Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Ingeniería del Software

DDVault: Aplicación web como soporte a la definición y gestión de diccionarios de datos

---

Alumno: Johana Andrea Ramírez Figueroa

Tutor: Yania Crespo González-Carvajal

A quienes me inspiraron a seguir adelante

Agradecimientos

Me gustaría expresar mi agradecimiento a Yania, por haberme guiado durante todo este proceso. Cada vez que me he sentido perdida, ha sabido encaminarme. También agradezco infinitamente su paciencia y su actitud positiva.

Resumen

Un diccionario de datos documenta los metadatos más ligados a su almacenamiento en bases de datos. Incluye aspectos como la definición de cada campo, su tipo de dato, formato, longitud, posibles valores que puede tomar, reglas que debe cumplir en relación con otros campos e, incluso, transformaciones sufridas. Estos metadatos ayudan a los usuarios a entender los datos desde el punto de vista técnico para poder explotarlos adecuadamente. Por este motivo, cada base de datos debería contar con su diccionario de datos asociado.

En este trabajo se ha desarrollado una aplicación web que ayuda a organizaciones a gestionar la documentación de sus bases de datos mediante la creación de diccionarios de datos y la asignación de usuarios a roles específicos. Para su desarrollo, se han utilizado los frameworks Angular para el cliente y Spring Boot para el servidor. Además, se ha utilizado Scrum para la organización y el seguimiento del trabajo.

Abstract

A data dictionary stores metadata related to the structure and constraints of data in a database. It includes information such as field definitions, data types, formats, lengths, valid values, relational rules, and any transformations applied to the data. This metadata is essential for understanding the data from a technical perspective and ensuring its correct usage. Therefore, every database should be accompanied by a data dictionary.

This Final Degree Project presents a web application designed to support organizations in documenting their databases by facilitating the creation of data dictionaries and the assignment of users to specific roles. The application was developed using the Angular framework for the client side and Spring Boot for the server side. The Scrum framework was adapted and applied for project organization and task management.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto

Vivimos en la era de la información, su inicio está asociado a la revolución digital, la cual comenzó a mediados del siglo XX y continua a día de hoy. Esta revolución ha supuesto un cambio en el funcionamiento de la sociedad y de su economía [31].

La información es obtenida mediante la recolección de datos y su posterior procesamiento. Asimismo, esta información es transformada en conocimiento, generando un valor. Este proceso es la base de la economía del conocimiento [44].

La cantidad de datos disponibles es cada vez mayor y su procesamiento se va complicando. La productividad de una empresa está fuertemente ligada a su capacidad de gestionar los datos que posee. Es por esto que, actualmente, las empresas dependen del uso de programas informáticos que optimicen la gestión de los datos. Un diccionario de datos es un ejemplo de este tipo de programas.

Un **diccionario de datos** [4] funciona como una guía de referencia donde se puede consultar el significado de los datos que forman una base de datos, Dicho de otra manera, un diccionario de datos consiste en un conjunto de **metadatos** que describe la estructura de una base de datos, el significado de sus campos y las relaciones que existen entre los datos que la forman. Los diccionarios ayudan a que los datos sean fiables, consistentes y más fáciles de entender.

Algunos de los componentes que debe incluir un diccionario de datos son:

- Una lista de elementos con sus nombres y definiciones.
- Propiedades detalladas de los elementos, como el tipo de dato, el tamaño, la opcionalidad, etc.

- Relaciones entre los elementos.
- Reglas de negocio.
- Fecha y hora de creación y de modificación.

1.2. Motivación

Dentro de una organización puede haber diferentes equipos de trabajo o departamentos, cada uno con sus propias tareas. Sin embargo, aún tratándose de tareas distintas entre sí, probablemente compartan un contexto y es necesario una puesta en común de los términos a utilizar.

En el ámbito de desarrollo software, es importante que los desarrolladores compartan una nomenclatura común para evitar errores o posibles perdidas de información. Los diccionarios de datos son utilizados como repositorios centrales donde consultar la definición y las restricciones de los términos.

Es importante aclarar que no es lo mismo un catálogo de datos que un diccionario de datos. Un **catálogo de datos** [3] es un inventario que facilita la búsqueda de datos desde distintas fuentes y da información sobre ellos. A diferencia del catálogo, un diccionario de datos está pensado para almacenar metadatos técnicos en el contexto de una base de datos como, por ejemplo, el valor máximo que puede alcanzar un campo de tipo entero. Se podría decir que un diccionario de datos es una parte de un catálogo de datos.

1.3. Alternativas

Existen multitud de herramientas que posibilitan la documentación de bases de datos, algunas son complejas y se tratan más bien de catálogos de datos que de diccionarios. La elección de una herramienta u otra dependerá de sus características. Una empresa pequeña no necesitará una herramienta demasiado compleja ni tendrá un presupuesto muy elevado, mientras que una empresa grande preferirá tener varias herramientas unificadas, como un catálogo de datos.

A continuación, se enumerarán algunas de las alternativas más populares para la creación de diccionarios de datos, así como los pros y los contras de cada una de ellas.

1.3.1. Dataedo

Catálogo de datos que permite, entre otras cosas, la creación y gestión de diccionarios de datos y del glosario empresarial [12].

Ventajas:

- Engloba varias herramientas para la gestión de datos.
- Creación automática de un diccionario de datos a partir de una base de datos.
- Compatibilidad con un gran número diferente de bases de datos.
- Visualización de diagramas entidad-relación.
- Variedad de formatos para exportar los datos (pdf, html, excel).

Inconvenientes:

- Precio muy elevado.

1.3.2. Database Note Taker

Herramienta para la creación de diccionarios de datos muy simple. Su objetivo principal es la documentación [40].

Ventajas:

- Totalmente gratuita.
- Creación automática de un diccionario de datos a partir de una base de datos.

Inconvenientes:

- Simplicidad, la funcionalidad es limitada.
- Los cambios son locales y no se comparten entre usuarios.

1.3.3. Redgate SQL Doc

Redgate ofrece una gran variedad de productos relacionados con el manejo de bases de datos, entre ellos se encuentra SQL Doc, una herramienta que facilita la documentación de bases de datos SQL [36].

Ventajas:

- Creación automática de un diccionario de datos a partir de una base de datos.
- Variedad de formatos para exportar los datos (pdf, html, Markdown).
- Precio asequible para pequeñas empresas.

- Interfaz sencilla y fácil de usar.

Inconvenientes:

- No tiene representación visual de las relaciones.

1.4. Objetivos

Dentro del contexto en el que se realiza este trabajo, se pueden diferenciar dos grupos de objetivos. El primero está relacionado con el aprendizaje y las competencias a adquirir durante la realización del proyecto, denominados **objetivos personales**. El segundo grupo corresponde a los **objetivos propios del proyecto**, que establecen los requisitos que la aplicación web debe cumplir una vez finalizada su implementación

Objetivos personales:

- Reforzar los conocimientos sobre desarrollo web con Spring y Angular.
- Aprender a gestionar un proyecto por completo.
- Ampliar el conocimiento en tecnologías utilizadas en el mundo laboral.
- Aplicar el marco de trabajo ágil SCRUM y respetar la calendarización de los sprints.

Objetivos del proyecto:

- Gestionar los usuarios del sistema.
- Crear el diccionario de datos de una base de datos.
- Modificar un diccionario y registrar los cambios en un historial.
- Asegurar la fiabilidad y consistencia de los datos entre todos los usuarios, incluso cuando se hayan modificado.
- Exportar los metadatos a un fichero.
- Ilustrar las relaciones de los datos mediante diagramas.

1.5. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 1 Introducción: Presenta el contexto y la motivación detrás del desarrollo de la aplicación, comparando alternativas existentes y sus ventajas e inconvenientes. También se definen los objetivos personales y del proyecto.

Capítulo 2 Requisitos y planificación: Detalla el marco de trabajo utilizado, los stakeholders, los roles de los usuarios en la aplicación y las épicas divididas en historias de usuario. También se recogen las reglas de negocio, así como la planificación del proyecto y el plan de riesgos.

Capítulo 3 Análisis: Aborda el modelado del dominio, incluyendo las clases y entidades principales del sistema. También se analiza el comportamiento e interacción de estas entidades, utilizando diagramas de actividades y máquinas de estados para representar los procesos y las interacciones clave.

Capítulo 4 Tecnologías utilizadas: Presenta las principales tecnologías empleadas en el proyecto y su utilización en cada área.

Capítulo 5 Diseño: Detalla las decisiones de diseño del proyecto, incluyendo la interfaz de usuario, la arquitectura general del sistema y las arquitecturas específicas para el cliente y el servidor. También se aborda el diseño de la base de datos, la comunicación entre los objetos del sistema y el despliegue del proyecto.

Capítulo 6 Implementación y pruebas: Recoge la licencia y organización del proyecto, las dificultades encontradas durante la implementación y el uso de Jasmine para las pruebas automatizadas.

Capítulo 7 Seguimiento del proyecto: Aborda la planificación de los sprints, su revisión y las propuestas de mejora para los siguientes. También se registra el tiempo dedicado a cada tarea y se señala si han surgido problemas durante la ejecución del sprint.

Capítulo 8 Conclusiones: Expone las conclusiones sacadas tras la finalización del proyecto, también se mencionan posibles líneas de trabajo futuras.

Anexo A Manuales: Incluye manuales de mantenimiento, de instalación/despliegue, y de uso.

Anexo B Resumen de enlaces adicionales: Incluye enlaces de interés sobre el proyecto, como el repositorio de código.

Capítulo 2

Requisitos y Planificación

2.1. Scrum

Scrum es un marco de trabajo ágil, se puede definir como un proceso empírico, donde la experiencia es la base del conocimiento y las decisiones se toman en base a lo observado [16] [38].

El trabajo se divide en pequeñas partes y cada equipo debe trabajar en ellas durante un corto periodo de tiempo. A cada periodo se le denomina **sprint** y su duración se fija al inicio del proyecto, siendo lo más común una duración de 2 semanas. Al final del sprint, todos los equipos deben entregar el resultado de su trabajo, el **incremento de valor**.

Los equipos suelen ser de tamaño reducido, aproximadamente de 10 personas. Un equipo Scrum consiste en un **Scrum Master**, un **Product Owner** y los **desarrolladores**, todos ellos tienen un objetivo común, crear un incremento valioso y útil para cada sprint.

- **Scrum Master**: Su tarea es garantizar que el equipo aplique los principios Scrum de manera correcta. Gestiona el product backlog (ver definición más adelante) y promueve la colaboración entre los miembros del equipo.
- **Product Owner**: Representa las necesidades de los stakeholders. Define las tareas del product backlog y proporciona claridad al equipo sobre la visión y el objetivo del producto.
- **Desarrollador**: Son los miembros que trabajan en el desarrollo del producto. Está formado por personas con diferentes habilidades, dependiendo del tipo de trabajo que realicen.

Los **stakeholders** son todas aquellas partes interesadas en el desarrollo exitoso del producto. Entre los stakeholders se pueden encontrar el equipo Scrum, los clientes y los usuarios

finales. La colaboración de todas las partes es fundamental para garantizar que el producto final cumpla con las expectativas.

Scrum establece tres artefactos que ayudan a gestionar el trabajo del equipo. Los **artefactos** ofrecen información importante que el equipo utiliza para definir el producto y el esfuerzo que hay que dedicar para crearlo. Los principales artefactos son:

- **Product Backlog**: Lista de tareas necesarias para crear el producto. Se puede ir refinando y dividiendo en tareas más pequeñas.
- **Sprint Backlog**: Conjunto de tareas seleccionadas del Product Backlog para realizar en un sprint específico.
- **Incremento**: Resultado de un sprint que contribuye de manera tangible al objetivo del producto. Cada incremento se añade a los anteriores y debe ser completamente funcional.

Los artefactos se ajustan a través de una serie de eventos diseñados para promover la transparencia del trabajo. A continuación, se definirán brevemente los eventos en los que participan los miembros de un equipo Scrum:

- **Sprint**: Como ya se dijo, es el periodo de tiempo durante el cual el equipo trabaja para realizar el incremento. Todos los demás eventos ocurren durante el sprint.
- **Sprint Planning**: Reunión que se realiza al principio del sprint para establecer las tareas del Sprint Backlog.
- **Daily Scrum**: Reunión diaria muy breve entre los desarrolladores en la que se comenta el trabajo realizado y las dificultades que se hayan detectado.
- **Sprint Review**: Reunión del equipo con los stakeholders para valorar el incremento.
- **Sprint Retrospective**: Reunión en la que los miembros del equipo reflexionan sobre cómo ha ido el sprint y lo que se debe cambiar para mejorar la efectividad.

2.1.1. Adaptación del marco de trabajo

Es necesario adaptar el marco de trabajo a la asignatura, Trabajo de Fin de Grado, para que se ajuste al tiempo que se le debe dedicar, que será de aproximadamente 300 horas equivalentes a 12 ECTS. También deben ajustarse los roles del equipo Scrum, ya que el trabajo será realizado por una estudiante con el apoyo de su tutora. La estudiante asumirá tanto el rol de Product Owner como el de desarrollador, mientras que la tutora actuará como Scrum Master, asegurándose de que la estudiante cumpla con los principios Scrum.

Por lo general, la finalización de un sprint coincidirá con el inicio del siguiente. Esto quiere decir que los eventos Sprint Review, Sprint Retrospective y Sprint Planning se unificarán en una sola reunión. No se celebrarán la reuniones diarias que promueve Scrum ya que resulta

inviable, sin embargo, se fijará una reunión intermedia en cada sprint para monitorizar el progreso de la estudiante.

Se dedicarán unas 40 horas de trabajo por sprint. Se han establecido un total de 8 sprints, aunque se planificarán 2 sprints adicionales en caso de que el trabajo se vea retrasado o tome más tiempo de lo esperado.

Para organizar el trabajo, se definirán unas épicas que se dividirán en historias de usuario. La estimación del esfuerzo se realizará mediante puntos de historia, estos se utilizan para calcular el esfuerzo necesario para completar una historia de usuario. A cada historia se le asignará un número de puntos, siguiendo una serie numérica lineal a partir del 1, donde cada unidad equivaldrá a 5 horas de trabajo. La cantidad de puntos asignados dependerá de la percepción sobre la complejidad de la tarea. Cada sprint incluirá las historias de usuario o tareas necesarias para alcanzar un total de 8 puntos de historia.

2.2. Stakeholders, roles y épicas

Los stakeholders de este proyecto son: la estudiante, como principal interesado del correcto desarrollo del proyecto; la tutora, quien forma parte del equipo Scrum; y las posibles organizaciones o usuarios finales que podrían beneficiarse con la utilización de la aplicación. La tutora actuará simulando el papel como stakeholder de una organización o usuario final.

Los usuarios desempeñarán un rol diferente en cada diccionario de datos. Cada rol otorga un conjunto específico de permisos. Estos roles serán:

- **Administrador**: Se encarga de gestionar los usuarios del sistema y sus permisos. Aunque no tiene acceso a los diccionarios, puede gestionar los permisos de los usuarios en ellos. Puede crear nuevos usuarios.
- **Arquitecto**: Puede modificar la estructura de los elementos existentes y editar el contenido de cualquier campo. Además, es responsable de revisar las modificaciones realizadas por los editores.
- **Editor**: puede modificar el contenido de ciertos campos, pero no puede cambiar la estructura ni el nombre de los elementos.
- **Lector**: Solo puede consultar la información del diccionario.

Durante la definición de las épicas, se utilizaron los términos “usuario” y “usuario normal” para referirse a diferentes tipos de usuarios. “Usuario” hacía referencia a cualquier usuario del sistema, mientras que “usuario normal” agrupaba a aquellos que podían desempeñar los roles de arquitecto, editor o lector en los diccionarios de datos. Más adelante, durante el sprint 11, se decidió reemplazar el término “usuario normal” por “gestor de metadatos”.

2.2.1. Épicas

Una **épica** [9] describe un gran conjunto de trabajo relacionado. Generalmente representa una característica o funcionalidad amplia que necesita ser desarrollada. Dado que las épicas suelen ser demasiado grandes o complejas para abordarse directamente, se dividen en varias **historias de usuario**, que son unidades más pequeñas y manejables de trabajo, que pueden desarrollarse de forma independiente dentro de un solo sprint. Cada historia de usuario describe una funcionalidad específica desde la perspectiva del usuario final.

Tanto las épicas como sus historias asociadas no son elementos estáticos y se pueden refinar y reorganizar a medida que el proyecto progresa, adaptándose a los cambios en las prioridades o en las necesidades del negocio.

Las épicas y las historias de usuario se definirán siguiendo el formato: Como <stakeholder>, quiero <funcionalidad> para <beneficio>.

El Product Backlog inicial está formado por las siguientes épicas:

EP01 - Gestión de usuarios

Como administrador, quiero gestionar los usuarios del sistema para asignarles diferentes roles en diferentes diccionarios de datos.

EP02 - Inicio de sesión

Como usuario, quiero iniciar sesión en la aplicación para poder realizar diferentes tareas dependiendo de mi rol.

EP03 - Creación de diccionario

Como usuario normal, quiero crear un diccionario de datos para documentar una o varias bases de datos relacionadas y brindar información sobre su contenido. Como consecuencia, el usuario que lo cree pasará a tener el rol de arquitecto en ese diccionario de datos.

EP04 - Modificación del diccionario

Como arquitecto o editor, quiero modificar un diccionario de datos para añadir, eliminar o cambiar su contenido.

EP05 - Exploración del diccionario de datos

Como usuario normal, quiero ver todos los diccionarios de datos en los que tengo permisos para consultarlos.

EP06 - Búsqueda de diccionario

Como usuario normal, quiero filtrar los diccionarios a partir de su nombre u otro campo para facilitar su búsqueda.

EP07 - Registro de operaciones

Como administrador, quiero tener acceso a un registro de operaciones por cada diccionario para saber las modificaciones y consultas realizadas por los usuarios.

EP08 - Historial de modificaciones

Como usuario normal, quiero ver el historial de modificaciones de un diccionario para saber cómo ha cambiado.

EP09 - Exportación de diccionario

Como usuario normal, quiero exportar un diccionario de datos a distintos formatos para facilitar su distribución.

EP10 - Selección de idioma

Como usuario, quiero cambiar el idioma de la web para entender lo que estoy leyendo.

EP11 - Representación visual

Como usuario normal, quiero ver una representación visual de las relaciones entre los elementos de un diccionario de datos para hacerme una idea de cómo están relacionadas.

2.3. División de épicas en historias de usuario

Las épicas se descomponen en historias de usuario para facilitar la planificación y ejecución del trabajo dentro de cada sprint. Esta división permite fragmentar el trabajo en tareas más pequeñas y manejables.

Cada historia de usuario estará representada por una referencia única con el formato HUX, donde X es un número. Para la descripción, se empleará el mismo esquema utilizado en la definición de las épicas, manteniendo así la coherencia. También se indicará el esfuerzo estimado en puntos de historia para distribuir el trabajo de manera uniforme a lo largo de los sprints.

De la Tabla 2.1 a la Tabla 2.9 se encuentran las historias de usuario correspondientes a las épicas de la EP01 a la EP09. En concreto, la épica EP10 no se ha dividido en historias de usuario ya que se prevé que la tarea de traducción se realice de manera continua, cada historia de usuario incluirá la implementación de la aplicación tanto en inglés como en español. En cuanto a la épica EP11, se ha decidido que no será implementada en el desarrollo actual, quedando como una posible ampliación futura.

En la historia HU23, el "registro de operaciones" se refiere a un conjunto de operaciones realizadas sobre un diccionario, que incluyen lecturas, modificaciones, creación o eliminación de elementos. En la historia HU24, el término "lista de operaciones" se utiliza como sinónimo de registro de operaciones. Por su parte, en la historia HU25, la "lista de modificaciones" hace referencia al historial de cambios realizados en un diccionario de datos, que puede incluir cambios en el contenido de elementos existentes, así como la creación y eliminación

Referencia	Historia de usuario	Puntos
HU01	Como administrador, quiero crear nuevos usuarios para poblar el sistema.	3
HU02	Como administrador, quiero ver una lista con todos los usuarios del sistema para gestionarlos eficientemente.	2
HU03	Como administrador, quiero modificar los roles de los usuarios para denegar/facilitar el acceso a un determinado diccionario de datos.	1
HU04	Como administrador, quiero habilitar/deshabilitar cuentas para controlar el inicio de sesión de los usuarios.	1
HU05	Como administrador, quiero modificar la información de un usuario para mantenerla actualizada.	1
HU06	Como administrador, quiero restablecer la contraseña de un usuario para facilitarle una nueva contraseña en caso de que se le haya olvidado.	1
HU30	Como administrador, quiero ver los diccionarios a los que tiene acceso un usuario normal para ver su rol.	1
HU31	Como administrador, quiero modificar el rol de un usuario normal en un determinado diccionario para otorgar o revocarle sus privilegios.	1
HU32	Como administrador, quiero facilitar el acceso de un usuario normal a un diccionario, para que pueda visualizar su contenido.	1
HU33	Como administrador, quiero denegar el acceso de un usuario normal a uno de sus diccionarios, para que no pueda visualizar su contenido.	1

Tabla 2.1: Historias de usuario EP01 - Gestión de usuarios

de elementos. El registro de operaciones será visible únicamente para los administradores, mientras que el historial de modificaciones será visible por cualquier gestor de metadatos que tenga acceso al diccionario. En resumen, la principal diferencia entre el registro de operaciones y el historial de modificaciones radica en si se incluyen o no las operaciones de lectura y en quién tiene acceso a cada uno de ellos.

Referencia	Historia de usuario	Puntos
HU07	Como usuario, quiero ingresar mi nombre de usuario y contraseña para iniciar sesión y acceder a una pantalla distinta dependiendo mi tipo de usuario.	3
HU08	Como usuario, quiero solicitar que mi contraseña se restablezca en caso de que se me olvide, para volver acceder a mi cuenta.	2
HU09	Como usuario, quiero cerrar sesión para que mi cuenta no quede accesible a usuarios externos a la aplicación.	1

Tabla 2.2: Historias de usuario EP02 - Inicio de sesión

Referencia	Historia de usuario	Puntos
HU10	Como usuario normal, quiero crear un nuevo diccionario de datos indicando un nombre y una descripción para documentar todas las bases de datos relacionadas a un proyecto.	2
HU11	Como arquitecto, quiero crear una base de datos dentro de un diccionario de datos indicando un nombre, una descripción y el SGBD para dar información sobre esa base de datos.	2
HU12	Como arquitecto, quiero crear tantas tablas/colecciones como necesite para documentar una base de datos indicando, como mínimo, un nombre y una descripción.	2

Tabla 2.3: Historias de usuario EP03 - Creación de diccionario

Referencia	Historia de usuario	Puntos
HU13	Como arquitecto, quiero añadir tantas columnas/campos como necesite a una tabla/colección para dar información más precisa.	2
HU14	Como arquitecto o editor, quiero modificar el nombre y la descripción de los elementos del diccionario para corregir errores o reflejar cambios realizados en las bases de datos reales.	1
HU15	Como arquitecto o editor, quiero modificar el contenido de las columnas/campos de una tabla/colección para dar información más precisa.	3
HU16	Como arquitecto, quiero eliminar elementos de un diccionario o el propio diccionario para borrar información innecesaria o que ya no exista.	1
HU28	Como arquitecto, quiero ver los cambios realizados por un editor antes de aplicarlos a los elementos para aceptarlos o rechazarlos. Si los rechazo, quiero indicar el motivo para informar al editor.	2
HU29	Como arquitecto, quiero establecer un número de arquitectos revisores que deben aprobar los cambios realizados por editores para garantizar la fiabilidad de la información modificada.	1

Tabla 2.4: Historias de usuario EP04 - Modificación del diccionario

Referencia	Historia de usuario	Puntos
HU17	Como usuario normal, quiero ver todos los diccionarios a los que tengo acceso para seleccionar uno.	1
HU18	Como usuario con rol, quiero ver todas las bases de datos de los diccionarios a los que tengo acceso para seleccionar una.	1
HU19	Como usuario normal, quiero visualizar todas las tablas/-colecciones de las bases de datos a las que tengo acceso para seleccionar una y explorar su contenido.	1
HU20	Como usuario normal, quiero visualizar todas las columnas/campos de las tablas/colecciones a las que tengo acceso para seleccionar una y explorar su contenido.	1

Tabla 2.5: Historias de usuario EP05 - Exploración del diccionario de datos

Referencia	Historia de usuario	Puntos
HU21	Como usuario normal, quiero buscar un diccionario a partir de su nombre para facilitar su selección.	1
HU22	Como usuario normal, quiero ordenar los diccionarios por un campo específico para organizar mejor la información.	1

Tabla 2.6: Historias de usuario EP06 - Búsqueda de diccionario

Referencia	Historia de usuario	Puntos
HU23	Como administrador, quiero ver todos los diccionarios del sistema para acceder a su registro de operaciones.	1
HU24	Como administrador, quiero ver la lista de operaciones realizadas sobre un diccionario, ordenada cronológicamente, para llevar un control sobre él.	2

Tabla 2.7: Historias de usuario EP07 - Registro de operaciones

Referencia	Historia de usuario	Puntos
HU25	Como usuario normal, quiero ver la lista de modificaciones realizadas sobre un diccionario, ordenada cronológicamente, para ver cómo ha evolucionado.	1

Tabla 2.8: Historias de usuario EP08 - Historial de modificaciones

Referencia	Historia de usuario	Puntos
HU26	Como usuario normal, quiero descargar la información de un diccionario en un documento PDF para acceder a la información de forma offline.	2
HU27	Como usuario normal, quiero descargar la información de un diccionario en un fichero JSON para facilitar su utilización en otras aplicaciones.	1

Tabla 2.9: Historias de usuario EP09 - Exportación de diccionario

2.4. Reglas de negocio

Las **reglas de negocio** [28] son principios fundamentales que definen cómo debe comportarse un sistema en función de las necesidades del negocio. Estas reglas determinan cómo se manipulan los datos, qué acciones están permitidas y en qué condiciones.

En el desarrollo de software, estas reglas actúan como una guía durante el diseño y la implementación del sistema, asegurando que las funcionalidades estén alineadas con las necesidades del negocio.

A continuación, se presentan las reglas de negocio obtenidas a partir de un análisis inicial de los requisitos.

RN01 - Existen dos tipos de usuarios en el sistema, los administradores y los gestores de metadatos. El tipo de usuario se asigna al momento de la creación y no puede ser modificado.

RN02 - Las tareas de los administradores son crear y gestionar usuarios, así como controlar el acceso a los diccionarios de datos asignando roles a los usuarios. También pueden acceder al historial de operaciones realizadas sobre un diccionario de datos.

RN03 - Los gestores de metadatos realizan operaciones sobre los diccionarios, deben tener un rol asignado a un diccionario de datos para poder acceder a él, este rol sólo puede ser uno pero puede cambiar. Las operaciones que un usuario puede realizar en un diccionario de datos varían según el rol asignado.

RN04 - Los roles que se le pueden asignar a un gestor de metadatos en un diccionario de datos son arquitecto, editor y lector.

RN05 - Los gestores de metadatos con rol de arquitecto pueden crear, modificar y eliminar cualquier elemento de un diccionario de datos.

RN06 - Los gestores de metadatos con rol de editor pueden modificar únicamente las metapropiedades que aporten documentación relevante al diccionario de datos, sin afectar su estructura.

RN07 - Los gestores de metadatos con rol de lector pueden consultar el contenido del diccionario de datos, pero no pueden realizar ninguna modificación.

RN08 - Cada cuenta tiene asociado un correo electrónico, este es único y sirve como nombre de usuario.

RN09 - Cuando una cuenta es creada, se le asigna una contraseña aleatoria.

RN10 - Las cuentas de usuarios recién creadas estarán inactivas hasta que se cambie la contraseña por primera vez.

RN11 - Las contraseñas deben cambiarse cada 90 días, los usuarios reciben una notificación para que cambien de contraseña. Si a los 2 días de haberse cumplido el plazo, el usuario no ha cambiado la contraseña la cuenta pasará a estar inactiva.

RN12 - Un usuario inactivo solo puede configurar su cuenta, el resto de la funcionalidad está deshabilitada hasta que se vuelva a activar.

RN13 - Las cuentas inactivas se pueden activar cambiando la contraseña.

RN14 - Las contraseñas deben tener un mínimo de 8 caracteres, entre los cuales debe haber, al menos, un carácter mayúscula, uno minúscula y un número.

RN15 - Los administradores pueden habilitar o deshabilitar cuentas de otros usuarios.

RN16 - Los usuarios con cuentas deshabilitadas no pueden iniciar sesión.

RN17 - Cuando un gestor de metadatos crea un diccionario se le asigna el rol de arquitecto en ese diccionario.

RN18 - Un arquitecto o editor solo puede editar elementos que no tengan cambios pendientes de aprobación.

RN19 - Los arquitectos pueden revisar los cambios propuestos por editores. Si se aceptan, los cambios se aplican a los elementos correspondientes del diccionario, y si se rechazan, se debe indicar el motivo. Una vez terminada la revisión, el elemento volverá a estar disponible para su edición.

RN20 - Las propuestas rechazadas pueden ser consultadas por sus autores, quienes podrán ver la información original y la propuesta realizada, así como el motivo del rechazo.

RN22 - En un diccionario de datos, se puede configurar el número de arquitectos necesarios para aprobar una propuesta y hacerla definitiva. Se puede configurar el diccionario de tal forma que no haga falta la validación de los arquitectos y el cambio sea inmediato.

RN23 - Basta que un arquitecto rechace la propuesta para que esta se considere rechazada.

2.5. Plan de riesgos

El **riesgo** de un proyecto es un evento incierto que, en caso de producirse, puede influir de manera positiva o negativa en la consecución de los objetivos del proyecto [24]. Se considera **oportunidad** a un riesgo con resultados positivos, mientras que una **amenaza** es un riesgo con efectos negativos.

Todo proyecto está sujeto a riesgos. Si no se gestionan adecuadamente, los riesgos pueden multiplicar los problemas, ya que la falta de control sobre las amenazas aumenta la probabilidad de que se materialicen.

Para determinar el nivel de riesgo se empleará la matriz mostrada en la Tabla 2.10, que evalúa la probabilidad de ocurrencia y el impacto que supondría. El diseño de esta matriz se ha tomado de los apuntes de la asignatura Planificación y Gestión de Proyectos [22].

Prob \ Imp	**Bajo**	**Medio**	**Alto**
Baja	Bajo	Bajo	Medio
Media	Bajo	Medio	Alto
Alta	Medio	Alto	Alto

Tabla 2.10: Matriz de probabilidad-impacto. Tomada de [22]

Riesgo R01	
Título	Enfermedad de la estudiante
Descripción	La estudiante no podrá dedicar tiempo a trabajar si está enfermo. Como el equipo de desarrollo solo está formado por la estudiante esto retrasaría el proyecto.
Probabilidad	Baja
Impacto	Alto
Nivel de riesgo	Medio
Plan de mitigación	▪ Establecer un margen extra de tiempo para realizar las tareas retrasadas.
Plan de contingencia	▪ Mover las tareas no finalizadas al siguiente sprint.

Tabla 2.11: Riesgo R01 - Enfermedad de la estudiante

En las Tablas 2.11, 2.12, 2.13, 2.14, 2.15, 2.16 se presenta una descripción detallada de cada riesgo identificado. Para cada riesgo se aporta una descripción, el valor de probabilidad, impacto y nivel de riesgo en una escala ordinal dada por los valores Bajo, Medio y Alto. Se presenta también el plan de mitigación, que incluye las medidas a tomar para reducir la probabilidad de ocurrencia del riesgo o el impacto que pueda causar, y el plan de contingencia, que establece las acciones a tomar si el riesgo se materializa [19].

Riesgo R02	
Título	Avería del equipo informático
Descripción	El ordenador con el que trabaja la estudiante podría sufrir algún tipo de avería, la cual impediría trabajar en el proyecto causando un retraso.
Probabilidad	Baja
Impacto	Alto
Nivel de riesgo	Medio
Plan de mitigación	▪ Establecer un margen extra de tiempo para realizar las tareas retrasadas. ▪ Realizar un mantenimiento al equipo periodicamente, ▪ Mantener el trabajo actualizado en un repositorio remoto de tal forma que facilite el trabajo entre diferentes dispositivos.
Plan de contingencia	▪ Comprar un ordenador nuevo.

Tabla 2.12: Riesgo R02 - Avería del equipo informático

Riesgo R03	
Título	Mala planificación de las historias de usuario
Descripción	La estimación de las historias de usuario podría no ser la adecuada, tomando más tiempo del inicialmente estimado.
Probabilidad	Media
Impacto	Medio
Nivel de riesgo	Medio
Plan de mitigación	▪ Utilizar el marco de trabajo Scrum el cual permite ajustar el backlog de cada sprint según sea necesario. ▪ Evitar poner tiempos muy ajustados ▪ Buscar referencias en otros TFGs para saber cuanto tiempo le ha llevado a otras personas realizar un trabajo parecido.
Plan de contingencia	▪ Ajustar el product backlog teniendo en cuenta las nuevas estimaciones.

Tabla 2.13: Riesgo R03 - Mala planificación de las historias de usuario

Riesgo R04	
Título	Gold Plating
Descripción	Desarrollar más funcionalidad de la especificada inicialmente en el product backlog podría provocar retrasos en los sprints correspondientes y reducir la calidad de los entregables.
Probabilidad	Baja
Impacto	Medio
Nivel de riesgo	Bajo
Plan de mitigación	▪ No realizar más funcionalidad que la establecida para el sprint. ▪ Terminar primero todas las tareas del sprint y, si sobra tiempo, realizar un análisis de las nuevas funcionalidades que se desean añadir para comprobar que aporten algún valor.
Plan de contingencia	▪ Descartar las modificaciones si están llevando demasiado tiempo. ▪ Ajustar el product backlog con las nuevas tareas.

Tabla 2.14: Riesgo R04 - Gold Plating

Riesgo R05	
Título	Otras asignaturas
Descripción	La estudiante está cursando más asignaturas aparte del TFG. El tiempo que puede dedicar a la realización del trabajo dependerá de los exámenes y entregas que deba realizar en las otras asignaturas, provocando retrasos puntuales.
Probabilidad	Media
Impacto	Alto
Nivel de riesgo	Alto
Plan de mitigación	▪ Tener en consideración las fechas en las que se realicen exámenes y ajustar la planificación a estos.
Plan de contingencia	▪ Mover las tareas no finalizadas al siguiente sprint.

Tabla 2.15: Riesgo R05 - Otras asignaturas

Riesgo R06	
Título	Desconocimiento de las tecnologías
Descripción	Se realizará un proyecto software al completo por lo que se deberán utilizar muchas tecnologías y muy diferentes entre sí, posiblemente haya alguna que no sepa utilizar, requiriendo un tiempo de aprendizaje.
Probabilidad	Alta
Impacto	Alto
Nivel de riesgo	Alto
Plan de mitigación	▪ Investigar el uso de las tecnologías elegidas para el desarrollo.
Plan de contingencia	▪ Cambiar la tecnología escogida por otra conocida.

Tabla 2.16: Riesgo R06 - Desconocimiento de las tecnologías

2.6. Planificación

El proyecto se inició en febrero de 2024 y, en un escenario optimista, se estima que podría finalizar a finales de julio. Sin embargo, se ha establecido un periodo extra en septiembre, en caso de que no se complete a tiempo. Dado que la defensa del TFG no tendrá lugar en el curso académico 2023-2024, se ha podido extender la planificación de los sprints hasta septiembre.

Al comienzo del proyecto, se estableció un sprint 0 dedicado principalmente a tareas de investigación, como el estudio de los diccionarios de datos, la selección de las tecnologías a utilizar y la lectura de otros TFGs para familiarizarse con el formato de este tipo de trabajos. Además, se llevaron a cabo otras actividades, como iniciar la redacción de los primeros capítulos de introducción y planificación, así como la elaboración de la calendarización para los sprints posteriores.

La planificación inicial de los sprints, del uno en adelante, puede verse en la Tabla 2.17. Por lo general, cada sprint tiene una duración de dos semanas y comienza inmediatamente después de la finalización del anterior. Sin embargo, algunos de ellos han tenido que ajustarse para tener en cuenta períodos vacacionales y de exámenes. Específicamente, los sprints que presentan ajustes son los siguientes:

- Sprint 1: se extiende la duración a tres semanas debido a las vacaciones de Semana Santa.
- Sprints 4 y 5: se deja un espacio de tiempo entre ambos para la preparación de los exámenes de la convocatoria ordinaria.
- Sprints 7 y 8: hay un mes de diferencia entre la finalización de uno y el inicio del siguiente, debido a las vacaciones de verano.

2.7. Presupuesto

El desarrollo de un proyecto software conlleva una serie de costes que varían en función de los recursos necesarios. Dada la situación en la que se desarrollará el proyecto, se presentarán dos presupuestos distintos. Primero, se mostrará un presupuesto simulado, suponiendo que el proyecto se realizara en un entorno empresarial. Después, se detallará el presupuesto real ajustado al contexto académico.

