uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Informática
Curso 2022-2023

Trabajo Fin de Grado

“Desarrollo de una aplicación web para un módulo de valoraciones de servicios de OPAC (Online Public Access Catalog)”

Alejandro Cebrecos Largo

Tutor

Alejandro Rey López

Leganés, septiembre 2023

RESUMEN

Esta Memoria se corresponde con el trabajo realizado sobre el módulo de valoraciones añadido a una aplicación de OPAC (Online Public Access Catalog). Esta aplicación web se llama ODA y pertenece a la empresa Baratz Servicios de Teledocumentación. El objetivo principal del proyecto es brindar a los usuarios la posibilidad de acceder directamente a opiniones de otros usuarios sobre los distintos productos de una biblioteca para mejorar su experiencia con la aplicación.

Tras realizar un estudio analizando las mejores tecnologías actuales para desarrollar aplicaciones web similares, se explican las tecnologías utilizadas en la aplicación. Está desarrollada en Java, con el framework Spring. La base de datos utilizada para guardar toda la información de las valoraciones es MariaDB. Los usuarios de la aplicación son tres: usuarios anónimos, que pueden valorar también los productos; usuarios autenticados, que disponen de una página de perfil y administrador de la biblioteca, que puede decidir si las nuevas valoraciones añadidas son apropiadas o no.

Una parte del desarrollo del módulo se corresponde con su planificación, en la que se diseñan requisitos acorde a las necesidades de las bibliotecas y los usuarios. Además, el proyecto se desarrolla siguiendo una metodología ágil, utilizando scrum como marco de trabajo y Redmine como herramienta de gestión de proyectos.

Palabras clave: Biblioteca; Valoraciones; Tecnología;

AGRADECIMIENTOS

Quiero mostrar mi agradecimiento hacia todas las personas que han estado a mi lado y que me han ayudado durante la realización de este proyecto.

Muchas gracias a mi familia por estar siempre apoyándome y aconsejándome hasta en los momentos más difíciles. Muchas gracias también a todos los profesores de la universidad que han puesto su granito de arena para enseñarme todo lo que he aprendido, en especial a mi tutor, Alejandro Rey. Por último, muchas gracias a todos los compañeros de Baratz que me han aconsejado y ayudado en las partes más complicadas de este trabajo, así como en mis primeros pasos en el mercado laboral.

Gracias por todo.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1 Contexto

Antes de empezar las asignaturas del segundo cuatrimestre del último curso del grado, acordé con la empresa Baratz la realización de mis prácticas curriculares. Después de un mes en el que pude comprobar cómo es un entorno real de trabajo, tras realizar un proyecto pequeño con ayuda de mis compañeros de trabajo, decidí seguir con las prácticas hasta el mes de agosto, formando parte del equipo de desarrollo Java, encargado del Back-end de una de sus aplicaciones.

Baratz es una empresa que se dedica a crear tecnología para las bibliotecas, con el objetivo de ayudarlas, evitar que los últimos avances tecnológicos acaben con ellas y, sobre todo, conectarlas con las personas. En total, más de 3000 bibliotecas en el mundo confían en Baratz, incluyendo el 80% de las bibliotecas públicas españolas, para proporcionarles el software de gestión bibliotecaria. Este producto se llama Absys.

La plataforma en la que estoy trabajando desde el inicio de mis prácticas se conoce como ODA (Opac Discovery by Absys). Desde hace un tiempo, la empresa ya tiene disponible para los clientes un producto muy similar, pero trabaja en una nueva versión que lo mejore. El anterior producto a ODA se llama Opac o mOPAC, la versión móvil. Un OPAC (Online Public Access Catalog) es una aplicación que ofrece a los usuarios acceso en línea a los recursos bibliográficos disponibles en la colección de una biblioteca. ODA permite a cada biblioteca acceder a todas las colecciones que posee, incluyendo acceso a muchos más catálogos que Opac, desde una única plataforma que combina simplicidad y potencia.

Uno de los módulos de los que estaba pendiente de desarrollo en la empresa era el módulo de valoraciones o reseñas. ODA cuenta con multitud de herramientas y módulos que facilitan a las bibliotecas la gestión de sus recursos y permiten a sus usuarios tener todos los productos disponibles de una forma muy simple, pero le falta algo muy importante en la actualidad. Al acceder a los productos, todo tipo de información está al alcance del usuario, menos las opiniones de otros usuarios sobre ese producto.

Este Trabajo de Fin de Grado consiste en el desarrollo del mencionado módulo de valoraciones del catálogo bibliotecario. Este permitirá mejorar la experiencia de los usuarios con la aplicación, pudiéndose consultar opiniones de otros usuarios para tomar decisiones informadas sobre artículos de las bibliotecas.

