Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DO SOFTWARE

Deseño e Implementación dunha Aplicación Utilizando Tecnoloxías Java EE

Estudante: Álvaro Martinez Cotelo
Dirección: Paula Montoto Castelao

A Coruña, setembro de 2019.

Por mi y por todos mis compañeros.

Agradecimientos

A Mamá, a Sabe y a Antón.

Resumen

En colaboración con los integrantes del equipo de investigación ”SURHI”, surge la necesidad de desarrollar una aplicación web que tiene como principal objetivo la visualización de revistas (en formato PDF) e información relativa a estas (metadatos, estudio, índices..), y, su administración mediante una herramienta de gestión de contenidos. Para ello, se creará un espacio web, accesible por cualquier usuario, en el que se muestren las revistas y la información relativa a ellas y un gestor, accesible mediante autenticación, encargado de la administración de estos contenidos.

Esta propuesta será llevada a cabo utilizando tecnologías software como Spring Boot y Thymeleaf, además de otras herramientas para la gestión del desarrollo (como GitLab).

Como metodología de trabajo seguiremos Scrum adaptado a nuestra situación.

Abstract

In collaboration with the members of the research team ”SURHI”, the need arises to develop a web application that has as its main objective the visualization of magazines (in PDF format) and information related to them (metadata, study, indexes ..), and its administration through a content management tool. To do this, a web space will be created, accessible by any user, in which the magazines and the information related to them are displayed and a manager, accessible through authentication, responsible for the administration of these contents.

This proposal will be carried out using software technologies such as Spring Boot and Thymeleaf, as well as other tools for development management (such as GitLab).

As a working methodology we will use Scrum adapted to our situation.

Palabras clave:

- Aplicación Web
- Apache POI
- Bootstrap
- Hibernate
- Java
- JPA
- Maven
- MySQL
- Scrum
- Spring Boot
- Spring Security
- Thymeleaf
- Revista
- Número de revista
- Estudio
- Índice de autores
- Índice de ilustraciones
- Índice onomástico

Keywords:

- Web application
- Apache POI
- Bootstrap
- Hibernate
- Java
- JPA
- Maven
- MySQL
- Scrum
- Spring Boot
- Spring Security
- Thymeleaf
- Journal
- Journal number
- Study
- Authors index
- Illustration index
- Onomastic Index

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este primer capítulo de la memoria se explica la motivación y los objetivos de la realización de nuestra aplicación web.

1.1 Motivación

Internet es un importante medio de comunicación, por ello han surgido aplicaciones Web como intermediario para propagar información, así como para ofrecer servicios a los usuarios.

Uno de estos primeros medios de comunicación son las revistas, publicaciones de edición periódica impresas en papel, que ofrecen un tratamiento exhaustivo de los sucesos o temas que desarrollan, de interés general para el público de estas.

A menudo, ediciones antiguas de distintas revistas se han perdido con el paso del tiempo o no son accesibles fácilmente por los usuarios interesados.

Los integrantes del equipo de investigación ”SURHI” (ver sección 1.2), se encuentran actualmente en el desarrollo de una investigación dedicada a la recolección y análisis de revistas antiguas del surrealismo hispanoamericano. Surge así la idea de crear una aplicación web que permita visualizar y gestionar estos contenidos.

Para ponernos en situación, los contenidos de los que disponen actualmente para cada revista son los siguientes:

Los **números**, que son una entrega o fascículo de una revista, los cuales están disponibles en formato PDF. El número de estos es variable según cada revista.

Un **estudio**, que es una contextualización y análisis literal de la revista completa, el cual está disponible en formato Word.

Los **índices de autores**, un único documento en formato Excel, en el que cada fila contiene información sobre los autores y traductores que aparecen en la revista, especificando a que numero pertenecen concretamente.

Los **índices de ilustraciones**, un único documento en formato Excel, en el que cada

fila contiene información sobre las imágenes que aparecen en la revista, especificando a que numero pertenecen concretamente.

El **índice onomástico**, un único documento en formato Excel, donde aparece el nombre de personas mencionadas en el texto, por orden alfabético con el número de las páginas donde aparecen. Este índice, al contrario que los de autores e ilustraciones no especifica en que numero, es de índole general.

Debido a esto, para adaptarnos a su metodología de trabajo nuestro gestor deberá disponer de un mecanismo de subida de archivos para los diferentes documentos citados anteriormente. Estos serán detallados más a fondo en el análisis (véase cap. 4, sec. 4.1).

1.2 SURHI

SURHI es el acrónimo del proyecto de investigación sobre surrealismo hispanoamericano titulado ”Caracterización del surrealismo hispánico. Recuperación, digitalización y edición de las revistas surrealistas de México, Perú y República Dominicana”.

Participan en este proyecto financiado por el Ministerio de Ciencia, Innovación y Universidades del Gobierno de España, las siguientes instituciones: Universidad de A Coruña, Universidad Autónoma de Madrid, Universidad Autónoma de México, Universidad Pontificia Católica de Perú y la Universidad Federal de Santa Catarina.

El equipo de investigación está formado por los siguientes académicos: Eva Valcárcel, Fernando Agrasar, Eduardo Becerra, Rita Eder, Luis Fernando Chueca y Raúl Antelo.

Eva Valcárcel, la investigadora principal del proyecto, y que forma parte del grupo de investigación ”HISPANIA” (Grupo de Investigación en Lengua, Literatura y Cultura Hispánica)” ha sido la que nos ha propuesto la colaboración en el proyecto mediante el desarrollo de la aplicación web.

1.3 Estado del arte

En la actualidad existen páginas web de revistas físicas que ofrecen parte del contenido, o avances de estas a los lectores. A continuación, mostramos algunos ejemplos:

- **La Guirnalda Polar** (fig. 1.1)

La Guirnalda Polar es una revista electrónica de cultura latinoamericana en Canadá. El espacio web está dedicado a una única revista, de la cual se muestran todos los números disponibles sin ningún tipo de clasificación u organización. El contenido de estos números está dividido en capítulos, accesibles mediante un enlace a un documento en formato PDF. Esta revista no dispone de estudios o análisis de contenido de estas.

Figura 1.1: Web La Guirnalda Polar (https://lgpolar.com/)

- **El surrealismo y sus derivas** (fig. 1.2)

Este espacio web está dedicado a la visualización de revistas antiguas del surrealismo hispanoamericano. Estas revistas están clasificadas por país de origen. El contenido de estas está dividido en números o entregas, accesibles mediante un enlace a una versión PDF. Cada revista dispone de un estudio, un índice de autores, un índice de ilustraciones y un índice onomástico. La web ha sido desarrollada estáticamente, por lo que no permite la gestión de contenidos.

Esta página ha sido la fuente de inspiración por parte del cliente para la realización de nuestro proyecto.

Figura 1.2: Web El surrealismo y sus derivas (https://www.uam.es/proyectosinv/surreal)

1.4 Objetivos

Los principales objetivos a cumplir con el desarrollo de la aplicación se dividen en dos grupos:

El primero de ellos se centra desarrollar un **espacio web** al que cualquier usuario de la red pueda acceder y visualizar todas las revistas y sus respectivos números, así como los datos de estudio de estas (Estudio de la revista, índices de autores, índice onomástico e índices de ilustraciones).

Para ello, se dispondrá de una página principal en la que encontraremos una introducción general a los datos mostrados, un espacio de información en el que se explicará la forma de lectura e interpretación de los distintos índices y el estudio, y un listado con todas las revistas disponibles y un enlace a cada una de ellas.

De cada una de estas revistas se quiere mostrar una página con información relevante. Esta información es la siguiente:

- La imagen de portada junto con el título de la revista y el país en el que fue editada.
- Un titular y un texto introductorio y explicativo.
- Todos los números de cada revista, los cuales podrán ser visualizados en formato PDF mediante un enlace.
- Un estudio de la revista y sus respectivos índices de autores, onomástico y de ilustraciones.

En segundo lugar, para que mostrar todos estos datos sea posible se desarrollará un **gestor** en el cual administraremos todos los contenidos. Esta página solo podrá ser accedida por un administrador mediante autenticación.

Este gestor permitirá crear una revista con sus datos identificativos, como el título y el país de origen, añadir números de esta en formato PDF, un estudio en formato Word y tres archivos Excel los cuales contendrán la información de los índices de autores, de ilustraciones y onomástico respectivamente. La información que contienen estos archivos Excel y Word será tratada e implementada en una base de datos.

Una vez que estas revistas hayan sido creadas, se mostrarán en el gestor y se podrán realizar distintas operaciones con ellas, como la visualización de sus respectivos datos, su modificación, o su borrado. Ocurrirá lo mismo con sus respectivos números, estudio e índices.

1.5 Estructura de la memoria

La estructura de la memoria está dividida en nueve capítulos en los que se explica el desarrollo del TFG.

El primer capítulo, el actual, es una **introducción** (ver cap. 1) en la que se explica la motivación y el contexto para realizar este proyecto, un estado del arte de aplicaciones similares y los objetivos a cumplir por nuestra aplicación web.

El segundo y tercer capitulo, **tecnologías y herramientas utilizadas** (ver cap. 2) y **metodología para el desarrollo** (ver cap. 3), introducen aspectos teóricos y su adaptación a nuestro proyecto.

Los siguientes cinco capítulos se centran en la planificación, desarrollo de la aplicación y pruebas de esta. En el primero de ellos, el **análisis** (ver cap. 4), se identifican los actores y se enumeran y explican los requisitos necesarios para llevar a cabo la **planificación** (ver cap. 5) de nuestro proyecto. Después de esto, en el tercer capítulo, se muestra el **diseño** (ver cap. 6) de la aplicación. A continuación, en los siguientes dos capítulos se explica cómo se llevó a cabo la **implementación** (ver cap. 7) del proyecto y las **pruebas** (ver cap. 8) realizadas para la validación de su correcto funcionamiento.

Por último, se desarrolla un apartado dedicado a las **conclusiónes** (ver cap. 9) en el que se expone la situación final del proyecto, las lecciones aprendidas, la relación con las competencias de la titulación en general y la mención en particular y posibles líneas futuras.

Capítulo 2

Tecnologías y herramientas utilizadas

En este capítulo se explican las tecnologías y las herramientas utilizadas durante el proceso de desarrollo.

2.1 Lenguajes utilizados

2.1.1 Java

Java [1] es un lenguaje de programación de propósito general. Se trata de un lenguaje basado en clases y orientado a objetos, compatible con Spring Boot para el desarrollo de aplicaciones web. Es el lenguaje principal elegido para el desarrollo de la lógica de negocio y los controladores de la aplicación.

2.1.2 Javascript

Lenguaje de programación interpretada que se ejecuta en el navegador. Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas [2].

2.1.3 HTML

HTML (Hypertext Markup Language) [3], se trata de un sistema estandarizado para marcar la estructura de páginas web, lo que permite modificar su apariencia y funcionalidades con ayuda de CSS y JavaScript. Es el lenguaje utilizado para los componentes de la vista de la aplicación.

2.1.4 CSS

CSS (Cascading Style Sheets) [4], es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en HTML, como el caso de nuestra aplicación.

2.1.5 SQL

SQL (Structured Query Language) [5] es un lenguaje de dominio específico utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales. Se utilizó para la visualización, adición, modificación y borrado de datos en la aplicación.

2.2 Frameworks y librerías

2.2.1 Spring Boot

[6] Es una tecnología open source que puede ser utilizada como punto de inicio de cualquier tipo de proyecto basado en el framework de Spring. Está diseñada para su puesta en marcha inmediata, con la mínima cantidad de configuración XML. Spring es un framework para el desarrollo de aplicaciones JavaEE. Una de las mayores ventajas que tiene este framework es la inyección de dependencias, de esta forma permite instanciar, inicializar y conectar objetos con la aplicación. Estos objetos serán conocidos como beans. Para utilizar esto beans es necesaria su configuración, que en el caso de Spring es mediante configuración XML o con anotaciones y objetos Java.

Esta tecnología dispone de una herramienta, Spring Initializer, que crea una aplicación de Spring Framework y otras librerías externas preconfiguradas de forma sencilla, la cual ha sido usada para la generación de este proyecto.

2.2.2 Bootstrap

Bootstrap [7] es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. Se ha utilizado en gran parte para la vista de la gestión de contenido.

2.2.3 Thymeleaf

Thymeleaf [8] es una biblioteca Java que implementa un motor de plantillas de XML/XHTML/HTML5. Se acopla muy bien para trabajar en la capa vista del MVC de aplicaciones web. Proporciona un módulo opcional para la integración con Spring MVC. El objetivo principal de Thymeleaf es permitir la creación de plantillas de una manera elegante y un código bien formateado. Sus dialectos Standard y SpringStandard permiten crear potentes plantillas naturales que se pueden visualizar correctamente en los navegadores de Internet.

2.2.4 Apache POI

Apache POI [9] proporciona bibliotecas Java puras para leer y escribir archivos en formatos de Microsoft Office, como Word, PowerPoint y Excel.

2.3 Herramientas para el desarrollo

2.3.1 Eclipse

Eclipse [10] es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (IDE), como el IDE de Java llamado Java Development Toolkit (JDT).

2.3.2 Maven

Maven [11] es una herramienta de gestión y construcción de proyectos basado en Java. Permite al desarrollador comprender el estado del desarrollo en el menor tiempo posible. Para ello se centra en ciertas áreas, como facilitar el proceso de construcción, ofrecer un proceso de construcción uniforme u ofrecer guías para el desarrollo de buenas prácticas.

2.3.3 Git

Git [12] es una herramienta para el control de versiones. Esta herramienta lleva el registro de todos los cambios realizados sobre el código, para coordinar mejor el trabajo realizado por los equipos de desarrollo.

2.3.4 GitLab

Gitlab [13] es un servicio web de control de versiones y desarrollo de software colaborativo basado en Git. Además de gestor de repositorios, el servicio ofrece también alojamiento de

wikis y un sistema de seguimiento de errores, todo ello publicado bajo una Licencia de código abierto.

2.4 Sistema de gestión de bases de datos

2.4.1 MySQL

MySQL [14] es un sistema de gestión de bases de datos relacional open source, y está considerada como la base datos de código abierto más popular del mundo para entornos de desarrollo web, en gran parte a su rendimiento, fiabilidad y facilidad de uso.

2.4.2 H2

H2 [15] es un sistema administrador de bases de datos relacionales programado en Java. Puede ser incorporado en aplicaciones Java o ejecutarse de modo cliente-servidor.

Capítulo 3

Metodología para el desarrollo

En este capítulo, se introduce la metodología utilizada durante el desarrollo de la aplicación.

3.1 Scrum

Scrum [16] es un proceso de desarrollo ágil. Permite un desarrollo rápido y por iteraciones, lo que nos permite inspeccionar el software creado y repriorizar el trabajo, en lugar de la planificación y ejecución completa del producto. Es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo y obtener el mejor resultado posible. Los equipos se autoorganizan para determinar que funcionalidades deben cubrir y cuáles son las de mayor prioridad de manera que al final de cada iteración sea posible realizar una demostración del producto desarrollado hasta el momento. A continuación, detallaremos como se lleva esto a cabo.

3.1.1 Roles

Esta metodología consta de los siguientes roles (fig. 3.1):

- **Product Owner**: Es el encargado de fijar las metas del sprint y de la priorización del Product Backlog. Intenta que el producto desarrollado por el equipo maximice su valor.

- **Scrum Master**: Es el responsable de promover las prácticas Scrum. Es el mediador entre el Equipo de desarrollo y el Product Owner. Su función debe estar divida a partes iguales entre estas tareas y desarrollo del producto. Dirige los Daily Scrum y las Sprint Review.

- **Equipo de desarrollo**: Se autoorganizan para desarrollar el proyecto siguiendo el Sprint Backlog.

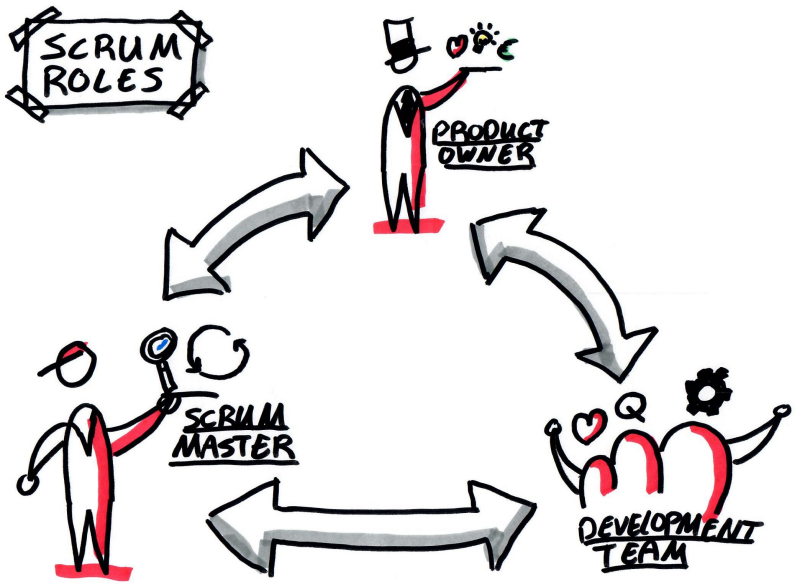

Figura 3.1: Roles de Scrum

3.1.2 Eventos o reuniones Scrum

- **Daily Scrum**: Reunión diaria del equipo de desarrollo, organizada por el Scrum Master, en la que cada integrante responde a 3 preguntas (”¿Qué hiciste ayer?”, ”¿Qué vas a hacer hoy?” y ”¿Que problemas encontraste?”) para ver el estado actual del proyecto. La duración aproximada es de 15 minutos, en el mismo lugar y hora. Puede asistir gente que no pertenece al equipo, pero no pueden participar.

