UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Asistente de prácticas ágiles para repositorios en GitHub Documentación Técnica

Presentado por Lucas Olmedo Díez
en Universidad de Burgos — Junio de 2025
Tutor: Carlos López Nozal

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

El Plan de Proyecto Software describe el proceso de desarrollo de la aplicación web documentada en esta memoria. Describe los procesos seguidos durante la planificación y su seguimiento para el desarrollo continuo del proyecto. Para la planificación del proyecto se ha usado la aplicación **Zube**, que permite un seguimiento de las *issues* del repositorio de GitHub que implementa diferentes prácticas ágiles.

Este apartado del Anexo describirá, además, la viabilidad del proyecto, que representa recursos y costes valorados para el mismo, tanto humanos como materiales. Este punto de vista de carácter económico es necesario para conocer los límites del proyecto y saber en qué medida sus objetivos entran dentro de los mismos. Para ello se calculará una aproximación de los fondos que requeriría un trabajador que desarrolle el proyecto, y se valorarán los posibles riesgos surgidos durante el desarrollo y cómo actuar frente a ellos.

Con el enfoque estructurado desarrollado este plan de la documentación, el equipo de desarrollo será capaz de cumplir con los objetivos establecidos del proyecto y entregar un resultado satisfactorio dentro de los límites calculados.

A.2. Planificación temporal

Como se ha mencionado anteriormente, la planificación temporal del proyecto se ha llevado a cabo con el uso de la herramienta de las *issues* y

milestones de GitHub, y la herramienta de **Zube** basando todo el desarrollo en tareas. Cada tarea representa una Issue de GitHub y representan una historia de usuario o pequeña tarea interna. Las tareas están bien estructuradas y clasificadas con etiquetas y *milestones* que indican con qué parte de la estructura del proyecto están relacionadas, y a qué fase del desarrollo pertenecen, respectivamente.

Un *sprint*, por su parte, representa una fase del desarrollo del software. En la planificación de este proyecto la mayoría de los sprints se han calculado con una duración de dos semanas y han abarcado entre 10 y 18 tareas o *issues*.

Además, **Zube** incluye varias herramientas de agilidad que favorecen el seguimiento del desarrollo de la aplicación como los Gráficos *burnup* y *burndown*, que representan la evolución del desarrollo de tareas realizadas en cada *sprint*. Más adelante en esta misma sección se pueden visualizar ejemplos de los gráficos *burnup* y *burndown* resultado de nuestros *sprints*.

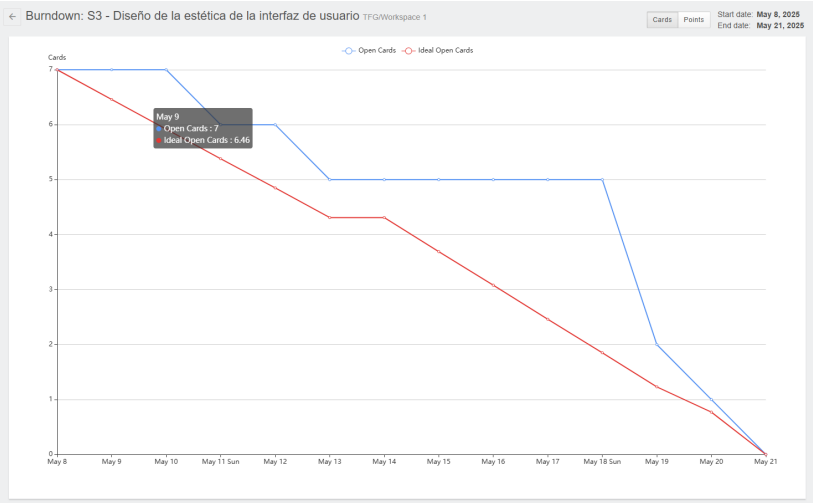

Figura A.1: Gráfico Burndown del Sprint 3

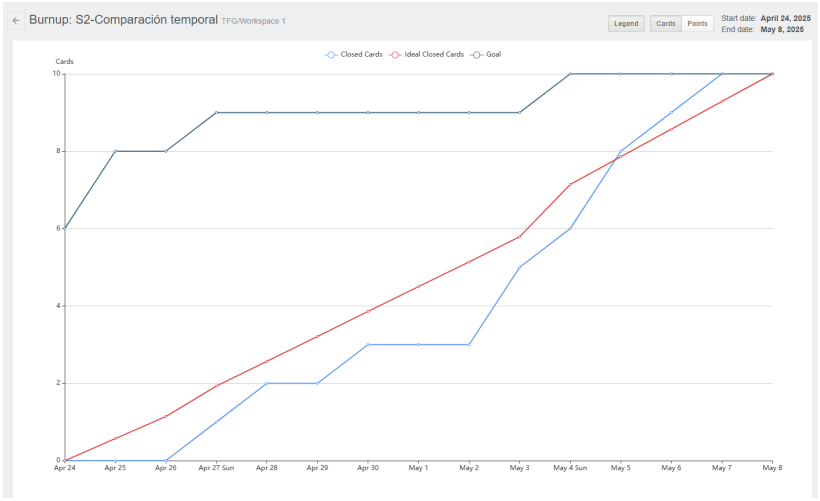

Figura A.2: Gráfico Burnup del Sprint 2

A continuación se explicará cómo se ha desarrollado, por medio de *sprints*, la planificación temporal del proyecto, siguiendo sus respectivas tareas y representando el flujo seguido para la creación y puesta en producción de la aplicación, así como la redacción de esta documentación del proyecto.

Cada tarea comprendía una historia de usuario o una tarea interna de pequeño tamaño, cuya completación comprende alrededor de un día de duración.

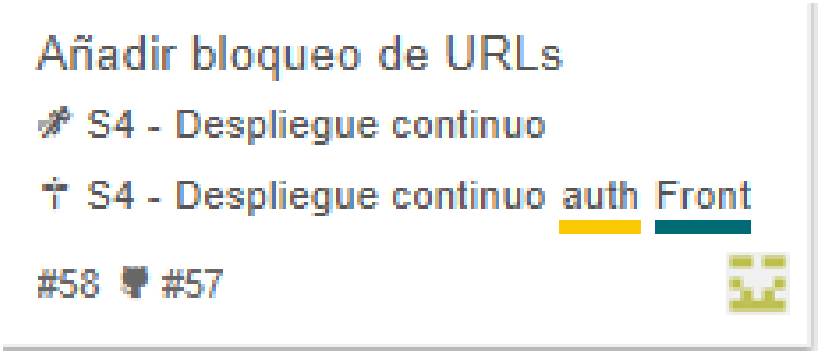

Figura A.3: Ejemplo de una tarea vista en el tablero Kanban de Zube

Los *sprints*, por su parte han tenido una duración media de quince días, a excepción del primero, pues requirió de varias semanas acordar las bases teóricas y objetivos principales de la aplicación.

Figura A.4: Ejemplo de un Sprint en Zube

La gestión de las tareas de cada *sprint* ha sido posible gracias a la herramienta del tablero KanBan, que permite gestionar de forma ágil y visual el proceso de desarrollo en el que se encuentra cada tarea. Una tarea se crea y pasa a estado *ready*, preparada para ser comenzada. Una vez esta se empiece a desarrollar, la tarea pasa al estado *in progress*. Una vez terminado el progreso pasará al estado de *in review*, donde se revisará nuevamente el contenido de la tarea para verificar que sea correcto y esté bien implementado, y, en caso positivo, finalizar en el estado *done*, como se puede ver en la siguiente imagen.

