Universidad de Cádiz
Escuela Superior de Ingeniería

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

APLICACIÓN DE MODELOS DE APRENDIZAJE PROFUNDO AL PREDICTIVE MUTATION TESTING

AUTOR: MARIO ROSANO BARROSO

Puerto Real, enero 2026

Universidad de Cádiz
Escuela Superior de Ingeniería

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

APLICACIÓN DE MODELOS DE APRENDIZAJE PROFUNDO AL PREDICTIVE MUTATION TESTING

DIRECTORES:
SARA BALDERAS DÍAZ
GABRIEL GUERRERO CONTRERAS

AUTOR: MARIO ROSANO BARROSO

Puerto Real, enero 2026

Agradecimientos

A mi **familia** por acompañarme y apoyarme toda mi vida. Sin vosotros, nada de esto habría sido posible. Gracias por vuestra paciencia, cariño y confianza incondicional.

A **Arturo, Jesús y Pablo**, por estar siempre ahí, por cada conversación, cada risa y cada momento que ha hecho este camino más llevadero. Gracias por ser un apoyo constante dentro y fuera de la universidad. Por más tiempo a vuestro lado.

A **La Ganga**: Sergio, Jorge, Ismael, Kane, Canijo, Coco, Hugo, Rafa, Fernando, Karma, Javi y Jesús. Gracias por todos los buenos ratos, las desconexiones necesarias y por hacer que cualquier día mejore solo con vuestra compañía, tanto en los buenos momentos como en los malos. Sois increíbles.

A **Real Ingeniería Informática**: Bertiny, Pedro, Dani, Mario Aranda, Guti, Víctor, Kike, Sergio, Raya, Mario Escobar, Salas, Rafa y Ángel. Gracias por todos los momentos que hemos compartido en la universidad: los descansos en ping pong, la ayuda que hemos tenido entre nosotros, los agobios de Víctor, todas las fiestas que hemos tenido, etc. Me alegro mucho de haber coincidido con vosotros en esta etapa.

A **Alejandro** por nuestros sin sentidos que hacen que no paremos de reir.

A **Mahe** por su compañía, su apoyo constante y por nuestras conversaciones frikis.

A **Quique** por ser el mejor manager que se pueda tener.

A **Jose** por ser una de las amistades más antiguas que tengo y por los buenos momentos que hemos pasado.

Finalmente, quiero agradecer a mi **tutora Sara** y a mi **cotutor Gabriel** por su dedicación, orientación y apoyo durante el desarrollo de este TFG. Gracias por vuestra paciencia, por vuestras indicaciones y por ayudarme a crecer a nivel académico.

Resumen

Este proyecto surge de una de las principales limitaciones de la técnica de *Prueba de Mutantes*, o *Mutation Testing*. Este método de validación del software permite evaluar la eficacia de un conjunto de pruebas introduciendo modificaciones pequeñas e intencionadas, denominadas mutantes, en el código fuente. El objetivo es comprobar si las pruebas implementadas son capaces de detectar los cambios generados, lo que sirve como indicador de su capacidad para encontrar errores reales.

En este contexto, es el proyecto elaborado el que se centra en abordar los principales desafíos asociados a esta técnica, el elevado coste computacional que implica ejecutar la totalidad de las pruebas sobre cada mutante generado. Con este trabajo se propone avanzar en la prueba de mutantes predictiva, conocida como PMT (Predictive Mutation Testing), que plantea reemplazar la ejecución directa de los mutantes por un proceso de predicción automatizada, mediante el diseño y análisis de modelos basados en técnicas de deep learning, con el objetivo de explorar su capacidad para identificar patrones más complejos en las métricas del código y de las ejecuciones de pruebas. El propósito es determinar si los enfoques investigados logran una mayor precisión en la predicción de mutantes, sin que ello implique un aumento significativo en el coste computacional del entrenamiento o la inferencia.

Para dar soporte al proceso completo de análisis y predicción, se ha desarrollado una plataforma web que centraliza todas las etapas implicadas en el estudio del PMT. Esta herramienta permite cargar proyectos, gestionar las características previamente extraídas y preparar estos datos para su uso por los algoritmos predictivos. Asimismo, facilita la administración de los modelos utilizados en la predicción, incluyendo su incorporación, actualización y evaluación, y ofrece un entorno unificado para visualizar los resultados obtenidos.

La plataforma constituye una solución pionera en su ámbito, al integrar en un único entorno web un flujo de trabajo modular y extensible que cubre desde la gestión de datos hasta la experimentación con modelos de predicción. Gracias a esta infraestructura, el proceso de investigación en PMT se vuelve más accesible, reproducible y escalable, lo que permitirá explorar nuevos enfoques, comparar modelos y avanzar en el desarrollo de técnicas más precisas y eficientes.

Palabras clave: Mutation Testing, Predictive Mutation Testing, mutantes, deep learning, modelos predictivos, métricas de código, plataforma web.

Abstract

This project arises from one of the main limitations of the Mutant Testing technique, also known as Mutation Testing. This software validation method evaluates the effectiveness of a test suite by introducing small and intentional modifications, called mutants, into the source code. The goal is to determine whether the implemented tests are able to detect the generated changes, which serves as an indicator of their ability to find real faults.

In this context, the present project focuses on addressing one of the main challenges associated with this technique: the high computational cost of executing the full test suite for every generated mutant. To tackle this issue, this work proposes advancing Predictive Mutation Testing (PMT), which replaces the direct execution of mutants with an automated prediction process. This is achieved through the design and analysis of models based on deep learning techniques, aimed at identifying more complex patterns in code metrics and test execution data. The objective is to determine whether the proposed approaches can achieve higher accuracy in mutant prediction without a significant increase in the computational cost of training or inference.

To support the complete analysis and prediction workflow, a web platform has been developed that centralizes all stages involved in the study of PMT. This tool allows users to upload projects, manage previously extracted features, and prepare these data for use by predictive algorithms. It also facilitates the management of the models used for prediction, including their incorporation, updating, and evaluation, and provides a unified environment for visualizing the obtained results.

The platform represents a pioneering solution in its domain, as it integrates into a single web environment a modular and extensible workflow covering everything from data management to experimentation with predictive models. Thanks to this infrastructure, PMT research becomes more accessible, reproducible, and scalable, enabling the exploration of new approaches, comparison of models, and advancement in the development of more accurate and efficient techniques.

Keywords: Mutation Testing, Predictive Mutation Testing, mutants, deep learning, predictive models, code metrics, web platform.

Índice general

Índice de figuras **15**

Índice de tablas **18**

I Prolegómeno **1**

1. Introducción **2**
1.1. Motivación . . . 2
1.2. Alcance y objetivos . . . 4
1.2.1. Alcance . . . 4
1.2.2. Objetivos . . . 4
1.3. Glosario de términos . . . 5
1.4. Organización del documento . . . 6

2. Antecedentes **9**
2.1. Contexto y estado de la técnica . . . 9

3. Plan de gestión de proyecto **12**
3.1. Metodología de desarrollo . . . 12
3.2. Tecnologías . . . 14
3.2.1. React . . . 14
3.2.2. Vite . . . 15
3.2.3. FastAPI . . . 15
3.2.4. PostgreSQL . . . 16
3.2.5. Figma . . . 17
3.2.6. Draw.io . . . 17
3.2.7. Docker . . . 17
3.2.8. Visual Studio Code . . . 18
3.2.9. GitHub . . . 18
3.2.10. Overleaf . . . 19
3.2.11. Wave . . . 19
3.2.12. Lighthouse . . . 20
3.3. Planificación del proyecto . . . 21
3.4. Costes . . . 21
3.4.1. Costes de recursos humanos . . . 21
3.4.2. Costes de recursos materiales . . . 22
3.5. Riesgos . . . 23

II Desarrollo **25**

4. Requisitos del sistema **26**

4.1. Objetivos de negocio . . . 26
4.2. Objetivos del sistema . . . 26
4.3. Actores del sistema . . . 26
4.4. Requisitos funcionales . . . 27
4.4.1. RF-01: Registro de usuario . . . 27
4.4.2. RF-02: Inicio de sesión . . . 27
4.4.3. RF-03: Menú principal y cierre de sesión . . . 27
4.4.4. RF-04: Visualización de usuarios registrados . . . 28
4.4.5. RF-05: Visualización del historial de sesiones IP de un usuario . 28
4.4.6. RF-06: Modificación del estado de la cuenta de un usuario . . . 28
4.4.7. RF-07: Modificación del rol de un usuario . . . 29
4.4.8. RF-08: Análisis de proyectos . . . 29
4.4.9. RF-09: Eliminación de proyectos del historial . . . 29
4.4.10. RF-10: Visualización métricas de proyectos analizados . . . 30
4.4.11. RF-11: Descarga de métricas de proyectos analizados . . . 30
4.4.12. RF-12: Subida de modelos PMT . . . 30
4.4.13. RF-13: Evaluación de modelos PMT . . . 30
4.4.14. RF-14: Filtrado y ordenación de datos . . . 31
4.4.15. RF-15: Selección de la paleta de colores de la plataforma . . . 31
4.5. Requisitos no funcionales . . . 31
4.5.1. RNF-01: Usabilidad . . . 31
4.5.2. RNF-02: Accesibilidad . . . 32
4.5.3. RNF-03: Fiabilidad . . . 32
4.5.4. RNF-04: Rendimiento y eficiencia . . . 32
4.5.5. RNF-05: Seguridad . . . 32
4.5.6. RNF-06: Mantenibilidad . . . 33
4.5.7. RNF-07: Portabilidad . . . 33
4.5.8. RNF-08: Escalabilidad . . . 33
4.5.9. RNF-09: Idioma de la plataforma . . . 33
4.6. Requisitos de información . . . 33
4.6.1. RI-01: Información de usuarios registrados . . . 33
4.6.2. RI-02: Estado de la cuenta de usuario . . . 34
4.6.3. RI-03: Historial de sesiones de usuario . . . 34
4.6.4. RI-04: Información sobre proyectos analizados . . . 34
4.6.5. RI-05: Métricas generadas por análisis . . . 35
4.6.6. RI-06: Información de modelos PMT . . . 35
4.6.7. RI-07: Resultados de evaluación de modelos . . . 35

5. Análisis del sistema **37**
5.1. Modelo conceptual . . . 37
5.2. Modelo de casos de uso . . . 38
5.3. Casos de uso del sistema . . . 41
5.4. Diagramas de actividad . . . 57
5.4.1. Registrar usuario . . . 57
5.4.2. Iniciar sesión . . . 58
5.4.3. Seleccionar paleta de colores . . . 59
5.4.4. Añadir proyecto . . . 60

5.4.5. Analizar proyecto . . . 61
5.4.6. Eliminar proyecto . . . 62
5.4.7. Visualizar métricas del análisis . . . 63
5.4.8. Exportar métricas . . . 64
5.4.9. Evaluar modelo de predicción . . . 65
5.4.10. Visualizar registro de sesiones de un usuario . . . 66
5.4.11. Modificar estado de un usuario . . . 67
5.4.12. Modificar rol de un usuario . . . 68
5.4.13. Subir modelo de predicción . . . 69

6. Diseño del sistema 71
6.1. Arquitectura del sistema . . . 71
6.1.1. Diseño de alto nivel . . . 71
6.1.1.1. Diagrama de contexto . . . 71
6.1.1.2. Diagrama de contenedores . . . 73
6.1.1.3. Diagrama de componentes . . . 74
6.1.1.4. Diagrama de arquitectura general . . . 76
6.1.1.5. Diagrama de flujo . . . 77
6.2. Diseño físico de datos . . . 79
6.2.1. Tabla: iplogs . . . 81
6.2.2. Tabla: users . . . 81
6.2.3. Tabla: models . . . 81
6.2.4. Tabla: project_uploads . . . 82
6.2.5. Tabla: project . . . 82
6.2.6. Tabla: project_results . . . 83
6.2.7. Tabla: analysis_tool . . . 83
6.2.8. Tabla: coverage_tool . . . 83
6.2.9. Tabla: mutation_tool . . . 84
6.2.10. Tabla: mutant . . . 84
6.2.11. Tabla: execution . . . 84
6.2.12. Tabla: file . . . 85
6.2.13. Tabla: static_analysis . . . 86
6.3. Diseño de la interfaz de usuario . . . 86
6.3.1. Sistema de paletas de colores . . . 86
6.4. Diseño de los modelos predictivos . . . 87
6.4.1. Modelos desarrollados . . . 87
6.4.2. Datos utilizados para el entrenamiento . . . 87
6.4.3. Configuración y optimización . . . 88
6.4.4. Métricas de los modelos entrenados . . . 89
6.4.5. Actualización y gestión de versiones . . . 89

7. Construcción del sistema 92
7.1. Entorno de construcción . . . 92
7.2. Código fuente . . . 92
7.2.1. Inserción del plugin OpenClover en el pom.xml . . . 94
7.2.2. Evaluación de modelos . . . 95
7.2.2.1. Extracto 1: Carga dinámica de modelos . . . 95
7.2.2.2. Extracto 2: Cálculo de métricas de evaluación . . . 96

7.2.3. Gestión de temas en la interfaz 96
7.3. Scripts de base de datos 98

8. Pruebas del sistema **101**
8.1. Pruebas de accesibilidad 101
8.1.1. Prueba 1: Navegación mediante teclado 101
8.1.2. Prueba 2: Contraste de colores 103
8.1.3. Prueba 3: Tamaño de fuente y escalado 104
8.2. Pruebas unitarias 105
8.2.1. Pruebas unitarias del inicio de sesión 105
8.2.1.1. Prueba 1: Inicio de sesión correcto 105
8.2.1.2. Prueba 2: Contraseña incorrecta 106
8.2.1.3. Prueba 3: Usuario no registrado 107
8.2.1.4. Prueba 4: Formato de correo inválido 107
8.2.1.5. Prueba 5: Campos vacíos 108
8.2.1.6. Prueba 6: Cierre de sesión 109
8.2.2. Pruebas unitarias del registro de usuario 110
8.2.2.1. Prueba 1: Registro correcto 110
8.2.2.2. Prueba 2: Campos obligatorios vacíos 111
8.2.2.3. Prueba 3: Formato de correo inválido 112
8.2.2.4. Prueba 4: Usuario ya existente 113
8.2.3. Pruebas unitarias del módulo de Proyectos 114
8.2.3.1. Prueba 1: Subida y análisis correcto de un proyecto mediante URL válida 114
8.2.3.2. Prueba 2: Subida y análisis incorrecto de un proyecto mediante URL válida 115
8.2.3.3. Prueba 3: Subida de proyecto con URL inválida 116
8.2.3.4. Prueba 4: Carga masiva de proyectos mediante fichero CSV 117
8.2.3.5. Prueba 5: Descarga de resultados de proyectos analizados 118
8.2.3.6. Prueba 6: Visualización de métricas de análisis 119
8.2.3.7. Prueba 7: Eliminación de un proyecto existente 119
8.2.4. Pruebas del módulo de Modelos 120
8.2.4.1. Prueba 1: Subida de un modelo con formato válido . . 120
8.2.4.2. Prueba 2: Subida de un modelo con una extensión no permitida 121
8.2.4.3. Prueba 3: Evaluación correcta de un proyecto con un modelo específico 122
8.2.4.4. Prueba 4: Evaluación de varios modelos sobre un mismo proyecto 122
8.2.5. Pruebas unitarias del módulo de gestión de usuarios 125
8.2.5.1. Prueba 1: Modificación de estado de un usuario 125
8.2.5.2. Prueba 2: Modificación de rol de un usuario 125
8.3. Pruebas de sistema 126
8.3.1. Pruebas funcionales 126
8.3.1.1. Prueba del RF-01 126
8.3.1.2. Prueba del RF-02 126

8.3.1.3. Prueba del RF-03 . . . 126
8.3.1.4. Prueba del RF-04 . . . 127
8.3.1.5. Prueba del RF-05 . . . 127
8.3.1.6. Prueba del RF-06 . . . 127
8.3.1.7. Prueba del RF-07 . . . 128
8.3.1.8. Prueba del RF-08 . . . 128
8.3.1.9. Prueba del RF-09 . . . 128
8.3.1.10. Prueba del RF-10 . . . 128
8.3.1.11. Prueba del RF-11 . . . 129
8.3.1.12. Prueba del RF-12 . . . 129
8.3.1.13. Prueba del RF-13 . . . 129
8.3.1.14. Prueba del RF-14 . . . 129
8.3.1.15. Prueba del RF-15 . . . 130
8.3.2. Pruebas no funcionales . . . 130
8.3.2.1. Prueba del RNF-01 . . . 130
8.3.2.2. Prueba del RNF-02 . . . 130
8.3.2.3. Prueba del RNF-03 . . . 131
8.3.2.4. Prueba del RNF-04 . . . 131
8.3.2.5. Prueba del RNF-05 . . . 133
8.3.2.6. Prueba del RNF-06 . . . 134
8.3.2.7. Prueba del RNF-07 . . . 134
8.3.2.8. Prueba del RNF-08 . . . 134
8.3.2.9. Prueba del RNF-09 . . . 135

9. Despliegue del sistema 137
9.1. Arquitectura física . . . 137
9.1.1. Requisitos físicos . . . 137
9.1.2. Requisitos software . . . 138
9.1.3. Modelo de despliegue . . . 139
9.2. Instrucciones de despliegue . . . 139
9.2.1. Procedimientos de instalación . . . 139
9.2.1.1. Acceso al repositorio del sistema . . . 140
9.2.2. Procedimientos de despliegue . . . 141
9.2.3. Pruebas de implantación . . . 142
9.3. Instrucciones para la operación del sistema y mantenimiento del nivel de servicio . . . 142
9.3.1. Backups y recuperación de datos . . . 143
9.3.2. Chequeo de logs . . . 143
9.3.3. Mantenimiento preventivo . . . 143
9.3.4. Disponibilidad y rendimiento . . . 144

III Epílogo 146

10. Conclusiones 147
10.1. Objetivos alcanzados . . . 147
10.2. Lecciones aprendidas . . . 148
10.3. Trabajo futuro . . . 148

Información sobre Licencia **151**

Bibliografía **153**

IV Anexos 156

A. Manual del desarrollador **157**
A.1. Preparación del entorno de trabajo 157
A.1.1. Requisitos físicos 157
A.1.2. Requisitos software 157
A.1.3. Control de versiones 157
A.2. Instrucciones de despliegue 158
A.2.1. Procedimiento de instalación 158
A.2.2. Acceso al repositorio del sistema 159
A.2.3. Procedimientos de despliegue 159
A.2.4. Pruebas de implantación 160
A.3. Consideraciones generales sobre el desarrollo 161
A.3.1. Buenas prácticas de desarrollo 161
A.3.2. Extensión del sistema 161
A.3.3. Pruebas y verificación de cambios 162

B. Manual de usuario **165**
B.1. Introducción 165
B.2. Uso del sistema 165
B.2.1. Registrar usuario 165
B.2.2. Iniciar sesión 166
B.2.3. Seleccionar paleta de colores 167
B.2.4. Menú lateral 169
B.2.5. Gestionar proyectos 170
B.2.5.1. Añadir y analizar proyecto 171
B.2.5.2. Borrar proyecto 172
B.2.5.3. Exportar métricas 173
B.2.5.4. Visualizar métricas de proyecto 173
B.2.6. Gestionar modelos 174
B.2.6.1. Subir modelo 175
B.2.6.2. Evaluar modelos 176
B.2.7. Gestionar usuarios 177
B.2.7.1. Gestionar información de un usuario 178
B.2.7.2. Modificar estado de un usuario 179
B.2.7.3. Modificar rol de un usuario 181

C. Lista de métricas **185**

Índice de figuras

3.1. Diagrama de metodología iterativa e incremental 12
3.2. React 15
3.3. Vite 15
3.4. FastAPI 16
3.5. PostgreSQL_elephant 16
3.6. Figma 17
3.7. Draw.io 17
3.8. Docker 18
3.9. Visual Studio Code 18
3.10. GitHub 19
3.11. Overleaf 19
3.12. Wave 20
3.13. Lighthouse 21
3.14. Diagrama de Gantt 21

5.1. Diagrama conceptual 37
5.2. Diagrama de casos de uso: Usuario 38
5.3. Diagrama de casos de uso: Admin 39
5.4. Diagrama de casos de uso: Superuser 39
5.5. Diagrama de actividad: Registrar usuario 57
5.6. Diagrama de actividad: Iniciar sesión 58
5.7. Diagrama de actividad: Seleccionar paleta de colores 59
5.8. Diagrama de actividad: Añadir proyecto 60
5.9. Diagrama de actividad: Analizar proyecto 61
5.10. Diagrama de actividad: Eliminar proyecto 62
5.11. Diagrama de actividad: Visualizar métricas del análisis 63
5.12. Diagrama de actividad: Exportar métricas 64
5.13. Diagrama de actividad: Evaluar modelo de predicción 65
5.14. Diagrama de actividad: Visualizar registro de sesiones de un usuario . . 66
5.15. Diagrama de actividad: Modificar estado de un usuario 67
5.16. Diagrama de actividad: Modificar rol de un usuario 68
5.17. Diagrama de actividad: Subir modelo de predicción 69

6.1. Diagrama de contexto 71
6.2. Diagrama de contenedores 73
6.3. Diagrama de componentes 74
6.4. Diagrama de arquitectura general 76
6.5. Diagrama de flujo 77
6.6. Esquema físico de la base de datos 80

8.1. Navegación mediante teclado con foco visible en el botón “ADD PROJECT” 102
8.2. Navegación mediante teclado con foco visible en el botón “Menu” . . . 102

8.3. Navegación mediante teclado con foco visible en la pestaña “Models” del menú lateral . . . 102
8.4. Prueba de escalado y adaptabilidad visual con zoom del 150 % . . . 104
8.5. Prueba de escalado y adaptabilidad visual con zoom del 200 % . . . 105
8.6. Login correcto . . . 106
8.7. Login incorrecto . . . 107
8.8. Usuario no registrado . . . 107
8.9. Formato de correo electrónico incorrecto . . . 108
8.10. Correo electrónico vacío . . . 108
8.11. Contraseña vacía . . . 109
8.12. Log out correcto . . . 109
8.13. Registro de usuario correcto . . . 111
8.14. Correo electrónico vacío . . . 112
8.15. Formato de correo electrónico incorrecto . . . 113
8.16. Usuario ya existente . . . 113
8.17. Proyecto subido y analizado correctamente . . . 115
8.18. Mensaje de análisis erróneo . . . 116
8.19. Proyecto subido y error en el análisis . . . 116
8.20. Mensaje de URL inválida . . . 117
8.21. Mensaje de éxito con la carga de tres proyectos por CSV . . . 117
8.22. Proyectos subidos y análizados correctamente . . . 118
8.23. Resultados descargados en un fichero CSV . . . 118
8.24. Visualización de métricas y filtros aplicados . . . 119
8.25. Mensaje de eliminación exitosa de proyectos . . . 120
8.26. Mensaje de confirmación de subida . . . 120
8.27. Modelo subido en el datagrid . . . 121
8.28. Mensaje de aviso de formato incorrecto . . . 121
8.29. Evaluación correcta de un modelo sobre un proyecto analizado . . . 122
8.30. Comparación de métricas entre varios modelos . . . 124
8.31. Mensaje de éxito en la modificación de estado . . . 125
8.32. Mensaje de éxito en la modificación de rol . . . 126
8.33. Rendimiento en Admin Dashboard . . . 132
8.34. Rendimiento en pantalla de proyectos sin análisis en ejecución . . . 132
8.35. Rendimiento en pantalla de proyectos mientras un proyecto está siendo analizado . . . 133
8.36. Rendimiento en pantalla de modelos . . . 133

B.1. Enlace a registro de usuario . . . 166
B.2. Pantalla de registro de usuario . . . 166
B.3. Pantalla de inicio de sesión . . . 167
B.4. Selector de la paleta de colores . . . 167
B.5. Pantalla de inicio de sesión con tema Claro . . . 168
B.6. Pantalla de inicio de sesión con tema Oscuro . . . 168
B.7. Pantalla de inicio de sesión con tema Azul . . . 169
B.8. Botón del menú . . . 169
B.9. Menú para usuarios administrador y superusuario . . . 170
B.10. Menú para usuarios corrientes . . . 170

B.11.Pantalla principal de proyectos 171
B.12.Botón para añadir proyecto 171
B.13.Pop-up para introducir enlaces de los repositorios de los proyectos . . . 172
B.14.Proyecto analizándose 172
B.15.Botón para borrar proyectos 172
B.16.Botón para exportar las métricas 173
B.17.Botón para visualizar las métricas 173
B.18.Pantalla de visualización de métricas 173
B.19.Selector de proyecto para visualizar métricas 174
B.20.Selector de columnas del datagrid 174
B.21.Pantalla de modelos de usuarios admin y superuser 175
B.22.Pantalla de modelos de usuarios corrientes 175
B.23.Botón para subir un modelo 175
B.24.Pop-up para subir un modelo 176
B.25.Botón para evaluar modelos 176
B.26.Pop-up para seleccionar el proyecto con el que evaluar los modelos . . . 176
B.27.Pantalla con las métricas resultantes de los modelos 177
B.28.Selector de modelo para visualizar resultados 177
B.29.Pantalla de gestión de usuarios 178
B.30.Pantalla de detalles del usuario 178
B.31.Selector del estado del usuario 179
B.32.Estado bloqueado del usuario en el Admin Dashboard 180
B.33.Selector del estado no disponible 180
B.34.Selector del rol del usuario 181
B.35.Selector del rol no disponible 182

Índice de tablas

3.1. Costes de recursos humanos 22
3.2. Costes de recursos materiales 22
3.3. Riesgos del proyecto, impacto, probabilidad y planes de mitigación . . . 23

5.1. Caso de uso: Registrar usuario 41
5.2. Caso de uso: Iniciar sesión 42
5.3. Caso de uso: Seleccionar paleta de colores 43
5.4. Caso de uso: Gestionar proyectos 44
5.5. Caso de uso: Añadir proyecto 45
5.6. Caso de uso: Analizar proyecto 46
5.7. Caso de uso: Eliminar proyecto 47
5.8. Caso de uso: Visualizar métricas del análisis 48
5.9. Caso de uso: Exportar métricas 49
5.10. Caso de uso: Evaluar modelo de predicción 50
5.11. Caso de uso: Gestionar usuarios 51
5.12. Caso de uso: Visualizar registro de sesiones de un usuario 52
5.13. Caso de uso: Ver registro de sesiones de un usuario 53
5.14. Caso de uso: Modificar rol de un usuario 54
5.15. Caso de uso: Gestionar modelos de predicción 55
5.16. Caso de uso: Subir modelo de predicción 56

6.1. Colores principales de cada tema con representación visual 87
6.2. Métricas de validación de los modelos predictivos seleccionados tras Grid Search. 89

Parte I

Prolegómeno

1. Introducción

A continuación, se describe la motivación del presente proyecto y su alcance. También se incluye un glosario de términos y la organización del resto de la presente documentación.

1.1. Motivación

En el desarrollo de software, garantizar la calidad del código es una tarea crítica y constante. Entre las técnicas más potentes para evaluar la efectividad de las pruebas se encuentra la *Prueba de Mutantes* o *Mutation Testing*. Esta técnica consiste en introducir pequeños errores intencionados (conocidos como mutantes) en el código fuente para comprobar si los casos de prueba implementados son capaces de detectarlos. Si las pruebas "matan" al mutante (es decir, fallan al ejecutarse sobre el código modificado), se considera que son efectivas. De lo contrario, se revela una posible debilidad en la cobertura del test, como se expone en trabajos seminales sobre el tema (Jia and Harman, 2011).

Sin embargo, esta rigurosidad tiene un precio, el *Mutation Testing* tradicional es extremadamente costoso desde el punto de vista computacional. Evaluar miles de mutantes implica una cantidad enorme de ejecuciones, lo que lo hace poco práctico para proyectos grandes o en entornos con recursos limitados.

Aquí es donde surge una pregunta inevitable: ¿es posible hacer esta técnica más eficiente sin sacrificar su efectividad? En un contexto donde la inteligencia artificial y el uso de machine learning para optimizar procesos complejos y de gran escala están revolucionando distintos campos, resulta natural explorar cómo estas tecnologías pueden optimizar también el proceso de validación del software. Surge así el concepto de Predictive Mutation Testing (Balderas-Díaz et al., 2025; Guerrero-Contreras et al., 2025), un enfoque que utiliza modelos de machine learning para identificar los mutantes más relevantes a partir de métricas y características de los proyectos. De este modo, el sistema puede predecir qué mutantes tienen mayor probabilidad de influir en los resultados de las pruebas, reduciendo de forma significativa el número de ejecuciones necesarias. Esto no solo disminuye el coste computacional, sino que también permite enfocar los recursos en los mutantes que realmente aportan información útil para evaluar la calidad de las pruebas, optimizando así todo el proceso de validación del software. Además, esta aproximación facilita la experimentación con distintos modelos y configuraciones, proporcionando un entorno más flexible y eficiente para mejorar las técnicas de testing predictivo.

Como parte de esta investigación, se ha desarrollado una plataforma web diseñada para centralizar y automatizar las principales etapas del proceso de Mutation Testing predictivo. La herramienta permite gestionar proyectos de software, organizar las características previamente extraídas por módulos externos, cargar y administrar

diferentes modelos de predicción y ejecutar evaluaciones sobre los mutantes generados. Además, ofrece al usuario una visualización clara de los resultados y métricas obtenidas, lo que facilita la comparación entre modelos y la interpretación del comportamiento del sistema durante el análisis.

La creación de esta plataforma es el resultado del trabajo realizado para integrar, en un mismo entorno operativo, todas las fases necesarias para aplicar Predictive Mutation Testing sobre proyectos reales. Durante el desarrollo se entrenaron modelos capaces de predecir el comportamiento de mutantes utilizando las métricas generadas por herramientas externas de análisis estático y dinámico. Estos modelos permiten identificar qué mutantes aportan información relevante sin necesidad de ejecutar el conjunto completo de pruebas, reduciendo de forma significativa el coste computacional y facilitando el uso de esta técnica en contextos académicos y experimentales.

El desarrollo de esta plataforma responde a una necesidad real tanto en el ámbito académico como en el industrial: disponer de herramientas capaces de mejorar la calidad del software sin incurrir en costes computacionales excesivos. Actualmente, el Mutation Testing es reconocido como una de las técnicas más fiables para evaluar la efectividad de las pruebas, pero su elevado coste lo ha mantenido limitado a entornos de investigación, dejando de lado su adopción en empresas y proyectos reales. Esta situación genera una brecha entre el potencial teórico de la técnica y su aplicabilidad práctica.

La plataforma creada en este TFG contribuye directamente a reducir esta brecha al permitir experimentar con modelos predictivos que disminuyen el número de mutantes que requieren ser ejecutados. Esto abre la puerta a mecanismos de validación más accesibles, rápidos y sostenibles, alineados con las necesidades actuales de la industria, donde la calidad del software debe garantizarse en ciclos de desarrollo cada vez más cortos y exigentes.

Además, al ofrecer un entorno centralizado, la herramienta facilita la experimentación y la comparación entre distintas estrategias predictivas. Este tipo de infraestructura resulta clave para avanzar en líneas de investigación emergentes, pero también para que empresas y equipos de desarrollo puedan incorporar técnicas más avanzadas de aseguramiento de la calidad sin requerir conocimientos especializados en aprendizaje automático.

En conjunto, esta plataforma no solo impulsa la investigación en Predictive Mutation Testing, sino que también promueve su transferencia hacia el mundo profesional, fomentando prácticas de validación más precisas y eficientes. Su uso contribuye a mejorar la calidad del software, reducir costes operativos y fortalecer un sector tecnológico que demanda herramientas cada vez más inteligentes, automatizadas y basadas en datos.

1.2. Alcance y objetivos

1.2.1. Alcance

El alcance del proyecto se resume en lo siguiente:

- **Desarrollo de una plataforma web** que integre herramientas de Mutation Testing con modelos de predicción basados en inteligencia artificial.
- La plataforma permitirá a los usuarios:
  - Gestionar la subida de proyectos Java mediante enlaces a repositorios públicos de GitHub.
  - Orquestar la ejecución de herramientas externas encargadas de la generación de mutantes y del cálculo de métricas estáticas y dinámicas.
  - Centralizar y organizar las métricas obtenidas por estos módulos externos para que puedan ser empleadas posteriormente en la evaluación de modelos predictivos.
  - Seleccionar y aplicar modelos predictivos cargados en el sistema para estimar el estado de los mutantes a partir de las características generadas.
  - Visualizar los resultados mediante tablas, métricas agregadas y reportes exportables que facilitan el análisis posterior.
  - Exportar resultados en distintos formatos (CSV o ZIP) para su uso en estudios académicos o análisis adicionales.
- **Facilitación del acceso a herramientas avanzadas** para estudiantes, docentes e investigadores, promoviendo el estudio y la adopción de técnicas modernas de aseguramiento de la calidad del software.

1.2.2. Objetivos

Se consideran los siguientes objetivos:

Objetivo general:

- Crear una plataforma web integral de *Predictive Mutation Testing* que permita a usuarios académicos y de investigación analizar proyectos de software, generar y evaluar mutantes, aplicar modelos predictivos y visualizar métricas de manera clara y accesible.

Objetivos específicos:

- Diseñar y entrenar modelos de aprendizaje automático capaces de predecir el estado de los mutantes en un proyecto de software.
- Implementar una interfaz web amigable que permita a los usuarios subir sus proyectos, gestionar el proceso de análisis y visualizar los resultados obtenidos,

tanto las métricas asociadas a cada mutante como las predicciones generadas por los modelos cargados en la plataforma.

- Reducir significativamente el coste computacional del Mutation Testing tradicional mediante técnicas de predicción basadas en datos.
- Fomentar el uso y la experimentación con técnicas de Mutation Testing al proporcionar una plataforma que simplifica su ejecución, reduce su coste computacional y facilita el análisis de resultados, permitiendo que este tipo de validación sea más accesible y práctica en distintos entornos de desarrollo y evaluación de software.

1.3. Glosario de términos

- **API** – Application Programming Interface
- **CSS** – Cascading Style Sheets
- **CU** – Caso de Uso
- **HTTP** – HyperText Transfer Protocol
- **HTTPS** – HyperText Transfer Protocol Secure
- **IA** – Inteligencia Artificial
- **IDE** – Integrated Development Environment
- **JWT** – JSON Web Token
- **ML** – Machine Learning
- **PMT** – Predictive Mutation Testing
- **RF** – Requisito Funcional
- **RNF** – Requisito No Funcional
- **RI** – Requisito de Información
- **SQL** – Structured Query Language
- **SPA** – Single Page Application
- **UI** – User Interface
- **URL** – Uniform Resource Locator
- **VCS** – Version Control System

1.4. Organización del documento

Esta memoria describe de forma estructurada el proceso completo seguido para el desarrollo de *PMTlab*, la plataforma web creada en el contexto de este Trabajo de Fin de Grado. A continuación, se detallan los contenidos de cada capítulo y su relación con el sistema desarrollado:

- **Capítulo 1: Introducción**

  Expone la motivación que da origen al proyecto, centrada en las limitaciones actuales del Mutation Testing tradicional y la necesidad de soluciones más eficientes basadas en predicción. Incluye los objetivos del trabajo, el alcance real de la plataforma desarrollada y un glosario de términos técnicos específicos del PMT.

- **Capítulo 2: Antecedentes**

  Presenta el contexto teórico necesario para comprender el funcionamiento del PMT, así como un análisis del estado de la técnica. Se revisan las herramientas existentes de Mutation Testing, las propuestas académicas relacionadas con PMT y se justifica por qué ninguna de ellas proporciona una plataforma integrada como la desarrollada en este TFG.

- **Capítulo 3: Plan de gestión de proyecto**

  Describe la planificación seguida para llevar a cabo el desarrollo de PMTlab. Incluye la metodología aplicada, las tecnologías que han sustentado tanto el frontend como el backend, la planificación temporal del proyecto, los recursos empleados, los costes estimados y los riesgos identificados durante la fase de desarrollo y despliegue.

- **Capítulo 4: Requisitos del sistema**

  Define con precisión lo que la plataforma debía permitir hacer: gestión de proyectos, análisis automático, manejo de mutantes, evaluación mediante modelos predictivos y visualización de resultados. Se describen los actores reales del sistema y se presentan los requisitos funcionales, no funcionales y de información sobre los que después se construyó la arquitectura.

- **Capítulo 5: Análisis del Sistema**

  Detalla el análisis previo necesario para comprender el funcionamiento interno de PMTlab. Incluye el modelo conceptual de datos, directamente relacionado con las métricas, mutantes y resultados generados por el sistema, el modelo de casos de uso que describe cómo los usuarios interactúan con la plataforma y los diagramas de actividad que representan los flujos principales del sistema.

- **Capítulo 6: Diseño del Sistema**

  Expone cómo se ha diseñado técnicamente la plataforma: la arquitectura cliente-servidor con React y FastAPI, el diseño físico de la base de datos PostgreSQL, la estructura interna de los componentes y la interfaz gráfica. Todo el diseño está

directamente vinculado al soporte de la carga de proyectos, su análisis automático y la evaluación de modelos predictivos.

- **Capítulo 7: Construcción del Sistema**

  Describe la implementación real del proyecto. Se detalla cómo está organizado el repositorio, cómo se implementa cada módulo (análisis, evaluación, gestión de usuarios, subida de modelos, etc.), el entorno de construcción con Docker y la estructura del código tanto en backend como en frontend.

- **Capítulo 8: Pruebas del Sistema**

  Incluye las pruebas realizadas sobre la plataforma: pruebas unitarias del analizador, del módulo de subida de proyectos y del sistema de gestión de usuarios; pruebas de sistema; y pruebas de accesibilidad. Cada prueba está acompañada de capturas y resultados reales obtenidos con PMTlab.

- **Capítulo 9: Despliegue del Sistema**

  Explica la arquitectura física utilizada durante el despliegue, basada en contenedores Docker. Se detallan los pasos necesarios para arrancar la plataforma, inicializar la base de datos, ejecutar el backend y acceder al frontend. También incluye instrucciones de operación diaria y mantenimiento del servicio.

- **Capítulo 10: Conclusiones**

  Resume los objetivos alcanzados, analiza críticamente las decisiones tomadas durante el desarrollo y presenta las principales lecciones aprendidas. También se proponen líneas de trabajo futuro relacionadas con la ampliación del soporte a nuevos lenguajes, la mejora de los modelos predictivos y la extensión de capacidades de análisis.

- **Información sobre Licencia**

  Expone los derechos de uso, distribución y propiedad del software desarrollado durante el TFG, así como las condiciones bajo las que pueden emplearse los modelos y scripts incluidos en la plataforma.

- **Bibliografía**

  Recoge todos los artículos científicos, obras de referencia y recursos técnicos utilizados para sustentar la investigación y el desarrollo del sistema.

- **Anexo A: Manual del desarrollador**

  Incluye una guía completa para entender y ampliar el código del proyecto: instrucciones para levantar el entorno, estructura interna del backend y frontend, dependencias y notas para futura evolución del sistema.

- **Anexo B: Manual de usuario**

  Proporciona explicaciones claras para que cualquier usuario pueda utilizar la plataforma: registro, subida de proyectos, visualización de métricas, evaluación de modelos, descarga de resultados, etc.

2. Antecedentes

2.1. Contexto y estado de la técnica

A pesar de los avances teóricos del PMT, actualmente no existe una plataforma web que permita aplicar estos conceptos de manera accesible y automatizada. Las herramientas tradicionales de *Mutation Testing*, como PIT, MutPy, Major, Javalanche o Stryker, se centran en la generación y ejecución de mutantes y requieren configuraciones manuales, lo que dificulta su uso. Además, estas herramientas comparten una limitación fundamental, pues no integran técnicas predictivas basadas en inteligencia artifical ni ofrecen interfaces gráficas que faciliten la carga de proyectos o la visualización de los resultados.

En el plano científico, varios estudios han explorado la incorporación de enfoques predictivos al *Mutation Testing* (**?**), aplicando modelos de aprendizaje automático para reducir el número de mutantes generados o estimar la efectividad de los casos de prueba. Sin embargo, estas propuestas se han centrado principalmente en la validación experimental y carecen de implementaciones accesibles que puedan ser utilizadas de forma directa por investigadores, estudiantes o profesionales.

Por otra parte, dentro del ámbito más amplio del aseguramiento de la calidad del software, existen plataformas y herramientas modernas que combinan analítica de software con técnicas inteligentes o de machine learning. Por ejemplo, CodeScene (CodeScene AB, 2025) realiza análisis de comportamiento del código, identifica "hotspots" y riesgos ocultos mediante algoritmos de ML. También existen plataformas de inspección de código como SonarQube (SonarSource, 2025), que proporcionan métricas de calidad, reportes web y reglas de calidad configurables. Además, Qodana (JetBrains, 2025) de JetBrains funciona como una plataforma de análisis estático integrada con CI/CD y con soporte para muchos lenguajes. A nivel más académico, el "Metrics Dashboard" (Thiruvathukal et al., 2018) ofrece una aplicación web para estudiar métricas de calidad de software a lo largo del tiempo, como densidad de defectos o productividad. Y, en un ámbito más relacionado con la investigación de modelos de Deep Learning reutilizables, trabajos como PTMTorrent (Zhang et al., 2023) han generado un gran conjunto de datos para el análisis sistemático de paquetes de modelos pre-entrenados de código abierto.

Aunque estas plataformas muestran avances importantes en la analítica de calidad de software, ninguna de ellas aborda de forma específica la integración práctica de modelos predictivos dentro del flujo de trabajo del Mutation Testing. Este hecho refuerza la originalidad de la propuesta de este trabajo: desarrollar una plataforma web integral que no solo genere y analice mutantes, sino que también aplique modelos predictivos y presente los resultados de forma interactiva.

No obstante, aunque estas plataformas representan avances significativos en la analítica de calidad de software, ninguna aborda específicamente la integración práctica de modelos predictivos dentro del flujo de trabajo del *Mutation Testing*. Esta falta de

soluciones accesibles evidencia una brecha tecnológica entre la investigación en PMT y su aplicación práctica. De este modo, surge la necesidad de una plataforma web integral que permita cargar proyectos, generar y analizar mutantes, aplicar modelos predictivos y presentar métricas de forma visual. Una herramienta de este tipo no solo cubriría una carencia en el ecosistema actual, sino que también fortalecería la relación entre la investigación y su implementación en el contexto académico, además del uso profesional.

3. Plan de gestión de proyecto

En esta sección se describen todos los aspectos relativos a la gestión del proyecto: metodología, organización, costes, planificación, riesgos y aseguramiento de la calidad.

3.1. Metodología de desarrollo

El desarrollo del proyecto se ha llevado a cabo siguiendo un enfoque iterativo e incremental. No se ha aplicado de manera estricta una metodología tradicional como el modelo en cascada. En su lugar, se ha adoptado una estrategia híbrida que combina características de ambos enfoques, permitiendo alternar entre fases de investigación, diseño, implementación y validación según el avance y los requerimientos específicos de cada etapa.

Este enfoque flexible ha permitido compaginar la exploración de técnicas de aprendizaje automático con la construcción de la plataforma web, ya que ambas partes requerían tiempos distintos de análisis, experimentación y pruebas. El desarrollo se organizó en iteraciones sucesivas, cada una de las cuales generó un incremento funcional del sistema, añadiendo o mejorando componentes como nuevos modelos predictivos o funcionalidades de la interfaz web. Al finalizar cada iteración, se llevaron a cabo actividades de evaluación y retroalimentación que sirvieron para ajustar los objetivos y planificar las mejoras del siguiente ciclo.

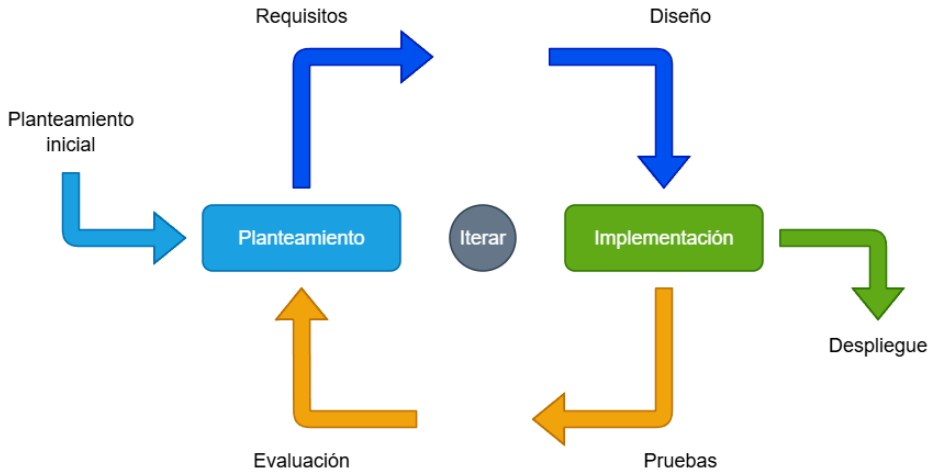


Figura 3.1: Diagrama de metodología iterativa e incremental

Las principales fases del ciclo de vida del proyecto serán las siguientes:

Fase de requisitos

En esta etapa se establecerán los objetivos generales y específicos del proyecto, como la reducción del coste computacional del *Predictive Mutation Testing*, la implementación de una plataforma web accesible y la validación de modelos de aprendizaje automático. Además, se recogerán los requisitos técnicos de la infraestructura, la base de datos y la interacción entre componentes.

Los requisitos se revisarán y refinarán a lo largo de las diferentes iteraciones conforme se obtengan nuevos resultados experimentales.

Fase de diseño

Se definirá la arquitectura del sistema y la organización de sus diferentes módulos: backend, frontend, base de datos y scripts auxiliares. Se diseñará también el flujo de interacción entre la API, la capa de análisis de mutantes y los modelos predictivos. El diseño tendrá en cuenta la necesidad de integrar contenedores Docker para simplificar el despliegue y asegurar la reproducibilidad de los experimentos.

Cada iteración permitirá ajustar el diseño en función de la experiencia adquirida durante las pruebas e implementaciones previas.

Fase de implementación

Durante esta etapa se desarrollarán tanto la plataforma web como los modelos de predicción. El trabajo se organizará en dos líneas complementarias:

- Por un lado, la parte **investigadora**, dedicada a probar y entrenar diferentes modelos de aprendizaje automático con el fin de evaluar su capacidad para predecir la relevancia de mutantes.
- Por otro lado, la parte **práctica**, enfocada en construir la aplicación web, incluyendo la interfaz de usuario, el servidor y la integración con la base de datos.

De esta manera, mientras se avance en la investigación de los modelos, también se asegurará que el sistema final fuera utilizable y cumpliera los objetivos planteados.

Cada iteración concluirá con un incremento funcional que integrará mejoras en todo el sistema.

Fase de pruebas

Se realizarán pruebas de verificación y validación en distintos niveles:

- **Pruebas unitarias** sobre scripts de análisis y funciones críticas.

- **Pruebas de integración** para garantizar la correcta comunicación entre backend, frontend y base de datos.
- **Pruebas funcionales** desde el punto de vista del usuario final, incluyendo la subida de proyectos, la ejecución de análisis y la visualización de métricas.

Estas pruebas también servirán para evaluar la precisión y eficiencia de los modelos predictivos desarrollados.

Después de cada iteración se revisarán los resultados obtenidos, identificando posibles mejoras. Este proceso permitirá ir perfeccionando el proyecto de forma progresiva y asegurando que cada versión sea más completa y estable que la anterior.

Fase de mantenimiento

Aunque se trate de un proyecto académico, se han previsto tareas básicas de mantenimiento con el objetivo de asegurar la continuidad del sistema en el futuro. Entre ellas se incluyen la actualización periódica de dependencias, la posibilidad de añadir nuevos modelos de predicción, la adaptación a distintos lenguajes de programación y el seguimiento del sistema mediante registros y copias de seguridad.

3.2. Tecnologías

El proyecto ha contado con el uso de una amplia gama de herramientas y tecnologías. A continuación, se presentan todas aquellas que han respondido eficazmente a las necesidades de cada etapa del desarrollo, habiéndose seleccionado las opciones más adecuadas por su funcionalidad, rendimiento y compatibilidad con los objetivos del sistema.

3.2.1. React

React (Meta Platforms, 2025) es una biblioteca desarrollada por Meta (anteriormente conocida como Facebook), la cual se utiliza para construir interfaces de usuario, especialmente en aplicaciones web de una sola página (Single Page Applications). Está basada en JavaScript, lo que permite integrar la lógica del lenguaje con estructuras visuales de forma eficiente. Su característica principal es la capacidad de implementar componentes reutilizables, que permiten desarrollar interfaces complejas de forma eficiente y organizada.

React usa un sistema llamado virtual DOM (Document Object Model) que actualiza solo las partes necesarias de la página cuando cambian los datos, mejorando el rendiento notablemente. Además, promueve una forma declarativa de construir interfaces, facilitando la comprensión y el mantenimiento del código.

Se ha utilizado React para el desarrollo de la interfaz de la herramienta web PMT debido a su eficiencia en la creación de interfaces dinámicas, su enfoque basado en com-

ponentes reutilizables, y el uso del Virtual DOM, que mejora el rendimiento. Además, permite una fácil integración con bibliotecas externas, lo que agiliza el desarrollo y favorece la escalabilidad del proyecto.