El proyecto se divide en sprints, los cuales se irán completando hasta realizar todas las tareas del Product Backlog.

- **Sprint Planning**: Durante esta reunión se seleccionan las tareas a realizar durante el sprint y una estimación de su duración, de este modo se obtiene el Sprint Backlog. Esta planificación es realizada por el equipo de desarrollo y el Scrum Master.

- **Sprint Review**: Esta reunión se realiza al final de cada sprint y tiene como objetivo presentar los trabajos realizados en este. Se pueden hacer adaptaciones en el Product Backlog si fuese necesario.

- **Sprint Retrospective**: Esta reunión se lleva a cabo al final de cada sprint y tiene como finalidad analizar lo que se ha realizado, que funcionalidades son operativas y cuales no y que se podría mejorar.

Además, el Sprint Review y el Sprint Retrospective sirven como punto de partida para la planificación del siguiente Sprint Planning.

3.1.3 Artefactos Scrum

- **Product Backlog** - Es la lista que recoge todo lo que necesita el producto, ordenado por prioridades. Esta lista es dinámica, ya que está continuamente evolucionando con la consecución de cada sprint. Se realiza gracias a la obtención de requisitos mediante la cooperación entre Product Owner y el Equipo de desarrollo.
- **Sprint Backlog** - Agrupa en tareas los elementos del Product Backlog seleccionados para el sprint actual. Los miembros del equipo seleccionan que tareas realizara cada uno.

3.2 Scrum aplicado al proyecto

Debido a las circunstancias de este proyecto, se modificaron algunos aspectos de la metodología para adaptarlos a nuestro caso.

Al realizarse de forma individual, y no disponer de un equipo, los roles de Scrum Master y del Equipo de desarrollo han sido llevados a cabo por el alumno, mientras que el de Product Owner, por la investigadora del grupo ”HISPANIA” de la facultad de Filología de la Coruña, Eva Valcárcel López.

Cada Sprint Planning ha tenido una duración variable de 2 a 3 semanas.

En nuestro caso, las Daily Scrum han sido sustituidas por un repaso individual de las tareas realizadas y el Sprint Planning realizado de manera individual.

Para la realización del Product Backlog se mantuvieron una serie de reuniones iniciales para concretar los aspectos fundamentales de la aplicación. Este fue modificado a medida que se realizaban demostraciones de versiones funcionales del producto.

Los Sprint Backlog fueron seleccionados por el alumno en función de prioridades establecidas inicialmente en consenso con el Product Owner. Además, se trató de seguir un orden lógico de implementación de funcionalidades, de las más básicas y que darían forma al proyecto, hasta otras más específicas y que definían la aplicación.

Para el Sprint Review y el Sprint Retrospective se han mantenido reuniones con el Product Owner. Algunas de estas reuniones de finalización de sprint se han realizado de manera individual por el alumno debido a la dificultad para reunirse.

Capítulo 4

Análisis

En colaboración con los integrantes del proyecto ”SURHI” (ver cap. 1 sec. 1.2), surge la necesidad de crear un espacio web dedicado a la muestra de revistas antiguas del surrealismo hispanoamericano y sus respectivos datos de estudio. Los contenidos de estas revistas son difíciles de encontrar y no existen ediciones digitales de estas. Debido a esto se decide crear un espacio web para su visualización y un gestor accesible por un administrador que le permita manipular estas revistas y sus respectivos datos.

En este capítulo, se muestran los datos referentes a las revistas y el análisis de requisitos.

4.1 Datos de las revistas

En las primeras reuniones con el Product Owner (ver cap. 3, sec. 3.2), se nos muestra información relevante a las revistas, que se detalla a continuación:

- Cada **revista** tiene los siguientes datos identificativos y descriptivos:
  - Título de la revista.
  - País en el que fue editada.
  - Un bloque con un título y texto de introducción a esta.
- Cada revista está formada por **números** o entregas, cuyo número es variables según cada revista. Estos están formados por:
  - Numero al que pertenecen.
  - Documento en formato PDF que contiene la edición física del número.
  - Una imagen de portada.

- Cada revista dispone de un **estudio**, que es una contextualización y análisis literal de la revista completa, el cual está disponible en formato Word.

- Cada revista dispone de un **índice de autores**, un único documento en formato Excel, en el que cada fila contiene la siguiente información:

  - Columna A: Número de revista al que pertenece.
  - Columna B: Título del fragmento de texto que han publicado los autores que van desde la columna C hasta la H, y que han traducido los de las columnas I y J.
  - Columna C, D, E, F, G y H: Autores.
  - Columna I, J: Traductores.
  - Columna K: Primera página donde aparece el título de la columna B.
  - Columna L: Notas.

- Cada revista dispone de un **índice de ilustraciones**, un único documento en formato Excel, en el que cada fila contiene la siguiente información:

  - Columna A: Número de revista al que pertenece.
  - Columna B: Título de la imagen o grupo de imágenes a las que hace referencia y ha publicado el autor de la columna C.
  - Columna C: Autor.
  - Columna D: Página o páginas donde aparecen las imágenes de la columna B. En caso de especificarse más de una página, estas serán separadas mediante el símbolo ";". También se puede especificar un rango separando dos números mediante el símbolo "_" (Ejem: 2_9)
  - Columna E: Notas.

- Cada revista dispone de un **índice onomástico**, un único documento en formato Excel, de índole general, en el que cada fila contiene la siguiente información:

  - Columna A: Nombre de un autor que aparece en la revista.
  - N Columnas: Título del fragmento de texto que han publicado los autores que van desde la columna C hasta la H, y que han traducido los de las columnas I y J.

En reuniones posteriores, se debatieron funcionalidades necesarias y se refinaron los requisitos de la aplicación que describiremos más adelante (ver sección 4.3).

4.2 Actores

Un actor en una aplicación especifica un rol jugado por un usuario o cualquier otro sistema que interactúa con el sujeto. Nuestra aplicación dispondrá de dos tipos de actores, **usuario** y **administrador**.

El rol de administrador nace debido a la necesidad de crear un gestor de contenidos. Este puede acceder mediante autenticación al bloque de la aplicación dedicado a la administración.

El otro rol disponible, será el de usuario, que puede ser cualquier persona que acceda a la web donde se muestran los contenidos. No necesita autenticación.

El administrador también puede acceder a todas las funcionalidades de las que dispone el usuario.

4.3 Requisitos de la aplicación

Un requisito es una necesidad física o funcional que el producto trata de satisfacer. Los requisitos se dividen entre funcionales y no funcionales. Los requisitos no funcionales son aquellos que no están específicamente relacionados con la funcionalidad del sistema. Los requisitos funcionales son aquellos que describen el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones.

De las reuniones con el Product Owner se extrajeron los siguientes, que se detallan a continuación y que podemos ver plasmados en el diagrama de casos de uso (ver fig. 4.1).

4.3.1 Requisitos funcionales

Se realizó una división de los requisitos funcionales en dos áreas. La primera para los referentes al espacio web accesible por todos los usuarios y la segunda para la gestión de contenidos, accesible solo por un administrador (ver sec. 4.2).

- **Espacio web** (accesible por todos los usuarios)
  - Mostrar una introducción e información para la interpretación de los datos de los índices.
  - Mostrar todas las revistas activadas.
  - Mostrar datos identificativos y descriptivos de estas revistas.
  - Mostrar en versión PDF cada número de cada revista.
  - Mostrar el índice de autores de cada revista.

  - Mostrar el índice de ilustraciones de cada revista.
  - Mostrar el índice onomástico de cada revista.
  - Mostrar el estudio de cada revista.

- **Gestión de contenidos** (solo accesible por administradores)
  - Securizar el gestor mediante inicio de sesión con usuario y contraseña.
  - Mostrar todas las revistas y sus datos.
  - Añadir revista nueva.
  - Eliminar una revista existente.
  - Modificar datos identificativos y descriptivos de una revista.
  - Mostrar todos los números de una revista y sus datos.
  - Añadir numero de una revista. Para ello debemos gestionar la subida de documentos en formato PDF y de sus correspondientes imágenes.
  - Eliminar numero de una revista.
  - Modificar numero de una revista.
  - Añadir estudio de una revista mediante archivo en formato Word.
  - Modificar estudio de una revista mediante archivo en formato Word.
  - Añadir grupos de índices de autores agrupados en un documento en formato Excel.
  - Añadir un registro de índice de autores.
  - Eliminar un registro de índice de autores.
  - Añadir grupos de índices de ilustraciones agrupados en un documento en formato Excel.
  - Añadir un registro de índice de ilustraciones.
  - Eliminar un registro de índice de ilustraciones.
  - Añadir grupos de índices onomásticos agrupados en un documento en formato Excel.
  - Añadir un registro de índice onomástico.
  - Eliminar un registro de índice onomástico.
  - Activar y desactivar visualización de revista en la web para usuarios.
  - Añadir país de origen de revista.
  - Eliminar país de origen de revista.
  - Modificar credenciales.

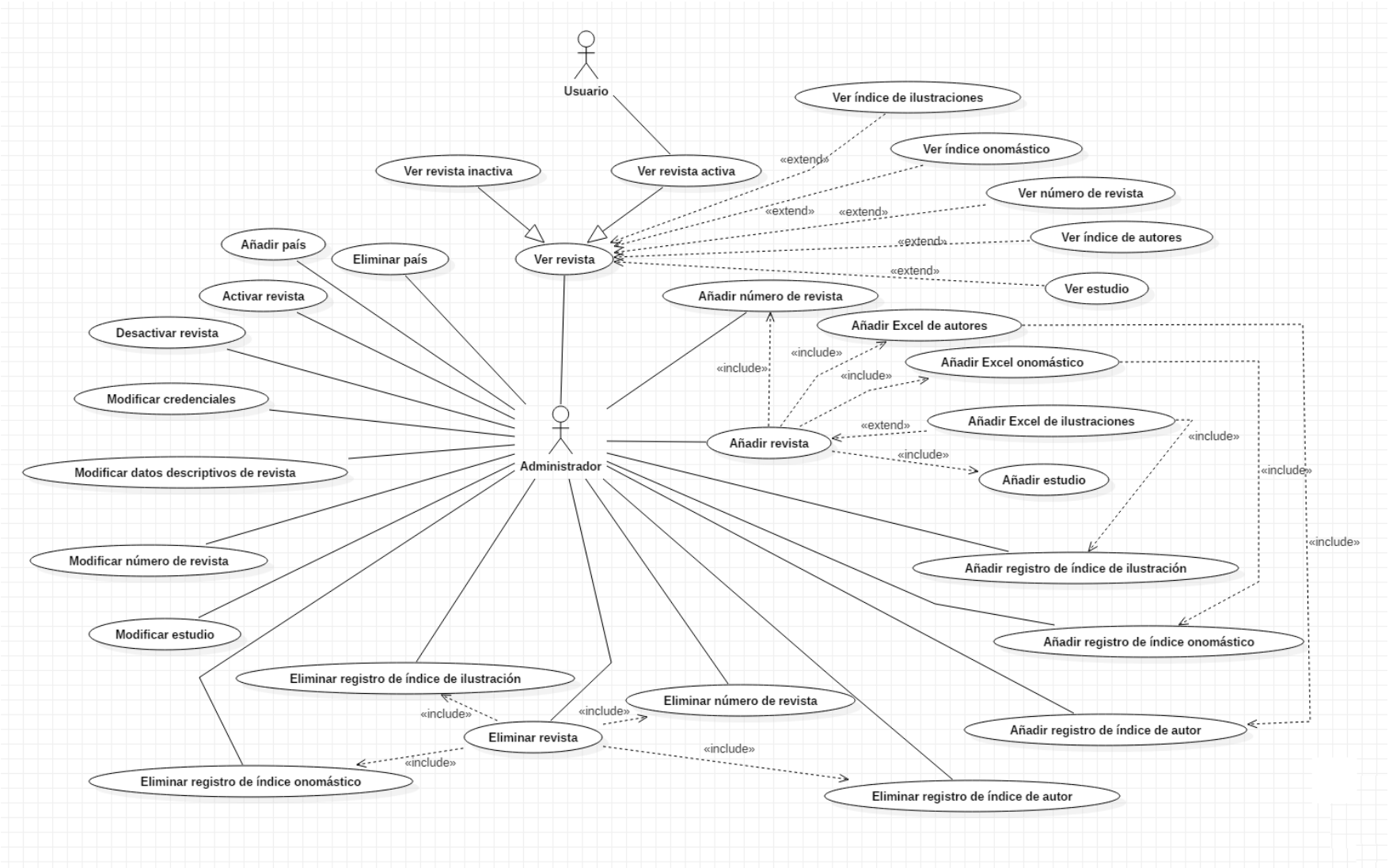

Figura 4.1: Diagrama de casos de uso

4.3.2 Requisitos no funcionales

De estas reuniones se concluyeron también los siguientes requisitos no funcionales:

- La aplicación debe estar en Español, ya que el ámbito de las revistas es de habla hispana.
- La aplicación web accesible por el usuario debe tener un diseño minimalista y de fácil uso.
- La acción de subir una revista en el gestor debe realizarse en una única vista.

4.4 Historias de usuarios

Una historia de usuario es una descripción de una funcionalidad que debe incorporar un sistema de software, y cuya implementación aporta valor al cliente. Se utilizaron estas para dividir el trabajo en tareas, tomando como referencia principal los requisitos funcionales detallados anteriormente (ver subsec. 4.3.1).

A continuación, se puede ver un ejemplo de un requisito transformado en historias de usuario:

- Requisito:
  - Activar y desactivar visualización de revista en la web para usuarios.
- Historias de usuario:
  - Como administrador, activar revista.
  - Como administrador, desactivar revista.

Capítulo 5

Planificación

En este capítulo se explican las etapas de la planificación durante el proceso de desarrollo de nuestra aplicación. Esta planificación ha sido acorde a la metodología Scrum (ver cap. 3, sec. 3.2).

5.1 Planificación inicial

Una vez realizado el análisis (ver cap. 4) que nos permitió conocer el alcance de la aplicación, se decidió dividir el trabajo en 5 áreas lógicas:

- **Generación inicial del proyecto y configuración**: Recolección de información necesaria para la puesta en marcha del proyecto, preparación y configuración de herramientas y entorno de desarrollo y construcción del arquetipo base para la aplicación.

- **Implementación de la persistencia**: Diseño de una base de datos, adición de la configuración necesaria e implementación de esta.

- **Gestión de acceso a la administración**: Adición de la configuración adicional necesaria e implementación del modelo de inicio de sesión en la aplicación para usuarios administradores (ver cap. 4, sec. 4.2).

- **Generación del espacio web dedicado a la administración** (ver cap. 4, subsec. 4.3.1): Planificación e implementación de las vistas, controladores, modelo y la lógica de negocio del gestor de contenidos (ver cap. 6, sec. 6.1).

- **Generación del espacio web destinado a todos los usuarios** (ver cap. 4, sec. 4.2): Diseño e implementación de la vista y controladores dedicados a la muestra de los contenidos de las revistas (ver cap. 4, sec. 4.1)

5.2 Adaptación de la planificación a Scrum

En este apartado se detallan las fases de desarrollo del proyecto siguiendo la metodología Scrum adaptada a nuestras necesidades (ver cap. 3, sec. 3.2). Al inicio de cada Sprint se realizó un Sprint Planning (ver cap. 3, subsec. 3.1.2), del cual obtuvimos el Sprint Backlog (ver cap. 3, subsec. 3.1.3), y al finalizar este, se procedió a la realización de los Sprint Review y Sprint Retrospective (ver cap. 3, subsec. 3.1.2).

El desarrollo del proyecto se dividió en seis fases o sprints que se detallan a continuación. La duración de estos fue de entre dos y tres semanas, dependiendo de la complejidad de los mismos. El proyecto alcanzó finalmente una duración de 15 semanas.

5.2.1 Primer sprint

En este primer sprint, se decidió llevar a cabo la implementación de la primera de las cinco áreas lógicas en las que se dividió nuestro proyecto (ver sec. 5.1). En primer lugar, se llevó a cabo la creación del Product Backlog (ver cap. 3, subsec. 3.1.3) con sus historias de usuario (ver cap. 4, sec. 4.4) obtenidas a partir de los requisitos funcionales (ver cap. 4, subsec. 4.3.1). A continuación se instalaron y configuraron herramientas como Eclipse, Gitlab y Maven (ver cap. 2, sec 2.3). Una vez terminado este proceso, se generó el arquetipo de la aplicación mediante Sprint Boot Initializer (ver cap. 2, subsec 2.2.1) y las dependencias iniciales necesarias.

Durante el primer Sprint Review se realizó una prueba básica de ejecución e inicialización del proyecto satisfactoria. El Product Owner no estuvo presente debido al carácter de configuración del sprint. Acto seguido se realizó el primer Sprint Retrospective, del que se sacaron conclusiones positivas debido a la sencillez de Spring Boot para ser configurado.