1.2 Motivación

En la actualidad, es muy útil contar con las opiniones y valoraciones de la gente que tiene experiencia en el uso de un producto, especialmente si se está interesado en su adquisición.

Hoy en día, es prácticamente imposible encontrar una web que ofrezca o venda productos a clientes que no cuente con un sistema que permita a los clientes valorar esos productos para que otros clientes cuenten con opiniones a tener en cuenta antes de adquirirlos. Además, se ha observado que este componente es actualmente una carencia del software existente en la empresa.

Por todo ello, considero este trabajo como una oportunidad para resolver un problema que mejore notablemente la experiencia de los usuarios de las bibliotecas involucradas, y también es una oportunidad muy buena para participar en un entorno real de trabajo, aprender mucho y crecer profesionalmente.

1.3 Objetivos

Como principal objetivo quiero conseguir que los usuarios de la aplicación puedan disponer de valoraciones y comentarios de otros usuarios cuando estén mirando un producto, ya sea un libro o libro electrónico o una película del que disponga la biblioteca en la que esté buscando. Para que la aplicación pueda disponer de este módulo y funcione de la mejor manera posible, se necesitan cumplir algunos objetivos más específicos:

- Que el usuario pueda valorar y comentar un producto, tener acceso a todos los comentarios que ha creado ya y modificar o eliminar cualquiera de ellos.

- Que cualquier usuario que acceda a la página a buscar un libro para pedir un préstamo, o simplemente para ver qué opina la gente sobre él, pueda hacerlo de la manera más simple posible, sin necesitar darse de alta en la aplicación.

- Desarrollar una parte del módulo que permita a los administradores de la biblioteca comprobar que un comentario es correcto antes de ser publicado y eliminarlo si procede, para evitar que aparezcan comentarios no deseados en la aplicación.

Personalmente, también quiero conseguir ciertos objetivos:

- Aprender todo lo posible sobre desarrollo de aplicaciones, aplicando todo lo aprendido en el grado para poder aprovechar al máximo el tiempo de trabajo.

- Ganar experiencia al participar en un entorno real de trabajo, incluyendo ya ciertos aspectos que en previos trabajos no existían, como el estricto cumplimiento de los tiempos marcados para las tareas del proyecto y el ajuste a las exigencias de diversos clientes.

Al trabajar en el grupo de desarrollo Java, no soy el encargado de diseñar el Front-end de ODA, por lo que el diseño visual de la aplicación lo realiza el equipo encargado del Front-end. Aun así, es imprescindible saber cómo funciona esta parte de la aplicación porque sí es mi responsabilidad la comunicación entre el Back-end y el Front-end, para que el equipo de Front-end pueda disponer de todos los datos necesarios para crear la parte que ve el cliente.

1.4 Estructura

Este documento está dividido en ocho secciones que explican detalladamente el desarrollo del proyecto.

- Apartado 1: Introducción. En ella se explica el contexto tanto del alumno como de la empresa, la motivación que ha llevado al alumno a la realización de este proyecto y los objetivos iniciales que se plantean, además de incluir una lista de términos para facilitar la comprensión de este documento.

- Apartado 2: Estado del arte. Estudio completo de las tecnologías actuales para el desarrollo web. Incluye dos ejemplos de trabajos similares ya diseñados y la justificación de la elección de las distintas tecnologías, así como una explicación de cada una de ellas.

- Apartado 3: Casos de uso y requisitos. Detalle de todos los casos de uso y requisitos del proyecto, tanto los de usuario como los de sistema, además de las matrices de trazabilidad.

- Apartado 4: Diseño del sistema. Incluye una explicación breve de la estructura y los módulos del proyecto ya existentes en la aplicación que se necesitan conocer para el desarrollo del proyecto. Además, se detallan y justifican todas las decisiones tomadas a lo largo del desarrollo.

- Apartado 5: Gestión del proyecto. Explicación de la metodología utilizada en el desarrollo del proyecto, así como la planificación, presupuesto y pruebas realizadas.

- Apartado 6: Marco regulador. Detalle de las leyes y licencias aplicadas en el desarrollo del proyecto.

- Apartado 7: Entorno socioeconómico. Explicación del efecto y posibles repercusiones del proyecto en la sociedad.

- Apartado 8: Conclusiones. Incluye una retrospectiva del proyecto, refresco de los objetivos y explicación breve de su consecución

Además de estas secciones, se incluyen al final la bibliografía utilizada y los anexos, que muestran imágenes de la aplicación para entender completamente su funcionamiento.

1.5 Lista de términos

SEO → "Search Engine Optimization", técnicas para mejorar posicionamiento web.