2.7.1. Presupuesto simulado

Según el portal de empleos **Jobted** [26], el salario medio de un **desarrollador web** en España es de 31.600 € brutos anuales. En este caso, se supondrá que el desarrollador es junior y no tiene experiencia laboral, por lo que se le asignará un salario de 20.200 €. Para

Sprint	Fecha de inicio	Fecha de finalización	Observaciones
Sprint 1	14/03/2024	04/04/2024	Duración 2 semanas. Calendarización expandida por vacaciones de Semana Santa
Sprint 2	04/04/2024	18/04/2024	
Sprint 3	18/04/2024	02/05/2024	
Sprint 4	02/05/2024	16/05/2024	
Sprint 5	12/06/2024	27/06/2024	Comienza después de la convocatoria ordinaria
Sprint 6	27/06/2024	11/07/2024	
Sprint 7	11/07/2024	25/07/2024	
Sprint 8	29/08/2024	12/09/2024	Comienza después de las vacaciones de verano
Sprint 9	12/09/2024	26/09/2024	Extra
Sprint 10	26/09/2024	10/10/2024	Extra

Tabla 2.17: Planificación inicial

una empresa, el coste total es mayor, aproximadamente, un 32 % más debido a la **seguridad social**. Por lo tanto, se pagaría un total de 26.664 € anuales por dicho empleado. Suponiendo que el empleado trabajase a tiempo completo, unas 160 horas mensuales, el coste por hora sería de 13,89 €. Teniendo en cuenta que, idealmente, el proyecto se desarrollará durante 8 sprints de 40 horas cada uno, se estima que el empleado trabajará en este proyecto durante 320 horas. Es decir, tener a un desarrollador trabajando en este proyecto costaría a una empresa **4.444,8 €**.

Para trabajar, el desarrollador necesitaría, al menos, un portátil. Como requisitos mínimos, se han contemplado estos componentes: 16 GB de RAM, 500 GB de SSD y un procesador Intel Core i5 de 13ª generación o equivalente. Tras revisar varias opciones, se seleccionó un portátil Lenovo [34] con un precio de 649 €, cuyas especificaciones se consideran adecuadas para soportar el desarrollo de manera holgada. Este coste se prorrateará considerando un periodo de amortización de 4 años, por lo que el coste proporcional al tiempo dedicado al proyecto será de **67,6 €**.

El proyecto se desarrollará en 5 meses, algunos de los programas que se utilizarán en el proyecto son de pago y, por lo tanto, se deberá pagar una mensualidad para poder utilizarlos, estos programas son:

- GitLab Premium - 26,72 € por usuario y mes
- IntelliJ Idea Ultimate para organizaciones - 72,48 € por usuario y mes
- GitHub Copilot Business - 17,51 € por usuario y mes
- Astah professional individual - 8,99 € por mes

También se tendrá en cuenta el consumo eléctrico del ordenador. Suponiendo que el portátil consume 0,2 kWh y se utilizará durante 320 horas, el consumo total quedaría en 64 kW. Siendo el precio medio del kW 0,12 €, la **electricidad** supondría un gasto de **7,68 €**.

Es posible que durante el desarrollo del proyecto se materialice algún riesgo que suponga un atraso o un aumento en el presupuesto, o que algunos gastos no estuvieran previstos, es por esto que se establece un **fondo de contingencia** del **20 %** del presupuesto inicial.

En la Tabla 2.18 se puede ver el resumen del presupuesto simulado.

2.7.2. Presupuesto real

El proyecto se realizará en un ordenador personal con más de 5 años de antigüedad, por lo que se considera que ya está totalmente amortizado.

Al tratarse de un proyecto desarrollado en un entorno académico, no se cobrará nada por el trabajo realizado. No será necesario pagar por las licencias de uso de los programas, ya que la Universidad posee licencias académicas que permiten su uso gratuito. El consumo eléctrico del ordenador tampoco se asumirá de manera directa.

En conclusión, el desarrollo del proyecto **no supondrá coste alguno**.

Recurso	Precio
Empleado	4.444,8 €
Ordenador	67,6 €
Licencias de programas	628,5 €
Electricidad	7,68 €
Total	**5.148,58 €**
Fondo de contingencia (20 %)	1.029,72 €
Total con fondo	**6.178,3 €**

Tabla 2.18: Presupuesto simulado

2.8. Replanificación del proyecto

El objetivo de este apartado es justificar la necesidad de replanificar el proyecto debido a una desviación significativa en las horas de trabajo dedicadas. Tras la realización del sprint 7, se habían estimado 280 horas de trabajo, sin embargo, solo se han dedicado 180 horas. Debido a esta diferencia de 100 horas se requiere una reorganización de las actividades para asegurar que el proyecto se desarrolle correctamente.

La desviación en el tiempo de trabajo se ha producido debido a varios factores, como tener que dedicar tiempo a otras asignaturas y sus exámenes, el inicio de prácticas en empresa. Estas han supuesto un impacto considerable, aunque también hay que mencionar la incorporación de funcionalidades que en un principio no se tuvieron en cuenta pero que aumentó la complejidad del proyecto, como es la validación de los arquitectos en los cambios realizados en una base de datos.

Se estima que actualmente, tras haber finalizado el Sprint 7, se ha completado aproximadamente el 30 % de las historias de usuario del proyecto. La funcionalidad fundamental, que requiere más tiempo y esfuerzo, ya ha sido desarrollada. El restante 70 % se espera completar en un plazo menor.

Para corregir la desviación producida, se ha decidido añadir dos sprints regulares al cronograma del proyecto. En la Tabla 2.19 se presenta el nuevo cronograma, este incluye los 2 sprints añadidos, con los cuales se pretende alcanzar las 300 horas de trabajo cuando hayan finalizado los sprints regulares. También se han atrasado los sprints extras que se utilizarán en caso de ser necesario y que se realizarían una vez se hayan terminado los sprints regulares.

Sprint	Fecha de inicio	Fecha de finalización	Observaciones
Sprint 1	14/03/2024	04/04/2024	Duración 2 semanas. Calendarización expandida por vacaciones de Semana Santa
Sprint 2	04/04/2024	18/04/2024	
Sprint 3	18/04/2024	02/05/2024	
Sprint 4	02/05/2024	16/05/2024	
Sprint 5	12/06/2024	27/06/2024	Comienza después de la convocatoria ordinaria
Sprint 6	27/06/2024	11/07/2024	
Sprint 7	11/07/2024	25/07/2024	
Sprint 8	29/08/2024	12/09/2024	Comienza después de las vacaciones de verano
Sprint 9	12/09/2024	26/09/2024	Añadido tras la replanificación
Sprint 10	26/09/2024	10/10/2024	Añadido tras la replanificación
Sprint 11 (Extra)	10/10/2024	24/10/2024	Aplazado por la replanificación
Sprint 12 (Extra)	14/11/2024	28/11/2024	Aplazado por la replanificación

Tabla 2.19: Replanificación

2.9. Product backlog final

El product backlog final está formado por las historias de usuario que se muestran en las tablas 2.20 y 2.21. Este product backlog incluye únicamente aquellas historias de usuario que han sido desarrolladas y que, por lo tanto, forman el **producto mínimo viable** (**MVP**). Las historias están organizadas por orden de prioridad, de mayor a menor, y siguen el mismo orden en que fueron desarrolladas.

El resto de las historias de usuario han sido colocadas en el **icebox**, una zona para aquellas funcionalidades que se han pausado, pero que podrían considerarse en un futuro desarrollo.

Referencia	Historia de usuario
HU01	Como administrador, quiero crear nuevos usuarios para poblar el sistema.
HU07	Como usuario, quiero ingresar mi nombre de usuario y contraseña para iniciar sesión y acceder a una pantalla distinta dependiendo mi tipo de usuario.
HU10	Como usuario normal, quiero crear un nuevo diccionario de datos indicando un nombre y una descripción para documentar todas las bases de datos relacionadas a un proyecto.
HU09	Como usuario, quiero cerrar sesión para que mi cuenta no quede accesible a usuarios externos a la aplicación.
HU17	Como usuario normal, quiero ver todos los diccionarios a los que tengo acceso para seleccionar uno.
HU18	Como usuario con rol, quiero ver todas las bases de datos de los diccionarios a los que tengo acceso para seleccionar una.
HU11	Como arquitecto, quiero crear una base de datos dentro de un diccionario de datos indicando un nombre, una descripción y el SGBD para dar información sobre esa base de datos.
HU02	Como administrador, quiero ver una lista con todos los usuarios del sistema para gestionarlos eficientemente.
HU04	Como administrador, quiero habilitar/deshabilitar cuentas para controlar el inicio de sesión de los usuarios.
HU05	Como administrador, quiero modificar la información de un usuario para mantenerla actualizada.
HU06	Como administrador, quiero restablecer la contraseña de un usuario para facilitarle una nueva contraseña en caso de que se le haya olvidado.
HU30	Como administrador, quiero ver los diccionarios a los que tiene acceso un usuario normal para ver su rol.

Tabla 2.20: Product backlog final [Parte 1]

Referencia	Historia de usuario
HU31	Como administrador, quiero modificar el rol de un usuario normal en un determinado diccionario para otorgar o revocarle sus privilegios.
HU33	Como administrador, quiero denegar el acceso de un usuario normal a uno de sus diccionarios, para que no pueda visualizar su contenido.
HU32	Como administrador, quiero facilitar el acceso de un usuario normal a un diccionario, para que pueda visualizar su contenido.
HU19	Como usuario normal, quiero visualizar todas las tablas/colecciones de las bases de datos a las que tengo acceso para seleccionar una y explorar su contenido.
HU12	Como arquitecto, quiero crear tantas tablas/colecciones como necesite para documentar una base de datos indicando, como mínimo, un nombre y una descripción.
HU20	Como usuario normal, quiero visualizar todas las columnas/campos de las tablas/colecciones a las que tengo acceso para seleccionar una y explorar su contenido.
HU13	Como arquitecto, quiero añadir tantas columnas/campos como necesite a una tabla/colección para dar información más precisa.
HU15	Como arquitecto o editor, quiero modificar el contenido de las columnas/campos de una tabla/colección para dar información más precisa.
HU14	Como arquitecto o editor, quiero modificar el nombre y la descripción de los elementos del diccionario para corregir errores o reflejar cambios realizados en las bases de datos reales.
HU16	Como arquitecto, quiero eliminar elementos de un diccionario o el propio diccionario para borrar información innecesaria o que ya no exista.
HU28	Como arquitecto, quiero ver los cambios realizados por un editor antes de aplicarlos a los elementos para aceptarlos o rechazarlos. Si los rechazo, quiero indicar el motivo para informar al editor.

Tabla 2.21: Product backlog final [Parte 2]

Capítulo 3

Análisis

En este capítulo se detalla el proceso de análisis llevado a cabo a lo largo de los diferentes sprints. Los modelos se han ido desarrollando y refinando progresivamente a medida que se adquiría un mayor conocimiento sobre el tema. Para modelar el dominio se ha utilizado un diagrama de clases y para desarrollar el modelo dinámico se han utilizado tanto diagramas de estados como diagramas de actividades.

3.1. Modelado del dominio

En la Figura 3.1 se presenta un diagrama de clases que ilustra todas las entidades del dominio identificadas. Al tratarse de un modelo conceptual no se incluyen operaciones. En este diagrama se pueden apreciar tres grupos de entidades: las que modelan los diccionarios de datos, como bases de datos, tablas, columnas y atributos, las que modelan a los usuarios y las que modelan acciones sobre el diccionario.

La entidad *Dictionary* tiene una estructura similar a los elementos que la forman, es decir, a las bases de datos. *Database*, a su vez, está compuesta por otros elementos a los cuales se les ha denominado *MainElement*, que representan tablas o colecciones. Esta abstracción responde a la necesidad de modelar tanto bases de datos relacionales, en las que la información se organiza en tablas, como bases de datos NoSQL basadas en documentos, en las que se utilizan colecciones. Cada uno de estos *MainElement* está formado por otros elementos denominados *InternalElement*, que podrían ser columnas o campos, dependiendo del tipo de base de datos. Dado que todas estas entidades tienen atributos comunes, se extrajeron en una superclase de la que heredan todas, llamada *Element*.

La utilización de la superclase Element facilita el modelado de las acciones. De esta forma se puede identificar fácilmente el elemento sobre el cual se realiza la acción.

Existen dos tipos de usuarios en el sistema, administradores y usuarios normales, dependiendo de su rol tendrán funcionalidades totalmente diferentes. Aunque lo realmente

importante en el dominio son las relaciones que tiene un usuario normal con los diccionarios, ya que son fundamentales para definir cómo pueden interactuar y acceder a los diferentes diccionarios.

3.2. Modelado dinámico

El modelado dinámico se ha utilizado para representar el comportamiento y las interacciones de las principales entidades del sistema. Las entidades que cambian su estado son las cuentas de los usuarios y los elementos que conforman los diccionarios de datos.

3.2.1. Modelo de proceso de negocio

El diagrama de actividades de la Figura 3.2 muestra el proceso de creación y activación de una cuenta de usuario. Cuando un administrador crea una cuenta, esta comienza en estado inactivo. Hasta que el administrador no comunica la contraseña al usuario y este ingresa para cambiarla, la cuenta se encuentra en estado inactivo.

El diagrama de la Figura 3.3 describe la modificación de un elemento del diccionario. Si el usuario con rol de editor desea modificar un elemento, primero se verifica si está disponible o bloqueado. Si está bloqueado, no puede editarse. Si está disponible, el editor puede realizar las modificaciones y enviar una propuesta de cambio. La propuesta será revisada por varios revisores, usuarios con rol de arquitecto que deben dar el visto bueno a las propuestas de cambio. Una vez aceptada por todos los revisores, los cambios se consolidarán y el elemento volverá a ser editable. Si algún revisor rechaza la propuesta, deberá justificar el motivo, permitiendo al editor conocer la razón. Tras el rechazo, el elemento quedará de nuevo disponible para su edición.

3.2.2. Modelado de objetos como máquinas de estados

Se han desarrollado dos diagramas que complementan a los diagramas de actividades anteriores.

El diagrama de la Figura 3.4 muestra los diferentes estados en los que se puede encontrar una cuenta de usuario. Una cuenta tiene dos estados simultáneos, es decir, puede estar habilitada o deshabilitada, y a su vez, activa o inactiva. El estado “Enabled” determina si el usuario puede iniciar sesión, mientras que el estado “Active” define si el usuario puede usar la funcionalidad del sistema. Cuando una cuenta es creada, se encuentra habilitada e inactiva. El usuario debe iniciar sesión y cambiar la contraseña para activarla. Si el usuario no cambia su contraseña en 90 días, la cuenta volverá al estado inactivo. Por otra parte, la habilitación o deshabilitación de una cuenta dependerá de que un administrador decida cambiarla de estado.

En la Figura 3.5 se puede apreciar los posibles estados de un elemento dentro del diccionario de datos. Inicialmente, todos los elementos son editables, pero cuando un editor envía una propuesta de cambio, el estado del elemento cambia a bloqueado. La propuesta debe ser aprobada por el número de revisores establecido en el diccionario para volver al estado editable o rechazada por uno de ellos. En este diagrama no se representa, pero si un arquitecto realiza una modificación, esta se consolida de forma inmediata, sin necesidad de revisión.

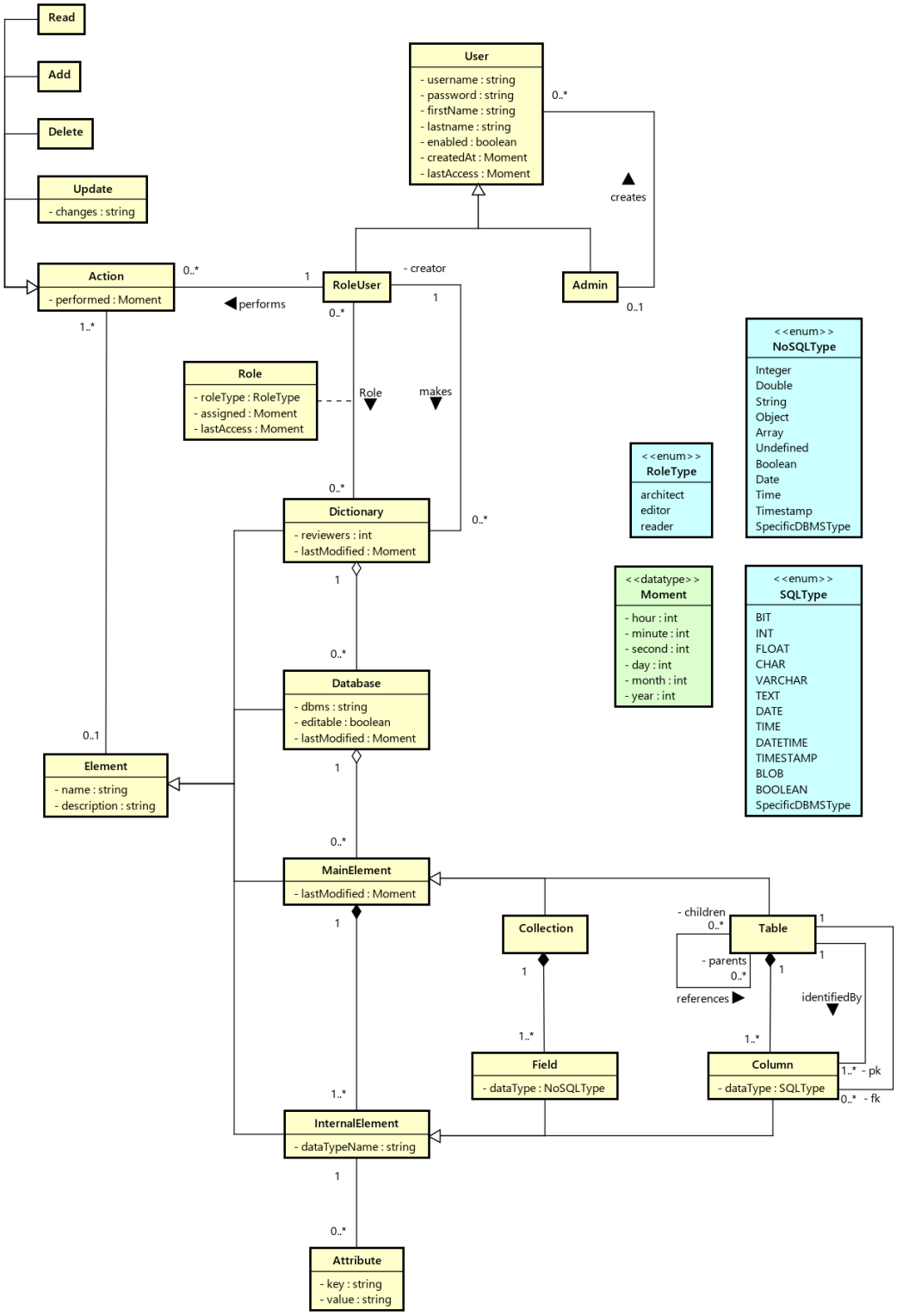


Figura 3.1: Modelo de dominio

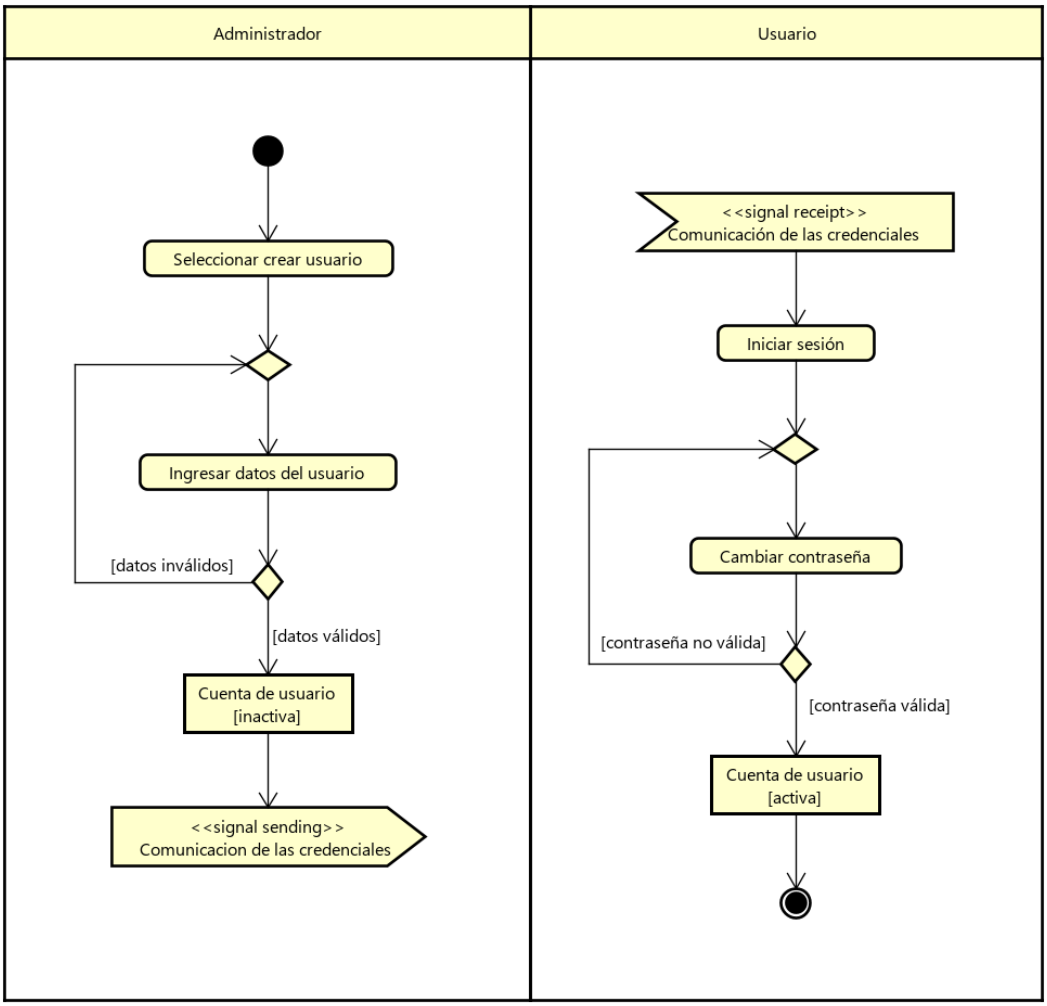


Figura 3.2: Proceso de negocio - Creación y activación de una cuenta de usuario

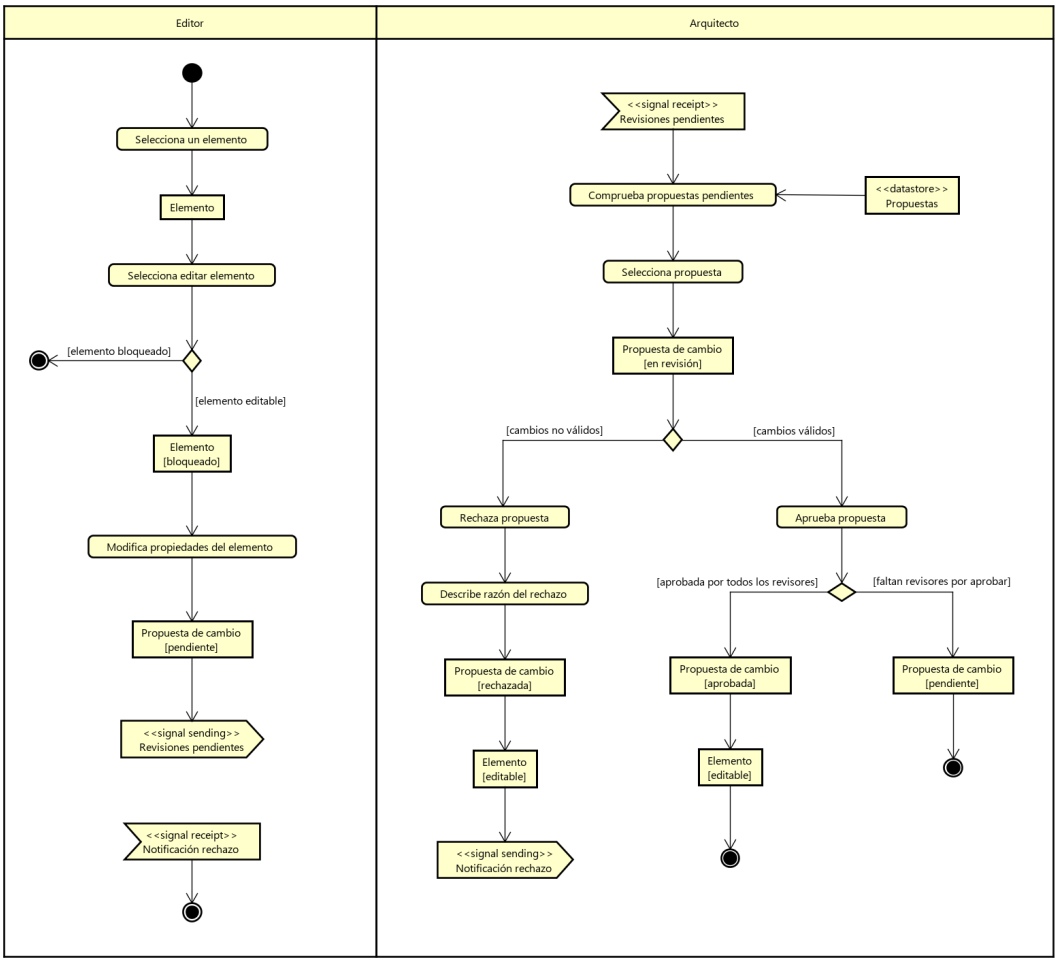


Figura 3.3: Proceso de negocio - Modificación de un elemento de diccionario

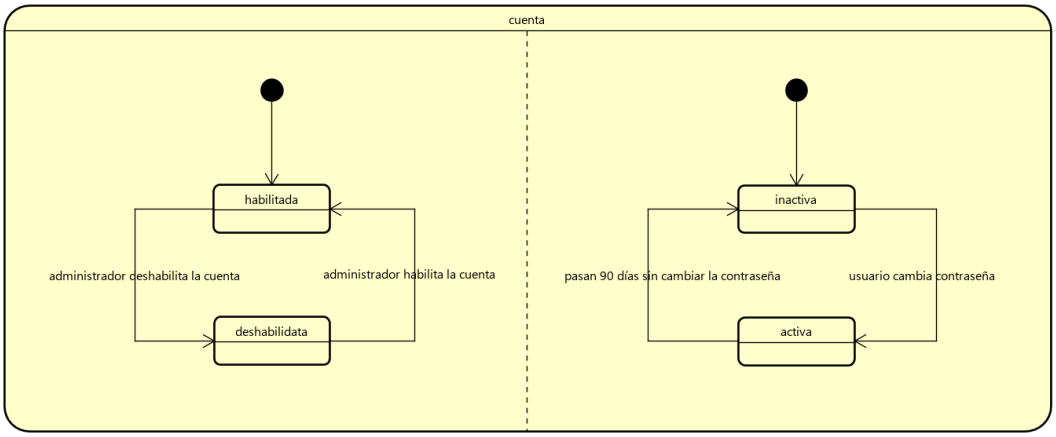


Figura 3.4: Máquina de estados - Cuenta de usuario

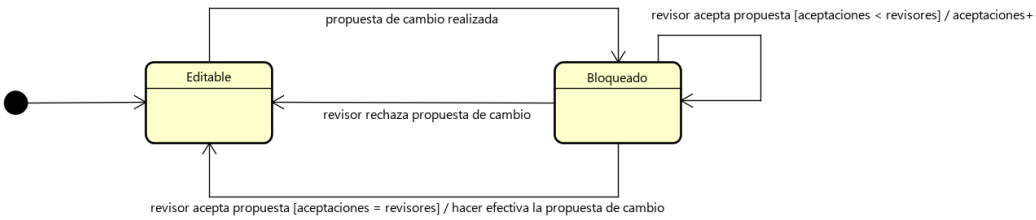


Figura 3.5: Máquina de estados - Elemento del diccionario

Capítulo 4

Tecnologías utilizadas

En este capítulo se presenta un resumen de las tecnologías utilizadas en el proyecto y cómo se han aplicado algunas de ellas en las tareas de gestión, desarrollo y documentación.

4.1. Herramientas de comunicación

4.1.1. Telegram

Telegram es una aplicación de mensajería instantánea gratuita que permite la sincronización en la nube, lo que facilita su uso en varios dispositivos, como teléfonos, tabletas y ordenadores. Esta funcionalidad favorece el intercambio rápido de información a través de mensajes, archivos y enlaces, convirtiéndola en una herramienta eficaz para la comunicación habitual.

4.2. Herramientas de prototipado, análisis y diseño

4.2.1. Figma

Figma es una herramienta de prototipado que sirve para diseñar interfaces web o de aplicaciones. Es una herramienta muy útil en el entorno educativo, ya que permite la colaboración entre los miembros de un equipo haciendo que los cambios sean visibles en tiempo real para todos. Posee un plan gratuito bastante completo, aunque también es posible ampliar la funcionalidad usando una cuenta educativa. Con Figma se pueden realizar prototipos interactivos y de esta forma probar la navegación entre pantallas.

Este programa se ha utilizado para realizar el prototipo de las páginas web desarrolladas, y por tanto, los elementos o colores mostrados en los prototipos pueden no coincidir exactamente con los del diseño final.

4.2.2. Astah Professional

Astah es una herramienta de modelado UML, al igual que Visual Paradigm. Sin embargo, se ha optado por utilizar Astah ya que tiene una interfaz mucho más amigable y por su sencillez. En Astah un proyecto se puede estructurar en varios modelos, como el modelo de análisis y el modelo de diseño, y cada uno se puede se puede dividir a su vez en tantos submodelos o subsistemas como sea necesario. Astah Professional es de pago pero los estudiantes pueden obtener una licencia gratuita para utilizarlo.

Los diagramas que se han utilizado para este proyecto han sido diagramas de clases, de secuencia, de componentes y de despliegue.

4.3. Herramientas de desarrollo y pruebas

4.3.1. Intellij IDEA

IntelliJ IDEA [13] es un IDE creado por JetBrains, diseñado principalmente para el desarrollo en Java y Kotlin. Destaca por su enfoque en la productividad del desarrollador, ofreciendo herramientas avanzadas que facilitan la escritura, depuración y mantenimiento del código.

Entre sus características más útiles se encuentra el autocompletado. También cuenta con potentes herramientas de refactorización que permiten modificar y mejorar el código. Su integración con herramientas como Maven y Gradle facilita la gestión de dependencias y la automatización del proceso de construcción del software. Su sistema de plugins también permite personalizar y extender sus funcionalidades, adaptándose a diferentes necesidades de desarrollo.

4.3.2. GitHub Copilot

GitHub Copilot es un asistente de programación con IA que ayuda a los desarrolladores a escribir código de manera más eficiente. Se integra con editores como IntelliJ IDEA o Visual Studio Code, proporcionando sugerencias contextuales basadas en el código existente. Su objetivo es reducir el tiempo dedicado a tareas repetitivas y permitir que los programadores se enfoquen en la resolución de problemas y la colaboración.

Para integrar GitHub Copilot a IntelliJ IDEA solo es necesario instalar su plugin oficial. Ha resultado especialmente útil para escribir código repetitivo, como constructores, getters

y setters, además de ayudar en la documentación de métodos al describir con precisión su funcionalidad.

4.3.3. Spring Boot

Spring Boot [18] es un framework de código abierto diseñado para simplificar el desarrollo de aplicaciones basadas en el ecosistema Spring. Su objetivo principal es reducir la configuración manual, permitiendo a los desarrolladores centrarse en la lógica de negocio en lugar de en tareas repetitivas de configuración y despliegue.

Spring Boot emplea un sistema interno de servidores de aplicaciones. Para ello, suele usar Tomcat. Con todo esto, sumado a su gestor de dependencias interno, este framework permite compilar las aplicaciones web desarrolladas en un único archivo con formato jar. Esto facilita su distribución y ejecución como cualquier otra aplicación Java.

Se ha utilizado para el desarrollo del back-end.

4.3.4. Angular

Angular [7] es un framework de desarrollo de aplicaciones web desarrollado por Google. Utiliza TypeScript, una versión mejorada de JavaScript.

Se basa en módulos, componentes y servicios. Los módulos agrupan componentes y recursos relacionados, mientras que los componentes son las unidades básicas de la interfaz de usuario. Los servicios, por su parte, proporcionan funcionalidades compartidas, como la gestión de datos y la comunicación con servidores. También cuenta con un sistema de enlace de datos bidireccional, lo que permite que cualquier cambio en los datos se refleje automáticamente en la interfaz de usuario y viceversa.

Se ha utilizado para el desarrollo del front-end.

4.3.5. MySQL

MySQL [17] es un sistema gestor de bases de datos desarrollado por Oracle, basado en el modelo relacional y en el lenguaje de consulta SQL. Es un software de código abierto ampliamente utilizado para almacenar y gestionar datos en aplicaciones web y servicios.

En MySQL, los datos se organizan en tablas y el sistema opera bajo una arquitectura cliente-servidor. La base de datos actúa como un servidor, donde se almacena toda la información, mientras que el software funciona como un cliente que permite a los usuarios realizar consultas, conocidas como “queries”, en SQL. Estas consultas son procesadas por el sistema de base de datos, facilitando el acceso y manejo de los datos.

En este proyecto, MySQL se utiliza como base de datos y Spring Boot interactúa con ella a través de Java Persistence API (JPA).

JPA es un estándar que permite trabajar con bases de datos relacionales de forma abstracta, definiendo clases especiales denominadas entidades que representan los registros de las tablas. Además, en Spring se añade otra capa de abstracción llamada **Spring Data JPA**, una extensión que simplifica aún más el acceso a los datos mediante interfaces llamadas repositorios, donde se pueden declarar métodos como búsquedas o actualizaciones [30].

4.3.6. Jasmine

Jasmine es un framework de testing para JavaScript que se usa en Angular. Da soporte a la metodología **Behavior Driven Development** (**BDD**), lo que significa que se centra en describir el comportamiento esperado del código. Jasmine puede ejecutarse tanto en navegadores como en Node.js, pero en el entorno de Angular, suele utilizarse junto con **Karma**, que es el test-runner encargado de ejecutar las pruebas de manera automatizada [15].

En el capítulo 6 se explica con detalle la sintaxis de los tests, con ejemplos concretos de tests realizados para probar el funcionamiento de la aplicación.

Figura 4.1: Jasmine + Karma. Imagen tomada de [45]

4.4. Herramientas de gestión y documentación

4.4.1. Overleaf

Overleaf es un editor de LaTeX online que se puede usar directamente desde el navegador, sin necesidad de instalar nada. Ofrece dos formas de edición, mediante código o a través de un editor visual, lo que permite trabajar sin tener un conocimiento profundo de LaTeX.

Es una herramienta gratuita, aunque requiere tener una cuenta para poder utilizarla. En su versión gratuita permite compartir los proyectos con una persona y la edición se realiza de forma colaborativa en tiempo real. Sin embargo, para este proyecto se ha utilizado una licencia educativa compartida por la tutora.

Además, Overleaf permite compilar el documento para generar el PDF y visualizarlo. En

la Figura 4.2 se muestra la interfaz de Overleaf, que incluye tanto el editor como el visor del documento.

Todo esto convierte a Overleaf en una opción ideal para redactar la memoria del proyecto, ya que facilita la revisión conjunta y permite ver los cambios al instante.

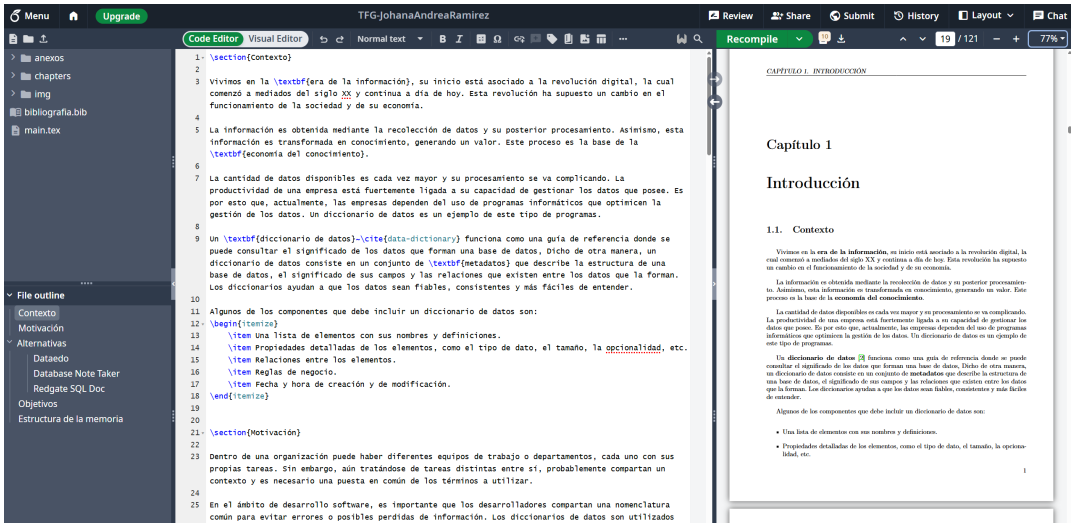


Figura 4.2: Interfaz de Overleaf

4.4.2. ChatGPT

ChatGPT es un modelo de lenguaje desarrollado por OpenAI. Se trata de un sistema basado en aprendizaje automático, entrenado con grandes cantidades de información, capaz de procesar y generar texto en lenguaje natural. Puede responder preguntas, explicar conceptos, traducir, resolver dudas de programación, entre otras tareas, todo ello manteniendo una conversación fluida con el usuario.

En este proyecto, ChatGPT se ha utilizado como herramienta de apoyo en la redacción de la memoria, ayudando a reformular párrafos y a mejorar la fluidez y coherencia del texto. En la Figura 4.3 se muestra cómo se ha utilizado ChatGPT para la redacción de este apartado. El tipo de prompt utilizado es "... mejorar el texto para que suene más fluido".

4.4.3. Git

Un **sistema de control de versiones** (**VCS**) es un tipo de software que permite hacer un seguimiento de los cambios realizados en los archivos de un proyecto a lo largo del tiempo. Cuando se realizan cambios en los archivos, el sistema guarda una instantánea de esos archivos. Estas instantáneas se almacenan de forma permanente y pueden recuperarse en cualquier momento.