La duración de este sprint fue de 2 semanas.

5.2.2 Segundo sprint

En este sprint se llevó a cabo la implementación de la segunda área lógica (ver sec. 5.1). Para ello se diseñó una primera aproximación del esquema de bases de datos de la aplicación, se instaló y configuró un sistema gestor de bases de datos, en concreto MySQL (ver cap. 2, sec. 2.4), y se implementó la configuración necesaria en el proyecto.

Mediante este Sprint Review se realizó una prueba básica de ejecución y conexión a la base de datos. El Product Owner no estuvo presente debido a la poca funcionalidad que ofrecía todavía el proyecto. Acto seguido se realizó el Sprint Retrospective, del que se sacaron conclusiones positivas debido a la facilidad de configuración y puesta en marcha de la conexión a la base de datos.

La duración de este sprint fue de 2 semanas.

5.2.3 Tercer sprint

En este sprint se llevó a cabo la implementación del modelo de inicio de sesión en la aplicación para usuarios administradores especificada en la tercera área lógica (ver sec. 5.1). Se utilizó Spring Boot Security, un framework de Spring Boot (ver cap. 2, subsec. 2.2.1), que proporciona autenticación , autorización y otras características de seguridad para aplicaciones empresariales. Para ello, se implementó la vista de inicio de sesión y la lógica de negocio necesaria para la realización de la autenticación de credenciales.

Mediante este Sprint Review se realizó una prueba de inicio de sesión en el sistema de administración. El Product Owner estuvo presente en esta reunión y dio el visto bueno al sistema de autenticación. Acto seguido se realizó el Sprint Retrospective, del que se sacaron conclusiones positivas debido a la correcta ejecución de la prueba realizada durante el Sprint Review.

La duración de este sprint fue de 2 semanas.

5.2.4 Cuarto sprint

En este sprint se llevó a cabo la primera parte de la cuarta área lógica (ver sec. 5.1). Se decidió implementar en primer lugar, el modelo de datos mediante la creación de las entidades (ver cap. 6, subsec. 6.2.1) y los repositorios (ver cap. 6, subsec. 6.2.2) necesarios para llevar a cabo la lógica de la aplicación. Acto seguido se desarrollaron servicios (ver cap. 6, subsec. 6.5) básicos para dotar de una base sólida al proyecto sobre la que implementar funcionalidades más complejas en el siguiente sprint. También se crearon las vistas necesarias para la visualización de estos servicios.

Mediante este Sprint Review se realizó una prueba en la que se mostraron servicios básicos de búsqueda y representación de información referente a las revistas (ver cap. 4, sec. 4.1). De esta reunión cabe destacar la aceptación por parte del Product Owner del diseño inicial de la vista. Acto seguido se realizó el Sprint Retrospective, del que se sacaron conclusiones positivas debido a la correcta ejecución de la prueba realizada durante el Sprint Review.

La duración de este sprint fue de 3 semanas.

5.2.5 Quinto sprint

En este sprint se llevó a cabo la segunda parte de la cuarta área lógica (ver sec. 5.1). Se implementaron los servicios necesarios para la subida de archivos al servidor (imágenes, PDF, Excel y Word), y toda la lógica de negocio necesaria para las operaciones de creación, modificación y eliminación de contenidos. También se crearon las vistas restantes para dotar de total funcionalidad al gestor.

Mediante este Sprint Review se realizó una prueba en la que se mostró todo el contenido

completo de la página de administración. El Product Owner realizo una subida de una revista, modificaciones en los distintos campos y componentes de las ya existentes y eliminaciones de distintos campos de datos. En esta reunión se destacó por parte del Product Owner la necesidad de añadir ventanas modales para la confirmación de las eliminaciones de datos.

Acto seguido se realizó el Sprint Retrospective, del que se sacaron conclusiones positivas debido a la correcta ejecución de la prueba realizada durante el Sprint Review. También se encontraron aspectos a mejorar, como la inclusión de ventanas modales para evitar el borrado por error. Además, a nivel de código, se vio la necesidad de incluir en la lógica de negocio controles para evitar que el usuario elimine todos los registros en lugares en los que debe haber al menos uno de ellos (Ejemplo: Todas las revistas deben tener al menos un numero).

La duración de este sprint fue de 3 semanas.

5.2.6 Sexto sprint

En este último sprint, se llevó a cabo la implementación de la última de las cinco áreas lógicas (ver sec. 5.1). Para ello, se crearon todos los componentes necesarios para desarrollar un espacio web sencillo, accesible a todos los usuarios. Además, se realizaron correcciones y mejoras en el bloque de administración.

Mediante este Sprint Review se realizó una prueba en la que se mostró todo el contenido completo de la página web, además de las mejoras implementadas en la administración. El Product Owner destacó la facilidad de uso de la aplicación. Acto seguido se realizó el Sprint Retrospective, del que se sacaron conclusiones positivas debido a la correcta ejecución de la prueba realizada durante el Sprint Review.

La duración de este sprint fue de 3 semanas.

Capítulo 6

Diseño

En este capítulo se explican aspectos relevantes del diseño de nuestra aplicación, como la arquitectura adoptada y los servicios utilizados.

6.1 Arquitectura

Como arquitectura global de la aplicación, se ha utilizado el patrón Modelo - Vista - Controlador (fig. 6.1).

Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características básicas para facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

Para ello MVC utiliza tres componentes:

- **Modelo** - Es la representación de la información con la cual el sistema opera.
- **Vista** - Presenta la información y lógica de negocio en un formato adecuado para interactuar por parte del usuario.
- **Controlador** - Es el intermediario entre la 'vista' y el 'modelo'. Responde a eventos por parte del usuario e invoca peticiones al modelo cuando se hace alguna solicitud sobre la información.

La mayor ventaja de esta arquitectura es la separación de responsabilidades dentro de la aplicación, de esta forma si hay cambios en los datos (Modelo) no afecta a la vista y esta simplemente se actualizará con los nuevos contenidos. Además, el usuario nunca interactúa directamente con los datos.

En nuestra arquitectura MVC, el controlador no actúa directamente con el modelo, sino que lo hace a través de una capa de servicios.

Esta capa nos permite la separación de responsabilidades, ya que el modelo se ocupa de realizar las llamadas a la base de datos y el servicio de realizar la lógica de negocio. Así, reducimos también el acoplamiento, ya que, en caso de realizar cambios en el modelo, no es necesario modificar los controladores, estos cambios se harían en el servicio sin la necesidad de modificar estos. Con esta implementación se aumenta la seguridad de la aplicación ya que el controlador no está en contacto directo con el modelo.

Esta capa es la encargada de recibir peticiones de los controladores, realizar llamadas al modelo para obtener los datos necesarios para la lógica de negocio y devolver una respuesta a los controladores.

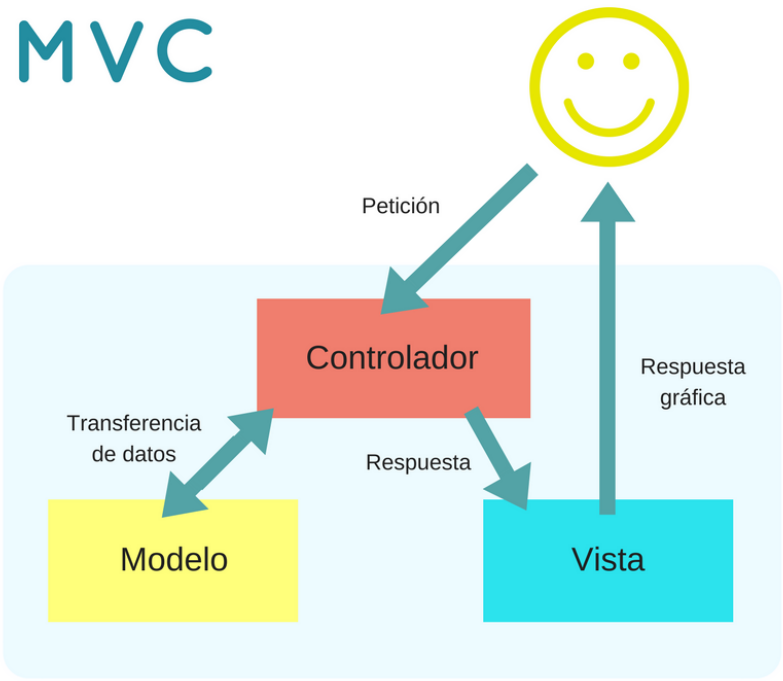

Figura 6.1: Arquitectura MVC

El proyecto se divide también en front end y back end:

- **Front end**: Se ocupa de la interacción con el usuario (Vista).
- **Back end**: Se ocupa de los datos y la lógica de negocio de la aplicación (Modelo).

Los controladores son los encargados de la comunicación entre la vista y el modelo.

6.2 Modelo

Engloba todos los componentes encargados de la gestión de los accesos a la información (ver subsec. 6.1)

6.2.1 Entidades

El modelo del dominio de la aplicación está formado por las entidades. Estas son clases que están mapeadas a una tabla en la base de datos. Este mapeo se realiza mediante una herramienta ORM (Object Relational Mapping), en este caso Hibernate (a través de Spring JPA), que se ocupa del ciclo de vida de estos objetos. La finalidad es la manipulación de la base de datos como si se tratase de un objeto Java en memoria.

Las entidades están relacionadas con otras entidades, y estas relaciones son expresadas a través de meta datos objeto/relacional. Los meta datos del objeto/relacional son especificados directamente en el fichero de la clase, usando las anotaciones de Java.

A continuación, en la figura 6.2, se muestra el diagrama de clases de la aplicación, que fue completado incrementalmente en cada sprint, en los cuales se fueron aumentando las clases necesarias hasta finalizar el desarrollo.

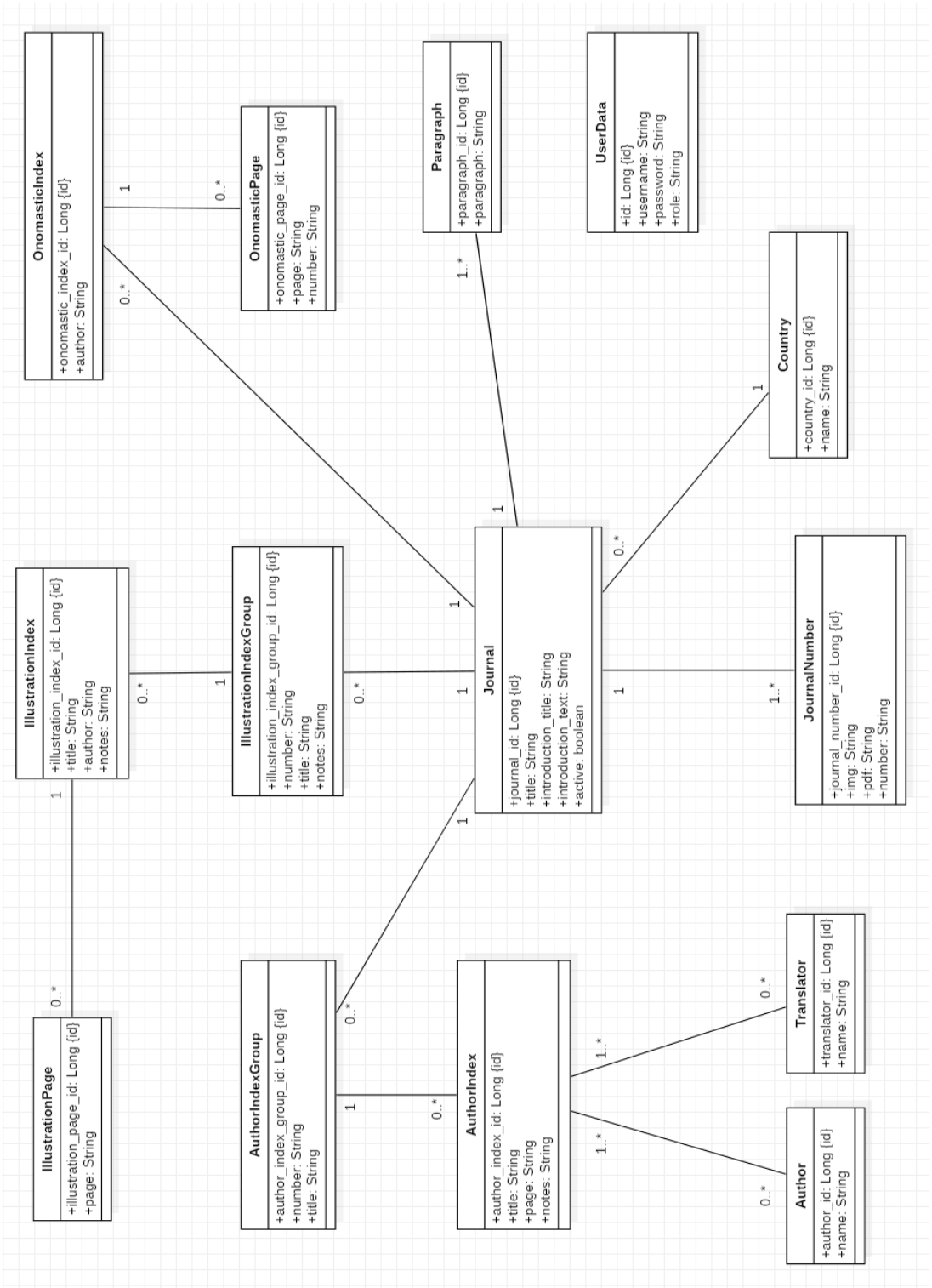

Figura 6.2: Diagrama de clases de la aplicación

6.2.2 Repositorios

El módulo JPA de Spring Data contiene interfaces que realizan consultas sobre la base de datos. Estos repositorios sirven de interfaz abstracta a una base de datos. JPA implementa consultas básicas como guardado, borrado y búsqueda por identificador. También permite hacer consultas personalizadas contra la base de datos utilizando SQL (ver cap. 2, sec. 2.4) e incluso autoimplementar consultas siguiendo convenciones de nombrado [17]. Los repositorios funcionan como intermediario entre la base de datos y la capa de servicios, donde se implementa la lógica de negocio con ayuda también de las entidades (ver subsec. 6.2.1). En nuestra aplicación, cada entidad dispone de un repositorio propio.
A continuación, mostramos el ejemplo de un repositorio encargado de las consultas referentes a países (fig. 6.3):

Figura 6.3: Repositorio de países (CountryRepository)

6.3 Controladores

Los controladores son los nexos principales entre la vista y el modelo (ver sec. 6.1). Son los encargados de responder a eventos por parte del usuario e invoca peticiones al modelo cuando se hace alguna solicitud sobre la información. Nuestra aplicación dispondrá de los siguientes:

- **HomeController** - Se encarga de controlar las peticiones por parte de la vista realizadas en la página web de todos usuarios (ver cap. 4, subsec. 4.3.1). Solo recibe llamadas de visualización de componentes.

- **LoginController** - Se encarga de gestionar las peticiones de inicio de sesión.

- **AdminController** - Se encarga de controlar las peticiones por parte de la vista realizadas en el gestor para administradores (ver cap. 4, subsec. 4.3.1).

6.4 Vistas

Las vistas son las encargadas de presentar la información y lógica de negocio en un formato adecuado para interactuar por parte del usuario (ver subsec. 6.1). Para ello el controlador delega a los objetos de la vista la información del modelo necesaria para que estos muestren la interfaz de usuario.

Para la implementación de estas interfaces (ver cap. 7, sec. 7.5), utilizaremos HTML (ver cap. 2, sec. 2.1) y Thymeleaf (ver cap. 2, sec. 2.2) para el marcado de la estructura de las páginas y la creación de plantillas respectivamente, CSS (ver cap. 2, sec. 2.1) para dotarlas de estilo y JavaScript (ver cap. 2, sec. 2.1) para añadir mejoras en estas. También nos ayudaremos de Bootstrap (ver cap. 2, sec. 2.2) y sus plantillas de diseño de elementos basado en HTML y CSS, así como de sus extensiones de JavaScript adicionales.

6.5 Servicios

Los servicios actúan como capa intermedia entre el controlador y el modelo y se encargan de la lógica de negocio del proyecto (ver sec. 6.1). Estos se han divido agrupándolos por funcionalidades, lo que permite que cada uno de ellos sea independiente de los otros. Se disminuye así el acoplamiento y la cohesión.

Se decidió realizar la siguiente división en servicios para el proyecto:

- **Servicio de revistas (JournalService) (fig. 6.4)**

Se encarga de todo lo referente a adición de revistas, modificación de sus datos o borrado de estas en el gestor para la administración (ver cap. 4, sec. 4.3.1). Además, también implementa los servicios de búsqueda necesarios para la visualización de estas.

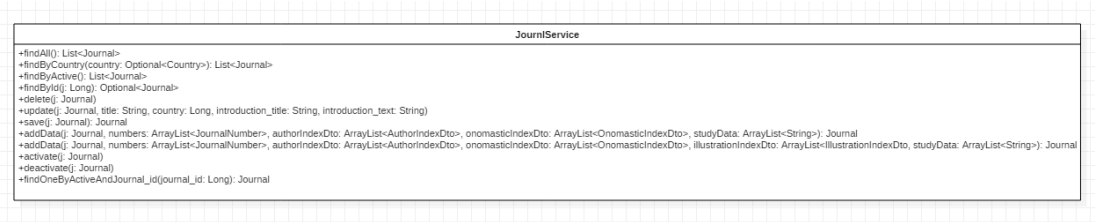

Figura 6.4: Servicio de revistas

– **Servicio de números de revistas (JournalNumberService) (fig. 6.5)**

Se encarga de todo lo referente a adición de números de revistas, modificación de sus datos o borrado de estos en el gestor para la administración (ver cap. 4, sec. 4.3.1). Además, también implementa los servicios de búsqueda necesarios para la visualización de estos.