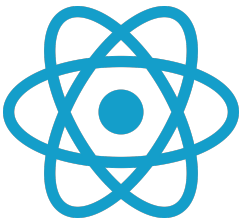


Figura 3.2: React

3.2.2. Vite

Vite (VoidZero Inc. and Vite Contributors, 2025) es una herramienta moderna de construcción y desarrollo para aplicaciones web, especialmente para proyectos con frameworks como React. Su principal ventaja es la velocidad: utiliza un servidor de desarrollo basado en módulos ES que permite recargas instantáneas al cambiar el código, eliminando tiempos de espera largos en el desarrollo.

Además, Vite realiza una compilación optimizada y eficiente para producción, generando paquetes ligeros y rápidos, lo que mejora el rendimiento final de la aplicación web.

Se ha utilizado Vite como herramienta de desarrollo y construcción para la plataforma web debido a su rápida velocidad de arranque y recarga en caliente, lo que mejora significativamente la experiencia durante el desarrollo. Además, Vite optimiza la compilación para producción, generando paquetes eficientes que contribuyen a un mejor rendimiento de la aplicación.



Figura 3.3: Vite

3.2.3. FastAPI

FastAPI (Ramírez, 2025) es un framework moderno y eficiente para construir APIs en Python. Se destaca por su velocidad, gracias a su soporte nativo para programación asíncrona, lo que permite manejar muchas solicitudes simultáneas sin perder rendimiento. Utiliza las anotaciones de tipo de Python para realizar validaciones automáticas de datos y generar documentación interactiva, facilitando el desarrollo y

mantenimiento del código. Su simplicidad y potencia lo convierten en una excelente opción para desarrollar backend robustos, escalables y fáciles de mantener.

Se ha utilizado FastAPI para desarrollar el backend de la plataforma debido a su capacidad para manejar de forma eficiente múltiples solicitudes simultáneas y procesar grandes volúmenes de datos provenientes del análisis de proyectos y modelos predictivos. Su validación automática de datos y rendimiento optimizado permitieron crear un servicio robusto y escalable, fundamental para la correcta ejecución de los procesos de predicción y análisis dentro de la herramienta.



Figura 3.4: FastAPI

3.2.4. PostgreSQL

PostgreSQL (PostgreSQL Global Development Group, 2025) es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto y altamente avanzado. Es conocido por su estabilidad, robustez y cumplimiento con los estándares SQL, además de ofrecer características avanzadas como soporte para datos JSON, extensiones, transacciones complejas y replicación.

Se utiliza ampliamente en aplicaciones que requieren integridad, escalabilidad y flexibilidad en el manejo de datos.

Se ha utilizado PostgreSQL como sistema de gestión de bases de datos para la plataforma debido a su robustez, escalabilidad y capacidad para manejar grandes volúmenes de datos relacionados con proyectos de software y resultados de análisis de mutantes. Su soporte avanzado para consultas complejas y manejo eficiente de información estructurada garantiza un almacenamiento fiable y rápido acceso a los datos necesarios para la predicción y evaluación dentro de la herramienta.

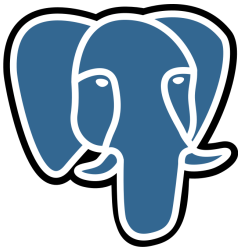


Figura 3.5: PostgreSQL_elephant

3.2.5. Figma

Figma (Figma Inc, 2025) es una herramienta de diseño de interfaces y prototipado que funciona directamente en el navegador. Permite crear y compartir diseños de forma colaborativa en tiempo real, facilitando la planificación visual de aplicaciones web antes del desarrollo.

Se ha utilizado Figma para diseñar los mockups de la plataforma web, ya que permite crear prototipos visuales de forma rápida y colaborativa. Esto facilitó la planificación de la estructura y apariencia de la interfaz antes de su implementación con React.

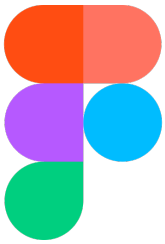


Figura 3.6: Figma

3.2.6. Draw.io

Draw.io (draw.io Ltd, 2025) es una herramienta para la creación de diagramas que puede ejecutarse tanto en el navegador como en aplicaciones de escritorio. Ofrece una amplia variedad de formas, conectores y plantillas que permiten representar procesos, arquitecturas de software y modelos conceptuales de manera clara y estructurada.

En este proyecto se ha utilizado Draw.io para elaborar los diagramas incluidos en la memoria, tales como diagramas de casos de uso, diagramas de actividad y representaciones conceptuales del sistema. Gracias a su facilidad de uso y a la posibilidad de exportar en múltiples formatos, ha resultado una herramienta práctica para documentar gráficamente el diseño y análisis del sistema.



Figura 3.7: Draw.io

3.2.7. Docker

Docker (Docker Inc, 2025) es una herramienta que permite crear contenedores, los cuales encapsulan una aplicación junto con todas sus dependencias. Esto asegura que la

aplicación se ejecute de forma idéntica en distintos entornos, facilitando su despliegue, prueba y distribución sin errores causados por diferencias en configuración o sistema operativo.

Se ha utilizado Docker para facilitar el despliegue de la plataforma web en servidores de alto rendimiento. Gracias al uso de contenedores, es posible garantizar que todos los componentes (análisis de proyectos, ejecución de modelos y gestión de múltiples peticiones) funcionen correctamente y de forma consistente, independientemente del entorno donde se instale la herramienta.

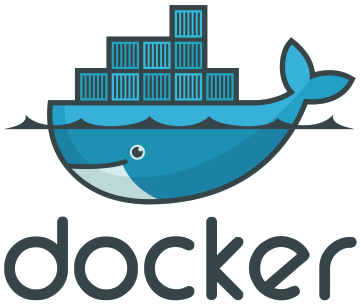


Figura 3.8: Docker

3.2.8. Visual Studio Code

Visual Studio Code (Microsoft, 2025) es un editor de código fuente desarrollado por Microsoft. VS Code se destaca por su extensibilidad, gracias a un amplio ecosistema de extensiones que permiten personalizarlo para diversos lenguajes de programación y flujos de trabajo.

Se ha utilizado Visual Studio Code como entorno de desarrollo debido a su ligereza, facilidad de uso y compatibilidad con los lenguajes y herramientas utilizadas. Sus extensiones y funcionalidades integradas han facilitado el desarrollo de la plataforma web, tanto en el frontend como en el backend y en el desarrollo de modelos, permitiendo trabajar de forma eficiente y organizada.



Figura 3.9: Visual Studio Code

3.2.9. GitHub

GitHub (GitHub Inc, 2025) es una plataforma en la nube que permite almacenar y controlar versiones del código fuente utilizando Git. Facilita el seguimiento de cambios,

el manejo de diferentes versiones y la organización del proyecto, asegurando un control ordenado y seguro durante todo el proceso de desarrollo.

Se ha utilizado GitHub para gestionar el código fuente de la plataforma y la elaboración de los modelos predictivos, permitiendo un control eficiente de las versiones y un seguimiento detallado de los cambios realizados. Esto contribuyó a un desarrollo ordenado y confiable tanto del sistema web como de los modelos de análisis.

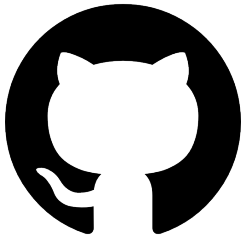


Figura 3.10: GitHub

3.2.10. Overleaf

Overleaf (Overleaf, 2025) es una plataforma online para la creación y edición de documentos en LaTeX, ampliamente utilizada en entornos académicos y científicos. Permite redactar textos técnicos, tesis y artículos de forma colaborativa y eficiente, sin necesidad de instalar software local. Su entorno facilita la compilación automática, el manejo de referencias y la organización estructurada del contenido, haciendo que la producción de documentos complejos sea más sencilla y rápida.

Se ha utilizado Overleaf para la elaboración de la documentación del proyecto debido a su facilidad para crear documentos en LaTeX de forma organizada y profesional. Esto permitió generar informes técnicos claros y bien estructurados de manera eficiente.



Figura 3.11: Overleaf

3.2.11. Wave

Wave (WebAIM, 2025) es una herramienta de evaluación de accesibilidad web que permite identificar de forma visual problemas relacionados con las WCAG. Destaca errores, alertas y elementos que pueden afectar la experiencia de usuarios con distintas

capacidades, proporcionando información sobre contraste, estructura semántica, navegación y otros aspectos clave. Su análisis visual facilita la detección y corrección de incidencias sin requerir conocimientos avanzados en accesibilidad.

En este proyecto se ha utilizado Wave a través de su extensión para Chrome con el objetivo de realizar pruebas de accesibilidad sobre la plataforma web. Gracias a esta herramienta, fue posible identificar y corregir elementos que podían dificultar el acceso a ciertos usuarios, garantizando así una experiencia más inclusiva y alineada con las buenas prácticas de diseño accesible.



Figura 3.12: Wave

3.2.12. Lighthouse

Lighthouse (Google, 2026) es una herramienta de auditoría web integrada en Google Chrome que permite evaluar de forma automática distintos aspectos de una aplicación web, incluyendo rendimiento, accesibilidad, buenas prácticas y SEO. En el ámbito de la accesibilidad, Lighthouse analiza el cumplimiento de criterios definidos en las WCAG, como el contraste de colores, la estructura semántica del HTML, la correcta definición de etiquetas, la navegación mediante teclado y el uso de atributos ARIA, proporcionando métricas y recomendaciones concretas para mejorar la experiencia de usuarios con distintas capacidades.

En este proyecto se ha utilizado Lighthouse a través de las herramientas de desarrollo de Chrome para realizar pruebas de rendimiento sobre la plataforma web PMTlab. Estas auditorías se han aplicado sobre distintas pantallas del sistema, con el objetivo de evaluar el tiempo de respuesta, la carga de recursos y la fluidez de la interfaz en escenarios representativos de uso real. De este modo, se ha podido comprobar si la plataforma ofrece un nivel de rendimiento adecuado para su utilización práctica.



Figura 3.13: Lighthouse

3.3. Planificación del proyecto

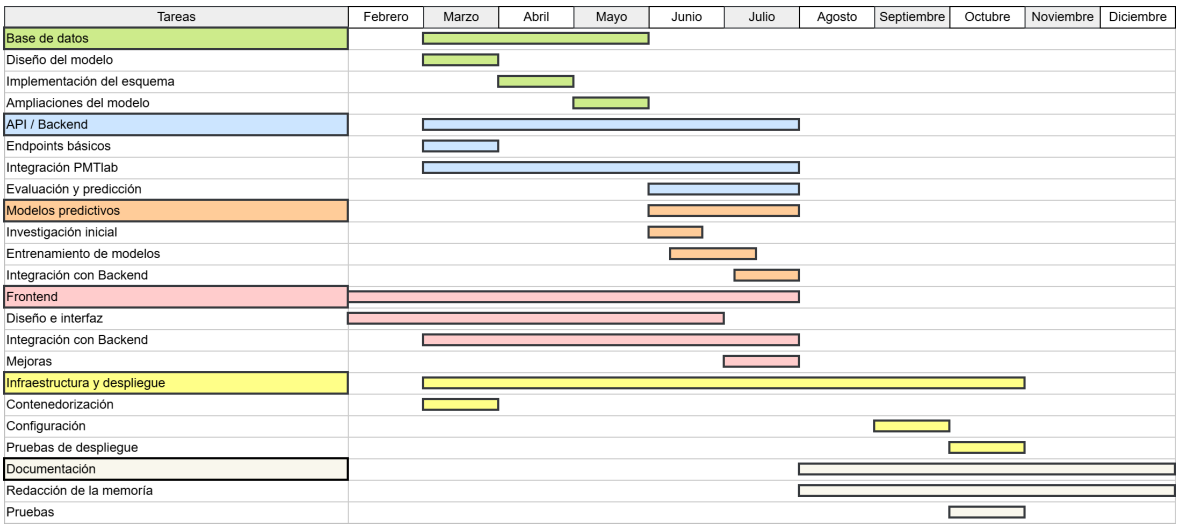


Figura 3.14: Diagrama de Gantt

3.4. Costes

En esta sección se presenta una estimación de los costes asociados al TFG, considerando tantos los recursos humanos como los materiales necesarios. El objetivo es aproximar el valor económico que tendría la realización del trabajo en un entorno profesional.

3.4.1. Costes de recursos humanos

Para realizar la estimación económica se ha consultado el salario promedio de un ingeniero de software junior en España en 2025, que se sitúa aproximadamente en 15,18 €/hora (Jooble, 2025).

La dedicación total al proyecto se ha estimado en unas 360 horas de trabajo distribuidas entre febrero y diciembre.

De este modo, los costes de recursos humanos asociados al desarrollo del TFG se calcula de la siguiente manera:

Rol	**Horas estimadas**	**Coste por hora (€)**	**Coste total (€)**
Ingeniero de software junior	360	15,18	5464,80

Tabla 3.1

Costes de recursos humanos

3.4.2. Costes de recursos materiales

Además de los recursos humanos, el desarrollo del proyecto requiere recursos materiales. Dentro de los materiales se incluyen el ordenador personal, el consumo eléctrico y la conexión a Internet. En cuanto al software, se han utilizado herramientas de carácter libre, por lo que el coste asociado es nulo.

Recurso material	**Coste unitario (€)**	**Coste total (€)**
React	0	0
Vite	0	0
FastAPI	0	0
PostgreSQL	0	0
Figma	0	0
Draw.io	0	0
Docker	0	0
Visual Studio Code	0	0
Git y GitHub	0	0
Overleaf	0	0
Figma	0	0
Ordenador personal	1500	1500
Electricidad (0,16779 €/kWh, consumo aprox. 68 kWh en 360h) (TotalEnergies, 2025)	0,16779	11,40
Conexión a Internet (tarifa mensual 35 €, 11 meses)	35	385
Total materiales		**1.896,40**

Tabla 3.2

Costes de recursos materiales

3.5. Riesgos

Riesgo	Impacto	Prob.	Plan de mitigación
Retrasos en la planificación	Aplazamiento de entregas y falta de tiempo para pruebas o documentación.	Media	Ajustar la planificación, priorizar tareas críticas y reservar tiempo extra.
Sobrecarga de trabajo académico/personal	Disminución de la dedicación al proyecto, afectando a la calidad final.	Alta	Organizar el calendario de entregas, dividir tareas en hitos pequeños y realistas.
Problemas técnicos con dependencias externas (librerías, frameworks, Docker, etc.)	Bloqueo temporal del desarrollo o necesidad de buscar alternativas.	Media	Documentar dependencias, mantener versiones estables y buscar soluciones alternativas.
Errores en la integración entre frontend y backend	Retrasos en pruebas, mal funcionamiento del sistema.	Media	Realizar pruebas incrementales y de integración frecuentes.
Dificultades en la implementación de modelos predictivos de IA	Resultados menos precisos o incompletos en la evaluación.	Media-Alta	Simplificar el modelo inicial, usar datasets de prueba y validar iterativamente.
Limitaciones de rendimiento en la ejecución de pruebas de mutación	Tiempo excesivo de ejecución, menor usabilidad de la plataforma.	Media	Optimizar la ejecución, aplicar técnicas de muestreo o limitar el número de mutantes.
Fallo de hardware personal (ordenador)	Pérdida de productividad y riesgo de pérdida de datos.	Baja	Realizar copias de seguridad en la nube y contar con equipos alternativos.
Pérdida de datos por fallo en control de versiones o copias de seguridad	Rehacer trabajo ya realizado.	Baja-Media	Uso riguroso de Git/-GitHub y copias en la nube.
Problemas de documentación o falta de claridad en requisitos	Rehacer parte del análisis o diseño.	Media	Revisar los requisitos con frecuencia y mantener documentación actualizada.
Subestimación del esfuerzo necesario	Proyecto incompleto o con menor calidad de la esperada.	Media-Alta	Definir objetivos alcanzables, priorizar funcionalidades clave y ajustar expectativas.

Tabla 3.3

Riesgos del proyecto, impacto, probabilidad y planes de mitigación

Parte II

Desarrollo

4. Requisitos del sistema

4.1. Objetivos de negocio

El desarrollo de esta plataforma tiene como propósito cubrir una necesidad actual en el ámbito de las pruebas de software, la inexistencia de herramientas que integren enfoques del PMT. Los principales objetivos son:

- Facilitar el uso del PMT mediante una plataforma que centralice la gestión de proyectos, mutantes, métricas y modelos, ofreciendo un entorno unificado y accesible para realizar análisis completos.
- Reducir el coste computacional asociado a la evaluación de la calidad de las pruebas, introduciendo un sistema que aprovecha modelos predictivos para priorizar mutantes y acelerar los análisis.
- Proveer una solución accesible y utilizable tanto en entornos académicos como profesionales, que permita experimentar con modelos, comparar resultados y explorar nuevas líneas de estudio o mejora en procesos de validación.

4.2. Objetivos del sistema

Para dar soporte a los objetivos de negocio, el sistema debe cumplir con una serie de metas técnicas y funcionales. Los objetivos principales del sistema son:

- Ofrecer una interfaz web intuitiva que permita a los usuarios interactuar con la plataforma sin necesidad de conocimientos técnicos avanzados.
- Permitir el registro, gestión y control de usuarios con distintos roles y niveles de permisos.
- Integrar un mecanismo de análisis de proyectos software, generando y almacenando mutantes y métricas asociadas.
- Incorporar modelos predictivos de PMT para optimizar la evaluación de pruebas.
- Facilitar la visualización y exportación de resultados de forma clara y accesible.

4.3. Actores del sistema

La plataforma debe identificar y gestionar distintos tipos de usuarios (actores) que interactúan con el sistema, asignando a cada uno un rol determinado con distintos niveles de permisos. Los roles definidos son los siguientes:

- `user`: Rol por defecto para los usuarios registrados. Puede acceder a las funcionalidades básicas de la plataforma, como el análisis de proyectos, la visualización y descarga de métricas, la selección de paleta de colores, y la evaluación de modelos previamente cargados (si están disponibles).
- `admin`: Este rol tiene, además de las funcionalidades del rol `user`, la capacidad de visualizar el listado de usuarios registrados, ver el historial de sesiones IP de un usuario, modificar el estado de la cuenta de un usuario (activar o bloquear), subir nuevos modelos predictivos a la plataforma, y gestionar proyectos analizados por otros usuarios.
- `superuser`: Es el rol con mayor nivel de permisos. Puede realizar todas las acciones disponibles para los roles anteriores, además de modificar el rol de otros usuarios.

Cada vez que un nuevo usuario se registra, el sistema asigna automáticamente el rol `user`. La promoción a `admin` o `superuser` solo puede ser realizada por un usuario con rol `superuser`.

Este esquema de actores permite controlar el acceso a funcionalidades sensibles del sistema y garantiza que cada usuario tenga una experiencia acorde a su nivel de responsabilidad dentro de la plataforma.

4.4. Requisitos funcionales

4.4.1. RF-01: Registro de usuario

La plataforma debe permitir que un usuario se registre proporcionando su dirección de correo eléctrónico, una contraseña segura, su nombre, su primer apellido. Además, el usuario puede introducir opcionalmente un segundo apellido, a qué organización pertenece (si es que pertenece a alguna) y marcar si es un estudiante o no. El usuario registrado a partir de la plataforma se le asignará el rol de `user`.

4.4.2. RF-02: Inicio de sesión

La plataforma debe ofrecer la opción de iniciar sesión a los usuarios previamente registrados, usando su correo electrónico y contraseña. El usuario dispondrá de un botón para visualizar la contraseña ocultada si es que lo desea.

4.4.3. RF-03: Menú principal y cierre de sesión

La plataforma debe mostrar un menú principal adaptado al rol del usuario autenticado.

- Los usuarios con rol `user` únicamente deben ver las opciones:
  - **Projects**
  - **Models**
  - **Cerrar sesión**
- Los usuarios con roles `admin` y `superuser` deben ver, además de las anteriores:
  - **Admin Dashboard**

El menú debe actualizarse automáticamente según el rol del usuario y garantizar que no se muestren opciones para las que no tenga permisos.

4.4.4. RF-04: Visualización de usuarios registrados

El sistema debe permitir a los usuarios con rol `admin` o `superuser` consultar el listado completo de usuarios registrados. La información mostrada debe incluir, al menos, los datos identificativos y el estado de la cuenta de cada usuario, de manera clara y estructurada para facilitar su revisión.

4.4.5. RF-05: Visualización del historial de sesiones IP de un usuario

La plataforma debe permitir la visualización del historial de sesiones de un usuario específico. En este historial se mostrarán la dirección IP del dispositivo desde el cual se accedió a la plataforma, así como la fecha y hora de inicio y cierre de sesión. Esta funcionalidad será accesible únicamente para usuarios con rol "admin" o "superuser".

4.4.6. RF-06: Modificación del estado de la cuenta de un usuario

La plataforma debe ofrecer la funcionalidad de cambiar el estado de la cuenta de un usuario. Los estados posibles son `Activo` y `Bloqueado`.

Si la cuenta está en estado `Activo`, el usuario podrá acceder a la plataforma y utilizar sus servicios. En cambio, si la cuenta se encuentra en estado `Bloqueado`, el usuario no podrá iniciar sesión ni hacer uso de la plataforma. Esta funcionalidad está destinada a gestionar el acceso de usuarios que hagan un uso inadecuado del sistema.

Solo los usuarios con rol `admin` o `superuser` tendrán acceso a esta funcionalidad.

4.4.7. RF-07: Modificación del rol de un usuario

La plataforma debe ofrecer la funcionalidad de modificar el rol asignado a un usuario. Los roles disponibles son: `user`, `admin` y `superuser`.

Esta funcionalidad será accesible únicamente para usuarios con rol `superuser`.

4.4.8. RF-08: Análisis de proyectos

La plataforma debe ofrecer la posibilidad de analizar proyectos subidos a GitHub, ya sea proporcionando un enlace individual a un repositorio o subiendo un archivo CSV que contenga una lista de enlaces. Esta lista podrá estar organizada en una columna o fila, con los enlaces separados por comas.

Los enlaces deberán corresponder a repositorios válidos, y el sistema tomará siempre la rama principal del proyecto (por defecto `main` o aquella configurada como rama predeterminada en el repositorio) para realizar el análisis.

Estos proyectos estarán identificados por su nombre y su *commit*. Si se sube un proyecto que ya fue analizado y no ha cambiado su *commit* desde el último análisis, no será necesario volver a analizarlo, ya que los datos correspondientes ya estarán almacenados en la base de datos.

En caso de que se detecte el mismo *commit* en un proyecto cuyo análisis haya fallado previamente, se solicitará al usuario que suba un nuevo *commit* a GitHub, con el objetivo de evitar errores repetidos en el análisis y el desperdicio de recursos.

Durante el análisis, el sistema generará los mutantes del proyecto y obtendrá sus métricas mediante herramientas de terceros encargadas del análisis estático y dinámico del código, las cuales son PIT, OpenClover y JaSoMe. Estas herramientas permiten extraer información relevante del mutante y de su contexto en el código, incluyendo métricas estructurales, de complejidad, cobertura y características específicas del operador aplicado. Todos los datos resultantes serán almacenados en la base de datos para su posterior consulta y utilización por otras funcionalidades de la plataforma.

4.4.9. RF-09: Eliminación de proyectos del historial

La plataforma debe ofrecer la opción de eliminar, del historial de proyectos subidos por el propio usuario, aquellos proyectos que este considere necesario retirar. Esta acción únicamente elimina la asociación entre el usuario y el proyecto analizado, pero no borra las métricas ni los resultados almacenados en la base de datos, ya que estos pueden seguir siendo utilizados por otras funcionalidades del sistema, como la evaluación de modelos predictivos o mostrar las métricas sin tener que analizar de nuevo el proyecto si otro usuario sube el mismo proyecto.

4.4.10. RF-10: Visualización métricas de proyectos analizados

La plataforma debe ofrecer la opción de visualizar las métricas generadas durante el proceso de análisis. El usuario podrá seleccionar uno o varios proyectos previamente analizados y consultar sus métricas en una interfaz que permita elegir y cambiar entre los distintos proyectos.

4.4.11. RF-11: Descarga de métricas de proyectos analizados

La plataforma debe ofrecer la opción de descargar en formato CSV las métricas generadas durante el proceso de análisis. El usuario podrá seleccionar uno o varios proyectos previamente analizados. Si se selecciona un solo proyecto, se descargará un archivo CSV con sus métricas. Si se seleccionan varios, se generará un archivo ZIP que contendrá un CSV individual por cada proyecto seleccionado. Esto facilita el almacenamiento y el análisis externo de los resultados.

4.4.12. RF-12: Subida de modelos PMT

La plataforma debe permitir la subida de modelos predictivos utilizados para estimar el estado de los mutantes generados durante el análisis de los proyectos. Para ello, el sistema admite únicamente modelos previamente entrenados y exportados en formatos compatibles. Los tipos de modelos actualmente operativos y sus extensiones permitidas son:

- **RandomForest**: `.joblib`, `.pkl`
- **XGBoost**: `.bin`, `.joblib`
- **TabNet**: `.zip`, `.pt`

La interfaz incluye otros tipos de modelos con el fin de mostrar la capacidad de extensión de la plataforma; sin embargo, dichos modelos no están habilitados en esta versión y no permiten la carga de archivos.

Para realizar la subida, el usuario deberá seleccionar el tipo de modelo compatible y proporcionar el archivo correspondiente desde su dispositivo. El sistema validará la extensión del fichero y almacenará el modelo en el servidor para su posterior uso en el proceso de evaluación. Esta funcionalidad estará disponible únicamente para usuarios con rol `admin` o `superuser`.

4.4.13. RF-13: Evaluación de modelos PMT

La plataforma debe permitir la evaluación de los modelos predictivos previamente subidos al servidor. Esta evaluación se realiza aplicando dichos modelos sobre las métricas generadas durante el análisis de un proyecto, con el objetivo de predecir el estado de cada mutante.

El usuario podrá seleccionar uno o varios modelos disponibles, así como un proyecto ya analizado. El sistema utilizará directamente las métricas almacenadas en la base de datos para calcular las predicciones sin necesidad de reentrenar el modelo.

Es importante señalar que la plataforma no realiza procesos de entrenamiento: los modelos deben estar completamente entrenados antes de su subida. Por tanto, el uso de datos no empleados en el entrenamiento previo (por ejemplo, nuevos proyectos destinados a prueba) debe ser gestionado manualmente por el usuario, siendo la plataforma únicamente responsable de ejecutar la inferencia.

Una vez finalizada la evaluación, el sistema redirigirá al usuario a una pantalla donde se mostrarán las métricas obtenidas para cada modelo seleccionado (p. ej., precisión, recall, F1-score y MCC), permitiendo comparar y visualizar de forma independiente los resultados.

4.4.14. RF-14: Filtrado y ordenación de datos

La plataforma debe ofrecer una serie de filtros y métodos de ordenación que permitan al usuario visualizar los datos de manera sencilla e intuitiva.

4.4.15. RF-15: Selección de la paleta de colores de la plataforma

La plataforma debe permitir al usuario seleccionar la paleta de colores que considere visualmente más atractiva o accesible. Las opciones disponibles incluirán los temas: `Claro`, `Oscuro` y `Azul`.

4.5. Requisitos no funcionales

4.5.1. RNF-01: Usabilidad

La plataforma debe permitir al usuario filtrar y ordenar la información mostrada en las tablas de datos, con el fin de facilitar la búsqueda, comparación y análisis de mutantes, métricas o modelos evaluados.

Los filtros disponibles serán los que proporciona el componente de tabla utilizado en la interfaz e incluirán, entre otros:

- **Filtrado por texto**: permitir buscar coincidencias parciales en columnas como el identificador del mutante, el nombre del proyecto o el operador de mutación.
- **Filtrado por valores numéricos**: permitir establecer condiciones tales como "mayor que", "menor que" o "igual a" en métricas cuantitativas (por ejemplo, *MTLOC*, *VG*, *WMC*, etc.).

- **Filtrado por categorías**: permitir seleccionar valores concretos en columnas categóricas, como el estado del mutante (*killed/survived*) o el resultado de un modelo predictivo.

  La ordenación disponible incluirá:

- **Ordenación ascendente y descendente** por cualquiera de las columnas visualizadas en la tabla.

  Además, la plataforma debe permitir:

- **Seleccionar y ocultar columnas**: ofreciendo al usuario la posibilidad de mostrar únicamente las columnas que considere relevantes para su análisis.

El usuario podrá aplicar filtros y ordenaciones desde los controles integrados en el encabezado de cada columna, actualizándose automáticamente la información que aparece en la tabla sin necesidad de recargar la página.

4.5.2. RNF-02: Accesibilidad

La plataforma debe cumplir con los principios de accesibilidad definidos en las Pautas de Accesibilidad al Contenido en la Web (WCAG 2.1), incluyendo soporte para navegación mediante teclado, contraste adecuado entre texto y fondo, y soporte para lectores de pantalla. También debe ofrecer opciones de personalización como el cambio de paleta de colores.

4.5.3. RNF-03: Fiabilidad

La plataforma debe ser capaz de mantener su funcionamiento continuo durante un 99.5 % del tiempo mensual (disponibilidad), excluyendo mantenimientos programados. Además, ante errores en el análisis de proyectos o evaluación de modelos, el sistema debe registrar los fallos y notificar adecuadamente al usuario sin comprometer la estabilidad general.

4.5.4. RNF-04: Rendimiento y eficiencia

La plataforma debe mantener un comportamiento estable y tiempos de respuesta aceptables en las principales pantallas del sistema, tanto en estado inactivo como durante la ejecución de procesos intensivos, como el análisis de proyectos.

4.5.5. RNF-05: Seguridad

La plataforma debe proteger los datos personales de los usuarios mediante encriptación de contraseñas (por ejemplo, usando bcrypt) y el uso de HTTPS para todas las comunicaciones. La autenticación se gestionará mediante tokens JWT (JSON Web

Tokens), que serán firmados y tendrán una expiración configurable (por defecto: 60 minutos). Tras ese tiempo, el token expirará y el usuario deberá autenticarse nuevamente.

Además, el acceso a funcionalidades sensibles estará restringido exclusivamente a usuarios con los roles adecuados, siguiendo el principio de mínimo privilegio para garantizar una mayor protección del sistema.

4.5.6. RNF-06: Mantenibilidad

El sistema debe estar desarrollado siguiendo principios de arquitectura modular y buenas prácticas de codificación (por ejemplo, separación de responsabilidades y uso de control de versiones). Esto facilitará la corrección de errores y la incorporación de nuevas funcionalidades en el futuro.

4.5.7. RNF-07: Portabilidad

La plataforma debe ser accesible desde los principales navegadores web modernos (Chrome, Firefox, Edge, Opera) y debe funcionar correctamente en dispositivos de escritorio, portátiles y tablets con resolución mínima de 768px de ancho. La adaptación a dispositivos móviles de 480px será progresiva y no prioritaria en la primera versión.

4.5.8. RNF-08: Escalabilidad

El sistema debe diseñarse de forma que pueda ampliarse para soportar un mayor número de usuarios concurrentes, nuevos tipos de análisis, o modelos predictivos adicionales sin necesidad de una reestructuración significativa del código base.

4.5.9. RNF-09: Idioma de la plataforma

La plataforma debe estar disponible en idioma inglés, con el objetivo de garantizar una mayor accesibilidad e inclusión a nivel internacional. Todos los textos de la interfaz, mensajes del sistema y etiquetas deben estar en inglés.

4.6. Requisitos de información

4.6.1. RI-01: Información de usuarios registrados

El sistema debe almacenar la información básica de cada usuario registrado. Esta incluye: dirección de correo electrónico (única por usuario), contraseña cifrada (por ejemplo, usando `bcrypt`), nombre, primer apellido, segundo apellido (opcional), organización (opcional) y si el usuario se identifica como estudiante. Esta información

permitirá personalizar la experiencia del usuario. Además, debe registrarse automáticamente el rol asignado al usuario al momento del registro, siendo por defecto `user`.

4.6.2. RI-02: Estado de la cuenta de usuario

El sistema debe almacenar el estado actual de la cuenta de cada usuario, que puede ser `Activo` o `Bloqueado`. Este estado condiciona la posibilidad de iniciar sesión: los usuarios con estado `Activo` podrán acceder y utilizar la plataforma, mientras que aquellos con estado `Bloqueado` no podrán iniciar sesión ni hacer uso del sistema.

4.6.3. RI-03: Historial de sesiones de usuario

El sistema debe registrar cada sesión iniciada por los usuarios, incluyendo dirección IP, fecha y hora de inicio y de cierre de sesión. Esta información es importante para motivos de auditoría, detección de accesos sospechosos y gestión de la actividad en la plataforma. Estos datos serán accesibles solo para usuarios con rol `admin` o `superuser`.

4.6.4. RI-04: Información sobre proyectos analizados

La información correspondiente a cada proyecto analizado se almacena en la base de datos mediante un conjunto de tablas relacionadas. Cada proyecto incluye datos como el nombre del repositorio, el hash del *commit* analizado, la fecha de análisis, el estado de la ejecución, los estados intermedios del análisis, los logs generados durante el proceso y un identificador único asociado.

Dado que un mismo proyecto puede ser subido por distintos usuarios, la plataforma gestiona esta relación mediante una tabla intermedia que vincula usuarios y proyectos, permitiendo identificar quiénes han solicitado el análisis de cada uno de ellos sin duplicar la información del propio proyecto.

Durante el análisis, el sistema actualiza el estado del proyecto según el punto del flujo en el que se encuentre. Estos estados están definidos mediante tipos enumerados en la base de datos e indican tanto el progreso general del análisis como su resultado técnico. Entre ellos se incluyen fases como la inicialización del proyecto, la clonación correcta del repositorio, la finalización del análisis y la subida de resultados, así como posibles fallos provocados por errores de compilación, ejecución de pruebas o tiempos de espera superados. Esta estructura permite identificar de forma precisa en qué paso se encuentra cada proyecto y facilita la detección de incidencias durante el proceso.

Los logs almacenados incluyen mensajes sobre el resultado de la clonación, compilación y análisis, advertencias emitidas por las herramientas externas, detalles de errores internos y cualquier incidencia relevante ocurrida durante el proceso. Estos registros permiten trazar el análisis completo y facilitan la identificación y corrección de problemas.

Para evitar análisis redundantes, la plataforma comprueba el *commit hash* del proyecto antes de iniciar el análisis. Si ese mismo *commit* ya ha sido procesado previamente, el sistema reutiliza los resultados ya existentes, evitando un análisis innecesario y reduciendo significativamente el consumo de recursos.

4.6.5. RI-05: Métricas generadas por análisis

El sistema debe registrar las métricas obtenidas tras el análisis de cada proyecto. Estas métricas incluyen, entre otras: operador de mutación, número de tests cubiertos, tipo de retorno, complejidad ciclomática y métricas específicas asociadas a los mutantes generados. Cada conjunto de métricas debe estar vinculado a un proyecto concreto y a un *commit* determinado. Esta información será utilizada posteriormente para su visualización, descarga en formato CSV y evaluación mediante modelos predictivos.

4.6.6. RI-06: Información de modelos PMT

El sistema debe almacenar los datos de cada modelo predictivo subido, incluyendo: nombre del modelo, tipo (por ejemplo, `RandomForest`, `XGBoost`, `TabNet`), fecha de subida, formato de archivo, usuario que lo subió, versión y ruta del fichero que contiene el modelo.

El tipo de modelo no puede ser creado libremente por el usuario: debe seleccionarse entre los tipos predefinidos que ofrece la plataforma. De este modo, se garantiza la consistencia y compatibilidad con los modelos soportados internamente.

4.6.7. RI-07: Resultados de evaluación de modelos

El sistema debe guardar los resultados obtenidos al evaluar un modelo predictivo con un proyecto. Esto incluye el modelo utilizado, el proyecto analizado, métricas de rendimiento obtenidas (precisión, F1-score, mcc, auc_roc y métricas específicas de cada clase, en este caso, el estado de los mutantes), y la fecha de evaluación. Esta información podrá visualizarse posteriormente para comparar distintos modelos.

5. Análisis del sistema

5.1. Modelo conceptual

Dado que el sistema hace uso de una base de datos relacional para almacenar la información gestionada por la plataforma, el modelo conceptual se ha representado mediante un diagrama entidad–relación (ER).

Este modelo se ha definido a un nivel de abstracción alto, centrándose en identificar las principales entidades del sistema y las relaciones existentes entre ellas, sin entrar en el detalle de los atributos específicos de cada entidad. La definición concreta de dichos atributos se aborda posteriormente en la fase de diseño físico de datos.

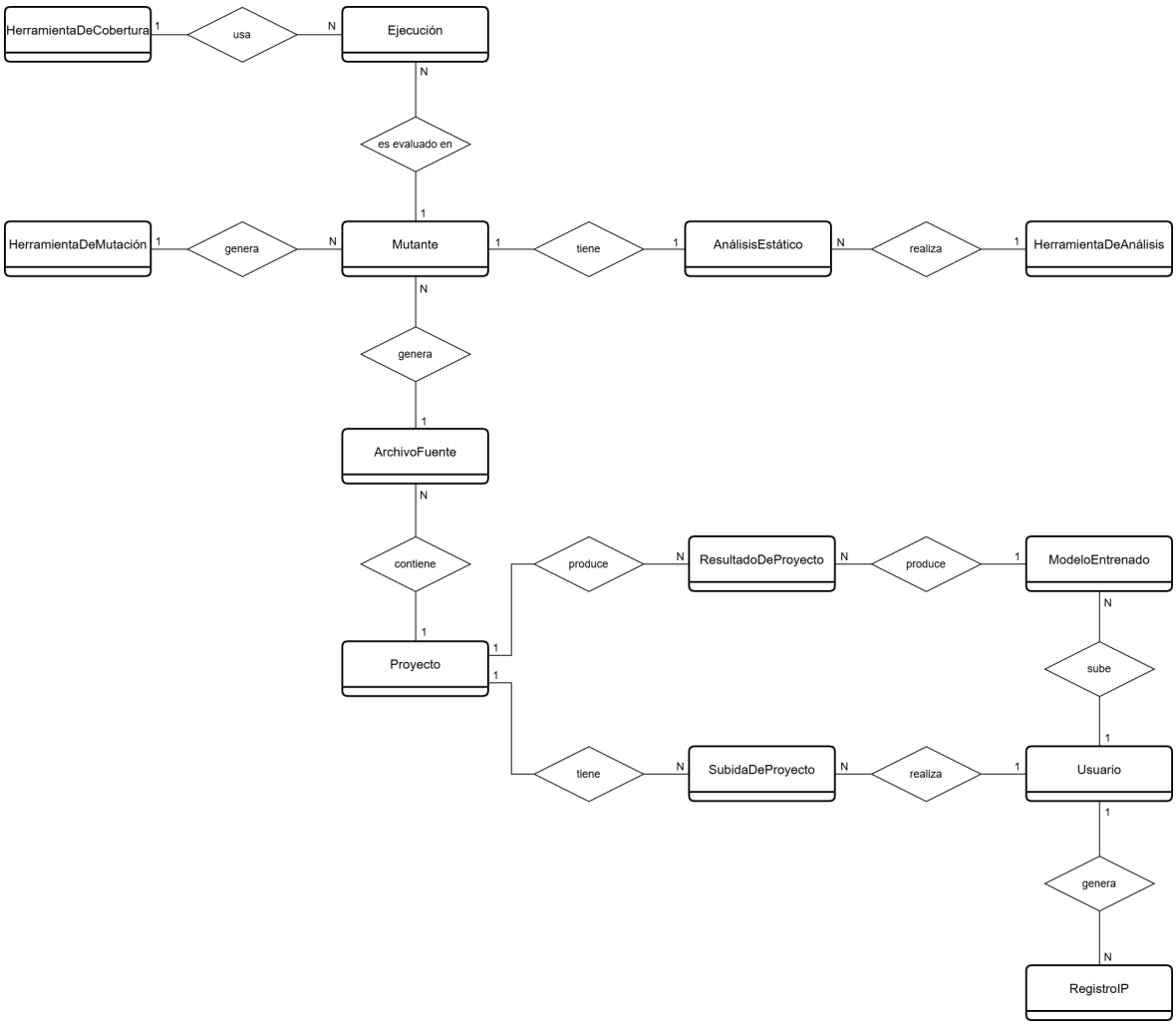


Figura 5.1: Diagrama conceptual

5.2. Modelo de casos de uso

Se han realizado tres diagramas, cada uno correspondiente a un actor distinto. Los casos de uso definidos para los usuarios corrientes también se aplican a los actores `Admin` y `Superuser`.

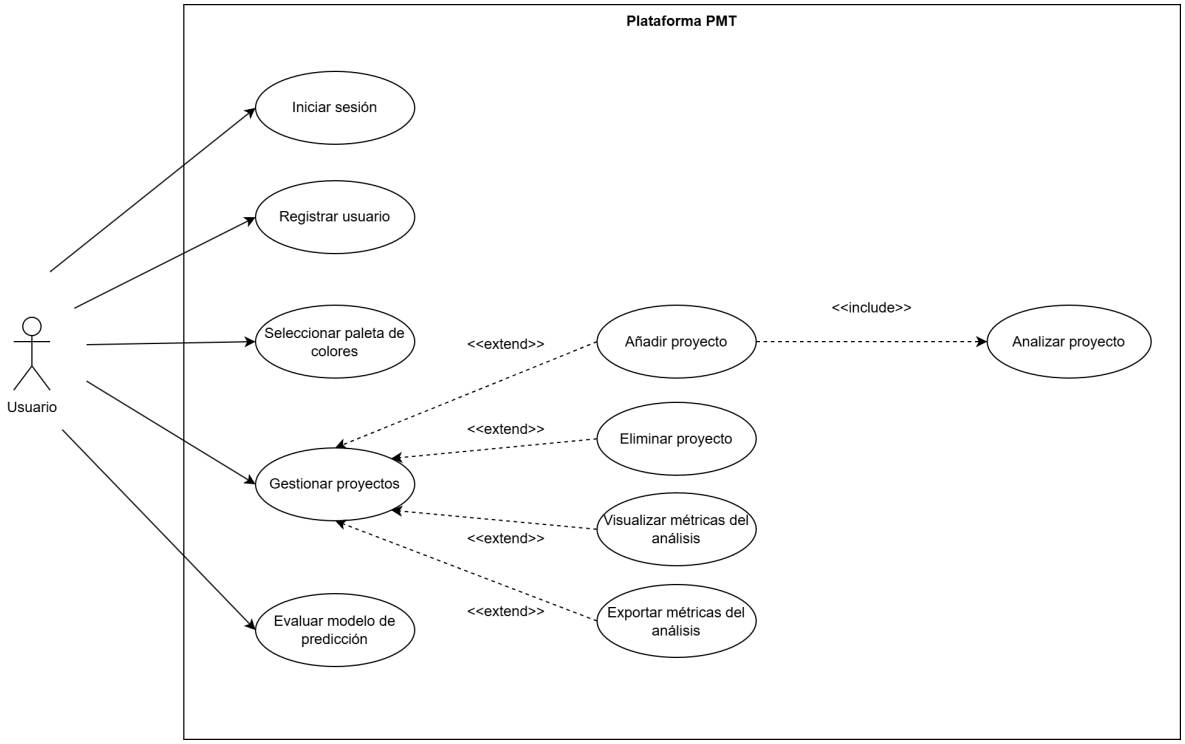


Figura 5.2: Diagrama de casos de uso: Usuario

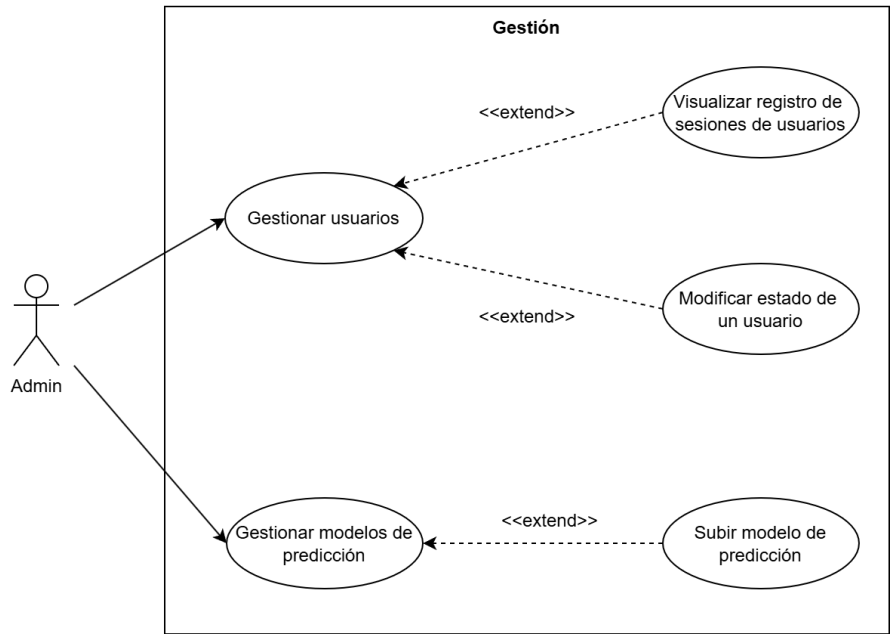


Figura 5.3: Diagrama de casos de uso: Admin

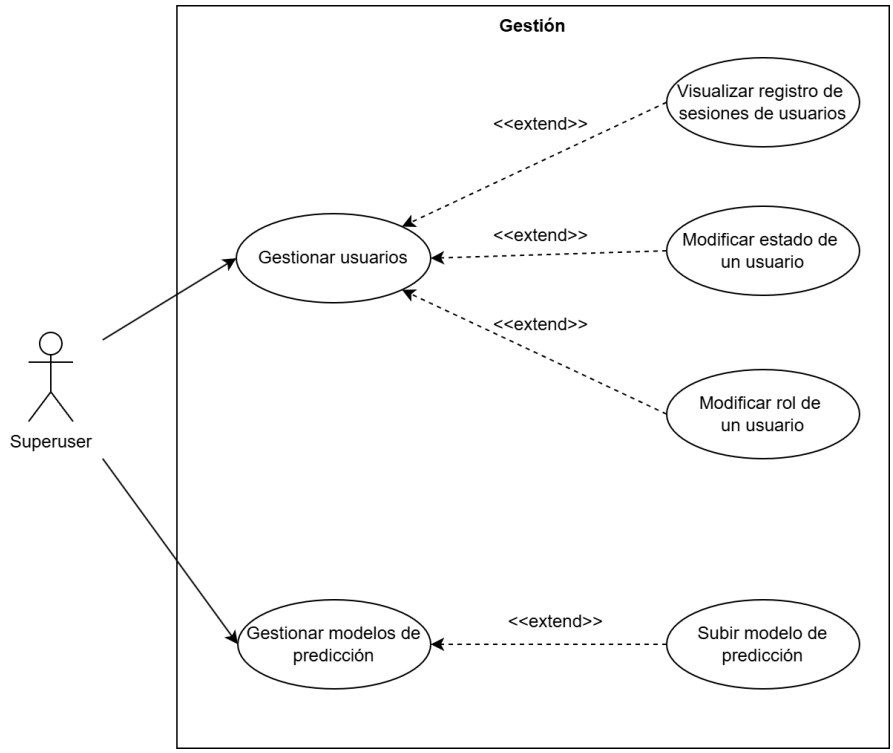


Figura 5.4: Diagrama de casos de uso: Superuser

5.3. Casos de uso del sistema

CU-01	**Registrar usuario**
Actor	Usuario corriente
Precondiciones	No existe un usuario con el mismo correo electrónico.
Postcondiciones	Se crea una cuenta para el usuario.
Escenario principal	
1	El usuario accede a la ventana de registro de usuario.
2	El usuario introduce la información en los campos obligatorios (Correo electrónico, contraseña, nombre y primer apellido).
3	El usuario envía la solicitud de registro de usuario.
4	El sistema comprueba que los campos obligatorios hayan sido cumplimentados.
5	El sistema verifica los datos ingresados (E-mail con formato correcto y no registrado previamente). Si son válidos, se crea una nueva cuenta y el usuario es redirigido a la ventana de inicio de sesión.
Escenarios alternativos	
2.A	**Registro con campos obligatorios y opcionales**
2.A.1	El usuario introduce la información en los campos obligatorios (Correo electrónico, contraseña, nombre y primer apellido) y opcionales (Segundo apellido, organización y condición de estudiante).
2.A.2	Se continua en el paso 3.
4.A	**Campos obligatorios sin cumplimentar**
4.A.1	El usuario deja algunos campos obligatorios sin completar y el sistema muestra un mensaje indicando cuáles faltan para poder continuar con el registro.
4.B	**Formato de correo electrónico erróneo**
4.B.1	Si el usuario introduce un correo con formato inválido, el sistema muestra un mensaje de error y no permite continuar hasta que se corrija.
5.A	**Correo electrónico previamente registrado**
5.A.1	Si el correo electrónico ingresado ya está registrado, el sistema notifica por pantalla al usuario con un mensaje de error e impide completar el registro.