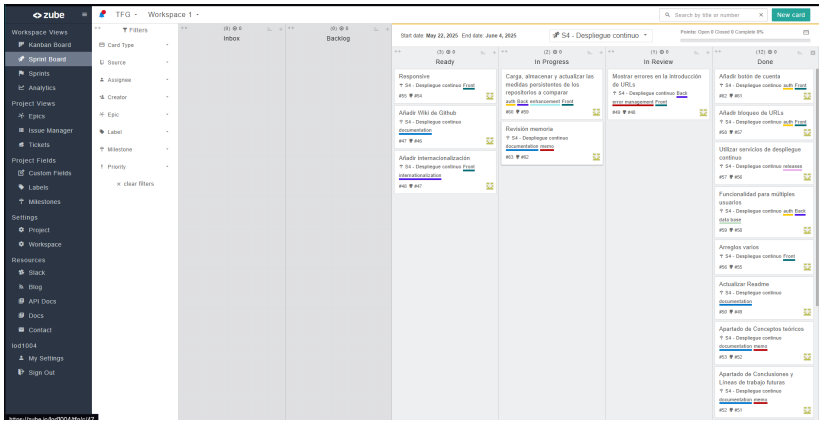

Figura A.5: Ejemplo del estado de un sprint visto en el tablero Kanban

La herramienta de Zube no solo ha permitido el seguimiento del desarrollo del proyecto en tareas, sino que también ha facilitado la comunicación entre el tutor y el estudiante autor de este TFG, por medio de los comentarios disponibles en cada tarea. Al terminar cada tarea se ha puesto un comentario en la misma indicando los resultados de las actividades asociadas, en ocasiones con imágenes representativas, y el commit que los subió al repositorio del proyecto. Esto ha permitido al tutor exponer su valoración del resultado de la tarea, puntos a mejorar y/o aspectos que necesiten cambios, como se puede ver a continuación:

Figura A.6: Ejemplo de comentarios en Zube del tutor y alumno en una tarea del proyecto

Organización en *sprints*

Para finalizar esta sección se van a desarrollar individualmente los *sprints* definidos y seguidos durante el desarrollo software. Estos *sprints* han sido pensados y creados durante el propio desarrollo en función del estado en el que se encontraba el proyecto, y los objetivos más adecuados que debían ser implementados en cada momento.

Sprint 0 - Kick-off (1/03/2025 - 03/04/2025)

Este sprint fue el primero y más largo de todos, pues abarcaría no solo el comienzo del desarrollo del proyecto, sino también la planificación del mismo.

Se definió qué aplicación se desarrollaría, sus bases teóricas, el *abstract* y los objetivos principales. Contiene las siguientes 9 tareas:

1. **Definir título y descripción del proyecto**: Consistió en, tras terminar de decidir los objetivos y bases del proyecto, redactar el título y descripción iniciales del mismo.

2. **Introducción del documento**: Redacción inicial del apartado de introducción de la memoria, planteando el contexto y motivación del proyecto.

3. **Revisión del resumen**: Corrección y mejora del resumen del proyecto, afinando su redacción y adecuación a los objetivos propuestos.

4. **Anexos**: Preparación de la estructura base de los anexos que acompañarán a la memoria del proyecto.

5. **Mockups**: Diseño preliminar de la interfaz de usuario para visualizar cómo se estructuraría la aplicación de forma gráfica.

6. **Definir requisitos funcionales**: Identificación de las funciones clave que debía cumplir la aplicación para alcanzar sus objetivos.

7. **Rama de pruebas**: Creación de una rama en el repositorio destinada a pruebas y experimentación durante el desarrollo que se usaría en este sprint.

8. **Estructura del front**: Primer esqueleto del frontend de la aplicación, estableciendo la organización interna de los componentes básicos.

9. **Estructura del back**: Configuración inicial del backend y definición de su arquitectura básica.

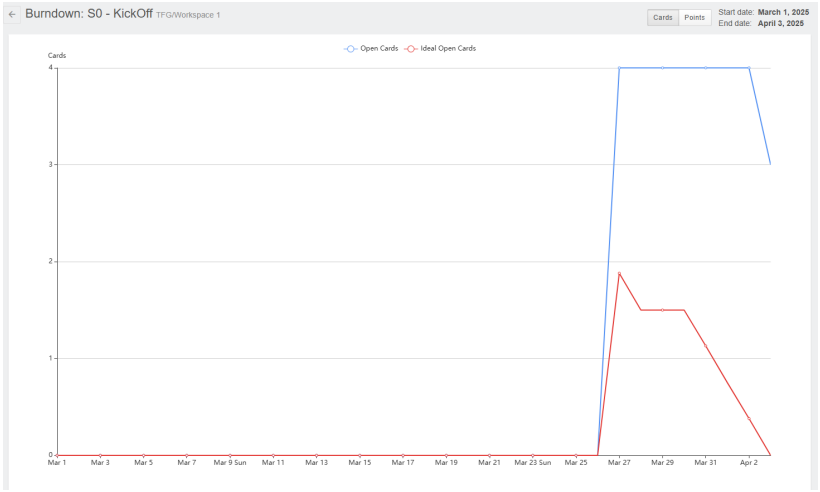

Figura A.7: Gráfico Burndown del Sprint 0

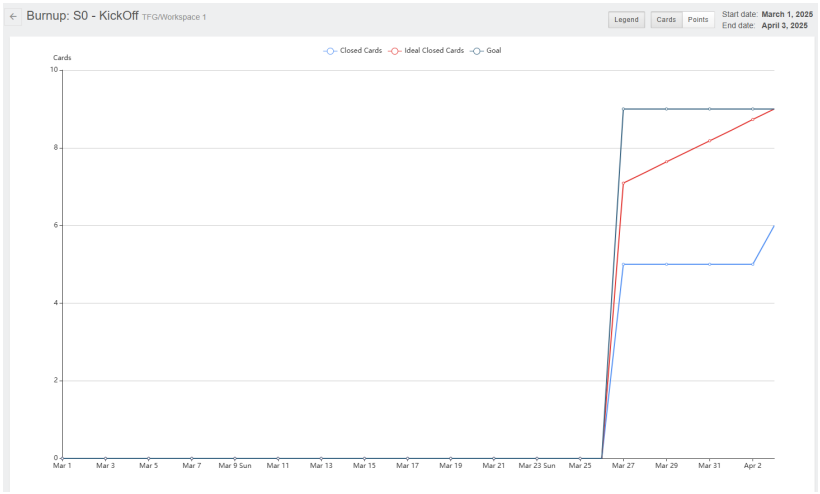

Figura A.8: Gráfico Burnup del Sprint 0

Sprint 1 - Primer prototipo de la aplicación (8/04/2025 - 22/04/2025)

Instaladas las herramientas básicas del proyecto y asentadas las ideas del mismo, comienza el desarrollo con un pequeño prototipo de la aplicación. Se perseguían los siguientes objetivos: conectar el backend con el frontend, definir las reglas que se evaluarán, continuar con la documentación y realizar los primeros análisis de datos.

1. **Conexión Front - Back**: Establecimiento de la comunicación entre el frontend y el backend.

