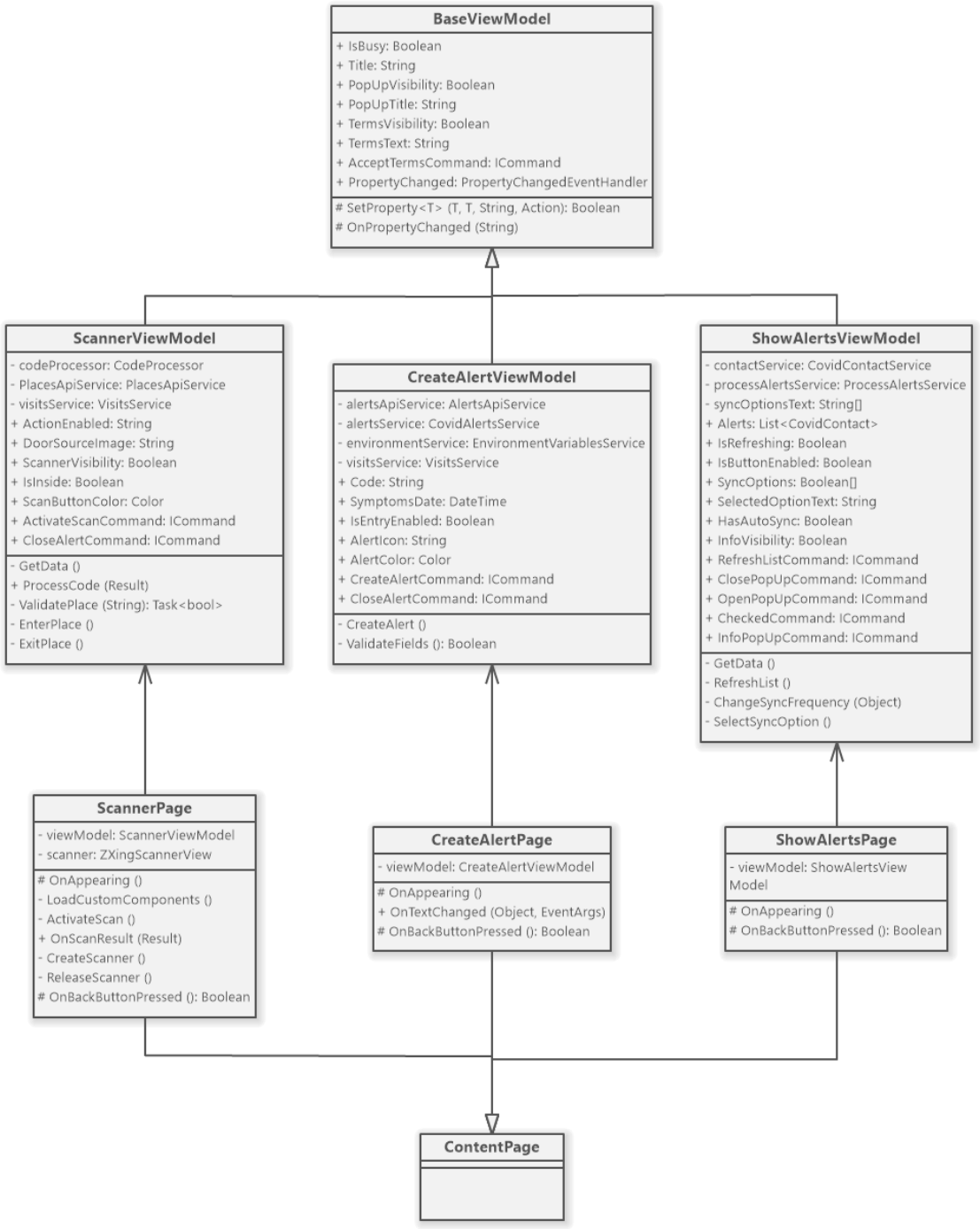

Figura 8.14. Diagrama de clases de las vistas y vistas modelo

8.2.3.3 Servicios y repositorios de acceso a la base de datos local

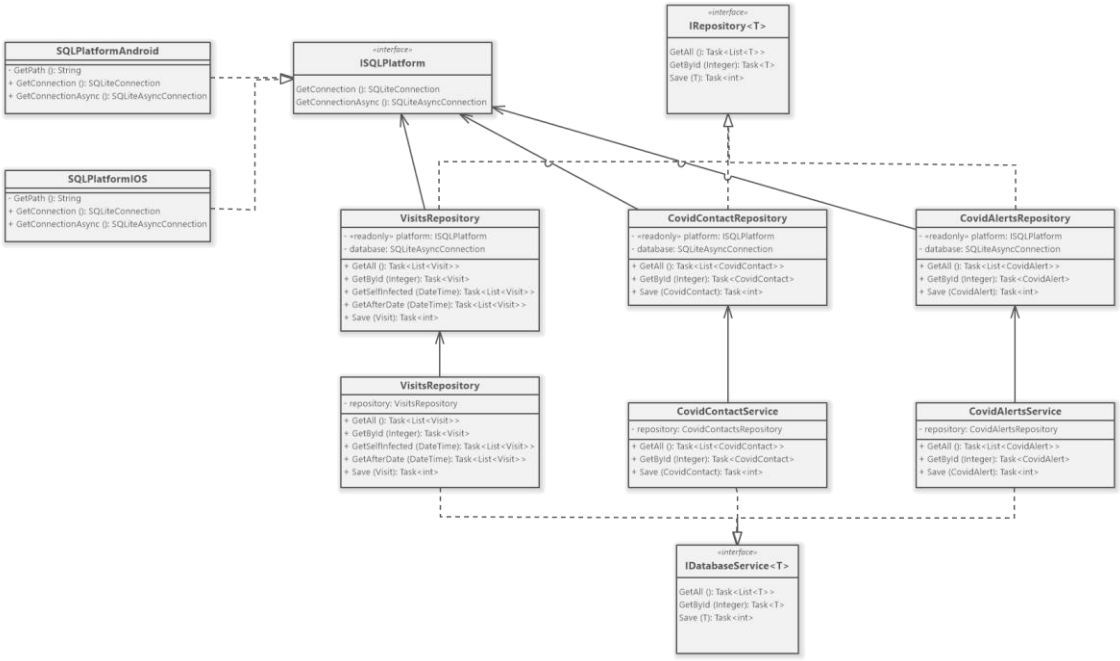

Figura 8.15. Diagrama de clases de acceso a la base de datos local

8.3 Diagramas de Actividades

En este apartado se describen los procesos de creación de alertas y obtención de posibles contactos con positivos mediante diagramas de actividad.

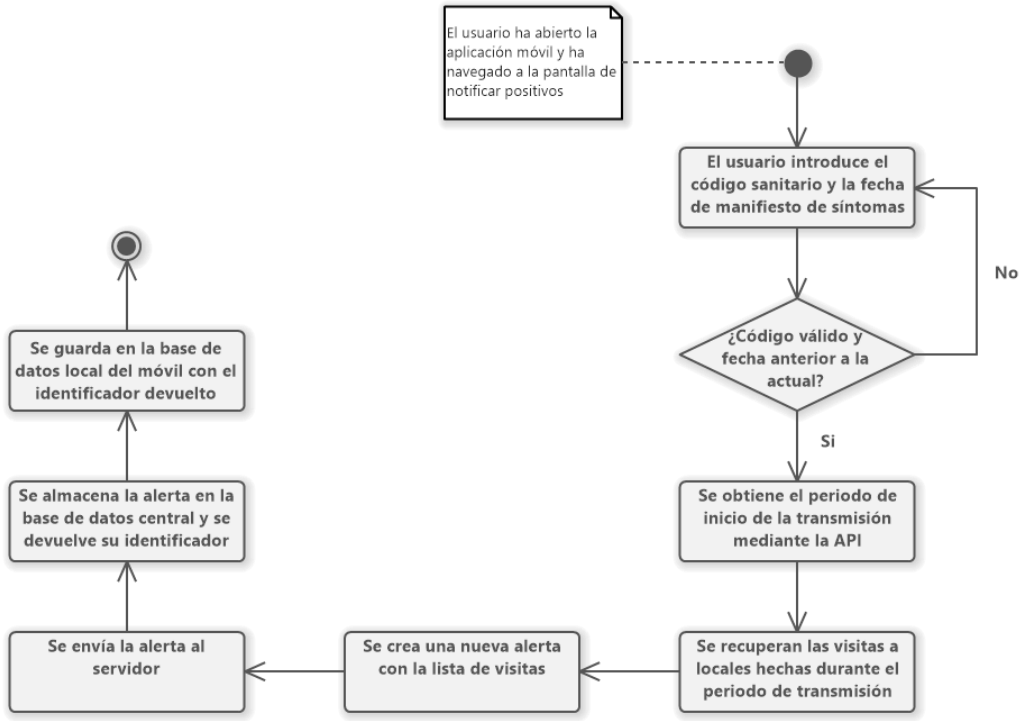

Figura 8.16. Diagrama de actividad del proceso de creación de alertas

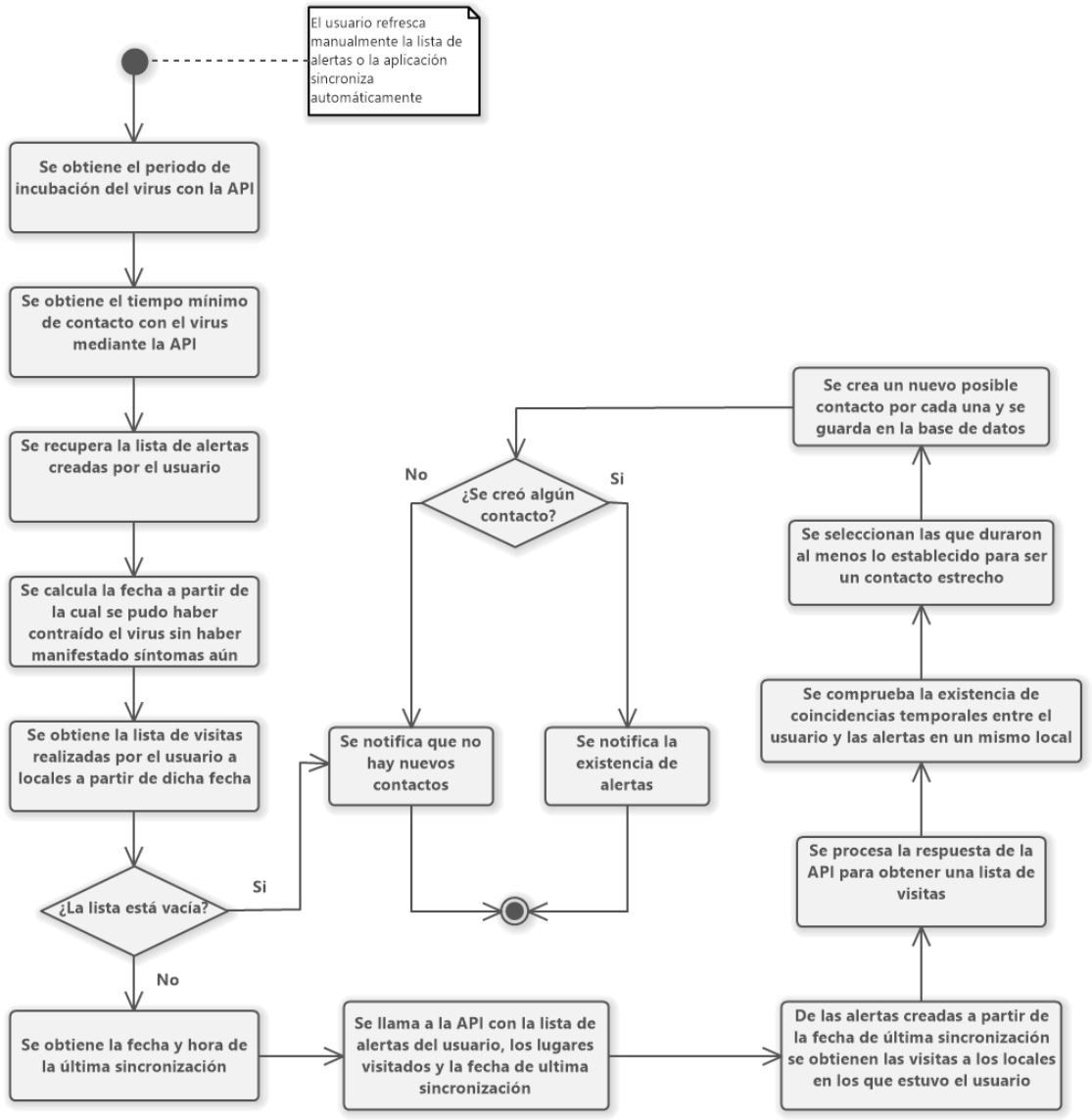

Figura 8.17. Diagrama de actividad del proceso de obtención de alertas

8.4 Diseño de la Base de Datos

En este apartado se describe el diseño de las bases de datos existentes en el sistema de rastreo. Tal y como se explicó en el capítulo 6, para implantar este sistema será necesario utilizar una base de datos central y a su vez las bases de datos locales de cada dispositivo que utilice la aplicación móvil. Puesto que la función de las bases de datos locales es completamente distinta a la de la base de datos central, en los próximos apartados se describirán ambas por separado.

8.4.1 Descripción de los SGBDs usados

8.4.1.1 Base de datos central

A la hora de elegir un sistema de gestión de bases de datos para el servidor central se buscó que este fuera fácilmente integrable con Node.JS, tecnología con la que se creará dicho servidor. Además, debido al enfoque ágil del proyecto y a las evoluciones y cambios que puede sufrir el dominio de este sistema, se optó por utilizar una base de datos NoSQL orientada a documentos. Estas bases de datos aportan una gran flexibilidad al desarrollo, convirtiéndolas en ideales para un desarrollo ágil.

De entre todas las opciones existentes en el mercado rápidamente se desmarcaron dos: Firebase Firestore y MongoDB Atlas. La primera, perteneciente a la solución Firebase de Google, se descartó debido a, precisamente, pertenecer a Google. El principal motivo de este descarte fue tratar de alejar este sistema de rastreo de las controversias que existieron alrededor de uno de sus predecesores, Radar Covid. Radar Covid hizo uso de servicios pertenecientes a la solución Firebase para realizar tareas de seguimiento del funcionamiento de la aplicación durante su etapa de desarrollo. Sin embargo, pese a que estos servicios ya no se usaban en su puesta en producción, en el código de Radar Covid aún había trazas de haber usado Firebase. Debido a la conocida intrusiva política de tratamiento de datos de Google, se generó mucho desconcierto entre la población con pocos conocimientos técnicos sobre software (la amplia mayoría de ciudadanos españoles), y fue una piedra más en el camino para el inalcanzado éxito de esta aplicación. Por tanto, para alejar de la población la idea de que el sistema de rastreo desarrollado en este proyecto no es fiable en cuanto al tratamiento de datos, se decidió usar MongoDB Atlas.

MongoDB Atlas ofrece servicios de bases de datos orientadas a documentos en la nube que garantizan una gran escalabilidad, disponibilidad y el cumplimiento de los estándares de privacidad y seguridad de datos más exigentes. Gracias a ser un servicio en la nube, MongoDB Atlas permite ajustar el tamaño, la capacidad de memoria, o la capacidad de almacenamiento de sus clusters de forma dinámica sin necesidad de parar el servicio. Gracias a esto se consigue ajustar los presupuestos del proyecto, pagando únicamente por las capacidades que se necesiten en cada momento, y da la posibilidad de extender el sistema de rastreo a otros países que así lo desearan sin perder velocidad de respuesta gracias a contar servicios en la nube en más de 75 regiones.

8.4.1.2 Base de datos de los dispositivos móviles

A la hora de gestionar y almacenar las visitas a locales o los posibles contactos con positivos es necesario el uso de almacenamiento persistente en los dispositivos móviles. Para ello se utilizarán las bases de datos SQL embebidas que proporcionan los sistemas operativos Android y iOS gracias al sistema de gestión de bases de datos SQLite.

SQLite es el motor de bases de datos más usado del mundo debido a su pequeño tamaño, velocidad y a que es un sistema autocontenido. Está embebido en la gran mayoría de dispositivos móviles y ordenadores, y es un sistema estable, multiplataforma y retrocompatible.

8.4.2 Integración de los SGBDs en el sistema

8.4.2.1 Base de datos central

Pese a que MongoDB cuenta con sus propios controladores para aplicaciones basadas en Node.JS, a la hora de integrarlo con el servidor se ha optado por usar la librería Mongoose. Mongoose es un ODM (*Object-Document Mapper*) para Node.JS que permite realizar transformaciones entre objetos en el código y documentos en la base de datos. Gracias a esto, implementar lógica de negocio y validaciones se vuelve una tarea mucho más directa, agilizando satisfactoriamente el desarrollo.

Para utilizar MongoDB con Mongoose en Node.JS el habrá que agregar ambas dependencias (*mongodb* y *mongoose*) al fichero *package.json* del servidor. Tras esto, mediante la función *connect* de la librería Mongoose y la URL de la base de datos MongoDB Atlas, se establece una conexión entre el servidor y dicha base de datos a través de la cuál se producirán los intercambios de datos. Por cada modelo creado (*Schema,* esquema en español), Mongoose creará una colección de documentos en la base de datos, a la que pertenecerán todas las instancias de dicho esquema. De esta manera, una vez definidas las entidades del modelo en dichos esquemas, Mongoose evitará que sea necesario diseñar aparte la base de datos.

8.4.2.2 Base de datos de los dispositivos móviles

Android y iOS, debido a sus grandes diferencias en cuanto a implementación, utilizan versiones distintas de la API de SQLite. Mientras que Android utiliza la implementación para Java, iOS hace uso de la API de SQLite para C. Por este motivo, para desarrollar la aplicación móvil manteniendo una única base de código encargada de la lógica de negocio y el acceso a base de datos, es necesario el uso de una librería que aporte un nivel mayor de abstracción en cuanto al acceso a dicha base de datos, y que por debajo adapte dichas operaciones en función del dispositivo que ejecute la aplicación.

De esta función se encargará la librería SQLite para Xamarin.Forms. Para integrarla en el proyecto será necesario instalar el paquete NuGet “SQLite” e implementar la creación de la base de datos para ambas plataformas mediante una interfaz común y el servicio de inyección de dependencias de Xamarin.Forms, que permitirán acceder a ellas indistintamente de si es un

dispositivo Android o iOS. Además de unificar el acceso a las bases de datos, esta librería actúa de ORM (*Object-Relational Mapper*), lo que permite, al igual que Mongoose en Node.JS, que no sea necesario diseñar la base de datos una vez estén definidas las entidades, puesto que para cada clase del modelo se creará una tabla en la base de datos.

8.5 Diseño de la Interfaz

En esta sección se describirán detalladamente las interfaces de usuario diseñadas a partir de los bocetos creados durante el análisis de cada una de las funcionalidades y pantallas a desarrollar. Además de mostrar los diseños de las pantallas, se detallarán los cambios que han sufrido con respecto a sus bocetos o como se han implementado ciertas características mencionadas en ellos. En cuanto a las pantallas de la aplicación móvil, se mostrará su formato en dispositivos Android, dado que para el desarrollo del proyecto no se ha contado con un dispositivo Mac OS en el que poder compilar la versión de iOS.

8.5.1 Interfaces de usuario de la aplicación web

8.5.1.1 Formulario de registro de locales

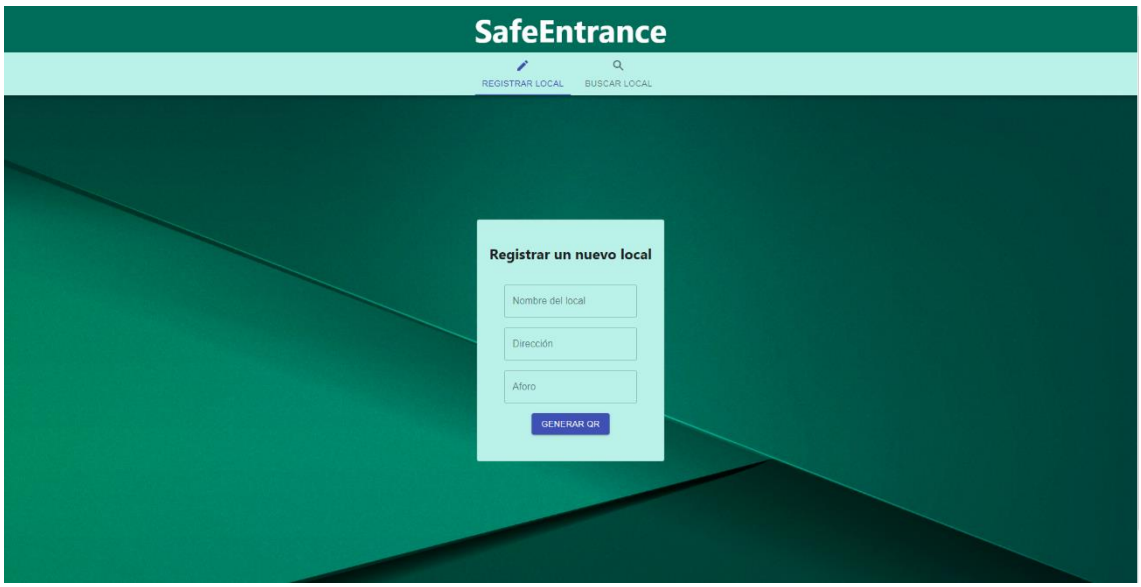

Figura 8.18. Formulario de registro de locales

Tal y como se bocetó, la página web cuenta con una cabecera y un menú de navegación en la parte superior. En el centro de la pantalla está el formulario de registro de locales con los campos determinados en las historias de usuario. En caso de que algún campo sea incorrecto, se mostrará como en la imagen 8.19.

