Universidad de Oviedo

ESCUELA DE INGENIERÍA INFORMÁTICA

Trabajo Fin de Máster

Máster Universitario en Ingeniería Web

SISTEMA WEB PARA EL ANÁLISIS ESTÁTICO DE PROGRAMAS JAVA MEDIANTE PROGQUERY

AUTOR:
Inés Díaz del Rey

DIRIGIDO POR:
Miguel García Rodríguez
Francisco Ortín Soler

FECHA:
08/07/2024

Resumen

El desarrollo de código fuente en los últimos años ha tenido un crecimiento muy significativo. Una muestra de ello es el incremento exponencial en el número de repositorios existentes en sistemas tales como GitHub, GitLab y Bitbucket. Este elevado número de aplicaciones existentes demanda sistemas de análisis y búsqueda de código en repositorios masivos de código fuente.

ProgQuery es un proyecto de investigación desarrollado en la Universidad de Oviedo que permite la realización de análisis y búsquedas de un modo eficiente en grandes cantidades de código Java. En lugar de utilizar el código fuente, convierte éste a estructuras sintácticas de árbol y semánticas de grafo, permitiendo una consulta del código desde un mayor elevado nivel de abstracción. Sin embargo, su uso requiere un proceso complejo de implantación y configuración, así como un conocimiento técnico de Neo4j, la base de datos orientada a grafos utilizada para almacenar las distintas representaciones de código fuente. Este proyecto busca mejorar la sencillez en la utilización de ProgQuery, permitiendo a los usuarios cargar, analizar y compartir programas Java de manera accesible a través de internet, permitiendo además la colaboración en la comunidad de desarrollo de software.

Para ello, este trabajo fin de máster plantea el diseño y la implementación de un sistema web con dos interfaces fundamentales para la utilización de ProgQuery. Primero, se provee una Web API destinada a construir aplicaciones cliente que consuman los servicios que proporciona el sistema. Posteriormente, se desarrolla una aplicación web que consume la Web API para que los usuarios puedan acceder con cualquier navegador web con el fin de cargar y analizar programas y obtener los resultados.

El sistema se estructura en torno a cuatro componentes clave: gestión de usuarios, gestión de programas, gestión de análisis y gestión de resultados. La gestión de usuarios incluye el registro, autenticación, edición y eliminación de perfiles de usuario. La gestión de programas permite a los usuarios cargar programas Java, ya sea como archivos individuales o proyectos comprimidos, y también importar programas desde repositorios como GitHub. La gestión de análisis proporciona un sistema de nomenclatura único para análisis, con opciones de visibilidad, y permite a los usuarios crear, listar, actualizar y eliminar análisis. Por último, la gestión de resultados permite ejecutar análisis en programas, almacenar los resultados y acceder a ellos posteriormente.

Por todo ello, este proyecto se presenta como una solución integral que aborda desafíos fundamentales en el desarrollo de software y fomenta la colaboración, el aprendizaje y el avance en la comunidad de desarrolladores de software.

Palabras clave

ProgQuery, bases de datos orientadas a grafos, análisis de código fuente, Neo4J, Cypher, detección de errores, Java

Abstract

The development of source code in recent years has experienced a significant growth. An indicator of this is the exponential increase in the number of projects on platforms repositories such as GitHub, GitLab, and Bitbucket. The large number of existing applications requires systems for analyzing and searching code in massive source code repositories.

ProgQuery is a research project developed at the University of Oviedo that enables efficient analysis and code search within large volumes of Java code. Instead of using the textual representation of source code, it translates it into syntactic trees and semantic graph structures, allowing for code queries at a higher level of abstraction. However, its use requires a complex deployment and configuration process, as well as the technical knowledge of Neo4j—the graph-oriented database used to store the various representations of source code. This project aims to simplify the use of ProgQuery, enabling users to upload, analyze, and share Java programs in an accessible manner via the internet, thereby facilitating collaboration within the software development community.

To achieve this, this master's thesis proposes the design and implementation of a web system with two fundamental interfaces for using ProgQuery. First, a Web API to build client applications that consume the services offered by the system. Second, a web application that consumes the Web API, allowing users to access it with any web browser to upload and analyze programs and obtain results.

The system is structured around four key components: user management, program management, analysis management, and results management. User management includes the registration, authentication, editing, and deletion of user profiles. Program management allows users to upload Java programs, either as individual files or compressed projects, and also import programs from repositories like GitHub. Analysis management provides a unique naming system for analyses, with visibility options, and allows users to create, list, update, and delete analyses. Lastly, results management enables the execution of analyses on programs, storing the results, and accessing them later.

Overall, this project presents a comprehensive solution that addresses fundamental challenges in software development and promotes collaboration, learning, and advancement within the software development community.

Keywords

ProgQuery, graph databases, source code analysis, Neo4J, Cypher, source code error detection, Java

Agradecimientos

Quiero agradecer a Miguel García Rodríguez, mi tutor, por su guía experta y constante apoyo durante este trabajo de fin de máster. Su orientación fue fundamental para alcanzar los resultados obtenidos.

A mi familia, gracias por su apoyo incondicional y motivación en cada paso de este proceso académico.

Este trabajo ha sido financiado con fondos de la Universidad de Oviedo a través de su apoyo a grupos de investigación oficiales (GR-2011-0040).

Índice General

Resumen **2**

Palabras clave **2**

Abstract **3**

Keywords **3**

Agradecimientos **4**

Capítulo 1. Introducción **21**

1.1. Propósito y justificación del proyecto 21

1.2. Objetivo del proyecto 22

1.3. Alcance del proyecto 22

1.4. Trabajo relacionado 22

Capítulo 2. Aspectos teóricos **25**

2.1. ProgQuery 25

2.1.1. Representaciones de programas superpuestas 26

2.2. Lenguajes de programación y entornos 27

2.2.1. Java con Spring Boot 27

2.2.2. IntelliJ IDEA 28

2.2.3. .NET 28

2.2.4. Visual Studio 28

2.3. Bases de datos 29

2.3.1. PostgreSQL: Gestión de Datos Relacionales 29

2.3.2. Neo4j: Gestión de Datos Basados en Grafos 29

Capítulo 3. Análisis **30**

3.1. Definición del sistema 30

3.2. Requisitos del sistema 31

3.2.1. Requisitos funcionales 31

3.2.1.1. Requisitos funcionales de la gestión de autenticación 31

3.2.1.2. Requisitos funcionales de la gestión de usuarios 32

3.2.1.3. Requisitos funcionales de la gestión de programas 34

3.2.1.4. Requisitos funcionales de la gestión de análisis 36

3.2.1.5. Requisitos funcionales de la gestión de resultados 37

3.2.1.6. Requisitos funcionales adicionales 39
3.2.2. Requisitos no funcionales 40
3.3. Casos de uso y escenarios 40
3.3.1. Identificación de actores del sistema 40
3.3.1.1. Actores primarios 40
3.3.2. Especificación de casos de uso 41
3.3.2.1. Usuario no autenticado 41
3.3.2.1.1. CU-W-01 Registrarse en el sistema 41
3.3.2.1.2. CU-W-02 Identificarse en el sistema 42
3.3.2.1.3. CU-W-03 Registrarse en el sistema con Google 43
3.3.2.1.4. CU-W-04 Identificarse en el sistema con Google 44
3.3.2.1.5. CU-W-05 Visualizar los programas públicos 44
3.3.2.1.6. CU-W-06 Visualizar los detalles de un programa público 45
3.3.2.1.7. CU-W-07 Visualizar el listado de resultados de un programa público 46
3.3.2.1.8. CU-W-08 Visualizar los detalles del resultado de un programa público 47
3.3.2.1.9. CU-W-09 Visualizar los análisis públicos 48
3.3.2.1.10. CU-W-10 Visualizar los detalles de un análisis público 49
3.3.2.1.11. CU-W-11 Visualizar los detalles de un usuario 50
3.3.2.1.12. CU-W-12 Visualizar los programas públicos de un usuario 51
3.3.2.1.13. CU-W-13 Visualizar los análisis públicos de un usuario 52
3.3.2.1.14. CU-W-14 Analizar código java sin autenticarse 53
3.3.2.2. Usuario autenticado 55
3.3.2.2.1. CU-W-15 Cerrar sesión 55
3.3.2.2.2. CU-W-16 Visualizar el perfil de un usuario 56
3.3.2.2.3. CU-W-17 Visualizar los programas de un usuario 57
3.3.2.2.4. CU-W-18 Visualizar los análisis de un usuario 58
3.3.2.2.5. CU-W-19 Modificar los datos del perfil 59
3.3.2.2.6. CU-W-20 Visualizar el listado de programas 60
3.3.2.2.7. CU-W-21 Visualizar los detalles de un programa 60
3.3.2.2.8. CU-W-22 Visualizar el listado de resultados de un programa 61
3.3.2.2.9. CU-W-23 Visualizar los detalles del resultado de un programa desde el programa 62
3.3.2.2.10. CU-W-24 Registrar un nuevo programa 63
3.3.2.2.11. CU-W-25 Modificar un programa 65
3.3.2.2.12. CU-W-26 Eliminar un programa 66

3.3.2.2.13. CU-W-27 Ver un listado de los usuarios que tienen acceso a un programa....67
3.3.2.2.14. CU-W-28 Dar acceso a un usuario a un programa....68
3.3.2.2.15. CU-W-29 Eliminar el acceso de un usuario a un programa....69
3.3.2.2.16. CU-W-30 Visualizar el listado de análisis....70
3.3.2.2.17. CU-W-31 Visualizar los detalles de un análisis....71
3.3.2.2.18. CU-W-32 Registrar un nuevo análisis....71
3.3.2.2.19. CU-W-33 Modificar un análisis....72
3.3.2.2.20. CU-W-34 Eliminar un análisis....73
3.3.2.2.21. CU-W-35 Ver un listado de los usuarios que tienen acceso a un análisis....74
3.3.2.2.22. CU-W-36 Dar acceso a un usuario a un análisis....75
3.3.2.2.23. CU-W-37 Eliminar el acceso de un usuario a un análisis....76
3.3.2.2.24. CU-W-38 Visualizar el listado de resultados....77
3.3.2.2.25. CU-W-39 Visualizar los detalles de un resultado....78
3.3.2.2.26. CU-W-40 Visualizar los detalles del resultado de un programa....79
3.3.2.2.27. CU-W-41 Realizar el análisis de un programa....80
3.3.2.2.28. CU-W-42 Eliminar un resultado....80
3.3.2.2.29. CU-W-43 Eliminar el resultado de un programa....81
3.3.2.3. Usuario administrador....83
3.3.2.3.1. CU-W-44 Ver estadísticas generales....83
3.3.2.3.2. CU-W-45 Ver estadísticas de los programas....84
3.3.2.3.3. CU-W-46 Ver estadísticas de los análisis....84
3.3.2.3.4. CU-W-47 Ver estadísticas de los resultados....85
3.3.2.3.5. CU-W-48 Ver estadísticas de programas-análisis....86
3.4. Matrices de trazabilidad....87
3.4.1. Matriz de trazabilidad de los requisitos funcionales de la gestión de autenticación....87
3.4.2. Matriz de trazabilidad de los requisitos funcionales de la gestión de usuarios87
3.4.3. Matriz de trazabilidad de los requisitos funcionales de la gestión de programas....88
3.4.4. Matriz de trazabilidad de los requisitos funcionales de la gestión de análisis..88
3.4.5. Matriz de trazabilidad de los requisitos funcionales de la gestión resultados..88
3.4.6. Matriz de trazabilidad de los requisitos funcionales adicionales....89
Capítulo 4. Diseño del sistema....90
4.1. Arquitectura del sistema....90

4.2. Diagrama y modelo de datos 92
4.3. Interfaz de usuario 94
4.3.1. Mapa de navegación 94
4.3.2. Diseño de pantallas 96
Capítulo 5. Implementación del sistema 105
5.1. Herramientas y entorno de desarrollo 105
5.1.1. Lenguajes de Programación 105
5.1.1.1. Java 105
5.1.1.2. C# 105
5.1.2. Herramientas de desarrollo 105
5.1.2.1. IntelliJ IDEA 105
5.1.2.2. Visual Studio 106
5.1.3. Entorno de ejecución 106
5.1.3.1. Servidor 106
5.1.3.2. Tomcat 106
5.1.3.3. Kestrel 107
5.1.3.4. PostgreSQL 107
5.1.3.5. Neo4J 107
5.2. Gestión de usuarios y autenticación 108
5.3. Gestión de programas 110
5.4. Gestión de análisis 112
5.5. Gestión de resultados 114
5.6. Desarrollo de la aplicación web 116
Capítulo 6. Desarrollo de las pruebas 119
6.1. Pruebas unitarias para la gestión de usuarios y autenticación 119
6.2. Pruebas unitarias para la gestión de programas 119
6.3. Pruebas unitarias para la gestión de análisis 121
6.4. Pruebas unitarias para la gestión de resultados 123
6.5. Pruebas unitarias adicionales 123
6.6. Pruebas de integración 124
6.6.1. CU-W-01 Registrarse en el sistema 124
6.6.2. CU-W-02 Identificarse en el sistema 124
6.6.3. CU-W-03 Registrarse en el sistema con Google 124

6.6.4. CU-W-04 Identificarse en el sistema con Google 125
6.6.5. CU-W-05 Visualizar los programas públicos 125
6.6.6. CU-W-06 Visualizar los detalles de un programa público 125
6.6.7. CU-W-07 Visualizar el listado de resultados de un programa público 125
6.6.8. CU-W-08 Visualizar los detalles del resultado de un programa público 126
6.6.9. CU-W-09 Visualizar los análisis públicos 126
6.6.10. CU-W-10 Visualizar los detalles de un análisis público 126
6.6.11. CU-W-11 Visualizar los detalles de un usuario 126
6.6.12. CU-W-12 Visualizar los programas públicos de un usuario 127
6.6.13. CU-W-13 Visualizar los análisis públicos de un usuario 127
6.6.14. CU-W-14 Analizar código java sin autenticarse 127
6.6.15. CU-W-15 Cerrar sesión 128
6.6.16. CU-W-16 Visualizar el perfil de un usuario 128
6.6.17. CU-W-17 Visualizar los programas de un usuario 128
6.6.18. CU-W-18 Visualizar los análisis de un usuario 128
6.6.19. CU-W-19 Modificar los datos del perfil 128
6.6.20. CU-W-20 Visualizar el listado de programas 129
6.6.21. CU-W-21 Visualizar los detalles de un programa 129
6.6.22. CU-W-22 Visualizar el listado de resultados de un programa 129
6.6.23. CU-W-23 Visualizar los detalles del resultado de un programa desde el programa 130
6.6.24. CU-W-24 Registrar un nuevo programa 130
6.6.25. CU-W-25 Modificar un programa 130
6.6.26. CU-W-26 Eliminar un programa 131
6.6.27. CU-W-27 Ver un listado de los usuarios que tienen acceso a un programa 131
6.6.28. CU-W-28 Dar acceso a un usuario a un programa 131
6.6.29. CU-W-29 Eliminar el acceso de un usuario a un programa 132
6.6.30. CU-W-30 Visualizar el listado de análisis 132
6.6.31. CU-W-31 Visualizar los detalles de un análisis 132
6.6.32. CU-W-32 Registrar un nuevo análisis 132
6.6.33. CU-W-33 Modificar un análisis 133
6.6.34. CU-W-34 Eliminar un análisis 133

6.6.35. CU-W-35 Ver un listado de los usuarios que tienen acceso a un análisis.. 134
6.6.36. CU-W-36 Dar acceso a un usuario a un análisis 134
6.6.37. CU-W-37 Eliminar el acceso de un usuario a un análisis 134
6.6.38. CU-W-38 Visualizar el listado de resultados 134
6.6.39. CU-W-39 Visualizar los detalles de un resultado 135
6.6.40. CU-W-40 Visualizar los detalles del resultado de un programa 135
6.6.41. CU-W-41 Realizar el análisis de un programa 135
6.6.42. CU-W-42 Eliminar un resultado 135
6.6.43. CU-W-43 Eliminar el resultado de un programa 136
6.6.44. CU-W-44 Ver estadísticas generales 136
6.6.45. CU-W-45 Ver estadísticas de los programas 136
6.6.46. CU-W-46 Ver estadísticas de los análisis 136
6.6.47. CU-W-47 Ver estadísticas de los resultados 137
6.6.48. CU-W-48 Ver estadísticas de programas-análisis 137
6.7. Pruebas de rendimiento 137
6.7.1. Pruebas de Carga 137
6.7.2. Pruebas de Estrés 139
6.8. Pruebas de usabilidad 140
Capítulo 7. Manuales del sistema 143
7.1. Manual de usuario 143
7.1.1. Instalación y Despliegue de la API 143
7.1.2. Instalación y Despliegue de la Web 143
7.1.3. Acceder a ProgQuery 144
7.1.4. Registro e inicio de sesión 145
7.1.5. Gestión de programas 146
7.1.6. Gestión de análisis 151
7.1.7. Gestión de resultados 155
7.1.8. Gestión de usuario y cierre de sesión 159
7.1.9. Análisis básico 161
7.1.10. Estadísticas generales 161
Capítulo 8. Dirección y gestión del proyecto 165
8.1. Planificación del proyecto 165

8.1.1. Identificación de los interesados 165
8.1.2. OBS, PBS 165
8.1.3. Planificación inicial. WBS 167
8.1.4. Riesgos 171
8.1.4.1. Plan de gestión de riesgos 171
8.1.4.1.1. Metodología 171
8.1.4.1.1.1. Metodología general 172
8.1.4.1.1.2. Metodología de gestión de riesgos 172
8.1.4.1.2. Herramientas y tecnologías 173
8.1.4.1.2.1. Tormenta de ideas 173
8.1.4.1.2.2. Evaluaciones periódicas 173
8.1.4.1.2.3. Reuniones de riesgos 173
8.1.4.1.2.4. Método Delphi 173
8.1.4.1.2.5. Checklists 173
8.1.4.1.2.6. Estudio de incidencias 173
8.1.4.1.3. Roles y responsabilidades 173
8.1.4.1.4. Presupuesto 174
8.1.4.1.5. Calendario 174
8.1.4.1.6. Categorías de riesgo 174
8.1.4.1.7. Definiciones de probabilidad 175
8.1.4.1.8. Definiciones de impacto por objetivos 175
8.1.4.1.9. Matriz de probabilidad e impacto 176
8.1.4.1.10. Niveles de tolerancia 176
8.1.4.1.11. Planes de contingencia 176
8.1.4.1.11.1. Plan de presupuesto 176
8.1.4.1.11.2. Plan de planificación 176
8.1.4.1.11.3. Plan de alcance y calidad 176
8.1.4.1.12. Formatos de la documentación 177
8.1.4.1.13. Seguimiento 177
8.1.4.2. Identificación de riesgos 177
8.1.4.3. Registro de riesgos 178
8.1.5. Presupuesto inicial 179
8.1.5.1. Presupuesto de costes 179
8.2. Ejecución del proyecto 182

8.2.1. Plan de seguimiento de planificación 182
8.2.2. Bitácora de incidencias del proyecto 184
8.2.3. Riesgos 185
8.2.3.1. Datos del primer riesgo identificado 185
8.2.3.2. Datos del segundo riesgo identificado 187
8.2.3.3. Datos del tercer riesgo identificado 190
8.2.3.4. Datos del cuarto riesgo identificado 193
8.2.3.5. Datos del primer riesgo identificado 195
8.3. Cierre del proyecto 197
8.3.1. Planificación final 197
8.3.2. Informe final de riesgos 202
8.3.3. Presupuesto final de costes 203
8.3.4. Informe de lecciones aprendidas 205
8.3.4.1. Introducción 205
8.3.4.2. Resumen del Proyecto 206
8.3.4.3. Planificación y Gestión del Proyecto 206
8.3.4.4. Desarrollo Técnico 206
8.3.4.5. Gestión de Datos 207
8.3.4.6. Colaboración y Comunicación 207
8.3.4.7. Evaluación de Resultados 207
8.3.4.8. Conclusión 208
Capítulo 9. Conclusiones y ampliaciones 209
9.1. Conclusiones 209
9.2. Ampliaciones 210
Capítulo 10. Referencias bibliográficas 211

Índice de Ilustraciones

Ilustración 1. Arquitectura de ProgQuery [1] 25

Ilustración 2. Consulta Cypher para detectar el patrón incorrecto de enumerados identificado como el ítem 31 de [12] “use instance fields instead of ordinals” 26

Ilustración 3. Diagrama del sistema 31

Ilustración 4. Diagrama de casos de uso - Usuario no autenticado 41

Ilustración 5. Diagrama de casos de uso - Usuario autenticado 55

Ilustración 6. Diagrama de casos de uso - Usuario administrador 83

Ilustración 7. Diagrama de capas 91

Ilustración 8. Diagrama ER de la base de datos 93

Ilustración 9. Mapa de navegación 95

Ilustración 10. Diseño pantalla principal 96

Ilustración 11. Diseño pantalla de inicio de sesión 96

Ilustración 12. Diseño pantalla de registro 97

Ilustración 13. Diseño pantalla de listado de programas 97

Ilustración 14. Diseño pantalla de detalles de un programa 98

Ilustración 15. Diseño pantalla de añadir un programa 98

Ilustración 16. Diseño pantalla de modificar un programa 99

Ilustración 17. Diseño pantalla de listado de análisis 99

Ilustración 18. Diseño pantalla de detalles de un análisis 100

Ilustración 19. Diseño pantalla de añadir un análisis 100

Ilustración 20. Diseño pantalla de modificar un análisis 101

Ilustración 21. Diseño pantalla de listado de resultados 101

Ilustración 22. Diseño pantalla de detalles de un resultado 102

Ilustración 23. Diseño pantalla de detalles del resultado de un programa 102

Ilustración 24. Diseño pantalla de añadir un resultado 103

Ilustración 25. Diseño pantalla del perfil de usuario 103

Ilustración 26. Diseño pantalla de modificar el perfil de usuario 103

Ilustración 27. Diseño pantalla de análisis básico 104

Ilustración 28. Diseño pantalla de estadísticas generales 104

Ilustración 29. Pruebas unitarias de autenticación 119
Ilustración 30. Pruebas unitarias de gestión de usuarios 119
Ilustración 31. Pruebas unitarias de gestión de programas (I) 120
Ilustración 32. Pruebas unitarias de gestión de programas (II) 121
Ilustración 33. Pruebas unitarias de gestión de análisis (I) 122
Ilustración 34. Pruebas unitarias de gestión de análisis (II) 122
Ilustración 35. Pruebas unitarias de gestión de resultados 123
Ilustración 36. Pruebas unitarias de estadísticas generales 123
Ilustración 37. Configuración de pruebas de carga usando JMeter 138
Ilustración 38. Resultados de las pruebas de carga 138
Ilustración 39. Configuración de pruebas de estrés usando JMeter 139
Ilustración 40. Resultados de las pruebas de estrés 140
Ilustración 41. Pantalla principal 144
Ilustración 42. Pantalla de inicio de sesión 145
Ilustración 43. Pantalla de registro 145
Ilustración 44. Pantalla de confirmación de registro 146
Ilustración 45. Pantalla principal con la sesión iniciada 146
Ilustración 46. Pantalla del listado de programas 147
Ilustración 47. Pantalla de inserción de un programa 147
Ilustración 48. Pantalla de confirmación de inserción de un programa 148
Ilustración 49. Pantalla del listado de programas propios 148
Ilustración 50. Pantalla de detalles de un programa 149
Ilustración 51. Pantalla de modificación de un programa 149
Ilustración 52. Pantalla de detalles de un programa modificado 150
Ilustración 53. Pantalla de confirmación de eliminación de un programa 150
Ilustración 54. Pantalla del listado de resultados asociados a un programa 151
Ilustración 55. Pantalla del listado de análisis 151
Ilustración 56. Pantalla de creación de un análisis 152
Ilustración 57. Pantalla de confirmación de creación de un análisis 152
Ilustración 58. Pantalla del listado de análisis propios 153
Ilustración 59. Pantalla de detalles de un análisis 153
Ilustración 60. Pantalla de modificación de un análisis 154

Ilustración 61. Pantalla de detalles de un análisis modificado 154
Ilustración 62. Pantalla de confirmación de eliminación de un análisis 155
Ilustración 63. Pantalla del listado de resultados propios 155
Ilustración 64. Pantalla de creación de un análisis de un programa 156
Ilustración 65. Pantalla de confirmación de realización del análisis de programa(s) 156
Ilustración 66. Pantalla del listado de resultados propios con uno nuevo guardado 157
Ilustración 67. Pantalla de detalles de un resultado 157
Ilustración 68. Pantalla de detalles de un resultado de un programa 158
Ilustración 69. Pantalla de confirmación de eliminación del resultado de un programa 158
Ilustración 70. Pantalla de confirmación de eliminación de un resultado 159
Ilustración 71. Pantalla con menú del usuario desplegado - Selección del perfil 159
Ilustración 72. Pantalla del perfil de usuario 160
Ilustración 73. Pantalla de modificación del perfil de usuario 160
Ilustración 74. Pantalla con menú del usuario desplegado - Selección de cierre de sesión 161
Ilustración 75. Pantalla de análisis básico 161
Ilustración 76. Pantalla de estadísticas generales 162
Ilustración 77. Pantalla de estadísticas de programas 162
Ilustración 78. Pantalla de estadísticas de análisis 163
Ilustración 79. Pantalla de estadísticas de resultados 163
Ilustración 80. Pantalla de estadísticas de programas-análisis 164
Ilustración 81. Icono API 164
Ilustración 82. API 164
Ilustración 83. Reuniones con el tutor 167
Ilustración 84. Tareas de análisis, dirección y gestión del proyecto e hitos 168
Ilustración 85. Tareas de diseño del sistema 168
Ilustración 86. Tareas de desarrollo del software 169
Ilustración 87. Tareas de pruebas y despliegue 169
Ilustración 88. Tareas de documentación y cierre del proyecto 170
Ilustración 89. Diagrama de Gantt I 170
Ilustración 90. Diagrama de Gantt II 171
Ilustración 91. Diagrama de Gantt III 171

Ilustración 92. Presupuesto inicial I 180
Ilustración 93. Presupuesto inicial II 180
Ilustración 94. Presupuesto inicial III 180
Ilustración 95. Presupuesto inicial IV 181
Ilustración 96. Presupuesto inicial V 181
Ilustración 97. Presupuesto inicial VI 181
Ilustración 98. Presupuesto inicial VII 182
Ilustración 99. Presupuesto inicial VIII 182
Ilustración 100. Reuniones con el tutor 198
Ilustración 101. Tareas de análisis, dirección y gestión del proyecto e hitos 198
Ilustración 102. Tareas de diseño del sistema 199
Ilustración 103. Tareas de desarrollo del software 199
Ilustración 104. Tareas de pruebas y despliegue 200
Ilustración 105. Tareas de documentación y cierre del proyecto 200
Ilustración 106. Diagrama de Gantt I 201
Ilustración 107. Diagrama de Gantt II 201
Ilustración 108. Diagrama de Gantt III 202
Ilustración 109. Presupuesto final I 203
Ilustración 110. Presupuesto final II 204
Ilustración 111. Presupuesto final III 204
Ilustración 112. Presupuesto final IV 204
Ilustración 113. Presupuesto final V 204
Ilustración 114. Presupuesto final VI 205
Ilustración 115. Presupuesto final VII 205
Ilustración 116. Presupuesto final VIII 205

Índice de Tablas

Tabla 1. Actores primarios del sistema 41
Tabla 2. Caso de uso: Registrarse en el sistema 42
Tabla 3. Caso de uso: Identificarse en el sistema 43
Tabla 4. Caso de uso: Registrarse en el sistema con Google 44
Tabla 5. Caso de uso: Identificarse en el sistema con Google 44
Tabla 6. Caso de uso: Visualizar los programas públicos 45
Tabla 7. Caso de uso: Visualizar los detalles de un programa público 46
Tabla 8. Caso de uso: Visualizar el listado de resultados de un programa público 47
Tabla 9. Caso de uso: Visualizar los detalles del resultado de un programa público 48
Tabla 10. Caso de uso: Visualizar los análisis públicos 49
Tabla 11. Caso de uso: Visualizar los detalles de un análisis público 50
Tabla 12. Caso de uso: Visualizar los detalles de un usuario 51
Tabla 13. Caso de uso: Visualizar los programas públicos de un usuario 52
Tabla 14. Caso de uso: Visualizar los análisis públicos de un usuario 53
Tabla 15. Caso de uso: Analizar código java sin autenticarse 54
Tabla 16. Caso de uso: Cerrar sesión 56
Tabla 17. Caso de uso: Visualizar el perfil de un usuario 57
Tabla 18. Caso de uso: Visualizar los programas de un usuario 58
Tabla 19. Caso de uso: Visualizar los análisis de un usuario 59
Tabla 20. Caso de uso: Modificar los datos del perfil 60
Tabla 21. Caso de uso: Visualizar el listado de programas 60
Tabla 22. Caso de uso: Visualizar los detalles de un programa 61
Tabla 23. Caso de uso: Visualizar el listado de resultados de un programa 62
Tabla 24. Caso de uso: Visualizar los detalles del resultado de un programa desde el programa 63
Tabla 25. Caso de uso: Registrar un nuevo programa 64
Tabla 26. Caso de uso: Modificar un programa 66
Tabla 27. Caso de uso: Eliminar un programa 67
Tabla 28. Caso de uso: Ver un listado de los usuarios que tienen acceso a un programa 68

Tabla 29. Caso de uso: Dar acceso a un usuario a un programa....69
Tabla 30. Caso de uso: Eliminar el acceso de un usuario a un programa....70
Tabla 31. Caso de uso: Visualizar el listado de análisis....70
Tabla 32. Caso de uso: Visualizar los detalles de un análisis....71
Tabla 33. Caso de uso: Registrar un nuevo análisis....72
Tabla 34. Caso de uso: Modificar un análisis....73
Tabla 35. Caso de uso: Eliminar un análisis....74
Tabla 36. Caso de uso: Ver un listado de los usuarios que tienen acceso a un análisis....75
Tabla 37. Caso de uso: Dar acceso a un usuario a un análisis....76
Tabla 38. Caso de uso: Eliminar el acceso de un usuario a un análisis....77
Tabla 39. Caso de uso: Visualizar el listado de resultados....78
Tabla 40. Caso de uso: Visualizar los detalles de un resultado....78
Tabla 41. Caso de uso: Visualizar los detalles del resultado de un programa....79
Tabla 42. Caso de uso: Realizar el análisis de un programa....80
Tabla 43. Caso de uso: Eliminar un resultado....81
Tabla 44. Caso de uso: Eliminar el resultado de un programa....82
Tabla 45. Caso de uso: Ver estadísticas generales....84
Tabla 46. Caso de uso: Ver estadísticas de los programas....84
Tabla 47. Caso de uso: Ver estadísticas de los análisis....85
Tabla 48. Caso de uso: Ver estadísticas de los resultados....86
Tabla 49. Caso de uso: Ver estadísticas de programas-análisis....87
Tabla 50. Matriz de trazabilidad de los requisitos funcionales de la gestión de autenticación....87
Tabla 51. Matriz de trazabilidad de los requisitos funcionales de la gestión de usuarios....88
Tabla 52. Matriz de trazabilidad de los requisitos funcionales de la gestión de programas....88
Tabla 53. Matriz de trazabilidad de los requisitos funcionales de la gestión de análisis....88
Tabla 54. Matriz de trazabilidad de los requisitos funcionales de la gestión resultados....89
Tabla 55. Matriz de trazabilidad de los requisitos funcionales adicionales....89
Tabla 56. Pruebas de integración – Registrarse en el sistema....124
Tabla 57. Pruebas de integración – Identificarse en el sistema....124
Tabla 58. Pruebas de integración – Registrarse en el sistema con Google....125
Tabla 59. Pruebas de integración – Identificarse en el sistema con Google....125

Tabla 60. Pruebas de integración – Visualizar los programas públicos 125

Tabla 61. Pruebas de integración – Visualizar los detalles de un programa público 125

Tabla 62. Pruebas de integración – Visualizar el listado de resultados de un programa público 126

Tabla 63. Pruebas de integración – Visualizar los detalles del resultado de un programa público 126

Tabla 64. Pruebas de integración – Visualizar los análisis públicos 126

Tabla 65. Pruebas de integración – Visualizar los detalles de un análisis público 126

Tabla 66. Pruebas de integración – Visualizar los detalles de un usuario 127

Tabla 67. Pruebas de integración – Visualizar los programas públicos de un usuario 127

Tabla 68. Pruebas de integración – Visualizar los análisis públicos de un usuario 127

Tabla 69. Pruebas de integración – Analizar código java sin autenticarse 127

Tabla 70. Pruebas de integración – Cerrar sesión 128

Tabla 71. Pruebas de integración – Visualizar el perfil de un usuario 128

Tabla 72. Pruebas de integración – Visualizar los programas de un usuario 128

Tabla 73. Pruebas de integración – Visualizar los análisis de un usuario 128

Tabla 74. Pruebas de integración – Modificar los datos del perfil 129

Tabla 75. Pruebas de integración – Visualizar el listado de programas 129

Tabla 76. Pruebas de integración – Visualizar los detalles de un programa 129

Tabla 77. Pruebas de integración – Visualizar el listado de resultados de un programa 129

Tabla 78. Pruebas de integración – Visualizar los detalles del resultado de un programa desde el programa 130

Tabla 79. Pruebas de integración – Registrar un nuevo programa 130

Tabla 80. Pruebas de integración – Modificar un programa 131

Tabla 81. Pruebas de integración – Eliminar un programa 131

Tabla 82. Pruebas de integración – Ver un listado de los usuarios que tienen acceso a un programa 131

Tabla 83. Pruebas de integración – Dar acceso a un usuario a un programa 132

Tabla 84. Pruebas de integración – Eliminar el acceso de un usuario a un programa 132

Tabla 85. Pruebas de integración – Visualizar el listado de análisis 132

Tabla 86. Pruebas de integración – Visualizar los detalles de un análisis 132

Tabla 87. Pruebas de integración – Registrar un nuevo análisis 133

Tabla 88. Pruebas de integración – Modificar un análisis 133

Tabla 89. Pruebas de integración – Eliminar un análisis 133

Tabla 90. Pruebas de integración – Ver un listado de los usuarios que tienen acceso a un análisis 134

Tabla 91. Pruebas de integración – Dar acceso a un usuario a un análisis 134

Tabla 92. Pruebas de integración – Eliminar el acceso de un usuario a un análisis 134

Tabla 93. Pruebas de integración – Visualizar el listado de resultados 135

Tabla 94. Pruebas de integración – Visualizar los detalles de un resultado 135

Tabla 95. Pruebas de integración – Visualizar los detalles del resultado de un programa 135

Tabla 96. Pruebas de integración – Realizar el análisis de un programa 135

Tabla 97. Pruebas de integración – Eliminar un resultado 136

Tabla 98. Pruebas de integración – Eliminar el resultado de un programa 136

Tabla 99. Pruebas de integración – Ver estadísticas generales 136

Tabla 100. Pruebas de integración – Ver estadísticas de los programas 136

Tabla 101. Pruebas de integración – Ver estadísticas de los análisis 137

Tabla 102. Pruebas de integración – Ver estadísticas de los resultados 137

Tabla 103. Pruebas de integración – Ver estadísticas de programas-análisis 137

Tabla 104. Resultado de las pruebas de estrés 140

Tabla 105. Resultados de las pruebas de usabilidad 141

Tabla 106. Interesados 165

Tabla 107. Identificación de riesgos 177

Tabla 108. Registro de riesgos 179

Tabla 109. Incidencias registradas 184

Capítulo 1. Introducción

1.1. Propósito y justificación del proyecto

En el mundo del desarrollo de software, la creciente complejidad de los programas ha generado la necesidad de herramientas avanzadas para procesar el código fuente para tareas tales como la detección de errores potenciales, vulnerabilidades, cuellos de botella en el rendimiento, cumplimiento de guías de estilo e incluso la búsqueda de código por criterios semánticos y sintácticos.

ProgQuery, una potente herramienta desarrollada en el grupo de investigación Computational Reflection de la Universidad de Oviedo, orientada a satisfacer las necesidades previamente citadas [1]. Para ello, tal y como se detalla en la Sección 2.3, ProgQuery toma el código Java, se lo pasa a una modificación de un compilador, crea estructuras sintácticas de árbol y semánticas de grafo y las almacena en una base de datos Neo4j basada en grafos. Posteriormente, empleando el lenguaje declarativo de consulta Cypher, el usuario puede realizar análisis de código con un elevado nivel de abstracción (utilizando las abstracciones de árbol y grafos), de un modo eficiente.

Sin embargo, el uso de ProgQuery requiere una configuración técnica considerable y un conocimiento especializado, lo cual puede ser un obstáculo para muchos desarrolladores. Para superar este desafío, este trabajo fin de máster propone un proyecto que simplifique y democratice el acceso a las capacidades de ProgQuery, permitiendo a los usuarios cargar, analizar y compartir programas Java de manera fácil y accesible a través de internet. Esta iniciativa no solo busca facilitar el uso de ProgQuery, sino también fomentar la colaboración y el intercambio de conocimientos dentro de la comunidad de desarrolladores de software.

El proyecto se centra en el diseño y la implementación de un sistema web robusto y versátil, compuesto por dos partes fundamentales que trabajarán en conjunto para proporcionar un servicio completo. En primer lugar, una Web API servirá como base para aplicaciones cliente permitiendo que usen los servicios del sistema, facilitando la integración con diversas herramientas. Y, en segundo lugar, una aplicación web ofrecerá a los usuarios la posibilidad de acceder a estos servicios desde cualquier navegador, brindándoles una interfaz sencilla para cargar, analizar y obtener resultados de sus programas Java.

El sistema estará estructurado en torno a cuatro componentes clave, diseñados para proporcionar una experiencia de usuario completa y eficiente. La gestión de usuarios incluirá funciones de registro, autenticación, y administración de perfiles, asegurando que cada usuario tenga un acceso seguro y personalizado. La gestión de programas permitirá la carga de archivos Java individuales o proyectos comprimidos, así como la importación directa desde repositorios conocidos como *GitHub*, *GitLab*, *Bitbucket* o en general cualquier repositorio *Git* [2]. En cuanto a la gestión de análisis, se establecerá un sistema de nombres único con opciones de visibilidad, permitiendo a los usuarios crear, listar, actualizar y eliminar análisis de manera flexible. Finalmente, la gestión de resultados se encargará de ejecutar los análisis, guardar los resultados y facilitar su acceso posterior.

Este proyecto se propone no solo como una herramienta técnica, sino como una manera de mejorar continuamente la práctica del desarrollo de software en Java. Al proporcionar un entorno accesible para el análisis avanzado de código, se espera estimular la colaboración y el aprendizaje entre desarrolladores, promoviendo el intercambio de análisis y resultados que puedan beneficiar a la comunidad en general. Además, este sistema tiene el potencial de impulsar investigaciones futuras y fomentar innovaciones en la industria del software, consolidándose como una contribución significativa al avance tecnológico y al desarrollo profesional en el campo de la programación.

1.2. Objetivo del proyecto

El objetivo principal de este proyecto de software radica en la creación de un sistema que facilite a los usuarios de ProgQuery la creación y el intercambio de análisis estáticos para programas Java, permitiendo procesar grandes volúmenes de código Java de un modo eficiente. Este sistema estará disponible a través de múltiples interfaces, incluyendo una aplicación web accesible desde navegadores web y una Web API diseñada para permitir la construcción de aplicaciones cliente que aprovechen los servicios de ProgQuery.

Los usuarios podrán cargar programas Java para su análisis, ya sea de forma directa o desde repositorios de código fuente populares como *GitHub*, *GitLab* y *Bitbucket*. El acceso al sistema será abierto a todos los usuarios, con el objetivo de fomentar la contribución y el intercambio de análisis en toda la comunidad de desarrollo de software. En resumen, el proyecto aspira a mejorar significativamente la práctica del desarrollo de software en Java al ofrecer análisis específicos y alentar la colaboración entre los usuarios de ProgQuery, con un potencial importante para futuras investigaciones y avances en la industria tecnológica.

1.3. Alcance del proyecto

El alcance de este proyecto se centra en el diseño e implementación de un sistema web con dos interfaces principales: una aplicación web y una Web API. Estas interfaces permitirán a los usuarios cargar programas Java, analizarlos utilizando la herramienta ProgQuery y compartir análisis estáticos resultantes. Los programas podrán ser cargados tanto como archivos individuales o como proyectos comprimidos, y también se podrá importar desde repositorios de código fuente *Git*.

La aplicación web estará diseñada para ofrecer una experiencia intuitiva y amigable para los usuarios, permitiendo la gestión de usuarios, programas, análisis y resultados. El enfoque principal será brindar a los usuarios una forma accesible de colaborar en análisis de programas Java y promover la colaboración dentro de la comunidad de desarrollo de software. Sin embargo, es importante destacar que el proyecto no incluirá la implementación técnica detallada de ProgQuery ni aspectos relacionados con su infraestructura subyacente.

1.4. Trabajo relacionado

El análisis de código Java ha avanzado significativamente, con el desarrollo de herramientas sofisticadas que permiten a los desarrolladores garantizar la calidad, seguridad y eficiencia

del código. Entre las herramientas más destacadas en el análisis estático se encuentran *SonarQube*, *SpotBugs* y *PMD*. *SonarQube* es una plataforma ampliamente utilizada que realiza revisiones automáticas del código para detectar problemas de calidad, seguridad y adherencia a estándares. Su integración con sistemas CI/CD (*Continuous Integration / Continuous Delivery*) la convierte en una herramienta indispensable en entornos empresariales [3]. *SpotBugs*, por su parte, examina el bytecode Java para identificar posibles errores comunes y problemas de rendimiento, y se integra fácilmente con entornos de desarrollo como Eclipse e IntelliJ IDEA [4]. *PMD*, otra herramienta popular, se enfoca en detectar malas prácticas y patrones de diseño incorrectos en el código fuente, permitiendo la personalización de reglas por parte del usuario [5].

Las herramientas basadas en grafos, como ProgQuery [1] y *JArchitect* [6], ofrecen una visión más estructurada y visual de la arquitectura del código. *JArchitect* proporciona una visualización avanzada de la arquitectura de aplicaciones Java, generando reportes detallados sobre la calidad del código y la arquitectura, y es utilizado principalmente en entornos empresariales para mantener y mejorar grandes bases de código. De manera similar, Wiggle [7] es una herramienta prototipo que permite realizar consultas sobre código fuente basado en el modelo de datos de grafos. Wiggle se distingue por traducir el Árbol de Sintaxis Abstracta (AST) de los archivos de código fuente en nodos de grafos en Neo4j. Cada elemento del AST y sus propiedades se convierten en nodos, y las relaciones entre estos elementos se traducen en aristas etiquetadas, lo que permite una representación detallada y manipulable del código. Esta metodología facilita consultas complejas y proporciona una visión profunda de la estructura del código, complementando las capacidades ofrecidas por herramientas como ProgQuery y JArchitect. Una de las características principales de ProgQuery es que permite a cualquier usuario hacer sus propios análisis desde un elevado nivel de abstracción y de un modo declarativo.

Además de las herramientas de análisis estático, existen herramientas de análisis dinámico que proporcionan una perspectiva diferente sobre la ejecución del código. *Java PathFinder* (JPF) es una de estas herramientas, que explora todos los posibles estados de ejecución de un programa Java para detectar errores concurrentes y problemas de tiempo de ejecución, siendo especialmente útil en software crítico donde la concurrencia es un factor importante [8]. *JProfiler* es otra herramienta clave en este ámbito, enfocada en el análisis de rendimiento y perfilado de aplicaciones Java [9]. Ofrece monitoreo detallado de uso de CPU, memoria y tráfico de red, ayudando a identificar cuellos de botella y optimizar el rendimiento de las aplicaciones.

En cuanto a la integración de herramientas de análisis de código en sistemas web, plataformas como *CodeClimate* y *GitHub Actions* han revolucionado la forma en que los desarrolladores realizan análisis de código. *CodeClimate* [10] ofrece una solución basada en la nube que se integra con repositorios de código para proporcionar retroalimentación continua sobre la calidad del código y adherencia a estándares, mientras que *GitHub Actions* [11] permite ejecutar análisis de código directamente desde repositorios de *GitHub*, facilitando la integración y el despliegue continuos (CI/CD). Ambas plataformas proporcionan interfaces web y APIs que facilitan su uso e integración en diversos entornos de desarrollo.

En conclusión, el análisis de código Java se beneficia de una amplia gama de herramientas que van desde el análisis estático y dinámico hasta representaciones basadas en grafos.

Estas herramientas no solo ayudan a identificar errores, optimizar el rendimiento, satisfacer el cumplimiento de guías de estilo, detectar código vulnerable y repetido, sino que también facilitan la comprensión de la estructura del código y garantizan la adherencia a las mejores prácticas de programación. La integración de estas herramientas en sistemas web, como se propone en este proyecto, mejora la accesibilidad y colaboración, permitiendo a los desarrolladores realizar análisis complejos de manera más sencilla y eficiente. Este enfoque está orientado a simplificar y democratizar el análisis de código Java, fomentando la colaboración y el aprendizaje en la comunidad de desarrolladores.

Capítulo 2. Aspectos teóricos

2.1. ProgQuery

ProgQuery [1] es una herramienta avanzada de análisis de código fuente diseñada inicialmente para Java, aunque se está trabajando en la incorporación de Python y una representación común agnóstica al lenguaje. ProgQuery está desarrollado por el grupo de investigación Computational Reflection en colaboración con la Universidad de Cambridge, como una extensión del sistema Wiggle creado por Raoul Urma y Alan Mycroft [7].

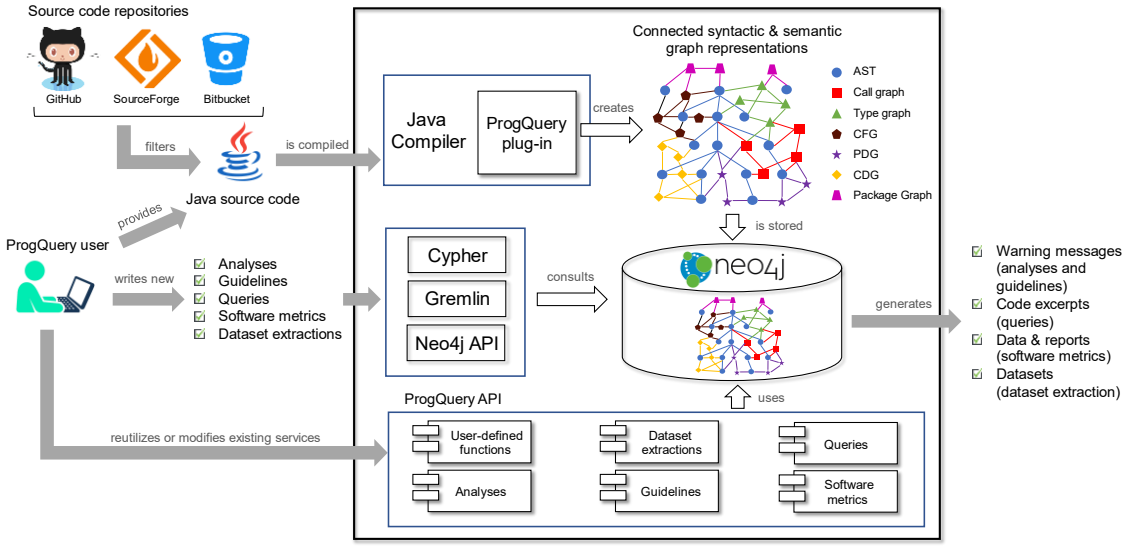


Ilustración 1. Arquitectura de ProgQuery [1]

La Ilustración 1 muestra la arquitectura de ProgQuery. Los programas Java que se van a procesar pueden obtenerse de repositorios de código abierto existentes, actualmente soporta *GitHub*, *Gitlab* y *Bitbucket*, es decir, cualquier sistema basado en Git, o bien ser proporcionados por el usuario. El código Java se compila mediante una modificación del compilador estándar de Java que se ha implementado previamente. Dicha modificación, además de generar código, crea siete grafos diferentes para cada programa, uno de ellos es el árbol de sintaxis abstracta o AST[1]. Estas representaciones se superponen, lo que significa que un nodo sintáctico puede estar conectado a otras representaciones semánticas diferentes a través de relaciones semánticas, y viceversa. El compilador Java modificado crea las siete diferentes representaciones superpuestas y las almacena en una base de datos orientada a grafos Neo4j, ofreciendo un elevado nivel de abstracción con información sintáctica y semántica, no simplemente el texto del código fuente. Por ejemplo, podemos obtener un método de una clase (sintáctica) que utilice un objeto de otro tipo (semántica) y conocer si éste tiene un método que pueda llamar a otro determinado (semántica).

Los usuarios de ProgQuery pueden consultar los distintos grafos de un programa dado de varias maneras. Pueden escribir análisis estáticos de programas, verificar el cumplimiento de guías de estilo o directrices, buscar código utilizando consultas avanzadas, obtener

[1] Puesto que un árbol es un caso particular de un grafo, se refiere de forma general a grafos, incluyendo el árbol de sintaxis abstracta de un programa.

métricas de software y extraer conjuntos de datos con información sintáctica y semántica. ProgQuery plataforma ofrece una colección de servicios existentes, principalmente análisis de detección de errores, como parte de la API de ProgQuery. De esta manera, los usuarios pueden utilizar dichos servicios existentes con su código Java. La API de ProgQuery también incluye funciones definidas por el usuario de Neo4j para facilitar la creación de nuevos análisis.

Las representaciones de programas almacenadas en Neo4j pueden consultarse de diferentes maneras. El principal mecanismo que utilizamos actualmente es Cypher, un lenguaje de consulta gráfica declarativo ampliamente utilizado con Neo4j. No obstante, también soporta otros lenguajes como Gremlin, basado en recorrido de grafos que admite consultas imperativas y declarativas para Neo4j. También es factible utilizar el API de Neo4j para el recorrido de grafos.

Dado el código fuente Java y una consulta (por ejemplo, en Cypher), ProgQuery puede generar varios tipos de salida. Para análisis de programas y cumplimiento de directrices, ProgQuery devolverá una colección de mensajes de advertencia para mejorar el código Java de entrada. Para consultas avanzadas, se devuelven los fragmentos de código encontrados, etiquetados con sus ubicaciones.

2.1.1. Representaciones de programas superpuestas

Una de las características clave de ProgQuery es la información sintáctica y semántica proporcionada como diferentes representaciones de grafos. El AST es la estructura central, donde los nodos representan construcciones sintácticas, vinculadas jerárquicamente a través de relaciones sintácticas. Los nodos del AST también están vinculados a otros nodos (semánticos o sintácticos) a través de relaciones semánticas. Por lo tanto, las representaciones gráficas sintácticas y semánticas están superpuestas. Esto facilita la expresión de consultas que combinan información sintáctica y semántica, como la que se muestra en la Ilustración 2.

A modo de ejemplo, el código Cypher de la Ilustración 2 implementa en ProgQuery la detección del error número 31 identificado por Joushua Bloch en su famoso libro "Effective Java" (segunda edición) [12]. La regla a comprobar se denomina "use instance fields instead of ordinals" y trata de evitar el uso ordinal de los valores de un enumerado, debido a que es un código propenso a errores.

```
MATCH (enclosingCU)-[:HAS_TYPE_DEF |
    HAS_INNER_TYPE_DEF]->()-[:DECLARES_METHOD |
    DECLARES_CONSTRUCTOR]->(enclosingM)-[:CALLS]->(inv)-[:HAS_DEF]->(md)
WHERE md.fullyQualifiedName ='java.lang.Enum:ordinal()int'
RETURN '{"rule":"CMU-DCL56","location":"'+enclosingCU.fileName +
    '","node":"java.lang.Enum:ordinal()","line":"'+inv.lineNumber+'","column":"' +
    inv.column +'","message":
    "You should not attach significance to the ordinal of an enum."}'

```

Ilustración 2. Consulta Cypher para detectar el patrón incorrecto de enumerados identificado como el ítem 31 de [12] "use instance fields instead of ordinals"

Las siete presentaciones de grafos creadas por ProgQuery son las siguientes:

1. *Abstract Syntax Tree (AST)*: Para un programa dado, se crean diferentes AST donde los nodos raíz son las diferentes unidades de compilación, es decir, archivos Java. Todas están unidas por un nodo Programa.
2. *Control Flow Graph (CFG)*: Los CFGs representan los caminos de ejecución que o bien pueden tomar (may) o bien ciertamente toman (must) cuando el programa se ejecuta.
3. *Program Dependency Graph (PDG)*: Los PDGs en ProgQuery proporcionan información sobre cuándo las variables (campos, parámetros y variables locales) y el estado del objeto al que apuntan (en caso de que las variables sean referencias) son leídas o modificadas.
4. *Call* Graph: Esta representación proporciona información sobre invocaciones de métodos y constructores en el código fuente. Los nodos de definición de métodos/constructores están conectados a todas las invocaciones en sus cuerpos a través de relaciones CALLS.
5. *Type Graph*: Los grafos de tipos representan toda la información de tipos en el programa fuente. Todos los nodos de expresión están vinculados con su tipo estático.
6. *Class Dependency Graph (CDG)*: La representación CDG está destinada a definir las relaciones de uso entre tipos. La única relación proporcionada es USES_TYPE, que conecta dos definiciones de tipos (clase, interfaz, enumeración o registro).
7. *Package* Graph: El Grafo de Paquetes proporciona información sobre la dependencia entre paquetes y une todos los ASTs en un programa.

En resumen, ProgQuery permite realizar análisis complejos sobre el código Java mediante un lenguaje de consulta específico y declarativo. Ofrece una infraestructura eficiente y escalable que permite extraer y utilizar propiedades semánticas y sintácticas avanzadas del código fuente Java utilizando representaciones altamente expresivas. Estas consultas pueden abarcar desde la identificación de dependencias hasta la detección de patrones específicos de diseño o anti-patrones dentro del código, proporcionando a los desarrolladores herramientas poderosas para mejorar la calidad y seguridad del software.

Para un mayor detalle acerca de ProgQuery, consúltese [1].

2.2. Lenguajes de programación y entornos

En esta sección se exploran los aspectos teóricos relacionados con los lenguajes de programación y los entornos de desarrollo utilizados en el proyecto. Se describen los fundamentos y las características de Java con Spring Boot, .NET, así como los entornos de desarrollo IntelliJ IDEA y Visual Studio, que han sido empleados para desarrollar la API y la interfaz del sistema, respectivamente.

2.2.1. Java con Spring Boot

Java [13] es un lenguaje de programación de propósito general, orientado a objetos y concurrente, diseñado para tener las menores dependencias de implementación posibles. Su filosofía "write once, run anywhere" (escribe una vez, ejecuta en cualquier lugar) permite

que el código compilado en Java se ejecute en cualquier plataforma que soporte una máquina virtual Java (JVM). Entre las características destacadas de Java se incluyen la robustez, la seguridad, la portabilidad y el alto rendimiento. *Spring Boot* [14] es un framework que simplifica el desarrollo de aplicaciones Java basadas en el ecosistema de Spring, proporcionando un entorno de configuración mínimo y convencional. Spring Boot facilita la creación de aplicaciones autónomas y listas para producción mediante el uso de dependencias y configuraciones automáticas. Entre sus principales características se encuentran la auto-configuración, la facilidad para crear microservicios, el empaquetado embebido y herramientas integradas para el monitoreo de aplicaciones. Spring Data JPA es un módulo de Spring que proporciona una abstracción sobre JPA (*Java Persistence API*), permitiendo a los desarrolladores realizar operaciones de persistencia de manera sencilla y con menos código. JPA es una especificación que permite el mapeo objeto-relacional (ORM) para gestionar datos en bases de datos relacionales de forma transparente. Con Spring Data JPA, se aprovechan las capacidades de JPA junto con la facilidad de uso y las configuraciones automáticas de Spring Boot.

2.2.2. IntelliJ IDEA

IntelliJ IDEA [15] es un entorno de desarrollo integrado (IDE) para el desarrollo de software, particularmente conocido por su soporte robusto para Java. Proporciona herramientas avanzadas de desarrollo, refactorización, depuración y pruebas. Algunas de sus características incluyen análisis de código en tiempo real, refactorización inteligente, integración con sistemas de control de versiones como *Git*, *SVN*, *Mercurial*, y una amplia gama de plugins para ampliar sus funcionalidades. IntelliJ IDEA es especialmente útil para manejar grandes bases de código y realizar refactorizaciones de manera segura.

2.2.3. .NET

.NET [16] es una plataforma de desarrollo de software desarrollada por Microsoft que proporciona un entorno controlado para desarrollar y ejecutar aplicaciones web. .NET soporta múltiples lenguajes de programación, siendo C# uno de los más populares. La plataforma incluye bibliotecas para el desarrollo de aplicaciones web y servicios en la nube. Entre sus características se encuentran la capacidad de desarrollar aplicaciones robustas y eficientes, así como herramientas y bibliotecas que facilitan el desarrollo rápido y eficiente. La plataforma .NET está optimizada para ofrecer alto rendimiento y escalabilidad, lo que la hace ideal para una amplia gama de aplicaciones web.

2.2.4. Visual Studio

Visual Studio [17] es un entorno de desarrollo integrado (IDE) de Microsoft utilizado principalmente para el desarrollo de aplicaciones en .NET. Ofrece una amplia gama de herramientas para el desarrollo, depuración y despliegue de aplicaciones web. Sus características incluyen un editor de código avanzado con resaltado de sintaxis, autocompletado y refactorización, herramientas potentes para depuración y diagnóstico de problemas en las aplicaciones, integración con Azure para facilitar el desarrollo y despliegue de aplicaciones en la nube de Microsoft, y soporte para múltiples lenguajes de

programación, como C#, VB.NET, F#, JavaScript, y TypeScript. Visual Studio proporciona una experiencia de desarrollo fluida y eficiente, aprovechando las capacidades avanzadas de la plataforma .NET.

2.3. Bases de datos

En esta sección se exploran los fundamentos teóricos de las bases de datos utilizadas en el proyecto, centrándose en PostgreSQL y Neo4j, seleccionadas por sus capacidades únicas en el manejo de datos estructurados y basados en grafos, respectivamente.

2.3.1. PostgreSQL: Gestión de Datos Relacionales

PostgreSQL [18] es un sistema de gestión de bases de datos relacional de código abierto conocido por su robustez y conformidad con los estándares SQL. Utiliza un modelo de datos relacional basado en tablas, filas y columnas para almacenar y organizar datos estructurados de manera eficiente. Ofrece soporte completo para transacciones ACID, garantizando la Atomicidad, Consistencia, Aislamiento y Durabilidad de las operaciones. Además, PostgreSQL es altamente escalable gracias a sus capacidades avanzadas de replicación y partición, permitiendo manejar grandes volúmenes de datos con eficacia. La extensibilidad es otra de sus fortalezas, permitiendo a los usuarios crear extensiones y funciones personalizadas para adaptarse a requerimientos específicos. En términos de seguridad, PostgreSQL cuenta con mecanismos robustos para proteger la integridad de los datos y del sistema en general.

2.3.2. Neo4j: Gestión de Datos Basados en Grafos

Neo4j [19] es una base de datos orientada a grafos diseñada para modelar y consultar datos altamente interconectados. Utiliza un modelo de datos basado en nodos (entidades) y relaciones (conexiones entre nodos) para representar estructuras complejas de datos de manera intuitiva y eficiente. Su modelo de grafo permite representar relaciones complejas entre entidades de manera natural, lo que es ideal para aplicaciones como redes sociales, análisis de redes, sistemas de recomendación y detección de fraudes. *Cypher* [20], el lenguaje de consulta de Neo4j, proporciona un enfoque declarativo y poderoso para explorar y manipular patrones en el grafo, facilitando análisis complejos y la extracción de información útil. Neo4j también destaca por su escalabilidad horizontal, permitiendo distribuir datos a través de múltiples nodos para mejorar el rendimiento y la capacidad de respuesta en aplicaciones con grandes volúmenes de datos y alta demanda de consultas complejas.

Capítulo 3. Análisis

3.1. Definición del sistema

Este sistema web para el análisis estático de programas Java mediante ProgQuery está diseñado para facilitar la creación y el intercambio de análisis, fomentando la colaboración entre desarrolladores de software. Proporciona una plataforma accesible y colaborativa, permitiendo a los usuarios cargar, analizar y compartir resultados de programas Java de manera eficiente. El sistema se compone de una Web API y una aplicación web, abordando las necesidades de diferentes tipos de usuarios. Utiliza PostgreSQL como base de datos principal para almacenar datos relacionales, como usuarios, programas, análisis y resultados, mientras que Neo4j se emplea para almacenar programas en forma de grafos.

El sistema está compuesto por cuatro microservicios principales y dos interfaces para interactuar con ellos, junto con dos bases de datos para almacenar los datos relacionales y los programas en forma de grafos.

Los microservicios son:

- Gestión de Usuarios: Se encarga de todas las operaciones relacionadas con la administración de usuarios en el sistema. Desde el registro inicial hasta la gestión de perfiles y la autenticación, su objetivo es proporcionar un acceso seguro y personalizado a cada usuario.
- Gestión de Programas: Se realizan las acciones relacionadas con la manipulación de programas Java. Los usuarios pueden cargar archivos individuales o proyectos comprimidos, así como importar programas desde repositorios externos como GitHub, GitLab y Bitbucket.
- Gestión de Análisis: Establece un sistema único para la gestión de análisis, ofreciendo opciones de visibilidad y flexibilidad en su manejo. Los usuarios pueden crear, visualizar, actualizar y eliminar análisis según sus necesidades específicas.
- Gestión de Resultados: Se encarga de ejecutar los análisis sobre los programas cargados, almacenar los resultados obtenidos y facilitar su acceso posterior para su revisión y análisis. Además, garantiza la integridad y disponibilidad de los datos de resultado.

Las dos interfaces principales para interactuar con estos microservicios son:

- Web API: Proporciona servicios web REST para la interacción externa con el sistema, permitiendo la construcción de aplicaciones cliente.
- Aplicación Web: Interfaz de usuario accesible desde cualquier navegador web, que permite a los usuarios cargar programas, ejecutar análisis y visualizar resultados.

Las bases de datos utilizadas son:

- PostgreSQL: Almacena datos relacionales de usuarios, programas, análisis y resultados.
- Neo4j: Utilizado para almacenar programas en forma de grafos.