Tabla 5.1

Caso de uso: Registrar usuario

CU-02	**Iniciar sesión**
Actor	Usuario, Admin y Superuser
Precondiciones	Debe existir la cuenta del usuario que intenta iniciar sesión.
Postcondiciones	El usuario inicia sesión.
Escenario principal	
1	El usuario accede a la ventana de inicio de sesión.
2	El usuario introduce el correo electrónico y su contraseña para iniciar sesión.
3	El usuario envía la solicitud de inicio de sesión.
4	El sistema comprueba que los campos hayan sido cumplimentados.
5	El sistema verifica las credenciales ingresadas. Si son válidas, se inicia sesión correctamente y el usuario es redirigido a la página de inicio correspondiente según rol.
Escenarios alternativos	
4.A	**Campos obligatorios sin cumplimentar**
4.A.1	El usuario deja algunos campos obligatorios (Correo electrónico o contraseña) sin completar y el sistema muestra un mensaje indicando cuáles faltan para poder continuar con el inicio de sesión.
4.B	**Formato de correo electrónico erróneo**
4.B.1	Si el usuario introduce un correo con formato inválido, el sistema muestra un mensaje de error y no permite continuar hasta que se corrija.
5.A	**Credenciales erróneas**
5.A.1	Si el usuario introduce un correo electrónico o contraseña incorrectos, el sistema muestra un mensaje de error indicando que las credenciales no son válidas y solicita que se ingresen nuevamente para continuar.

Tabla 5.2

Caso de uso: Iniciar sesión

CU-03	**Seleccionar paleta de colores**
Actor	Usuario, Admin y Superuser
Precondiciones	Ninguna.
Postcondiciones	El usuario selecciona la paleta de color y se aplican los colores de la paleta a la plataforma.
Escenario principal	
1	El usuario pulsa el selector de paletas de colores.
2	El usuario selecciona la paleta entre las opciones disponibles (`Claro`, `Oscuro` y `Azul`).
3	El sistema cambia los colores determinados por la paleta a los distintos componentes de la plataforma.
Escenarios alternativos	
***.A**	**Cancelar**
*.A.1	El usuario toca cualquier parte de la pantalla que no sea el selector para cancelar la acción.
*.A.2	El sistema oculta las opciones del selector.

Tabla 5.3

Caso de uso: Seleccionar paleta de colores

CU-04	**Gestionar proyectos**
Actor	Usuario, Admin y Superuser
Precondiciones	El usuario ha sido autenticado correctamente.
Postcondiciones	Ninguna.
Escenario principal	
1	El usuario accede a la ventana Proyectos.
2	El sistema muestra un datagrid con la información de todos los proyectos subidos a la plataforma por el usuario. En caso de ser Admin o Superuser, se mostrarán todos los proyectos cargados a la plataforma.
Extensiones	
2.A	**Añadir proyecto**
2.A.1	El usuario pulsa el botón Añadir proyecto y el sistema muestra un pop-up para poder añadir un proyecto para analizar. Se inicia el caso de uso **CU-05**: "Añadir proyecto".
2.B	**Borrar proyecto**
2.B.1	El usuario selecciona un proyecto y el sistema habilita el botón Borrar Proyectos. Se inicia el caso de uso **CU-07**: "Borrar proyecto".
2.C	**Ver métricas del análisis**
2.C.1	El usuario selecciona un proyecto y el sistema habilita el botón Visualizar Métricas. Se inicia el caso de uso **CU-08**: "Ver métricas del análisis".
2.D	**Exportar métricas del análisis**
2.D.1	El usuario selecciona un proyecto y el sistema habilita el botón Exportar Métricas. Se inicia el caso de uso **CU-09**: "Exportar métricas del análisis".

Tabla 5.4

Caso de uso: Gestionar proyectos

CU-05	**Añadir proyecto**
Actor	Usuario, Admin y Superuser
Precondiciones	El usuario ha sido autenticado correctamente.
Postcondiciones	Se añade el proyecto a la plataforma y se procede a su análisis.
Escenario principal	
1	El usuario accede a la ventana Proyectos.
2	El usuario pulsa el botón Añadir Proyecto y el sistema muestra un formulario con los campos necesarios para completar el caso de uso.
3	El usuario introduce el enlace al repositorio de GitHub.
4	El usuario pulsa el botón Confirmar.
5	El sistema guarda en la base de datos el proyecto añadido y muestra un mensaje de éxito. Se procede directamente al análisis de dicho proyecto.
Escenarios alternativos	
2.A	**Cancelar**
2.A.1	El usuario pulsa el botón Cancelar o pulsa fuera del formulario.
2.A.2	El sistema oculta el pop-up para añadir proyectos.
3.A	**Introducir varios repositorios**
3.A.1	El usuario sube un fichero CSV con una lista de enlaces a repositorios (separados por comas o dispuestos en columna).
3.A.2	Se continua en el paso 4.
6.A	**Error al añadir proyecto**
6.A.1	El sistema intenta añadir el proyecto.
6.A.2	Ocurre un error durante el proceso de obtención del proyecto.
6.A.3	El sistema muestra un mensaje indicando que no se ha podido añadir el proyecto de predicción.

Tabla 5.5

Caso de uso: Añadir proyecto

CU-06	**Analizar proyecto**
Actor	Usuario, Admin y Superuser
Precondiciones	El usuario ha sido autenticado correctamente. El proyecto se encuentra almacenado en la plataforma y es accesible para su análisis.
Postcondiciones	El sistema almacena las métricas y resultados obtenidos del análisis del proyecto, dejándolos disponibles para consulta y exportación.
Escenario principal	
1	El sistema recibe la solicitud de analizar un proyecto (de forma automática tras su adición).
2	El sistema obtiene el último commit del repositorio del proyecto.
3	El sistema clona el repositorio.
4	El sistema ejecuta las herramientas de análisis para extraer métricas relevantes.
5	El sistema procesa y almacena los resultados obtenidos en la base de datos.
6	El sistema notifica al usuario que el análisis ha finalizado con éxito y que las métricas están disponibles para su consulta.
Escenarios alternativos	
2.A	**Proyecto previamente analizado**
2.A.1	Se detecta que el proyecto ya existe en el sistema y que ha sido analizado correctamente.
2.A.2	El sistema muestra un mensaje de proyecto analizado.
2.B	**Proyecto con análisis incorrecto**
2.B.1	Se detecta que el proyecto ya existe en el sistema y que ha habido un error en un análisis previo.
2.B.2	El sistema muestra un mensaje de proyecto con análisis incorrecto.
3.A	**Error en la obtención del repositorio**
3.A.1	El sistema no puede clonar el repositorio (por URL inválida, permisos o fallo de conexión).
3.A.2	El sistema muestra un mensaje de error indicando el motivo y no se inicia el análisis.
4.A	**Error durante el análisis**
4.A.1	Ocurre un fallo en la ejecución de las herramientas de análisis.
4.A.2	El sistema registra el error y notifica al usuario que no se han podido obtener todas las métricas.

Tabla 5.6

Caso de uso: Analizar proyecto

CU-07	**Eliminar proyecto**
Actor	Usuario, Admin y Superuser
Precondiciones	El usuario ha sido autenticado correctamente. Deben existir proyectos añadidos.
Postcondiciones	Se eliminan los proyectos de la lista de proyectos del usuario.
Escenario principal	
1	El usuario accede a la ventana Proyectos.
2	El usuario selecciona los proyectos que desea eliminar y el sistema habilita el botón de ”Borrar Proyectos”.
3	El usuario pulsa el botón de “Eliminar Proyectos”.
4	El sistema elimina de la lista de proyectos del usuario los proyectos seleccionados.
Escenarios alternativos	
2.A	**Cancelar**
2.A.1	El usuario deselecciona los proyectos.
2.A.2	El sistema oculta el botón de “Eliminar Proyectos”.

Tabla 5.7

Caso de uso: Eliminar proyecto

CU-08	**Visualizar métricas del análisis**
Actor	Usuario, Admin y Superuser
Precondiciones	El usuario ha sido autenticado correctamente. Los proyectos deben haber sido analizados con éxito previamente.
Postcondiciones	Se redirige al usuario a la página donde se muestran las métricas de los proyectos seleccionados.
Escenario principal	
1	El usuario accede a la ventana Proyectos.
2	El usuario selecciona los proyectos de los que desea visualizar las métricas y el sistema habilita el botón "Visualizar Métricas".
3	El usuario pulsa el botón "Visualizar Métricas".
4	El sistema redirige al usuario a la página de métricas de los proyectos seleccionados.
Escenarios alternativos	
2.A	**Cancelar**
2.A.1	El usuario deselecciona los proyectos.
2.A.2	El sistema oculta el botón "Visualizar Métricas".

Tabla 5.8

Caso de uso: Visualizar métricas del análisis

CU-09	**Exportar métricas**
Actor	Usuario, Admin y Superuser
Precondiciones	El usuario ha sido autenticado correctamente. Los proyectos han sido analizados correctamente.
Postcondiciones	El usuario obtiene un fichero CSV con las métricas del análisis de los proyectos.
Escenario principal	
1	El usuario accede a la ventana Proyectos.
2	El usuario selecciona los proyectos, de la lista de proyectos disponibles, y el sistema habilita el botón Exportar Métricas.
3	El usuario pulsa el botón Exportar Métricas.
4	El sistema inicia la descarga de las métricas del análisis en formato CSV y muestra un mensaje de éxito.
Escenarios alternativos	
2.A	**Cancelar**
2.A.1	El usuario deselecciona los proyectos.
2.A.2	El sistema oculta el botón Exportar Métricas.
4.A	**Error en la exportación**
4.A.1	Se produce un error en el sistema al intentar exportar las métricas.
4.A.2	El sistema muestra un mensaje de error.

Tabla 5.9

Caso de uso: Exportar métricas

CU-10	**Evaluar modelo de predicción**
Actor	Usuario, Admin y Superuser
Precondiciones	El usuario ha sido autenticado correctamente. El proyecto con el que se desea evaluar debe haber sido analizado con éxito previamente y debe existir un modelo subido a la plataforma.
Postcondiciones	Se redirige al usuario a la página donde se muestran los resultados de la evaluación del modelo de predicción y proyecto seleccionados.
Escenario principal	
1	El usuario accede a la ventana de Modelos.
2	El usuario selecciona el modelo que desea evaluar, de la lista de modelos disponibles, y el sistema habilita el botón Evaluar Modelos.
3	El usuario pulsa el botón de Evaluar Modelos y el sistema muestra un formulario para seleccionar el proyecto con el que evaluar.
4	El usuario selecciona el proyecto con el que evaluar el modelo de predicción seleccionado.
5	El usuario pulsa el botón Confirmar.
6	El sistema evalúa el modelo y, al finalizar, redirige al usuario a la página de resultados para consultar los datos obtenidos.
Escenarios alternativos	
2.A	**Cancelar evaluación de modelo**
2.A.1	El usuario deselecciona el modelo.
2.A.2	El sistema oculta el botón de Evaluar Modelos.
4.A	**Cancelar selección de proyecto**
4.A.1	El usuario pulsa el botón Cancelar o pulsa fuera del formulario.
4.A.2	El sistema oculta el pop-up de selección de proyecto.
6.A	**Error en la evaluación**
6.A.1	El sistema intenta ejecutar la evaluación del modelo.
6.A.2	Ocurre un error durante el proceso de evaluación.
6.A.3	El sistema muestra un mensaje indicando que no se ha podido completar la evaluación.

Tabla 5.10

Caso de uso: Evaluar modelo de predicción

CU-11	**Gestionar usuarios**
Actor	Admin y Superuser
Precondiciones	El usuario ha sido autenticado correctamente. Deben existir usuarios en la base de datos.
Postcondiciones	Ninguna.
Escenario principal	
1	El usuario accede a la ventana Admin Dashboard.
2	El sistema muestra un datagrid con la información de todos los usuarios registrados en la plataforma.
Extensiones	
2.A	**Ver registro de sesiones de un usuario**
2.A.1	El usuario pulsa el botón Acceder de otro usuario y el sistema lo redirige a la página de detalles del usuario donde puede ver las sesiones de este último. Se inicia el caso de uso **CU-12**: "Ver registro de sesiones de un usuario".
2.B	**Modificar estado de un usuario**
2.B.1	El usuario pulsa el botón Acceder de otro usuario y el sistema lo redirige a la página de detalles del usuario donde puede modificar el estado de este último. Se inicia el caso de uso **CU-13**: "Modificar estado de un usuario".
2.C	**Modificar rol de un usuario**
2.C.1	El usuario pulsa el botón Acceder de otro usuario y el sistema lo redirige a la página de detalles del usuario donde puede modificar el rol de este último. Esta acción es exclusiva para usuarios con rol Superuser. Se inicia el caso de uso **CU-14**: "Modificar rol de un usuario".

Tabla 5.11

Caso de uso: Gestionar usuarios

CU-12	**Visualizar registro de sesiones de un usuario**
Actor	Admin y Superuser
Precondiciones	El usuario ha sido autenticado correctamente. Deben existir usuarios en la base de datos.
Postcondiciones	El usuario accede al registro de sesiones del usuario seleccionado.
Escenario principal	
1	El usuario accede a la ventana Detalles del Usuario.
2	El sistema despliega un datagrid que contiene el registro completo de las sesiones iniciadas por el usuario seleccionado en la plataforma.

Tabla 5.12

Caso de uso: Visualizar registro de sesiones de un usuario

CU-13	**Modificar estado de un usuario**
Actor	Admin y Superuser
Precondiciones	El usuario ha sido autenticado correctamente. Deben existir usuarios en la base de datos.
Postcondiciones	Se le cambia el estado al usuario gestionado.
Escenario principal	
1	El usuario accede a la ventana Detalles del Usuario.
2	El usuario pulsa el selector del estado del usuario gestionado y selecciona un estado entre Activo y Bloqueado.
3	El sistema habilita el botón Aplicar Cambios.
4	El usuario pulsa el botón Aplicar Cambios.
5	El sistema actualiza el estado del usuario gestionado y muestra un mensaje de éxito.
Escenarios alternativos	
2.A	**Cancelar**
2.A.1	El usuario pulsa fuera del selector o, si ha seleccionado un estado distinto al original, restablece el valor original.
2.A.2	El sistema oculta las opciones del selector.
5.A	**Error al actualizar el estado**
5.A.1	Se produce un error al actualizar el estado en la base de datos.
5.A.2	El sistema muestra un mensaje de error.

Tabla 5.13

Caso de uso: Ver registro de sesiones de un usuario

CU-14	**Modificar rol de un usuario**
Actor	Superuser
Precondiciones	El usuario ha sido autenticado correctamente. Deben existir usuarios en la base de datos.
Postcondiciones	Se le cambia el rol al usuario gestionado.
Escenario principal	
1	El usuario accede a la ventana Detalles del Usuario.
2	El usuario pulsa el selector del rol del usuario gestionado y selecciona un rol entre User, Admin y Superuser.
3	El sistema habilita el botón Aplicar Cambios.
4	El usuario pulsa el botón Aplicar Cambios.
5	El sistema actualiza el rol del usuario gestionado y muestra un mensaje de éxito.
Escenarios alternativos	
2.A	**Cancelar**
2.A.1	El usuario pulsa fuera del selector o, si ha seleccionado un rol distinto al original, restablece el valor original.
2.A.2	El sistema oculta las opciones del selector.
5.A	**Error al actualizar el rol**
5.A.1	Se produce un error al actualizar el rol en la base de datos.
5.A.2	El sistema muestra un mensaje de error.

Tabla 5.14

Caso de uso: Modificar rol de un usuario

CU-15	**Gestionar modelos de predicción**
Actor	Admin y Superuser
Precondiciones	El usuario ha sido autenticado correctamente.
Postcondiciones	Ninguna.
Escenario principal	
1	El usuario accede a la ventana Modelos.
2	El sistema muestra un datagrid con la información de todos los modelos subidos a la plataforma.
Extensiones	
2.A	**Subir modelo de predicción**
2.A.1	El usuario pulsa el botón Subir Modelo y el sistema muestra un pop-up para poder subir un modelo de predicción. Se inicia el caso de uso **CU-16**: "Subir modelo de predicción".

Tabla 5.15

Caso de uso: Gestionar modelos de predicción

CU-16	**Subir modelo de predicción**
Actor	Admin y Superuser
Precondiciones	El usuario ha sido autenticado correctamente.
Postcondiciones	El modelo se almacena en el sistema.
Escenario principal	
1	El usuario accede a la ventana Modelos.
2	El usuario pulsa el botón Subir Modelo y el sistema muestra un formulario con los campos necesarios para completar el caso de uso.
3	El usuario selecciona el tipo de modelo (Random Forest, TabNet, XGBoost, etc.) y el sistema muestra los formatos permitidos.
4	El usuario carga el fichero con el modelo.
5	El usuario pulsa el botón Confirmar.
6	El sistema almacena el modelo en el sistema y muestra un mensaje de éxito.
Escenarios alternativos	
2.A	**Cancelar**
2.A.1	El usuario pulsa el botón Cancelar o pulsa fuera del formulario.
2.A.2	El sistema oculta el formulario para añadir proyectos.
3.A	**Formato de fichero erróneo**
3.A.1	El usuario carga un fichero con un modelo cuyo formato no corresponde con los permitidos.
3.A.2	El sistema muestra un mensaje de error.
6.A	**Error en el almacenamiento del modelo**
6.A.1	El sistema intenta almacenar el modelo.
6.A.2	Ocurre un error durante el proceso de almacenamiento.
6.A.3	El sistema muestra un mensaje indicando que no se ha podido subir el modelo de predicción.

Tabla 5.16

Caso de uso: Subir modelo de predicción

5.4. Diagramas de actividad

5.4.1. Registrar usuario

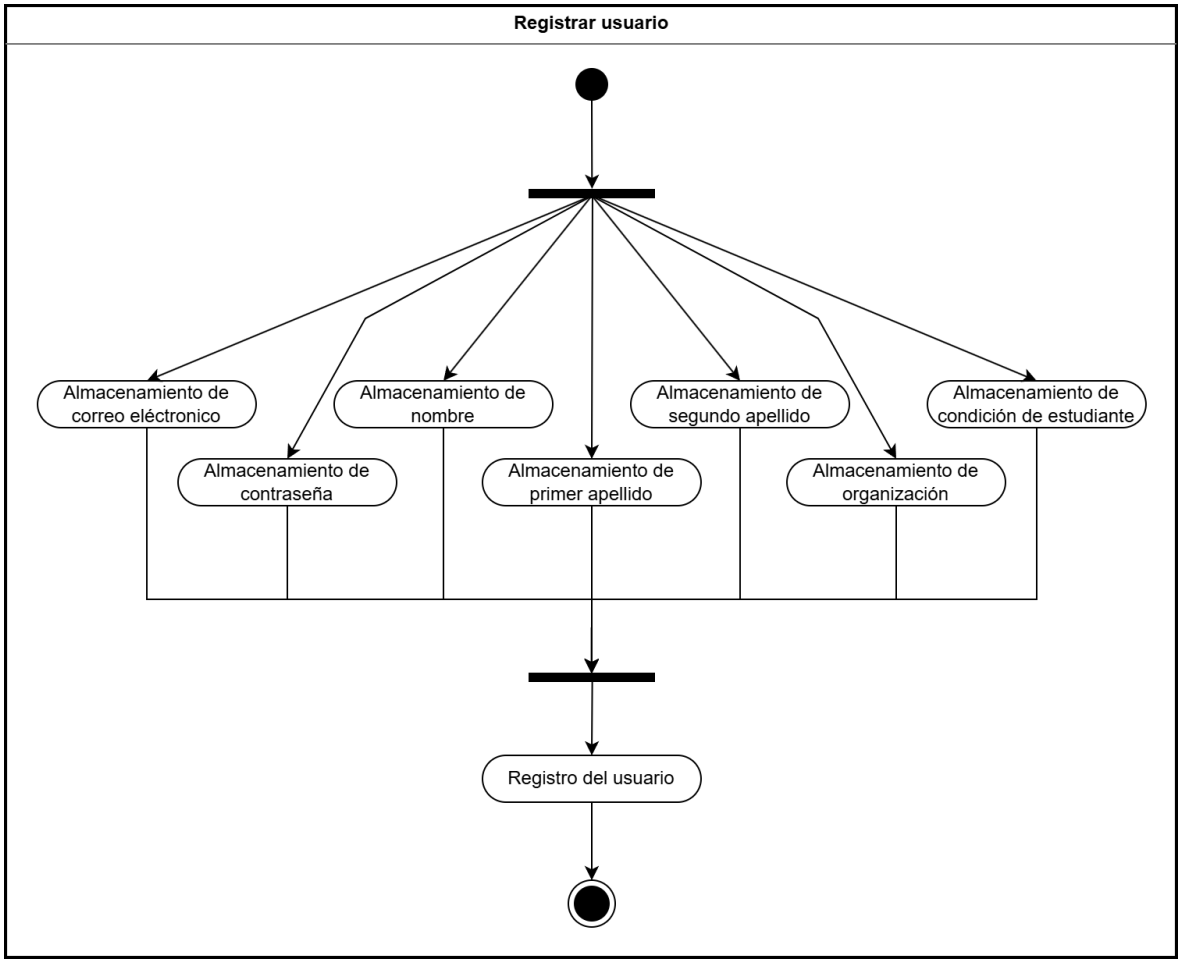


Figura 5.5: Diagrama de actividad: Registrar usuario

5.4.2. Iniciar sesión

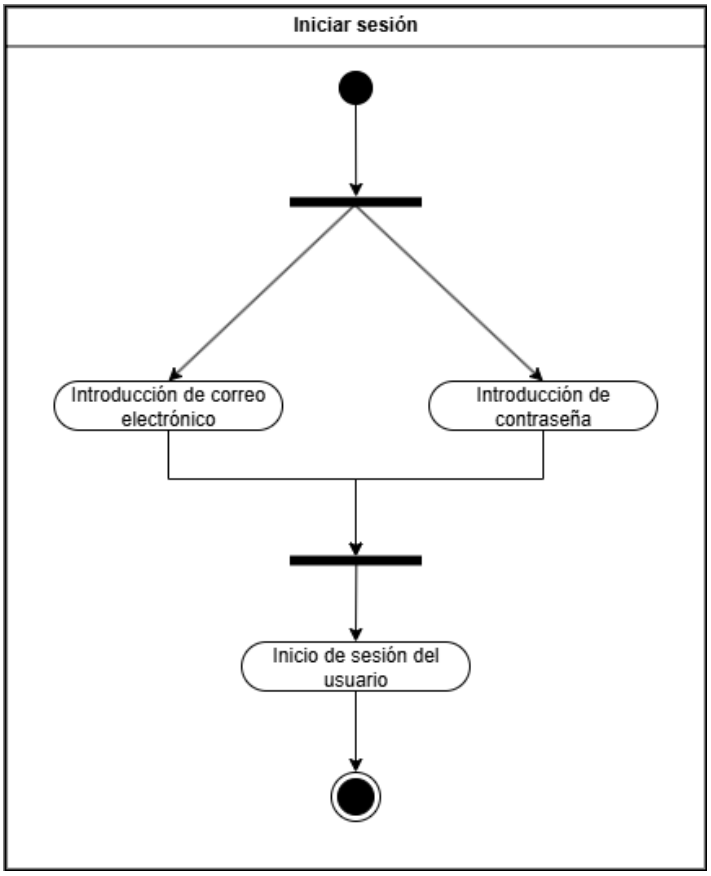


Figura 5.6: Diagrama de actividad: Iniciar sesión

5.4.3. Seleccionar paleta de colores

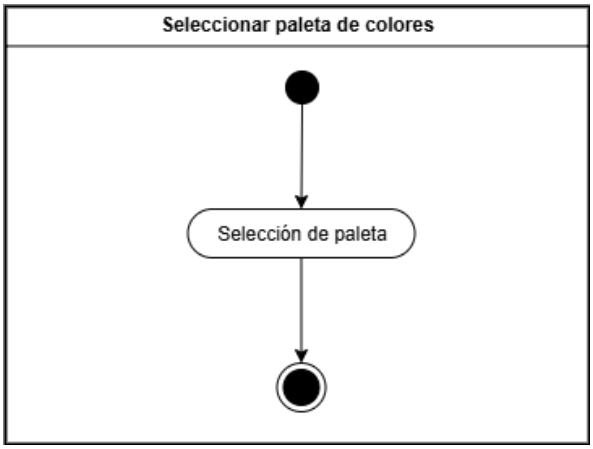


Figura 5.7: Diagrama de actividad: Seleccionar paleta de colores

5.4.4. Añadir proyecto

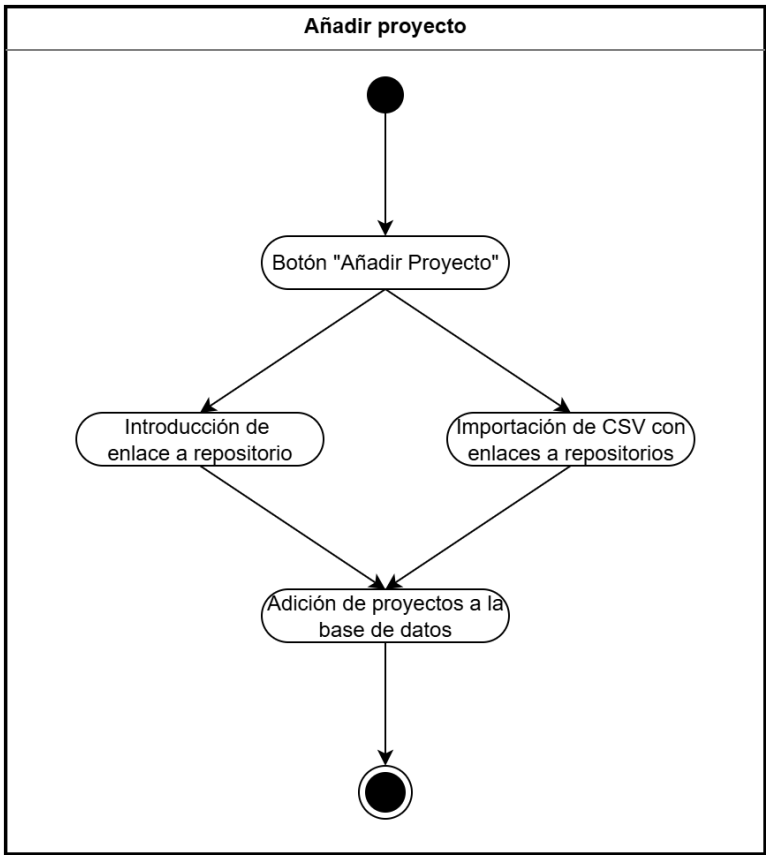


Figura 5.8: Diagrama de actividad: Añadir proyecto

5.4.5. Analizar proyecto

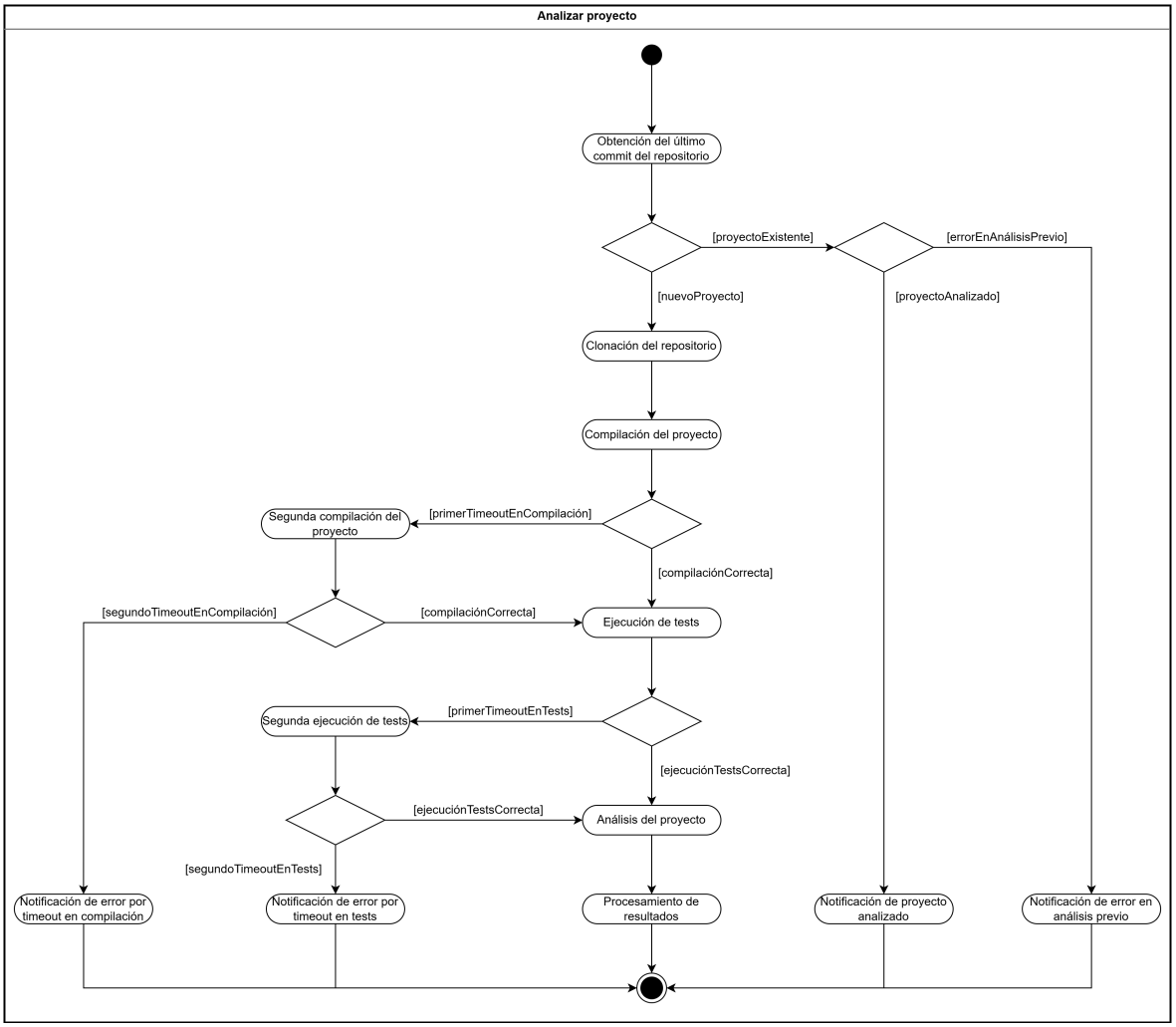


Figura 5.9: Diagrama de actividad: Analizar proyecto

5.4.6. Eliminar proyecto

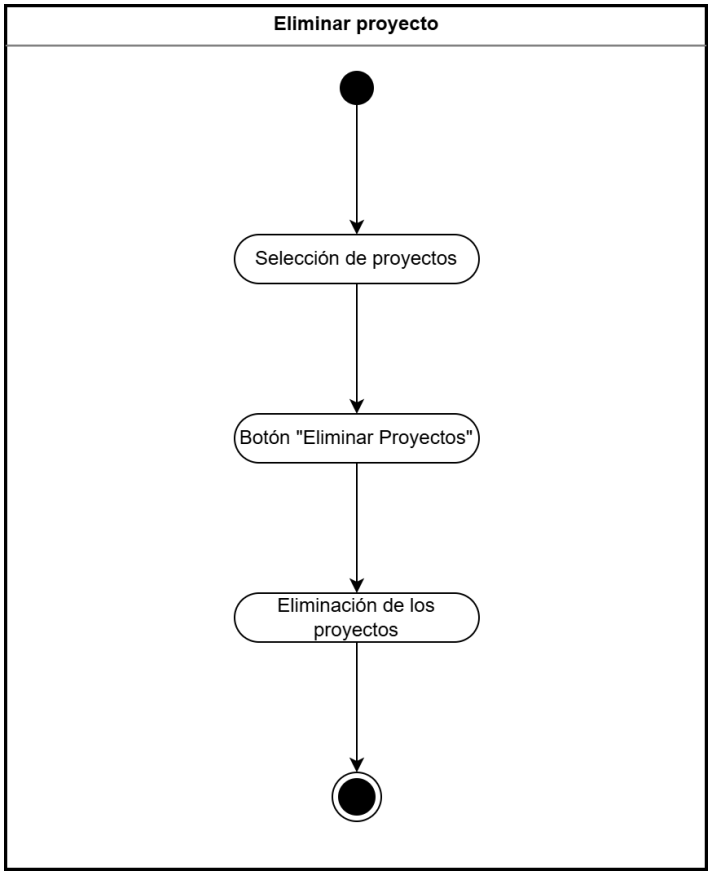


Figura 5.10: Diagrama de actividad: Eliminar proyecto

5.4.7. Visualizar métricas del análisis

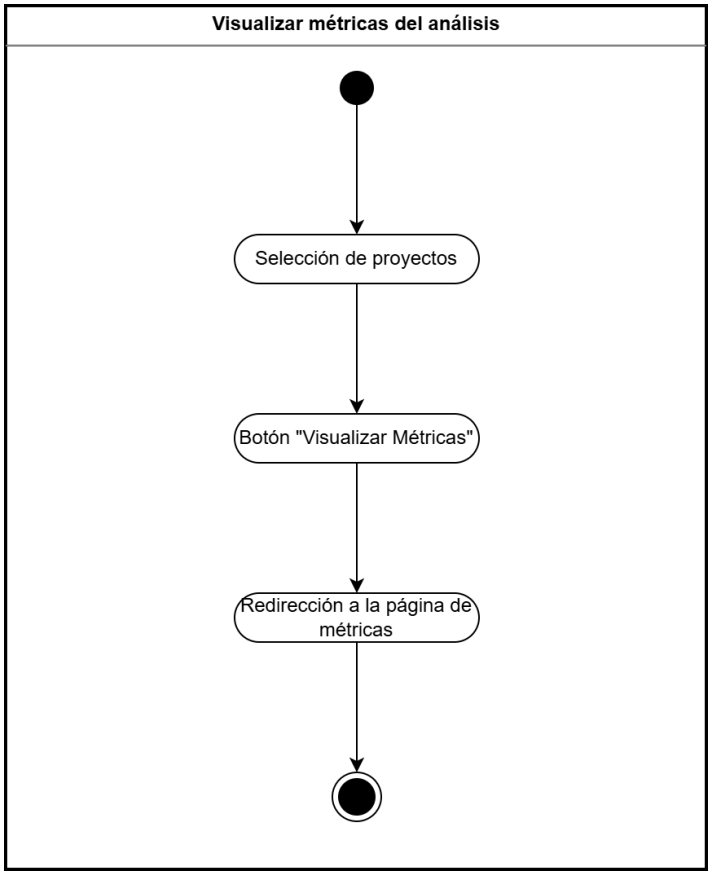


Figura 5.11: Diagrama de actividad: Visualizar métricas del análisis

5.4.8. Exportar métricas

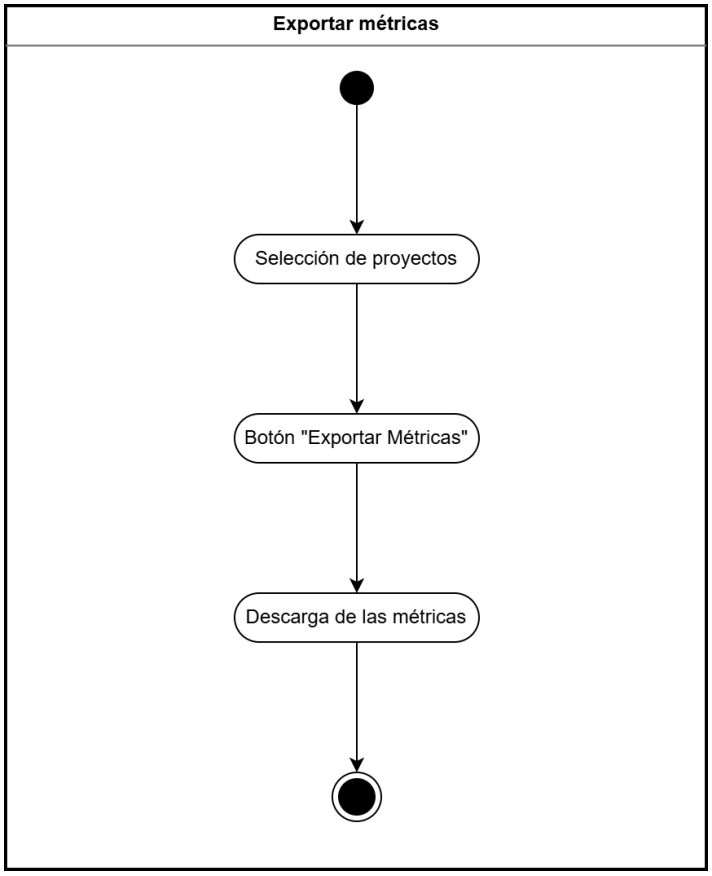


Figura 5.12: Diagrama de actividad: Exportar métricas

5.4.9. Evaluar modelo de predicción

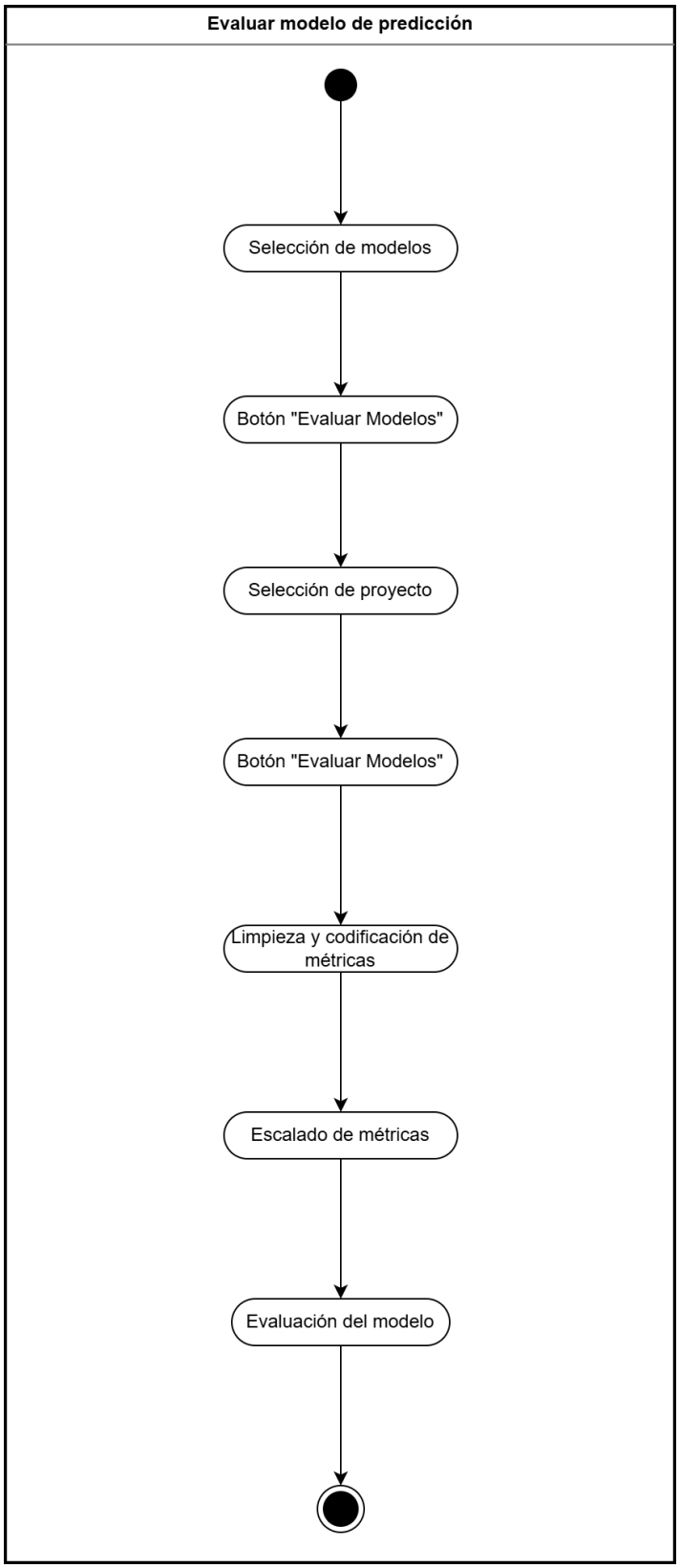


Figura 5.13: Diagrama de actividad: Evaluar modelo de predicción

5.4.10. Visualizar registro de sesiones de un usuario

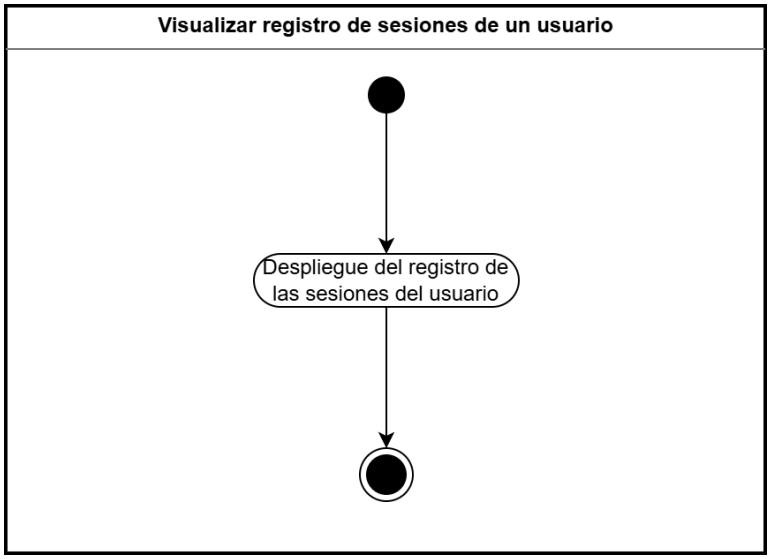


Figura 5.14: Diagrama de actividad: Visualizar registro de sesiones de un usuario

5.4.11. Modificar estado de un usuario

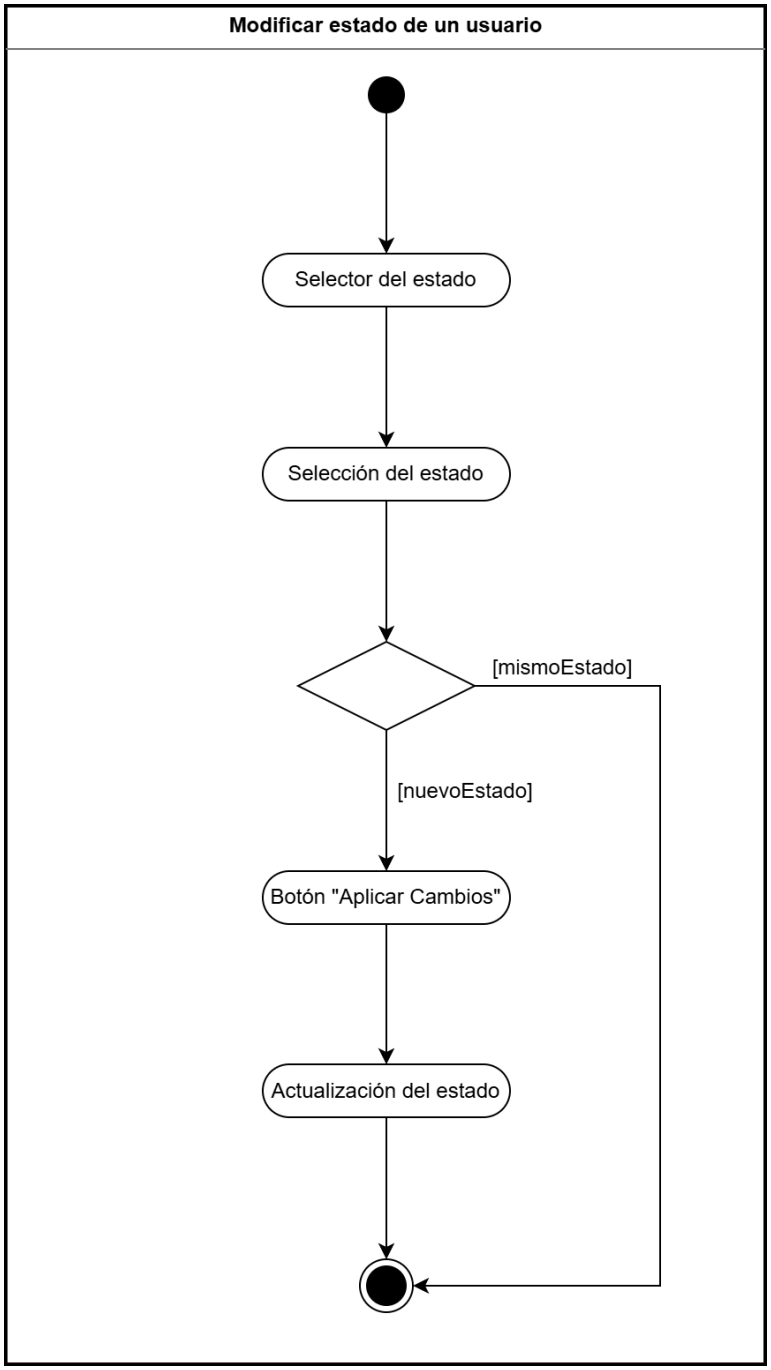


Figura 5.15: Diagrama de actividad: Modificar estado de un usuario

5.4.12. Modificar rol de un usuario

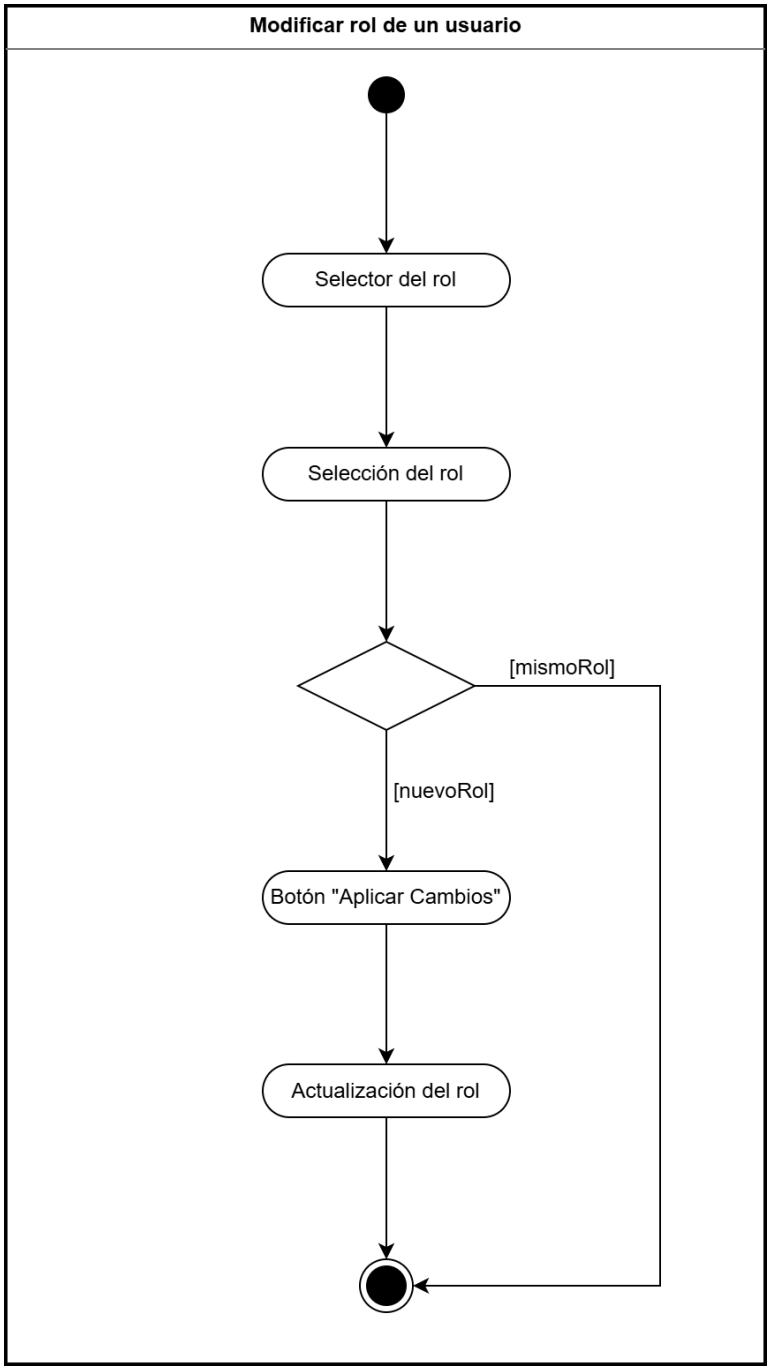


Figura 5.16: Diagrama de actividad: Modificar rol de un usuario

5.4.13. Subir modelo de predicción

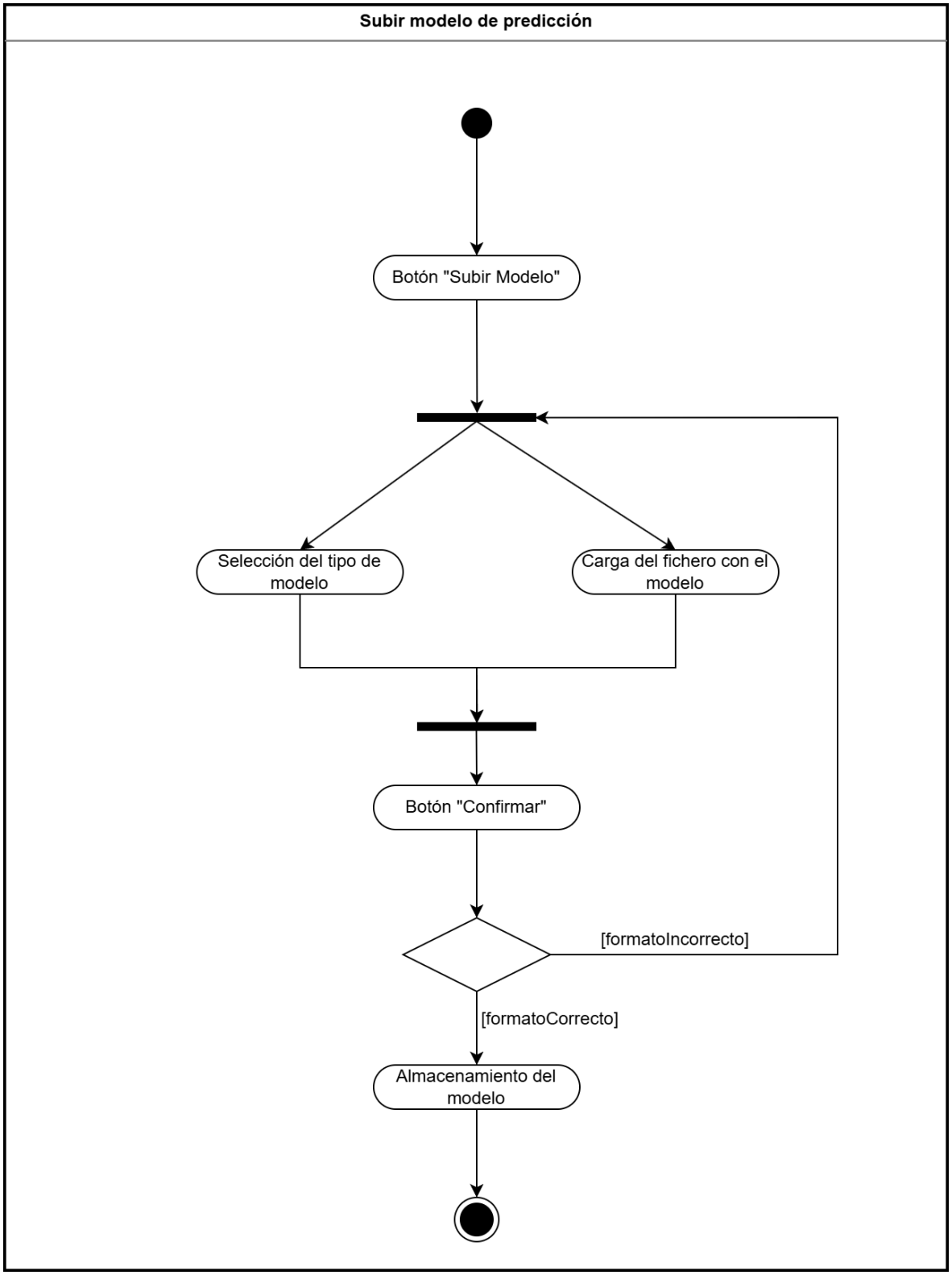


Figura 5.17: Diagrama de actividad: Subir modelo de predicción

6. Diseño del sistema

6.1. Arquitectura del sistema

En esta sección se expone la arquitectura general del sistema, definiendo la infraestructura tecnológica necesaria y la organización de sus componentes. Para su representación se emplean diagramas C4, un diagrama de arquitectura general y un diagrama de flujo, que muestran de forma clara la estructura y el funcionamiento del sistema.