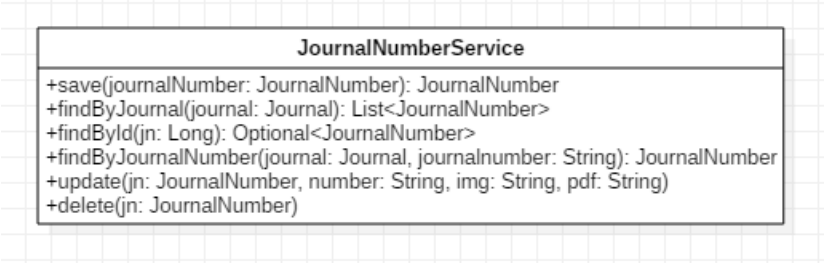

Figura 6.5: Servicio de números de revistas

– **Servicio de países (CountryService) (fig. 6.6)**

Se encarga de gestionar los países de origen a los que pertenecen las revistas.

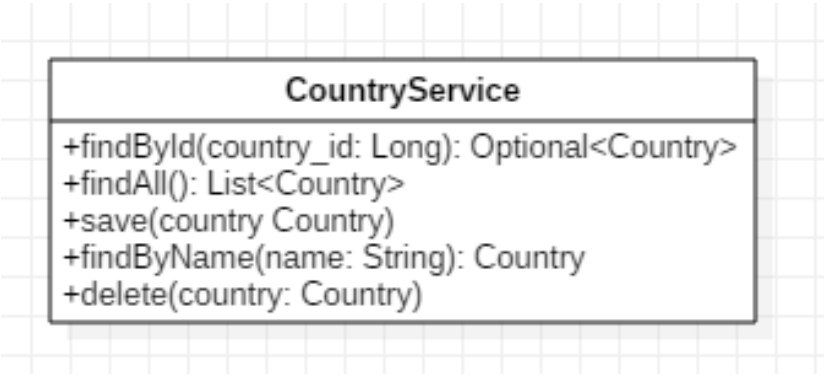

Figura 6.6: Servicio de países

– **Servicio de archivos (FileStorageService) (fig. 6.7)**

Se encarga de la subida de archivos al servidor, la eliminación de estos y la sustitución cuando se realiza una modificación.

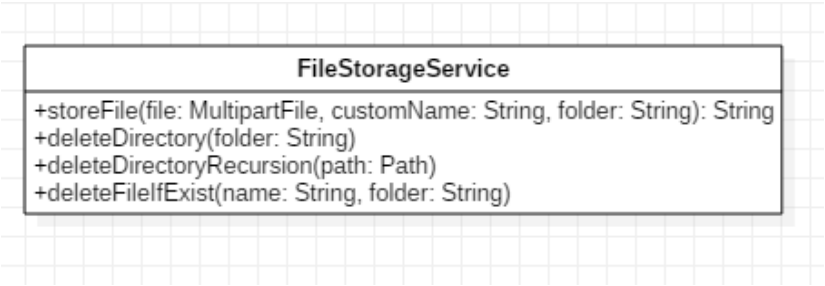

Figura 6.7: Servicio de archivos

– **Servicio de Excel (ExcelUtilitiesService) (fig. 6.8)**

Se encarga de leer archivos Excel y traducirlos a objetos del dominio para ser guardados en la base de datos. Ofrece funcionalidades para índices de autores, de ilustraciones y onomástico.

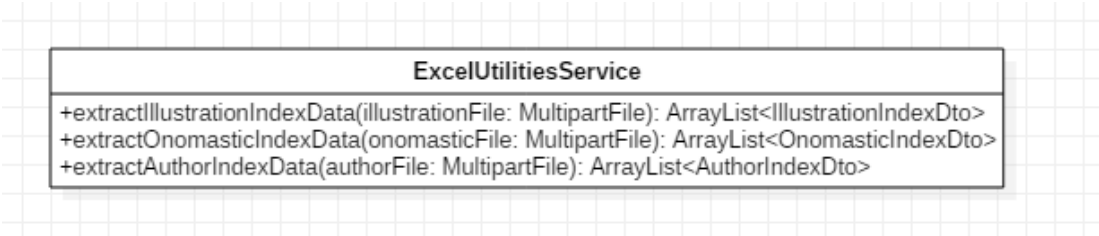

Figura 6.8: Servicio de Excel

– **Servicio de Word (WordUtilitiesService) (fig. 6.9)**

Se encarga de leer archivos Word y traducirlos a objetos del dominio para ser guardados en la base de datos. Ofrece funcionalidad para el estudio de la revista.

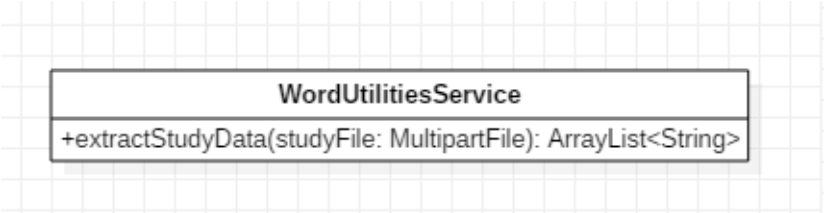

Figura 6.9: Servicio de Word

– **Servicio de usuario (UserService) (fig. 6.10)**

Se encarga de la gestión de usuarios.

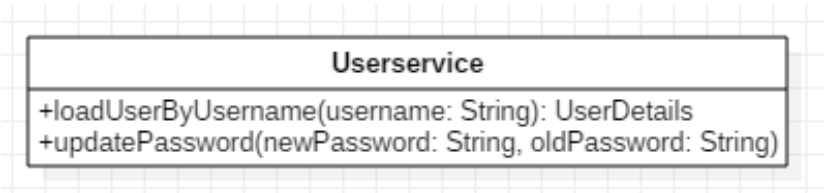

Figura 6.10: Servicio de Usuarios

– **Servicio de grupos de índices de autores (AuthorIndexGroupService) (fig. 6.11)**

Se encarga de agrupar por números de revistas los índices de autores.

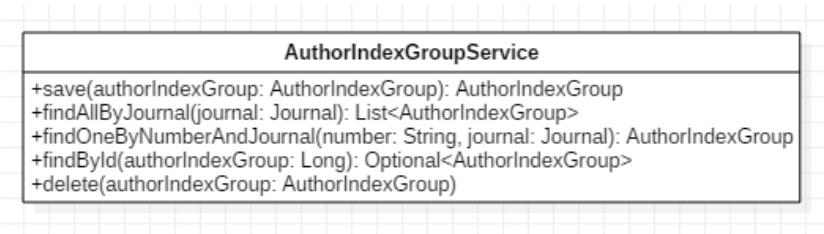

Figura 6.11: Servicio de grupos de índice autores

– **Servicio de índice de autores (AuthorIndexService) (fig. 6.12)**

Se encarga de almacenar y mostrar los índices de autores de las revistas.

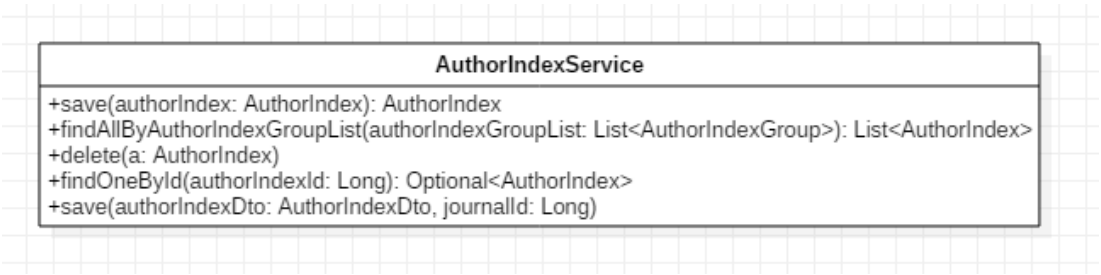

Figura 6.12: Servicio de índice de autores

– **Servicio de autores (AuthorService) (fig. 6.13)**

Se encarga de gestionar los autores de los índices autores.

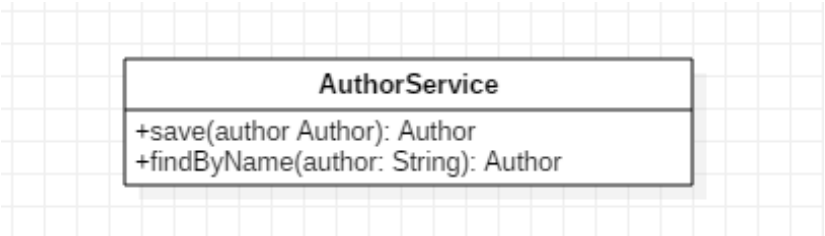

Figura 6.13: Servicio de autores

– **Servicio de traductores (TranslatorService) (fig. 6.14)**

Se encarga de gestionar los traductores de los índices autores.

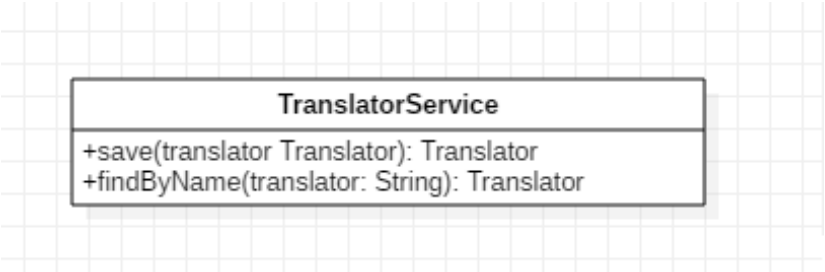

Figura 6.14: Servicio de traductores

– **Servicio de grupos de índices de ilustraciones (IllustrationIndexGroupService) (fig. 6.15)**

Se encarga de agrupar por números de revistas los índices de ilustraciones.

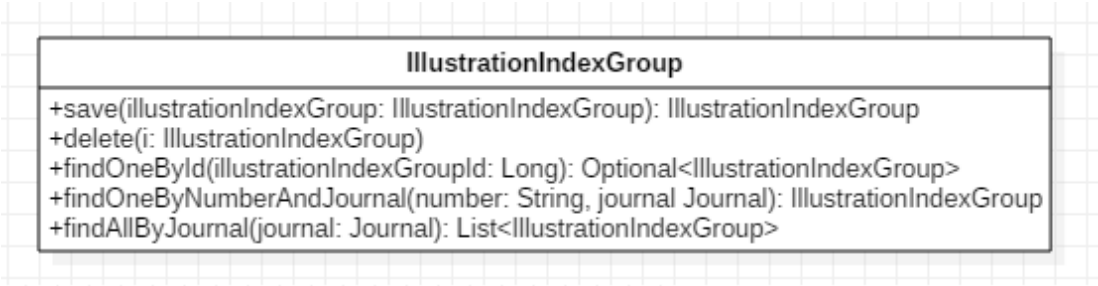

Figura 6.15: Servicio de grupos de índices de ilustraciones

– **Servicio de índice de ilustraciones (IllustrationIndexService) (fig. 6.16)**

Se encarga de almacenar y mostrar los índices de ilustraciones de las revistas.

Figura 6.16: Servicio de índice de ilustraciones

– **Servicio de páginas de índices de ilustraciones (IllustrationPageService) (fig. 6.17)**

Se encarga de agrupar las páginas de los índices de ilustraciones.

Figura 6.17: Servicio de páginas de índices de ilustraciones

– **Servicio de índice onomástico (OnomasticIndexService) (fig. 6.18)**

Se encarga de almacenar y mostrar los índices onomásticos de las revistas.

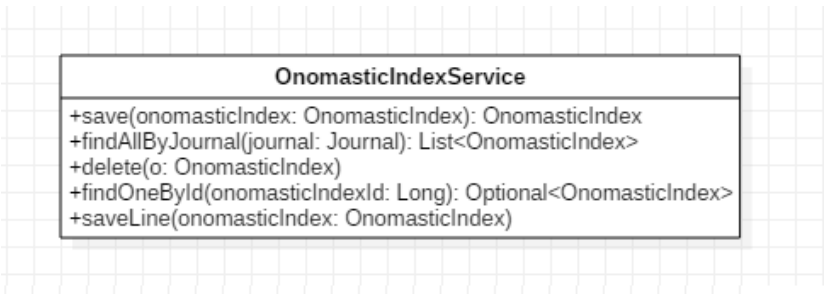

Figura 6.18: Servicio de índice onomástico

– **Servicio de páginas de índice onomástico (OnomasticPageService) (fig. 6.19)**

Se encarga de gestionar las páginas de los índices onomásticos.

Figura 6.19: Servicio de páginas de índice onomástico

– **Servicio de párrafos de estudio (ParagraphService) (fig. 6.20)**

Se encarga de gestionar los estudios de las revistas.

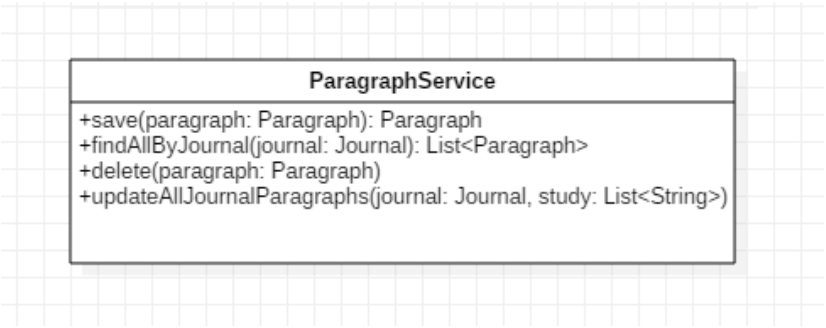

Figura 6.20: Servicio de párrafos de estudio

Capítulo 7

Implementación

En este capítulo se muestran aspectos importantes de la implementación del proyecto.

7.1 Estructura y configuración

7.1.1 Estructura del proyecto

Mediante la utilización de Spring Initializr (ver cap. 2, subsec. 2.2.1) creamos el arquetipo base del proyecto. Esta es la organización de los paquetes más importantes:

```
├── config
│
├── domain
│   ├── Journal.java
│   ├── JournalRepository.java
│   ├── Author.java
│   ├── AuthorRepository.java
│   └── ...
│
├── dto
│   ├── AuthorIndexDto.java
│   ├── JournalDto.java
│   └── ...
│
├── service
│   ├── OnomasticPageService.java
│   ├── OnomasticPageServiceImpl.java
│   ├── AuthorService.java
│   ├── AuthorServiceImpl.java
│   └── ...
│
├── exceptions
│   ├── FileStorageException.java
│   └── ...
│
└── web
    ├── AdminController.java
    ├── LoginController.java
    └── ...
```

7.1.2 Configuración

La construcción de la aplicación (incluye compilación y ejecución de test) es gestionada por Maven (ver cap. 2, sec. 2.3). Maven permite la creación de módulos (en nuestro caso el proyecto está formado por un único modulo) los cuales contienen un archivo POM (pom.xml). Este archivo contiene información y detalles de configuración para que Maven pueda construir el proyecto. En este documento se instalaron las dependencias iniciales que utilizará Spring Boot (ver cap. 2, subsec. 2.2.1), las cuales son:

- **spring-boot-starter-web** - Dependencia utilizada para el desarrollo de aplicaciones web usando Spring MVC.
- **spring-boot-devtools** - Dependencia útil durante el desarrollo, que permite la realización de cambios en caliente en el proyecto.
- **spring-boot-starter-data-jpa** - Dependencia inicial para implementar la capa de persistencia relacional usando Spring Data.
- **spring-boot-starter-security** - Dependencia necesaria para la implementación de la seguridad en la aplicación.
- **spring-boot-starter-test** - Dependencia utilizada por Spring Boot para la implementación de pruebas.
- **mysql-connector-java** - Dependencia necesaria para la implementación de la base de datos principal con MySQL.
- **h2** - Dependencia necesaria para la implementación de la base de datos de pruebas.
- **spring-boot-starter-thymeleaf** - Dependencia necesaria para la utilización de la biblioteca Thymeleaf (ver cap. 2, sec. 2.2) y su motor de plantillas.

Otras dependencias fueron añadidas posteriormente al proyecto cuando fueron necesarias.