2. **Recogida de Repositorio vía URL**: Implementación de una funcionalidad para introducir un repositorio mediante su URL.

3. **Guardar repositorio en base de datos**: Añadir la funcionalidad de almacenamiento de repositorios para su análisis posterior.

4. **Recoger número de Issues abiertas y cerradas**: Desarrollo de una métrica base para evaluar las *issues* del repositorio.

5. **Mostrar número de Issues**: Visualización en la interfaz del número de *issues*, como parte de las estadísticas presentadas.

6. **Evaluar mensajes de los commits**: Análisis del contenido de los mensajes de los commits para detectar buenas prácticas.

7. **Evaluar Issues en el tiempo**: Cálculo de métricas temporales sobre la evolución de los *issues*.

8. **Evaluar Pull Requests**: Análisis de la actividad de las *pull requests* como indicador de colaboración.

9. **Evaluar Releases**: Inclusión de información sobre las versiones liberadas del proyecto.

10. **Evaluar Acciones**: Estudio de las GitHub Actions empleadas como parte del flujo de trabajo.

11. **Definir reglas**: Redacción y codificación de las reglas de evaluación de buenas prácticas basadas en metodología ágil.

Figura A.9: Gráfico Burndown del Sprint 1

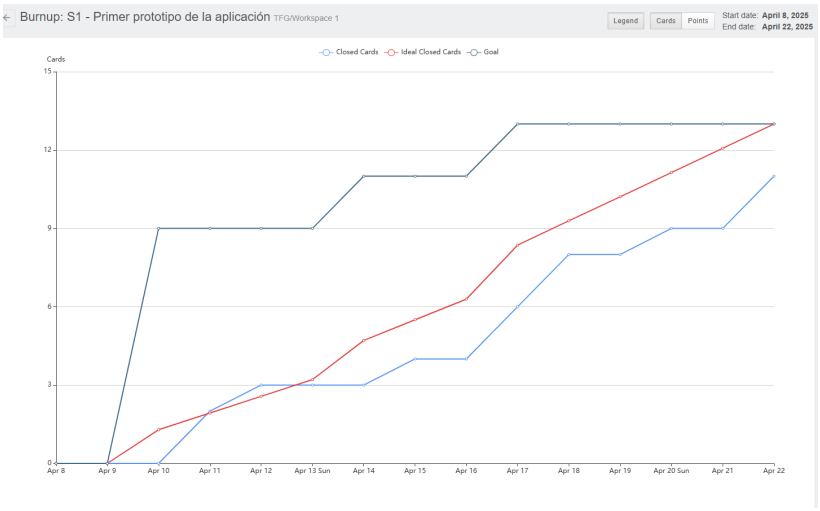

Figura A.10: Gráfico Burnup del Sprint 1

Sprint 2 - Comparación temporal (24/04/2025 - 08/05/2025)

La aplicación ya obtiene estadísticas básicas. En este sprint se centró en implementar una comparación temporal entre repositorios, añadir control de errores y documentar la evolución temporal de los proyectos.

1. **Crear funciones de las reglas**: Implementación concreta de las funciones para evaluar las reglas definidas.

2. **Mostrar cumplimiento de reglas**: Inclusión en el frontend de indicadores que muestran si se cumplen las reglas.
3. **Extraer intervalos de tiempo del repositorio**: Implementar la funcionalidad de los intervalos de tiempo para comparar la actividad del repositorio.
4. **Selección de intervalos de tiempo**: Interfaz para que el usuario pueda seleccionar intervalos de tiempo a usar en el análisis.
5. **Añadir estadísticas temporales**: Implementación de métricas que reflejen la evolución cada cierto número de días.
6. **Añadir estadísticas restantes**: Inclusión de estadísticas que aún no habían sido implementadas.
7. **Estadísticas de participantes**: Incorporación de datos sobre los miembros del repositorio.
8. **Ordenar Base de datos**: Refactorización y optimización de la estructura de almacenamiento de repositorios y medidas de calidad de proceso.
9. **Añadir control de errores**: Añadir gestión de errores para situaciones como problemas de red o repositorios no válidos.
10. **Documentar sobre la evolución de proyectos en el tiempo**: Redacción teórica en la memoria sobre las fases de un proyecto ágil.

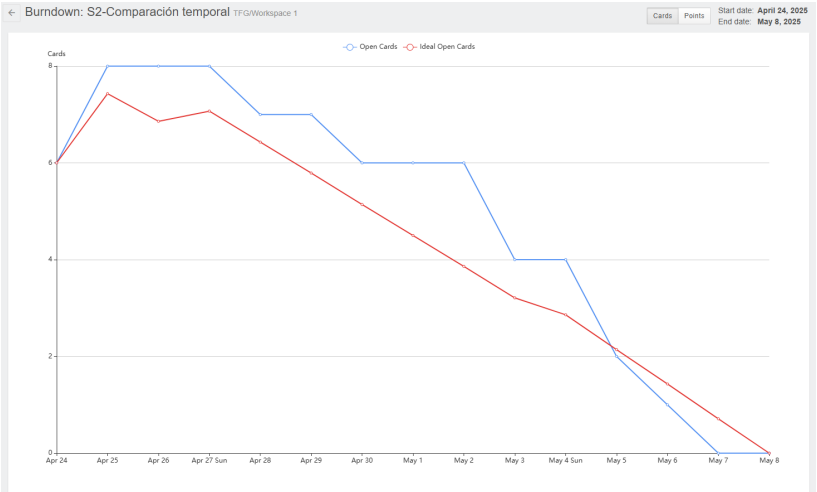

Figura A.11: Gráfico Burndown del Sprint 2

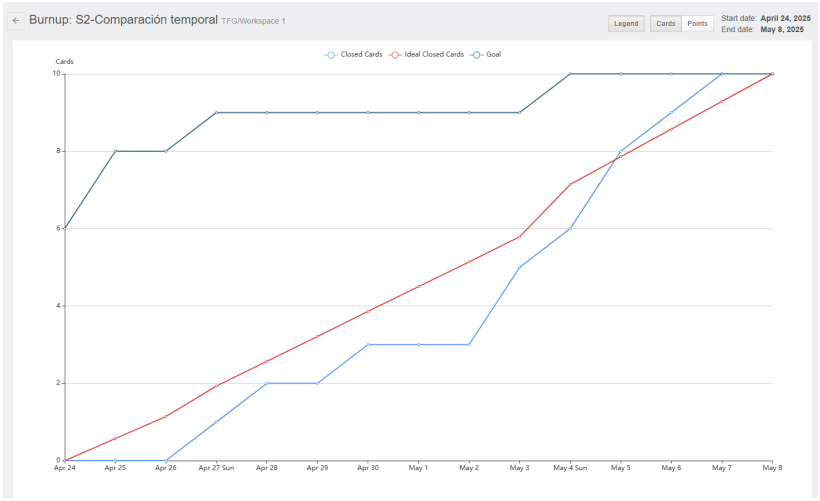

Figura A.12: Gráfico Burnup del Sprint 2

Sprint 3 - Diseño de la estética de la interfaz de usuario (08/05/2025 - 21/05/2025)

Con una versión funcional disponible, se procedió a mejorar la experiencia de usuario y seguir avanzando en la documentación del proyecto.