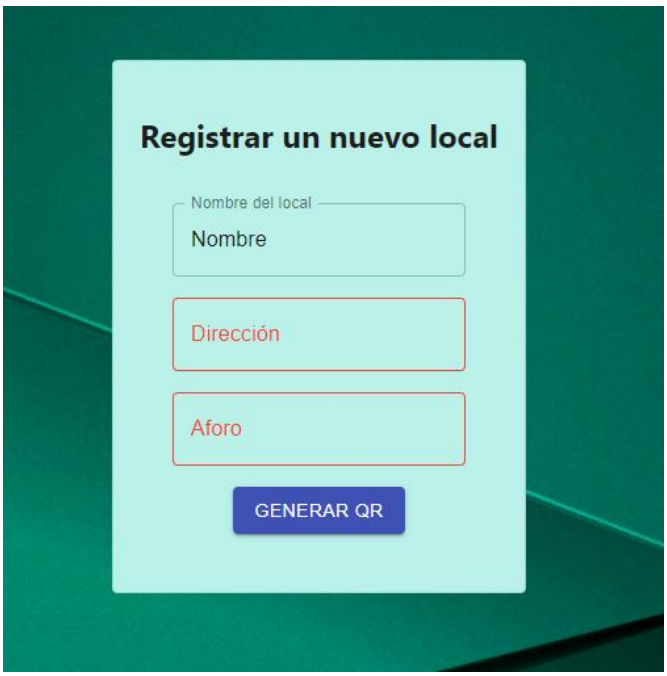

Figura 8.19. Formulario de registro de locales con campos incorrectos

8.5.1.2 Pantalla de búsqueda de locales

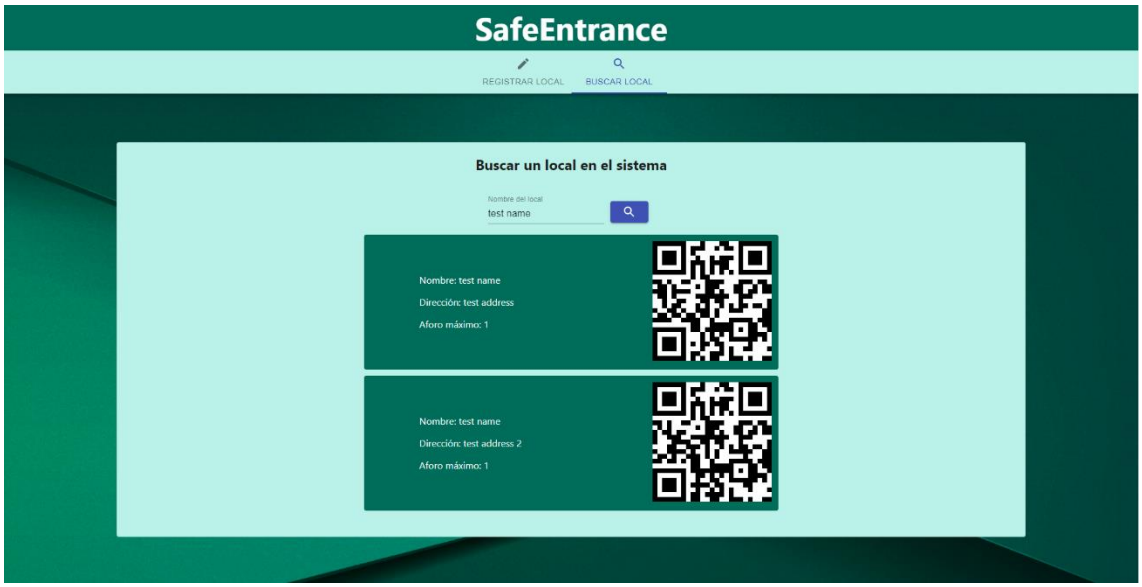

Figura 8.20. Formulario de registro de locales con campos incorrectos

La pantalla de búsqueda de locales finalmente se diseñó de acuerdo con el boceto. Sin embargo, se optó por esconder la barra de desplazamiento vertical en la lista de locales, lo que mejora el aspecto visual sin quitarle la funcionalidad de realizar dicho desplazamiento cuando la lista de locales con un determinado nombre no cabe dentro de la pantalla. En las tarjetas de los locales, como se había previsto, se incluyen el nombre, la dirección, el aforo máximo y el QR.

8.5.1.3 Pantalla de descarga del código QR

Figura 8.21. Pantalla de descarga del QR

En esta pantalla, consecuencia de haber registrado correctamente el local en el sistema, se muestra el QR generado junto con dos botones para descargarlo y volver atrás. En caso de que se introduzca una dirección falsa con un identificador que no exista, se mostrará un mensaje de error y el botón de volver atrás.

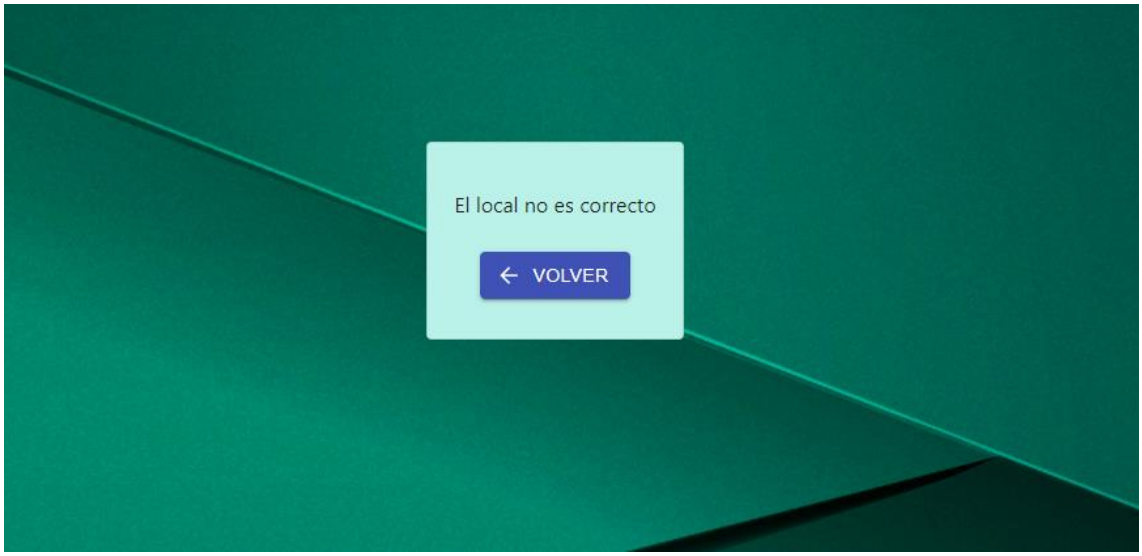

Figura 8.22. Pantalla de descarga del QR con el parámetro incorrecto

8.5.1.4 Formulario de autenticación del administrador

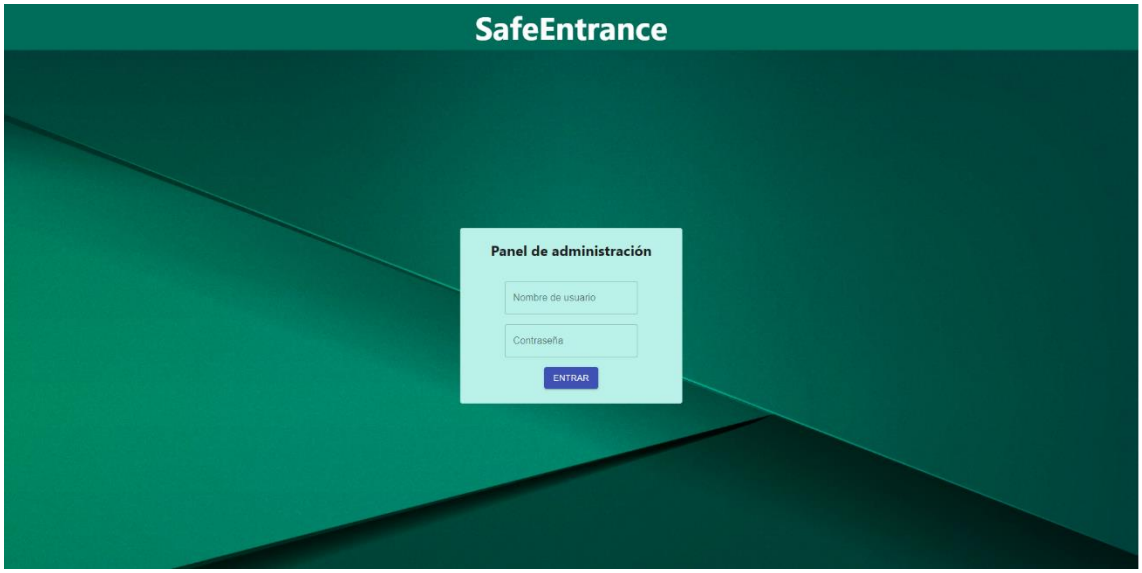

Figura 8.23. Formulario de inicio de sesión de administrador

Este formulario cuenta con dos campos, como se definió en las historias de usuario. En caso de que se introduzcan unas credenciales incorrectas, se mostrarán los campos en rojo y aparecerá un mensaje durante cinco segundos.

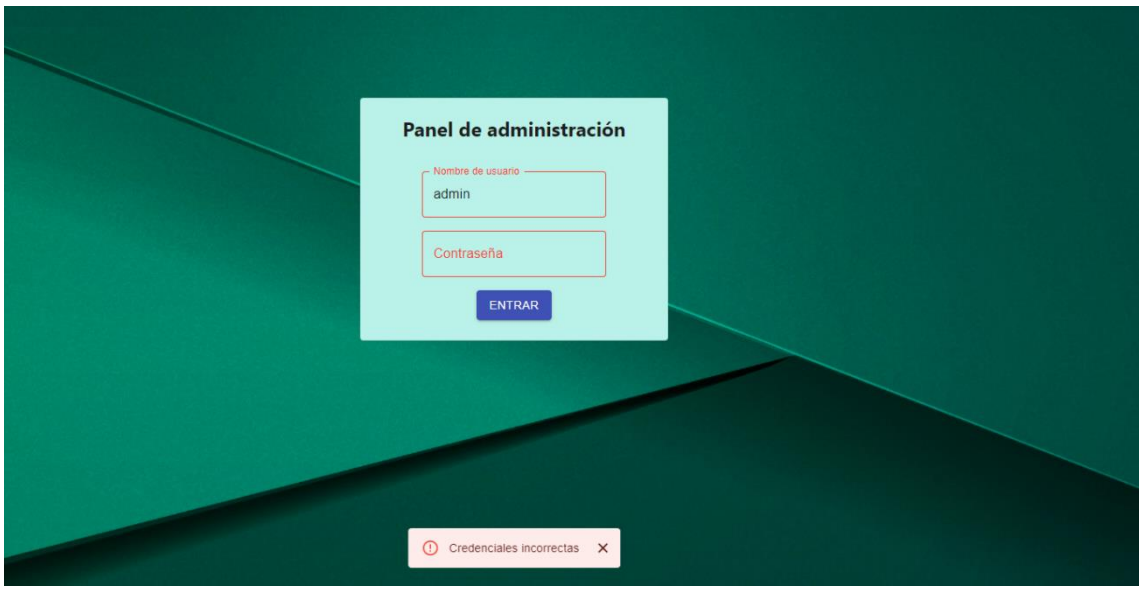

Figura 8.24. Fallo al iniciar de sesión de administrador

8.5.1.5 Panel de control de variables de contagio

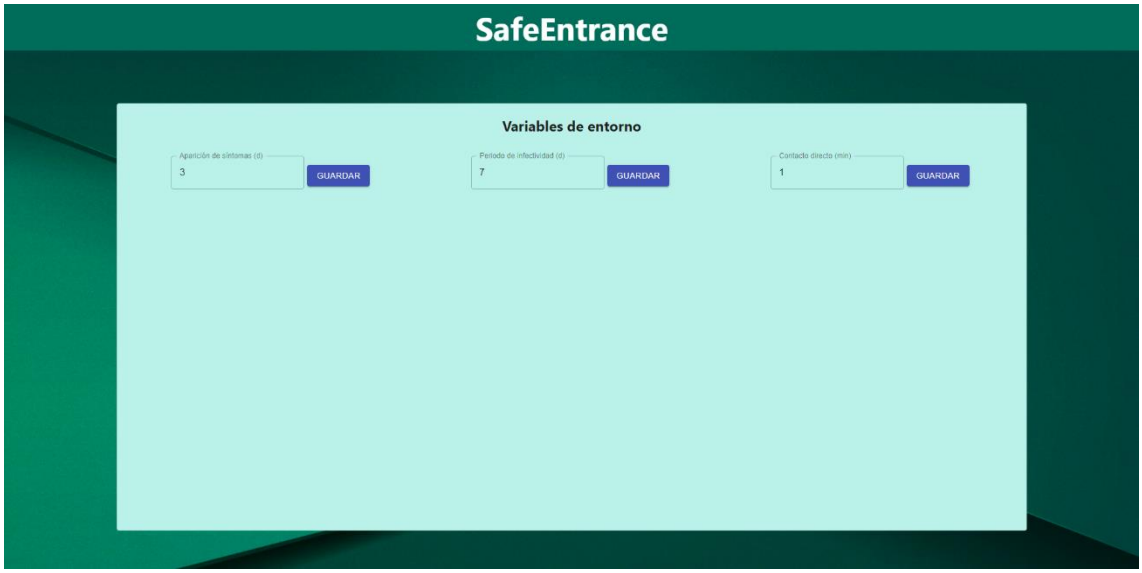

Figura 8.25. Panel de control de variables de contagio

Tal y como se bocetó, el panel de control tiene un campo para cada variable, acompañado de un botón para guardar su valor. Para dar confirmación al usuario de que actualizó el valor correctamente, se mostrará un mensaje similar al de error de la pantalla previa, pero en este caso de confirmación.

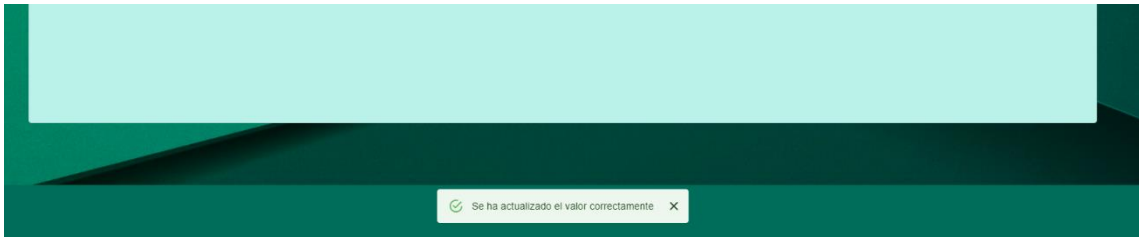

Figura 8.26. Mensaje de confirmación de actualización de variable

8.5.1.6 Lista de validación de nuevos positivos

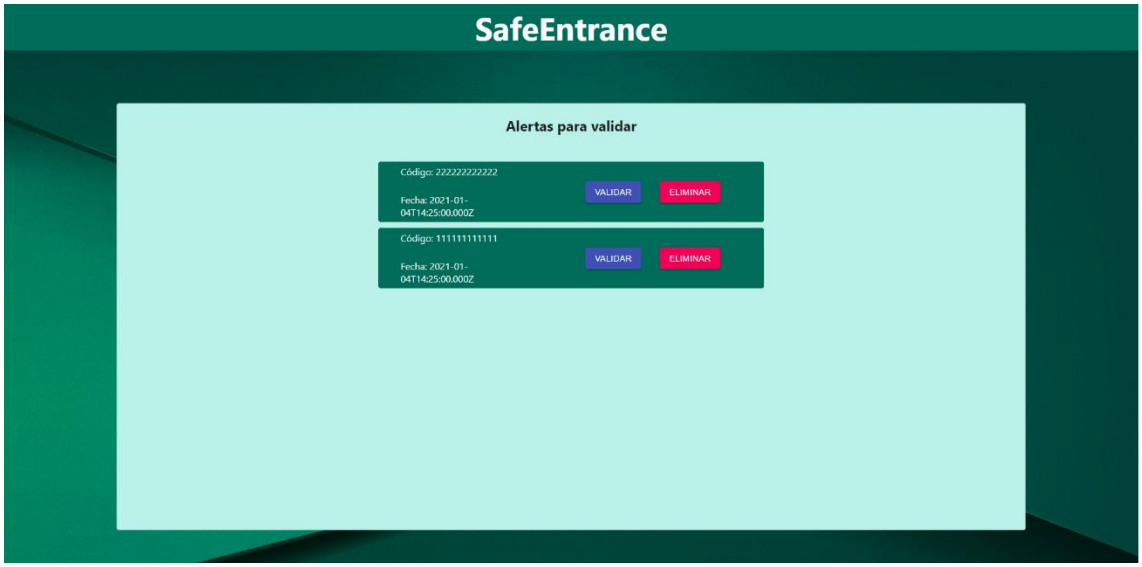

Figura 8.27. Lista de nuevos positivos pendientes de validación

En esta pantalla, de acuerdo con lo establecido en las historias de usuario, se muestran los códigos y fechas de los nuevos positivos reportados, junto con los botones de validar y eliminar.

8.5.1.7 Pantalla de aviso legal y política de privacidad

Si el usuario se desplaza hacia abajo en cualquiera de las pantallas de la aplicación web se encontrará con un pie de página que contiene un enlace. Dicho enlace lleva a la página en la que se explica la política de privacidad. En las siguientes imágenes se muestran dicho pie de página y la pantalla de aviso legal.

Figura 8.28. Pie de página de la aplicación web

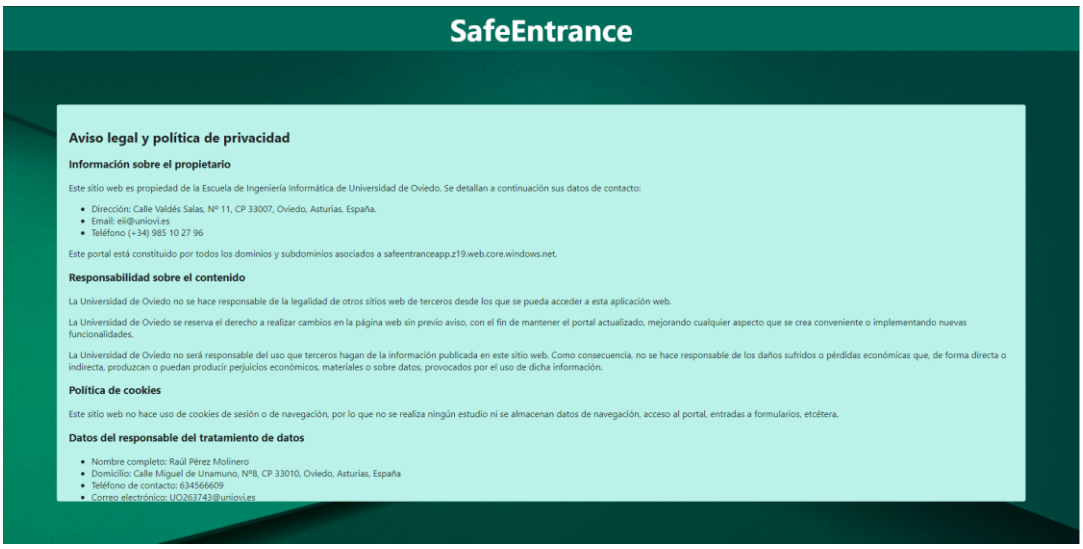

Figura 8.29. Pantalla de aviso legal y política de privacidad

8.5.2 Interfaces de usuario de la aplicación móvil

8.5.2.1 Pantalla de bienvenida

Figura 8.30. Pantalla de bienvenida

Tal y como se describe en las historias de usuario, se diseñó una pantalla de bienvenida que muestra el icono de la aplicación en el centro. Vemos también como este icono representa una entrada libre de virus.

8.5.2.2 Pantalla del lector de códigos QR

Figura 8.31. Pantalla del lector QR con él desactivado y el usuario fuera de un local

En esta primera imagen de la pantalla del lector QR vemos como se implementó una barra inferior de navegación tal y como se diseñó en los bocetos, con sus iconos y títulos. En cuanto al escáner de códigos QR, en esta imagen se encuentra desactivado, y la aplicación le indica al usuario que para entrar a un local debe escanear un código. Pulsando en el botón amarillo, se activará el escáner, que ocupará el lugar en el que se encuentra la imagen de la puerta, tal y como se puede apreciar en la figura 8.32.

Figura 8.32. Pantalla del lector QR con él activado

En esta imagen se muestra cómo quedaría la pantalla cuando se activa el lector QR. En el espacio previamente ocupado por la imagen de la puerta aparece el lector QR, con una barra horizontal que ayudará al usuario a centrar el dispositivo sobre el código. Pulsando de nuevo el botón amarillo se desactivaría el escáner. En caso de escanear el QR de un local, se desactivará el escáner y se mostrará que el usuario ha entrado al local.

Figura 8.33. Pantalla del lector QR tras entrar a un local

Tras escanear el QR de un local, la aplicación mostrará al usuario la puerta abierta y el texto y el color del botón cambiarán con el fin de hacerle saber que se encuentra dentro de un local. El funcionamiento de la pantalla en este estado es idéntico a cuando el usuario está fuera.

8.5.2.3 Pantalla para reportar positivos

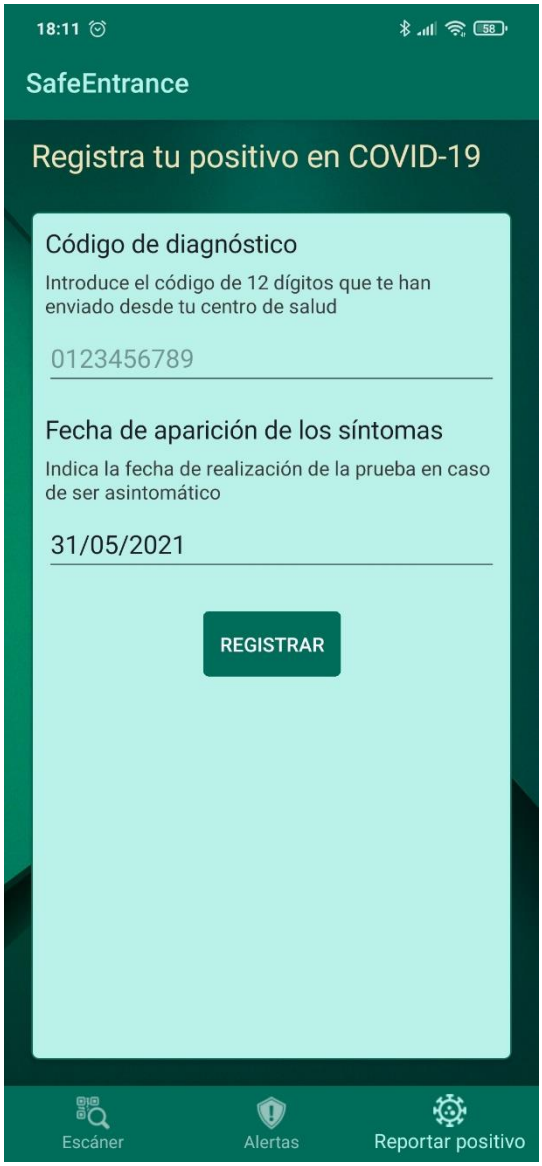

Figura 8.34. Pantalla para reportar positivos

En esta pantalla se encuentra el formulario para reportar positivos. En lugar de dividir la entrada del código sanitario en 12, finalmente se decidió que fuera un único campo con control de longitud. Además del nombre de los campos, se incluyó una breve descripción para que el usuario sepa que se le pide introducir. En caso de que algún campo sea incorrecto se mostrará un mensaje de error con el formato que se describirá más adelante, en el apartado de ventanas flotantes.