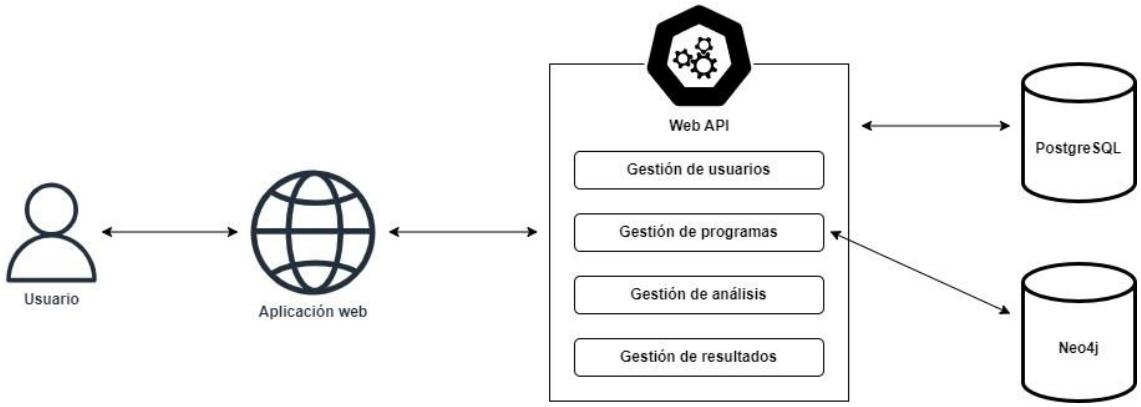


Ilustración 3. Diagrama del sistema

3.2. Requisitos del sistema

3.2.1. Requisitos funcionales

3.2.1.1. Requisitos funcionales de la gestión de autenticación

RFGAu1. El sistema permitirá a un usuario no autenticado registrarse.

RFGAu1.1. Solicitará los siguientes campos obligatorios: nombre, apellidos, email, confirmación email y contraseña.

RFGAu1.2. Validará los datos introducidos para el registro:

RFGAu1.2.1. Validará que ninguno de los campos esté vacío.

RFGAu1.2.2. Validará que el nombre tenga una longitud mínima de 3 caracteres y una máxima de 20.

RFGAu1.2.3. Validará que el email es no tenga una longitud máxima a 50 caracteres y corresponda a un formato de correo electrónico.

RFGAu1.2.4. Validará que el email y la confirmación del email coinciden.

RFGAu1.2.5. Validará que no existe en el sistema otro usuario con el mismo email.

RFGAu1.2.6. Validará que la contraseña tenga una longitud mínima de 6 caracteres y una máxima de 40.

RFGAu1.2.7. Si las validaciones anteriores son correctas:

RFGAu1.2.7.1. El nuevo usuario se registrará en el sistema.

RFGAu1.2.7.2. Redirigirá a la página de confirmación de registro del usuario.

RFGAu1.2.8. Si alguna de las validaciones anteriores es incorrecta, se redirigirá a una página de error con el código de error 400 y un mensaje descriptivo.

RFGAu2. El sistema permitirá a un usuario no autenticado iniciar sesión.

RFGAu2.1. Solicitará los siguientes campos obligatorios: email y contraseña.

RFGAu2.2. Comprobará que exista un usuario en la base de datos que coincida con los datos introducidos.

RFGAu2.2.1. Si la comprobación es correcta:

RFGAu2.2.1.1. Redirigirá a la página principal con la sesión del usuario iniciada.

RFGAu2.2.1.2. Generará un token que se usará para hacer las futuras peticiones al sistema.

RFGAu2.2.2. Si la comprobación es incorrecta, se redirigirá a una página de error con el código de error 401 y un mensaje descriptivo.

RFGAu3. El sistema permitirá a un usuario autenticado cerrar sesión.

RFGAu4. El sistema permitirá a un usuario autenticarse con Google.

RFGAu4.1. El sistema permitirá a un usuario iniciar sesión con Google.

RFGAu4.1.1. Si el usuario está registrado, iniciará sesión y redirigirá a la página principal.

RFGAu4.1.2. Si el usuario no está registrado, se registrará en el sistema, iniciará sesión y se redirigirá a la página principal.

RFGAu4.2. El sistema permitirá a un usuario registrarse con Google.

RFGAu4.2.1. Si el usuario ya está registrado, se redirigirá a una página de error con el código de error 400 y un mensaje descriptivo.

RFGAu4.2.2. Si el usuario no está registrado, se registrará en el sistema, iniciará sesión y se redirigirá a la página principal.

3.2.1.2. Requisitos funcionales de la gestión de usuarios

RFGU1. El sistema permitirá a un usuario administrador ver la lista de usuarios registrados.

RFGU1.1. El listado de usuarios mostrará los siguientes datos de cada usuario: id, email, nombre y apellidos.

RFGU2. El sistema permitirá a un usuario ver los detalles de un usuario registrado.

RFGU2.1. Se realizará una búsqueda del usuario por el email.

RFGU2.1.1. Si la búsqueda se realiza correctamente, se mostrará la siguiente información del usuario: nombre, apellidos, email, número de programas, número de análisis y número de resultados pertenecientes al usuario.

RFGU2.1.2. Si no se encuentra un usuario registrado con dicho email, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

RFGU3. El sistema permitirá a un usuario autenticado ver los programas a los que tenga acceso de un cierto usuario registrado y a un usuario no autenticado ver los programas públicos de este.

RFGU3.1. El listado de programas a los que tiene acceso un usuario mostrará la siguiente información de cada programa: id, rdn, nombre, descripción, path, base de datos, código fuente, url, opciones javac, fecha, visibilidad, estado, actualización, número de nodos, números de ficheros, líneas de código, mensaje de estado, número de análisis, email del propietario y nombre completo del propietario.

RFGU4. El sistema permitirá a un usuario autenticado ver los análisis a los que tenga acceso de un cierto usuario registrado y a un usuario no autenticado ver los análisis públicos de este.

RFGU4.1. El listado de análisis a los que tiene acceso un usuario mostrará la siguiente información de cada análisis: id, rdn, nombre, descripción, fecha, visibilidad, consulta, categoría, construcción sintáctica, email del propietario y nombre completo del propietario.

RFGU5. El sistema permitirá a un usuario administrador editar los datos de cualquier usuario registrado y a un usuario autenticado editar sus propios datos.

RFGU5.1. Los datos que se podrán editar son los siguientes: nombre, apellidos, correo y contraseña.

RFGU5.2. Se procederá a validad los datos con el mismo procedimiento que en **RFGAu1.2.**

RFGU5.3. Si no se encuentra el usuario que se desea editar, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

RFGU6. El sistema permitirá a un usuario administrador eliminar a cualquier usuario registrado y a un usuario autenticado eliminar su propio usuario.

RFGU6.1. Si no se encuentra el usuario que se desea eliminar, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

3.2.1.3. Requisitos funcionales de la gestión de programas

RFGP1. El sistema permitirá a un usuario administrador ver todos los programas registrados, a un usuario autenticado ver aquellos programas a los que tenga acceso y a un usuario no autenticado ver aquellos programas públicos.

RFGP1.1. El listado de programas a los que tenga acceso cierto usuario mostrará la siguiente información de cada programa: id, rdn, nombre, descripción, path, base de datos, código fuente, url, opciones javac, fecha, visibilidad, estado, actualización, número de nodos, números de ficheros, líneas de código, mensaje de estado, número de análisis, email del propietario y nombre completo del propietario.

RFGP2. El sistema permitirá ver un programa concreto a un usuario administrador, a un usuario autenticado que tenga acceso, y a un usuario no autenticado siempre y cuando el programa sea público.

RFGP2.1. Se realizará una búsqueda del programa por el rdn.

RFGP2.1.1. Si la búsqueda se realiza correctamente, se mostrará la siguiente información del programa: id, rdn, nombre, descripción, path, base de datos, código fuente, url, opciones javac, fecha, visibilidad, estado, actualización, número de nodos, números de ficheros, líneas de código, mensaje de estado, número de análisis, email del propietario y nombre completo del propietario.

RFGP2.1.2. Si no se encuentra un programa registrado con dicho rdn, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

RFGP3. El sistema permitirá ver los resultados de un programa concreto a un usuario administrador, a un usuario autenticado que tenga acceso y a un usuario no autenticado siempre y cuando el programa y los análisis utilizados sean públicos.

RFGP3.1. El listado de resultados de un programa concreto a los que tiene acceso un usuario mostrará la siguiente información de cada resultado: id del resultado, rdn del programa, comentario, fecha, puntuación, nodos inválidos, número de análisis, estado y actualización.

RFGP4. El sistema permitirá a un usuario autenticado registrar un nuevo programa.

RFGP4.1. Solicitará los siguientes campos obligatorios: nombre, rdn, visibilidad, aplicación java (fichero java, proyecto java en zip o url de Github); y los siguientes campos opcionales: descripción y opciones javac.

RFGP4.2. Validará que los campos obligatorios no estén vacíos.

RFGP4.2.1. Si las validaciones anteriores son correctas:

RFGP4.2.1.1. El nuevo programa se registrará en el sistema.

RFGP4.2.1.2. Redirigirá a la página de confirmación de registro del programa.

RFGP4.2.2. Si alguna de las validaciones anteriores es incorrecta, se redirigirá a una página de error con el código de error 400 y un mensaje descriptivo.

RFGP5. El sistema permitirá a un usuario autenticado actualizar un programa existente.

RFGP5.1. Los datos que se podrán editar son los siguientes: nombre, rdn, descripción, visibilidad, aplicación java y opciones javac.

RFGP5.2. Validará que los campos obligatorios no estén vacíos.

RFGP5.3. Si no se encuentra el programa que se desea editar, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

RFGP6. El sistema permitirá a un usuario autenticado eliminar un programa existente.

RFGP6.1. Si no se encuentra el programa que se desea eliminar, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

RFGP7. El sistema permitirá a un usuario autenticado ver un listado de los usuarios que tienen acceso a cualquier programa propio de visibilidad compartida.

RFGP7.1. El listado de usuarios mostrará los siguientes datos de cada usuario: id, email, nombre y apellidos.

RFGP8. El sistema permitirá a un usuario autenticado compartir un programa propio de visibilidad compartida con otro usuario autenticado.

RFGP8.1. Solicitará el correo electrónico del usuario al que se desea dar acceso.

RFGP8.1.1. Si existe dicho usuario, se le dará acceso al programa.

RFGP9. El sistema permitirá a un usuario autenticado ver los datos de cualquier usuario que tenga acceso a cualquier programa propio de visibilidad compartida.

RFGP9.1. Se realizará una búsqueda del usuario por el email.

RFGP9.1.1. Si la búsqueda se realiza correctamente, se mostrará la siguiente información del usuario: id, nombre, apellidos y email.

RFGP10. El sistema permitirá a un usuario autenticado eliminar a cualquier usuario que tenga acceso a cualquier programa propio de visibilidad compartida.

RFGP10.1. Se realizará una búsqueda del usuario por el email.

RFGP10.1.1. Si la búsqueda se realiza correctamente, se eliminará dicho usuario del listado de usuarios con acceso al programa.

3.2.1.4. Requisitos funcionales de la gestión de análisis

RFGAn1. El sistema permitirá a un usuario administrador ver todos los análisis registrados, a un usuario autenticado ver aquellos análisis a los que tenga acceso y a un usuario no autenticado ver aquellos análisis públicos.

RFGAn1.1. El listado de análisis a los que tenga acceso cierto usuario mostrará la siguiente información de cada análisis: id, rdn, nombre, descripción, fecha, visibilidad, consulta, categoría, construcción sintáctica, email del propietario y nombre completo del propietario.

RFGAn2. El sistema permitirá ver un análisis concreto a un usuario administrador, a un usuario autenticado que tenga acceso, y a un usuario no autenticado siempre y cuando el análisis sea público.

RFGAn2.1. Se realizará una búsqueda del análisis por el rdn.

RFGAn2.1.1. Si la búsqueda se realiza correctamente, se mostrará la siguiente información del análisis: id, rdn, nombre, descripción, fecha, visibilidad, consulta, categoría, construcción sintáctica, email del propietario y nombre completo del propietario.

RFGAn2.1.2. Si no se encuentra un análisis registrado con dicho rdn, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

RFGAn3. El sistema permitirá a un usuario autenticado registrar un nuevo análisis.

RFGAn3.1. Solicitará los siguientes campos obligatorios: nombre, rdn, visibilidad, categoría, construcción sintáctica y consulta Cypher; y el siguiente campo opcional: descripción.

RFGAn3.2. Validará que los campos obligatorios no estén vacíos.

RFGAn3.2.1. Si las validaciones anteriores son correctas:

RFGAn3.2.1.1. El nuevo análisis se registrará en el sistema.

RFGAn3.2.1.2. Redirigirá a la página de confirmación de registro del análisis.

RFGAn3.2.2. Si alguna de las validaciones anteriores es incorrecta, se redirigirá a una página de error con el código de error 400 y un mensaje descriptivo.

RFGAn4. El sistema permitirá a un usuario autenticado actualizar un análisis existente.

RFGAn4.1. Los datos que se podrán editar son los siguientes: nombre, rdn, descripción, visibilidad, categoría, construcción sintáctica y consulta Cypher**.**

RFGAn4.2. Validará que los campos obligatorios no estén vacíos.

RFGAn4.3. Si no se encuentra el análisis que se desea editar, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

RFGAn5. El sistema permitirá a un usuario autenticado eliminar un análisis existente.

RFGAn5.1. Si no se encuentra el análisis que se desea eliminar, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

RFGAn6. El sistema permitirá a un usuario autenticado ver un listado de los usuarios que tienen acceso a cualquier análisis propio de visibilidad compartida.

RFGAn6.1. El listado de usuarios mostrará los siguientes datos de cada usuario: id, email, nombre y apellidos.

RFGAn7. El sistema permitirá a un usuario autenticado compartir un análisis propio de visibilidad compartida con otro usuario autenticado.

RFGAn7.1. Solicitará el correo electrónico del usuario al que se desea dar acceso.

RFGAn7.1.1. Si existe dicho usuario, se le dará acceso al análisis.

RFGAn8. El sistema permitirá a un usuario autenticado ver los datos de cualquier usuario que tenga acceso a cualquier análisis propio de visibilidad compartida.

RFGAn8.1. Se realizará una búsqueda del usuario por el email.

RFGAn8.1.1. Si la búsqueda se realiza correctamente, se mostrará la siguiente información del usuario: id, nombre, apellidos y email.

RFGAn9. El sistema permitirá a un usuario autenticado eliminar a cualquier usuario que tenga acceso a cualquier análisis propio de visibilidad compartida.

RFGAn9.1. Se realizará una búsqueda del usuario por el email.

RFGAn9.1.1. Si la búsqueda se realiza correctamente, se eliminará dicho usuario del listado de usuarios con acceso al análisis.

3.2.1.5. Requisitos funcionales de la gestión de resultados

RFGR1. El sistema permitirá a un usuario autenticado ver todos los resultados de los análisis que ha realizado.

RFGR1.1. El listado de resultados mostrará la siguiente información de cada resultado: id, comentario, fecha, números de programas, número de análisis, número de ficheros, líneas de código, número de nodos, nodos comprobados, nodos inválidos, estado, actualización, nombre del propietario, nombre del email y programas asociados.

RFGR2. El sistema permitirá a un usuario administrador y a un usuario autenticado que tenga acceso ver los detalles del resultado concreto.

RFGR2.1. Se realizará una búsqueda del resultado por el id.

RFGR2.1.1. Si la búsqueda se realiza correctamente, se mostrará la siguiente información del resultado: id, comentario, fecha, números de programas, número de análisis, número de ficheros, líneas de código, número de nodos, nodos comprobados, nodos inválidos, estado, actualización, nombre del propietario, nombre del email y programas asociados.

RFGR2.1.2. Si no se encuentra un resultado registrado con dicho id, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

RFGR3. El sistema permitirá ver los detalles del resultado de un programa concreto a un usuario administrador, a un usuario autenticado que tenga acceso y a un usuario no autenticado siempre y cuando el programa y los análisis utilizados sean públicos.

RFGR3.1. Se realizará una búsqueda del programa por el rdn.

RFGR3.1.1. Si la búsqueda se realiza correctamente, se mostrará la siguiente información del resultado del programa: id, rdn, nombre, fecha, número de análisis, número de ficheros, líneas de código, número de nodos, nodos comprobados, nodos inválidos, actualización, nombre del propietario, nombre del email y análisis realizados.

RFGR3.1.2. Si no se encuentra un programa registrado con dicho rdn, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

RFGR4. El sistema permitirá a un usuario autenticado crear un nuevo resultado, es decir, realizar análisis a programas.

RFGR4.1. Solicitará los siguientes campos obligatorios: comentario, rdn de uno o varios programas y rdn de uno o varios análisis.

RFGR4.2. Validará que los campos obligatorios no estén vacíos.

RFGR4.3. Comprobará la existencia de los programas y análisis introducidos.

RFGR4.3.1. Si existe cada uno de ellos, se creará el nuevo resultado.

RFGR4.3.2. Si no se encuentra algún programa o análisis introducido, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

RFGR5. El sistema permitirá a un usuario administrador y al usuario propietario de un resultado concreto eliminar dicho resultado.

RFGR5.1. Si se elimina el resultado, en consecuencia, se eliminarán los resultados de cada programa asociado a dicho resultado.

RFGR5.2. Si no se encuentra el resultado que se desea eliminar, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

RFGR6. El sistema permitirá a un usuario administrador y al usuario propietario de un resultado concreto eliminar los resultados de cada uno de los programas asociados que tenga dicho resultado.

RFGR6.1. Si el resultado del programa que se ha eliminado es el único asociado al resultado concreto, en consecuencia, se eliminará el resultado concreto.

RFGR6.2. Si no se encuentra el resultado del programa que se desea eliminar, se redirigirá a una página de error con el código de error 404 y un mensaje descriptivo.

3.2.1.6. Requisitos funcionales adicionales

RFAd1. El sistema permitirá al usuario administrador ver un listado de estadísticas generales de la aplicación.

RFAd1.1. El listado de estadísticas mostrará los siguientes datos: número total de usuarios, número total de programas, número total de programas agrupados por estado, número total de programas agrupados por visibilidad, número total de programas agrupados por fecha de creación, número total de ficheros, número medio de fichero por programa, número total de nodos, número medio de nodos por programa, número medio de programas por usuario, número total de análisis, número total de análisis agrupados por categoría, número total de análisis agrupados por construcción sintáctica, número total de análisis agrupados por fecha de creación, número medio de análisis por usuario, número total de resultados, número medio de programas distintos por resultado, número medio de análisis distintos por resultado, número total de programas-análisis, número total de programas distintos, número total de análisis distintos, número medio de programas-análisis por programa, número medio de programas-análisis por análisis, número medio de programas-análisis por resultado, número total de programas-análisis agrupados por estado, número total de programas-análisis agrupados por fecha de creación, número total de nodos comprobados, número medio de nodos comprobados por programas-análisis, número total de nodos inválidos y número medio de nodos inválidos por programas-análisis.

RFAd2. El sistema permitirá a un usuario no autenticado analizar un programa que introduzca.

> **RFAd2.1.** Se solicitará la selección de un programa público o introducir un código java, y la selección de un análisis público o introducir una consulta Cypher.
>
> > **RFAd2.1.1.** Si los datos introducidos son correctos, se realizará el análisis correspondiente al programa y se mostrarán los resultados.

3.2.2. Requisitos no funcionales

RNF1. La comunicación entre la aplicación web y la API REST será en formato JSON, garantizando consistencia y eficiencia en las transmisiones de datos.

RNF2. Todas las comunicaciones entre la API REST y los clientes del sistema estarán cifradas utilizando certificados SSL, asegurando la seguridad de la información transmitida.

RNF3. La aplicación web debe implementar mecanismos seguros para la gestión y validación de tokens JWT utilizados en la autenticación y autorización de usuarios.

RNF4. La aplicación web deberá disponer de un manual de usuario, proporcionando respuestas a las posibles dudas que puedan surgir durante su utilización.

RNF5. La interfaz de la aplicación web será diseñada para ser usable e intuitiva, permitiendo que cualquier usuario pueda manejarla sin necesidad de experiencia previa.

RNF6. La API REST será implementada de manera que sea independiente del cliente que la consuma, facilitando la interoperabilidad y futuras integraciones.

RNF7. El sistema proporcionará mensajes de error descriptivos y comprensibles para los usuarios, facilitando la identificación y resolución de problemas.

RNF8. Todas las funcionalidades del sistema y las transacciones de negocio responderán al usuario en menos de 5 segundos, garantizando una experiencia ágil y eficiente.

RNF9. El sistema estará diseñado para manejar hasta 100 usuarios con sesiones concurrentes de manera efectiva, asegurando un rendimiento óptimo bajo carga máxima.

RNF10. La aplicación web debe ser compatible con una amplia gama de navegadores web.

3.3. Casos de uso y escenarios

En esta sección se detallan los actores del sistema, así como los casos de uso y los escenarios asociados.

3.3.1. Identificación de actores del sistema

A continuación, se identificarán los actores primarios del sistema, que son aquellos que interactúan directamente con él.

3.3.1.1. Actores primarios

Actor	Interacción con el sistema

Usuario no autenticado	Un usuario no autenticado es aquel que no ha iniciado sesión en el sistema.
Usuario autenticado	Un usuario autenticado es aquel que ha iniciado sesión en el sistema.
Usuario administrador	Un usuario administrador es aquel usuario autenticado con permisos especiales característicos de un administrador.

Tabla 1. Actores primarios del sistema

3.3.2. Especificación de casos de uso

En esta sección, se detallan los casos de uso del sistema asociados a cada uno de los actores identificados anteriormente.

3.3.2.1. Usuario no autenticado

Ilustración 4. Diagrama de casos de uso - Usuario no autenticado

3.3.2.1.1. CU-W-01 Registrarse en el sistema

Caso de uso	Registrarse en el sistema
Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-

Precondiciones	El usuario no puede estar registrado en el sistema
Postcondiciones	El usuario estará registrado en el sistema
Requisitos que satisface	**RFGAu1**
Propósito	
Registrar a un nuevo usuario en el sistema a través de la aplicación web que consume la Web API.	
Resumen de alto nivel	
Un usuario no autenticado accede a la aplicación web y se registra por medio de la página de registro, introduciendo los datos solicitados. Si estos son correctos, el usuario de registrará en el sistema y podrá iniciar sesión con las credenciales introducidas.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario accede a la aplicación web, pulsa sobre el botón *Login* y después sobre "Create an account". 2. Rellena los datos del formulario.	3. El sistema comprueba que los datos introducidos son válidos y que no existe un usuario con el mismo correo electrónico. 4. Registra el nuevo usuario mediante la API REST. 5. Redirecciona a una página de confirmación de registro.
Flujo de acciones de caminos alternativos	
3a. Si los datos introducidos no son válidos, se muestra un error y se vuelve al punto 2. 3b. Si existe un usuario con el mismo correo, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 4a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 2. Caso de uso: Registrarse en el sistema

3.3.2.1.2. CU-W-02 Identificarse en el sistema

Caso de uso	Identificarse en el sistema
Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario no puede haber iniciado sesión en el sistema
Postcondiciones	El usuario estará autenticado en el sistema
Requisitos que satisface	**RFGAu2**
Propósito	
Autenticar a un usuario en el sistema a través de la aplicación web que consume la Web API.	

Resumen de alto nivel	
Un usuario no autenticado accede a la aplicación web e inicia sesión por medio de la página de Login, introduciendo las credenciales indicadas, email y contraseña. Si estas son correctas, se autenticará en el sistema.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario accede a la aplicación web y pulsa sobre el botón *Login*. 2. Introduce las credenciales para iniciar sesión.	3. El sistema comprueba que los datos introducidos son válidos y corresponden con un usuario registrado en el sistema mediante la API REST. 4. Redirecciona a la página principal con la sesión iniciada.
Flujo de acciones de caminos alternativos	
3a. Si los datos introducidos no son válidos, se muestra un error y se vuelve al punto 2. 3b. Si no existe un usuario con esas credenciales, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 3. Caso de uso: Identificarse en el sistema

3.3.2.1.3. CU-W-03 Registrarse en el sistema con Google

Caso de uso	Registrarse en el sistema con Google
Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario no puede estar registrado en el sistema
Postcondiciones	El usuario estará registrado en el sistema
Requisitos que satisface	**RFGAu4.2**
Propósito	
Registrar a un nuevo usuario en el sistema a través de la autenticación con Google de la aplicación web que consume la Web API.	
Resumen de alto nivel	
Un usuario no autenticado accede a la aplicación web, se redirige a la página de registro y se registra por medio de la autenticación de Google. Si no existe el usuario en el sistema, el usuario de registrará en el sistema y podrá iniciar sesión con la cuenta de Google registrada.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA

1. El usuario accede a la aplicación web, pulsa sobre el botón *Login* y después sobre "Create an account". 2. Pulsa sobre el botón "Sign un with Google" e selecciona la cuenta de Google con la que se quiere registrar en el sistema.	3. El sistema comprueba que no existe un usuario con el mismo correo electrónico mediante la API REST. 4. Redirecciona a la página principal con la sesión iniciada.
Flujo de acciones de caminos alternativos	
3a. Si existe un usuario con el mismo correo, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 4. Caso de uso: Registrarse en el sistema con Google

3.3.2.1.4. CU-W-04 Identificarse en el sistema con Google

Caso de uso	Identificarse en el sistema con Google
Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario no puede haber iniciado sesión en el sistema
Postcondiciones	El usuario estará autenticado en el sistema
Requisitos que satisface	**RFGAu4.1**
Propósito	
Autenticar a un usuario en el sistema a través de la autenticación con Google de la aplicación web que consume la Web API.	
Resumen de alto nivel	
Un usuario no autenticado accede a la aplicación web, se redirige a la página de Login e inicia sesión por medio de la autenticación de Google. Si existe el usuario en el sistema, se autenticará. Si no existe, se registrará en el sistema y se autenticará.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario accede a la aplicación web, pulsa sobre el botón *Login* y después sobre "Sign in with Google". 2. Selecciona la cuenta de Google con la que iniciar sesión en el sistema.	3. El sistema comprueba que coinciden los datos de la cuenta con un usuario registrado mediante la API REST. 4. Redirecciona a la página principal con la sesión iniciada.
Flujo de acciones de caminos alternativos	
3a. Si no existe el usuario, el sistema lo registrará y se continúa en el punto 4.	

Tabla 5. Caso de uso: Identificarse en el sistema con Google

3.3.2.1.5. CU-W-05 Visualizar los programas públicos

Caso de uso	Visualizar los programas públicos

Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario no puede haber iniciado sesión en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGP1**
Propósito	
Visualizar un listado de los programas públicos registrados en el sistema sin autenticarse.	
Resumen de alto nivel	
Un usuario no autenticado accede a la aplicación web, se redirige a la página de programas y visualiza un listado de los programas púbicos que hay registrados en el sistema.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario accede a la aplicación web y pulsa sobre el botón *Programs*.	2. El sistema consulta a la API REST los programas registrados con visibilidad pública. 3. Redirecciona a una página con el listado de dichos programas.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 6. Caso de uso: Visualizar los programas públicos

3.3.2.1.6. CU-W-06 Visualizar los detalles de un programa público

Caso de uso	Visualizar los detalles de un programa público
Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario no puede haber iniciado sesión en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGP2**
Propósito	
Visualizar los detalles de un programa público registrado en el sistema sin autenticarse.	
Resumen de alto nivel	

Un usuario no autenticado accede a la aplicación web, se redirige a la página de programas, visualiza un listado de los programas púbicos que hay registrados en el sistema y pulsa sobre uno de ellos. Se redireccionará a una página con los detalles del programa.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario accede a la aplicación web y pulsa sobre el botón *Programs*.	2. El sistema consulta a la API REST los programas registrados con visibilidad pública. 3. Redirecciona a una página con el listado de dichos programas.
4. El usuario pulsa sobre un programa del listado.	5. El sistema consulta a la API REST los detalles de dicho programa. 6. Redirecciona a una página con los detalles del programa.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 7. Caso de uso: Visualizar los detalles de un programa público

3.3.2.1.7. CU-W-07 Visualizar el listado de resultados de un programa público

Caso de uso	Visualizar el listado de resultados de un programa público
Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario no puede haber iniciado sesión en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGP3**
Propósito	
Visualizar los resultados de un programa público registrado en el sistema sin autenticarse.	
Resumen de alto nivel	
Un usuario no autenticado accede a la aplicación web, se redirige a la página de programas, visualiza un listado de los programas púbicos que hay registrados en el sistema y pulsa sobre uno de ellos. Se redireccionará a una página con los detalles del programa. El usuario pulsa sobre el icono de resultados y visualiza un listado con los resultados visibles asociados al programa.	

Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario accede a la aplicación web y pulsa sobre el botón *Programs*.	2. El sistema consulta a la API REST los programas registrados con visibilidad pública. 3. Redirecciona a una página con el listado de dichos programas.
4. El usuario pulsa sobre un programa del listado.	5. El sistema consulta a la API REST los detalles de dicho programa. 6. Redirecciona a una página con los detalles del programa.
7. El usuario pulsa sobre el icono de resultados.	8. El sistema consulta a la API REST los resultados visibles asociados al programa. 9. Redirecciona a una página con el listado de dichos resultados.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 8. Caso de uso: Visualizar el listado de resultados de un programa público

3.3.2.1.8. CU-W-08 Visualizar los detalles del resultado de un programa público

Caso de uso	Visualizar los detalles del resultado de un programa público
Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario no puede haber iniciado sesión en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGR3**
Propósito	
Visualizar los detalles del resultado visible de un programa público registrado en el sistema sin autenticarse.	
Resumen de alto nivel	
Un usuario no autenticado accede a la aplicación web, se redirige a la página de programas, visualiza un listado de los programas púbicos que hay registrados en el sistema y pulsa sobre uno de ellos. Se redireccionará a una página con los detalles del	

programa. El usuario pulsa sobre el icono de resultados y visualiza un listado con los resultados visibles asociados al programa. El usuario pulsa sobre uno de los resultados asociados al programa. Se redireccionará a una página con los detalles del resultado del programa.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario accede a la aplicación web y pulsa sobre el botón *Programs*.	2. El sistema consulta a la API REST los programas registrados con visibilidad pública. 3. Redirecciona a una página con el listado de dichos programas.
4. El usuario pulsa sobre un programa del listado.	5. El sistema consulta a la API REST los detalles de dicho programa. 6. Redirecciona a una página con los detalles del programa.
7. El usuario pulsa sobre el icono de resultados.	8. El sistema consulta a la API REST los resultados visibles asociados al programa. 9. Redirecciona a una página con el listado de dichos resultados.
10. El usuario pulsa sobre un resultado visible.	11. El sistema consulta a la API REST los detalles dicho resultado. 12. Redirecciona a una página con los detalles del resultado del programa.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 11a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 9. Caso de uso: Visualizar los detalles del resultado de un programa público

3.3.2.1.9. CU-W-09 Visualizar los análisis públicos

Caso de uso	Visualizar los análisis públicos
Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario no puede haber iniciado sesión en el sistema

Postcondiciones	-
Requisitos que satisface	**RFGAn1**
Propósito	
Visualizar un listado de los análisis públicos registrados en el sistema sin autenticarse.	
Resumen de alto nivel	
Un usuario no autenticado accede a la aplicación web, se redirige a la página de análisis y visualiza un listado de los análisis púbicos que hay registrados en el sistema.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario accede a la aplicación web y pulsa sobre el botón *Analyses*.	2. El sistema consulta a la API REST los análisis registrados con visibilidad pública. 3. Redirecciona a una página con el listado de dichos análisis.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 10. Caso de uso: Visualizar los análisis públicos

3.3.2.1.10. CU-W-10 Visualizar los detalles de un análisis público

Caso de uso	Visualizar los detalles de un análisis público
Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario no puede haber iniciado sesión en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGAn2**
Propósito	
Visualizar los detalles de un análisis público registrado en el sistema sin autenticarse.	
Resumen de alto nivel	
Un usuario no autenticado accede a la aplicación web, se redirige a la página de análisis, visualiza un listado de los análisis púbicos que hay registrados en el sistema y pulsa sobre uno de ellos. Se redireccionará a una página con los detalles del análisis.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA

1. El usuario accede a la aplicación web y pulsa sobre el botón *Analyses*.	2. El sistema consulta a la API REST los análisis registrados con visibilidad pública. 3. Redirecciona a una página con el listado de dichos análisis.
4. El usuario pulsa sobre un análisis del listado.	5. El sistema consulta a la API REST los detalles de dicho análisis. 6. Redirecciona a una página con los detalles del análisis.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 11. Caso de uso: Visualizar los detalles de un análisis público

3.3.2.1.11. CU-W-11 Visualizar los detalles de un usuario

Caso de uso	Visualizar los detalles de un usuario
Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario no puede haber iniciado sesión en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGU2**
Propósito	
Visualizar los detalles de un usuario registrado en el sistema sin autenticarse.	
Resumen de alto nivel	
Un usuario no autenticado accede a la aplicación web, se redirige a la página de programas o análisis y visualiza un listado de los programas o análisis púbicos que hay registrados en el sistema. Cada uno de ellos mostrará el correo electrónico del propietario. El usuario pulsa sobre dicho correo electrónico y visualiza los detalles de dicho propietario.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario accede a la aplicación web y pulsa sobre el botón *Programs* o *Analyses*.	2. El sistema consulta a la API REST los programas o análisis registrados con visibilidad pública. 3. Redirecciona a una página con el listado de dichos programas o análisis.

4. El usuario pulsa sobre el correo electrónico del propietario de un programa o análisis.	5. El sistema consulta a la API REST los detalles de dicho propietario. 6. Redirecciona a una página con los detalles del usuario.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 12. Caso de uso: Visualizar los detalles de un usuario

3.3.2.1.12. CU-W-12 Visualizar los programas públicos de un usuario

Caso de uso	Visualizar los programas públicos de un usuario
Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario no puede haber iniciado sesión en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGU3**
Propósito	
Visualizar un listado de los programas públicos de un usuario registrado en el sistema sin autenticarse.	
Resumen de alto nivel	
Un usuario no autenticado accede a la aplicación web, se redirige a la página de programas o análisis y visualiza un listado de los programas o análisis púbicos que hay registrados en el sistema. Cada uno de ellos mostrará el correo electrónico del propietario. El usuario pulsa sobre dicho correo electrónico y visualiza los detalles de dicho propietario. El usuario pulsa sobre el icono de programas y visualiza un listado con los programas públicos del otro usuario.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario accede a la aplicación web y pulsa sobre el botón *Programs* o *Analyses*.	2. El sistema consulta a la API REST los programas o análisis registrados con visibilidad pública. 3. Redirecciona a una página con el listado de dichos programas o análisis.
4. El usuario pulsa sobre el correo electrónico del propietario de un programa o análisis.	5. El sistema consulta a la API REST los detalles de dicho propietario.

	6. Redirecciona a una página con los detalles del usuario.
7. El usuario pulsa sobre el icono de programas.	8. El sistema consulta a la API REST los programas públicos asociados al otro usuario. 9. Redirecciona a una página con el listado de dichos programas.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 13. Caso de uso: Visualizar los programas públicos de un usuario

3.3.2.1.13. CU-W-13 Visualizar los análisis públicos de un usuario

Caso de uso	Visualizar los análisis públicos de un usuario registrado sin autenticarse
Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario no puede haber iniciado sesión en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGU4**
Propósito	
Visualizar un listado de los análisis públicos de un usuario registrado en el sistema sin autenticarse.	
Resumen de alto nivel	
Un usuario no autenticado accede a la aplicación web, se redirige a la página de programas o análisis y visualiza un listado de los programas o análisis púbicos que hay registrados en el sistema. Cada uno de ellos mostrará el correo electrónico del propietario. El usuario pulsa sobre dicho correo electrónico y visualiza los detalles de dicho propietario. El usuario pulsa sobre el icono de análisis y visualiza un listado con los análisis públicos del otro usuario.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario accede a la aplicación web y pulsa sobre el botón *Programs* o *Analyses*.	2. El sistema consulta a la API REST los programas o análisis registrados con visibilidad pública.

	3. Redirecciona a una página con el listado de dichos programas o análisis.
4. El usuario pulsa sobre el correo electrónico del propietario de un programa o análisis.	5. El sistema consulta a la API REST los detalles de dicho propietario. 6. Redirecciona a una página con los detalles del usuario.
7. El usuario pulsa sobre el icono de análisis.	8. El sistema consulta a la API REST los análisis públicos asociados al otro usuario. 9. Redirecciona a una página con el listado de dichos análisis.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 14. Caso de uso: Visualizar los análisis públicos de un usuario

3.3.2.1.14. CU-W-14 Analizar código java sin autenticarse

Caso de uso	Analizar código java sin autenticarse
Actores	Usuario no autenticado
Iniciado por	Usuario no autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario no puede haber iniciado sesión en el sistema
Postcondiciones	-
Requisitos que satisface	**RFAd2**
Propósito	
Hacer un análisis de un código java sin autenticarse.	
Resumen de alto nivel	
Un usuario no autenticado accede a la aplicación web, se redirige a la página de análisis básico, selecciona un programa o introduce un código java, y selecciona un análisis o introduce una consulta Cypher y analiza el código.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario accede a la página web, selecciona la quinta slide y pulsa sobre “Click here” para acceder a la página de análisis básico.	3. El sistema consulta a la API REST para realizar el análisis de dicho programa.

2. Selecciona un programa o introduce un código java, y selecciona un análisis o introduce una consulta Cypher y analiza.	4. Muestra los detalles del resultado obtenido.
Flujo de acciones de caminos alternativos	
3a. Si existe algún error en los datos introducidos, se muestra en la aplicación web y se vuelve al punto 2. 3b. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 15. Caso de uso: Analizar código java sin autenticarse

3.3.2.2. Usuario autenticado

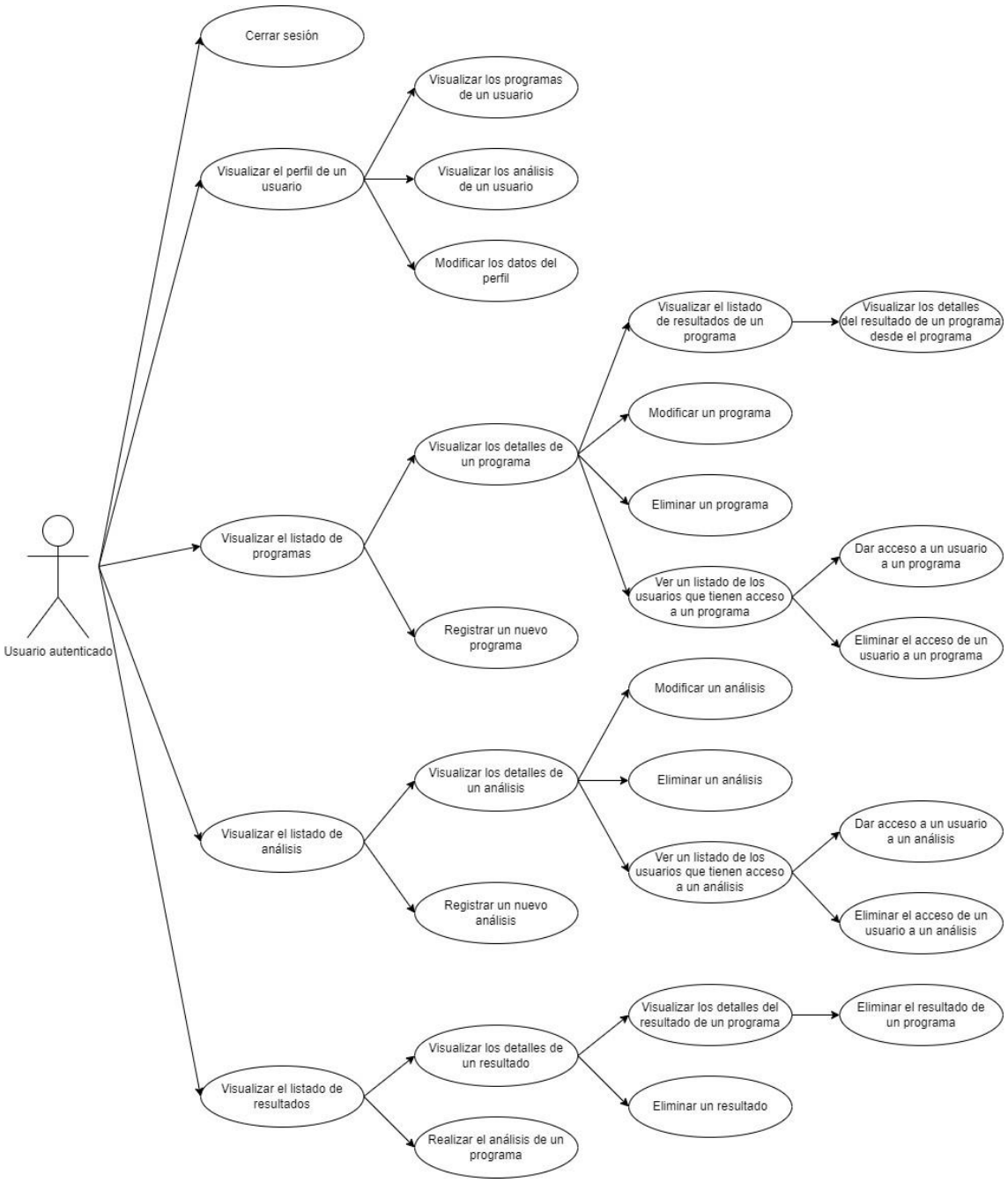


Ilustración 5. Diagrama de casos de uso - Usuario autenticado

3.3.2.2.1. CU-W-15 Cerrar sesión

Caso de uso	Cerrar sesión
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema

Postcondiciones	El usuario no estará autenticado en el sistema
Requisitos que satisface	**RFGAu3**
Propósito	
Cerrar la sesión en la aplicación web.	
Resumen de alto nivel	
Un usuario autenticado cierra sesión en la aplicación web.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre su nombre y sobre el botón *Logout*.	2. El sistema elimina los datos de sesión para cerrar la sesión del usuario. 3. Redirecciona a la página principal.
Flujo de acciones de caminos alternativos	

Tabla 16. Caso de uso: Cerrar sesión

3.3.2.2.2. CU-W-16 Visualizar el perfil de un usuario

Caso de uso	Visualizar el perfil de un usuario
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGU2**
Propósito	
Visualizar los detalles de un usuario registrado en el sistema.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de programas o análisis y visualiza un listado de los programas o análisis a los que tiene acceso. Cada uno de ellos mostrará el correo electrónico del propietario. El usuario pulsa sobre dicho correo electrónico y visualiza los detalles de dicho propietario.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Programs* o *Analyses*.	2. El sistema consulta a la API REST los programas o análisis a los que tiene acceso el usuario.

	3. Redirecciona a una página con el listado de dichos programas o análisis.
4. El usuario pulsa sobre el correo electrónico del propietario de un programa o análisis.	5. El sistema consulta a la API REST los detalles de dicho propietario. 6. Redirecciona a una página con los detalles del usuario.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 17. Caso de uso: Visualizar el perfil de un usuario

3.3.2.2.3. CU-W-17 Visualizar los programas de un usuario

Caso de uso	Visualizar los programas de un usuario
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGU3**
Propósito	
Visualizar un listado de los programas visibles de un usuario registrado.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de programas o análisis y visualiza un listado de los programas o análisis a los que tiene acceso. Cada uno de ellos mostrará el correo electrónico del propietario. El usuario pulsa sobre dicho correo electrónico y visualiza los detalles de dicho propietario. El usuario pulsa sobre el icono de programas y visualiza un listado con los programas visibles del otro usuario.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Programs* o *Analyses*.	2. El sistema consulta a la API REST los programas o análisis a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos programas o análisis.
4. El usuario pulsa sobre el correo electrónico del propietario de un programa o análisis.	5. El sistema consulta a la API REST los detalles de dicho propietario. 6. Redirecciona a una página con los detalles del usuario.

7. El usuario pulsa sobre el icono de programas.	8. El sistema consulta a la API REST los programas visibles asociados al otro usuario. 9. Redirecciona a una página con el listado de dichos programas.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 18. Caso de uso: Visualizar los programas de un usuario

3.3.2.2.4. CU-W-18 Visualizar los análisis de un usuario

Caso de uso	Visualizar los análisis de un usuario
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGU4**
Propósito	
Visualizar un listado de los análisis visibles de un usuario registrado	
Resumen de alto nivel	
Un usuario autenticado accede a la página de programas o análisis y visualiza un listado de los programas o análisis a los que tiene acceso. Cada uno de ellos mostrará el correo electrónico del propietario. El usuario pulsa sobre dicho correo electrónico y visualiza los detalles de dicho propietario. El usuario pulsa sobre el icono de análisis y visualiza un listado con los análisis visibles del otro usuario.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Programs* o *Analyses*.	2. El sistema consulta a la API REST los programas o análisis a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos programas o análisis.
4. El usuario pulsa sobre el correo electrónico del propietario de un programa o análisis.	5. El sistema consulta a la API REST los detalles de dicho propietario. 6. Redirecciona a una página con los detalles del usuario.

7. El usuario pulsa sobre el icono de análisis.	8. El sistema consulta a la API REST los análisis visibles asociados al otro usuario. 9. Redirecciona a una página con el listado de dichos análisis.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 19. Caso de uso: Visualizar los análisis de un usuario

3.3.2.2.5. CU-W-19 Modificar los datos del perfil

Caso de uso	Modificar los datos del perfil
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGU6**
Propósito	
Modificar los datos del perfil del usuario propio.	
Resumen de alto nivel	
Un usuario autenticado modificará algún dato de su perfil y guardará dicha modificación.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre su nombre y sobre el botón *Profile*.	2. El sistema consulta a la API REST los datos de perfil del usuario. 3. Redirecciona a una página con los detalles del usuario.
4. El usuario pulsa sobre el icono de editar.	5. El sistema consulta a la API REST los datos a editar del usuario. 6. Redirecciona a una página de edición de los datos del usuario.
7. El usuario modifica sus datos y pulsa sobre el botón “Save user”	8. El sistema consulta a la API REST para modificar los datos del usuario. 9. Redirecciona a una página con los detalles del usuario actualizados.
Flujo de acciones de caminos alternativos	

2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.

Tabla 20. Caso de uso: Modificar los datos del perfil

3.3.2.2.6. CU-W-20 Visualizar el listado de programas

Caso de uso	Visualizar el listado de programas
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGP1**
Propósito	
Visualizar un listado de los programas visibles del sistema.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de programas y visualiza un listado de los programas visibles del sistema.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Programs*.	2. El sistema consulta a la API REST los programas a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos programas.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 21. Caso de uso: Visualizar el listado de programas

3.3.2.2.7. CU-W-21 Visualizar los detalles de un programa

Caso de uso	Visualizar los detalles de un programa
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-

Requisitos que satisface	**RFGP2**
Propósito	
Visualizar los detalles de un programa visible del sistema.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de programas, visualiza un listado de los programas visibles del sistema y pulsa sobre uno de ellos. Se redireccionará a una página con los detalles del programa.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Programs*.	2. El sistema consulta a la API REST los programas a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos programas.
4. El usuario pulsa sobre un programa del listado.	5. El sistema consulta a la API REST los detalles de dicho programa. 6. Redirecciona a una página con los detalles del programa.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 22. Caso de uso: Visualizar los detalles de un programa

3.3.2.2.8. CU-W-22 Visualizar el listado de resultados de un programa

Caso de uso	Visualizar el listado de resultados de un programa
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGP3**
Propósito	
Visualizar el listado de resultado de un programa visible del sistema.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de programas, visualiza un listado de los programas visibles del sistema y pulsa sobre uno de ellos. Se redireccionará a una página con los detalles del programa. El usuario pulsa sobre el icono de resultados y visualiza un listado con los resultados visibles asociados al programa.	

Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Programs*.	2. El sistema consulta a la API REST los programas a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos programas.
4. El usuario pulsa sobre un programa del listado.	5. El sistema consulta a la API REST los detalles de dicho programa. 6. Redirecciona a una página con los detalles del programa.
7. El usuario pulsa sobre el icono de resultados.	8. El sistema consulta a la API REST los resultados asociados al programa a los que tiene acceso el usuario. 9. Redirecciona a una página con el listado de dichos resultados.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 23. Caso de uso: Visualizar el listado de resultados de un programa

3.3.2.2.9. CU-W-23 Visualizar los detalles del resultado de un programa desde el programa

Caso de uso	Visualizar los detalles del resultado de un programa desde el programa
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGR3**
Propósito	
Visualizar los detalles del resultado visible de un programa registrado en el sistema.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de programas, visualiza un listado de los programas visibles del sistema y pulsa sobre uno de ellos. Se redireccionará a una página con los detalles del programa. El usuario pulsa sobre el icono de resultados y visualiza un listado con los resultados visibles asociados al programa. El usuario pulsa sobre uno de	

los resultados asociados al programa. Se redireccionará a una página con los detalles del resultado del programa.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Programs*.	2. El sistema consulta a la API REST los programas a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos programas.
4. El usuario pulsa sobre un programa del listado.	5. El sistema consulta a la API REST los detalles de dicho programa. 6. Redirecciona a una página con los detalles del programa.
7. El usuario pulsa sobre el icono de resultados.	8. El sistema consulta a la API REST los resultados asociados al programa a los que tiene acceso el usuario. 9. Redirecciona a una página con el listado de dichos resultados.
10. El usuario pulsa sobre un resultado visible.	11. El sistema consulta a la API REST los detalles dicho resultado. 12. Redirecciona a una página con los detalles del resultado del programa.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 11a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 24. Caso de uso: Visualizar los detalles del resultado de un programa desde el programa

3.3.2.2.10. CU-W-24 Registrar un nuevo programa

Caso de uso	Registrar un nuevo programa
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGP4**

<table>
<tr><td colspan="2">Propósito</td></tr>
<tr><td colspan="2">Registrar un nuevo programa en el sistema.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Resumen de alto nivel</td></tr>
<tr><td colspan="2">Un usuario autenticado accede a la página de programas y pulsa sobre el botón de nuevo programa. Se redireccionará a un formulario con los datos a introducir del programa, el usuario rellena dicho formulario y pulsa sobre insertar el programa. Se redireccionará a una página de confirmación del registro del programa.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Descripción detallada</td></tr>
<tr><td colspan="2">Flujo de acciones del camino básico</td></tr>
<tr><td>ACTOR</td><td>SISTEMA</td></tr>
<tr><td>El usuario pulsa sobre el botón Programs y sobre el botón New.
Rellena los datos del formulario y pulsa sobre “Insert program”.</td><td><ol start="3">
El sistema comprueba que los datos introducidos son válidos y que no existe ningún programa con el mismo rdn.
Registra el nuevo programa mediante la API REST.
Redirecciona a una página de confirmación de registro del programa.</td></tr>
<tr><td colspan="2">Flujo de acciones de caminos alternativos</td></tr>
<tr><td colspan="2">3a. Si los datos introducidos no son válidos, se muestra un error y se vuelve al punto 2.
3b. Si existe un programa con el mismo rdn, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.
4a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.</td></tr>
</table>

Tabla 25. Caso de uso: Registrar un nuevo programa

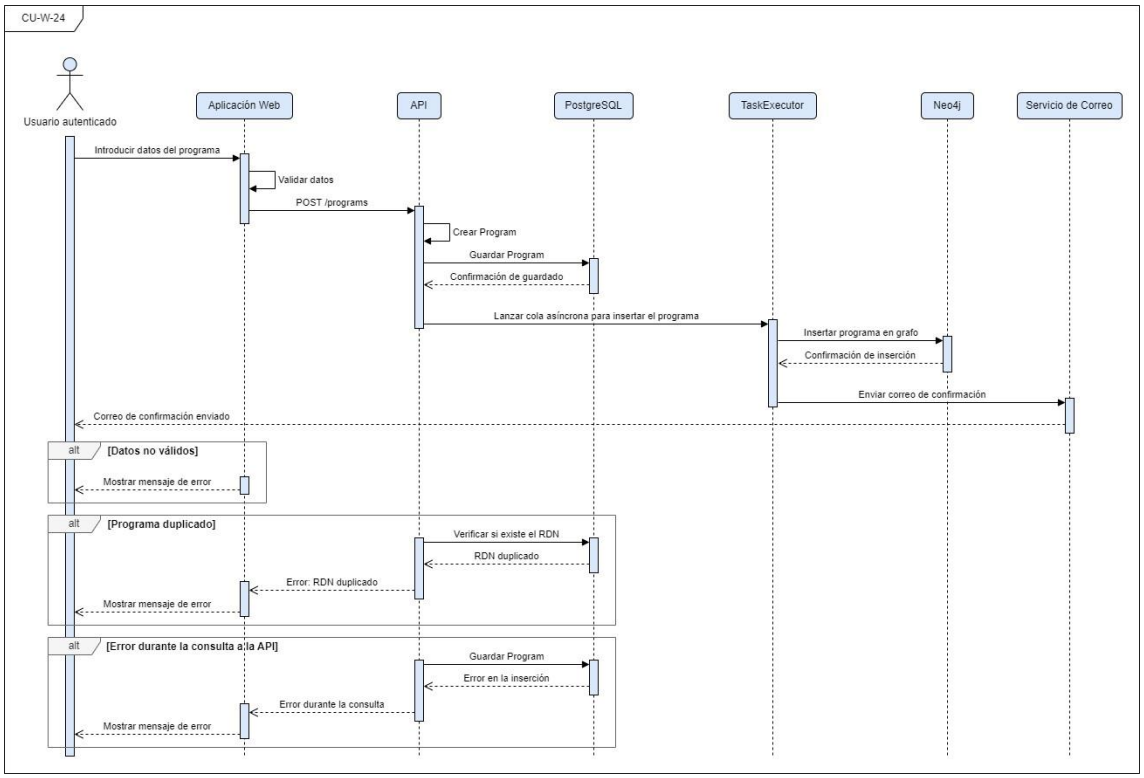


Ilustración 4. Diagrama de interacción CU-W-24

3.3.2.2.11. CU-W-25 Modificar un programa

Caso de uso	Modificar un programa
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGP5**

Propósito
Modificar los datos de un programa propio.
Resumen de alto nivel
Un usuario autenticado accede a la página de programas, visualiza un listado de los programas visibles del sistema y pulsa sobre un programa propio. Se redireccionará a una página con los detalles del programa. El usuario pulsa sobre el icono de editar y modifica los datos de su programa.
Descripción detallada
Flujo de acciones del camino básico

ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Programs*.	2. El sistema consulta a la API REST los programas a los que tiene acceso el usuario.

	3. Redirecciona a una página con el listado de dichos programas.
4. El usuario pulsa sobre uno de sus propios programas.	5. El sistema consulta a la API REST los detalles de dicho programa. 6. Redirecciona a una página con los detalles del programa.
7. El usuario pulsa sobre el icono de editar. 8. Modifica los datos del programa y pulsa sobre "Save program".	9. El sistema comprueba que los datos introducidos son válidos. 10. Actualiza la información del programa mediante la API REST. 11. Redirecciona a la página de detalles del programa.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 9a. Si los datos introducidos no son válidos, se muestra un error y se vuelve al punto 8. 10a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 26. Caso de uso: Modificar un programa

3.3.2.2.12. CU-W-26 Eliminar un programa

Caso de uso	Eliminar un programa
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGP6**
Propósito	
Eliminar un programa propio.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de programas, visualiza un listado de los programas visibles del sistema y pulsa sobre un programa propio. Se redireccionará a una página con los detalles del programa. El usuario pulsa sobre el icono de eliminar y confirma la eliminación del programa.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Programs*.	2. El sistema consulta a la API REST los programas a los que tiene acceso el usuario.

	3. Redirecciona a una página con el listado de dichos programas.
4. El usuario pulsa sobre uno de sus propios programas.	5. El sistema consulta a la API REST los detalles de dicho programa. 6. Redirecciona a una página con los detalles del programa.
7. El usuario pulsa sobre el icono de eliminar y confirma la eliminación de su programa.	8. El sistema elimina el programa mediante la API REST. 9. Redirecciona a la página del listado de los programas del usuario.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 27. Caso de uso: Eliminar un programa

3.3.2.2.13. CU-W-27 Ver un listado de los usuarios que tienen acceso a un programa

Caso de uso	Ver un listado de los usuarios que tienen acceso a un programa
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGP7**
Propósito	
Visualizar el listado de los usuarios que tiene acceso a un programa propio de visibilidad compartida.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de programas, visualiza un listado de los programas visibles del sistema y pulsa sobre un programa propio de visibilidad compartida. Se redireccionará a una página con los detalles del programa, pulsa sobre el listado desplegable “Shared Users” y visualiza los usuarios que tienen acceso al programa.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Programs*.	2. El sistema consulta a la API REST los programas a los que tiene acceso el usuario.

	3. Redirecciona a una página con el listado de dichos programas.
4. El usuario pulsa sobre uno de sus propios programas de visibilidad compartida.	5. El sistema consulta a la API REST los detalles de dicho programa. 6. Redirecciona a una página con los detalles del programa.
7. El usuario pulsa sobre el listado desplegable "Shared Users".	8. El sistema muestra el listado de usuarios.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 28. Caso de uso: Ver un listado de los usuarios que tienen acceso a un programa

3.3.2.2.14. CU-W-28 Dar acceso a un usuario a un programa

Caso de uso	Dar acceso a un usuario a un programa
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGP8**
Propósito	
Dar acceso a un usuario registrado a un programa propio de visibilidad compartida.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de programas, visualiza un listado de los programas visibles del sistema y pulsa sobre un programa propio de visibilidad compartida. Se redireccionará a una página con los detalles del programa, pulsa sobre el listado desplegable "Shared Users", pulsa sobre el botón "Add user", introduce el email de un usuario registrado y pulsa sobre el botón "Add".	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Programs*.	2. El sistema consulta a la API REST los programas a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos programas.
4. El usuario pulsa sobre uno de sus propios programas de visibilidad compartida.	5. El sistema consulta a la API REST los detalles de dicho programa. 6. Redirecciona a una página con los detalles del programa.

7. El usuario pulsa sobre el listado desplegable "Shared Users" y sobre el botón "Add user". 8. Introduce el email de un usuario registrado y pulsa sobre el botón "Add".	9. El sistema añade al usuario en el listado de usuarios compartidos mediante la API REST.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 9a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 29. Caso de uso: Dar acceso a un usuario a un programa

3.3.2.2.15. CU-W-29 Eliminar el acceso de un usuario a un programa

Caso de uso	Eliminar el acceso de un usuario a un programa
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGP10**
Propósito	
Eliminar el acceso de un usuario a un programa propio de visibilidad compartida.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de programas, visualiza un listado de los programas visibles del sistema y pulsa sobre un programa propio de visibilidad compartida. Se redireccionará a una página con los detalles del programa, pulsa sobre el listado desplegable "Shared Users" y visualiza los usuarios que tienen acceso al programa. El usuario pulsa sobre el botón "Remove" asociado al usuario que desea eliminarle el acceso al programa.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Programs*.	2. El sistema consulta a la API REST los programas a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos programas.

4. El usuario pulsa sobre uno de sus propios programas de visibilidad compartida.	5. El sistema consulta a la API REST los detalles de dicho programa. 6. Redirecciona a una página con los detalles del programa.
7. El usuario pulsa sobre el listado desplegable "Shared Users".	8. El sistema muestra el listado de usuarios.
9. El usuario pulsa sobre el botón "Remove" asociado al usuario que desea eliminarle el acceso.	10. El sistema eliminará el acceso al programa a dicho usuario mediante la API REST.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 10a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 30. Caso de uso: Eliminar el acceso de un usuario a un programa

3.3.2.2.16. CU-W-30 Visualizar el listado de análisis

Caso de uso	Visualizar el listado de análisis
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGAn1**
Propósito	
Visualizar un listado de los análisis visibles del sistema.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de análisis y visualiza un listado de los análisis visibles del sistema.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Analyses*.	2. El sistema consulta a la API REST los análisis a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos análisis.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 31. Caso de uso: Visualizar el listado de análisis

3.3.2.2.17. CU-W-31 Visualizar los detalles de un análisis

Caso de uso	Visualizar los detalles de un análisis
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGAn2**
Propósito	
Visualizar los detalles de un análisis visible del sistema.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de análisis, visualiza un listado de los análisis visibles del sistema y pulsa sobre uno de ellos. Se redireccionará a una página con los detalles del análisis.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Analyses*.	2. El sistema consulta a la API REST los análisis a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos análisis.
4. El usuario pulsa sobre un análisis del listado.	5. El sistema consulta a la API REST los detalles de dicho análisis. 6. Redirecciona a una página con los detalles del análisis.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 32. Caso de uso: Visualizar los detalles de un análisis

3.3.2.2.18. CU-W-32 Registrar un nuevo análisis

Caso de uso	Registrar un nuevo análisis
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGAn3**

Propósito	
Registrar un nuevo análisis en el sistema.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de análisis y pulsa sobre el botón de nuevo análisis. Se redireccionará a un formulario con los datos a introducir del análisis, el usuario rellena dicho formulario y pulsa sobre insertar el análisis. Se redireccionará a una página de confirmación del registro del análisis.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Analyses* y sobre el botón *New.* 2. Rellena los datos del formulario y pulsa sobre "Insert analysis".	3. El sistema comprueba que los datos introducidos son válidos y que no existe ningún análisis con el mismo rdn. 4. Registra el nuevo análisis mediante la API REST. 5. Redirecciona a una página de confirmación de registro del análisis.
Flujo de acciones de caminos alternativos	
3a. Si los datos introducidos no son válidos, se muestra un error y se vuelve al punto 2. 3b. Si existe un análisis con el mismo rdn, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 4a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 33. Caso de uso: Registrar un nuevo análisis

3.3.2.2.19. CU-W-33 Modificar un análisis

Caso de uso	Modificar un análisis
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGAn4**
Propósito	
Modificar los datos de un análisis propio.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de análisis, visualiza un listado de los análisis visibles del sistema y pulsa sobre un análisis propio. Se redireccionará a una página con los detalles del análisis. El usuario pulsa sobre el icono de editar y modifica los datos de su análisis.	

Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Analyses*.	2. El sistema consulta a la API REST los análisis a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos análisis.
4. El usuario pulsa sobre uno de sus propios análisis.	5. El sistema consulta a la API REST los detalles de dicho análisis. 6. Redirecciona a una página con los detalles del análisis.
7. El usuario pulsa sobre el icono de editar. 8. Modifica los datos del análisis y pulsa sobre "Save analysis".	9. El sistema comprueba que los datos introducidos son válidos. 10. Actualiza la información del análisis mediante la API REST. 11. Redirecciona a la página de detalles del análisis.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 9a. Si los datos introducidos no son válidos, se muestra un error y se vuelve al punto 8. 10a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 34. Caso de uso: Modificar un análisis

3.3.2.2.20. CU-W-34 Eliminar un análisis

Caso de uso	Eliminar un análisis
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGAn5**
Propósito	
Eliminar un análisis propio.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de análisis, visualiza un listado de los análisis visibles del sistema y pulsa sobre un análisis propio. Se redireccionará a una página con los detalles del análisis. El usuario pulsa sobre el icono de eliminar y confirma la eliminación del análisis.	

Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Analyses*.	2. El sistema consulta a la API REST los análisis a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos análisis.
4. El usuario pulsa sobre uno de sus propios análisis.	5. El sistema consulta a la API REST los detalles de dicho análisis. 6. Redirecciona a una página con los detalles del análisis.
7. El usuario pulsa sobre el icono de eliminar y confirma la eliminación de su análisis.	8. El sistema elimina el análisis mediante la API REST. 9. Redirecciona a la página del listado de los análisis del usuario.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 35. Caso de uso: Eliminar un análisis

3.3.2.2.21. CU-W-35 Ver un listado de los usuarios que tienen acceso a un análisis

Caso de uso	Ver un listado de los usuarios que tienen acceso a un análisis
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGAn6**
Propósito	
Visualizar el listado de los usuarios que tiene acceso a un análisis propio de visibilidad compartida.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de análisis, visualiza un listado de los análisis visibles del sistema y pulsa sobre un análisis propio de visibilidad compartida. Se redireccionará a una página con los detalles del análisis, pulsa sobre el listado desplegable “Shared Users” y visualiza los usuarios que tienen acceso al análisis.	
Descripción detallada	

Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Analyses*.	2. El sistema consulta a la API REST los análisis a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos análisis.
4. El usuario pulsa sobre uno de sus propios análisis de visibilidad compartida.	5. El sistema consulta a la API REST los detalles de dicho análisis. 6. Redirecciona a una página con los detalles del análisis.
7. El usuario pulsa sobre el listado desplegable "Shared Users".	8. El sistema muestra el listado de usuarios.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 36. Caso de uso: Ver un listado de los usuarios que tienen acceso a un análisis

3.3.2.2.22. CU-W-36 Dar acceso a un usuario a un análisis

Caso de uso	Dar acceso a un usuario a un análisis
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGAn7**
Propósito	
Dar acceso a un usuario registrado a un análisis propio de visibilidad compartida.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de análisis, visualiza un listado de los análisis visibles del sistema y pulsa sobre un análisis propio de visibilidad compartida. Se redireccionará a una página con los detalles del análisis, pulsa sobre el listado desplegable "Shared Users", pulsa sobre el botón "Add user", introduce el email de un usuario registrado y pulsa sobre el botón "Add".	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Analyses*.	2. El sistema consulta a la API REST los análisis a los que tiene acceso el usuario.

	3. Redirecciona a una página con el listado de dichos análisis.
4. El usuario pulsa sobre uno de sus propios análisis de visibilidad compartida.	5. El sistema consulta a la API REST los detalles de dicho análisis. 6. Redirecciona a una página con los detalles del análisis.
7. El usuario pulsa sobre el listado desplegable "Shared Users" y sobre el botón "Add user". 8. Introduce el email de un usuario registrado y pulsa sobre el botón "Add".	9. El sistema añade al usuario en el listado de usuarios compartidos mediante la API REST.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 9a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 37. Caso de uso: Dar acceso a un usuario a un análisis

3.3.2.2.23. CU-W-37 Eliminar el acceso de un usuario a un análisis

Caso de uso	Eliminar el acceso de un usuario a un análisis
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGAn9**
Propósito	
Eliminar el acceso de un usuario a un análisis propio de visibilidad compartida.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de análisis, visualiza un listado de los análisis visibles del sistema y pulsa sobre un análisis propio de visibilidad compartida. Se redireccionará a una página con los detalles del análisis, pulsa sobre el listado desplegable "Shared Users" y visualiza los usuarios que tienen acceso al programa. El usuario pulsa sobre el botón "Remove" asociado al usuario que desea eliminarle el acceso al análisis.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA

1. El usuario pulsa sobre el botón *Analyses*.	2. El sistema consulta a la API REST los análisis a los que tiene acceso el usuario. 3. Redirecciona a una página con el listado de dichos análisis.
4. El usuario pulsa sobre uno de sus propios análisis de visibilidad compartida.	5. El sistema consulta a la API REST los detalles de dicho análisis. 6. Redirecciona a una página con los detalles del análisis.
7. El usuario pulsa sobre el listado desplegable "Shared Users".	8. El sistema muestra el listado de usuarios.
9. El usuario pulsa sobre el botón "Remove" asociado al usuario que desea eliminarle el acceso.	10. El sistema eliminará el acceso al análisis a dicho usuario mediante la API REST.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 10a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 38. Caso de uso: Eliminar el acceso de un usuario a un análisis

3.3.2.2.24. CU-W-38 Visualizar el listado de resultados

Caso de uso	Visualizar el listado de resultados
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGR1**
Propósito	
Visualizar un listado de los resultados propios.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de resultados y visualiza un listado de sus propios resultados registrados en el sistema.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Results*.	2. El sistema consulta a la API REST los resultados pertenecientes al usuario.

	3. Redirecciona a una página con el listado de dichos resultados.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 39. Caso de uso: Visualizar el listado de resultados

3.3.2.2.25. CU-W-39 Visualizar los detalles de un resultado

Caso de uso	Visualizar los detalles de un resultado
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGR2**
Propósito	
Visualizar los detalles de un resultado propio.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de resultados, visualiza un listado de sus propios resultados registrados en el sistema y pulsa sobre uno de ellos. Se redireccionará a una página con los detalles del resultado.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Results*.	2. El sistema consulta a la API REST los resultados pertenecientes al usuario. 3. Redirecciona a una página con el listado de dichos resultados.
4. El usuario pulsa sobre uno de sus propios resultados.	5. El sistema consulta a la API REST los detalles de dicho resultado. 6. Redirecciona a una página con los detalles del resultado.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 40. Caso de uso: Visualizar los detalles de un resultado

3.3.2.2.26. CU-W-40 Visualizar los detalles del resultado de un programa

Caso de uso	Visualizar los detalles del resultado de un programa
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGR3**
Propósito	
Visualizar los detalles del resultado de un programa asociado a un resultado propio.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de resultados, visualiza un listado de sus propios resultados registrados en el sistema y pulsa sobre uno de ellos. Se redireccionará a una página con los detalles del resultado. El usuario pulsa sobre el icono de resultados asociado a uno de los programas analizados. Se redireccionará a una página con los detalles del resultado del programa.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Results*.	2. El sistema consulta a la API REST los resultados pertenecientes al usuario. 3. Redirecciona a una página con el listado de dichos resultados.
4. El usuario pulsa sobre uno de sus propios resultados.	5. El sistema consulta a la API REST los detalles de dicho resultado. 6. Redirecciona a una página con los detalles del resultado.
7. El usuario pulsa sobre el icono de resultados de uno de los programas analizados.	8. El sistema consulta a la API REST los detalles del resultado del programa. 9. Redirecciona a una página con los detalles de dicho resultado.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 41. Caso de uso: Visualizar los detalles del resultado de un programa

3.3.2.2.27. CU-W-41 Realizar el análisis de un programa

Caso de uso	Realizar el análisis de un programa
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGR4**
Propósito	
Analizar uno o varios programas visibles y/o propios con uno o varios análisis visibles y/o propios.	
Resumen de alto nivel	
Un usuario autenticado accede a la página de resultados y pulsa sobre el botón de nuevo resultado. Se redireccionará a un formulario donde seleccionar uno o varios programas visibles y uno o varios análisis visibles que realizarles a los programas.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Results* y sobre el botón *New.* 2. Rellena los datos del formulario y pulsa sobre “Analyze”.	3. El sistema comprueba que los datos introducidos son válidos. 4. Registra el nuevo resultado mediante la API REST. 5. Redirecciona a una página de confirmación de registro del resultado.
Flujo de acciones de caminos alternativos	
3a. Si los datos introducidos no son válidos, se muestra un error y se vuelve al punto 2. 4a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 42. Caso de uso: Realizar el análisis de un programa

3.3.2.2.28. CU-W-42 Eliminar un resultado

Caso de uso	Eliminar un resultado
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGR5**
Propósito	
Eliminar un resultado propio.	

Resumen de alto nivel	
Un usuario autenticado accede a la página de resultados, visualiza un listado de sus propios resultados registrados en el sistema y pulsa sobre uno de ellos. Se redireccionará a una página con los detalles del resultado. El usuario pulsa sobre el icono de eliminar y confirma la eliminación del resultado.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Results*.	2. El sistema consulta a la API REST los resultados pertenecientes al usuario. 3. Redirecciona a una página con el listado de dichos resultados.
4. El usuario pulsa sobre uno de sus propios resultados.	5. El sistema consulta a la API REST los detalles de dicho resultado. 6. Redirecciona a una página con los detalles del resultado.
7. El usuario pulsa sobre el icono de eliminar y confirma la eliminación de su resultado.	8. El sistema elimina el resultado mediante la API REST. 9. Redirecciona a la página del listado de los resultados del usuario.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 43. Caso de uso: Eliminar un resultado

3.3.2.2.29. CU-W-43 Eliminar el resultado de un programa

Caso de uso	Eliminar el resultado de un programa
Actores	Usuario autenticado
Iniciado por	Usuario autenticado
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema
Postcondiciones	-
Requisitos que satisface	**RFGR6**
Propósito	
Eliminar el resultado de un programa asociado a un resultado propio.	
Resumen de alto nivel	

<table>
<tr><td colspan="2">Un usuario autenticado accede a la página de resultados, visualiza un listado de sus propios resultados registrados en el sistema y pulsa sobre uno de ellos. Se redireccionará a una página con los detalles del resultado. El usuario pulsa sobre el icono de resultados asociado a uno de los programas analizados. Se redireccionará a una página con los detalles del resultado del programa. El usuario pulsa sobre el icono de eliminar y confirma la eliminación de dicho resultado del programa.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Descripción detallada</td></tr>
<tr><td colspan="2">Flujo de acciones del camino básico</td></tr>
<tr><td>ACTOR</td><td>SISTEMA</td></tr>
<tr><td>1. El usuario pulsa sobre el botón Results.</td><td>2. El sistema consulta a la API REST los resultados pertenecientes al usuario.
3. Redirecciona a una página con el listado de dichos resultados.</td></tr>
<tr><td>4. El usuario pulsa sobre uno de sus propios resultados.</td><td>5. El sistema consulta a la API REST los detalles de dicho resultado.
6. Redirecciona a una página con los detalles del resultado.</td></tr>
<tr><td>7. El usuario pulsa sobre el icono de resultados de uno de los programas analizados.</td><td>8. El sistema consulta a la API REST los detalles del resultado del programa.
9. Redirecciona a una página con los detalles de dicho resultado.</td></tr>
<tr><td>10. El usuario pulsa sobre el icono de eliminar y confirma la eliminación del resultado del programa.</td><td>11. El sistema elimina dicho resultado mediante la API REST.
12. Redirecciona a la página del listado de los resultados del usuario.</td></tr>
<tr><td colspan="2">Flujo de acciones de caminos alternativos</td></tr>
<tr><td colspan="2">2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.
5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.
8a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.
11a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.</td></tr>
</table>

Tabla 44. Caso de uso: Eliminar el resultado de un programa

3.3.2.3. Usuario administrador

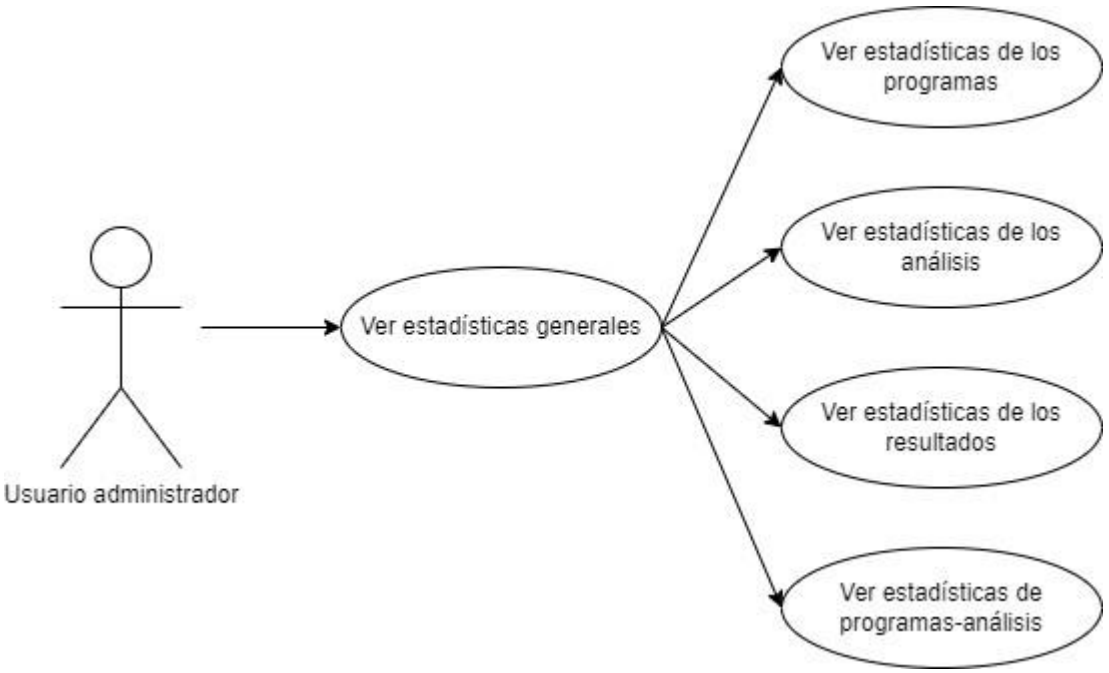


Ilustración 6. Diagrama de casos de uso - Usuario administrador

3.3.2.3.1. CU-W-44 Ver estadísticas generales

Caso de uso	Ver estadísticas generales
Actores	Usuario administrador
Iniciado por	Usuario administrador
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema y tendrá rol de administrador
Postcondiciones	-
Requisitos que satisface	**RFAd1**
Propósito	
Visualizar las estadísticas generales de la aplicación web.	
Resumen de alto nivel	
Un usuario administrador accede a la página de estadísticas y visualiza un conjunto de estadísticas generales del sistema.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Statistics*.	2. El sistema consulta a la API REST las estadísticas generales del sistema. 3. Redirecciona a una página con el conjunto de dichas estadísticas.

Flujo de acciones de caminos alternativos
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.

Tabla 45. Caso de uso: Ver estadísticas generales

3.3.2.3.2. CU-W-45 Ver estadísticas de los programas

Caso de uso	Ver estadísticas de los programas
Actores	Usuario administrador
Iniciado por	Usuario administrador
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema y tendrá rol de administrador
Postcondiciones	-
Requisitos que satisface	**RFAd1**

Propósito
Visualizar las estadísticas de los programas registrados en el sistema.

Resumen de alto nivel
Un usuario administrador accede a la página de estadísticas y visualiza un conjunto de estadísticas generales del sistema. El usuario pulsa sobre el icono de programa. Se redireccionará a una página con estadísticas de los programas registrados.

Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Statistics*.	2. El sistema consulta a la API REST las estadísticas generales del sistema. 3. Redirecciona a una página con el conjunto de dichas estadísticas.
4. El usuario pulsa sobre el icono de programa.	5. El sistema consulta a la API REST las estadísticas de los programas registrados en el sistema. 6. Redirecciona a una página con un listado de dichas estadísticas.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 46. Caso de uso: Ver estadísticas de los programas

3.3.2.3.3. CU-W-46 Ver estadísticas de los análisis

Caso de uso	Ver estadísticas de los análisis

Actores	Usuario administrador
Iniciado por	Usuario administrador
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema y tendrá rol de administrador
Postcondiciones	-
Requisitos que satisface	**RFAd1**
Propósito	
Visualizar las estadísticas los análisis registrados en el sistema.	
Resumen de alto nivel	
Un usuario administrador accede a la página de estadísticas y visualiza un conjunto de estadísticas generales del sistema. El usuario pulsa sobre el icono de análisis. Se redireccionará a una página con estadísticas de los análisis registrados.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Statistics*.	2. El sistema consulta a la API REST las estadísticas generales del sistema. 3. Redirecciona a una página con el conjunto de dichas estadísticas.
4. El usuario pulsa sobre el icono de análisis.	5. El sistema consulta a la API REST las estadísticas de los análisis registrados en el sistema. 6. Redirecciona a una página con un listado de dichas estadísticas.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 47. Caso de uso: Ver estadísticas de los análisis

3.3.2.3.4. CU-W-47 Ver estadísticas de los resultados

Caso de uso	Ver estadísticas de los resultados
Actores	Usuario administrador
Iniciado por	Usuario administrador
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema y tendrá rol de administrador
Postcondiciones	-
Requisitos que satisface	**RFAd1**

Propósito	
Visualizar las estadísticas de los resultados registrados en el sistema.	
Resumen de alto nivel	
Un usuario administrador accede a la página de estadísticas y visualiza un conjunto de estadísticas generales del sistema. El usuario pulsa sobre el icono de resultados. Se redireccionará a una página con estadísticas de los resultados registrados.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Statistics*.	2. El sistema consulta a la API REST las estadísticas generales del sistema. 3. Redirecciona a una página con el conjunto de dichas estadísticas.
4. El usuario pulsa sobre el icono de resultado.	5. El sistema consulta a la API REST las estadísticas de los resultados registrados en el sistema. 6. Redirecciona a una página con un listado de dichas estadísticas.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 48. Caso de uso: Ver estadísticas de los resultados

3.3.2.3.5. CU-W-48 Ver estadísticas de programas-análisis

Caso de uso	Ver estadísticas de programas-análisis
Actores	Usuario administrador
Iniciado por	Usuario administrador
Tipo	Primario
Referencias	-
Precondiciones	El usuario estará autenticado en el sistema y tendrá rol de administrador
Postcondiciones	-
Requisitos que satisface	**RFAd1**
Propósito	
Visualizar las estadísticas de los conjuntos programa-análisis registrados en el sistema.	
Resumen de alto nivel	
Un usuario administrador accede a la página de estadísticas y visualiza un conjunto de estadísticas generales del sistema. El usuario pulsa sobre el icono de programa-análisis. Se redireccionará a una página con estadísticas de los conjuntos programa-análisis	

registrados. Dichos conjuntos corresponden con los análisis que se hayan realizado a ciertos programas.	
Descripción detallada	
Flujo de acciones del camino básico	
ACTOR	SISTEMA
1. El usuario pulsa sobre el botón *Statistics*.	2. El sistema consulta a la API REST las estadísticas generales del sistema. 3. Redirecciona a una página con el conjunto de dichas estadísticas.
4. El usuario pulsa sobre el icono de programa-análisis.	5. El sistema consulta a la API REST las estadísticas de los conjuntos programa-análisis registrados en el sistema. 6. Redirecciona a una página con un listado de dichas estadísticas.
Flujo de acciones de caminos alternativos	
2a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1. 5a. Si se produce algún error durante la consulta, se redirige a una página de error mostrando el mismo y se vuelve al punto 1.	

Tabla 49. Caso de uso: Ver estadísticas de programas-análisis

3.4. Matrices de trazabilidad

3.4.1. Matriz de trazabilidad de los requisitos funcionales de la gestión de autenticación

	CU-W-01	CU-W-02	CU-W-03	CU-W-04	CU-W-15
RFGAu1	✓				
RFGAu2		✓			
RFGAu3					✓
RFGAu4.1			✓		
RFGAu4.2				✓	

Tabla 50. Matriz de trazabilidad de los requisitos funcionales de la gestión de autenticación

3.4.2. Matriz de trazabilidad de los requisitos funcionales de la gestión de usuarios

	CU-W-11	CU-W-12	CU-W-13	CU-W-16	CU-W-17	CU-W-18	CU-W-19
RFGU1							
RFGU2	✓			✓			
RFGU3		✓			✓		
RFGU4			✓			✓	
RFGU5							

RFGU6							✓

Tabla 51. Matriz de trazabilidad de los requisitos funcionales de la gestión de usuarios

3.4.3. Matriz de trazabilidad de los requisitos funcionales de la gestión de programas

	CU-W-05	CU-W-06	CU-W-07	CU-W-20	CU-W-21	CU-W-22	CU-W-24	CU-W-25	CU-W-26	CU-W-27	CU-W-28	CU-W-29
RFGP1	✓			✓								
RFGP2		✓			✓							
RFGP3			✓			✓						
RFGP4							✓					
RFGP5								✓				
RFGP6									✓			
RFGP7										✓		
RFGP8											✓	
RFGP9												
RFGP10												✓

Tabla 52. Matriz de trazabilidad de los requisitos funcionales de la gestión de programas

3.4.4. Matriz de trazabilidad de los requisitos funcionales de la gestión de análisis

	CU-W-09	CU-W-10	CU-W-30	CU-W-31	CU-W-32	CU-W-33	CU-W-34	CU-W-35	CU-W-36	CU-W-37
RFGAn1	✓		✓							
RFGAn2		✓		✓						
RFGAn3					✓					
RFGAn4						✓				
RFGAn5							✓			
RFGAn6								✓		
RFGAn7									✓	
RFGAn8										
RFGAn9										✓

Tabla 53. Matriz de trazabilidad de los requisitos funcionales de la gestión de análisis

3.4.5. Matriz de trazabilidad de los requisitos funcionales de la gestión resultados

	CU-W-08	CU-W-23	CU-W-38	CU-W-39	CU-W-40	CU-W-41	CU-W-42	CU-W-43
RFGR1			✓					
RFGR2				✓				
RFGR3	✓	✓			✓			
RFGR4						✓		
RFGR5							✓	

RFGR6								✓

Tabla 54. Matriz de trazabilidad de los requisitos funcionales de la gestión resultados

3.4.6. Matriz de trazabilidad de los requisitos funcionales adicionales

	CU-W-14	CU-W-44	CU-W-45	CU-W-46	CU-W-47	CU-W-48
RFGAd1		✓	✓	✓	✓	✓
RFGAd2	✓					

Tabla 55. Matriz de trazabilidad de los requisitos funcionales adicionales

Capítulo 4. Diseño del sistema

4.1. Arquitectura del sistema

Presenta la estructura general del sistema y cómo sus componentes interactúan.

El sistema desarrollado se compone de una aplicación web diseñada para ofrecer herramientas de análisis y gestión de código a usuarios registrados. La aplicación cuenta con funcionalidades como autenticación y gestión de usuarios, gestión de programas y análisis, visualización de resultados y de estadísticas.

La arquitectura del sistema se ha diseñado con los siguientes objetivos principales:

- Escalabilidad: Permitir el crecimiento del sistema con un aumento en el número de usuarios y la complejidad de las funcionalidades.
- Seguridad: Garantizar la protección de datos sensibles y la integridad de la aplicación frente a amenazas externas.
- Mantenibilidad: Facilitar la introducción de nuevas funcionalidades y la corrección rápida de errores mediante una estructura modular y bien organizada.
- Rendimiento: Asegurar tiempos de respuesta rápidos y eficientes, especialmente en las operaciones críticas como la ejecución de análisis de código.

El sistema se organiza en las siguientes capas y componentes:

- Capa de Presentación:
    - Controladores (Controllers)
    - Vistas (Views)
- Capa de Aplicación:
    - Servicios (Services)
    - Modelos (Models)
- Capa de Persistencia:
    - Repositorios (Repositories)
- Capa de Seguridad:
    - Implementación de seguridad utilizando JWT (JSON Web Tokens).

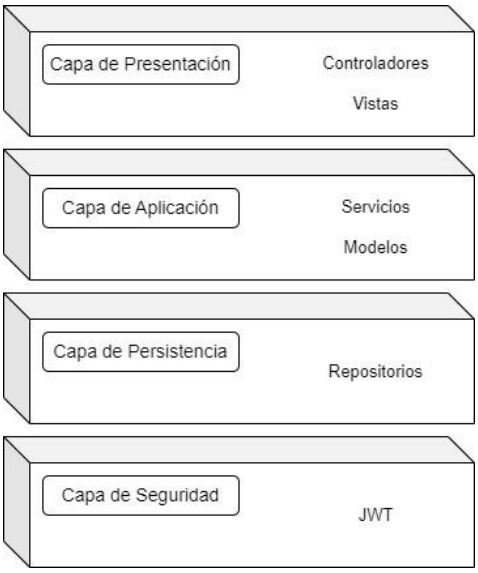


Ilustración 7. Diagrama de capas

El sistema se despliega en un entorno que soporta aplicaciones web, utilizando un servidor de aplicaciones Java para ejecutar el backend desarrollado en Spring Boot. Este backend gestiona la lógica de negocio, la seguridad y la interacción con la base de datos, asegurando una conexión segura mediante SSL/TLS para las comunicaciones con los clientes. Por otro lado, el frontend desarrollado en .NET se ejecuta en un servidor web compatible interactuando con el backend a través de APIs RESTful para obtener y procesar datos, proporcionando así una interfaz de usuario dinámica y segura.

Entre los patrones arquitectónicos utilizados se encuentran:

- MVC (Modelo-Vista-Controlador): Utilizado para separar la lógica de negocio (servicios y modelos), la presentación (vistas) y la gestión de solicitudes (controladores).
- Seguridad con JWT: Para autenticación y autorización basadas en tokens JWT, asegurando la seguridad y la gestión de sesiones de usuario de manera eficiente.

Entre los componentes claves del sistema se encuentran:

- Controladores (Controllers): Gestión de endpoints HTTP y lógica de enrutamiento.
- Servicios (Services): Implementación de la lógica de negocio y coordinación entre controladores y repositorios.
- Modelos (Models): Representación de entidades y datos del dominio.
- Repositorios (Repositories): Acceso a datos y operaciones de persistencia.
- Seguridad (Security): Implementación de medidas de seguridad como autenticación y autorización.
- Vistas (Views): Interfaces de usuario en formato HTML y posiblemente utilizando algún framework de frontend.

Por otro lado, entre las consideraciones de seguridad y rendimiento se encuentran:

- Implementación de cifrado SSL/TLS para todas las comunicaciones entre el cliente y el servidor, garantizando la confidencialidad y la integridad de los datos.
- Utilización de tokens JWT para la gestión de sesiones de usuario, garantizando autenticación segura y autorización basada en roles.
- Diseño orientado a servicios para facilitar la escalabilidad horizontal mediante la replicación de componentes y el balanceo de carga.
- Optimización de consultas y operaciones de base de datos para mantener un rendimiento óptimo incluso bajo cargas elevadas.

4.2. Diagrama y modelo de datos

Esta sección es crucial para entender la estructura interna de la aplicación y cómo se manejan y relacionan los datos. A continuación, se detallan los componentes clave del modelo de datos utilizado en el proyecto, junto con el diagrama que ilustra las relaciones entre las distintas entidades.

En este proyecto, se han utilizado PostgreSQL y Neo4j como sistemas de gestión de bases de datos para almacenar y gestionar la información. PostgreSQL se ha elegido por su robustez y capacidad para manejar datos relacionales, mientras que Neo4j se ha seleccionado por su potencia en la gestión de grafos, lo cual es esencial para las funcionalidades avanzadas de análisis de código basadas en grafos.

Neo4j se utiliza en este proyecto exclusivamente para almacenar el código de los programas en un formato gráfico. Esta base de datos orientada a grafos permite representar de manera eficiente las relaciones y dependencias entre las diferentes partes del código, facilitando el análisis y la visualización de la estructura de los programas. Sin embargo, dado que su uso está limitado a esta función específica, no se incluye su diagrama en la sección.

Por otro lado, PostgreSQL se utiliza para gestionar toda la información relacional del sistema, incluyendo la gestión de usuarios, programas, análisis y resultados. La estructura de esta base de datos relacional es más compleja y cubre la mayoría de las operaciones fundamentales del sistema. Por este motivo, se considera más relevante y útil presentar el diagrama de la base de datos de PostgreSQL, proporcionando una visión detallada de cómo se organiza y maneja la información principal del proyecto.

El modelo de datos de la aplicación se compone de varias entidades principales que interactúan entre sí. Estas entidades incluyen usuarios, programas, análisis y resultados. Cada una de estas entidades tiene una serie de atributos que definen sus propiedades y una serie de relaciones que determinan cómo interactúan entre sí.

A continuación, se muestra el diagrama de la base de datos de PostgreSQL.

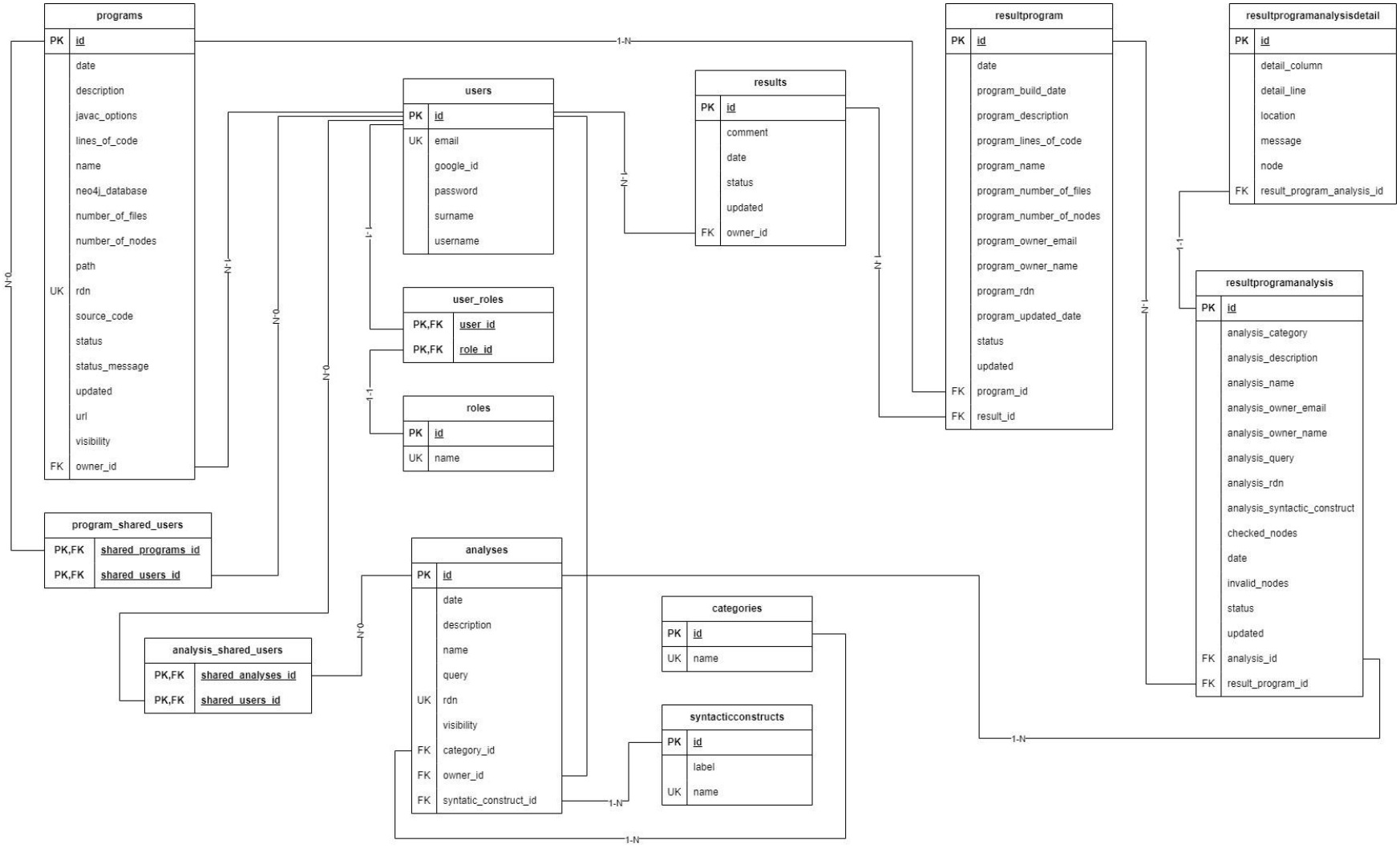


Ilustración 8. Diagrama ER de la base de datos

4.3. Interfaz de usuario

En esta sección, se detalla el diseño y la estructura de las interfaces de usuario desarrolladas para el proyecto, que incluyen el mapa de navegación y el diseño de las distintas pantallas.

4.3.1. Mapa de navegación

El mapa de navegación proporciona una visión general de la estructura y la jerarquía de las pantallas dentro de la aplicación, mostrando cómo navegar a través de las distintas secciones y funcionalidades del sistema de manera intuitiva y eficiente

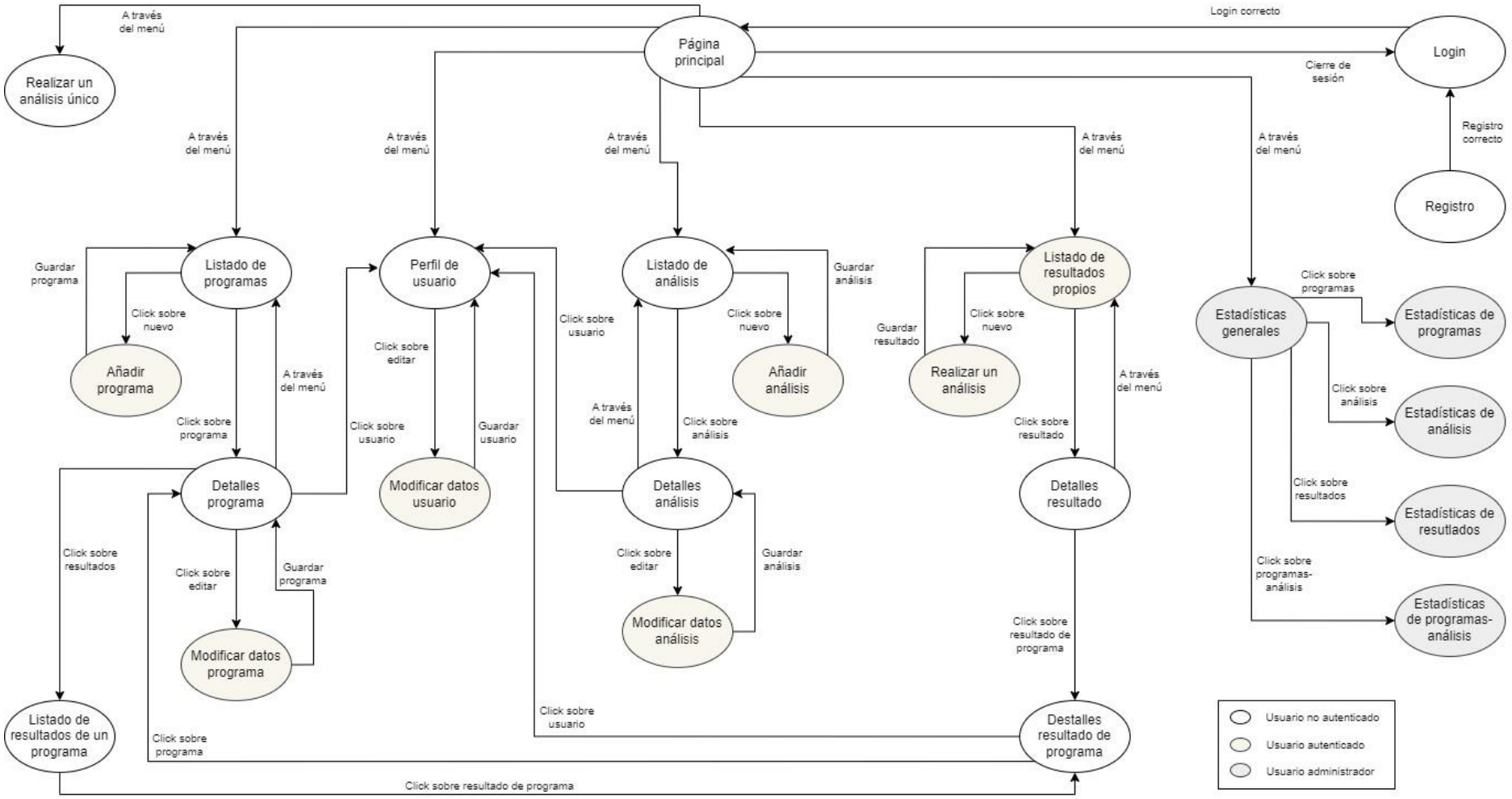


Ilustración 9. Mapa de navegación

4.3.2. Diseño de pantallas

A continuación, se presentan los diseños detallados de las diferentes pantallas de la aplicación. Cada diseño ha sido desarrollado teniendo en cuenta los principios de usabilidad, accesibilidad y experiencia de usuario, con el objetivo de proporcionar una interfaz intuitiva y atractiva para los usuarios finales.