1. **Maquetación del FrontEnd**: Rediseño visual del frontend para mejorar la experiencia y estética.
2. **Nuevas estadísticas y revisión**: Mejora de algunas métricas, añadido de alguans nuevasy validación de los resultados obtenidos.
3. **Añadir Spinner**: Mejora visual mediante un componente que indica la carga de información.
4. **Implementar logs**: Sistema de registro de eventos para facilitar el mantenimiento.
5. **Información sobre resultados de las reglas**: Mostrar explicaciones detalladas de cada regla y su resultado.
6. **Apartado de "Trabajos Relacionados"**: Redacción del apartado que contextualiza el trabajo con investigaciones previas.
7. **Apartado de .ªspectos relevantes del desarrollo"**: Inclusión en la memoria de las decisiones clave durante el desarrollo.

8. **Apartado de .ºbjetivos del proyecto"**: Definición clara de los objetivos concretos que se querían alcanzar.
9. **Apartado de "Técnicas y herramientas"**: Descripción de las herramientas usadas y su justificación.
10. **Primera Release del proyecto**: Publicación de la primera versión oficial v0.1.0 de la aplicación.

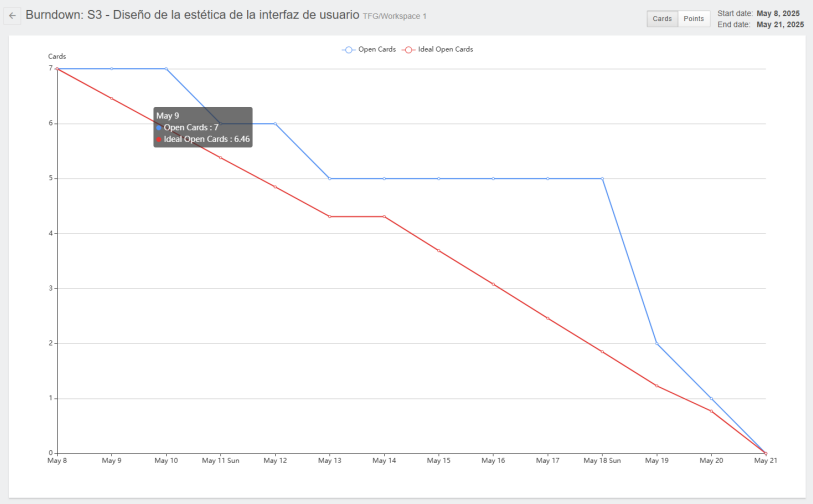

Figura A.13: Gráfico Burndown del Sprint 3

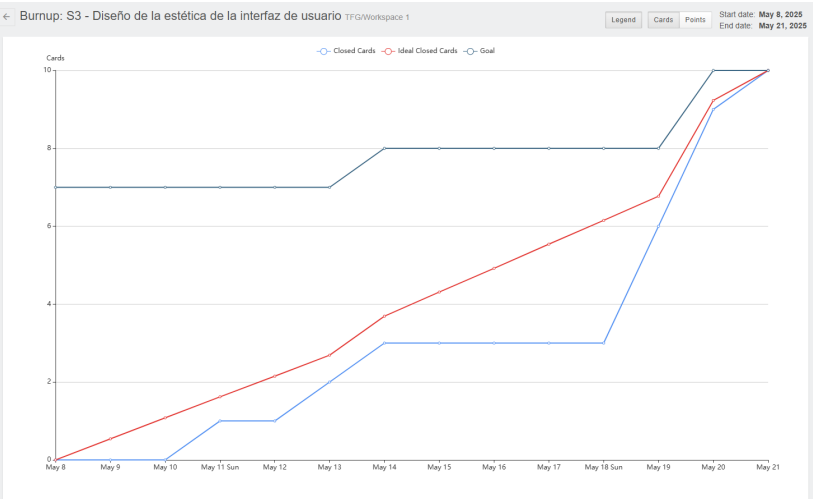

Figura A.14: Gráfico Burnup del Sprint 3

Sprint 4 - Despliegue continuo (22/05/2025 - 04/06/2025)

En este último sprint se trabajó en el despliegue de la aplicación, correcciones finales, soporte para múltiples usuarios y finalización de la memoria.

1. **Utilizar servicios de despliegue continuo**: Implementación de un flujo de despliegue automático tras cada actualización.
2. **Responsive**: Adaptación de la interfaz a distintos tamaños de pantalla.
3. **Mostrar errores en la introducción de URLs**: Validación del input del usuario para evitar URLs no válidas.
4. **Añadir internacionalización**: Soporte para diferentes idiomas en la interfaz de usuario.
5. **Añadir página de inicio**: Creación de una landing page para la aplicación que serviría como formulario de login.
6. **Cambiar título y nomenclaturas**: Revisión de nombres de variables y componentes para mayor claridad.
7. **Actualizar Readme**: Actualización de la documentación principal del repositorio con instrucciones claras.
8. **Añadir Wiki de Github**: Creación de una wiki complementaria al README para documentar el uso.
9. **Añadir usuarios y login**: Implementación básica de autenticación de usuarios.
10. **Funcionalidad para múltiples usuarios**: Adaptación del backend para permitir análisis separados por usuario.
11. **Añadir botón de cuenta**: Crear un botón de apertura del menú de usuario.
12. **Revisión memoria**: Revisión general del contenido escrito de la memoria.
13. **PDF completo de la memoria**: Generación del documento final completo en formato PDF.
14. **Apartado de Conclusiones y Líneas de trabajo futuras**: Reflexión final sobre el proyecto y posibles continuaciones en la memoria.

15. **Apartado A de los anexos**: Redacción del apartado A de los anexos.

16. **Revisión del apartado B de los anexos**: Revisión del apartado B de los anexos.

17. **Apartado C de los anexos**: Redacción del apartado C de los anexos.

18. **Apartado D de los anexos**: Redacción del apartado D de los anexos.

19. **Apartado E de los anexos**: Redacción del apartado E de los anexos.

20. **Apartado F de los anexos**: Redacción del apartado F de los anexos.

21. **Anexos (finales)**: Inclusión de todos los anexos restantes con los elementos complementarios.

22. **Test eficiencia v.0.2.0**: Evaluación de la eficiencia del sistema tras la última versión.

23. **Arreglos varios**: Correcciones menores y ajustes visuales o funcionales.

Figura A.15: Gráfico Burndown del Sprint 4

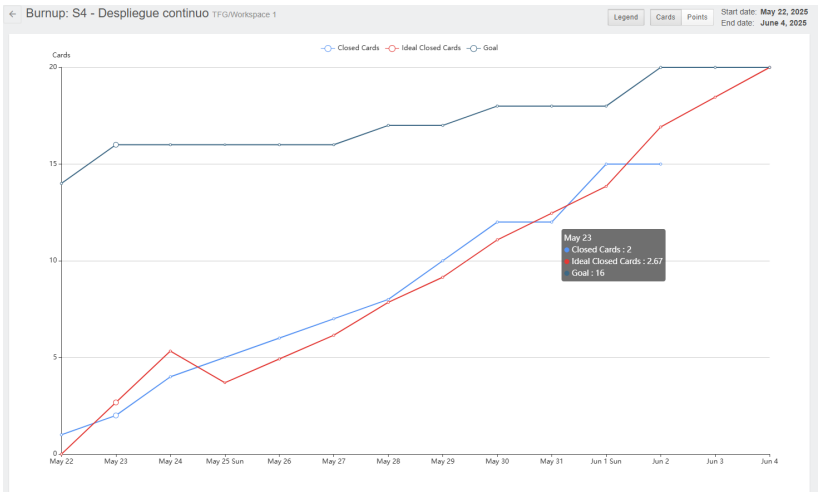

Figura A.16: Gráfico Burnup del Sprint 4

A.3. Estudio de viabilidad

Este apartado recoge un análisis detallado sobre la viabilidad del proyecto desde una doble perspectiva: económica y legal. Se examinan los recursos necesarios para el desarrollo de la aplicación web **Asistente de prácticas ágiles para repositorios en GitHub** en equipos de desarrollo que utilizan GitHub, así como los requisitos y restricciones legales que podrían condicionar su puesta en marcha. Dado su enfoque académico, automatizado y extensible, el proyecto se plantea como una herramienta útil tanto para entornos académicos como profesionales, y su desarrollo es asumido por un único responsable técnico.