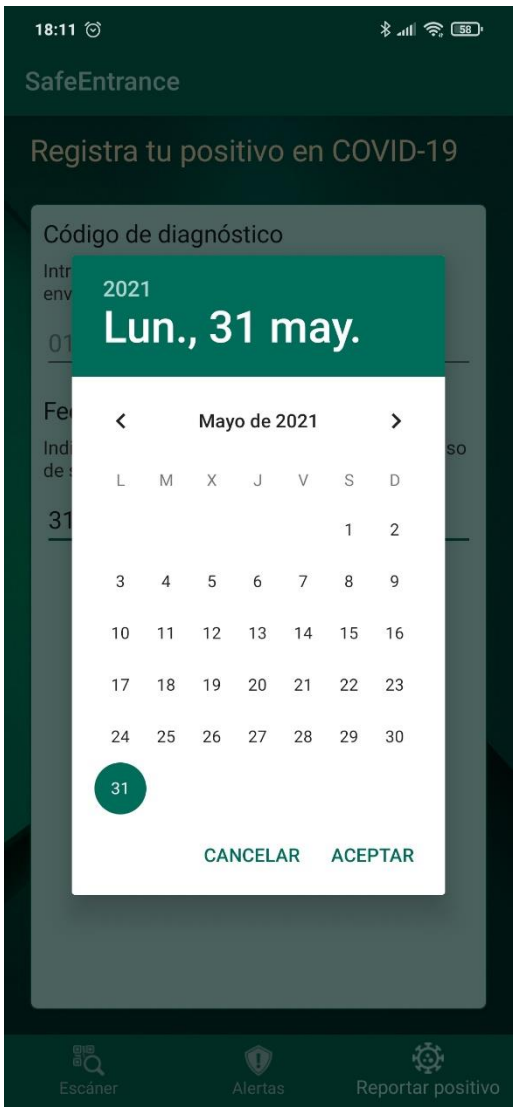

Figura 8.35. Calendario de selección de fechas

Para seleccionar una fecha, la aplicación mostrará un calendario, tal y como se especificó en el análisis de esta funcionalidad. El calendario tendrá también los colores del tema de la aplicación.

8.5.2.4 Listado de posibles contactos con positivos

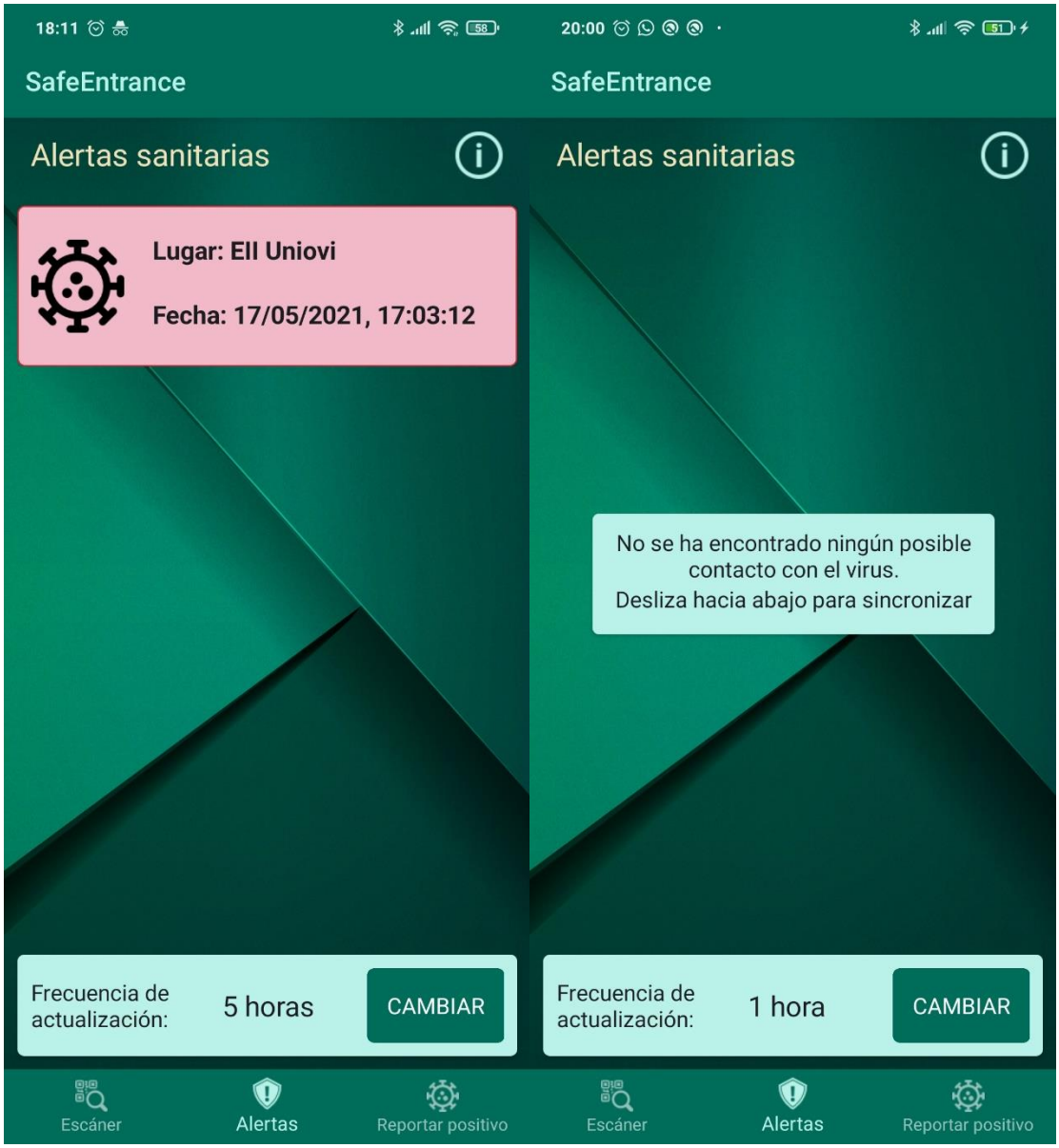

Figura 8.36. Listado de posibles contactos con positivos

La pantalla para listar alertas tiene un diseño que se adecúa en casi todo al boceto realizado previamente. En caso de existir algún posible contacto, se muestra en una tarjeta roja como la que aparece en la imagen de la izquierda. Además, en la parte inferior se muestra, en los dispositivos Android, la sección de selección de frecuencia de sincronización. Como se puede comprobar en la imagen de la derecha, en caso de no haber posibles contactos con positivos, se muestra un mensaje.

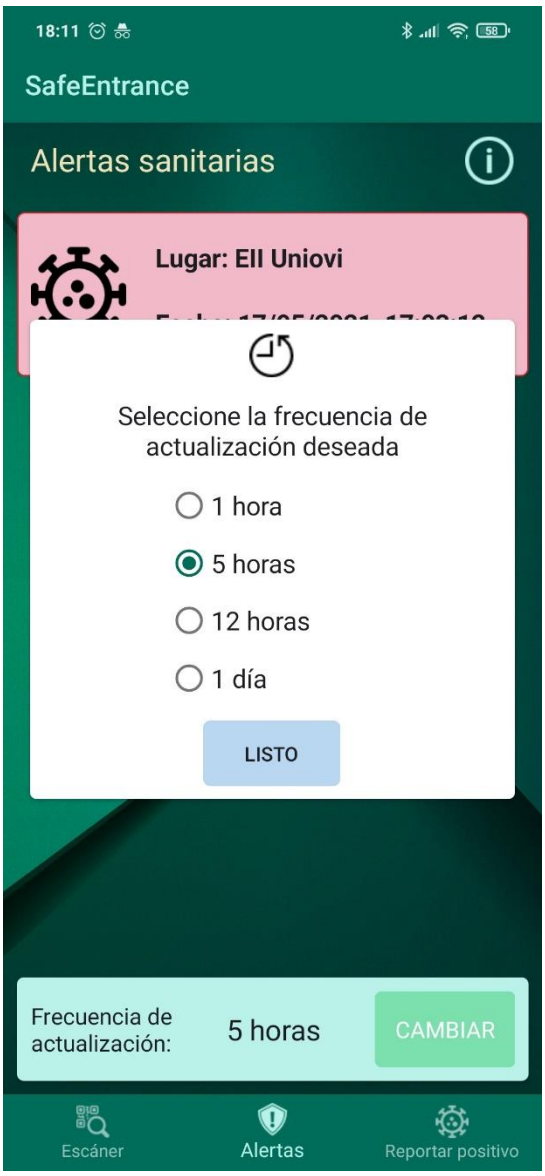

Figura 8.37. Ventana de selección de frecuencia de sincronización

Como muestra la imagen 8.37, la ventana de selección de sincronización permite al usuario elegir entre 4 posibles frecuencias. El formato de esta ventana es el que se describió en las historias de usuario. Además, como se explicará en el próximo apartado, su estilo es común al resto de ventanas flotantes.

8.5.2.5 Ventanas flotantes

Para que el aspecto visual de la aplicación sea consistente, todas las ventanas flotantes tienen el mismo diseño. Por ventanas flotantes se entienden aquellas que pasan a un primer plano en la pantalla sin ocupar la totalidad de esta, focalizando la atención del usuario en ellas sin que pueda interactuar con el resto de los elementos de la vista actual hasta que cierre dicha ventana flotante. En la aplicación hay tres tipos de ventanas flotantes: errores, ventanas informativas y la ventana de selección de frecuencia. A continuación, muestran ejemplos de los dos tipos que aún no se han enseñado, los errores y las ventanas informativas.

Figura 8.38. Ventana de error y ventana de términos y condiciones

En la imagen de la izquierda se muestra un mensaje de error producido al escanear un código incorrecto, mientras que a la derecha aparece la ventana de términos y condiciones de la aplicación. Como se puede observar, el estilo es común en ambas, y están compuestas por una tarjeta blanca con icono en la parte superior, texto en el medio y un botón de aceptar al final. En el caso de los términos y condiciones, el texto es desplazable verticalmente, debido a su gran tamaño.

8.6 Especificación Técnica del Plan de Pruebas

En esta sección se presenta el diseño del plan de pruebas del sistema. Este plan de pruebas se fue desarrollando a lo largo de las 4 iteraciones, a medida que se fueron desarrollando e implementando las funcionalidades asociadas a cada historia de usuario. Las pruebas aquí expuestas han servido para que historias de usuario en revisión pudieran cerrarse. A su vez, todas estas pruebas deberán ejecutarse cada vez que se pretenda desplegar una nueva versión de alguno de los módulos que componen el sistema, con el fin de garantizar que la nueva versión cumple con todos los criterios de aceptación definidos en las historias de usuario y que no se introducen fallos en ninguna de las funcionalidades que no deberían haber sufrido cambios.

8.6.1 Pruebas Unitarias

Las pruebas unitarias en este sistema estarán enfocadas a probar el comportamiento de la API REST gestionada por el servidor backend, puesto que esta será la que realice la mayor parte de la lógica de negocio. Debido a que estas pruebas están enfocadas a la lógica de negocio, carece de sentido diseñar pruebas unitarias para la aplicación web.

Para la realización de estas pruebas unitarias en el servidor se habilitará un entorno de pruebas. Dicho entorno se conectará a una base de datos distinta a la que se utilizará en producción, con el fin de poder realizar todo tipo de operaciones sobre ella sin afectar al sistema real. Además de realizarse sobre este entorno, las pruebas unitarias se automatizarán e integrarán dentro del sistema de CI/CD del servidor. De esta manera, se ejecutarán todas automáticamente cada vez que se compile una nueva versión de la rama principal del sistema. En caso de que alguna de estas pruebas falle, el entorno de despliegue continuo abortará el despliegue de la nueva versión.

A continuación, se expone el plan de pruebas unitarias, dividido por módulos del servidor. Para cada una de las pruebas se indicará la acción a realizar y el resultado esperado que devolverá la API REST.

Módulo de locales	
Acción	**Resultado esperado**
Registrar un local correctamente	201 (CREATED)
Registrar un local con un valor no numérico para su capacidad	400 (BAD REQUEST)
Registrar un local sin cubrir todos los campos del registro (Nombre, Dirección y Aforo)	400 (BAD REQUEST)
Registrar un local en una dirección ya en uso por otro local	400 (BAD REQUEST)
Registrar otro local con un nombre ya en uso por otro local y una dirección nueva	201 (CREATED)
Obtener un local sin aportar un identificador	404 (NOT FOUND)

Obtener un local a partir de un identificador con formato incorrecto	404 (NOT FOUND)
Obtener un local a partir de un identificador no existente en el sistema	404 (NOT FOUND)
Obtener el nombre de un local a partir de un identificador incorrecto	404 (NOT FOUND)
Obtener locales por nombre sin aportar un nombre	404 (NOT FOUND)
Obtener locales a partir de un nombre existente en el sistema	Lista con dos locales con el mismo nombre

Módulo de alertas	
Acción	**Resultado esperado**
Obtener posibles alertas en un local que no existe en la base de datos	Lista vacía
Crear una nueva alerta en estado VALIDADA con tres visitas	201 (CREATED)
Crear una nueva alerta sin validar con tres visitas	201 (CREATED)
Crear una alerta sin cubrir todos los campos necesarios (Fecha de creación, Fecha de comienzo de síntomas y Estado)	400 (BAD REQUEST)
Crear una alerta cuyas visitas carecen de algún campo (Identificador del local, Fecha de entrada y Fecha de salida)	400 (BAD REQUEST)
Crear una alerta con una lista vacía de visitas	201 (CREATED)
Obtener una alerta de posible contagio en un local visitado en dos alertas, una validada y otra sin validar	Lista conteniendo únicamente la segunda visita de la alerta validada
Obtener las alertas que están sin validar	Lista conteniendo las dos últimas alertas creadas

Módulo de autenticación como administrador	
Acción	**Resultado esperado**
Registrarse como administrador con las credenciales correctas	Token de sesión
Acceder al panel de control con el token generado	200 (OK)
Acceder al panel de control con un token incorrecto	403 (FORBIDDEN)
Acceder al panel de control sin un token	403 (FORBIDDEN)

Registrarse como administrador con un nombre de usuario incorrecto	403 (FORBIDDEN)
Registrarse como administrador con una contraseña incorrecta	403 (FORBIDDEN)

Módulo de variables de contagio	
Acción	**Resultado esperado**
Actualizar el valor de una variable de entorno correctamente con un token de sesión	201 (OK)
Asignar un valor no numérico a la variable de entorno	400 (BAD REQUEST)
Obtener el valor de una variable de entorno	200 (OK)

Tras haber mostrado el plan de pruebas unitarias es necesario aclarar que, debido a la generación aleatoria de identificadores tanto de alertas como de locales, ciertas operaciones no se pueden probar mediante pruebas automatizadas. Sin embargo, esto no es un problema puesto que las pruebas unitarias se completarán con pruebas del sistema para comprobar todas las funcionalidades completamente.

8.6.2 Pruebas de Aceptación del Sistema

Las pruebas de aceptación del sistema se realizarán sobre la aplicación móvil y la página web con el fin de comprobar que ambas se integran bien con el servidor y que el comportamiento de ambas es el esperado en todas las situaciones que se puedan llegar a dar.

Además de realizar exhaustivas pruebas manuales para verificar que se cumplen los criterios de aceptación definidos en las historias de usuario, se automatizarán todas aquellas pruebas que no necesiten de acciones externas para ser realizadas, con el fin de facilitar el trabajo del equipo de desarrollo. Para llevar a cabo dicha automatización se decidió utilizar la plataforma *TestProject*. A continuación, se presenta el plan de pruebas de sistema, indicando para cada prueba sus pasos y el resultado esperado de cada uno.

8.6.2.1 Pruebas automatizadas de la aplicación móvil

Notificar un nuevo positivo	
Pasos a realizar	**Resultado esperado**
Con la aplicación abierta, acceder a la pantalla de registro de positivos pulsando en el botón correspondiente de la barra de navegación	Se accede a la pantalla de registro de positivos
Pulsar el botón “REGISTRAR”	Se muestra una ventana de error que indica que el código es incorrecto

Cerrar la ventana de error, escribir un código de rastreo con 11 dígitos de longitud y pulsar el botón de "REGISTRAR"	Se muestra una ventana de error que indica que el código es incorrecto
Cerrar la ventana de error y escribir un código con más de 12 dígitos	La aplicación impide superar los 12 dígitos en la entrada del código de rastreo
Pulsar en el campo de la fecha de aparición de los síntomas	Se muestra un calendario con el día actual seleccionado.
Seleccionar el día posterior al actual, cerrar el calendario y registrar el positivo	Se muestra una ventana de error indicando que la fecha es incorrecta
Seleccionar la fecha actual en el calendario y registrar el positivo	Se muestra una ventana indicando que el positivo se notificó correctamente

Configurar frecuencia de sincronización	
Pasos a realizar	**Resultado esperado**
Con la aplicación abierta, acceder a la pantalla de listado de alertas pulsando en el botón correspondiente de la barra de navegación	Se accede a la pantalla de listado de alertas
Pulsar el botón para cambiar la frecuencia de sincronización	Se muestra la ventana de selección de frecuencia
Seleccionar la segunda opción y pulsar el botón "OK"	Se muestra la frecuencia de sincronización correspondiente a la segunda opción
Pulsar el botón para cambiar la frecuencia de sincronización, seleccionar la tercera opción y pulsar el botón "OK"	Se muestra la frecuencia de sincronización correspondiente a la tercera opción
Ir a la vista de escáner y volver a la vista de listado de alertas	Se muestra la frecuencia de sincronización correspondiente a la tercera opción
Enviar la aplicación a segundo plano y traerla de nuevo al frente	Se muestra la frecuencia de sincronización correspondiente a la tercera opción
Cerrar por completo la aplicación, abrirla de nuevo y acceder a la pantalla de listado de alertas	Se muestra la frecuencia de sincronización correspondiente a la tercera opción

Vista de información de alertas	
Pasos a realizar	**Resultado esperado**
Con la aplicación abierta, acceder a la pantalla de listado de alertas pulsando en el botón correspondiente de la barra de navegación	Se accede a la pantalla de listado de alertas
Pulsar el botón de información	Se muestra la ventana de información de alertas
Pulsar el botón "OK"	Se cierra la ventana de información de alertas

Pulsar nuevamente el botón de información	Se muestra la ventana de información de alertas
Pulsar el botón de “ATRÁS” del dispositivo	Se cierra la ventana de información de alertas

8.6.2.2 Pruebas manuales de la aplicación móvil

Registrar entrada en locales	
Pasos a realizar	**Resultado esperado**
Con la aplicación abierta, pulsar el botón de activar escáner	Se activa la cámara con el lector de QRs
Enfocar un código no perteneciente al sistema	Se desactiva la cámara y se muestra un mensaje indicando que el código es incorrecto
Cerrar la ventana de error, activar el escáner y enfocar un código de un local que ya no pertenezca al sistema	Se desactiva la cámara y se una ventana de error que indica que el código es incorrecto
Cerrar la ventana de error, activar el escáner y enfocar un código de un local registrado en el sistema que se encuentre con la ocupación completa	Se desactiva la cámara y se muestra una ventana de error que indica que el local está lleno
Cerrar la ventana de error, activar el escáner y enfocar un código de un local registrado en el sistema que se encuentre con la ocupación a uno por debajo del máximo	Se desactiva la cámara, se abre la puerta en la aplicación y se sustituye el texto “Entrar a un local” por “Salir del local” indicando al usuario que ha entrado. Se crea una nueva entrada en la tabla de visitas de la base de datos
Activar el escáner y enfocar el código escaneado previamente	Se desactiva la cámara, se cierra la puerta y se sustituye el texto “Salir del local” por “Entrar a un local” indicando al usuario que ha salido. Se añade a la visita creada la fecha de salida
Activar el escáner y enfocar el código escaneado previamente	Se desactiva la cámara, se abre la puerta en la aplicación y se sustituye el texto “Entrar a un local” por “Salir del local” indicando al usuario que ha entrado. Se crea una nueva entrada en la tabla de visitas de la base de datos
Activar el escáner y enfocar el código de otro local perteneciente al sistema	Se desactiva la cámara y se sigue mostrando al usuario que está dentro, con la puerta abierta y el texto “Salir del local”. Se añade a la última visita creada la fecha de salida y se crea una nueva visita

Búsqueda manual de alertas	
Pasos a realizar	**Resultado esperado**
DISPOSITIVO 1: Activar el escáner y enfocar un código de un local registrado en el sistema	Se desactiva la cámara, se abre la puerta en la aplicación y se sustituye el texto "Entrar a un local" por "Salir del local" indicando al usuario que ha entrado. Se crea una nueva entrada en la tabla de visitas de la base de datos
DISPOSITIVO 1: Acceder a la pantalla de listado de alertas pulsando en el botón correspondiente de la barra de navegación	Se accede a la pantalla de listado de alertas
DISPOSITIVO 1: Deslizar hacia abajo para recargar la lista	Tras un tiempo de carga, se muestra una notificación que indica la ausencia de nuevas alertas y la lista se mantiene vacía
DISPOSITIVO 2: Activar el escáner y enfocar el código del local escaneado por el otro dispositivo	Se desactiva la cámara, se abre la puerta en la aplicación y se sustituye el texto "Entrar a un local" por "Salir del local" indicando al usuario que ha entrado. Se crea una nueva entrada en la tabla de visitas de la base de datos
DISPOSITIVO 2: Esperar el tiempo mínimo establecido para considerar a un contacto estrecho, activar el escáner y enfocar el código de nuevo	Se desactiva la cámara, se cierra la puerta y se sustituye el texto "Salir del local" por "Entrar a un local" indicando al usuario que ha salido. Se añade a la visita creada la fecha de salida
DISPOSITIVO 1: Activar el escáner y enfocar el código de nuevo	Se desactiva la cámara, se cierra la puerta y se sustituye el texto "Salir del local" por "Entrar a un local" indicando al usuario que ha salido. Se añade a la visita creada la fecha de salida
DISPOSITIVO 2: Navegar a la pantalla para notificar positivos, escribir un código de 12 dígitos y pulsar en "REGISTRAR"	Se muestra una ventana indicando que el positivo se notificó correctamente
DISPOSITIVO 1: Acceder a la pantalla de listado de alertas y deslizar hacia abajo para recargar la lista	Tras un tiempo de carga, se muestra una notificación que indica la ausencia de nuevas alertas y la lista se mantiene vacía
WEB: Acceder al listado de alertas sin confirmar de la aplicación web y validar la que contenga el código escrito en el DISPOSITIVO 2	Se elimina la alerta de la lista y pasa a estado VALIDADA
DISPOSITIVO 1: Acceder a la pantalla de listado de alertas y deslizar hacia abajo para recargar la lista	Tras un tiempo de carga, se muestra una notificación que indica la existencia de nuevas alertas. En la lista aparece un nuevo posible contacto con el nombre del local escaneado y la fecha y hora en la que comenzó la coincidencia entre los dos dispositivos