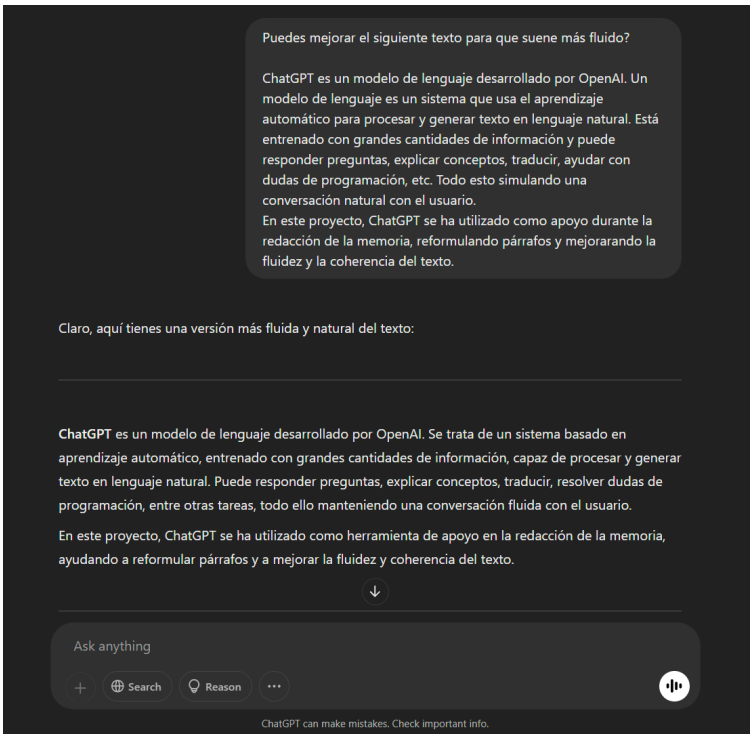


Figura 4.3: Ejemplo de uso de ChatGPT

Git, en particular, es un sistema de control de versiones **distribuido**. A diferencia de otros sistemas donde existe un repositorio centralizado, en Git cada desarrollador tiene una copia completa del proyecto en su máquina local. Esto permite trabajar sin necesidad de conexión constante con un servidor, ya que las modificaciones se pueden hacer localmente y luego sincronizar con el repositorio central.

Aspectos básicos de Git [6]:

- **Confirmaciones** (Commits): Son las instantáneas y se crean con el comando 'git commit'. Si un archivo no ha cambiado de una confirmación a la siguiente, Git usa el archivo almacenado anteriormente. Este enfoque difiere de otros sistemas, que guardan solo las diferencias entre versiones. Es posible revertir el código a una confirmación anterior, inspeccionar cómo cambian los archivos de una confirmación a la siguiente y revisar información como dónde y cuándo se realizaron los cambios. Las confirmaciones se identifican en Git mediante un hash criptográfico único. Dado que todo tiene hash, es imposible realizar cambios y perder o dañar la información sin que Git lo detecte.
- **Ramas** (Branches): Como cada desarrollador trabaja en su propio repositorio local, puede haber muchos cambios diferentes basados en la misma confirmación. Las ramas son punteros ligeros para el trabajo en curso. Una vez finalizado el trabajo creado en una rama, se puede combinar de nuevo en la rama principal.

- **Estados**: Los archivos en Git pueden estar en tres estados:
  - **Modificados**: El archivo ha sido cambiado, pero los cambios aún no forman parte de una confirmación.
  - **Almacenados provisionalmente**: Los cambios están preparados para ser confirmados.
  - **Confirmados**: Los cambios pasan a ser definitivos y a formar parte del historial de desarrollo.

4.4.4. Gitlab

Gitlab es una plataforma web de código abierto que permite gestionar proyectos de desarrollo de software utilizando Git como sistema de control de versiones y que sirve también como repositorio online para almacenar proyectos. Ofrece un conjunto de herramientas integradas que cubren todo el ciclo de vida del desarrollo, desde la planificación y la colaboración entre desarrolladores, hasta la integración y el despliegue continuo (CI/CD).

En este proyecto, se ha utilizado GitLab como repositorio remoto, lo que ha permitido clonar el proyecto en diferentes dispositivos, como el ordenador utilizado para el desarrollo y la máquina virtual donde se ha realizado el despliegue. Además, se han aprovechado las herramientas de planificación propias de GitLab, como los **issues** e **issue boards**, para organizar los sprints.

Issues

Los issues [21] ayudan a la planificación, seguimiento y entrega del trabajo dentro de un proyecto en GitLab. Permiten gestionar propuestas de funcionalidades, tareas, solicitudes de soporte y reportes de errores.

En este proyecto, los issues se han utilizado para representar épicas, historias de usuario y otro tipo de tareas denominadas hotfix. Se ha creado un issue por cada épica e historia de usuario definida en el Capítulo 2, mientras que los hotfix se han ido generando durante el desarrollo y hacen referencia a tareas necesarias para corregir o completar funcionalidades existentes.

Las épicas se identifican con títulos en el formato "EPXX – <Nombre>" y llevan la etiqueta "Epic". Están bloqueadas por los issues de sus historias de usuario asociadas y no deben cerrarse hasta que todas estas hayan sido completadas. En la Figura 4.4 puede verse el issue correspondiente a la épica EP02.

Las historias de usuario siguen el formato "HUXX – <Nombre>" y están relacionadas con el issue de su correspondiente épica. En cada issue se indica una estimación del tiempo de desarrollo y, una vez finalizada, se registra el tiempo real empleado. En la Figura 4.5 se muestra el issue de la historia de usuario HU30.

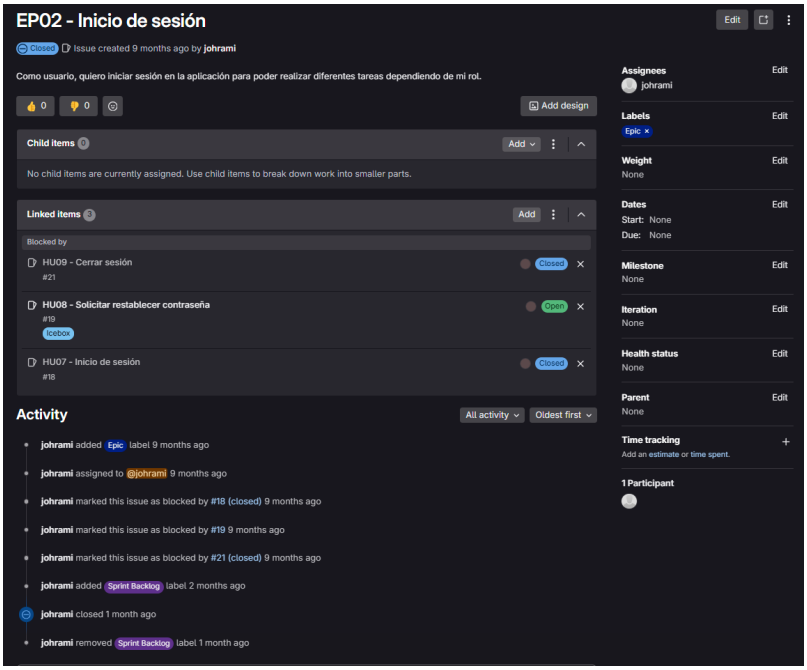


Figura 4.4: Issue correspondiente a la épica EP02

Por último, las tareas hotfix se identifican mediante el formato "HFXX – <Nombre>" y llevan la etiqueta "Hotfix". También incluyen una estimación y un tiempo real de desarrollo, aunque no están relacionados con ningún otro issue.

Issue boards

Los issue boards [20] son tableros que ofrecen una forma visual de gestionar y supervisar las tareas en un proyecto. Los issues se presentan como tarjetas organizadas en columnas. Cada columna contiene aquellos issues que cumplan con la condición seleccionada, como una etiqueta, un hito o una persona asignada. Este sistema permite visualizar el avance a lo largo de las distintas etapas del flujo de trabajo y es compatible con Scrum o Kanban.

Para ajustar el tablero al marco de trabajo Scrum, se ha organizado en cinco columnas:

- **Open**: Contiene todas las tareas al inicio del proyecto. Funciona como el product backlog de Scrum.
- **Sprint Backlog**: Agrupa las tareas seleccionadas para ser completadas en el sprint actual.
- **In Progress**: Representa las tareas en desarrollo. En general, solo habrá una tarea en esta columna a la vez.

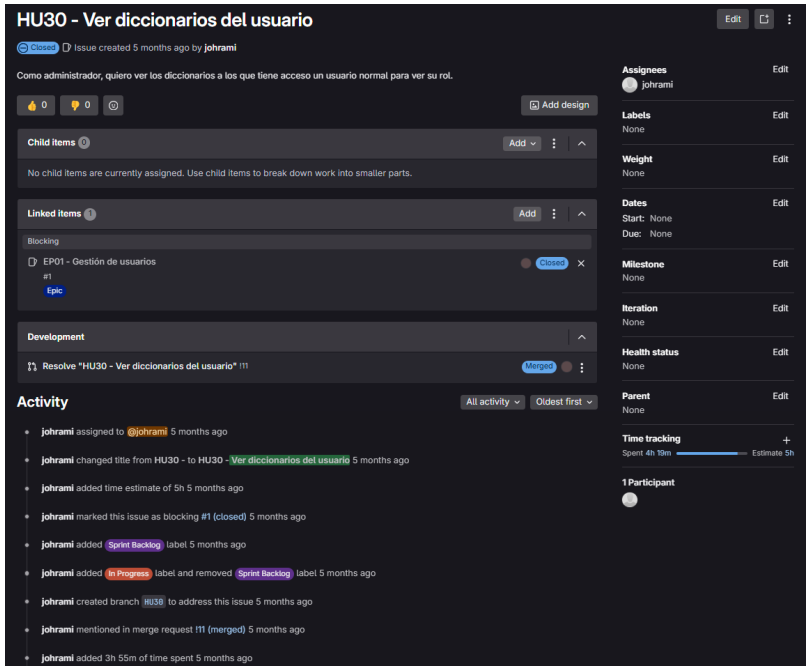


Figura 4.5: Issue correspondiente a la historia de usuario HU30

- **Pending Review**: Aquí se encuentran las tareas completadas, pero pendientes de revisión. La revisión se realiza al final del sprint durante el sprint review. Si es necesario hacer ajustes, las tareas pueden volver al sprint backlog.
- **Closed**: Contiene las tareas que se consideran finalizadas.

La Figura 4.6 muestra una captura del tablero durante el desarrollo de uno de los sprints, con las columnas organizadas según lo descrito previamente.

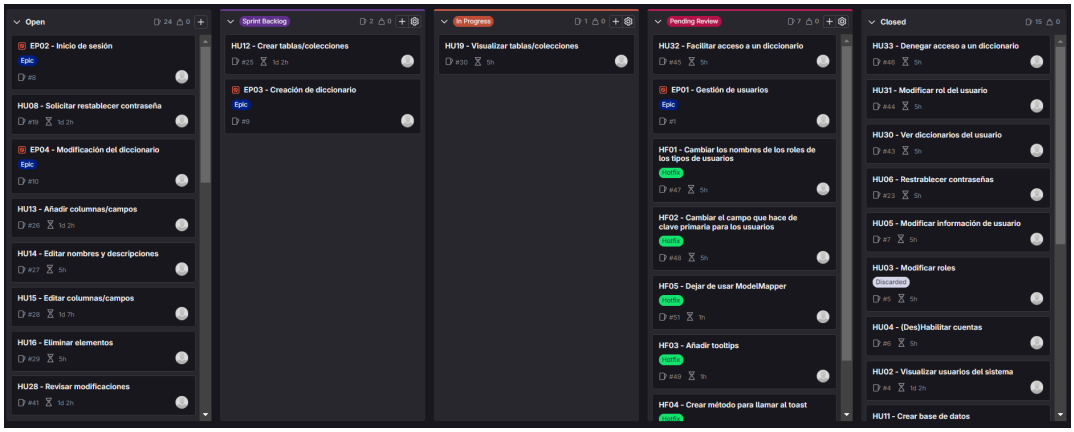


Figura 4.6: Issue board del proyecto

Capítulo 5

Diseño

En este capítulo se describe el diseño de la aplicación, incluyendo el diseño de la interfaz de usuario, la arquitectura utilizada, el diseño de datos y el despliegue de la aplicación.

5.1. Arquitectura

El diseño de la aplicación sigue una arquitectura **cliente-servidor**, con un back-end **monolítico** que centraliza la lógica de negocio, el acceso a datos y la seguridad, y un front-end basado en **componentes** y **servicios**

5.1.1. Arquitectura cliente-servidor

La arquitectura cliente-servidor [42] permite el procesamiento cooperativo de la información mediante la interacción entre clientes y servidores. En esta estructura, uno o varios clientes solicitan servicios a uno o más servidores. El acceso a los servicios se realiza de manera transparente, es decir, los usuarios no perciben la cantidad de servidores involucrados en el proceso. Es una de las arquitecturas más utilizadas en el desarrollo de aplicaciones.

- **Cliente** (**Front-end**): Es todo proceso que solicita servicios de otro proceso, en este caso, un servidor. Maneja la interfaz de usuario, gestionando tanto la presentación como la interacción con los datos. Las funciones principales del proceso cliente son:
  - Administrar la interfaz de usuario.
  - Interactuar con el usuario.
  - Hacer validaciones locales.
  - Generar peticiones.

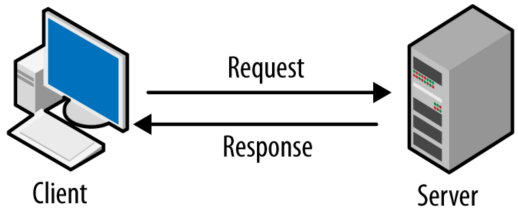


Figura 5.1: Esquema cliente-servidor. Imagen tomada de [11]

  - Recibir y formatear resultados.

- **Servidor** (**Back-end**): Es cualquier proceso que proporciona servicios a otros. Su tarea es atender a múltiples clientes que solicitan los recursos que proporciona. El servidor gestiona la lógica de negocio y el acceso a los datos. Dependiendo del tipo de servidor, puede ofrecer una API para facilitar la comunicación con los clientes o interactuar directamente con bases de datos u otros sistemas. Sus funciones principales son:

  - Aceptar o rechazar solicitudes de los clientes.
  - Procesar peticiones aplicando la lógica de la aplicación.
  - Formatear datos para trasmitirlos a los clientes.
  - Realizar validaciones a nivel de bases de datos.

El esquema de funcionamiento de un sistema cliente-servidor se resume en los siguientes pasos y se esquematiza en la Figura 5.1:

1. El cliente envía una solicitud al servidor.
2. El servidor recibe la solicitud del cliente.
3. El servidor procesa la solicitud.
4. El servidor genera una respuesta y se la envía al cliente.
5. El cliente recibe la respuesta y la procesa.

Aplicación en el Proyecto

El sistema sigue una arquitectura **cliente-servidor**, con el front-end desarrollado en **Angular** y el back-end utilizando **Spring Boot**. Para la comunicación entre ambos, el back-end expone una **API RESTful** que el front-end consume mediante **peticiones HTTP**.

El back-end se ha diseñado siguiendo un enfoque por **capas**. El servidor también ejecuta la base de datos **MySQL**. Para garantizar la seguridad de los recursos, la autenticación se gestiona mediante **JWT** (JSON Web Token), el cual debe incluirse en todas las peticiones

HTTP realizadas por el cliente. Esto asegura la protección de los endpoints del servidor mediante los filtros de **Spring Security**, garantizando que solo los usuarios autenticados puedan acceder a los servicios.

En el front-end, se ha utilizado **Angular Material** para la construcción de las interfaces de usuario. Para asegurar que todas las peticiones al servidor incluyan el token de autenticación, se emplea un **interceptor** que lo añade automáticamente a las cabeceras de las solicitudes HTTP. Además, también se utilizan **guards** para proteger las rutas y controlar la interacción del usuario con determinadas páginas, permitiendo el acceso únicamente a usuarios autenticados y con roles específicos.

5.1.2. Arquitectura del servidor: Patrón capas

La **arquitectura en capas** [2], también conocida como arquitectura de múltiples capas (**multilayer**), es un patrón arquitectónico ampliamente utilizado. Consta de varias capas, cada una con un conjunto específico de responsabilidades en el contexto de la aplicación. Esto se muestra esquemáticamente en la Figura 5.2.

Las capas están organizadas de forma jerárquica unas encima de otras y las dependencias siempre van hacia abajo. Es decir, que una capa concreta dependerá solamente de las capas inferiores, pero nunca de las superiores. Podemos diferenciar además entre sistemas estrictos y relajados [8] en función de las relaciones de dependencia con las capas inferiores. Un **sistema estricto** es aquel en el que una capa solo depende directamente de la capa inmediatamente inferior, mientras que en un **sistema relajado** puede hacerlo de todas las que hay por debajo, aunque no sean contiguas.

Ventajas de la arquitectura en capas:

- **Modularidad**: La separación de preocupaciones hace que el código sea más comprensible, organizado y manejable. Fomenta la reutilización de componentes en diferentes aplicaciones.
- **Escalabilidad**: Cada capa se puede escalar y optimizar de forma independiente de acuerdo con los requisitos.
- **Mantenibilidad**: Las modificaciones pueden realizarse en componentes específicos sin afectar a todo el sistema.
- **Capacidad de prueba**: Se puede probar cada capa de forma individual.
- **Interoperabilidad**: Facilita la integración y comunicación entre diferentes sistemas y servicios.

Aplicación en el Proyecto

El servidor sigue una arquitectura por capas, organizada en **tres capas estrictas** y **una relajada**. Esto permite una separación clara de las responsabilidades y un diseño mantenible.

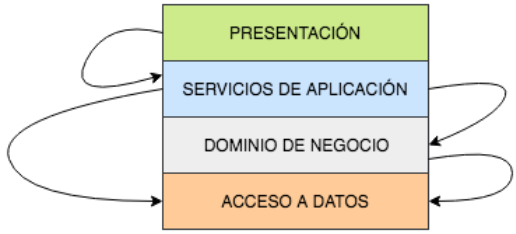


Figura 5.2: Sistema relajado de capas. Imagen tomada de [8]

La arquitectura del servidor se puede apreciar en la Figura 5.3

Capas estrictas:

- **controllers** (Capa superior): Esta capa se encarga de gestionar las solicitudes entrantes del cliente. Su función principal es recibir las peticiones, determinar qué servicio debe procesarlas y, finalmente, construir una respuesta para el cliente.
- **business** (Capa intermedia): Agrupa los servicios que contienen la lógica de negocio del sistema. Se encarga de orquestar las acciones necesarias para procesar correctamente la solicitud del cliente. Además, maneja los errores que puedan surgir durante el procesamiento.
- **persistence** (Capa inferior): Es la capa encargada de gestionar los datos persistentes, definiendo la estructura de las entidades y facilitando su almacenamiento y recuperación desde la base de datos.

Capa relajada:

- **common services**: Esta capa contiene funcionalidades comunes que pueden ser reutilizadas por las demás capas del sistema, así como configuraciones generales de la aplicación.

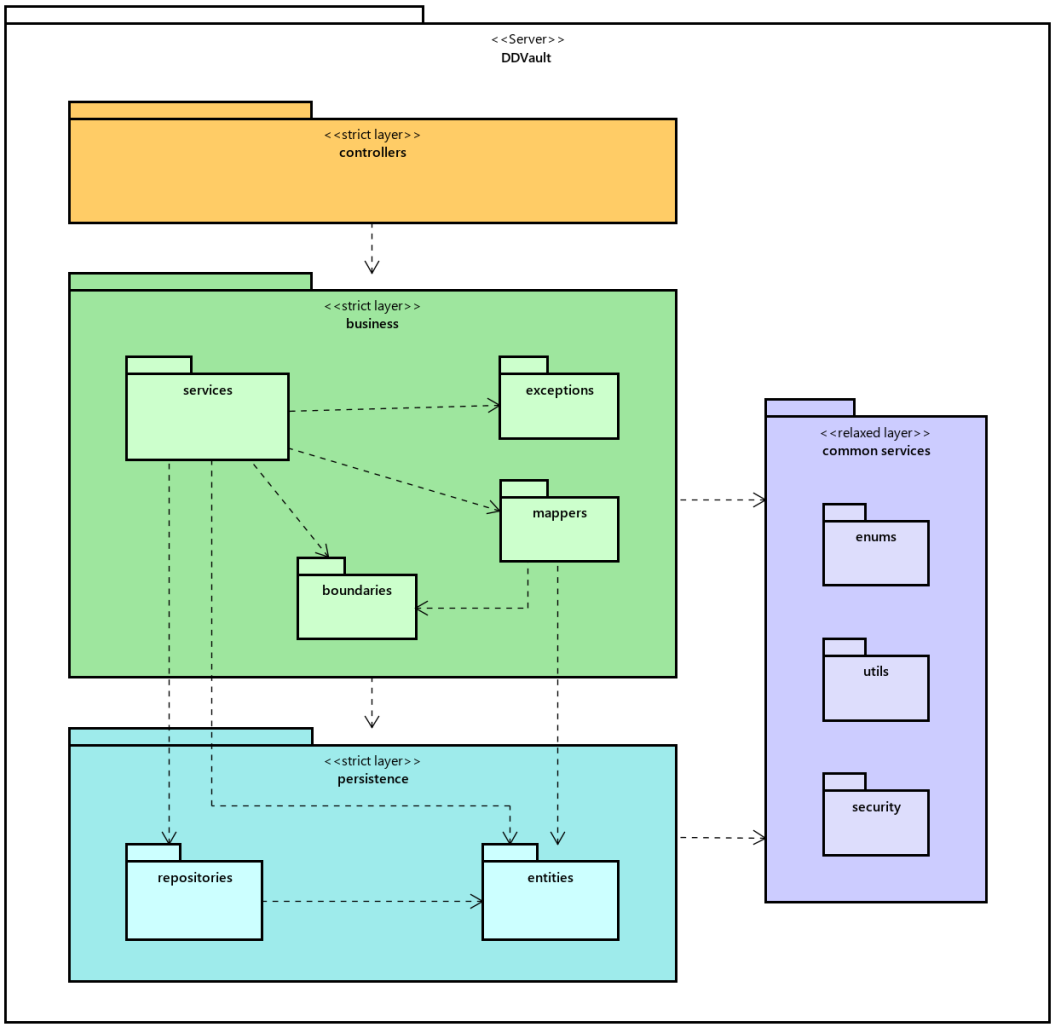


Figura 5.3: Arquitectura general del servidor

5.1.3. Arquitectura del cliente: Patrón MVVM

El patrón **Modelo-Vista-Modelo de Vista** (**MVVM**) es una arquitectura que busca separar la lógica de la aplicación de su interfaz de usuario , facilitando el mantenimiento, las pruebas y la evolución de la aplicación [5].

MVVM consta de tres componentes principales [14]:

- **Modelo**: Representa la lógica de negocio y los datos de la aplicación. Se encarga de gestionar y recuperar la información.
- **Vista**: Se encarga de la presentación visual. Muestra los datos procesados por el modelo de vista y capta las acciones del usuario para transmitirlas al modelo de vista.
- **Modelo de Vista**: Actúa como intermediario entre el modelo y la vista. Prepara los datos del modelo para ser presentados en la vista y maneja las interacciones del usuario. Mantiene la vista sincronizada con los datos.

La interacción entre estos componentes es la siguiente: la vista depende del modelo de vista y el modelo de vista depende del modelo, pero ni la vista conoce al modelo, ni el modelo conoce al modelo de vista. Esto se muestra esquemáticamente en la Figura 5.4.

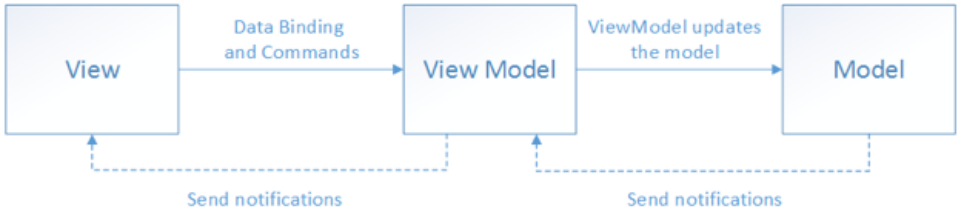


Figura 5.4: Relación entre los componentes del patrón MVVM [5]

Aplicación en el Proyecto

Aunque Angular no sigue estrictamente el patrón MVVM, se puede adaptar a él mediante su estructura basada en componentes y servicios:

- Modelo: **Servicios** o clases responsables de gestionar los datos, realizar operaciones y comunicarse con las APIs.
- Vista: **Plantillas HTML** dentro de los componentes, que definen cómo se presentan los datos del modelo al usuario.
- Modelo de Vista: **Clases TypeScript** asociadas a los componentes. El enlace entre la vista y el modelo de vista se logra mediante el **data binding**.

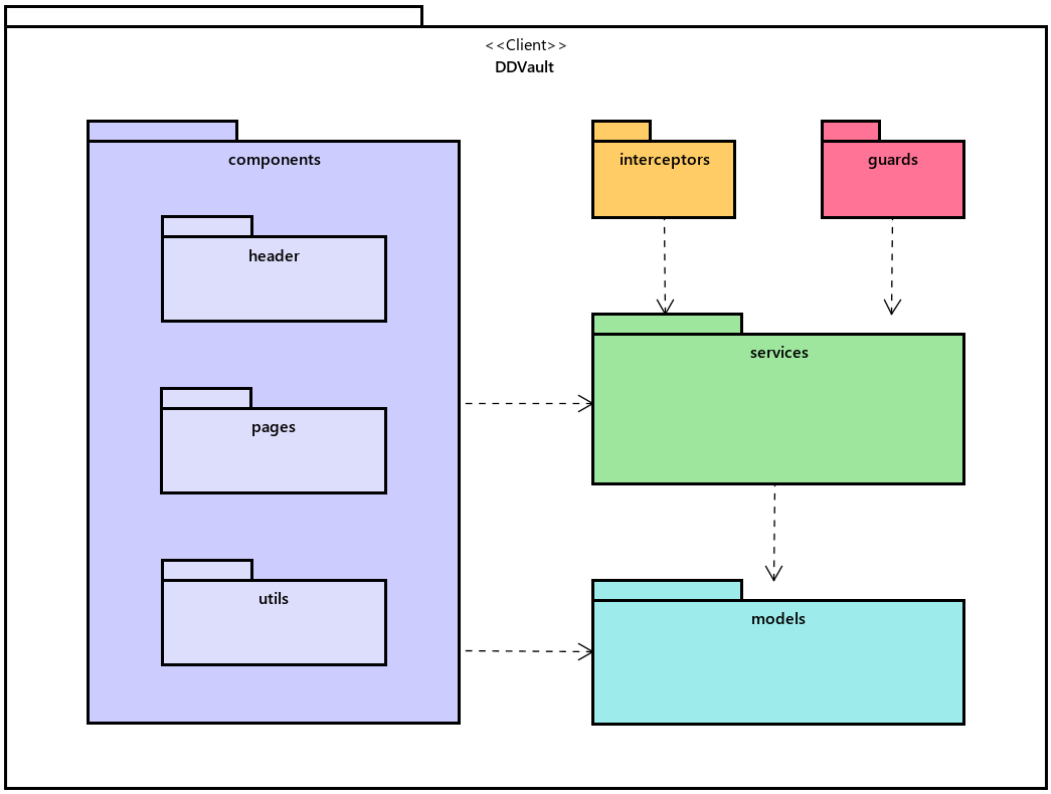


Figura 5.5: Arquitectura general del cliente

En la Figura 5.5 se presenta el diagrama con la arquitectura del cliente, en el que se puede observar la siguiente organización:

- **interceptors**: Son clases que interceptan las solicitudes HTTP antes de que sean enviadas al servidor. En este caso, se utilizan para añadir el token de autenticación en las cabeceras de cada petición, asegurando que el usuario esté correctamente autenticado.
- **guards**: Se encargan de proteger las rutas, evitando que usuarios sin el rol adecuado accedan a ciertas páginas.
- **services**: Son clases que contienen los métodos necesarios para interactuar con el backend o para otras funciones como la configuración de idioma.
- **components**: Los componentes se agrupan en tres grupos según su uso:
  - **header**: Componentes que forman la cabecera de la aplicación, visible en todas las páginas.
  - **pages**: Cada página tiene su propio componente, que puede estar compuesto por varios subcomponentes, organizados jerárquicamente.

    - **utils**: Componentes que pueden reutilizarse en diferentes páginas.

- **models**: Contienen las interfaces y los enumerados que definen la estructura de los datos utilizados en la aplicación.

Diseño basado en componentes

Los componentes son la unidad fundamental de construcción de una aplicación Angular [1]. No deben confundirse con los componentes del patrón MVVM. Cada componente representa una parte concreta de la interfaz y encapsula tanto su lógica como su apariencia.

Todo componente está compuesto por:

- Una clase TypeScript con su comportamiento.
- Una plantilla HTML que controla lo que se renderiza en el DOM.
- Un selector CSS que define cómo se usa el componente en otras plantillas HTML.

Una aplicación Angular se construye mediante la composición de componentes, lo cuales pueden anidarse entre sí para formar vistas más complejas. Esta estructura facilita el aislamiento de funcionalidades, lo que contribuye a que el código sea más mantenible y escalable.

En la Figura 5.6 se muestran todos los componentes utilizados para construir la interfaz de la aplicación. Las líneas de dependencia representan la jerarquía entre componentes. El componente “app” es el componte raíz y “header” es un componente que está siempre visible en la parte superior de la interfaz. Los componentes en color azul representan páginas completas, mientras que los de color rosa corresponden a subcomponentes de dichas páginas. Los componentes reutilizables, representados en color verde, son aquellos que pueden ser utilizados por cualquier otro componente.

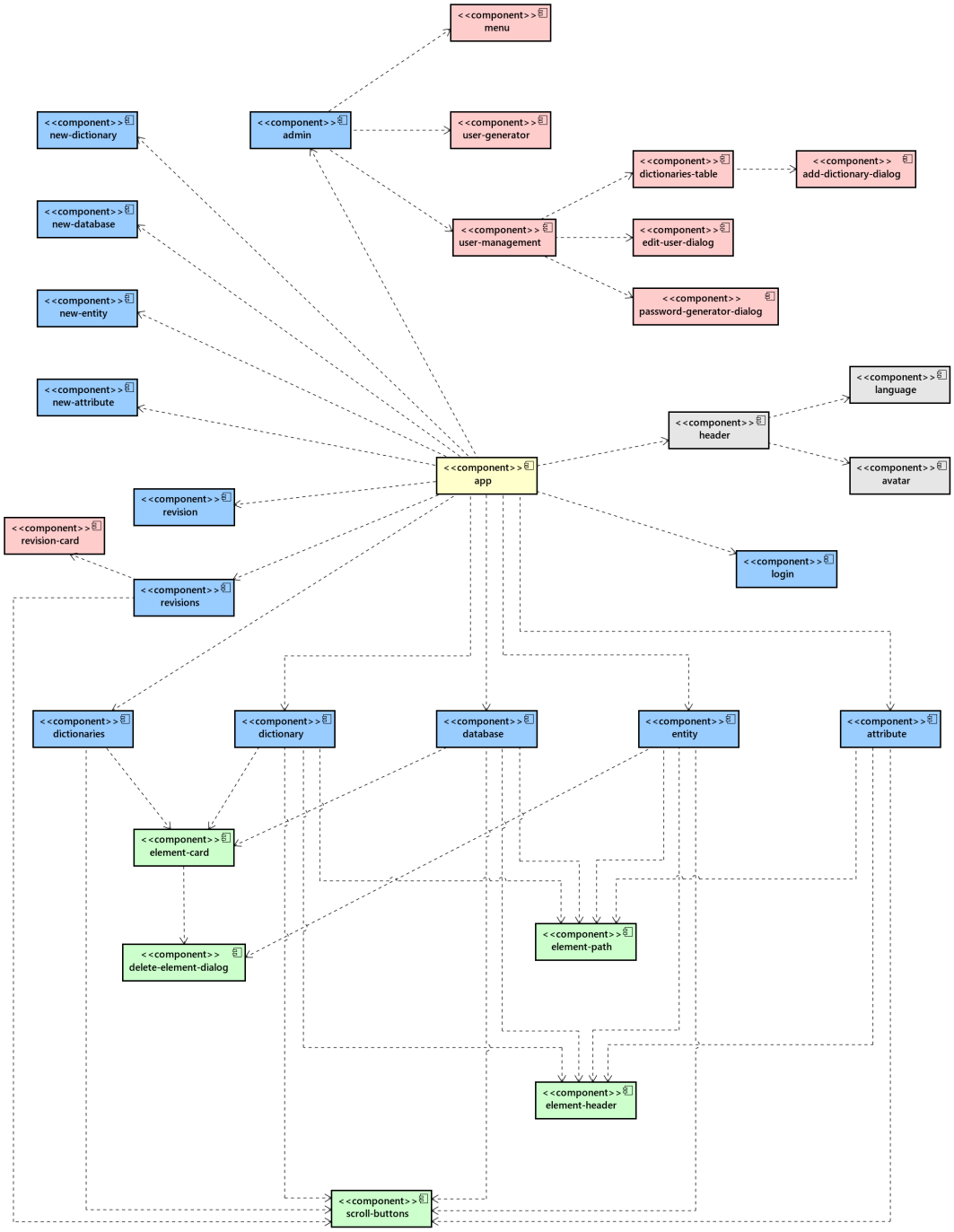


Figura 5.6: Diagrama de componentes

5.2. Diseño de la interfaz de usuario

A continuación, se muestran los prototipos creados para las diferentes páginas de la aplicación. Estos prototipos se han ido creando y completando durante las tareas de diseño de cada historia de usuario. Una sola página puede ser producto de una o varias historias de usuario o las historias de usuario más complejas han necesitado la creación de más de una página.

El inicio de sesión, Figura 5.7, es la primera página con la que interactúa el usuario. Es la única página accesible para todos. Una vez iniciada la sesión, el usuario será redirigido a una de las dos páginas de inicio, la página de inicio del administrador, Figura 5.8, o la página de inicio del gestor de metadatos, Figura 5.9.

En la pantalla de inicio, los administradores pueden elegir entre dos opciones que modifican parcialmente el contenido mostrado. Seleccionando "Crear usuario", se mostrará el formulario de la Figura 5.10, que permite registrar nuevos administradores o gestores de metadatos. Por otro lado, al seleccionar "Gestionar usuarios" (Figura 5.11), se mostrará una lista con todos los usuarios del sistema. Al hacer clic en un usuario, su fila se expandirá para revelar opciones adicionales.

Para las funcionalidades "Editar información", "Restablecer contraseña" y "Añadir diccionario" se ha optado por el uso de ventanas de diálogo, ya que son tareas simples que no justifican la creación de páginas individuales, evitando así una complejidad innecesaria. Las figuras 5.12, 5.13 y 5.14 muestran las interfaces diseñadas para cada una de estas funciones, respectivamente.

El gestor de metadatos puede acceder a tres grupos de ventanas en el sistema.

El primer grupo corresponde a las ventanas que muestran el contenido de los distintos elementos del diccionario de datos: diccionario (Figura 5.15), base de datos (Figura 5.16), entidad (Figura 5.17) y atributo (Figura 5.18). En estas ventanas, el nombre y la descripción de cada elemento pueden editarse directamente. Al activar el modo edición mediante un botón, los campos se vuelven editables.

El segundo grupo lo conforman las páginas de creación de elementos. Estas presentan formularios similares, pero adaptados a los distintos tipos de elementos, ya que cada uno requiere información específica. La Figura 5.19 muestra la interfaz para crear un diccionario y la Figura 5.20 muestra la de una base de datos.

Por último, están las páginas destinadas a la revisión de propuestas. Primero se encuentra una lista con las revisiones pendientes, aceptadas o rechazadas (historial de revisiones), así como a las propuestas realizadas por el propio usuario (Figura 5.21). Al seleccionar una propuesta, se abre una ventana con información detallada sobre los cambios propuestos y otros datos relevantes. Aunque la ventana de la propuesta es la misma, su contenido varía según su estado y el rol del usuario que la visualiza. En la Figura 5.22 se muestra un ejemplo de una propuesta pendiente desde la perspectiva de un revisor.