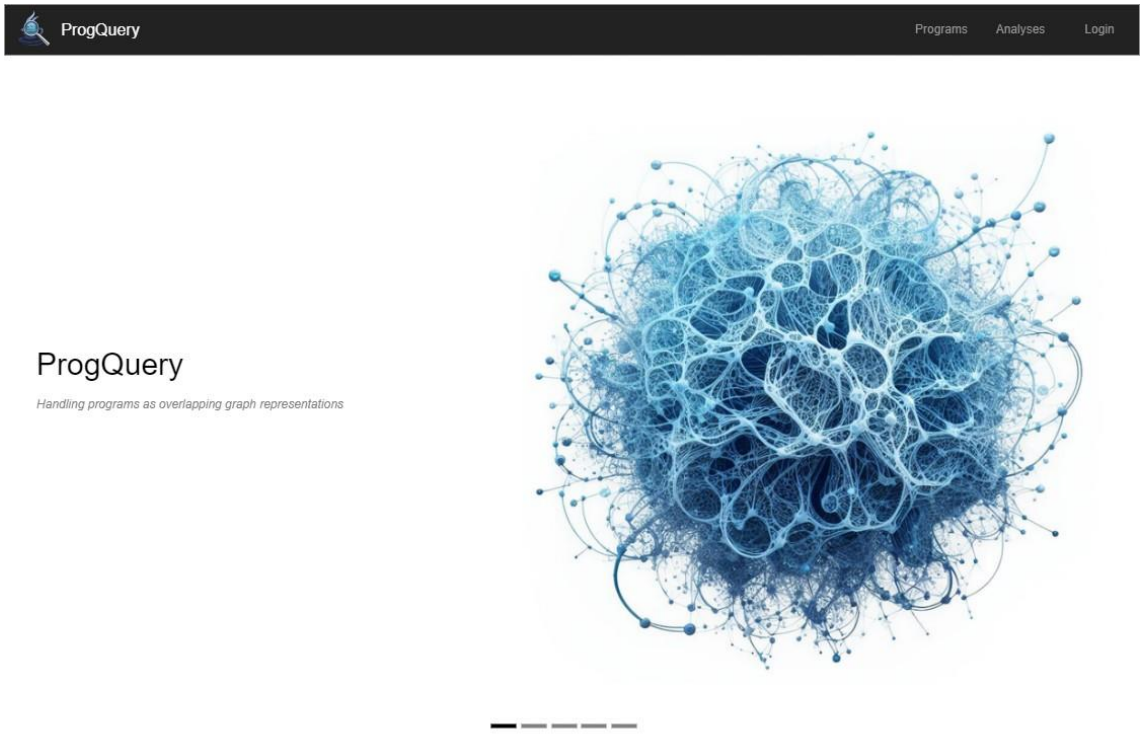


Ilustración 10. Diseño pantalla principal

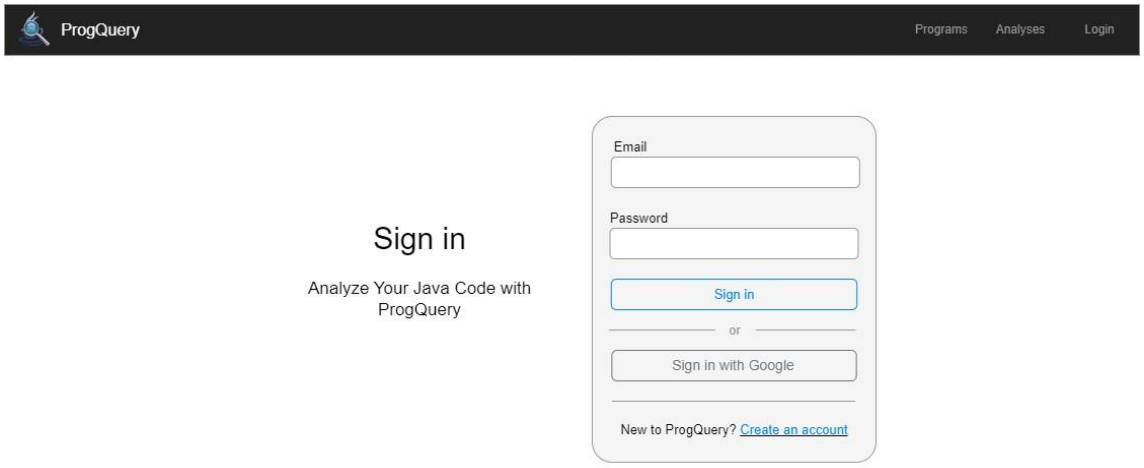


Ilustración 11. Diseño pantalla de inicio de sesión

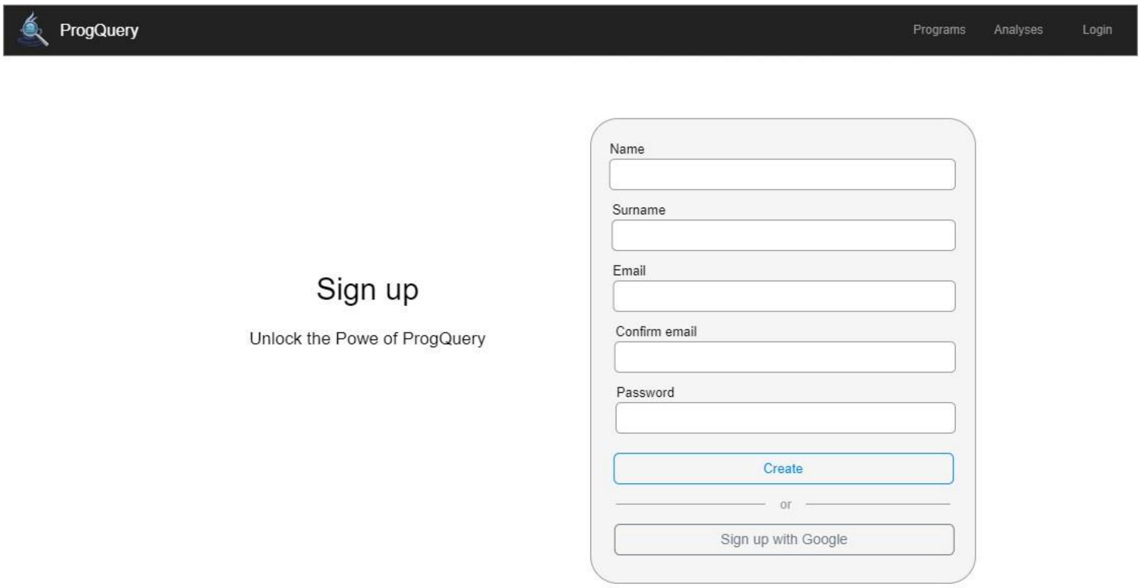


Ilustración 12. Diseño pantalla de registro

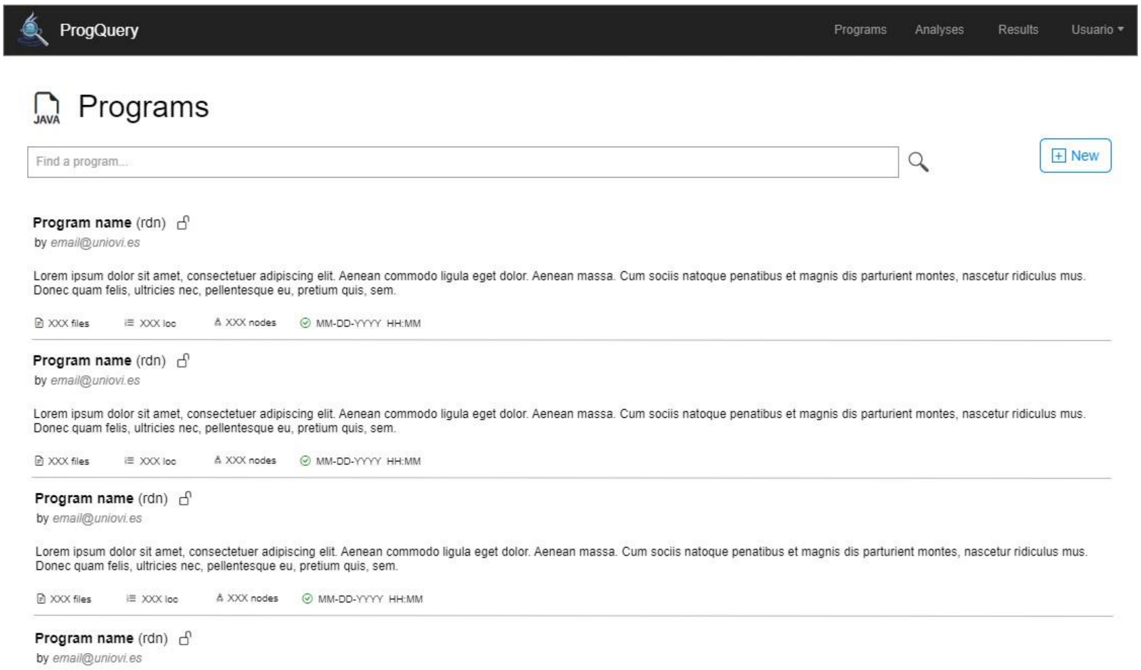


Ilustración 13. Diseño pantalla de listado de programas

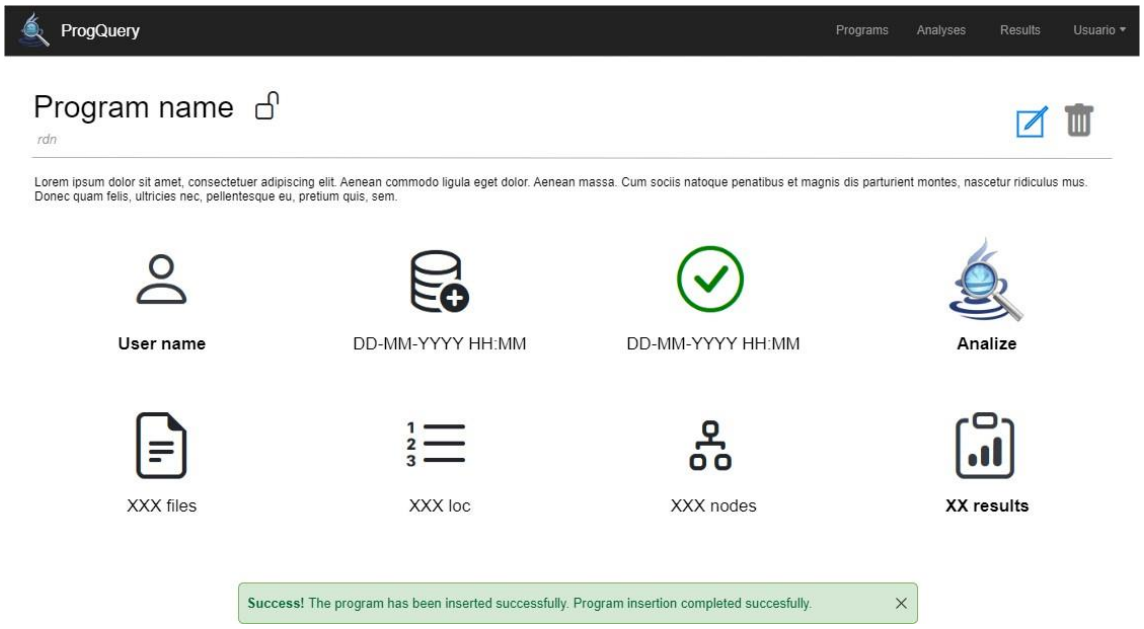


Ilustración 14. Diseño pantalla de detalles de un programa

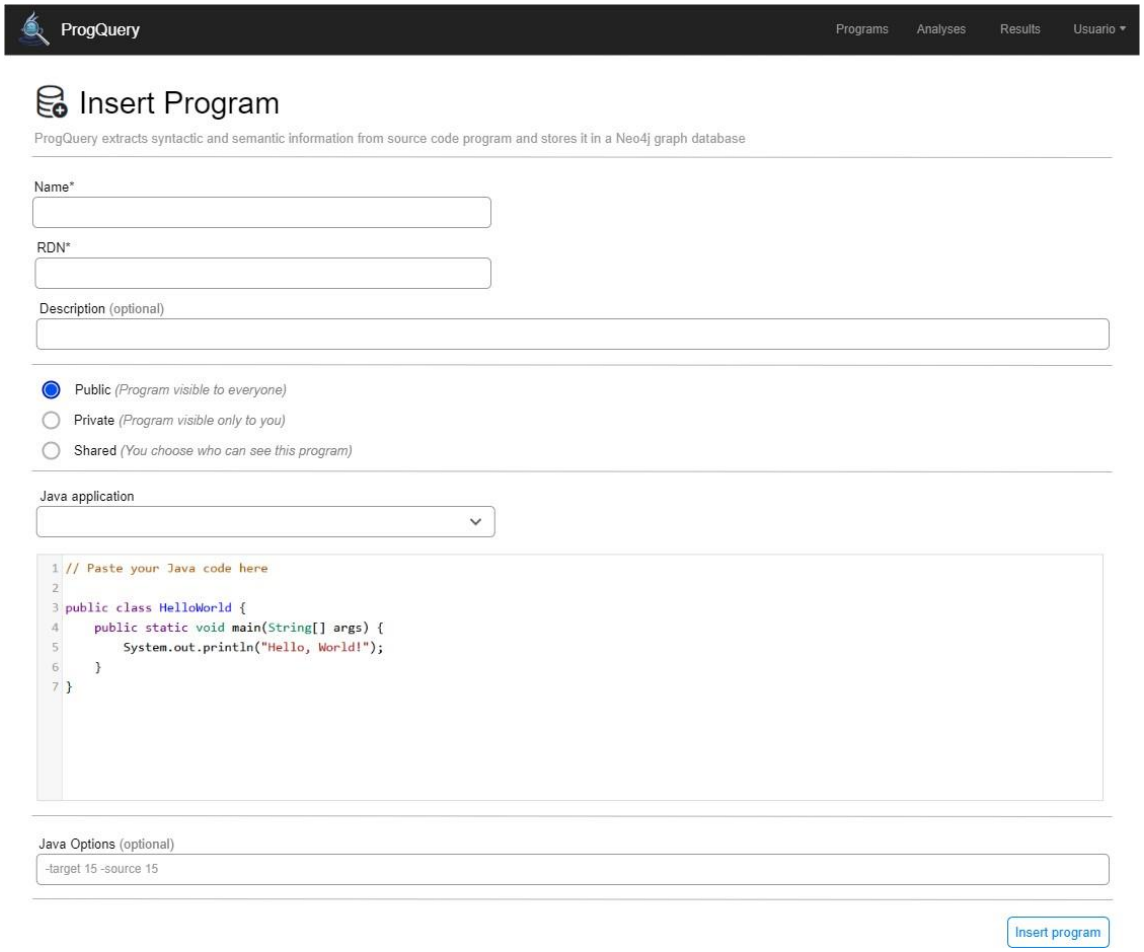


Ilustración 15. Diseño pantalla de añadir un programa

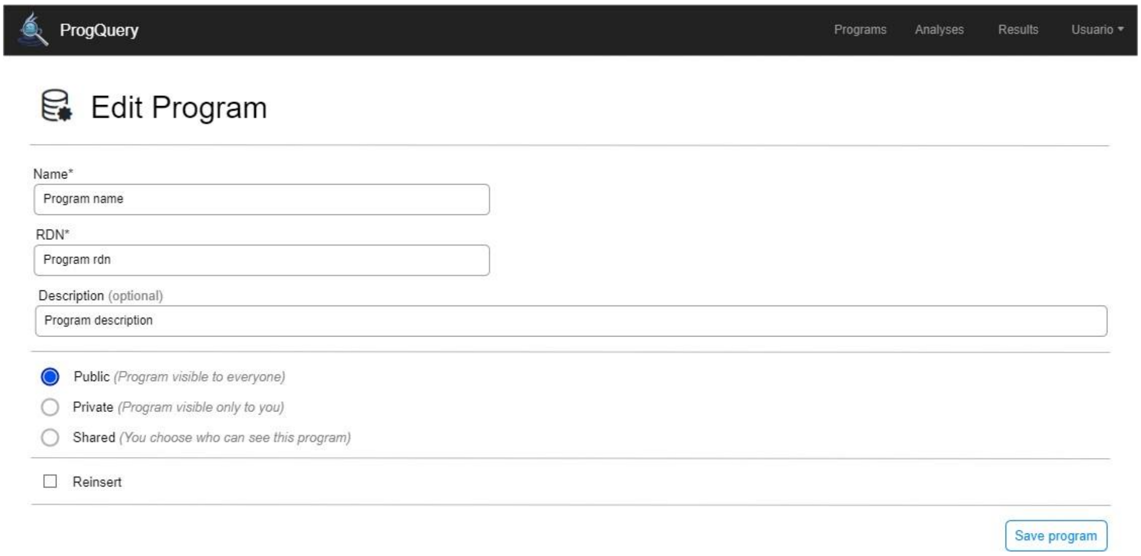


Ilustración 16. Diseño pantalla de modificar un programa

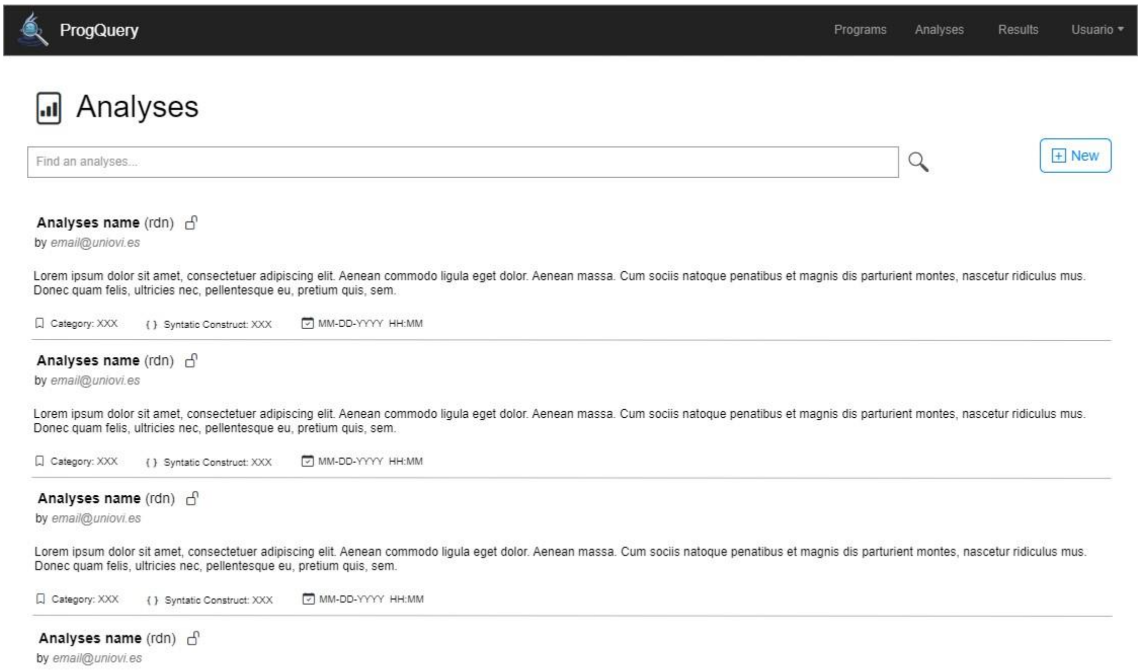


Ilustración 17. Diseño pantalla de listado de análisis

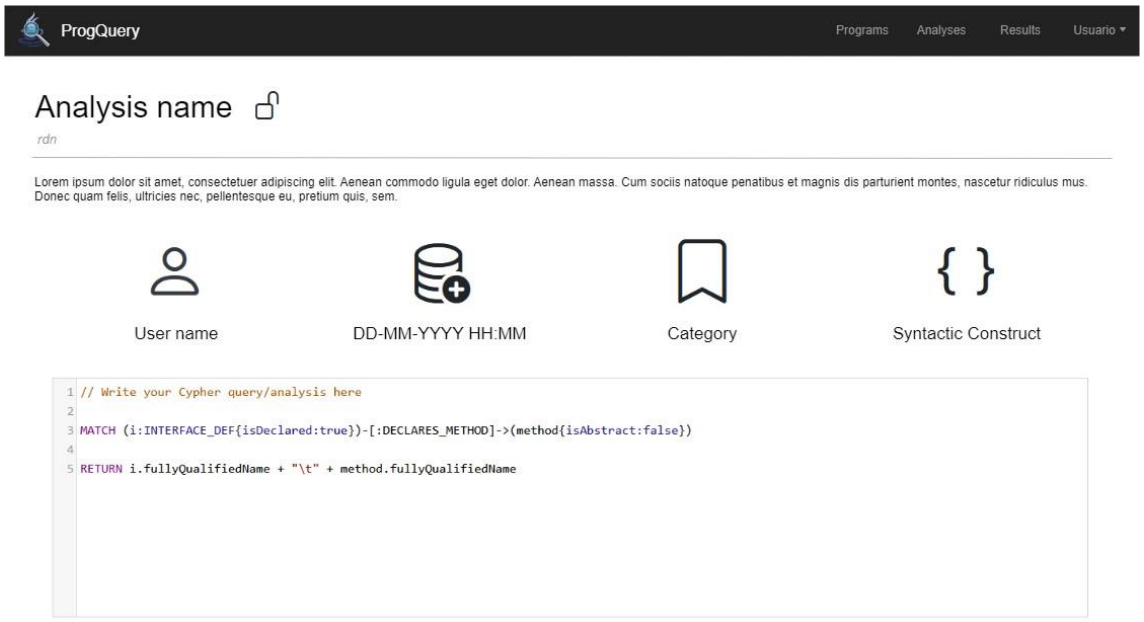


Ilustración 18. Diseño pantalla de detalles de un análisis

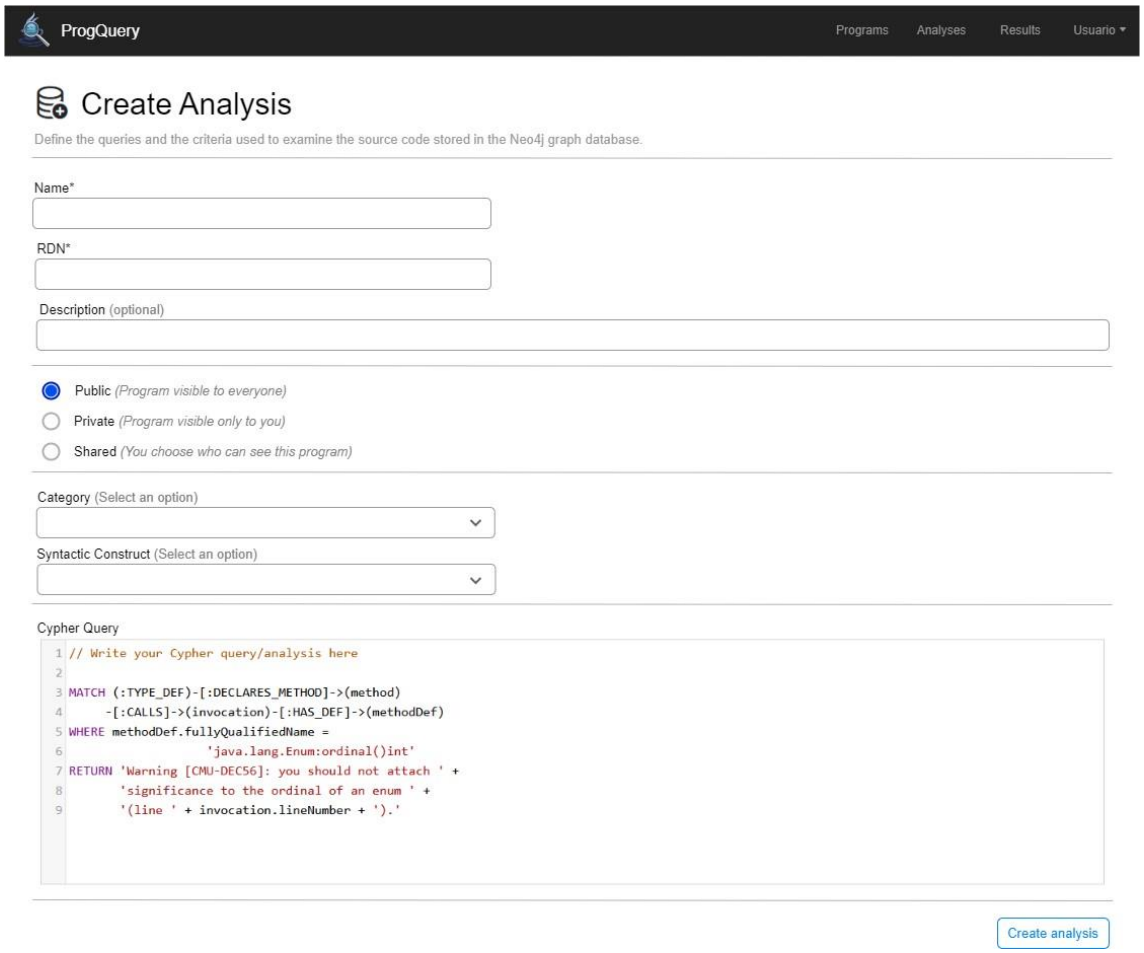


Ilustración 19. Diseño pantalla de añadir un análisis

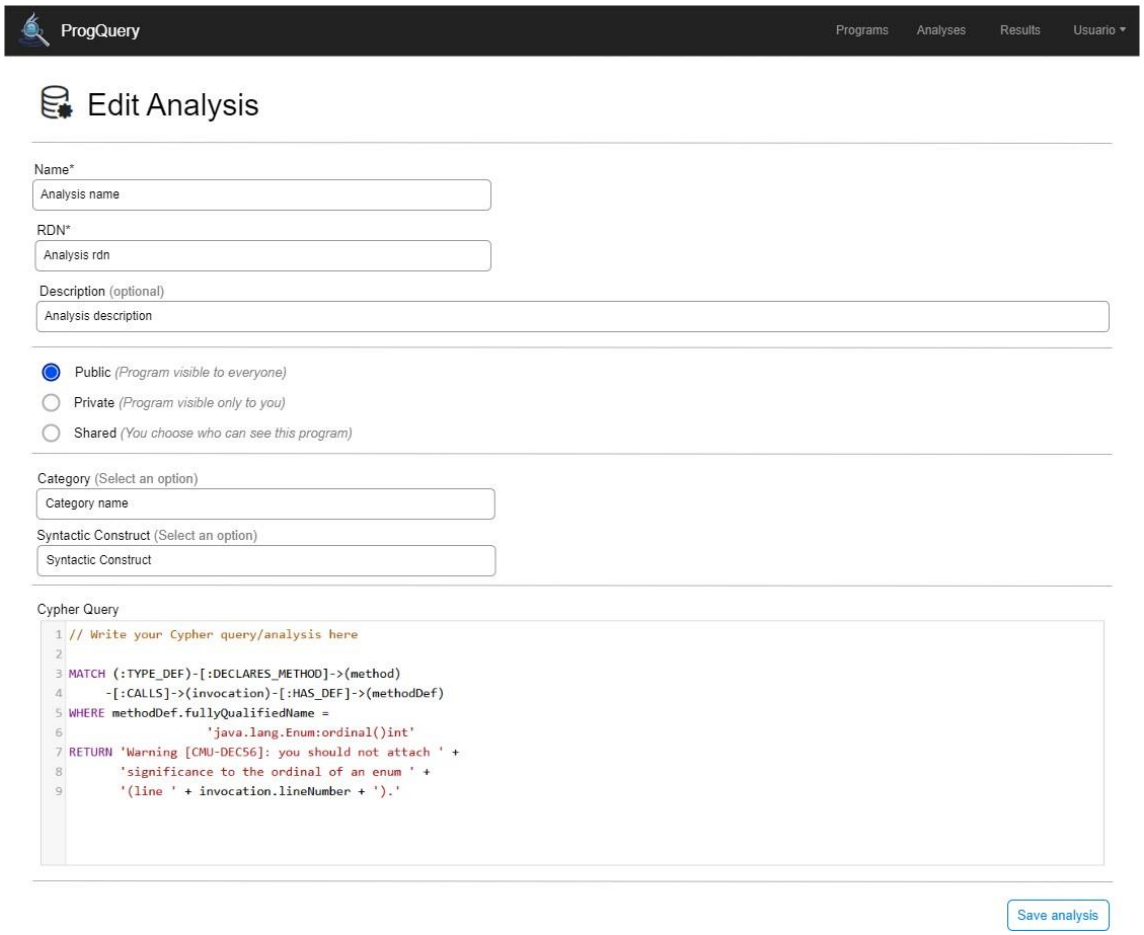


Ilustración 20. Diseño pantalla de modificar un análisis

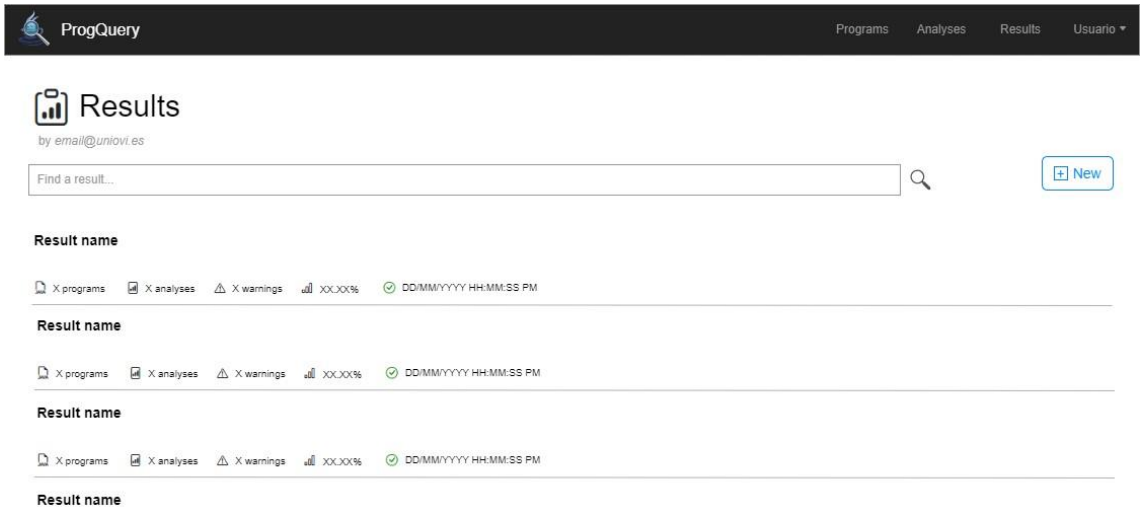


Ilustración 21. Diseño pantalla de listado de resultados

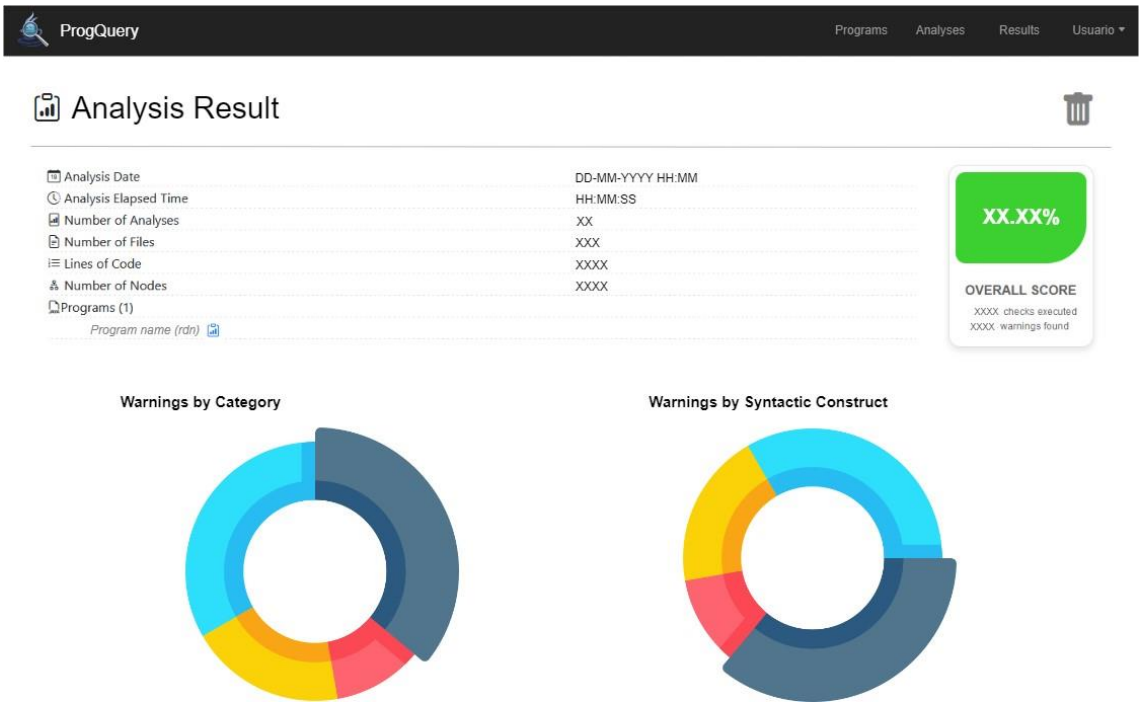


Ilustración 22. Diseño pantalla de detalles de un resultado

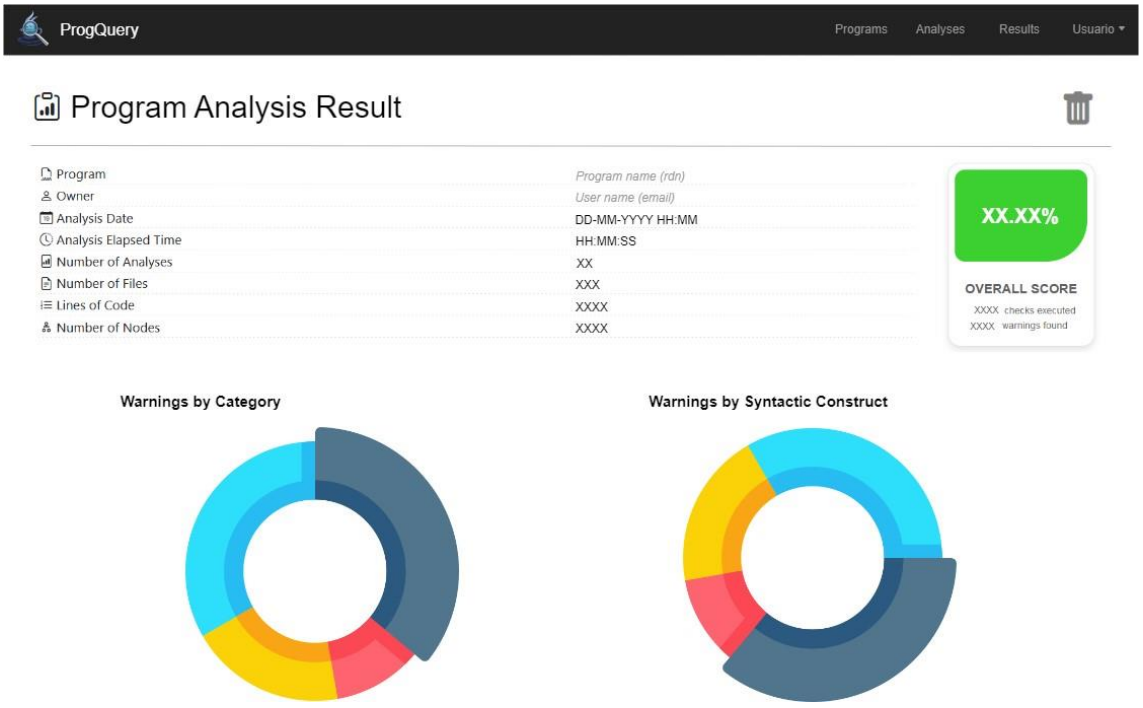


Ilustración 23. Diseño pantalla de detalles del resultado de un programa

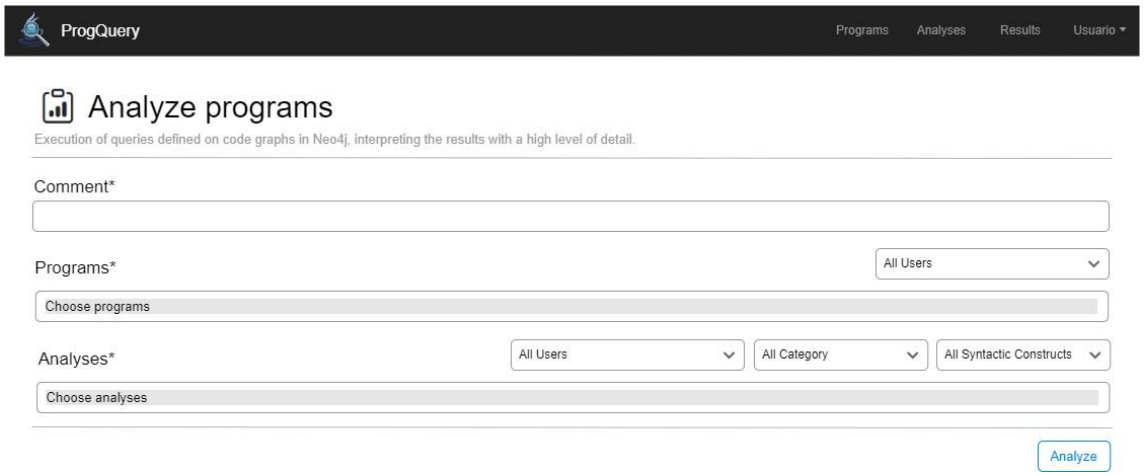


Ilustración 24. Diseño pantalla de añadir un resultado

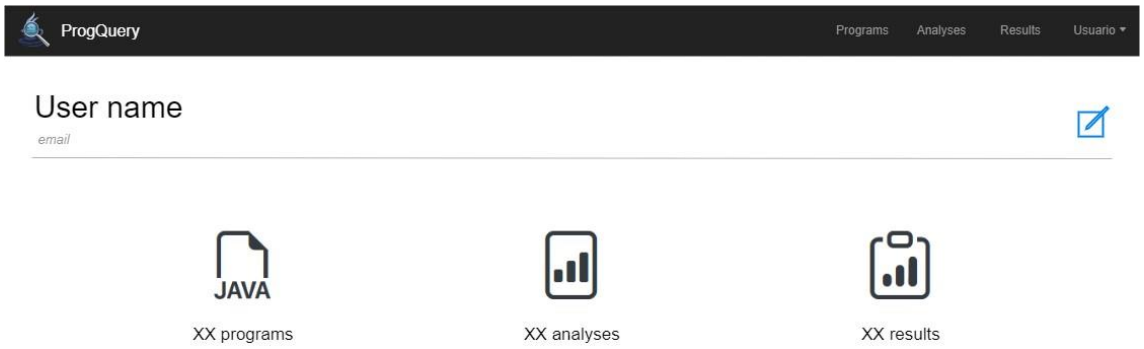


Ilustración 25. Diseño pantalla del perfil de usuario

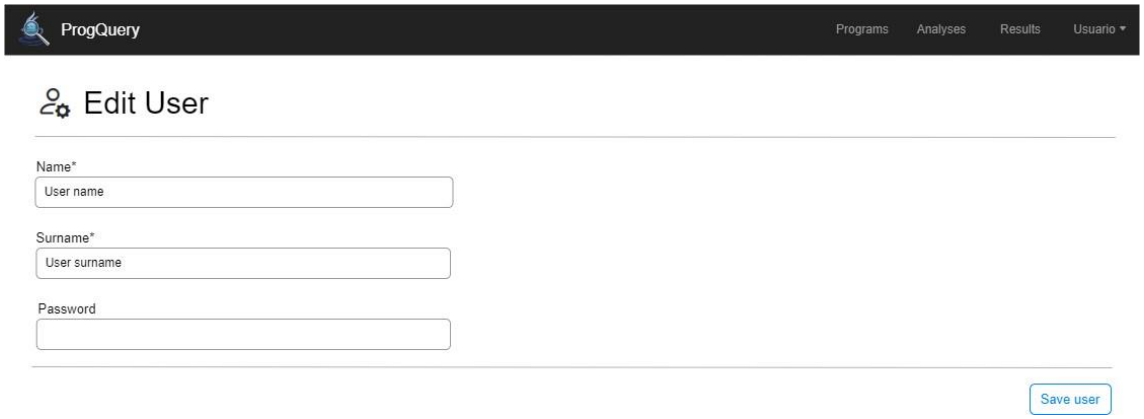


Ilustración 26. Diseño pantalla de modificar el perfil de usuario

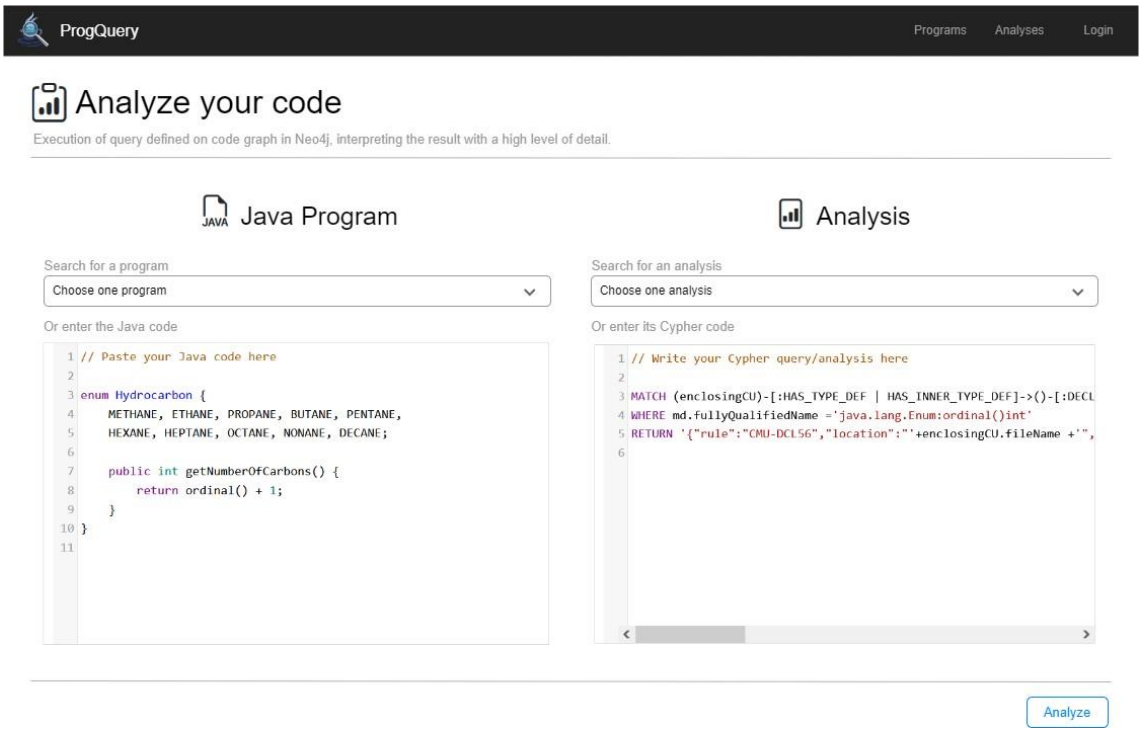


Ilustración 27. Diseño pantalla de análisis básico

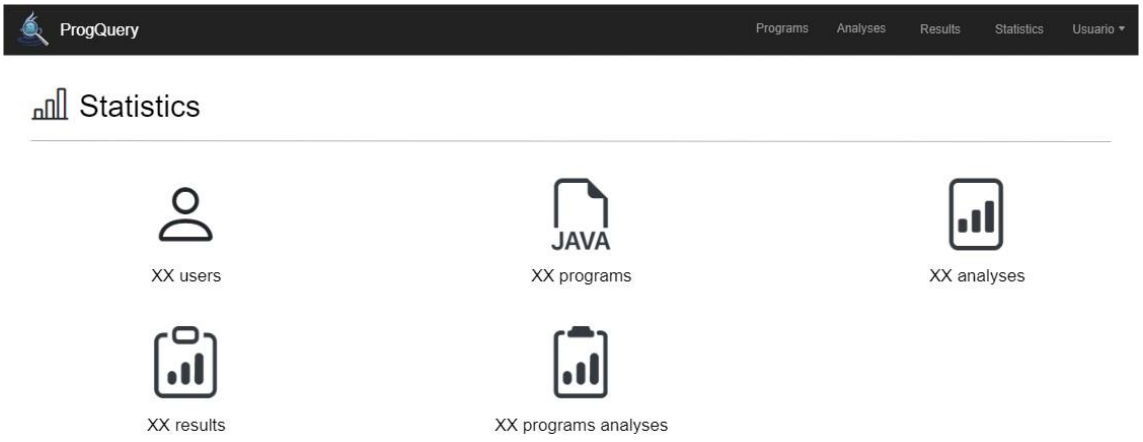


Ilustración 28. Diseño pantalla de estadísticas generales

Capítulo 5. Implementación del sistema

5.1. Herramientas y entorno de desarrollo

En esta sección se hará un breve repaso por las herramientas usadas para las diferentes fases del proyecto.

5.1.1. Lenguajes de Programación

5.1.1.1. Java

Java es un lenguaje de programación y plataforma informática desarrollado inicialmente por Sun Microsystems en 1995. Se destaca por su versatilidad como entorno de ejecución para aplicaciones, gracias a su máquina virtual que permite que el código escrito en Java sea portable entre diferentes sistemas operativos sin necesidad de recompilación. Es uno de los lenguajes más solicitados en la actualidad, conocido por ser orientado a objetos, admitir programación multihilo, ofrecer simplicidad y seguridad, y ser compatible con múltiples plataformas.

Debido a estas características y a la elección de Spring como *framework* para el desarrollo del backend, se optó por utilizar Java 15 para implementar la API REST en este proyecto de grado de Ingeniería Informática del Software.

5.1.1.2. C#

C# es un lenguaje de programación desarrollado por Microsoft, introducido por primera vez en 2000 como parte de su plataforma .NET. Es conocido por su sintaxis elegante, su orientación a objetos robusta y su capacidad para desarrollar aplicaciones tanto para escritorio como para web. En este proyecto se utilizó C# 10 en conjunto con ASP.NET Core MVC versión 7 para desarrollar la parte *frontend* de la aplicación web.

ASP.NET Core MVC es un *framework* moderno y eficiente para construir aplicaciones web y APIs utilizando el ecosistema de .NET Core. Esta versión 7 de ASP.NET Core MVC continúa mejorando en términos de rendimiento, seguridad y características nuevas, lo cual hizo de esta combinación una elección ideal para satisfacer los requisitos del proyecto.

5.1.2. Herramientas de desarrollo

5.1.2.1. IntelliJ IDEA

Para el desarrollo de la API REST se utilizó IntelliJ IDEA 2023.2.4, un entorno de desarrollo integrado (IDE) altamente capacitado y ergonómico para la Java Virtual Machine (JVM). Este IDE ofrece una amplia gama de funciones, incluyendo integración con las tecnologías y *frameworks* más modernos del mercado, lo que facilita trabajar con diversos lenguajes de programación como Java, Scala y Groovy, entre otros.

IntelliJ IDEA destaca por su finalización inteligente de código, análisis continuo del código en tiempo real, capacidades avanzadas de refactorización, atajos que simplifican tareas

como la importación de clases, integración con sistemas de control de versiones y una extensa colección de *plugins*. Actualmente, es reconocido como una de las mejores opciones para el desarrollo con Java.

5.1.2.2. Visual Studio

Para el desarrollo del *frontend* web, se utilizó Visual Studio 2022, un entorno de desarrollo integrado (IDE) desarrollado por Microsoft. Visual Studio 2022 es una herramienta poderosa y versátil que ofrece soporte integral para la creación de aplicaciones web, entre otras plataformas. Este IDE proporciona un conjunto robusto de características que incluyen un editor de código avanzado, depuración integrada, herramientas para el diseño de interfaces de usuario y una amplia gama de extensiones y *plugins* que mejoran la productividad del desarrollador.

Visual Studio 2022 es conocido por su capacidad para trabajar con múltiples lenguajes de programación, incluyendo C#, HTML, CSS, JavaScript y TypeScript, facilitando el desarrollo frontend de aplicaciones web modernas. Su integración con el ecosistema de .NET y su soporte para tecnologías como ASP.NET Core hacen de Visual Studio 2022 una elección destacada para desarrolladores que buscan eficiencia y potencia en sus proyectos de ingeniería informática del software.

5.1.3. Entorno de ejecución

5.1.3.1. Servidor

El servidor utilizado para desplegar todo el sistema es un Dell PowerEdge R540 equipado con dos procesadores Intel Xeon Silver 4210R de 2.4GHz, sumando un total de 40 núcleos de procesamiento. El servidor cuenta con 160GB de memoria RAM DDR4 a 3.200 MHz, proporcionando una capacidad robusta para manejar cargas de trabajo intensivas y demandantes en términos de recursos computacionales y memoria.

Este servidor está configurado con el sistema operativo Ubuntu Server 22.04.1 de 64 bits, conocido por su estabilidad, seguridad y soporte extendido para aplicaciones de servidor. La combinación de hardware y software asegura un entorno de ejecución confiable y eficiente para las aplicaciones y servicios desplegados en este proyecto de Ingeniería Informática del Software.

5.1.3.2. Tomcat

Para la ejecución del servidor de aplicación en este proyecto, se ha utilizado Apache Tomcat 10. Tomcat es un servidor web de código abierto desarrollado por la Apache Software Foundation, diseñado específicamente para ejecutar aplicaciones Java Servlet y JavaServer Pages (JSP). La versión 10 de Tomcat ofrece mejoras significativas en términos de rendimiento y compatibilidad con las últimas especificaciones Java EE y Jakarta EE. Además de ser ligero y fácil de configurar, Tomcat 10 soporta protocolos como HTTP/2 y proporciona herramientas integradas para la gestión y monitoreo de aplicaciones desplegadas, lo que lo convierte en una opción popular para desplegar aplicaciones web Java en diversos entornos de producción.

5.1.3.3. Kestrel

Para el servidor web utilizado en el frontend de este proyecto, se ha optado por Kestrel. Kestrel es un servidor web de código abierto desarrollado por Microsoft y es la opción predeterminada para ejecutar aplicaciones ASP.NET Core. Destacado por su alto rendimiento y eficiencia, Kestrel está diseñado para manejar cargas de trabajo intensivas y escalar de manera efectiva en entornos de producción.

Kestrel soporta HTTP/1.x y HTTP/2, y está integrado estrechamente con ASP.NET Core, lo que facilita su configuración y despliegue junto con aplicaciones web desarrolladas en C#. Además de su velocidad y bajo consumo de recursos, Kestrel proporciona opciones avanzadas para la configuración de seguridad, gestión de conexiones y monitoreo del rendimiento, haciendo de él una elección robusta para servir aplicaciones web modernas con ASP.NET Core en diversas plataformas.

5.1.3.4. PostgreSQL

Para la gestión de datos en este proyecto, se ha utilizado PostgreSQL 14.11 como sistema de gestión de base de datos relacional. PostgreSQL es una opción popular entre los desarrolladores debido a su robustez, fiabilidad y capacidad para manejar grandes volúmenes de datos de manera eficiente. La versión 14.11 ofrece mejoras significativas en términos de rendimiento, seguridad y características avanzadas.

PostgreSQL es conocido por su conformidad con los estándares SQL, soporte para transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad), y la disponibilidad de extensiones que permiten ampliar sus capacidades según las necesidades del proyecto. Además, proporciona características como índices avanzados, replicación nativa, soporte para JSON y tipos de datos personalizados, lo que lo convierte en una opción ideal para aplicaciones que requieren manejo y almacenamiento de datos estructurados de manera eficiente y segura.

5.1.3.5. Neo4J

Para la gestión de datos basados en grafos en este proyecto, se ha utilizado Neo4j versión 4.5 Entreprise. Neo4j es un sistema de gestión de bases de datos basado en grafos, diseñado específicamente para almacenar, consultar y procesar datos interconectados de manera eficiente. La versión 4.5 de Neo4j ofrece mejoras significativas en cuanto a rendimiento, escalabilidad y funcionalidades avanzadas.

Neo4j se destaca por su modelo de datos flexible y su lenguaje de consulta Cypher, optimizado para trabajar con estructuras de grafos. Además de ser altamente eficiente en consultas complejas y operaciones de navegación entre nodos y relaciones, Neo4j ofrece características como la replicación y la alta disponibilidad para asegurar la integridad y la disponibilidad de los datos en entornos de producción.

Con soporte para transacciones ACID y una amplia comunidad de usuarios y desarrolladores, Neo4j es una elección popular para aplicaciones que requieren modelar y analizar relaciones complejas entre datos, como redes sociales, recomendaciones personalizadas, análisis de redes y más.

5.2. Gestión de usuarios y autenticación

La gestión de usuarios y autenticación en el sistema se implementa utilizando una combinación de controladores, modelos, repositorios, servicios de seguridad y payloads. A continuación, se detalla cada uno de estos componentes y su papel en el sistema.

Controladores (Controllers)

Los controladores manejan las solicitudes HTTP y definen los endpoints de la API para la gestión de usuarios y autenticación.

AuthController

- Maneja las solicitudes de autenticación.
- Endpoints principales:
    - /signin: Autentica a un usuario mediante correo electrónico y contraseña.
    - /signin-google: Autentica a un usuario mediante Google ID.
- Genera y devuelve un token JWT tras la autenticación exitosa.

UsersController

- Maneja las operaciones CRUD (Crear, Leer, Actualizar, Borrar) para los usuarios.
- Endpoints principales:
    - GET /api/users: Lista todos los usuarios (solo para administradores).
    - GET /api/users/{user}: Obtiene detalles de un usuario específico.
    - POST /api/users: Registra un nuevo usuario.
    - PUT /api/users/{user}: Actualiza los detalles de un usuario específico.
    - DELETE /api/users/{user}: Elimina un usuario y sus datos asociados.

Modelos (Models)

Los modelos representan las entidades del sistema y se utilizan para mapear los datos a las tablas de la base de datos.

User

- Representa a un usuario en el sistema.
- Campos principales: id, username, surname, email, password, roles, googleId.

Role

- Representa un rol de usuario en el sistema.
- Campos principales: id, name.

ERole (Enum)

- Define los roles posibles que pueden asignarse a un usuario.

- Valores: ROLE_USER, ROLE_ADMIN.

Repositorios (Repositories)

Los repositorios proporcionan abstracciones sobre el acceso a los datos y permiten realizar operaciones CRUD en las entidades.

UserRepository

- Proporciona métodos para buscar usuarios por correo electrónico y Google ID.
- Métodos principales: findByEmail, existsByEmail, findUserByEmailAndGoogleId.

RoleRepository

- Proporciona métodos para buscar roles por nombre.
- Métodos principales: findByName.

Seguridad (Security)

La seguridad se encarga de gestionar la autenticación y autorización en el sistema.

jwt

- *AuthEntryPointJwt*: Maneja los errores de autenticación.
- *AuthTokenFilter*: Filtra y valida los tokens JWT en las solicitudes.
- *JwtUtils*: Proporciona métodos para generar y validar tokens JWT.

services

- *UserDetailsImpl*: Implementa UserDetails y proporciona detalles del usuario autenticado.
- *UserDetailsServiceImpl*: Carga los detalles del usuario desde la base de datos para la autenticación.

WebSecurityConfig

- Configura la seguridad de la aplicación, incluyendo la protección de los endpoints y la gestión de las políticas de autenticación y autorización.

Payloads

Los payloads se utilizan para transferir datos entre el cliente y el servidor.

Requests

- *LoginRequest*: Contiene los datos para la solicitud de inicio de sesión (correo electrónico y contraseña).
- *LoginGoogleRequest*: Contiene los datos para la solicitud de inicio de sesión con Google (correo electrónico y Google ID).
- *UserPostRequest*: Contiene los datos para registrar un nuevo usuario.
- *UserPutRequest*: Contiene los datos para actualizar un usuario existente.

- *AccessPostRequest*: Contiene los datos para una solicitud específica de acceso.

Responses

- *JwtResponse*: Contiene el token JWT y los detalles del usuario autenticado.
- *MessageResponse*: Contiene mensajes de respuesta genéricos.
- *UserResponse*: Contiene los detalles de un usuario.

La implementación de la gestión de usuarios y autenticación en el sistema se basa en una arquitectura bien estructurada que utiliza controladores para manejar las solicitudes HTTP, modelos para representar las entidades, repositorios para el acceso a los datos, componentes de seguridad para la autenticación y autorización, y payloads para la transferencia de datos. Esta estructura permite una gestión eficiente y segura de los usuarios y sus datos en el sistema. La inclusión de un enum ERole facilita la definición y gestión de roles, y los componentes de seguridad aseguran que solo los usuarios autenticados y autorizados puedan acceder a los recursos protegidos.

5.3. Gestión de programas

En esta sección se detalla cómo se gestiona la creación, actualización, consulta y eliminación de programas en la API. Se utilizan controlador, modelo, repositorio y payloads para facilitar estas operaciones.

Controlador (Controller)

El controlador maneja las solicitudes HTTP relacionadas con la gestión de programas y define los endpoints de la API correspondientes.

ProgramsController

- Maneja las operaciones CRUD (Crear, Leer, Actualizar, Borrar) para los programas.
- Endpoints principales:
    - GET /api/programs: Lista todos los programas visibles.
    - GET /api/programs/{rdn}: Obtiene detalles de un programa específico.
    - POST /api/programs: Crea un nuevo programa.
    - PUT /api/programs/{rdn}: Actualiza los detalles de un programa específico.
    - DELETE /api/programs/{id}: Elimina un programa y sus datos asociados.
    - GET /api/programs/{rdn}/accesses: Obtiene los usuarios con acceso a cierto programa.
    - GET /api/programs/{rdn}/accesses/{user}: Obtiene detalles de un usuario específico con acceso a cierto programa.
    - POST /api/programs/{rdn}/accesses: Otorga acceso a un usuario específico a cierto programa.

- DELETE /api/programs/{rdn}/accesses/{user}: Revoca el acceso de un usuario específico a cierto programa.
- GET /api/programs/{rdn}/results: Retorna los resultados de los análisis ejecutados en cierto programa.

Modelos (Models)

Los modelos representan las entidades del sistema y se utilizan para mapear los datos a las tablas de la base de datos.

Program

- Representa un programa en el sistema.
- Campos principales: id, rdn, name, description, path, neo4j_database, sourceCode, url, javacOptions, date, visibility, status, updated, numberOfNodes, numberOfFiles, linesOfCode, statusMessage, owner, sharedUsers, resultPrograms.

EVisibility

- Define la visibilidad de los programas y los análisis del sistema.
- Valores: PUBLIC, PRIVATE, SHARED.

EStatus

- Define el estado de los programas y los resultados del sistema.
- Valores: WAITING, CLONING, CLONING_FAIL, CLONED, INSERTING, INSERTING_FAIL, INSERTED, ANALYZING, ANALYZING_FAIL, ANALYZED.

Repositorios (Repositories)

Los repositorios proporcionan abstracciones sobre el acceso a los datos y permiten realizar operaciones CRUD en las entidades.

ProgramRepository

- Proporciona métodos para buscar programas por rdn y otros criterios.
- Métodos principales: findByRdn, findByOwner_Email, findPublicPrograms, findProgramsSharedWithUser, findProgramsAccessedByUser, updateStatus.

ProgramRepositoryImpl

- Agrega al anterior el método findProgramsAccessedByUser.

Payloads

Los payloads se utilizan para transferir datos entre el cliente y el servidor.

Requests

- *ProgramPostRequest*: Contiene los datos para la solicitud de creación de un nuevo programa.
- *ProgramPutRequest*: Contiene los datos para la solicitud de actualización de un programa existente.

La implementación de la gestión de programas en el sistema se basa en una arquitectura estructurada que utiliza un controlador para manejar las solicitudes HTTP, modelos para representar las entidades, repositorios para el acceso a los datos, servicios para la lógica de negocio y payloads para la transferencia de datos. Esta estructura facilita una gestión eficiente y segura de los programas y sus datos en el sistema.

5.4. Gestión de análisis

En esta sección se detalla cómo se gestiona la creación, actualización, consulta y eliminación de análisis en la API. Se utilizan controlador, modelo, repositorio y payloads para facilitar estas operaciones.

Controlador (Controller)

El controlador maneja las solicitudes HTTP relacionadas con la gestión de análisis y define los endpoints de la API correspondientes.

AnalysisController

- Maneja las operaciones CRUD (Crear, Leer, Actualizar, Borrar) para los análisis.
- Endpoints principales:
    - GET /api/analyses: Lista todos los análisis visibles.
    - GET /api/analyses/{rdn}: Obtiene detalles de un análisis específico.
    - POST /api/analyses: Crea un nuevo análisis.
    - PUT /api/analyses/{rdn}: Actualiza los detalles de un análisis específico.
    - DELETE /api/analyses/{rdn}: Elimina un análisis y sus datos asociados.
    - GET /api/analyses/{rdn}/accesses: Obtiene los usuarios con acceso a cierto análisis.
    - GET /api/analyses/{rdn}/accesses/{user}: Obtiene detalles de un usuario específico con acceso a cierto análisis.
    - POST /api/analyses/{rdn}/accesses: Otorga acceso a un usuario específico a cierto análisis.
    - DELETE /api/analyses/{rdn}/accesses/{user}: Revoca el acceso de un usuario específico a cierto análisis.

CategoriesController

- Endpoint:
    - GET /api/categories: Lista todas las categorías.

SyntacticConstructController

- Endpoint:
    - GET /api/syntacticconstructs: Lista todas las construcciones sintácticas.

Modelos (Models)

Los modelos representan las entidades del sistema y se utilizan para mapear los datos a las tablas de la base de datos.

Analysis

- Representa un análisis en el sistema.
- Campos principales: id, rdn, name, category, syntacticConstruct, description, query, date, visibility, owner, sharedUsers, resultProgramAnalyses.

Category

- Representa la categoría a la que pertenece un análisis.
- Campos principales: id, name.

SyntacticConstruct

- Representa la sintaxis utilizada en el análisis.
- Campos principales: id, name label.

Repositorios (Repositories)

Los repositorios proporcionan abstracciones sobre el acceso a los datos y permiten realizar operaciones CRUD en las entidades.

AnalysisRepository

- Proporciona métodos para buscar análisis por rdn y otros criterios.
- Métodos principales: findByRdn, findByOwner_Email, findPublicAnalyses, findAnalysesSharedWithUser, findAnalysesAccessedByUser.

AnalysisRepositoryImpl

- Agrega al anterior el método findAnalysesAccessedByUser.

CategoryRepository

- Método: findByName.

SyntacticConstructRepository

- Método: findByName.

Payloads

Los payloads se utilizan para transferir datos entre el cliente y el servidor.

Requests

- AnalysisPostRequest: Contiene los datos para la solicitud de creación de un nuevo análisis.
- AnalysisPutRequest: Contiene los datos para la solicitud de actualización de un análisis existente.

La implementación de la gestión de análisis en el sistema se basa en una arquitectura estructurada que utiliza un controlador para manejar las solicitudes HTTP, modelos para representar las entidades, repositorios para el acceso a los datos, servicios para la lógica de negocio y payloads para la transferencia de datos. Esta estructura facilita una gestión eficiente y segura de los análisis y sus datos en el sistema.

5.5. Gestión de resultados

En esta sección se detalla cómo se gestiona la creación, consulta y eliminación de resultados en la API. Se utilizan controladores, modelos, repositorios y payloads para facilitar estas operaciones.

Controlador (ResultsController)

El controlador maneja las solicitudes HTTP relacionadas con la gestión de resultados y define los endpoints de la API correspondientes.

ResultsController

- Maneja las operaciones CRUD (Crear, Leer, Actualizar, Borrar) para los resultados.
- Endpoints principales:
    - GET /api/results: Lista todos los resultados visibles.
    - GET /api/results/{id}: Obtiene detalles de un resultado específico.
    - GET /api/results/{id}/programs/{programRdn}: Obtiene detalles del resultado de un programa específico.
    - POST /api/results: Crea un nuevo resultado.
    - DELETE /api/results/{id}: Elimina un resultado y sus datos asociados.
    - DELETE /api/results/{id}/programs/{programRdn}: Elimina el resultado de un programa específico.

Modelos (Models)

Los modelos representan las entidades del sistema y se utilizan para mapear los datos a las tablas de la base de datos.

Result

- Representa un resultado en el sistema, asociado a un usuario propietario y a varios programas analizados.
- Campos principales: id, comment, date, updated, status, owner, resultPrograms.

ResultProgram

- Representa un programa específico asociado a un resultado, con detalles de análisis realizados sobre él.

- Campos principales: id, date, updated, status, program_owner_email, program_owner_name, program_rdn, program_name, program_description, program_build_date, program_updated_date, program_numberOfNodes, program_numberOfFiles, program_linesOfCode, result, program, resultProgramAnalyses.

ResultProgramAnalysis

- Representa un análisis específico realizado sobre un programa asociado a un resultado.
- Campos principales: id, date, updated, status, checkedNodes, invalidNodes, analysis_rdn, analysis_owner_name, analysis_owner_email, analysis_category, analysis_syntacticConstruct, analysis_name, analysis_description, analysis_query, resultProgram, analysis, details.

ResultProgramAnalysisDetail

- Representa ciertos detalles del anterior.
- Campos principales: id, location, message, detailLine, detailColumn, node.

Repositorios (Repositories)

Los repositorios proporcionan abstracciones sobre el acceso a los datos y permiten realizar operaciones CRUD en las entidades.

ResultRepository

- Proporciona métodos para buscar y manipular resultados en la base de datos.
- Métodos principales: findByOwner_Email, updateStatus.

ResultProgramRepository

- Proporciona métodos para buscar y manipular resultados de programas en la base de datos.
- Métodos principales: updateStatus, setupDate.

ResultProgramAnalysisRepository

- Proporciona métodos para buscar y manipular análisis realizados a programas asociados a resultados en la base de datos.
- Métodos principales: updateStatus, setupDate.

Payloads

Los payloads se utilizan para transferir datos entre el cliente y el servidor.

Requests

- *ResultPostRequest*: Contiene los datos para la solicitud de creación de un nuevo resultado.

Responses

- *ResultResponse*: Contiene los detalles de un resultado para responder a las solicitudes de consulta.
- *ResultProgramResponse*: Contiene los detalles del resultado de un programa para responder a las solicitudes de consulta.
- *ResultProgramListResponse*: Contiene los detalles de una lista de resultados de programas para responder a las solicitudes de consulta.
- *ResultProgramAnalysisResponse*: Contiene los detalles de un análisis realizado a un programa asociado a un resultado para responder a las solicitudes de consulta.
- *ResultProgramAnalysisDetailResponse*: Contiene los detalles de un detalle específico de un análisis realizado a un programa asociado a un resultado para responder a las solicitudes de consulta.

La implementación de la gestión de resultados en el sistema se basa en una arquitectura estructurada que utiliza un controlador para manejar las solicitudes HTTP, modelos para representar las entidades principales, repositorios para el acceso eficiente a los datos en la base de datos, servicios para la lógica de negocio y payloads para la transferencia de datos entre el cliente y el servidor. Esta estructura facilita una gestión eficiente y segura de los resultados y sus componentes relacionados en el sistema, asegurando la integridad de los datos y el cumplimiento de las reglas de negocio establecidas.

5.6. Desarrollo de la aplicación web

En el desarrollo de aplicaciones web modernas, una estructura organizativa clara y eficiente es crucial para asegurar la mantenibilidad, escalabilidad y claridad del código. En esta aplicación, se ha adoptado una arquitectura bien definida que facilita la gestión de las funcionalidades y la experiencia del usuario.

Controllers

Los controladores manejan las solicitudes HTTP y definen los endpoints de la aplicación web.

- *AnalysesController*: Controlador para la gestión de análisis.
- *AuthController*: Controlador para la autenticación.
- *HomeController*: Controlador para las vistas principales y rutas generales.
- *ProgramsController*: Controlador para la gestión de programas.
- *ResultsController*: Controlador para la gestión de resultados.
- *StatisticsController*: Controlador para la visualización y gestión de estadísticas.
- *UsersController*: Controlador para la gestión de usuarios.

Models

Los modelos representan las entidades y estructuras de datos utilizadas en la aplicación web. Pueden estar organizados en diferentes carpetas según el contexto o el controlador al que pertenezcan.

- *Analyses/*: Modelos relacionados con los análisis.
- *Auth/*: Modelos relacionados con la autenticación y usuarios.
- *Programs*/: Modelos relacionados con los programas.
- *Results*/: Modelos relacionados con los resultados.
- *Statistics*/: Modelos relacionados con las estadísticas.
- *Users/*: Modelos relacionados con los usuarios.

Services

Los servicios contienen la lógica de negocio de la aplicación web y se comunican con la capa de persistencia si es necesario.

- *AnalysesService*: Lógica de negocio relacionada con los análisis.
- *AuthService*: Lógica de negocio relacionada con la autenticación y gestión de usuarios.
- *CategoriesService*: Lógica de negocio relacionada con las categorías.
- *ProgramsService*: Lógica de negocio relacionada con los programas.
- *ResultsService*: Lógica de negocio relacionada con los resultados.
- *StatisticsService*: Lógica de negocio relacionada con las estadísticas.
- *SyntacticConstructsService*: Lógica de negocio relacionada con las construcciones sintácticas.
- *UsersService*: Lógica de negocio relacionada con los usuarios.

Views

Las vistas son plantillas o páginas HTML que se renderizan y presentan al usuario final. Pueden estar organizadas en diferentes carpetas según la funcionalidad o el controlador al que correspondan.

- *Analyses/*: Vistas relacionadas con la gestión de análisis.
- *Auth/*: Vistas relacionadas con la autenticación y gestión de usuarios.
- *Home/*: Vistas relacionadas con las páginas principales.
- *Programs/*: Vistas relacionadas con la gestión de programas.
- *Results/*: Vistas relacionadas con la gestión de resultados.
- *Statistics/*: Vistas relacionadas con la visualización de estadísticas.
- *Shared/*: Vistas compartidas entre diferentes secciones de la aplicación.

- *Users/*: Vistas relacionadas con la gestión de usuarios.

Assets

Los activos estáticos como CSS, JavaScript, imágenes, etc., utilizados por las vistas y la interfaz de usuario.

- *css/*: Hojas de estilo CSS.
- *js/*: Scripts JavaScript.
- *images/*: Imágenes utilizadas en la aplicación.

Capítulo 6. Desarrollo de las pruebas

6.1. Pruebas unitarias para la gestión de usuarios y autenticación

En esta sección se presentan capturas de pantalla de pruebas unitarias exitosas realizadas para validar la funcionalidad de la gestión de usuarios y autenticación en el proyecto. Las pruebas unitarias son fundamentales para asegurar que los componentes clave del sistema, como el registro, autenticación, edición y eliminación de perfiles de usuario. El número total de pruebas unitarias que se han realizado para la gestión de usuarios es de 20 pruebas y para la autenticación de 4 pruebas.

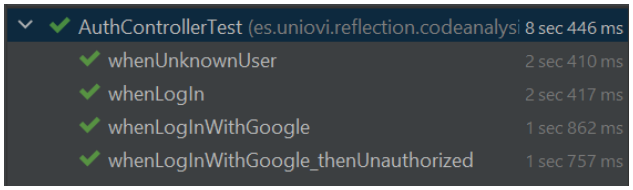


Ilustración 29. Pruebas unitarias de autenticación

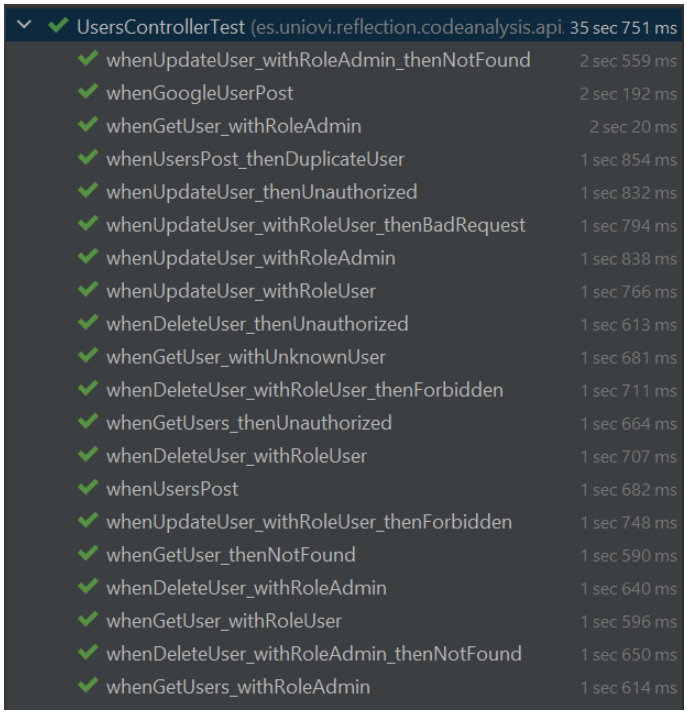


Ilustración 30. Pruebas unitarias de gestión de usuarios

6.2. Pruebas unitarias para la gestión de programas

En esta sección se presentan capturas de pantalla que muestran el resultado satisfactorio de las pruebas unitarias implementadas para validar la gestión de programas en el proyecto. Las pruebas unitarias son fundamentales para asegurar que las funcionalidades

relacionadas con la carga y gestión de programas Java se ejecuten de manera precisa y confiable. El número total de pruebas unitarias que se han realizado para la gestión de programas es de 51 pruebas.

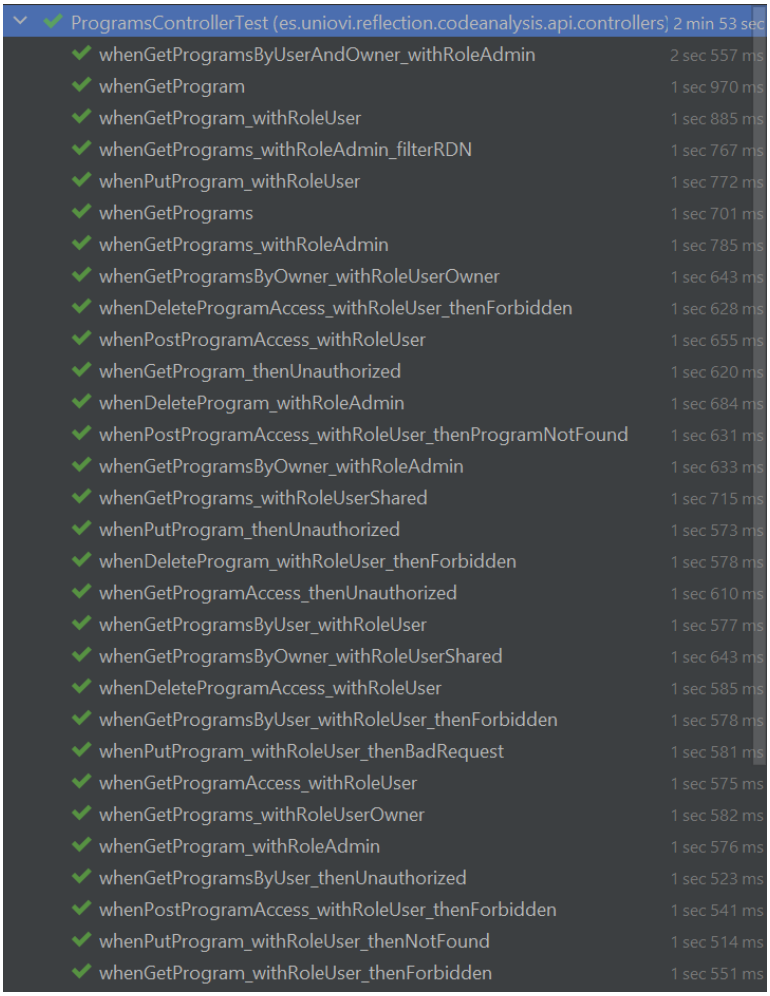


Ilustración 31. Pruebas unitarias de gestión de programas (I)

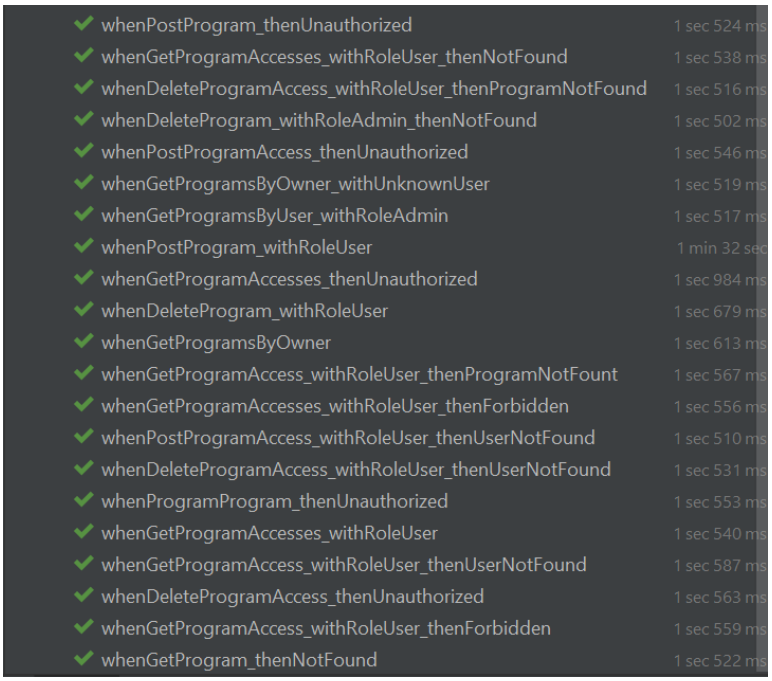


Ilustración 32. Pruebas unitarias de gestión de programas (II)

6.3. Pruebas unitarias para la gestión de análisis

En esta sección se presentan capturas de pantalla que muestran el resultado satisfactorio de las pruebas unitarias implementadas para validar la gestión de análisis en el proyecto. Las pruebas unitarias desempeñan un papel crucial en asegurar que las funciones relacionadas con la creación, actualización y eliminación de análisis se ejecuten correctamente y cumplan con los requisitos del sistema. El número total de pruebas unitarias que se han realizado para la gestión de análisis es de 56 pruebas.

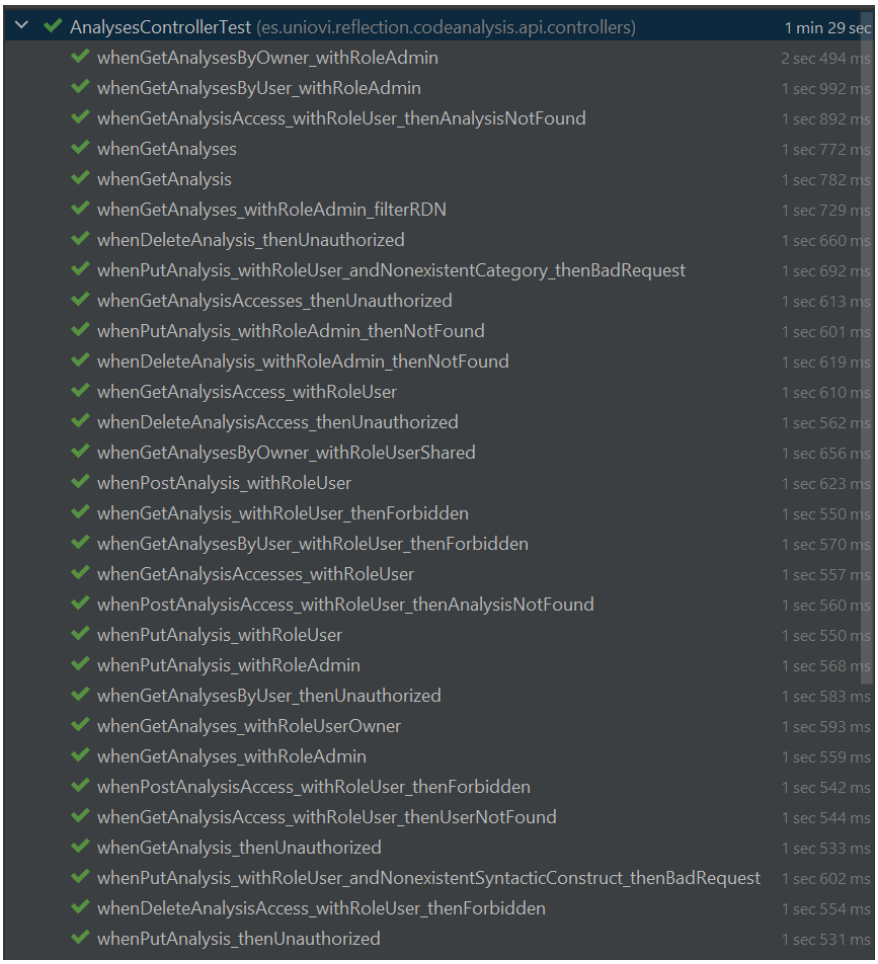


Ilustración 33. Pruebas unitarias de gestión de análisis (I)

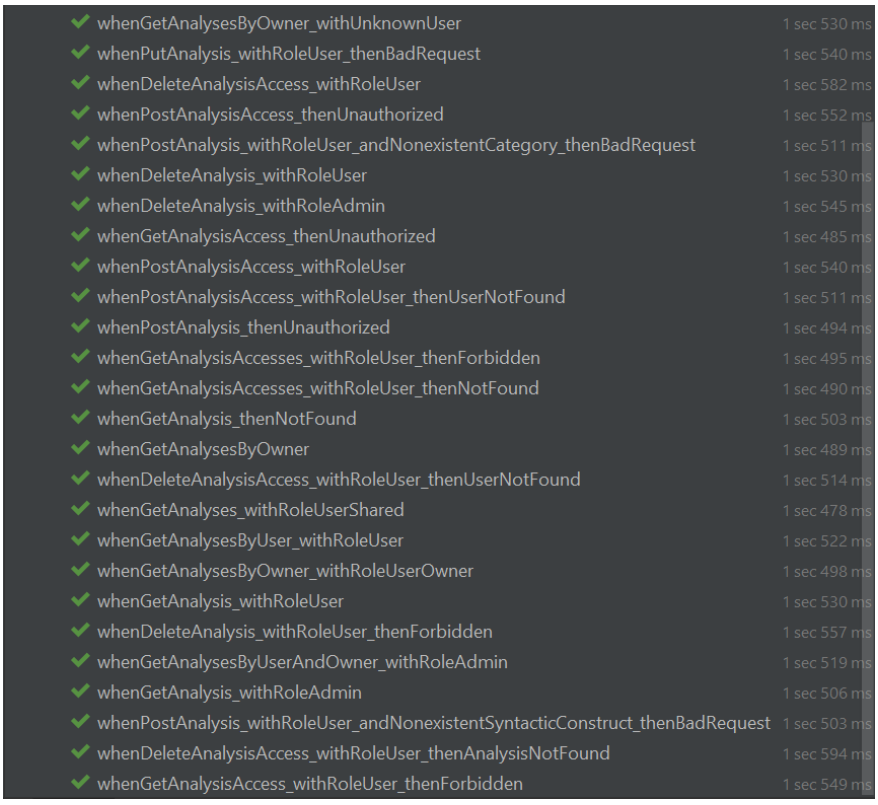


Ilustración 34. Pruebas unitarias de gestión de análisis (II)

6.4. Pruebas unitarias para la gestión de resultados

En esta sección se presentan las capturas de pantalla de las pruebas unitarias exitosas implementadas para validar la gestión de resultados en el proyecto. Estas pruebas son fundamentales para garantizar que las funcionalidades relacionadas con la ejecución de análisis, almacenamiento y acceso a resultados se realicen correctamente y cumplan con los estándares del sistema. El número total de pruebas unitarias que se han realizado para la gestión de resultados es de 29 pruebas.

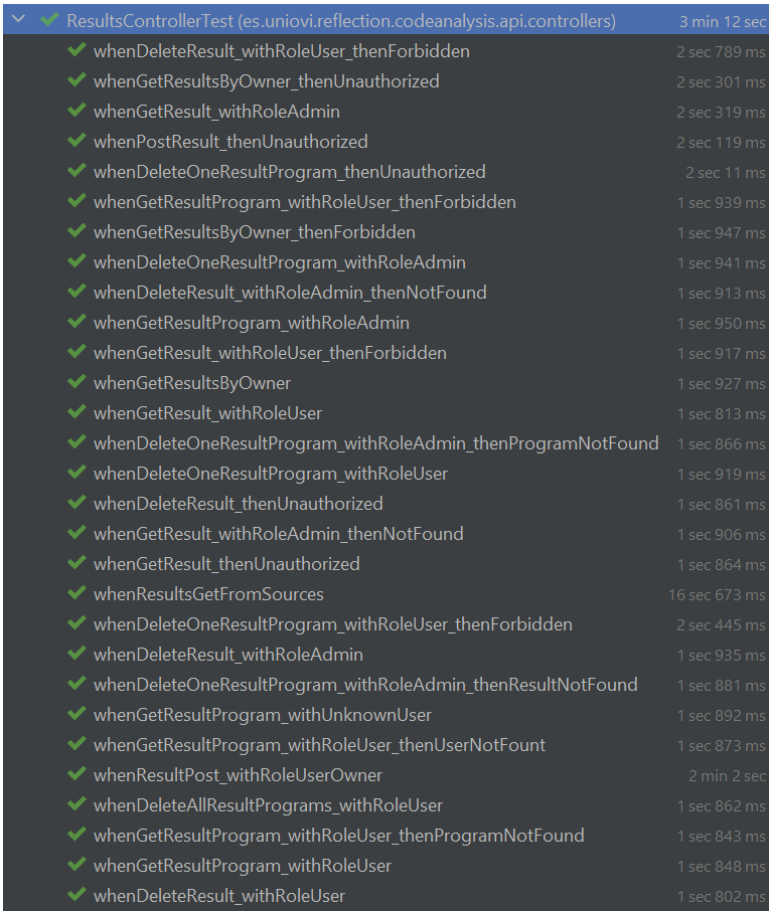


Ilustración 35. Pruebas unitarias de gestión de resultados

6.5. Pruebas unitarias adicionales

En esta sección se presentan detalles sobre la implementación de pruebas unitarias adicionales enfocadas en las estadísticas generales del sistema. Estas pruebas unitarias son esenciales para asegurar que los componentes encargados de generar y manejar estadísticas generales funcionen correctamente y proporcionen datos precisos y fiables. El número total de pruebas unitarias adicionales que se han realizado es de 3 pruebas.