JVM→ "Java Virtual Machine", la máquina virtual de Java.

XML → "eXtensible Markup Language", lenguaje de marcado para almacenar datos.

CSV → "Comma-Separated Values", archivo para representar datos tabulares.

PDF → "Portable Document Format", archivo para representar documentos.

URL → "Uniform Resource Locator", cadena de caracteres que identifican la dirección de un recurso en Internet.

IoC → "Inversion of Control", patrón de diseño y un principio de programación.

ORM → "Object-Relational Mapping", técnica para mapear objetos a estructuras relacionales de una base de datos .

REST → "Representational State Transfer", estilo arquitectónico para facilitar interoperabilidad y escalabilidad.

RESTful → dicho de un servicio, "Representational State Transfer", que se basa en la utilización de recursos, la utilización de operaciones HTTP, una interfaz uniforme y múltiples representaciones posibles, como JSON, HTML o XML.

MVC → "Model-View Controller", patrón de diseño en aplicaciones software

WAR → "Web ARchive", archivo para empaquetar y distribuir aplicaciones en Java.

HTTP → "Hypertext Transfer Protocol", protocolo de comunicación en la web.

HTML → "Hypertext Markup Language", lenguaje de marcado para crear páginas web.

JSP → "JavaServer Pages", tecnología de Java para crear páginas web dinámicas.

SAML → "Security Assertion Markup Language", estándar para el intercambio de información entre dominios de seguridad.

API → "Application Programming Interface", conjunto de reglas y protocolos.

DAO → "Data Access Object", patrón de diseño de software para separar la lógica de acceso a datos de la lógica de negocio.

CRUD → "Create, Read, Update, Delete", operaciones de manipulación de datos.

GPL → "General Public License", licencia de software que establece las condiciones bajo las cuales un programa de software puede ser distribuido, utilizado y modificado.

GNU → "GNU's Not Unix", proyecto para crear un sistema operativo completo y libre.

IDE → "Integrated Development Environment", herramienta de software para facilitar su desarrollo.

Catálogo → en ODA, los catálogos son todos los tipos de productos que almacena la biblioteca. Pueden ser libros, libros electrónicos o películas, entre otros.

Framework → estructura o marco de trabajo.

Back-end → parte de una aplicación web que se encarga del procesamiento y la gestión de los datos.

Front-end → parte visible de una aplicación web con la que los usuarios interactúan.

Jetty y Undertow → servidores web y contenedores de servlets de código abierto escritos en Java.

Beans → clases especiales que siguen una convención de nomenclatura específica.

Getter → método para obtener el valor actual de una propiedad privada de una clase.

Setter → método para modificar el valor de una propiedad privada de una clase.

Datasource → objeto que proporciona una conexión a una fuente de datos.

CrudRepository y JpaRepository → interfaces proporcionadas por Spring Data para facilitar las operaciones de acceso y manipulación de datos en una base de datos.

PagingAndSortingRepository → interfaz proporcionada por Spring Data para realizar operaciones de paginación y ordenación en una colección de datos.

Servlet → clase Java utilizada para extender las capacidades de un servidor web y procesar solicitudes y respuestas HTTP.

HttpServletRequest → interfaz de Java que representa una solicitud HTTP recibida.

Linux → sistema operativo de código abierto basado en el núcleo Linux.

2. ESTADO DEL ARTE

2.1 Trabajos similares

2.1.1 Amazon

Es muy difícil encontrar en la actualidad una aplicación web que ofrezca productos a los clientes y no cuente con una sección en la que se muestren valoraciones y opiniones realizadas por los usuarios de la aplicación, por lo tanto, existen muchos trabajos similares a la idea de proyecto que se tiene. Entre ellos, me fijo en la sección de valoraciones que tiene disponible la plataforma Amazon para cada uno de los productos que ofrece.

Fig. 2.1. Valoraciones en Amazon [1]

En esta aplicación, están disponibles las puntuaciones y valoraciones que los usuarios añaden a un producto para todos los usuarios de la aplicación, se hayan registrado o no. Como se puede ver en la Figura 2.2, existe una funcionalidad que se asemeja un poco a lo que se pretende en este proyecto con la tarea de los administradores de las bibliotecas, que tendrán que validar que un comentario creado es apropiado, y si no lo es, eliminarlo u ocultarlo. En Amazon, se indica si el usuario que ha hecho cada valoración ha adquirido el producto o no. Por lo tanto, existe una labor de búsqueda por parte de los administradores de Amazon para verificar si un usuario que valora un producto lo ha comprado o no.