Inicio de la aplicación	
Pasos a realizar	**Resultado esperado**
Instalar la aplicación y abrirla por primera vez	Se muestra la pantalla de bienvenida y un mensaje para pedir permiso para utilizar la cámara del dispositivo
Denegar el permiso de usar la cámara	Se cierra la ventana y arranca la aplicación. Se muestra la ventana de términos y condiciones
Tratar de navegar por la aplicación sin cerrar la pantalla de términos y condiciones	Se sigue mostrando y no deja hacer nada más
Cerrar la aplicación y abrirla otra vez	Se vuelven a mostrarla petición de permisos y los términos y condiciones
Aceptar los términos y condiciones	Se cierra la ventana de términos y condiciones y se permite utilizar la aplicación
Pulsar en el botón para activar el escáner	No se activa la cámara
Cerrar la aplicación, abrirla de nuevo y otorgar permiso para utilizar la cámara	No se muestran los términos y condiciones y además se realiza una sincronización con el servidor en busca de alertas
Pulsar en el botón para activar el escáner	Se activa la cámara

8.6.2.3 Pruebas automatizadas de la aplicación web

Registrar un local	
Pasos a realizar	**Resultado esperado**
Navegar a la URL de la página web	Se muestra el formulario de registro de locales
Pulsar el botón “Generar QR”	Se resaltan en rojo todos los campos del formulario
Escribir un nombre en el local y pulsar el botón “Generar QR”	Se resaltan en rojo todos los campos del formulario menos el nombre
Escribir una dirección para el local y pulsar el botón “Generar QR”	Se resalta en rojo el campo del aforo máximo
Tratar de escribir un aforo no numérico	La entrada bloquea la entrada de caracteres no numéricos
Escribir un aforo que sea un número positivo y generar el QR	La aplicación web carga una nueva página en la que se muestra un código QR y dos botones
Pulsar en el botón “Descargar”	Se descarga un archivo de imagen
Pulsar el botón “Volver”	La aplicación web carga de nuevo el formulario de registro de locales
Registrar un local en la misma dirección que el anterior	Se resalta en rojo el campo de la dirección y se muestra un mensaje indicando que ya

	existe un local registrado en esa dirección
Navegar a la dirección "<Dominio general>/generated_qr/ dummyinvalidaddressqrnotvalid" (dirección con el formato aceptado pero que no representa un local)	Se muestra un mensaje diciendo que el local es incorrecto y botón para volver atrás

Buscar locales	
Pasos a realizar	**Resultado esperado**
Navegar a la URL de la página web	Se muestra el formulario de registro de locales
Pulsar en la opción de menú "BUSCAR LOCAL"	Se muestra la pantalla de búsqueda de locales
Pulsar en el botón de la lupa	Se resalta en rojo la barra de búsqueda vacía
Escribir el nombre de un local existente y pulsar el botón de la lupa	Se muestra la tarjeta de dicho local. La tarjeta contiene el nombre, la dirección, el aforo máximo y el código QR
Escribir un nombre perteneciente a varios locales en el sistema	Se muestran las tarjetas de todos los locales con ese nombre.
Pulsar en la opción de menú "REGISTRAR LOCAL"	Se muestra la pantalla de registro de locales

Acceso al panel de administrador	
Pasos a realizar	**Resultado esperado**
Acceder a la dirección "<Dominio general>/login"	Se muestra el formulario de inicio de sesión
Se pulsa en el botón "ENTRAR"	Se resaltan en rojo los campos de usuario y contraseña y se muestra un mensaje de error
Se introduce el nombre de usuario correcto y una contraseña incorrecta y se pulsa "ENTRAR"	Se resaltan en rojo los campos de usuario y contraseña y se muestra un mensaje de error
Se introduce el nombre de usuario incorrecto y una contraseña correcta y se pulsa "ENTRAR"	Se resaltan en rojo los campos de usuario y contraseña y se muestra un mensaje de error
Se introducen las credenciales correctas	La aplicación web redirige al usuario al panel de control de variables
Se actualiza el valor de una variable y se pulsa en "ACTUALIZAR"	Se muestra un mensaje indicando que se actualizó el valor de la variable

Validar alertas	
Pasos a realizar	**Resultado esperado**
Acceder a la dirección "<Dominio general>/alerts"	Se muestra una lista con dos alertas sin validar
Se pulsa sobre el botón "VALIDAR" de la primera alerta	Se elimina la alerta de la lista y en la base de datos pasa a estado "VALIDADA"
Se pulsa sobre el botón "ELIMINAR" de la primera alerta	Se elimina la alerta de la lista y de la base de datos

Pantalla de política de privacidad	
Pasos a realizar	**Resultado esperado**
Navegar a la URL de la página web	Se muestra el formulario de registro de locales, la cabecera de la aplicación y un pie de página con un enlace
Pulsar en el enlace del pie de página	La aplicación carga la pantalla de aviso legal y política de privacidad
Pulsar en el título de la cabecera	La aplicación carga la pantalla de inicio con el formulario de registro de locales

8.6.3 Pruebas de Usabilidad

Las pruebas de usabilidad se realizarán sobre un grupo de usuarios reales que actuará a modo de muestra de la población a la que va orientado este sistema, es decir, toda la población española sin importar su habilidad con las nuevas tecnologías. Con estas pruebas se pretende evaluar la adecuación de las aplicaciones web y móvil a las necesidades de los usuarios.

A continuación, se presentan los modelos de cuestionarios que se utilizarán en estas pruebas, junto con las listas de actividades guiadas que seguirán los usuarios según su rol.

8.6.3.1 Preguntas de carácter general

¿Usa un dispositivo móvil frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza más con el dispositivo móvil?
1. Es parte de mi trabajo o profesión 2. Lo uso principalmente para jugar y escuchar música 3. Empleo aplicaciones de mensajería instantánea y redes sociales 4. Únicamente leo el correo y navego ocasionalmente por internet

¿Ha usado alguna vez un sistema de rastreo para el COVID-19?
1. Sí, he utilizado Radar Covid 2. Si, pero no he utilizado Radar Covid 3. No, aunque me lo he planteado 4. No, nunca
¿Qué busca Vd. principalmente en una aplicación de rastreo de enfermedades?
1. Que sea fácil de usar 2. Que respete la privacidad de los usuarios 3. Que no interfiera en mi día a día 4. Que tenga todas las funciones necesarias para ser lo más efectiva posible

8.6.3.2 Actividades guiadas

8.6.3.2.1 Usuario estándar

- Buscar un local
- Comprobar la política de privacidad
- Abrir la aplicación móvil y leer los términos y condiciones
- Registrar la entrada a un local
- Notificar un positivo
- Actualizar la lista de alertas
- Cambiar la frecuencia de sincronización
- Registrar la salida del local
- Cerrar la aplicación y comprobar el funcionamiento de la sincronización automática

8.6.3.2.2 Dueño de un local

- Cumplimentar el formulario para registrar el local
- Descargar el código QR
- Realizar las actividades de los usuarios estándar
- Comprobar el funcionamiento del control de aforo

8.6.3.2.3 Administrador de sistemas

- Iniciar sesión de administrador
- Actualizar el valor de alguna variable de contagio
- Realizar las actividades de los usuarios estándar

8.6.3.3 Preguntas cortas sobre la aplicación y observaciones

Facilidad de Uso de la Aplicación Web	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la página?				
¿Existe ayuda para las funciones en caso de que tenga dudas?				
¿Le resulta sencillo el uso de la aplicación web?				
Funcionalidad de la aplicación Web	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. espera?				
¿El tiempo de respuesta de la página es muy grande?				
Facilidad de Uso de la Aplicación Móvil	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la aplicación?				
¿Existe ayuda para las funciones en caso de que tenga dudas?				
¿Le resulta sencillo el uso de la aplicación?				
Funcionalidad de la Aplicación Móvil	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. espera?				
¿El tiempo de respuesta de la aplicación es muy grande?				

Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es				
Los iconos e imágenes usados son				
Los colores empleados son				
Diseño de la Interfaz Móvil		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				
¿El diseño de las pantallas de la aplicación es claro y atractivo?				
¿Cree que la aplicación está bien estructurada?				
Diseño de la Interfaz Web		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				
¿El diseño de las páginas es claro y atractivo?				
¿Cree que la página web está bien estructurada?				
Observaciones				
Cualquier comentario del usuario				

Capítulo 9. Implementación del Sistema

9.1 Estándares y Normas Seguidos

Para llevar a cabo el desarrollo de este sistema se han seguido las leyes, normas y estándares indicados a continuación:

- Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.
- REGLAMENTO (UE) 2016/679 DEL PARLAMENTO EUROPEO Y DEL CONSEJO de 27 de abril de 2016 relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento General de Protección de Datos).
- Estándar de diseño de interfaces de usuario Material Design.
- Scrumban, metodología de desarrollo ágil de software. Esta metodología ha sido adaptada a las características específicas de este proyecto.

9.2 Lenguajes de Programación

9.2.1 C#

C# es un lenguaje de programación multiplataforma desarrollado y estandarizado por Microsoft como parte de la plataforma .NET. C# es una evolución de los lenguajes C y C++, de los que toma su sintaxis básica, que utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque cuenta con sustanciales mejoras inspiradas en otros lenguajes como Python o SQL. En la versión de .NET Core, su compilador se reconstruyó por completo, consiguiendo que las aplicaciones sean un 600% más rápidas que las desarrolladas con versiones anteriores.

Pese a ser un lenguaje moderno orientado a objetos y enfocado también al paradigma de programación funcional, C# conserva la potencia original de C, permitiendo al programador el acceso a bajo nivel al núcleo de los sistemas operativos, el trabajo directo con punteros a memoria y la interacción con elementos físicos como puertos USB o tarjetas gráficas.

Las principales características de C#, y los motivos por los que lo he escogido como lenguaje para este proyecto, son las siguientes:

- Sencillez: La sintaxis de C# es muy similar a la de otros lenguajes orientados a objetos, lo que hace que su curva de aprendizaje no sea muy pronunciada.
- Seguridad: C# incorpora mecanismos cuya función es asegurarse de que los accesos a datos se realizan de forma correcta.
- Extensibilidad: C# ofrece la posibilidad de agregar tipos de datos básicos, operadores y modificadores mediante código, lo que aporta una gran flexibilidad al desarrollo.
- Compatibilidad: El hecho de ser un lenguaje multiplataforma permite que la base de código de una aplicación sea reutilizable para varias plataformas.

9.2.1.1 Xamarin y Xamarin.Forms

Para desarrollar software para dispositivos móviles a través del lenguaje C# y la plataforma de desarrollo .NET, Microsoft creó el *framework* Xamarin. Xamarin es una capa de abstracción que administra la comunicación de código compartido con el código de la plataforma subyacente (Android, iOS o WPF). Gracias a Xamarin, es posible desarrollar aplicaciones para las tres plataformas mencionadas compartiendo de media un 90% del código de la aplicación entre todas ellas. Además, al ejecutarse en un entorno administrado, Xamarin ofrece ventajas como la asignación de memoria y la recolección automática de elementos no utilizados.

Con el fin de aumentar la cantidad de código compartido entre las implementaciones de una misma aplicación en cada una de las plataformas nace Xamarin.Forms. Tal y como se explica en el apartado 4.2.2, Xamarin.Forms es un marco de interfaz de usuario que permite, no solo reutilizar la lógica de negocio de la aplicación, sino también crear interfaces de usuario comunes a todas las plataformas, que en tiempo de compilación se transforman en vistas y controles nativos del sistema operativo al que están destinadas.

9.2.1.2 Paquetes NuGet

NuGet es una herramienta de compartición de paquetes creada por Microsoft para la plataforma .NET. Un paquete NuGet es un archivo comprimido en formar ZIP con la extensión ".nupkg" que contiene código compilado, archivos relacionados con este código y un manifiesto descriptivo en el que se incluye toda la información relativa al paquete y su versión. Estos paquetes se suben a un repositorio público o privado, y desde ahí otros desarrolladores pueden incorporarlos a sus proyectos. Gracias al administrador de paquetes NuGet incorporado en el IDE Visual Studio, para incorporar un paquete a un proyecto tan sólo hay que buscarlo en el repositorio, seleccionar la versión deseada y descargarlo, el administrador de paquetes se encarga del resto de tareas intermedias.

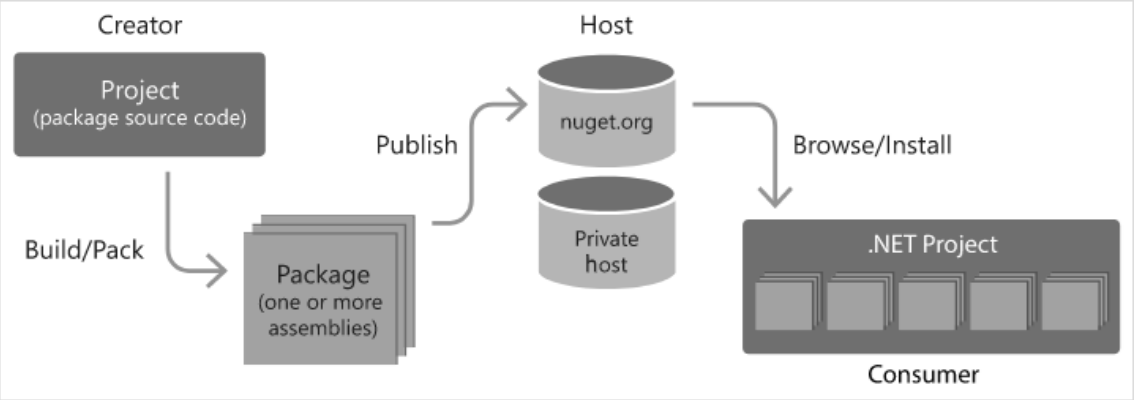

Figura 9.1. Flujo de los paquetes NuGet

Para desarrollar este proyecto se han utilizado los paquetes NuGet que se describen a continuación:

9.2.1.2.1 Xamarin.Essentials 1.6.1

Xamarin.Essentials proporciona una API multiplataforma para proyectos Xamarin accesible desde código compartido que permite acceder a características de los dispositivos y sistemas operativos Android, iOS y WPF que de otra forma solo serían accesibles mediante sus propias APIs nativas. En concreto, en este proyecto se ha utilizado esta librería para acceder y gestionar las preferencias persistentes de las aplicaciones móviles.

9.2.1.2.2 ZXing.Net.Mobile.Forms 2.4.1

ZXing.Net.Mobile es una librería de C#/.NET basada en el código abierto de la librería de lectura de códigos de barras ZXing (Zebra Crossing). El objetivo de esta librería es facilitar la incorporación del escaneo de códigos de barras a las aplicaciones móviles desarrolladas con Xamarin y Xamarin.Forms. ZXing soporta una gran variedad de formatos de códigos de barras, tanto unidimensionales como códigos QR.

9.2.1.2.3 SQLite-NET-PCL 1.7.335

SQLite-NET es un paquete ligero de código abierto que proporciona almacenamiento persistente mediante bases de datos SQLite a las aplicaciones móviles desarrolladas para .NET,

Mono y Xamarin. Esta versión de SQLite-NET proporciona versiones de SQLite propias para cada plataforma, que luego podrán ser tratadas de manera uniforme gracias al sistema de inyección de dependencias de Xamarin.Forms.

9.2.2 JavaScript

JavaScript es un lenguaje de programación interpretado con funciones de primera clase utilizado principalmente para el desarrollo de páginas web, aunque cada vez mes más común su uso fuera de los navegadores, en entornos como Node.JS y Apache. JavaScript surge como dialecto del estándar ECMAScript, y su sintaxis se diseñó en base a la del lenguaje C, aunque adopta nombres y convenciones del lenguaje Java. Desde 2012, todos los navegadores modernos soportan completamente ECMAScript 5.1, una versión de JavaScript. Las principales características de JavaScript son las siguientes:

- Es un lenguaje basado en prototipos
- Se trata de un lenguaje imperativo y estructurado
- Es muy dinámico
- Es un lenguaje débilmente tipado
- Soporta la programación orientada a objetos
- Está orientado hacia el paradigma de programación funcional

9.2.2.1 Node.JS

Node.JS es un entorno de ejecución de JavaScript, construido con el motor de JavaScript V8 de Chrome, diseñado para crear aplicaciones web escalables gracias a su manejo de eventos asíncronos en forma de *callbacks*. Un *callback* es simplemente una función *x* que se usa como argumento de otra función *y*. Cuando se llama a la función *y*, esta ejecuta la función *x*. En Node.JS, a cada petición se le asigna un callback, lo que permite procesar varias peticiones de manera asíncrona, e ir respondiéndolas a medida que se van obteniendo los resultados. Este modelo de concurrencia contrasta bastante con el modo más común de manejar trabajos concurrentes, que consiste en utilizar distintos hilos del sistema operativo para cada tarea. Además de difíciles de usar, las redes basadas en hilos son poco eficientes. Pese a no estar diseñado para ello, Node.JS también incluye funciones que permiten a los desarrolladores aprovecharse de los múltiples núcleos de los procesadores a la hora de generar subprocesos.

Para este proyecto se han elegido Node.JS y JavaScript como las tecnologías principales, tanto en el *back-end*, como en la aplicación web a desarrollar. Dado que el tratamiento de datos en este proyecto se realiza de manera descentralizada, el servidor central no realizará operaciones con un coste computacional elevado, para las que un *backend* desarrollado con un lenguaje compilado sería más adecuado. El principal cometido del servidor es atender las peticiones de los dispositivos mediante una API REST. De cara al *frontend*, Node.JS cuenta con varios *frameworks* de desarrollo de aplicaciones web, como por ejemplo ReactJS o Angular V2.

9.2.2.2 Express

Express es una infraestructura de aplicaciones web mínima y flexible diseñada para el entorno de ejecución Node.js. Proporciona un conjunto sólido de características para las aplicaciones web y móviles, así como miles de métodos y programas de utilidad HTTP y middleware que facilitan la creación de una API REST sólida, rápida y sencilla de mantener. Express ha sido la tecnología escogida para implementar el servidor *backend* del sistema de rastreo.

Además de Express, para desarrollar el servidor del sistema se han utilizado las siguientes librerías disponibles para Node.JS.

9.2.2.2.1 JSON Web Token 8.5.1

Este paquete contiene los mecanismos necesarios para la generación y validación de tokens de sesión, los cuales permiten identificar y autenticar a un usuario a través de las llamadas a los servicios de una API REST.

9.2.2.2.2 Mongoose 5.11.19

Tal y como se explicó en el apartado 8.4.2, Mongoose es un ODM (*Object-Document Mapper*) para Node.JS que permite realizar transformaciones entre objetos en el código y documentos en la base de datos.

9.2.2.3 ReactJS

ReactJS es una biblioteca para construir interfaces de usuario mediante una estructura de componentes, es decir, piezas de código autocontenidas que describen una parte de la interfaz. Estos componentes se juntan y combinan entre ellos para crear las interfaces de usuario completas. A la hora de hacer esto, React abstrae el grueso del trabajo de renderizado de dichos componentes con el fin de que el desarrollador pueda centrarse en el diseño y la creación de la interfaz y su lógica de presentación. A diferencia de otros *frameworks*, ReactJS no impone reglas estrictas sobre convenciones de código, arquitectura o estructuras de paquetes y código, por lo que permite a los equipos de desarrollo organizarse de la manera que se encuentre más eficiente en cada proyecto y favoreciendo también de este modo el desarrollo ágil.

Para desarrollar la aplicación web y sus interfaces de usuario, además de ReactJS se han usado las librerías que se indican a continuación.

9.2.2.3.1 Material UI 4.11

Material UI es una biblioteca de componentes e iconos de código abierto para ReactJS cuyo objetivo es simplificar la creación de interfaces de usuario con React que sigan el estándar Material Design.

9.2.2.3.2 QRCode.React 1.0.1

Este paquete es una API ligera que permite la codificación y descodificación de información en códigos QR, así como el renderizado de los mismos a través de un componente para ReactJS.

9.2.2.3.3 JS-Base64 3.6.0

Este módulo permite codificar y descodificar cadenas de información a través del algoritmo Base 64 con el fin de ser utilizadas a modo de URIs.

9.3 Herramientas y Programas Usados para el Desarrollo

9.3.1 Git y GitHub

Git es un sistema de control de versiones distribuido de código libre diseñado para gestionar todo tipo de proyectos de forma rápida y eficiente. Para almacenar y gestionar todo el código desarrollado en este proyecto se ha hecho uso de la plataforma de desarrollo colaborativo GitHub y de la versión de Git para Windows. Gracias a GitHub, se ha podido llevar a cabo un control de versiones de los tres subsistemas del proyecto (servidor, aplicación web y aplicación móvil) mediante el uso de ramas para cada una de las funcionalidades a desarrollar.

9.3.2 Visual Studio 2019

Visual Studio 2019 es un entorno de desarrollo integrado (IDE del inglés Integrated Development Environment) desarrollado por Microsoft orientado a la programación de aplicaciones en C#/.NET y C/C++, aunque también cuenta con soporte para muchos otros lenguajes de programación. Para el desarrollo de la aplicación móvil se ha usado la versión Community, puesto que no se han necesitado características especiales que incluyen las versiones superiores. Además de esto, Visual Studio 2019 está integrado con el sistema de control de versiones Git, lo que ha facilitado mucho la gestión del código a nivel de ramas y confirmaciones.

9.3.3 Visual Studio Code

Visual Studio Code es otro entorno de desarrollo creado por Microsoft. En este caso, se trata de un entorno más ligero, de código abierto, que gracias a los plugins creados por la comunidad permite trabajar con una multitud de lenguajes de programación distintos. En este proyecto, este entorno se ha utilizado para programar el servidor y la aplicación web, dado que permite ejecutar, analizar y probar aplicaciones de Node.JS de manera cómoda y eficiente gracias a su consola de comandos interna. Además, al igual que Visual Studio 2019, está integrado con Git.