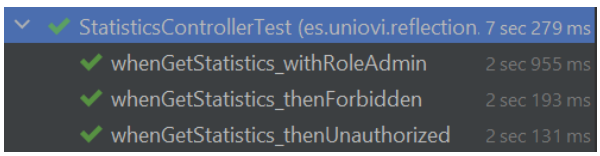


Ilustración 36. Pruebas unitarias de estadísticas generales

6.6. Pruebas de integración

6.6.1. CU-W-01 Registrarse en el sistema

Identificador	Título	Escenario	Resultado esperado
CP-W-01	Registro en el sistema de forma correcta	Un usuario rellena todos los campos del formulario de registro de forma correcta y pulsa sobre crear	El sistema lo registrar y redirige al usuario a la página de confirmación de registro
CP-W-02	Registro en el sistema con un email existente	Un usuario rellena todos los campos del formulario de registro, pero el email ya existe en el sistema y pulsa sobre crear	El sistema no permite el registro del usuario y muestra un error indicando que el email introducido ya existe

Tabla 56. Pruebas de integración – Registrarse en el sistema

6.6.2. CU-W-02 Identificarse en el sistema

Identificador	Título	Escenario	Resultado esperado
CP-W-03	Inicio de sesión con usuario y contraseña correctos	Se introduce un usuario y contraseña correctos	El sistema permite iniciar sesión de forma correcta
CP-W-04	Inicio de sesión con credenciales incorrectas	Se introducen credenciales incorrectas	El sistema no permite iniciar sesión y muestra un error de credenciales incorrectas
CP-W-05	Acceso a rutas de la aplicación sin estar autenticado en el sistema	Se intenta realizar acciones en la aplicación que requieren que el usuario esté autenticado	El sistema no permite realizar la acción y muestro un error de no autorizado

Tabla 57. Pruebas de integración – Identificarse en el sistema

6.6.3. CU-W-03 Registrarse en el sistema con Google

Identificador	Título	Escenario	Resultado esperado
CP-W-06	Registro en el sistema mediante Google de forma correcta	Un usuario pulsa sobre registrarse con Google y selecciona su cuenta de Google	El sistema registra al usuario y lo redirige a la página principal con la sesión iniciada

CP-W-07	Registro en el sistema con una cuenta de Google previamente registrada	Un usuario pulsa sobre registrarse con Google y selecciona su cuenta de Google	El sistema no permite el registro del usuario y muestra un error indicando que el email ya existe

Tabla 58. Pruebas de integración – Registrarse en el sistema con Google

6.6.4. CU-W-04 Identificarse en el sistema con Google

Identificador	Título	Escenario	Resultado esperado
CP-W-08	Inicio de sesión mediante una cuenta de Google registrada de forma correcta	Un usuario pulsa sobre iniciar sesión con Google y selecciona su cuenta	El sistema inicia sesión y lo redirige a la página principal
CP-W-09	Inicio de sesión mediante Google con una cuenta que no está registrada	Un usuario pulsa sobre iniciar sesión con Google y selecciona su cuenta	El sistema registra al usuario y lo redirige a la página principal con la sesión iniciada

Tabla 59. Pruebas de integración – Identificarse en el sistema con Google

6.6.5. CU-W-05 Visualizar los programas públicos

Identificador	Título	Escenario	Resultado esperado
CP-W-10	Visualización de los programas públicos sin autenticarse	Un usuario no autenticado accede a la vista de programas	El sistema muestra un listado con los programas públicos registrados

Tabla 60. Pruebas de integración – Visualizar los programas públicos

6.6.6. CU-W-06 Visualizar los detalles de un programa público

Identificador	Título	Escenario	Resultado esperado
CP-W-11	Visualización de los detalles de un programa público sin autenticarse	Un usuario no autenticado accede a la vista de programas y pulsa sobre un programa público	El sistema muestra los detalles de dicho programa público

Tabla 61. Pruebas de integración – Visualizar los detalles de un programa público

6.6.7. CU-W-07 Visualizar el listado de resultados de un programa público

Identificador	Título	Escenario	Resultado esperado
CP-W-12	Visualización del listado de resultados de un programa	Un usuario no autenticado accede a la vista de	El sistema muestra un listado con los resultados visibles de

	público sin autenticarse	programas, pulsa sobre un programa público y pulsa sobre resultados	dicho programa público

Tabla 62. Pruebas de integración – Visualizar el listado de resultados de un programa público

6.6.8. CU-W-08 Visualizar los detalles del resultado de un programa público

Identificador	Título	Escenario	Resultado esperado
CP-W-13	Visualización de los detalles de un resultado de un programa público sin autenticarse	Un usuario no autenticado accede a la vista de programas, pulsa sobre un programa público, pulsa sobre resultados y pulsa sobre un resultado	El sistema muestra los detalles del resultado visible de un programa público

Tabla 63. Pruebas de integración – Visualizar los detalles del resultado de un programa público

6.6.9. CU-W-09 Visualizar los análisis públicos

Identificador	Título	Escenario	Resultado esperado
CP-W-14	Visualización de los análisis públicos sin autenticarse	Un usuario no autenticado accede a la vista de análisis	El sistema muestra un listado con los análisis públicos registrados

Tabla 64. Pruebas de integración – Visualizar los análisis públicos

6.6.10. CU-W-10 Visualizar los detalles de un análisis público

Identificador	Título	Escenario	Resultado esperado
CP-W-15	Visualización de los detalles de un análisis público sin autenticarse	Un usuario no autenticado accede a la vista de análisis y pulsa sobre un análisis público	El sistema muestra los detalles de dicho análisis público

Tabla 65. Pruebas de integración – Visualizar los detalles de un análisis público

6.6.11. CU-W-11 Visualizar los detalles de un usuario

Identificador	Título	Escenario	Resultado esperado
CP-W-16	Visualización del perfil de un usuario sin autenticarse	Un usuario no autenticado accede a los detalles de un programa o un	El sistema muestra los detalles de dicho usuario propietario

		análisis público y pulsa sobre el propietario	

Tabla 66. Pruebas de integración – Visualizar los detalles de un usuario

6.6.12. CU-W-12 Visualizar los programas públicos de un usuario

Identificador	Título	Escenario	Resultado esperado
CP-W-17	Visualización de los programas públicos de un usuario	Un usuario no autenticado, desde el perfil de otro usuario, pulsa sobre los programas de dicho usuario	El sistema muestra los programas públicos del usuario

Tabla 67. Pruebas de integración – Visualizar los programas públicos de un usuario

6.6.13. CU-W-13 Visualizar los análisis públicos de un usuario

Identificador	Título	Escenario	Resultado esperado
CP-W-18	Visualización de los análisis públicos de un usuario	Un usuario no autenticado, desde el perfil de otro usuario, pulsa sobre los análisis de dicho usuario	El sistema muestra los análisis públicos del usuario

Tabla 68. Pruebas de integración – Visualizar los análisis públicos de un usuario

6.6.14. CU-W-14 Analizar código java sin autenticarse

Identificador	Título	Escenario	Resultado esperado
CP-W-19	Análisis básico de código java sin autenticarse	Un usuario no autenticado accede al análisis básico e introduce o selecciona un programa y un análisis	El sistema muestra los resultados de análisis básico realizado
CP-W-20	Análisis básico sin autenticarse con código o consulta incorrectos	Un usuario no autenticado accede al análisis básico e introduce incorrectamente un programa y/o un análisis	El sistema muestra un error relacionado con el código y/o la consulta introducidos

Tabla 69. Pruebas de integración – Analizar código java sin autenticarse

6.6.15. CU-W-15 Cerrar sesión

Identificador	Título	Escenario	Resultado esperado
CP-W-21	Cerrar sesión en el sistema	Un usuario autenticado pulsa sobre su nombre y sobre cerrar sesión	El sistema redirige a la página principal y elimina los datos de sesión del almacenamiento local del navegador
CP-W-22	Acceso a rutas de la aplicación tras haber cerrado sesión en el sistema	Se intenta realizar acciones en la aplicación que requieren que el usuario esté autenticado	El sistema no permite realizar la acción y muestro un error de no autorizado

Tabla 70. Pruebas de integración – Cerrar sesión

6.6.16. CU-W-16 Visualizar el perfil de un usuario

Identificador	Título	Escenario	Resultado esperado
CP-W-23	Visualización del perfil de un usuario	Un usuario autenticado pulsa sobre su nombre y sobre perfil	El sistema muestra los detalles del usuario

Tabla 71. Pruebas de integración – Visualizar el perfil de un usuario

6.6.17. CU-W-17 Visualizar los programas de un usuario

Identificador	Título	Escenario	Resultado esperado
CP-W-24	Visualización de los programas de un usuario	Un usuario autenticado, desde su perfil, pulsa sobre los programas	El sistema muestra los programas del usuario

Tabla 72. Pruebas de integración – Visualizar los programas de un usuario

6.6.18. CU-W-18 Visualizar los análisis de un usuario

Identificador	Título	Escenario	Resultado esperado
CP-W-25	Visualización de los análisis de un usuario	Un usuario autenticado, desde su perfil, pulsa sobre los análisis	El sistema muestra los análisis del usuario

Tabla 73. Pruebas de integración – Visualizar los análisis de un usuario

6.6.19. CU-W-19 Modificar los datos del perfil

Identificador	Título	Escenario	Resultado esperado

CP-W-26	Modificación de los datos del perfil de un usuario	Un usuario autenticado, desde su perfil, pulsa sobre editar, modifica los datos y guarda la nueva información	El sistema actualiza los datos del usuario y redirige al perfil del usuario
CP-W-27	Modificación de los datos del perfil de un usuario con campos introducidos erróneos	Un usuario autenticado, desde su perfil, pulsa sobre editar, modifica los datos e intenta guardar la nueva información	El sistema muestra un error asociado con los campos erróneos introducidos

Tabla 74. Pruebas de integración – Modificar los datos del perfil

6.6.20. CU-W-20 Visualizar el listado de programas

Identificador	Título	Escenario	Resultado esperado
CP-W-28	Visualización de los programas visibles	Un usuario autenticado accede a la vista de programas	El sistema muestra un listado con los programas visibles registrados

Tabla 75. Pruebas de integración – Visualizar el listado de programas

6.6.21. CU-W-21 Visualizar los detalles de un programa

Identificador	Título	Escenario	Resultado esperado
CP-W-29	Visualización de los detalles de un programa	Un usuario autenticado accede a la vista de programa y pulsa sobre un programa visible	El sistema muestra los detalles de dicho programa visible

Tabla 76. Pruebas de integración – Visualizar los detalles de un programa

6.6.22. CU-W-22 Visualizar el listado de resultados de un programa

Identificador	Título	Escenario	Resultado esperado
CP-W-30	Visualización del listado de resultados de un programa visible	Un usuario autenticado accede a la vista de programas, pulsa sobre un programa visible y pulsa sobre resultados	El sistema muestra un listado con los resultados visibles de dicho programa

Tabla 77. Pruebas de integración – Visualizar el listado de resultados de un programa

6.6.23. CU-W-23 Visualizar los detalles del resultado de un programa desde el programa

Identificador	Título	Escenario	Resultado esperado
CP-W-31	Visualización de los detalles de un resultado de un programa visible	Un usuario autenticado accede a la vista de programas, pulsa sobre un programa visible, pulsa sobre resultados y pulsa sobre un resultado	El sistema muestra los detalles del resultado visible de un programa visible

Tabla 78. Pruebas de integración – Visualizar los detalles del resultado de un programa desde el programa

6.6.24. CU-W-24 Registrar un nuevo programa

Identificador	Título	Escenario	Resultado esperado
CP-W-32	Registro de un nuevo programa en el sistema	Un usuario autenticado accede a la vista de programas, pulsa en nuevo, introduce los datos correspondientes y pulsa en insertar programa	El sistema registra el programa y redirige a una página de confirmación de registro
CP-W-33	Registro de un nuevo programa en el sistema con campos incorrectos	Un usuario autenticado accede a la vista de programas, pulsa en nuevo, introduce los datos e intenta insertar el programa	El sistema muestra el error correspondiente a los campos incorrectos

Tabla 79. Pruebas de integración – Registrar un nuevo programa

6.6.25. CU-W-25 Modificar un programa

Identificador	Título	Escenario	Resultado esperado
CP-W-34	Modificación de un programa propio	Un usuario autenticado, desde los detalles de un programa propio, pulsa sobre editar, introduce y guarda la nueva información	El sistema guarda las actualizaciones del programa y redirige a la pantalla de detalles del programa

CP-W-35	Modificación de un programa propio con campos incorrectos	Un usuario autenticado, pulsa sobre editar, introduce e intenta guardar la nueva información	El sistema muestra el error correspondiente a los campos incorrectos

Tabla 80. Pruebas de integración – Modificar un programa

6.6.26. CU-W-26 Eliminar un programa

Identificador	Título	Escenario	Resultado esperado
CP-W-36	Eliminación de un programa propio	Un usuario autenticado, desde los detalles de un programa propio, pulsa sobre eliminar y confirma la eliminación	El sistema elimina el programa y redirige a la vista de programas

Tabla 81. Pruebas de integración – Eliminar un programa

6.6.27. CU-W-27 Ver un listado de los usuarios que tienen acceso a un programa

Identificador	Título	Escenario	Resultado esperado
CP-W-37	Visualización de los usuarios que tienen acceso a un programa de visibilidad compartida	Un usuario autenticado accede a los detalles de un programa de visibilidad compartida y pulsa sobre usuarios compartidos para visualizar un listado de estos	El sistema muestra el listado de usuarios que tienen acceso a dicho programa

Tabla 82. Pruebas de integración – Ver un listado de los usuarios que tienen acceso a un programa

6.6.28. CU-W-28 Dar acceso a un usuario a un programa

Identificador	Título	Escenario	Resultado esperado
CP-W-38	Asignación de acceso a un usuario a un programa de visibilidad compartida	Un usuario autenticado accede a los detalles de un programa de visibilidad compartida, pulsa sobre usuarios	El sistema añade al usuario introducido en el listado de usuarios con acceso al programa

		compartidos y añade un usuario registrado	

Tabla 83. Pruebas de integración – Dar acceso a un usuario a un programa

6.6.29. CU-W-29 Eliminar el acceso de un usuario a un programa

Identificador	Título	Escenario	Resultado esperado
CP-W-39	Eliminación de acceso a un usuario a un programa de visibilidad compartida	Un usuario autenticado accede a los detalles de un programa de visibilidad compartida, pulsa sobre usuarios compartidos y elimina a un usuario del listado	El sistema elimina al usuario del listado de usuarios con acceso al programa

Tabla 84. Pruebas de integración – Eliminar el acceso de un usuario a un programa

6.6.30. CU-W-30 Visualizar el listado de análisis

Identificador	Título	Escenario	Resultado esperado
CP-W-40	Visualización de los análisis visibles	Un usuario autenticado accede a la vista de análisis	El sistema muestra un listado con los análisis visibles registrados

Tabla 85. Pruebas de integración – Visualizar el listado de análisis

6.6.31. CU-W-31 Visualizar los detalles de un análisis

Identificador	Título	Escenario	Resultado esperado
CP-W-41	Visualización de los detalles de un análisis	Un usuario autenticado accede a la vista de análisis y pulsa sobre un análisis visible	El sistema muestra los detalles de dicho análisis visible

Tabla 86. Pruebas de integración – Visualizar los detalles de un análisis

6.6.32. CU-W-32 Registrar un nuevo análisis

Identificador	Título	Escenario	Resultado esperado
CP-W-42	Registro de un nuevo análisis en el sistema	Un usuario autenticado accede a la vista de análisis, pulsa en nuevo, introduce los datos correspondientes y	El sistema registra el análisis y redirige a una página de confirmación de registro

		pulsa en crear análisis	
CP-W-43	Registro de un nuevo análisis en el sistema con campos incorrectos	Un usuario autenticado accede a la vista de análisis, pulsa en nuevo, introduce los datos e intenta crear el análisis	El sistema muestra el error correspondiente a los campos incorrectos

Tabla 87. Pruebas de integración – Registrar un nuevo análisis

6.6.33. CU-W-33 Modificar un análisis

Identificador	Título	Escenario	Resultado esperado
CP-W-44	Modificación de un análisis propio	Un usuario autenticado, desde los detalles de un análisis propio, pulsa sobre editar, introduce y guarda la nueva información	El sistema guarda las actualizaciones del análisis y redirige a la pantalla de detalles del análisis
CP-W-45	Modificación de un análisis propio con campos incorrectos	Un usuario autenticado, desde los detalles de un análisis propio, pulsa sobre editar, introduce e intenta guardar la nueva información	El sistema muestra el error correspondiente a los campos incorrectos

Tabla 88. Pruebas de integración – Modificar un análisis

6.6.34. CU-W-34 Eliminar un análisis

Identificador	Título	Escenario	Resultado esperado
CP-W-46	Eliminación de un análisis propio	Un usuario autenticado, desde los detalles de un análisis propio, pulsa sobre eliminar y confirma la eliminación	El sistema elimina el análisis y redirige a la vista de análisis

Tabla 89. Pruebas de integración – Eliminar un análisis

6.6.35. CU-W-35 Ver un listado de los usuarios que tienen acceso a un análisis

Identificador	Título	Escenario	Resultado esperado
CP-W-47	Visualización de los usuarios que tienen acceso a un análisis de visibilidad compartida	Un usuario autenticado accede a los detalles de un análisis de visibilidad compartida y pulsa sobre usuarios compartidos para visualizar un listado de estos	El sistema muestra el listado de usuarios que tienen acceso a dicho análisis

Tabla 90. Pruebas de integración – Ver un listado de los usuarios que tienen acceso a un análisis

6.6.36. CU-W-36 Dar acceso a un usuario a un análisis

Identificador	Título	Escenario	Resultado esperado
CP-W-48	Asignación de acceso a un usuario a un análisis de visibilidad compartida	Un usuario autenticado accede a los detalles de un análisis de visibilidad compartida, pulsa sobre usuarios compartidos y añade un usuario registrado	El sistema añade al usuario introducido en el listado de usuarios con acceso al análisis

Tabla 91. Pruebas de integración – Dar acceso a un usuario a un análisis

6.6.37. CU-W-37 Eliminar el acceso de un usuario a un análisis

Identificador	Título	Escenario	Resultado esperado
CP-W-49	Eliminación de acceso a un usuario a un análisis de visibilidad compartida	Un usuario autenticado accede a los detalles de un análisis de visibilidad compartida, pulsa sobre usuarios compartidos y elimina a un usuario del listado	El sistema elimina al usuario del listado de usuarios con acceso al análisis

Tabla 92. Pruebas de integración – Eliminar el acceso de un usuario a un análisis

6.6.38. CU-W-38 Visualizar el listado de resultados

Identificador	Título	Escenario	Resultado esperado

CP-W-50	Visualización del listado de resultados propios	Un usuario autenticado accede a la vista de resultados	El sistema muestra un listado con los resultados propios del usuario

Tabla 93. Pruebas de integración – Visualizar el listado de resultados

6.6.39. CU-W-39 Visualizar los detalles de un resultado

Identificador	Título	Escenario	Resultado esperado
CP-W-51	Visualización de los detalles de un resultado	Un usuario autenticado accede a la vista de resultados propios y pulsa sobre uno de ellos	El sistema muestra los detalles de dicho resultado

Tabla 94. Pruebas de integración – Visualizar los detalles de un resultado

6.6.40. CU-W-40 Visualizar los detalles del resultado de un programa

Identificador	Título	Escenario	Resultado esperado
CP-W-52	Visualización de los detalles del resultado de un programa	Un usuario autenticado, desde los detalles de un resultado propio, pulsa sobre el icono de resultados de un programa	El sistema muestra los detalles del resultado de dicho programa

Tabla 95. Pruebas de integración – Visualizar los detalles del resultado de un programa

6.6.41. CU-W-41 Realizar el análisis de un programa

Identificador	Título	Escenario	Resultado esperado
CP-W-53	Realización del análisis de uno o varios programas	Un usuario autenticado, desde la vista de resultados, pulsa sobre nuevo y crea un nuevo análisis seleccionando uno o varios programas y análisis	El sistema realiza el análisis, almacena la información registrando un nuevo resultado con los datos obtenidos y redirige a la vista de resultados

Tabla 96. Pruebas de integración – Realizar el análisis de un programa

6.6.42. CU-W-42 Eliminar un resultado

Identificador	Título	Escenario	Resultado esperado

CP-W-54	Eliminación de un resultado propio	Un usuario autenticado, desde los detalles de un resultado propio, pulsa sobre eliminar	El sistema elimina el resultado y redirige a la vista de resultados

Tabla 97. Pruebas de integración – Eliminar un resultado

6.6.43. CU-W-43 Eliminar el resultado de un programa

Identificador	Título	Escenario	Resultado esperado
CP-W-55	Eliminación del resultado de un programa	Un usuario autenticado, desde los detalles de un resultado propio de un programa, pulsa sobre eliminar	El sistema elimina el resultado del programa y redirige a la vista de resultados

Tabla 98. Pruebas de integración – Eliminar el resultado de un programa

6.6.44. CU-W-44 Ver estadísticas generales

Identificador	Título	Escenario	Resultado esperado
CP-W-56	Visualización de las estadísticas generales	Un usuario administrador accede a la vista de estadísticas generales	El sistema muestra las estadísticas generales

Tabla 99. Pruebas de integración – Ver estadísticas generales

6.6.45. CU-W-45 Ver estadísticas de los programas

Identificador	Título	Escenario	Resultado esperado
CP-W-57	Visualización de las estadísticas de programas	Un usuario administrador, desde la vista de estadísticas generales, pulsa sobre programas	El sistema muestra las estadísticas asociadas a los programas registrados

Tabla 100. Pruebas de integración – Ver estadísticas de los programas

6.6.46. CU-W-46 Ver estadísticas de los análisis

Identificador	Título	Escenario	Resultado esperado
CP-W-58	Visualización de las estadísticas de análisis	Un usuario administrador, desde la vista de estadísticas generales, pulsa sobre análisis	El sistema muestra las estadísticas asociadas a los análisis registrados

Tabla 101. Pruebas de integración – Ver estadísticas de los análisis

6.6.47. CU-W-47 Ver estadísticas de los resultados

Identificador	Título	Escenario	Resultado esperado
CP-W-59	Visualización de las estadísticas de resultados	Un usuario administrador, desde la vista de estadísticas generales, pulsa sobre resultados	El sistema muestra las estadísticas asociadas a los resultados registrados

Tabla 102. Pruebas de integración – Ver estadísticas de los resultados

6.6.48. CU-W-48 Ver estadísticas de programas-análisis

Identificador	Título	Escenario	Resultado esperado
CP-W-60	Visualización de las estadísticas de programas-análisis	Un usuario administrador, desde la vista de estadísticas generales, pulsa sobre programas-análisis	El sistema muestra las estadísticas asociadas a los programas-análisis

Tabla 103. Pruebas de integración – Ver estadísticas de programas-análisis

6.7. Pruebas de rendimiento

6.7.1. Pruebas de Carga

Las pruebas de carga permiten simular el acceso de varios usuarios concurrentes al sistema y eso es lo que se probó mediante la herramienta de JMeter [21].

Como se puede ver en la configuración del JMeter se ha establecido en primer lugar un plan de pruebas sobre la parte más relevante del sistema. Para ello se ha configurado una serie de peticiones entre las que se encuentran el login y las operaciones de acceso al listado de programas, a los detalles de un solo programa, listado de todos los análisis, detalles de un solo análisis, listado de todos los resultados y detalle de un solo resultado. Además, se ha determinado un número de hilos (usuarios) de 20, ramp-up time, que es el tiempo que se tarda en crear cada usuario de 0,1 y una duración de 60 segundos.

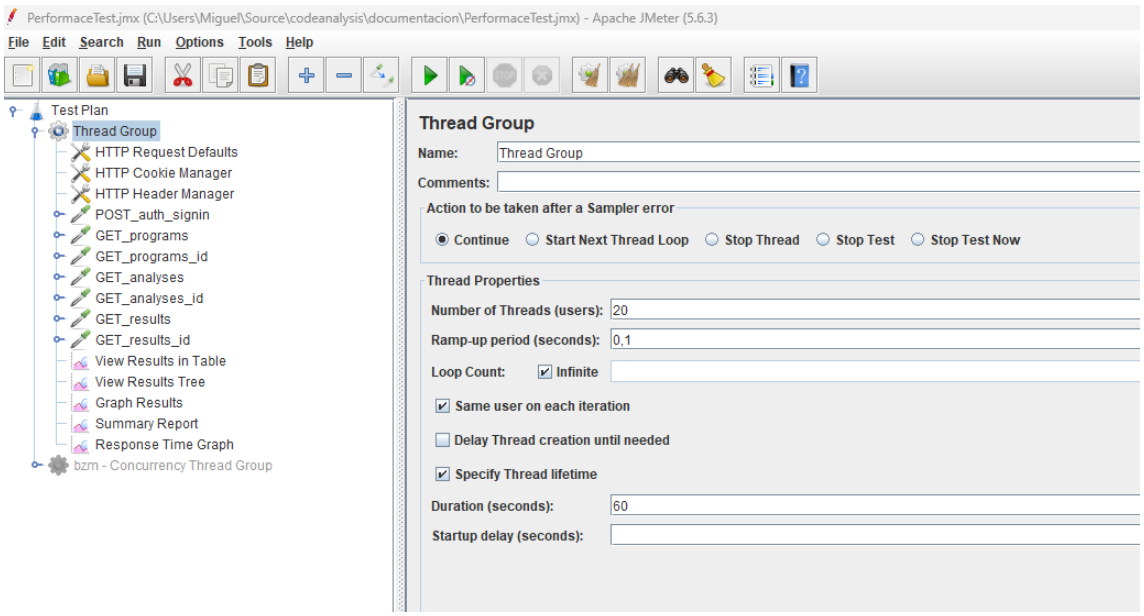


Ilustración 37. Configuración de pruebas de carga usando JMeter

Con todo lo anterior lo que simulamos es la realización de todas estas peticiones de forma concurrente por 20 usuarios al mismo tiempo durante 60 segundos, entrando al sistema con una diferencia de 0,1 segundos. Tras esto obtenemos una gráfica que analizaremos a continuación.

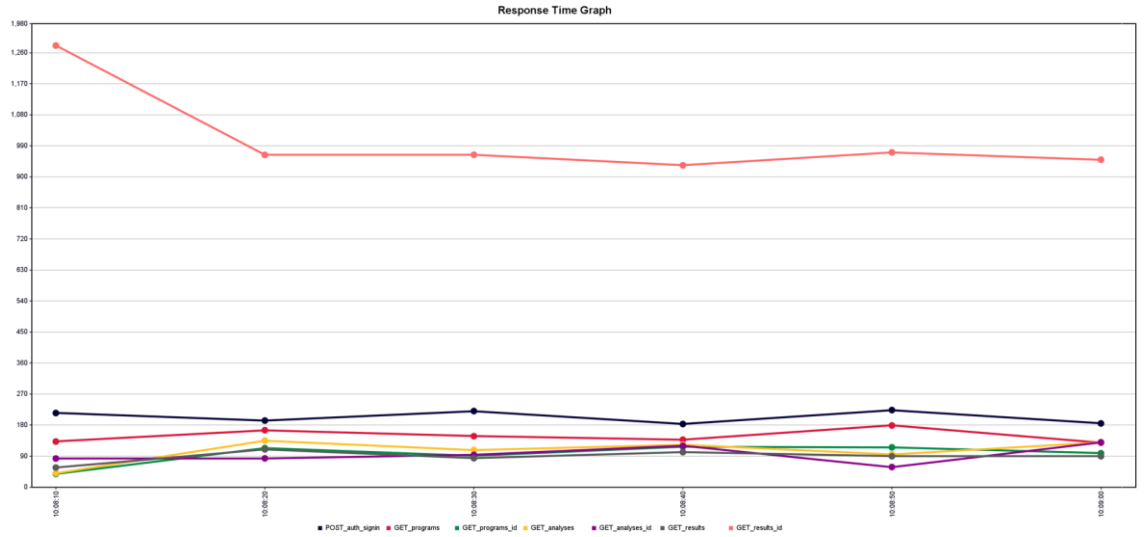


Ilustración 38. Resultados de las pruebas de carga

Como se puede ver la operación más costosa en tiempo es la obtención de los detalles de un resultado concreto ya que se necesita acceder a base de datos, combinar todos los resultados, todo el proceso de autorización y autenticación en el sistema. Una vez estabilizado el sistema el tiempo medio es inferior a 900 milisegundos. La segunda operación que más tiempo requiere es el inicio de sesión, ya que se necesita acceder a base de datos, comprobar las credenciales y generar el token, es decir, todo el proceso de autorización y autenticación en el sistema. El tiempo medio es algo superior a los 100 milisegundos. Finalmente, el resto de las operaciones vemos que no hay mucha diferencia entre ellas, con un tiempo medio inferior a los 100 milisegundos.

En general las pruebas de carga han tenido muy buenos resultados puesto que todas las operaciones se realizan en menos de 1 segundo y con ello demostraríamos que hemos conseguido hacer realidad el **RNF8** donde se requería que la funcionalidad del sistema y transacción de negocio debe responder al usuario en menos de 5 segundos.

6.7.2. Pruebas de Estrés

Este tipo de pruebas permiten encontrar la carga máxima que es capaz de soportar el sistema, tras la cual comienza a responder de forma lenta y producir errores. Dicho instante se llama punto de ruptura y es el que se trata de encontrar con estas pruebas.

Al igual que en el caso de las pruebas de carga se ha usado la herramienta de JMeter y se le ha configurado el mismo plan de pruebas que en el caso anterior lo único que han variado son las variables puesto que en este tipo de pruebas se añade alguna más: *target concurrency* es el número de usuarios, *ramp-up steps count* divide el número de usuarios en X bloques, *ramp-up time* especifica cada cuanto tiempo entra un nuevo bloque de usuarios y *hold target* es la duración del test.

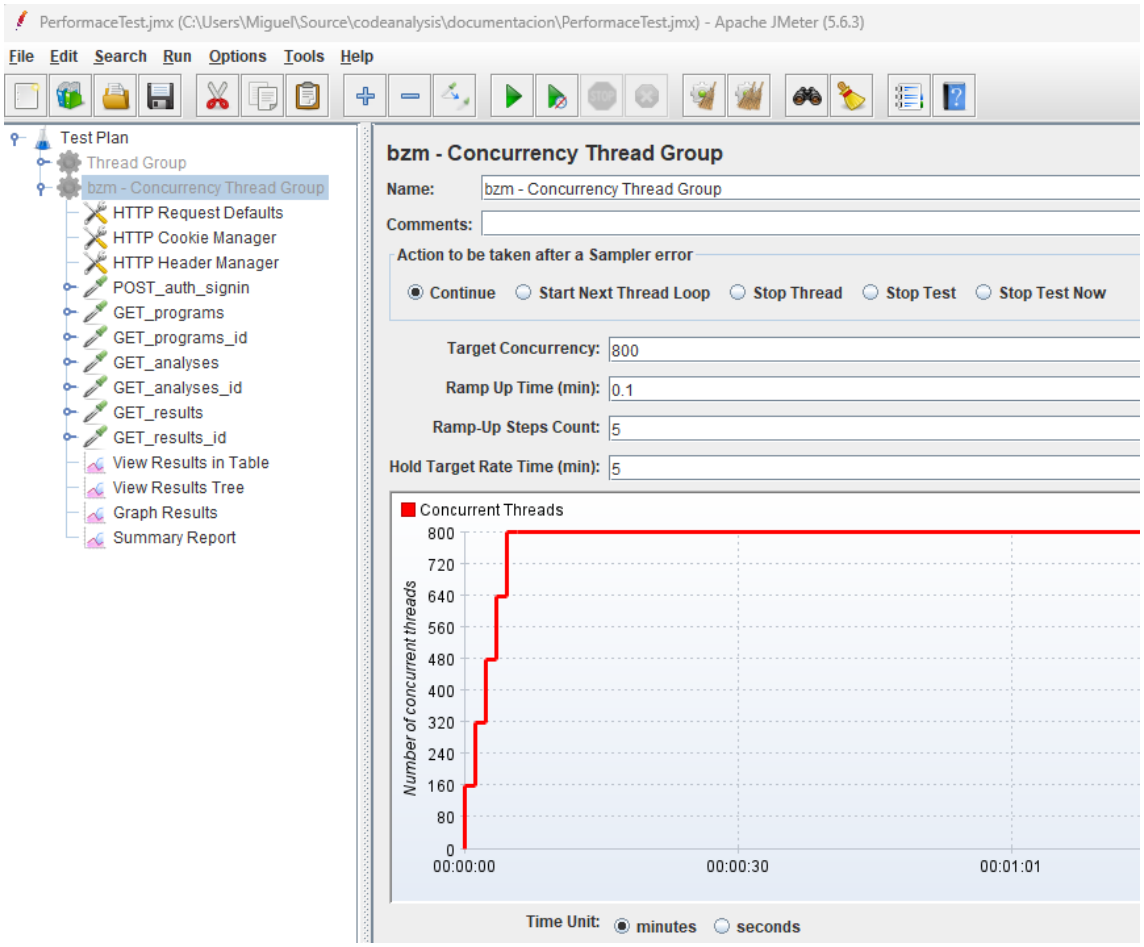


Ilustración 39. Configuración de pruebas de estrés usando JMeter

La principal diferencia entre las pruebas de carga y las de estrés es que en las de carga los usuarios llegan de 1 en 1 cada x segundos mientras que en las de estrés los usuarios llegan en bloques cada x segundos. Por tanto, los nuevos usuarios entran todos a la vez de forma concurrente. Para la realización de esta prueba se probó con usuarios entre 100 y 800 y se obtuvieron los siguientes resultados representados en la gráfica de abajo.

Usuarios Concurrentes	Throughput (op/segundo)	Error %
100	75,6/s	0 %
200	74,3/s	0 %
300	72,5/s	0 %
400	68,4/s	0 %
500	66,9/s	0 %
600	57,6/s	0 %
700	53,9/s	0 %
800	50,1/s	0 %

Tabla 104. Resultado de las pruebas de estrés

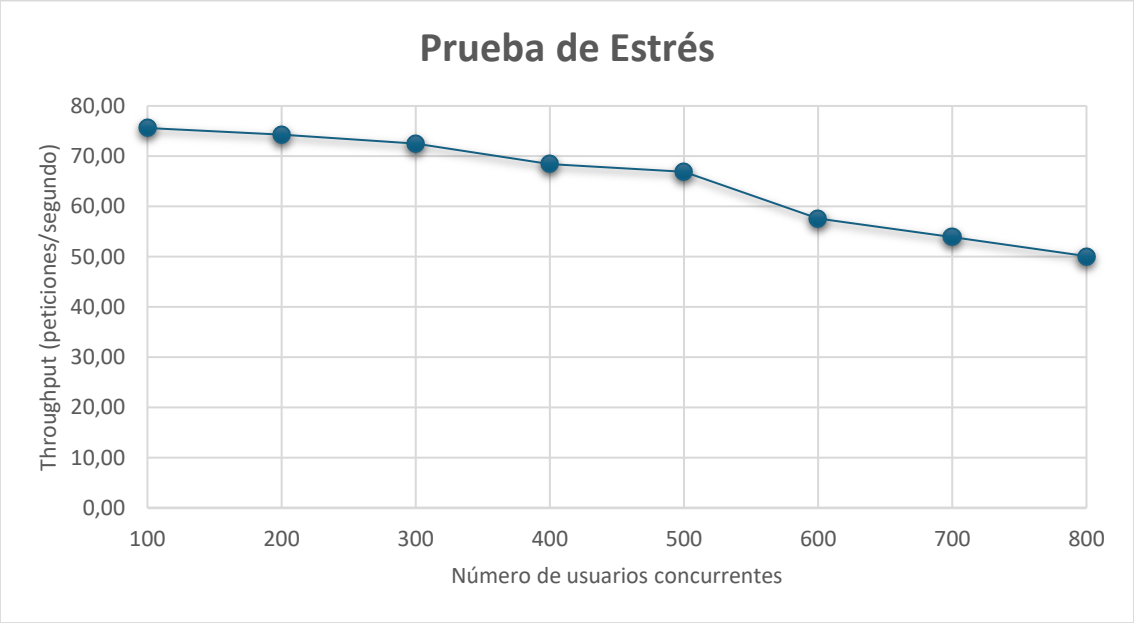


Ilustración 40. Resultados de las pruebas de estrés

Esta gráfica muestra en el eje X el número de usuario introducidos a la prueba y en el eje Y el *throughput*, es decir, la cantidad de peticiones que una aplicación de software puede procesar y responder dentro de un tiempo específico. Además, en la tabla se puede ver el % de errores que se producen con esas cargas de usuarios. Tal y como se aprecia en la tabla, aún con 800 usuarios concurrentes no se produjeron errores ni se aprecia una degradación significativa del sistema. En esta última prueba, el sistema fue capaz de procesar de forma correcta 16.550 operaciones en 5 minutos.

En general las pruebas de estrés han tenido muy buenos resultados puesto que el sistema es capaz de atender hasta 800 usuarios concurrentes sin ningún error, por lo que con ello demostraríamos que hemos conseguido hacer realidad el **RNF9** donde se requería que el sistema fuese capaz de operar adecuadamente con hasta 100 usuarios con sesiones concurrentes.

6.8. Pruebas de usabilidad

Para evaluar la usabilidad de la aplicación, se realizarán pruebas con un grupo de 10 usuarios que realizarán una serie de tareas específicas en la aplicación web. Estas tareas

están diseñadas para cubrir las funcionalidades principales del sistema y asegurar que los usuarios puedan completar las acciones requeridas sin dificultad.

Las pruebas de usabilidad consistirán en los siguientes pasos:

1. Registrarse en la aplicación.
2. Iniciar sesión con las credenciales del registro.
3. Acceder al perfil y modificar algún dato.
4. Acceder al listado de programas y visualizar los detalles de alguno.
5. Subir un nuevo programa, editar algún dato posteriormente y eliminarlo.
6. Acceder al listado de análisis y visualizar los detalles de alguno.
7. Crear un nuevo análisis de visibilidad compartida y, posteriormente, dar acceso a dicho programa a algún usuario existente.
8. Analizar varios programas. Para ello, crea un nuevo resultado eligiendo varios programas para analizar.
9. Acceder a los detalles del resultado anterior y eliminar el resultado de uno de los programas analizados.
10. Cerrar sesión.

Una vez completadas las tareas descritas anteriormente, se solicita a los usuarios que evalúen cada paso en una escala del 0 al 10, donde 10 indica que la tarea fue muy sencilla de realizar y 0 que fue imposible de completar. Los resultados de estas evaluaciones se detallan a continuación.

Número prueba de usabilidad	Nota media obtenida de los usuarios
1	9,8
2	10
3	9,7
4	9,2
5	8,4
6	9,2
7	7,6
8	9,3
9	6,6
10	9,8

Tabla 105. Resultados de las pruebas de usabilidad

Como se puede observar en los resultados anteriores, el uso de la aplicación no presentó una complejidad excesiva, aunque se identificaron dos casuísticas que podrían mejorarse. En la séptima prueba de usabilidad, algunos usuarios encontraron compleja la funcionalidad de acceso compartido a programas, debido a la falta de iconos y elementos

visuales claros. Para solucionar esto, se añadió un icono específico que hizo la función más intuitiva. En la novena prueba, los usuarios reportaron dificultad para comprender que el icono de resultados era un enlace interactivo. Para mejorar esta experiencia, se cambió el color del icono a azul, indicando de manera más clara que se trataba de un enlace. Tras recibir este feedback, se realizaron los ajustes necesarios para mejorar la usabilidad de la aplicación web.

En conclusión, se puede afirmar que la aplicación cumple con el requisito no funcional de ser usable. Ningún usuario manifestó la imposibilidad de completar las tareas planteadas, lo que indica que la aplicación es accesible y fácil de usar incluso sin experiencia previa.

Capítulo 7. Manuales del sistema

7.1. Manual de usuario

7.1.1. Instalación y Despliegue de la API

La API web está basada en Maven, por lo que el proceso de instalación y despliegue se simplifica considerablemente utilizando las herramientas estándar proporcionadas por Maven y Tomcat.

1. Generación del WAR con Maven:

   Para compilar y empaquetar la aplicación web en formato WAR, se utiliza Maven. Asegúrate de tener Maven instalado en el entorno de desarrollo.

   Desde la línea de comandos, dentro del directorio raíz del proyecto donde se encuentra el archivo pom.xml, ejecuta el siguiente comando:

   ```
   mvn clean compile package
   ```

   Este comando realiza varias tareas:

- `clean`: Limpia el directorio de salida para eliminar cualquier compilación previa.
- `compile`: Compila el código fuente de la aplicación.
- `package`: Empaqueta el código compilado en un archivo WAR (Web ARchive) listo para ser desplegado en un contenedor web como Tomcat.

2. Despliegue en Tomcat:

- Copiar el WAR a la carpeta de despliegue de Tomcat:

  Copia el archivo war generado en la carpeta de despliegue de Tomcat. Por lo general, esta carpeta se encuentra en <ruta-a-tomcat>/webapps/.

- Iniciar Tomcat.
- Verificar el despliegue:

  Una vez que Tomcat haya iniciado correctamente, la aplicación debería desplegarse automáticamente si todo ha sido configurado correctamente.

7.1.2. Instalación y Despliegue de la Web

El proceso para compilar y crear una aplicación frontend en ASP.NET Core es el siguiente

1. Compilación del Proyecto:

   Asegúrate de tener .NET Core SDK instalado en tu entorno de desarrollo. Desde la línea de comandos, dentro del directorio raíz del proyecto donde se encuentra el archivo de proyecto (.csproj), ejecuta el siguiente comando:

   ```
   dotnet build
   ```

Este comando compila el proyecto y verifica si hay errores de compilación en el código fuente.

2. Creación del Paquete de Publicación:

   Para crear el paquete de publicación que luego podrás desplegar en un servidor web como Kestrel, ejecuta el siguiente comando:

   ```
   dotnet publish -c Release -o <ruta-de-salida>
   ```

- `-c Release`: Especifica que se debe construir en modo Release, optimizando el código para producción.
- `-o <ruta-de-salida>`: Especifica la ruta de salida donde se generará el paquete de publicación.

3. Despliegue en Kestrel:

   Kestrel es el servidor web integrado por defecto en ASP.NET Core. Para ejecutar la aplicación directamente usando Kestrel, navega hasta la carpeta de publicación y ejecuta el siguiente comando:

   ```
   dotnet nombre-del-proyecto.dll
   ```

4. Verificar el despliegue:

   Una vez que Kestrel se haya iniciado correctamente, la aplicación debería desplegarse automáticamente si todo ha sido configurado correctamente.

7.1.3. Acceder a ProgQuery

En primer lugar, nos dirigimos a la aplicación desde nuestro navegador web por medio del enlace https://progquery.uniovi.es/.

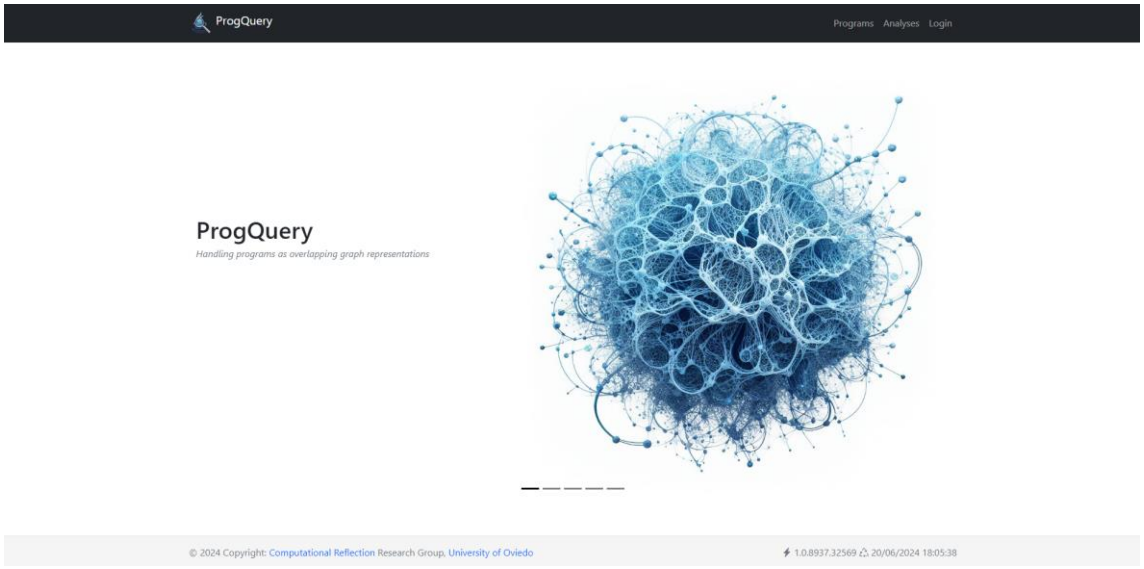


Ilustración 41. Pantalla principal

7.1.4. Registro e inicio de sesión

Para iniciar sesión y/o registrarse, pulsamos en *Login*.

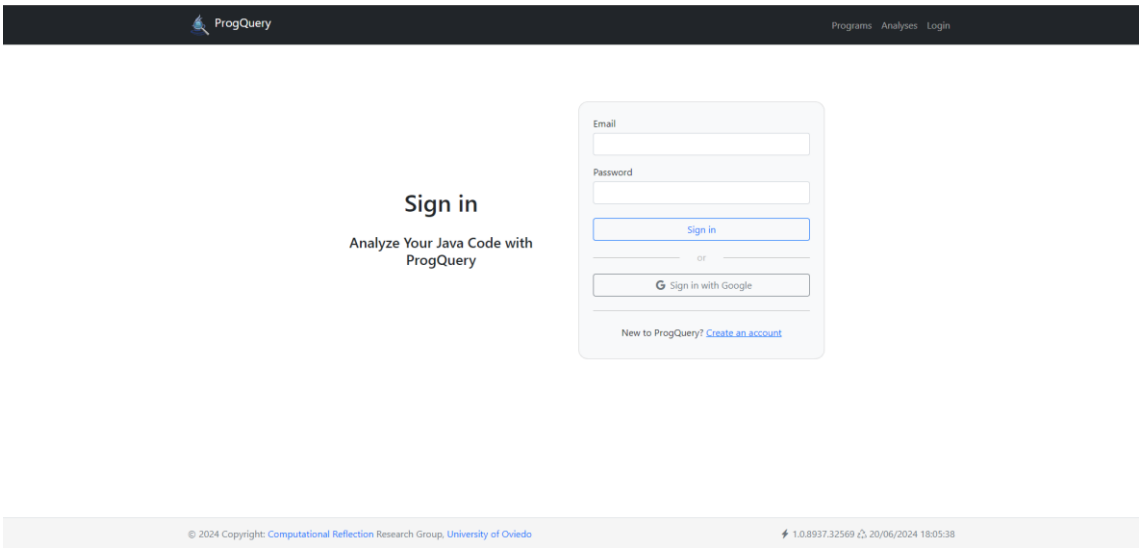


Ilustración 42. Pantalla de inicio de sesión

Pulsamos en *Create an account*, e introducimos los datos del usuario que deseamos registrar. Posteriormente, pulsamos en *Create*. También podemos registrarnos con Google.

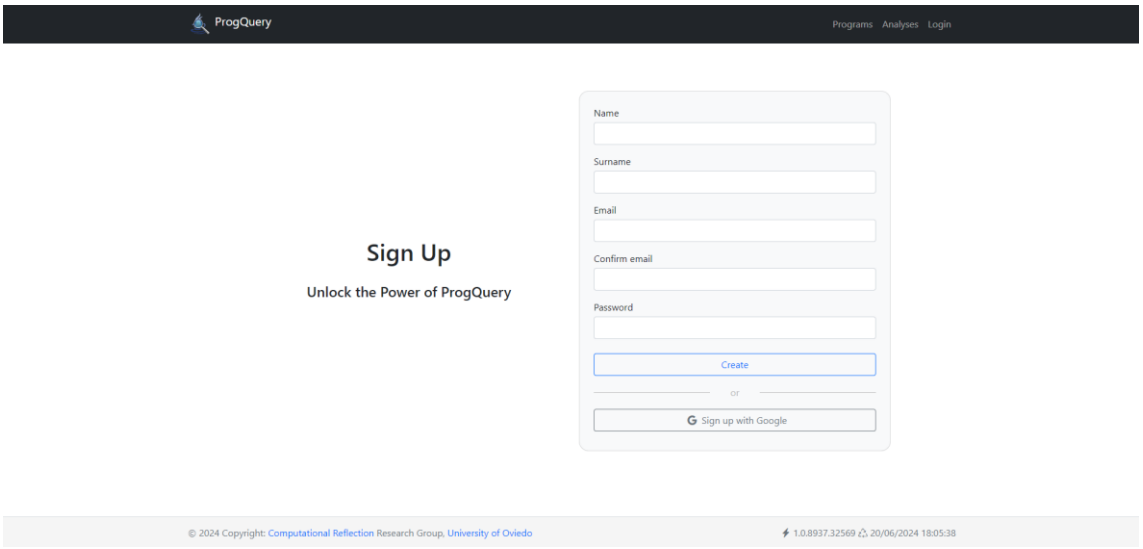


Ilustración 43. Pantalla de registro

Si el usuario se registra con éxito, se confirma por la aplicación.

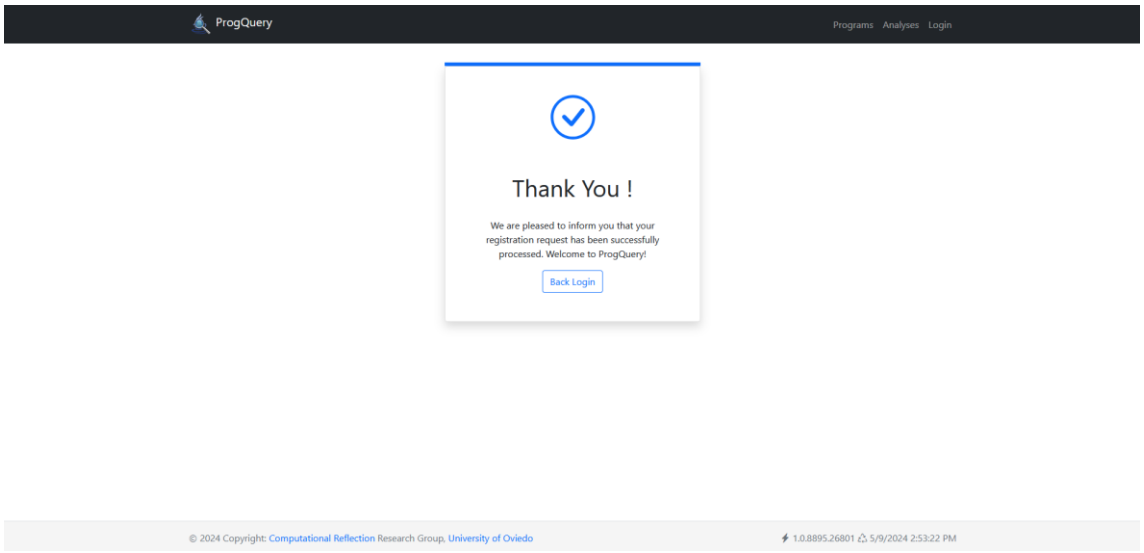


Ilustración 44. Pantalla de confirmación de registro

Volvemos a la página de inicio de sesión e introducimos las credenciales registradas.

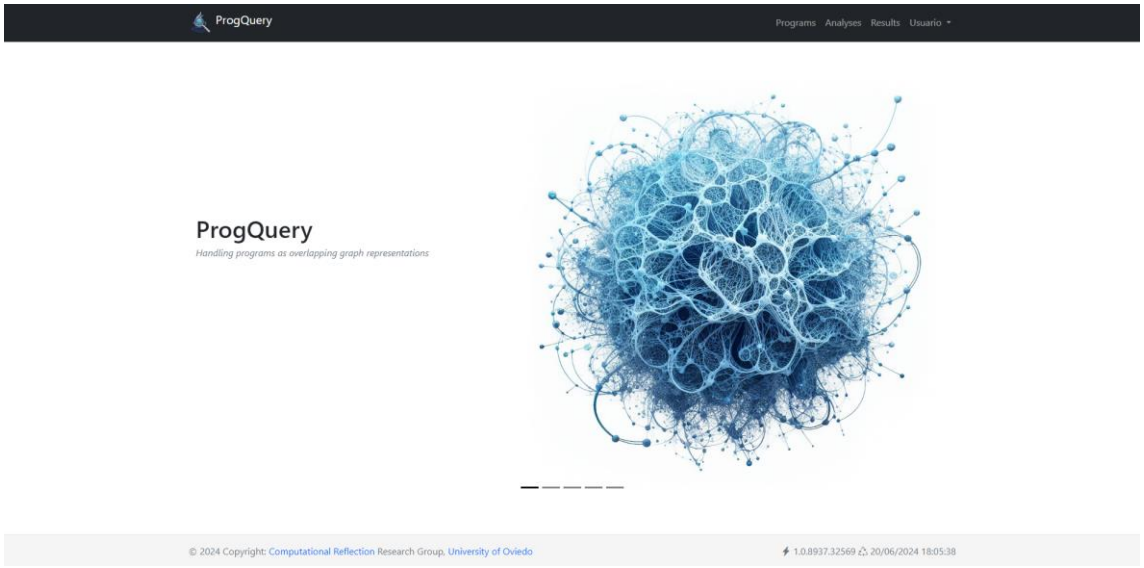


Ilustración 45. Pantalla principal con la sesión iniciada

7.1.5. Gestión de programas

A continuación, accederemos al listado de programas.

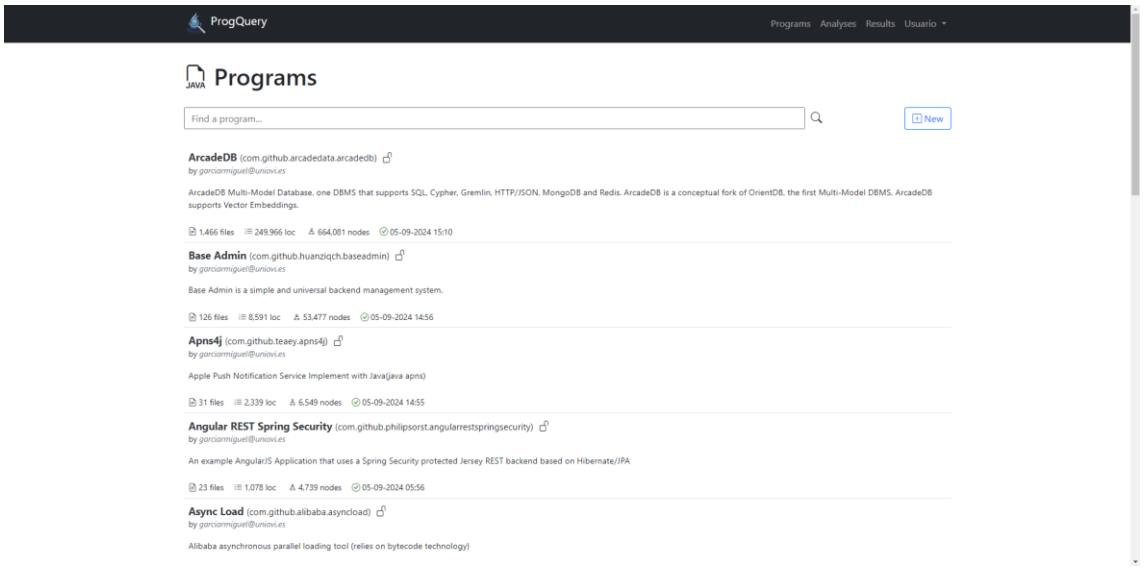


Ilustración 46. Pantalla del listado de programas

Pulsamos en *New* para agregar un nuevo programa. Rellenamos los datos correspondientes y pulsamos en *Insert program*.

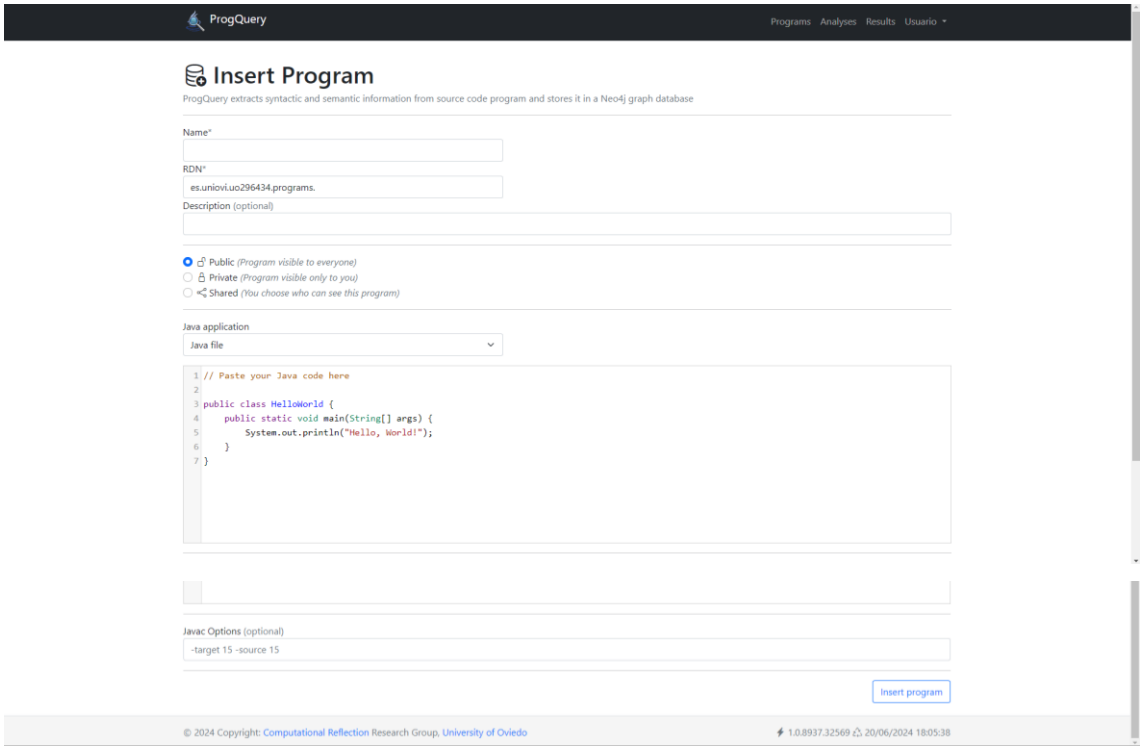


Ilustración 47. Pantalla de inserción de un programa

Si el programa se registra con éxito, se confirma por la aplicación.

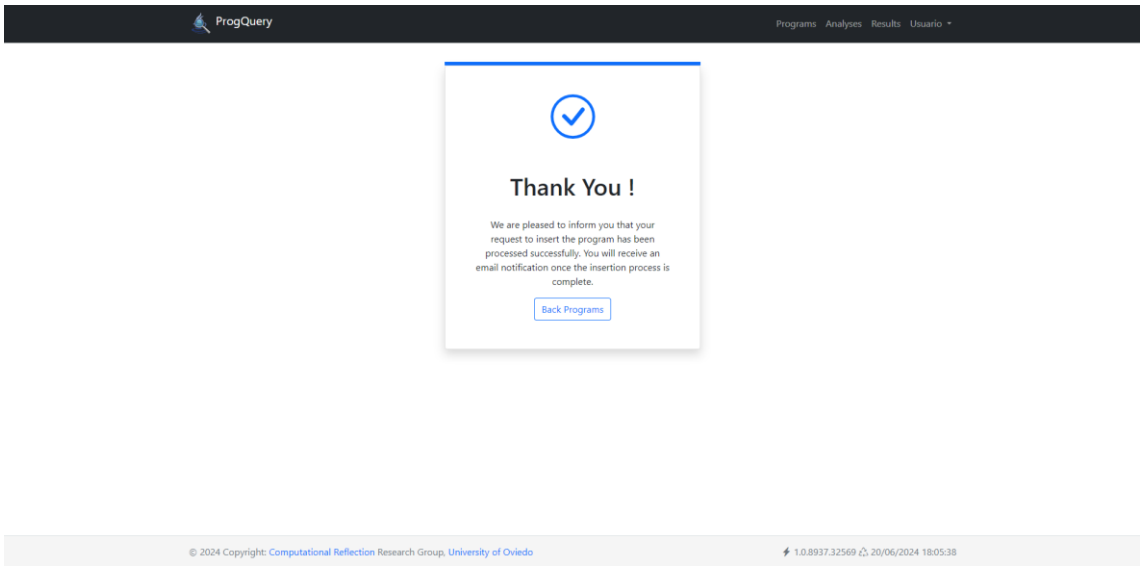


Ilustración 48. Pantalla de confirmación de inserción de un programa

Al pulsar en *Back Programs*, nos redirige a nuestros programas.

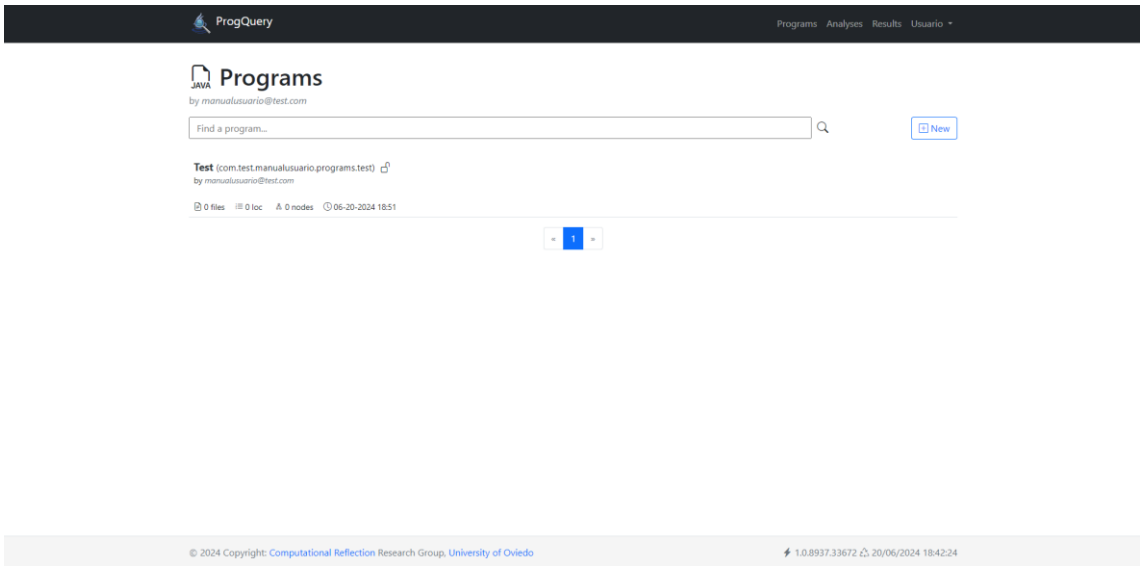


Ilustración 49. Pantalla del listado de programas propios

Pulsamos sobre el nombre del programa para acceder a los detalles de dicho programa.

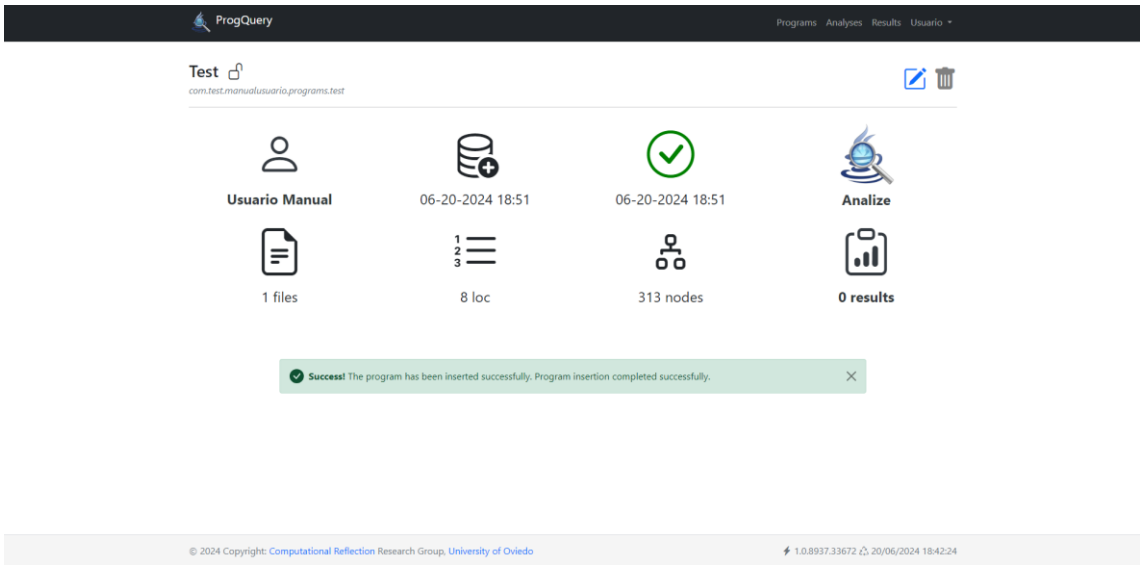


Ilustración 50. Pantalla de detalles de un programa

Para editar el programa, pulsamos sobre el icono de editar y al finalizar la edición pulsamos en *Save program* para guardar los nuevos datos.