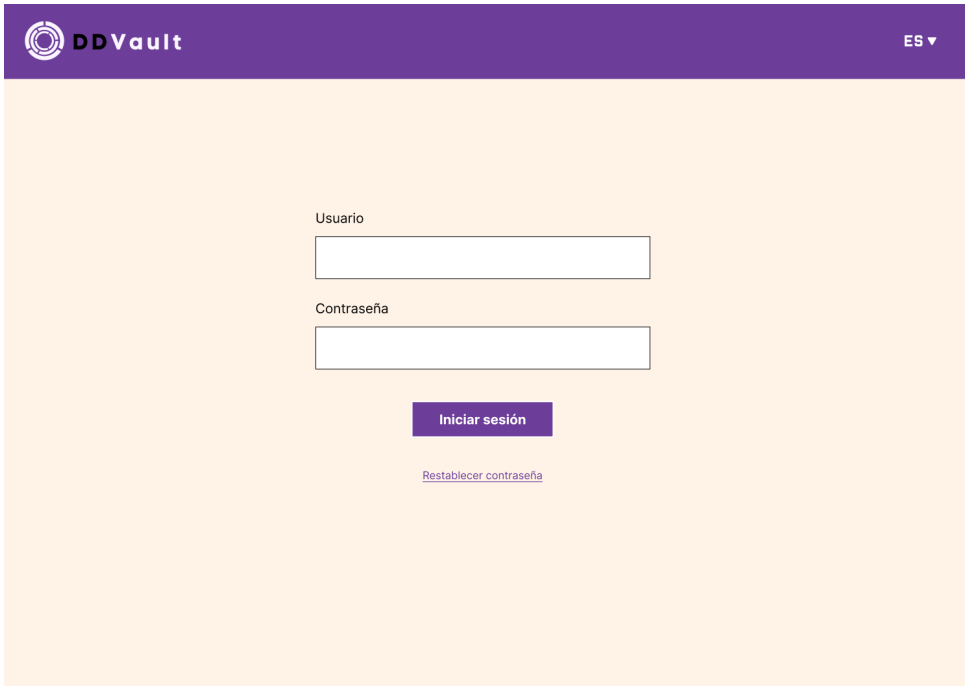


Figura 5.7: Pantalla inicio de sesión

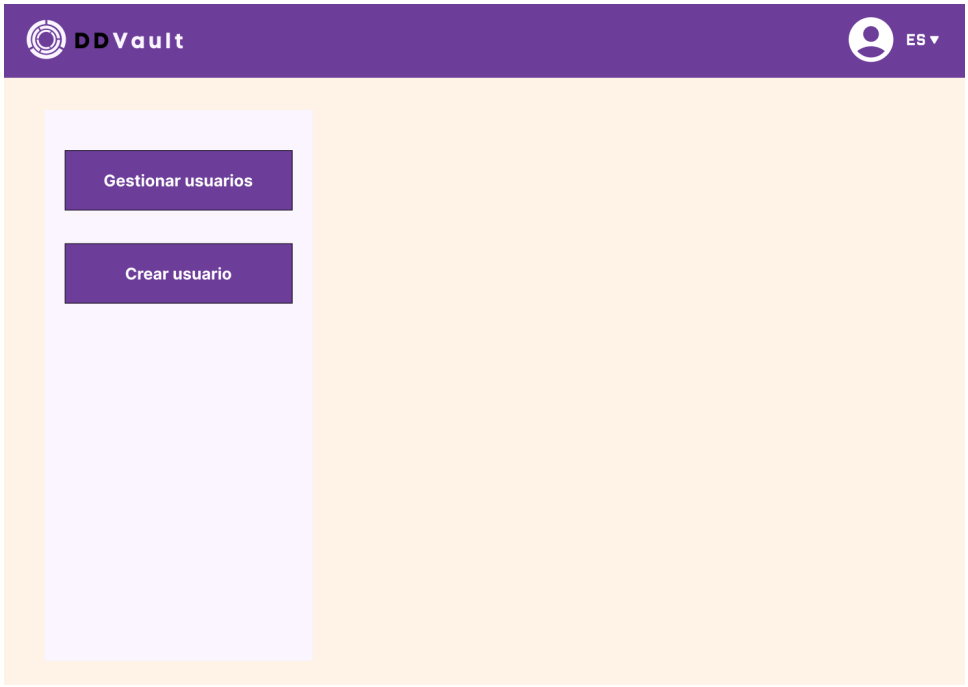


Figura 5.8: Pantalla inicio del administrador

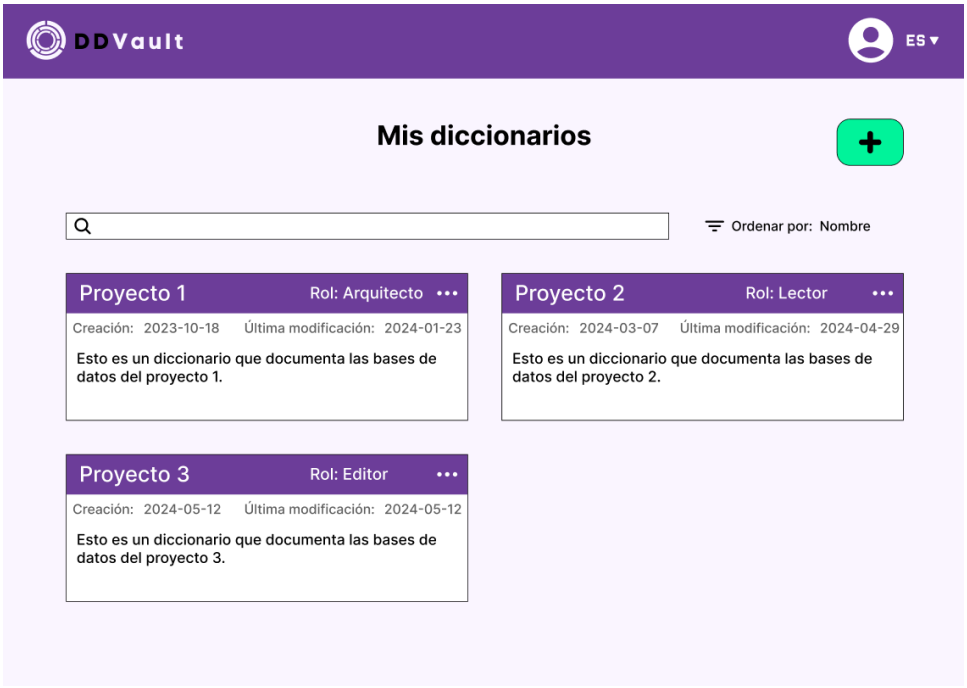


Figura 5.9: Pantalla inicio del gestor de metadatos

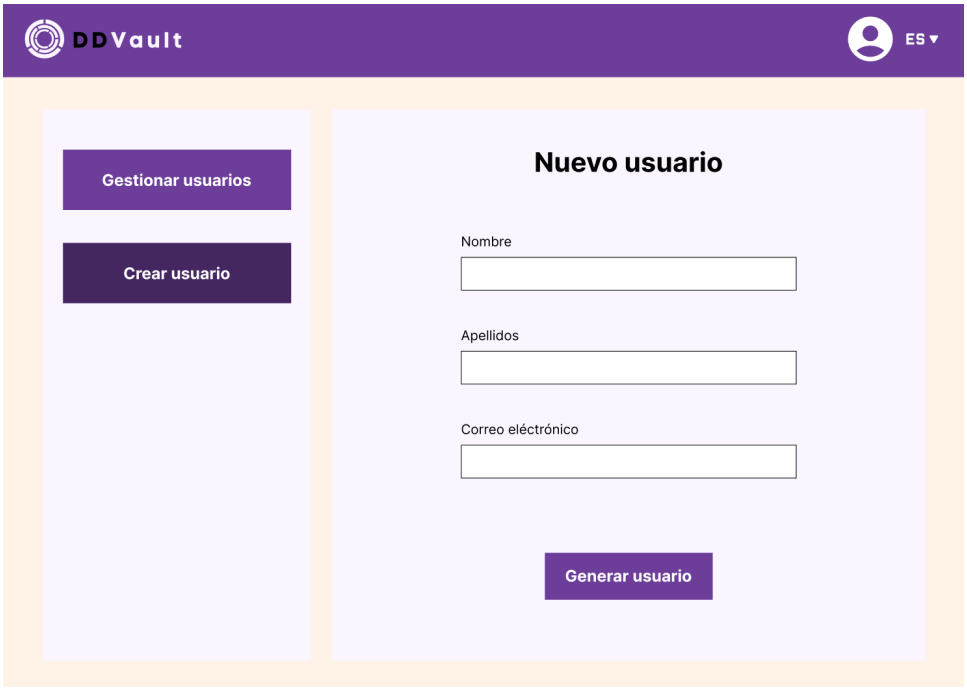


Figura 5.10: Pantalla para la creación de usuarios

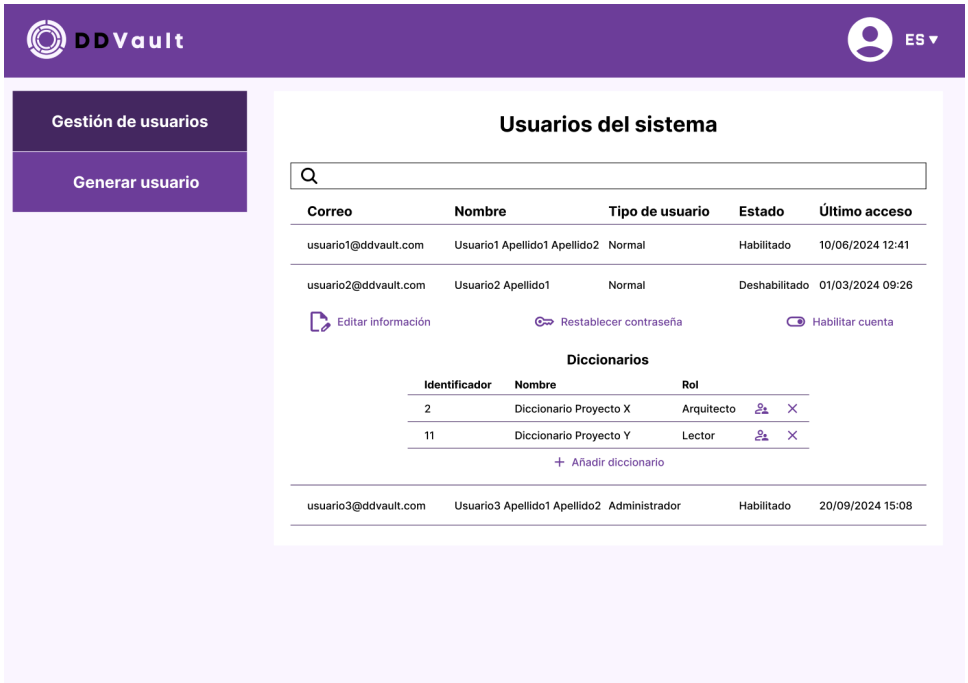


Figura 5.11: Pantalla para la gestión de usuarios

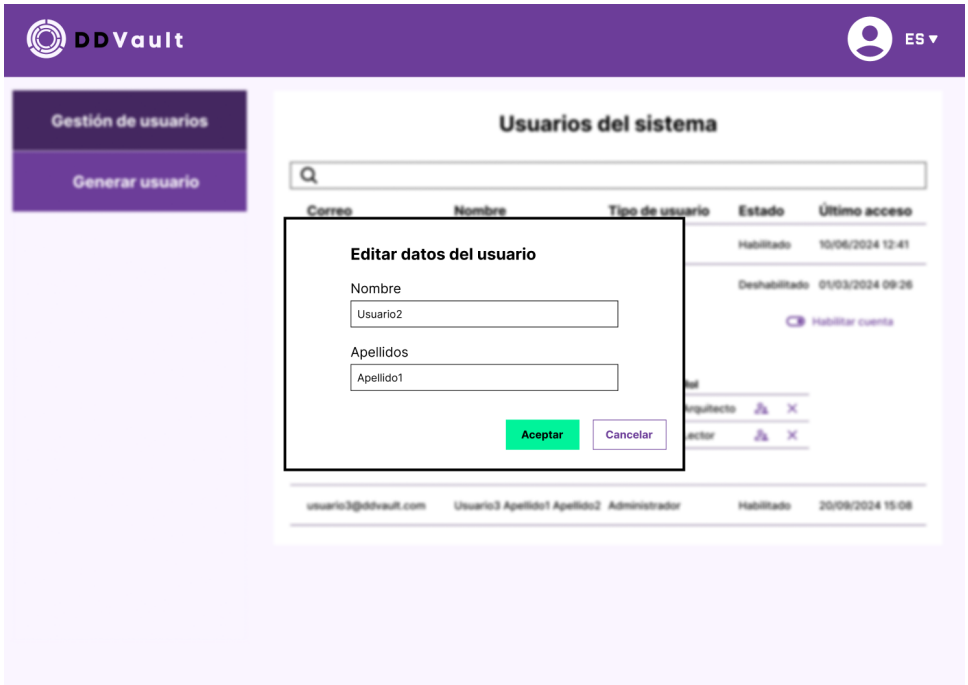


Figura 5.12: Ventana de diálogo para la edición de información del usuario

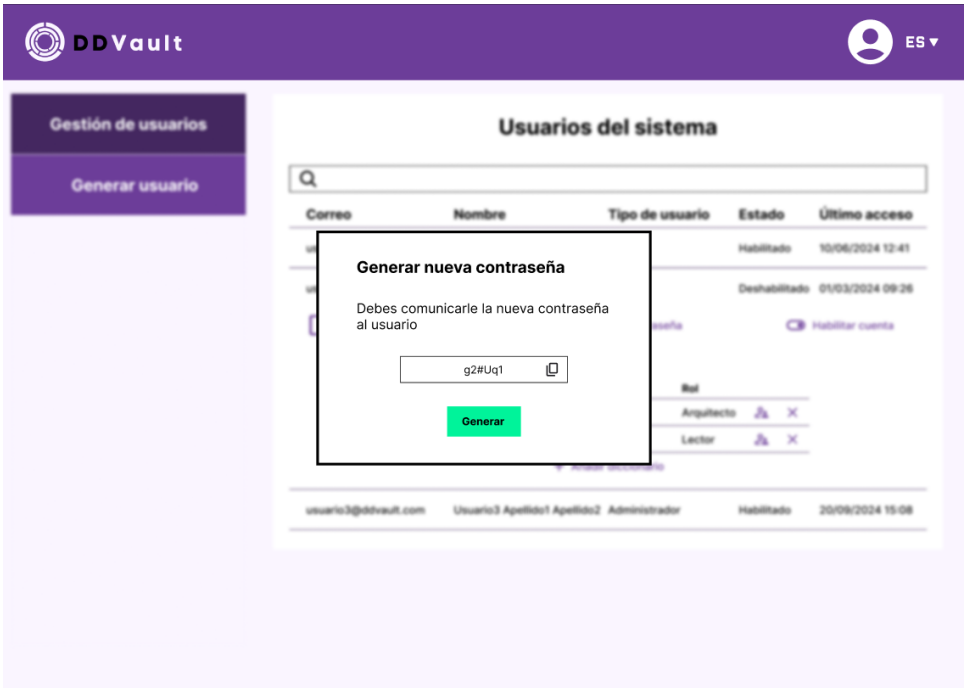


Figura 5.13: Ventana de diálogo para la generación de contraseñas

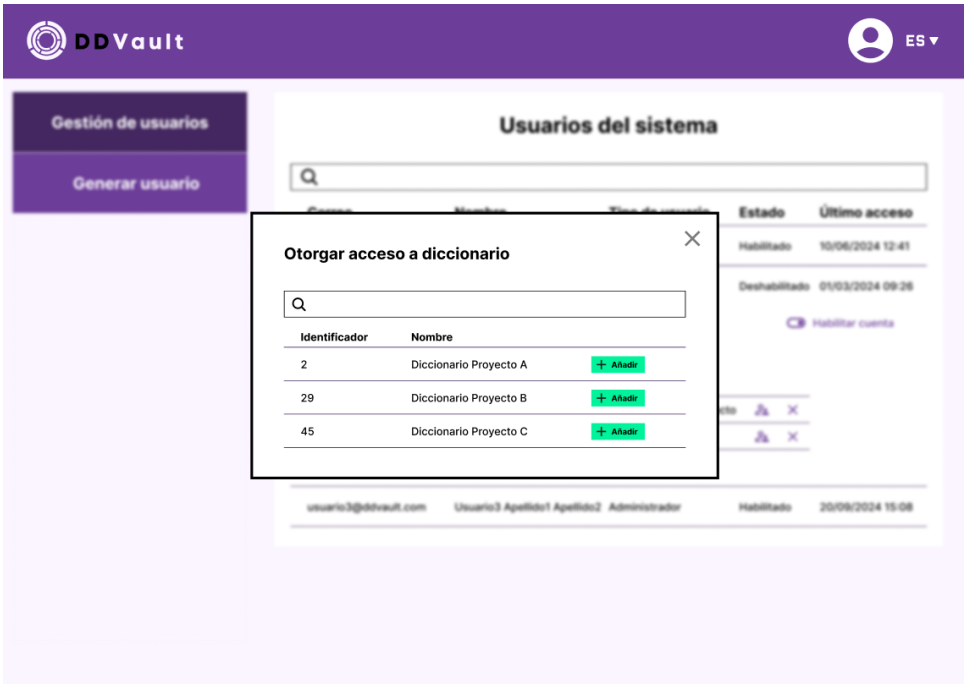


Figura 5.14: Ventana de diálogo para conceder acceso a diccionarios

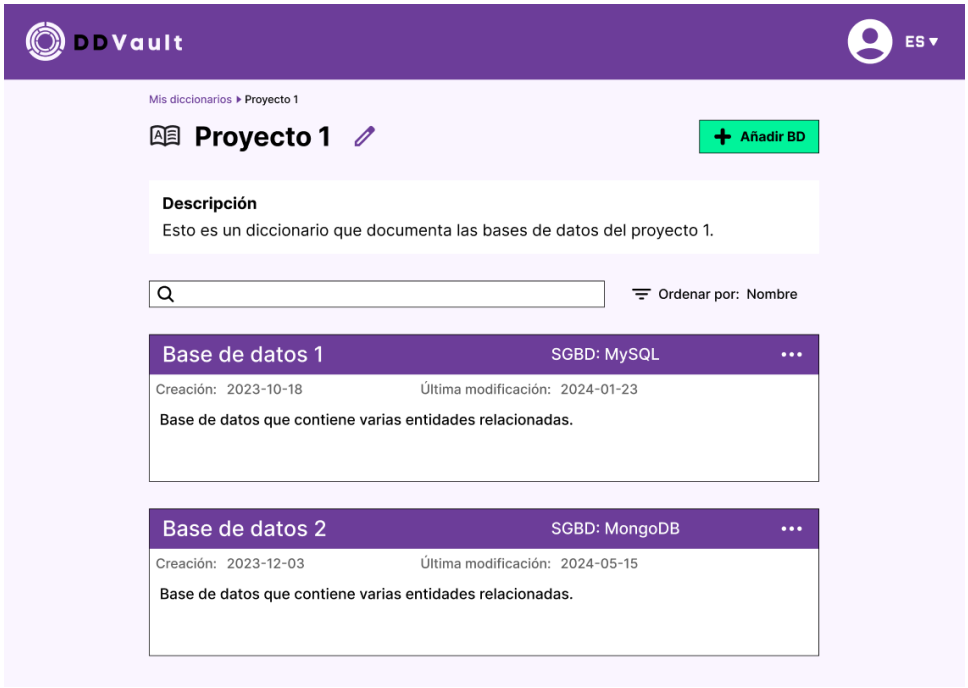


Figura 5.15: Pantalla para la visualización de un diccionario

Figura 5.16: Pantalla para la visualización de una base de datos

Figura 5.17: Pantalla para la visualización de una entidad

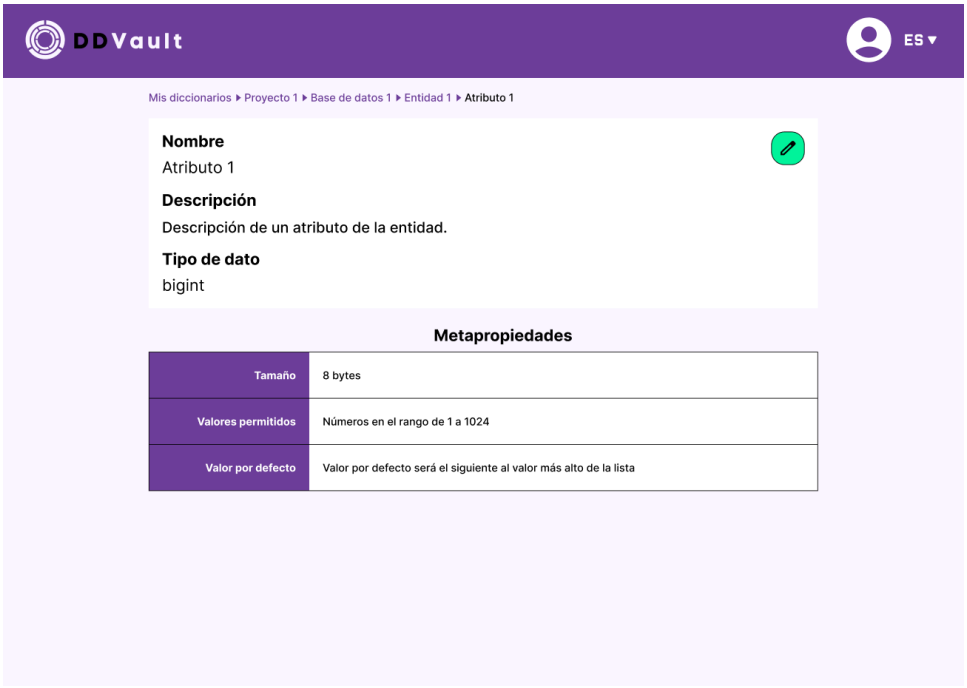


Figura 5.18: Pantalla para la visualización de un atributo

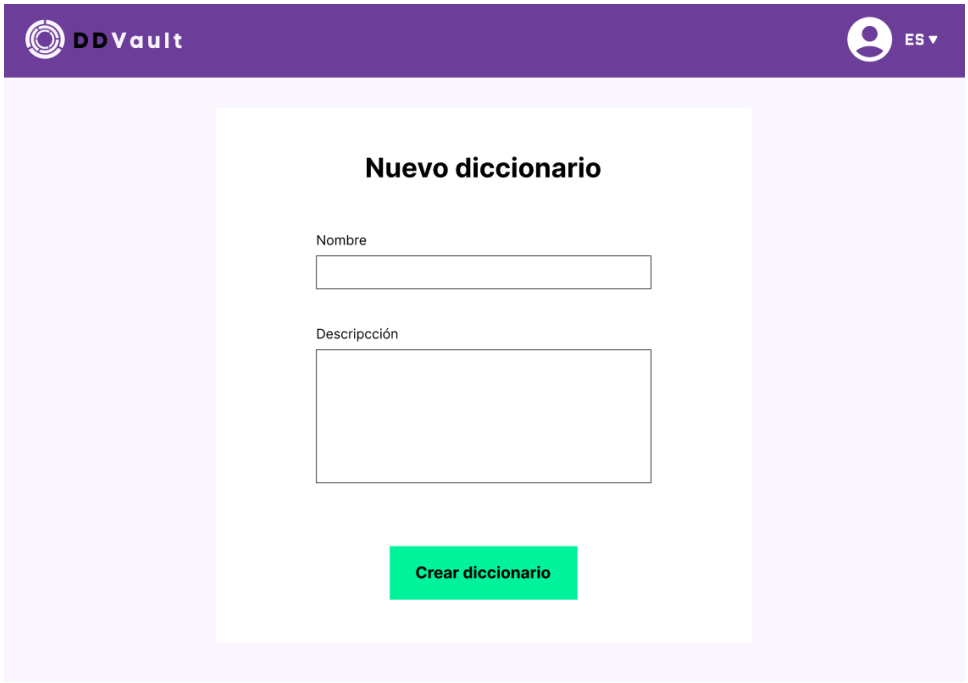


Figura 5.19: Pantalla para la creación de un diccionario

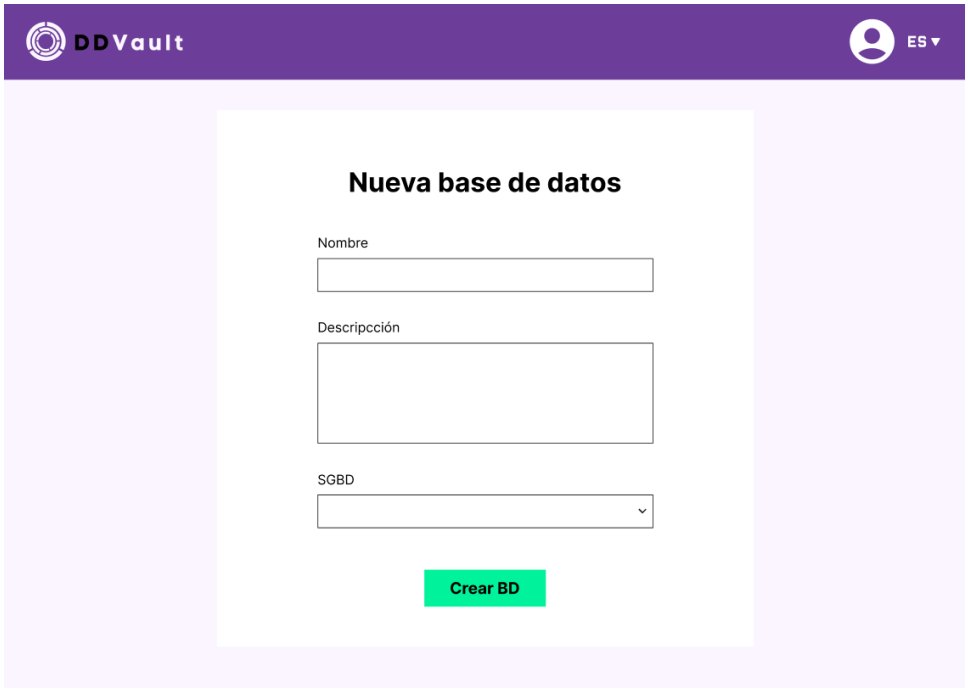


Figura 5.20: Pantalla para la creación de una base de datos

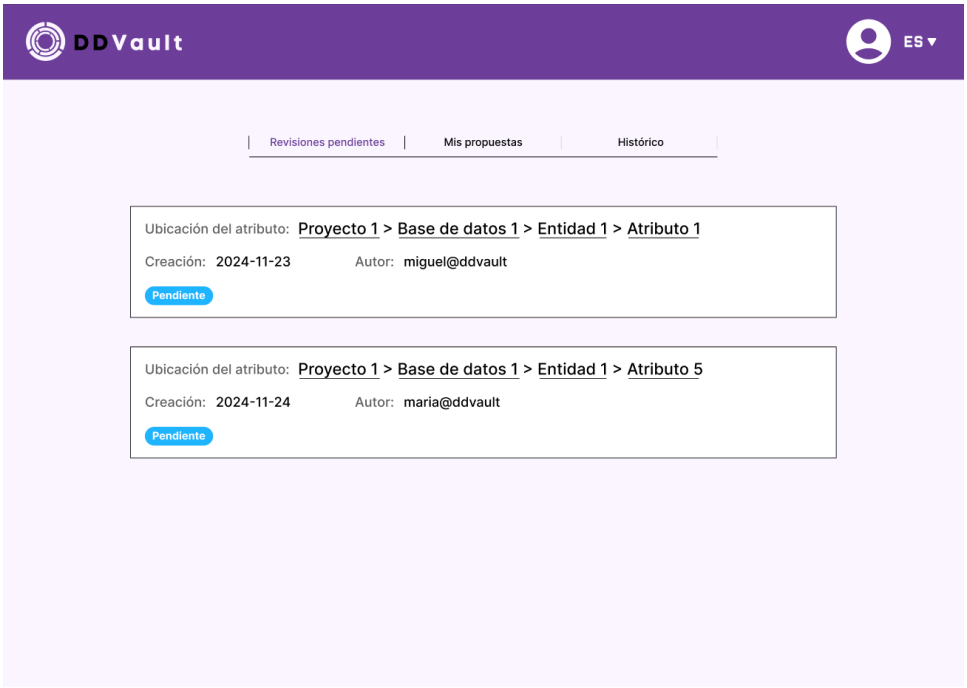


Figura 5.21: Pantalla con la lista de propuestas

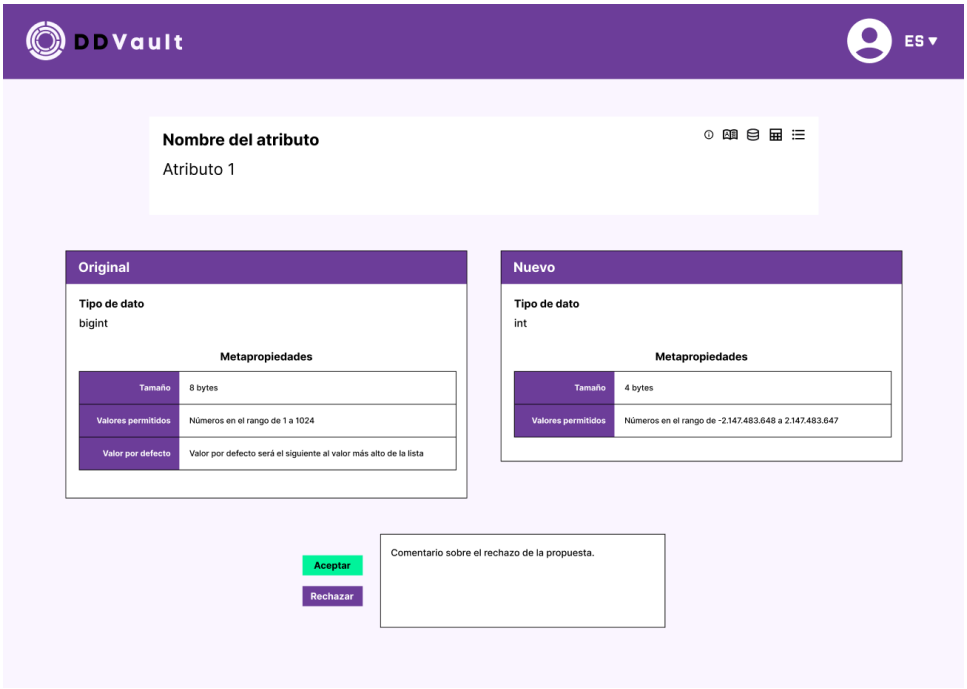


Figura 5.22: Pantalla con la revisión de una propuesta

5.3. Diseño de datos

En esta sección se presenta el diseño lógico de los datos a través de un diagrama UML adaptado con estereotipos que representa el diseño relacional de la base de datos del sistema, sus tablas, con sus campos, y las relaciones entre ellas. En la Figura 5.23 se muestra dicho diagrama, el cual se ha ido refinando a medida que avanzaba el desarrollo de las historias de usuario, adaptándose así a los requisitos y cambios del sistema.

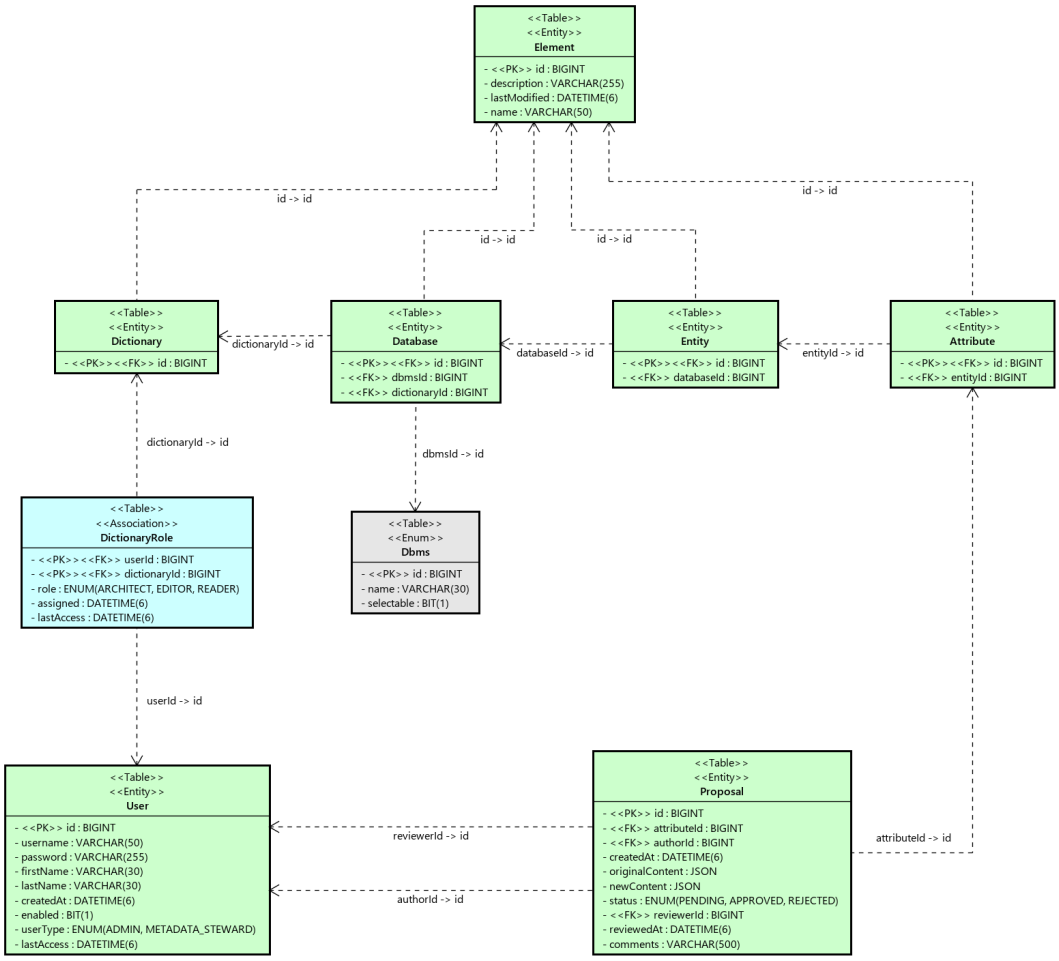


Figura 5.23: Diagrama entidad-relación

5.4. Diseño de la comunicación

Con el objetivo de mostrar un ejemplo de la comunicación entre objetos en el sistema, se ha escogido la historia de usuario "**HU10 - Crear diccionario de datos**". Esta se ha dividido en dos diagramas de secuencia que representan las interacciones de los objetos en el cliente (Figura 5.24) y en el servidor (Figura 5.25).

La comunicación se inicia cuando el usuario completa el formulario y solicita la creación de un nuevo diccionario. El componente de Angular capta el evento "onSubmit" y delega la acción en el servicio correspondiente. En el servicio se forma la solicitud HTTP a la API REST del servidor, la cual se realiza de forma asíncrona.

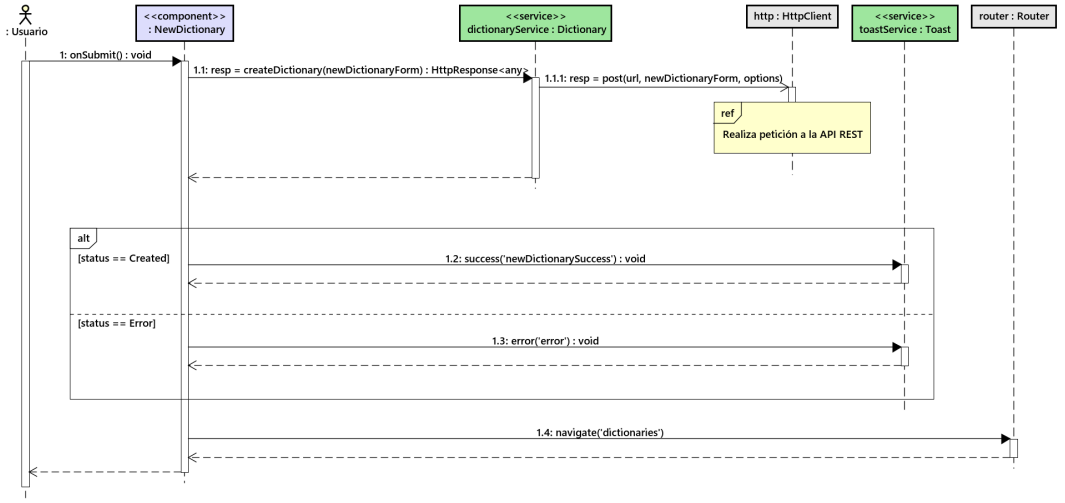


Figura 5.24: Diagrama de secuencia de "HU10 - Crear diccionario de datos" en el front-end

En el servidor, la petición es recibida por el controlador de diccionarios, que se encarga de redirigirla al servicio correspondiente. El servicio extrae la información necesaria de la solicitud y se encarga de la creación de las entidades. Dado que un usuario puede estar asociado a múltiples diccionarios y cada diccionario puede tener varios usuarios, además de crear una entidad "Dictionary", también se crea una entidad denominada "DictionaryRoleEntity". Esta entidad representa la asociación entre un usuario y un diccionario, y almacena el rol que desempeña dicho usuario en el mismo.

Si el proceso se completa correctamente, y no se han producido errores de comunicación, el servidor genera una respuesta HTTP con el estado CREATED que se envía de vuelta al cliente, confirmando la creación del nuevo diccionario.