Viabilidad económica

Costes de personal

El desarrollo ha sido asumido por una sola persona con perfil de estudiante de ingeniería informática. De acuerdo con el XVIII Convenio Colectivo Estatal de Empresas de Consultoría y Tecnologías de la Información [**?**], se estima un coste bruto mensual en torno a 2.000 €. Para el desarrollo completo del proyecto se contemplan hasta 4 meses de trabajo.

Concepto	Coste
Salario mensual	1.500€
Seguridad Social y retenciones	500€
Total 4 meses	**8.000€**

Tabla A.1: Costes de *personal*

Coste de hardware

Se presupone la utilización de un equipo de sobremesa ya disponible con prestaciones suficientes para ejecutar herramientas de desarrollo, e instalar aplicaciones y herramientas de programación y puesta en producción, así como entornos de inferencia local. A efectos contables, se considera una amortización del equipo a 3 años.

Concepto	Coste	Coste amortizado
Ordenador de desarrollo	500€	113€
Teléfono móvil para pruebas para interfaces responsive	250€	83€
Total	**750€**	**196€**

Tabla A.2: Costes de *hardware*

Coste de conectividad

El uso continuo de internet es fundamental, tanto para descargar dependencias como para utilizar APIs de inferencia o conectarse a servicios de terceros como GitHub, Vercel o Render. Se estima un coste medio mensual de 35€.

- **Total en conectividad (4 meses):** 140€

Coste de software

Todo el software utilizado en el desarrollo es de código abierto o con licencias gratuitas para su uso individual: Visual Studio Code, GitHub, Vercel (usando entornos públicos gratuitos) y React. El sistema operativo

empleado es Windows 10 Home, de un coste de alrededor de 10 euros y ampliamente compatible con las herramientas mencionadas.

Coste estimado: 10€

Auditoría y supervisión académica

Se reserva un coste temporal asociado al seguimiento del proyecto por parte del tutor, que incluye la corrección del código, revisión de entregables y reuniones de orientación técnica.

Concepto	**Coste**
Mano de obra	8.000€
Hardware amortizado	196€
Conectividad	140€
Software Windows 10	10€
Revisión académica	550€
Total	**9.306€**

Tabla A.3: Costes totales del asistente inteligente para prácticas ágiles

Sostenibilidad económica del proyecto

Una vez completado el desarrollo inicial, el coste de mantenimiento es bajo, especialmente si se utilizan modelos locales. La monetización del asistente podría basarse en:

- **Licencia freemium:** acceso básico gratuito y funciones avanzadas bajo suscripción.
- **Integración en GitHub Marketplace:** cobro por instalación en organizaciones.
- **Modelo educativo:** cesión gratuita a universidades con apoyo institucional.

Viabilidad legal

Licencias del software utilizado

Se ha garantizado el uso exclusivo de herramientas de código abierto o con licencias permissivas Esto incluye tanto las librerías públicas usadas en el frontend y el backend como herramientas de despliegue continuo como Vercel o React

- Librearías de Angular y Node-js.
- Integraciones continuas con GitHub, Vercel y React.

Términos de uso de las APIs de terceros

Al utilizar servicios de inferencia por API (En este caso GitHub API) deben respetarse las políticas de uso responsable, límites de almacenamiento de datos y uso no comercial sin licencia explícita.

Requisitos:

- Consentimiento del usuario en el tratamiento de repositorios.
- Prohibición del reentrenamiento sobre inputs sin autorización.

Protección de datos

En caso de desplegar el asistente en organizaciones reales, será necesario cumplir con la normativa de protección de datos como el RGPD:

- Anonimización de inputs del usuario.
- Consentimiento explícito para recoger métricas.
- Política de privacidad visible.

Restricciones de publicación

Si el asistente se publica en plataformas como GitHub Marketplace o se distribuye como SaaS, deberá incluir:

- Condiciones de uso claras.
- Política de privacidad.
- Declaración de uso de APIs externas.

Checklist legal

1. Verificación de licencias de todas las librerías.
2. Inclusión de política de privacidad si se recoge algún dato.
3. Revisión de los términos de cada API externa usada.
4. Consentimiento de usuario informado en contextos reales.
5. Uso responsable de la inferencia (sin outputs automatizados sin revisión humana en contextos críticos).

Cumpliendo estas condiciones, la legalidad de la aplicación está asegurada y no representa impedimentos para su desarrollo ni distribución futura.

Apéndice B

Especificación de Requisitos

B.1. Introducción

Este apéndice recoge los requisitos funcionales de la aplicación propuesta en este proyecto para evaluar la adopción de prácticas ágiles en repositorios de GitHub utilizados por estudiantes para realizar diversos trabajos como, por ejemplo, Trabajos de Fin de Grado (TFG). Se incluyen los casos de uso posibles que podrá tener un usuario de la aplicación desarrollada, junto a tablas y diagramas que ayuden a comprender los mismos. Los casos de uso comprenderán cierto número de requisitos funcionales y no funcionales, listados y explicados más adelante en este apartado para especificar clárame nte todas las formas que tiene el usuario de interactuar con el software propuesto. Además se incluirá el listado de objetivos generales del proyecto para ayudar a contextualizar tanto los casos de uso como los requisitos funcionales.

B.2. Objetivos generales

La visión general de este proyecto abarca los objetivos descritos a continuación:

- **Análisis de métricas de calidad de proceso de software de los repositorios**: La aplicación web analiza detalladamente todas las métricas de calidad de proceso y características que pueden resultar útiles de un repositorio durante el proceso de desarrollo de software.

- **Ajuste temporal del análisis**: Consiste en la posibilidad de ajustar distintos parámetros termporales del análisis para elegir en qué etapas del desarrollo se analizarán los repositorios.
- **Comparación de las métricas de calidad de proceso obtenidas de los repositorios de referencia**: La aplicación es capaz de, dadas todas las medidas de calidad de proceso de los repositorios que el usuario ha elegido como referencia, compararlas para ofrecer al usuario una visualización clara de los diferentes valores de estas medidas en los distintos repositorios.
- **Prácticas ágiles y medidas de repositorios**: La aplicación utiliza las definiciones de prácticas ágiles recogidas en [**?**] para presentar los resultados al usuario tras analizar el repositorio.
- **Asistencia a estudiantes**: Se pueden utilizar otros TFGs púplicos realizados en la Universidad de Burgos como casos de estudio, introduciéndolos en la aplicación y que estos sirvan como referencia para los estudiantes a la hora de realizar sus propios TFG.