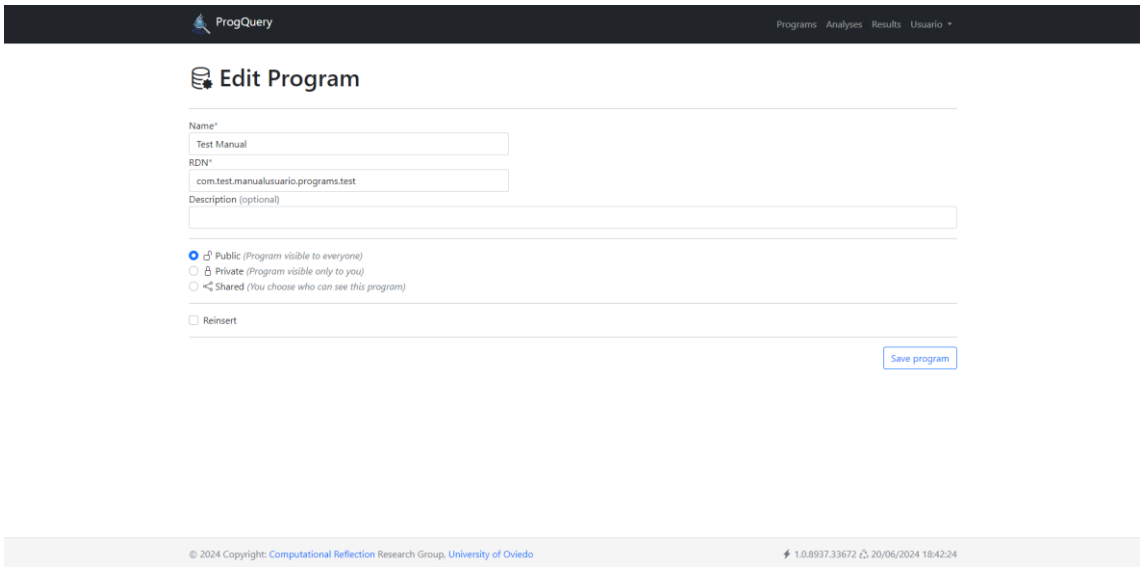


Ilustración 51. Pantalla de modificación de un programa

Podemos observar que se ha realizado correctamente la modificación.

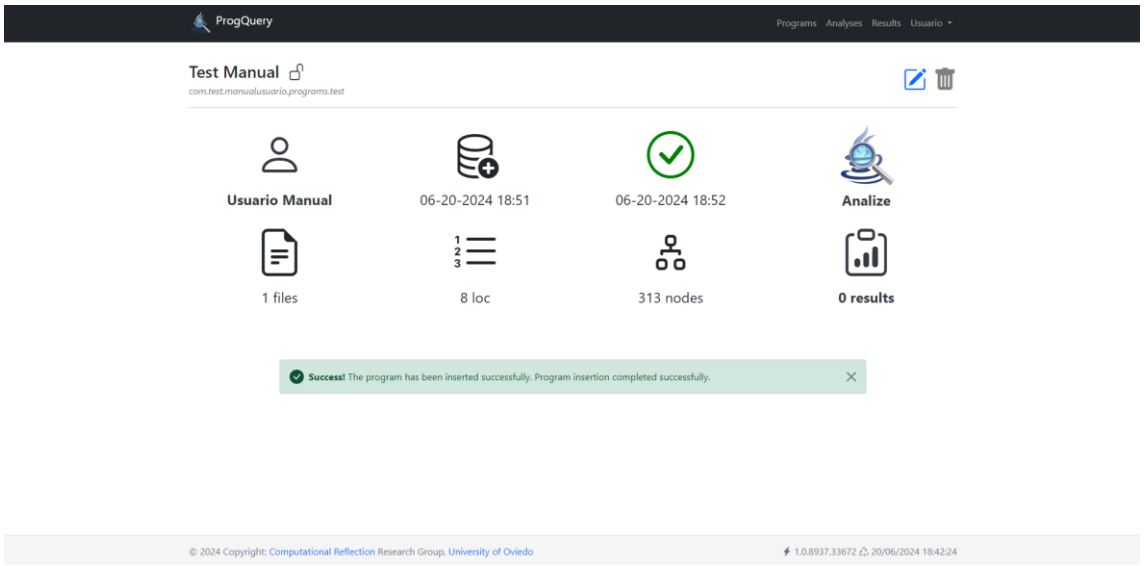


Ilustración 52. Pantalla de detalles de un programa modificado

A continuación, para borrar el programa, pulsaremos el icono de borrado, y nos saldrá una confirmación de borrado, para asegurarnos de que realmente queremos borrarlo o no. En este caso, pulsaremos sobre *Cancel*.

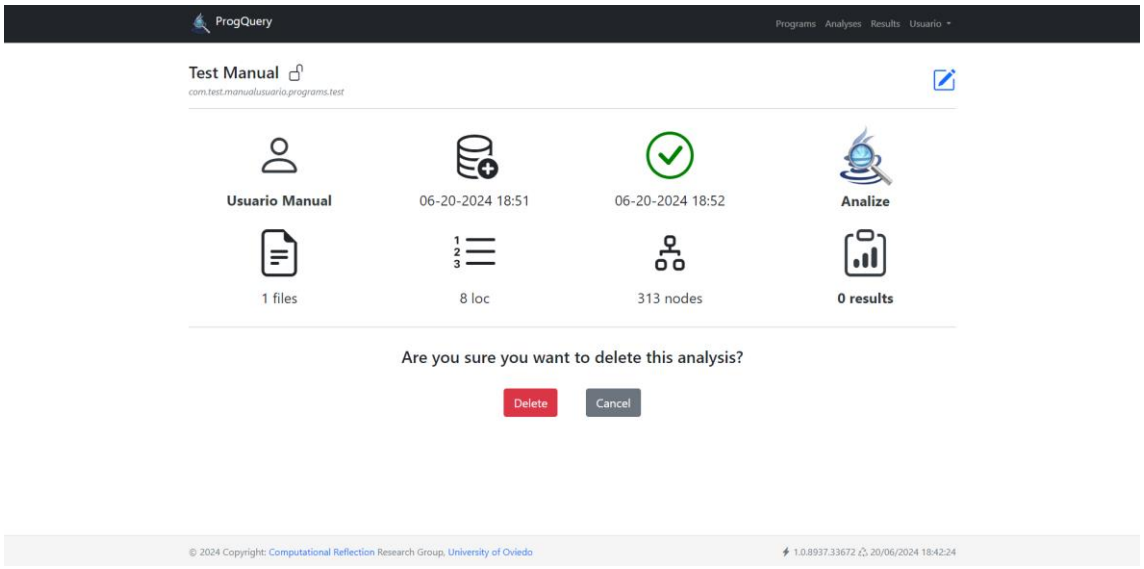


Ilustración 53. Pantalla de confirmación de eliminación de un programa

Además, desde los detalles del programa podemos acceder a los resultados asociados a los análisis realizados a dicho programa pulsando sobre el icono de resultados. En este caso aún no tenemos ningún resultado asociado al programa.

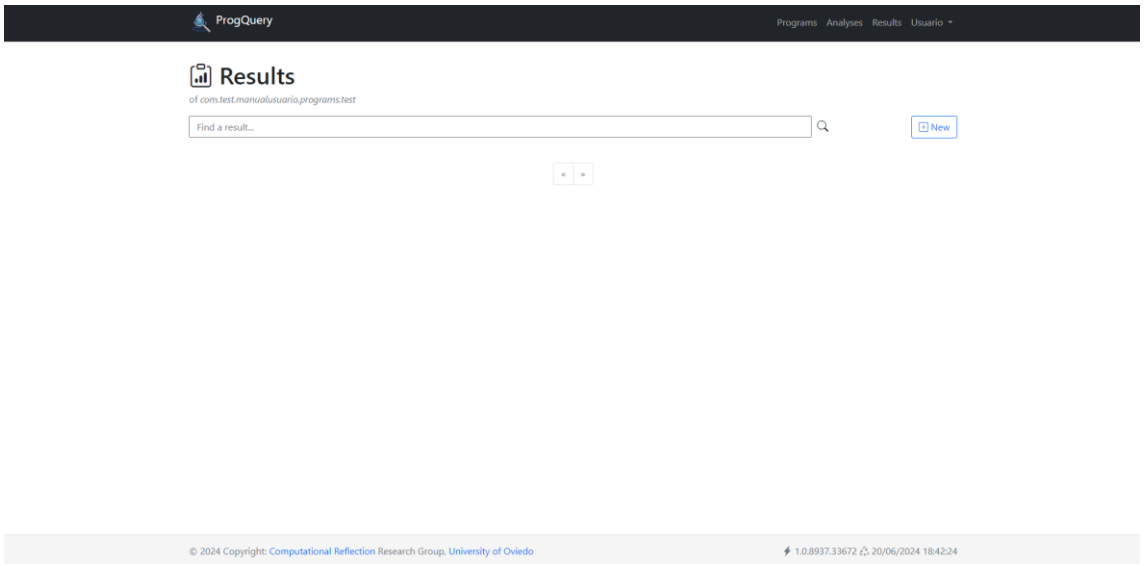


Ilustración 54. Pantalla del listado de resultados asociados a un programa

7.1.6. Gestión de análisis

Para continuar, accederemos al listado de análisis.

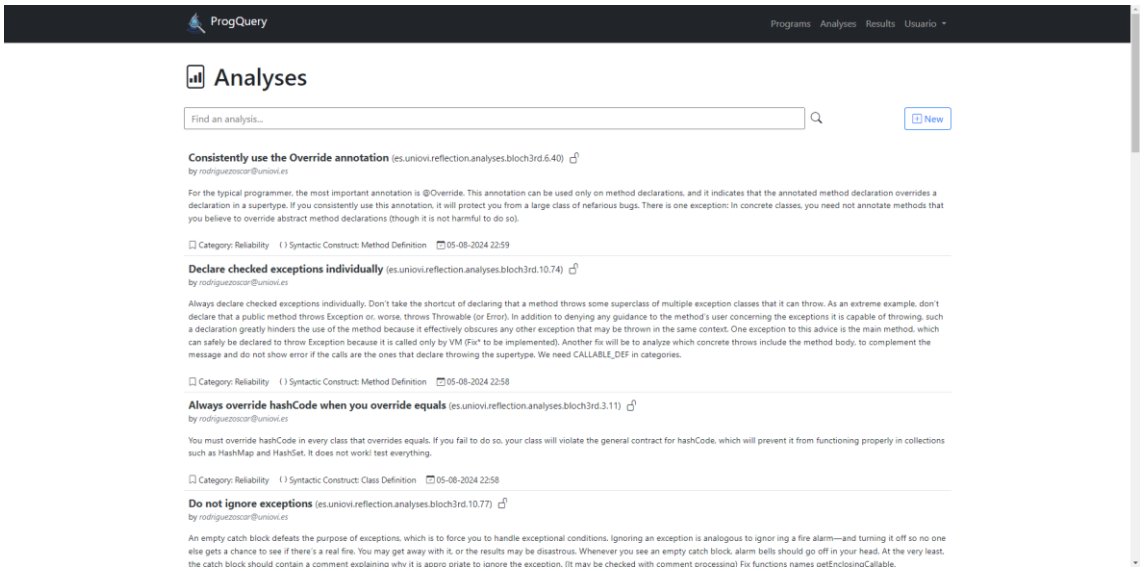


Ilustración 55. Pantalla del listado de análisis

Pulsamos en *New* para agregar un nuevo análisis. Rellenamos los datos correspondientes y pulsamos en *Create analysis*.

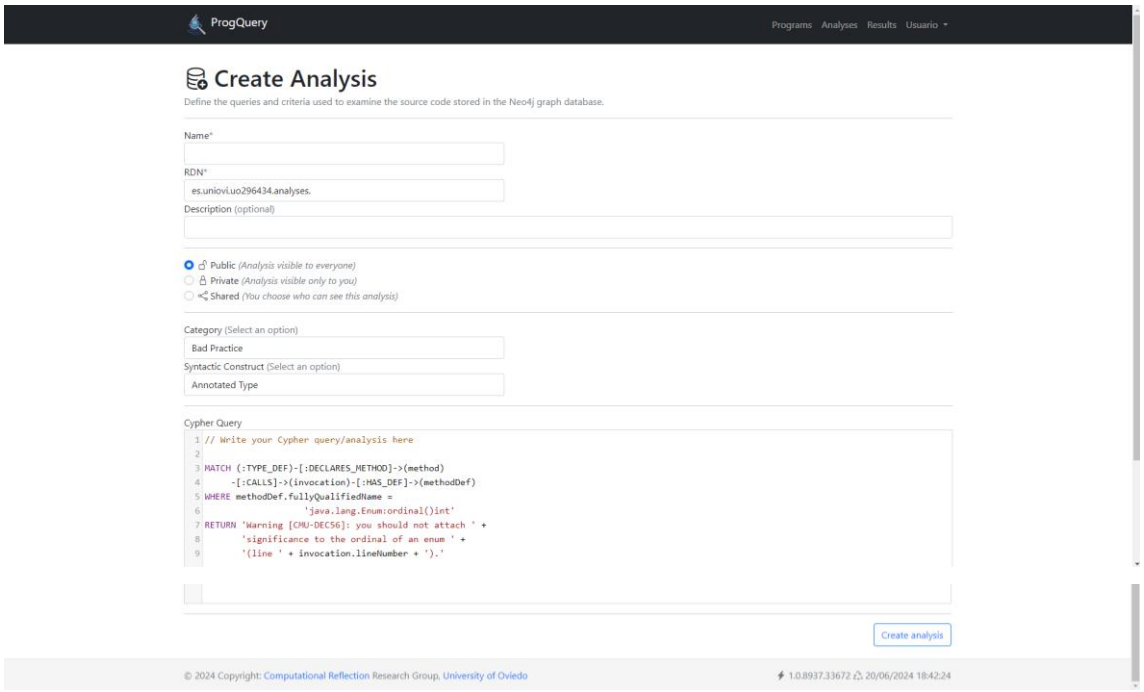


Ilustración 56. Pantalla de creación de un análisis

Si el análisis se registra con éxito, se confirma por la aplicación.

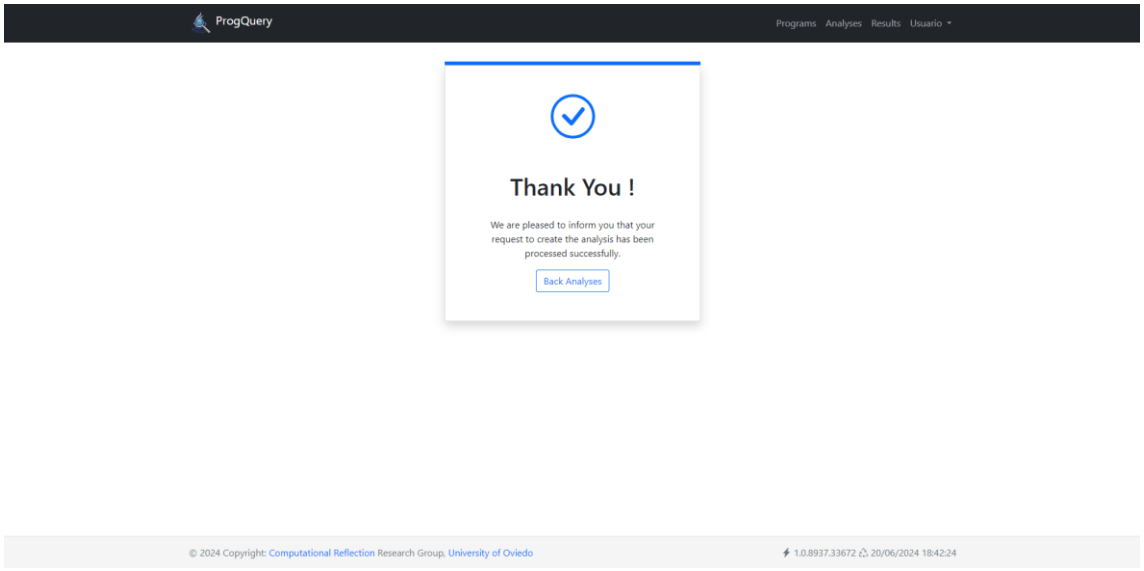


Ilustración 57. Pantalla de confirmación de creación de un análisis

Al pulsar en *Back Analyses*, nos redirige a nuestros análisis.

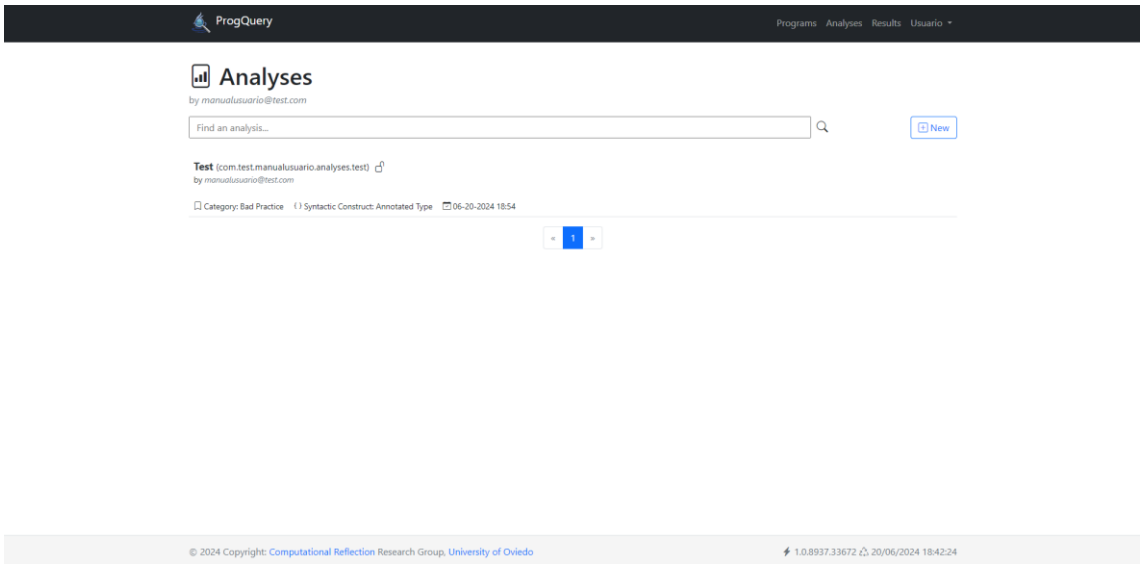


Ilustración 58. Pantalla del listado de análisis propios

Pulsamos sobre el nombre del análisis para acceder a los detalles de dicho análisis.

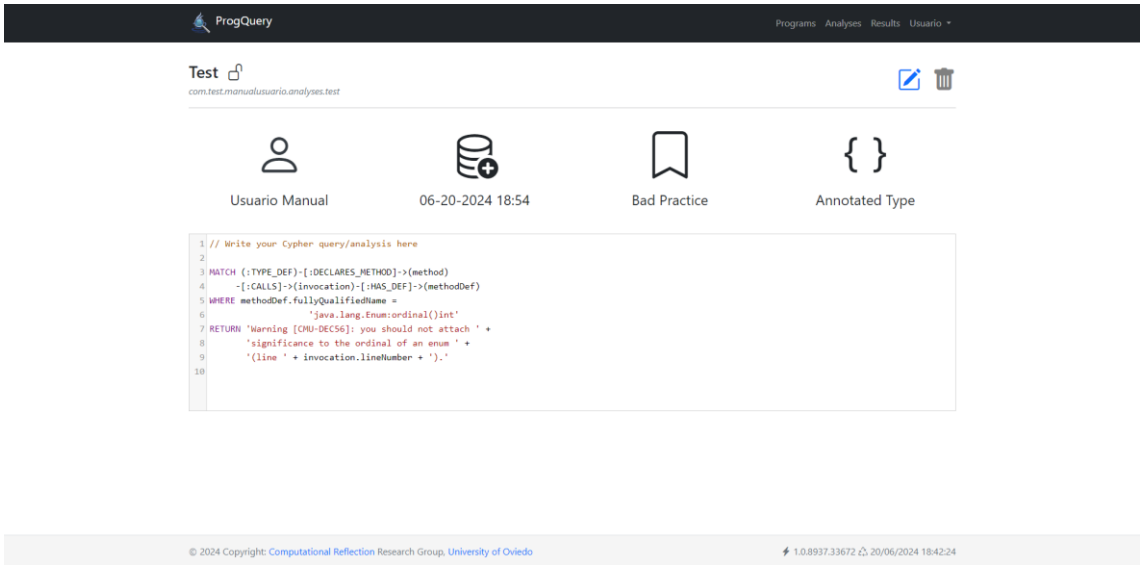


Ilustración 59. Pantalla de detalles de un análisis

Para editar el análisis, pulsamos sobre el icono de editar y al finalizar la edicción pulsamos en *Save analysis* para guardar los nuevos datos.

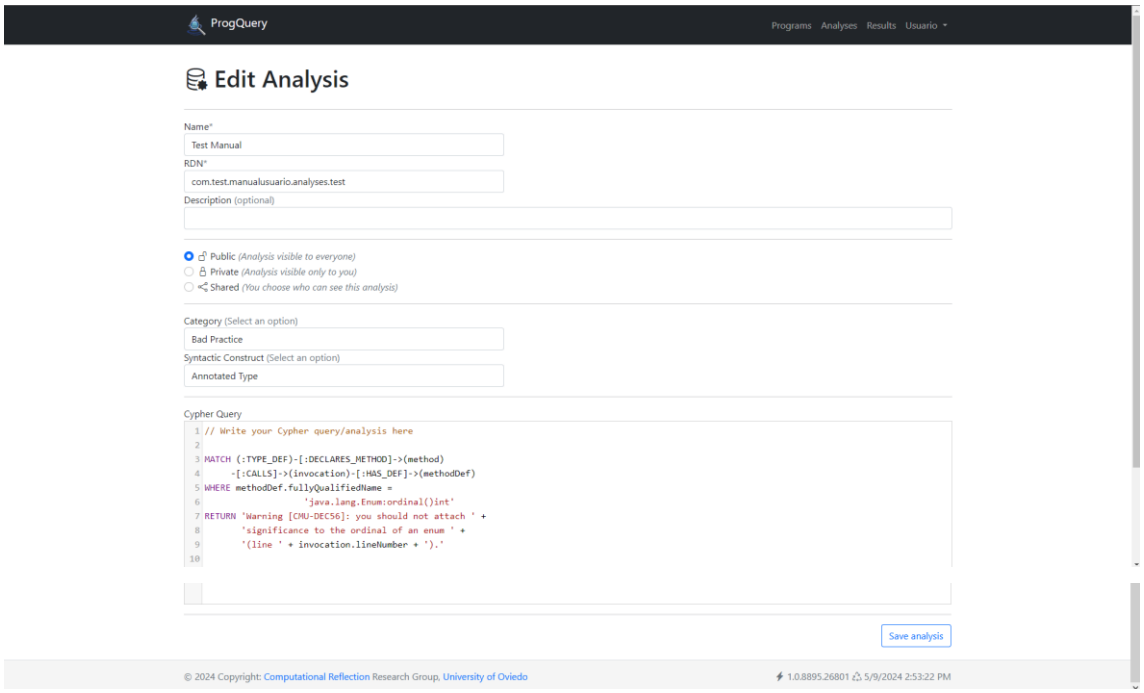


Ilustración 60. Pantalla de modificación de un análisis

Podemos observar que se ha realizado correctamente la modificación.

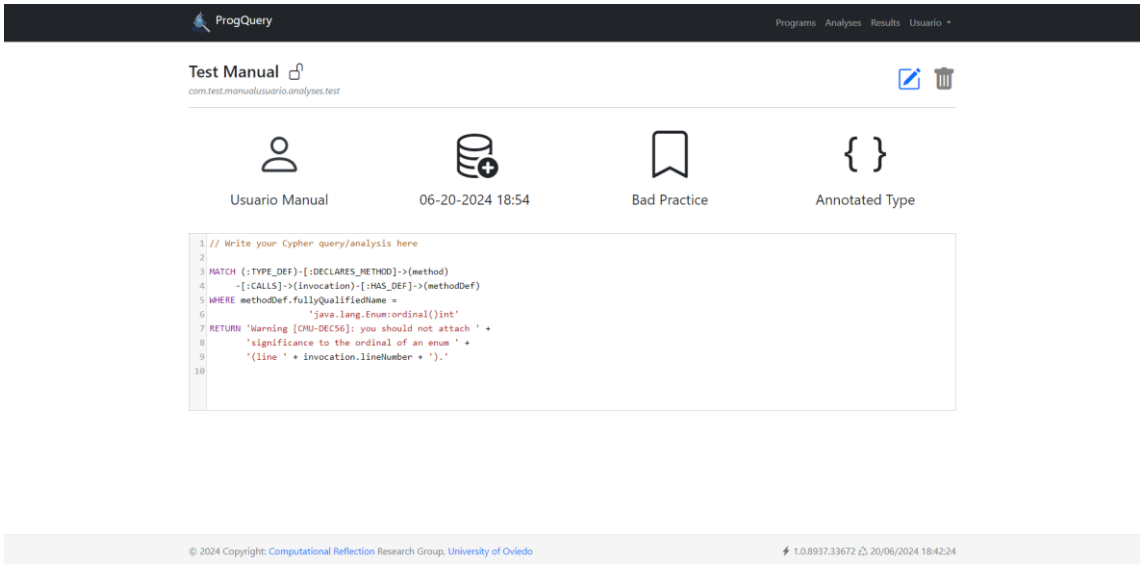


Ilustración 61. Pantalla de detalles de un análisis modificado

A continuación, para borrar el análisis, pulsaremos el icono de borrado, y nos saldrá una confirmación de borrado, para asegurarnos de que realmente queremos borrarlo o no. En este caso, pulsaremos sobre *Cancel*.

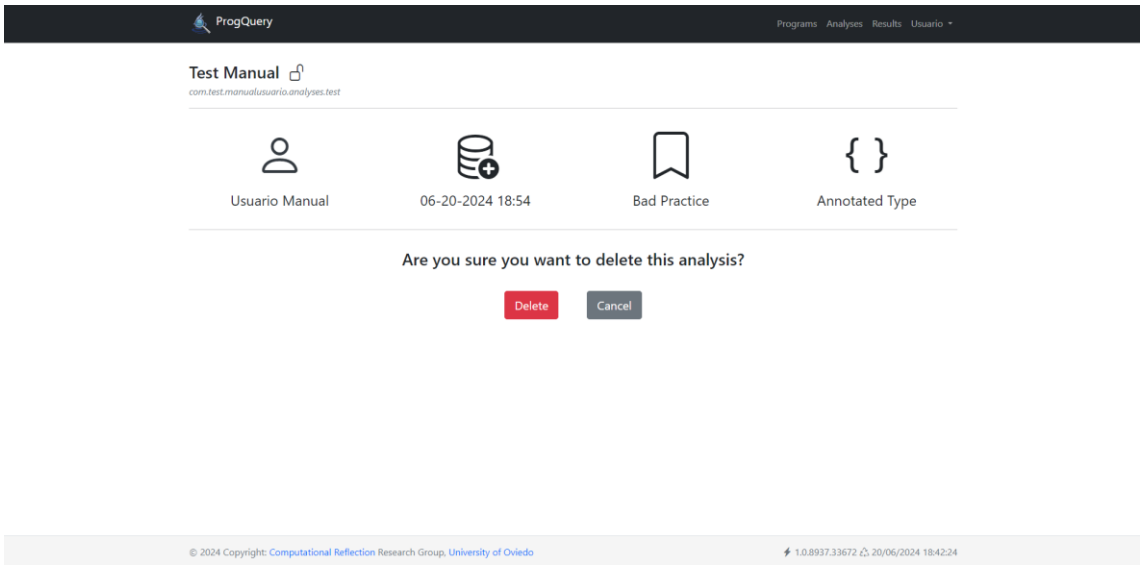


Ilustración 62. Pantalla de confirmación de eliminación de un análisis

7.1.7. Gestión de resultados

Seguidamente, accederemos al listado de nuestros resultados.

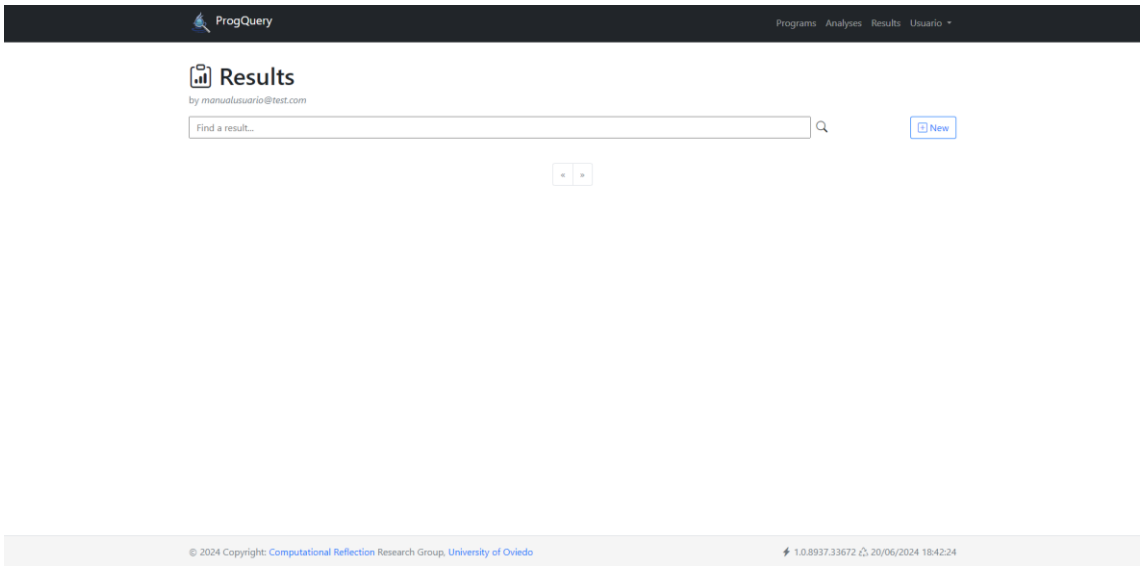


Ilustración 63. Pantalla del listado de resultados propios

Pulsamos en *New* para agregar un nuevo resultado. Rellenamos los datos correspondientes, eligiendo entre uno o varios programas y análisis, y pulsamos en *Analyze*.

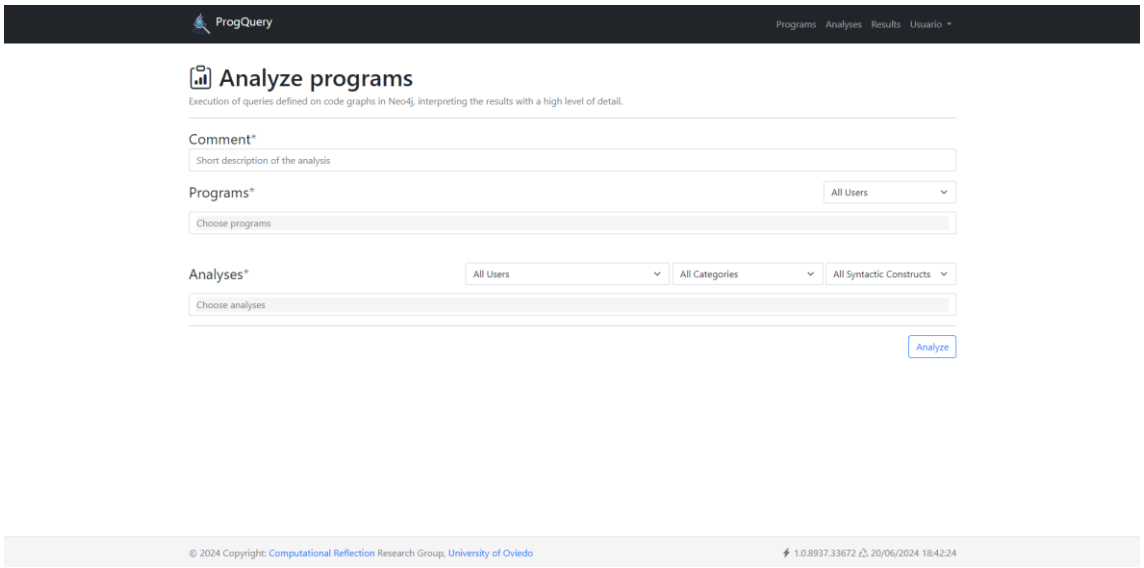


Ilustración 64. Pantalla de creación de un análisis de un programa

Si el resultado se registra con éxito, se confirma por la aplicación.

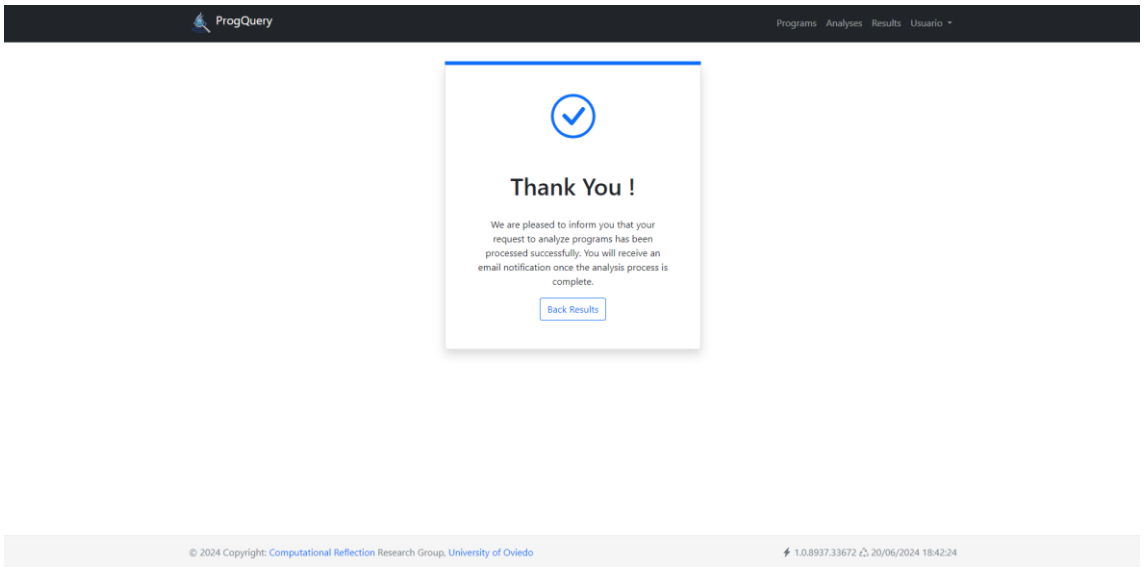


Ilustración 65. Pantalla de confirmación de realización del análisis de programa(s)

Al pulsar en *Back Results*, nos redirige a nuestros resultados.

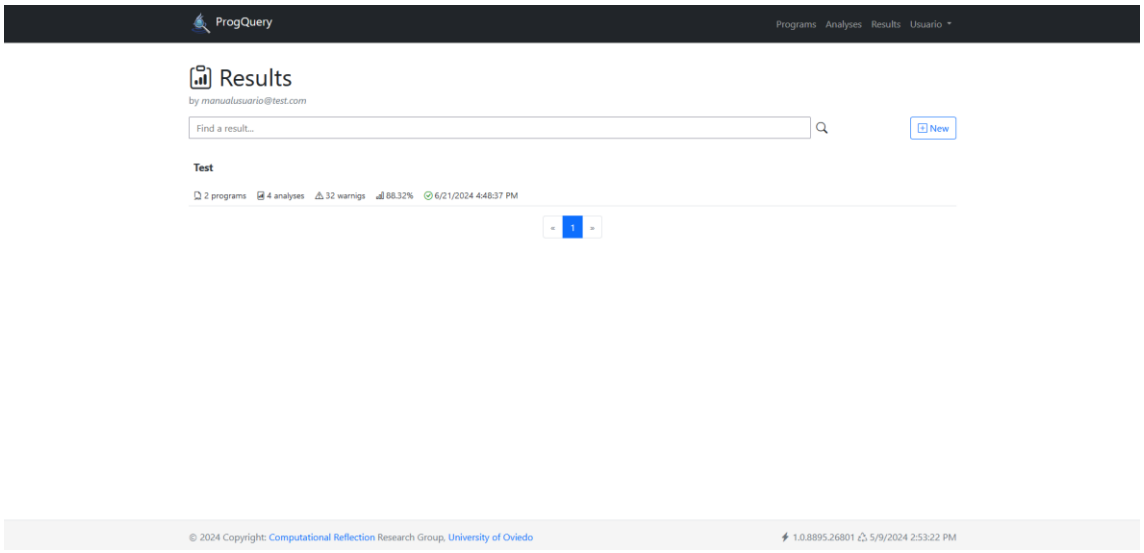


Ilustración 66. Pantalla del listado de resultados propios con uno nuevo guardado

Pulsamos sobre el nombre del resultado para acceder a los detalles de dicho resultado.

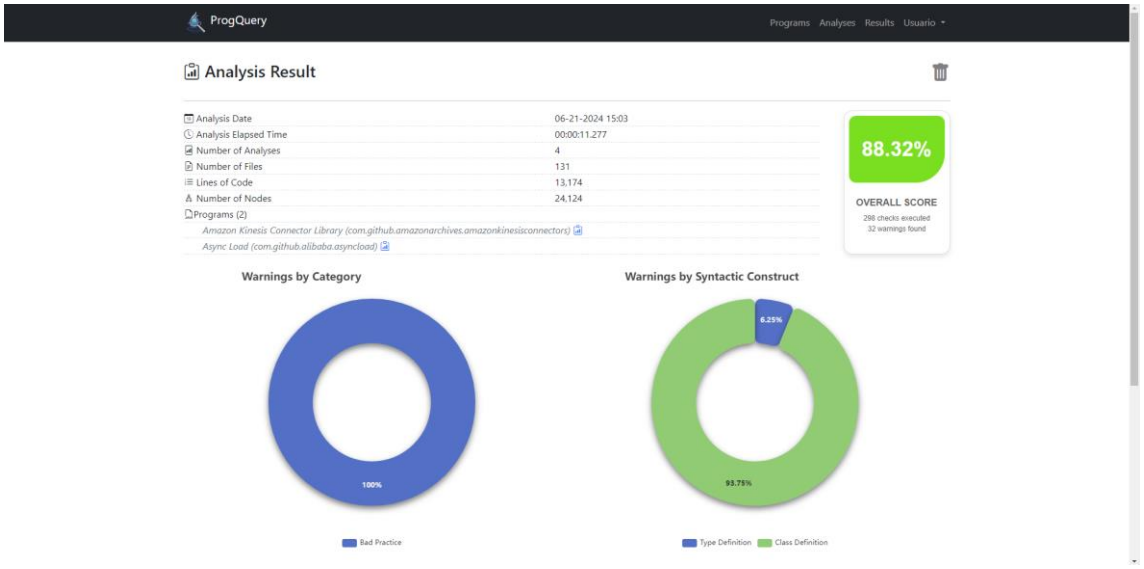


Ilustración 67. Pantalla de detalles de un resultado

Pulsamos sobre el icono de resultados al lateral del programa para acceder a los detalles del resultado de dicho programa.

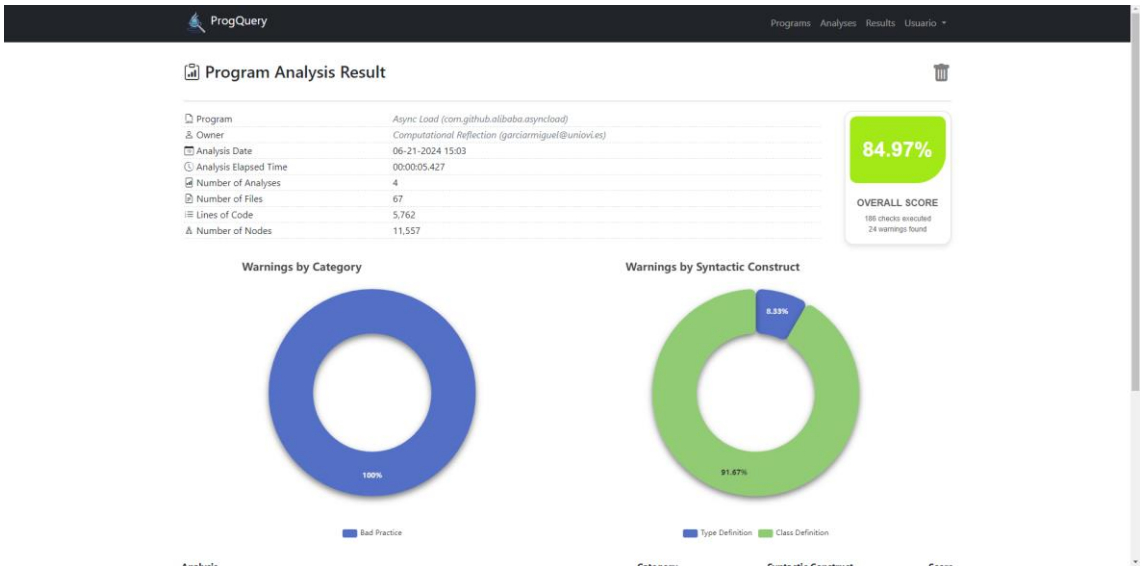


Ilustración 68. Pantalla de detalles de un resultado de un programa

Para borrar el resultado del programa, pulsaremos el icono de borrado, y nos saldrá una confirmación de borrado, para asegurarnos de que realmente queremos borrarlo o no. En este caso, pulsaremos sobre *Delete*.

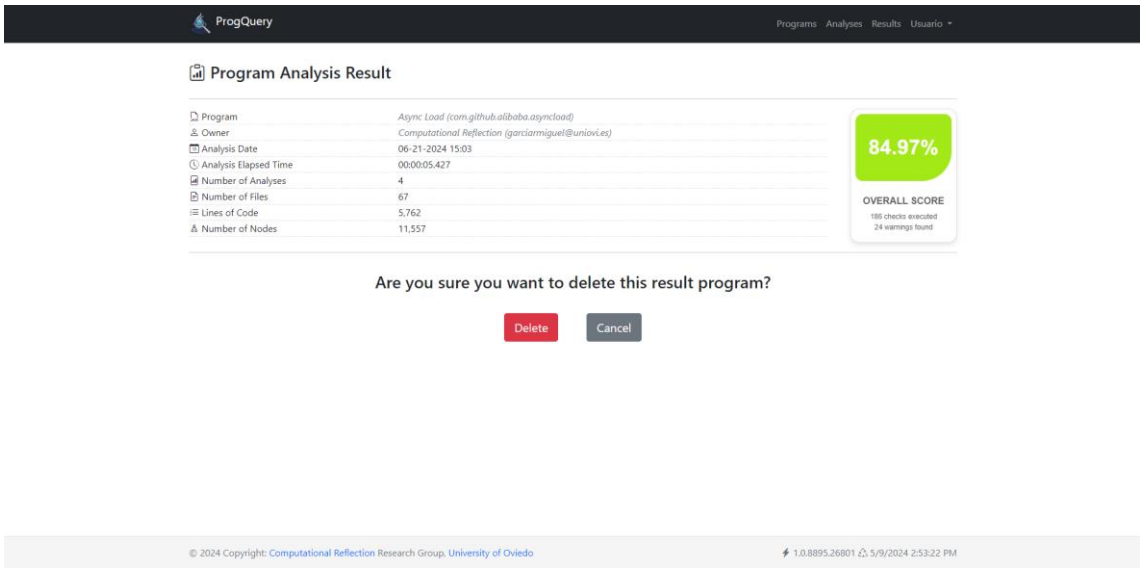


Ilustración 69. Pantalla de confirmación de eliminación del resultado de un programa

Podemos observar que ya no aparece el programa en el resultado general.

Para borrar este resultado general, pulsaremos el icono de borrado, y nos saldrá una confirmación de borrado, para asegurarnos de que realmente queremos borrarlo o no. En este caso, pulsaremos sobre *Cancel*.

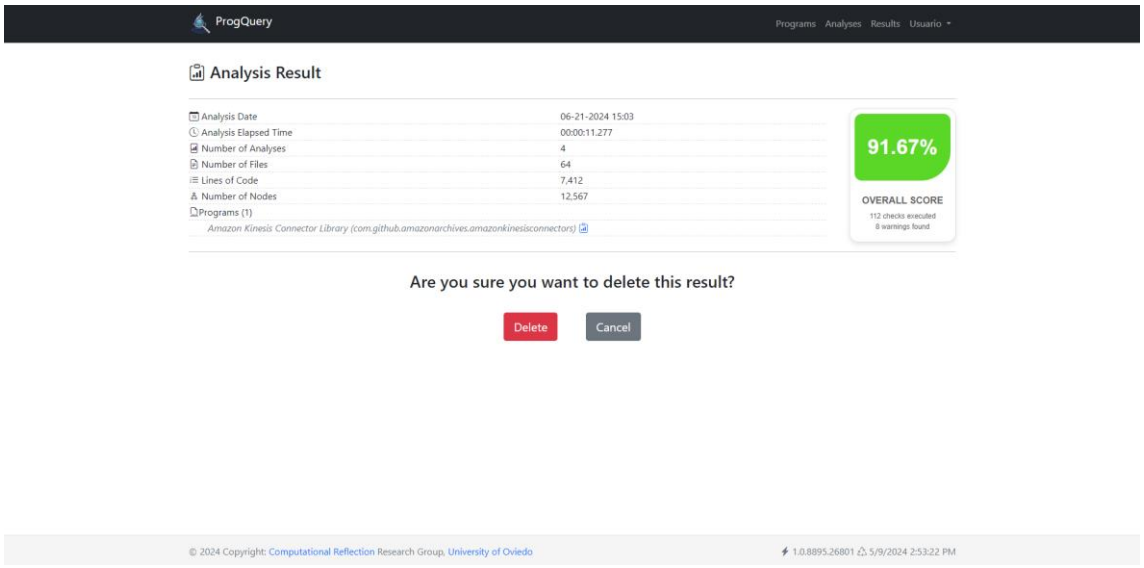


Ilustración 70. Pantalla de confirmación de eliminación de un resultado

7.1.8. Gestión de usuario y cierre de sesión

A continuación, para acceder a nuestro perfil de usuario, pulsamos sobre nuestro nombre y en *Profile.*

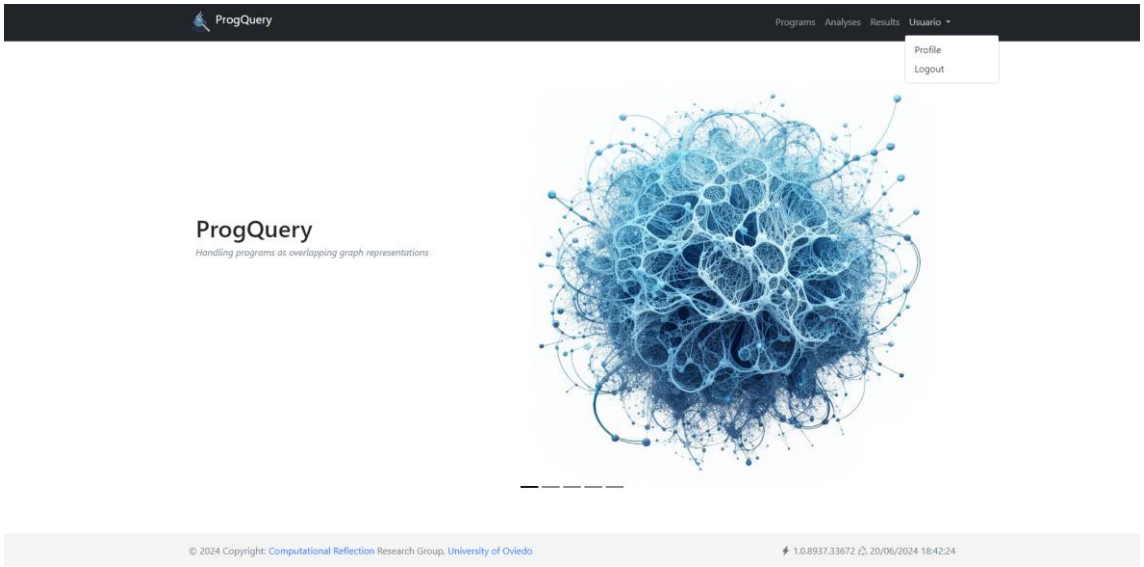


Ilustración 71. Pantalla con menú del usuario desplegado - Selección del perfil

Podemos observar los detalles básicos de nuestro usuario. Como creamos un programa, un análisis y un resultado, se muestra reflejado. Además, pulsando en cada uno de ellos, podremos acceder a sus respectivos listados.

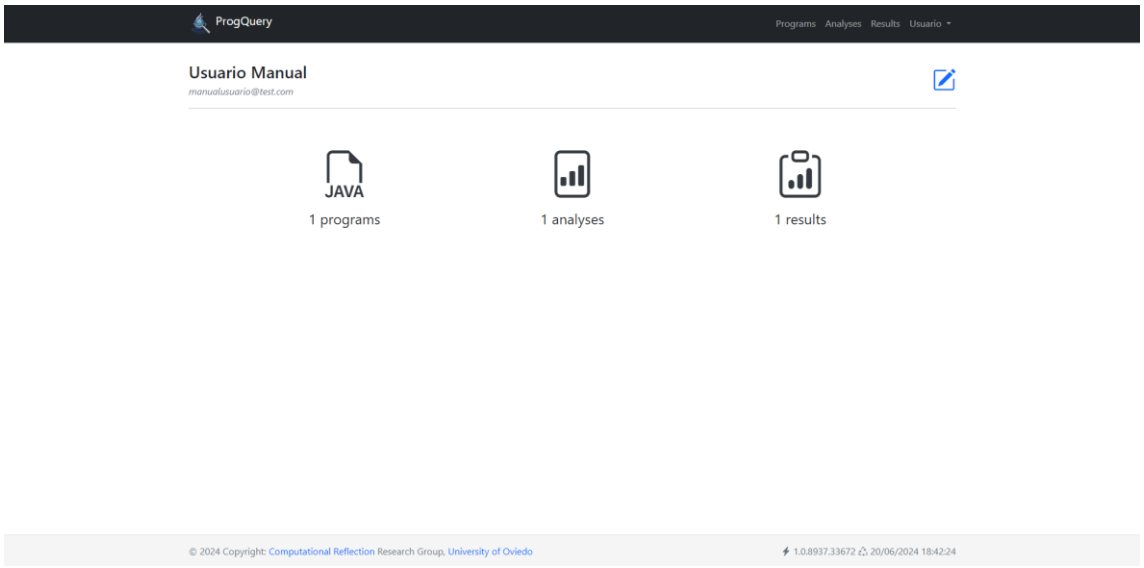


Ilustración 72. Pantalla del perfil de usuario

Para editar los datos de nuestro usuario, pulsamos sobre el icono de editar y al finalizar la edición pulsamos en *Save user* para guardar los nuevos datos introducidos.

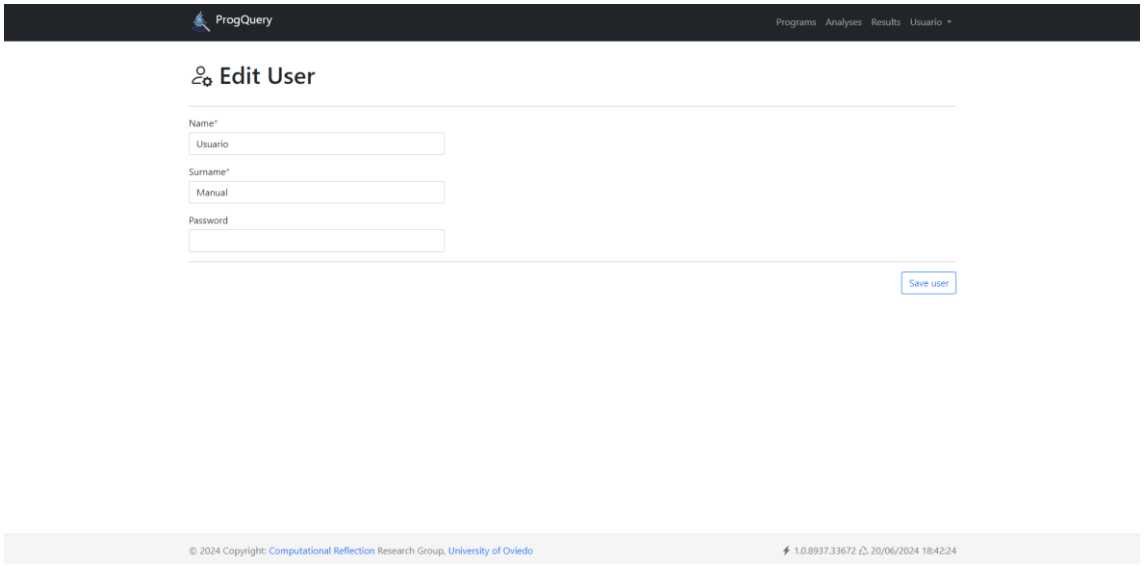


Ilustración 73. Pantalla de modificación del perfil de usuario

Finalmente, para cerrar sesión, pulsamos sobre nuestro nombre y en *Logout*.

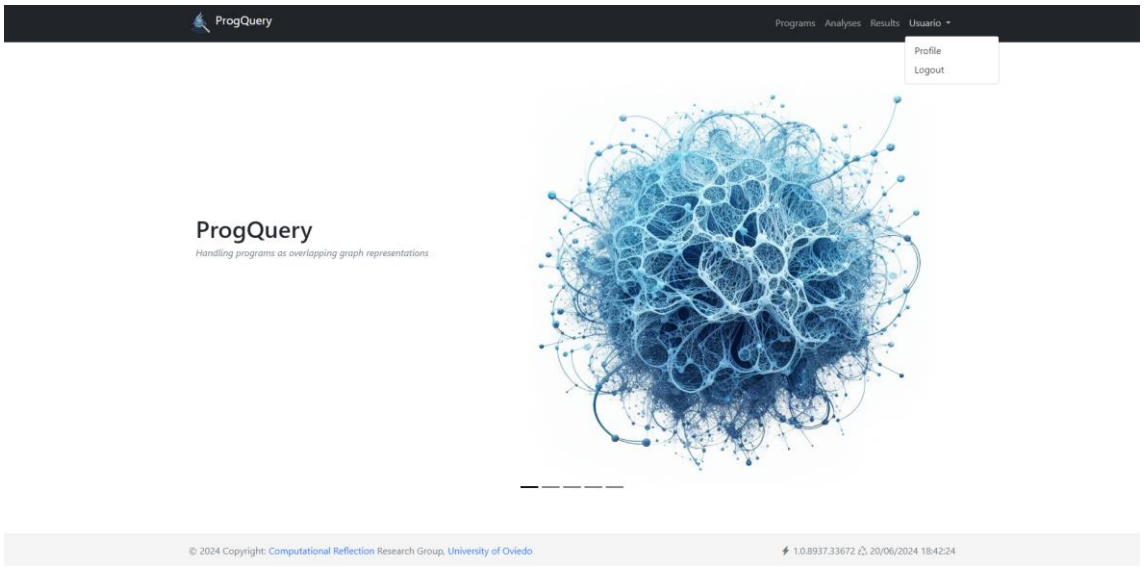


Ilustración 74. Pantalla con menú del usuario desplegado - Selección de cierre de sesión

7.1.9. Análisis básico

Por otro lado, un usuario no autenticado ni registrado, podrá realizar un análisis básico de un programa específico con una consulta específica. Tanto el programa a evaluar como el análisis a realizar podrán seleccionarse entre las opciones públicas disponibles en la aplicación, o bien, ingresarse manualmente mediante la introducción del código correspondiente.

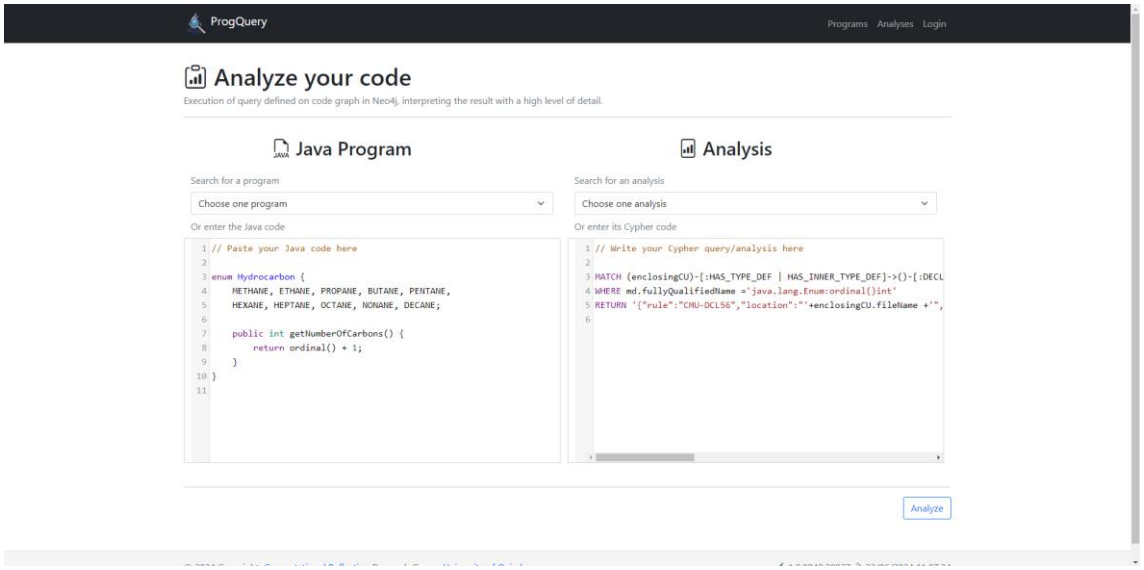


Ilustración 75. Pantalla de análisis básico

7.1.10. Estadísticas generales

Para finalizar, un usuario administrador, podrá visualizar las estadísticas de la aplicación accediendo a dicha opción. Pulsamos sobre “Statistics” y observamos las estadísticas generales de la aplicación.

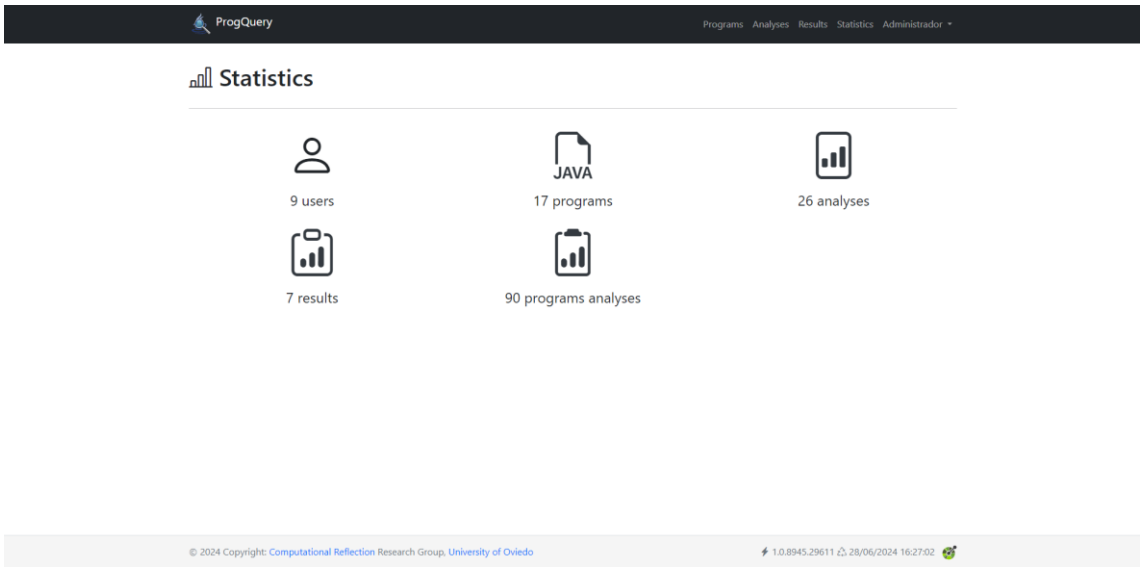


Ilustración 76. Pantalla de estadísticas generales

Si pulsamos sobre los programas, podemos observar las estadísticas asociadas a programas hasta el momento.

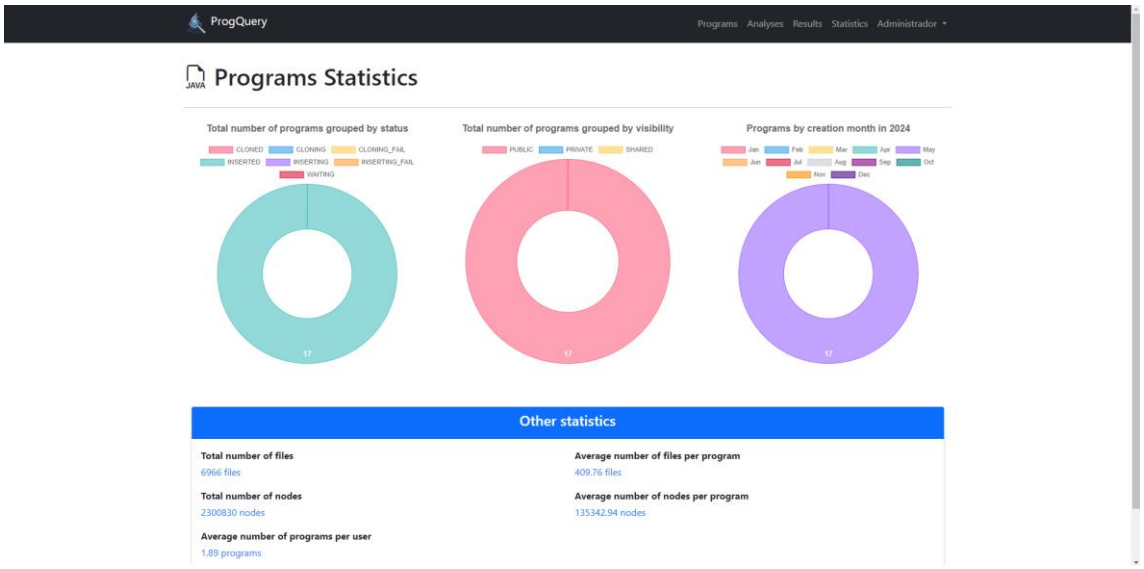


Ilustración 77. Pantalla de estadísticas de programas

Si pulsamos sobre los análisis, podemos observar las estadísticas asociadas a análisis hasta el momento.