7.2 Persistencia

Para la implementación de la persistencia se empleó una base de datos relacional, MySQL (ver cap. 2, sec. 2.4). Para el modelo del dominio de la aplicación se utilizaron entidades (ver cap. 6, subsec. 6.2.1). Estas son clases que están mapeadas a una tabla en la base de datos. Este mapeo se realizó mediante una herramienta ORM (Object Relational Mapping), en este caso

Hibernate (a través de Spring JPA), que se ocupa del ciclo de vida de estos objetos. Para ello se utilizaron anotaciones como ”@Id” y ”@Entity” en nuestras entidades. Hibernate facilita también el manejo de las relaciones entre entidades mediante la utilización de anotaciones como podemos ver en la figura 7.1.

```java
@Entity
public class Journal {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long journal_id;

    private String title;

    private String introduction_title;

    @Lob
    @Column(columnDefinition = "MEDIUMTEXT")
    private String introduction_text;

    @ManyToOne
    @JoinColumn(name = "country_id")
    private Country country;

    @OneToMany(mappedBy = "journal", orphanRemoval = true)
    private List<JournalNumber> number;

    @OneToMany(mappedBy = "journal", orphanRemoval = true)
    private List<AuthorIndex> authorIndex;

    @OneToMany(mappedBy = "journal", orphanRemoval = true)
    private List<OnomasticIndex> onomasticIndex;

    @OneToMany(mappedBy = "journal", orphanRemoval = true)
    private List<IllustrationIndex> illustrationIndex;

    private boolean active = false;

    @OneToMany(mappedBy = "journal", orphanRemoval = true)
    private List<Paragraph> study;
```

Figura 7.1: Entidad de una revista (Journal)

Posteriormente, JPA fue utilizada para el desarrollo de los repositorios (ver 7.2), los cuales son interfaces que extienden de la especificación JpaRepository, utilizados para el acceso a los objetos de la capa de persistencia. Estos repositorios proporcionan funciones básicas CRUD (Create, Read, Update and Delete) y además nos permiten la creación de nuestras propias operaciones.

```java
@Repository
public interface JournalRepository extends JpaRepository<Journal, Long> {

    List<Journal> findAllByCountry(Optional<Country> country);

    List<Journal> findAllByActive(boolean active);

}
```

Figura 7.2: Repositorio de una revista (JournalRepository)

Otras anotaciones de interés son aquellas que nos permiten especificar mediante las entidades aspectos relacionados con base de datos. En la siguiente imagen (7.3) vemos como mediante el uso de las anotaciones @Lob, para el tratamiento de objetos pesados, en este caso un estudio de una revista, y @Column y el atributo columnDefinition de esta, podemos establecer el String ”paragraph”, mapeado de forma predefinida como ”varchar”, como un ”mediumtext” en la base de datos.

```java
@Entity
public class Paragraph {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long paragraph_id;

    @Lob
    @Column(columnDefinition = "MEDIUMTEXT")
    private String paragraph;

    @ManyToOne
    @JoinColumn(name = "journal_id")
    private Journal journal;
```

Figura 7.3: Ejemplo de anotaciones (Paragraph)

7.3 Servicios

En la implementación de nuestra aplicación MVC, el controlador interactúa con el modelo a través de una capa de servicios.

Esta capa permite la separación de responsabilidades, ya que el modelo se ocupa de reali-

zar las llamadas a la base de datos y el servicio de realizar la lógica de negocio. Así se reduce también el acoplamiento, ya que, en caso de realizar cambios en el modelo, no es necesario modificar los controladores, estos cambios se harían en el servicio sin la necesidad de modificar estos. Con esta implementación se aumenta la seguridad de la aplicación ya que el controlador no está en contacto directo con el modelo.

Esta capa es la encargada de recibir peticiones de los controladores, realizar llamadas al modelo para obtener los datos necesarios para la lógica de negocio y devolver una respuesta a los controladores.

La implementación de estos servicios se lleva a cabo mediante una interface, en la cual se declaran las funciones de la lógica de negocio, y que es llamada por los controladores, y una clase que implementa esta, en la que se desarrolla la lógica de negocio y se hacen las peticiones de datos a los repositorios, los cuales son inyectados mediante la anotación @Autowired.

A continuación, podemos ver un ejemplo de una interface (7.4) y la clase que implementa esta (7.5).

```
public interface JournalService {

    public List<Journal> findAll();

    public List<Journal> findByCountry(Country optional);

    public List<Journal> findByActive();

    public Optional<Journal> findById(Long j);

    public void delete(Journal j);

    public void update(Journal j, String title, Long country, String introduction_title, String introduction_text);

    public Journal save(Journal j);

    public Journal addData(Journal j, ArrayList<JournalNumber> numbers, ArrayList<AuthorIndexDto> authorIndexDto,
            ArrayList<OnomasticIndexDto> onomasticIndexDto, ArrayList<String> studyData);

    public Journal addData(Journal j, ArrayList<JournalNumber> numbers, ArrayList<AuthorIndexDto> authorIndexDto,
            ArrayList<OnomasticIndexDto> onomasticIndexDto, ArrayList<IllustrationIndexDto> illustrationIndexDto,
            ArrayList<String> studyData);

    public void activate(Journal j);

    public void deactivate(Journal j);

    public Journal findOneByActiveAndJournal_id(Long journal_id);
}
```

Figura 7.4: Ejemplo de interface (JournalService)

```java
@Service
public class JournalServiceImpl implements JournalService {

    @Autowired
    private JournalRepository journalRepository;

    @Autowired
    private JournalNumberService journalNumberService;

    @Autowired
    private AuthorService authorService;

    @Autowired
    private CountryService countryService;

    @Autowired
    private TranslatorService translatorService;

    @Autowired
    private AuthorIndexService authorIndexService;

    @Autowired
    private AuthorIndexGroupService authorIndexGroupService;

    @Autowired
    private OnomasticIndexService onomasticIndexService;

    @Autowired
    private OnomasticPageService onomasticPageService;

    @Autowired
    private IllustrationIndexService illustrationIndexService;

    @Autowired
    private IllustrationPageService illustrationPageService;

    @Autowired
    private IllustrationIndexGroupService illustrationIndexGroupService;

    @Autowired
    private ParagraphService paragraphService;

    @Override
    @Transactional(readOnly = true)
    public List<Journal> findAll() {
        return journalRepository.findAll();
    }

    @Override
    @Transactional(readOnly = true)
    public List<Journal> findByCountry(Country country) {
        return journalRepository.findAllByCountry(country);
    }
```

Figura 7.5: Ejemplo de clase que implementa una interface (JournalServiceImpl)

7.3.1 Implementación de servicios de transformación de datos

Como se puede ver en el análisis (ver cap. 4, sec. 4.1), cada revista dispone de un índice de autores en el que cada fila contiene información sobre los autores y traductores que aparecen en la revista, un índice de ilustraciones en el que cada fila contiene información sobre las imágenes que aparecen en la revista y un índice onomástico, donde aparece el nombre de personas mencionadas en el texto, por orden alfabético con el número de las páginas donde aparecen. Este índice, al contrario que los de autores e ilustraciones, es de índole general. Estos tres tipos de índices están contenidos en tres archivos de formato Excel respectivamente.

Como requisito de nuestra aplicación (ver cap. 4, sec. 4.3.1), debemos disponer de un mecanismo de subida de documentos que transforme la información obtenida de estos archivos a una base de datos [18]. Para ello hemos utilizado una librería Java, Apache POI, la cual ha sido añadida a nuestro proyecto mediante la declaración de la dependencia ”poi-ooxml” en el POM (ver sec. 7.1.2). Esta librería nos permite instanciar un objeto de clase Excel, del cual podremos extraer sus hojas, y dentro de estas leer sus filas y celdas.

En la siguiente imagen, 7.6, se muestra un ejemplo de archivo Excel de índices de autores.

	A	B	C	D	E	F	G	H	I	J	K	L
1	**Nº**	**TITULO**	**AUTOR 1**	**AUTOR 2**	**AUTOR 3**	**AUTOR 4**	**AUTOR 5**	**AUTOR 6**	**TRADUCTOR 1**	**TRADUCTOR 2**	**PP.**	**NOTAS**
2		**1 ABRIL-MAYO 1942**										
3		1 The New Image	Wolfgang Paalen						Robert Motherwell		7	
4		1 Aphorisms	Georg Christoph Lichtenberg (1742-1799)								16	
5		1 Suggestion for an Objective Morality	John Dawson	Charles Givors	Cesar Moro	Wolfgang Paalen	Alice Paalen	Eva Sulzer			17	
6		1 Les nouveaux Sauriens	Charles Givors								20	
7		1 Regionalism in Painting	Edward Renouf								21	
8		1 The Sleeping Woman	Alice Paalen								24	
9		1 Farewell au Surrealisme	Wolfgang Paalen								26	
10		1 Seeing and Showing	Wolfgang Paalen								27	
11		1 Poèmes	César Moro								28	
12		1 L'auto	Charles Givors								29	
13		1 Apercu pour une Morale Objective	Wolfgang Paalen								32	
14		1 L'Image Nouvelle	Wolfgang Paalen								36	Difiere en ciertos pasajes de la versión inglesa.
15		1 (French and English)	Médecine								43	
16		1 à l'Intacolhuatl	Alice Paalen								45	
17		1 Paysage Totémique	Wolfgang Paalen								46	
18		**1 JULIO-AGOSTO 1942**										
19	II	Surprise and Inspiration	Wolfgang Paalen								5	
20	II	Three Tests	Edward Renouf								11	
21	II	About the Origins of the Doric Column and the Guitar-woman	Wolfgang Paalen								14	
22	II	On certain Functions of Modern Paiting	Edward Renouf								19	
23	II	Did Henri Rousseau ever get to Mexico?	Eva Sulzer								26	
24	II	Poème	Alice Paalen								30	
25	II	Xilonen	Charles Givors								31	
26	II	Pierre-Mère	César Moro								34	
27	II	Surprise et Inspiration	Wolfgang Paalen								36	
28	II	Paysage Totémique	Wolfgang Paalen								42	
29	II	The Dialectical Gospel	Wolfgang Paalen								54	
30	II	L'Evangile Dialectique	Wolfgang Paalen								59	
31	**III**	**OTOÑO 1942**										
32	III	Art and Science	Wolfgang Paalen								4	
33	III	A Mary Margaret Miller	Alice Paalen								9	No está en la página 9, sino en una página sin nú
34	III	Nothing but Toys	Gustav Regler								10	
35	III	Notes on Meaning, Conflict and Creation	Edward Renouf								12	
36	III	Downshire Hill	Valentine Penrose								17	
37	III	Au fond du Temps	César Moro								20	
38	III	Prométhée parle	Charles Givors								21	
39	III	Le Grand Malentendu (trad. "Art and Science")	Wolfgang Paalen								22	
40	III	Paysage Totémique	Wolfgang Paalen								27	
41	III	A preface to Parker Tyler's "America's Hallucination"	Henry Miller								32	
42	III	Two Poems	Marian Castleman								39	
43	III	Through the Streetz of my own labyrinth	Anaiz Nin								40	
44	III	The Ghetto, fragment from the Rosy Crucifission	Henry Miller								41	
45	III	Book Reviews	Wolfgang Paalen								44	
46	**IV-V**	**AMERINDIAN NUMBER**										
47	IV-V	The Codices of Azoyu	Alfonso Caso								3	
48	IV-V	Totem Art	Wolfgang Paalen								7	
49	IV-V	Those Hunting Grounds	Gustav Regler								38	
50	IV-V	Tlatilco, Archaic Mexican Art and Culture	Miguel Covarrubias								40	
51	IV-V	The Enigma of Maya Astronomy	Maud Worcester Makemson								47	
52	IV-V	Seals of the Ancient Mexicans	Jorge Enciso								54	
53	IV-V	New Discoveries in the Temple of the Sun in Palenque	Miguel Ángel Fernández								55	
54	IV-V	The Painting in Mexican Codices	Carlos R. Margain Araujo								59	
55	IV-V	Gabriel Vicente Gahona, Mexican Artist of the XIX Century	Francisco Díaz de León								68	
56	IV-V	Birth of Fire	Wolfgang Paalen								71	
57	IV-V	Coricancha	César Moro								73	
58	**VI**	**NUMBER VI**										
59	VI	On the Meaning of Cubism Today	Wolfgang Paalen								4	
60	VI	The Modern Painter's World	Robert Motherwell								9	
61	VI	During the Eclipse	Carter Stone	Wolfgang Paalen							15	
62	VI	Esposition Alice Paalen	Jacqueline Johnson								21	
63	VI	La Venta - Colossal Heads and Jaguar Gods	Miguel Covarrubias								24	
64	VI	The Eye's Journey	Anaiz Nin								34	
65	VI	The Earth	Jacqueline Johnson								37	

Hoja1 | Hoja2 | Hoja3

Figura 7.6: Ejemplo de índice de autores

Los índices de autores poseen un formato estándar de doce columnas (Descripción esquematizada en cap. 4, sec. 4.1). En la primera fila de todos los índices el valor de cada columna es el título del campo al que representan. En este caso, la primera columna (A) es el número de revista al que pertenecen, el segundo campo (B) es el título del fragmento de texto que han publicado los autores que van desde la columna C hasta la H, y que han traducido los de las columnas I y J, y que aparece en la página de la columna K. También hay un último campo auxiliar para posibles notas (L). Si una fila solo posee número (el cual aparece por primera vez) y titulo, se tratará como un nuevo grupo de índice de autores, y el titulo será añadido a este nuevo grupo.

Para los índices de ilustraciones se siguió el mismo proceso que para los de autores. En la siguiente imagen, 7.7, tenemos un ejemplo de ellos.

	A	B	C	D	E
1	**Nº**	**TÍTULO**	**AUTOR 1**	**PP.**	**NOTAS**
2	**I**	**ABRIL-MAYO 1942**			En el índice de la revista aparecen así, a pesar de que en el orden de las páginas están desordenados.
3	I	"What the sailor will say"	John Dawson	23	Oil-painting
4	I	"Figure pandynamique"	Wolfgang Paalen	6	Oil-painting
5	I	"Polarités Majeures"	Wolfgang Paalen	11	Oil-painting
6	I	Woodcut (Original)	Wolfgang Paalen		
7	I	"Paysage errant"	Jean Caroux	14	Oil-painting
8	I	Drawings	Edward Renouf	16; 28	
9	I	Tableau-Poème	Alice Paalen	35	Gouache
10	I	Photographies de la Colombie Britannique	Eva Sulzer	47; 49	
11	**II**	**JULIO-AGOSTO 1942**			
12	II	Photographs	Brassaï	16_17	
13	II	"Portrait of the Eternal"	Manuel Álvarez Bravo	18	Photography
14	II	"Paysage errant"	Jean Caroux	38	Oil-painting
15	II	Brush Drawing	John Dawson	25	
16	II	Drawing	Carlos Mérida	35	
17	II	Drawings, Sculpture	Henry Moore	28_29	
18	II	"Moraines"	Alice Paalen	23	Oil-painting
19	II	Brush-Drawing	Alice Paalen	40	
20	II	Polarités chromatiques	Wolfgang Paalen	4	Oil-painting
21	II	Polarités chromatiques	Wolfgang Paalen	41	Oil-painting
22	II	Woodcut (original) gravé par F. Díaz de León	Wolfgang Paalen	31	
23	II	"Hell Bird"	Edward Renouf	9	Oil-painting
24	II	4 Brush Drawings	Edward Renouf	11_13	
25	II	Drawing	Henri Rousseau (?)	27	
26	II	Photographs	Eva Sulzer	15; 43; 44	
27	**III**	**OTOÑO 1942**			
28	III	Photo	Manuel Álvarez Bravo		
29	III	Sketch for the ballet "Aleko"	Marc Chagall		Water-color
30	III	Sketches for Mobiles	Alexander Calder	9; 16; 26	Brush-drawings
31	III	Photo	Doris Heydn		
32	III	A little window toward the sea	Carlos Mérida		Wax-painting
33	III	A hunting design after a Maya theme	Carlos Mérida		Water-color
34	III	Sculptures and Drawings	Henry Moore		
35	III	Etude d'espace	Robert Motherwell		Gouache
36	III	Rendez-vous des rivières	Alice Paalen		Oil-painting
37	III	Quaternity	Edward Renouf		Oil-painting
38	III	Personnage spatial	Wolfgang Paalen		Oil-painting
39	III	Photo	I. Russell Sorgi		
40	III	Photos	Eva Sulzer	30	
41	**IV-V**	**AMERINDIAN NUMBER**			
42	IV-V	Water-colors and drawings; Collection	Miguel Covarrubias	42; 43_44; 47	
43	IV-V		Rosa Rolanda	66_67	
44	IV-V		Miguel Ángel Fernández	52	
45	IV-V		Florence Arquin		
46	IV-V		Francisco Díaz de León	68	

Hoja1 Hoja2 Hoja3

Figura 7.7: Ejemplo de índice de ilustraciones

Los índices de ilustraciones poseen un formato estándar de cinco columnas (Descripción esquematizada en cap. 4, sec. 4.1). En la primera fila de todos los índices el valor de cada columna es el título del campo al que representan. En este caso, la primera columna (A) es el número de revista al que pertenecen, el segundo campo (B) es el título de la imagen o grupo

de imágenes a las que hace referencia y ha publicado el autor de la columna C, y que aparece o aparecen en la página o páginas de la columna D. En caso de especificarse más de una página, estas serán separadas mediante el símbolo ";". Además, se puede especificar un rango mediante el uso del símbolo "_" entre 2 números. También hay un último campo auxiliar para posibles notas (E). Si una fila solo posee número (el cual aparece por primera vez), titulo y notas, se tratará como un nuevo grupo de índice de ilustraciones, y el titulo será añadido a este nuevo grupo, junto a la nota descriptiva.

Para el índice onomástico se siguió el mismo proceso que para los anteriores. En la siguiente imagen, 7.8, tenemos un ejemplo.