B.3. Catálogo de requisitos funcionales

A continuación se puede observar un listado de todos los requisitos funcionales considerados para la aplicación realizada en este proyecto.

- **RF-01** – Creación una cuenta para usar la aplicación.
- **RF-02** – Entrada del perfil creado iniciando sesión.
- **RF-03** – Salida del perfil creado cerrando sesión.
- **RF-04** – Cambio de la contraseña del perfil de la cuenta creada.
- **RF-05** – Cambio del idioma de la interfaz de usuario de la aplicación.
- **RF-06** – Introducción y validación de URLs de repositorios.
- **RF-07** – Elección del número de días para realizar el cálculo de las medidas de calidad temporales.
- **RF-08** – Elección del ámbito termporal de extracción de medidas de calidad de proceso de los repositorios.

- **RF-09** – Elección del tipo de intervalos de tiempo a usar para extraer las medidas de calidad de proceso de los repositorios introducidos.
- **RF-10** – Ajuste de los intervalos de tiempo específicos para elegir de qué momento en el tiempo de vida de los repositorios se extraerán las medidas de calidad de proceso.
- **RF-11** – Carga de grupos de repositorios usados como fuente de referencia en la aplicación con anterioridad.
- **RF-12** – Borrado de grupos de repositorios usados como fuente de referencia en la aplicación con anterioridad.
- **RF-13** – Visualización de la evaluación de la aplicación de prácticas ágiles por parte del repositorio a evaluar.
- **RF-14** – Visualización de la comparación de los valores numéricos de las medidas de calidad de proceso entre el repositorio a analizar y los repositorios de referencia.

B.4. Especificación de requisitos funcionales

A continuación se describen de forma detallada los requisitos funcionales del sistema.

RF-01 – Creación de una cuenta de usuario

El sistema debe permitir a los usuarios crear una cuenta personal mediante un formulario de registro. Esta funcionalidad es necesaria para garantizar la personalización de la experiencia, el almacenamiento de configuraciones y el acceso a funcionalidades avanzadas, como la carga de repositorios de referencia o la consulta del historial de análisis realizados; además de garantizar seguridad y control de acceso a la aplicación.

RF-02 – Inicio de sesión

Los usuarios registrados deben poder iniciar sesión en el sistema para acceder a sus perfiles y funcionalidades asociadas mediante un nombre de usuario único y una contraseña que se almacenará hasheada para garantizar la seguridad del almacenaje de perfiles de usuario en la base de datos.

RF-03 – Cierre de sesión

Los usuarios pueden cerrar sesión de forma segura al finalizar su uso, tanto para finalizar de forma segurasu actividad o cambiar de cuenta por diferentes motivos como el uso de varias cuentas personalizadas con diferentes ajustes y/o grupos de repositorios almacenados.

RF-04 – Cambio de contraseña

El sistema debe permitir a los usuarios modificar la contraseña de su cuenta en cualquier momento, con el fin de mantener la seguridad y el control sobre su acceso. El proceso debe incluir medidas de validación para evitar contraseñas débiles o inseguras.

RF-05 – Cambio de idioma de la interfaz

La interfaz de la aplicación debe ser multilingüe, permitiendo a los usuarios seleccionar el idioma que prefieran entre las opciones disponibles. Esto contribuye a la internacionalización del software, mejora la accesibilidad y facilita su uso en contextos internacionales o multilingües.

RF-06 – Introducción y validación de URLs de repositorios

El sistema debe permitir a los usuarios introducir la URL de uno o varios repositorios públicos de GitHub. La aplicación validará que las URLs introducidas sean correctas, que los repositorios estén disponibles y que se pueda acceder a ellos para su análisis para infromar al usuario de que está introduciendo repositorios válidos y de manera correcta.

RF-07 – Selección del número de días para métricas temporales

El sistema debe permitir al usuario definir un número de días a considerar como ventana temporal para el cálculo de medidas de calidad de proceso relacionadas con medias. Esta opción permite realizar análisis adaptados a periodos de mayor o menor longitud de actividad dentro del proyecto.

RF-08 – Selección del ámbito temporal del análisis

Los usuarios podrán definir si los repositorios se analizarán al completo, lo cual abarcaría todo su tiempo de vida desde elñ primer *commit* hasta el último, o por intervalos de tiempo ajustables.

RF-09 – Selección del tipo de intervalos de tiempo

El sistema debe ofrecer la posibilidad de seleccionar el tipo de intervalo temporal para el análisis, ya sean intervalos relativos, que abarquen cuartos de la vida del repositorio, o intervalos absolutos que equivalgan a meses de vida. Esta opción facilita la visualización de tendencias y eficiencia de trabajo en distintas fases del desarrollo de software.

RF-10 – Ajuste manual de intervalos de tiempo

Además de seleccionar el tipo de intervalo, el usuario podrá definir manualmente los intervalos exactos de tiempo que desea analizar (Por ejemplo el primer y segundo cuarto, lo cual es equivalente a la primera mitad del repositorio para los intervalos relativos, o del tercer al sexto mes de vida del repositorio para los intervalos absolutos). Esta funcionalidad avanzada permite focalizar el análisis en momentos clave del desarrollo del proyecto.

RF-11 – Carga de grupos de repositorios de referencia

La aplicación debe permitir al usuario cargar grupos de repositorios previamente definidos como referencia. Esto permite al usuario repetir comparaciones o realizar comparaciones muy similares evitando el tiempo de extracción de medidas de calidad de proceso.

RF-12 – Eliminación de grupos de referencia cargados*

El usuario también debe tener la posibilidad de eliminar grupos de repositorios de referencia previamente cargados, con el fin de mantener organizada su área de trabajo y adaptar el análisis a nuevos contextos o criterios de comparación.

RF-13 – Visualización de la evaluación de prácticas ágiles

Una vez realizado el análisis, el sistema debe mostrar una evaluación del repositorio analizado en función de su cumplimiento de buenas prácticas ágiles. Esta evaluación debe representarse de forma clara y comprensible en forma de reglas que indiquen visualmente en qué aspectos se usan de forma sólida las metodologías ágiles y en qué aspectos hay opción de mejora.

RF-14 – Comparación con repositorios de referencia

La aplicación debe permitir comparar los valores de todas las métricas extraídas del repositorio analizado con los correspondientes valores de los repositorios de referencia seleccionados. Esta comparación se visualizará mediante listas que ayuden a respaldar la evaluación del uso de prácticas ágiles en los repositorios.

B.5. Catálogo de requisitos no funcionales

A continuación se puede observar un listado de todos los requisitos no funcionales considerados para la aplicación realizada en este proyecto.

- **RNF-01** Modularidad
- **RNF-02** Usabildiad
- **RNF-03** Reusabilidad
- **RNF-04** Separación entre frontend y backend
- **RNF-05** Validaciones informativas
- **RNF-06** Navegación intuitiva

B.6. Especificación de requisitos no funcionales

A continuación se describen de forma detallada los requisitos no funcionales del sistema.

RNF-01 – Modularidad

La aplicación debe estar diseñada utilizando una arquitectura modular basada en componentes reutilizables. Debe ser comparible con patrones de diseño modernos y fácil de escalar o modificar por partes sin afectar al conjutno total del software. Esta estructura permite que futuras modificaciones o ampliaciones (como la adición de nuevas métricas o funcionalidades) se integren de forma controlada y sencilla.