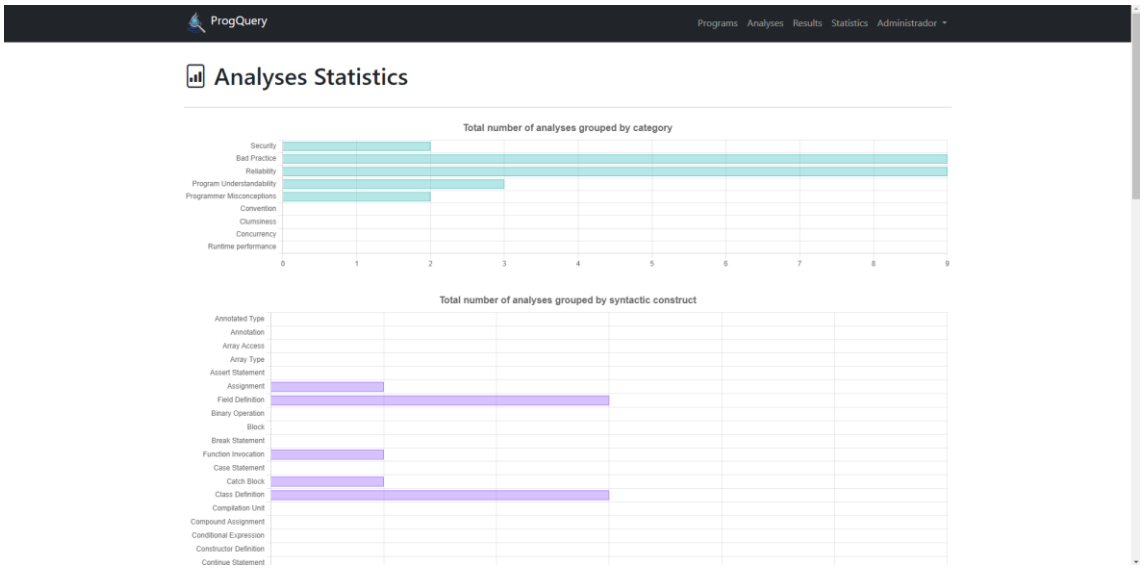


Ilustración 78. Pantalla de estadísticas de análisis

Si pulsamos sobre los resultados, podemos observar las estadísticas asociadas a resultados hasta el momento.

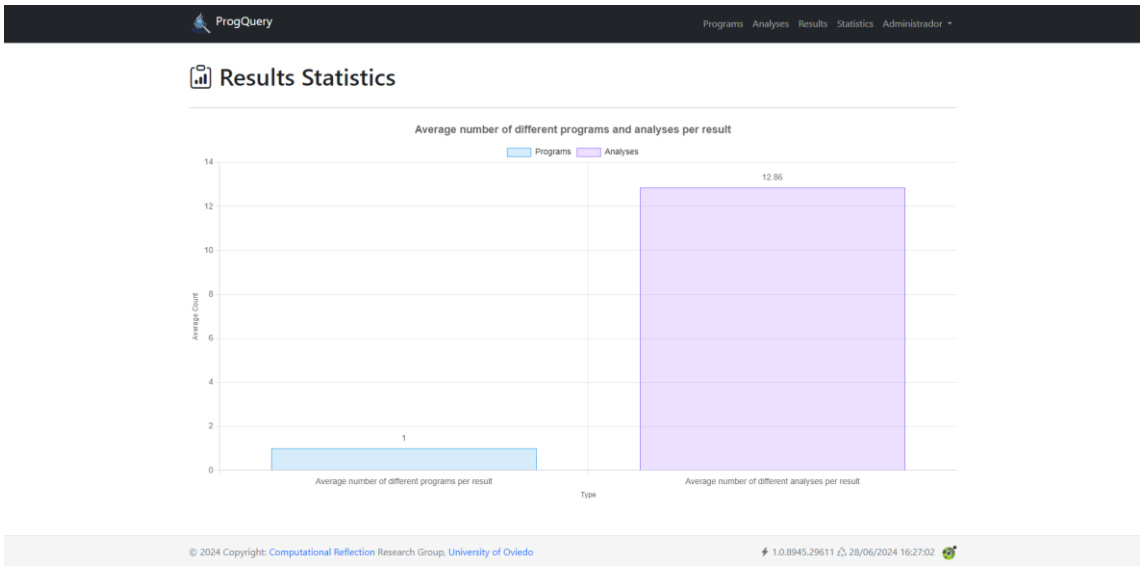


Ilustración 79. Pantalla de estadísticas de resultados

Si pulsamos sobre programas-análisis, podemos observar las estadísticas asociadas a programas-análisis hasta el momento.

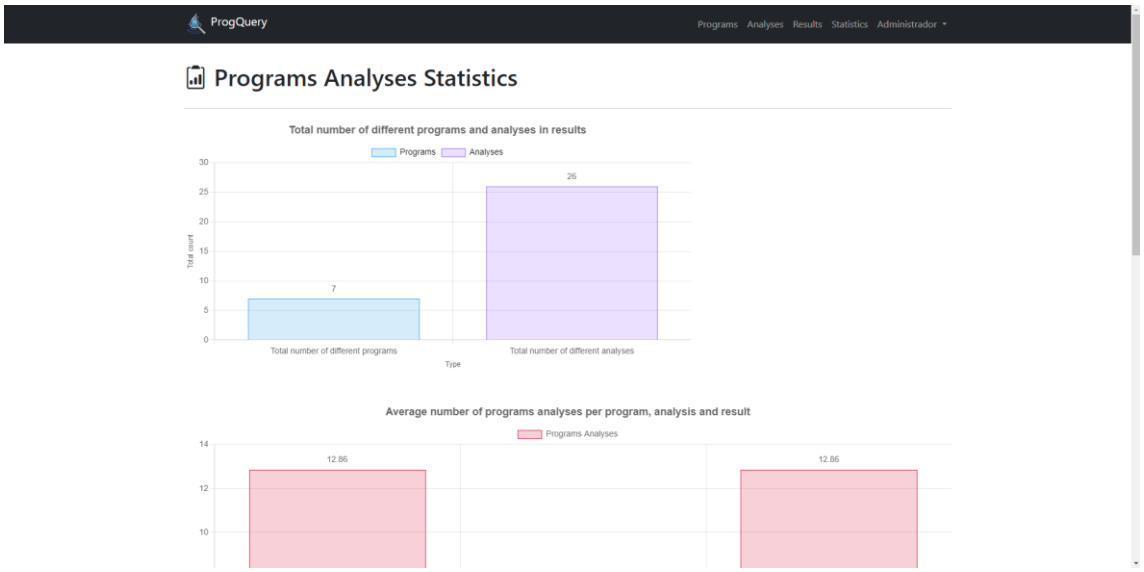


Ilustración 80. Pantalla de estadísticas de programas-análisis

Y, por último, en el pie de página, observamos un icono de API que al pulsar sobre él, nos redirige a nuestra API.

Ilustración 81. Icono API

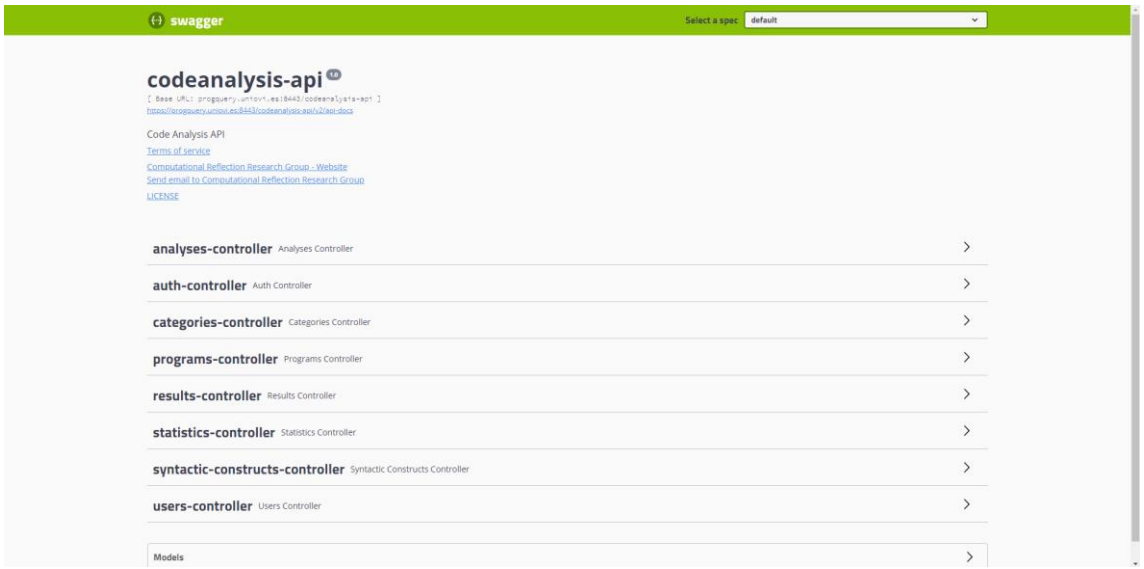


Ilustración 82. API

Capítulo 8. Dirección y gestión del proyecto

8.1. Planificación del proyecto

La planificación de un proyecto es una fase crucial que establece la estructura y el enfoque necesarios para lograr los objetivos propuestos de manera eficiente y efectiva.

8.1.1. Identificación de los interesados

La identificación de los interesados es crucial para reconocer a todas las personas, grupos o entidades que pueden influir o ser influenciados por el proyecto. Comprender sus roles y expectativas facilita la gestión de sus necesidades, contribuyendo al éxito del proyecto. Los interesados identificados en este proyecto son los siguientes.

Nombre	Cargo	Departamento/División
Inés Díaz del Rey	Desarrolladora	Estudiante
Miguel García Rodríguez	Tutor académico	Departamento de Informática
Grupo de Investigación Computational Reflection Research Group	Investigadores	Computational Reflection Research Group
Desarrolladores de software (usuarios)	Desarrolladores de software (usuarios)	Desarrolladores de software (usuarios)
Desarrolladores	Desarrolladores	Desarrolladores

Tabla 106. Interesados

8.1.2. OBS, PBS

En la planificación, es fundamental definir tanto la estructura organizacional como la de los productos. La OBS describe la jerarquía y responsabilidades dentro del equipo del proyecto, mientras que la PBS descompone los entregables del proyecto en componentes manejables. Estos desgloses permiten una gestión eficiente de recursos y tareas.

OBS (Organizational Breakdown Structure):

1. *Estudiante (Inés Díaz del Rey)*
    - Cargo: Desarrolladora
    - Departamento/División: Estudiante de Máster
    - Responsabilidades:
        - Desarrollo y codificación de la Web API y la aplicación web.
        - Documentación del proyecto y preparación del Proyecto Final.
        - Pruebas y aseguramiento de la calidad del software.

    - Coordinación con el tutor académico y el grupo de investigación para asegurar la alineación con los objetivos del proyecto.

2. *Tutor Académico (Miguel García Rodríguez)*
    - Cargo: Tutor Académico
    - Departamento/División: Departamento de Informática, Universidad de Oviedo
    - Responsabilidades:
        - Supervisión general del proyecto.
        - Proveer orientación técnica y académica a la estudiante.
        - Revisión de la documentación y los entregables del proyecto.
        - Evaluación del progreso del proyecto y retroalimentación continua.
3. *Grupo de Investigación Computational Reflection Research Group*
    - Cargo: Investigadores
    - Departamento / División: Computational Reflection Research Group
    - Responsabilidades:
        - Apoyo en la investigación y desarrollo de soluciones técnicas avanzadas.
        - Colaboración en la implementación de funcionalidades complejas.
        - Proveer recursos y conocimiento especializado para el desarrollo del proyecto.
        - Participación en la validación y prueba de los componentes del sistema.

PBS (Project Breakdown Structure)

1. *Web API*
    - Descripción: Una API web desarrollada para proporcionar servicios específicos del proyecto.
    - Componentes Clave:
        - Endpoints para la gestión de usuarios, programas, análisis y resultados.
        - Integración con bases de datos PostgreSQL y Neo4j.
    - Responsables: Estudiante, con orientación del Tutor Académico y apoyo del Grupo de Investigación.
2. *Aplicación Web*
    - Descripción: Una interfaz web que interactúa con la API web para ofrecer funcionalidades a los usuarios finales.
    - Componentes Clave:

  - Interfaz de usuario intuitiva y responsiva.
  - Funcionalidades de registro, inicio de sesión y gestión de perfiles.
  - Visualización y gestión de programas, análisis y resultados.

- Responsables: Estudiante, con orientación del Tutor Académico y apoyo del Grupo de Investigación.

3. *Proyecto Final*

- Descripción: El entregable final que incluye toda la documentación relevante.
- Componentes Clave:
  - Documentación completa del desarrollo y uso del sistema.
  - Código fuente organizado y comentado.
  - Manuales de usuario y de instalación.
  - Responsables: Estudiante, con revisión del Tutor Académico.

8.1.3. Planificación inicial. WBS

En este apartado, se muestra la planificación inicial que se ha elaborado. Las tareas siguen un orden lineal puesto que se han desarrollado por una única persona, aunque según el puesto que ejercía dicha persona se le ha asignado un recurso diferente acorde al cargo. El calendario utilizado está formado por 40 horas semanales de trabajo distribuidas en cinco días (lunes, martes, miércoles, jueves y viernes), considerando los días no lectivos o festivos del calendario del curso 2023/2024 como días no laborales. La planificación del proyecto está dividida en varias tareas, subtareas e hitos. El proyecto tiene una duración total de 550 horas.

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
1		**Realizar una reunión cada dos semanas con el tutor**	**64 días**	**lun 08/01/24**	**lun 15/04/24**	
2		Realizar una reunión ca	2 horas	lun 08/01/24	lun 08/01/24	InésJP
3		Realizar una reunión ca	2 horas	lun 22/01/24	lun 22/01/24	InésJP
4		Realizar una reunión ca	2 horas	lun 05/02/24	lun 05/02/24	InésJP
5		Realizar una reunión ca	2 horas	lun 19/02/24	lun 19/02/24	InésJP
6		Realizar una reunión ca	2 horas	lun 04/03/24	lun 04/03/24	InésJP
7		Realizar una reunión ca	2 horas	lun 18/03/24	lun 18/03/24	InésJP
8		Realizar una reunión ca	2 horas	lun 01/04/24	lun 01/04/24	InésJP
9		Realizar una reunión ca	2 horas	lun 15/04/24	lun 15/04/24	InésJP

Ilustración 83. Reuniones con el tutor

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
10		**Análisis, Dirección y Gestión del proyecto**	**14 días**	**lun 08/01/24**	**lun 29/01/24**	
11		Definir el propósito y la justificación del proyecto	3 horas	lun 08/01/24	lun 08/01/24	InésJP
12		Definir el objetivo del proyecto	3 horas	lun 08/01/24	lun 08/01/24	InésJP
13		Definir el alcance del proyecto	4 horas	mar 09/01/24	mar 09/01/24	InésJP
14		Estudio del arte	12 horas	mar 09/01/24	mié 10/01/24	InésA
15		Familiarizarse con los lenguajes de programación, el entorno y las bases de datos	8 horas	jue 11/01/24	jue 11/01/24	InésDS
16		Comprender ProgQuery	3 horas	vie 12/01/24	vie 12/01/24	InésDS
17		Definir el sistema	5 horas	vie 12/01/24	vie 12/01/24	InésA
18		Identificar los stackholders y definir sus roles	3 horas	lun 15/01/24	lun 15/01/24	InésA
19		Realizar un análisis de requisitos	13 horas	lun 15/01/24	mar 16/01/24	InésA
20		Realizar un análisis de casos de uso y sus escenarios	22 horas	mié 17/01/24	lun 22/01/24	InésA
21		Definir la planificación del proyecto	8 horas	mar 23/01/24	mar 23/01/24	InésJP
22		Definir el presupuesto del proyecto	5 horas	mié 24/01/24	mié 24/01/24	InésJP
23		Definir los riesgos del proyecto	19 horas	mié 24/01/24	lun 29/01/24	InésA
24		Entrega de la Planificación del Proyecto	0 días	mié 31/01/24	mié 31/01/24	
25		Entrega del Presupuesto del Proyecto	0 días	mié 31/01/24	mié 31/01/24	

Ilustración 84. Tareas de análisis, dirección y gestión del proyecto e hitos

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
26		**Diseño del Sistema**	**9 días**	**mar 30/01/24**	**vie 09/02/24**	
27		Diseñar la arquitectura del sistema	8 horas	mar 30/01/24	mar 30/01/24	InésAS
28		Crear diagramas y modelos de datos	16 horas	mié 31/01/24	jue 01/02/24	InésAS
29		Diseñar la interfaz de usuario	14 horas	vie 02/02/24	lun 05/02/24	InésAS
30		Definir los componentes del sitema y sus interacciones	24 horas	mar 06/02/24	vie 09/02/24	InésAS

Ilustración 85. Tareas de diseño del sistema

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
31		**Desarrollo del Software**	**29 días**	**lun 12/02/24**	**jue 21/03/24**	
32		**Desarrollar la Web API**	**23 días**	**lun 12/02/24**	**mié 13/03/24**	
33		Configurar proyecto y entorno de desarrollo de la Web API	8 horas	lun 12/02/24	lun 12/02/24	InésDS
34		Implementar la gestión de usuarios y autenticación	40 horas	mar 13/02/24	mar 20/02/24	InésDS
35		Implementar la gestión de programas	42 horas	mar 20/02/24	mar 27/02/24	InésDS
36		Implementar la gestión de análisis	40 horas	mar 27/02/24	mar 05/03/24	InésDS
37		Implementar la gestión de resultados	42 horas	mié 06/03/24	mié 13/03/24	InésDS
38		Desarrollar la aplicación web	46 horas	mié 13/03/24	jue 21/03/24	InésDS

Ilustración 86. Tareas de desarrollo del software

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
40		**Pruebas**	**8 días**	**vie 22/03/24**	**mar 09/04/24**	
41		Realizar pruebas unitarias para la gestión de usuarios	10 horas	vie 22/03/24	lun 01/04/24	InésDS
42		Realizar pruebas unitarias para la gestión de programas	10 horas	lun 01/04/24	mar 02/04/24	InésDS
43		Realizar pruebas unitarias para la gestión de análisis	10 horas	mar 02/04/24	mié 03/04/24	InésDS
44		Realizar pruebas unitarias para la gestión de resultados	10 horas	jue 04/04/24	vie 05/04/24	InésDS
45		Realizar pruebas de integración	14 horas	vie 05/04/24	lun 08/04/24	InésDS
46		Identificar y solucionar errores y problemas	8 horas	mar 09/04/24	mar 09/04/24	InésDS
47		**Despliegue**	**2 días**	**mié 10/04/24**	**jue 11/04/24**	
48		Configurar el servidor y alojar la aplicación web y la Web API	12 horas	mié 10/04/24	jue 11/04/24	InésDS
49		Realizar pruebas finales	4 horas	jue 11/04/24	jue 11/04/24	InésDS

Ilustración 87. Tareas de pruebas y despliegue

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
50		**Documentación**	**9 días**	**vie 12/04/24**	**mié 24/04/24**	
51		Realizar manuales de usuario	14 horas	vie 12/04/24	lun 15/04/24	InésDS
52		Realizar documentación final del proyecto	50 horas	mar 16/04/24	mié 24/04/24	InésJP
53		**Cierre del Proyecto**	**1 día**	**mié 24/04/24**	**mié 24/04/24**	
54		Realizar una revisión final del proyecto	2 horas	mié 24/04/24	mié 24/04/24	InésJP
55		Realizar una reunión de cierre del proyecto	2 horas	mié 24/04/24	mié 24/04/24	InésJP
56		Entrega Final del Proyecto	0 días	jue 30/05/24	jue 30/05/24	

Ilustración 88. Tareas de documentación y cierre del proyecto

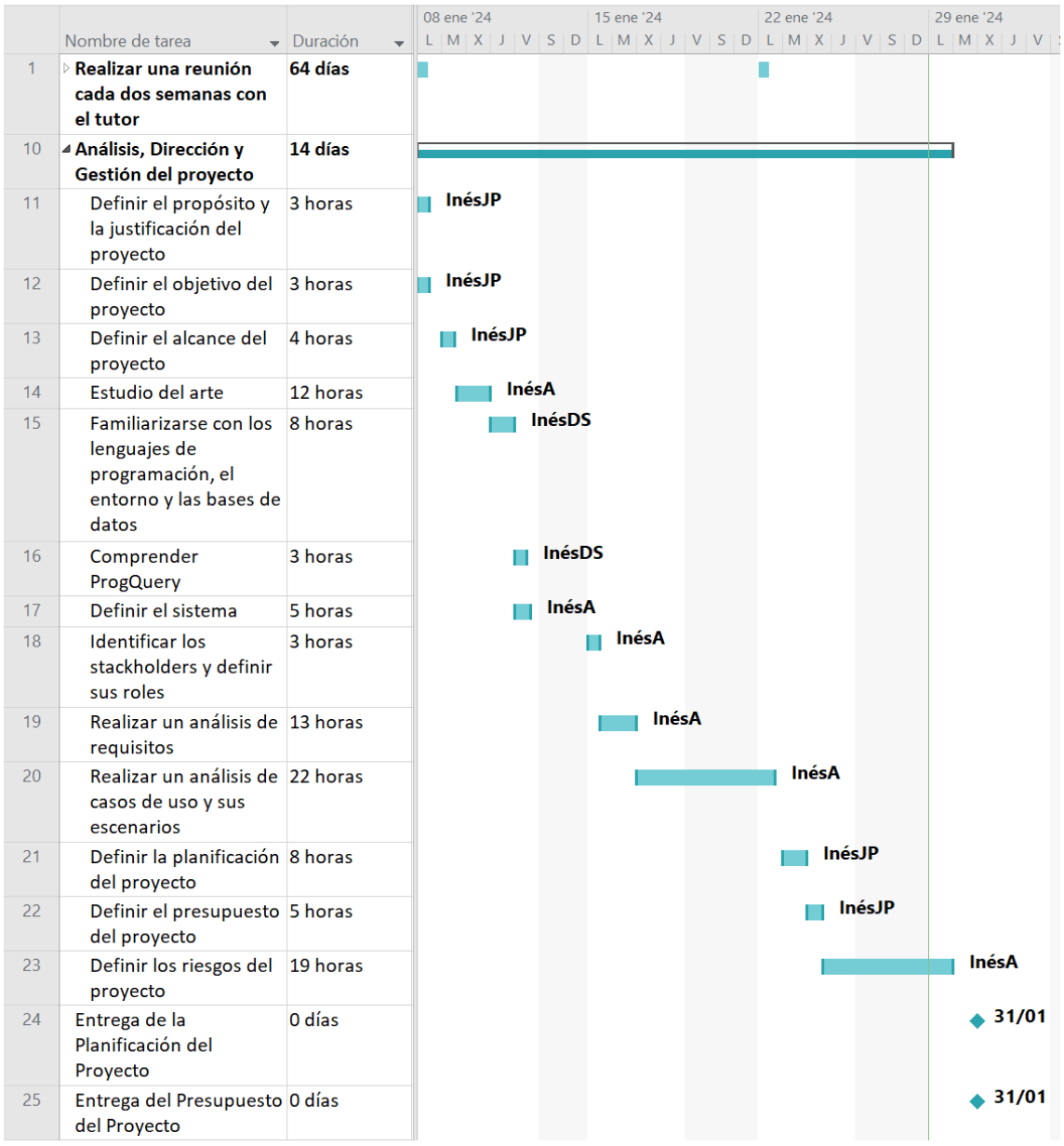


Ilustración 89. Diagrama de Gantt I

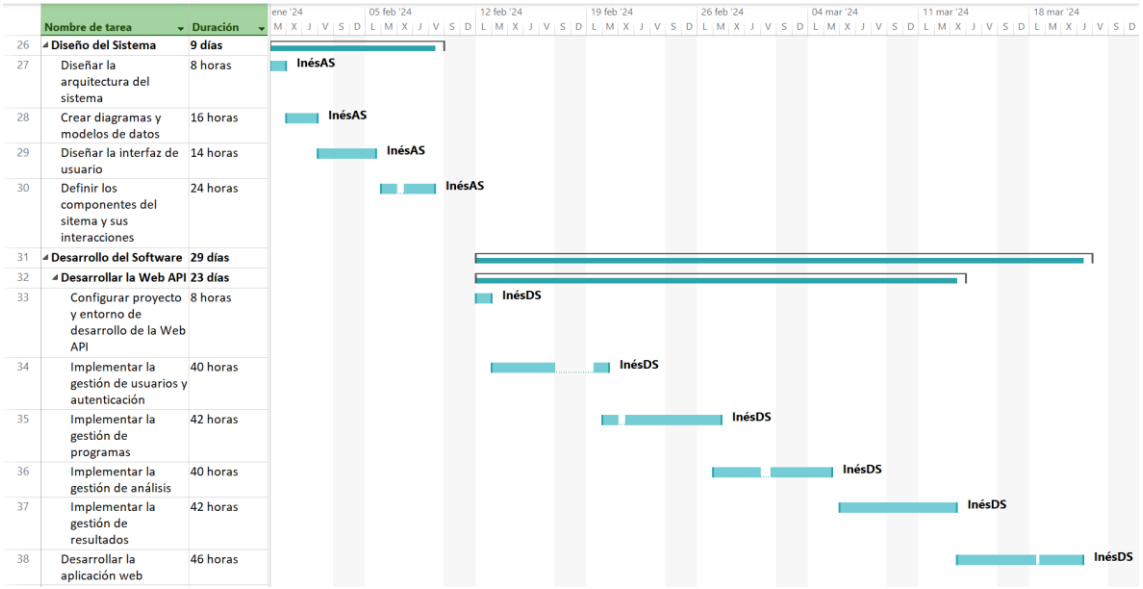


Ilustración 90. Diagrama de Gantt II

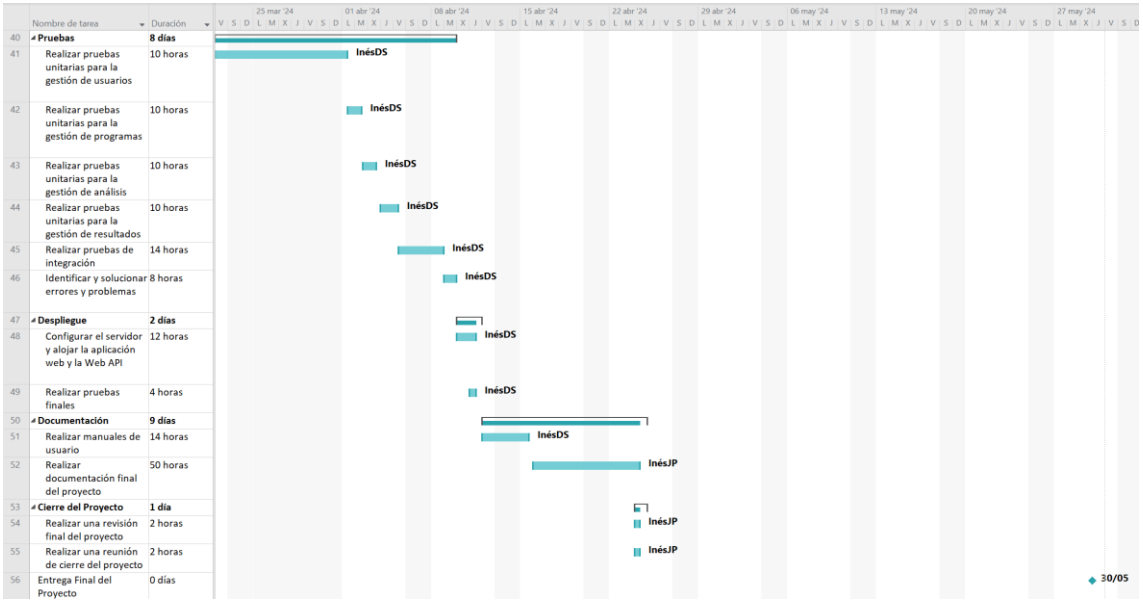


Ilustración 91. Diagrama de Gantt III

8.1.4. Riesgos

8.1.4.1. Plan de gestión de riesgos

8.1.4.1.1. Metodología

La gestión de riesgos es un componente crítico en cualquier proyecto, especialmente en un entorno dinámico y complejo como el desarrollo de software. Para abordar de manera efectiva los riesgos que puedan surgir en el proyecto, se ha establecido una metodología sólida que consta de dos aspectos esenciales: una metodología general y una metodología de gestión de riesgos.

El objetivo de esta metodología es anticipar los riesgos, reducir su impacto y, en última instancia, garantizar el éxito del proyecto. A continuación, se detallarán ambas metodologías, proporcionando un enfoque estructurado para la gestión de riesgos del proyecto.

8.1.4.1.1.1. Metodología general

La metodología general de gestión de riesgos se basa en dos fases que abarcan la identificación, análisis, priorización y manejo de los riesgos del proyecto:

1. *Identificación de riesgos iniciales*: En esta fase, se llevará a cabo un proceso de tormenta de ideas con todos los socios involucrados en el proyecto para identificar los posibles riesgos. Esta etapa es crucial para capturar riesgos desde el principio.
2. *Elaboración del plan de gestión de riesgos*: Se creará un plan de gestión de riesgos que incluye los criterios para la gestión de riesgos, un registro de riesgos y una hoja de datos de riesgos para todos los riesgos priorizados.
3. *Monitorización continua de riesgos*: Durante todo el proyecto, se realizarán evaluaciones regulares de riesgos para garantizar que no representen amenazas significativas. Se actualizará el plan de gestión de riesgos a medida que cambien las condiciones del proyecto y se tomarán decisiones para mitigar los riesgos.
4. *Informe final de riesgos*: Al final del proyecto, se preparará un informe final que documentará la evolución de los riesgos desde su identificación inicial hasta el cierre del proyecto.

8.1.4.1.1.2. Metodología de gestión de riesgos

Esta metodología se enfoca en los aspectos técnicos de la gestión de riesgos y se divide en seis pasos:

1. *Planificar la gestión de riesgos*: Definir cómo se llevarán a cabo las actividades de gestión de riesgos en el proyecto.
2. *Identificar los riesgos*: Determinar los riesgos que podrían afectar el proyecto y documentar sus características.
3. *Realizar el análisis cualitativo de riesgos*: Priorizar los riesgos en función de su probabilidad e impacto, lo que ayudará a determinar si se requieren análisis adicionales o acciones posteriores.
4. *Realizar el análisis cuantitativo de riesgos*: Analizar numéricamente el efecto de los riesgos en los objetivos generales del proyecto.
5. *Planificar la respuesta a los riesgos*: Desarrollar estrategias y acciones para mejorar las oportunidades y reducir las amenazas a los objetivos del proyecto.
6. *Monitorizar y controlar los riesgos*: Implementar planes de respuesta, rastrear riesgos identificados, supervisar riesgos residuales, identificar nuevos riesgos y evaluar la efectividad del proceso contra riesgos a lo largo del proyecto.

8.1.4.1.2. Herramientas y tecnologías

En el proyecto se emplearán diversas herramientas y técnicas para gestionar los riesgos de manera efectiva.

8.1.4.1.2.1. Tormenta de ideas

En una fase inicial, se utilizará esta técnica para identificar una lista amplia de riesgos potenciales. Los miembros del equipo participarán en sesiones colaborativas para discutir abiertamente los principales objetivos del proyecto y explorar posibles desafíos y oportunidades.

8.1.4.1.2.2. Evaluaciones periódicas

Se realizarán evaluaciones regulares de los indicadores según el plan establecido en la hoja de riesgos. Estos resultados se analizarán en reuniones de seguimiento de riesgos.

8.1.4.1.2.3. Reuniones de riesgos

En cada reunión de seguimiento del proyecto, se incluirá un punto de discusión dedicado a los riesgos. Esto permitirá tomar decisiones y abordar los riesgos de manera proactiva.

8.1.4.1.2.4. Método Delphi

En situaciones de desacuerdo en la evaluación de riesgos o en las decisiones relacionadas, se recurrirá al Método Delphi. Esta técnica implica la recopilación de opiniones de expertos para llegar a consensos sobre los riesgos y las acciones a tomar.

8.1.4.1.2.5. Checklists

Se emplearán listas de verificación (checklists) que contienen elementos específicos para revisar o considerar. Estas listas ayudarán a prevenir errores comunes y garantizar la adopción de mejores prácticas desde el inicio del proyecto.

8.1.4.1.2.6. Estudio de incidencias

Se analizarán incidentes y problemas que hayan surgido en proyectos anteriores. Este enfoque permitirá detectar riesgos recurrentes y aprender de la experiencia pasada para evitar la repetición de errores.

8.1.4.1.3. Roles y responsabilidades

En este caso, las responsabilidades estarían a cargo del estudiante, siendo las siguientes:

1. Liderar y supervisar la gestión de riesgos en el proyecto, integrándola en la planificación y toma de decisiones.
2. Coordinar las acciones de mitigación y seguimiento de cada riesgo.
3. Identificar nuevos riesgos y realizar su gestión.

8.1.4.1.4. Presupuesto

Item	**Concepto**	**Asignación (€)**
1	Identificación de riesgos	350 €
2	Análisis y priorización de los riesgos	470 €
3	Planificación de los riesgos	350 €
4	Definición de planes de contingencia	350 €
5	Actualización y monitorización de los riesgos	470 €
	TOTAL	**1990 €**

8.1.4.1.5. Calendario

Hito / Actividad	**Fecha**
Identificación de riesgos	DD/MM/YYYY
Análisis y priorización de los riesgos	DD/MM/YYYY
Planificación de los riesgos	DD/MM/YYYY
Definición de planes de contingencia	DD/MM/YYYY

8.1.4.1.6. Categorías de riesgo

Para facilitar la identificación y comprensión de los riesgos, los clasificamos en diferentes categorías., pudiendo cada uno de ellos pertenecer a una o más categorías.

1. Técnico
    1.1. Requisitos
    1.2. Tecnología
    1.3. Complejidad e interfaces
    1.4. Prestaciones y fiabilidad
    1.5. Calidad
2. Externo
    2.1. Subcontratistas y proveedores
    2.2. Regulación
    2.3. Mercado
    2.4. Usuario
    2.5. Tiempo
3. Organizacional
    3.1. Dependencias del proyecto
    3.2. Recursos
    3.3. Financiación

3.4. Personal

4. Gestión del proyecto

4.1. Estimación

4.2. Planificación

4.3. Control

4.4. Comunicación

8.1.4.1.7. Definiciones de probabilidad

Muy Baja	(0%...20%] El valor usado en la matriz de probabilidad e impacto es 10%. La probabilidad de que el riesgo se materialice es extremadamente baja, casi improbable.
Baja	(20%...40%] El valor usado en la matriz de probabilidad e impacto es 30%. La probabilidad de que el riesgo ocurra es baja, aunque no se puede descartar por completo.
Media	(4%...60%] El valor usado en la matriz de probabilidad e impacto es 50%. Existe una probabilidad moderada de que el riesgo se materialice; es una posibilidad significativa.
Alta	(60%...80%] El valor usado en la matriz de probabilidad e impacto es 70%. Las probabilidades de que el riesgo ocurra son considerables; se debe prestar atención y considerar medidas preventivas.
Muy Alta	(80%...100%] El valor usado en la matriz de probabilidad e impacto es 90%. El riesgo afectará al proyecto con una probabilidad muy alta; es altamente probable que se materialice y requiere acción inmediata.

8.1.4.1.8. Definiciones de impacto por objetivos

Impacto sobre los objetivos principales					
Objetivos	*Escalas relativas o numéricas*				
	Muy Bajo / 5 %	Bajo / 15 %	Medio / 30 %	Alto / 55 %	Crítico / 90 %
Presupuesto	Incremento insignificante del presupuesto	Incremento del presupuesto menor al 10 %	Incremento del presupuesto entre el 10%-20%	Incremento del presupuesto entre el 20%-35%	Incremento del presupuesto superior al 35%
Planificación	Incremento insignificante de tiempo	Incremento de tiempo menor al 5%	Incremento de tiempo entre el 5%-10%	Incremento de tiempo entre el 10%-20%	Incremento de tiempo superior al 20%
Alcance	Reducciones del alcance	Afectadas áreas poco	Afectadas áreas	Afectadas áreas que impidan	Reducciones que provocan

	poco apreciables	relevantes del alcance	importantes del alcance	lograr requisitos de alto nivel	que el proyecto deje de ser funcional
Calidad	Reducción de la calidad inapreciable	Solo se ven afectadas aplicaciones muy exigentes	La reducción de la calidad requiere la aprobación del jefe de proyecto	La reducción de la calidad no es aceptada por el cliente	Reducciones que provocan que el proyecto deje de ser útil

8.1.4.1.9. Matriz de probabilidad e impacto

Probabilidad	**Muy Alta**	***0,90***	0,05	0,14	0,27	0,50	0,81
	Alta	***0,70***	0,04	0,11	0,21	0,39	0,63
	Media	***0,50***	0,03	0,08	0,15	0,28	0,45
	Baja	***0,30***	0,02	0,05	0,09	0,17	0,27
	Muy Baja	***0,10***	0,01	0,02	0,03	0,06	0,09
			0,05	***0,15***	***0,30***	***0,55***	***0,90***
			Muy Bajo	**Bajo**	**Medio**	**Alto**	**Crítico**
			Impacto				

8.1.4.1.10. Niveles de tolerancia

El umbral de riesgo se establece en 0,35 como el punto a partir del cual se considerará que los riesgos representan una amenaza significativa para el proyecto.

8.1.4.1.11. Planes de contingencia

8.1.4.1.11.1. Plan de presupuesto

Si se identifican costos adicionales que superan el 5% del presupuesto total del proyecto, se realizará una revisión detallada del alcance y se buscarán formas de reducir los costos sin comprometer la calidad.

8.1.4.1.11.2. Plan de planificación

En caso de retrasos imprevistos que amenacen con superar la fecha límite de entrega del proyecto, se realizará una revisión exhaustiva del cronograma y se asignarán recursos adicionales si es necesario para asegurar que el proyecto se entregue a tiempo.

8.1.4.1.11.3. Plan de alcance y calidad

Este plan se implementará en caso de que se produzca un desvío significativo en el alcance del proyecto o que los entregables no cumplan con los estándares de calidad establecidos. Se identificarán los problemas o cambios, se evaluarán y se tomarán medidas correctivas.

8.1.4.1.12. Formatos de la documentación

Para orientar la gestión de la documentación, se seguirán las siguientes normas como punto de referencia:

- UNE-ISO 31000:2010: Gestión del riesgo. Proporciona un marco general para identificar, evaluar y tratar los riesgos en proyectos.
- UNE-EN 31010:2011: Gestión del riesgo. Se enfoca en las técnicas para evaluar y apreciar los riesgos.

8.1.4.1.13. Seguimiento

Para garantizar un control efectivo de los riesgos en el proyecto, se establece la siguiente política de seguimiento:

- Cada mes durante el desarrollo del proyecto, se llevará a cabo un análisis exhaustivo de los riesgos identificados. En este proceso, se recalculará la probabilidad de ocurrencia y el impacto de cada riesgo en función de los cambios en el entorno y el avance del proyecto.
- Mensualmente, se realizará una revisión para identificar nuevas amenazas que puedan surgir en el contexto del proyecto. Esto garantiza que cualquier riesgo emergente se detecte a tiempo y se incluya en el plan de gestión.
- La revisión de los riesgos se llevará a cabo a través de reuniones regulares con los stakeholders involucrados en el proyecto, aquellos que se verán afectados por los riesgos; con el fin de discutir cambios en los riesgos, tomar decisiones y ajustar las estrategias de mitigación según sea necesario.

8.1.4.2. Identificación de riesgos

ID	Nombre
1	Dificultades en la integración de la Web API con la aplicación web
2	Retraso en el desarrollo de uno de los módulos de la API
3	Cambios significativos en los requisitos del proyecto
4	Problemas de compatibilidad de la aplicación web con algún navegador
5	Errores en el análisis del programa a través de consultas Cypher
6	Complejidad técnica por falta de experiencia
7	Problemas con la integración de repositorios externos para importar programas
8	Problemas de rendimiento en la aplicación web
9	Implementación insuficiente de medidas de seguridad
10	Fallos de ciertas funcionalidades inesperados
11	Problemas con la gestión de la base de datos
12	Problemas con la usabilidad de la aplicación web
13	Falta de coherencia entre la Web API y la aplicación web
14	Problemas en la subida de programas simultáneamente
15	Insatisfacción del usuario por falta de conocimiento del uso de la aplicación web

Tabla 107. Identificación de riesgos

8.1.4.3. Registro de riesgos

ID	Nombre	Responsable	Probabilidad	Impacto				Impacto	0,35	Response
				Presup.	Planific.	Alcance	Calidad		Priorización	
1	Dificultades en la integración de la Web API con la aplicación web	Desarrollador de software	Media	Bajo	Medio	Alto	Alto	0,28		Realizar pruebas de integración tempranas y planificar la integración gradual de componentes.
2	Retraso en el desarrollo de uno de los módulos de la API	Desarrollador de software	Media	Medio	Alto	Muy Bajo	Bajo	0,28		Asignar recursos adicionales al módulo y realizar un seguimiento detallado del progreso.
3	Cambios significativos en los requisitos del proyecto	Jefe de Proyecto y Analista	Baja	Medio	Alto	Alto	Medio	0,17		Mantener una comunicación constante con los interesados para comprender y documentar adecuadamente los requisitos.
4	Problemas de compatibilidad de la aplicación web con algún navegador	Desarrollador de software y Arquitecto de software	Media	Bajo	Medio	Medio	Alto	0,28		Realizar pruebas de compatibilidad en varios navegadores durante el desarrollo.
5	Errores en el análisis del programa a través de consultas Cypher	Desarrollador de software y Analista	Baja	Bajo	Medio	Medio	Alto	0,17		Realizar revisiones de código y pruebas rigurosas de diversos análisis antes de su implementación.
6	Complejidad técnica por falta de experiencia	Desarrollador de software y Arquitecto de software	Alta	Medio	Alto	Medio	Medio	0,39		Proporcionar capacitación y recursos adicionales para aumentar los conocimientos del desarrollador.
7	Problemas con la integración de repositorios externos para importar programas	Desarrollador de software	Media	Bajo	Medio	Bajo	Medio	0,15		Definir protocolos de integración claros y realizar seguimiento y pruebas regulares de dicha integración.
8	Problemas de rendimiento en la aplicación web	Desarrollador de software y Arquitecto de software	Media	Medio	Medio	Bajo	Alto	0,28		Realizar pruebas de rendimiento periódicas y optimizaciones durante el desarrollo para garantizar que la aplicación funcione de manera eficiente.
9	Implementación insuficiente de medidas de seguridad	Desarrollador de software	Media	Bajo	Medio	Medio	Alto	0,28		Implementar medidas adecuadas de autenticación y autorización en todo el proyecto.

10	Fallos de ciertas funcionalidades inesperados	Desarrollador de software	Media	Bajo	Bajo	Medio	Medio	0,15		Realizar pruebas unitarias de todos los posibles escenarios de las diferentes funcionalidades
11	Problemas con la gestión de la base de datos	Desarrollador de software y Arquitecto de software	Media	Bajo	Bajo	Medio	Medio	0,15		Incluir un diseño eficiente del esquema de la base de datos, realizar consultas eficientes y considerar la consistencia de los datos.
12	Problemas con la usabilidad de la aplicación web	Desarrollador de software	Baja	Bajo	Bajo	Medio	Alto	0,17		Realizar pruebas de usabilidad y recopilar retroalimentación de los usuarios para mejorar la experiencia de estos.
13	Falta de coherencia entre la Web API y la aplicación web	Desarrollador de software	Baja	Bajo	Bajo	Bajo	Medio	0,09		Mantener una coherencia entre la API y la UI para facilitar el desarrollo y la integración de nuevas funcionalidades
14	Problemas en la subida de programas simultáneamente	Desarrollador de software	Alta	Bajo	Medio	Medio	Alto	0,39		Implementar colas de trabajo para posibilitar la subida de varios programas al mismo tiempo.
15	Insatisfacción del usuario por falta de conocimiento del uso de la aplicación web	Desarrollador de software y Analista	Baja	Bajo	Bajo	Bajo	Medio	0,09		Realizar un manual de usuario que aborde todas las funcionalidades de la aplicación web.

Tabla 108. Registro de riesgos

8.1.5. Presupuesto inicial

Con respecto al presupuesto inicial elaborado, se ha divido en varias partidas según la temática. Se muestran dichas partidas y el presupuesto total detallado y resumido.

8.1.5.1. Presupuesto de costes

Partida 1: Análisis, Dirección y Gestión del proyecto

I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
1			**Análisis, Dirección y Gestión del proyecto**						**6.236,25 €**
	1		Reuniones cada dos semanas con el tutor					960,00 €	
		1	Inés (Jefe de proyecto)	16	horas	60,00 €	960,00 €		
	2		Definición del propósito y la justificación del proyecto					180,00 €	
		1	Inés (Jefe de proyecto)	3	horas	60,00 €	180,00 €		
	3		Definición del objetivo del proyecto					180,00 €	
		1	Inés (Jefe de proyecto)	3	horas	60,00 €	180,00 €		
	4		Definifición del alcance del proyecto					240,00 €	
		1	Inés (Jefe de proyecto)	4	horas	60,00 €	240,00 €		
	5		Estudio del arte					585,00 €	
		1	Inés (Analista)	12	horas	48,75 €	585,00 €		
	6		Familiarización con los lenguajes de programación, el entorno y las bases de datos					210,00 €	
		1	Inés (Desarrollador de Software)	8	horas	26,25 €	210,00 €		
	7		Comprensión ProgQuery					78,75 €	
		1	Inés (Desarrollador de Software)	3	horas	26,25 €	78,75 €		
	8		Definición del sistema					243,75 €	
		1	Inés (Analista)	5	horas	48,75 €	243,75 €		
	9		Identificación de stackholders y roles					146,25 €	
		1	Inés (Analista)	3	horas	48,75 €	146,25 €		
	10		Análisis de requisitos					633,75 €	
		1	Inés (Analista)	13	horas	48,75 €	633,75 €		
	11		Análisis de casos de uso y escenarios					1.072,50 €	
		1	Inés (Analista)	22	horas	48,75 €	1.072,50 €		
	12		Definición de la planificación del proyecto					480,00 €	
		1	Inés (Jefe de proyecto)	8	horas	60,00 €	480,00 €		
	13		Definición del presupuesto del proyecto					300,00 €	
		1	Inés (Jefe de proyecto)	5	horas	60,00 €	300,00 €		
	14		Definición de los riesgos del proyecto					926,25 €	
		1	Inés (Analista)	19	horas	48,75 €	926,25 €		
								TOTAL	**6.236,25 €**

Ilustración 92. Presupuesto inicial I

Partida 2: Diseño del sistema

I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
1			**Diseño del sistema**						**3.255,00 €**
	1		Diseño de la arquitectura del sistema					420,00 €	
		1	Inés (Arquitecto de Software)	8	horas	52,50 €	420,00 €		
	2		Creación de diagramas y modelos de datos					840,00 €	
		1	Inés (Arquitecto de Software)	16	horas	52,50 €	840,00 €		
	3		Diseño de la interfaz de usuario					735,00 €	
		1	Inés (Arquitecto de Software)	14	horas	52,50 €	735,00 €		
	4		Definición de componentes del sistema y sus interacciones					1.260,00 €	
		1	Inés (Arquitecto de Software)	24	horas	52,50 €	1.260,00 €		
								TOTAL	**3.255,00 €**

Ilustración 93. Presupuesto inicial II

Partida 3: Desarrollo del sistema

I1	I2	I3	I4	Descripción	Cantidad	Unidades	Precio	Subtotal (4)	Subtotal (3)	Subtotal (2)	Total
1				**Desarrollo del sistema**							**5.722,50 €**
	1			Desarrollo de la WEB API						4.515,00 €	
		1		Configuración del proyecto y entorno de desarrollo					210,00 €		
			1	Inés (Desarrollador de Software)	8	horas	26,25 €	210,00 €			
		2		Implementación de la gestión de usuarios y autenticación					1.050,00 €		
			1	Inés (Desarrollador de Software)	40	horas	26,25 €	1.050,00 €			
		3		Implementación de la gestión de programas					1.102,50 €		
			1	Inés (Desarrollador de Software)	42	horas	26,25 €	1.102,50 €			
		4		Implementación de la gestión de análisis					1.050,00 €		
			1	Inés (Desarrollador de Software)	40	horas	26,25 €	1.050,00 €			
		5		Implementación de la gestión de resultados					1.102,50 €		
			1	Inés (Desarrollador de Software)	42	horas	26,25 €	1.102,50 €			
	2			Desarrollo de la aplicación web						1.207,50 €	
		1		Inés (Desarrollador de Software)	46	horas	26,25 €		1.207,50 €		
										TOTAL	**5.722,50 €**

Ilustración 94. Presupuesto inicial III

Partida 4: Pruebas y Despliegue

I 1	I 2	I 3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
1			**Pruebas**						**1.627,50 €**
	1		Pruebas unitarias para la gestión de usuarios					262,50 €	
		1	Inés (Desarrollador de Software)	10	horas	26,25 €	262,50 €		
	2		Pruebas unitarias para la gestión de programas					262,50 €	
		1	Inés (Desarrollador de Software)	10	horas	26,25 €	262,50 €		
	3		Pruebas unitarias para la gestión de análisis					262,50 €	
		1	Inés (Desarrollador de Software)	10	horas	26,25 €	262,50 €		
	4		Pruebas unitarias para la gestión de resultados					262,50 €	
		1	Inés (Desarrollador de Software)	10	horas	26,25 €	262,50 €		
	5		Pruebas de integración					367,50 €	
		1	Inés (Desarrollador de Software)	14	horas	26,25 €	367,50 €		
	6		Identificación y solución de errores y problemas					210,00 €	
		1	Inés (Desarrollador de Software)	8	horas	26,25 €	210,00 €		
2			**Despliegue**						**420,00 €**
	1		Configuración del servidor y alojamiento de la aplicación web y la Web API					315,00 €	
		1	Inés (Desarrollador de Software)	12	horas	26,25 €	315,00 €		
	2		Realización de pruebas finales					105,00 €	
		1	Inés (Desarrollador de Software)	4	horas	26,25 €	105,00 €		

TOTAL	**2.047,50 €**

Ilustración 95. Presupuesto inicial IV

Partida 5: Documentación y Cierre del proyecto

I 1	I 2	I 3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
1			**Documentación**						**3.367,50 €**
	1		Realización de manuales de usuario					367,50 €	
		1	Inés (Desarrollador de Software)	14	horas	26,25 €	367,50 €		
	2		Documentación final del proyecto					3.000,00 €	
		1	Inés (Jefe de proyecto)	50	horas	60,00 €	3.000,00 €		
2			**Cierre del proyecto**						**240,00 €**
	1		Revisión final del proyecto					120,00 €	
		1	Inés (Jefe de proyecto)	2	horas	60,00 €	120,00 €		
	2		Reunión de cierre del proyecto con el cliente					120,00 €	
		1	Inés (Jefe de proyecto)	2	horas	60,00 €	120,00 €		

TOTAL	**3.607,50 €**

Ilustración 96. Presupuesto inicial V

Partida 6: Costes indirectos

I 1	I 2	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Total
1		**Costes de producción**					**265,00 €**
	1	Servidor	4	meses	20,00 €	80,00 €	
	2	Portátil	4	meses	31,25 €	125,00 €	
	3	Licencias de desarrollo	4	meses	15,00 €	60,00 €	
2		**Otros**					**140,00 €**
	1	Consumo eléctrico	4	meses	35,00 €	140,00 €	

TOTAL	**405,00 €**

Ilustración 97. Presupuesto inicial VI

Presupuesto detallado del proyecto

Presupuesto del proyecto			
Partida	***Item***	***Partida***	***Importe***
1		**Análisis, Dirección y Gestión del proyecto**	**6.236,25 €**
2		**Diseño del sistema**	**3.255,00 €**
3		**Desarrollo del sistema**	**5.722,50 €**
4		**Pruebas y Despliegue**	**2.047,50 €**
	1	Pruebas	1.627,50 €
	2	Despliegue	420,00 €
5		**Documentación y Cierre**	**3.607,50 €**
	1	Documentación	3.367,50 €
	2	Cierre del proyecto	240,00 €
6		**Costes indirectos**	**405,00 €**
	1	Costes de producción	265,00 €
	2	Otros	140,00 €
		TOTAL PROYECTO	**21.273,75 €**

Ilustración 98. Presupuesto inicial VII

Presupuesto general del proyecto

Presupuesto de cliente		
Cod.	***Partida***	***Total***
1	**Análisis, Dirección y Gestión del proyecto**	**6.236,25 €**
2	**Diseño del sistema**	**3.255,00 €**
3	**Desarrollo del sistema**	**5.722,50 €**
4	**Pruebas y Despliegue**	**2.047,50 €**
5	**Documentación y Cierre**	**3.607,50 €**
6	**Costes indirectos**	**405,00 €**
	TOTAL PROYECTO	**21.273,75 €**

Ilustración 99. Presupuesto inicial VIII

8.2. Ejecución del proyecto

8.2.1. Plan de seguimiento de planificación

El plan de seguimiento de la planificación del proyecto se basa en la realización de reuniones periódicas y en el monitoreo detallado de las tareas y actividades planificadas. A continuación, se presenta un resumen de las actividades y los hitos más relevantes.

Reuniones Periódicas

Se planifican reuniones quincenales con el tutor académico para revisar el progreso del proyecto, discutir cualquier problema encontrado y ajustar la planificación según sea necesario. Las fechas programadas para estas reuniones son las siguientes:

1. Reunión 1: Lunes 08/01/24
2. Reunión 2: Lunes 22/01/24
3. Reunión 3: Lunes 05/02/24

4. Reunión 4: Lunes 19/02/24
5. Reunión 5: Lunes 04/03/24
6. Reunión 6: Lunes 18/03/24
7. Reunión 7: Lunes 01/04/24
8. Reunión 8: Lunes 15/04/24
9. Reunión 9: Lunes 29/04/24
10. Reunión 10: Lunes 27/05/24

Estas reuniones serán clave para asegurar que el proyecto se mantenga en el camino correcto y que se puedan tomar decisiones informadas y oportunas.

Fases del Proyecto

El proyecto está dividido en varias fases, cada una con sus tareas específicas y plazos de entrega. A continuación, se presenta un resumen de las principales fases y actividades:

1. **Análisis, Dirección y Gestión del Proyecto** (08/01/24 - 29/01/24)
    - Definir propósito, objetivos y alcance del proyecto.
    - Estudio del estado del arte.
    - Análisis de requisitos y casos de uso.
    - Planificación, presupuesto y análisis de riesgos.
    - **Entrega de Planificación y Presupuesto del Proyecto**: 31/01/24
2. **Diseño del Sistema** (01/02/24 - 12/02/24)
    - Diseño de la arquitectura del sistema.
    - Creación de diagramas y modelos de datos.
    - Diseño de la interfaz de usuario y definición de componentes.
3. **Desarrollo del Software** (15/02/24 - 02/04/24)
    - Desarrollo de la Web API y la aplicación web.
    - Implementación de funcionalidades clave como gestión de usuarios, programas, análisis y resultados.
4. **Pruebas** (15/04/24 - 29/04/24)
    - Pruebas unitarias y de integración.
    - Identificación y solución de errores.
5. **Despliegue** (02/05/24 - 03/05/24)
    - Configuración del servidor y alojamiento de la aplicación.

- Pruebas finales antes del despliegue.

6. **Documentación** (27/05/24 - 06/06/24)
   - Elaboración de manuales de usuario.
   - Documentación final del proyecto.
7. **Cierre del Proyecto** (07/06/24)
   - Revisión final del proyecto.
   - Reunión de cierre del proyecto.
8. **Entrega Final del Proyecto** (04/07/24)

Cada fase y actividad tiene una duración específica y se supervisará mediante reuniones y revisiones periódicas para asegurar el cumplimiento de los plazos y la calidad del trabajo realizado.

Control de Incidencias

Se mantendrá una bitácora de incidencias del proyecto para registrar y hacer seguimiento de cualquier problema o desviación que ocurra durante el desarrollo del proyecto. Cada incidencia será documentada con su descripción, fecha de detección, impacto, responsable y estado de resolución.

Este enfoque de seguimiento detallado y reuniones periódicas garantizará que el proyecto se ejecute de manera efectiva y que cualquier problema se aborde de manera oportuna para minimizar su impacto en el desarrollo del proyecto.

8.2.2. Bitácora de incidencias del proyecto

Durante la ejecución del proyecto, se han registrado las siguientes incidencias. La bitácora detalla cada incidencia, su impacto en el proyecto y las medidas tomadas para solucionarlas y minimizar su efecto.

Descripción	Impacto	Acciones tomadas
Cambios en los requisitos del proyecto a mitad del desarrollo.	Medio	Revisión y actualización del plan del proyecto y del desarrollo.
Falta de experiencia técnica.	Bajo	Adquisición de los conocimientos necesarios.
Subida de varios programas simultáneamente.	Medio	Añadida una cola de trabajo para manejar las subidas simultáneas de programas, permitiendo una carga eficiente.
Retroalimentación de los usuarios sobre la usabilidad de la aplicación web.	Bajo	Realización de pruebas de usabilidad y ajustes en la interfaz de usuario basados en la retroalimentación recibida.

Tabla 109. Incidencias registradas

8.2.3. Riesgos

8.2.3.1. Datos del primer riesgo identificado

ID: 1

Nombre: Dificultades en la integración de la Web API con la aplicación web

Descripción:

La integración de la API web con la aplicación web es un aspecto crítico para el proyecto. El éxito del proyecto depende en gran medida de la implementación exitosa de esta integración. Sólo si esta integración tiene éxito sin costos significativamente mayores, el presupuesto del proyecto se mantendrá dentro de límites aceptables.

El costo inicial para la integración de la API web se ha estimado y se encuentra dentro de los parámetros presupuestarios. Sin embargo, cualquier desviación significativa de estos costes estimados podría poner en peligro la viabilidad financiera del proyecto.

La gestión eficaz de este riesgo es esencial para garantizar que la integración se produzca dentro de los parámetros presupuestarios y permita el éxito del proyecto. Las medidas proactivas y el monitoreo constante son esenciales para cerrar brechas significativas en los costos de integración.

Categoría(s) de riesgo: Requisitos, Tecnologías, Planificación

Causas del Riesgo:

- Si la Web API y la Aplicación Web utilizan tecnologías incompatibles, podría surgir dificultades en la integración.
- Si el desarrollador carece de experiencia en la integración de sistemas, esto podría contribuir al riesgo.
- La falta de rendimiento de la Web API o la Aplicación Web podría generar dificultades en su integración.

Probabilidad	Impacto				Impacto Total	Respuestas
	Presupuesto	Planificación	Alcance	Calidad		
Media	Bajo	Medio	Alto	Alto	0,28	Realizar pruebas de integración tempranas y planificar la integración gradual de componentes.

Riesgos derivados de este: Riesgo de retraso en la planificación del proyecto. Riesgo de que el proyecto sea inviable.

Riesgo residual: Problemas de compatibilidad de la aplicación web con algún navegador.

Plan de Contingencia:

1. *Gestión de Cambios*: No se permitirán cambios en la integración sin una evaluación adecuada por parte del desarrollador.

   a. En caso de cambios, estos deben ser evaluados y documentados. Se establecerá un límite de 16 horas de trabajo para la evaluación y resolución de cambios no planificados (16h * 26,25€/h = 420€).

2. *Formación del Desarrollador*: Se proporcionará formación al desarrollador en caso de que carezca de experiencia en integraciones similares.

   a. Se estima un costo de formación de 5 horas para el desarrollador (5h * 26,25€/h = 131,25€).

3. *Compatibilidad del Entorno*: Antes de la integración, se llevará a cabo una revisión exhaustiva del entorno de explotación para asegurar la compatibilidad con la Web API.

   a. Si surgen problemas, se realizará un análisis detallado para estimar si el módulo se puede adaptar en un plazo de 16 horas (16h * 26,25€/h = 420€).

4. *Planificación y Entrega*: a. Se establecerá un plan detallado para la entrega y aceptación de la integración una vez terminada. b. En caso de errores o problemas en la entrega, se limitarán a un máximo de 16 horas de trabajo (16h * 26,25€/h = 420€).

Presupuesto para contingencias:

Coste 1: Probabilidad Media – 420€ * 0,5 = 210€

Coste 2: Probabilidad Baja – 131,25€ * 0,3 = 39,38€

Coste 3: Probabilidad Baja – 420€ * 0,3 = 126€

Coste 4: Probabilidad Baja – 420€ * 0,3 = 126€

Presupuesto total: *501,38€*

Planificación temporal de las contingencias:

La gestión de cambios y la evaluación de impacto en la integración serán una preocupación constante durante todo el desarrollo del proyecto.

La formación del desarrollador debe realizarse al inicio del proyecto.

La revisión y estudio del entorno tecnológico para identificar incompatibilidades potenciales se llevará a cabo en las etapas iniciales del proyecto.

En caso de que la aceptación de la integración se realice con errores, se deberá solucionar con la mayor brevedad para obtener una aceptación total.

	Valor del riesgo				
INDICADORES	***Muy alto***	***Alto***	***Medio***	***Bajo***	***Muy Bajo***
Indicador 1: Número de intentos de integración fallidos.	>10	8	5	2	0
Indicador 2: Cambios en los requisitos de integración.	>16	12	8	4	0

Indicador 3: Tiempo de incompatibilidad del entorno.	>16	12	8	4	0
Indicador 4: Nivel de experiencia del desarrollador.	<10,00%	30,00%	50,00%	70,00%	>90,00%
Indicador 5: Consumo de presupuesto de contingencia.	>80,00%	60,00%	40,00%	20,00%	<10,00%

Monitorización:

Indicadores:

Indicador 1: Número de intentos de integración fallidos. • Se refiere al total de veces que se ha intentado integrar la Web API con la aplicación web, sin éxito.	**Evaluación:** Se evalúa en tiempo real durante el proceso de integración. Cada intento fallido se registra y suma al indicador.
Indicador 2: Cambios en los requisitos de integración. • Se mide en horas de trabajo no planificadas que surgen debido a cambios en los requisitos de integración, los cuales no se han incluido en la planificación inicial.	**Evaluación:** Se evalúa cada vez que se identifica un cambio en los requisitos de integración que no estaba previamente planificado.
Indicador 3: Tiempo de incompatibilidad del entorno. • Refleja el tiempo total en horas que se dedica a abordar incompatibilidades en el entorno tecnológico.	**Evaluación:** Se evalúa durante el proceso de estudio del entorno tecnológico y cada vez que se abordan incompatibilidades.
Indicador 4: Nivel de experiencia del desarrollador. • Mide el nivel de experiencia del desarrollador en proyectos de integración similares.	**Evaluación:** Se evalúa al inicio del proyecto al analizar las experiencias previas del desarrollador.
Indicador 5: Consumo de presupuesto de contingencia. • Refleja el importe utilizado de la reserva de contingencia destinada a abordar riesgos no previstos.	**Evaluación:** Se evalúa cada vez que se necesita asignar dinero de la reserva por contingencias para abordar los riesgos relacionados con la integración.