	A	B	C	D	E	F	G	H	I	J	K
1	**AUTOR**	**P.**	**P.**	**P.**	**P.**	**P.**	**P.**	**P.**	**P.**	**P.**	**P.**
2	Álvarez Bravo, Manuel	18 (II)	(III)	17 (IV-V)	23 (IV-V)	26 (IV-V)	40 (VI)				
3	Arquin, Florence	72-73 (IV-V)									
4	Baziotes, William	8 (VI)	(VI)								
5	Brassai	16 (II)	17 (II)								
6	Braque, Georges	(VI)									
7	Cage, Xenia	(VI)									
8	Calder, Alexander	4-5 (III)	9 (III)	16 (III)	20-21 (III)	26 (III)	(VI)				
9	Caroux, Jean	14 (I)	38 (II)								
10	Caso, Alfonso	3 (IV-V)									
11	Castleman, Marian	39 (III)									
12	Chagall, Marc	(III)									
13	Chambi, Martín	73-77 (IV-V)									
14	Cordry, Donald	51 (VI)									
15	Covarrubias, Miguel	16-17 (IV-V)	40 (IV-V)	43 (IV-V)	44 (IV-V)	47 (IV-V)	24 (VI)	(VI)			
16	Dawson, John	17 (I)	23 (I)	25 (II)							
17	Díaz de León, Francisco	68 (IV-V)	68 (IV-V)	(VI)							
18	Enciso, Jorge	54 (IV-V)	54 (IV-V)								
19	Fernández, Miguel Ángel	52 (IV-V)	55 (IV-V)								
20	Fett, William	(VI)									
21	Givors, Charles	20 (I)	29 (I)	31 (II)	21 (III)	48 (VI)					
22	Greville-Healey, Giles	(VI)									
23	Hayter, William	(VI)									
24	Helion, Jean	(VI)									
25	Heydn, Doris	(III)	(VI)								
26	Hofer, Evelyn	(VI)									
27	Holtzman, Harry	(VI)									
28	Johnson, Jacqueline	21 (VI)	37 (VI)								
29	Lichtenberg, Georg Christoph	16 (I)									
30	Limón Aragón, Luis	22 (IV-V)	36 (IV-V)								
31	Margain Araujo, Carlos R.	59 (IV-V)									
32	Makemson, Maud Worcester	47 (IV-V)									
33	Matta	(VI)									
34	Médecine	43 (I)									
35	Mérida, Carlos	35 (II)	(III)	(VI)							
36	Miller, Henry	32 (III)	41 (III)								
37	Moore, Henry	28 (II)	29 (II)	(III)	12-14 (VI)						
38	Moro, César	28 (I)	34 (II)	20 (III)	73 (IV-V)	45 (VI)					
39	Motherwell, Robert	(III)	9 (VI)	(VI)							
40	Nin, Anais	40 (III)	34 (VI)								
41	Onslow-Ford, Gordon	(VI)									
42	Paalen, Alice	24 (I)	35 (I)	45 (I)	23 (II)	30 (II)	40 (II)	(III)	9 (III)	16 (IV-V)	(VI)
43	Paalen, Wolfgang	6 (I)	7 (I)	11 (I)	26 (I)	27 (I)	32 (I)	36 (I)	46 (I)	4 (II)	5 (II)
44	Penrose, Valentine	17 (III)									
45	Picasso, Pablo	(VI)									
46	Pollock, Jackson	(VI)									
47	Regler, Gustav	10 (III)	38 (IV-V)	41 (VI)	43 (VI)						
48	Renouf, Edward	16 (I)	21 (I)	28 (I)	9 (II)	11 (II)	12 (II)	13 (II)	19 (II)	(III)	12 (III)
49	Reuter, Walter	22 (VI)									
50	Reynal, Jeanne	(VI)									
51	Rolanda, Rosa	66 (IV-V)	67 (IV-V)	72-73 (IV-V)	(VI)						
52	Rousseau, Henri	27 (II)									
53	Russell Sorgi, I.	(III)									
54	Smith, David	(VI)									
55	Stone, Carter	15 (VI)									

Hoja1 Hoja2 Hoja3

Figura 7.8: Ejemplo de índice onomástico

El índice onomástico, a diferencia de los otros, posee un formato estándar, pero de columnas variables (Descripción esquematizada en cap. 4, sec. 4.1). En la primera fila de todos los índices el valor de cada columna es el título del campo al que representan. En este caso, la primera columna (A) es el nombre de un autor que aparece en la revista y las siguientes columnas (el número de estas es variable) representan las páginas y el número de revista en el que aparece.

A continuación se muestra un ejemplo de uso de la librería Apache POI (ver cap. 2, sec. 2.2) con la que se extrajeron los datos de los archivos Excel.

```
public ArrayList<OnomasticIndexDto> extractOnomasticIndexData(MultipartFile onomasticFile) throws
    IOException {

  ArrayList<OnomasticIndexDto> data = new ArrayList<>();
  InputStream file = onomasticFile.getInputStream();
  XSSFWorkbook worbook = new XSSFWorkbook(file);
  XSSFSheet sheet = worbook.getSheetAt(0);
  int rows = sheet.getLastRowNum();

  Row row;
  Cell cell;

  for (int r = 1; r <= rows; r++) {
    row = sheet.getRow(r);
    OnomasticIndexDto onomasticIndexDto = new OnomasticIndexDto();
    if (row == null) {
      break;
    } else {
      onomasticIndexDto.setAuthor(row.getCell(0, Row.RETURN_BLANK_AS_NULL).toString());
      for (int c = 1; c < row.getLastCellNum(); c++) {
        cell = row.getCell(c, Row.RETURN_BLANK_AS_NULL);
        if (cell != null)
          onomasticIndexDto.addPage(cell.toString());
      }
    }
    data.add(onomasticIndexDto);
  }

  return data;

}
```

Para la extracción de datos del estudio (ver cap. 4, sec. 4.1), se utilizó la misma librería que para los índices, pero implementando las funcionalidades que ofrece para Word.

7.3.2 Implementación de gestión de archivos

Como se puede ver en el análisis (ver cap. 4, sec. 4.1), cada revista está formada por números de revista. Estos están disponibles en formato PDF. El número de ellos es variable según cada revista. Además, se necesita una imagen de portada para cada uno.

Como requisito de nuestra aplicación (ver cap. 4, sec. 4.3.1), debemos disponer de un mecanismo de subida, borrado y actualización de archivos a nuestro servidor.

Para ello hemos implementado un servicio, FileStorageService [19], el cual se encarga de almacenar los PDF con el contenido de los números y las imágenes de portada de estos.

Para evitar el borrado por sustitución de documentos con el mismo nombre, se han organizado los documentos por directorios, donde cada revista posee el suyo propio, y cuyo nombre será el identificador de la revista, asegurándonos así de que sea único. Dentro del directorio de cada revista, los números y las imágenes serán nombrados con el número de revista, el cual es único también.

Además, este servicio implementa funciones de eliminación de documentos y directorios que son necesarias cuando se elimina o actualiza una revista.

El siguiente fragmento de código representa el método de almacenamiento de un archivo y su nomenclatura personalizada.

```
1  public String storeFile(MultipartFile file, String customName, String folder) throws
       FileStorageException {
2
3      Path fileStorageLocation = Paths.get("src/main/resources/static/journals/" +
         folder).toAbsolutePath().normalize();
4
5      try {
6
7        Files.createDirectories(fileStorageLocation);
8
9      } catch (Exception ex) {
10
11       throw new FileStorageException("Could not create the directory where the uploaded files will be
         stored.", ex);
12     }
13
14     // Normalize file name
15     String fileName = StringUtils.cleanPath(customName);
16
17     try {
18
19       // Check if the file's name contains invalid characters
20       if (fileName.contains("..")) {
21
22         throw new FileStorageException("Sorry! Filename contains invalid path sequence " + fileName);
23
24       }
25
26       // Copy file to the target location (Replacing existing file with the same name)
27       Path targetLocation = fileStorageLocation.resolve(fileName);
28
29       Files.copy(file.getInputStream(), targetLocation, StandardCopyOption.REPLACE_EXISTING);
30
31       return fileName;
32
33     } catch (IOException ex) {
34
35       throw new FileStorageException("Could not store file " + fileName + ". Please try again!", ex);
36
37     }
38
39 }
```

Como control de validación para el formulario donde se adjuntan estos documentos fue necesario crear una anotación personalizada (@FileExist) para verificar que el archivo había sido subido. El siguiente código muestra cómo se llevó a cabo su implementación:

```
@Target({ FIELD, ANNOTATION_TYPE })
@Retention(RUNTIME)
@Constraint(validatedBy =
    com.surhi.journals.utilities.FileExistValidator.class)
@Documented
public @interface FileExists {

  String message() default "";

  Class<?>[] groups() default {};

  Class<? extends Payload>[] payload() default {};
}

@Component
class FileExistValidator implements
    ConstraintValidator<com.surhi.journals.utilities.FileExists,
    MultipartFile> {

  public FileExistValidator() {
  }

  @Override
  public void initialize(com.surhi.journals.utilities.FileExists
    constraintAnnotation) {}

  @Override
  public boolean isValid(MultipartFile value,
    ConstraintValidatorContext context) {

    if (value.getOriginalFilename().equals(""))
      return false;
    else
      return true;

  }
}
```

7.4 Seguridad

Para la implementación de la seguridad de nuestra aplicación se empleó Spring Security (ver sec. 7.1.2). Por defecto, Spring Boot ya ofrece un método de seguridad básico. En el proyecto se sobrescribió para la implementación de nuestro propio servicio de usuarios. Esto nos permitió también emplear nuestra propia interfaz personalizada de inicio de sesión acorde al diseño de la página.

Para llevar a cabo esto, como primer paso se crearon los componentes necesarios para la persistencia de los datos de los usuarios. Acto seguido se creó el servicio encargado de autenticar usuarios, el cual implementa la interfaz UserDetailsService que se utiliza para recuperar datos relacionados con el usuario. Tiene un método denominado loadUserByUsername() que se puede sobrescribir para personalizar el proceso de búsqueda del usuario.

Una vez hecho esto, mediante un archivo de configuración (ver código 7.4) sobrescribimos la seguridad básica de Spring añadiendo nuestra propia página de inicio de sesión y delegando la autenticación a nuestro servicio.

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

  @Autowired
  UserService userDetailsService;

  BCryptPasswordEncoder bCryptPasswordEncoder;

  String[] resources = new String[] { "/css/**", "/images/**", "/journals/**", "/js/**" };

  @Override
  protected void configure(HttpSecurity http) throws Exception {

    http
      .authorizeRequests()
      .antMatchers(resources).permitAll()
      .antMatchers("/revista*", "/").permitAll()
        .anyRequest().authenticated()
        .and().formLogin().loginPage("/login").permitAll()
        .defaultSuccessUrl("/admin").failureUrl("/login?error=true")
        .usernameParameter("username").passwordParameter("password")
        .and().logout().permitAll().logoutSuccessUrl("/login?logout");

  }

  @Bean
  public BCryptPasswordEncoder passwordEncoder() {

    bCryptPasswordEncoder = new BCryptPasswordEncoder(4);
    return bCryptPasswordEncoder;

  }

  @Autowired
  public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

    auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());

  }
}
```

7.5 Controladores e interfaz

En este apartado se introducen aspectos relevantes a la interfaz y los controladores.

Los controladores son los encargados de responder a eventos por parte del usuario e invocar peticiones al modelo cuando se hace alguna solicitud sobre la información. En nuestra aplicación se han implementado tres controladores para interactuar entre las vistas y el modelo (ver cap. 6, sec. 6.3). El primero de ellos, ”AdminController” es el encargado de interactuar entre las vistas del gestor de contenidos y el modelo. Aquí ocurren todas las peticiones relacionadas con la adición, modificación o borrado de datos. El segundo controlador, ”HomeController” es el encargado de gestionar los eventos lanzados desde el lado cliente en la página web accesible para todos los usuarios. Solo realiza peticiones de lectura de datos contra el modelo. Por último, se implementó un controlador encargado de las peticiones de inicio de sesión (”LoginController”). Una vez que el controlador recibe una respuesta del modelo, este delega a los objetos de la vista la información necesaria para que estos muestren la interfaz de usuario.

Para la implementación de la estructura de estas interfaces, utilizamos HTML (ver cap. 2, sec. 2.1) y Thymeleaf (ver cap. 2, sec. 2.2), que nos permitieron la creación de un código sencillo y bien formateado que interpreta los datos del modelo recibidos, ocupándose así de la dinamización de las páginas (ver ejemplo 7.5). Cabe destacar que se han creado plantillas que permiten la reutilización de código como en el caso de algunos componentes base como el menú de navegación y el pie de página. Se utilizaron otros lenguajes como CSS (ver cap. 2, sec. 2.1), para dotar de estilo las páginas, y JavaScript (ver cap. 2, sec. 2.1) para añadir mejoras en estas.

El diseño visual de las interfaces sigue dos patrones. Uno para las vistas de la web accesible por todos los usuarios y otro para el gestor de contenidos. El diseño de la interfaz de usuario es simple, con una organización de contenidos estructurada y de fácil acceso, tal y como pedía el Producrt Owner. Tampoco se complicó el desarrollo de la interfaz introduciendo animaciones ni recargas innecesarias. El diseño de la interfaz de gestión de usuarios (Ver captura 7.9), principalmente implementado con componentes Bootstrap (ver cap. 2, sec. 2.2) como tablas y botones sigue un patrón de estructura y colores en todas sus páginas (Rojo para borrar, azul para guardar, verde para añadir y blanco para modificar).

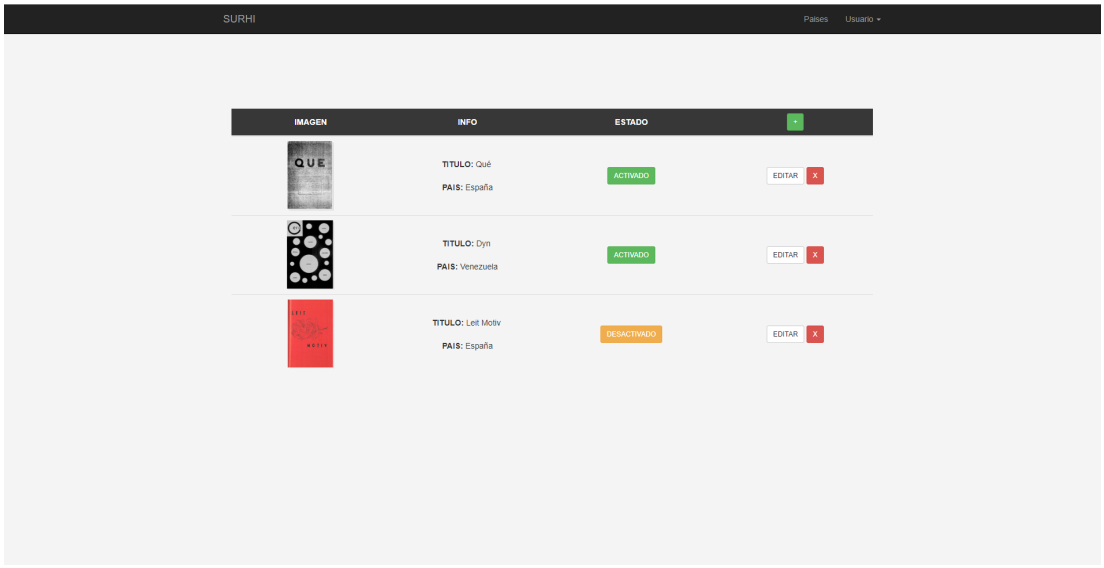

Figura 7.9: Interfaz de usuario del gestor de contenidos

Ejemplo de código HTML y Thymeleaf:

```
<section class="text-center row" th:unless="${#lists.isEmpty(journals)}">
  <div class="col-md-6 col-lg-3 journal" th:each="j : ${journals}">
    <div class="card">
      <div class="card-img-top">
        <a href="#" th:href="@{/revista(id=${j.journal_id})}">
          <img class="ind-journal-img" th:src="@{${'journals/' + j.journal_id + '/' +
    j.number[0].img}}" alt="Responsive image"/>
        </a>
            </div>
            <div class="card-body">
        <p><span class="card-body-title" th:text="${j.title != null} ? ${j.title} : ''"></span></p>
        <p><span class="card-body-country" th:text="${j.country.name != null} ? ${j.country.name} :
    ''"></span></p>
      </div>
    </div>
  </div>
</section>
```

Capítulo 8

Pruebas

En este capítulo se muestran las pruebas realizadas.

8.1 Pruebas unitarias

En programación, una prueba unitaria es una forma de comprobar el correcto funcionamiento de una unidad de código. Esto sirve para asegurar que cada unidad funcione correctamente por separado. Las pruebas se han realizado sobre los servicios de la aplicación intentando cubrir los máximos casos posibles de forma que cada caso sea independiente del resto.

La configuración de las pruebas con Spring Boot (ver cap. 2, subsec. 2.2.1) simplifica el código necesario para la puesta en marcha de las mismas. Para la realización de estas se ha configurado una base de datos H2 (ver cap. 2, sec. 2.4) en memoria mediante un archivo de configuración ".properties" ubicado en el directorio "resources" dentro de la rama pruebas.

Para que los test sean independientes, se ha encapsulado cada uno de ellos en una transacción (Mediante la anotación @Transactional). Esta transacción se revertirá al final del método de prueba, independientemente de su resultado, eliminando todos los registros de la base de datos.

A continuación, se muestran dos ejemplos de prueba, uno de ellos comprobando distintas funcionalidades con éxito, y el otro forzando un error.

El primero de ellos (Ver código 8.1) se trata de una prueba sobre el servicio de revistas (JournalService (cap. 6, sec. 6.5)). En ella creamos distintas revistas con sus datos mínimos necesarios y comprobamos que las funciones de búsqueda de todos los elementos, de búsqueda por país y búsqueda por identificador funcionan correctamente. Una vez hecho esto, "desactivamos" (ver requisito en cap. 4, subsec. 4.3.1) una revista y comprobamos que la búsqueda por activas devuelve los resultados esperados.

El segundo caso (Ver código 8.1) muestra el salto de una excepción personalizada cuando se trata de buscar un numero de una revista que no existe en el servicio de números de revista (JournalNumberService (ver cap. 6, sec. 6.5)).