RNF-02 – Usabildiad

La interfaz debe ser intuitiva, clara y fácil de usar. Los formularios deben presentar instrucciones comprensibles, validaciones dinámicas y retroalimentación inmediata ante errores o acciones del usuario. Se prioriza un diseño simple y directo que permita a usuarios no técnicos comprender la aplicación sin curva de aprendizaje.

RNF-03 – Reusabilidad

El código debe seguir principios como DRY (Don't Repeat Yourself) y SOLID, en particular el principio Open/Closed, que facilita la extensión de funcionalidades sin modificar el núcleo existente. La organización del código promueve la reutilización de servicios, componentes y lógica, favoreciendo la eficiencia en el desarrollo y la reducción de errores.

RNF-04 –Separación entre frontend y backend

El sistema debe mantener una separación estricta entre las capas de presentación (frontend) y lógica de negocio (backend), utilizando servicios HTTP para el intercambio de datos de forma asíncrona. Esto mejora la escalabilidad y permite gestionar respuestas y errores de forma controlada desde la interfaz.

RNF-05 – Validaciones informativas

Los formularios deben incorporar validación reactiva utilizando sistemas de formularios y alertas, proporcionando retroalimentación inmediata al usuario en caso de errores o datos incompletos. Esto reduce la frustración del usuario y mantiende informado al mismo, y además mejora la precisión y seguridad de los datos introducidos.

RNF-06 – Navegación intuitiva

La aplicación debe organizar sus funcionalidades de forma jerárquica, utilizando tabs, rutas y botones de navegación claros que guíen al usuario durante el análisis. Los componentes de navegación deben ser reutilizables y consistentes en toda la aplicación para facilitar el acceso rápido a las diferentes secciones.

B.7. Casos de uso

En esta sección se expondrán los diferentes casos de uso que recogen los requisitos ya definidos anteriormente para cumplir con los objetivos de la aplicación establecidos.

CU-1	**Creación y acceso a la cuenta**
Versión	1.0
Autor	Lucas Olmedo Diéz
Requisitos asociados	RF-01, RF-02
Descripción	Permite al usuario crear una cuenta y acceder a la aplicación mediante inicio de sesión.
Precondición	El usuario debe introducir un usuario único y una contraseña válida
Acciones	1. El usuario accede a la página de registro o inicio de sesión. 2. El usuario introduce los datos necesarios (usuario y contraseña). 3. El sistema verifica la validez de los datos. 4. El sistema registra al usuario o permite el acceso según corresponda.
Postcondición	El usuario ha creado una cuenta o ha accedido a su perfil.
Excepciones	Datos incorrectos o cuenta inexistente/ya registrada.
Importancia	Alta

Tabla B.1: CU-1 Creación y acceso a la cuenta.

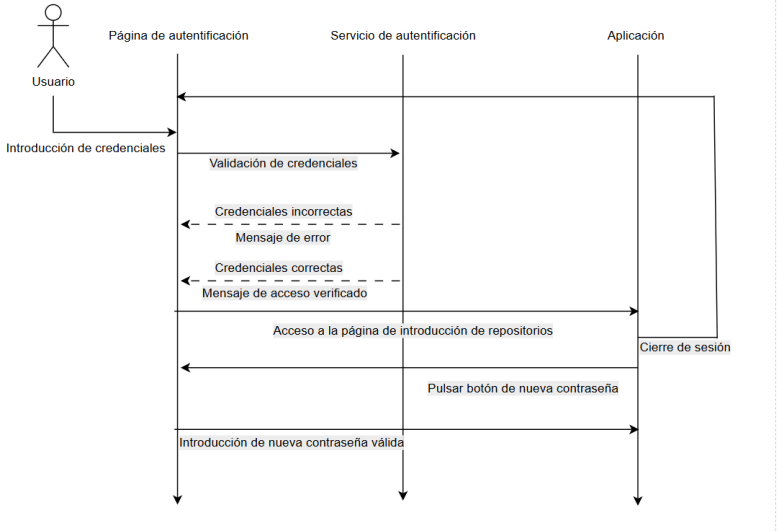

Figura B.1: Diagrama de los Casos de Uso 1 y 2

CU-2	Configuración de la cuenta
Versión	1.0
Autor	Lucas Olmedo Diéz
Requisitos asociados	RF-02, RF-03
Descripción	Permite al usuario cambiar la contraseña y modificar el idioma de la aplicación.
Precondición	El usuario debe haber iniciado sesión.
Acciones	1. El usuario accede a la sección de configuración. 2. Selecciona cambiar contraseña o idioma. 3. El sistema valida y aplica los cambios.
Postcondición	Se actualizan las preferencias del usuario.
Excepciones	
Importancia	Media

Tabla B.2: CU-2 Configuración de la cuenta.

CU-3	Introducción y validación de URLs de repositorios
Versión	1.0
Autor	Lucas Olmedo Diéz
Requisitos asociados	RF-04
Descripción	Permite al usuario introducir URLs de repositorios GitHub y valida su formato y accesibilidad.
Precondición	El usuario ha iniciado sesión.
Acciones	1. El usuario introduce una o varias URLs. 2. El sistema valida la estructura y el acceso. 3. Se informa al usuario del resultado.
Postcondición	Se aceptan o rechazan las URLs introducidas.
Excepciones	URL malformada o repositorio no accesible.
Importancia	Alta

Tabla B.3: CU-3 Introducción y validación de URLs de repositorios.

CU-4	Configurar análisis del repositorio
Versión	1.0
Autor	Lucas Olmedo Díez
Requisitos asociados	RF-05, RF-06, RF-07, RF-08
Descripción	Permite al usuario configurar cómo se analizarán los datos del repositorio: intervalos absolutos o relativos, número de días para calcular las métricas temporales, períodos de los intervalos.
Precondición	El usuario debe haber elegido la opción de .ªnalizar por intervalos de tiempo".
Acciones	1. El usuario selecciona si desea usar fechas absolutas o relativas. 2. Ajusta manualmente los intervalos de fechas específicos. 3. Elige el tipo de intervalos para segmentar la información. 4. (Opcional) Se cambia el númerod e días usado para obtener métricas de calidad temporales.
Postcondición	El sistema ha almacenado la configuración de análisis para ser utilizada en el cálculo de métricas.
Excepciones	Intervalos inválidos, número de días no válido, configuración incompleta.
Importancia	Alta

Tabla B.4: CU-4 Configurar análisis del repositorio

CU-5	Gestionar grupos de repositorios de comparación
Versión	1.0
Autor	Lucas Olmedo Díez
Requisitos asociados	RF-09, RF-10
Descripción	Permite al usuario cargar o borrar grupos de repositorios previamente guardados para usarlos como referencia comparativa en los análisis.
Precondición	El usuario debe haber hecho algún análisis previo con el que se guardaron repositorios de referencia.
Acciones	1. El usuario accede a la sección de gestión de repositorios guardados. 2. Selecciona un grupo de repositorios guardado previamente y lo carga. 3. (Opcional) Elimina uno o más grupos guardados si lo desea.
Postcondición	Los grupos seleccionados han sido cargados o eliminados exitosamente.
Excepciones	No hay grupos guardados disponibles o error en la carga/borrado.
Importancia	Media