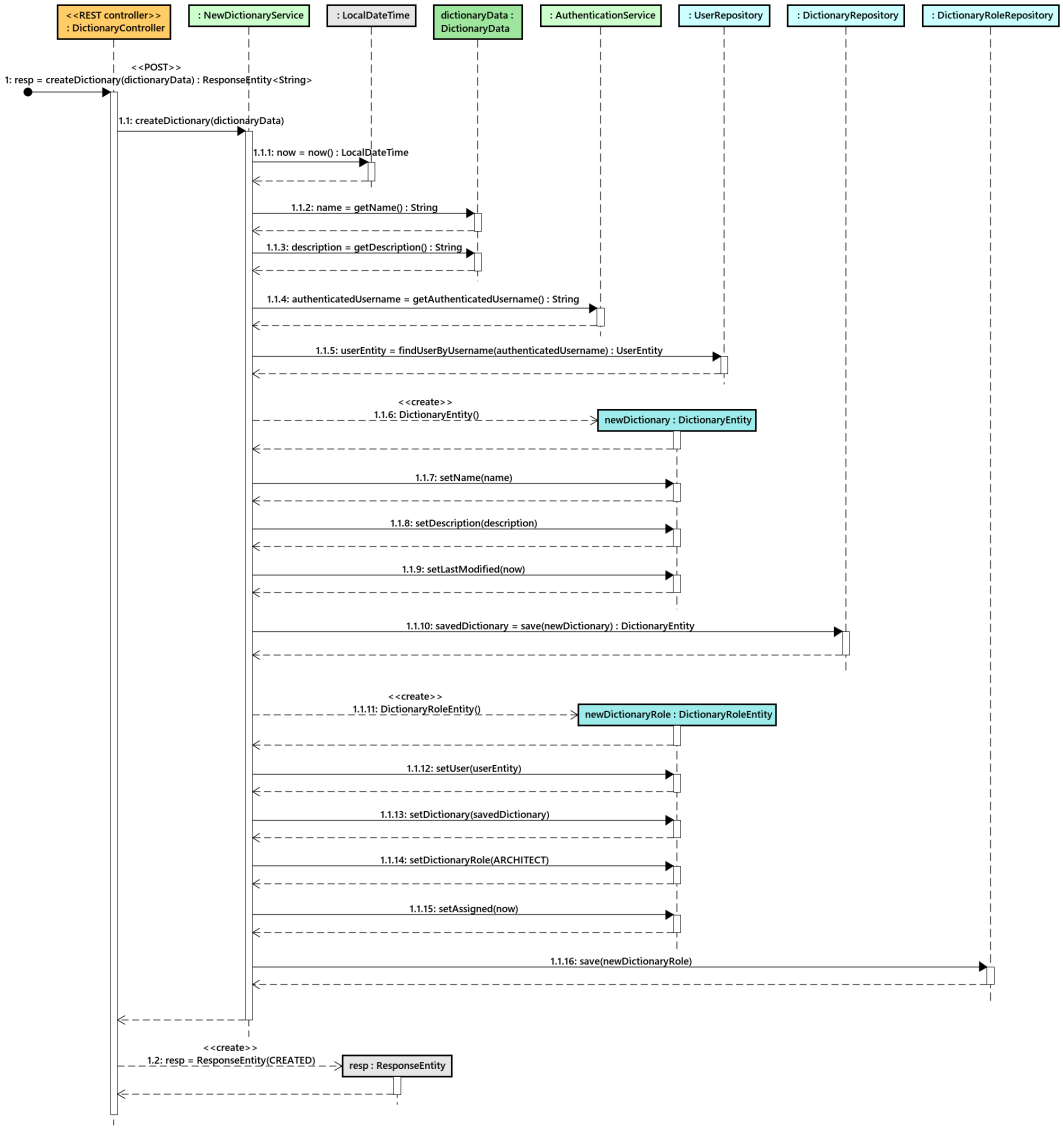


Figura 5.25: Diagrama de secuencia de “HU10 - Crear diccionario de datos” en el back-end

5.5. Despliegue de la aplicación

A continuación, se describe el despliegue de la aplicación tanto en el entorno de pruebas como en el entorno de producción.

En ambos entornos, la comunicación entre el navegador, el front-end y el back-end se realiza mediante peticiones HTTP. El front-end interactúa con el back-end a través de una API REST, mientras que el back-end gestiona el acceso a la base de datos mediante JPA.

En el entorno de desarrollo y pruebas, todo el sistema se ejecuta en un dispositivo con sistema operativo Windows. Este entorno es ideal para la fase de desarrollo, permitiendo realizar pruebas rápidas y ajustes en la aplicación. En este caso, el front-end se ejecuta en un servidor Node.js. La Figura 5.26 presenta el diagrama de despliegue en el entorno de pruebas local utilizado durante el desarrollo de la aplicación.

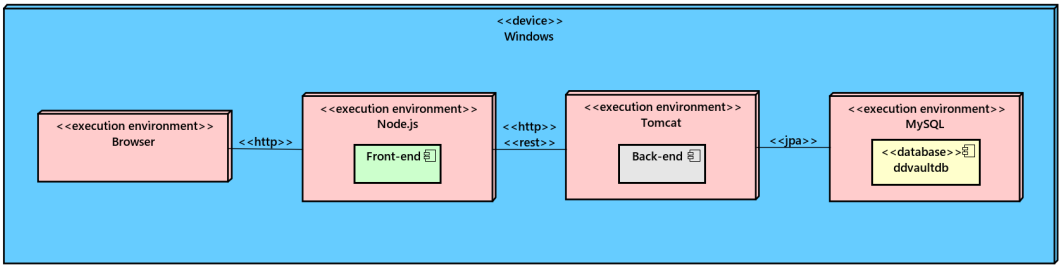


Figura 5.26: Diagrama de despliegue en el entorno local de desarrollo y pruebas

En el entorno de producción, uno o varios usuarios acceden a la aplicación desde dispositivos conectados a la red, utilizando un navegador. Para el despliegue de los servicios, se emplea una máquina virtual proporcionada por la universidad, que utiliza Linux. En este entorno, el front-end se ejecuta en un servidor Nginx. El diagrama de la Figura 5.27 muestra el despliegue realizado en el entorno de producción.

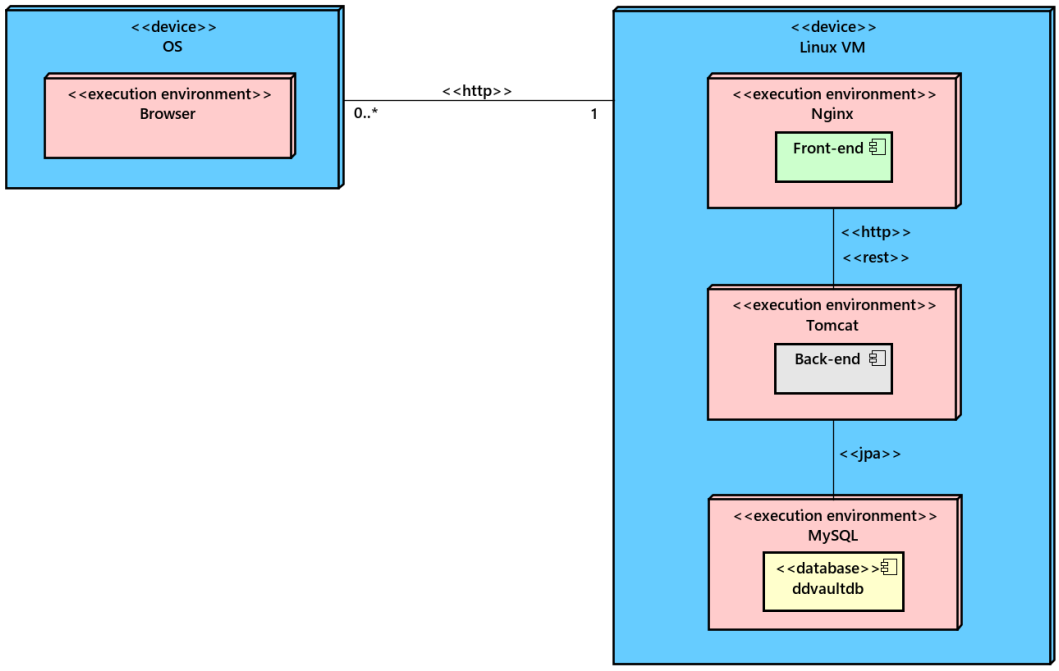


Figura 5.27: Diagrama de despliegue en el entorno de producción

Capítulo 6

Implementación y pruebas

6.1. Licencia

Este proyecto se declara bajo una licencia **CC BY 4.0**. La Figura 6.2 muestra algunas de las características y términos clave de la licencia. Para más información se debe consultar `https://creativecommons.org/licenses/by/4.0/`

Figura 6.1: CC BY 4.0

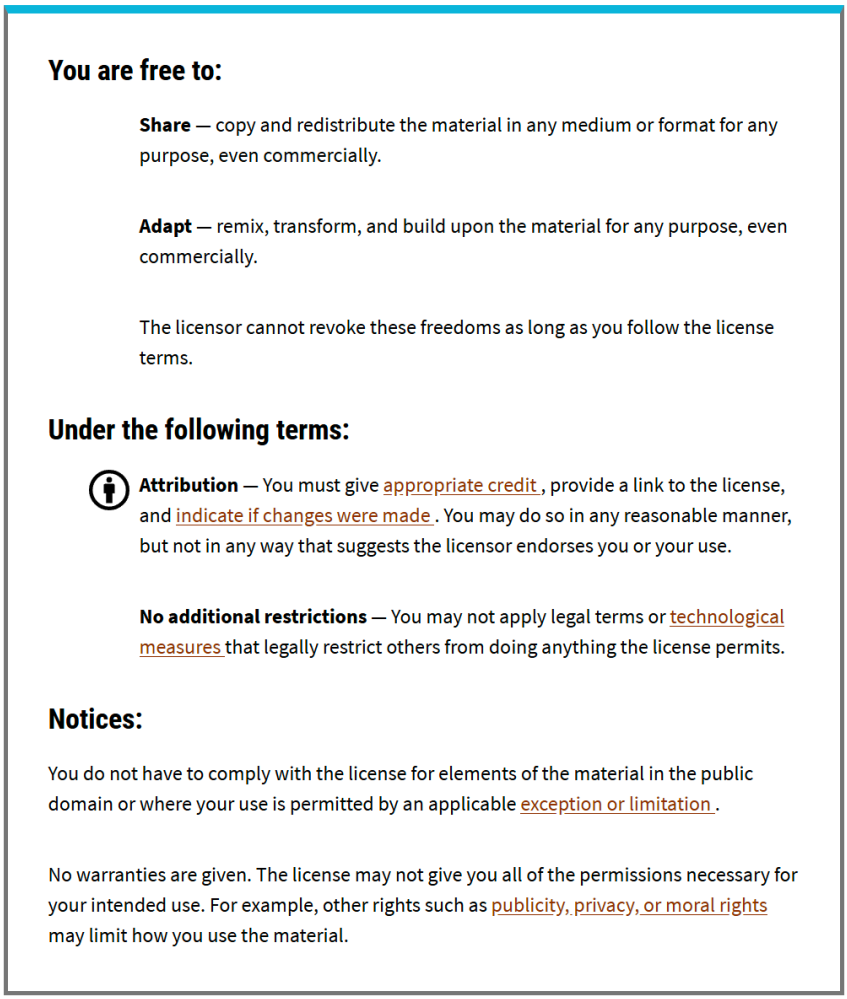
6.2. Implementación

6.2.1. Organización del proyecto

A continuación, se describe la organización del proyecto por directorios, detallando el tipo de contenido de las carpetas y el papel que desempeñan algunos ficheros. Se han omitido principalmente ficheros específicos del framework o del editor de código.

Estructura del repositorio

El proyecto se encuentra en un único repositorio, que alberga tanto el front-end como el back-end. La estructura del repositorio se puede apreciar en la Figura 6.3. Dentro de la carpeta *app* se encuentran las aplicaciones desarrolladas, organizadas en subcarpetas según el framework utilizado, *Angular* y *Spring*.



You are free to:

Share — copy and redistribute the material in any medium or format for any purpose, even commercially.

Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation .

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

Figura 6.2: Resumen de la licencia CC BY 4.0. Captura tomada de [10]

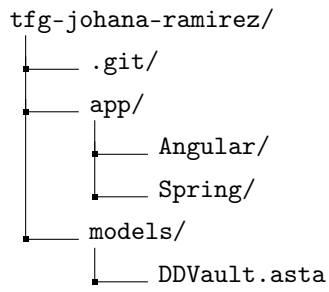


Figura 6.3: Estructura del repositorio

Por otra parte, la carpeta *models* contiene los ficheros relacionados con la documentación. En este caso, el fichero astah donde se encuentran todos los diagramas realizados durante las fases de análisis y diseño.

Estructura del código en Spring Boot

La estructura inicial del proyecto realizado con Spring Boot se genera al crear la aplicación. El código fuente se encuentra en la carpeta *src*. Dentro de *src/main/resources* se encuentra el fichero *application.properties*, que se utiliza para configurar aspectos como la conexión a la base de datos y otros parámetros.

En *src/main/java*, dentro del paquete *uva.inf.tfg.ddvault*, se encuentra el fichero *DDVaultApplication.java*, el cual se encarga de iniciar el contexto de la aplicación. El fichero *ApplicationConfig.java* establece los componentes para la autenticación de usuarios mediante Spring Security, como el repositorio de usuario y el cifrado de contraseñas. El resto de carpetas son las creadas de acuerdo con la arquitectura definida en la Sección 5.1.2. Estas carpetas son:

- ***business***: En esta carpeta se encuentran los *boundaries*, estos ayudan a formatear los datos que se envían y reciben, por eso se distinguen dos tipos, los de entrada y los de salida. No se consideran DTOs ya que pueden estar formados por campos de varias entidades o contener solo información parcial de la entidad. Los *mappers* ayudan a formar boundaries de salida, tomando los datos necesarios de entidades u otros objetos. En *exceptions* se encuentran las excepciones personalizadas, que proporcionan mensajes descriptivos para identificar errores durante la ejecución. Se lanzan en los servicios, pero se capturan en los controladores, lo que permite generar respuestas HTTP detalladas en caso de error. En *services* están los servicios donde se procesan las peticiones. Normalmente cada servicio creado se corresponde con una historia de usuario, sin embargo, en ocasiones es necesario crear más de uno. En la carpeta *core* se encuentran los servicios más importantes, lo cuales son utilizados repetidamente por otros servicios.
- ***commonservices***: En esta carpeta se incluyen configuraciones de seguridad, enumerados que se utilizan en varios puntos de la aplicación y otros clases que no encajan en

ninguna de las otras capas pero son utilizadas por estas.

- ***controllers***: En los controladores se definen los endpoints de la aplicación. Se han ordenado, principalmente, por el nombre de la entidad a la que afecta la operación
- ***persistence***: agrupa los ficheros relacionados con la gestión de los datos persistentes, aquí se encuentran repositorios y entidades.

El proyecto utiliza Maven, lo cual se refleja en la existencia del archivo *pom.xml*. Este archivo se usa para gestionar las dependencias y configurar el proceso de compilación y ejecución de la aplicación.

En la Figura 6.4 puede verse la estructura de la aplicación desarrollada para el back-end.

Estructura del código en Angular

Al igual que Spring Boot, Angular define una estructura inicial al crear el proyecto, la cual se ha respetado y se le han añadido otras carpetas para organizar mejor el proyecto. En la Figura 6.5 se muestra la estructura final del proyecto.

Los ficheros *angular.json* y *package.json* se utilizan en la configuración de la aplicación. El primero gestiona la configuración global del proyecto, mientras que el segundo se encarga de gestionar las dependencias mediante npm.

Dentro de *src* se encuentran los siguientes directorios:

- ***app***: contiene el código principal de la aplicación, la estructura de este directorio coincide con la arquitectura del cliente descrita en la Sección 5.1.3. Aquí se encuentra el componente raíz, formado por los tres ficheros característicos de un componente Angular (ts, html y css), además de su fichero de pruebas. En *app-routing.module.ts* se asocian URLs con componentes y guardas. En el fichero *app.module.ts* se declaran los componentes creados y se importan los módulos necesarios.
  - ***components***: Contiene el resto de componentes de la aplicación, cada uno organizado en su propia carpeta.
  - ***services***: Agrupa todos los servicios. Cada servicio dispone de su propia carpeta, que incluye también su fichero de pruebas.
  - ***models***: En esta carpeta se encuentran los ficheros que definen la estructura de los datos. Al no tener funcionalidad, se utilizan interfaces en lugar de clases. También se incluyen aquí los enums.
  - ***guards***: Las guardas creadas se utilizan para comprobar que el usuario haya iniciado sesión y que tenga el rol adecuado al intentar acceder a determinadas páginas.
  - ***interceptors***: Contiene un único interceptor que se encarga de insertar en la cabecera de las peticiones HTTP el token de autenticación generado al iniciar sesión.

- ***assets***: En la carpeta *i18n* se encuentran los ficheros de internacionalización, cada idioma debe tener un fichero JSON (*en.json* para inglés y *es.json* para español). En *images* están los logotipos diseñados para la aplicación.
- ***environments***: Aquí se encuentran los ficheros que configuran el cambio automático de la URL de la API según si el entorno es de desarrollo o de producción.

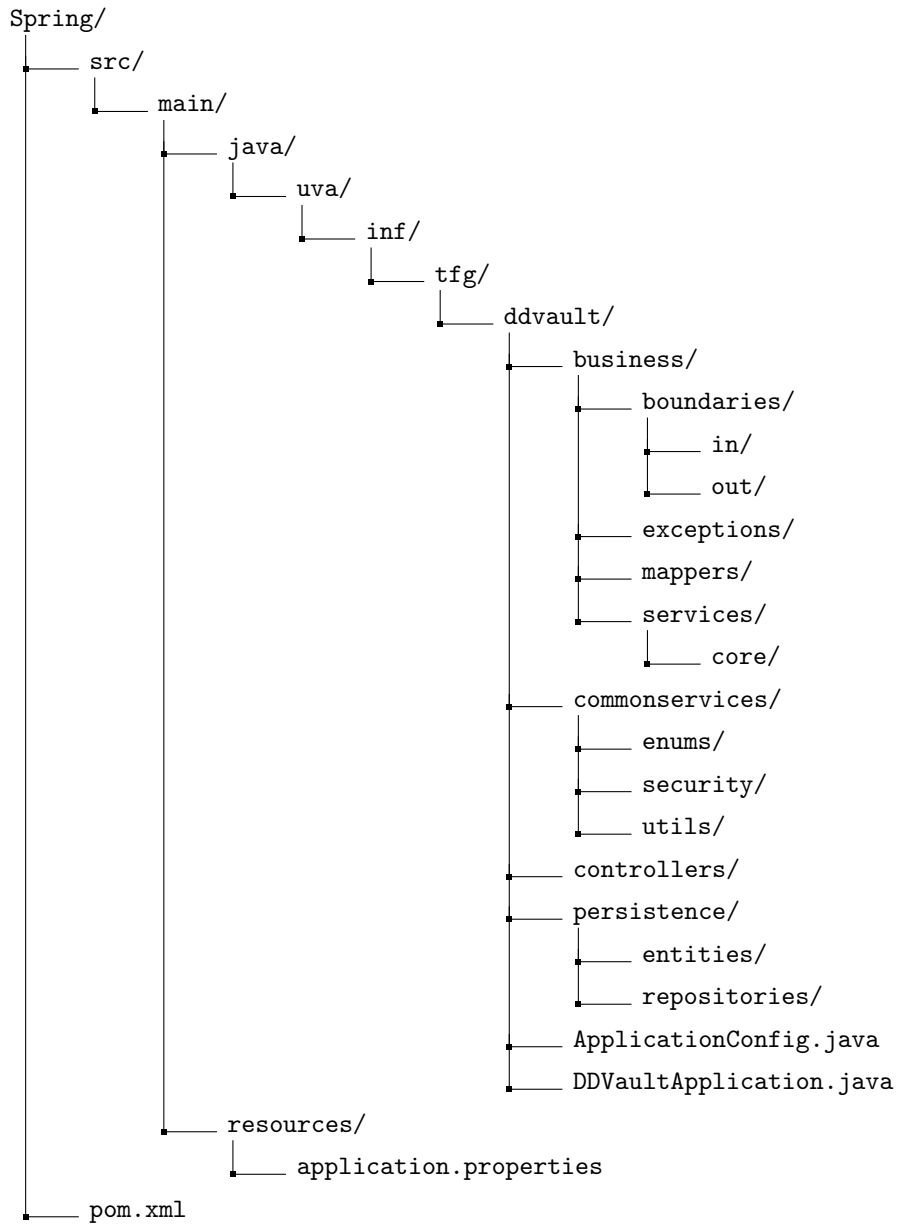


Figura 6.4: Estructura del código en Spring Boot

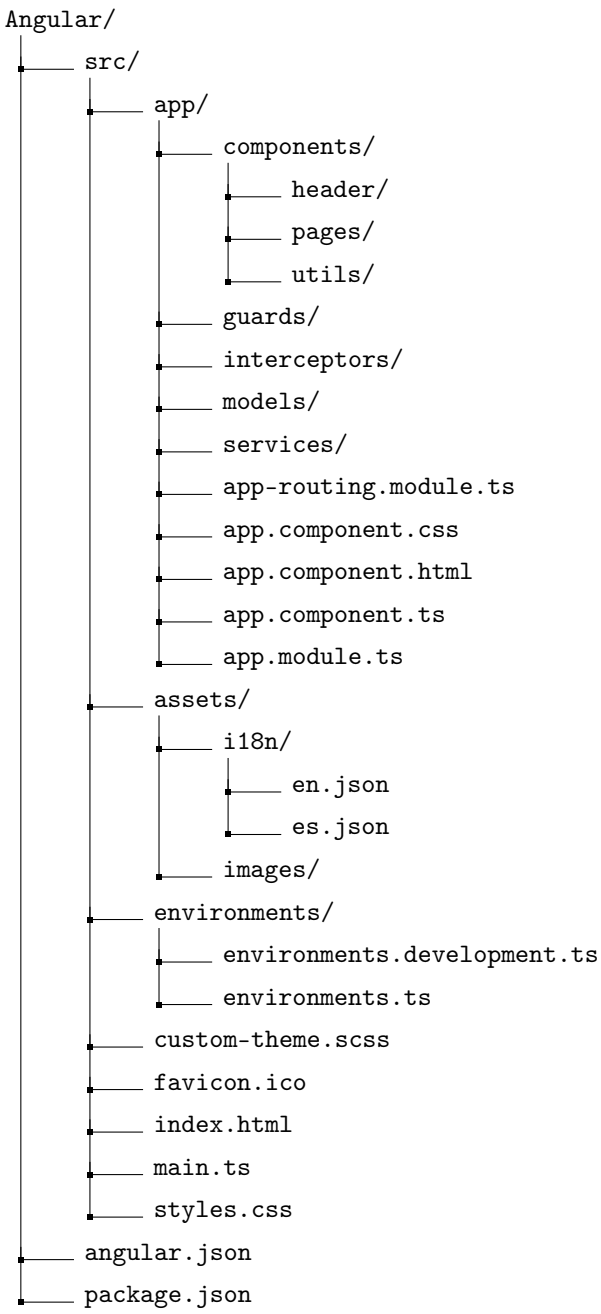


Figura 6.5: Estructura del código en Angular

6.2.2. Dificultades encontradas

Durante el desarrollo de la aplicación surgieron algunas dificultades al implementar determinadas funcionalidades. Algunas de ellas tuvieron que ser replanteadas para simplificarlas y adaptarlas a las limitaciones de tiempo. A continuación, se describen algunos de estos retos y las soluciones adoptadas:

- **Bloqueo de atributos durante su edición**: En un principio, se planeó impedir que varios usuarios editaran simultáneamente un mismo atributo. La idea era mantener un estado en el servidor que indicara si un atributo estaba siendo editado y bloquear su edición. Sin embargo, no es posible detectar de manera sencilla si un usuario ha abandonado la página, por ejemplo, porque ha cerrado el navegador. Una posible solución consistía en introducir temporizadores que desbloquearan el atributo automáticamente pasado un cierto tiempo. Sin embargo, esta alternativa también requería contemplar varios casos adicionales. Por ejemplo, si el editor pasa demasiado tiempo editando y el tiempo está a punto de acabarse, habría que ampliar el temporizador. O si el editor sale por accidente de la página debería ser capaz de volver y retomar su trabajo desde donde lo dejó. Debido a la introducción del sistema de revisiones, se decidió optar por la solución más básica, que es: los atributos solo se bloquean en el momento en que un editor envía una propuesta de cambio. Esto también tiene inconvenientes, como que dos editores pueden trabajar simultáneamente sobre el mismo atributo, pero solo los cambios del primero que finalice serán los que se guarden.

- **Sistema de mensajería**: Para la gestión de contraseñas lo ideal habría sido tener un sistema de mensajería automático que enviase al correo electrónico del usuario la contraseña generada al crear la cuenta. Esto le permitiría acceder al sistema y cambiarla por una más segura. También sería muy útil para el restablecimiento de contraseñas. Esto tampoco se ha implementado debido a la limitación del tiempo y a que no se consideraba algo primordial. En su lugar, es un administrador del sistema quien se encarga de restablecer las contraseñas y comunicarlas manualmente al usuario.

- **Pruebas con Jasmine**: Dado que era la primera vez que se trabajaba con Jasmine, surgieron diversos problemas causados por la falta de conocimiento. Durante las primeras etapas del desarrollo, las pruebas se ejecutaban correctamente, lo que llevó a pensar que estaban bien definidas. Sin embargo, a medida que aumentaba la complejidad de los elementos a probar, comenzaron a surgir errores. Estos errores no eran consistentes, no siempre fallaban las mismas pruebas y para solucionarlos se aplicaban "parches" que ocultaban el problema real en lugar de resolverlo. La situación empeoró hasta el punto de que no era posible conseguir que todas las pruebas se ejecutaran correctamente. La aleatoriedad en los fallos sugería una configuración incorrecta, especialmente en lo relativo a la inicialización del entorno y la configuración de los mocks. Finalmente, fue necesario rehacer por completo todas las pruebas existentes, prestando especial atención a la correcta preparación del entorno antes de cada prueba. Además de esto, los mensajes de error proporcionados por la consola en caso de fallo no siempre resultan claros o informativos, lo que ha dificultado en gran medida la identificación y resolución de los problemas.

6.3. Pruebas

Para garantizar que una aplicación funcione correctamente y cumpla con los requisitos definidos es necesario realizar pruebas. Estas pueden ser manuales o automatizadas, en función del enfoque que se desee adoptar [35].

En las **pruebas manuales**, una persona introduce datos en el sistema y analiza la respuesta obtenida para comprobar que el comportamiento es el correcto. Estas pruebas se basan en **casos de prueba**, los cuales contemplan distintos escenarios, como situaciones de uso comunes, casos límite o condiciones de error. Cada caso de prueba especifica qué se debe probar, los datos de entrada y el resultado esperado. Las pruebas manuales requieren tiempo y esfuerzo en cada ejecución y están sujetas a errores humanos.

Por otro lado, las **pruebas automatizadas** se basan en **scripts** que simulan de forma automática las acciones que realizaría un usuario. Una vez definidos, estos scripts pueden ejecutarse repetidamente sin intervención manual, comparando los resultados obtenidos con los esperados y notificando cualquier discrepancia. Aunque su creación puede ser costosa, resultan mucho más eficientes a largo plazo. Además, se pueden integrar fácilmente con herramientas de CI/CD, lo que permite una validación constante del sistema conforme se introducen cambios.

En este proyecto se han implementado **pruebas automatizadas** utilizando el framework **Jasmine**, que viene integrado por defecto en los proyectos de Angular.

Como ya se mencionó en el capítulo dedicado a las tecnologías, Jasmine es un framework diseñado para realizar pruebas basadas en **Behavior-Driven Development** (**BDD**). Este enfoque busca describir el comportamiento esperado del sistema desde la perspectiva del usuario. Una práctica habitual en BDD es escribir las pruebas antes de implementar la funcionalidad, lo que permite comprobar desde el inicio que el desarrollo se alinea con lo que el usuario quiere.

No obstante, Jasmine también es perfectamente válido para realizar pruebas unitarias o aplicar enfoques como el Test-Driven Development [41]. Las **pruebas unitarias** se centran en verificar pequeñas unidades del código, como funciones, asegurando que funcionen de manera correcta y **aislada**. Este tipo de pruebas es especialmente útil para garantizar la mantenibilidad del sistema, ya que permiten detectar rápidamente si una modificación rompe alguna funcionalidad existente.

Las pruebas realizadas para este proyecto combinan ambos enfoques. Por un lado, se han definido tests que validan el comportamiento esperado desde la perspectiva del usuario. Por otro, se han implementado pruebas centradas en comprobar que métodos o servicios individuales funcionan correctamente de forma aislada. La aplicación de ambos tipos de pruebas ha permitido alcanzar una buena cobertura de código. También hay que señalar que las pruebas se escribieron una vez se había implementado la funcionalidad, ya que aún no se contaba con la experiencia suficiente de la tecnología utilizada.

Las pruebas se definen en los ficheros ***.spec.ts***. Estos ficheros suelen generarse automáticamente junto al elemento correspondiente cuando se utiliza el comando *ng generate*. Por

lo tanto, cada componente, servicio, guarda e interceptor cuenta con su propio fichero de pruebas. De esta forma, los tests de un elemento son fácilmente localizables y se mantienen organizados junto al código que validan.

6.3.1. Sintaxis de Jasmine

A continuación, se va a describir la sintaxis de Jasmine con la ayuda del ejemplo de la Figura 6.6. Este ejemplo no forma parte de las pruebas realizadas para validar el proyecto pero permitirá identificar los principales elementos utilizados en la escritura de este tipo de pruebas [25].

```
describe('SaludoComponent', () => {
    let saludoComponent: SaludoComponent;

    beforeEach(() => {
        const mockUsuarioService = jasmine.createSpyObj('UsuarioService', ['getNombre']);

        mockUsuarioService.getNombre.and.returnValue('Johana');

        // Otros ajustes relacionados con la inicialización del componente
    });

    it('debería saludar al usuario que ha iniciado sesión', () => {
        saludoComponent.saludar();

        expect(saludoComponent.mensaje).toBe('Hola, Johana');
    });

});
```

Figura 6.6: Ejemplo de una *test suite* en Jasmine

- **Suites** (***describe***): La función *describe* se utiliza para agrupar un conjunto de pruebas relacionadas. Cada fichero de pruebas comienza con una suite. El primer argumento sirve para describir el grupo de pruebas. En el ejemplo, la suite agrupa el conjunto de pruebas del componente *SaludoComponent*.
- **Specs** (***it***): Las pruebas se definen con la función *it*. Habrá tantas como casos de prueba se deseen cubrir y dentro de cada una se definen las expectativas. También recibe una cadena descriptiva como argumento. En el ejemplo, el caso de prueba consiste en comprobar que el componente construya correctamente un saludo personalizado con la ayuda del servicio de usuarios, al cual se le solicita el nombre del usuario registrado.
- **Expectations** (***expect***): Las expectativas son las condiciones que se deben cumplir para que una prueba pase. Se construyen con la función *expect*, que recibe el valor real como argumento y se utiliza junto con un matcher que define la condición esperada.
- **Matchers**: Son métodos encadenados a *expect* que comparan el valor recibido con el valor esperado. Realizan comparaciones booleanas, como por ejemplo, *toBe*, *toContain*,

toBeLessThan, entre otros. En el ejemplo, se espera que el saludo generado, almacenado en la variable *mensaje* del componente, sea igual al valor esperado "Hola, Johana".

También existen otras funciones que permiten configurar las pruebas antes y después de las pruebas. Estas son las funciones: ***beforeEach***, ***afterEach***, ***beforeAll***, ***afterAll***. De esta forma se evita repetir código al preparar o limpiar el entorno de pruebas. En el ejemplo, se ha utilizado la función *beforeEach* para configurar el **mock** del servicio que utiliza el componente y simular una respuesta.

En la Figura 6.7, se muestra una prueba real del proyecto. En este caso, se trata de un test unitario del componente *LoginComponent*, donde se verifica que, al enviar el formulario mediante el método *onSubmit*, el sistema realiza correctamente las operaciones necesarias si el usuario es un administrador. Entre estas operaciones se comprueba que se almacenen el token y la sesión recibidos, que se redirija al usuario a la ruta correspondiente y que se muestre un mensaje de éxito.

```
it('should navigate to admin route when user is admin', () => {
    mockLoginService.login.and.returnValue(of({status: 200, body: token}));
    mockLoginService.getUserSession.and.returnValue(of({status: 200, body: adminSession}));
    mockLoginService.isAdmin.and.returnValue(true);

    loginComponent.onSubmit()

    expect(mockLoginService.login).toHaveBeenCalledTimes(1)
    expect(mockLoginService.setToken).toHaveBeenCalledWith(token.token)
    expect(mockLoginService.getUserSession).toHaveBeenCalledTimes(1)
    expect(mockLoginService.setSession).toHaveBeenCalledWith(adminSession)
    expect(mockRouter.navigate).toHaveBeenCalledWith(['admin'])
    expect(mockToastService.success).toHaveBeenCalledTimes(1)
    expect(mockToastService.success).toHaveBeenCalledWith('loginSuccess')
});
```

Figura 6.7: Prueba perteneciente a la suite del componente *LoginComponent*

6.3.2. Cobertura de las pruebas

Angular permite medir la cobertura de las pruebas automatizadas, es decir, qué porcentaje del código ha sido ejecutado al realizar los tests.

Para ejecutar las pruebas se utiliza **Karma**, que ejecuta las pruebas en un navegador y proporciona un resumen de los resultados, incluyendo el número total de pruebas superadas y fallidas. En la Figura 6.8, se muestra el resumen generado para este proyecto, donde se indica que se han ejecutado correctamente **212 pruebas** y no ha habido ningún fallo.

La cobertura se obtiene ejecutando el comando `ng test --code-coverage`. Tras la ejecución, se genera la carpeta *coverage* en la raíz del proyecto, donde se encuentra el informe generado. En la Figura 6.9, se puede observar que la cobertura es prácticamente del **100 %**. El único componente con una línea sin cubrir es *ElementCardComponent*, debido al uso de

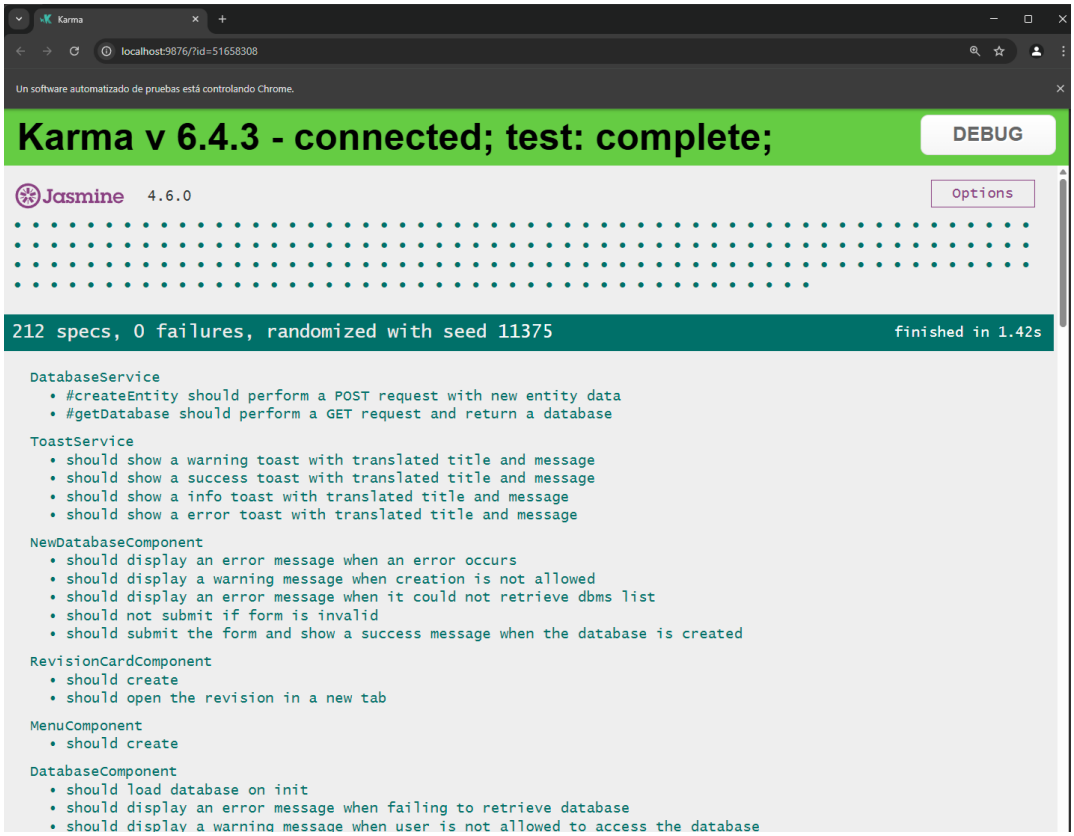
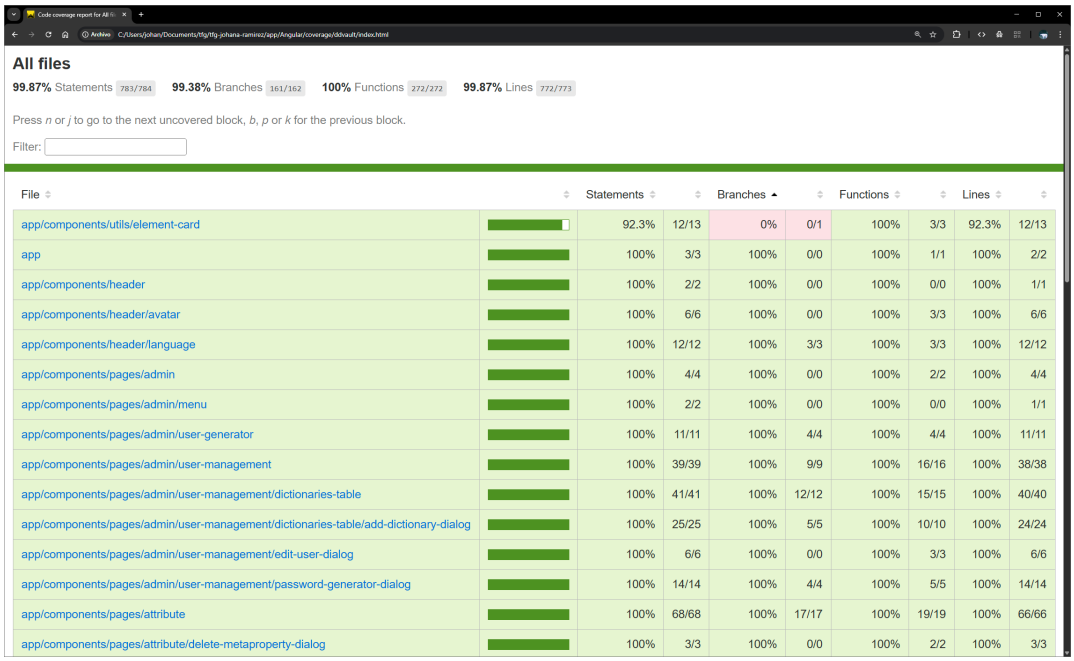


Figura 6.8: Karma mostrando los resultados de las pruebas ejecutadas

window.location.reload(), que provoca una recarga completa del componente y rompe el flujo de ejecución de la prueba.