6.1.1. Diseño de alto nivel

En esta sección se presenta el diseño de alto nivel del sistema, describiendo su arquitectura mediante distintos diagramas que permiten visualizar tanto la estructura general como la interacción entre sus componentes. Para ello se utiliza el modelo C4, que contempla diagramas de contexto, contenedores y componentes, ofreciendo una visión jerárquica y detallada del sistema. Además, se incluyen un diagrama de arquitectura general y un diagrama de flujo, que facilitan la comprensión de los procesos y la organización interna del software.

6.1.1.1. Diagrama de contexto

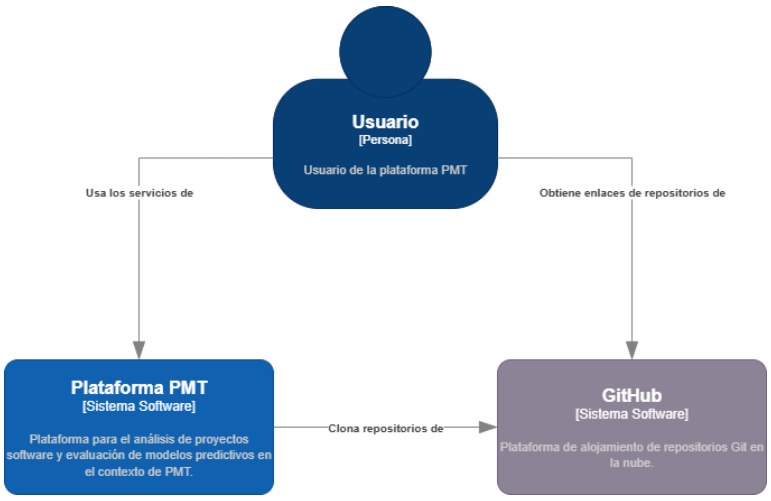


Figura 6.1: Diagrama de contexto

El diagrama de contexto representa la plataforma PMTlab como un sistema único y delimitado, mostrando su relación con los elementos externos que interactúan con ella. En particular, se refleja la integración con los repositorios GitHub, desde los que se obtienen los proyectos de software que serán analizados. Este diagrama permite comprender de forma clara el alcance del sistema desarrollado, sus dependencias externas

y los principales intercambios de información que se producen durante el uso de la plataforma.

Sistemas

- Plataforma PMT: Sistema desarrollado en este trabajo que permite gestionar proyectos software, almacenar los resultados de su análisis, administrar modelos predictivos previamente entrenados y ejecutar evaluaciones PMT sobre proyectos analizados, mostrando las métricas obtenidas de forma centralizada.
- GitHub: Sistema que combina control de versiones con Git y colaboración en la nube para desarrollar software en equipo.

Relaciones

- El usuario interactúa con la Plataforma PMT para subir proyectos, consultar el estado de sus análisis y visualizar los resultados derivados de la evaluación PMT.
- La Plataforma PMT obtiene los proyectos que se van a analizar clonándolos desde sus respectivos repositorios remotos.

6.1.1.2. Diagrama de contenedores

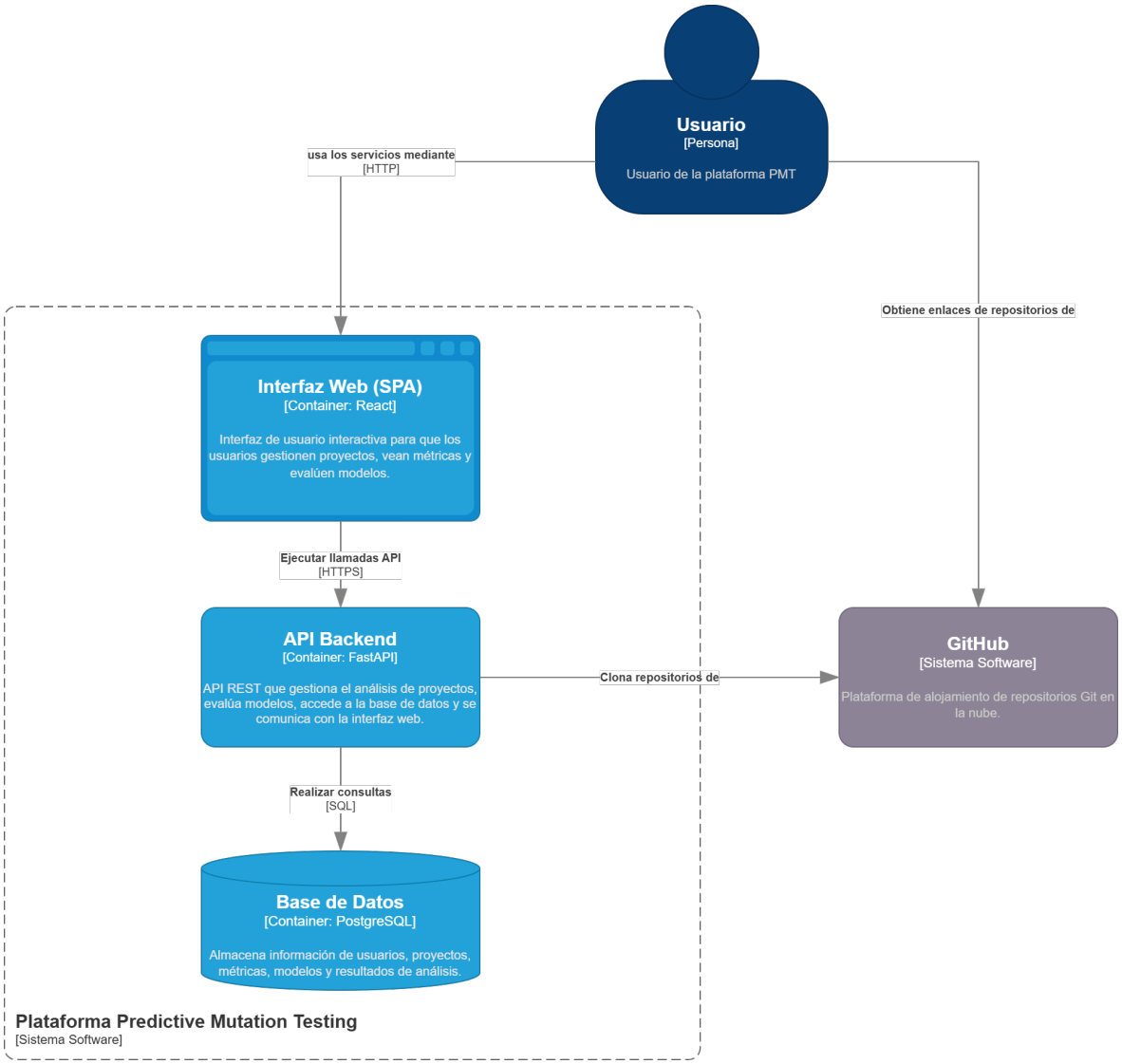


Figura 6.2: Diagrama de contenedores

El diagrama de contenedores descompone el sistema en las aplicaciones, servicios y almacenes de datos que lo conforman. Permite identificar qué tecnologías y responsabilidades corresponden a cada contenedor, así como las relaciones y comunicaciones entre ellos. De esta forma, se proporciona una visión clara de la organización interna del sistema y de cómo los diferentes elementos colaboran para ofrecer la funcionalidad global.

Contenedores

- Interfaz Web (SPA): Contenedor que compone la interfaz de la Plataforma PMT donde los usuarios hacen uso del sistema.
- API Backend: Componente que gestiona las solicitudes del frontend y e implementa toda la lógica y funcionalidades del sistema. Desarrollada con FastAPI.

- Base de datos: Contenedor que almacena y gestiona los datos del sistema en una base de datos relacional PostgreSQL.

Relaciones

- Los usuarios usan los servicios de la Plataforma PMT mediante la Interfaz Web vía HTTP.
- La Interfaz Web ejecuta llamadas a la API Backend, llevando a cabo las peticiones de los usuarios.
- La API Backend obtiene la información necesaria mediante consultas a la Base de Datos.

6.1.1.3. Diagrama de componentes

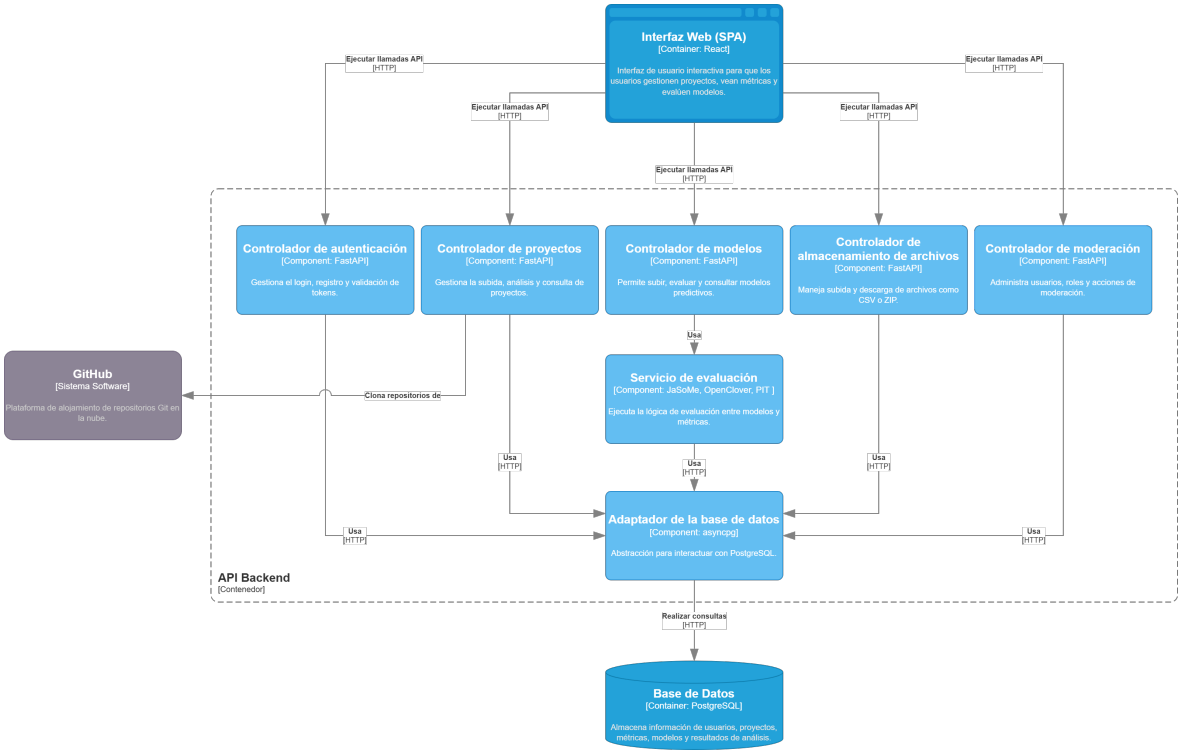


Figura 6.3: Diagrama de componentes

El diagrama de componentes profundiza en la estructura interna de cada contenedor, mostrando los módulos, clases o subsistemas que lo conforman y las relaciones entre ellos. Su propósito es detallar la organización lógica del software, facilitando la comprensión de cómo se implementan las responsabilidades de cada contenedor y cómo se integran los distintos elementos en el sistema.

Componentes

- Controlador de autenticación: Componente que gestiona el registro de usuarios, inicio de sesión y validación de tokens.

- Controlador de proyectos: Componente que gestiona la subida, análisis y validación de tokens.
- Controlador de modelos: Componente que permite la subida, evaluación y consulta de modelos predictivos.
- Servicio de evaluación: Componente que ejecuta la lógica de evaluación entre modelos y métricas.
- Controlador de almacenamiento de archivos: Componente responsable de gestionar la subida, almacenamiento y acceso a los archivos asociados al sistema, incluyendo los ficheros de modelos predictivos subidos por los usuarios y los ficheros de resultados generados durante el análisis de proyectos y la evaluación de modelos.
- Controlador de moderación: Componente que administra los usuarios, roles y acciones de moderación.
- Adaptador de la base de datos: Componente abstracto que interactúa con PostgreSQL.

Relaciones

- Los controladores se comunican con el Adaptador de la base de datos como puente para la Base de Datos.

6.1.1.4. Diagrama de arquitectura general

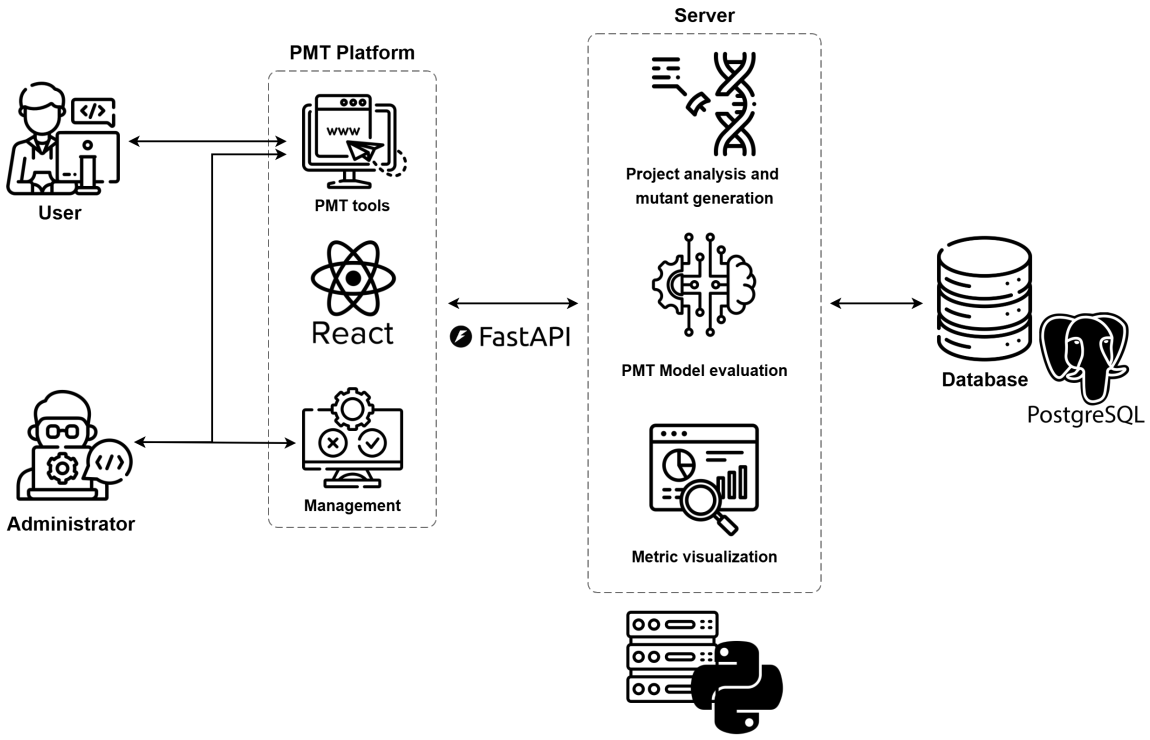


Figura 6.4: Diagrama de arquitectura general

El diagrama de arquitectura general muestra cómo se organizan y relacionan los principales componentes de la plataforma PMT dentro del entorno de despliegue. En él se distinguen claramente el frontend web, encargado de la interacción con el usuario, el backend que implementa la lógica de negocio y coordina los procesos de análisis y evaluación, y la base de datos que almacena la información de usuarios, proyectos, métricas y modelos predictivos.

Asimismo, el diagrama refleja el flujo principal de datos del sistema: desde la solicitud de análisis o evaluación iniciada por el usuario, pasando por la ejecución de los procesos correspondientes en el backend, hasta el almacenamiento y posterior consulta de los resultados. Esta representación permite comprender de forma global cómo se integran los distintos módulos desarrollados y cómo se soporta el funcionamiento completo de la plataforma.

6.1.1.5. Diagrama de flujo

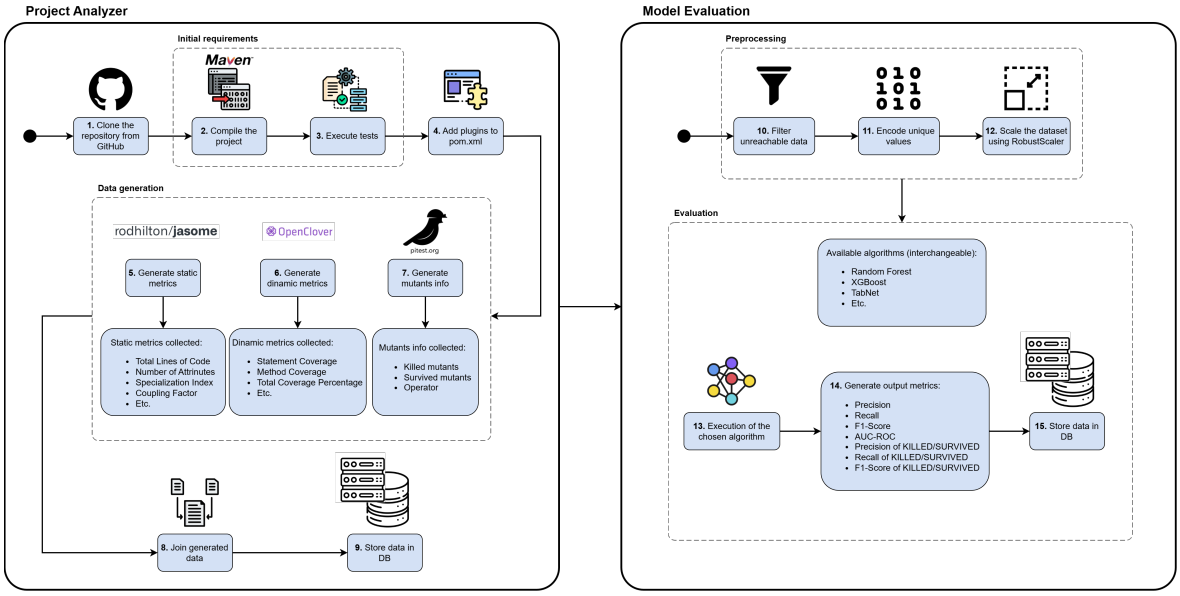


Figura 6.5: Diagrama de flujo

El proceso completo de PMTlab se divide en dos grandes fases: el análisis de proyectos y la evaluación de modelos predictivos. A continuación se describen cada uno de los nodos que conforman este flujo de trabajo mostrado en la figura 6.5.

Fase de Análisis

1. **Clonar el repositorio desde GitHub**. El sistema descarga automáticamente el código fuente del proyecto desde su repositorio remoto en GitHub, utilizando la URL proporcionada por el usuario. Esto garantiza que el análisis se realiza sobre una versión concreta del proyecto, identificada por su *commit*.

2. **Compilar el proyecto**. Una vez clonado, el proyecto se compila para verificar que el código es válido y que puede ejecutarse correctamente. Si la compilación falla, el proceso de análisis se detiene.

3. **Ejecutar los tests**. Se ejecutan los casos de prueba del proyecto original sin mutaciones para comprobar que el proyecto es estable antes de introducir cambios en el código.

4. **Añadir plugins al *pom.xml***. Se incorporan dinámicamente los plugins necesarios (PIT, OpenClover y JaSoMe) al fichero de configuración Maven del proyecto, lo que permite generar mutantes y extraer métricas sin modificar permanentemente el repositorio original.

5. **Generar métricas estáticas**. Se calculan métricas estáticas del código mediante herramientas como JaSoMe.

6. **Generar métricas dinámicas**. A través de OpenClover se obtienen métricas durante la ejecución de los tests, reflejando el comportamiento real del código.

7. **Generar información de mutantes**. Utilizando PIT se generan mutantes del código y se ejecutan los tests sobre ellos, obteniendo información como el operador de mutación y si el mutante ha sido *KILLED* o *SURVIVED*.
8. **Unir los datos generados**. Todas las métricas y la información de los mutantes se combinan en un único conjunto de datos estructurado, donde cada fila representa un mutante y cada columna una característica.
9. **Almacenar los datos en la base de datos**. El conjunto de datos resultante se guarda en la base de datos de la plataforma, asociado al proyecto y al *commit* analizado, permitiendo reutilizar la información en evaluaciones posteriores.

Fase de Evaluación

10. **Filtrar datos no alcanzables**. Se eliminan los mutantes no alcanzables o con valores inválidos para evitar que afecten negativamente al rendimiento de los modelos.
11. **Codificar valores únicos**. Las variables categóricas se transforman en valores numéricos mediante codificadores previamente entrenados, permitiendo su uso por los modelos predictivos.
12. **Escalar el conjunto de datos con RobustScaler**. Las métricas numéricas se escalan para reducir el impacto de valores extremos y mejorar la estabilidad de los modelos.
13. **Ejecutar el algoritmo seleccionado**. Se aplica el modelo predictivo elegido (por ejemplo, Random Forest, XGBoost o TabNet) sobre el conjunto de datos procesado para obtener las predicciones de cada mutante.
14. **Generar métricas de salida**. Se calculan las métricas de evaluación del modelo, incluyendo:
    - Precision
    - Recall
    - F1-Score
    - AUC-ROC
    - Precision para las clases *KILLED* y *SURVIVED*
    - Recall para las clases *KILLED* y *SURVIVED*
    - F1-Score para las clases *KILLED* y *SURVIVED*
15. **Almacenar los resultados en la base de datos**. Las métricas obtenidas y las predicciones generadas se almacenan en la base de datos, permitiendo su posterior consulta y comparación entre modelos.

Cabe destacar que el listado completo de métricas empleadas en el proceso de análisis y evaluación, junto con su descripción detallada, se encuentra recogido en el anexo C. En dicho anexo se especifican tanto las métricas estáticas y dinámicas

extraídas de los proyectos como aquellas utilizadas por los modelos predictivos para la evaluación de los mutantes.

6.2. Diseño físico de datos

En esta sección se presenta la traducción del modelo conceptual al esquema físico de la base de datos, como se muestra en la figura 6.6. Se definen las tablas, claves, relaciones e índices necesarios para garantizar la integridad y el acceso eficiente a la información en el SGBD seleccionado.



Figura 6.6: Esquema físico de la base de datos

6.2.1. Tabla: iplogs

Esta tabla consta del registro de entrada y salida de la plataforma de cada usuario.

- ***log_id***: Clave primaria.
- ***user_id***: Clave foránea que apunta a *id* de la tabla *users*.
- ***ip***: IP del usuario.
- ***entry_time***: Fecha y hora de inicio de sesión del usuario.
- ***entry_time***: Fecha y hora de cierre de sesión del usuario.

6.2.2. Tabla: users

Esta tabla gestiona la información de los usuarios registrados.

- ***id***: Clave primaria.
- ***email***: Correo electrónico del usuario.
- ***password***: Contraseña de la cuenta.
- ***role***: Rol de la cuenta del usuario.
- ***name***: Nombre del usuario.
- ***lastname1***: Primer apellido del usuario.
- ***lastname2***: Segundo apellido del usuario.
- ***organization***: Organización a la que pertenece el usuario.
- ***isstudent***: Booleano que indica si el usuario es estudiante o no.
- ***created_at***: Fecha de la creación de la cuenta.
- ***account_status***: Estado de la cuenta (Activo o Bloqueado).

6.2.3. Tabla: models

Esta tabla almacena los datos de los modelos de predicción subidos a la plataforma.

- ***id***: Clave primaria.
- ***name***: Nombre del modelo.
- ***path***: Ruta en la que se encuentra el fichero del modelo en el servidor.
- ***version***: Versión del modelo.
- ***model_type***: Tipo de modelo predictivo subido a la plataforma (por ejemplo, `RandomForest`, `XGBoost`, `TabNet`, etc.).

- ***format***: Formato del fichero del modelo.
- ***user_id***: Clave foránea que apunta a *id* de la tabla *users*.
- ***uploaded_at***: Fecha en la que se subió el modelo al servidor.

6.2.4. Tabla: project_uploads

Esta tabla relaciona a los usuarios con los proyectos que han subido a la plataforma.

- ***code_repository***: Clave primaria y foránea que apunta a *code_repository* de la tabla *projects*.
- ***commit***: Clave primaria y foránea que apunta a *commit* de la tabla *projects*.
- ***id_user***: Clave primaria y foránea que apunta a *id* de la tabla *users*.
- ***upload_date***: Fecha de subida del proyecto.

6.2.5. Tabla: project

Esta tabla almacena y gestiona la información de los proyectos subidos a la plataforma.

- ***code_repository***: Clave primaria. Código del repositorio del proyecto.
- ***name***: Nombre del proyecto.
- ***commit***: Clave primaria. Commit del repositorio del proyecto.
- ***id_project***: ID del proyecto en la tabla.
- ***status***: Estado actual del proyecto, definido por el tipo *enum_status_project*. Indica la fase del flujo en la que se encuentra el proyecto, pudiendo tomar los valores:
  - *INITIAL*: El proyecto ha sido registrado en la plataforma pero aún no se ha clonado.
  - *CLONED*: El repositorio del proyecto ha sido clonado correctamente.
  - *ANALYZED*: El proyecto ha sido analizado y se han generado métricas, tal como se muestra en el anexo C.
  - *UPLOADED*: Los resultados del análisis han sido subidos exitosamente a la plataforma.
  - *FAILED*: El proceso de análisis o subida ha fallado en alguna etapa.
- ***initial_execution***: Etapa inicial del proyecto dentro de la plataforma, definida por el tipo *enum_status_analysis_project*. Indica en qué paso se encuentra el proyecto al momento de su primera ejecución:

- *CORRECT*: Inicialización correcta del proyecto.
- *TEST_FAIL*: Error detectado en las pruebas del proyecto.
- *COMPILE_FAIL*: Error en la compilación inicial.
- *COMPILE_TIMEOUT*: La compilación inicial superó el tiempo límite.
- *TEST_TIMEOUT*: Las pruebas iniciales excedieron el tiempo límite.

- ***logs***: Registro textual o referencia al archivo de logs generado durante la ejecución del análisis del proyecto. Contiene información de seguimiento, errores o advertencias.

6.2.6. Tabla: project_results

Esta tabla almacena los resultados generados por los distintos modelos de predicción aplicados a cada proyecto.

- ***id***: Clave primaria.
- ***model_id***: Clave foránea que apunta a *id* de la tabla *models*.
- ***project_id***: Clave foránea que apunta a *id_project* de la tabla *project*.
- ***path***: Ruta del archivo CSV donde se almacena el resultado generado por el modelo.
- ***version***: Versión del modelo utilizado. Permite diferenciar resultados generados por distintas versiones de un mismo modelo.
- ***uploaded_at***: Fecha y hora en que se registró el resultado en la plataforma.

6.2.7. Tabla: analysis_tool

Esta tabla almacena la información de las herramientas de análisis utilizadas.

- ***name***: Clave primaria. Nombre de la herramienta de análisis.
- ***version***: Versión de la herramienta de análisis..
- ***id_analysis_tool***: Identificador interno de la herramienta dentro de la base de datos.

6.2.8. Tabla: coverage_tool

Esta tabla almacena la información de las herramientas de cobertura utilizadas.

- ***name***: Clave primaria. Nombre de la herramienta de cobertura.
- ***version***: Versión de la herramienta de cobertura.

- ***id_coverage_tool***: Identificador interno de la herramienta dentro de la base de datos.

6.2.9. Tabla: mutation_tool

Esta tabla almacena la información de las herramientas de mutación utilizadas.

- ***name***: Clave primaria. Nombre de la herramienta de mutación.
- ***version***: Versión de la herramienta de mutación.
- ***id_mutation_tool***: Identificador interno de la herramienta dentro de la base de datos.

6.2.10. Tabla: mutant

Esta tabla almacena la información de los mutantes generados por las herramientas de mutación. Cada registro representa un mutante concreto introducido en un archivo del proyecto durante el proceso de mutation testing.

- ***operator***: Clave primaria. Nombre del operador de mutación aplicado. Define el tipo de modificación realizada sobre el código fuente.
- ***line***: Número de línea del archivo donde se aplicó el operador de mutación.
- ***method***: Nombre del método afectado por el mutante.
- ***type_return***: Tipo de retorno del método donde se introdujo la mutación.
- ***id_mutant***: Clave primaria. Identificador único del mutante dentro de la base de datos.
- ***id_mutation_tool***: Clave foránea que apunta a *id_mutation_tool* de la tabla *mutation_tool*.
- ***id_file***: Clave foránea que apunta a *id_file* de la tabla *file*.

6.2.11. Tabla: execution

Esta tabla almacena la información de las ejecuciones de pruebas y cobertura realizadas sobre los mutantes generados en los proyectos. Cada registro representa la ejecución de un conjunto de pruebas sobre un mutante específico utilizando una herramienta de cobertura determinada.

- ***excluded_tests***: Lista de pruebas que fueron excluidas de la ejecución para este mutante.
- ***target_tests***: Pruebas objetivo que fueron ejecutadas durante esta ejecución.
- ***added_tests***: Pruebas adicionales añadidas para asegurar la cobertura mínima o evaluar mutantes específicos.

- ***num_executed_covered***: Número de pruebas ejecutadas que efectivamente cubrieron alguna línea de código modificada por el mutante.
- ***num_test_covered***: Número total de pruebas que cubrieron al menos una instrucción del código evaluado.
- ***num_mutant_assertion***: Número de aserciones ejecutadas relacionadas con mutantes durante la ejecución.
- ***num_class_assertion***: Número de aserciones ejecutadas a nivel de clase en esta ejecución.
- ***id_mutant***: Clave foránea que apunta a *id_mutant* de la tabla *mutant*.
- ***id_execution***: Clave primaria. Identificador único de la ejecución dentro de la base de datos.
- ***id_coverage_tool***: Clave foránea que apunta a *id_coverage_tool* de la tabla *coverage_tool*.
- ***status***: Estado de la ejecución, definido por el tipo *enum_status*. Indica el resultado final de la ejecución, pudiendo tomar los valores:
  - *KILLED*: La ejecución detectó correctamente el mutante y la prueba falló como se esperaba.
  - *SURVIVED*: El mutante sobrevivió; las pruebas no lo detectaron.
  - *TIMED_OUT*: La ejecución excedió el tiempo límite permitido.
  - *MEMORY_ERROR*: La ejecución falló debido a un error de memoria.
  - *RUN_ERROR*: Ocurrió un error durante la ejecución del test.
  - *NON_VIABLE*: La ejecución no fue viable para este mutante.
  - *NO_COVERAGE*: Ninguna prueba cubrió el mutante durante la ejecución.

6.2.12. Tabla: file

Esta tabla almacena información sobre los archivos de código fuente incluidos en los proyectos.

- ***name***: Clave primaria. Nombre del archivo de código.
- ***programming_language***: Lenguaje de programación del archivo.
- ***extension***: Extensión del archivo.
- ***package***: Clave primaria. Nombre del paquete al que pertenece el archivo dentro del proyecto.
- ***id_file***: Identificador único del archivo dentro de la base de datos.
- ***id_project***: Clave primaria y foránea que apunta a *id_project* de la tabla *project*.

6.2.13. Tabla: static_analysis

Esta tabla almacena los resultados del análisis estático realizados sobre los mutantes de los proyectos. Cada registro representa un conjunto de métricas extraídas por una herramienta de análisis estático para un mutante específico.

- ***id_mutant***: Clave foránea que apunta a *id_mutant* de la tabla *mutant*.
- ***id_analysis_tool***: Clave foránea que apunta a *id_analysis_tool* de la tabla *analysis_tool*.
- ***id_static_analysis***: Clave primaria. Identificador único de este registro de análisis estático.

  Para ver el contenido completo de las métricas generadas, véase el anexo C.

6.3. Diseño de la interfaz de usuario

6.3.1. Sistema de paletas de colores

La plataforma incorpora un sistema de tematización que permite alternar entre tres paletas de colores: *Claro*, *Oscuro* y *Azul*. Cada una de estas paletas define de forma consistente los colores utilizados en los principales elementos de la interfaz, incluyendo el fondo de la aplicación, los componentes visuales, los textos, los botones, los bordes, las tablas de datos y los enlaces.

El cambio de tema se aplica de manera global a toda la interfaz y se mantiene persistente durante la sesión del usuario, garantizando una experiencia visual uniforme en todas las pantallas del sistema.

Las tres paletas han sido diseñadas con objetivos diferenciados:

- **Tema Claro**: orientado a ofrecer una visualización limpia y luminosa, adecuada para entornos bien iluminados.
- **Tema Oscuro**: diseñado para reducir la fatiga visual en entornos con poca iluminación, utilizando fondos oscuros y colores de alto contraste.
- **Tema Azul**: proporciona una estética intermedia con una identidad visual diferenciada y buena legibilidad.

Cada tema define un conjunto de variables de estilo que controlan los colores de los principales elementos de la interfaz. En la Tabla 6.1 se resumen los colores más relevantes de cada paleta.

Elemento	Tema Claro	Tema Oscuro	Tema Azul
Fondo principal	#f5f5f5	#121212	#e0f7fa
Texto	#333333	#ffffff	#333333
Color primario	#00796b	#107C41	#01579B
Botón hover	#005B50	#087F32	#09487b
Enlaces	#076258	#1DB954	#01579B

Tabla 6.1

Colores principales de cada tema con representación visual

6.4. Diseño de los modelos predictivos

Dentro del diseño del sistema, uno de los elementos fundamentales es la definición y entrenamiento de los modelos de predicción utilizados para el PMT. Estos modelos son los encargados de estimar si un mutante será *KILLED* o *SURVIVED*. a partir de las métricas generadas durante el análisis del proyecto.

La plataforma está diseñada para ser extensible: los modelos incorporados no son estáticos, sino que pueden actualizarse o sustituirse por nuevas versiones más precisas. De hecho, el apartado de gestión de modelos permite cargar nuevas arquitecturas, versiones optimizadas o enfoques diferentes, manteniendo así la plataforma abierta a la evolución y al progreso en líneas de investigación futuras.

6.4.1. Modelos desarrollados

Para este trabajo se han entrenado tres modelos principales, seleccionados por su rendimiento, interpretabilidad o capacidad para manejar grandes volúmenes de métricas:

- **Random Forest**: modelo basado en árboles de decisión, robusto y tradicionalmente eficaz para tareas de clasificación en métricas de software.
- **XGBoost**: algoritmo de *gradient boosting* optimizado, capaz de capturar relaciones no lineales y ofrecer un alto rendimiento con tiempos de inferencia bajos.
- **TabNet**: arquitectura basada en *deep learning* diseñada específicamente para datos tabulares, capaz de realizar aprendizaje basado en atención e identificar patrones complejos entre características.

6.4.2. Datos utilizados para el entrenamiento

El conjunto de datos utilizado para entrenar los modelos proviene de las métricas generadas por módulos externos de análisis de mutación (Véase el anexo C).

El preprocesamiento aplicado consistió en:

- Codificación de valores categóricos mediante *Label Encoding*.

- Escalado de características numéricas mediante *RobustScaler*.
- Eliminación de registros incompletos o no alcanzables.

6.4.3. Configuración y optimización

Los modelos predictivos no fueron entrenados utilizando configuraciones arbitrarias, sino mediante un proceso sistemático de búsqueda de hiperparámetros basado en *Grid Search*. Para cada tipo de modelo se definió un espacio de búsqueda compuesto por los hiperparámetros más relevantes, y se evaluaron múltiples combinaciones sobre un conjunto de validación con el objetivo de maximizar el rendimiento predictivo.

Este procedimiento permitió explorar distintas configuraciones y seleccionar, para cada algoritmo, aquella que ofrecía el mejor equilibrio entre capacidad de discriminación y estabilidad en la predicción de mutantes *KILLED* y *SURVIVED*.

Los modelos finalmente seleccionados y cargados inicialmente en la plataforma corresponden a las siguientes configuraciones óptimas:

- **Random Forest**
  - Número de árboles (*n_estimators*): 100
  - Profundidad máxima (*max_depth*): 7
  - Muestras mínimas por división: 10
  - Muestras mínimas por hoja: 1
  - Bootstrap: Desactivado
- **XGBoost**
  - Número de árboles: 100
  - *Learning rate*: 0.05
  - Profundidad máxima: 7
  - *Subsample*: 0.8
  - *Colsample by tree*: 0.9
  - *Gamma*: 0.1
  - *Min child weight*: 1
- **TabNet**
  - Dimensión de decisión (*n_d*): 8
  - Dimensión de atención (*n_a*): 32
  - Número de pasos: 7
  - Parámetro de regularización (*lambda_sparse*): 0.001
  - *Momentum*: 0.02

- Optimizador: Adam (*learning rate* = 0.02)

Estas configuraciones fueron seleccionadas por ofrecer los mejores resultados globales en las métricas de clasificación durante el proceso de validación.

6.4.4. Métricas de los modelos entrenados

En la Tabla 6.2 se muestran los resultados obtenidos por las versiones finales de cada modelo, que son las que se cargan inicialmente en la plataforma PMTlab.

Modelo	F1-macro	F1-KILLED	F1-SURVIVED	MCC	ROC-AUC
Random Forest	0.594	0.840	0.347	0.190	0.646
XGBoost	0.612	0.855	0.370	0.225	0.663
TabNet	0.617	0.834	0.400	0.247	0.682

Tabla 6.2

Métricas de validación de los modelos predictivos seleccionados tras Grid Search.

TabNet presenta el mejor rendimiento global en términos de F1-macro y MCC, lo que indica una mayor estabilidad y equilibrio entre ambas clases. XGBoost obtiene el mayor valor de ROC-AUC, reflejando una excelente capacidad discriminativa. Random Forest, aunque más conservador, ofrece una línea base sólida y fácilmente interpretable.

Estos resultados justifican la incorporación de los tres modelos en la plataforma, permitiendo comparar distintos enfoques y facilitando la actualización futura de versiones más avanzadas.

6.4.5. Actualización y gestión de versiones

La plataforma permite subir nuevos modelos mediante el módulo de *Gestión de Modelos*. Cada modelo se acompaña de:

- Su tipo (RandomForest, XGBoost, TabNet, etc.).
- Su versión.
- El archivo correspondiente en el formato requerido.
- La fecha de subida.

Esto garantiza un sistema flexible, en el que es posible:

- Sustituir modelos por versiones mejoradas.
- Añadir nuevas arquitecturas.
- Comparar diferentes enfoques predictores.
- Evaluar modelos en distintos proyectos directamente desde la interfaz.

De esta manera, el sistema queda preparado para evolucionar y adaptarse a nuevos avances en investigación sin necesidad de modificaciones internas en el código.

7. Construcción del sistema

Este capítulo trata los aspectos relacionados con la implementación del sistema en código, haciendo uso de un determinado entorno tecnológico.

7.1. Entorno de construcción

El entorno de desarrollo del sistema se ha configurado a partir de un conjunto de tecnologías y herramientas que permiten cubrir todas las fases del desarrollo.

El desarrollo se ha llevado a cabo en Visual Studio Code (VS Code) como entorno principal de programación. Para el control de versiones se ha utilizado Git, con GitHub como repositorio remoto de código y plataforma de colaboración.

Para la implementación del frontend se ha empleado React, junto con Vite como herramienta de construcción y empaquetado, lo que ha permitido una mayor rapidez en el desarrollo y pruebas locales.

El backend se ha desarrollado en Python utilizando el framework FastAPI, por su ligereza y eficiencia en el manejo de APIs REST. La gestión de la base de datos se ha realizado con PostgreSQL, complementada con el uso de Docker para garantizar entornos reproducibles tanto en desarrollo como en despliegue.

Este conjunto de tecnologías garantiza un flujo de trabajo organizado, eficiente y replicable, lo que contribuye tanto a la calidad del desarrollo como a la mantenibilidad futura del sistema.

7.2. Código fuente

En esta sección se describe la organización del código fuente del sistema, detallando la estructura de directorios y la función de los principales ficheros que lo componen. Además, se incluyen extractos representativos de código que ilustran la implementación de algoritmos o funcionalidades clave de la plataforma.

- **/web**: Carpeta raíz del proyecto web. Contiene todos los elementos necesarios para la configuración, inicialización y despliegue de la plataforma. Entre sus componentes se incluyen los archivos de configuración que permiten levantar la base de datos, creando las tablas necesarias e insertando los datos iniciales para que la plataforma funcione correctamente desde el primer momento. Esto incluye archivos de Docker Compose, que facilitan la creación de contenedores para los distintos servicios del sistema, asegurando que la base de datos, la API backend y cualquier otro servicio dependiente puedan iniciarse de manera automatizada y consistente en cualquier entorno.

- **/backend**: Contiene todo el código del servidor y lógica de negocio. Agrupa todos los componentes necesarios para el funcionamiento del servidor de la plataforma. Aquí se encuentran los scripts que gestionan el análisis, la evaluación y el preprocesamiento de los proyectos subidos por los usuarios, así como las funcionalidades relacionadas con la API que comunica el frontend con la base de datos. Además, incluye utilidades generales que facilitan tareas recurrentes del sistema, el controlador específico para PostgreSQL, que permite interactuar de forma eficiente con la base de datos, ejecutar consultas y almacenar los resultados de los análisis y métricas generadas, y un **Dockerfile** que define la construcción de la imagen del contenedor para el backend.
    - **/encoders**: Utilidades para codificación y transformación de datos.
    - **/jasome-0.6.8-alpha**: Librería externa utilizada para el análisis estático de código Java.
    - **/outputMetrics**: Directorio donde se encuentran las métricas procesadas de las evaluaciones de modelos.
    - **/Resultados**: Directorio donde se almacenan métricas generadas por los análisis.
    - **/scalers**: Utilidades para la normalización y escalado de datos.
    - **/uploads**: Carpeta donde se almacenan los modelos predictivos subidos por los usuarios para su evaluación.
- **/frontend**: Contiene la parte del cliente de la plataforma, desarrollada con React. Además, dispone de un Dockerfile que define la construcción de la imagen del contenedor del frontend.
    - **/node_modules**: Dependencias instaladas de Node.js necesarias para el frontend.
    - **/public**: Archivos estáticos accesibles desde el navegador (HTML, imágenes, favicon).
    - **/src**: Código fuente principal de la aplicación React.
        - **/assets**: Recursos como imágenes, iconos y fuentes utilizadas en la interfaz.
        - **/components**: Componentes reutilizables de React que componen la interfaz.
        - **/context**: Archivos que gestionan el estado global de la aplicación.
        - **/styles**: Hojas de estilo y temas visuales para la plataforma.

7.2.1. Inserción del plugin OpenClover en el pom.xml

Tal como se observa en el fragmento de código 7.1, se tiene como objetivo añadir automáticamente el plugin *OpenClover* al archivo pom.xml de un proyecto Maven. La función se encarga de abrir y analizar el XML del pom.xml, comprobar si el plugin ya existe y, en caso contrario, insertarlo junto con sus configuraciones necesarias, asegurando que el plugin esté disponible para el proceso de análisis.

OpenClover es una herramienta que permite medir la cobertura de código en proyectos Java, es decir, qué porcentaje del código fuente es ejecutado durante la ejecución de las pruebas. Esto resulta muy útil para evaluar la calidad del conjunto de pruebas de un proyecto.

En el contexto del proyecto, se utiliza *OpenClover* para enriquecer los análisis de los proyectos subidos, obteniendo métricas adicionales sobre la calidad de sus pruebas (véase el anexo C).

Extracto de código 7.1: Inserción del plugin OpenClover en el pom.xml

```
def add_OpenClover_plugin_to_pom(pom_file, plugin_data):
    # Parsear el archivo XML
    parser = etree.XMLParser(remove_blank_text=True)
    tree = etree.parse(pom_file, parser)
    root = tree.getroot()

    # Namespaces
    nsmap = {None: 'http://maven.apache.org/POM/4.0.0'}

    # Encontrar el elemento <plugins>
    build = root.find('.//{http://maven.apache.org/POM/4.0.0}build')
    if build is None:
        # Crear <build> si no existe
        build = etree.SubElement(root, '{http://maven.apache.org/POM/4.0.0}build')
    plugins = build.find('{http://maven.apache.org/POM/4.0.0}plugins')
    if plugins is None:
        # Crear <plugins> si no existe
        plugins = etree.SubElement(build, '{http://maven.apache.org/POM/4.0.0}plugins'
            )

    # Primero comprobar que no existe ya
    if not plugin_exists(plugins, plugin_data):

        # Crear el nuevo plugin
        plugin = etree.Element('{http://maven.apache.org/POM/4.0.0}plugin')

        # Anadir groupId, artifactId, y version
        group_id_elem = etree.SubElement(plugin, '{http://maven.apache.org/POM/4.0.0}
            groupId')
        group_id_elem.text = plugin_data['groupId']
        artifact_id_elem = etree.SubElement(plugin, '{http://maven.apache.org/POM
            /4.0.0}artifactId')
        artifact_id_elem.text = plugin_data['artifactId']
        version_elem = etree.SubElement(plugin, '{http://maven.apache.org/POM/4.0.0}
            version')
        version_elem.text = plugin_data['version']

        # Anadir <executions> si existe
        if 'executions' in plugin_data:
           executions = etree.SubElement(plugin, '{http://maven.apache.org/POM/4.0.0}
               executions')
           for execution_data in plugin_data['executions']:
                execution = etree.SubElement(executions, '{http://maven.apache.org/POM
                    /4.0.0}execution')
                goals = etree.SubElement(execution, '{http://maven.apache.org/POM
                    /4.0.0}goals')
```

```
            for goal in execution_data['goals']:
                goal_elem = etree.SubElement(goals, '{http://maven.apache.org/POM
                    /4.0.0}goal')
                goal_elem.text = goal

    # Anadir <configuration> si existe
    if 'configuration' in plugin_data:
        configuration = etree.SubElement(plugin, '{http://maven.apache.org/POM
            /4.0.0}configuration')
        for key, value in plugin_data['configuration'].items():
            elem = etree.SubElement(configuration, '{http://maven.apache.org/POM
                /4.0.0}' + key)
            if isinstance(value, list):
                for item in value:
                    item_elem = etree.SubElement(elem, '{http://maven.apache.org/
                        POM/4.0.0}exclude')
                    item_elem.text = item
            else:
                elem.text = value

    # Anadir el nuevo plugin a <plugins>
    plugins.append(plugin)

    # Escribir el XML modificado nuevamente en el archivo sin declaracion XML
    tree.write(pom_file, pretty_print=True, xml_declaration=False, encoding='UTF-8
        ')
```

7.2.2. Evaluación de modelos

7.2.2.1. Extracto 1: Carga dinámica de modelos

El fragmento 7.2 implementa un cargador dinámico de modelos, capaz de reconocer y abrir diferentes formatos utilizados en machine learning y deep learning, siendo:

- `pkl`
- `joblib`
- `h5`
- `bin`
- `pt`
- `zip`

Gracias a esta función, la plataforma no se limita a un único framework, sino que soporta modelos de Scikit-learn, Keras/TensorFlow, PyTorch y TabNet.