9.3.4 Azure App Service

Azure App Service es un servicio para construir, desplegar y escalar aplicaciones web integrado dentro del paquete Azure de Microsoft. Integrado con varios sistemas de control de versiones, AppService permite compilar y desplegar aplicaciones web desde código escrito en casi

cualquier lenguaje de programación alojado en un repositorio de casi cualquier sistema de control de versiones. Azure App Service está integrado además con Visual Studio Code, lo cual simplifica aún más la compilación y el despliegue de las aplicaciones, y cuenta con un sistema de CI/CD a través de AzureDevOps que agiliza dichas tareas de despliegue. Además, App Service garantiza una disponibilidad del 99,95% y proporciona numerosos mecanismos de monitorización y protección de las aplicaciones, así como servicios sanitarios para arreglar posibles fallos que puedan ocurrir en ellas. La gran variedad de cuotas y sus posibilidades de escalada de recursos, junto con las garantías que ofrece una empresa como Microsoft, además de las ya mencionadas integraciones con Azure DevOps, Visual Studio Code y GitHub, han sido las razones principales para elegir App Service por delante de otras opciones como por ejemplo Heroku.

9.3.5 Azure DevOps

Azure DevOps es un conjunto de servicios de desarrollo enfocados a metodologías ágiles que busca mejorar el planeamiento y la colaboración dentro de un proyecto, así como agilizar la integración y el despliegue de aplicaciones. Los servicios englobados dentro de Azure DevOps son Azure Boards, Azure Pipelines, Azure Repos, Azure Test Plans y Azure Artifacts, de los cuales en este proyecto se han usado los dos primeros.

9.3.5.1 Azure Boards

Azure Boards es una herramienta que permite crear backlogs, iteraciones y tableros Kanban para gestionar historias de usuario, tareas, bugs, etc. Se trata de un buen servicio para gestionar proyectos de software desarrollados bajo metodologías ágiles. En este proyecto ha sido de gran utilidad para organizar las iteraciones y gestionar el ciclo de vida de las historias de usuario.

9.3.5.2 Azure Pipelines

A la hora de aplicar la integración y distribución continuas en cualquier proyecto es necesario utilizar una plataforma que permita incorporar un nivel alto de automatización permanente y supervisión constante al desarrollo del software. Azure Pipelines combina la integración y la distribución continuas para probar, compilar y enviar el software de manera constante y coherente al destino de producción. Las principales ventajas de Azure Pipelines con respecto a otros sistemas de canalización CI/CD son las siguientes:

- Funciona con cualquier lenguaje o plataforma gracias al uso de ficheros en formato YML.
- Cuenta con una potente interfaz gráfica que acelera y simplifica los procesos de canalización de la integración y el despliegue.
- Permite implementar en distintos tipos de destinos a la vez.
- Ofrece la posibilidad de compilar en máquinas Windows, Linux y Mac OS.
- Funciona con proyectos de código abierto.
- Está integrado con GitHub.

9.3.6 Postman

Postman es un cliente HTTP gratuito pensado para hacer pruebas sobre APIs web. Permite realizar peticiones, obtener las respuestas que generan dichas peticiones e incluso automatizar el envío de dichas peticiones. En este proyecto, Postman se ha usado durante las tareas de desarrollo del servidor para comprobar el funcionamiento de la API REST que estaba implementando. Además, ha servido como herramienta para realizar pruebas del sistema de manera manual sobre dicha API REST.

9.4 Patrones empleados

9.4.1 MVVM

El patrón de arquitectura de software Modelo-Vista-Vista Modelo es una adaptación del patrón Modelo de Presentación, de Martin Fowler, creado por un equipo de desarrollo de Microsoft. El principal objetivo de este patrón es desacoplar al máximo la interfaz de usuario de una aplicación, ya no solo de su lógica de negocio, sino también de la lógica de presentación. De esta manera, aplicando este patrón correctamente se consigue desarrollar código funcional para interfaces de usuario muy distintas que hará que se comporten de la misma manera. El patrón MVVM ha sido adoptado por plataformas como Android, iOS o Windows Presentation Foundation como estándar para el desarrollo de aplicaciones.

La arquitectura MVVM, como su nombre indica, cuenta con tres partes:

- Modelo: Representa la capa de negocio y datos de la aplicación. Esta capa contiene las entidades y modelos de datos que se contemplan en la aplicación, pero en ningún caso ha de contener las acciones los servicios, métodos o acciones que los manipulan. Esta capa, por tanto, no tendrá dependencias con ninguna de las otras capas de la aplicación.
- Vista: Esta capa representa la interfaz de usuario propiamente dicha. Ha de contener los controles, vistas y elementos necesarios para formar las distintas interfaces de la aplicación, y puede también contener comportamientos y lógica especifica de dicha interfaz. Esta lógica propia, aunque se debe evitar, puede aparecer generalmente en forma de eventos, animaciones propias del diseño de la interfaz, transiciones, etc. Siempre que sea posible se ha de delegar el comportamiento en la capa de Vista Modelo, de la que dependerá esta capa.
- Vista modelo: A modo de intermediario entre la vista y el modelo, esta capa contiene la gran mayoría de la lógica de las aplicaciones con arquitectura MVVM. Estas vistas modelo se comportan a modo de abstracción las vistas, implementando la lógica de presentación de la aplicación sin conocer el diseño particular de las pantallas. Gracias a esto, es posible variar el comportamiento de una vista en tiempo de ejecución, así como desarrollar varias vistas muy distintas que cumplirán las mismas funciones sin necesidad de duplicar la lógica. Para comunicar las vistas modelo con las vistas, generalmente se usan enlaces a datos y el patrón *Command*.

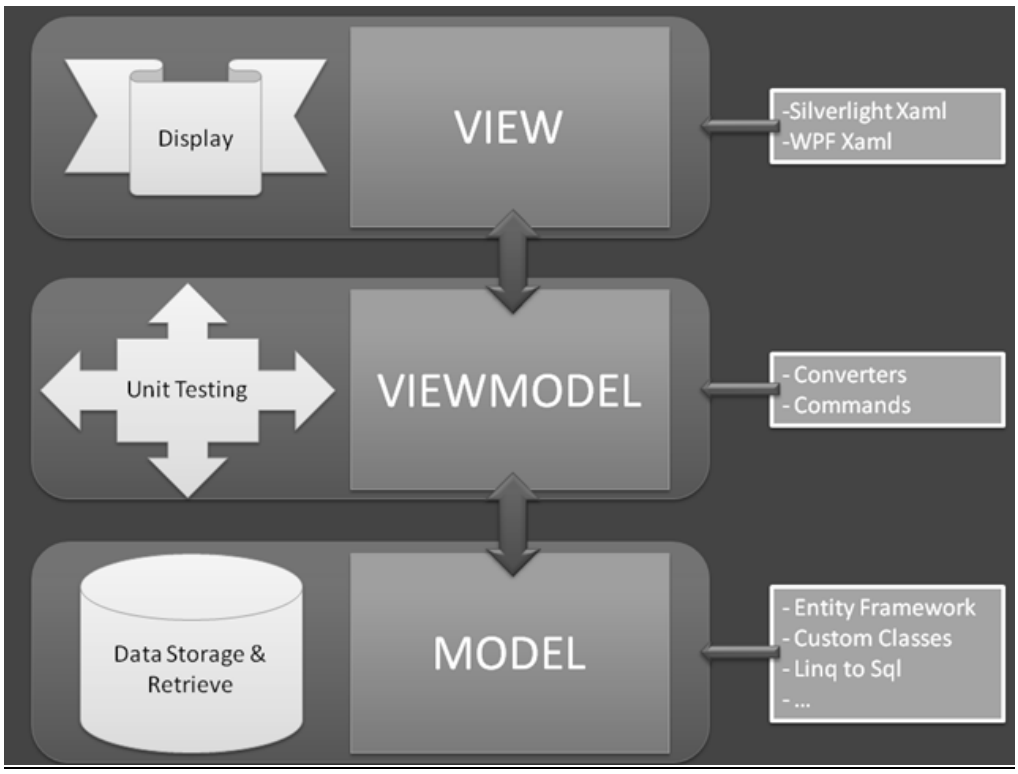

Figura 9.2. Arquitectura MVVM

9.4.2 Patrón repositorio

Este patrón permite crear una capa de abstracción entre la lógica de la aplicación y la implementación concreta del sistema de gestión de bases de datos. De esta manera se obtienen unas fachadas con una interfaz común que simplifican el acceso a datos y permiten cambiar de SGBD sin necesidad de modificar la lógica de la aplicación.

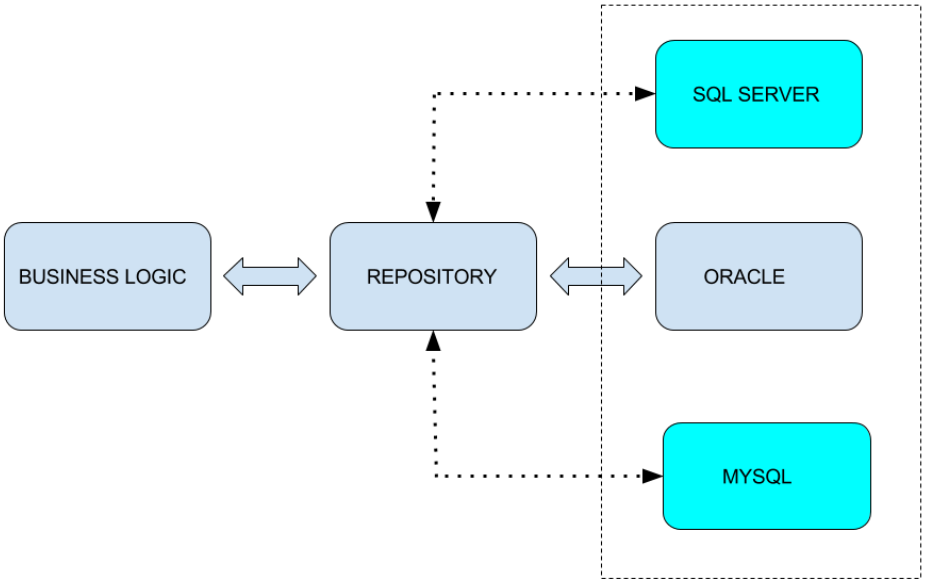

Figura 9.3. Patrón repositorio

9.5 Creación del Sistema

9.5.1 Problemas Encontrados

9.5.1.1 Despliegue de la aplicación web en Azure App Service

En la fase previa al desarrollo, durante las tareas de preparación y configuración del entorno de despliegue y el sistema de CI/CD, pese a no haber existido casi ningún problema a la hora de hacerlo para el servidor, cuando se trató de configurar App Service para alojar la aplicación web surgieron varios problemas. Pese a haber seguido la guía oficial de Microsoft, desplegar la aplicación fue bastante complicado, puesto que hubo que aplicar configuraciones distintas a las que sugería la guía. Tras conseguir el despliegue, sin embargo, la integración con Azure Pipelines seguía fallando, obteniendo una aplicación desplegada pero que no estaba integrada con el sistema CI/CD. Por estos motivos, se decidió alojar la página web desde un contenedor Azure Web Storage. Pese a no contar con todas las opciones de configuración y monitorización que otorga App Service, la función de mantener la aplicación web desplegada e integrada con el sistema CI/CD la cumple con creces, por lo que finalmente se trasladó definitivamente a Web Storage. El servidor, por el contrario, se mantiene satisfactoriamente desplegado con App Service.

9.5.1.2 Programación de próximas sincronizaciones en Android

Partimos del hecho de que la sincronización automática en segundo plano de la aplicación móvil ha de implementarse por separado para Android y par iOS, debido al funcionamiento y las medidas de seguridad que implementan cada uno de los sistemas operativos. Dejando de un lado la implementación para iOS, en Android surgió un problema que hizo que hubiera que modificar algunos criterios de aceptación.

La plataforma de desarrollo Android cuenta con un mecanismo para programar alarmas repetitivas llamado “Alarm Manager”. Este mecanismo permite programar alarmas para horas y fechas concretas, que independientemente del estado del dispositivo, se activarán y ejecutarán aquello que se haya programado. A la hora de desarrollar una aplicación nativa para Android en Java, este mecanismo funciona tal y como se espera. Sin embargo, la adaptación de Xamarin Android de este mecanismo, al menos en los dispositivos en los que se han realizado pruebas, no permite que estas alarmas se activen si el sistema operativo ha liberado la memoria correspondiente a la ejecución de la aplicación. Explicado de otra manera, cuando la aplicación se encuentra en primer o segundo plano, estas alarmas funcionan correctamente. Del mismo modo lo hacen cuando se cierra la aplicación, pero no se “mata” por completo. Sin embargo, cuando el usuario decide limpiar el uso de la RAM, o el sistema operativo necesita liberar espacio y libera la memoria correspondiente a esta aplicación, el “Alarm Manager” deja de funcionar.

Tras numerosas pruebas e implementaciones distintas, se optó por aceptar este comportamiento, tal y como se indica en las historias de usuario correspondientes, y en este caso se planteó la aplicación para que actúe como estaba pensado en un primer momento,

pero indicándole al usuario que tiene la posibilidad de cancelar la sincronización automática cerrando por completo la aplicación. Además, se optó por sincronizar de manera automática la aplicación cada vez que se inicia.

9.5.1.3 Respuestas de la API en las pruebas unitarios

A la hora de realizar las pruebas unitarias para la API REST, para probar algunas situaciones de prueba es necesario utilizar los valores obtenidos como respuesta de otras llamadas a la API. Sin embargo, la biblioteca de *testing* elegida da bastantes problemas para recuperar estos valores y utilizarlos en otros casos de uso. Por este motivo, ciertas situaciones de prueba pertenecientes al comúnmente denominado *happy path* no forman parte del plan de pruebas unitarias, sino que se comprueban de forma manual en las pruebas funcionales del sistema. El resultado de esta decisión es un plan de pruebas unitarias enfocado a comprobar que la API REST no acepta peticiones que no debe aceptar y no realiza acciones que no debe realizar.

9.5.1.4 Compilación y ejecución de pruebas en dispositivos iOS

Debido a la política de desarrollo de software de Apple, para poder crear aplicaciones para iOS se necesita un dispositivo Mac OS con el entorno XCode instalado. Desde un ordenador con Windows y Visual Studio se podría compilar una aplicación para iOS mediante una conexión con un dispositivo físico Mac Os con XCode, y se podría ejecutar en un emulador. Debido a que no dispongo de ningún dispositivo con Mac OS, la realización de estas pruebas se ha pospuesto a futuras ampliaciones. Sin embargo, no llevar las pruebas a cabo en dispositivos iOS no resta prácticamente fiabilidad al sistema, puesto que al ser tanto la lógica como las interfaces compartidas, el comportamiento en Android y iOS será idéntico. El único aspecto susceptible de tener fallos por no haber sido probado es la sincronización automática en iOS, dado que esta funcionalidad si que se desarrolló de manera independiente para cada plataforma. Por este motivo, en lugar de no realizarlas, se aplaza la ejecución de estas pruebas a futuras ampliaciones, y será en este momento cuando se adquieran los dispositivos de pruebas necesarios.

9.5.2 Descripción Detallada de las Clases

A continuación, se muestra la descripción detallada de las clases y archivos de código más importantes para comprender el funcionamiento del sistema, tanto del servidor como de la aplicación móvil.

9.5.2.1 place

Nombre	**Tipo**	**Descripción**	**Hereda de…**
place		Contiene las rutas de la API relativas a la gestión de locales	
Responsabilidades			
Número	**Descripción**		
1	Creación de nuevos locales		
2	Obtención de locales		
3	Gestionar el control de ocupación de los locales		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	JSON	POST(“/addPlace”)	name : String address : String capacity : Integer
Público	JSON	GET(“/getPlace”)	id : String
Público	JSON	POST(“/scanPlace”)	id : String isEntry : Boolean
Público	JSON	GET(“/getPlaceName”)	id : String
Público	JSON	GET(“/searchPlaceByName”)	name : String
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Observaciones			

9.5.2.2 alert

Nombre	**Tipo**	**Descripción**	**Hereda de…**
alert		Contiene las rutas de la API relativas a la gestión de alertas	
Responsabilidades			
Número	**Descripción**		
1	Crear nuevas alertas		
2	Obtener alertas en base a las visitas realizadas por un cliente		
3	Validar alertas		
4	Eliminar alertas inválidas		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	JSON	POST(“/addAlert”)	symptomsDate : DateTime alertDate : DateTime state : String visits : Visit[]
Público	JSON	POST(“/getAffectingAlerts”)	places : String[] fromDate : DateTime exclude : String[]
Público	JSON	GET(“/getNotValidated”)	
Público	JSON	POST(“/validate)	_id : String
Público	JSON	POST(“/deleteNotValid”)	_id : String
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Observaciones			

9.5.2.3 login

Nombre	**Tipo**	**Descripción**	**Hereda de…**
login		Contiene las rutas de la API relativas a la autenticación del administrador	
Responsabilidades			
Número	**Descripción**		
1	Autenticar al cliente como administrador		
2	Gestionar el acceso al panel de control de variables		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	JSON	POST("/")	username : String password : String
Público	JSON	GET("/panel")	token : String
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Observaciones			

9.5.2.4 environment

Nombre	**Tipo**	**Descripción**	**Hereda de…**
environment		Contiene las rutas de la API relativas a la gestión de las variables de contagio	
Responsabilidades			
Número	**Descripción**		
1	Obtener las variables de contagio		
2	Actualizar el valor de las variables de contagio		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	JSON	GET("/getVariables")	
Público	JSON	GET("/getVariable")	name : String
Público	JSON	POST("/setVariable")	name : String value : Integer token : String
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Observaciones			

9.5.2.5 index

Nombre	**Tipo**	**Descripción**	**Hereda de…**
index		Inicializa y configura los módulos del servidor	
Responsabilidades			
Número	**Descripción**		
1	Inicializar los módulos		
2	Configurar el servidor		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Privado	Final	Module	express
Privado	Final	Module	getDBConnection
Privado	Final	Express	app
Privado	Final	String	NODE_ENV
Privado		String	DB_URI
Privado	Final	String	hostname
Privado	Final	String	port
Privado		Server	server
Observaciones			

9.5.2.6 ScannerViewModel

Nombre	**Tipo**	**Descripción**	**Hereda de…**
ScannerViewModel		Implementa la funcionalidad de escanear QRs y controla la lógica de presentación asociada a dicha funcionalidad	BaseViewModel
Responsabilidades			
Número	**Descripción**		
1	Controlar la lógica de presentación de la vista de escáner		
2	Registrar visitas a locales		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Privado	void	GetData	
Público | Asíncrono	void	ProcessCode	result : Result
Privado | Asíncrono	Task<Boolean>	ValidatePlace	placeId : String
Privado | Asíncrono	void	EnterPlace	placeId : String scanTime : DateTime
Privado | Asíncrono	void	ExitPlace	placeId : String scanTime : DateTime
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Público		String	ActionEnabled
Privado		String	_actionEnabled
Público		String	DoorSourceImage
Privado		String	_doorSourceImage
Público		Boolean	ScannerVisibility
Privado		Boolean	_scannerVisibility
Público		Boolean	IsInside
Privado		Boolean	_isInside
Público		Color	ScanButtonColor
Privado		Color	_scanButtonColor
Privado		CodeProcessor	codeProcessor
Privado		PlacesApiService	placesApiService
Privado		VisitsService	visitsService
Privado		String	currentVisitId
Público		ICommand	ActivateScanCommand
Público		ICommand	CloseAlertCommand
Observaciones			
No confundir alertas (mensajes popup de alerta) con alertas sanitarias (posibles contactos). En esta clase todas las menciones a alertas hacen referencia a los popups.			

9.5.2.7 CreateAlertViewModel

Nombre	**Tipo**	**Descripción**	**Hereda de…**
CreateAlertViewModel		Implementa el proceso de notificación de nuevos positivos	BaseViewModel
Responsabilidades			
Número	**Descripción**		
1	Controlar la lógica de presentación de la vista de notificación de nuevos positivos		
2	Registrar nuevos positivos como alertas de covid		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Privado | Asíncrono	void	CreateAlert	
Privado	Boolean	ValidateFields	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Público		String	Code
Privado		String	_code
Público		DateTime	SymptomsDate
Privado		DateTime	_symptomsDate
Público		Boolean	IsEntryEnabled
Privado		Boolean	_isEntryEnabled
Público		String	AlertIcon
Privado		String	_alertIcon
Público		Color	AlertColor
Privado		Color	_alertColor
Privado		VisitsService	visitsService
Privado		AlertsApiService	alertsApiService
Privado		CovidAlertsService	alertsService
Privado		EnvironmentVariablesService	environmentService
Público		ICommand	CreateAlertCommand
Público		ICommand	CloseAlertCommand
Observaciones			
No confundir alertas (mensajes popup de alerta) con alertas sanitarias (posibles contactos). Las alertas sanitarias se denominan CovidAlert, y se hablará de ellas como nuevos positivos.			

9.5.2.8 ShowAlertsViewModel

Nombre	**Tipo**	**Descripción**	**Hereda de…**
ShowAlertsViewModel		Implementa la funcionalidad de cambiar la frecuencia de sincronización y controla la sincronización manual	BaseViewModel
Responsabilidades			
Número	**Descripción**		
1	Refrescar manualmente la lista de alertas		
2	Cambiar la frecuencia de sincronización automática		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Privado | Asíncrono	void	GetData	
Privado | Asíncrono	void	RefreshList	
Privado	void	ChangeSyncFrequency	param : object
Privado	void	SelectSyncOption	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Público		List<CovidAlert>	Alerts
Privado		List<CovidAlert>	_alerts
Público		Boolean	IsRefreshing
Privado		Boolean	_isRefreshing
Público		Boolean	IsButtonEnabled
Privado		Boolean	_isButtonEnabled
Público		Boolean[]	SyncOptions
Privado		Boolean[]	_syncOptions
Público		String	SelectedOptionText
Privado		String	_selectedOptionText
Público		Boolean	HasAutoSync
Privado		Boolean	_hasAutoSync
Público		Boolean	InfoVisibility
Privado		Boolean	_infoVisibiliy
Público		String	InfoText
Privado		CovidContactService	contactService
Privado		ProcessAlertsService	processAlertsService
Privado		String[]	syncOptionsText
Público		ICommand	RefreshListCommand
Público		ICommand	ClosePopUpCommand
Público		ICommand	OpenPopUpCommand
Público		ICommand	CheckedCommand
Público		ICommand	InfoPopUpCommand
Observaciones			

9.5.2.9 ProcessAlertsService

Nombre	**Tipo**	**Descripción**	**Hereda de…**
ProcessAlertsService		Servicio encargado de obtener y almacenar las alertas sanitarias que puedan afectar al usuario	
Responsabilidades			
Número	**Descripción**		
1	Realizar llamadas a servicios para obtener las variables de contagio y las posibles alertas		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público | Asíncrono	Task<Integer>	Process	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Privado		AlertsApiService	alertsApiService
Privado		VisitsService	visitsService
Privado		EnvironmentVariablesService	environmentService
Privado		PlacesApiService	placesService
Privado		CovidContactService	contactService
Privado		CovidAlertsService	covidAlertsService
Observaciones			

9.5.2.10 NotificationManager (Implementación para Android)

Nombre	**Tipo**	**Descripción**	**Hereda de…**
NotificationManager		Esta clase genera las notificaciones push para Android informando sobre si se han encontrado nuevas alertas	INotificationManager
Responsabilidades			
Número	**Descripción**		
1	Crear las notificaciones push		
2	Enviar las notificaciones		
3	Recibir notificaciones push		
4	Mostrar las notificaciones		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	void	Initialize	
Público	void	ReceiveNotification	title : String message : String
Público	void	SendNotification	manualRefresh : Boolean title : String message : String notifyTime : DateTime?
Público | Asíncrono	void	Process	title : String message : String
Privado	void	Show	title : String message : String
Paquete	void	CreateNotificationChannel	
Paquete	void	GetNotifyTime	notifyTime : DateTime
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Privado		ProcessAlertsService	processAlertsService
Paquete	Final	String	channelId
Paquete	Final	String	channelName
Paquete	Final	String	channelDescription
Paquete		Boolean	channelInitialized
Paquete		Integer	messageId
Paquete		Integer	pendingIntentId
Paquete		Android.App.NotificationManager	manager
Público	Evento	EventHandler	NotificationReceived
Público	Estático	NotificationManager	Instance
Observaciones			
Esta clase implementa el patrón Singleton			

Capítulo 10. Desarrollo de las Pruebas

10.1 Pruebas Unitarias

Para desarrollar las pruebas unitarias del servidor se empleó la librería *Mocha*. Se trata de un paquete para Node.JS pensado para realizar todo tipo de pruebas en aplicaciones JavaScript. Para realizar las pruebas a la API, se ha complementado con el uso de otras dos librerías, *Supertest* y *Chai*, las cuales proporcionan características adicionales como aserciones más complejas.