Tabla B.5: CU-5 Gestionar grupos de repositorios de comparación

CU-6	Visualizar evaluación de buenas prácticas ágiles
Versión	1.0
Autor	Lucas Olmedo Díez
Requisitos asociados	RF-11
Descripción	Permite al usuario visualizar el grado de adopción de buenas prácticas ágiles evaluadas automáticamente por el sistema a partir de los datos del repositorio.
Precondición	Se han introducido URLS válidad y parámetros del análisis correctos.
Acciones	1. El sistema muestra los resultados del análisis de buenas prácticas en formato gráfico y textual. 2. El usuario puede consultar qué reglas se cumplen, cuáles no, y por qué.
Postcondición	El usuario ha accedido a la información evaluada y entendible de las prácticas ágiles del repositorio.
Excepciones	No se han podido calcular algunas reglas por errores en añgún repositorio.
Importancia	Alta

Tabla B.6: CU-6 Visualizar evaluación de buenas prácticas ágiles

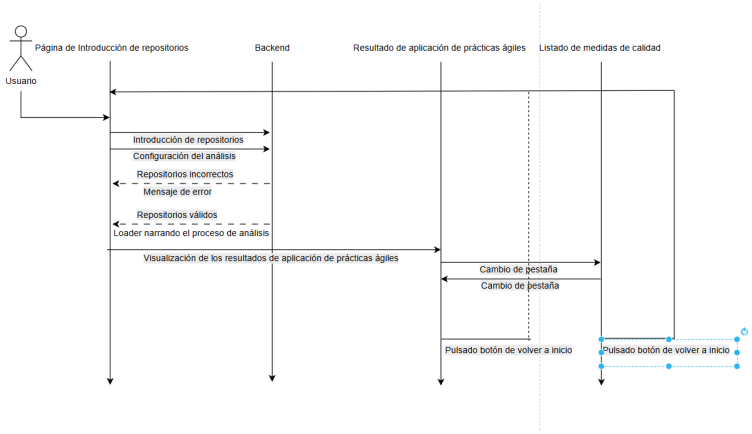

Figura B.2: Diagrama de los Casos de Uso 3, 4, 6 y 7

CU-7	Comparar medidas de calidad de proceso con repositorios de referencia
Versión	1.0
Autor	Lucas Olmedo Díez
Requisitos asociados	RF-12
Descripción	Permite comparar visualmente las métricas del repositorio analizado con las de los repositorios de referencia cargados.
Precondición	El repositorio analizado y al menos un grupo de referencia, junto a sus medidas de calidad deben estar cargados.
Acciones	1. El usuario accede a la sección de comparación. 2. El sistema genera y muestra listas comparativas de las métricas seleccionadas. 3. El usuario interpreta la comparación en función de los valores visualizados.
Postcondición	El usuario obtiene una visión comparativa clara del estado de su proyecto.
Excepciones	Error al cargar los datos de referencia o métricas incomparables.
Importancia	Alta

Tabla B.7: CU-7 Comparar con repositorios de referencia

CU-8	Navegar por la aplicación
Versión	1.0
Autor	Lucas Olmedo Díez
Requisitos asociados	RF-11, RF-03
Descripción	Permite al usuario desplazarse cómodamente por las diferentes secciones de la aplicación (login, análisis, comparación, configuración, etc.), garantizando una experiencia de usuario fluida.
Precondición	El usuario debe haber iniciado sesión. La aplicación debe estar desplegada correctamente.
Acciones	1. El usuario interactúa con los tabs y botones de navegación disponibles. 2. El sistema muestra el contenido correspondiente sin errores.
Postcondición	El usuario ha accedido correctamente a las diferentes funcionalidades sin fricciones.
Excepciones	Error de carga, o comportamiento inesperado.
Importancia	Alta

Tabla B.8: CU-8 Navegar por la aplicación

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño arquitectónico

C.4. Diseño procedimental

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

D.3. Manual del programador

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Apéndice F

Anexo de sostenibilización curricular

F.1. F.1. Introducción

Durante el desarrollo de este TFG, centrado en la creación de una aplicación web para el análisis automatizado de prácticas ágiles y métricas de calidad de proceso en proyectos alojados en GitHub, he aplicado diversas competencias de sostenibilidad de forma transversal. Este trabajo ha sido una oportunidad para reflexionar sobre cómo el desarrollo software puede contribuir de manera activa al cumplimiento de los Objetivos de Desarrollo Sostenible (ODS), especialmente al ods9, centrado en la industria, innovación e infraestructura, y al ods12.

El análisis del desarrollo software, combinado con el uso de las prácticas ágiles permiten fomentar procesos de mejora continua que reduzcan el desperdicio de recursos humanos y computacionales, a la vez que promueven modelos colaborativos, transparentes y sostenibles. La herramienta desarrollada contribuye a una mayor conciencia en torno a la eficiencia y responsabilidad dentro de los equipos de desarrollo software.

F.2. F.2. Competencias de Sostenibilidad en el proyecto

F.2.1. Adquisición de conocimiento de competencias de sostenibilidad

A través de este proyecto, he sido capaz de relacionar el desarrollo tecnológico con los retos globales de sostenibilidad. La aplicación se enfoca en facilitar el análisis reflexivo de proyectos software, lo que permite a los equipos evaluar sus prácticas y alinear sus metodologías con objetivos más amplios como la eficiencia, la justicia laboral (mediante la mejora del clima de equipo), o el acceso equitativo al conocimiento abierto.

F.2.2. Sostenibilidad al tomar decisiones de proyecto

He priorizado el uso eficiente de recursos en la arquitectura del sistema, optando por tecnologías open-source, integraciones ligeras con la API de GitHub para minimizar el consumo energético y computacional. Se evita la persistencia innecesaria de datos, lo que contribuye al uso responsable de almacenamiento y procesamiento en servidores.

F.2.3. Fomento de la participación colectiva

Este trabajo promueve el uso de métricas abiertas y replicables para evaluar proyectos en GitHub, lo que incentiva la participación colaborativa en comunidades de desarrollo software. Al facilitar la evaluación colectiva de buenas prácticas, la herramienta ayuda a fortalecer ecosistemas de software más transparentes, resilientes y responsables.

F.2.4. Ética del proyecto

Durante el desarrollo se ha seguido un enfoque ético que evita la recopilación innecesaria de datos personales, se promueve el análisis de repositorios públicos y se fomenta el uso responsable de la tecnología como instrumento para mejorar la transparencia y la calidad en el trabajo en equipo, sin fomentar modelos competitivos insostenibles o presiones laborales indebidas.

F.2.5. Conciencia sobre Sostenibilidad en el Desarrollo Software

Una parte clave del proyecto ha sido su valor educativo. La herramienta actúa como recurso de concienciación para estudiantes y desarrolladores que buscan mejorar sus prácticas en el desarrollo software y planificación para el mismo. Fomenta la reflexión sobre el ciclo de vida de los proyectos de software, sus tareas y componentes, la calidad de sus procesos y su sostenibilidad a largo plazo.

F.3. F.3. Conclusión

El desarrollo de este proyecto ha fortalecido mi comprensión de cómo el software no es únicamente una herramienta técnica, sino un elemento transformador con capacidad de influir en dimensiones sociales y económicas. Aplicar competencias de sostenibilidad en el diseño, implementación y propósito de esta herramienta me ha hecho más consciente de las decisiones éticas y técnicas que tomamos como desarrolladores software. Estoy convencido de que este tipo de soluciones tecnológicas son fundamentales para alcanzar un desarrollo sostenible.

Bibliografía