Extracto de código 7.2: Carga dinámica de modelos

```
def load_model_dynamic(model_path):
    ext = os.path.splitext(model_path)[1].lower()
    if ext == '.pkl' or ext == '.joblib':
        return joblib.load(model_path)
    elif ext == '.h5':
        from tensorflow.keras.models import load_model
        return load_model(model_path)
    elif ext == '.bin' or ext == '.pt':
        return torch.load(model_path, map_location=torch.device('cpu'))
    elif ext == '.zip' or ext == '.pt':
        model = TabNetClassifier()
        model.load_model(model_path)
        return model
    else:
        raise ValueError(f"Formato de modelo no soportado: {ext}")
```

7.2.2.2. Extracto 2: Cálculo de métricas de evaluación

En el extracto de código 7.3 se calculan las principales métricas de evaluación utilizadas en la plataforma:

- **Precision, recall y F1-score**, que permiten evaluar el rendimiento de los modelos en términos clásicos de clasificación.
- **MCC (Matthews Correlation Coefficient)**, una métrica robusta especialmente útil en problemas desbalanceados (Balderas-Díaz et al., 2025).
- **AUC-ROC**, que mide la capacidad discriminativa global del modelo.

Extracto de código 7.3: Cálculo de métricas de evaluación

```
precision = TP / (TP + FP) if (TP + FP) > 0 else 0
recall = TP / (TP + FN) if (TP + FN) > 0 else 0
f1 = 2 * precision * recall / (precision + recall) if (precision + recall) > 0
    else 0

mcc_num = (c * s) - np.dot(t_k, p_k)
mcc_den = np.sqrt((s**2 - np.dot(p_k, p_k)) * (s**2 - np.dot(t_k, t_k)))
mcc = mcc_num / mcc_den if mcc_den != 0 else 0

auc = roc_auc_score(y_true, y_score[:, 1], average='weighted', multi_class='ovr')
```

7.2.3. Gestión de temas en la interfaz

El extracto de código 7.4 implementa el sistema de personalización de temas de la plataforma. Se definen tres paletas de colores (`Claro`, `Oscuro` y `Azul`) que se aplican dinámicamente mediante la *Context API* de React y variables CSS. El tema seleccionado se almacena en *localStorage*, garantizando que la preferencia del usuario se conserve entre sesiones.

Extracto de código 7.4: Gestión de temas en la interfaz

```
import React, { createContext, useState, useContext, useEffect } from "react";

// Definir los temas con colores personalizados
const themes = {
  light: {
    background: "#f5f5f5",
    componentBackground: "#ffffff",
    text: "#333333",
    primary: "#00796b",
    inputBackground: "#ffffff",
    buttonBackground: "#00796b",
    buttonText: "#ffffff",
    border: "#00796b",
    buttonHover:"#0cad9b",
    borderHover: "#00796b",
    borderFocused: "#00796b",
    labelFocused: "#00796b",
    rowEven: "#f9f9f9",
    rowOdd: "#ffffff",
    hoverRow: "#e0e0e0"
  },
  dark: {
    background: "#121212",
    componentBackground: "#383535",
    text: "#ffffff",
    primary: "#1DB954",
    inputBackground: "#333",
    buttonBackground: "#1DB954",
    buttonText: "#ffffff",
    border: "#1DB954",
    buttonHover:"#0cab44",
    borderHover: "#0cab44",
    borderFocused: "#0cab44",
    labelFocused: "#0cab44",
    rowEven: "#2a2a2a",
    rowOdd: "#383535",
    hoverRow: "#444444"
  },
  blue: {
    background: "#e0f7fa",
    componentBackground: "#ffffff",
    text: "#333333",
    primary: "#0288d1",
    inputBackground: "#ffffff",
    buttonBackground: "#0288d1",
    buttonText: "#ffffff",
    border: "#0288d1",
    buttonHover:"#61b9ff",
    borderHover: "#01579b",
    borderFocused: "#0288d1",
    labelFocused: "#0288d1",
    rowEven: "#d6eef7",
    rowOdd: "#ffffff",
    hoverRow: "#b3e5fc"
  }
};


// Crear el contexto
const ThemeContext = createContext();

// Proveedor del contexto de temas
export const ThemeProvider = ({ children }) => {
  const savedTheme = localStorage.getItem("theme") || "light";
  const [theme, setTheme] = useState(savedTheme);

  useEffect(() => {
    localStorage.setItem("theme", theme);
    const currentTheme = themes[theme];
```

```
    Object.keys(currentTheme).forEach((key) => {
      document.documentElement.style.setProperty(`--${key}`, currentTheme[key]);
    });
  }, [theme]);

  return (
    <ThemeContext.Provider value={{ theme, setTheme, themes }}>
      {children}
    </ThemeContext.Provider>
  );
};

// Hook para acceder al contexto
export const useTheme = () => useContext(ThemeContext);
```

7.3. Scripts de base de datos

La plataforma incluye un script principal denominado `init.sql`, encargado de inicializar la base de datos. Este script crea todas las tablas necesarias para el funcionamiento del sistema y carga los datos iniciales, como usuarios por defecto, roles y configuraciones básicas de la plataforma.

La ejecución del script se automatiza mediante Docker: al levantar los contenedores, el contenedor de PostgreSQL ejecuta automáticamente `init.sql` gracias a la configuración del volumen:

```
volumes:
- ./init.sql:/docker-entrypoint-initdb.d/init.sql
```

De esta forma, cada vez que se inicia la plataforma en un entorno nuevo, la base de datos queda lista para su uso sin intervención manual.

Ejemplo de fragmento de init.sql

A modo de ilustración, se muestra el extracto del script 7.5, donde se crea la tabla `users` y se inserta un usuario:

Extracto de código 7.5: Creación de la tabla users e inserción de usuario

```
CREATE TABLE public.users (
    id integer NOT NULL,
    email character varying(100) NOT NULL,
    password character varying(255) NOT NULL,
    role character varying(50) DEFAULT 'user'::character varying NOT NULL,
    name character varying(100) NOT NULL,
    lastname1 character varying(100) NOT NULL,
    lastname2 character varying(100),
    organization character varying(255),
    isstudent boolean NOT NULL,
    created_at timestamp without time zone DEFAULT CURRENT_TIMESTAMP,
    account_status character varying(20) DEFAULT 'active'::character varying NOT NULL,
    CONSTRAINT users_account_status_check CHECK (((account_status)::text = ANY ((ARRAY
        ['active'::character varying, 'blocked'::character varying])::text[]))),
    CONSTRAINT users_rol_check CHECK (((role)::text = ANY ((ARRAY['user'::character
        varying, 'admin'::character varying, 'superuser'::character varying])::text[])
        ))
);
```

```
INSERT INTO users (email, password, role, name, lastname1, isStudent)
VALUES ('superuser@pmt.com', '$2b$12$2bux0oIP9.LiAsrySytIN.yzDayeOnp2dgYFH1kMEdOOR.
    FEv3fRC', 'superuser', 'Mario', 'Rosano', false);
```

Este ejemplo demuestra cómo se establece la estructura básica de la plataforma y cómo se introducen los datos esenciales para su funcionamiento.

8. Pruebas del sistema

En este capítulo se presenta el plan de pruebas del sistema de información, incluyendo los diferentes tipos de pruebas que se han llevado a cabo, tanto las manuales (mediante listas de comprobación) como las automatizadas mediante herramientras específicas de pruebas.

8.1. Pruebas de accesibilidad

Estas pruebas tienen como objetivo verificar que la plataforma **PMTlab** cumple con los principios básicos de accesibilidad web, garantizando que cualquier usuario pueda interactuar de forma efectiva con la aplicación.

Se evalúan aspectos visuales, de navegación por teclado y contraste de colores.

8.1.1. Prueba 1: Navegación mediante teclado

Objetivo: Comprobar que es posible navegar por la plataforma sin necesidad de ratón, utilizando únicamente el teclado.

Procedimiento:

1. Acceder a la plataforma **PMTlab**.
2. Usar la tecla **Tab** o las flechas direccionales para desplazarse entre los distintos elementos interactivos (botones, campos de texto, menús, etc.).
3. Utilizar **Enter** o **Espacio** para activar cada elemento.

Resultado esperado: Todos los elementos son accesibles mediante teclado, el foco visual se resalta correctamente y no existen zonas inaccesibles.

Evidencia:

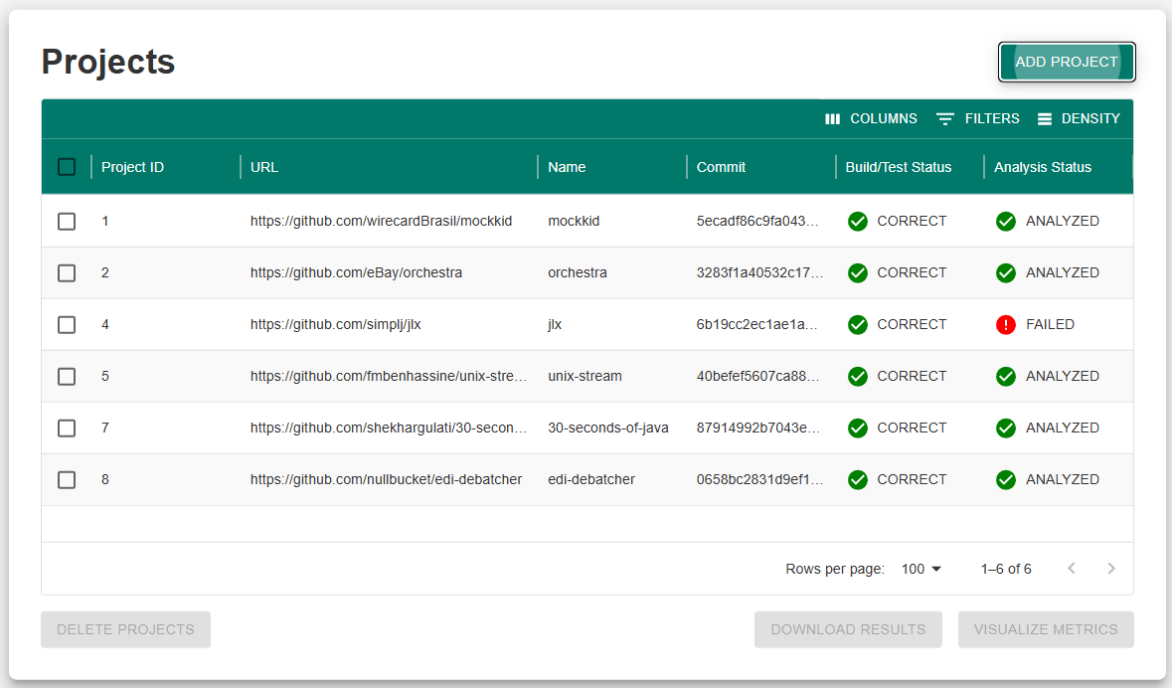


Figura 8.1: Navegación mediante teclado con foco visible en el botón “ADD PROJECT”

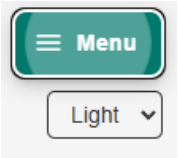


Figura 8.2: Navegación mediante teclado con foco visible en el botón “Menu”

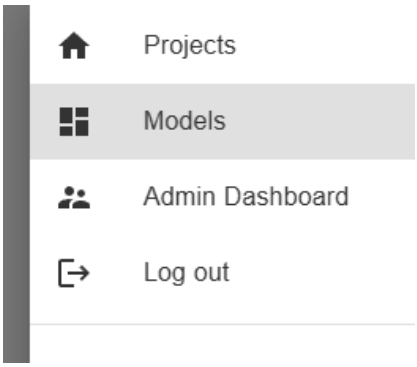


Figura 8.3: Navegación mediante teclado con foco visible en la pestaña “Models” del menú lateral

8.1.2. Prueba 2: Contraste de colores

Objetivo: Verificar que los colores empleados en la interfaz presentan suficiente contraste entre texto y fondo para garantizar la legibilidad.

Procedimiento:

1. Acceder a la plataforma PMTlab desde Google Chrome.
2. Activar la extensión WAVE desde la barra de extensiones del navegador.
3. Ejecutar el análisis de accesibilidad sobre la página principal de la plataforma.
4. Acceder a la pestaña *Contrast* de WAVE para visualizar los resultados específicos de contraste de color.
5. Revisar los avisos y errores de contraste detectados para los distintos elementos de la interfaz.
6. Cambiar la paleta de colores desde la plataforma (*Light*, *Dark* y *Blue*) y repetir el análisis en cada una de ellas.
7. Analizar si los posibles errores de contraste afectan a elementos funcionales propios de la plataforma o a componentes externos.

Resultado esperado:

Los textos, botones, formularios y elementos interactivos desarrollados específicamente para PMTlab deben cumplir los niveles mínimos de contraste exigidos por WCAG 2.1. Los posibles avisos generados por componentes externos no controlables no deben comprometer la legibilidad ni la interacción del usuario.

Resultado obtenido:

El análisis con WAVE detectó algunos avisos de contraste en determinadas celdas del componente de tabla (*DataGrid*) utilizado por la plataforma. Dichos avisos se deben al comportamiento interno y a las hojas de estilo del componente de terceros (Material UI), que no permite modificar completamente ciertos estilos de contraste sin alterar su funcionamiento.

No obstante, todos los elementos clave desarrollados para PMTlab (botones, formularios, textos, etiquetas, enlaces y cabeceras) cumplen los niveles de contraste requeridos. En las tres paletas de colores disponibles (*Light*, *Dark* y *Blue*), la información principal y los controles de interacción se muestran de forma clara y legible.

Evidencia:

- Informes visuales proporcionados por la extensión WAVE, mostrando los avisos de contraste asociados al componente *DataGrid*.
- Ausencia de errores de contraste en los elementos de interfaz propios de la plataforma (formularios, botones, encabezados y textos).
- Resultados consistentes al repetir el análisis en las tres paletas de colores disponibles.

8.1.3. Prueba 3: Tamaño de fuente y escalado

Objetivo: Comprobar que la interfaz mantiene su legibilidad y estructura al modificar el nivel de zoom del navegador.

Procedimiento:

1. Acceder a la plataforma desde el navegador.
2. Incrementar el zoom al 150 % y 200 %.
3. Verificar que el texto sigue siendo legible y que la maquetación no se rompe.

Resultado esperado: La interfaz se adapta sin perder estructura ni legibilidad.

Evidencia:

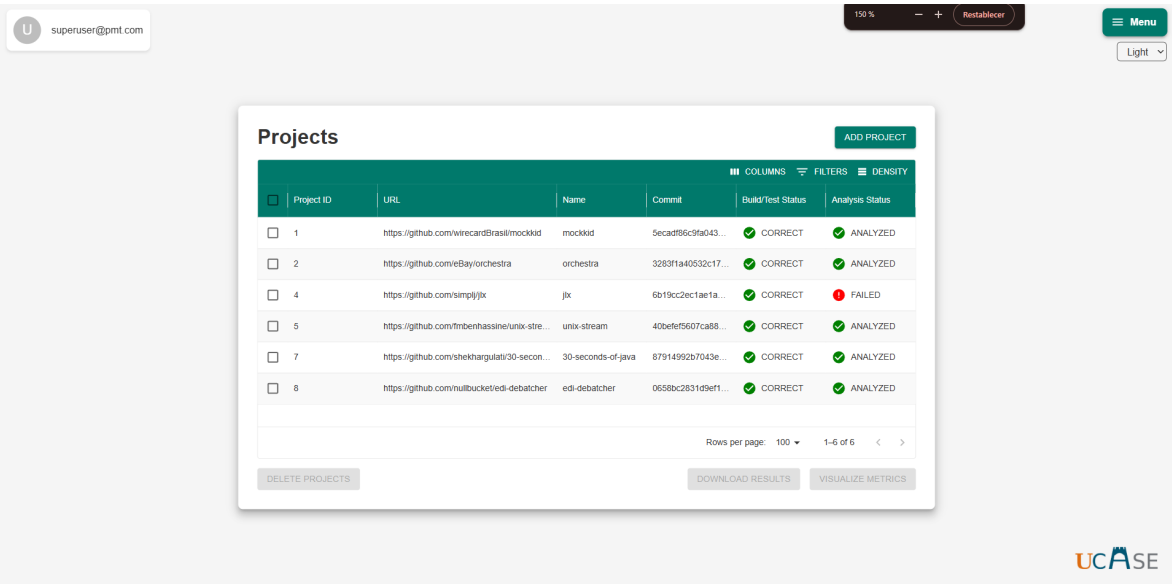


Figura 8.4: Prueba de escalado y adaptabilidad visual con zoom del 150 %

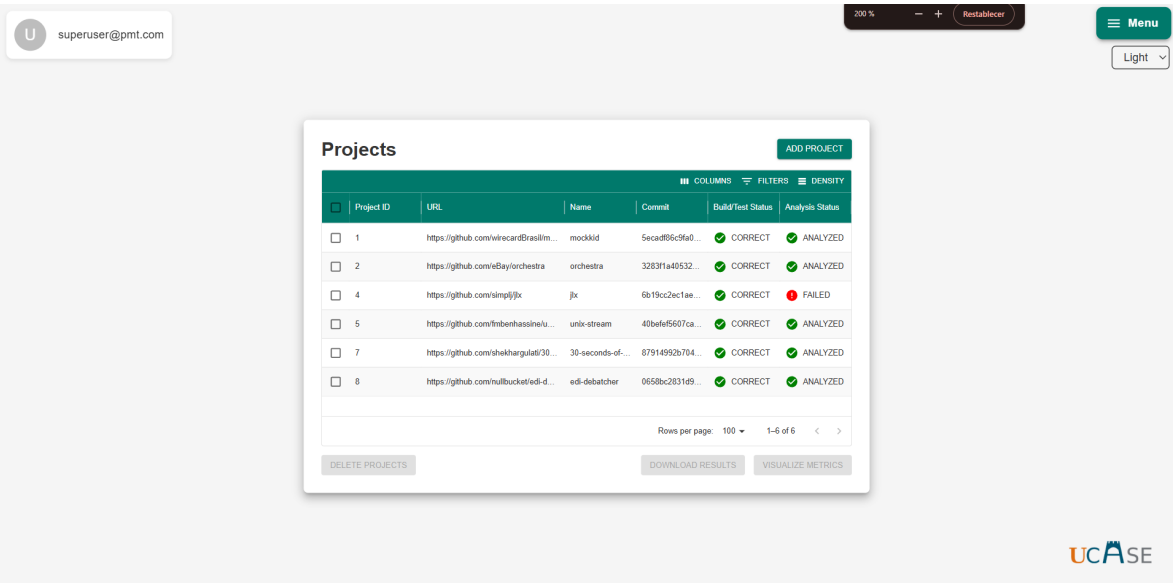


Figura 8.5: Prueba de escalado y adaptabilidad visual con zoom del 200 %

8.2. Pruebas unitarias

En esta sección se presentan las pruebas unitarias realizadas para verificar el correcto funcionamiento de los distintos módulos que componen la plataforma **PMTlab**. Estas pruebas permiten asegurar que cada componente del sistema cumple con los requisitos funcionales definidos y responde adecuadamente ante distintas condiciones de uso.

Las pruebas se han ejecutado de forma manual comprobando tanto el comportamiento esperado como la respuesta ante entradas erróneas. Cada prueba incluye su objetivo, procedimiento, resultado esperado y evidencia gráfica del funcionamiento del sistema.

8.2.1. Pruebas unitarias del inicio de sesión

El siguiente conjunto de pruebas tiene como objetivo comprobar el correcto funcionamiento del proceso de autenticación. Estas pruebas se realizaron de manera manual, verificando tanto el comportamiento esperado como la respuesta visual del sistema en diferentes escenarios.

8.2.1.1. Prueba 1: Inicio de sesión correcto

Objetivo: Verificar que un usuario registrado puede acceder al sistema con credenciales válidas.

Procedimiento:

1. Acceder a la página de inicio de sesión de PMTlab.

2. Introducir un correo electrónico registrado: `superuser@pmt.com`.
3. Introducir la contraseña correcta: `root`.
4. Pulsar el botón **"LOG IN"**.

   Resultado esperado: El sistema redirige al panel principal del usuario.

   Resultado obtenido: Conforme al esperado.

   Evidencia:

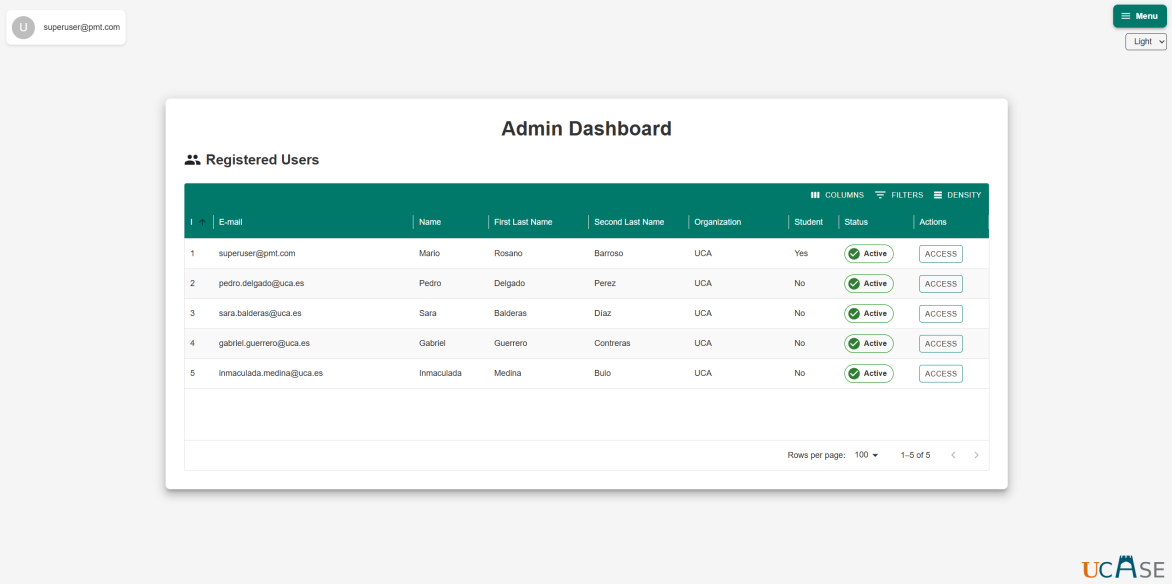


Figura 8.6: Login correcto

8.2.1.2. Prueba 2: Contraseña incorrecta

Objetivo: Comprobar que el sistema muestra un mensaje de error cuando la contraseña no es válida.

Procedimiento:

1. Introducir el correo electrónico: `superuser@pmt.com`.
2. Introducir una contraseña incorrecta, por ejemplo: `12345`.
3. Pulsar el botón **"LOG IN"**.

Resultado esperado: El sistema rechaza el acceso y muestra un mensaje informativo.

Resultado obtenido: Conforme al esperado.

Evidencia:

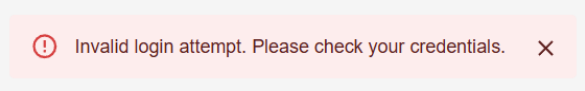


Figura 8.7: Login incorrecto

8.2.1.3. Prueba 3: Usuario no registrado

Objetivo: Validar la respuesta del sistema ante un intento de inicio de sesión con un correo inexistente.

Procedimiento:

1. Introducir el correo electrónico: `usuarioinexistente@mail.com`.
2. Introducir una contraseña cualquiera.
3. Pulsar el botón **"LOG IN"**.

Resultado esperado: El sistema rechaza el acceso y muestra un mensaje informativo.

Resultado obtenido: Conforme al esperado.

Evidencia:

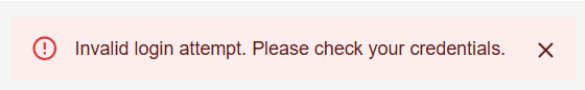


Figura 8.8: Usuario no registrado

8.2.1.4. Prueba 4: Formato de correo inválido

Objetivo: Verificar que el sistema detecta y bloquea la introducción de un correo electrónico con formato incorrecto en la pantalla de inicio de sesión.

Procedimiento:

1. Introducir en el campo *E-mail* el valor: `usuario.prueba`, sin incluir el símbolo "@" ni dominio.
2. Introducir una contraseña cualquiera, por ejemplo: `Test1234`.
3. Pulsar el botón **"LOG IN"**.

Resultado esperado: El sistema muestra un mensaje de error indicando que el formato del correo electrónico no es válido. No se permite continuar con el proceso de autenticación.

Resultado obtenido: Conforme al esperado.

Evidencia:

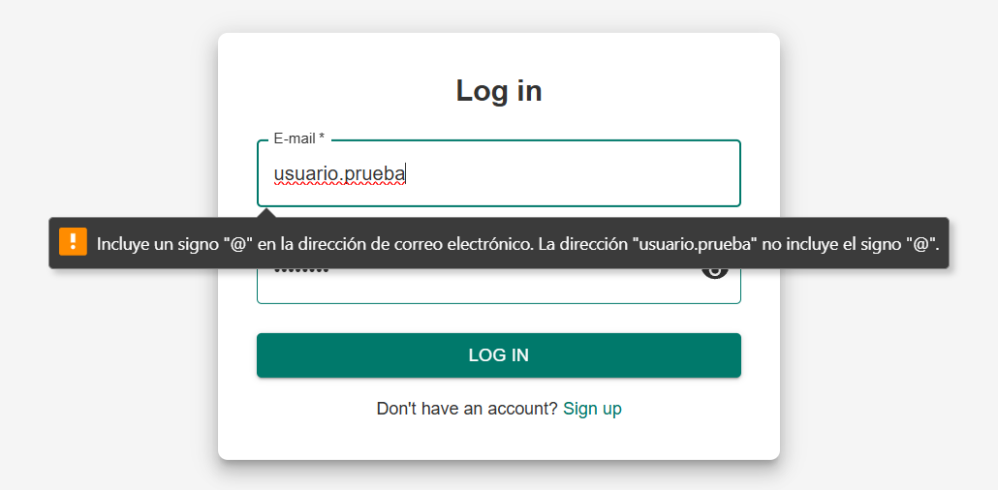


Figura 8.9: Formato de correo electrónico incorrecto

8.2.1.5. Prueba 5: Campos vacíos

Objetivo: Verificar que el sistema no permite el envío del formulario si no se completan los campos requeridos.

Procedimiento:

1. Dejar vacíos los campos de correo y contraseña.
2. Pulsar el botón **"LOG IN"**.

Resultado esperado: Aparece un mensaje de validación indicando que se deben completar los campos.

Resultado obtenido: Conforme al esperado.

Evidencia:

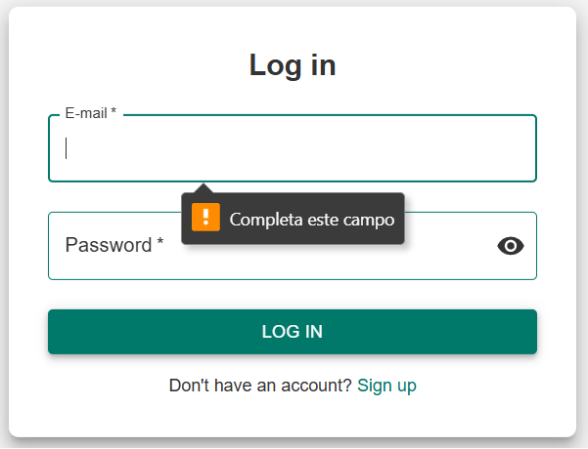


Figura 8.10: Correo electrónico vacío

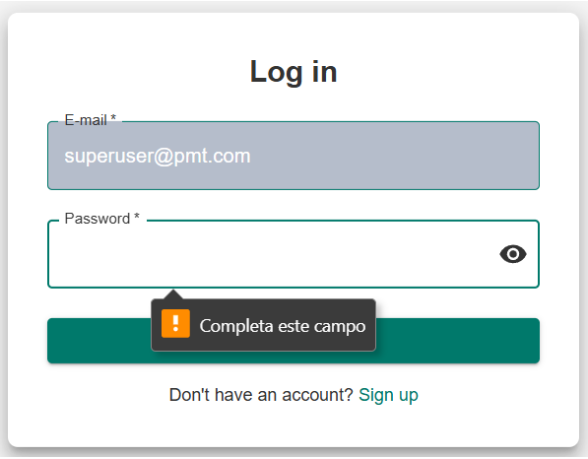


Figura 8.11: Contraseña vacía

8.2.1.6. Prueba 6: Cierre de sesión

Objetivo: Comprobar que el cierre de sesión invalida correctamente la sesión activa.

Procedimiento:

1. Iniciar sesión correctamente.
2. Abrir el menú lateral y pulsar el botón **"Log out"**.

Resultado esperado: El sistema redirige a la pantalla de inicio de sesión, eliminando la sesión activa del navegador.

Resultado obtenido: Conforme al esperado.

Evidencia:

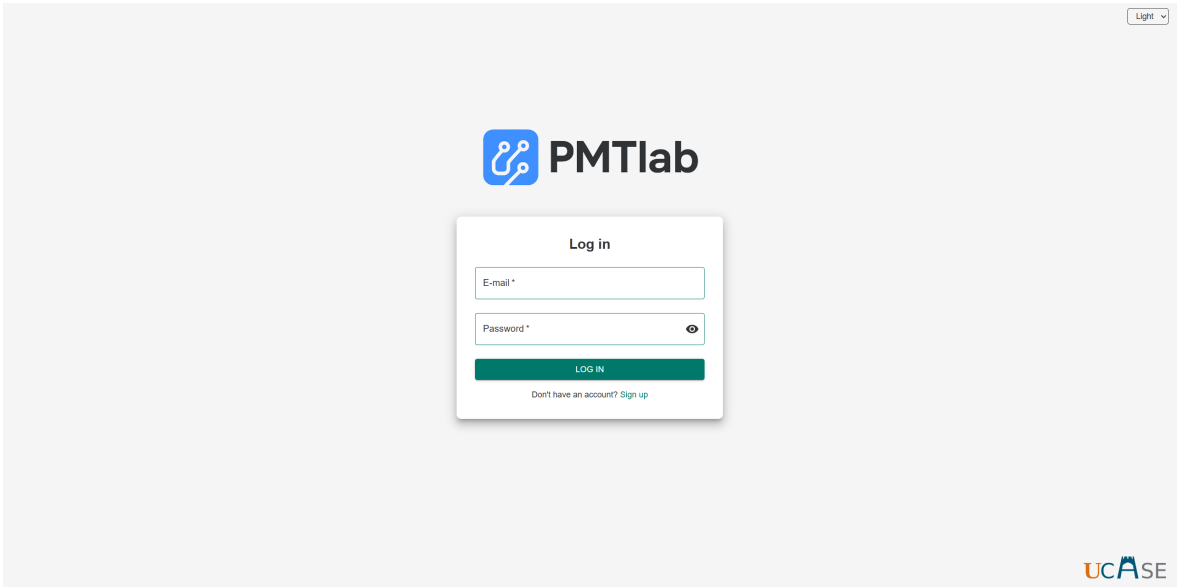


Figura 8.12: Log out correcto

8.2.2. Pruebas unitarias del registro de usuario

Estas pruebas tienen como objetivo verificar el correcto funcionamiento del proceso de registro de nuevos usuarios. Se ha comprobado tanto la validación de los campos del formulario como la creación efectiva del usuario en la base de datos.

8.2.2.1. Prueba 1: Registro correcto

Objetivo: Verificar que un nuevo usuario puede registrarse correctamente con todos los datos válidos.

Procedimiento:

1. Acceder a la pantalla de registro mediante el enlace *"Don't have an account? Sign up"*.
2. Completar los campos con los siguientes datos:
   - **Email:** `usuario.prueba@pmt.com`
   - **Contraseña:** `Test1234`
   - **Nombre:** Mario
   - **Apellido 1:** Rosano
   - **Apellido 2:** Barroso
   - **Organización:** UCA
   - **¿Es estudiante?:** Sí
3. Pulsar el botón **"SIGN UP"**.

Resultado esperado: El sistema redirige a la pantalla de inicio de sesión.

Resultado obtenido: Conforme al esperado.

Evidencia:

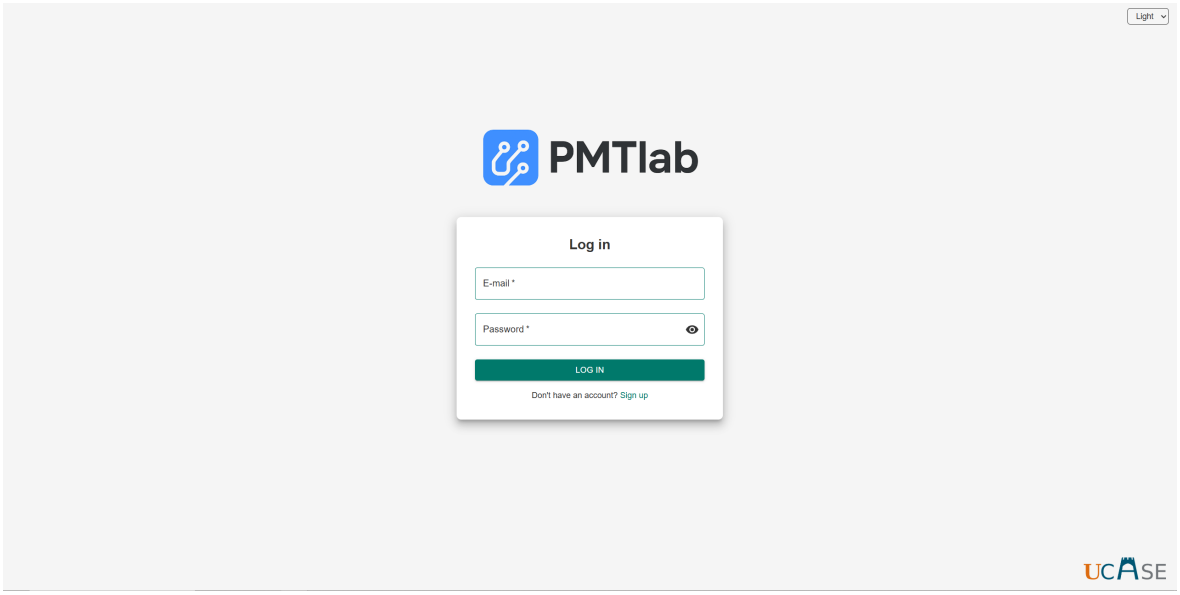


Figura 8.13: Registro de usuario correcto

8.2.2.2. Prueba 2: Campos obligatorios vacíos

Objetivo: Comprobar que el sistema no permite registrar un usuario si no se completan los campos requeridos.

Procedimiento:

1. Acceder a la pantalla de registro.
2. Dejar vacío el campo *E-mail*.
3. Pulsar el botón **"SIGN UP"**.

Resultado esperado: El sistema muestra mensajes de validación indicando que el campos es obligatorio y no permite continuar.

Resultado obtenido: Conforme al esperado.

Evidencia:

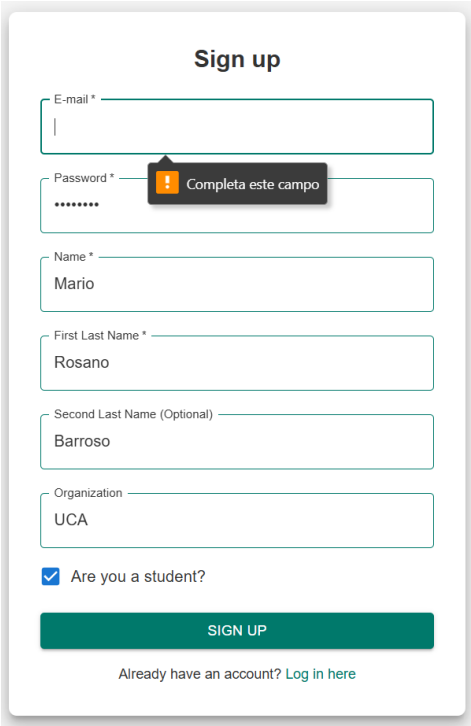


Figura 8.14: Correo electrónico vacío

8.2.2.3. Prueba 3: Formato de correo inválido

Objetivo: Verificar que el sistema detecta y bloquea correos con formato incorrecto.

Procedimiento:

1. Introducir en el campo de correo electrónico el valor: `usuario.prueba`.
2. Completar el resto de los campos correctamente.
3. Pulsar el botón **"SIGN UP"**.

Resultado esperado: Se muestra un mensaje de error indicando que el formato del correo electrónico no es válido.

Resultado obtenido: Conforme al esperado.

Evidencia:

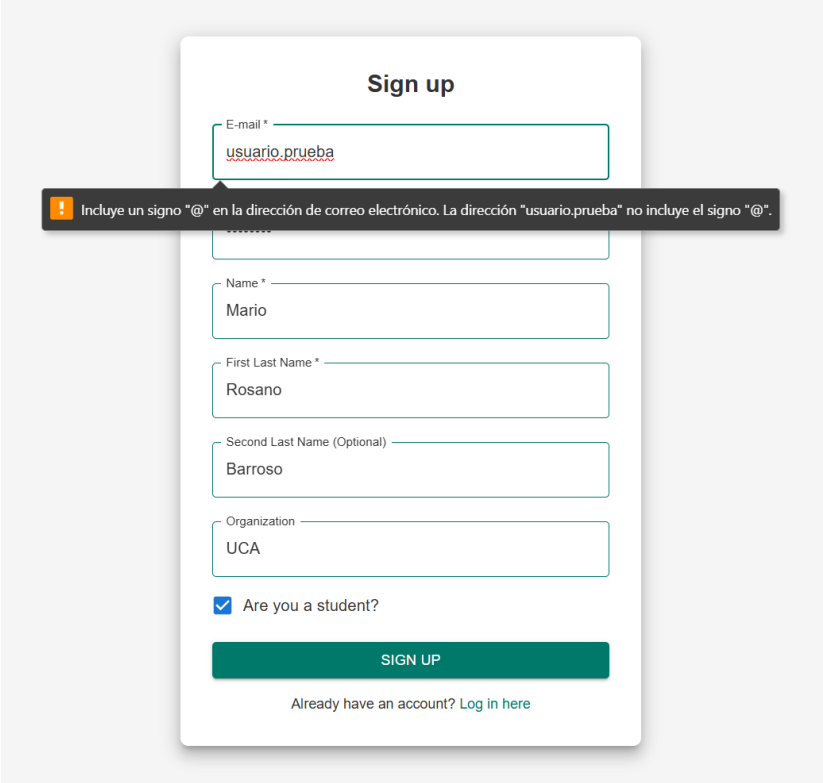


Figura 8.15: Formato de correo electrónico incorrecto

8.2.2.4. Prueba 4: Usuario ya existente

Objetivo: Verificar que el sistema impide el registro de un correo electrónico que ya está asociado a una cuenta existente.

Procedimiento:

1. Introducir un correo electrónico ya registrado, como `superuser@pmt.com`.
2. Completar el formulario correctamente.
3. Pulsar el botón **"SIGN UP"**.

Resultado esperado: El sistema muestra un mensaje de error.

Resultado obtenido: Conforme al esperado.

Evidencia:

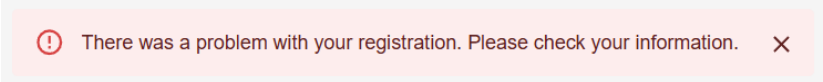


Figura 8.16: Usuario ya existente

8.2.3. Pruebas unitarias del módulo de Proyectos

El módulo de **Proyectos** permite la carga y análisis automático de repositorios o ficheros CSV. Estas pruebas verifican que los proyectos se suben correctamente, que el análisis se inicia de forma automática y que los resultados se muestran adecuadamente. También se comprueba el manejo de errores ante entradas no válidas o fallos durante el análisis.

8.2.3.1. Prueba 1: Subida y análisis correcto de un proyecto mediante URL válida

Objetivo: Verificar que el sistema acepta una URL válida de un repositorio público y realiza el análisis correspondiente.

Procedimiento:

1. Acceder al apartado **"Projects"** desde el menú lateral.
2. Pulsar el botón **"ADD PROJECT"**
3. Introducir la siguiente URL en el campo correspondiente: `https://github.com/wirecardBrasil/mockkid`.
4. Pulsar el botón **"CONFIRM"**.

Resultado esperado: El sistema comienza el proceso de análisis. El proyecto aparecerá en el datagrid y se mostrarán los estados del proceso hasta llegar al análisis correcto.

Resultado obtenido: Conforme al esperado.

Evidencia:

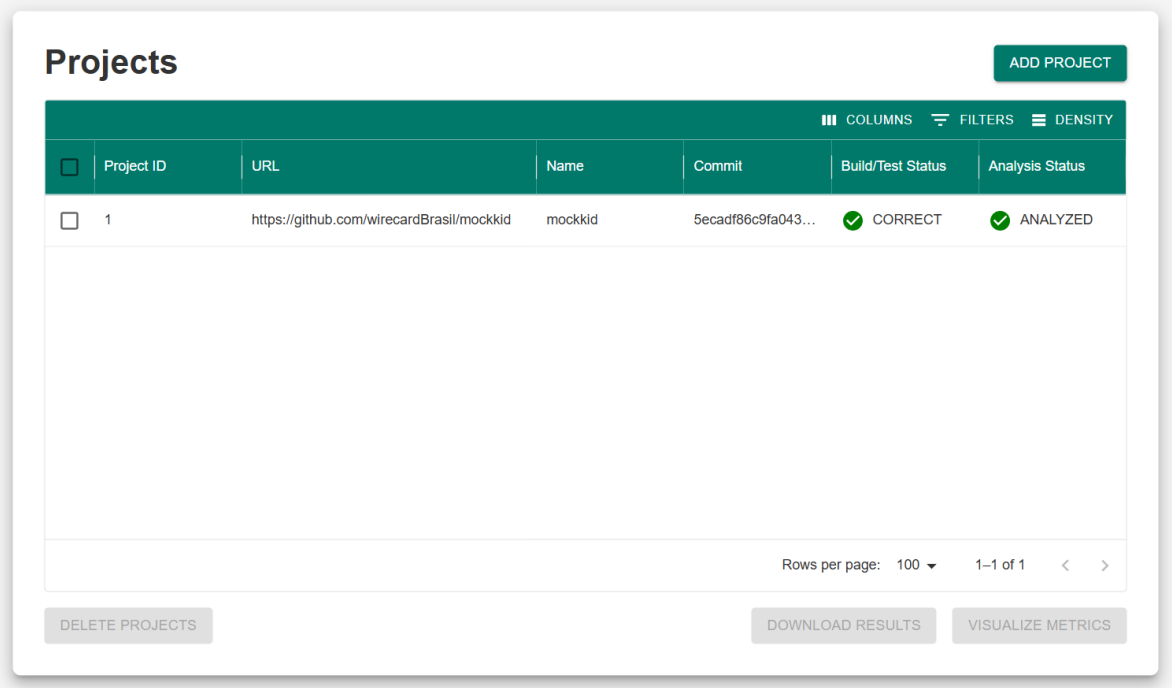


Figura 8.17: Proyecto subido y analizado correctamente

8.2.3.2. Prueba 2: Subida y análisis incorrecto de un proyecto mediante URL válida

Objetivo: Verificar el comportamiento del sistema cuando se introduce una URL válida de un repositorio público, pero el análisis no puede completarse correctamente por un error interno o por incompatibilidades del proyecto.

Procedimiento:

1. Acceder al apartado **"Projects"** desde el menú lateral.
2. Pulsar el botón **"ADD PROJECT"**
3. Introducir la siguiente URL en el campo correspondiente: `https://github.com/simplj/jlx`.
4. Pulsar el botón **"CONFIRM"**.

Resultado esperado: El sistema acepta la URL e inicia el proceso de análisis de forma automática. Durante la ejecución, se produce un error que impide completar el análisis, mostrando un mensaje de fallo y el estado de fallo en el datagrid.

Resultado obtenido: Conforme al esperado.

Evidencia:

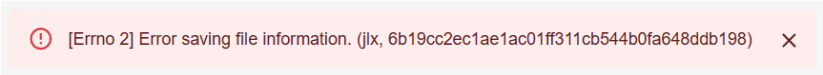


Figura 8.18: Mensaje de análisis erróneo

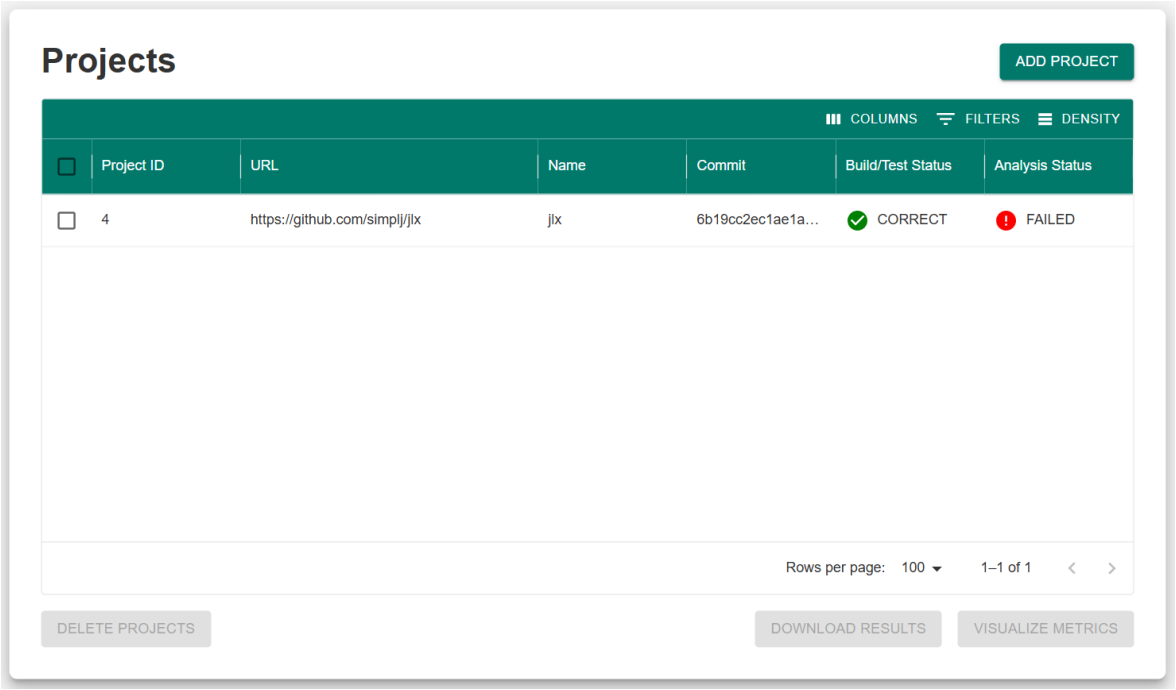


Figura 8.19: Proyecto subido y error en el análisis

8.2.3.3. Prueba 3: Subida de proyecto con URL inválida

Objetivo: Comprobar que el sistema valida el formato de la URL antes de iniciar el análisis.

Procedimiento:

1. Acceder al apartado **"Projects"** desde el menú lateral.
2. Pulsar el botón **"ADD PROJECT"**
3. Introducir en el campo de URL el texto: `mockkid-github`.
4. Pulsar el botón **"CONFIRM"**.

Resultado esperado: El sistema muestra un mensaje de error informando que la URL no tiene un formato válido y no inicia el análisis.

Resultado obtenido: Conforme al esperado.

Evidencia:

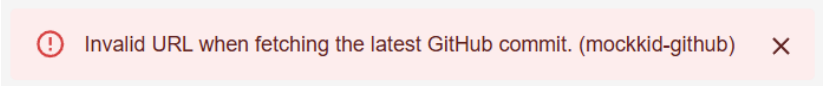


Figura 8.20: Mensaje de URL inválida

8.2.3.4. Prueba 4: Carga masiva de proyectos mediante fichero CSV

Objetivo: Verificar que el sistema permite subir varios proyectos simultáneamente desde un fichero CSV.