Las pruebas unitarias se ejecutan de manera automática al subir cambios a la rama principal del proyecto del servidor, tal y como se explica en el apartado 8.6.1, gracias al sistema de CI/CD Azure Pipelines. Además de esto, es conveniente ejecutarlas antes en local con el fin de asegurarse de antemano que los cambios implementados funcionan correctamente, por lo que se podrán ejecutar estas pruebas también desde Visual Studio Code.

A continuación, se muestran dos capturas de pantalla en las que se aprecia el resultado de las pruebas unitarias en Visual Studio Code y el resultado de la compilación del servidor en Azure Pipelines.

```
  Alerts API tests
    √ BEFORE Initialize Alerts DB collection
(node:8400) DeprecationWarning: Listening to events on the Db class has been deprecated and will be removed in the next major version.
    √ POST Should not obtain any possible contact (1397ms)
    √ POST Should create a validated alert with some visits (225ms)
    √ POST Should create a non validated alert with some visits (116ms)
    √ POST Should not create an alert missing any field
    √ POST Should not create an alert whose visits miss fields
    √ POST Should create a new alert with an empty list of visits (121ms)
    √ POST Should obtain only the first alert as a possible contact (113ms)
    √ GET Should obtain the second and third alerts as not validated ones (110ms)

  The environment variables API
    √ POST Should log in with admin credentials
    √ POST Should allow to set a value for a new environment variable (220ms)
    √ POST Should not allow to set a non-numeric value for the variable (110ms)
    √ GET Should return the previously created variable (110ms)

  The admin login service
    √ POST Should log in with admin credentials
    √ GET Should access the panel with the generated token
    √ GET Should not access the panel with an invalid token
    √ GET Should not access the panel with an empty token
    √ POST Should not login with wrong username
    √ POST Should not login with wrong password

  Places API tests
    √ BEFORE Initialize DB collection
    √ POST Should add a place to the collection (224ms)
    √ POST Should not add a place with a non-number capacity (110ms)
    √ POST Should not add a place with missing fields (109ms)
    √ POST Should not add a place in the same address as the previous (110ms)
    √ POST Should add a place in another address (220ms)
    √ GET Should return an error if we dont provide an id to search
    √ GET Should return an error if the id does not match the required format
    √ GET Should return an error if the id does not exist
    √ GET Should return an error if we dont provide an id to search
    √ GET Should return an error if the id does not match the required format
    √ GET Should return an error if the id does not exist
    √ GET Should return an error if we dont provide a name
    √ GET Should return a list with the places matching the name provided (111ms)


  33 passing (4s)
```

Figura 10.1. Resultado de las pruebas unitarias

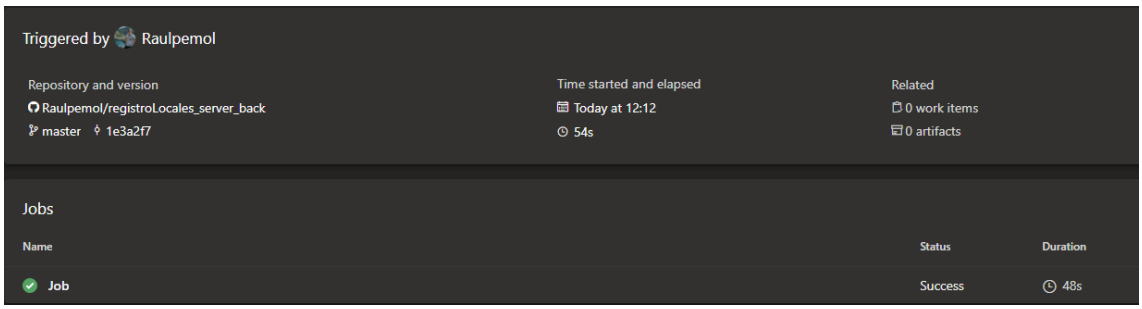

Figura 10.2. Resultado de la compilación en Azure Pipelines

10.2 Pruebas de Aceptación del Sistema

Las pruebas funcionales descritas en el capítulo 8 se ejecutaron numerosas veces, siempre que se modificaba alguna funcionalidad, o se añadían características nuevas a la aplicación móvil o a la aplicación web. Puesto que, hasta que el resultado de dichas pruebas no era el esperado no se cerraban las historias de usuario, finalmente todos los resultados obtenidos coinciden con los esperados.

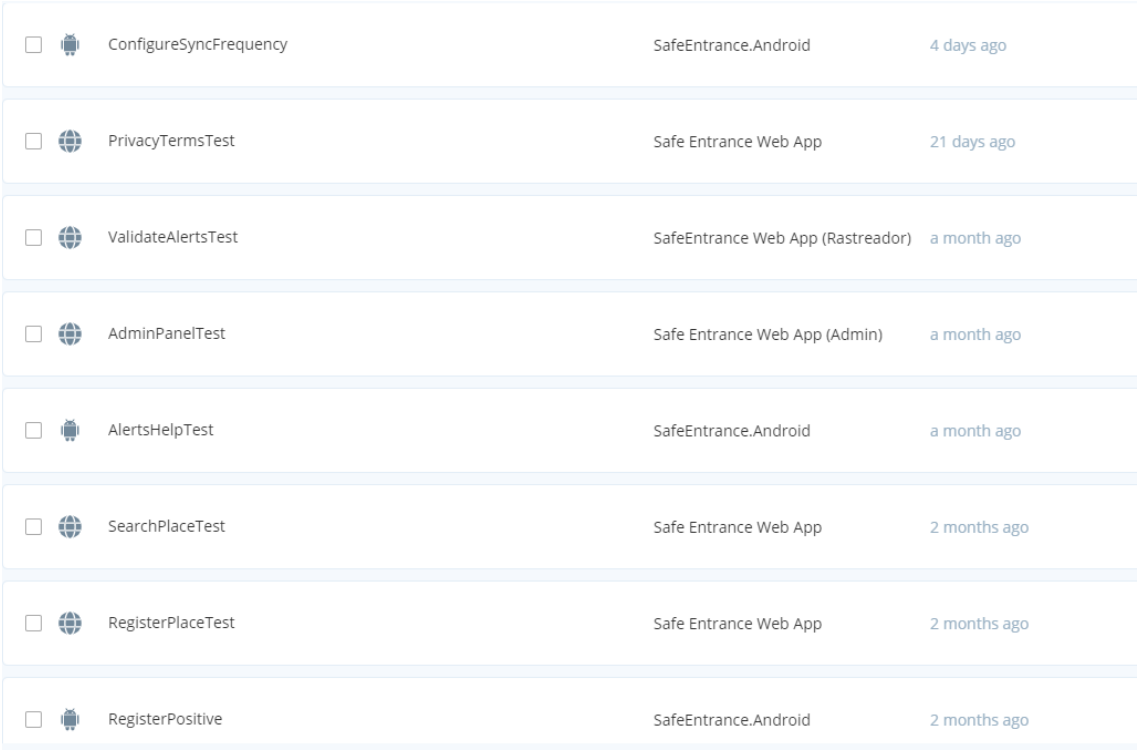

Figura 10.3. Batería de pruebas automatizadas con TestProject

10.3 Pruebas de Usabilidad

En el apartado 8.6.3 se definen los cuestionarios usados para evaluar la usabilidad del sistema en base a las respuestas de un conjunto de usuarios que realizaron las actividades guiadas que se especifican en dicho apartado. La muestra de población con la que se han realizado las pruebas de usabilidad está compuesta por las siguientes personas:

- 1 administrador de sistemas
- 1 dueño de un local de restauración
- 1 encargado de un centro social (a efectos prácticos cumple con el perfil de dueño de local)
- 2 personas entre 18 y 30 años
- 2 personas entre 30 y 50 años
- 2 personas entre 50 y 70 años
- 1 persona entre 70 y 100 años

A continuación, se muestran los resultados obtenidos en los cuestionarios y las conclusiones que se pueden extraer de ellos.

10.3.1 Resultado de las preguntas de carácter general

¿Usa un dispositivo móvil frecuentemente?
1. Todos los días – 90% 2. Varias veces a la semana – 0% 3. Ocasionalmente – 10% 4. Nunca o casi nunca – 0%
¿Qué tipo de actividades realiza más con el dispositivo móvil?
1. Es parte de mi trabajo o profesión – 0% 2. Lo uso principalmente para jugar y escuchar música – 0% 3. Empleo aplicaciones de mensajería instantánea y redes sociales – 100% 4. Únicamente leo el correo y navego ocasionalmente por internet – 0%
¿Ha usado alguna vez un sistema de rastreo para el COVID-19?
1. Sí, he utilizado Radar Covid – 20% 2. Si, pero no he utilizado Radar Covid – 0% 3. No, aunque me lo he planteado – 30% 4. No, nunca – 50%
¿Qué busca Vd. principalmente en una aplicación de rastreo de enfermedades?
1. Que sea fácil de usar – 10% 2. Que respete la privacidad de los usuarios – 40% 3. Que no interfiera en mi día a día – 10% 4. Que tenga todas las funciones necesarias para ser lo más efectiva posible – 40%

Al analizar las dos primeras respuestas de las preguntas del cuestionario general podemos darnos cuenta de que los dispositivos móviles forman una parte casi indispensable de la vida de toda la gente, siendo utilizados a diario por 9 de los 10 encuestados. El único que dijo usar su teléfono móvil solo ocasionalmente fue la persona del grupo de edad entre 70 y 100 años, lo cuál indica que, pese a que las nuevas tecnologías están menos integradas en la vida de los más mayores, también son usadas por este grupo poblacional. Todos los entrevistados, además, coincidieron en que usan los dispositivos móviles principalmente para acceder a redes sociales y utilizar aplicaciones de mensajería instantánea.

Observando los resultados de la tercera pregunta, se aprecia claramente como Radar Covid no tuvo un impacto muy grande en la población, habiendo sido utilizado por tan solo dos de los encuestados. Además, resulta interesante ver cómo únicamente tres de los ocho restantes se plantearon en algún momento su uso, para finalmente descartarlo.

Por último, la cuarta pregunta cuenta con más variedad de respuestas, aunque hay dos que son claramente mayoritarias. La facilidad de uso y que la aplicación no interfiera en el día a día de las personas solamente fueron votadas como característica principal de una aplicación de rastreo por una persona cada una. Por el contrario, que sea lo más efectiva posible y que respete la privacidad del usuario son las características que en más consideración tienen los potenciales usuarios de cara al uso de una aplicación de este tipo.

Tras analizar estas respuestas, queda aún más claro que los dispositivos móviles forman parte de la vida de todas las personas, y que a la hora de desarrollar una aplicación de rastreo es sumamente importante cuidar la privacidad del usuario al mismo nivel que la efectividad del sistema. Finalmente, parece necesario realizar un proceso de reeducación sobre la población si se pretende el uso masivo de este tipo de aplicaciones, puesto que la tendencia actual es a rechazarlas.

10.3.2 Resultados de las preguntas cortas sobre usabilidad

Facilidad de Uso de la Aplicación Web	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la página?	70%	30%	0%	0%
¿Existe ayuda para las funciones en caso de que tenga dudas?	0%	0%	0%	100%
¿Le resulta sencillo el uso de la aplicación web?	90%	10%	0%	0%
Funcionalidad de la aplicación Web	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. espera?	80%	20%	0%	0%
¿El tiempo de respuesta de la página es muy grande?	0%	0%	0%	100%
Facilidad de Uso de la Aplicación Móvil	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la	90%	10%	0%	0%

aplicación?				
¿Existe ayuda para las funciones en caso de que tenga dudas?	10%	30%	60%	0%
¿Le resulta sencillo el uso de la aplicación?	100%	0%	0%	0%
Funcionalidad de la Aplicación Móvil	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. espera?	70%	30%	0%	0%
¿El tiempo de respuesta de la aplicación es muy grande?	0%	0%	30%	70%
Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es	0%	90%	10%	0%
Los iconos e imágenes usados son	30%	70%	0%	0%
Los colores empleados son	40%	60%	0%	0%
Diseño de la Interfaz Móvil		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?		100%	0%	0%
¿El diseño de las pantallas de la aplicación es claro y atractivo?		80%	0%	20%
¿Cree que la aplicación está bien estructurada?		100%	0%	0%
Diseño de la Interfaz Web		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?		100%	0%	0%
¿El diseño de las páginas es claro y atractivo?		40%	10%	50%
¿Cree que la página web está bien estructurada?		80%	0%	20%
Observaciones				
- El pie de página lo prefería visible sin necesidad de desplazarme hacia abajo en la página web. - Esperaba que el lector de QRs se activara solo. - No pensaba que hubiera que escanear el código también para salir. - Es un poco raro que la pantalla de inicio sea el formulario para registrar locales, estaría mejor si fuera una pantalla de información general.				

Una vez recopiladas todas las respuestas al cuestionario, podemos concluir que, si bien las aplicaciones son ambas fáciles de usar, están bien estructuradas y por lo general tanto sus funcionalidades como los elementos gráficos son adecuados, existen ciertos puntos susceptibles de mejora.

Uno de ellos es la existencia de ayuda en la página web. Pese a que no fue necesaria para la mayoría de los usuarios, uno de ellos remarcó que le hubiera gustado que la pantalla de inicio contuviera información general acerca del uso y funcionamiento tanto de la página como de la aplicación móvil. Puesto que dicha propuesta pareció interesante, se considerará desarrollarla en un futuro.

Otro punto a tener en cuenta es que las fuentes empleadas en la aplicación móvil, pese a no suponer ningún problema para el público general, los usuarios de mayor edad o con peor vista necesitaron de más tiempo para leer la ayuda del formulario de registro de nuevo positivo. Se podría considerar rediseñar esta pantalla o tratar de adaptar el tamaño de la fuente.

Capítulo 11. Manuales del Sistema

11.1 Manual de Instalación

11.1.1 Instalación del servidor

El servidor backend del sistema se encuentra desplegado continuamente en Azure App Service, por lo que no sería necesario instalarlo en una máquina local. De todas maneras, en caso de que se quiera desplegar en local con el fin de probar alguna funcionalidad o realizar cambios en dicho servidor, existe la posibilidad de hacerlo, previa consulta con el equipo de desarrollo para obtener acceso al repositorio en el que se encuentra el código fuente.

Una vez se cuenta con dicho acceso, lo primero que se ha de hacer es descargarse el código fuente, ya sea mediante un archivo comprimido desde GitHub o clonando en local el repositorio. Se recomienda consultar la siguiente guía, en caso de no tener conocimientos sobre GIT, si se desea clonar el repositorio en local: Cómo clonar repositorios con GitHub.

Una vez se tiene el código fuente en la máquina desde la que se va a desplegar, se ha de abrir una consola de comandos (CMD, PowerShell, o similar), en la cual se escribirá el siguiente comando para instalar los paquetes y librerías necesarias para compilar el proyecto.

```
npm install
```

Tras introducir este comando se podrá ya desplegar el servidor en la máquina local. El comando que realizará el despliegue es el siguiente:

```
npm run start
```

Tras ejecutar dicho comando se habrá desplegado el servidor en local, y se podrá acceder a él a través de la ruta http://localhost:8080.

11.1.2 Instalación de la aplicación web

Del mismo modo que el servidor, la aplicación web se encuentra desplegada en Azure, en este caso en Azure Web Storage. Por este motivo, no es necesario instalar nada en la máquina desde la que se desea acceder a ella.

En caso de querer desplegarla en local, al igual que el servidor, los pasos a seguir para obtener el código fuente e instalar los paquetes y librerías necesarias para su despliegue son los ya descritos en el apartado 11.1.1.

Una vez se cuenta con el código fuente y han sido instalados todos los paquetes necesarios, se puede desplegar la versión local de la página web a través del siguiente comando:

```
npm run start
```

En este caso, tras ejecutar el anterior comando, la página web será accesible a través de la ruta http://localhost:3000.

11.1.3 Instalación de la aplicación móvil

Cuando la aplicación se encuentre en producción, la instalación de la aplicación se realizará a través de los mercados de aplicaciones Play Store y App Store, para dispositivos Android y iOS respectivamente. Instalar una aplicación desde estas plataformas es muy sencillo, puesto que bastará con buscarla y pulsar sobre el botón de "Instalar".

Hasta que el sistema esté implantado de manera oficial, la instalación de la aplicación se tendrá que realizar a través de fuentes desconocidas para los sistemas operativos. Para que un dispositivo móvil permita la instalación de aplicaciones a través de fuentes desconocidas, se han de seguir los siguientes pasos:

- Android: En la aplicación de Ajustes, entrar en la opción de Seguridad. Una vez dentro, buscar la opción "Orígenes desconocidos" o similar (el nombre dependerá de la capa de personalización que utilice el dispositivo concreto), y activar el interruptor correspondiente. Tras haber hecho esto, se ha de descargar el archivo APK en el terminal móvil, y al pulsar sobre él comenzará su instalación.
- iOS: En dispositivos iOS no es posible instalar aplicaciones de terceros sin hacer uso de software de terceros poco fiable.

11.2 Manual de Ejecución

Tras realizar los pasos descritos en el manual de instalación, no será necesario realizar ninguna acción para ejecutar el servidor y la aplicación web, puesto que ya se encontrarán desplegadas, tanto en Azure como en local si así lo quiso el usuario. Para detener tanto el servidor como la aplicación web desplegados en local, bastará con enviar la combinación de teclas *CTRL + C* a la consola desde la que se desplegó cada uno de los subsistemas, y escribir la letra *S* para confirmar que se desea parar la ejecución.

En el caso de la aplicación móvil, una vez instalada, el usuario tan solo ha de pulsar sobre su icono creado en el menú de aplicaciones para que se ejecute la misma. Al igual que para el resto de las aplicaciones móviles, esta se cerrará cuando el usuario salga de ella con la tecla "Atrás".

11.3 Manual de Usuario

11.3.1 Manual de usuario de la aplicación móvil

El propósito de esta aplicación es que todos los usuarios que dispongan de ella en su dispositivo móvil puedan utilizar el sistema de rastreo para acceder a cualquier local que este registrado en él de manera segura, sabiendo que en caso de que exista la posibilidad de que coincidan con un positivo, se enterarán lo antes posible para poder tomar las precauciones necesarias.

Cuando se abre por primera vez la aplicación, esta nos pedirá permiso para poder acceder a la cámara del dispositivo. Con el fin de poder utilizar el lector de QRs para acceder a los locales, es imprescindible otorgarle dicho permiso, que podrá ser revocado en cualquier momento desde el menú de Ajustes del sistema operativo. Tras esto, tal y como se muestra en la figura 11.1, la aplicación mostrará la ventana de términos y condiciones de uso. Es muy importante leer bien esta información para poder conocer cómo funciona la aplicación y cuáles son los derechos de todos sus usuarios. Una vez se haya leído y entendido, para poder utilizar la aplicación se deben aceptar dichos términos pulsando en el botón de “Aceptar y continuar”.

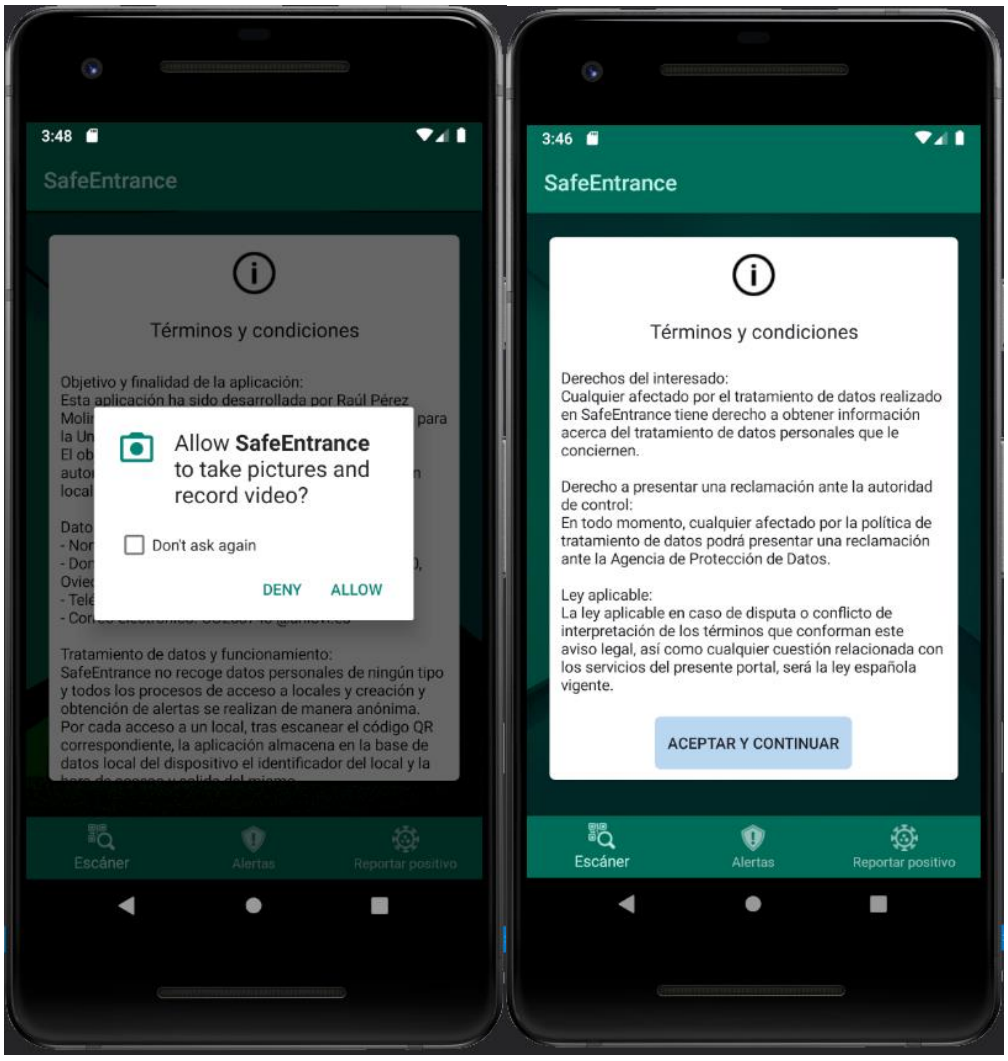

Figura 11.1. Aceptación de permisos y condiciones de uso

Tras aceptar los términos y condiciones, se podrá utilizar ya la aplicación. Como muestran las imágenes, en la parte inferior de la pantalla se encuentra el menú de navegación, desde el que se accederá a las tres pantallas de la aplicación. Por defecto, la pantalla de inicio es la del escáner de códigos QR.