8.2.3.2. Datos del segundo riesgo identificado

ID: 2

Nombre: Retraso en el desarrollo de uno de los módulos de la API

Descripción:

El retraso en el desarrollo de uno de los módulos de la API podría comprometer significativamente el cronograma general del proyecto. La entrega puntual de cada módulo es crucial para cumplir con los plazos establecidos y garantizar la finalización exitosa de este. Cualquier retraso en el desarrollo de un módulo puede generar una reacción en cadena

que afecte a las etapas posteriores del proyecto, resultando en la extensión de los plazos de entrega y posibles costos adicionales.

El plan del proyecto se ha elaborado con márgenes de tiempo específicos para cada fase del desarrollo. Sin embargo, cualquier desviación en el cronograma previsto para uno de los módulos puede desestabilizar el plan general, comprometiendo los plazos y el presupuesto acordados. La gestión eficaz y la supervisión constante del progreso son esenciales para mitigar este riesgo y asegurar la finalización del proyecto dentro del tiempo y presupuesto previstos.

Categoría(s) de riesgo: Planificación, Ejecución, Requisitos

Causas del Riesgo:

- Problemas técnicos imprevistos en el desarrollo del módulo, como errores complejos de codificación o dificultades en la integración de tecnologías.
- Falta de recursos, como un desarrollador con la experiencia necesaria para resolver problemas técnicos avanzados.
- Requisitos mal definidos o cambios frecuentes en los requisitos que llevan a retrabajos y ajustes continuos.
- Dependencias con otros módulos o componentes del sistema que no están listos a tiempo, afectando el progreso del módulo en cuestión.

Probabilidad	Impacto				Impacto Total	Respuestas
	Presupuesto	Planificación	Alcance	Calidad		
Media	Medio	Alto	Muy Bajo	Bajo	0,28	Asignar recursos adicionales al módulo y realizar un seguimiento detallado del progreso.

Riesgos derivados de este: Riesgo de incumplimiento de los plazos del proyecto. Riesgo de aumento de los costos del proyecto.

Riesgo residual: Problemas de calidad del código debido a prisas en el desarrollo final.

Plan de Contingencia:

1. *Revisión y Ajuste del Cronograma*: Se realizarán revisiones periódicas del cronograma para identificar posibles retrasos y ajustar las fechas de entrega de forma proactiva.
    a. En caso de retraso, se permitirá una extensión de hasta 16 horas de trabajo para completar el módulo sin afectar significativamente el cronograma global (16h * 26,25€/h = 420€).
2. *Optimización del Trabajo del Desarrollador*: Se implementarán técnicas de gestión del tiempo y metodologías ágiles para optimizar el rendimiento del desarrollador.
    a. Se reservarán 5 horas para sesiones de planificación y revisión del progreso semanal (5h * 26,25€/h = 131,25€).

3. *Capacitación y Soporte Técnico*: Se proporcionará capacitación adicional al desarrollador en áreas donde se identifiquen carencias de conocimientos.
    a. El costo de capacitación se estima en 5 horas (5h * 26,25€/h = 131,25€).
4. *Mejora de la Comunicación*: a. Se implementarán reuniones diarias de actualización del progreso para asegurar que el desarrollador está al tanto del estado del proyecto y cualquier cambio en los requisitos.
    a. El tiempo invertido en estas reuniones será de 1 hora diaria durante 4 semanas (1h * 5 * 4 * 26,25€/h = 525€).

Presupuesto para contingencias:

Coste 1: Probabilidad Media – 420€ * 0,5 = 210€

Coste 2: Probabilidad Media – 131,25€ * 0,5 = 65,63€

Coste 3: Probabilidad Baja – 131,25€ * 0,3 = 39,38€

Coste 4: Probabilidad Media – 525€ * 0,5 = 262,5€

Presupuesto total: *577,51€*

Planificación temporal de las contingencias:

La revisión y ajuste del cronograma será una preocupación constante durante todo el desarrollo del proyecto.

La implementación de técnicas de optimización del trabajo del desarrollador debe realizarse al inicio del proyecto y mantenerse durante todo el desarrollo.

La capacitación del desarrollador debe realizarse al inicio del proyecto, tan pronto como se identifiquen las carencias de conocimientos.

Las reuniones diarias de actualización del progreso comenzarán desde el inicio del proyecto y se mantendrán durante todo el desarrollo para asegurar una comunicación fluida y oportuna.

Monitorización:

	Valor del riesgo				
INDICADORES	***Muy alto***	***Alto***	***Medio***	***Bajo***	***Muy Bajo***
Indicador 1: Retrasos acumulados en el cronograma.	>20	15	10	5	0
Indicador 2: Número de errores críticos en el módulo.	>16	12	8	4	0
Indicador 3: Frecuencia de cambios en los requisitos.	>50,00%	40,00%	30,00%	20,00%	<10,00%
Indicador 4: Nivel de experiencia del desarrollador.	<10,00%	30,00%	50,00%	70,00%	>90,00%
Indicador 5: Consumo de presupuesto de contingencia.	>80,00%	60,00%	40,00%	20,00%	<10,00%

Indicadores:

Indicador 1: Retrasos acumulados en el cronograma. • Se refiere al total de horas de retraso acumuladas en comparación con el cronograma inicial.	**Evaluación:** Se evalúa semanalmente, comparando el progreso real con el planificado.
Indicador 2: Número de errores críticos en el módulo. • Se refiere a la cantidad de errores críticos encontrados en el módulo que afectan su funcionalidad básica.	**Evaluación:** Se evalúa continuamente durante la fase de pruebas y desarrollo.
Indicador 3: Frecuencia de cambios en los requisitos. • Se mide por la cantidad de veces que se han solicitado cambios en los requisitos del módulo durante el desarrollo.	**Evaluación:** Se evalúa cada vez que se identifica un cambio en los requisitos que no estaba previamente planificado.
Indicador 4: Nivel de experiencia del desarrollador. • Mide el nivel de experiencia del desarrollador en proyectos de integración similares.	**Evaluación:** Se evalúa al inicio del proyecto al analizar las experiencias previas del desarrollador.
Indicador 5: Consumo de presupuesto de contingencia. • Refleja el importe utilizado de la reserva de contingencia destinada a abordar riesgos no previstos.	**Evaluación:** Se evalúa cada vez que se necesita asignar dinero de la reserva por contingencias para abordar los riesgos relacionados con el desarrollo del módulo.

8.2.3.3. Datos del tercer riesgo identificado

ID: 3

Nombre: Cambios significativos en los requisitos del proyecto

Descripción:

Los cambios significativos en los requisitos del proyecto representan un riesgo importante que puede afectar negativamente tanto al cronograma como al presupuesto. Estos cambios pueden surgir debido a una variedad de factores, como malentendidos en la comunicación, cambios en las necesidades del cliente o descubrimientos durante el proceso de desarrollo. Si no se gestionan adecuadamente, estos cambios pueden llevar a retrasos en la entrega, aumento de costos y, en última instancia, insatisfacción del cliente.

Categoría(s) de riesgo: Requisitos, Planificación, Costos

Causas del Riesgo:

- Comunicación inadecuada entre el equipo del proyecto y los stakeholders.
- Falta de claridad en los requisitos iniciales del proyecto.
- Cambios en las necesidades del cliente durante el desarrollo.

- Complejidad técnica no anticipada al inicio del proyecto.
- Limitaciones de recursos, como personal o tecnología.

Probabilidad	Impacto				Impacto Total	Respuestas
	Presupuesto	Planificación	Alcance	Calidad		
Baja	Medio	Alto	Alto	Medio	0,17	Mantener una comunicación constante con los interesados para comprender y documentar adecuadamente los requisitos.

Riesgos derivados de este: Riesgo de retraso en la entrega del proyecto. Riesgo de aumento de los costos del proyecto. Riesgo de insatisfacción del cliente.

Riesgo residual: Desalineación entre los requisitos finales del proyecto y las expectativas del cliente.

Plan de Contingencia:

1. *Evaluación y Análisis de los Cambios*: Se designará un equipo específico para realizar una evaluación exhaustiva de los cambios propuestos, incluyendo el jefe de proyecto y el analista.
    a. Se establecerá un límite de 16 horas de trabajo para la evaluación y análisis de cada cambio propuesto (8h * 60€/h = 480€ para el jefe de proyecto, 8h * 48,75€/h = 390€ para el analista).
2. *Priorización de Requisitos*: Se asignará un responsable para priorizar los nuevos requisitos según su impacto en el proyecto.
    a. En caso de cambios significativos, se reservará un tiempo adicional de 16 horas para revisar y ajustar la planificación del proyecto (8h * 60€/h = 480€ para el jefe de proyecto, 8h * 48,75€/h = 390€ para el analista).
3. *Comunicación Efectiva*: Se establecerá un protocolo de comunicación claro para informar a todas las partes interesadas sobre los cambios en los requisitos y su impacto en el proyecto.
    a. Se designará un tiempo adicional de 8 horas para la comunicación y actualización de las partes interesadas (4h * 60€/h = 240€ para el jefe de proyecto, 4h * 48,75€/h = 195€ para el analista).
4. *Asignación de Recursos*: Se garantizará la disponibilidad de recursos adicionales para abordar los cambios significativos en los requisitos.
    a. Se reservará un presupuesto específico para cubrir los costos adicionales asociados con la gestión de cambios (8h * 60€/h = 480€ para el jefe de proyecto, 8h * 48,75€/h = 390€ para el analista).
    b. Coste 4: Probabilidad Baja – 870€ * 0,3 = 261€

Presupuesto para contingencias:

Coste 1: Probabilidad Media – 870€ * 0,5 = 435€

Coste 2: Probabilidad Media – 870€ * 0,5 = 435€

Coste 3: Probabilidad Baja – 405€ * 0,3 = 121,5€

Coste 4: Probabilidad Baja – 870€ * 0,3 = 261€

Presupuesto total: 1252,5€

Planificación temporal de las contingencias:

La evaluación y análisis de los cambios se realizará tan pronto como se identifiquen, con reuniones programadas según sea necesario.

La priorización de requisitos y la asignación de recursos se llevarán a cabo de manera oportuna para minimizar cualquier impacto en el cronograma del proyecto.

La comunicación efectiva con todas las partes interesadas será continua a lo largo del proyecto para garantizar la transparencia y la alineación en relación con los cambios en los requisitos.

Monitorización:

	Valor del riesgo				
INDICADORES	***Muy alto***	***Alto***	***Medio***	***Bajo***	***Muy Bajo***
Indicador 1: Frecuencia de cambios significativos en los requisitos.	>20,00%	15,00%	10,00%	5,00%	0,00%
Indicador 2: Impacto en el cronograma del proyecto.	>20	15	10	5	0
Indicador 3: Impacto en el presupuesto del proyecto.	>20,00%	15,00%	10,00%	5,00%	0,00%
Indicador 4: Nivel de comunicación con los stakeholders.	1	2	3	4	5

Indicadores:

Indicador 1: Frecuencia de cambios significativos en los requisitos. • Refleja la frecuencia con la que se producen cambios significativos en los requisitos del proyecto.	**Evaluación:** Se realiza durante el desarrollo del proyecto y se actualiza periódicamente.
Indicador 2: Impacto en el cronograma del proyecto. • Cuantifica el impacto que tienen los cambios en los requisitos en el cronograma planificado del proyecto.	**Evaluación:** Se realiza durante la gestión activa del proyecto y se actualiza con cada cambio importante en los requisitos.
Indicador 3: Impacto en el presupuesto del proyecto. • Indica el impacto financiero de los cambios en los requisitos en el presupuesto total del proyecto.	**Evaluación:** Se realiza durante la gestión financiera del proyecto y se actualiza con cada cambio importante en los requisitos.

Indicador 4: Nivel de comunicación con los stakeholders. • Evalúa la eficacia de la comunicación con los stakeholders del proyecto sobre los cambios en los requisitos.	**Evaluación:** Se basa en la retroalimentación de los stakeholders y en la percepción del equipo de proyecto sobre la comunicación.

8.2.3.4. Datos del cuarto riesgo identificado

ID: 10

Nombre: Fallos de ciertas funcionalidades inesperados

Descripción:

Los fallos inesperados de ciertas funcionalidades representan un riesgo potencial para el proyecto, ya que pueden afectar negativamente la calidad del producto final y la satisfacción del usuario. Estos fallos pueden surgir durante el desarrollo del software o después de su implementación, y pueden ser causados por diversos factores, como errores en el código, falta de pruebas adecuadas, o interacciones inesperadas entre diferentes componentes del sistema.

El impacto de estos fallos puede variar desde leves molestias hasta graves interrupciones en el uso del producto, dependiendo de la naturaleza de las funcionalidades afectadas y la criticidad del sistema. Los fallos graves pueden provocar la pérdida de datos, la corrupción del sistema, o incluso la indisponibilidad total del servicio, lo que podría tener repercusiones significativas en la reputación de la empresa y en la viabilidad del proyecto.

Categoría(s) de riesgo: Desarrollo de Software, Calidad

Causas del Riesgo:

- Errores en el código durante el desarrollo.
- Falta de pruebas exhaustivas para identificar posibles fallos.
- Interacciones inesperadas entre diferentes componentes del sistema.
- Cambios en los requisitos del cliente no detectados durante el desarrollo.

Probabilidad	Impacto				Impacto Total	Respuestas
	Presupuesto	Planificación	Alcance	Calidad		
Media	Bajo	Bajo	Medio	Medio	0,15	Realizar pruebas unitarias de todos los posibles escenarios de las diferentes funcionalidades

Riesgos derivados de este: Pérdida de confianza del usuario en el producto. Retrasos en la entrega del proyecto debido a la corrección de fallos. Costos adicionales asociados con el mantenimiento y la resolución de problemas.

Riesgo residual: Posible pérdida de usuarios o clientes debido a la insatisfacción con la calidad del producto.

Plan de Contingencia:

1. *Pruebas Exhaustivas de Funcionalidades*: Se asignará tiempo adicional al desarrollador responsable para realizar pruebas exhaustivas de todas las funcionalidades del sistema.
    a. Se reservará un total de 20 horas para llevar a cabo estas pruebas adicionales (20h * 26,25€/h = 525€).
2. *Revisión de Código Regular*: Se implementará un proceso de revisión de código regular para detectar y corregir posibles errores antes de que afecten al usuario final.
    a. Se reservará un total de 20 horas para la revisión de código (20h * 26,25€/h = 525€).
3. *Comunicación Proactiva con el Cliente*: Se mantendrá una comunicación abierta y proactiva con el cliente para identificar cualquier cambio en los requisitos que pueda afectar la calidad del producto.
    a. Se asignará un tiempo adicional de 16 horas para la comunicación con el cliente (16h * 26,25€/h = 420€).
4. *Seguimiento de Errores*: Se establecerá un sistema de seguimiento de errores para registrar y priorizar los fallos detectados durante las pruebas y su posterior corrección.
    a. Se reservará un total de 16 horas para el seguimiento y resolución de errores (16h * 26,25€/h = 420€).

Presupuesto para contingencias:

Coste 1: Probabilidad Media – 525€ * 0,5 = 262,5€

Coste 2: Probabilidad Baja – 525€ * 0,3 = 157,5€

Coste 3: Probabilidad Baja – 420€ * 0,3 = 126€

Coste 4: Probabilidad Media – 420€ * 0,5 = 210€

Presupuesto total: *756€*

Planificación temporal de las contingencias:

Se asignará un período de tiempo específico al desarrollador responsable al finalizar cada ciclo de desarrollo para realizar pruebas exhaustivas de todas las funcionalidades del sistema.

Se implementará un proceso de revisión de código regular al finalizar cada semana laboral para detectar y corregir posibles errores antes de que afecten al usuario final.

Se mantendrá una comunicación abierta y proactiva con el cliente a lo largo del proyecto para identificar cualquier cambio en los requisitos que pueda afectar la calidad del producto.

Se establecerá un sistema de seguimiento de errores para registrar y priorizar los fallos detectados durante las pruebas y su posterior corrección.

Monitorización:

	Valor del riesgo				
INDICADORES	***Muy alto***	***Alto***	***Medio***	***Bajo***	***Muy Bajo***
Indicador 1: Número de fallos en un cierto tiempo.	>10	8	5	2	0
Indicador 2: Impacto en la experiencia del usuario.	5	4	3	2	1
Indicador 3: Nivel de cobertura de pruebas.	<30,00%	50,00%	70,00%	90,00%	>90,00%

Indicadores:

Indicador 1: Número de fallos en un cierto tiempo. • Refleja la frecuencia con la que se producen fallos en ciertas funcionalidades del proyecto.	**Evaluación:** Se realiza durante las pruebas del sistema y las pruebas de usuario, registrando cada fallo detectado.
Indicador 2: Impacto en la experiencia del usuario. • Evalúa el impacto que tienen los fallos de funcionalidades en la experiencia general del usuario.	**Evaluación:** Se basa en la gravedad y frecuencia de los fallos reportados por los usuarios o identificados durante las pruebas.
Indicador 3: Nivel de cobertura de pruebas. • Indica el porcentaje de funcionalidades del sistema que han sido probadas con respecto al total de funcionalidades.	**Evaluación:** Se realiza durante la planificación y ejecución de pruebas, registrando el alcance de las pruebas realizadas.

8.2.3.5. Datos del primer riesgo identificado

ID: 15

Nombre: Insatisfacción del usuario por falta de conocimiento del uso de la aplicación web

Descripción:

La insatisfacción del usuario debido a la falta de comprensión sobre cómo utilizar la aplicación web es un riesgo que puede tener un impacto significativo en la percepción del producto y la satisfacción del cliente. Este riesgo puede surgir cuando la interfaz de usuario es confusa o poco intuitiva, lo que dificulta que los usuarios naveguen y utilicen la aplicación de manera efectiva. Además, la falta de documentación clara y detallada sobre el funcionamiento de la aplicación puede aumentar la frustración del usuario al intentar comprender sus características y funcionalidades. Asimismo, la ausencia de capacitación adecuada para el usuario final puede llevar a una experiencia deficiente y generar una percepción negativa de la aplicación.

Categoría(s) de riesgo: Experiencia del usuario, Usabilidad

Causas del Riesgo:

- Interfaz de usuario confusa o poco intuitiva.

- Documentación insuficiente o poco clara sobre el uso de la aplicación.
- Falta de capacitación para el usuario final.

Probabilidad	Impacto				Impacto Total	Respuestas
	Presupuesto	Planificación	Alcance	Calidad		
Baja	Bajo	Bajo	Bajo	Medio	0,09	Realizar un manual de usuario que aborde todas las funcionalidades de la aplicación web.

Riesgos derivados de este: Aumento de solicitudes de soporte debido a la dificultad de los usuarios para utilizar la aplicación. Posible pérdida de usuarios y daño a la reputación de la empresa debido a la insatisfacción del usuario.

Riesgo residual: Persistencia de problemas de usabilidad a pesar de los esfuerzos de mejora.

Plan de Contingencia:

1. *Mejora de la Interfaz de Usuario*: Se asignará un tiempo adicional al desarrollador y al analista para revisar y mejorar la interfaz de usuario, haciéndola más intuitiva y fácil de usar.
    a. Se reservará un total de 30 horas para esta tarea (30h * 26,25€/h = 787,50€).
2. *Elaboración de Documentación Detallada*: Se dedicará tiempo a la elaboración de documentación detallada que explique claramente el uso de la aplicación web.
    a. Se reservará un total de 20 horas para esta tarea (20h * 26,25€/h = 525€).
3. *Capacitación del Usuario Final*: Se organizarán sesiones de capacitación para el usuario final, donde se proporcionará orientación sobre cómo utilizar la aplicación de manera efectiva.
    a. Se reservará un total de 16 horas para estas sesiones (16h * 26,25€/h = 420€).

Presupuesto para contingencias:

Coste 1: Probabilidad Media – 787,5€ * 0,5 = 393,75€

Coste 2: Probabilidad Media – 525€ * 0,5 = 262,5€

Coste 3: Probabilidad Baja – 420€ * 0,3 = 126€

Presupuesto total: *782,25€*

Planificación temporal de las contingencias:

La mejora de la interfaz de usuario se llevará a cabo en las primeras etapas del desarrollo, con revisiones continuas a lo largo del proceso.

La elaboración de documentación detallada se realizará simultáneamente con el desarrollo de la aplicación y estará lista antes del lanzamiento.

Las sesiones de capacitación para el usuario final se programarán cerca de la finalización del desarrollo, antes del lanzamiento oficial de la aplicación.

Monitorización:

	Valor del riesgo				
INDICADORES	***Muy alto***	***Alto***	***Medio***	***Bajo***	***Muy Bajo***
Indicador 1: Número de consultas de soporte.	>20	15	10	5	0
Indicador 2: Tasa de abandono de usuarios.	>18,00%	14,00%	10,00%	6,00%	<2,00%
Indicador 3: Comentarios negativos de los usuarios.	>20	15	10	5	0

Indicadores:

Indicador 1: Número de consultas de soporte. • Registra la cantidad de consultas de soporte recibidas relacionadas con la dificultad de los usuarios para utilizar la aplicación correctamente.	**Evaluación:** Un aumento en las consultas de soporte indica una mayor dificultad de los usuarios para utilizar la aplicación, lo que aumenta el riesgo de insatisfacción.
Indicador 2: Tasa de abandono de usuarios. • Mide el porcentaje de usuarios que abandonan la aplicación después de utilizarla por primera vez debido a problemas de usabilidad.	**Evaluación:** Una tasa de abandono más alta indica que más usuarios encuentran la aplicación difícil de usar, lo que aumenta el riesgo de insatisfacción.
Indicador 3: Comentarios negativos de los usuarios. • Recopila los comentarios y reseñas de los usuarios sobre la aplicación, especialmente aquellos que expresan confusión sobre cómo usarla.	**Evaluación:** Una alta frecuencia de comentarios negativos indica una insatisfacción generalizada entre los usuarios, aumentando el riesgo de insatisfacción del usuario.

8.3. Cierre del proyecto

8.3.1. Planificación final

En este apartado, se muestra la planificación final que se ha llevado a cabo, teniendo las mismas consideraciones que en la planificación inicial.

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
1		**Realizar una reunión cada dos semanas con el tutor**	**94 días**	**lun 08/01/24**	**lun 27/05/24**	
2		Realizar una reunión ca	2 horas	lun 08/01/24	lun 08/01/24	InésJP
3		Realizar una reunión ca	2 horas	lun 22/01/24	lun 22/01/24	InésJP
4		Realizar una reunión ca	2 horas	lun 05/02/24	lun 05/02/24	InésJP
5		Realizar una reunión ca	2 horas	lun 19/02/24	lun 19/02/24	InésJP
6		Realizar una reunión ca	2 horas	lun 04/03/24	lun 04/03/24	InésJP
7		Realizar una reunión ca	2 horas	lun 18/03/24	lun 18/03/24	InésJP
8		Realizar una reunión ca	2 horas	lun 01/04/24	lun 01/04/24	InésJP
9		Realizar una reunión ca	2 horas	lun 15/04/24	lun 15/04/24	InésJP
10		Realizar una reunión ca	2 horas	lun 29/04/24	lun 29/04/24	InésJP
11		Realizar una reunión ca	2 horas	lun 27/05/24	lun 27/05/24	InésJP

Ilustración 100. Reuniones con el tutor

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
12		**Análisis, Dirección y Gestión del proyecto**	**14 días**	**lun 08/01/24**	**lun 29/01/24**	
13		Definir el propósito y la justificación del proyecto	3 horas	lun 08/01/24	lun 08/01/24	InésJP
14		Definir el objetivo del proyecto	3 horas	lun 08/01/24	lun 08/01/24	InésJP
15		Definir el alcance del proyecto	3 horas	mar 09/01/24	mar 09/01/24	InésJP
16		Estudio del arte	13 horas	mar 09/01/24	jue 11/01/24	InésA
17		Familiarizarse con los lenguajes de programación, el entorno y las bases de datos	8 horas	jue 11/01/24	jue 11/01/24	InésDS
18		Comprender ProgQuery	3 horas	vie 12/01/24	vie 12/01/24	InésDS
19		Definir el sistema	5 horas	vie 12/01/24	vie 12/01/24	InésA
20		Identificar los stackholders y definir sus roles	1 hora	lun 15/01/24	lun 15/01/24	InésA
21		Realizar un análisis de requisitos	13 horas	lun 15/01/24	mar 16/01/24	InésA
22		Realizar un análisis de casos de uso y sus escenarios	24 horas	mar 16/01/24	lun 22/01/24	InésA
23		Definir la planificación del proyecto	8 horas	mar 23/01/24	mar 23/01/24	InésJP
24		Definir el presupuesto del proyecto	4 horas	mié 24/01/24	mié 24/01/24	InésJP
25		Definir los riesgos del proyecto	20 horas	mié 24/01/24	lun 29/01/24	InésA
26		Entrega de la Planificación del Proyecto	0 días	mié 31/01/24	mié 31/01/24	
27		Entrega del Presupuesto del Proyecto	0 días	mié 31/01/24	mié 31/01/24	

Ilustración 101. Tareas de análisis, dirección y gestión del proyecto e hitos

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
28		**Diseño del Sistema**	**8 días**	**jue 01/02/24**	**lun 12/02/24**	
29		Diseñar la arquitectura del sistema	8 horas	jue 01/02/24	jue 01/02/24	InésAS
30		Crear diagramas y modelos de datos	14 horas	vie 02/02/24	lun 05/02/24	InésAS
31		Diseñar la interfaz de usuario	16 horas	mar 06/02/24	mié 07/02/24	InésAS
32		Definir los componentes del sitema y sus interacciones	24 horas	jue 08/02/24	lun 12/02/24	InésAS

Ilustración 102. Tareas de diseño del sistema

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
33		**Desarrollo del Software**	**29 días**	**jue 02/15/24**	**mar 04/02/24**	
34		**Desarrollar la Web API**	**22 días**	**jue 02/15/24**	**vie 03/15/24**	
35		Configurar proyecto y entorno de desarrollo de la Web API	8 horas	jue 02/15/24	jue 02/15/24	InésDS
36		Implementar la gestión de usuarios y autenticación	40 horas	vie 02/16/24	vie 02/23/24	InésDS
37		Implementar la gestión de programas	40 horas	vie 02/23/24	vie 03/01/24	InésDS
38		Implementar la gestión de análisis	40 horas	vie 03/01/24	vie 03/08/24	InésDS
39		Implementar la gestión de resultados	40 horas	lun 03/11/24	vie 03/15/24	InésDS
40		Desarrollar la aplicación web	46 horas	lun 03/18/24	mar 04/02/24	InésDS

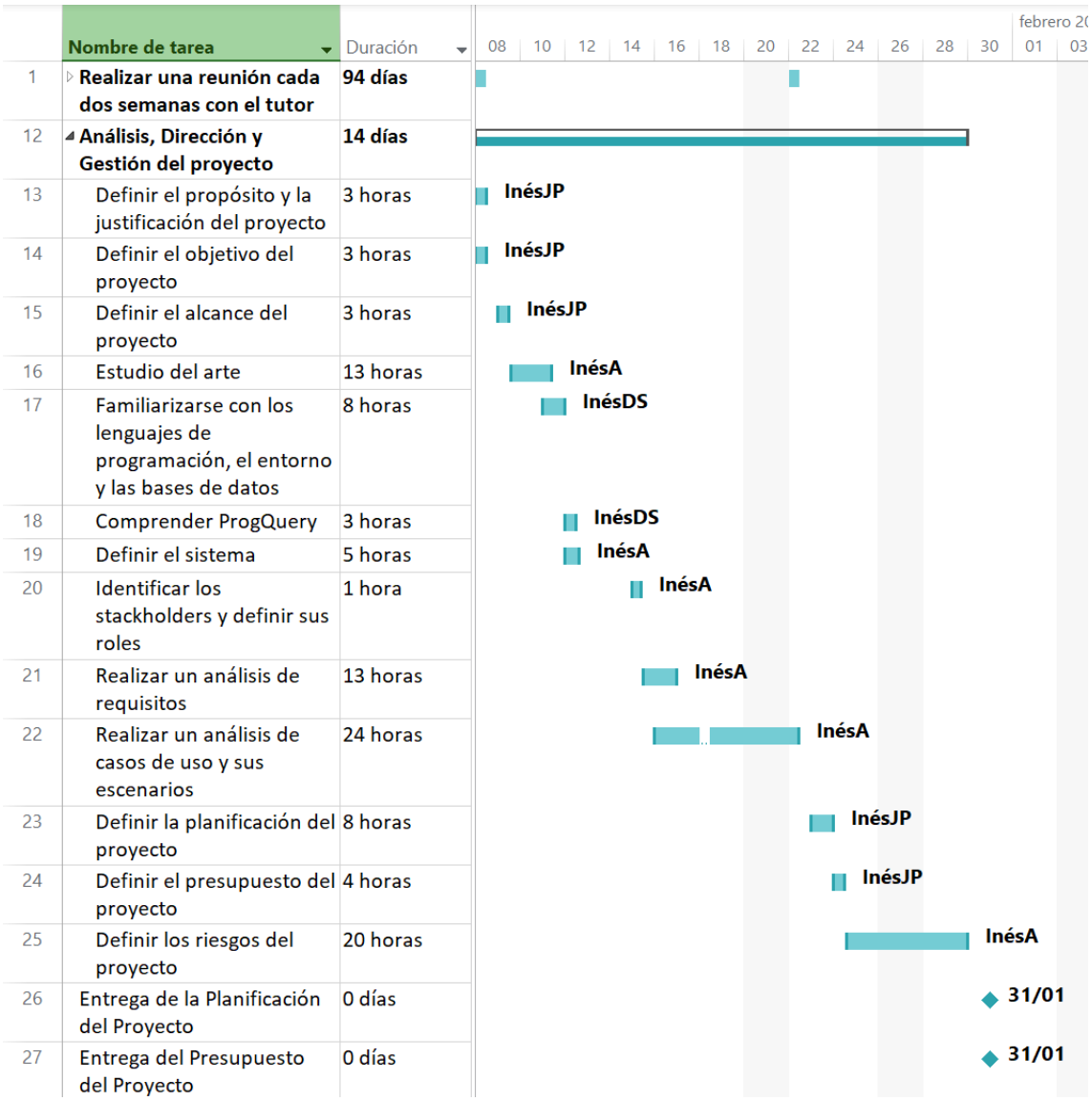
Ilustración 103. Tareas de desarrollo del software

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
42		**Pruebas**	**11 días**	**lun 15/04/24**	**lun 29/04/24**	
43		Realizar pruebas unitarias para la gestión de usuarios	10 horas	lun 15/04/24	mar 16/04/24	InésDS
44		Realizar pruebas unitarias para la gestión de programas	10 horas	mié 17/04/24	jue 18/04/24	InésDS
45		Realizar pruebas unitarias para la gestión de análisis	10 horas	vie 19/04/24	lun 22/04/24	InésDS
46		Realizar pruebas unitarias para la gestión de resultados	10 horas	mar 23/04/24	mié 24/04/24	InésDS
47		Realizar pruebas de integración	14 horas	jue 25/04/24	vie 26/04/24	InésDS
48		Identificar y solucionar errores y problemas	8 horas	vie 26/04/24	lun 29/04/24	InésDS
49		**Despliegue**	**2 días**	**jue 02/05/24**	**vie 03/05/24**	
50		Configurar el servidor y alojar la aplicación web y la Web API	12 horas	jue 02/05/24	vie 03/05/24	InésDS
51		Realizar pruebas finales	4 horas	vie 03/05/24	vie 03/05/24	InésDS

Ilustración 104. Tareas de pruebas y despliegue

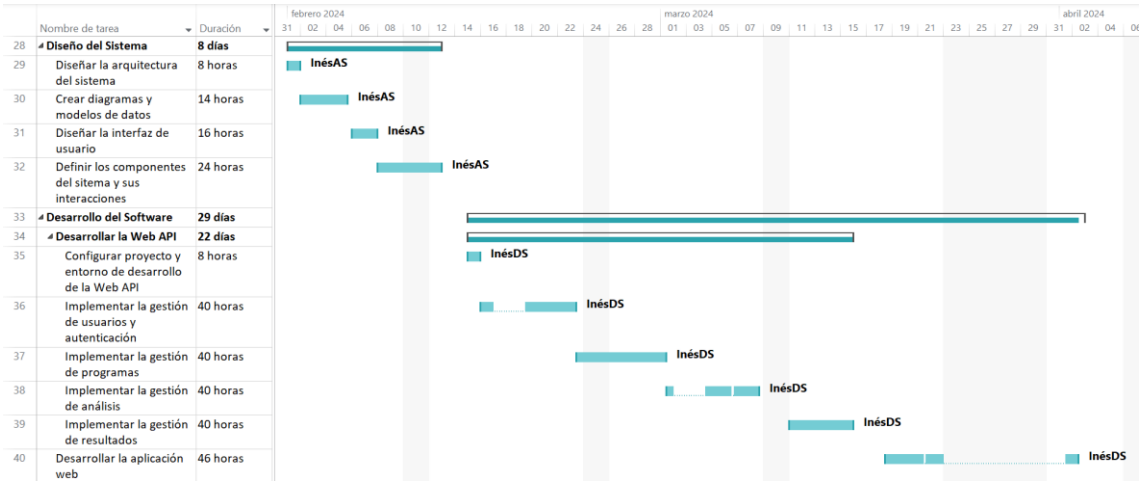
	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
52		**Documentación**	**9 días**	**lun 27/05/24**	**jue 06/06/24**	
53		Realizar manuales de usuario	14 horas	lun 27/05/24	mar 28/05/24	InésDS
54		Realizar documentación final del proyecto	50 horas	mié 29/05/24	jue 06/06/24	InésJP
55		**Cierre del Proyecto**	**1 día**	**vie 07/06/24**	**vie 07/06/24**	
56		Realizar una revisión final del proyecto	2 horas	vie 07/06/24	vie 07/06/24	InésJP
57		Realizar una reunión de cierre del proyecto	2 horas	vie 07/06/24	vie 07/06/24	InésJP
58		Entrega Final del Proyecto	0 días	jue 04/07/24	jue 04/07/24	

Ilustración 105. Tareas de documentación y cierre del proyecto



	Nombre de tarea	Duración
1	**Realizar una reunión cada dos semanas con el tutor**	**94 días**
12	**Análisis, Dirección y Gestión del proyecto**	**14 días**
13	Definir el propósito y la justificación del proyecto	3 horas
14	Definir el objetivo del proyecto	3 horas
15	Definir el alcance del proyecto	3 horas
16	Estudio del arte	13 horas
17	Familiarizarse con los lenguajes de programación, el entorno y las bases de datos	8 horas
18	Comprender ProgQuery	3 horas
19	Definir el sistema	5 horas
20	Identificar los stackholders y definir sus roles	1 hora
21	Realizar un análisis de requisitos	13 horas
22	Realizar un análisis de casos de uso y sus escenarios	24 horas
23	Definir la planificación del proyecto	8 horas
24	Definir el presupuesto del proyecto	4 horas
25	Definir los riesgos del proyecto	20 horas
26	Entrega de la Planificación del Proyecto	0 días
27	Entrega del Presupuesto del Proyecto	0 días

Ilustración 106. Diagrama de Gantt I



	Nombre de tarea	Duración
28	**Diseño del Sistema**	**8 días**
29	Diseñar la arquitectura del sistema	8 horas
30	Crear diagramas y modelos de datos	14 horas
31	Diseñar la interfaz de usuario	16 horas
32	Definir los componentes del sitema y sus interacciones	24 horas
33	**Desarrollo del Software**	**29 días**
34	**Desarrollar la Web API**	**22 días**
35	Configurar proyecto y entorno de desarrollo de la Web API	8 horas
36	Implementar la gestión de usuarios y autenticación	40 horas
37	Implementar la gestión de programas	40 horas
38	Implementar la gestión de análisis	40 horas
39	Implementar la gestión de resultados	40 horas
40	Desarrollar la aplicación web	46 horas

Ilustración 107. Diagrama de Gantt II

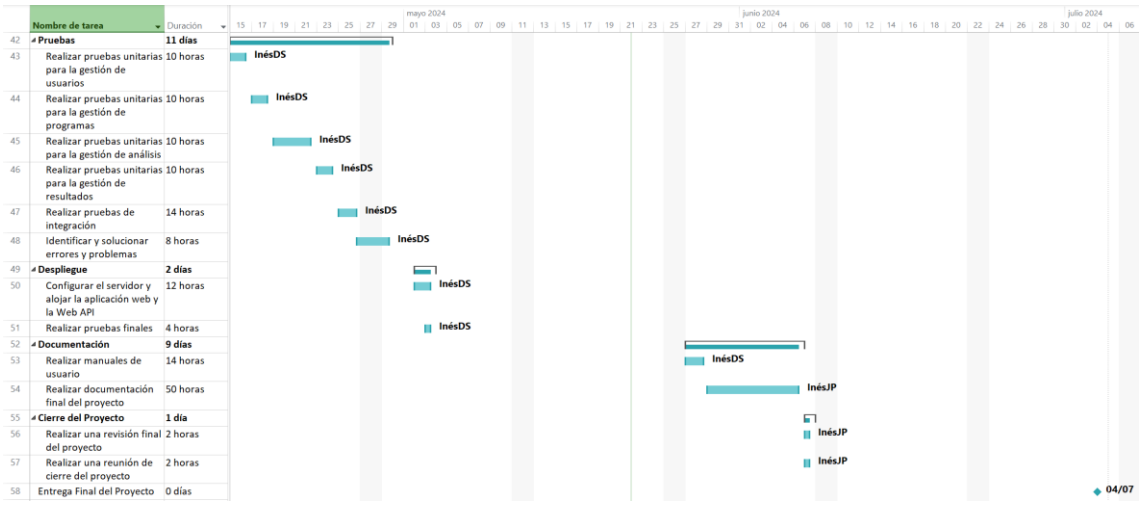


Ilustración 108. Diagrama de Gantt III

8.3.2. Informe final de riesgos

Este informe final de riesgos detalla las incidencias y riesgos gestionados durante la ejecución del proyecto de desarrollo del sistema web para el análisis de programas Java utilizando ProgQuery. El objetivo del informe es proporcionar una visión completa de los riesgos identificados, las acciones tomadas para mitigarlos y los resultados obtenidos, así como las lecciones aprendidas y recomendaciones para futuros proyectos.

Riesgo 1: Cambios en los requisitos del proyecto a mitad del desarrollo

Descripción: Modificaciones en los requisitos iniciales del proyecto durante la fase de desarrollo.

Impacto: Medio. Este riesgo podría causar retrasos en el cronograma del proyecto y requerir recursos adicionales para la replanificación y ajuste del desarrollo.

Acciones tomadas: Se llevó a cabo una revisión exhaustiva y actualización del plan del proyecto y del desarrollo. Se mantuvo una comunicación constante con los stakeholders para asegurarse de que los cambios se integraran de manera efectiva y eficiente.

Resultados: La actualización del plan permitió acomodar los cambios sin causar retrasos significativos en el proyecto. La comunicación continua ayudó a minimizar la confusión y asegurar que todos los miembros del equipo estuvieran alineados con los nuevos requisitos.

Riesgo 2: Falta de experiencia técnica

Descripción: Falta de conocimientos específicos necesarios para completar ciertas tareas del proyecto.

Impacto: Bajo. Este riesgo podría haber afectado la calidad del trabajo y causado errores en el desarrollo.

Acciones tomadas: Se adquirieron los conocimientos necesarios a través de cursos específicos.

Resultados: La formación adicional permitió adquirir las habilidades necesarias, mejorando la calidad del trabajo y reduciendo los errores técnicos.

Riesgo 3: Subida de varios programas simultáneamente

Descripción: La necesidad de manejar la carga de múltiples programas Java subidos simultáneamente.

Impacto: Medio. Este riesgo podría haber causado cuellos de botella en el sistema y reducido la eficiencia del proceso de carga.

Acciones tomadas: Se implementó una cola de trabajo para gestionar las subidas simultáneas de programas, asegurando que cada carga se procesara de manera eficiente y ordenada.

Resultados: La cola de trabajo mejoró significativamente la capacidad del sistema para manejar múltiples subidas, evitando cuellos de botella y asegurando una carga eficiente.

Riesgo 4: Retroalimentación de los usuarios sobre la usabilidad de la aplicación web

Descripción: Comentarios y retroalimentación de los usuarios respecto a la facilidad de uso de la aplicación web.

Impacto: Bajo. Este riesgo podría haber afectado la satisfacción del usuario y la adopción de la aplicación.

Acciones tomadas: Se realizaron pruebas de usabilidad y se ajustó la interfaz de usuario basándose en la retroalimentación recibida. Esto incluyó la implementación de mejoras en la navegación y la interfaz general.

Resultados: Las pruebas de usabilidad y los ajustes en la interfaz de usuario resultaron en una aplicación más intuitiva y fácil de usar, lo que incrementó la satisfacción de los usuarios.

8.3.3. Presupuesto final de costes

Con respecto al presupuesto final elaborado, se ha divido en varias partidas según la temática. Se muestran dichas partidas y el presupuesto total detallado y resumido.

Partida 1: Análisis, Dirección y Gestión del proyecto

I 1	I 2	I 3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
1			**Análisis, Dirección y Gestión del proyecto**						**6.453,75 €**
	1		Reuniones cada dos semanas con el tutor					1.200,00 €	
		1	Inés (Jefe de proyecto)	20	horas	60,00 €	1.200,00 €		
	2		Definición del propósito y la justificación del proyecto					180,00 €	
		1	Inés (Jefe de proyecto)	3	horas	60,00 €	180,00 €		
	3		Definición del objetivo del proyecto					180,00 €	
		1	Inés (Jefe de proyecto)	3	horas	60,00 €	180,00 €		
	4		Definifición del alcance del proyecto					180,00 €	
		1	Inés (Jefe de proyecto)	3	horas	60,00 €	180,00 €		
	5		Estudio del arte					633,75 €	
		1	Inés (Analista)	13	horas	48,75 €	633,75 €		
	6		Familiarización con los lenguajes de programación, el entorno y las bases de datos					210,00 €	
		1	Inés (Desarrollador de Software)	8	horas	26,25 €	210,00 €		
	7		Comprensión ProgQuery					78,75 €	
		1	Inés (Desarrollador de Software)	3	horas	26,25 €	78,75 €		
	8		Definición del sistema					243,75 €	
		1	Inés (Analista)	5	horas	48,75 €	243,75 €		
	9		Identificación de stackholders y roles					48,75 €	
		1	Inés (Analista)	1	hora	48,75 €	48,75 €		
	10		Análisis de requisitos					633,75 €	
		1	Inés (Analista)	13	horas	48,75 €	633,75 €		
	11		Análisis de casos de uso y escenarios					1.170,00 €	
		1	Inés (Analista)	24	horas	48,75 €	1.170,00 €		
	12		Definición de la planificación del proyecto					480,00 €	
		1	Inés (Jefe de proyecto)	8	horas	60,00 €	480,00 €		
	13		Definición del presupuesto del proyecto					240,00 €	
		1	Inés (Jefe de proyecto)	4	horas	60,00 €	240,00 €		
	14		Definición de los riesgos del proyecto					975,00 €	
		1	Inés (Analista)	20	horas	48,75 €	975,00 €		
								TOTAL	**6.453,75 €**

Ilustración 109. Presupuesto final I

Partida 2: Diseño del sistema

I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
1			**Diseño del sistema**						**3.255,00 €**
	1		Diseño de la arquitectura del sistema					420,00 €	
		1	Inés (Arquitecto de Software)	8	horas	52,50 €	420,00 €		
	2		Creación de diagramas y modelos de datos					735,00 €	
		1	Inés (Arquitecto de Software)	14	horas	52,50 €	735,00 €		
	3		Diseño de la interfaz de usuario					840,00 €	
		1	Inés (Arquitecto de Software)	16	horas	52,50 €	840,00 €		
	4		Definición de componentes del sistema y sus interacciones					1.260,00 €	
		1	Inés (Arquitecto de Software)	24	horas	52,50 €	1.260,00 €		
								TOTAL	**3.255,00 €**

Ilustración 110. Presupuesto final II

Partida 3: Desarrollo del sistema

I1	I2	I3	I4	Descripción	Cantidad	Unidades	Precio	Subtotal (4)	Subtotal (3)	Subtotal (2)	Total
1				**Desarrollo del sistema**							**5.617,50 €**
	1			Desarrollo de la WEB API						4.410,00 €	
		1		Configuración del proyecto y entorno de desarrollo					210,00 €		
			1	Inés (Desarrollador de Software)	8	horas	26,25 €	210,00 €			
		2		Implementación de la gestión de usuarios y autenticación					1.050,00 €		
			1	Inés (Desarrollador de Software)	40	horas	26,25 €	1.050,00 €			
		3		Implementación de la gestión de programas					1.050,00 €		
			1	Inés (Desarrollador de Software)	40	horas	26,25 €	1.050,00 €			
		4		Implementación de la gestión de análisis					1.050,00 €		
			1	Inés (Desarrollador de Software)	40	horas	26,25 €	1.050,00 €			
		5		Implementación de la gestión de resultados					1.050,00 €		
			1	Inés (Desarrollador de Software)	40	horas	26,25 €	1.050,00 €			
	2			Desarrollo de la aplicación web						1.207,50 €	
		1		Inés (Desarrollador de Software)	46	horas	26,25 €		1.207,50 €		
										TOTAL	**5.617,50 €**

Ilustración 111. Presupuesto final III

Partida 4: Pruebas y Despliegue

I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
1			**Pruebas**						**1.627,50 €**
	1		Pruebas unitarias para la gestión de usuarios					262,50 €	
		1	Inés (Desarrollador de Software)	10	horas	26,25 €	262,50 €		
	2		Pruebas unitarias para la gestión de programas					262,50 €	
		1	Inés (Desarrollador de Software)	10	horas	26,25 €	262,50 €		
	3		Pruebas unitarias para la gestión de análisis					262,50 €	
		1	Inés (Desarrollador de Software)	10	horas	26,25 €	262,50 €		
	4		Pruebas unitarias para la gestión de resultados					262,50 €	
		1	Inés (Desarrollador de Software)	10	horas	26,25 €	262,50 €		
	5		Pruebas de integración					367,50 €	
		1	Inés (Desarrollador de Software)	14	horas	26,25 €	367,50 €		
	6		Identificación y solución de errores y problemas					210,00 €	
		1	Inés (Desarrollador de Software)	8	horas	26,25 €	210,00 €		
2			**Despliegue**						**420,00 €**
	1		Configuración del servidor y alojamiento de la aplicación web y la Web API					315,00 €	
		1	Inés (Desarrollador de Software)	12	horas	26,25 €	315,00 €		
	2		Realización de pruebas finales					105,00 €	
		1	Inés (Desarrollador de Software)	4	horas	26,25 €	105,00 €		
								TOTAL	**2.047,50 €**

Ilustración 112. Presupuesto final IV

Partida 5: Documentación y Cierre del proyecto

I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
1			**Documentación**						**3.367,50 €**
	1		Realización de manuales de usuario					367,50 €	
		1	Inés (Desarrollador de Software)	14	horas	26,25 €	367,50 €		
	2		Documentación final del proyecto					3.000,00 €	
		1	Inés (Jefe de proyecto)	50	horas	60,00 €	3.000,00 €		
2			**Cierre del proyecto**						**240,00 €**
	1		Revisión final del proyecto					120,00 €	
		1	Inés (Jefe de proyecto)	2	horas	60,00 €	120,00 €		
	2		Reunión de cierre del proyecto con el cliente					120,00 €	
		1	Inés (Jefe de proyecto)	2	horas	60,00 €	120,00 €		
								TOTAL	**3.607,50 €**

Ilustración 113. Presupuesto final V

Partida 6: Costes indirectos

I 1	*I 2*	*Descripción*	*Cantidad*	*Unidades*	*Precio*	*Subtotal (3)*	*Total*
1		**Costes de producción**					**331,25 €**
	1	Servidor	5	meses	20,00 €	100,00 €	
	2	Portátil	5	meses	31,25 €	156,25 €	
	3	Licencias de desarrollo	5	meses	15,00 €	75,00 €	
2		**Otros**					**175,00 €**
	1	Consumo eléctrico	5	meses	35,00 €	175,00 €	

TOTAL	**506,25 €**

Ilustración 114. Presupuesto final VI

Presupuesto detallado del proyecto

Presupuesto del proyecto			
Partida	***Item***	***Partida***	***Importe***
1		**Análisis, Dirección y Gestión del proyecto**	**6.453,75 €**
2		**Diseño del sistema**	**3.255,00 €**
3		**Desarrollo del sistema**	**5.617,50 €**
4		**Pruebas y Despliegue**	**2.047,50 €**
	1	Pruebas	1.627,50 €
	2	Despliegue	420,00 €
5		**Documentación y Cierre**	**3.607,50 €**
	1	Documentación	3.367,50 €
	2	Cierre del proyecto	240,00 €
6		**Costes indirectos**	**506,25 €**
	1	Costes de producción	331,25 €
	2	Otros	175,00 €

TOTAL PROYECTO	**21.487,50 €**

Ilustración 115. Presupuesto final VII

Presupuesto general del proyecto

Presupuesto de cliente		
Cod.	***Partida***	***Total***
1	**Análisis, Dirección y Gestión del proyecto**	**6.453,75 €**
2	**Diseño del sistema**	**3.255,00 €**
3	**Desarrollo del sistema**	**5.617,50 €**
4	**Pruebas y Despliegue**	**2.047,50 €**
5	**Documentación y Cierre**	**3.607,50 €**
6	**Costes indirectos**	**506,25 €**

TOTAL PROYECTO	**21.487,50 €**

Ilustración 116. Presupuesto final VIII

8.3.4. Informe de lecciones aprendidas

8.3.4.1. Introducción

Este informe recoge las lecciones aprendidas durante el desarrollo del proyecto "Sistema web para el análisis estático de programas Java mediante ProgQuery". El objetivo de este

informe es documentar las experiencias y conocimientos adquiridos para mejorar futuros proyectos y compartir estas lecciones con la comunidad de desarrollo de software.

8.3.4.2. Resumen del Proyecto

El proyecto consistió en diseñar e implementar un sistema web que facilite la creación y el intercambio de análisis estáticos de programas Java utilizando la herramienta ProgQuery. El sistema incluye una Web API y una aplicación web, soportado por bases de datos PostgreSQL y Neo4j.

8.3.4.3. Planificación y Gestión del Proyecto

1. Planificación Inicial
    - Fortalezas: La planificación inicial fue exhaustiva, cubriendo todos los aspectos del proyecto desde la gestión de usuarios hasta la integración de bases de datos.
    - Debilidades: La planificación inicial no consideró adecuadamente el manejo de la carga de programas simultáneos, lo que se tuvo que considerar en fases avanzadas del proyecto.
2. Gestión de Tiempo
    - Fortalezas: Se mantuvo un cronograma riguroso, con revisiones semanales para ajustar las tareas.
    - Debilidades: Algunas tareas tomaron más tiempo del esperado debido a problemas técnicos imprevistos.

8.3.4.4. Desarrollo Técnico

1. Elección de Tecnologías
    - Fortalezas: La elección de Java con Spring Boot, .NET y las bases de datos PostgreSQL y Neo4j resultó adecuada para los objetivos del proyecto.
    - Debilidades: Se tuvo que realizar aprendizaje para algunas herramientas y framework.
2. Desarrollo de la Web API
    - Fortalezas: Asegura una comunicación eficiente y estandarizada entre el frontend y el backend, permitiendo una fácil integración y expansión futura.
    - Debilidades: En situaciones de alta carga, la API mostró problemas de rendimiento que requirieron optimización adicional.
3. Desarrollo de la Aplicación Web
    - Fortalezas: La aplicación web ofrece una interfaz intuitiva y amigable, facilitando la gestión de usuarios, programas y análisis.
    - Debilidades: Algún problema de usabilidad posteriormente resuelto.

8.3.4.5. Gestión de Datos

1. PostgreSQL
    - Fortalezas: Robustez y fiabilidad para el manejo de datos relacionales.
    - Debilidades: Necesidad de optimización en consultas complejas.
2. Neo4j
    - Fortalezas: Excelente para la representación y consulta de grafos complejos.
    - Debilidades: La carga y procesamiento de programas Java en forma de grafos presentaron desafíos algo significativos.

8.3.4.6. Colaboración y Comunicación

1. Comunicación
    - Fortalezas: Reuniones regulares y canales de comunicación efectivos.
    - Debilidades: Disponibilidad.

8.3.4.7. Evaluación de Resultados

1. Objetivos Cumplidos
    - Accesibilidad y Usabilidad: El sistema permite a los usuarios cargar programas Java para su análisis, tanto de forma directa como desde repositorios de código como GitHub, GitLab y Bitbucket.
    - Interfaces de Usuario: Se desarrollaron una aplicación web y una Web API, proporcionando múltiples formas de interactuar con el sistema.
    - Gestión de Usuarios: Funcionalidades de registro, autenticación y administración de perfiles implementadas con éxito.
    - Gestión de Programas y Análisis: Los usuarios pueden cargar, analizar, listar, actualizar y eliminar análisis de manera flexible.
    - Gestión de Resultados: Ejecución y almacenamiento de análisis, facilitando su acceso posterior.
    - Colaboración y Compartición: Fomentada la colaboración y el intercambio de conocimientos dentro de la comunidad de desarrolladores.
    - Documentación: Creación de documentación y ejemplos de uso detallados.
2. Áreas de Mejora
    - Mejora de la documentación para usuarios finales.
    - Optimización del rendimiento de la aplicación web y la API.
    - Mayor automatización en la integración y despliegue continuo (CI/CD).

8.3.4.8. Conclusión

El proyecto proporcionó valiosas experiencias y conocimientos en el desarrollo de sistemas complejos y la integración de herramientas avanzadas de análisis de código. Las lecciones aprendidas permitirán mejorar la planificación, gestión y ejecución de futuros proyectos, así como compartir estas experiencias para el beneficio de la comunidad de desarrollo de software.

Capítulo 9. Conclusiones y ampliaciones

9.1. Conclusiones

El sistema web desarrollado para el análisis estático de programas Java utilizando ProgQuery está orientado a ser una herramienta efectiva para manejar la creciente complejidad en el desarrollo de software. Este proyecto facilita la carga, análisis y compartición de resultados de análisis en programas Java, promoviendo la colaboración y el aprendizaje dentro de la comunidad de desarrolladores.

Una de las principales contribuciones del proyecto es su accesibilidad y usabilidad. Con dos interfaces principales - una Web API y una aplicación web- los usuarios pueden acceder a las funcionalidades de ProgQuery desde diferentes plataformas y dispositivos. Esto mejora significativamente la accesibilidad del sistema, haciendo que sea más fácil para los desarrolladores interactuar con las herramientas de análisis de código.

Además, el proyecto presenta una gestión eficiente de todas las operaciones. Los cuatro componentes clave del sistema (gestión de usuarios, gestión de programas, gestión de análisis y gestión de resultados) proporcionan una estructura clara y organizada, permitiendo a los usuarios manejar sus programas y análisis de manera efectiva.

El sistema también fomenta la colaboración al permitir la carga y análisis de programas desde repositorios populares como GitHub, GitLab y Bitbucket. Esto facilita el intercambio de conocimientos y la colaboración entre desarrolladores, impulsando la innovación y el progreso en la comunidad de desarrollo de software.

La integración de tecnologías avanzadas como PostgreSQL para el almacenamiento de datos relacionales y Neo4j para el almacenamiento de programas en forma de grafos asegura la eficiencia y escalabilidad del sistema. Esto permite manejar grandes volúmenes de datos de manera efectiva, garantizando que el sistema pueda crecer y adaptarse a las necesidades cambiantes de los usuarios.

El proyecto utiliza una arquitectura basada en microservicios debido a sus numerosas ventajas. Esta arquitectura facilita la escalabilidad, ya que permite que los componentes individuales se desplieguen y escalen de manera independiente. Además, mejora la resiliencia del sistema, dado que una falla en un microservicio no necesariamente afecta a los demás. También permite una mayor flexibilidad en el desarrollo, ya que diferentes equipos pueden trabajar en distintos microservicios utilizando tecnologías adecuadas para cada uno. Sin embargo, esta arquitectura también conlleva ciertos desafíos en su mantenimiento. La integración de múltiples microservicios y bases de datos requiere un mantenimiento constante y actualizaciones regulares para asegurar un funcionamiento óptimo y la incorporación de nuevas funcionalidades.

En resumen, el sistema web desarrollado no solo simplifica el acceso a las capacidades avanzadas de ProgQuery, sino que también fomenta la colaboración y el aprendizaje continuo entre los desarrolladores de software. Al proporcionar una plataforma robusta y versátil, este proyecto contribuye significativamente al avance tecnológico y al desarrollo

profesional en el campo del análisis masivo de código Java, orientado a la mejora de su calidad (con los análisis existentes), así como otras futuras como el rendimiento y seguridad, a definir libremente por sus potenciales usuarios.

9.2. Ampliaciones

El desarrollo de este sistema web para el análisis estático de programas Java utilizando ProgQuery abre muchas oportunidades para futuras ampliaciones e investigaciones.

Una posible ampliación es el soporte para otros lenguajes de programación. Ampliar el sistema para soportar el análisis de programas escritos en lenguajes como Python, C++ o JavaScript aumentaría su utilidad y alcance. Actualmente el grupo de investigación está trabajando en la representación de Python, así como en una representación común agnóstica al lenguaje.

Otra dirección interesante es la implementación de algoritmos de análisis más avanzados. Integrar análisis de vulnerabilidades, seguimiento de guías de estilo y optimización del rendimiento podría proporcionar conocimientos más profundos y valiosos a los desarrolladores.

El campo de Big Code (Big data aplicado a Source Code) es otra área prometedora. Se basa en la obtención de grandes volúmenes de código fuente como datos para entrenar modelos predictivos profundos orientados a la mejora del desarrollo de software. ProgQuery se puede utilizar para extraer propiedades del código fuente que se utilizarían como conjunto de datos de entrada para entrenar modelos de aprendizaje automático.

La seguridad y privacidad también son áreas clave para futuras investigaciones. Implementar medidas adicionales para proteger los datos de los usuarios y los programas cargados, asegurando el cumplimiento con regulaciones como el GDPR, es esencial para mantener la confianza en el sistema.

Finalmente, los análisis realizados pueden utilizarse en el entorno educativo. Un proyecto en actual desarrollo se basa en codificar como consultas Cypher las rúbricas que los profesores de una asignatura de programación utilizan para corregir el código de los alumnos. ProgQuery se pretende utilizar como herramienta de ayuda a los alumnos para que vean qué partes de su código pueden ser mejoradas, reforzando así su proceso de aprendizaje.

En resumen, este proyecto tiene el potencial de evolucionar y expandirse significativamente, abordando nuevas necesidades y desafíos en el campo del desarrollo de software. Las futuras ampliaciones e investigaciones pueden contribuir a mejorar las capacidades del sistema e impulsar la innovación y el avance en la comunidad global de desarrolladores de software.

Capítulo 10. Referencias bibliográficas

[1] Rodriguez-Prieto, O., Mycroft, A., & Ortin, F., «An Efficient and Scalable Platform for Java Source Code Analysis Using Overlaid Graph Representations,» *IEEE Access,* vol. 8, pp. 72239-72260, 2020.

[2] «Git,» [En línea]. Available: https://git-scm.com/. [Último acceso: 29 05 2024].

[3] «Code Quality Tool & Secure Analysis with SonarQube,» [En línea]. Available: https://www.sonarsource.com/products/sonarqube/. [Último acceso: 29 05 2024].

[4] «SpotBugs,» [En línea]. Available: https://spotbugs.github.io/. [Último acceso: 29 05 2024].

[5] «PMD,» [En línea]. Available: https://pmd.github.io/. [Último acceso: 29 05 2024].

[6] «JArchitect,» [En línea]. Available: https://www.jarchitect.com. [Último acceso: 29 05 2024].

[7] Urma, R. G., & Mycroft, A., «Source-code queries with graph databases—with application to programming language usage and evolution,» *Science of Computer Programming,* vol. 97, pp. 127-134, 2015.

[8] «Java PathFinder,» [En línea]. Available: https://github.com/javapathfinder/jpf-core/wiki/Home. [Último acceso: 29 05 2024].

[9] «Java Profiler - JProfiler,» [En línea]. Available: https://www.ej-technologies.com/products/jprofiler/overview.html. [Último acceso: 29 05 2024].

[10] «Code Climate,» [En línea]. Available: https://codeclimate.com/. [Último acceso: 29 05 2024].

[11] «Documentación de GitHub Actions - Documentación de GitHub,» [En línea]. Available: https://docs.github.com/es/actions. [Último acceso: 29 05 2024].

[12] Joshua Block, «Effective Java, 2nd edition,» *Addison-Wesley,* 2008.

[13] «Oracle Help Center,» [En línea]. Available: https://docs.oracle.com/en/java/. [Último acceso: 29 05 2024].

[14] «Spring Boot,» [En línea]. Available: https://docs.spring.io/spring-boot/index.html. [Último acceso: 29 05 2024].

[15] «IntelliJ IDEA Help,» [En línea]. Available: https://www.jetbrains.com/help/idea/getting-started.html. [Último acceso: 29 05 2024].

[16] «.NET documentation,» [En línea]. Available: https://learn.microsoft.com/en-us/dotnet/. [Último acceso: 29 05 2024].

[17] «Visual Studio documentation,» [En línea]. Available: https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022. [Último acceso: 29 05 2024].

[18] «PostgreSQL: Documentation,» [En línea]. Available: https://www.postgresql.org/docs/. [Último acceso: 29 05 2024].

[19] «Neo4j Graph Data Platform,» [En línea]. Available: https://neo4j.com/docs/docs/. [Último acceso: 29 05 2024].

[20] «Neo4j Graph Data Platform,» [En línea]. Available: https://neo4j.com/docs/cypher-manual/5/introduction/. [Último acceso: 29 05 2024].

[21] «Apache JMeter - Apache JMeter™,» [En línea]. Available: https://jmeter.apache.org/. [Último acceso: 01 07 2024].