Procedimiento:

1. Acceder al apartado **"Projects"** desde el menú lateral.
2. Pulsar el botón **"ADD PROJECT"**
3. Cargar un archivo CSV con las URLs:
   - https://github.com/eBay/orchestra
   - https://github.com/nullbucket/edi-debatcher
   - https://github.com/shekhargulati/30-seconds-of-java
4. Pulsar el botón **"CONFIRM"**.

Resultado esperado: El sistema comienza el análisis de cada proyecto del fichero, mostrando su progreso individual y añadiéndolos a la lista una vez completados además de un mensaje de éxito.

Resultado obtenido: Conforme al esperado.

Evidencia:

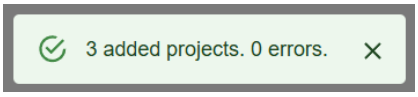


Figura 8.21: Mensaje de éxito con la carga de tres proyectos por CSV

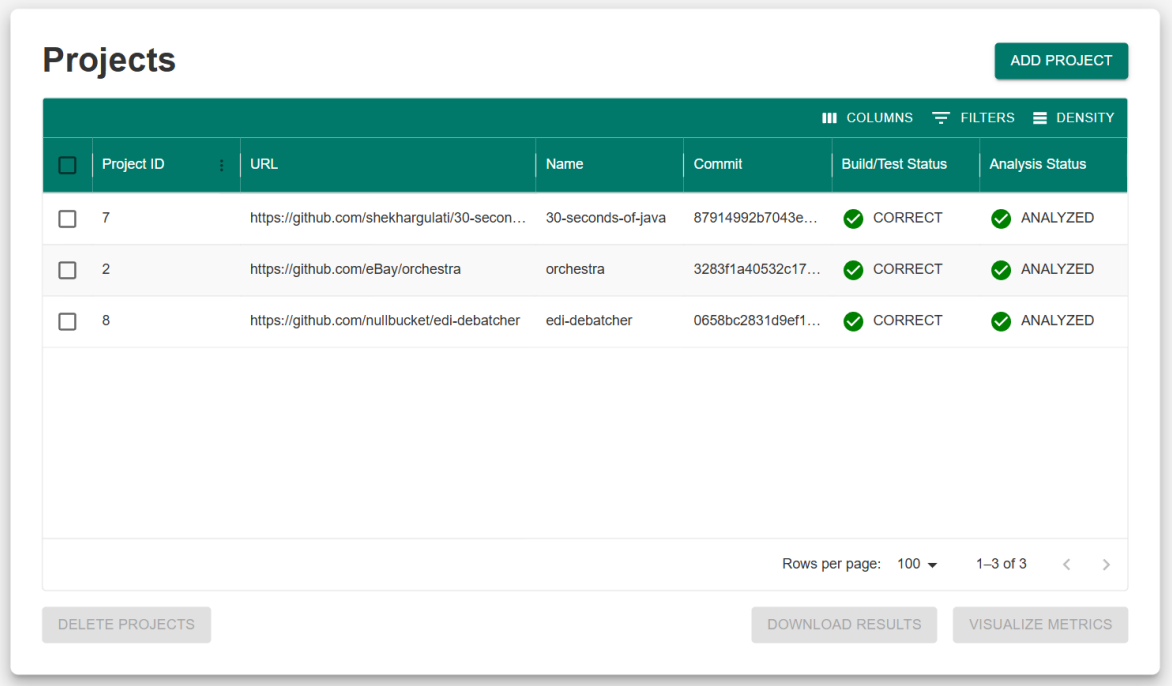


Figura 8.22: Proyectos subidos y análizados correctamente

8.2.3.5. Prueba 5: Descarga de resultados de proyectos analizados

Objetivo: Confirmar que los resultados pueden descargarse correctamente en formato CSV.

Procedimiento:

1. Acceder al apartado **"Projects"** desde el menú lateral.
2. Seleccionar uno o varios proyectos desde la lista.
3. Pulsar el botón **"DOWNLOAD RESULTS"**.
4. Comprobar los ficheros descargados.

Resultado esperado: El sistema descarga un archivo CSV por cada proyecto seleccionado, con las columnas y métricas correspondientes.

Resultado obtenido: Conforme al esperado.

Evidencia:

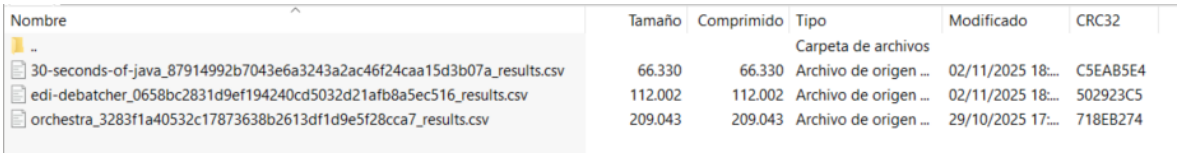


Figura 8.23: Resultados descargados en un fichero CSV

8.2.3.6. Prueba 6: Visualización de métricas de análisis

Objetivo: Comprobar que los datos de cada proyecto se visualizan correctamente en la interfaz.

Procedimiento:

1. Acceder al apartado **"Projects"** desde el menú lateral.
2. Seleccionar uno o varios proyectos previamente analizados.
3. Pulsar el botón **"VISUALIZE METRICS"**.
4. Filtrar las columnas visibles para mostrar únicamente `ID_MUTANT`, `LCOMSTAR`, `MTLOC` y `VG`.

Resultado esperado: La tabla actualiza su contenido y muestra solo las columnas seleccionadas, con datos consistentes.

Resultado obtenido: Conforme al esperado.

Evidencia:

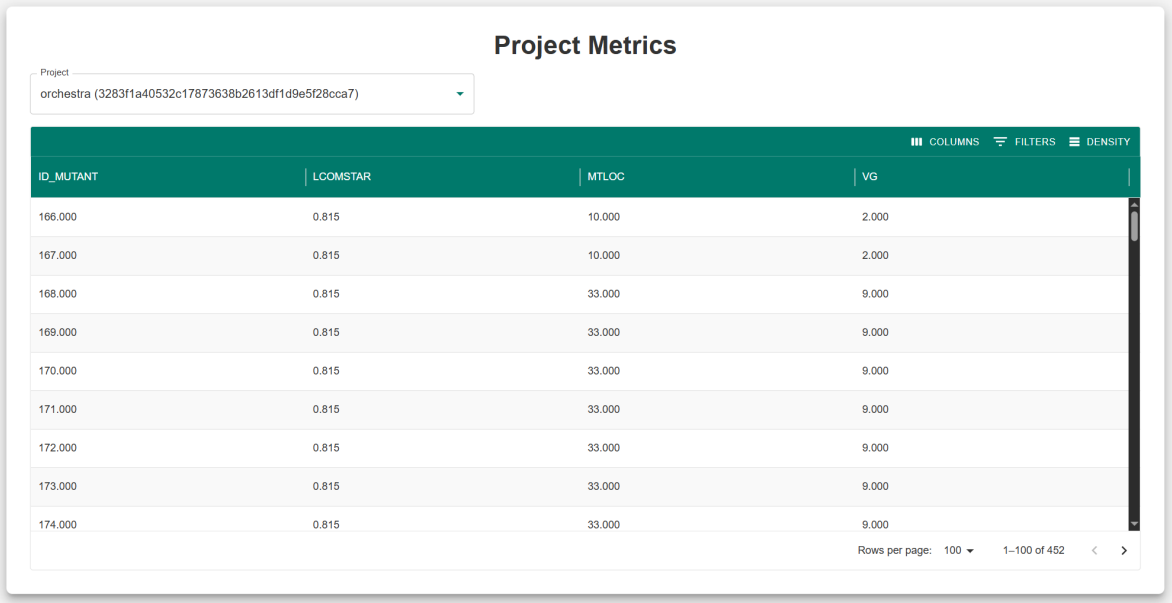


Figura 8.24: Visualización de métricas y filtros aplicados

8.2.3.7. Prueba 7: Eliminación de un proyecto existente

Objetivo: Verificar que el sistema permite eliminar correctamente un proyecto, actualizando la interfaz y eliminando su información asociada.

Procedimiento:

1. Acceder al apartado **"Proyects"** desde el menú lateral.
2. Seleccionar uno o varios proyectos desde la lista.

3. Pulsar el botón de **“DELETE PROJECTS”**.

Resultado esperado: El proyecto seleccionado se elimina de la tabla y desaparece de la lista de proyectos disponibles. El sistema muestra un mensaje de confirmación indicando que la eliminación se ha realizado correctamente.

Resultado obtenido: Conforme al esperado.

Evidencia:

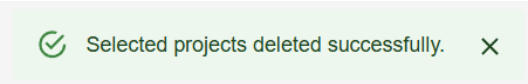


Figura 8.25: Mensaje de eliminación exitosa de proyectos

8.2.4. Pruebas del módulo de Modelos

En este apartado se verifican las funcionalidades principales relacionadas con la gestión y uso de modelos predictivos dentro de la plataforma. Estas pruebas se centran en la correcta visualización, selección y evaluación de modelos sobre proyectos previamente analizados.

8.2.4.1. Prueba 1: Subida de un modelo con formato válido

Objetivo: Comprobar que el sistema permite la subida de un modelo con una extensión compatible y registra correctamente su información en la base de datos.

Procedimiento:

1. Acceder al apartado **“Models”** desde el menú lateral.
2. Pulsar el botón **“UPLOAD MODEL”**.
3. Seleccionar el tipo modelo `Random Forest`.
4. Seleccionar un archivo con formato válido, por ejemplo `RandomForest_model.joblib`.
5. Pulsar el botón **“CONFIRM”**.

Resultado esperado: El sistema sube el modelo correctamente, muestra un mensaje de confirmación y lo añade al listado de modelos disponibles, mostrando su nombre, tipo y fecha de subida.

Resultado obtenido: Conforme al esperado.

Evidencia:

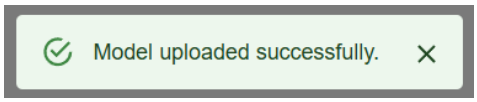


Figura 8.26: Mensaje de confirmación de subida

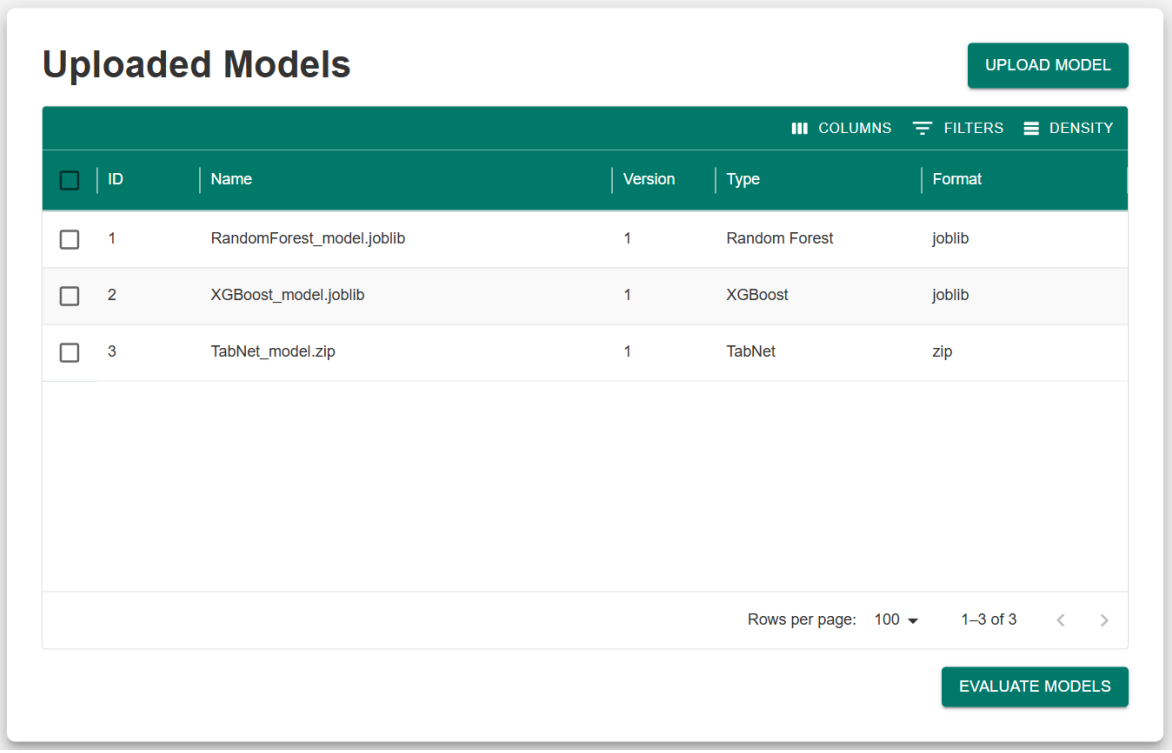


Figura 8.27: Modelo subido en el datagrid

8.2.4.2. Prueba 2: Subida de un modelo con una extensión no permitida

Objetivo: Verificar que el sistema impide la subida de archivos cuya extensión no sea compatible con los formatos de modelo admitidos.

Procedimiento:

1. Acceder al apartado **"Models"** desde el menú lateral.
2. Pulsar el botón **"UPLOAD MODEL"**.
3. Seleccionar un tipo de modelo cualquiera.
4. Seleccionar un archivo con una extensión no válida, por ejemplo `modelo.txt`.
5. Pulsar el botón **"CONFIRM"**.

Resultado esperado: El sistema muestra un mensaje de aviso indicando que el formato del archivo no es válido y no permite completar la subida.

Resultado obtenido: Conforme al esperado.

Evidencia:

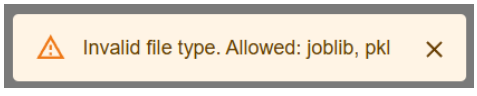


Figura 8.28: Mensaje de aviso de formato incorrecto

8.2.4.3. Prueba 3: Evaluación correcta de un proyecto con un modelo específico

Objetivo: Verificar que el sistema permite seleccionar un modelo y evaluar correctamente un proyecto previamente analizado.

Procedimiento:

1. Desde la pantalla de **Models**, seleccionar el modelo *RandomForest_model.joblib*.
2. Pulsar el botón **"EVALUATE MODELS"**.
3. En la ventana emergente, seleccionar un proyecto previamente analizado.
4. Pulsar el botón **"CONFIRM"**.

Resultado esperado: El sistema ejecuta la evaluación del modelo sobre el proyecto seleccionado. Una vez finalizada, se redirige automáticamente a la pantalla de resultados donde se muestran las métricas generadas (precision, recall, f1-score, AUC, etc.).

Resultado obtenido: Conforme al esperado.

Evidencia:

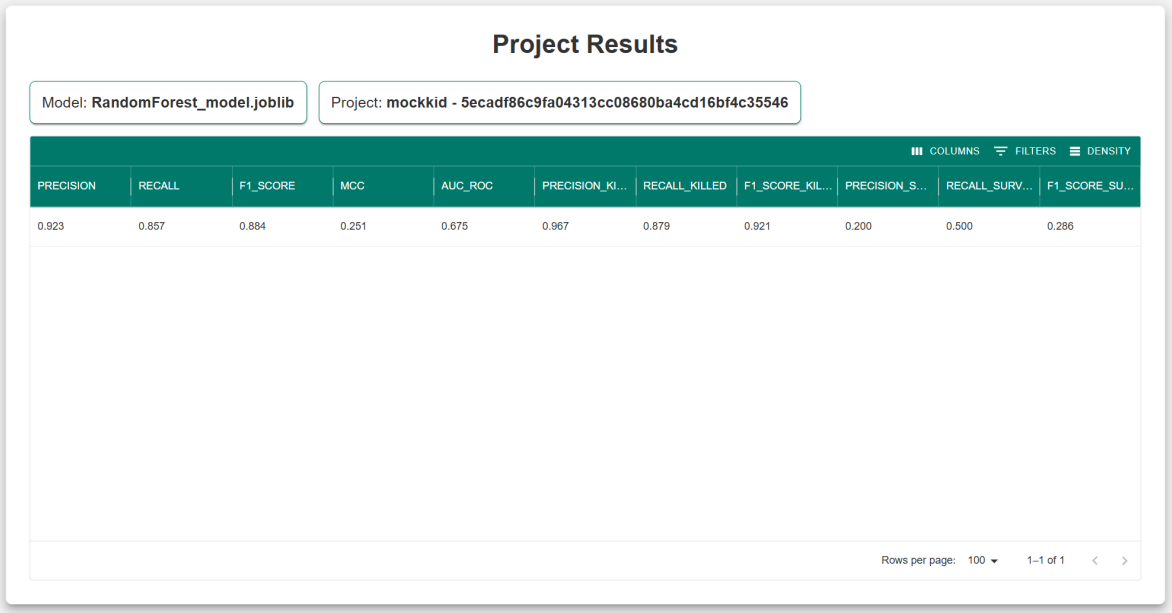


Figura 8.29: Evaluación correcta de un modelo sobre un proyecto analizado

8.2.4.4. Prueba 4: Evaluación de varios modelos sobre un mismo proyecto

Objetivo: Verificar que el sistema permite evaluar un mismo proyecto con múltiples modelos y comparar los resultados obtenidos.

Procedimiento:

1. Acceder a la sección **"Models"**.

2. Seleccionar todos los modelos disponibles.
3. Pulsar el botón **“EVALUATE MODELS”**.
4. Escoger un proyecto previamente analizado y confirmar la acción.

Resultado esperado: El sistema evalúa el proyecto con todos los modelos seleccionados y genera un conjunto de métricas comparativas. En la pantalla de resultados se muestran las diferencias entre modelos, permitiendo identificar cuál obtiene el mejor rendimiento.

Resultado obtenido: Conforme al esperado.

Evidencia:

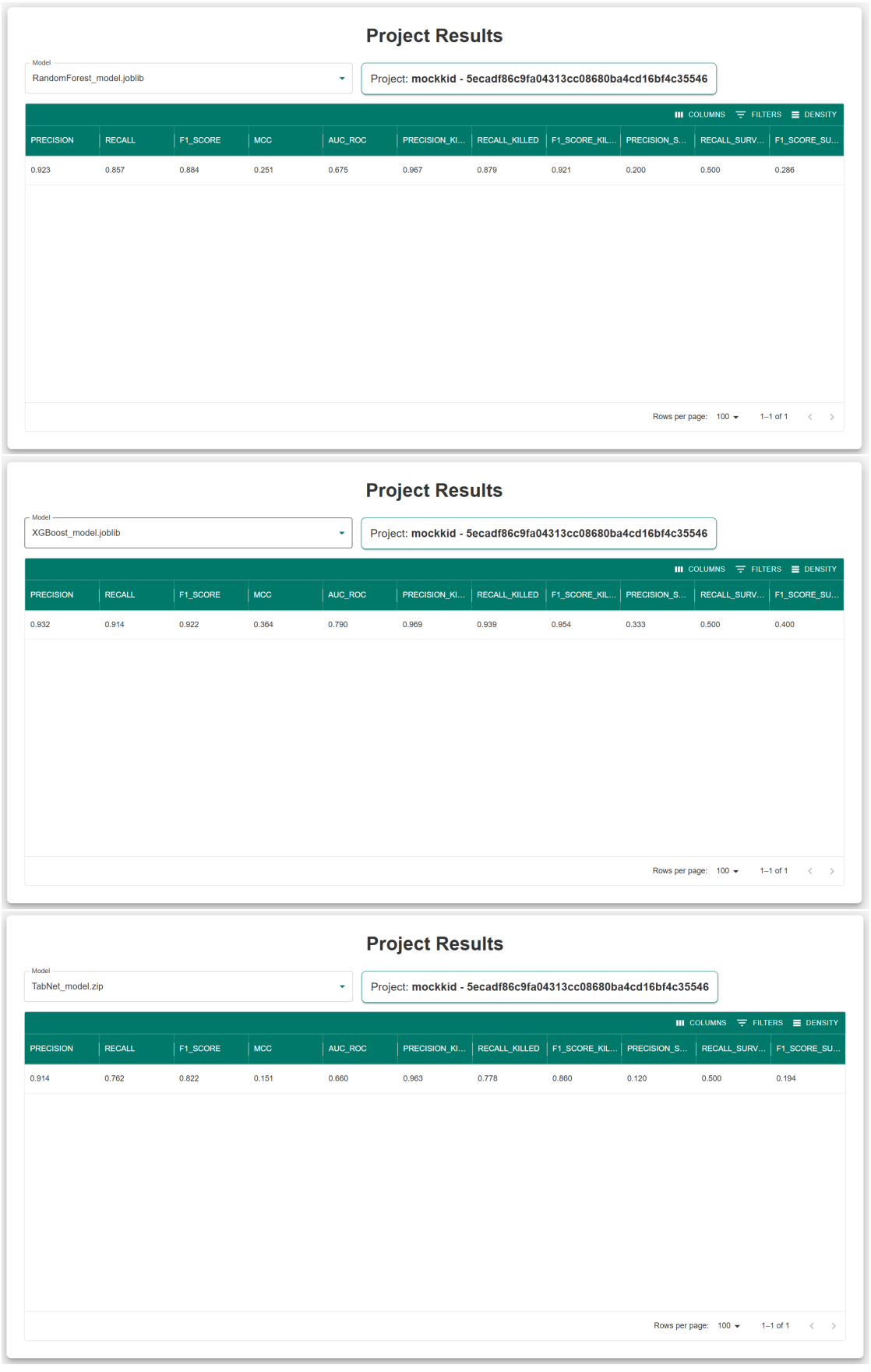


Figura 8.30: Comparación de métricas entre varios modelos

8.2.5. Pruebas unitarias del módulo de gestión de usuarios

En este apartado se recogen las pruebas realizadas sobre las funcionalidades relacionadas con la gestión de usuarios dentro de la plataforma PMTlab. Estas pruebas verifican que el sistema permite administrar los usuarios correctamente, controlar los permisos y gestionar los datos personales de forma segura.

8.2.5.1. Prueba 1: Modificación de estado de un usuario

Objetivo: Verificar que el administrador puede editar el estado de un usuario existente y que los cambios se guardan correctamente.

Procedimiento:

1. Acceder al apartado **"Admin Dashboard"** desde el menú lateral.
2. Desde el listado de usuarios, pulsa el botón **.ªCCESS"** del usuario deseado.
3. Pulsar en el selector de estado.
4. Modifica el estado a `Blocked`.

Resultado esperado: El sistema actualiza la información del usuario y muestra un mensaje de confirmación de éxito.

Resultado obtenido: Conforme al esperado.

Evidencia:

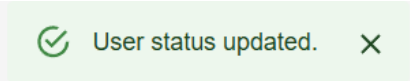


Figura 8.31: Mensaje de éxito en la modificación de estado

8.2.5.2. Prueba 2: Modificación de rol de un usuario

Objetivo: Verificar que el administrador puede editar el rol de un usuario existente y que los cambios se guardan correctamente.

Procedimiento:

1. Acceder al apartado **"Admin Dashboard"** desde el menú lateral.
2. Desde el listado de usuarios, pulsa el botón **"ACCESS"** del usuario deseado.
3. Pulsar en el selector de rol.
4. Modifica el rol a `admin`.

Resultado esperado: El sistema actualiza la información del usuario y muestra un mensaje de confirmación de éxito.

Resultado obtenido: Conforme al esperado. El cambio se refleja correctamente en el listado.

Evidencia:

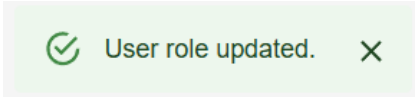


Figura 8.32: Mensaje de éxito en la modificación de rol

8.3. Pruebas de sistema

En esta actividad se realizan las pruebas de sistema de modo que se asegure que el sistema cumple con todos los requisitos establecidos: funcionales y no funcionales.

8.3.1. Pruebas funcionales

Con estas pruebas se analiza el buen funcionamiento de la implementación de los flujos normales y alternativos de los distintos casos de uso del sistema.

8.3.1.1. Prueba del RF-01

Objetivo: La plataforma debe permitir que un usuario se registre proporcionando su dirección de correo eléctrónico, una contraseña segura, su nombre, su primer apellido. Además, el usuario puede introducir opcionalmente un segundo apellido, a qué organización pertenece (si es que pertenece a alguna) y marcar si es un estudiante o no. El usuario registrado a partir de la plataforma se le asignará el rol de `user`.

Resultado obtenido: El requisito se valida mediante la prueba unitaria descrita en la sección 8.2.2.1, cuya ejecución satisfactoria confirma su correcto cumplimiento.

8.3.1.2. Prueba del RF-02

Objetivo: La plataforma debe ofrecer la opción de iniciar sesión a los usuarios previamente registrados, usando su correo electrónico y contraseña. El usuario dispondrá de un botón para visualizar la contraseña ocultada si es que lo desea.

Resultado obtenido: El requisito se valida mediante la prueba unitaria descrita en la sección 8.2.1.1, cuya ejecución satisfactoria confirma su correcto cumplimiento.

8.3.1.3. Prueba del RF-03

Objetivo: La plataforma debe adaptar el menú principal al rol del usuario autenticado, mostrando únicamente las opciones permitidas. Los usuarios estándar disponen de las funciones básicas, mientras que los roles administrativos incluyen acceso adicional al panel de administración.

Resultado obtenido: El requisito se valida mediante las pruebas unitarias descritas en las secciones 8.2.4.1, 8.2.5.1 y 8.2.5.2, cuya acceso único a los roles `admin` y `superuser` y ejecución satisfactoria confirma su correcto cumplimiento.

8.3.1.4. Prueba del RF-04

Objetivo: El sistema debe permitir a los usuarios con rol `admin` o `superuser` consultar el listado completo de usuarios registrados. La información mostrada debe incluir, al menos, los datos identificativos y el estado de la cuenta de cada usuario, de manera clara y estructurada para facilitar su revisión.

Resultado obtenido: El sistema permite a los usuarios con rol `admin` y `superuser` acceder correctamente al listado de usuarios registrados. La información se presenta de forma clara y estructurada, mostrando los datos identificativos de cada usuario junto con el estado de su cuenta. Asimismo, se verifica que los usuarios con rol `user` no pueden acceder a esta funcionalidad, confirmando que el control de permisos se aplica correctamente.

8.3.1.5. Prueba del RF-05

Objetivo: La plataforma debe permitir la visualización del historial de sesiones de un usuario específico. En este historial se mostrarán la dirección IP del dispositivo desde el cual se accedió a la plataforma, así como la fecha y hora de inicio y cierre de sesión. Esta funcionalidad será accesible únicamente para usuarios con rol "admin" o "superuser".

Resultado obtenido: La plataforma permite a los usuarios con rol `admin` y `superuser` acceder correctamente al historial de sesiones de un usuario seleccionado. La información mostrada incluye la dirección IP utilizada en cada acceso, así como la fecha y hora de inicio y cierre de sesión. Los usuarios que no disponen de los permisos necesarios no pueden visualizar esta información, cumpliéndose así las restricciones de acceso definidas.

8.3.1.6. Prueba del RF-06

Objetivo: La plataforma debe ofrecer la funcionalidad de cambiar el estado de la cuenta de un usuario. Los estados posibles son `Activo` y `Bloqueado`.

Si la cuenta está en estado `Activo`, el usuario podrá acceder a la plataforma y utilizar sus servicios. En cambio, si la cuenta se encuentra en estado `Bloqueado`, el usuario no podrá iniciar sesión ni hacer uso de la plataforma. Esta funcionalidad está destinada a gestionar el acceso de usuarios que hagan un uso inadecuado del sistema.

Solo los usuarios con rol `admin` o `superuser` tendrán acceso a esta funcionalidad.

Resultado obtenido: El requisito se valida mediante la prueba unitaria descrita en la sección 8.2.5.1, cuya ejecución satisfactoria confirma su correcto cumplimiento.

8.3.1.7. Prueba del RF-07

Objetivo: La plataforma debe ofrecer la funcionalidad de modificar el rol asignado a un usuario. Los roles disponibles son: `user`, `admin` y `superuser`.

Esta funcionalidad será accesible únicamente para usuarios con rol `superuser`.

Resultado obtenido: El requisito se valida mediante la prueba unitaria descrita en la sección 8.2.5.2, cuya ejecución satisfactoria confirma su correcto cumplimiento.

8.3.1.8. Prueba del RF-08

Objetivo: La plataforma permite analizar proyectos de GitHub mediante enlaces individuales o listas en formato CSV. El análisis se realiza sobre la rama principal y se identifica por el nombre del proyecto y su commit, evitando reprocesar versiones ya analizadas o con errores previos. Durante el proceso se generan mutantes y se obtienen métricas mediante herramientas externas, almacenando los resultados para su uso posterior en la plataforma.

Resultado obtenido: El requisito se valida mediante las pruebas unitarias descritas en las secciones 8.2.3.1 y 8.2.3.4, cuya ejecución satisfactoria confirma su correcto cumplimiento.

8.3.1.9. Prueba del RF-09

Objetivo: La plataforma debe ofrecer la opción de eliminar, del historial de proyectos subidos por el propio usuario, aquellos proyectos que este considere necesario retirar. Esta acción únicamente elimina la asociación entre el usuario y el proyecto analizado, pero no borra las métricas ni los resultados almacenados en la base de datos, ya que estos pueden seguir siendo utilizados por otras funcionalidades del sistema, como la evaluación de modelos predictivos o mostrar las métricas sin tener que analizar de nuevo el proyecto si otro usuario sube el mismo proyecto.

Resultado obtenido: El requisito se valida mediante la prueba unitaria descrita en la sección 8.2.3.7, cuya ejecución satisfactoria confirma su correcto cumplimiento.

8.3.1.10. Prueba del RF-10

Objetivo: La plataforma debe ofrecer la opción de visualizar las métricas generadas durante el proceso de análisis. El usuario podrá seleccionar uno o varios proyectos previamente analizados y consultar sus métricas en una interfaz que permita elegir y cambiar entre los distintos proyectos.

Resultado obtenido: El requisito se valida mediante la prueba unitaria descrita en la sección 8.2.3.6, cuya ejecución satisfactoria confirma su correcto cumplimiento.

8.3.1.11. Prueba del RF-11

Objetivo: La plataforma debe ofrecer la opción de descargar en formato CSV las métricas generadas durante el proceso de análisis. El usuario podrá seleccionar uno o varios proyectos previamente analizados. Si se selecciona un solo proyecto, se descargará un archivo CSV con sus métricas. Si se seleccionan varios, se generará un archivo ZIP que contendrá un CSV individual por cada proyecto seleccionado. Esto facilita el almacenamiento y el análisis externo de los resultados.

Resultado obtenido: El requisito se valida mediante la prueba unitaria descrita en la sección 8.2.3.5, cuya ejecución satisfactoria confirma su correcto cumplimiento.

8.3.1.12. Prueba del RF-12

Objetivo: La plataforma permite la carga de modelos predictivos previamente entrenados en formatos compatibles para la evaluación de mutantes. El sistema valida el tipo y la extensión del archivo antes de almacenarlo, y esta funcionalidad está restringida a usuarios con roles administrativos.

Resultado obtenido: El requisito se valida mediante la prueba unitaria descrita en la sección 8.2.4.1, cuya ejecución satisfactoria confirma su correcto cumplimiento.

8.3.1.13. Prueba del RF-13

Objetivo: La plataforma permite evaluar modelos predictivos aplicándolos sobre las métricas de proyectos ya analizados para predecir el estado de los mutantes. El sistema realiza únicamente la inferencia, sin procesos de entrenamiento, y muestra métricas de rendimiento para cada modelo seleccionado, facilitando su comparación.

Resultado obtenido: El requisito se valida mediante las pruebas unitarias descritas en las secciones 8.2.4.3 y 8.2.4.4, cuya ejecución satisfactoria confirma su correcto cumplimiento.

8.3.1.14. Prueba del RF-14

Objetivo: La plataforma debe ofrecer una serie de filtros y métodos de ordenación que permitan al usuario visualizar los datos de manera sencilla e intuitiva.

Resultado obtenido: La plataforma ofrece correctamente funcionalidades de filtrado, ordenación y selección de columnas sobre los datos mostrados, integradas en los componentes de visualización utilizados. El usuario puede ordenar la información por distintas columnas, aplicar filtros para acotar los resultados visibles y seleccionar qué columnas desea mostrar en cada momento. Estas opciones permiten adaptar la visualización de los datos a las necesidades del análisis sin afectar a la información almacenada, proporcionando una experiencia de uso flexible e intuitiva.

8.3.1.15. Prueba del RF-15

Objetivo: La plataforma debe permitir al usuario seleccionar la paleta de colores que considere visualmente más atractiva o accesible. Las opciones disponibles incluirán los temas: `Claro`, `Oscuro` y `Azul`.

Resultado obtenido: La plataforma permite al usuario seleccionar correctamente entre las paletas de colores disponibles: `Claro`, `Oscuro` y `Azul`. Al realizar la selección, la apariencia visual del sistema se actualiza de forma inmediata, aplicando los colores correspondientes a los distintos elementos de la interfaz. La preferencia de tema se mantiene durante la navegación por la plataforma, ofreciendo una experiencia visual personalizable y accesible.

8.3.2. Pruebas no funcionales

Estas pruebas pretenden comprobar el funcionamiento del sistema, con respecto a los requisitos no funcionales identificados: eficiencia, seguridad, etc.

8.3.2.1. Prueba del RNF-01

Objetivo: La plataforma permite filtrar, ordenar y personalizar las tablas de datos para facilitar el análisis de la información. Los usuarios pueden aplicar filtros por texto, valores numéricos y categorías, ordenar por cualquier columna y mostrar u ocultar columnas, con actualización dinámica de los resultados.

Resultado obtenido: Las tablas de la plataforma permiten aplicar filtros por texto y valores numéricos, ordenar los registros por cualquiera de las columnas disponibles y personalizar la visualización mostrando u ocultando columnas. Las modificaciones se reflejan de forma inmediata sin recargar la página, facilitando el análisis de la información.

8.3.2.2. Prueba del RNF-02

Objetivo: La plataforma debe cumplir con los principios de accesibilidad definidos en las Pautas de Accesibilidad al Contenido en la Web (WCAG 2.1), incluyendo soporte para navegación mediante teclado, contraste adecuado entre texto y fondo, y soporte para lectores de pantalla. También debe ofrecer opciones de personalización como el cambio de paleta de colores.

Resultado obtenido: El requisito se valida mediante las pruebas unitarias descritas en las secciones 8.1.1, 8.1.2 y 8.1.3, cuya ejecución satisfactoria confirma su correcto cumplimiento.

8.3.2.3. Prueba del RNF-03

Objetivo: La plataforma debe ser capaz de mantener su funcionamiento continuo durante un 99.5 % del tiempo mensual (disponibilidad), excluyendo mantenimientos programados. Además, ante errores en el análisis de proyectos o evaluación de modelos, el sistema debe registrar los fallos y notificar adecuadamente al usuario sin comprometer la estabilidad general.

Resultado obtenido: Durante las pruebas realizadas, la plataforma ha mantenido un funcionamiento estable, incluso cuando se producen errores en el análisis de proyectos o en la evaluación de modelos. En estos casos, los fallos se registran mediante logs internos y el usuario recibe notificaciones informativas sin que el sistema deje de estar operativo.

8.3.2.4. Prueba del RNF-04

Objetivo: La plataforma debe mantener un comportamiento estable y tiempos de respuesta aceptables en las principales pantallas del sistema, tanto en estado inactivo como durante la ejecución de procesos intensivos, como el análisis de proyectos.

Escenarios evaluados:

Se realizaron auditorías Lighthouse en los siguientes escenarios representativos:

- **Panel de administración (Admin Dashboard)** sin procesos activos.
- **Pantalla de proyectos** sin análisis en ejecución.
- **Pantalla de proyectos** mientras un proyecto está siendo analizado.
- **Pantalla de modelos** sin procesos de evaluación en ejecución.

Procedimiento

1. Acceder a PMTlab desde Google Chrome e iniciar sesión.
2. Navegar al **Admin Dashboard** y ejecutar una auditoría Lighthouse.
3. Acceder a la sección **Proyectos** sin iniciar ningún análisis y ejecutar una auditoría Lighthouse.
4. Subir un proyecto para iniciar su análisis.
5. Mientras el sistema se encuentra en estado de análisis, ejecutar una auditoría Lighthouse en la pantalla de **Proyectos**.
6. Acceder a la sección **Modelos** y ejecutar una auditoría Lighthouse.
7. Registrar los resultados obtenidos en cada escenario.

Resultado esperado:

El sistema debe seguir siendo navegable y funcional en todas las pantallas evaluadas, incluso cuando se está ejecutando el análisis de proyectos en segundo plano, sin bloqueos ni degradaciones severas de rendimiento.

Resultado obtenido:

Las auditorías realizadas muestran que el rendimiento del frontend se mantiene estable en las distintas pantallas evaluadas. En la pantalla de proyectos durante el análisis se observa un aumento moderado en los tiempos de carga, pero sin afectar a la interactividad ni a la capacidad de consultar el estado del proceso.

El Admin Dashboard, la pantalla de proyectos sin análisis y la pantalla de modelos presentan valores de rendimiento consistentes, lo que confirma que el sistema está correctamente desacoplado entre la ejecución de procesos de backend y la experiencia de usuario del frontend.

Evidencias:

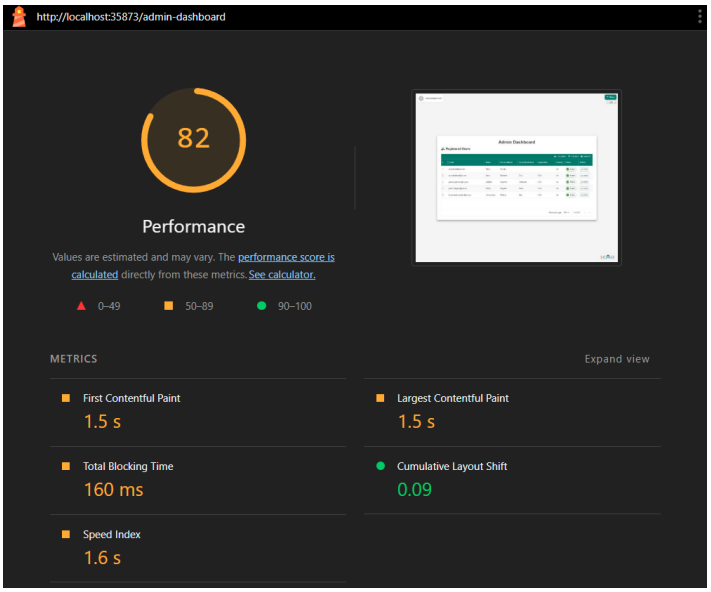


Figura 8.33: Rendimiento en Admin Dashboard

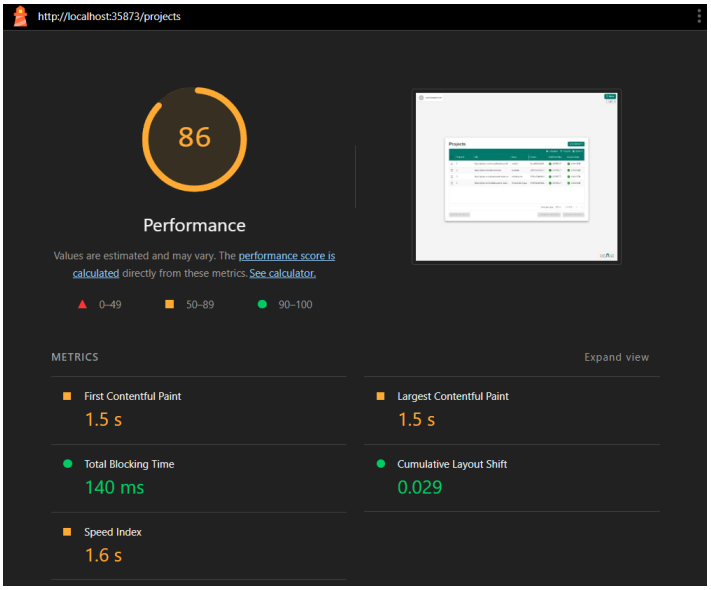


Figura 8.34: Rendimiento en pantalla de proyectos sin análisis en ejecución

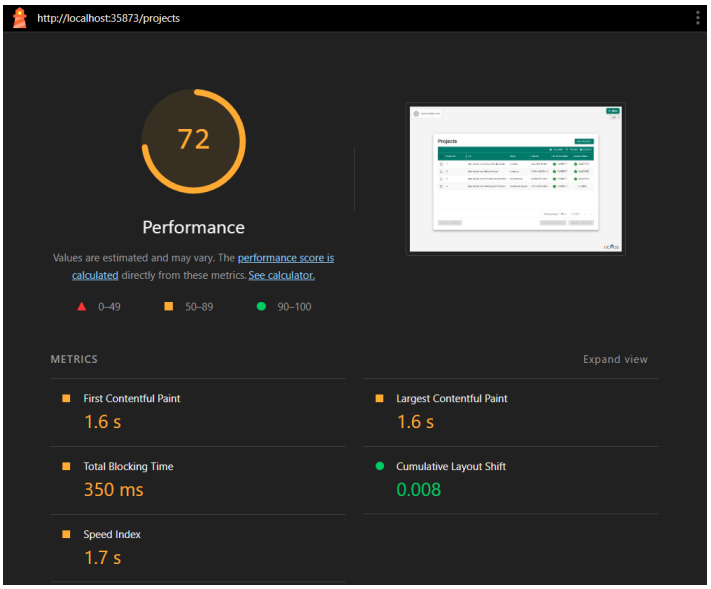


Figura 8.35: Rendimiento en pantalla de proyectos mientras un proyecto está siendo analizado

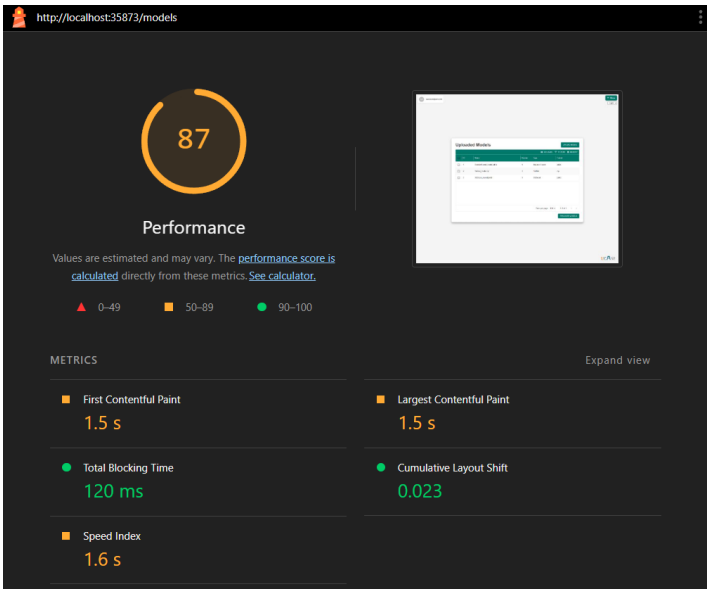


Figura 8.36: Rendimiento en pantalla de modelos

8.3.2.5. Prueba del RNF-05

Objetivo: La plataforma debe proteger los datos personales de los usuarios mediante encriptación de contraseñas (por ejemplo, usando bcrypt) y el uso de HTTPS para todas las comunicaciones. La autenticación se gestionará mediante tokens JWT (JSON Web Tokens), que serán firmados y tendrán una expiración configurable (por defecto: 60 minutos). Tras ese tiempo, el token expirará y el usuario deberá autenticarse nuevamente.

Además, el acceso a funcionalidades sensibles estará restringido exclusivamente a usuarios con los roles adecuados, siguiendo el principio de mínimo privilegio para garantizar una mayor protección del sistema.

Resultado obtenido: Las contraseñas de los usuarios se almacenan cifradas mediante algoritmos de hashing seguro y la autenticación se gestiona mediante tokens JWT con expiración configurada. El acceso a las funcionalidades del sistema se encuentra restringido según el rol del usuario, garantizando el principio de mínimo privilegio.

8.3.2.6. Prueba del RNF-06

Objetivo: El sistema debe estar desarrollado siguiendo principios de arquitectura modular y buenas prácticas de codificación (por ejemplo, separación de responsabilidades y uso de control de versiones). Esto facilitará la corrección de errores y la incorporación de nuevas funcionalidades en el futuro.

Resultado obtenido: El sistema presenta una arquitectura modular claramente definida, con separación entre frontend, backend y base de datos, lo que facilita su mantenimiento y evolución. El uso de control de versiones y buenas prácticas de codificación permite incorporar mejoras y corregir errores sin afectar al resto del sistema.

8.3.2.7. Prueba del RNF-07

Objetivo: La plataforma debe ser accesible desde los principales navegadores web modernos (Chrome, Firefox, Edge, Opera) y debe funcionar correctamente en dispositivos de escritorio, portátiles y tablets con resolución mínima de 768px de ancho. La adaptación a dispositivos móviles de 480px será progresiva y no prioritaria en la primera versión.

Resultado obtenido: La plataforma ha sido probada en los principales navegadores web modernos y funciona correctamente en dispositivos de escritorio y portátiles con resoluciones iguales o superiores a 768px de ancho. La interfaz se adapta adecuadamente al tamaño de pantalla, cumpliendo con los requisitos de compatibilidad definidos, como se puede ver en la prueba 8.1.3.

8.3.2.8. Prueba del RNF-08

Objetivo: El sistema debe diseñarse de forma que pueda ampliarse para soportar un mayor número de usuarios concurrentes, nuevos tipos de análisis, o modelos predictivos adicionales sin necesidad de una reestructuración significativa del código base.

Resultado obtenido: La arquitectura del sistema permite ampliar el número de usuarios, incorporar nuevos modelos predictivos y extender los tipos de análisis sin necesidad de una reestructuración significativa del código base. Este diseño facilita la evolución futura de la plataforma.

8.3.2.9. Prueba del RNF-09

Objetivo: La plataforma debe estar disponible en idioma inglés, con el objetivo de garantizar una mayor accesibilidad e inclusión a nivel internacional. Todos los textos de la interfaz, mensajes del sistema y etiquetas deben estar en inglés.

Resultado obtenido: La totalidad de la interfaz de usuario, incluidos los textos, mensajes del sistema y etiquetas, se presenta en idioma inglés.

9. Despliegue del sistema

Este capítulo recoge la arquitectura física planteada para el sistema, las instrucciones para su despliegue y las instrucciones para la operación y mantenimiento del nivel de servicio.

9.1. Arquitectura física

La arquitectura física del sistema se basa en un entorno virtualizado mediante contenedores, lo que permite desplegar y ejecutar los distintos componentes de manera independiente, aislada y fácilmente reproducible. En lugar de utilizar hardware físico dedicado, se emplea un servidor de propósito general capaz de ejecutar los servicios necesarios a través de *Docker* y *docker-compose*, lo que aporta portabilidad y escalabilidad.

9.1.1. Requisitos físicos

La infraestructura física necesaria para el despliegue del sistema no requiere de un gran número de servidores dedicados, ya que se trata de una plataforma orientada a investigación y desarrollo. Aun así, es fundamental garantizar que el hardware empleado sea capaz de soportar la ejecución concurrente de los servicios (frontend, backend y base de datos), así como los procesos de análisis y evaluación de proyectos que pueden resultar computacionalmente costosos.

El sistema puede ejecutarse en un servidor físico o, de forma más flexible, en una máquina virtual dentro de una infraestructura en la nube o local. Los requisitos mínimos recomendados son:

- **Procesador (CPU)**: Mínimo 8 núcleos físicos a 3.0 GHz o superior.
- **Memoria RAM**: Mínimo 16 GB, recomendado 32 GB. Esto permite cargar y evaluar modelos de machine learning de gran tamaño sin comprometer el rendimiento de la base de datos ni del backend.
- **Almacenamiento**: 1 TB en disco SSD para asegurar rapidez en la lectura/escritura de datos. El almacenamiento debe cubrir:
  - Los volúmenes persistentes de PostgreSQL.
  - Los proyectos subidos por los usuarios.
  - Los modelos de aprendizaje automático y sus resultados.
- **Tarjeta gráfica (GPU)**: no es estrictamente necesaria, pero se recomienda disponer de una GPU con al menos 8 GB de VRAM para acelerar la evaluación de modelos en escenarios más intensivos.

- **Conectividad de red**: al menos 1 Gbps de ancho de banda, con baja latencia, para garantizar un acceso fluido de los usuarios a la plataforma y una correcta comunicación entre los contenedores de Docker.