All files

99.87% Statements 783/784 **99.38%** Branches 161/162 **100%** Functions 272/272 **99.87%** Lines 772/773

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
app/components/utils/element-card	92.3%	12/13	0%	0/1	100%	3/3	92.3%	12/13
app	100%	3/3	100%	0/0	100%	1/1	100%	2/2
app/components/header	100%	2/2	100%	0/0	100%	0/0	100%	1/1
app/components/header/avatar	100%	6/6	100%	0/0	100%	3/3	100%	6/6
app/components/header/language	100%	12/12	100%	3/3	100%	3/3	100%	12/12
app/components/pages/admin	100%	4/4	100%	0/0	100%	2/2	100%	4/4
app/components/pages/admin/menu	100%	2/2	100%	0/0	100%	0/0	100%	1/1
app/components/pages/admin/user-generator	100%	11/11	100%	4/4	100%	4/4	100%	11/11
app/components/pages/admin/user-management	100%	39/39	100%	9/9	100%	16/16	100%	38/38
app/components/pages/admin/user-management/dictionaries-table	100%	41/41	100%	12/12	100%	15/15	100%	40/40
app/components/pages/admin/user-management/dictionaries-table/add-dictionary-dialog	100%	25/25	100%	5/5	100%	10/10	100%	24/24
app/components/pages/admin/user-management/edit-user-dialog	100%	6/6	100%	0/0	100%	3/3	100%	6/6
app/components/pages/admin/user-management/password-generator-dialog	100%	14/14	100%	4/4	100%	5/5	100%	14/14
app/components/pages/attribute	100%	68/68	100%	17/17	100%	19/19	100%	66/66
app/components/pages/attribute/delete-metaproperty-dialog	100%	3/3	100%	0/0	100%	2/2	100%	3/3

Figura 6.9: Cobertura del código

Capítulo 7

Seguimiento del proyecto

7.1. Introducción

En esta sección se detalla el seguimiento del proyecto, debido a que se ha adoptado el marco de trabajo ágil Scrum, la unidad básica de seguimiento son los sprints. Cada sprint está dividido en tres apartados, en el **sprint planning** se describen las tareas planificadas para ese sprint y su tiempo estimado, en el **sprint review** se señalan las valoraciones y correcciones a realizar respecto al trabajo realizado, y en el **sprint retrospective** se indican los riesgos materializados durante el desarrollo del sprint y las acciones tomadas para corregirlos, también se indican las posibles mejoras a realizar en el próximo sprint.

Todos los sprints están acompañados de una tabla en que se detallan las tareas realizadas en el sprint. En cada tarea se especifica la épica e historia de usuario a la que está asociada, si es que lo está, el tiempo estimado y el tiempo real empleado, una descripción de la tarea y, por último, el estado en el que se encuentra, el cual puede variar entre "No iniciada", "Iniciada", "Incompleta", "Completada". La diferencia entre los estados "Iniciada" e "Incompleta" será la cantidad de trabajo restante para terminar la tarea, significando "Incompleta" que queda poco para completarla. La última fila de la tabla indica el tiempo trabajado en ese sprint. En la Tabla 7.1 se puede ver un ejemplo de sprint backlog con dos tareas ficticias.

EP	HU	T. estimado	T. empleado	Tareas	Estado
-	-	1 hora	2 h 3 min	Descripción de la tarea 1	Completada
EP00	HU00	5 horas	3 h 50 min	Descripción de la tarea 2	Iniciada
Trabajo total			**5 h 53 min**		

Tabla 7.1: Ejemplo de Sprint backlog

EP	HU	T. estimado	T. empleado	Tareas	Estado
-	-	1 hora	2h 30 min	Qué es un diccionario de datos y ejemplos	Completada
-	-	30 min	15 min	Angular vs React	Completada
-	-	1 hora	2 h 13 min	Bases de datos no relacionales	Completada
-	-	30 min	18 min	Gherkin	Completada
-	-	4 horas	2 h	Latex	Completada
-	-	2 horas	4 h 27 min	Lectura de otros TFGs para ver su estructura	Completada
-	-	5 horas	6 h 41 min	Redactar capítulos 1 y 2 del informe	Iniciada
Trabajo total			**18 h 24 min**		

Tabla 7.2: Sprint 0

7.2. Seguimiento por sprints

7.2.1. Sprint 0 (15/02/2024 - 14/03/2024)

Durante este periodo se llevarán a cabo las tareas de investigación necesarias para realizar el proyecto y se elegirán las tecnologías a utilizar. Una vez realizada la investigación se empezará a redactar los capítulos 1 y 2 correspondientes a la introducción y planificación, en este último se establecerán unas fechas orientativas para los próximos sprints.

Las tareas a realizar en el sprint 0 se encuentran en la Tabla 7.2.

7.2.2. Sprint 1 (14/03/2024 - 04/04/2024)

Sprint planning

En este primer sprint, se trabajarán las historias de usuario HU01 y HU07. Además, se deberá dedicar un tiempo a trabajar las secciones del informe que quedaron pendientes del sprint 0.

Al ser las primeras historias de usuario en las que se trabaja, se estima que tomarán más tiempo de lo normal. A ambas historias se les ha asignado 3 puntos de usuario, mientras que a la tarea de completar el informe se le ha asignado 2 puntos.

En la Tabla 7.3 se puede observar el Sprint backlog de este periodo.

EP	HU	T. estimado	T. empleado	Tareas	Estado
-	-	10 horas	25 h 21 min	Completar secciones de introducción, requisitos y planificación de la memoria del TFG	Incompleta
EP01	HU01	15 horas	30 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Iniciada No iniciada No iniciada No iniciada
EP02	HU07	15 horas	1 h 23 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Iniciada No iniciada No iniciada No iniciada
Trabajo total			**27 h 14 min**		

Tabla 7.3: Sprint backlog - Sprint 1

Sprint review

No se han realizado la mayoría de las tareas planificadas para este sprint. Aún así, se deberán realizar algunos cambios en las realizadas.

En cuanto al prototipo, se debería elegir un nombre para la aplicación y realizar un logotipo.

Será necesario realizar un par de cambios en el modelo del dominio. El datatype nombrado *Fecha* se cambiará por *Momento* para reflejar que se quiere recoger tanto de la fecha como de la hora. La clase *Rol* se sustituirá por una clase asociación, para representar la relación entre un *Usuario* y un *Diccionario*.

Sprint retrospective

La tarea de completar la introducción y planificación de la memoria ha llevado mucho más tiempo del estimado, sumado a eso la parte de planificación sigue incompleta. Será necesario trabajar la soltura a la hora de redactar el informe para poder dedicarle más tiempo a otras tareas.

Se mantendrán las historias de usuario para el próximo sprint, además se deberá concluir la sección de planificación. También se empezará a redactar la sección de tecnologías utilizadas.

7.2.3. Sprint 2 (04/04/2024 - 18/04/2024)

Sprint planning

Se continuará con las historias de usuario iniciadas en el sprint 1. Se deberá completar el apartado de presupuesto de la sección de planificación y comenzar a redactar las tecnologías utilizadas para el desarrollo del proyecto.

En la Tabla 7.4 se puede observar el Sprint backlog de este periodo.

Sprint review

La historia de usuario HU01 se encuentra casi completada, solo falta la realización de los test. Se debe decidir si se realizarán los test solo de la parte del frontend o también los correspondientes al servidor, además de las tecnologías a utilizar para ello, se ha propuesto utilizar Jasmine. Se ha ajustado la tarea relacionada con la redacción de la sección tecnologías ya que se había puesto que se completaría, pero al tratarse de una tarea que se irá ajustando a lo largo del tiempo solo se iba a iniciar. Se han añadido algunas tareas que en un principio no estaban contempladas, investigación del funcionamiento de Angular, Spring y Angular Material.

Sprint retrospective

Será necesario investigar acerca del framework de pruebas Jasmine. También se investigará acerca del framework Spring Security para controlar el acceso de los usuarios. Para realizar la EP10 será necesario ir investigando cómo implementar el cambio de idioma, ya que si se deja para más adelante es posible que la tarea de traducción se haga más pesada.

EP	HU	T. estimado	T. empleado	Tareas	Estado
-	-	5 horas	3 h 13 min	Completar secciones de introducción, requisitos y planificación de la memoria del TFG	Completada
-	-	5 horas	8 min	Comenzar a redactar la sección de tecnologías utilizadas de la memoria del TFG	Iniciada
EP01	HU01	15 horas	18 min	Análisis Diseño Desarrollo Pruebas	Completada Completada No iniciada No iniciada
EP02	HU07	15 horas	22 h 13 min	Análisis Diseño Desarrollo Pruebas	Completada Completada Completada No iniciada
-	-	-	5 h 27 min	Investigación sobre funcionamiento y arquitectura de Spring y Angular	Completada
-	-	-	2 h 43 min	Investigación sobre funcionamiento de la biblioteca Angular Material y aplicación práctica	Completada
-	-	-	2 h	Configuración del tema predeterminado de la aplicación con Angular Material	Completada
Trabajo total			**36 h 2 min**		

Tabla 7.4: Sprint backlog - Sprint 2

7.2.4. Sprint 3 (18/04/2024 - 02/05/2024)

Sprint planning

Para el sprint 3 se continuará rellenando la sección de tecnologías utilizadas, se realizará el desarrollo de HU01 y se investigará sobre el framework Jasmine para realizar las pruebas de las historias HU01 y HU07. También se investigará sobre cómo utilizar el framework Spring Security para controlar el acceso a ciertos recursos. Por último, también se invesitagará cómo implementar el cambio de idioma en un aplicación angular para poder realizar la EP10. En la Tabla 7.5 se puede observar el Sprint backlog de este periodo.

Sprint review

En cuanto al modelo conceptual se habían añadido algunas clases relacionadas con la parte del modelado de diccionarios, en un principio se había planteado que solo habría una base de datos por diccionario. Sin embargo, se ha propuesta la opción de que cada diccionario pueda pertenecer a un proyecto con varias bases de datos, las cuales pueden ser de diferentes tipos. Por lo tanto, será necesario adaptar el modelo conceptual a estas nuevas condiciones. También se ha planteado el modelado de las operaciones.

Sprint retrospective

Durante la segunda semana del sprint se materializó el riesgo R05 recogido en la Tabla 2.15. Esto provocó que no se pudiera trabajar en el proyecto tanto como se esperaba y muchas tareas ni siquiera se iniciaron. Como acción correctiva, se pasarán todas las tareas sin finalizar al siguiente sprint.

EP	HU	T. estimado	T. empleado	Tareas	Estado
-	-	5 horas	-	Continuar con la sección de tecnologías utilizadas de la memoria del TFG	Iniciada
-	-	3 horas	2 h 9 min	Investigar sobre cambio de idiomas en Angular	Completada
-	-	3 horas	4h 8 min	Investigación sobre funcionamiento del framework Spring Security	Iniciada
-	-	3 horas	-	Investigación sobre funcionamiento del framework Jasmine	No iniciada
EP01	HU01	10 horas	22 h 28 min	Desarrollo Pruebas	Completada No iniciada
EP02	HU07	1 hora	-	Pruebas	No iniciada
EP03	HU10	15 horas	3 h 14 min	Prototipado Análisis Diseño Desarrollo Pruebas	Iniciada Iniciada No iniciada No iniciada No iniciada
EP10	-	-	40 min	Traducción de la página de creación de usuarios	Completada
Trabajo total			**32 h 39 min**		

Tabla 7.5: Sprint backlog - Sprint 3

7.2.5. Sprint 4 (02/05/2024 - 17/05/2024)

Sprint planning

En este sprint se continuará con las tareas que no se pudieron finalizar en el sprint anterior debido al riesgo manifestado. Si diera tiempo se añadirán algunas tareas más. En la Tabla 7.6 se recogen las tareas realizadas durante el sprint 4.

Sprint review

En el listado de bases de datos de un proyecto se añadirá un campo para indicar el tipo de BD, relacional o no, y otro para el sistema gestor de base de datos específico. Se ha establecido que el único usuario capaz de asignar roles a los usuarios en un proyecto será un administrador. Por defecto, cuando un usuario crea un proyecto tendrá el rol de arquitecto, sin embargo, esto podría ser cambiado por un administrador. Cuando se deshabilite la cuenta de un usuario se eliminarán todas los roles de sus proyectos.

Sprint retrospective

Antes de implementar Spring Security, el usuario tenía un email y un username, generado a partir del email, con el que iniciaba sesión, ya se había comentado que utilizar el email para iniciar sesión podía ser una opción más amigable para el usuario, sin embargo, como Sprint Security utiliza una configuración predeterminada en la que usa los atributos username y password para autenticar al usuario, se decidió directamente que el atributo email se convirtiese en el username.

En la Tabla 2.16 se puede ver el riesgo R06, el cual se ha materializado en este sprint, ya que debido al desconocimiento de Spring Security y Jasmine ha sido necesario invertir mucho tiempo en aprender estas tecnologías y no se pudieron completar todas las tareas contempladas para este sprint a pesar de haber dedicado más tiempo de trabajo total. Se continuará con el desarrollo de HU10 en el próximo sprint.

EP	HU	T. estimado	T. empleado	Tareas	Estado
-	-	5 horas	-	Continuar con la sección de tecnologías utilizadas de la memoria del TFG	Iniciada
-	-	5 horas	19 h 6 min	Continuar con la investigación sobre Spring Security y realizar su implementación	Completada
-	-	5 horas	11 h	Investigación sobre funcionamiento del framework Jasmine	Completada
EP01	HU01	1 hora	5 h 6 min	Pruebas	Completada
EP02	HU07	1 hora	1 h 36 min	Pruebas	Completada
EP03	HU10	10 horas	6 h 38 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Iniciada No iniciada
-	-	2 horas	2 h 18 min	Redactar en el informe el seguimiento del sprint anterior y el actual, completar introducción del capítulo Seguimiento del proyecto	Completada
EP02	HU09	5 horas	1 h 38 min	Diseño Desarrollo Pruebas	Completada Completada Completada
Trabajo total			**47 h 22 min**		

Tabla 7.6: Sprint backlog - Sprint 4

7.2.6. Sprint 5 (14/06/2024 - 28/06/2024)

Sprint planning

Para facilitar el seguimiento de historias de usuario y épicas y saber cuales están finalizadas, se ha sugerido utilizar la herramienta de planificación de GitLab, la cual utiliza **issues** y varios tableros en los que se repartirán dichos issues, de esta forma se identificará rápidamente el estado actual de cada tarea. Cada issue representará una épica o historia de usuario.

En este sprint se dará un poco más de importancia a la tarea de redacción del informe, específicamente a los apartados de tecnologías utilizadas y diseño. Aunque también se continuará implementando la HU10 iniciada en el sprint anterior y se intentará desarrollar por completo la HU11. En la Tabla 7.7 se resumen las tareas a realizar en este sprint.

Sprint review

En este sprint también se ha materializado el riesgo R05 detallado en la Tabla 2.15, debido a que se han tenido que realizar exámenes en la convocatorio extraordinaria. Por lo tanto no se ha podido realizar todo el trabajo planificado. Se pasarán todas las tareas sin finalizar al siguiente sprint.

Sprint retrospective

Ha sido necesario añadir las historias de usuario HU17 y HU18, ya que son necesarias para ver los diccionarios recién creados y para seleccionar un diccionario en específico.

Se ha visto necesario empezar a redactar algunas reglas de negocio necesarias para definir el funcionamiento del sistema.

EP	HU	T. estimado	T. empleado	Tareas	Estado
-	-	15 horas	3h 46 min	Continuar con la sección de tecnologías utilizadas de la memoria del TFG	Iniciada
-	-	5 horas	2 h 40 min	Conocer la herramienta de planificación de GitLab y ordenar las historias de usuario	Completada
-	-	5 horas	-	Comenzar a redactar en el capítulo de diseño la sección sobre la arquitectura del proyecto.	No iniciada
EP03	HU10	5 horas	3 h 39 min	Desarrollo Pruebas	Completada Completada
EP03	HU11	10 horas	-	Prototipado Análisis Diseño Desarrollo Pruebas	No iniciada No iniciada No iniciada No iniciada No iniciada
EP05	HU17	5 horas	4 h 16 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Completada Completada
EP05	HU18	5 horas	2 h 29 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Completada No iniciada
Trabajo total			**16 h 50 min**		

Tabla 7.7: Sprint backlog - Sprint 5

7.2.7. Sprint 6 (28/06/2024 - 11/07/2024)

Sprint planning

Se continuará con las tareas del sprint anterior y además se definirán las reglas de negocio coherentes con el funcionamiento de la aplicación. En la Tabla 7.8 se puede ver las tareas planificadas para este sprint.

Sprint review

Durante el transcurso del sprint se materializó el riesgo R01 el cual se menciona en la Tabla 2.11. Este sprint es el que ha tenido la menor dedicación de tiempo de trabajo hasta ahora, lo cual afectará seguramente a los próximos sprints. Se ha planteado la posibilidad de realizar una replanificación, la cual se haría en el próximo sprint.

Sprint retrospective

Para facilitar la definición de las reglas de negocio también se realizarán diagramas de estados que ayuden a describir el funcionamiento de la aplicación.

EP	HU	T. estimado	T. empleado	Tareas	Estado
-	-	5 horas	-	Continuar con la sección de tecnologías utilizadas de la memoria del TFG	Iniciada
-	-	10 horas	-	Comenzar a redactar en el capítulo de diseño la sección sobre la arquitectura del proyecto.	No iniciada
-	-	4 horas	1 h 34 min	Comenzar a realizar diagramas de estados.	Iniciada
-	-	10 horas	5 h 12 min	Comenzar a definir reglas de negocio	Iniciada
EP03	HU11	10 horas	-	Prototipado Análisis Diseño Desarrollo Pruebas	No iniciada No iniciada No iniciada No iniciada No iniciada
EP05	HU18	1 hora	46 min	Pruebas	Completada
Trabajo total			**7 h 32 min**		

Tabla 7.8: Sprint backlog - Sprint 6

7.2.8. Sprint 7 (11/07/2024 - 25/07/2024)

Sprint planning

Se continuará dando importancia a las tareas relacionadas con la redacción del informe, sobre todo a las relacionadas con la delimitación del funcionamiento, aunque también habrá que continuar con el desarrollo de la funcionalidad realizando las historias de usuario HU11 y HU18. El sprint backlog con las tareas a realizar se encuentra en la Tabla 7.9.

Sprint review

De nuevo ha habido poca dedicación en este sprint, en parte debido al riesgo R05, desarrollado en la Tabla 2.15. La estudiante está trabajando a jornada completa en sus prácticas de empresa, lo que ha reducido el tiempo disponible para avanzar en el proyecto.

Sprint retrospective

Debido a la escasa dedicación en los últimos sprints, se ha acumulado una cantidad significativa de tareas pendientes. Es probable que no sea posible realizar el proyecto en el tiempo inicialmente planificado. Por ello, se realizará una replanificación, estimando el

EP	HU	T. estimado	T. empleado	Tareas	Estado
-	-	5 horas	-	Continuar con la sección de tecnologías utilizadas de la memoria del TFG	Iniciada
-	-	10 horas	5 h 10 min	Continuar con los diagramas de estados y empezar con los de flujo	Incompleta
-	-	5 horas	-	Comenzar a redactar en el capítulo de diseño la sección sobre la arquitectura del proyecto.	No iniciada
EP01	HU02	10 horas	-	Prototipado Análisis Diseño Desarrollo Pruebas	No iniciada No iniciada No iniciada No iniciada No iniciada
EP03	HU11	10 horas	8 h 48 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Incompleta No iniciada
Trabajo total			**14 h 58 min**		

Tabla 7.9: Sprint backlog - Sprint 7

porcentaje de trabajo completado hasta ahora y cuánto queda para finalizar. A partir de esta estimación, se ajustará el número de sprints necesarios para concluir el proyecto.

7.2.9. Sprint 8 (29/08/2024 - 12/09/2024)

Sprint planning

Para retomar el ritmo de trabajo, se ha decidido que este sprint se enfoque únicamente en el desarrollo de historias de usuario. Específicamente, se han seleccionado las historias HU02, HU11 y HU12. En la Tabla 7.10 se encuentra el sprint backlog detallado.

Sprint review

Se ha invertido una gran parte del tiempo en refactorización del código y mejora de algunos test, ya que se detectaron ciertas partes del código con mucho margen de mejora. Abordar esto en este momento evita un aumento de la deuda técnica que podría haber generado problemas en el futuro.

EP	HU	T. estimado	T. empleado	Tareas	Estado
EP03	HU11	10 horas	6 h 14 min	Desarrollo Pruebas	Completada Completada
EP03	HU12	15 horas	-	Prototipado Análisis Diseño Desarrollo Pruebas	No iniciada No iniciada No iniciada No iniciada No iniciada
EP01	HU02	10 horas	-	Prototipado Análisis Diseño Desarrollo Pruebas	No iniciada No iniciada No iniciada No iniciada No iniciada
-	-	5 horas	23 h 8 min	Refactoring del código escrito en sprints anteriores.	Completada
-	-	5 horas	16 h 40 min	Reescritura de test y aumento de la cobertura añadiendo nuevos test.	Completada
Trabajo total			**46 h 2 min**		

Tabla 7.10: Sprint backlog - Sprint 8

Sprint retrospective

Será necesario profundizar en el uso de Jasmine para realizar los test, ya que aún hay comportamientos en el código que no termino de comprender. Con el ritmo de trabajo ya restablecido, en el próximo sprint se retomará el desarrollo del informe, aunque no es seguro que en el próximo sprint pueda tener la dedicación planificada debido a un viaje de trabajo y a la memoria de prácticas externas que también debo completar.

7.2.10. Sprint 9 (12/09/2024 - 26/09/2024)

Sprint planning

Se continuará con las historias de usuario HU12 y HU02 que no se pudieron empezar en el sprint anterior y también se ha añadido la historia HU04 que está relacionada con la HU02 y es más corta. En cuanto al informe se trabajará en el capítulo de análisis.

En la Tabla 7.11 se encuentra el sprint backlog detallado.

Sprint review

Como se había anticipado, durante la primera semana de este sprint no pude avanzar en ninguna de las tareas seleccionadas. Esto se debió a un viaje de trabajo y la necesidad de finalizar, antes de la fecha límite, la memoria de la asignatura Prácticas externas, por lo que tuve que dedicar todo el tiempo disponible en esto.

La HU02 ha llevado mucho más tiempo del esperado debido a complejidades durante el desarrollo del front-end y al mapeo de objetos en el back-end. Haber enfrentado estas dificultades en este momento servirá para agilizar el desarrollo de las tareas en los próximos sprints.

Actualmente, las tareas de prototipado y desarrollo del front-end, especialmente la parte estética, me llevan mucho más tiempo que el desarrollo del back-end, que suele ser un proceso más automático.

Sprint retrospective

Sería recomendable continuar con el resto de tareas de la EP01 ya que están relacionadas con las historias HU02 y HU04 y son tareas sencillas de realizar.

Una sugerencia para el próximo sprint sería empezar realizando las tareas de redacción de la memoria del TFG, dado que suele dejarse para el final, provocando un retraso en su realización que finalmente impida presentar el TFG cerca de la fecha prevista.

EP	HU	T. estimado	T. empleado	Tareas	Estado
EP03	HU12	15 horas	-	Prototipado Análisis Diseño Desarrollo Pruebas	No iniciada No iniciada No iniciada No iniciada No iniciada
EP01	HU02	10 horas	19 h 56 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Completada Completada
EP01	HU04	5 horas	3 h 33 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Completada Completada
-	-	10 horas	-	Redactar el capítulo de análisis	No iniciada
Trabajo total			**23 h 29 min**		

Tabla 7.11: Sprint backlog - Sprint 9

7.2.11. Sprint 10 (26/09/2024 - 10/10/2024)

Sprint planning

Como se indicó en la retrospectiva del sprint anterior, en este sprint se desarrollarán las historias de usuario de la épica EP01 para así dar por terminado el desarrollo de la gestión de usuarios. Sin embargo, se ha detectado que una de las historias, HU03, se podía dividir en varias ya que implicaba desarrollar 4 funcionalidades distintas. Por ello, han surgido 4 nuevas historias de usuario, específicamente HU30, HU31, HU32, HU33.

La primera tarea a realizar en este sprint será la redacción del capítulo de Análisis de la memoria del TFG, de esta forma se asegurará haberla completado al final del sprint.

Por lo tanto el sprint backlog quedará como se indica en la Tabla 7.12.

Sprint review

Algunos botones no son lo suficientemente descriptivos con su función, por lo que se ha recomendado añadir tooltips de tal forma que al pasar el ratón por encima se muestre una pequeña pista de lo que hace.

Sprint retrospective

Se deberá investigar acerca de las responsabilidades en el gobierno de datos para elegir un nombre adecuado para el rol que desempeñan los usuarios que tienen acceso a los diccionarios de datos, estos se llamaban usuarios normales o usuarios con rol, pero estos nombres pueden llevar a malentendidos.

El correo electrónico se utiliza como clave primaria de los usuarios, ya que es un campo único. Sin embargo, sería más conveniente usar un identificador que sea, por ejemplo, un número autogenerado. Esto evitaría tener que actualizar el correo electrónico en otras tablas cuando se modifica en la tabla de usuarios, ya que no se usaría como clave foránea.

Tabla 7.12: Sprint backlog - Sprint 10

EP	HU	T. estimado	T. empleado	Tareas	Estado
-	-	10 horas	6 h 40 min	Redactar el capítulo de análisis	Completada
EP01	HU05	5 horas	6 h 47 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Completada Completada
EP01	HU06	5 horas	4 h 3 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Completada Completada
EP01	HU30	5 horas	4 h 19 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Completada Completada
EP01	HU31	5 horas	4 h 6 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Completada Completada
EP01	HU32	5 horas	-	Prototipado Análisis Diseño Desarrollo Pruebas	No iniciada No iniciada No iniciada No iniciada No iniciada
EP01	HU33	5 horas	38 min	Prototipado Análisis Diseño	Completada Completada Completada

Continúa en la siguiente página

Tabla 7.12 – viene de la página anterior

EP	HU	T. estimado	T. empleado	Tareas	Estado
				Desarrollo Pruebas	Completada Completada
-	-	1 horas	15 h 22 min	Reflexión general sobre el proyecto como la implementación de las funcionalidades pendientes, su complejidad y alcance, revisión de entidades en el modelado del dominio.	Completada
-	-	1 horas	2 h 53 min	División de HU03 en historias de usuario más específicas. Actualización del informe para reflejar estos cambios e inclusión de las tareas faltantes en los issuess de GitLab.	Completada
Trabajo total			**44 h 48 min**		

7.2.12. Sprint 11 (10/10/2024 - 24/10/2024)

Sprint planning

Se empezará por la historia HU32 que quedó pendiente en el sprint anterior y así finalizar la épica EP01. A continuación se realizarán las historias HU19 y HU12 que están relacionadas entre sí, ya que se trata de la visualización de las tablas/colecciones de una base de datos y su creación. Una vez realizada la historia HU12 se dará por completada la épica EP03.

También se trabajará en la memoria del TFG redactando parte de los capítulos sobre las tecnologías utlizadas y el diseño. Además, se añadirán algunas tareas para realizar las mejoras contempladas en la revisión y la retrospectiva del sprint anterior. El sprint backlog se encuentra en la Tabla 7.13.

Sprint review

Para referirse al tipo de usuario que no es administrador se ha elegido el nombre "Metadata steward", que se traducirá como "Gestor de metadatos". Este nombre se ha elegido a partir de los tipos de administradores en el gobierno de datos [37], siendo "Metadata steward" el más adecuado debido a la temática de la aplicación.

Se añadió el campo "id" como identificador para los usuarios, debido a este cambio, el mapeo de las entidades a objetos de envío de información empezó a dar problemas. Para este mapeo se utilizaba la biblioteca ModelMapper [32], la cual simplificaba en gran parte la transformación de objetos de una clase a otra. ModelMapper utiliza los nombres y tipos de los atributos para determinar automáticamente qué atributos de una clase se corresponden con los atributos de la otra clase. Sin embargo, esta simplicidad desaparecía cuando había más de un atributo que podía ser adecuado, por ejemplo, cuando necesitaba transformar dos entidades en un solo objeto, como es el caso de DictionaryEntity y DictionaryRoleEntity a DictionaryData. Por estas dificultades se ha decidido eliminar el uso de la biblioteca y realizar el mapeo de forma manual.

La llamada al toast implicaba la duplicación de código en varias partes de cada componente. Para mejorar esta parte del código se ha creado un componente que simplifica este proceso. Para utilizarlo solo es necesario llamar al método correspondiente al tipo de toast que se necesita, "success", "info", "warning" o "error", añadiendo como parámetro una palabra clave que identifica el valor deseado de los ficheros de traducción.

Hasta ahora, las entidades de los elementos de un diccionario se habían mapeado de manera independiente en JPA, pero esto implicaba duplicación de código por esto se ha investigado la manera de mapear la herencia de los elementos. Se ha observado que existen tres estrategias [29] dependiendo del número de tablas que se quiera generar, SINGLE_TABLE, JOINED y TABLE_PER_CLASS. Se ha concluido que la estrategia más adecuada en este caso es JOINED, lo que implica tener un tabla para la superclase y una por cada subclase, en la tabla de la superclase se almacenan lo atributos comunes y en las de la subclase los atributos específicos de cada uno. Las razones de esta elección son varias, es conveniente distinguir los elementos según su tipo, cada elemento tiene un lista de subelementos. Tener una sola tabla con todos los elementos facilitará la realización de acciones en elementos específicos sin necesidad de diferenciar entre tipos.

En cuanto a los nombres de los elementos que se refieren a tablas o colecciones los cuales se denominan "MainElement" y a sus subelementos columnas o campos, llamados "InternalElement" se ha decidido asignarles nombres más acordes al contexto de las bases de datos, que sean abstractos y que no se vinculen específicamente a bases de datos relacionales o no relaciones. Por esto, se ha decidido utilizar los nombres "Entidad" y "Atributo", estos son conceptos ampliamente utilizados en el modelado de datos [39] y aunque se suele asociar con bases de datos relacionales se puede usar de manera abstracta para representar y describir conceptos. A su vez la clase llamada "Atributo" se pasará a llamar "Metapropiedad" para evitar la duplicación de nombres, además resulta más descriptivo.

Para evitar una posible confusión del usuario en las páginas con desplazamiento vertical, donde podría haber más contenido y no darse cuenta, se han añadido unos botones para facilitar la navegación hacia arriba o hacia abajo [27]. Estos botones solo aparecen cuando hay más contenido en las respectivas direcciones.

EP	HU/HF	T. estimado	T. empleado	Tareas	Estado
-	-	5 horas	-	Redactar parte del capítulo de Tecnologías	No iniciada
-	-	5 horas	-	Redactar parte del capítulo de Diseño	No iniciada
EP01	HU32	5 horas	5 h 14 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Completada Completada
EP05	HU19	5 horas	12 h 2 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Completada Completada
EP03	HU12	10 horas	4 h 8 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Completada Completada
-	HF01	5 horas	2 h 23 min	Cambiar los nombres de los roles de los tipos de usuarios	Completada
-	HF02	5 horas	6 h 9 min	Cambiar el campo que hace de clave primaria para los usuarios	Completada
-	HF03	1 hora	45 min	Añadir tooltips a los botones sin texto	Completada
-	HF04	2 horas	1 h	Crear método para llamar al toast	Completada
-	HF05	1 hora	39 min	Dejar de usar ModelMapper	Completada
-	HF06	5 horas	4 h 22 min	Modelar la herencia de los elementos con JPA	Completada
-	HF08	2 horas	3h 23 min	Despliegue de la aplicación en la MV	Completada
Trabajo total			**40 h 5 min**		

Tabla 7.13: Sprint backlog - Sprint 11

Sprint retrospective

El próximo sprint es el último planificado, en la próxima reunión será esencial identificar y priorizar las historias de usuario que aporten al proyecto la funcionalidad esencial y que sea coherente. Puede que sea necesario planificar un sprint extra.

7.2.13. Sprint 12 (14/11/2024 - 28/11/2024)

Sprint planning

En este sprint, se trabajará exclusivamente en el desarrollo de historias de usuario, principalmente serán tareas relacionadas con la edición de los elementos del diccionario. La HU20 permitirá visualizar los atributos de las entidades y sus metapropiedades, mientras que con la HU13 se podrán añadir nuevos atributos a una entidad. Una vez realizado esto, la HU15 permitirá editar o enviar propuestas de cambio de las metapropiedades de un atributo. Por último, con el desarrollo de la HU28 los arquitectos podrán revisar cambios propuestos por editores.

Se ha echado en falta que la sesión de un usuario permanezca activa en varias pestañas del navegador, permitiendo la consulta de varias páginas de manera simultánea. Para ello, se ha incluido la tarea HF07, que tratará el cambio en el alcance de la sesión.

Las tareas a desarrollar en este sprint se encuentran en la Tabla 7.14.

Sprint review

Debido al cambio en el alcance de la sesión, la cuenta del usuario permanece activa incluso cuando el navegador se cierra. Sin embargo, el token JWT tiene un tiempo de validez limitado, al caducar, las peticiones enviadas al servidor generaban errores, aunque en el lado del cliente parecía que el usuario seguía autenticado. Para resolver este problema, se ha añadido una comprobación adicional en el interceptor de autenticación [23]. Con esto, si el servidor devuelve un error 401 (Unauthorized) debido a la invalidez o caducidad del token, el sistema redirige automáticamente al usuario a la página de inicio de sesión. Se esperaba que la realización de esta tarea fuera breve. Sin embargo, debido a este error, el tiempo requerido fue bastante mayor al previsto.

La HU20 también ha llevado más tiempo del esperado debido a que aún no se había establecido la manera de almacenar las metapropiedades de los atributos. Gran parte del tiempo se ha dedicado a explorar las diferentes estructuras de datos y elegir la más adecuada. Finalmente, se optó por utilizar el tipo Record [33] en el frontend y HashMap en el backend.

Por otra parte, la HU13 solo ha requerido una pequeña parte del tiempo estimado, ya que se ha simplificado la creación de un atributo para que sea parecida al resto de elementos del diccionario. En un principio, se había considerado permitir que las metapropiedades se

EP	HU/HF	T. estimado	T. empleado	Tareas	Estado
EP05	HU20	5 horas	9 h 36 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Completada Completada
EP04	HU13	10 horas	1 h 23 min	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Completada Completada
EP04	HU15	15 horas	19 h	Prototipado Análisis Diseño Desarrollo Pruebas	Completada Completada Completada Incompleta No iniciada
EP04	HU28	10 horas	-	Prototipado Análisis Diseño Desarrollo Pruebas	No iniciada No iniciada No iniciada No iniciada No iniciada
-	HF07	1 hora	5 h 19 min	Cambiar el alcance de la sesión al navegador	Completada
Trabajo total			**35 h 18 min**		

Tabla 7.14: Sprint backlog - Sprint 12

añadieran durante la creación del atributo, pero finalmente solo será necesario especificar el nombre, la descripción y el tipo de dato. Para añadir metapropiedades se deberá editar el atributo después de su creación.

Durante el desarrollo de la HU15, el tipo de dato elegido para las metapropiedades comenzó a ser un problema. Aunque inicialmente se eligieron mapas, esta estructura no era suficiente para determinar el orden en el que se almacenarían las metapropiedades. De esta manera se decidió almacenar las metapropiedades en su propia clase, compuesta por tres campos: posición, metaclave y metavalor. Solo se considera que se ha realizado un cambio en un atributo cuando al menos uno de los tres campos de alguna de las metapropiedades es diferente, si no hay ningún cambio, se detecta en el servidor. Por lo tanto, un atributo puede tener las mismas metapropiedades pero en distinto orden y se consideraría un cambio. También fue necesario verificar que las posiciones fueran números consecutivos, del 1 al número total de metapropiedades, ya que de lo contrario no se mostraban correctamente en el frontend.

Sprint retrospective

Aún quedan aspectos relacionados con el tipo de dato utilizado para las metapropiedades que deben ser considerados, por lo que no se ha podido finalizar la HU15 ni comenzar con la HU28, revisión de los cambios propuestos, antes será necesario establecer una manera efectiva de almacenar tanto los datos originales como los nuevos. Además, aún queda pendiente determinar cómo se bloqueará la edición de un atributo cuando haya propuestas pendientes de aprobación o rechazo.

Por ahora, se añadirá un sprint adicional para intentar finalizar las historias de usuario relacionadas con la edición. Asimismo, se decidirá cuáles no se desarrollarán y se dejarán como líneas de trabajo futuras. Más adelante, una vez finalizada la época de exámenes, se evaluará cómo proceder.

7.2.14. Sprint 13 (28/11/2024 - 12/12/2024)

Sprint planning

En este sprint, se deberá completar la HU15, iniciada en el sprint anterior, la cual trata la edición de los atributos. Una vez finalizada, se continuará con la HU28, correspondiente a la aprobación de las propuestas de cambio, y la HU14, enfocada en la edición de nombres y descripciones de los elementos.

También se deberá redactar parte del capítulo de Diseño, ya que la parte de redacción de la memoria del TFG se está quedando atrasada respecto a la parte de desarrollo.

En la Tabla 7.15 se resumen las tareas planificadas para este sprint.

Sprint review

Han seguido surgiendo problemas con respecto al desarrollo de la HU15. Por una parte, la forma de almacenar las propuestas no era del todo óptima. El tipo de dato de un atributo estaba separado de las propuestas de cambio, cuando en realidad debería formar parte de ellas. Por lo tanto, en la parte del servidor, ya no es la clase Atributo quien contiene el tipo de dato y la metapropiedad. En su lugar, se ha creado una nueva clase llamada Contenido, que agrupa ambos. Cada propuesta tiene un contenido con los datos originales y otro con los datos modificados, además de otros campos como el estado de la propuesta. Se considera que el contenido modificado de la última propuesta de un atributo, cuyo estado es “aceptado”, es el contenido real del atributo.