```
1
2  @Test
3  @Transactional
4  public void journalTest() {
5
6    Country c1 = countryService.save(new Country("Espana"));
7    Country c2 = countryService.save(new Country("Argentina"));
8
9    Journal j1 = journalService.save(new Journal("One",c1));
10   Journal j2 = journalService.save(new Journal("Two",c2));
11   Journal j3 = journalService.save(new Journal("Three",c2));
12   Journal j4 = journalService.save(new Journal("Four",c1));
13   Journal j5 = journalService.save(new Journal("Five",c1));
14
15   assertEquals(5,journalService.findAll().size());
16
17   assertEquals(3,journalService.findByCountry(c1).size());
18
19 assertEquals(j1,journalService.findById(j1.getJournal_id()).get());
20
21   journalService.activate(j1);
22
23   assertEquals(1,journalService.findByActive().size());
24
25 }
```

```
1
2  @Test(expected = JournalNumberNotFoundException.class)
3  @Transactional
4  public void JournalNumberNotFoundExceptionTest() throws
       JournalNumberNotFoundException {
5
6    Country c1 = countryService.save(new Country("Espana"));
7
8    Journal j1 = journalService.save(new Journal("One",c1));
9
10   JournalNumber jn1 = journalNumberService.save(new
       JournalNumber(j1, "1", "leitmotiv.png"));
11   JournalNumber jn2 = journalNumberService.save(new
       JournalNumber(j1, "2", "cero.png"));
12
13   journalNumberService.findByJournalNumber(j1, "3");
14
15 }
```

8.2 Pruebas de sistema

Las pruebas de sistema tienen como objetivo comprobar la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica. Dan una visión muy similar a su comportamiento en el entorno de producción. Se llevo cabo la realización de estas de manera manual sobre la aplicación web completa.

Capítulo 9

Conclusiones

En este capítulo final de la memoria se presenta la situación final del proyecto, las lecciones aprendidas, la relación con las competencias de la titulación en general y la mención en particular y posibles líneas futuras.

9.1 Competencias de la titulación

Algunas de las competencias de la titulación que se han completado con el desarrollo de este proyecto han sido:

- La capacidad para la resolución de problemas: Un trabajo puede ser planteado como un problema. En este caso el desarrollo de la aplicación.
- La capacidad de organización y planificación: A lo largo del desarrollo de la aplicación se ha ido planificando y organizando las tareas en iteraciones.
- La capacidad de comunicación con clientes reales: Durante el desarrollo del proyecto hemos estado en un entorno real con un cliente.

9.2 Competencias de la mención

Algunas de las competencias de la mención en Ingeniería del Software que se han completado con el desarrollo de este proyecto han sido:

- Extracción de requisitos satisfactoriamente a partir de reuniones con del cliente.
- Análisis de una aplicación, diseño, implementación y desarrollo de esta.
- Utilización de metodologías actuales

9.3 Lecciones aprendidas

Se ha aprendido a gestionar un proyecto en su totalidad, desde el principio hasta el final de su desarrollo. Se ha consolidado el uso de metodologías ágiles. Además, durante el transcurso del proyecto se han encontrado numerosos errores y se ha aprendido a solucionarlos, a intentar no cometerlos otra vez y a prevenirlos.

9.4 Situación actual de la aplicación y posibles líneas futuras

La solución final, que se puede observar en el apéndice A, se trata de un espacio web al que cualquier usuario de la red puede acceder y visualizar todas las revistas y sus respectivos números, así como los datos de estudio de estas (Estudio de la revista, índices de autores, índice onomástico e índices de ilustraciones). Además, se ha realizado también la implementación de un gestor, que solo puede ser accedido por un administrador mediante autenticación, en el cual se administran todos los contenidos.

Esta aplicación web ha sido diseñada para ser utilizada por personal de la facultad de Filología de la Coruña (ver cap. 1 sec. 1.2). En la última reunión, en la que se realizó una demo completa de la aplicación, el Product Owner (ver cap. 3 sec. 3.2) y su equipo dio el visto bueno a la aplicación. Esta, por tanto, está cerca de integrarse en un entorno real de producción. Para ello, deben concretarse detalles referentes al despliegue de esta en una última reunión no realizada todavía.

Por tanto, el estado final del proyecto es exitoso, ya que se han conseguido llevar a cabo los objetivos de la aplicación, cumpliendo las especificaciones del proyecto y su aprobación por parte del cliente.

A pesar de haberse cumplido estos objetivos, se podrían desarrollar cambios o mejoras a tener en cuenta para el futuro de la aplicación. Estas no se han llevado a cabo debido a que el departamento de investigación para el que se ha desarrollado esta aplicación, no consideró necesaria su implementación.

A continuación, se exponen algunas de estas ideas:

- Internalización: Permitir la selección de más idiomas en el espacio web. Para ello sería necesario la traducción de contenidos por parte del cliente.

- Descarga de contenidos: Permitir al administrador descargar los archivos previamente subidos al servidor como los números de las revistas (ver cap. 7 subsec. 7.3.2). También se podría añadir una funcionalidad que permitiese exportar los datos de índices a un documento Excel con la misma estructura que los utilizados por ellos (ver cap. 4 sec. 4.1).

Apéndices

Apéndice A

Solución final

Se han incluido las vistas mas representativas como ejemplo de uso de la aplicación o que contienen alguna información relevante para el usuario.

A.1 Gestor

El gestor de contenidos sigue un patrón de estructura y colores en todas sus páginas (Rojo para borrar, azul para guardar, verde para añadir y blanco para modificar).

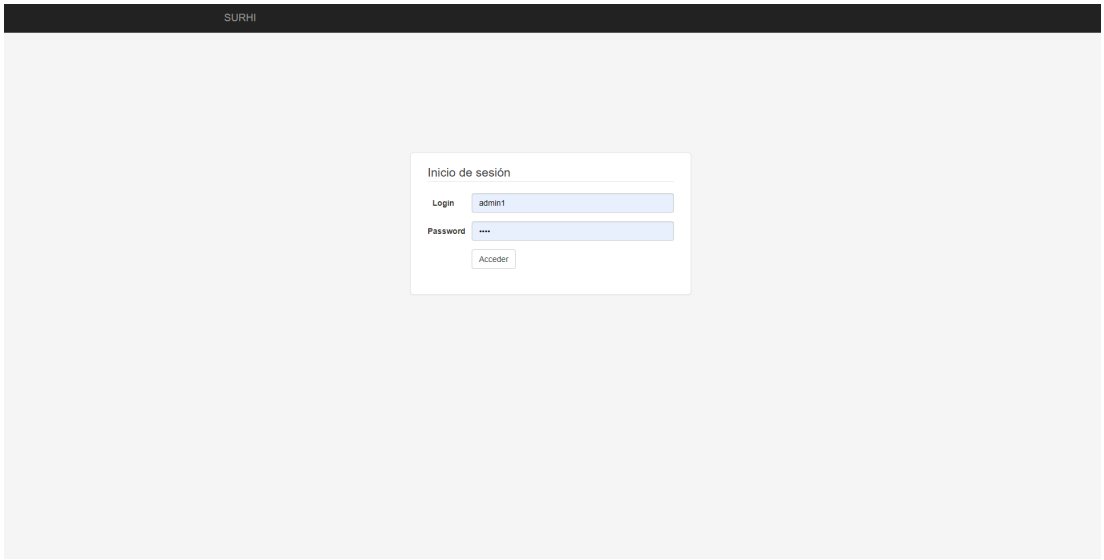

Figura A.1: Formulario de acceso a página de administración. Si las credenciales son correctas accedes a la página inicial de la administración (A.2).

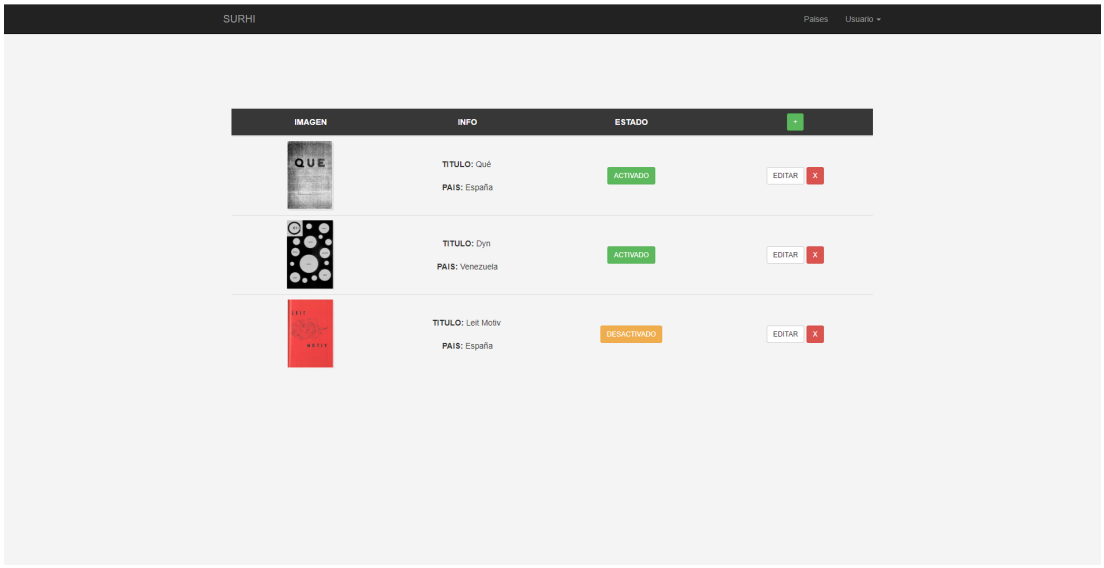