9.1.2. Requisitos software

El sistema se apoya en un conjunto de elementos software que permiten garantizar la portabilidad, escalabilidad y facilidad de mantenimiento de la plataforma. Cada componente cumple un rol específico dentro de la arquitectura:

- **Sistema operativo**: Linux (Ubuntu 22.04 LTS recomendado). Este sistema operativo ofrece estabilidad, compatibilidad con contenedores Docker y gran soporte en entornos de servidores.
- **Entorno de contenedores**: Docker y Docker Compose. Se emplean para aislar los servicios (frontend, backend, base de datos) en contenedores independientes, simplificando el despliegue y la replicación del sistema en diferentes entornos.
- **Servidor de base de datos**: PostgreSQL (última versión estable). Se encarga de gestionar las tablas de usuarios, proyectos, métricas de análisis y modelos predictivos. Su motor relacional garantiza integridad y consistencia en el almacenamiento.
- **Lenguajes de programación y runtimes**:
  - **Python 3.11**: utilizado en el backend para la lógica de negocio, análisis de proyectos y ejecución de modelos de machine learning.
  - **Node.js 20**: necesario para el frontend desarrollado con React y para la gestión de dependencias del lado cliente.
  - **OpenJDK 8**: requerido para ejecutar herramientas de análisis estático y cobertura de código en proyectos Java.
- **Frameworks y librerías principales**:
  - **React**: desarrollo de la interfaz gráfica de usuario.
  - **Express (Node.js)**: soporte a la comunicación API entre frontend y backend.
  - **Scikit-learn, XGBoost, PyTorch TabNet**: librerías empleadas para el preprocesamiento de datos y evaluación de modelos predictivos.
  - **lxml, pandas, numpy**: librerías de soporte para análisis estático, manipulación de datos y parsing de ficheros.
- **Gestión de versiones**: Git, con repositorios en GitHub. Permite la colaboración, control de cambios y despliegue continuo.

- **Otras herramientas**:
  - **Dockerfile + init.sql**: scripts de configuración y arranque inicial del sistema, que automatizan la creación de tablas y la inserción de datos iniciales en PostgreSQL.
  - **npm y pip**: gestores de paquetes para instalar dependencias tanto en frontend como en backend.

9.1.3. Modelo de despliegue

El modelo de despliegue de la plataforma se basa en la utilización de contenedores Docker:

- **Base de datos (PostgreSQL)**: Se ejecuta en el contenedor `pmt_database`. Este contenedor utiliza la imagen oficial de PostgreSQL y se encarga de almacenar los datos de usuarios, proyectos, métricas de análisis y modelos predictivos. Los datos persisten mediante un volumen externo (`backend_postgres_data`), mientras que el archivo `init.sql` se monta para inicializar las tablas y datos iniciales al arrancar el contenedor.
- **Servidor backend**: Ejecutado en el contenedor `pmt_server`, se construye a partir de un `Dockerfile` ubicado en el directorio `./backend`. Este contenedor gestiona la lógica de negocio, la API REST para la comunicación con el frontend, el análisis y evaluación de proyectos, y el preprocesamiento de datos para los modelos predictivos. Se montan varios volúmenes para persistir archivos de entrada, salida, modelos y utilidades.
- **Frontend**: El contenedor `pmt_frontend` ejecuta la interfaz web desarrollada en React. También se construye a partir de un `Dockerfile` en `./frontend` y se vincula al backend mediante la red interna. Los volúmenes montados permiten sincronizar el código fuente y las dependencias de Node.js.

Todos los contenedores están conectados a una red interna llamada `mynetwork`, que permite la comunicación entre ellos sin exponer puertos adicionales a la red externa, salvo los necesarios para acceso desde el host (p. ej., 13000 para el backend, 5173 para el frontend).

9.2. Instrucciones de despliegue

A continuación, se presentan las indicaciones para instalar el sistema sobre la infraestructura física mencionada anteriormente.

9.2.1. Procedimientos de instalación

Antes de proceder con el despliegue, es necesario preparar el entorno donde se ejecutará el sistema. A continuación, se indican los pasos necesarios para contar con

un entorno funcional:

1. **Actualizar el sistema operativo**. En caso de utilizar Linux (Ubuntu 22.04 LTS recomendado), ejecutar:

   ```
   sudo apt update && sudo apt upgrade -y
   ```

2. **Instalar Git**. Es necesario para clonar el repositorio del proyecto:

   ```
   sudo apt install git -y
   ```

3. **Instalar Docker**. El sistema utiliza contenedores para aislar los diferentes servicios. Instrucciones básicas de instalación en Ubuntu:

   ```
   sudo apt install ca-certificates curl gnupg lsb-release -y
   curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
   ↪  --dearmor -o /usr/share/keyrings/docker.gpg
   echo "deb [arch=$(dpkg --print-architecture)
   ↪  signed-by=/usr/share/keyrings/docker.gpg]
   ↪  https://download.docker.com/linux/ubuntu $(lsb_release -cs)
   ↪  stable" | sudo tee /etc/apt/sources.list.d/docker.list >
   ↪  /dev/null
   sudo apt update
   sudo apt install docker-ce docker-ce-cli containerd.io -y
   ```

4. **Instalar Docker Compose**. Se necesita para orquestar los diferentes contenedores.

   ```
   sudo apt install docker-compose -y
   ```

5. **Comprobar la instalación**. Verificar que tanto Docker como Docker Compose están correctamente instalados:

   ```
   docker --version
   docker-compose --version
   ```

6. **Permitir ejecución de Docker sin sudo** (opcional pero recomendado).

   ```
   sudo groupadd docker
   sudo usermod -aG docker $USER
   newgrp docker
   ```

9.2.1.1. Acceso al repositorio del sistema

El código fuente del sistema se encuentra almacenado en un repositorio privado https://github.com/MarioRosano14/TFG. Para poder acceder a él, es necesario solicitar permiso de acceso, ya que no está disponible de forma pública. Una vez concedido el acceso, el repositorio puede clonarse utilizando la herramienta `Git` previamente instalada.

```
git clone https://github.com/MarioRosano14/TFG
```

De esta manera se obtiene en local toda la estructura del proyecto, incluyendo el `docker-compose.yml`, los directorios de `backend` y `frontend`, así como los scripts necesarios para inicializar la base de datos y ejecutar la plataforma.

9.2.2. Procedimientos de despliegue

Una vez cumplidos los requisitos previos y obtenido el código fuente del repositorio, el sistema puede desplegarse siguiendo los pasos que se detallan a continuación.

1. **Acceder al directorio principal del proyecto**. Desde la raíz del repositorio clonado, situarse en la carpeta `web/`, que contiene los archivos de configuración y el fichero `docker-compose.yml`.

   ```
   cd web/
   ```

2. **Configurar el entorno de despliegue**. Antes de iniciar los contentedores, es necesario definir los parámetros de implantación mediante el fichero `.env` ubicado en el directorio `web/`. Este archivo permite establecer los puertos y hosts deseados para cada servicio según el entorno (desarrollo o producción).

   Un ejemplo de configuración es el siguiente:

   ```
   # ============================
   # CONFIGURACIÓN DE PUERTOS
   # ============================
   PMTLAB_PG_PORT=25433
   PMTLAB_BACKEND_PORT=18080
   PMTLAB_FRONTEND_PORT=35873
   PMTLAB_PGADMIN_PORT=49151

   # ============================
   # CONFIGURACIÓN DEL HOST
   # ============================
   # Desarrollo: localhost; Producción: 10.142.14.161
   PMTLAB_FRONTEND_HOST="localhost"

   # Desarrollo: localhost; Producción: 10.142.14.161
   PMTLAB_BACKEND_HOST="localhost"
   ```

3. **Construir e iniciar los contenedores**. Ejecutar el comando de Docker Compose para levantar todos los servicios en segundo plano. La opción `--build` garantiza que las imágenes se reconstruyan, incorporando posibles cambios en el código o en los ficheros de configuración.

   ```
   docker-compose up -d --build
   ```

   Tras completar estos pasos, el sistema queda desplegado y accesible.

9.2.3. Pruebas de implantación

Una vez finalizado el procedimiento de despliegue y levantados todos los contenedores mediante `docker-compose`, es necesario realizar una serie de pruebas básicas de implantación para verificar que el sistema se encuentra correctamente instalado y que los servicios están accesibles.

1. **Acceso al frontend**. Tras unos minutos, dependiendo del tiempo requerido para la instalación de dependencias en cada contenedor, el frontend de la plataforma puede consultarse en la dirección definida en el fichero `.env`.

   Por ejemplo, con la configuración mostrada anteriormente:

   `http://localhost:35873/`

   Una carga correcta de la página indica que el servidor de frontend y la comunicación con el backend están activos.

2. **Prueba de autenticación**. Se ha configurado un usuario *superusuario* que permite validar el correcto funcionamiento del sistema de autenticación y la conexión con la base de datos.

   - **Email**: `superuser@pmt.com`
   - **Contraseña**: `root`

   Si el inicio de sesión se realiza con éxito, se confirma que el flujo de autenticación funciona y que la tabla de usuarios de PostgreSQL está operativa.

3. **Carga de proyectos de prueba**. Para comprobar que el backend gestiona adecuadamente la subida y análisis de proyectos, es recomendable utilizar los modelos de ejemplo incluidos en el directorio:

   `Investigacion/`

   Estos modelos permiten verificar el proceso de carga de ficheros, almacenamiento en el contenedor correspondiente y posterior evaluación.

La superación satisfactoria de estas pruebas confirma que el sistema se encuentra implantado y preparado para su utilización.

9.3. Instrucciones para la operación del sistema y mantenimiento del nivel de servicio

Con el objetivo de asegurar la continuidad del servicio y prevenir posibles incidencias, se definen una serie de procedimientos de operación y mantenimiento. Estos procedimientos abarcan la disponibilidad del sistema, la gestión de la información almacenada, la monitorización de logs y el mantenimiento preventivo.

9.3.1. Backups y recuperación de datos

Dado que el sistema depende de la base de datos PostgreSQL, es importante implementar copias de seguridad periódicas.

- **Frecuencia recomendada**: diaria para entornos de producción, semanal para entornos de pruebas.
- **Método**: mediante el comando `pg_dump` desde el contenedor de la base de datos.

```
docker exec -t pmt_database pg_dump -U root db_test > backup.sql
```

- **Recuperación**: en caso de fallo, el backup puede restaurarse con:

```
docker exec -i pmt_database psql -U root db_test < backup.sql
```

9.3.2. Chequeo de logs

El seguimiento de logs permite detectar errores de ejecución, problemas de rendimiento o intentos de acceso no autorizados.

- **Logs de contenedores**: pueden consultarse con `docker logs <nombre-contenedor>`.
- **Registro en backend**: los scripts generan mensajes de error y advertencias que ayudan a localizar problemas en el análisis de proyectos o en la evaluación de modelos.

9.3.3. Mantenimiento preventivo

Para garantizar la estabilidad a largo plazo del sistema, se recomienda:

- **Actualización de imágenes Docker**: ejecutar periódicamente `docker-compose build` para regenerar las imágenes con las versiones más recientes y seguras de los componentes.
- **Gestión de dependencias**: revisar las versiones de librerías utilizadas en backend (Python, librerías de ML) y frontend (Node.js, React) para prevenir vulnerabilidades.
- **Control de espacio en disco**: vigilar el crecimiento de las carpetas de `uploads` y `outputMetrics`, eliminando proyectos obsoletos o trasladando resultados antiguos a almacenamiento externo.
- **Verificación periódica de credenciales**: comprobar que las contraseñas de acceso (por ejemplo, de PostgreSQL o del superusuario de la plataforma) son seguras y rotarlas periódicamente.

9.3.4. Disponibilidad y rendimiento

- **Disponibilidad**: el sistema puede reiniciarse de forma controlada con `docker-compose restart`, sin pérdida de datos gracias al uso de volúmenes persistentes en la base de datos.
- **Escalabilidad**: en caso de mayor carga de trabajo, es posible escalar el número de instancias del backend modificando el `docker-compose.yml`.
- **Pruebas de rendimiento**: se recomienda ejecutar periódicamente cargas de prueba para verificar que los tiempos de respuesta de la API y del frontend siguen siendo aceptables.

Estos procedimientos permiten mantener un nivel de servicio adecuado, garantizando la seguridad, disponibilidad y rendimiento del sistema a lo largo del tiempo.

Parte III

Epílogo

10. Conclusiones

En este último capítulo se detallan las lecciones aprendidas tras el desarrollo del presente proyecto y se identifican las posibles oportunidades de mejora sobre el software desarrollado.

10.1. Objetivos alcanzados

El objetivo general de este Trabajo de Fin de Grado ha sido el diseño y desarrollo de una plataforma web integral orientada al *Predictive Mutation Testing*, capaz de servir como entorno práctico para el análisis de proyectos software y la evaluación de la calidad de sus pruebas mediante técnicas predictivas. Este objetivo se ha materializado con la construcción de un sistema completo que integra un backend encargado del procesamiento y análisis, un frontend web para la interacción con los usuarios y una base de datos que almacena de forma estructurada toda la información generada durante los análisis y evaluaciones.

A lo largo del desarrollo del proyecto se ha llevado a cabo un trabajo significativo en la vertiente investigadora, centrado en el diseño, entrenamiento y evaluación de distintos modelos de aprendizaje automático aplicados al contexto del PMT. En concreto, se han entrenado y comparado modelos como *RandomForest*, *XGBoost* y *TabNet*, analizando su comportamiento sobre métricas extraídas de mutantes y validando su capacidad para predecir la relevancia de estos sin necesidad de ejecutar la totalidad de las pruebas.

Desde el punto de vista del desarrollo software, se ha implementado una interfaz web basada en React que permite a los usuarios registrar proyectos, lanzar análisis automáticos, gestionar modelos predictivos y consultar los resultados de manera clara y organizada. La plataforma incorpora funcionalidades de filtrado, ordenación y selección de columnas, así como opciones de personalización visual, lo que facilita la exploración de grandes volúmenes de datos generados durante los análisis.

Asimismo, se ha diseñado el sistema para que sea fácilmente extensible y reutilizable, permitiendo la incorporación de nuevos modelos predictivos entrenados externamente y su evaluación sobre proyectos previamente analizados. Esta característica refuerza el carácter experimental de la plataforma y la convierte en una herramienta válida tanto para entornos profesionales y académicos como para estudios más avanzados en el ámbito del aseguramiento de la calidad del software.

Finalmente, se han realizado pruebas de implantación y validación del sistema, utilizando proyectos de ejemplo y escenarios reales de uso, que han permitido comprobar la estabilidad de la plataforma, la correcta integración de sus distintos componentes y la fiabilidad de los resultados obtenidos. En conjunto, el trabajo realizado ha permitido cumplir los objetivos planteados inicialmente, dando lugar a una plataforma

funcional que combina investigación, desarrollo e implementación práctica en el ámbito del *Predictive Mutation Testing*.

10.2. Lecciones aprendidas

El desarrollo de este Trabajo de Fin de Grado ha permitido adquirir una serie de conocimientos y buenas prácticas, tanto en el ámbito tecnológico como en la gestión de proyectos de software. Entre los aspectos más destacados se encuentran:

- **Planificación y metodología de trabajo**: La combinación de investigación, diseño, implementación y pruebas en ciclos iterativos permitió avanzar de manera flexible, adaptando el desarrollo a los hallazgos y dificultades encontradas durante el proyecto. Esta experiencia reforzó la importancia de documentar cada fase y priorizar tareas críticas para mantener un flujo de trabajo eficiente.
- **Gestión de entornos y despliegue**: La integración de contenedores Docker para backend, frontend y base de datos proporcionó un aprendizaje profundo sobre la virtualización de servicios, la configuración de volúmenes, redes y persistencia de datos. Se comprendió la relevancia de mantener un entorno reproducible para facilitar pruebas, mantenimiento y entrega a terceros.
- **Diseño e implementación de sistemas complejos**: El trabajo con modelos predictivos, bases de datos relacionales y el desarrollo de una interfaz web interactiva permitió consolidar competencias en programación Python, React, PostgreSQL y frameworks de machine learning como XGBoost y TabNet. Asimismo, se adquirió experiencia en la manipulación de grandes volúmenes de datos y la integración de pipelines de preprocesamiento y evaluación.
- **Control de calidad y pruebas**: La definición de pruebas de implantación y verificación de métricas ayudó a comprender la importancia de evaluar la efectividad del sistema, la correcta funcionalidad de cada módulo y la consistencia de los resultados obtenidos.

En conclusión, este proyecto permitió adquirir habilidades valiosas en planificación, desarrollo, despliegue y evaluación de sistemas de software complejos, que serán de gran utilidad en futuros proyectos académicos o profesionales.

10.3. Trabajo futuro

Durante el desarrollo de este proyecto se han identificado algunas cosas para mejorar el sistema en futuras versiones. Entre las principales se destacan:

- **Optimización de modelos predictivos**: Aunque los modelos implementados cumplen su función, existe un margen de mejora en términos de precisión, capacidad de generalización y eficiencia computacional. Futuras iteraciones podrían incluir la exploración de algoritmos más avanzados para incrementar la efectividad en la predicción de mutantes relevantes.

- **Ampliación del conjunto de métricas**: Incluir nuevas métricas de análisis estático o dinámico que permitan obtener un diagnóstico más completo de los proyectos analizados, aumentando el valor informativo que el sistema proporciona a los usuarios.
- **Mejora de la interfaz y experiencia de usuario**: Incorporar funcionalidades adicionales como visualizaciones interactivas de métricas, dashboards personalizables o sugerencias automáticas basadas en los resultados obtenidos.
- **Automatización de flujos y pipelines**: Incrementar la automatización del preprocesamiento de datos, evaluación de modelos y generación de informes para reducir la intervención manual y mejorar la reproducibilidad de los análisis.
- **Compatibilidad con más lenguajes de programación**: Actualmente el sistema se centra en proyectos desarrollados en Java. Una línea de trabajo futuro sería ampliar su alcance para soportar distintos lenguajes de programación (por ejemplo, Python, C#, C++ o JavaScript), lo cual aumentaría significativamente la aplicabilidad y el impacto de la herramienta.

Información sobre Licencia

El software desarrollado en el marco de este Trabajo de Fin de Grado está sujeto a un Acuerdo de Licencia de Usuario Final (EULA). Dicha licencia regula los términos y condiciones bajo los cuales se permite el uso y acceso tanto al sistema software como a la documentación asociada.

La licencia autoriza el uso del software conforme a los fines para los que ha sido desarrollado, quedando prohibida su redistribución total o parcial sin el consentimiento expreso del autor. Asimismo, el software no podrá ser integrado en otros sistemas o plataformas ni utilizado en contextos no autorizados sin dicha autorización.

De acuerdo con los términos de la EULA, los usuarios no podrán modificar, distribuir ni crear obras derivadas del software sin el consentimiento explícito del autor. Estas condiciones tienen como objetivo proteger el trabajo realizado y garantizar un uso adecuado del sistema.

Bibliografía

Balderas-Díaz, S., Guerrero-Contreras, G., Delgado-Pérez, P., Medina-Bulo, I., 2025. An Analysis of Class Imbalance Challenges in Predictive Mutation Testing, in: Intelligent Information and Database Systems, Springer Nature Singapore, Singapore. pp. 319–331. doi:doi:10.1007/978-981-96-6005-6_23.

CodeScene AB, 2025. CodeScene: Behavioral Code Analysis. https://codescene.com. Accedido: 19 de noviembre de 2025.

Docker Inc, 2025. Docker. https://www.docker.com/. Accedido: 19 de noviembre de 2025.

draw.io Ltd, 2025. Diagrams.net: Flowchart Maker & Online Diagram Software. https://www.diagrams.net/. Accedido: 19 de noviembre de 2025.

Figma Inc, 2025. Figma: The collaborative interface design tool. https://www.figma.com/. Accedido: 19 de noviembre de 2025.

GitHub Inc, 2025. GitHub. https://github.com/. Accedido: 19 de noviembre de 2025.

Google, 2026. Lighthouse: Automated Website Auditing Tool. https://developer.chrome.com/docs/lighthouse/. Accedido: 12 de enero de 2026.

Guerrero-Contreras, G., Balderas-Díaz, S., Delgado-Pérez, P., Medina-Bulo, I., 2025. Explainable Artificial Intelligence to Improve Interpretability in Predictive Mutation Testing, in: Intelligent Information and Database Systems, Springer Nature Singapore, Singapore. pp. 307–318. doi:doi:10.1007/978-981-96-6005-6_22.

JetBrains, 2025. Qodana: Code Quality Platform. https://www.jetbrains.com/qodana. Accedido: 19 de noviembre de 2025.

Jia, Y., Harman, M., 2011. An analysis and survey of the development of mutation testing. IEEE Transactions on Software Engineering 37, 649–678. doi:doi:10.1109/TSE.2010.62.

Jooble, 2025. Ingeniero de software junior salario en España. https://es.jooble.org/salary/ingeniero-de-software-junior. Cifra de 15,18 € por hora; Accedido: 19 de noviembre de 2025.

Meta Platforms, 2025. React – A JavaScript library for building user interfaces. https://react.dev/. Accedido: 19 de noviembre de 2025.

Microsoft, 2025. Visual Studio Code - Code Editing. Redefined. https://code.visualstudio.com/. Accedido: 19 de noviembre de 2025.

Overleaf, 2025. Overleaf: Online LaTeX Editor. https://www.overleaf.com/. Accedido: 19 de noviembre de 2025.

PostgreSQL Global Development Group, 2025. PostgreSQL. https://www.postgresql.org/. Accedido: 19 de noviembre de 2025.

Ramírez, S.T., 2025. FastAPI. https://fastapi.tiangolo.com/. Accedido: 19 de noviembre de 2025.

SonarSource, 2025. SonarQube. https://www.sonarsource.com/products/sonarqube. Accedido: 19 de noviembre de 2025.

Thiruvathukal, G.K., Shilpika, Hayward, N.J., Läufer, K., 2018. Metrics Dashboard: A Hosted Platform for Software Quality Metrics. arXiv preprint arXiv:1804.02053 URL: https://arxiv.org/abs/1804.02053, arXiv:1804.02053.

TotalEnergies, 2025. Precio Luz Hoy: 19/11/2025. https://www.totalenergies.es/es/hogares/precio-luz-hoy. Precio medio del día para el mercado regulado (PVPC) en España: 0,16779 €/kWh; Accedido: 19 de noviembre de 2025.

VoidZero Inc. and Vite Contributors, 2025. Vite — Next Generation Frontend Tooling. https://vitejs.dev/. Accedido: 19 de noviembre de 2025.

WebAIM, 2025. WAVE Web Accessibility Evaluation Tool. https://wave.webaim.org/. Accedido: 19 de noviembre de 2025.

Zhang, T., Zhang, N., Wu, Y., Li, C., Li, Y., Fu, B., Liu, X., 2023. PTMTorrent: A Dataset for Mining Open-source Pre-trained Model Packages. arXiv preprint arXiv:2303.08934 URL: https://arxiv.org/abs/2303.08934, arXiv:2303.08934.

Parte IV

Anexos

A. Manual del desarrollador

Este manual describe las pautas para modificar y ampliar el software. Está dirigido a desarrolladores que necesiten trabajar sobre el código fuente para añadir nuevas funcionalidades o ajustar las existentes.

A.1. Preparación del entorno de trabajo

Esta sección describe los requisitos necesarios para instalar y ejecutar la plataforma, así como el proceso de preparación del entorno de desarrollo. Aunque se recomienda utilizar Linux (preferiblemente Ubuntu 22.04 LTS), el desarrollador puede emplear cualquier sistema operativo compatible con Docker.

A.1.1. Requisitos físicos

Los requisitos físicos del sistema se describen en la sección 9.1.1.

A.1.2. Requisitos software

La plataforma se apoya en un conjunto reducido de herramientas necesarias para su despliegue. No es necesario instalar manualmente librerías, lenguajes ni dependencias internas, ya que todo se ejecuta dentro de contenedores.

- **Docker y Docker Compose**: elementos esenciales para la ejecución del frontend, backend y base de datos.
- **Git**: empleado para clonar el repositorio y gestionar versiones.
- **Sistema operativo**: recomendado Ubuntu 22.04 LTS, aunque cualquier sistema compatible con Docker es válido.

Para el desarrollo del software se sugiere utilizar **Visual Studio Code** como entorno de trabajo, por su facilidad de uso, integración con Git y amplia disponibilidad de extensiones útiles.

A.1.3. Control de versiones

El proyecto utiliza Git como sistema de control de versiones, con repositorios alojados en GitHub. Esto permite mantener un historial claro de cambios, gestionar ramas de desarrollo y facilitar la colaboración.

A.2. Instrucciones de despliegue

A continuación se describen los pasos necesarios para preparar un entorno local. Aunque se muestran comandos para Ubuntu, estos pueden adaptarse fácilmente a otros sistemas compatibles.

A.2.1. Procedimiento de instalación

1. **Actualizar el sistema operativo** (si se utiliza Linux):

   ```
   sudo apt update && sudo apt upgrade -y
   ```

2. **Instalar Git**:

   ```
   sudo apt install git -y
   ```

3. **Instalar Docker**:

   ```
   sudo apt install ca-certificates curl gnupg lsb-release -y
   curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
   ↪  --dearmor -o /usr/share/keyrings/docker.gpg
   echo "deb [arch=$(dpkg --print-architecture)
   ↪  signed-by=/usr/share/keyrings/docker.gpg]
   ↪  https://download.docker.com/linux/ubuntu $(lsb_release -cs)
   ↪  stable" | sudo tee /etc/apt/sources.list.d/docker.list >
   ↪  /dev/null
   sudo apt update
   sudo apt install docker-ce docker-ce-cli containerd.io -y
   ```

4. **Instalar Docker Compose**:

   ```
   sudo apt install docker-compose -y
   ```

5. **Verificar la instalación**:

   ```
   docker --version
   docker-compose --version
   ```

6. **(Opcional) Permitir el uso de Docker sin sudo**:

   ```
   sudo groupadd docker
   sudo usermod -aG docker $USER
   newgrp docker
   ```

Una vez completados estos pasos, el entorno queda preparado para clonar el repositorio del proyecto y ejecutar el sistema mediante los archivos de configuración proporcionados.

A.2.2. Acceso al repositorio del sistema

El código fuente del sistema se encuentra almacenado en un repositorio privado https://github.com/MarioRosano14/TFG. Para poder acceder a él, es necesario solicitar permiso de acceso, ya que no está disponible de forma pública. Una vez concedido el acceso, el repositorio puede clonarse utilizando la herramienta `Git` previamente instalada.

```
git clone https://github.com/MarioRosano14/TFG
```

De esta manera se obtiene en local toda la estructura del proyecto, incluyendo el `docker-compose.yml`, los directorios de `backend` y `frontend`, así como los scripts necesarios para inicializar la base de datos y ejecutar la plataforma.

A.2.3. Procedimientos de despliegue

Una vez cumplidos los requisitos previos y obtenido el código fuente del repositorio, el sistema puede desplegarse siguiendo los pasos que se detallan a continuación.

1. **Acceder al directorio principal del proyecto**. Desde la raíz del repositorio clonado, situarse en la carpeta `web/`, que contiene los archivos de configuración y el fichero `docker-compose.yml`.

   ```
   cd web/
   ```

2. **Configurar el entorno de despliegue**. Antes de iniciar los contentedores, es necesario definir los parámetros de implantación mediante el fichero `.env` ubicado en el directorio `web/`. Este archivo permite establecer los puertos y hosts deseados para cada servicio según el entorno (desarrollo o producción).

   Un ejemplo de configuración es el siguiente:

   ```
   # ============================
   # CONFIGURACIÓN DE PUERTOS
   # ============================
   PMTLAB_PG_PORT=25433
   PMTLAB_BACKEND_PORT=18080
   PMTLAB_FRONTEND_PORT=35873
   PMTLAB_PGADMIN_PORT=49151

   # ============================
   # CONFIGURACIÓN DEL HOST
   # ============================
   # Desarrollo: localhost; Producción: 10.142.14.161
   PMTLAB_FRONTEND_HOST="localhost"

   # Desarrollo: localhost; Producción: 10.142.14.161
   PMTLAB_BACKEND_HOST="localhost"
   ```

3. **Construir e iniciar los contenedores**. Ejecutar el comando de Docker Compose para levantar todos los servicios en segundo plano. La opción `--build` garantiza que las imágenes se reconstruyan, incorporando posibles cambios en el código o en los ficheros de configuración.

   ```
   docker-compose up -d --build
   ```

   Tras completar estos pasos, el sistema queda desplegado y accesible.

A.2.4. Pruebas de implantación

Una vez finalizado el procedimiento de despliegue y levantados todos los contenedores mediante `docker-compose`, es necesario realizar una serie de pruebas básicas de implantación para verificar que el sistema se encuentra correctamente instalado y que los servicios están accesibles.

1. **Acceso al frontend**. Tras unos minutos, dependiendo del tiempo requerido para la instalación de dependencias en cada contenedor, el frontend de la plataforma puede consultarse en la dirección definida en el fichero `.env`.

   Por ejemplo, con la configuración mostrada anteriormente:

   `http://localhost:35873/`

   Una carga correcta de la página indica que el servidor de frontend y la comunicación con el backend están activos.

2. **Prueba de autenticación**. Se ha configurado un usuario *superusuario* que permite validar el correcto funcionamiento del sistema de autenticación y la conexión con la base de datos.

   - **Email**: `superuser@pmt.com`
   - **Contraseña**: `root`

   Si el inicio de sesión se realiza con éxito, se confirma que el flujo de autenticación funciona y que la tabla de usuarios de PostgreSQL está operativa.

3. **Carga de proyectos de prueba**. Para comprobar que el backend gestiona adecuadamente la subida y análisis de proyectos, es recomendable utilizar los modelos de ejemplo incluidos en el directorio:

   `Investigacion/`

   Estos modelos permiten verificar el proceso de carga de ficheros, almacenamiento en el contenedor correspondiente y posterior evaluación.

La superación satisfactoria de estas pruebas confirma que el sistema se encuentra implantado y preparado para su utilización.

A.3. Consideraciones generales sobre el desarrollo

Este apartado recoge una serie de recomendaciones y directrices generales que deben tenerse en cuenta a la hora de modificar, extender o mantener el código fuente de PMTlab. El objetivo es facilitar la evolución del sistema, asegurar la coherencia del desarrollo y minimizar la introducción de errores al incorporar nuevas funcionalidades o mejoras.

A.3.1. Buenas prácticas de desarrollo

Durante la evolución del código se deben seguir una serie de buenas prácticas generales:

- Mantener un **código modular**, reutilizable y correctamente documentado, especialmente en los componentes del backend encargados del análisis y evaluación de modelos.
- Utilizar nombres descriptivos para variables, funciones y ficheros, favoreciendo la legibilidad del código.
- Evitar duplicidades, centralizando la lógica común en funciones o módulos compartidos.
- Comentar aquellas partes del código que implementen lógica compleja o decisiones de diseño relevantes.

En el frontend, se recomienda seguir el enfoque basado en componentes, limitando la lógica de negocio y manteniendo los componentes lo más simples posible.

A.3.2. Extensión del sistema

La plataforma ha sido diseñada para facilitar su ampliación futura. Algunas consideraciones a tener en cuenta son:

- La incorporación de nuevos **modelos predictivos** debe realizarse respetando el mecanismo existente de carga y evaluación, asegurando la compatibilidad con los formatos soportados.
- La inclusión de nuevas métricas o resultados debe gestionarse a través de los módulos correspondientes, sin alterar el flujo general del análisis.
- Cualquier cambio en la estructura de datos debe reflejarse de forma consistente tanto en el backend como en la base de datos.

Este enfoque permite actualizar modelos, versiones o algoritmos sin necesidad de modificar el núcleo del sistema.

A.3.3. Pruebas y verificación de cambios

Cada nueva mejora o modificación introducida en el sistema debe ir acompañada de un proceso de verificación que asegure su correcto funcionamiento. Para ello se recomienda:

- Verificar que las funcionalidades existentes no se ven afectadas por los cambios realizados.
- Comprobar el correcto funcionamiento de los flujos principales del sistema, como la subida de proyectos, el análisis y la evaluación de modelos.
- Revisar los mensajes de error y logs generados para detectar posibles fallos o comportamientos inesperados.

Estas pruebas permiten mantener la estabilidad del sistema y facilitan la detección temprana de errores durante su evolución.

B. Manual de usuario

B.1. Introducción

El manual de usuario tiene como objetivo guiar al usuario en el uso y funcionamiento de la plataforma PMTlab. En él se describen las principales funcionalidades del sistema, los pasos necesarios para acceder y operar en la aplicación, así como las acciones básicas para la carga de proyectos, ejecución de análisis, etc.

B.2. Uso del sistema

Describir todos los aspectos necesarios para una utilización efectiva y eficiente de las características del sistema por parte de los usuarios.

B.2.1. Registrar usuario

Para utilizar la plataforma **PMTlab**, cada usuario debe disponer de una cuenta personal. El registro permite identificar a cada usuario, almacenar sus proyectos de forma independiente y ofrecer una experiencia personalizada dentro del sistema. A continuación se detallan los pasos para crear una cuenta nueva:

1. **Acceda a la pantalla de registro.**
   Desde la página de inicio de sesión de la plataforma, seleccione la opción **"Sign Up"**, ubicada en la parte inferior del formulario de inicio de sesión (Figura B.1).
2. **Complete el formulario de registro.**
   El sistema solicitará una serie de datos personales y de identificación necesarios para crear su perfil:
   - **Correo electrónico:** Dirección válida que servirá como identificador único del usuario.
   - **Contraseña:** Clave personal.
   - **Nombre:** Nombre de pila del usuario.
   - **Primer apellido:** Primer apellido del usuario.
   - **Segundo apellido:** Segundo apellido del usuario (Campo opcional).
   - **Organización:** Nombre de la institución, empresa o universidad a la que pertenece el usuario (por ejemplo, *Universidad de Cádiz*).
   - **¿Es estudiante?:** campo de selección (Sí/No) que permite categorizar a los usuarios para fines estadísticos y de evaluación académica.

3. **Confirme el registro.**
   Haga clic en el botón **"SIGN UP"** (Figura B.2) para finalizar el proceso. Si los datos son válidos, el sistema registrará su cuenta y se le redirigirá a la pantalla de inicio de sesión.

4. **Acceda a su cuenta.**
   Una vez registrado, podrá iniciar sesión introduciendo su correo electrónico y contraseña en la pantalla de inicio de sesión. Desde ese momento, tendrá acceso a las funcionalidades de **PMTlab**.

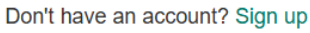


Figura B.1: Enlace a registro de usuario

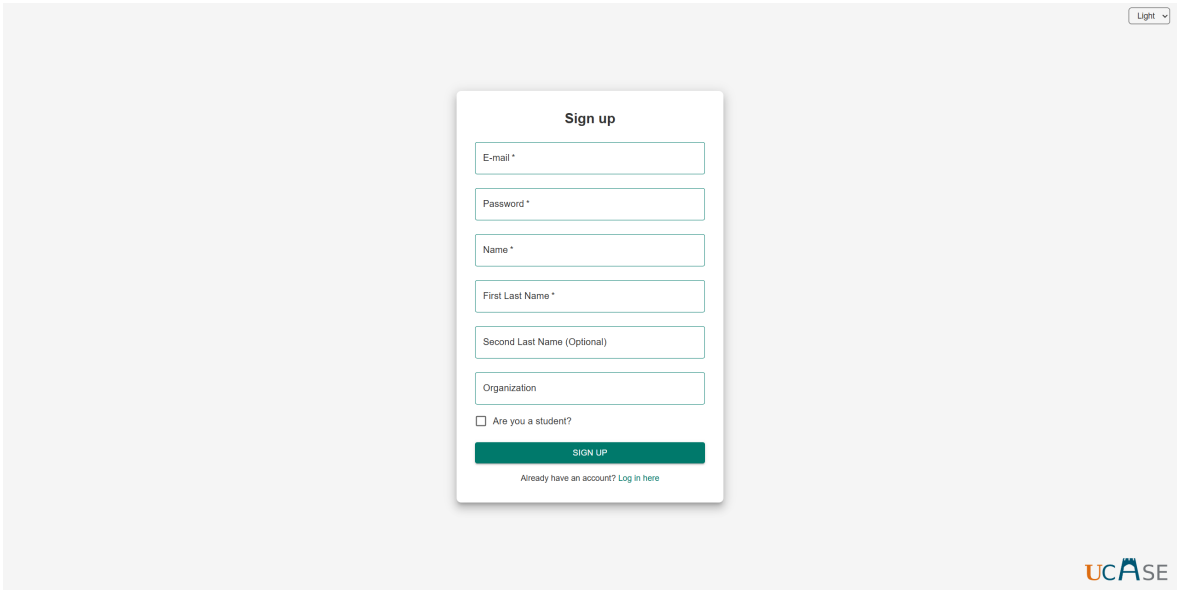


Figura B.2: Pantalla de registro de usuario

B.2.2. Iniciar sesión

Para acceder a las funcionalidades de la plataforma **PMTlab**, los usuarios deben iniciar sesión con su cuenta personal. Esto garantiza un acceso seguro y personalizado a los proyectos y análisis almacenados. Siga los siguientes pasos para iniciar sesión:

1. **Ingrese sus credenciales.**
   Desde la página de inicio de sesión de **PMTlab**, introduzca la información solicitada en los campos correspondientes:

   - **Correo electrónico:** La dirección registrada durante el proceso de creación de cuenta.
   - **Contraseña:** La clave personal definida durante el registro.

2. **Inicie sesión.**
   Haga clic en el botón **"LOG IN"** (Figura B.3) para acceder a su cuenta. Si los datos son correctos, el sistema mostrará la pantalla principal con sus proyectos y análisis disponibles.

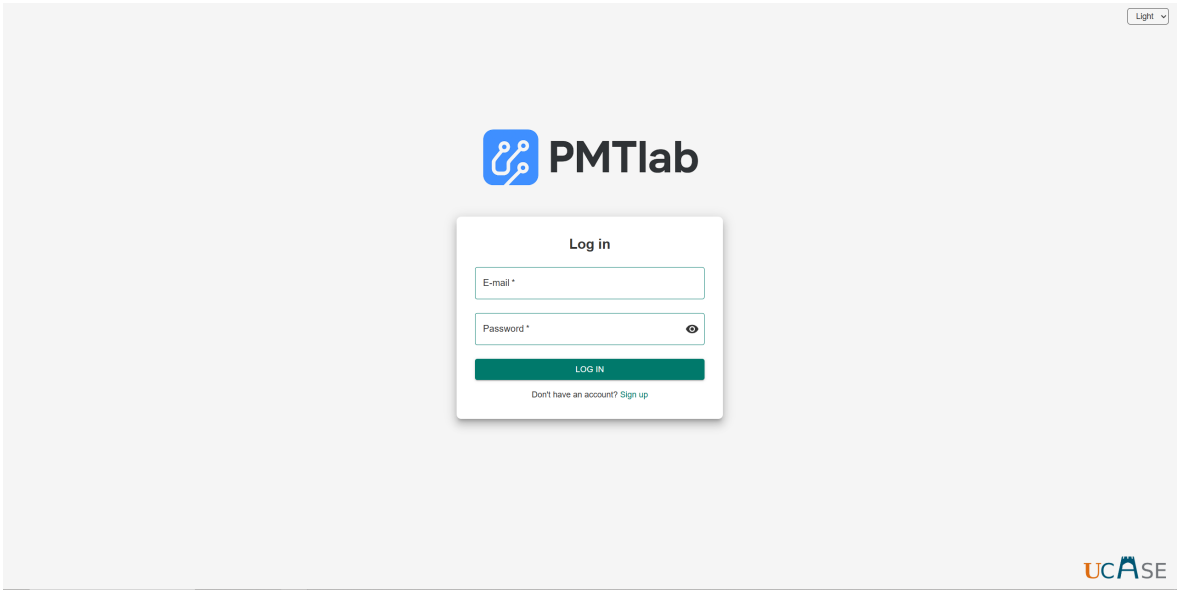


Figura B.3: Pantalla de inicio de sesión

B.2.3. Seleccionar paleta de colores

La plataforma **PMTlab** permite personalizar la apariencia del entorno de trabajo mediante la selección de diferentes paletas de colores. Esto facilita la adaptación visual del sistema a las preferencias del usuario, mejorando la experiencia y comodidad durante su uso.

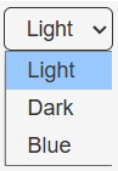


Figura B.4: Selector de la paleta de colores

Para cambiar la paleta de colores, siga los pasos a continuación:

1. **Acceda al selector de paletas.**
   Ubique el selector de la paleta de colores en la esquina superior derecha (Figura B.4).

2. **Seleccione la paleta deseada.**
   El sistema ofrece varias opciones de apariencia, por ejemplo:

   - **Claro:** fondo blanco y colores suaves para una visualización estándar.

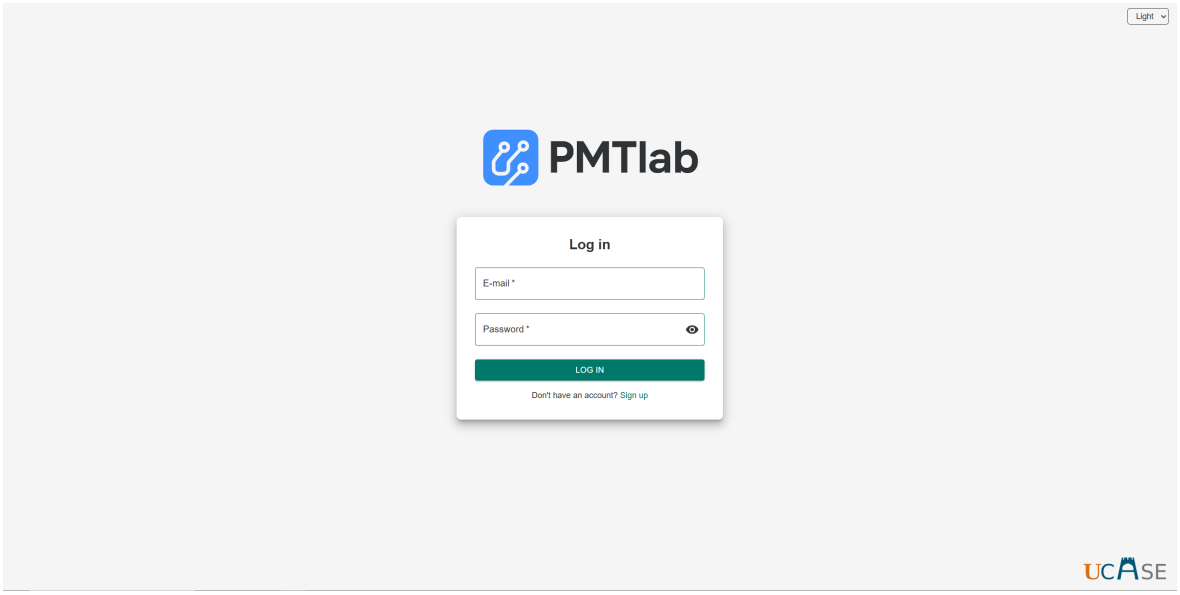


Figura B.5: Pantalla de inicio de sesión con tema Claro

- **Oscuro:** fondo oscuro y colores contrastantes para reducir la fatiga visual en entornos con poca luz.

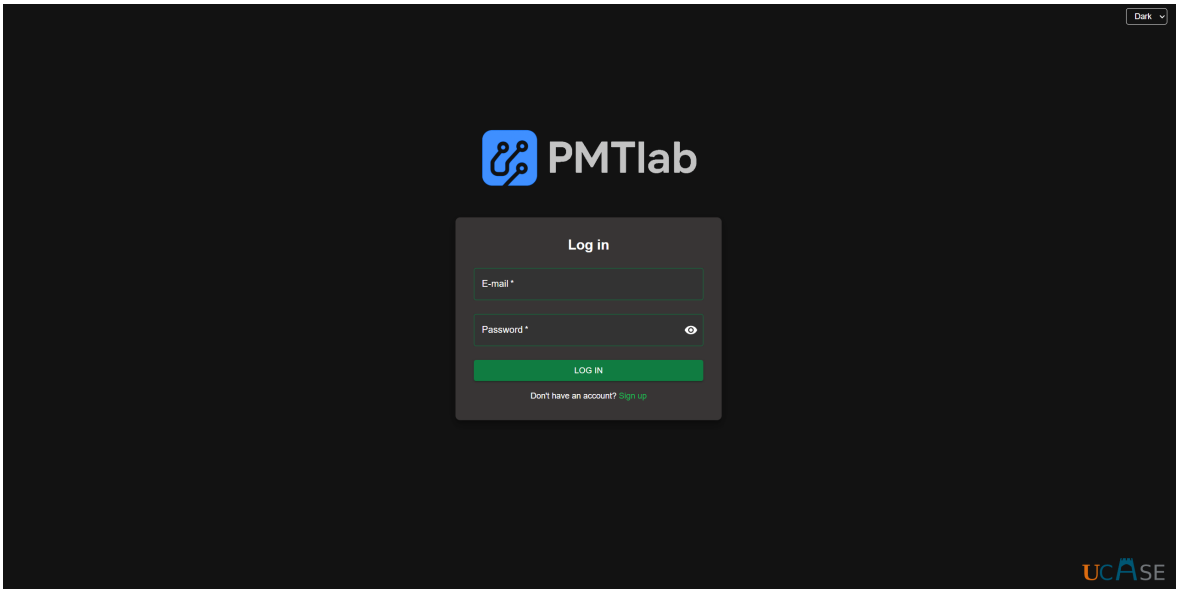


Figura B.6: Pantalla de inicio de sesión con tema Oscuro

- **Azul:** paleta con tonos azules, para una experiencia visual diferente y moderna.



Figura B.7: Pantalla de inicio de sesión con tema Azul

3. **Confirme la selección.**
   Una vez elegida la paleta, el sistema aplicará automáticamente los cambios a todas las pantallas y secciones de **PMTlab**.

Esta funcionalidad permite que cada usuario configure la plataforma según sus preferencias visuales, mejorando la comodidad y la eficiencia durante el uso.

B.2.4. Menú lateral

El **menú lateral** de la plataforma **PMTlab** proporciona acceso rápido a las principales secciones y herramientas del sistema. Su diseño facilita la navegación, agrupando las funcionalidades según el tipo de usuario y sus permisos asociados.



Figura B.8: Botón del menú

1. **Abrir el menú lateral.**
   Para desplegar el menú, haga clic en el **botón del menú** ubicado en la esquina superior derecha de la interfaz (Figura B.8). Al pulsarlo, se mostrará un panel lateral con las distintas opciones de navegación.
2. **Opciones del menú según el tipo de usuario.**
   El contenido del menú lateral varía en función del perfil del usuario:
   - **Usuarios administradores y superusuarios:** disponen de acceso completo a todas las secciones, incluyendo la gestión de usuarios, proyectos y modelos.

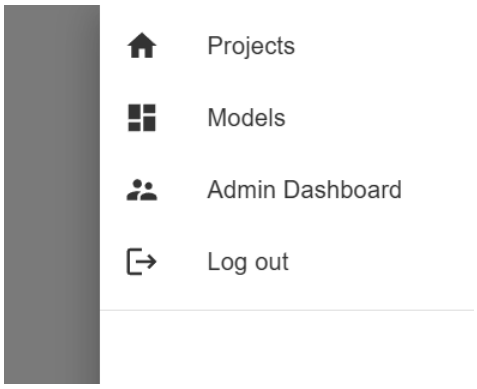


Figura B.9: Menú para usuarios administrador y superusuario

- **Usuarios corrientes:** tienen acceso únicamente a las funcionalidades relacionadas con sus propios proyectos y análisis, así como a la sección de modelos.

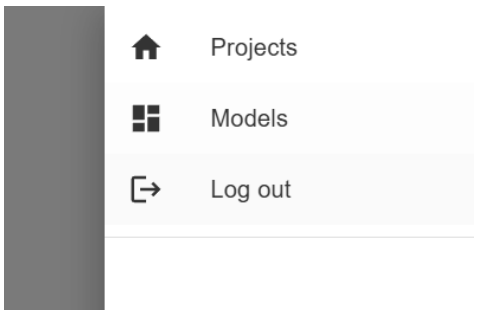


Figura B.10: Menú para usuarios corrientes

3. **Navegación dentro del menú.**
   Seleccione la opción deseada haciendo clic sobre el nombre o icono correspondiente. El sistema cargará automáticamente la sección asociada en el área principal de trabajo.
4. **Cerrar el menú lateral.**
   Puede ocultar nuevamente el menú seleccionando cualquier zona fuera del panel lateral.

El menú lateral constituye el punto central de navegación en **PMTlab**, permitiendo un acceso organizado y eficiente a todas las funcionalidades disponibles según el rol del usuario.

B.2.5. Gestionar proyectos

La plataforma **PMTlab** permite a los usuarios gestionar sus proyectos, desde su adición hasta el análisis, exportación de métricas y visualización de resultados. La pantalla principal de proyectos muestra un listado de todos los proyectos disponibles junto con sus estados y acciones asociadas.