Esta pantalla cuenta con dos elementos, el lector QR y un botón. El primero, situado en el centro de la pantalla, estará desactivado por defecto, y en su lugar habrá una imagen de una puerta. Si la puerta está cerrada, significa que aún no se ha entrado a ningún local. En caso de que se muestre una puerta abierta, esto supondrá que nos encontramos dentro de un local.

Para encender y apagar el escáner QR hay que pulsar sobre el botón rectangular situado debajo de dicho escáner o de la imagen de la puerta. Este botón, acompañando a dicha imagen, indicará también que acción se va a realizar al escanear el próximo código. Si nos encontramos fuera de un local, el botón tendrá un color amarillo, y mostrará el mensaje "Entrar a un local". Una vez dentro, el botón cambiará de color y de texto, pasando a ser rojo y a indicarnos que la acción a realizar es "Salir del local". En la siguiente imagen se muestra el aspecto de esta pantalla en las dos situaciones mencionadas anteriormente.

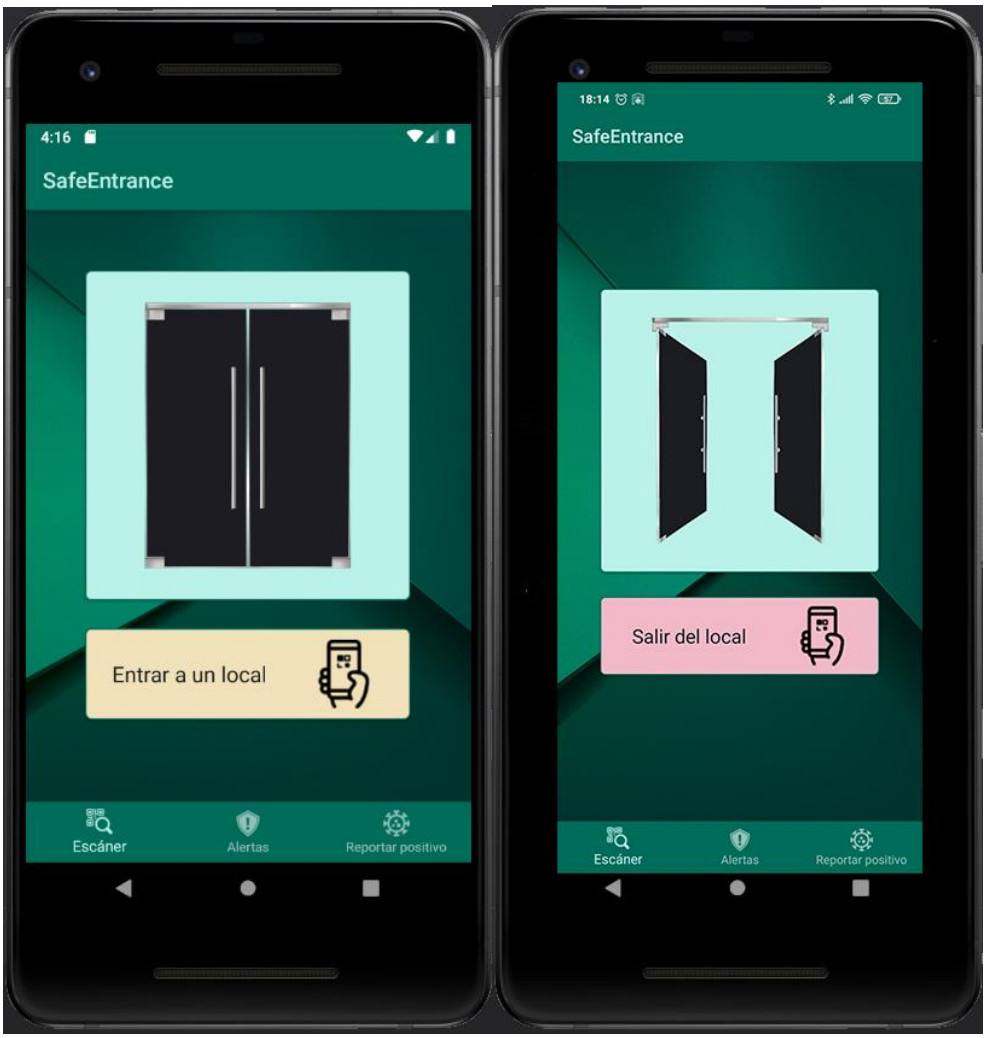

Figura 11.2. Estados de la pantalla del escáner

La segunda pantalla de la aplicación corresponde al listado de alertas de posibles contactos con positivos. Desde esta pantalla se podrá conocer en todo momento si se ha coincidido en algún local con un usuario que posteriormente haya dado positivo en coronavirus. En esta pantalla se mostrará una lista con todos estos posibles contactos. En caso de aparecer alguno nuevo, es primordial ponerse en contacto con un centro de salud con el fin de seguir los protocolos de seguridad indicados por las autoridades.

La aplicación actualizará periódicamente esta lista en busca de nuevas alertas, sin embargo, existe la posibilidad de refrescarla de manera manual en cualquier momento. Para ello, se ha de deslizar hacia abajo con el dedo en la lista. Al hacer este gesto, se mostrará un icono de cargando, y al finalizar la sincronización llegará una notificación indicando si se han encontrado o no nuevas alertas. En la imagen siguiente se muestra el aspecto de esta pantalla cuando no se han encontrado alertas, y a su lado se enseña un ejemplo de alerta. Por cada una de estas alertas la aplicación muestra el lugar en el que se produjo el posible contacto y la fecha a la que comenzó.

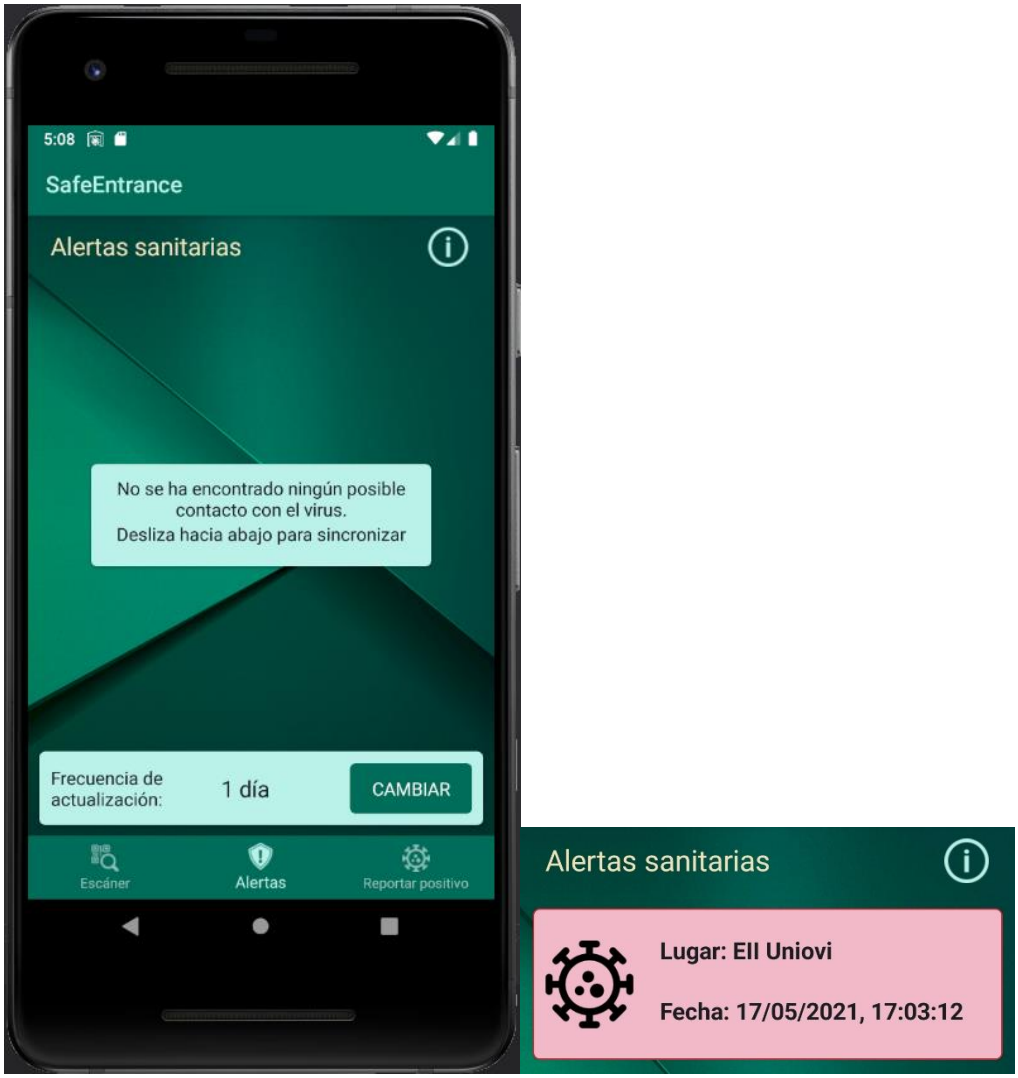

Figura 11.3. Listado de alertas vacío y ejemplo de alerta

Tal y como se puede apreciar en la figura 11.3, la pantalla cuenta también con un botón de información. Pulsaremos en él si no se tiene claro que considera la aplicación alerta, y se abrirá una ventana con una breve explicación.

La sección situada en la parte inferior de la pantalla, justo encima de la barra de navegación solamente está disponible en los dispositivos Android. En ella se muestra la frecuencia actual con la que la aplicación realiza la actualización automática de la lista de alertas. Esta actualización automática se realizará siempre que la aplicación esté en primer o en segundo plano. Además, en caso de que se cierre la aplicación, pero no decidamos liberar la memoria que se usa para ella, también seguirá actualizándose con la misma frecuencia. Para liberar dicha memoria y cancelar la sincronización automática accederemos a la bóveda de aplicaciones en ejecución y eliminaremos la aplicación. Para cambiar la frecuencia con la que se realiza la sincronización, pulsaremos en el botón “CAMBIAR”, y se abrirá una ventana con cuatro opciones: 1 hora, 5 horas, 12 horas y 1 día. Tras seleccionar la deseada, cerraremos la ventana pulsando el botón “OK” o la tecla “Atrás”. En dispositivos iOS, dado que el sistema operativo se encarga de decidir que frecuencia de actualización es la más optima, esta sección estará deshabilitada.

Por último, la tercera pantalla de la aplicación está destinada a que podamos notificar un resultado positivo en una prueba de COVID-19. En esta pantalla se muestra un breve formulario en el que se habrá de indicar el código de rastreo asociado a la prueba realizada y la fecha de aparición de los primeros síntomas. El código de rastreo lo proporcionarán las autoridades sanitarias en el momento de comunicar el resultado de la prueba realizada. En caso de que seamos asintomáticos, la fecha a introducir será la de realización de la prueba. A continuación, se muestra como es está pantalla.

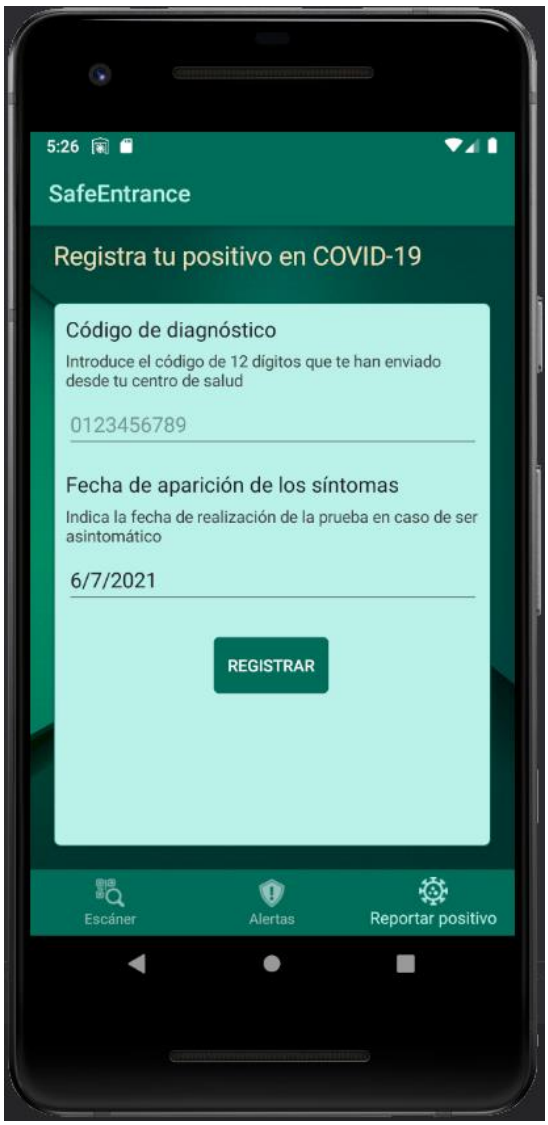

Figura 11.4. Formulario de registro de positivo por coronavirus

11.3.2 Manual de usuario de la aplicación web

El objetivo de esta aplicación web es facilitar el registro de nuevos locales en el sistema, proporcionar a los usuarios un portal desde el que poder buscar si un establecimiento utiliza este sistema de rastreo, y a la vez dar al administrador del sistema una interfaz cómoda desde la que poder modificar el valor de las variables de contagio del coronavirus en caso de que desde la comunidad científica se descubran nuevos datos sobre la enfermedad. Este manual, por tanto, se divide en dos secciones, una destinada a la web accesible para todos los usuarios y otra para las funciones de administrador.

11.3.2.1 Funciones estándar

Al acceder al Portal Web veremos que la página tiene la siguiente estructura: Cabecera – Menú – Contenido – Pie de página. En la cabecera se muestra el nombre del sistema de rastreo, y pulsando sobre él en cualquier pantalla navegaremos a la pantalla de inicio.

El menú de navegación cuenta con dos opciones: Registrar un local y Buscar un local. La primera de ellas está destinada a todos aquellos usuarios que sean propietarios de un establecimiento y quieran darlo de alta en el sistema. Para hacer esto, se deberá rellenar el formulario con los datos solicitados. Una vez rellenado, haremos click en el botón "Generar QR" para proceder a la generación del código que identificará al local en el sistema. Este código se deberá colocar en las entradas y salidas de local para que los clientes que accedan a él lo escaneen con la aplicación tanto a la entrada como a la salida. Tras darle al botón de "Generar QR", se mostrará el código generado y un botón que permitirá descargarlo en formato PNG. También se mostrará otro botón para volver al inicio.

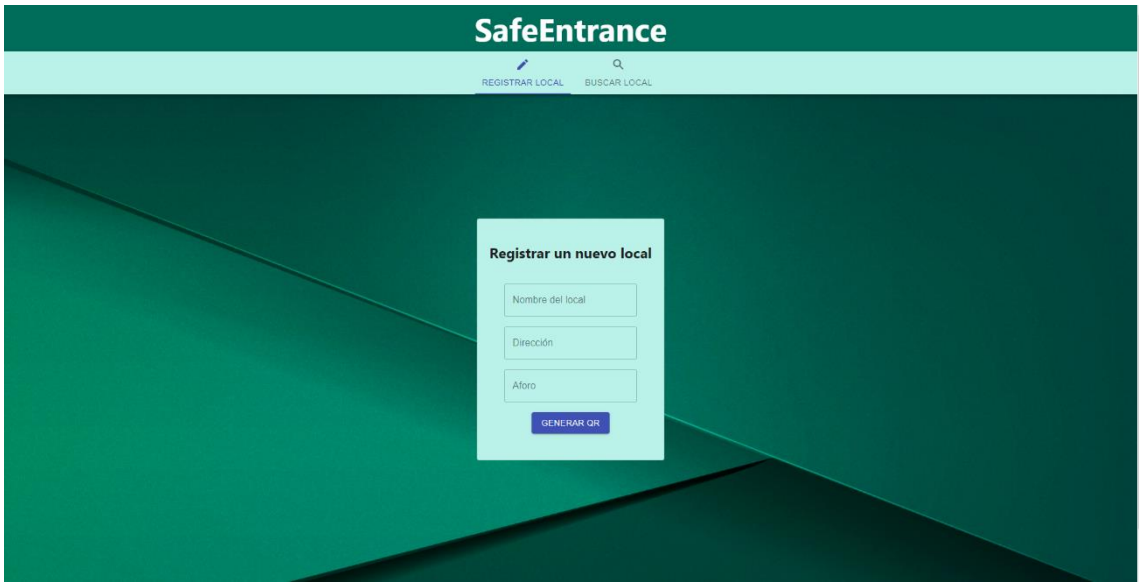

Figura 11.5. Aspecto del formulario de registro de locales

La segunda sección del menú de la página permite buscar locales a partir de su nombre. Cuando queramos conocer si un establecimiento utiliza el sistema de rastreo buscaremos su nombre. En caso de que haya algún local registrado con dicho nombre, se mostrarán sus datos (nombre, dirección y aforo máximo).

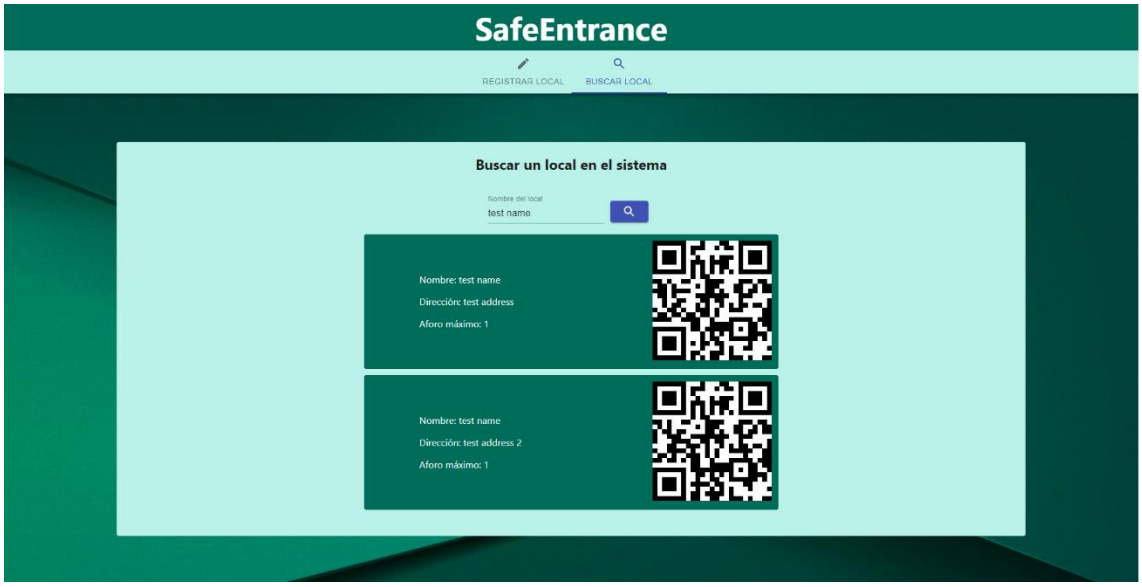

Figura 11.5. Aspecto de la sección de búsqueda de locales

Para cumplir con la ley de protección de datos vigente y para informar sobre el tratamiento de datos que se realiza en la página, desde el pie de página se podrá acceder mediante un enlace a una explicación de la política de privacidad del sistema.

11.3.2.2 Funciones de administrador

Para acceder al panel de control es necesario antes iniciar sesión en la aplicación. Para ello hemos de acceder al subdominio **/login**. En dicha dirección hay un formulario de inicio de sesión en el que introduciremos nuestras credenciales de administrador, y accederemos al panel de control de las variables de contagio. Dicho panel cuenta con un campo para cada variable, y en caso de querer actualizar el valor de alguna de ellas, modificaremos el valor actual y pulsaremos sobre el botón "Actualizar" asociado a la variable que acabamos de modificar. Si se actualizó correctamente el valor, la aplicación mostrará un mensaje indicándolo.

Es importante tener en cuenta que el acceso al panel y, por tanto, la capacidad para actualizar el valor de las variables dura tan solo 5 minutos por motivos de seguridad. Por ello, cuando finalice el periodo de 5 minutos la página pedirá de nuevo que se introduzcan las credenciales para poder manipular el panel de control. A continuación, se muestra el aspecto de dicho panel.