Figura A.2: Página principal de la administración. Se muestran todas las revistas disponibles con sus datos representativos y un menú en la parte superior en el que accedemos al gestor de países (A.6) y a las opciones de usuario (A.3) . En la tercera columna mediante un botón podremos activar o desactivar la revista para su visualización en la página. El botón verde en la esquina superior derecha de la tabla nos lleva a la vista de añadir una revista nueva (A.7). En la cuarta columna de la tabla se sitúan los botones de actualización (A.9) y eliminación de revista (A.4).

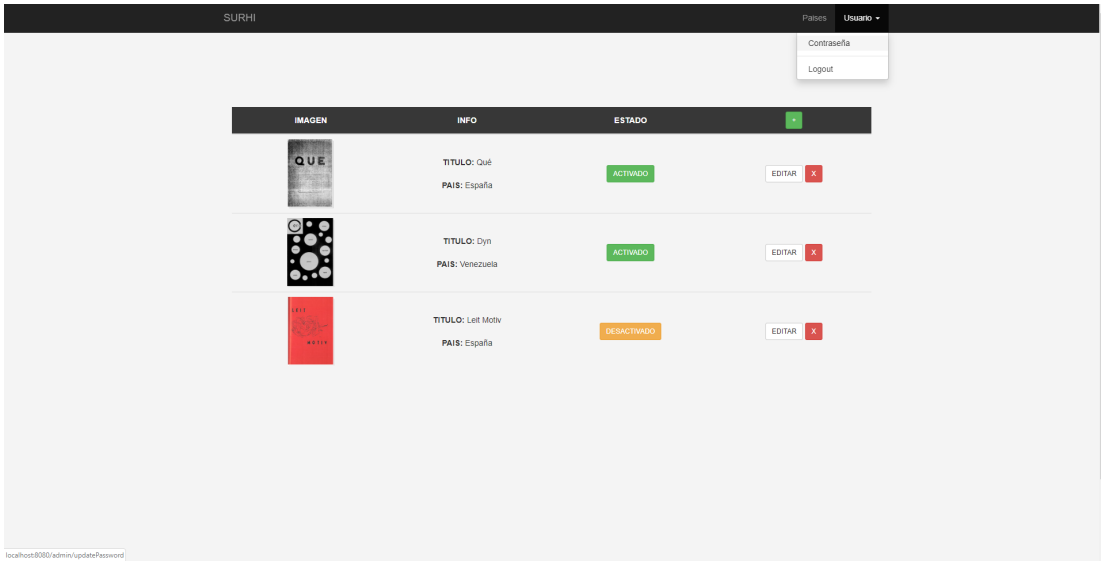

Figura A.3: Vista de la página de administración con opciones de Logout y cambio de contraseña desplegadas.

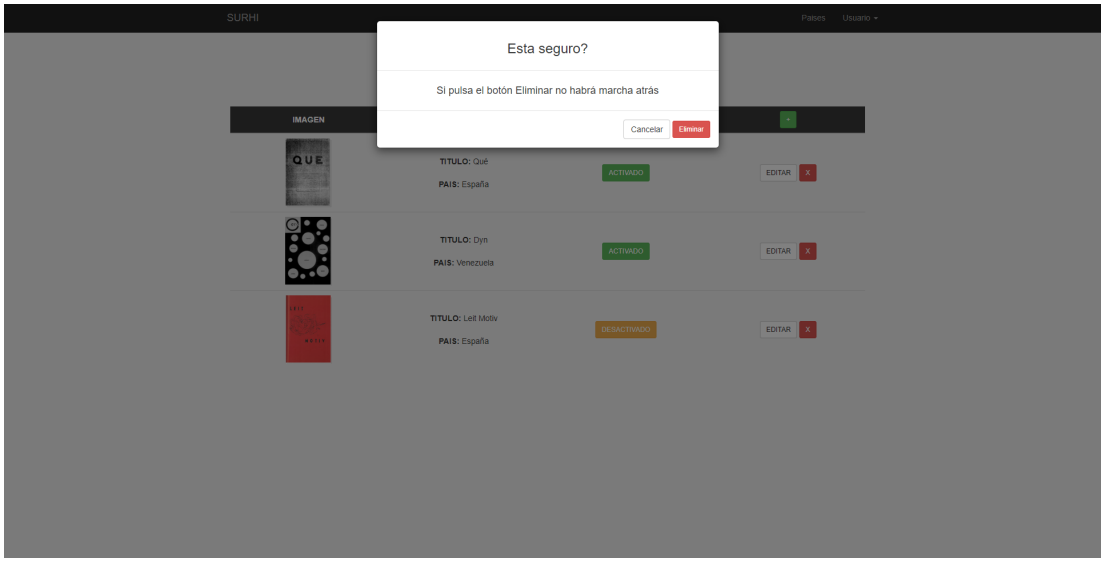

Figura A.4: Ventana desplegable para confirmar borrado de revista. Todos los elementos que contiene serán también eliminados.

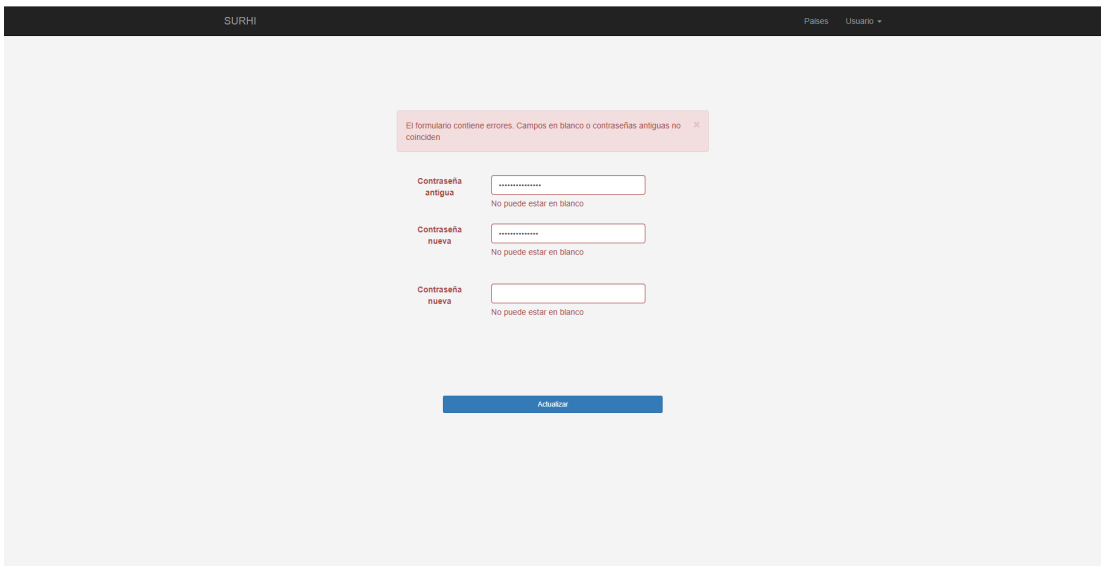

Figura A.5: Página de actualización de contraseña. Si las credenciales son incorrectas despliega un error.

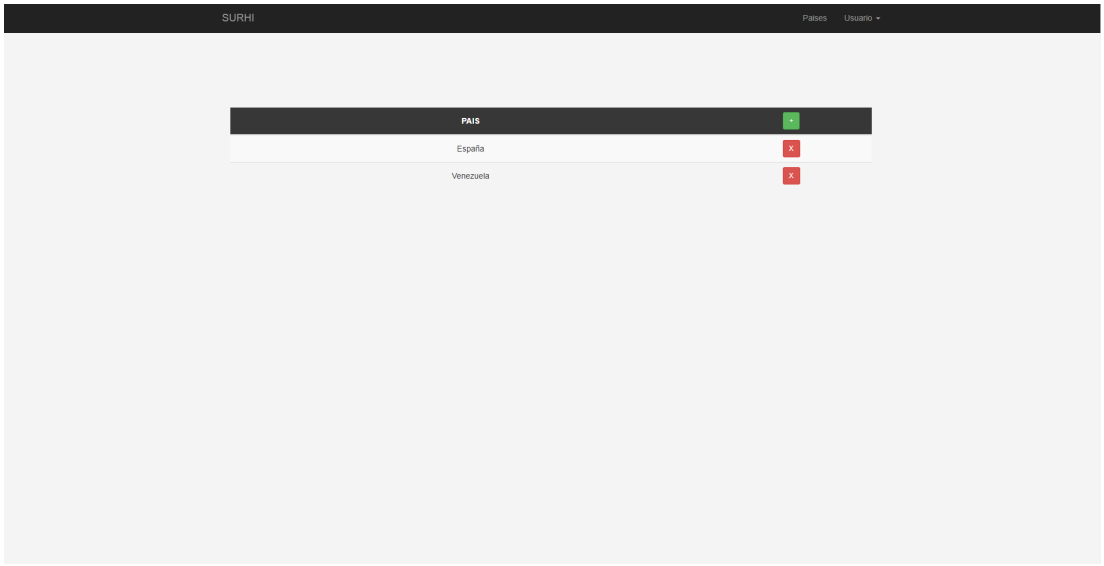

Figura A.6: Página de administración de países. El botón verde despliega la vista de adición de país. El botón rojo el de borrado. Un país no puede ser eliminado si alguna revista lo referencia.

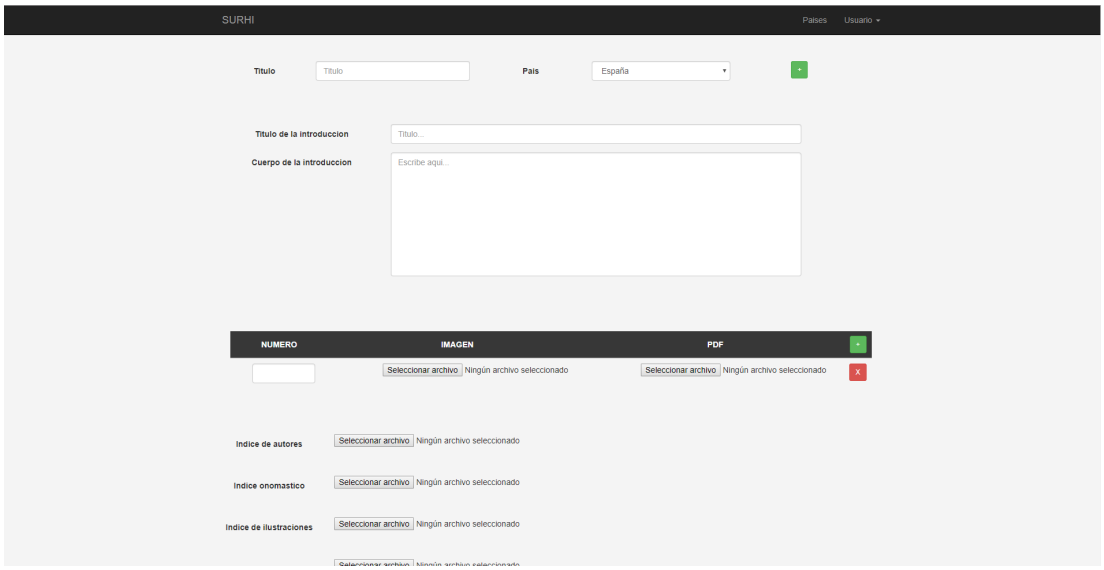

Figura A.7: Formulario de creación de revista. Todos los campos son obligatorios a excepción del índice de ilustraciones (ver A.8). Para añadir un numero debe pulsarse el botón verde de la esquina superior derecha de la tabla. Para eliminarlos se utilizarán los botones rojos.

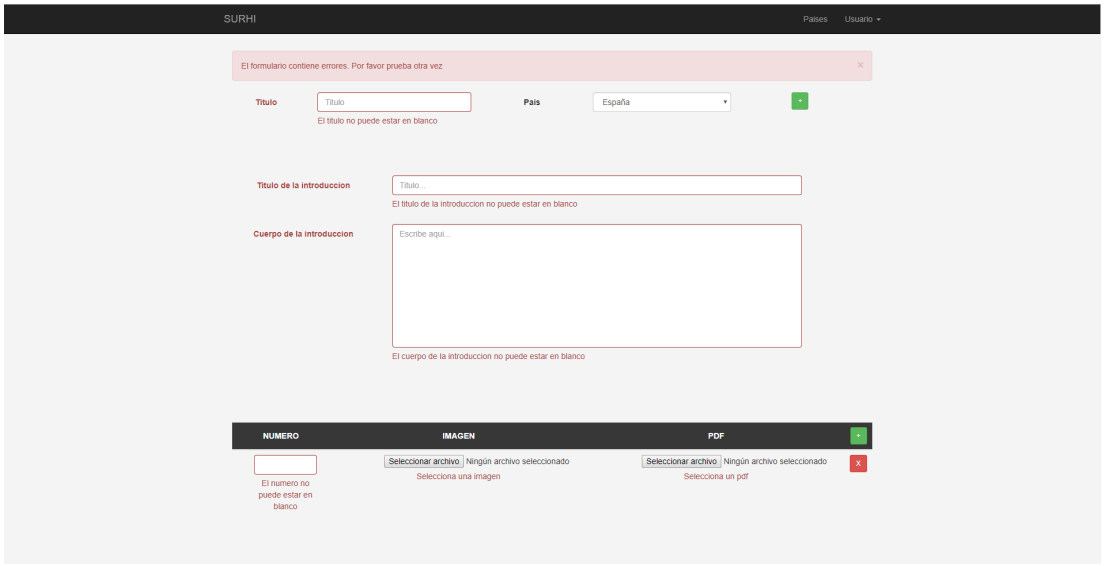

Figura A.8: Vista de error al crear una revista.

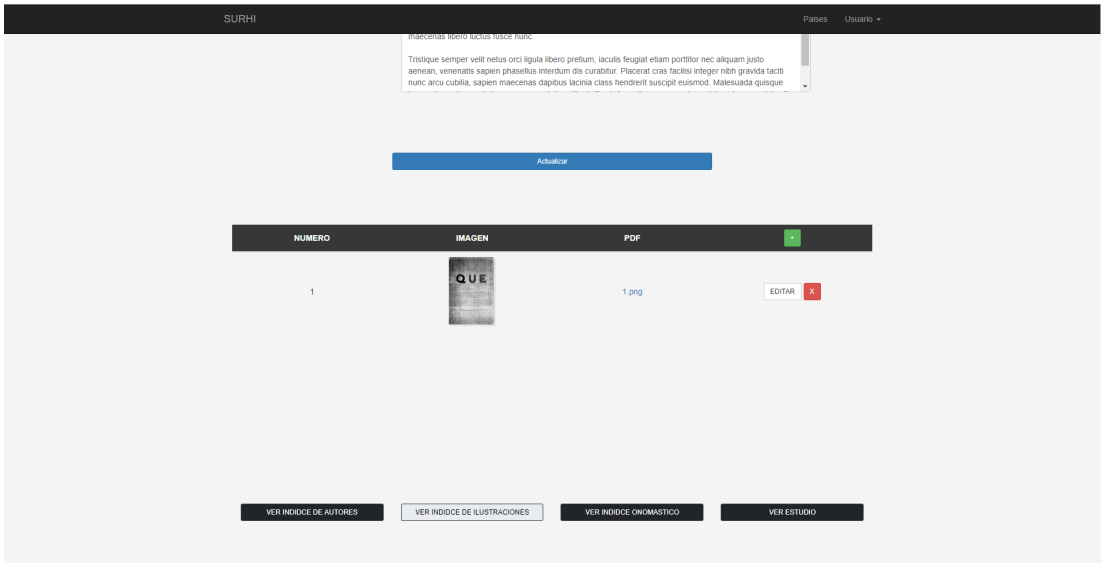

Figura A.9: Vista de actualización de una revista. Podremos actualizar sus metadatos y visualizar sus números de revistas, los cuales podremos eliminar previa confirmación (botón rojo) o actualizar (botón blanco). También se mostrarán enlaces a todos los índices y el estudio. (Ver acceso a índice de autores en fig. A.10).

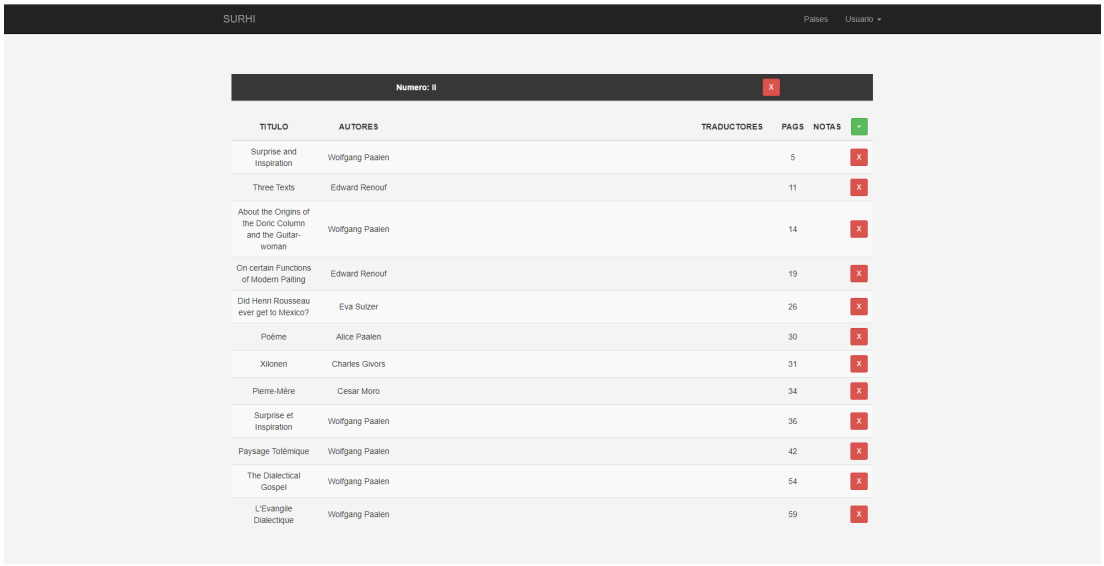

Figura A.10: Vista de índices de autores, en ella podremos crear grupos de índices o añadir líneas a estos con los botones verdes. Los botones rojos servirán de eliminación previa confirmación (ver fig. A.11).

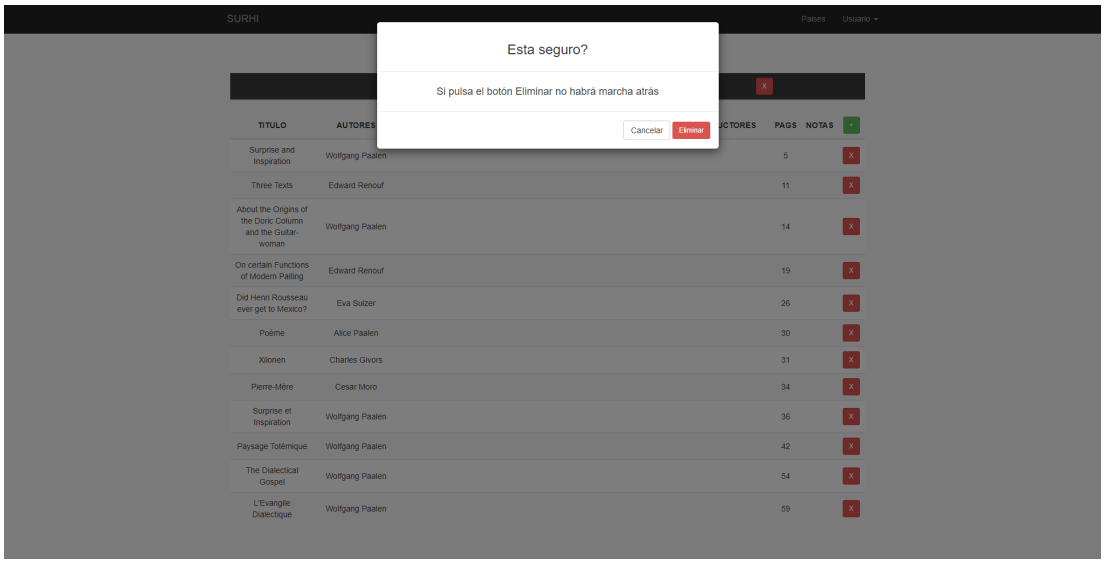

Figura A.11: Ventana de confirmación de borrado de un índice de autores.

A.2 Página

La página sigue el mismo patrón de diseño para todos sus contenidos.

Figura A.12: Vista principal de página.

Figura A.13: Vista de las revistas disponibles.

Lista de acrónimos

PDF *Portable Document Format.*

TFG *Trabajo Fin de Grado.*

HTML *Hypertext Markup Language.*

CSS *Cascading Style Sheets.*

SQL *Structured Query Language.*

XML *Extensible Markup Language.*

MVC *Model View Controller.*

POI *Poor Obfuscation Implementation.*

IDE *Integrated Development Environment.*

JDT *Java Development Toolkit.*

ORM *Object Relational Mapping.*

JPA *Java Persistence API.*

POM *Project Object Model.*

CRUD *Create, Read, Update and Delete.*

DTO *Data Transfer Object.*

SURHI *Surrealismo hispanoamericano.*

Glosario

Framework Software que ofrece una infraestructura (conjunto estandarizado de conceptos, prácticas y criterios) para la creación de otros programas.

Open Source Cualquier programa cuyo código fuente está disponible para su uso o modificación según lo consideren los usuarios u otros desarrolladores.

Índice onomástico Índice donde aparece el nombre de personas mencionadas en el texto, por orden alfabético con el número de las páginas donde aparecen.

Word Programa informático orientado al procesamiento de textos.

Excel Programa de hojas de cálculo y una herramienta avanzada de análisis y visualización de datos.

Bibliografía

[1] ¿qué es java y para qué es necesario? [En línea]. Disponible en: https://www.java.com/es/download/faq/whatis_java.xml

[2] Javascript a fondo. [En línea]. Disponible en: https://desarrolloweb.com/javascript/

[3] Qué es HTML. [En línea]. Disponible en: https://codigofacilito.com/articulos/que-es-html#targetText=%E2%80%9CHTML%20es%20un%20lenguaje%20de,Lenguaje%20de%20Marcas%20de%20Hipertexto%E2%80%9D.

[4] CSS tutorial. [En línea]. Disponible en: https://www.w3schools.com/css/

[5] SQL tutorial. [En línea]. Disponible en: https://www.w3schools.com/sql/

[6] Spring projects. [En línea]. Disponible en: https://spring.io/projects/spring-boot

[7] a. B. M. O. contributors, Jacob Thornton. Introduction. [En línea]. Disponible en: https://getbootstrap.com/docs/4.0/getting-started/introduction/

[8] Tutorial: Using thymeleaf. [En línea]. Disponible en: https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html#arrays

[9] Apache POI - the java API for microsoft documents. [En línea]. Disponible en: https://poi.apache.org/

[10] E. F. Inc. The platform for open innovation and collaboration | the eclipse foundation. [En línea]. Disponible en: https://www.eclipse.org/

[11] Maven – welcome to apache maven. [En línea]. Disponible en: https://maven.apache.org/

[12] Git. [En línea]. Disponible en: https://git-scm.com/

[13] The first single application for the entire DevOps lifecycle. [En línea]. Disponible en: https://about.gitlab.com/

[14] MySQL. [En línea]. Disponible en: https://www.mysql.com/

[15] H2 database engine. [En línea]. Disponible en: https://www.h2database.com/html/main.html

[16] ¿qué es scrum? [En línea]. Disponible en: https://www.scrum.org/resources/blog/que-es-scrum

[17] Spring data JPA - reference documentation. [En línea]. Disponible en: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/

[18] How to read excel files in java using apache POI. [En línea]. Disponible en: https://www.callicoder.com/java-read-excel-file-apache-poi/

[19] Spring boot file upload / download rest API example. [En línea]. Disponible en: https://www.callicoder.com/spring-boot-file-upload-download-rest-api-example/