Por otro lado, en cuanto al bloqueo de la edición del atributo cuando está siendo editado, inicialmente se planeaba usar un token que indicara si el atributo era editable o no. Sin embargo, este enfoque podría generar problemas si, por ejemplo, el usuario cierra el navegador sin cancelar la acción, ya que bloquearía el atributo indefinidamente. Aunque se

EP	HU/HF	T. estimado	T. empleado	Tareas	Estado
-	-	10 horas	8 h 33 min	Redactar parte del capítulo de Diseño	Iniciada
EP04	HU15	15 horas	26 h 19 min	Desarrollo Pruebas	Completada Completada
EP04	HU28	10 horas	-	Prototipado Análisis Diseño Desarrollo Pruebas	No iniciada No iniciada No iniciada No iniciada No iniciada
EP04	HU14	5 horas	-	Prototipado Análisis Diseño Desarrollo Pruebas	No iniciada No iniciada No iniciada No iniciada No iniciada
Trabajo total			**34 h 52 min**		

Tabla 7.15: Sprint backlog - Sprint 13

podría haber implementado un temporizador, esto habría alargado el tiempo de desarrollo de la HU15. Finalmente, se ha optado por bloquear la edición cuando exista una propuesta pendiente. Sin embargo, este enfoque trae consigo unos casos especiales a tener en cuenta. Por ejemplo, cuando dos usuarios acceden a la edición de un atributo, ambos realizarán sus cambios, pero solo se guardará la propuesta del primero que la envíe, mientras que la del segundo será descartada. Otro caso ocurre cuando un arquitecto y un editor editan el atributo simultáneamente. Dado que los cambios del arquitecto no requieren aprobación, el atributo no se bloqueará, y el editor podrá enviar su propuesta. Sin embargo, los datos originales de la propuesta del editor corresponderán al atributo antes de que el arquitecto realizara sus cambios.

Finalmente, se ha logrado completar la HU15. Esta ha sido, con diferencia, la historia de usuario que más tiempo de desarrollo ha requerido. Esto podría deberse, en primer lugar, a una división inadecuada de la historia de usuario, que abarcaba demasiadas funcionalidades y presentaba una complejidad excesiva. Por otro lado, la selección de los tipos de datos resultó problemática. Aunque se intentó seleccionar la opción más apropiada desde el principio, a medida que se avanzaba en el desarrollo, surgieron ciertas limitaciones que han provocado varias refactorizaciones del código.

Sprint retrospective

Será necesario al menos un sprint más para dar por cerrada la épica de edición de elementos de un diccionario, siendo esto la parte más importante de lo que queda de desarrollo de la aplicación, pero esto se detallará mejor en el próximo sprint.

También se deberá dar un mayor peso a la redacción de la memoria, ya que a pesar de haber dedicado algo de tiempo en este sprint aun queda mucho por redactar.

7.2.15. Sprint 14 (03/02/2025 - 17/02/2025)

Sprint planning

En este sprint se dará prioridad a la finalización de las historias de usuario que abarcan la funcionalidad básica del sistema. La HU14 contempla la edición de nombres y descripciones de los elementos por parte de un arquitecto y la HU16 el borrado de estos elementos y todos los subelementos que contiene. La HU28 por otra parte consiste en implementar las revisiones de las propuestas de cambios realizadas por editores y que deben ser validadas por los arquitectos del mismo diccionario. Una vez realizadas estas historias, se dará por completada la tarea de desarrollo. Una vez finalizadas las historias de usuario se dedicará el resto el tiempo a redactar la memoria, priorizando la finalización de los capítulos 2 y 5, sobre requisitos y diseño, respectivamente.

En la Tabla 7.16 se resumen las tareas planificadas para este sprint.

Sprint review

Para asegurar que la edición de nombres y descripciones de los elementos de la HU14 solo fuera accesible para usuarios con rol de arquitecto, ha sido necesario implementar un control de roles. Esta comprobación aún no estaba implementada en el frontend, por lo que la edición era visible para cualquier usuario.

La HU16 era bastante sencilla en cuanto a implementación, ya que solo era necesario habilitar el borrado en cascada en la base de datos para los subelementos del elemento a borrar. Sin embargo, surgió un inconveniente debido a que cada elemento atributo guardaba una referencia a la última propuesta de cambio, y a su vez, todas las propuestas de cambio tenían una referencia al atributo al que pertenecen. Esto generaba una dependencia circular de claves foráneas. La solución más viable consistió en eliminar la referencia a la última propuesta. De esta manera, cada vez que se necesita recuperar la última propuesta, se consulta la fecha de la propuesta más reciente, a pesar de que este campo se había creado explícitamente para evitar consultas adicionales.

La HU28 resultó ser mucho más compleja de lo que se había estimado. Por una parte, requería el diseño de dos páginas distintas, una para ver todas las propuestas y otra para ver la propuesta seleccionada. Por otra parte, no todas las propuestas son iguales, dependen tanto del rol del usuario como del estado en que se encuentran. Por lo tanto, ambas páginas varían mostrando contenido y opciones diferentes.

Dentro de HU28, también ha sido necesario crear un nuevo servicio para traducir las etiquetas de los paginadores [43], ya que hasta ese momento no se estaban traduciendo. Este servicio ahora se aplica automáticamente a todos los paginadores utilizados en el proyecto.

No se ha podido dedicar tiempo a la redacción de la memoria del TFG debido a que la implementación de las historias de usuario ha requerido más tiempo del previsto y era prioridad terminarlas.

EP	HU/HF	T. estimado	T. empleado	Tareas	Estado
-	-	20 horas	-	Redactar informe	No iniciada
EP04	HU14	5 horas	7 h 56 min	Prototipado	Completada
				Análisis	Completada
				Diseño	Completada
				Desarrollo	Completada
				Pruebas	Completada
EP04	HU16	5 horas	8 h 14 min	Prototipado	Completada
				Análisis	Completada
				Diseño	Completada
				Desarrollo	Completada
				Pruebas	Completada
EP04	HU28	10 horas	32 h 59 min	Prototipado	Completada
				Análisis	Completada
				Diseño	Completada
				Desarrollo	Completada
				Pruebas	Completada
Trabajo total			**49 h 9 min**		

Tabla 7.16: Sprint backlog - Sprint 14

Sprint retrospective

Este ha sido el último sprint, por lo que no se iniciarán nuevos sprints para finalizar la redacción de la memoria. No obstante, se deberá dedicar un tiempo adicional para completar y revisar la memoria del TFG.

7.3. Resumen de la ejecución del proyecto

7.3.1. Funcionalidad implementada

Se puede iniciar sesión en el sistema utilizando un correo electrónico y una contraseña. Una vez iniciada la sesión, la página de inicio varía según el tipo de usuario autenticado.

Los administradores tienen la capacidad de crear nuevos usuarios, para lo cual deben indicar el nombre, los apellidos y el correo electrónico, que debe ser único y no repetirse para ningún otro usuario del sistema. Además, deben elegir uno de los dos tipos de usuario, administrador o gestor de metadatos.

También pueden gestionar los usuarios del sistema a través de una lista que se puede filtrar por el contenido de los campos como el nombre o la fecha del último acceso. Para cada usuario, los administradores pueden editar la información asociada, generar una nueva contraseña, la cual debe ser comunicada al usuario de manera externa al sistema en casos como cuando un usuario accede por primera vez o ha olvidado su contraseña, y activar o

desactivar cuentas. La desactivación de una cuenta no conlleva pérdida de datos, sino que simplemente impide que el usuario inicie sesión.

Además, los administradores tienen la capacidad de gestionar el acceso de los gestores de metadatos a diccionarios específicos. Pueden otorgar o revocar el acceso a un diccionario de datos y modificar el rol del usuario dentro de ese diccionario. Para conceder acceso, el administrador puede ver una lista de diccionarios a los que el usuario seleccionado no tiene acceso y realizar búsquedas basadas en diferentes criterios.

En cuanto a la funcionalidad de los gestores de metadatos, estos pueden crear tantos diccionarios de datos como deseen. Al crear un diccionario, automáticamente se les asigna el rol de arquitecto en ese diccionario, aunque este rol puede ser modificado por un administrador. Además, solo el administrador puede añadir nuevos usuarios al diccionario.

Para crear un diccionario de datos, el gestor solo necesita proporcionar un nombre y una descripción. De manera similar, los demás elementos que conforman el diccionario, como bases de datos, entidades y atributos, se crean especificando un nombre y una descripción. La creación de estos elementos está restringida exclusivamente a los arquitectos del diccionario. En el caso de las bases de datos, también es necesario seleccionar el sistema gestor de base de datos. Para los atributos opcionalmente se indica el tipo de dato.

Por otro lado, cualquier usuario con acceso a un diccionario puede visualizar toda la información contenida en él. La edición del nombre y la descripción de los elementos del diccionario también es exclusiva para arquitectos y son cambios que se asientan siempre de manera inmediata.

Tanto arquitectos como editores pueden modificar los atributos existentes, específicamente el tipo de dato y las metapropiedades. Se pueden añadir tantas metapropiedades como se desee y el orden de estas importa. Cuando un editor guarda los cambios estos no se ven reflejados inmediatamente sino que se crea una propuesta cuyo estado es pendiente. Mientras exista una propuesta con este estado la edición del atributo se bloquea. Por otro lado, cuando los cambios son realizados por un arquitecto, estos se aplican de forma inmediata al atributo y no se bloquea su edición.

Los gestores de metadatos pueden visualizar tres tipos de propuestas. Las propuestas pendientes de revisión son aquellas realizadas por editores de un diccionario en el que el usuario tiene el rol de arquitecto y cuyo estado es “pendiente”. El historial de propuestas contiene aquellas en las que el usuario actúa como revisor, y las propuestas tienen los estados “aprobado” o “rechazado”. Por último, las propuestas del usuario son aquellas en las que el usuario es el autor, y estas pueden estar en cualquiera de los estados posibles.

Al acceder a una propuesta, siempre se podrá visualizar tanto el contenido original como el modificado, junto con la información asociada, como la hora de creación, el autor y, en caso de haber sido revisada, el revisor. Si el usuario es revisor, podrá aceptar o rechazar la propuesta, y en caso de rechazo, deberá indicar el motivo. El autor de la propuesta podrá consultar dicho motivo si su propuesta es rechazada.

Los usuarios pueden elegir entre dos idiomas disponibles, inglés o español. Además, esta funcionalidad se ha diseñado de tal manera, que es posible añadir fácilmente más idiomas.

El cierre de sesión puede realizarse manualmente seleccionando la opción en el icono de perfil, aunque también se cierra automáticamente después de un tiempo.

7.3.2. Dedicación

Se han completado un total de **23 historias de usuario** de las 33 inicialmente planteadas. En la Tabla 7.17 se muestran las historias desarrolladas, junto con su tiempo estimado y el tiempo real dedicado a cada una. La suma de las horas estimadas para estas historias era de 175 horas. Sin embargo, el tiempo real invertido ha sido de 261 horas y 29 minutos, lo que supone una desviación de 86 horas. Esta diferencia representa un incremento significativo respecto a lo previsto.

Las historias de usuario que más tiempo de desarrollo requirieron fueron:

- "HU01 – Como administrador, quiero crear nuevos usuarios para poblar el sistema." con 28 horas y 22 minutos.
- "HU07 – Como usuario, quiero ingresar mi nombre de usuario y contraseña para iniciar sesión y acceder a una pantalla distinta dependiendo mi tipo de usuario." con 25 horas y 12 minutos.
- "HU15 – Como arquitecto o editor, quiero modificar el contenido de las columnas/-campos de una tabla/colección para dar información más precisa." con 45 horas y 19 minutos.
- "HU28 – Como arquitecto, quiero ver los cambios realizados por un editor antes de aplicarlos a los elementos para aceptarlos o rechazarlos. Si los rechazo, quiero indicar el motivo para informar al editor." con 32 horas y 59 minutos.

El alto tiempo de desarrollo de las historias HU01 y HU07 se debió a que fueron las primeras en implementarse. En ese momento no se contaba con experiencia previa ni con la configuración inicial del sistema. En el caso de las historias HU15 y HU28, el motivo fue principalmente la complejidad de las funcionalidades.

El **tiempo total de trabajo** realizado a lo largo de los 14 sprints ha sido de **474 horas y 44 minutos**, superando ampliamente las 320 horas que se habían planteado inicialmente.

Este aumento de dedicación también implica un incremento en el coste simulado del proyecto. Volviendo a realizar los cálculos con 475 horas de trabajo y 12 meses de desarrollo, el coste estimado ascendería a 8.279,8 € , una cifra considerablemente superior a los 6.178,3 € del presupuesto simulado original. El desglose de este nuevo cálculo se muestra en la Tabla 7.18. Por otra parte, el coste real es, tal y como se estimó, de 0 €.

HU	Tiempo estimado	Tiempo empleado
HU01	15 horas	28 h 22 min
HU02	10 horas	19 h 56 min
HU04	5 horas	3 h 33 min
HU05	5 horas	6 h 47 min
HU06	5 horas	4 h 3 min
HU07	15 horas	25 h 12 min
HU09	5 horas	1 h 38 min
HU10	10 horas	13 h 31 min
HU11	10 horas	15 h 2 min
HU12	10 horas	4 h 8 min
HU13	10 horas	1 h 23 min
HU14	5 horas	7 h 56 min
HU15	15 horas	45 h 19 min
HU16	5 horas	8 h 14 min
HU17	5 horas	4 h 16 min
HU18	5 horas	3 h 15 min
HU19	5 horas	12 h 2 min
HU20	5 horas	9 h 36 min
HU28	10 horas	32 h 59 min
HU30	5 horas	4 h 19 min
HU31	5 horas	4 h 6 min
HU32	5 horas	5 h 14 min
HU33	5 horas	38 min
Total	**175 horas**	**261 h 29 min**

Tabla 7.17: Dedicación por historia de usuario

Los principales riesgos que se han materializado durante el desarrollo han sido los siguientes:

- **R04 – Gold Plating.** Esta práctica se ha dado de forma generalizada, aunque de manera inconsciente, tendiendo a complicar las soluciones para obtener un resultado más profesional en lugar de optar por enfoques más simples. No obstante, cuando se proponían mejoras que implicaban un esfuerzo más grande, se mitigaba el riesgo

Recurso	Precio
Empleado	6.597,75 €
Ordenador	162,25 €
Licencias de programas	1.508,4 €
Electricidad	11,4 €
Total	**8.279,8 €**

Tabla 7.18: Coste simulado

posponiéndolas para el final del sprint y solo si quedaba tiempo suficiente.

- **R05 – Carga de otras asignaturas.** Durante la mayor parte del desarrollo se han cursado simultáneamente otras asignaturas, así como prácticas de empresa y exámenes. Aunque se intentó mitigar este riesgo planificando en función de los períodos de evaluación, esto no fue suficiente y derivó en numerosos retrasos.
- **R06 – Desconocimiento de las tecnologías.** La necesidad de investigar y aprender las tecnologías utilizadas supuso una carga de trabajo considerable. A pesar de que se planificaron tareas específicas para esto, en general, requirieron más tiempo del estimado.

La materialización de los riesgos descritos previamente provocó la necesidad de introducir cambios en la planificación en distintos momentos del desarrollo. Inicialmente se habían previsto 8 sprints y 2 adicionales, pero la dedicación real resultó muy inferior a la estimada. Esto obligó a una primera replanificación, en la que se añadieron 2 sprints más. Sin embargo, ni siquiera con esta ampliación fue posible alcanzar un producto mínimo viable. A partir de ese momento, se optó por ir añadiendo sprints según las necesidades del proyecto, hasta que se decidió finalizar con el sprint 14. Como resultado, un proyecto que en un principio se había planificado para desarrollarse en 6 meses acabó extendiéndose a lo largo de casi un año.

Capítulo 8

Conclusiones

Al comienzo del proyecto, tras definir las historias de usuario, parecía que sería posible implementar la mayoría sin demasiada dificultad. Sin embargo, con el paso de los sprints, fue quedando claro que desarrollar un proyecto de software completo desde cero conlleva más complejidad de la que se pensaba. A lo largo del proceso surgieron muchos retos, tanto técnicos como de organización, e imprevistos.

El uso del marco de trabajo ágil Scrum ha permitido organizar el desarrollo por iteraciones y una evolución progresiva del sistema, pero tal vez no ha sido la opción más adecuada para este proyecto. En varios momentos ha sido necesario replantearse decisiones tomadas en sprints anteriores, como la estructura de las entidades, ya que algunos requisitos no se habían tenido en cuenta desde el principio. Esto llevó a reescrituras de código que podrían haberse evitado con una fase inicial de análisis y diseño más profunda. Esto demuestra que dedicar más tiempo al diseño antes de comenzar el desarrollo puede ahorrar mucho trabajo a largo plazo.

En cuanto a los objetivos personales, se han cumplido los relacionados con el aprendizaje y la profundización en las tecnologías utilizadas, especialmente Angular y Spring Boot. Respecto a los objetivos del proyecto, se ha cumplido el objetivo principal, enfocado en la creación y gestión de diccionarios de datos, así como en la administración de usuarios y sus roles.

8.1. Líneas de trabajo futuras

Este proyecto es altamente ampliable y muchas de las funcionalidades implementadas podrían mejorarse. Además, quedan pendientes algunas que ya estaban planteadas, pero que no se desarrollaron por falta de tiempo. Las posibles ampliaciones futuras podrían centrarse en los siguientes aspectos:

- **Sistema de mensajería.** Al crear un nuevo usuario, se debería enviar un correo informativo con las instrucciones para iniciar sesión. Este sistema también permitiría gestionar el restablecimiento de contraseñas.
- **Mejoras en la visualización de modificaciones para revisiones.** Implementar un código de colores para facilitar la identificación de cambios: verde para metapropiedades añadidas, rojo para eliminadas y amarillo para modificadas.
- **Configuración del número de revisores.** Cada diccionario de datos debería permitir establecer cuántos arquitectos deben aprobar una propuesta para que los cambios se apliquen.
- **Sugerencias de tipos de datos.** En función del SGBD seleccionado durante la creación de la base de datos, se podrían autocompletar los campos de tipo de datos o permitir la creación de tipos personalizados reutilizables.
- **Plantillas de metadatos para los atributos.** Por ejemplo, al seleccionar un tipo de dato String, se podrían añadir por defecto metapropiedades como longitud mínima y máxima.
- **Relaciones entre entidades.** Ofrecer mecanismos que permitan referenciar otras entidades del mismo diccionario de datos.
- **Perfil de usuario.** Añadir opciones de personalización, como imágenes de perfil y otras configuraciones.

Bibliografía

[1] angular.dev. Components. https://angular.dev/essentials/components. Última consulta: 2025-04-07.

[2] AppMaster. Arquitectura en capas. https://appmaster.io/es/glossary/arquitectura-en-capas. Última consulta: 2025-03-31.

[3] Team Atlan. Data catalog vs. data dictionary: Key differences & benefits. https://atlan.com/data-catalog-vs-data-dictionary/. Última consulta: 2024-03-13.

[4] Team Atlan. Data dictionary: Examples, templates, best practices, and how to make a data dictionary. https://atlan.com/what-is-a-data-dictionary/. Última consulta: 2024-03-13.

[5] Varios autores. El patrón modelo-vista-modelo de vista. https://learn.microsoft.com/es-es/previous-versions/xamarin/xamarin-forms/enterprise-application-patterns/mvvm. Última consulta: 2025-03-31.

[6] Varios autores. ¿qué es git? https://learn.microsoft.com/es-es/devops/develop/git/what-is-git. Última consulta: 2025-04-05.

[7] axarnet. Qué es angular, cómo funciona y para qué sirve. https://axarnet.es/blog/angular. Última consulta: 2025-04-01.

[8] Alejandro Acebes Cabrera. Capas, cebollas y colmenas: arquitecturas en el backend. https://adictosaltrabajo.com/2019/07/02/capas-cebollas-y-colmenas-arquitecturas-en-el-backend/. Última consulta: 2025-03-31.

[9] Dharma Consulting. Dominando las Épicas en la gestión de proyectos ágiles: Un enfoque estructurado para entregar resultados de negocio. https://dharmacon.net/2023/07/14/dominando-las-epicas-en-la-gestion-de-proyectos-agiles-un-enfoque-estructurado-para-entregar-resultados-de-negocio/. Última consulta: 2025-04-19.

[10] creative commons. Attribution 4.0 international deed. https://creativecommons.org/licenses/by/4.0/. Última consulta: 2025-04-07.

[11] Ali Darvish. Client-server architecture. https://darvishdarab.github.io/cs421_f20/docs/readings/client_server/. Última consulta: 2025-03-31.

[12] Dataedo. Data catalog software. `https://dataedo.com/product/data-catalog`. Última consulta: 2024-03-20.

[13] Equipo de Contenidos de GoDaddy. Intellij idea: Ventajas para el desarrollo. `https://www.godaddy.com/resources/es/crearweb/intellij-idea-que-es`. Última consulta: 2025-04-01.

[14] Equipo de Imagina. ¿qué es el patrón de arquitectura (mvvm)? `https://imaginaformacion.com/tutoriales/que-es-el-patron-de-arquitectura-mvvm`. Última consulta: 2025-03-31.

[15] Digital55. Cómo usar testing en angular con jasmine y karma. `https://digital55.com/blog/como-usar-testing-angular-jasmine-karma/`. Última consulta: 2025-04-03.

[16] Claire Drumond. Qué es scrum y cómo empezar. `https://www.atlassian.com/es/agile/scrum`. Última consulta: 2024-02-26.

[17] Equipo editorial de IONOS. ¿qué es mysql? `https://www.ionos.com/es-us/digitalguide/servidores/know-how/que-es-mysql/`. Última consulta: 2025-04-01.

[18] Esteban Canle Fernández. ¿qué es spring boot y para qué sirve? `https://www.tokioschool.com/noticias/spring-boot/`. Última consulta: 2025-04-01.

[19] Claudia Benavides Gallegos. Como crear un plan de mitigación o un plan de contingencia de riesgos. `https://es.linkedin.com/pulse/como-crear-un-plan-de-mitigaci%C3%B3n-o-contingencia-benavides-gallegos`. Última consulta: 2025-04-20.

[20] GitLab. Issue boards. `https://docs.gitlab.com/user/project/issue_board/`. Última consulta: 2025-04-06.

[21] GitLab. Issues. `https://docs.gitlab.com/user/project/issues/`. Última consulta: 2025-04-06.

[22] César Pablo Gutiérrez. *Práctica 1.b Excel básico en gestión de proyectos*. Apuntes de la asignatura Planificación y Gestión de Proyectos, 2023.

[23] Liron Hazan. Angular — how to intercept 401 err response and redirect to login page. `https://lironhazan.medium.com/angular-6-401-authentication-error-handling-888922def566`. Última consulta: 2025-01-08.

[24] Project Management Institute. *Guía de los fundamentos para la dirección de proyectos, (Guía del PMBOK®)*. Project Management Institute, Inc., 2013. 5ª edición. Capítulo 11.

[25] Jasmine. Your first suite. `https://jasmine.github.io/tutorials/your_first_suite`. Última consulta: 2025-04-14.

[26] Jobted. Sueldo del desarrollador web en españa. `https://www.jobted.es/salario/desarrollador-web`. Última consulta: 2024-04-08.

[27] Gaurav Kumar. Make scrolling easier with scroll-to-top and bottom buttons in angular. `https://medium.com/@gauravkrajput/make-scrolling-easier-with-scroll-to-top-and-bottom-buttons-in-angular-a8d0c54ccd43`. Última consulta: 2024-10-24.

[28] Nicolas Lapointe. Clean architecture: Business rules first! https://dev.to/nlapointe/clean-architecture-business-rules-first-4dlo. Última consulta: 2025-03-19.

[29] Federico Lloves. Mapeo de herencia con jpa. https://somospnt.com/blog/250-mapeo-de-herencia-con-jpa. Última consulta: 2024-10-24.

[30] Daniel Medina. Introducción a spring boot: Api rest y spring data jpa. https://danielme.com/2018/02/21/tutorial-spring-boot-web-spring-data-jpa/#jpa. Última consulta: 2025-04-20.

[31] MinnaLearn. ¿qué es la revolución digital? https://courses.minnalearn.com/es/courses/digital-revolution/the-digital-revolution/what-is-the-digital-revolution/. Última consulta: 2025-04-19.

[32] ModelMapper. Getting started. https://modelmapper.org/getting-started/. Última consulta: 2024-10-24.

[33] Dany Paredes. How to use record type in typescript. https://danywalls.com/how-to-use-record-type-in-typescript. Última consulta: 2025-01-08.

[34] PcComponentes. Portátil lenovo ideapad slim 3 15irh8. https://www.pccomponentes.com/portatil-lenovo-ideapad-slim-3-15irh8-intel-core-i5-13420h-16gb-1tb-ssd-156. Última consulta: 2025-03-20.

[35] QAlified. Aprende las diferencias entre pruebas manuales y pruebas automatizadas. https://qalified.com/es/blog/manual-vs-automatizadas-software-pruebas/. Última consulta: 2025-04-14.

[36] Redgate. Sql doc. https://www.red-gate.com/products/sql-doc/. Última consulta: 2024-04-01.

[37] Reltio. What is data stewardship? https://www.reltio.com/glossary/data-governance/what-is-data-stewardship/. Última consulta: 2024-10-24.

[38] Scrum.org. What is scrum? https://www.scrum.org/learning-series/what-is-scrum/. Última consulta: 2024-02-26.

[39] Sigma. What is data modeling? https://www.sigmacomputing.com/resources/learn/what-is-data-modeling. Última consulta: 2024-10-24.

[40] Database Note Taker. Database note taker. https://databasenotetaker.com/. Última consulta: 2024-03-20.

[41] Testim. Is jasmine bdd or tdd? here's what you need to know. https://www.testim.io/blog/is-jasmine-bdd-or-tdd/. Última consulta: 2025-04-14.

[42] Oposiciones TIC. Arquitectura cliente servidor. https://oposicionestic.blogspot.com/2011/06/arquitectura-cliente-servidor.html. Última consulta: 2025-03-31.

[43] Varios. How to use matpaginatorintl? https://stackoverflow.com/questions/46869616/how-to-use-matpaginatorintl. Última consulta: 2025-03-18.

[44] Wikipedia. Economía del conocimiento. `https://es.wikipedia.org/wiki/Econom%C3%ADa_del_conocimiento`. Última consulta: 2025-04-19.

[45] Hayk Yaghubyan. How to test the angular project with jasmine and karma. `https://medium.com/@haykoyaghubyan/how-to-test-the-angular-project-with-jasmine-and-karma-a4241fc8be20`. Última consulta: 2025-04-03.

Apéndice A

Manuales

A.1. Manual de despliegue e instalación

El despliegue aquí descrito está pensado para un dispositivo que funcione con Linux.

A.1.1. Prerrequisitos

Se deben tener instalados los siguientes programas:

- Git
- Java 21
- MySQL 8.0.41
- Maven 3.6.3
- Angular CLI 17.3.5
- Node 20.18.2
- npm 10.8.2
- Nginx 1.18.0

No es obligatorio que se utilicen las mismas versiones, pero sí es recomendable.

A.1.2. Instrucciones

Estos son los pasos a seguir para el despliegue de la aplicación:

1. En **MySQL**, crear el usuario con los siguientes comandos:

```
CREATE USER 'ddvault'@'localhost' IDENTIFIED BY 'ddvaultserv';

GRANT ALL PRIVILEGES ON ddvaultdb.* TO 'ddvault'@'localhost';
```

2. Clonar el repositorio

```
git clone https://gitlab.inf.uva.es/johrami/tfg-johana-ramirez.git
```

3. Situarse en el directorio *app/Spring*
4. Compilar la aplicación (back-end)

```
mvn clean package
```

5. Ejecutar el fichero *.jar*

```
java -jar /home/usuario/tfg-johana-ramirez/app/Spring/target/DDVault-0.0.1-SNAPSHOT.jar
```

6. Abrir otra consola o crear un servicio que ejecute la aplicación de fondo.
7. Situarse en el directorio *app/Angular*
8. Instalar los paquetes necesarios

```
npm install
```

9. Compilar la aplicación (front-end)

```
ng build --configuration production
```

10. Crear un directorio dentro de */var/www* cuyo nombre coincida con el nombre del dominio. En este caso será *virtual.lab.inf.uva.es*.
11. Copiar el contenido de la carpeta *dist/ddvault* dentro de */var/www/virtual.lab.inf.uva.es*.

```
sudo cp -r /home/usuario/tfg-johana-ramirez/app/Angular/dist/ddvault/*
    /var/www/virtual.lab.inf.uva.es/
```

12. Configurar Nginx creando el fichero */etc/nginx/sites-available/ddvault* con el siguiente contenido:

```
server {
    listen 80;
    listen [::]:80;

    server_name virtual.lab.inf.uva.es;

    root /var/www/virtual.lab.inf.uva.es/browser;
    index index.html;

    location / {
            try_files $uri $uri/ /index.html;
    }

    location /api/ {
            proxy_pass http://localhost:8080/;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

13. Crear un enlace del fichero recién creado en */etc/nginx/sites-enabled*

```
sudo ln -s /etc/nginx/sites-available/ddvault /etc/nginx/sites-enabled/
```

14. Reiniciar Nginx

```
sudo systemctl restart nginx
```

A.2. Manual de mantenimiento

La aplicación fue desarrollada en un entorno local con sistema operativo Windows, utilizando el entorno de desarrollo IntelliJ IDEA. Para poder ejecutar y mantener el proyecto, es necesario tener instalados los siguientes programas:

- Git
- Java 21
- Node.js 20.18.2
- MySQL 8.0.41
- Angular CLI 17.3.5

Una vez instaladas estas herramientas, el resto de dependencias necesarias serán gestionadas automáticamente por el IDE.

Para realizar una copia personal del proyecto en GitLab se utiliza el botón ***Fork*** que se encuentra en la página principal del repositorio, tal y como se muestra en la Figura A.1.

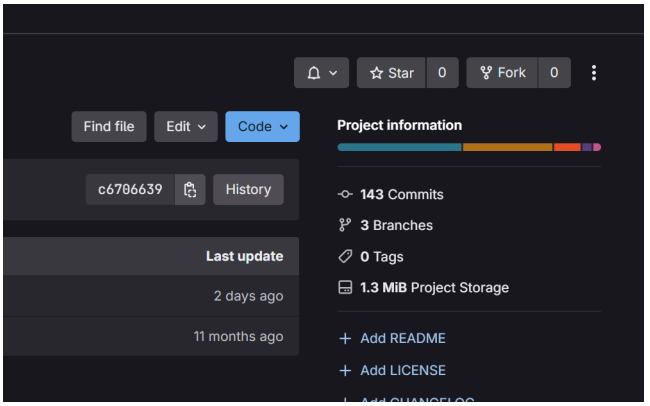


Figura A.1: Localización del botón *Fork*

Pulsando este botón se abre la página mostrada en la Figura A.2. Aquí se configura el nombre y otras opciones del repositorio que se va a crear.

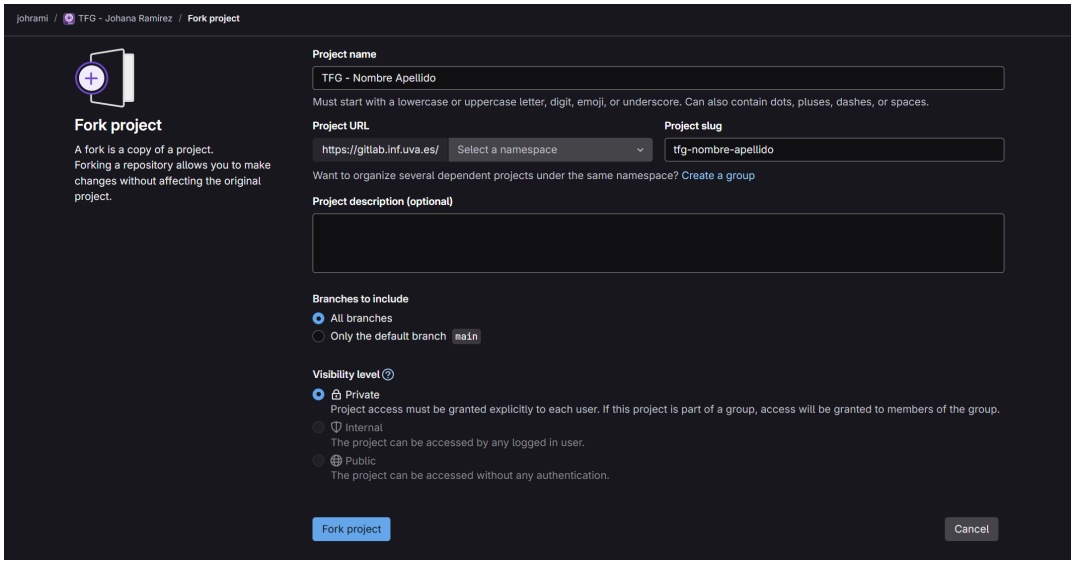


Figura A.2: Configuración del nuevo repositorio

Para clonar el repositorio remoto en uno local se utiliza el comando:

```
git clone https://gitlab.inf.uva.es/usuario/tfg-nombre-apellido.git
```

El proyecto se debe abrir en dos ventanas separadas, una por cada aplicación. Una vez abierto IntelliJ solo es necesario pulsar en "*File* > *Open...*" y buscar las carpetas Angular y Spring.

Para lanzar las pruebas y a la vez generar el reporte de cobertura, se debe ejecutar, en la consola de la aplicación Angular, el comando:

```
ng test --code-coverage
```

A.3. Manual de usuario

Este manual ofrece una guía de uso de la aplicación, describiendo la funcionalidad de cada página y la navegación entre ellas. El objetivo es ayudar a un usuario no familiarizado con la aplicación a entender su funcionamiento.

Inicio de sesión - Figura A.3

Es la primera página a la que se accede, solo pueden estar en ella usuarios sin identificación. A la derecha de la cabecera se encuentra el botón para cambiar de idioma, si se pulsa se abre el desplegable de la Figura A.4 y se deberá seleccionar el idioma deseado. La selección del idioma es accesible desde cualquier página. Para iniciar sesión se debe proporcionar un correo y una contraseña, si el usuario que inicia sesión es administrador se le redirige a la página de inicio del administrador, pero si es un gestor de metadatos se le redirige a la página de diccionarios.

Inicio del administrador - Figura A.5

En esta página, el administrador dispone de un menú con dos opciones. Al seleccionar una de ellas, cambia el contenido mostrado en pantalla. Si se elige "Gestión de usuarios" se muestra la subpágina de gestión de usuarios, mientras que, si se selecciona "Generar usuario" se muestra la subpágina para la creación de usuarios.

Gestión de usuarios - Figura A.6

Aquí se listan todos los usuarios del sistema. Además de mostrar su información, también se pueden realizar acciones sobre cada uno de ellos. Se debe seleccionar un usuario para que se expanda su fila y se muestren los diccionarios a los que tiene acceso el usuario. El botón "Editar información" abre la ventana de diálogo de la Figura A.7 y permite cambiar el nombre y el apellido del usuario. El botón "Restablecer contraseña" abre la ventana de diálogo de la Figura A.8 utilizada para restablecer la contraseña del usuario de manera aleatoria, cuenta con un botón que facilita el copiado de la contraseña en el portapapeles. El botón "Deshabilitar/Habilitar cuenta" cambia el estado

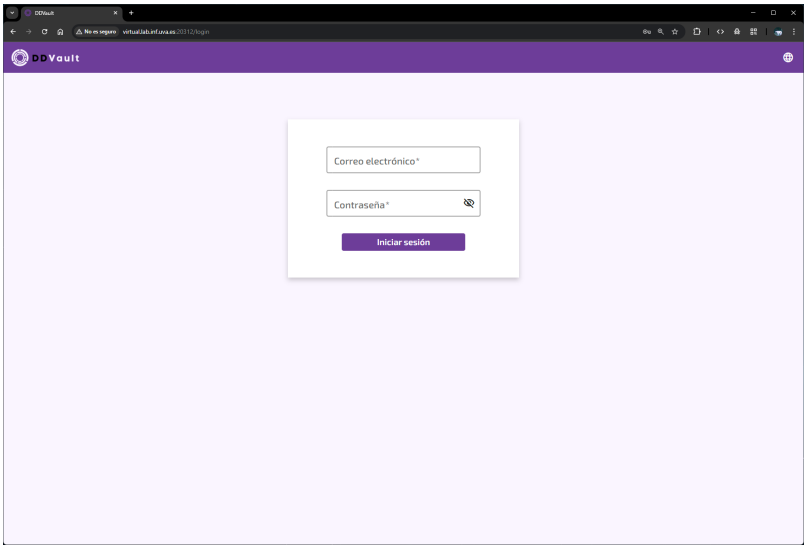


Figura A.3: Página de inicio de sesión

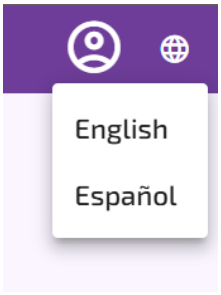


Figura A.4: Desplegable con los idiomas disponibles

de un usuario, si el estado es *Deshabilitado*, el usuario no puede iniciar sesión. Si el usuario tenía acceso a algún diccionario, lo volverá a tener cuando se habilite la cuenta de nuevo. La tabla de diccionarios también cuenta con varias opciones para la configuración de relaciones de los usuarios con los diccionarios. El botón que se encuentra a la derecha del rol, permite cambiar el rol del usuario en el diccionario de esa fila. Más a la derecha está el botón para eliminar cualquier tipo de relación del usuario con el diccionario. Pulsando en "Añadir diccionario", se abre la ventana de diálogo de la Figura A.9. En esta ventana se muestra una tabla con los diccionarios a los cuales el usuario no tiene acceso. Si se pulsa el botón "Añadir" se le otorga al usuario el rol de editor en ese diccionario.

Creación de usuarios - Figura A.10

Los usuarios se crean proporcionando un nombre, un apellido, correo electrónico y seleccionando el tipo de usuario que se desea generar. Pueden ser de dos tipos, administradores y gestores de metadatos.

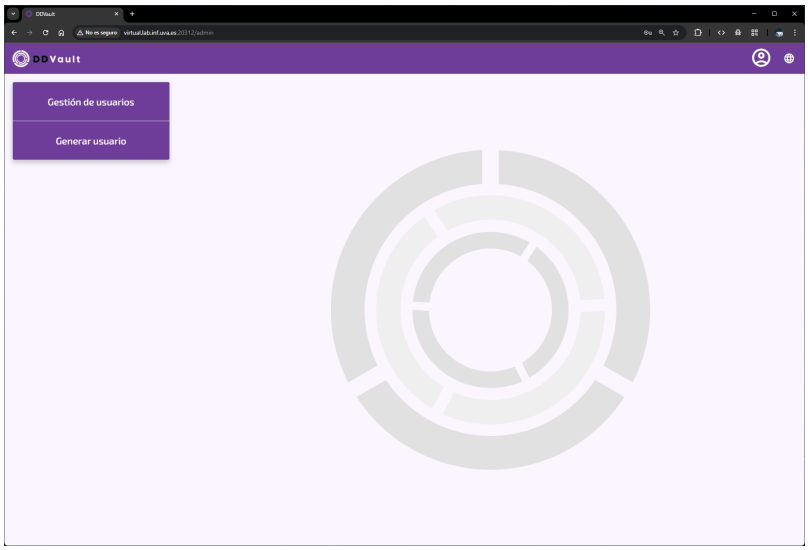


Figura A.5: Página de inicio del administrador

Diccionarios - Figura A.11

Esta es la página de inicio de un gestor de metadatos, en ella se muestran los diccionarios a los que tiene acceso el usuario. Cuando se pulsa en el botón "Nuevo" se redirige a la página de creación de un diccionario. Si se pulsa el botón "Revisión" se redirige a la página de revisiones. Cada tarjeta es un diccionario, pulsando sobre el nombre de una se redirige a la página del diccionario en cuestión. Si se pulsa en el botón con el icono de los tres puntos, se abre un desplegable en el que se puede seleccionar la opción "Eliminar". En el caso en el que se decida borrar un elemento, se abrirá la ventana de diálogo de la Figura A.12, en la que se debe confirmar si realmente se quiere borrar.

Creación de diccionario - Figura A.13

Para crear un diccionario se debe proporcionar un nombre y una descripción del diccionario. Pulsando el botón de "Crear diccionario" se redirige a la página de diccionarios donde se encontrará el diccionario creado.

Diccionario - Figura A.14

En esta página se muestra el nombre del diccionario, su descripción y una lista de bases de datos. El nombre y la descripción son editables, para que aparezca el botón de edición (el botón con el icono de un lápiz) se debe pasar el ratón por encima de la descripción. Aunque este solo aparecerá cuando el usuario tenga el rol de arquitecto en ese diccionario. Cuando se pulse el botón de edición la página cambiará y se mostrará como la página de la Figura A.15. Los cambios se podrán guardar o cancelar. Pulsar el botón "Nuevo" hará que se abra la página de creación de una base de datos. Cada tarjeta representa una base datos, pulsando en su nombre, se abrirá la página de base de datos correspondiente.

Creación de una base de datos - Figura A.16

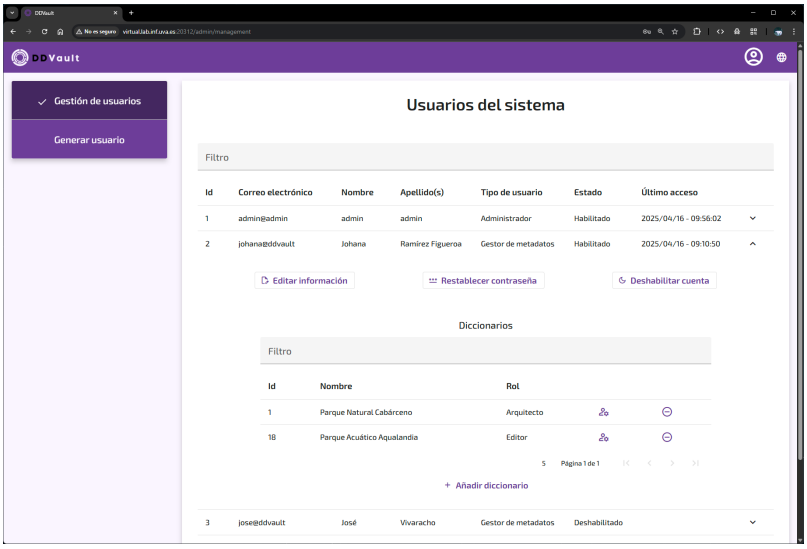


Figura A.6: Página para la gestión de usuarios del sistema

Figura A.7: Ventana de diálogo para modificar la información de un usuario

Figura A.8: Ventana de diálogo para restablecer la contraseña

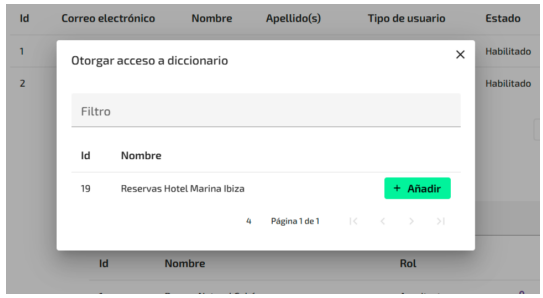


Figura A.9: Ventana de diálogo para otorgar acceso a los diccionarios

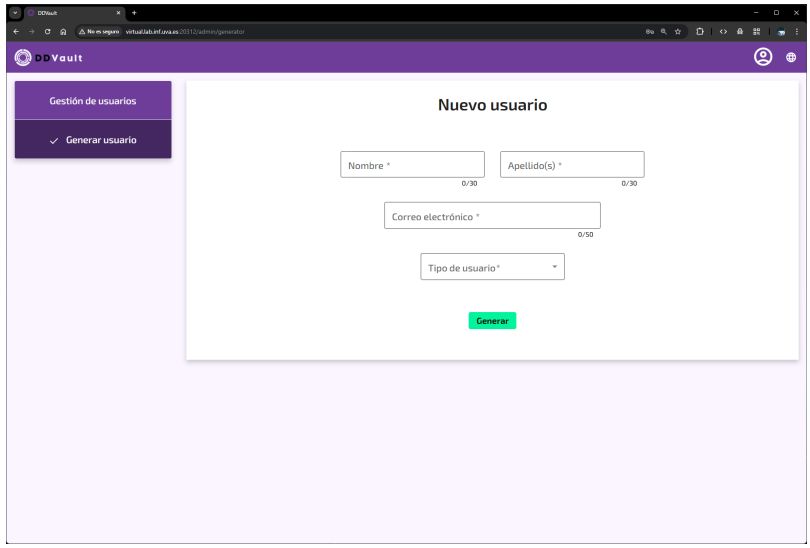


Figura A.10: Página para la creación de usuarios

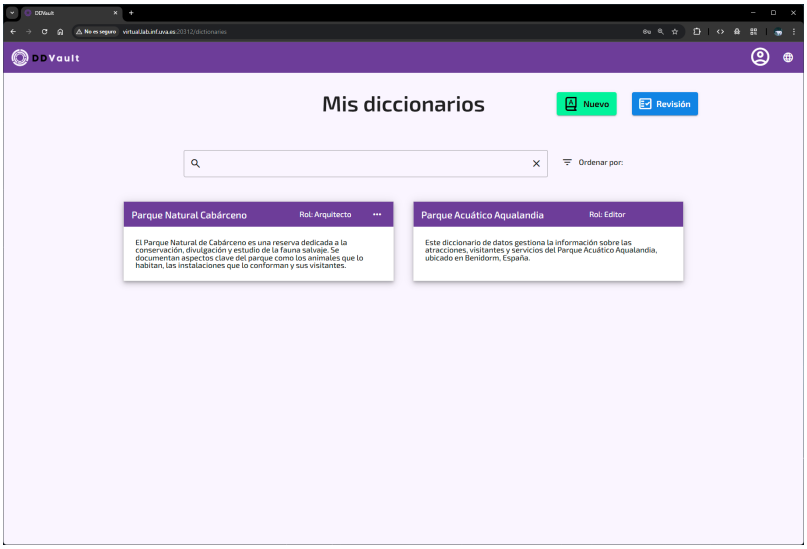


Figura A.11: Página con los diccionarios del gestor de metadatos

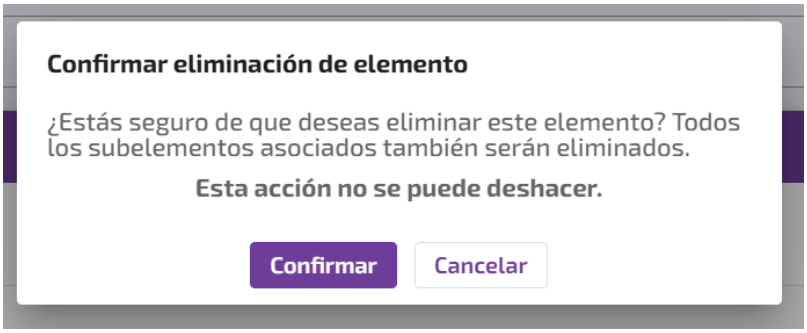


Figura A.12: Ventana de diálogo para confirmar la eliminación de un elemento

Para crear una base de datos, se debe especificar un nombre, una descripción y seleccionar un SGBD. Cuando se pulse el botón “Crear base de datos” se redirigirá a la página del diccionario que la contiene.

Base de datos - Figura A.17

En la página de una base de datos, se muestra su información, nombre y una lista de las entidades que forman la base de datos. Se puede acceder a la página de una entidad pulsando en su nombre. También es posible crear una entidad pulsando en el botón “Nueva”, esto redirige a la página de creación de una entidad.

Creación de una entidad - Figura A.18

Una entidad se crea estableciendo solo un nombre y una descripción, cuando se pulse el botón “Crear entidad” se volverá a la página de la base de datos a la que pertenece la entidad creada.

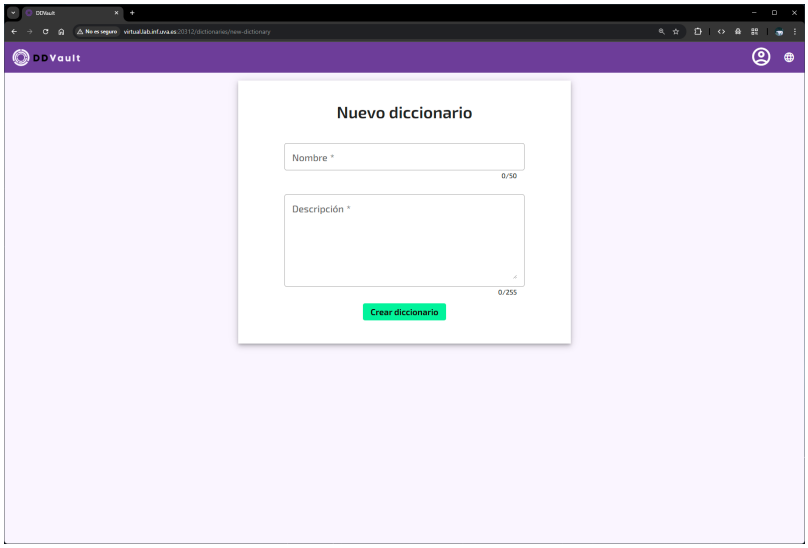


Figura A.13: Página para la creación de un diccionario

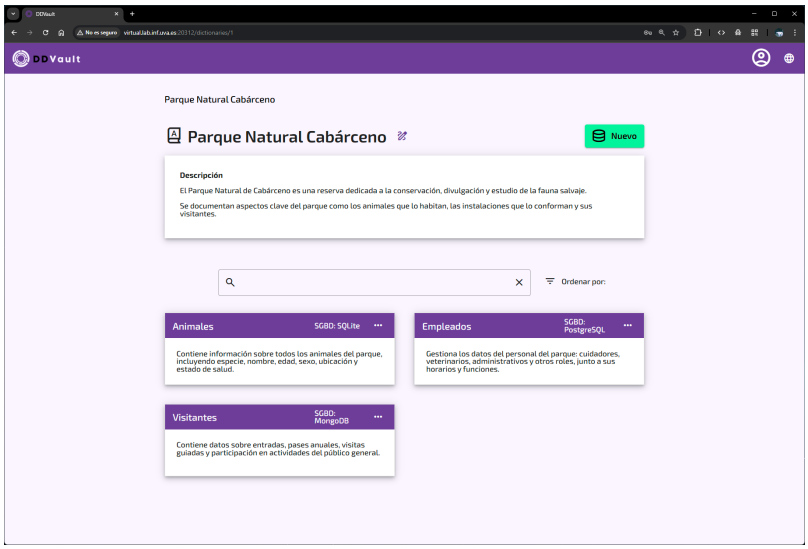


Figura A.14: Página de un diccionario de datos

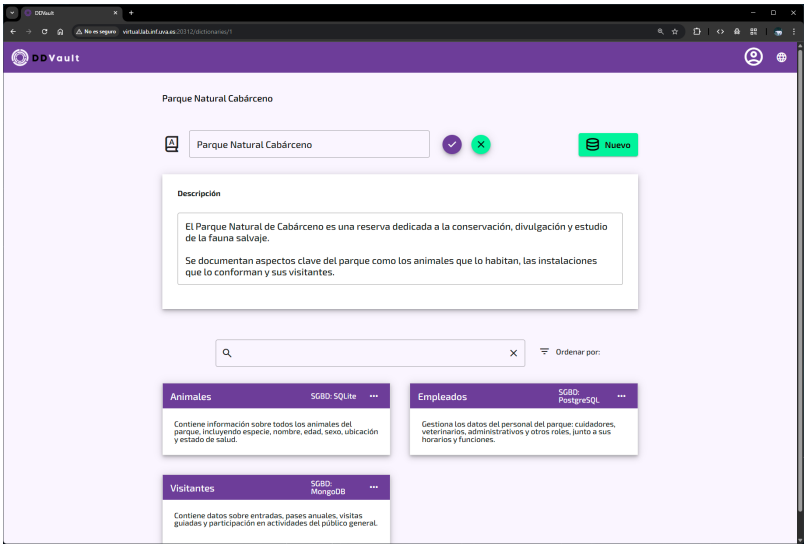


Figura A.15: Edición del nombre y la descripción de un diccionario

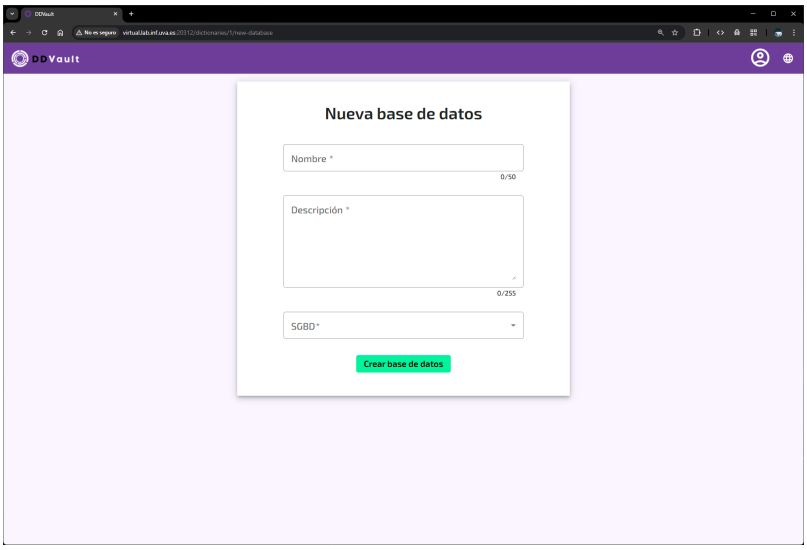


Figura A.16: Página para la creación de una base de datos

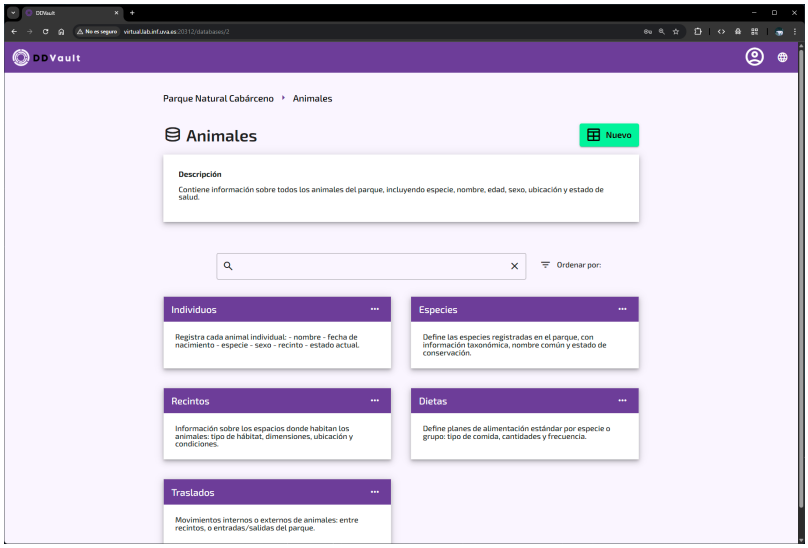


Figura A.17: Página de una base de datos

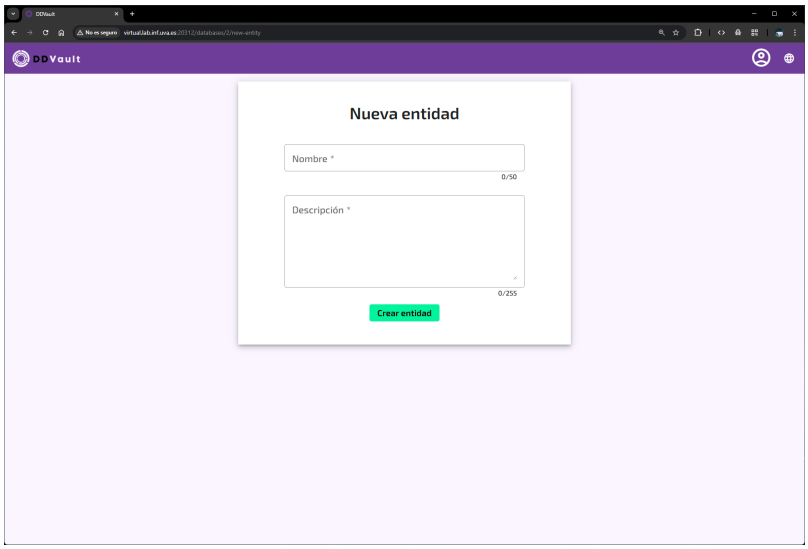


Figura A.18: Página para la creación de una entidad

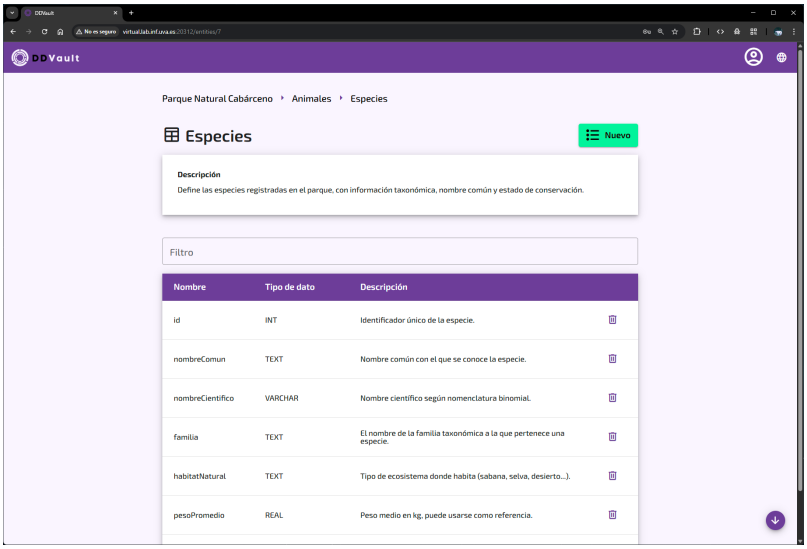


Figura A.19: Página de una entidad

Entidad - Figura A.19

Cada entidad está formada por un nombre, una descripción y un conjunto de atributos. Pulsando en cualquiera de las filas de la tabla se redirigirá a la página del correspondiente atributo. En caso de pulsar la papelera, se abrirá la ventana de diálogo de la Figura A.12. También se puede acceder a la página de creación de un atributo pulsando el botón “Nuevo”.

Creación de un atributo - Figura A.3

Para crear un atributo se debe proporcionar un nombre, una descripción y un tipo de dato. Pulsando en el botón “Crear atributo” se generará el nuevo atributo con la información suministrada y se redirigirá a la página de entidad, donde se encuentra la lista de atributos, incluyendo el nuevo.

Atributo - Figura A.21

En la página de un atributo se muestra su nombre, descripción, un tipo de dato y un conjunto de metapropiedades. Pulsando el botón “Editar” se activa la edición de el tipo de dato y las metapropiedades, dejando la vista como se muestra en la Figura A.22. En este modo se pueden modificar las metapropiesdades, añadir nuevas o eliminar las existentes. Pulsar el botón de borrado de una metapropiedad hará que se abra la ventana de diálogo de la Figura A.23. Para añadir metapropiedades se debe pulsar el botón “Añadir metapropiedad”. Las metapropiedades se pueden arrastrar para cambiarlas de posición, esto se consigue manteniendo pulsado los botones de la izquierda y arrastrando el atributo a la posición deseada. Si el botón de editar, se encuentra deshabilitado, significa que no es posible editar el atributo porque está a la espera de una revisión.

Revisiones - Figura A.24

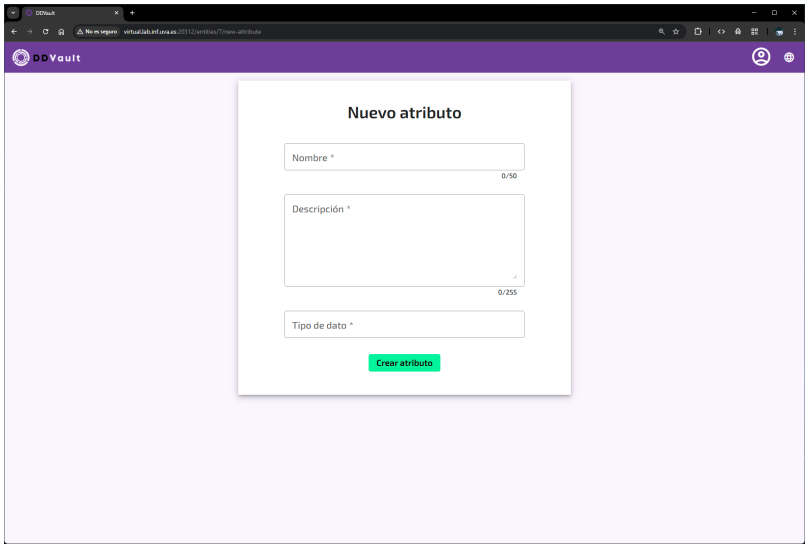


Figura A.20: Página para la creación de un atributo

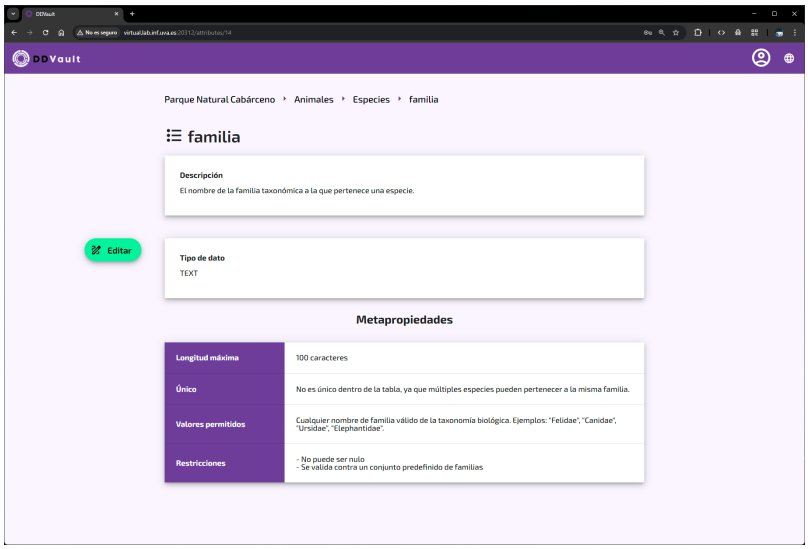


Figura A.21: Página de un atributo

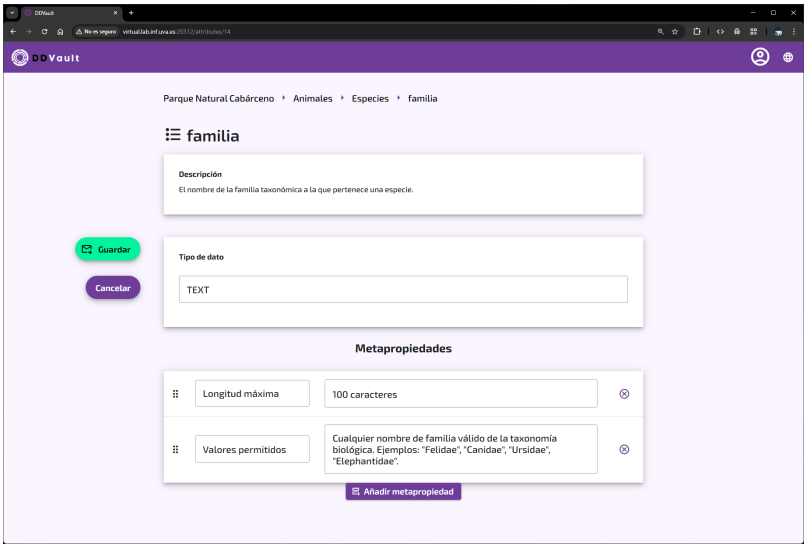


Figura A.22: Edición de las metapropiedades de un atributo

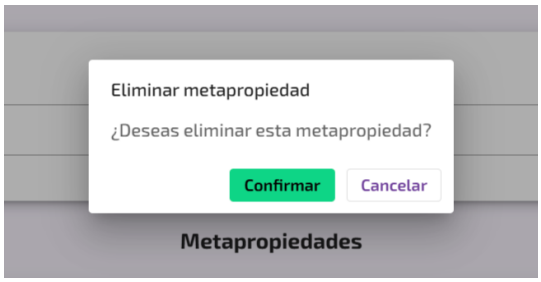


Figura A.23: Ventana de diálogo para confirmar el borrado de una metapropiedad

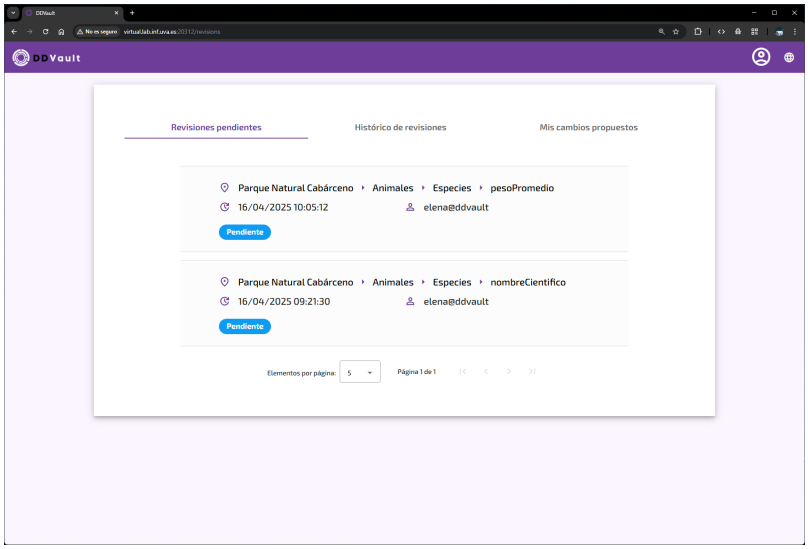


Figura A.24: Página de revisiones en la pestaña de "Revisiones pendientes"

En esta página las revisiones se dividen en tres pestañas. En la pestaña "Revisiones pendientes" (Figura A.24) se encuentran las propuestas realizadas por editores de un diccionario en el que el usuario es un arquitecto. En la pestaña "Histórico de revisiones", (Figura A.25) están todas las propuestas que el usuario ha aceptado o rechazado. En la pestaña "Mis cambios propuestos" (Figura A.26) se encuentran todas las propuestas hechas por el usuario, en cualquiera de los estados. La primera fila de cada revisión es la ubicación del atributo que se ha modificado. La ubicación no es más que el nombre de los elementos de la jerarquía en la que se encuentra el atributo, pulsando sobre el nombre de cualquiera de los elementos, se abrirá en otra pestaña, la información específica del elemento. Cuando se pulsa sobre una revisión, se abre en una nueva pestaña la página con la información detallada de la revisión.

Revisión - Figura A.27

La interfaz de la página de la revisión varía según el estado de la propuesta. En específico, cuando la propuesta está pendiente (Figura A.27), en la parte inferior de la página, aparecerán los botones para aceptar o rechazar los cambios propuestos. Cuando está aceptada (Figura A.28) o rechazada (Figura A.29), se mostrará el revisor. Adicionalmente, en las revisiones rechazadas se mostrará también el comentario del revisor, con los motivos del rechazo. Con los botones que se encuentran justo encima del estado, se puede consultar la información de los elementos del atributo.

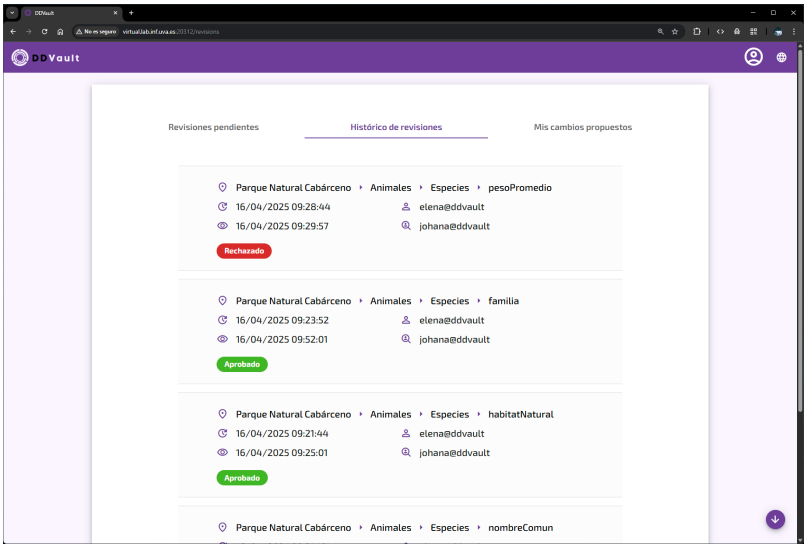


Figura A.25: Página de revisiones en la pestaña de "Histórico de revisiones"

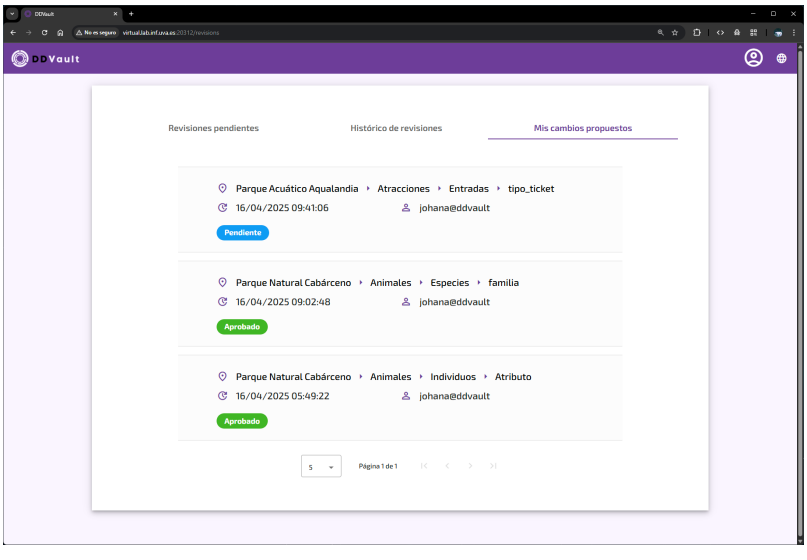


Figura A.26: Página de revisiones en la pestaña de "Mis cambios propuestos"

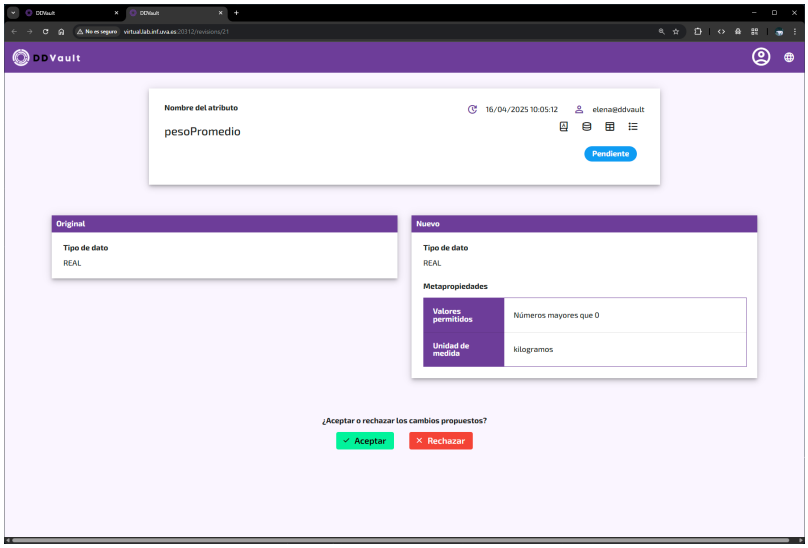


Figura A.27: Página de una revisión pendiente

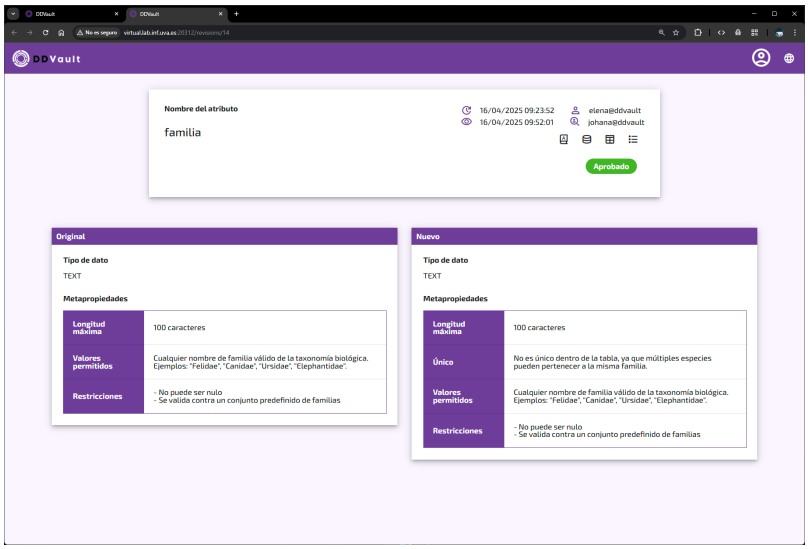


Figura A.28: Página de una revisión aceptada

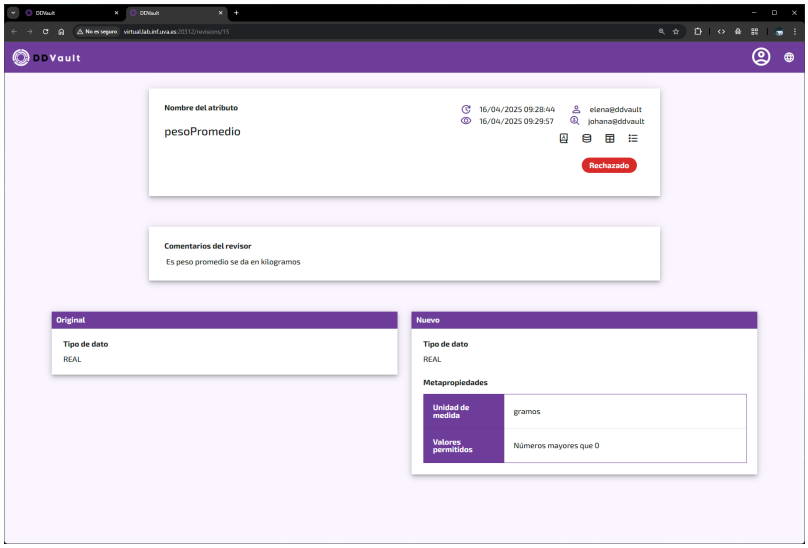


Figura A.29: Página de una revisión rechazada

Apéndice B

Resumen de enlaces adicionales

Los enlaces útiles de interés en este Trabajo Fin de Grado son:

- Repositorio del código:
  `https://gitlab.inf.uva.es/johrami/tfg-johana-ramirez`.
- Aplicación desplegada en una máquina virtual proporcionada por la Escuela:
  `http://www.virtual.lab.inf.uva.es:20312/login`.
  Este despliegue dejará de estar disponible una vez finalizado el presente curso académico cuando se liberen los recursos.