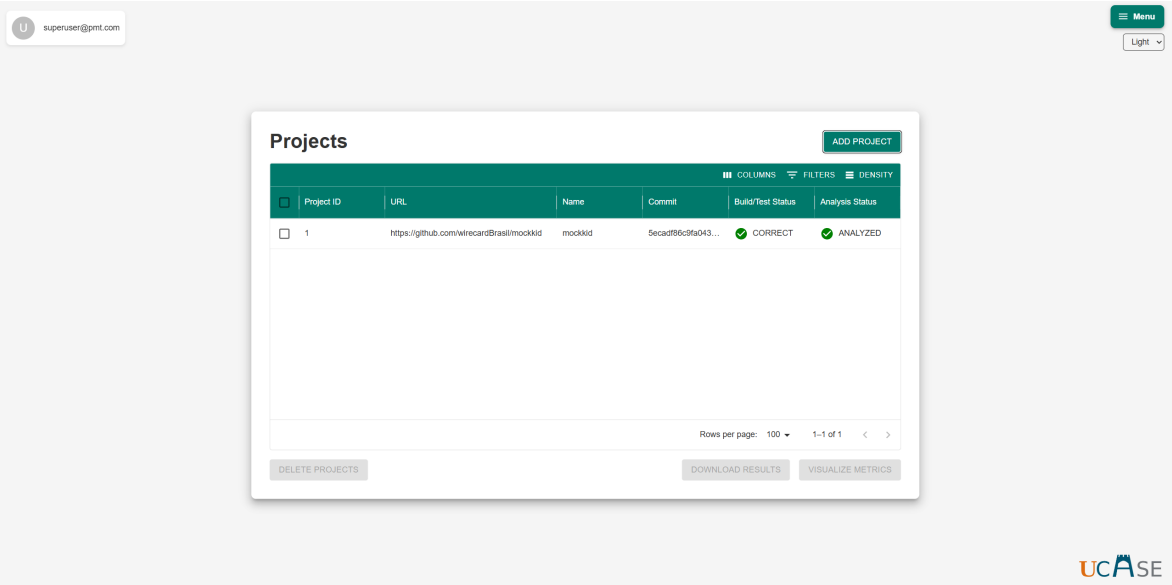


Figura B.11: Pantalla principal de proyectos

B.2.5.1. Añadir y analizar proyecto

Para crear un nuevo proyecto y comenzar su análisis, siga estos pasos:

1. **Abrir el formulario de añadir proyecto.**
   Haga clic en el botón **"ADD PROJECT"** ubicado en la pantalla principal de proyectos (Figura B.12).



Figura B.12: Botón para añadir proyecto

2. **Introducir enlaces de repositorios.**
   Complete la información solicitada en el pop-up, introduciendo el enlace al repositorio del proyecto de forma manual o cargando un fichero CSV que disponga de los enlaces de varios proyectos en columna o separados por comas (Figura B.13).

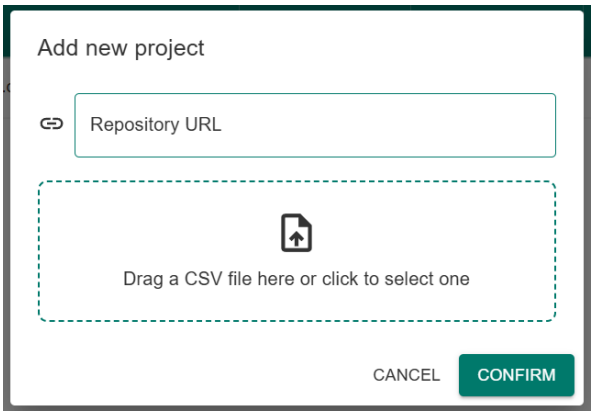


Figura B.13: Pop-up para introducir enlaces de los repositorios de los proyectos

3. **Analizar el proyecto.**
   Una vez añadido, el proyecto comenzará su análisis automáticamente, mostrando su progreso en la pantalla principal (ver Figura B.14).

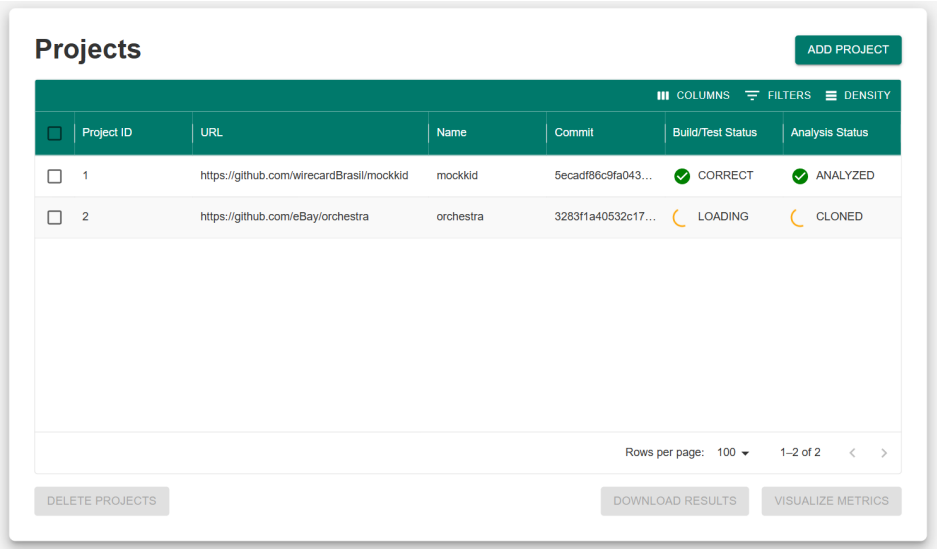


Figura B.14: Proyecto analizándose

B.2.5.2. Borrar proyecto

Para eliminar un proyecto de la plataforma:

1. Seleccione uno o varios proyectos y haga clic en el botón **"DELETE PROJECTS"** (Figura B.15).



Figura B.15: Botón para borrar proyectos

B.2.5.3. Exportar métricas

Para exportar los datos y métricas de un proyecto:

1. Seleccione los proyectos deseados y haga clic en el botón **"DOWNLOAD RESULTS"** (Figura B.16).



Figura B.16: Botón para exportar las métricas

2. Si ha seleccionado un único proyecto, el sistema generará un fichero CSV con todas las métricas del proyecto. Si ha seleccionado más de uno, el sistema generará un fichero ZIP con los CSV correspondientes a cada proyecto.

B.2.5.4. Visualizar métricas de proyecto

Para consultar las métricas de un proyecto de forma detallada:

1. Seleccione uno o varios proyectos y haga clic en el botón **"VISUALIZE METRICS"** (Figura B.17).



Figura B.17: Botón para visualizar las métricas

2. Se abrirá la **pantalla de visualización de métricas** con todos los datos disponibles (Figura B.18).

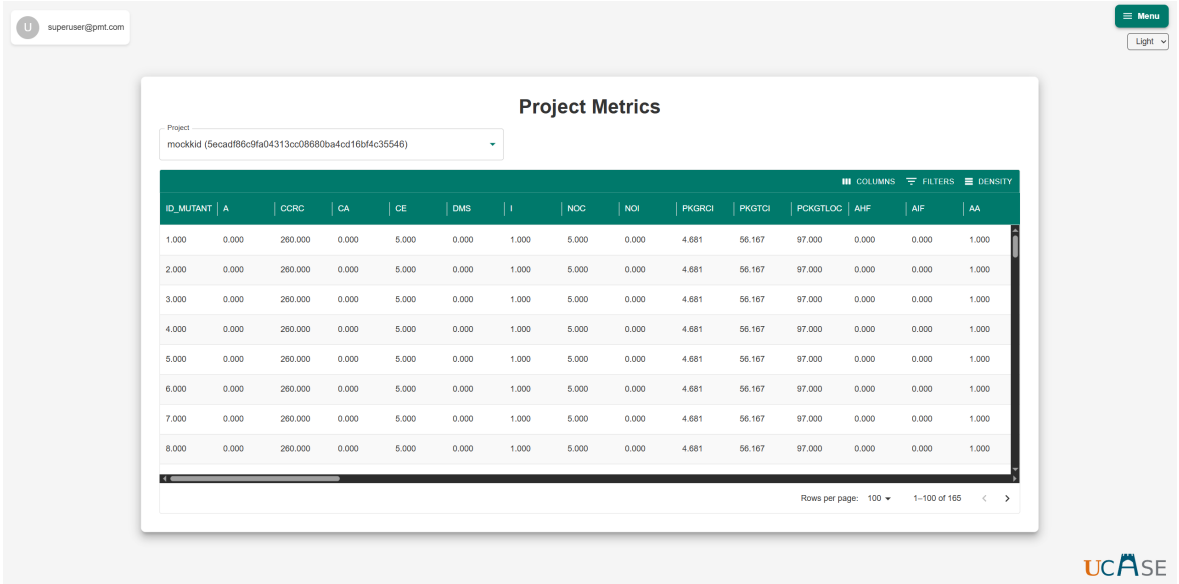


Figura B.18: Pantalla de visualización de métricas

3. Para seleccionar el proyecto cuyas métricas desea consultar, utilice el **selector de proyecto** (Figura B.19).

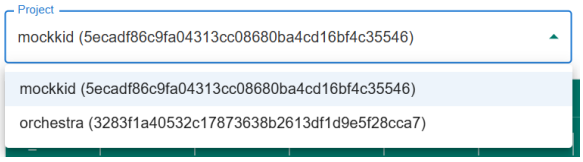


Figura B.19: Selector de proyecto para visualizar métricas

4. Puede personalizar la visualización del datagrid seleccionando las columnas que desea mostrar (Figura B.20).

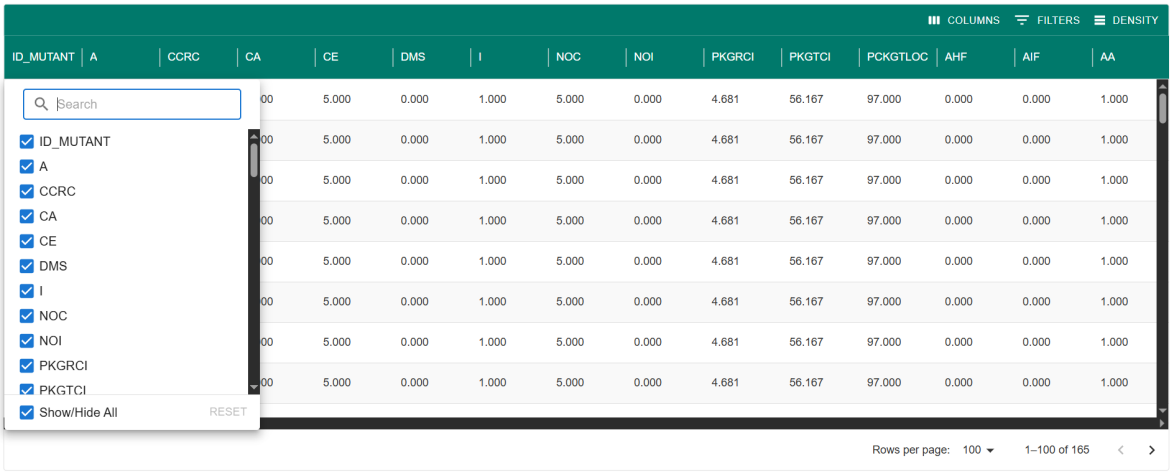


Figura B.20: Selector de columnas del datagrid

Esta sección permite al usuario administrar sus proyectos de manera completa: añadir nuevos proyectos, analizarlos, borrar los que ya no necesite, exportar métricas y visualizar datos detallados para la toma de decisiones.

B.2.6. Gestionar modelos

La sección de modelos subidos de la plataforma **PMTlab** permite a los usuarios administrar y evaluar. La disponibilidad de ciertas acciones depende del tipo de usuario: solo los usuarios **administrador** y **superusuario** pueden subir nuevos modelos, mientras que todos los usuarios pueden evaluar modelos existentes.

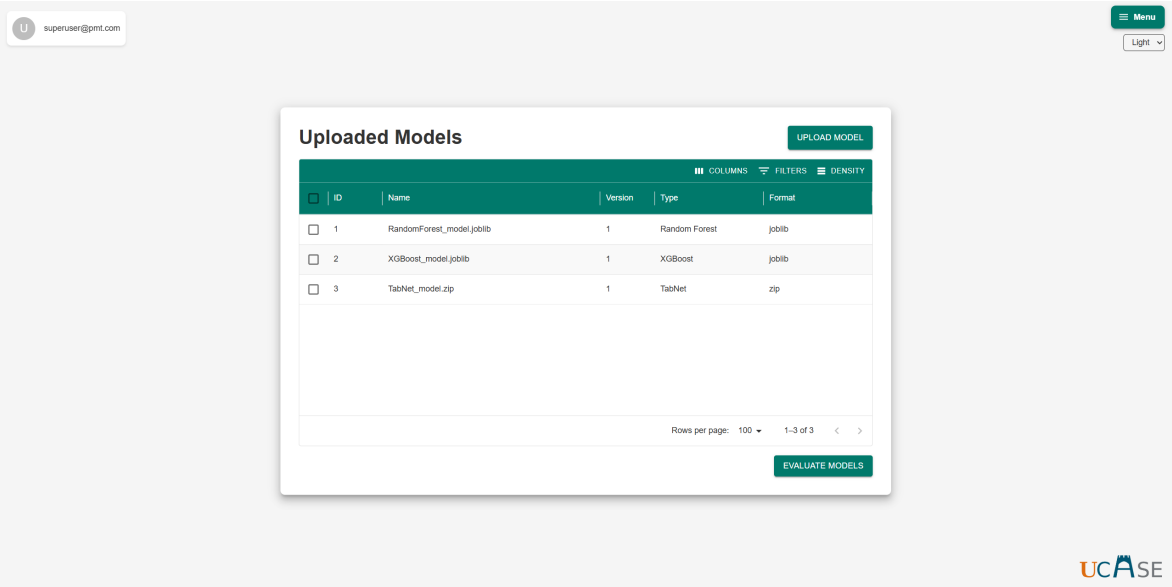


Figura B.21: Pantalla de modelos de usuarios admin y superuser

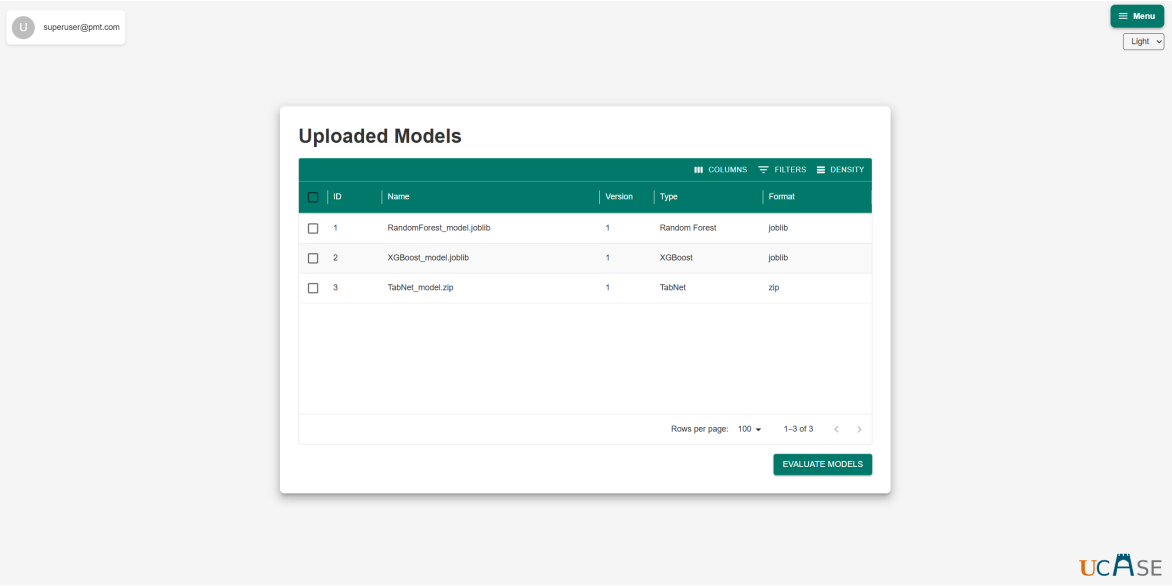


Figura B.22: Pantalla de modelos de usuarios corrientes

B.2.6.1. Subir modelo

Solo los usuarios **admin** y **superuser** pueden subir modelos nuevos a la plataforma:

1. Haga clic en el botón **"UPLOAD MODEL"** en la pantalla de gestión de modelos (Figura B.23).



Figura B.23: Botón para subir un modelo

2. Una vez mostrado el pop-up, seleccione el tipo de algoritmo del modelo y carguelo. Tenga en cuenta los formatos permitidos (Figura B.24).

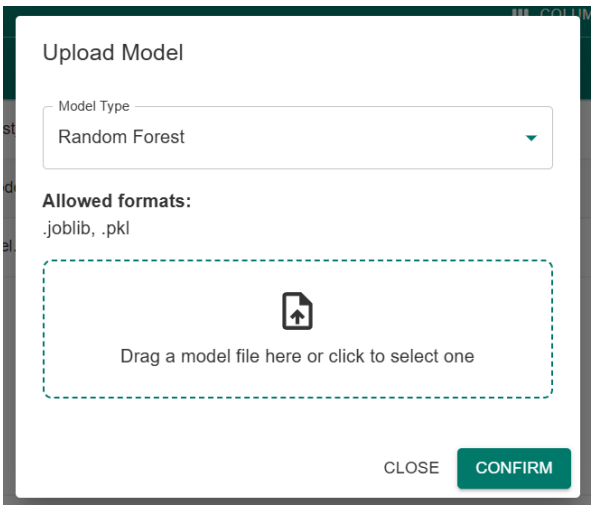


Figura B.24: Pop-up para subir un modelo

3. Una vez confirmado, el modelo estará disponible para su evaluación.

B.2.6.2. Evaluar modelos

Cualquier usuario puede evaluar modelos existentes utilizando proyectos disponibles analizados correctamente en la plataforma:

1. Seleccione los modelos que desee evaluar y haga clic en el botón **"EVALUATE MODELS"** (Figura B.25).



Figura B.25: Botón para evaluar modelos

2. En el pop-up, seleccione el proyecto con el que desea evaluar los modelos (Figura B.26).

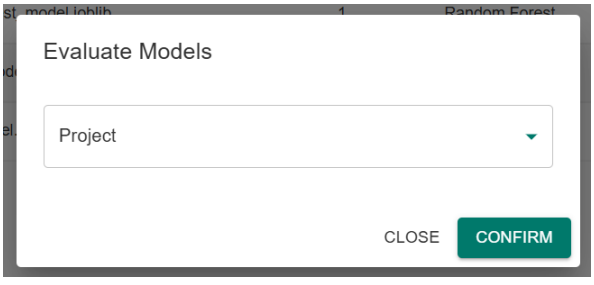


Figura B.26: Pop-up para seleccionar el proyecto con el que evaluar los modelos

3. Una vez realizado el análisis, se mostrarán las métricas resultantes de los modelos (Figura B.27).

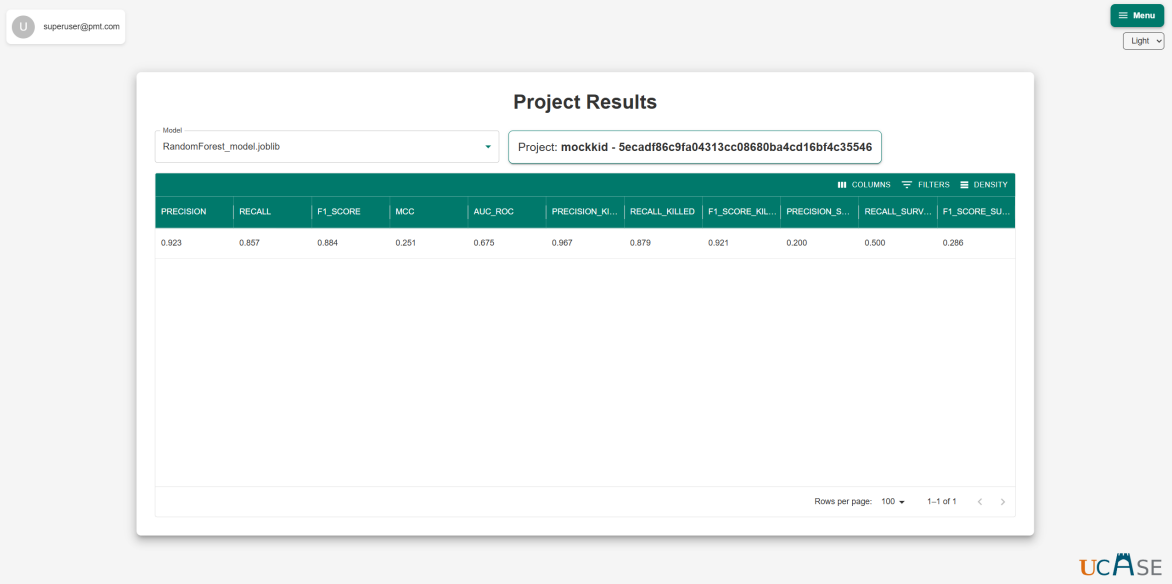


Figura B.27: Pantalla con las métricas resultantes de los modelos

4. Para consultar métricas específicas, utilice el **selector de modelo** para elegir el modelo deseado (Figura B.28).

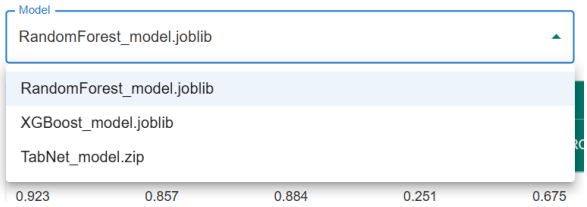


Figura B.28: Selector de modelo para visualizar resultados

Esta funcionalidad permite administrar y evaluar modelos de manera controlada según los permisos de usuario, asegurando que solo los administradores y superusuarios puedan subir modelos nuevos, mientras que todos los usuarios pueden evaluar los resultados.

B.2.7. Gestionar usuarios

La plataforma **PMTlab** permite a los usuarios con permisos de **admin** o **superuser** gestionar la información y los permisos de otros usuarios. Esto incluye visualizar detalles, modificar estados y cambiar roles, garantizando un control adecuado sobre el acceso a la plataforma.

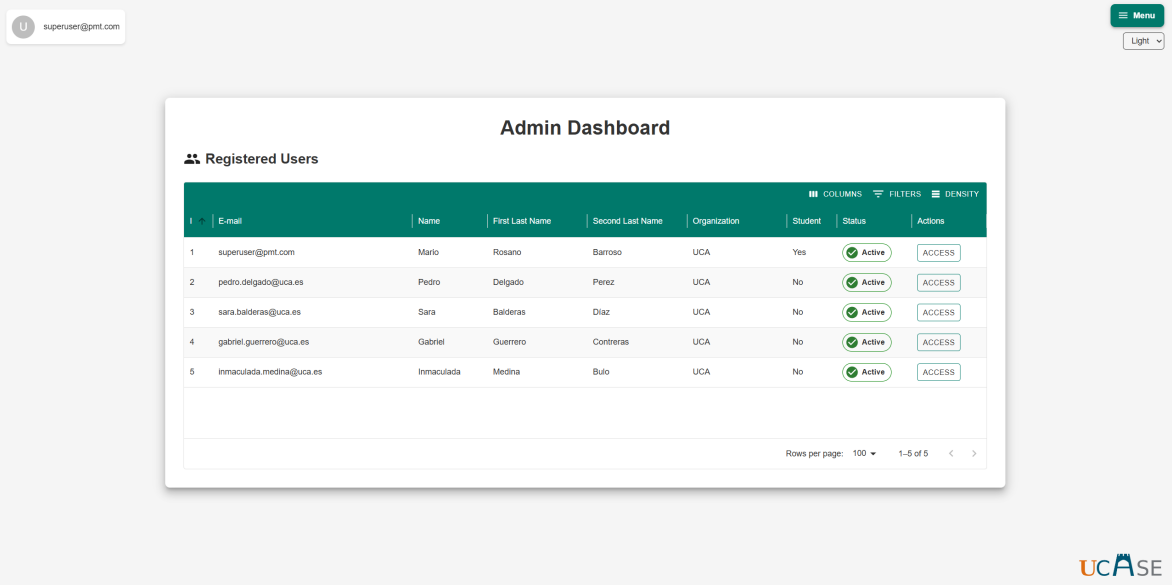


Figura B.29: Pantalla de gestión de usuarios

B.2.7.1. Gestionar información de un usuario

Para consultar o modificar los datos de un usuario:

1. Acceda a la pantalla de detalles del usuario deseado haciendo click en el botón **"ACCESS"** correspondiente, donde podrá visualizar y modificar el estado y rol (Figura B.30).

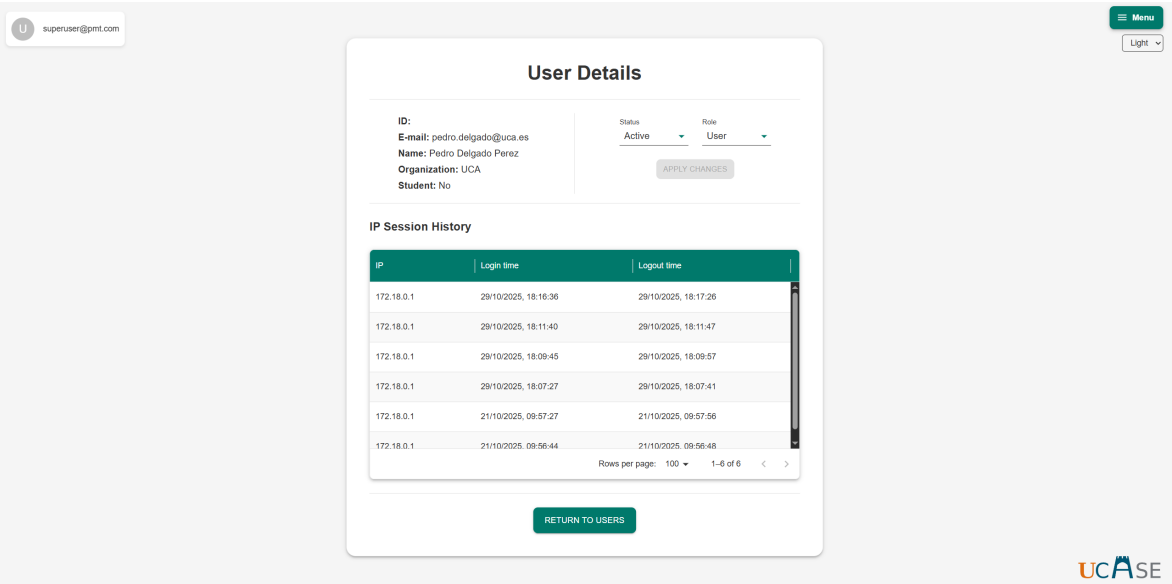


Figura B.30: Pantalla de detalles del usuario

Además, en esta pantalla se muestra el **registro de actividad del usuario**, que incluye:

- Dirección **IP** desde la que se ha accedido a la plataforma.
- Fecha y hora del **inicio de sesión**.
- Fecha y hora del **cierre de sesión**.

B.2.7.2. Modificar estado de un usuario

El estado de un usuario determina su acceso a la plataforma. Para cambiarlo:

1. Abra el **selector del estado del usuario** y elija la opción deseada (Activo o bloqueado) (Figura B.31).

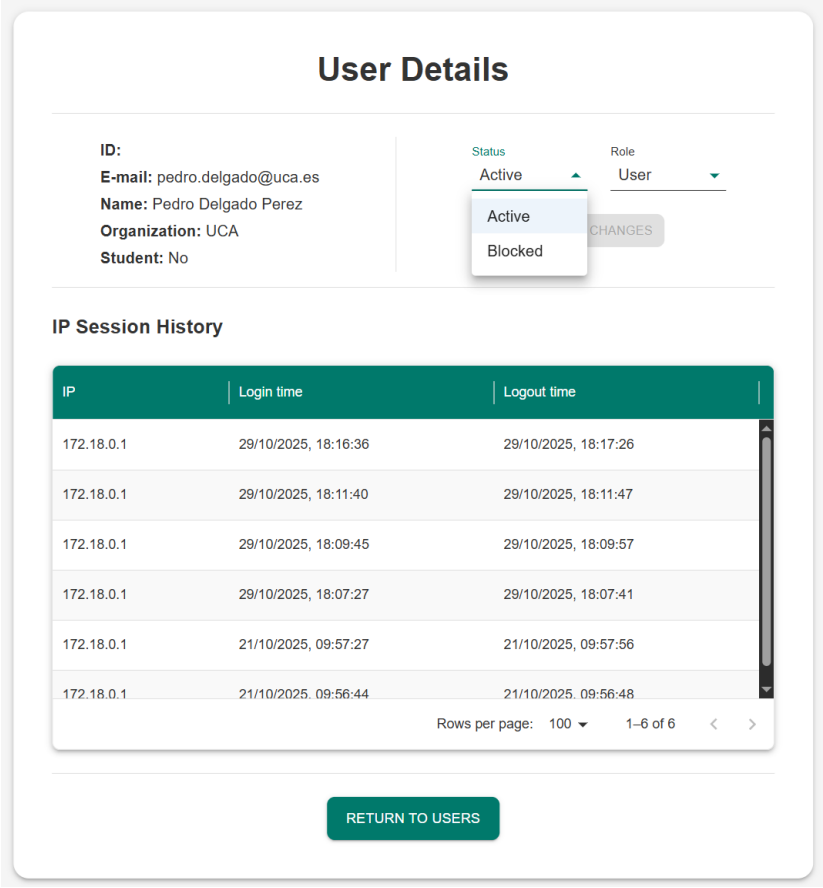


Figura B.31: Selector del estado del usuario

2. Una vez modificado, el estado se reflejará en el **Admin Dashboard**, mostrando usuarios bloqueados o activos (Figura B.32).

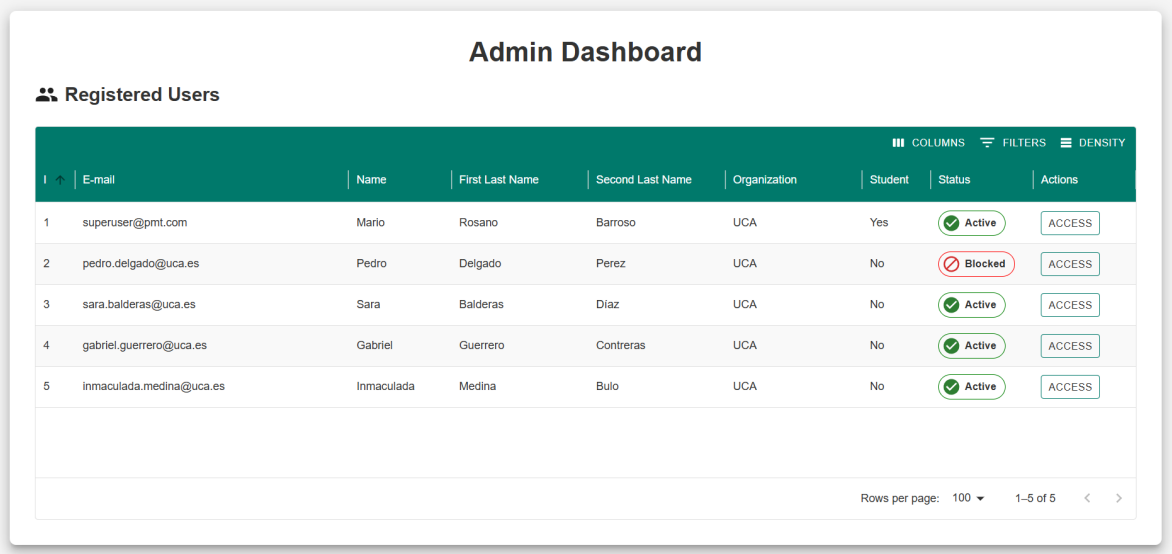


Figura B.32: Estado bloqueado del usuario en el Admin Dashboard

3. Si el rol del usuario es de un nivel inferior o igual al usuario al que se desea modificar el estado, el selector no estará disponible (Figura B.33).

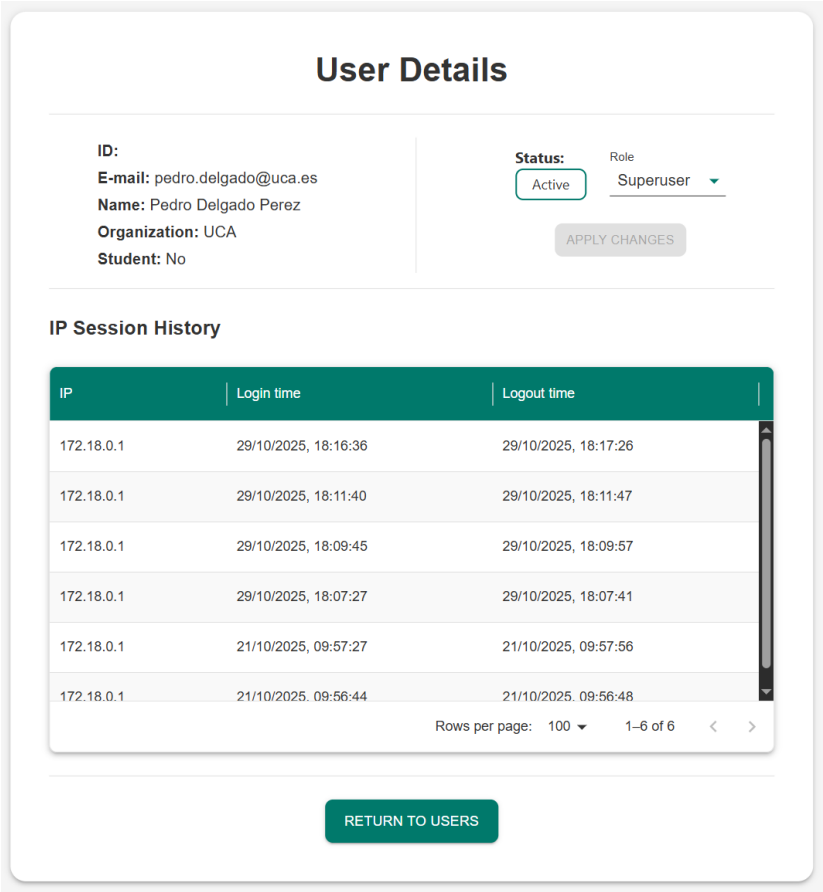


Figura B.33: Selector del estado no disponible

B.2.7.3. Modificar rol de un usuario

El rol de un usuario determina sus permisos y accesos dentro de la plataforma. Para cambiarlo:

1. Abra el **selector del rol del usuario** y elija el rol adecuado según las responsabilidades del usuario (Figura B.34).

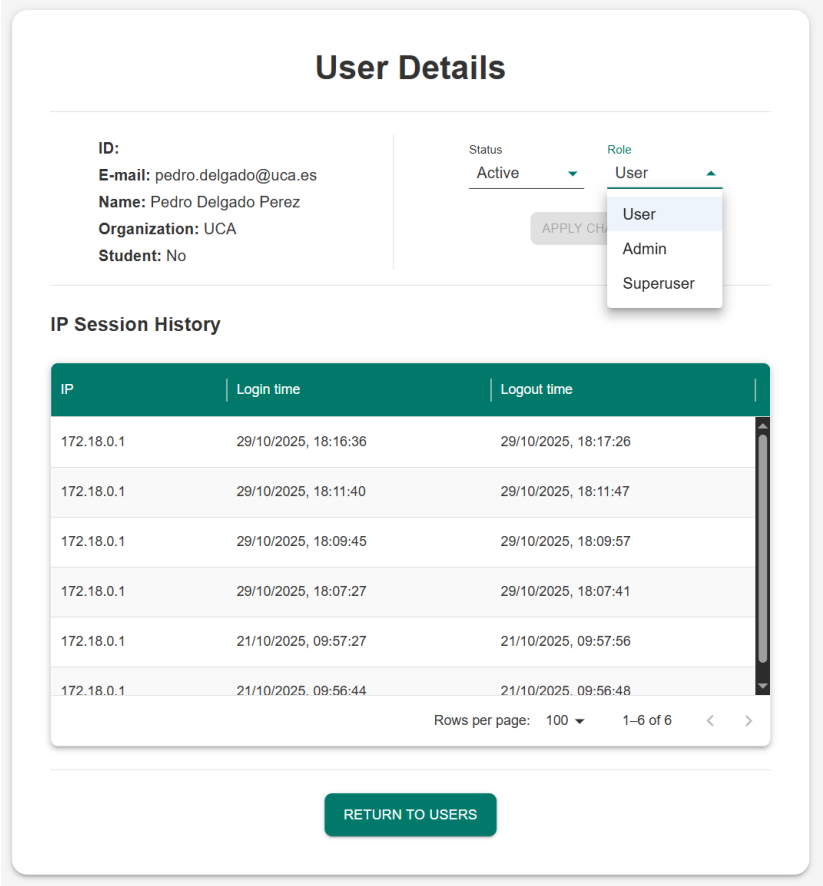


Figura B.34: Selector del rol del usuario

2. Está opción es solo aplicable a los usuarios con rol superuser, si su rol es admin el selector no estará disponible (Figura B.35).

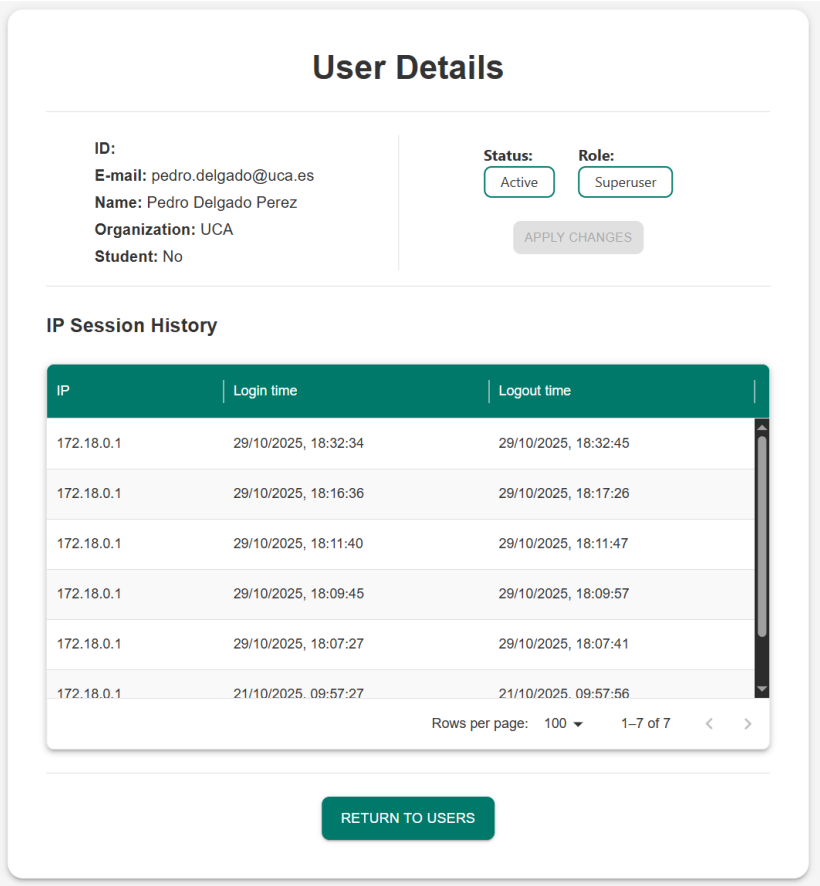


Figura B.35: Selector del rol no disponible

Esta sección permite a los administradores mantener un control eficiente sobre los usuarios de **PMTlab**, asegurando que cada usuario tenga los permisos y el acceso apropiados según su rol y estado.

C. Lista de métricas

- File - Nombre del archivo fuente en el que se encuentra el mutante generado.
- Class - Nombre de la clase del proyecto a la que pertenece el mutante.
- Method - Método o función concreta dentro de la clase donde se ha aplicado el mutante.
- Line - Número de línea del código fuente en la que se ha insertado el mutante.
- Operator - Operador de mutación aplicado (por ejemplo, reemplazo de operadores aritméticos, cambios en condiciones lógicas, etc.).
- Label - Etiqueta que indica el resultado real del mutante tras ejecutar los casos de prueba, `KILLED` (detectado por los tests) o `SURVIVED` (no detectado).
- NumTestCovered - Número de casos de prueba que ejecutan el código donde se encuentra el mutante.
- NumExecuteCovered - Número de veces que la línea mutada ha sido ejecutada durante la ejecución de los tests.
- NumMutantAssertion - Número de aserciones de los tests que están directamente relacionadas con el comportamiento del mutante.
- NumClassAssertion - Número de aserciones existentes en los casos de prueba de la clase afectada por el mutante.
- TypeReturn - Tipo de dato que devuelve el método donde se encuentra el mutante (por ejemplo, `int`, `boolean`, `String`, `void`, etc.).
- Raw Total Lines of Code (RTLOC) - El número real de líneas de código en una clase, utilizando los números de línea del propio archivo. Se cuentan los comentarios, los espacios en blanco y cualquier otro contenido (clase).
- Total Lines of Code (TLOC) - El número total de líneas de código, ignorando los comentarios, los espacios en blanco y las diferencias de formato (proyecto, paquete, clase, método).
- Number of Attributes (NF) - El número de campos/atributos (clase).
- Number of Static Attributes (NSF) - El número de atributos estáticos (clase).
- Number of Public Attributes (NPF) - El número de atributos públicos (clase).
- Number of Methods (NM) - El número de métodos (clase).
- Number of Static Methods (NSM) - El número de métodos estáticos (clase).
- Number of Public Methods (NPM) - El número de métodos públicos (clase).
- Number of Classes (NOC) - El número de clases dentro de un paquete (paquete).

- Number of Parameters (NOP) - El número de parámetros que recibe un método (método).
- Depth of Inheritance Tree (DIT) - La profundidad máxima de la jerarquía de herencia para una clase (clase).
- Number of Overridden Methods (NORM) - El número de métodos que una clase sobrescribe o implementa a partir de una clase padre (clase).
- Number of Inherited Methods (NMI) - El número de métodos que una clase hereda de las clases padre (clase).
- Number of Methods Added to Inheritance (NMA) - El número de métodos que una clase añade a la jerarquía de herencia; métodos definidos en la clase que no ha sobrescrito ni heredado (clase).
- Specialization Index (SIX) - El grado de especialización de una clase, definido como (DIT * NORM) / NOM (clase).
- Number of Methods Inherited Total (Mit) - Métodos heredados en total (clase).
- Number of Methods Inherited (Mi) - Métodos heredados pero no sobrescritos (clase).

  Number of Methods Defined (Md) - Métodos definidos dentro de la clase (igual que NMA) (clase).
- Number of Methods Overidden (Mo) - Métodos de la clase que sobrescriben un método que, de otro modo, sería heredado (clase).
- Number of Methods (All) (Ma) - Métodos que pueden invocarse en una clase (heredados, sobrescritos y definidos). Ma = Md + Mi. Igual que NORM (clase).
- Method Inheritance Factor (MIF) - Mi / Ma (clase).
- Number of Public Methods Defined (PMd) - Número de métodos definidos que son públicos (clase).
- Number of Public Methods Inherited (PMi) - Número de métodos heredados (pero no sobrescritos) que son públicos (clase).
- Public Methods Ratio (PMR) - (PMd+PMi)/(Md+Mi) (clase).
- Number of Hidden Methods Defined (HMd) - Número de métodos definidos que no son públicos (clase).
- Number of Hidden Methods Inherited (HMi) - Número de métodos heredados (pero no sobrescritos) que no son públicos (clase).
- Method Hiding Factor (MHF) - PMd / Md (clase) Number of Methods Inherited Ratio (NMIR) - (Mi / Mit) * 100 (de Lorenz y Kidd, renombrado aquí para distinguirlo de NMI) (clase).
- Coupling Factor (CF) - Véase http://www.cs.kent.edu/~jmaletic/cs63901/lectures/SoftwareMetrics.pdf (nota: esto se define de forma diferente a la del texto. El texto divide por $TC^2 - TC$ para el número total de relaciones. Esa

es una métrica a nivel de sistema, pero esta métrica es a nivel de clase, lo que significa que el número total de relaciones es 2(TC - 1). Si una clase usa a todas las demás clases y es usada por todas las demás, ese es su valor para un CF de 1) (clase).

- Polymorphism Factor (PF) - Mo / (Md * NOD) (clase).
- Number of Attributes Inherited Total (Ait) - Atributos heredados en total (clase).
- Number of Attributes Inherited (Ai) - Atributos heredados pero no sobrescritos (clase).
- Number of Attributes Defined (Ad) - Atributos definidos dentro de la clase (igual que NMA) (clase).
- Number of Attributes Overidden (Ao) - Atributos de la clase que sobrescriben un atributo que, de otro modo, sería heredado (clase).
- Number of Attributes (All) (Aa) - Atributos que pueden referenciarse en una clase (heredados, sobrescritos y definidos). Aa = Ad + Ai (clase).
- Attribute Inheritance Factor (AIF) - Ai / Aa (clase).
- Number of Public Attributes Defined (Av) - Número de atributos definidos que son públicos (clase).
- Number of Hidden Attributes Defined (Ah) - Número de atributos definidos que no son públicos (clase).
- Method Hiding Factor (AHF) - Ah / Ad (clase).
- McCabe Cyclomatic Complexity (VG) - El número de rutas posibles únicas a través del código (método).
- Weighed Methods per Class (WMC) - La suma de todas las complejidades ciclomáticas de todos los métodos de una clase (clase).
- Lack of Cohesion Methods (LCOM) - Una medida de la cohesión de una clase. Calculada mediante el método de Henderson-Sellers, basada en el número de conjuntos disjuntos formados al comparar los métodos con los atributos que utilizan (clase).
- Number of Interfaces (NOI) - El número de clases abstractas (e interfaces) en un paquete (paquete).
- Afferent Coupling (Ca) - Número de clases externas a un paquete que dependen de él (paquete).
- Efferent Coupling (Ca) - Número de clases dentro de un paquete que dependen de clases externas a él (paquete).
- Instability (I) - Efectivamente, el nivel de riesgo de un paquete, es decir, con qué frecuencia tiene razones para cambiar, Ce / (Ce + Ca) (paquete).
- Abstractness (A) - El número de clases abstractas (e interfaces) dividido entre el número total de tipos en un paquete, NOI / NOC (paquete).

- Normalized Distance from Main Sequence (DMS) - La métrica de Robert Martin para la distancia de un paquete respecto al ideal, — A + I - 1 — (paquete).
- Nested Block Depth (NBD) - La profundidad máxima del nivel más profundo de anidamiento dentro de un método (método).
- Number of Children (NOCh) - El número de clases que extienden directamente esta clase (clase).
- Number of Parents (NOPa) - El número de clases que esta clase extiende directamente (clase).

  Number of Descendants (NOD) - El número total de clases que tienen a esta clase como ancestro (clase).
- Number of Ancestors (NOA) - El número total de clases que tienen a esta clase como descendiente (clase).
- Number of Links (NOL) - Número de enlaces (asociaciones, generalizaciones, relaciones de uso) entre una clase y todas las demás (clase).
- Number of Dependants (NODa) - El número total de clases que dependen de esta clase (clase).
- Number of Dependencies (NODe) - El número total de clases de las que depende una clase (igual que NOL) (clase).
- Class Category Relational Cohesion (CCRC) - La tasa de cohesión entre las clases de un paquete. Suma(NOL) / NOC (paquete).
- Number of Comparisons (NCOMP) - El número de comparaciones en un método (método).
- Number of Control Variables (NVAR) - El número de variables de control referenciadas en un método (método).
- McClure's Complexity Metric (MCLC) - NCOP + NVAR (método).
- Fan-out (Fout) - El número de métodos inmediatamente subordinados a un método (método).
- Fan-in (Fin) - El número de métodos que invocan un método (método).
- Structural Complexity (Si) - $Fout^2$ (método).
- Input/Output Variables (IOVars) - NOP + 1 (0 si el tipo de retorno es void) (método).
- Data Complexity (Di) - (IOVars)/(Fout+1) (método).
- System Complexity (Ci) - Si + Di (método).
- Class Total System Complexity (ClTCi) - La suma de Ci sobre todos los métodos de la clase (clase).
- Class Relative System Complexity (ClRCi) - El promedio de Ci sobre todos los métodos de la clase (clase).

- Package Total System Complexity (PkgTCi) - La suma de Ci sobre todos los métodos de todas las clases de un paquete (paquete).
- Package Relative System Complexity (PkgRCi) - El promedio de Ci sobre todos los métodos de todas las clases de un paquete (paquete).
- Number of Collaborations (CRC) - El número de colaboraciones entre una clase y todas las demás (clase).
- Data Abstraction Coupling (DAC) - Número de atributos en una clase que representan tipos de otras clases (clase).
- Size2 (SIZE2) - Número de atributos y número de métodos locales definidos en una clase (clase).
- Data Access Metric (DAM) - Relación entre el número de atributos privados (protegidos) y el número total de atributos declarados en la clase (clase).
- Direct Class Coupling (DCC) - Conteo del número de clases distintas con las que una clase está directamente relacionada. La métrica incluye las clases que están directamente relacionadas mediante declaraciones de atributos y el paso de mensajes (parámetros) en los métodos (clase).
- Measure of Aggregation (MOA) - Esta métrica mide el grado de la relación parte-todo, realizada mediante el uso de atributos. La métrica consiste en contar el número de declaraciones de datos cuyos tipos son clases definidas por el usuario (clase).