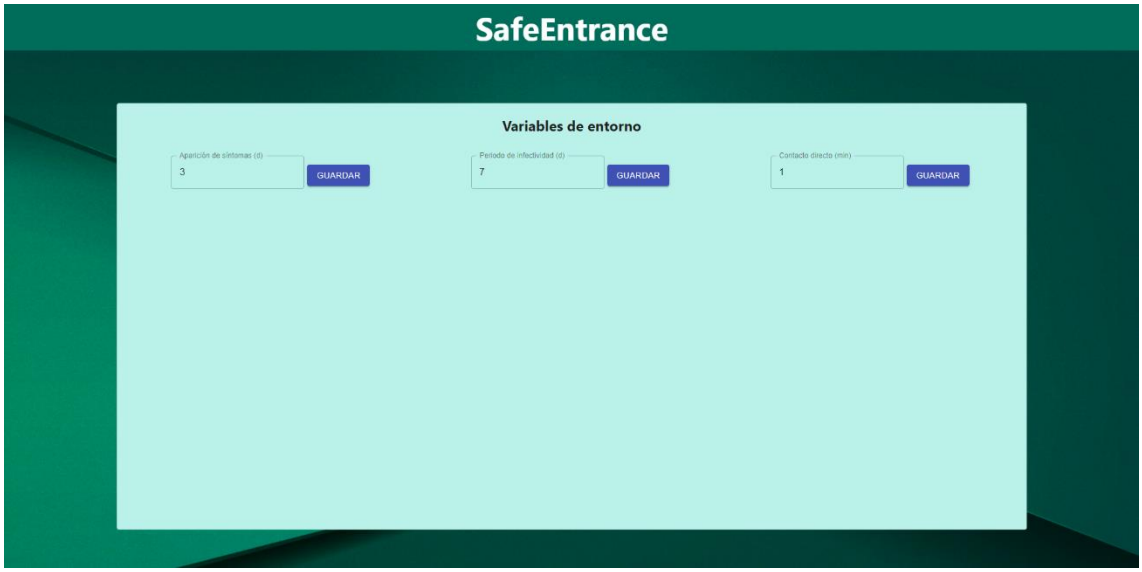

Figura 11.6. Aspecto del panel de control de variables

Cada vez que se cargue el panel se mostrarán los valores presentes en el sistema para cada variable.

11.4 Manual del Programador

11.4.1 Realizar y probar ampliaciones en el servidor

Para realizar cualquier ampliación o modificación de las funcionalidades provistas por la API REST es necesario conocer la arquitectura y los módulos descritos en el capítulo 8 de este documento. Una vez se tenga claro esto, se podrán añadir nuevos métodos a la API en cualquiera de los módulos, así como crear módulos nuevos que contengan métodos referidos a otra entidad o funcionalidad del sistema. También se podrán crear nuevas entidades mediante esquemas de la librería *Mongoose*.

Con el fin de agilizar el desarrollo y la prueba de nuevas funcionalidades en la API, el servidor cuenta con el módulo *Nodemon* instalado, el cual permite desplegar el servidor en local y redesplegarlo cada vez que se guardan cambios en algún archivo. Se recomienda trabajar de esta manera, con el servidor activo siempre en la máquina para poder llevar a cabo un desarrollo más ágil. Para desplegar el servidor con *Nodemon* se ha de ejecutar el siguiente comando:

```
npm run server
```

Se recomienda encarecidamente el uso de la herramienta *Postman* para realizar peticiones a la API REST y comprobar su funcionamiento a medida que se van desarrollando e implementando funcionalidades.

11.4.2 Estructura de la solución en Xamarin.Forms y aspectos a tener en cuenta para ampliar la aplicación móvil

La solución de la aplicación móvil, debido a que el objetivo de Xamarin.Forms es que se reutilice la mayor cantidad de código posible para Android y iOS, cuenta con tres proyectos distintos que se compilarán de manera conjunto para generar ejecutables para Android o iOS.

El primero de estos proyectos lleva el nombre de la aplicación, y es el que contiene la base de código común de la aplicación. La arquitectura y la estructura de paquetes que tiene este proyecto están definidas en el capítulo 8. Prácticamente cualquier modificación que se quiera hacer en las funcionalidades existentes se hará en este proyecto, y siempre que se quiera implementar una nueva vista se creará un archivo XAML en este proyecto.

Sin embargo, cabe la posibilidad de que algunas ampliaciones requieran de la implementación de código específico para cada plataforma, como fue el caso del sistema de notificaciones y sincronizaciones en segundo plano. En este caso, se ha de acudir a los proyectos con sufijos ".Android" y ".iOS", y crear en ellos las clases correspondientes. Para poder llamar a estas clases desde el proyecto principal, se ha de hacer uso del servicio de inyección de dependencias de Xamarin.Forms. Para ello bastará con crear una interfaz en el proyecto

principal, que ha de ser implementada por las clases específicas de cada plataforma. A través de este servicio de inyección de dependencias, desde el proyecto principal se llamará a los métodos de dicha interfaz, y en tiempo de compilación, según la plataforma de destino, se utilizará una clase u otra.

Por último, es importante tener en cuenta que, a la hora de agregar recursos al proyecto (imágenes, iconos, etcétera) se han de añadir a los proyectos específicos para cada plataforma. En el caso del proyecto de Android bastará con añadir los archivos correspondientes a la carpeta *Resources*, mientras que para iOS habrá también que crear una nueva entrada en el catálogo de activos por cada recurso a añadir. Para hacer esto, se tiene que abrir el archivo *Assets.xcassets* del proyecto de iOS, y ahí dentro crear un nuevo recurso del tipo deseado. El nombre que se le debe dar a dicho recurso ha de ser el mismo que tiene ese recurso en el proyecto de Android (sin extensión).

Capítulo 12. Conclusiones y Ampliaciones

12.1 Conclusiones

El objetivo de este proyecto era diseñar e implementar un sistema de rastreo que sirviera como ejemplo de cómo se pueden plantear este tipo de aplicaciones sin necesidad de realizar un tratamiento de datos invasivo, pero a la vez manteniendo la efectividad necesaria para controlar la propagación del virus. A lo largo de estos meses se ha conseguido llevar a cabo dicho sistema, aprendiendo mucho por el camino.

La idea de utilizar Xamarin.Forms para el desarrollo de la aplicación móvil, además de por los criterios explicados a lo largo del documento, se llevó a cabo con el fin aprender más acerca del desarrollo multiplataforma para dispositivos móviles, puesto que es un campo que no se abarca de manera exhaustiva durante el grado y consideré interesante profundizar en él. Además, la creación de una aplicación web completa (*backend* y *frontend*) con una arquitectura basada en microservicios me permitió afianzar los conocimientos adquiridos durante acerca de sistemas distribuidos. Si a esto le sumamos el uso de metodologías ágiles o integración y despliegue continuos, por mencionar un par de conceptos, considero que la realización de este proyecto no solo me ha servido a nivel académico para consolidar lo aprendido durante el grado, sino que también creo que me aporta un gran valor desde un punto de vista profesional, puesto que muchas de las tecnologías utilizadas en este proyecto están muy demandadas por la industria.

Desde un punto de vista del producto obtenido, pese a estar también bastante satisfecho con el resultado, me quedo con ganas de haber realizado algunas ampliaciones que detallaré en el próximo apartado.

12.2 Ampliaciones

12.2.1 Compilación del proyecto y realización de pruebas funcionales en dispositivos iOS

Como se explicó en el apartado 9.5.1, uno de los problemas encontrados en el proyecto fue la necesidad de un dispositivo Mac OS para poder compilar la aplicación móvil para iOS. Debido a que no se dispuso de un ordenador con dicho sistema operativo, una de las ampliaciones propuestas, y la de mayor prioridad, es compilar la aplicación para iOS y realizar las pruebas funcionales en dichos dispositivos.

12.2.2 Integración con los sistemas informáticos sanitarios actuales

Para que el sistema de rastreo desarrollado en este proyecto se pueda poner en marcha es necesario que exista alguna manera de validar que los códigos sanitarios introducidos por los usuarios a la hora de notificar su positivo por coronavirus son correctos. Con el objetivo de minimizar los esfuerzos necesitados para realizar dichas validaciones, lo que se propone en este proyecto es reutilizar los sistemas que validan los códigos que introducen los usuarios de Radar Covid para notificar sus positivos. De este modo se evitaría tener que diseñar una interfaz de comunicación entre la nueva aplicación y los sistemas actuales, pudiendo crear un servicio en la aplicación móvil para reaprovechar los mecanismos existentes. Una vez que se haya hecho esto, se podrá eliminar la funcionalidad de prueba que permite validar o eliminar nuevas alertas desde la página web.

12.2.3 Añadir campos para validar la propiedad del local en el formulario de registro de locales

Con el fin de validar que la persona que registra un local en el sistema es el dueño de dicho establecimiento o, en su defecto, alguien en quién se ha delegado para hacerlo, será necesario que, previo paso a la puesta en producción del sistema, se implemente algún método para verificar esta situación. Esta característica se consideró que estaba fuera del alcance de este proyecto y por ese motivo se ha decidido dejarla como ampliación.

12.2.4 Creación de una pantalla de información en la página web

En una de las últimas pruebas de usabilidad realizadas con potenciales usuarios, se sugirió que la aplicación web contase con una página de inicio en la que, de manera gráfica, a través de imágenes e iconos acompañados de textos cortos, se explicase al usuario el funcionamiento del sistema, sus ventajas y un pequeño manual de uso. Pese a tratarse de una buena idea, no fue incluida en el backlog de producto debido a que se consideró más bien competencia de un diseñador gráfico que de un ingeniero informático. Por este motivo, la futura ampliación tendrá como objetivo diseñar una serie de viñetas e imágenes a través de las cuales poder explicar a los usuarios como funciona el sistema de rastreo y cuales son sus ventajas, con el objetivo de incluirlas en una nueva pantalla de la aplicación web que se mostraría por defecto en lugar del formulario de registro de locales. Esta nueva página de inicio tendría también su propio botón en el menú de navegación de la página web.

Capítulo 13. Presupuesto

Para el desarrollo de este proyecto, el equipo encargado de llevarlo a cabo estará formado una única persona con conocimientos de ingeniería de software y sistemas informáticos. Para poder completar el desarrollo del sistema, este ingeniero informático deberá tener conocimientos principalmente de C#, desarrollo de aplicaciones móviles, React.JS y Azure. Realizando un estudio de mercado, podemos apreciar que el salario medio de un ingeniero con al menos un año de experiencia en las tecnologías mencionadas previamente es el que se muestra en la siguiente tabla. Para llevar a cabo dicho estudio y obtener un salario medio para un ingeniero con dichos conocimientos se han usado el portal de empleo *Glassdoor* y la calculadora de salarios de *Stack Overflow*.

Trabajador	Sueldo bruto mensual (€)	Productividad (%)	Coste directo mensual (€)	Coste indirecto mensual (€)
Ingeniero informático	2.800,00	80%	2.240,00	560,00

En la tabla se hace diferencia entre el coste directo e indirecto del trabajador. Esto se debe a que, pese a ser un perfil técnico, al ser el único miembro del equipo, parte de su tiempo tendrá que ser empleado en reuniones y realización de la documentación del proyecto. Por este motivo, se calcula que el tiempo que podrá dedicar íntegramente al desarrollo del proyecto es un 80% del total.

A continuación, se muestran como parte de los costes indirectos del proyecto, todos aquellos materiales y licencias que son necesarios para completarlo y posteriormente ponerlo en producción. Pese a que para la realización de este proyecto se han utilizado en su mayoría licencias y planes gratuitos, en caso de ser finalmente comercializado este producto sería necesario adquirir licencias profesionales de ciertas herramientas, así como planes de Azure y MongoDB de mayores capacidades. Por este motivo, los cálculos de presupuesto que se realizan en este capítulo tendrán en cuenta los costes de dichas licencias y servicios de pago con el fin de realizar una estimación más ajustada a la realizada de los costes de implantar un sistema de rastreo de estas características.

Recurso	Precio	Cantidad	Total al mes (€)
Gastos de internet	34,90 €/mes	1	34,90
Equipo para el desarrollo	970,00 €	1	20,20
Dispositivo Android para pruebas	200,00 €	1	5,55
Visual Studio Professional (Incluye Azure DevOps)	45,00 €/mes	1	45,00
MongoDB Atlas M40 Dedicated Cluster	1,04 €/h	1	748,80
Azure App Service P3V3	418,61 €/mes	1	418,61

Equipo Mac OS	1.200,00€	1	16,66
Licencia para publicar aplicaciones en iOS App Store	99,00 €/año	1	8,25
Cuenta de desarrollador para publicar aplicaciones en Play Store	25,00 €	1	25,00

En la tabla anterior se puede apreciar como los costes de los equipos de desarrollo y pruebas han sido amortizados con el fin de obtener su precio mensual. Para ello, se ha estimado que la vida útil de un ordenador como el utilizado es de 4 años, mientras que el periodo de amortización del dispositivo Android es de 3 años. Para el equipo Mac, dado que no se va a usar como equipo de desarrollo, sino como herramienta para compilar la aplicación par iOS, se amortizará en 6 años, puesto que sus capacidades productivas no serán tan importantes y por ende podrá alargarse su vida útil. En el caso de la cuota para publicar una aplicación en el mercado Play Store, debido a su bajo coste y su pago único, se realizará una única vez, sin ser fraccionado y sin necesidad de calcular una amortización a largo plazo.

Los costes mensuales (sin contar las licencias para publicar aplicaciones en App Store y Play Store) obtenidos a partir de los datos indicados anteriormente son los siguientes:

Costes directos (€)	Costes indirectos	Total (€)
2.240,00	1.849,72	4.089,72

A la hora de calcular los presupuestos finales, se desglosarán en tres partidas. Estas partidas, tal y como se explica en el apartado 5.2, no especificarán las tareas a realizar y sus costes, debido a que no se conocen de antemano. Sin embargo, en base a la duración de las etapas del proyecto y a los costes mensuales calculados anteriormente, se puede dar un presupuesto independiente para cada una de ellas. De esta manera, se podrá gestionar el coste del proyecto en base a su avance temporal, y se tendrán puntos de referencia en los que se podrá comparar el avance real con los costes presupuestados.

Partida	Coste (€)
Etapa inicial	4.089,72
Desarrollo del primer prototipo del proyecto	14.041,37
Ampliación y mantenimiento del sistema	49.200,64
TOTAL	**67.331,73 €**

A la hora de calcular el presupuesto de estas partidas, tan solo es necesario multiplicar el coste mensual del proyecto por la duración planificada de cada etapa, de acuerdo con la planificación descrita en el capítulo 5. En el caso de la fase de ampliación y mantenimiento, pese a que no se contempla en este documento, dicha etapa se llevará a cabo una vez se haya presentado la primera versión del sistema, desarrollada hasta el día 13 de junio. En ella, en caso de que se quiera implantar el producto presentado, se realizarán las ampliaciones necesarias para llevarlo a cabo y se continuará con el mantenimiento del sistema siguiendo la

misma metodología que en el desarrollo inicial. La cantidad presupuestada para dicha etapa de ampliación corresponde al periodo de un año trabajando en el sistema más el coste de las licencias de Play Store y App Store.

En cuanto a la diferenciación entre presupuesto de costes y presupuesto de cliente, debido a que el proyecto está pensado para ser llevado a cabo por una única persona en lugar de por una empresa, el beneficio que se pretende obtener de su realización equivale al salario del trabajador.

Capítulo 14. Referencias Bibliográficas

14.1 Libros y Artículos

[MinisterioDeSanidad20] "Evaluación del riesgo de la transmisión de SARS-CoV-2 mediante aerosoles. Medidas de prevención y recomendaciones". Ministerio de Sanidad, Gobierno de España. 2020.

[Jones20] Nicholas R. Jones, Zeshan U. Qureshi, Robert J. Temple, Jessica P. J. Larwood, Trisha Greenhalgh, Lydia Bourouiba. "Two metres or one: what is the evidence for physical distancing in covid-19?". BMJ. 2020. doi: https://doi.org/10.1136/bmj.m3223

[He20] He, X., Lau, E.H.Y., Wu, P. "Temporal dynamics in viral shedding and transmissibility of COVID-19". Nature Medicine. 2020. doi: https://doi.org/10.1038/s41591-020-0869-5

[SanMiguel20] Lucía G. San Miguel, Agustín P. Moreira. "Transmisibilidad del Covid-19". Ministerio de Sanidad, Gobierno de España. 2020.

[SanJosé12] Pablo P. San-José, Laura G. Pérez, Eduardo Á. Alonso, Susana de la F. Rodríguez, Cristina G. Borge. "Estudio sobre seguridad en dispositivos móviles y smartphones". Instituto Nacional de Tecnologías de la comunicación. 2012.

14.2 Referencias en Internet

[Hopkins21] Johns Hopkins University. "COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University". https://github.com/CSSEGISandData/COVID-19. Última consulta: 13/06/2021.

[GobiernoDeEspaña21] Gobierno de España. "Estadísticas de RadarCovid". https://radarcovid.gob.es/estadisticas/descargas-radar. Última consulta: 13/06/2021.

[Arganda21] Carlos Arganda. "Radar Covid: solo se han notificado 48.822 casos en la 'app', el 2,06%". https://www.diariofarma.com/2021/02/13/radarcovid-solo-se-han-notificado-48-822-casos-en-la-app-el-206. Última consulta: 13/06/2021.

[BeSoftware21] BeSoftware. "¿Qué es C# y para qué sirve?". https://bsw.es/que-es-c/. Última consulta: 13/06/2021.

[Vector21] Vector ITC. "Más de 5.000 millones de dispositivos Bluetooth en 2021". https://www.vectoritcgroup.com/tech-magazine/innovation-trends/mas-de-5-000-millones-de-dispositivos-bluetooth-en-2021/. Última consulta: 13/06/2021.

[StatCounter21] GlobalStats StatCounter. "Mobile Operating System Market Share Worldwide". https://gs.statcounter.com/os-market-share/mobile/worldwide/. Última consulta: 13/06/2021.

[RedHat21] Red Hat." ¿Qué es la metodología ágil?". https://www.redhat.com/es/devops/what-is-agile-methodology. Última consulta: 13/06/2021.

[RedHat21] Red Hat." ¿Qué son la integración/distribución continuas (CI/CD)?". https://www.redhat.com/es/topics/devops/what-is-ci-cd. Última consulta: 13/06/2021.

[Rehkopf21] Max Rehkopf. "Historias de usuario con ejemplos y plantilla". https://www.atlassian.com/es/agile/project-management/user-stories. Última consulta: 13/06/2021.

[StrongLoop21] StrongLoop. "Express: Infraestructura web rápida, minimalista y flexible para Node.js". https://expressjs.com/es/. Última consulta: 13/06/2021.

[Facebook21] Facebook Inc. "React: Una biblioteca de JavaScript para construir interfaces de usuario". https://es.reactjs.org/. Última consulta: 13/06/2021.

[Mozilla21] Mozilla. "Primeros pasos en React". https://developer.mozilla.org/es/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started. Última consulta: 13/06/2021.

[Yeray21] Josue Yeray. "Compartiendo Código: Móvil + Web + Escritorio (1/2)". https://geeks.ms/jyeray/2010/10/25/compartiendo-cdigo-mvil-web-escritorio-12/. Última consulta: 13/06/2021.

[Shukla21] Ashish Shukla. "Repository pattern in c#". https://www.codecompiled.com/csharp/repository-pattern-in-csharp/. Última consulta: 13/06/2021.

[Microsoft21] Microsoft. "¿Qué es Xamarin.Forms?". https://docs.microsoft.com/es-es/xamarin/get-started/what-is-xamarin-forms. Última consulta: 13/06/2021.

[Microsoft21] Microsoft. "¿Qué es Xamarin?". https://docs.microsoft.com/es-es/xamarin/get-started/what-is-xamarin. Última consulta: 13/06/2021.

[Microsoft21] Microsoft. "Una introducción a NuGet". https://docs.microsoft.com/es-es/nuget/what-is-nuget. Última consulta: 13/06/2021.

[Microsoft21] Microsoft. "Paseo por el lenguaje C#". https://docs.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/. Última consulta: 13/06/2021.

[Glassdoor21] Glassdoor Inc. "Glassdoor". https://www.glassdoor.es/Sueldos/index.htm. Última consulta: 13/06/2021.

[Exchange21] Stack Exchange Inc. "Stack Overflow Jobs Salary Calculator". https://stackoverflow.com/jobs/salary. Última consulta: 13/06/2021.

[Exchange21] Stack Exchange Inc. "Stack Overflow". https://stackoverflow.com/. Última consulta: 13/06/2021.

[Wikimedia21] Fundación Wikimedia Inc. "Wikipedia". https://es.wikipedia.org. Última consulta: 13/06/2021.

[Academia21] Real Academia Española. "Diccionario de la lengua española". https://www.rae.es/. Última consulta: 13/06/2021.

Capítulo 15. Apéndices

15.1 Glosario y Diccionario de Datos

- **Aplicación:** Programa informático preparado para una utilización específica.
- **Base de datos:** Conjunto de datos organizados de forma estructurada y sistematizada para permitir su rápida recuperación.
- **Clase:** Plantilla para la creación de objetos de datos según un modelo predefinido. Representan entidades o conceptos, y están formadas por un conjunto de métodos y variables.
- **Compilación:** Proceso mediante el cual se convierte un programa en lenguaje máquina a partir de otro programa de computadora escrito en otro lenguaje de más alto nivel.
- **Despliegue:** Conjunto de actividades interrelacionadas que hacen que un programa esté disponible para su uso.
- **EDT:** Siglas de Estructura de Desglose de Trabajo. Se trata de una descomposición jerárquica de las tareas a realizar por el equipo de trabajo para cumplir con los objetivos de un proyecto.
- **Servidor:** Unidad informática que proporciona diversos servicios a computadoras conectadas con ella a través de una red.
- **SGBD:** Sistema Gestor de Bases de Datos. Conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos.
- **Sincronizar:** Reflejar los cambios realizados en un archivo almacenado en una máquina a otro archivo situado en un equipo distinto con el fin de mantener una consistencia entre ambos.
- **Software**: Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

15.2 Contenido Entregado en el ZIP

15.2.1 Raíz del archivo comprimido

Directorio	Contenido
./Directorio raíz	Contiene un fichero leeme.txt explicando esta estructura.
./Código fuente	Contiene el código fuente de los tres proyectos que forman el sistema de rastreo. Su contenido se describe más adelante.
./Instalación	Ficheros de instalación generados. Contiene el APK de la aplicación móvil.
./Documentación	Contiene el PDF de documentación asociado al proyecto y una carpeta de imágenes. Su estructura se describe más adelante.

15.2.2 Código fuente

Directorio	Contenido
./Directorio raíz de "Código fuente"	Contiene los tres proyectos que forman el sistema.
./Aplicación móvil	Contiene la solución de Visual Studio de la aplicación móvil. Esta solución está formada por tres proyectos: uno general, uno para Android y otro para iOS.
./Servidor	Contiene el proyecto para Visual Studio Code del servidor del sistema.
./Aplicación web	Contiene el proyecto para Visual Studio Code de la aplicación web.

15.2.3 Documentación

Directorio	Contenido
./Directorio raíz de "Documentación"	Contiene el PDF de documentación asociado al proyecto y una carpeta con las imágenes utilizadas en ella. Se incluye también un fichero de texto con los enlaces a la página web y a la API REST.
./Imágenes	Contiene las imágenes usadas en la documentación del proyecto.
./Imágenes/Diagramas	Contiene en formato de imagen los diagramas del sistema.
./Imágenes/Pantallas/App	Contiene capturas de pantalla de la aplicación móvil.
./Imágenes/Pantallas/Web	Contiene capturas de pantalla de la página web.

15.3 Índice Alfabético