Fig. 2.2. Verificación de compra en Amazon [1]

La diferencia entre esta funcionalidad de Amazon y los objetivos de este proyecto es que en Amazon es imprescindible darse de alta en la página con una cuenta para poder añadir una valoración, en cambio, en ODA los usuarios anónimos también podrán añadir una valoración.

2.1.2 Casa del Libro

Lo más parecido que se puede encontrar actualmente a este módulo de valoraciones de catálogos bibliotecarios puede ser la página web de Casa del Libro, al ofrecer un servicio muy parecido. Esta plataforma es un tanto diferente a ODA porque no está diseñada para bibliotecas, es una librería online, por lo tanto, en ella es posible comprar libros, pero no pedir su préstamo.

Fig. 2.3. Valoraciones en listado en Casa del Libro [2]

Como se observa en la Figura 2.3, en el listado de libros que aparece al realizar una búsqueda se muestra la media de valoraciones que los usuarios le ha dado a cada libro y también el número de comentarios que se han realizado. En el segundo caso, no aparece ninguna de las dos cosas, ya que no existe todavía ninguna reseña de ese libro.

Además, en la Figura 2.4, se observa el diseño de este módulo una vez se selecciona un libro. Se muestra el número de veces que se le ha añadido cada puntuación distinta (del 1 al 5), los comentarios que han realizado los usuarios, incluyendo autor y fecha, y se ofrece la posibilidad de añadir una opinión al libro seleccionado. Al igual que en Amazon y, al contrario que en el proyecto de ODA, sólo se permite añadir valoraciones a usuarios identificados en la aplicación.

Fig. 2.4. Valoración y comentarios en detalle Casa del Libro [2]

2.2 Tecnologías para desarrollar aplicaciones web similares

2.2.1 Arquitectura de aplicaciones web

Para discutir sobre las posibles arquitecturas que se pueden utilizar en desarrollo web se diferencian dos aspectos: el tipo de renderizado (cliente, servidor o web estática) y el enfoque de la arquitectura (monolítico o microservicios).

2.2.1.1 CSR, SSR y SSG

CSR (Client Side Rendering), SSR (Server Side Rendering) y SSG (Static Site Generation) son los tres tipos de renderizado que existen en la actualidad. En desarrollo web, renderizar se refiere al proceso de convertir el código de una página web en elementos visuales y funcionales que los usuarios pueden ver y con los que pueden interactuar [3].

El Client Side Rendering es el método de renderizado que se realiza principalmente en el navegador. Cuando el navegador solicita un sitio web que utiliza este tipo de renderizado, normalmente recibe un archivo HTML con un contenido mínimo, ya que el encargado de cargar y mostrar el contenido es JavaScript. Entre sus ventajas más importantes destaca la rápida navegación entre vistas que ofrece. Por el contrario, entre las desventajas está el bajo rendimiento de la aplicación, el deficiente posicionamiento SEO, ya que los rastreadores deben realizar un trabajo adicional para analizar el código, y la complejidad de mantenimiento [3]. Las librerías más recomendadas para CSR son React, Vue, Svelte y Angular.

El Server Side Rendering es el tipo de renderizado que más se ha utilizado desde los inicios de la Web. Cuando un navegador solicita una página web, ésta se genera desde el servidor, compilando el código y los datos para devolver una página HTML completa. Los datos se actualizan dinámicamente en cada solicitud que realiza el navegador, lo que explica por qué en estos sitios es necesario recargar la página para obtener la versión más actualizada. Lo que en CSR eran desventajas, en SSR son ventajas. Al utilizar este método, el rendimiento de la aplicación es considerablemente mejor y el posicionamiento SEO también mejora mucho. En cambio, se depende de un proveedor de alojamiento para ejecutar la aplicación y si el sitio recibe grandes volúmenes de visitas el coste del proveedor puede ser mayor, al necesitar un servidor capaz de manejar todas las peticiones [4]. Para este tipo de renderizado se utilizan lenguajes de servidor, como PHP, Python o Java.

Por último, existe el tipo de renderizado Static Site Generation). Este renderizado se refiere a la creación de sitios web estáticos que funcionan con HTML, CSS y JavaScript, sin necesidad de código de servidor. Estos sitios se llaman estáticos debido a que todos los datos se encuentran directamente incorporados en el código, no se obtienen del servidor, lo cual puede resultar inconveniente al trabajar con proyectos extensos. Son muchas las ventajas de este renderizado, como la facilidad de desarrollo, el buen rendimiento, el posicionamiento SEO, los bajos costes que se pagan por los servicios de alojamiento y, sobre todo, la amplia variedad de herramientas disponibles en diferentes lenguajes. En cambio, es una gran desventaja la necesidad de reconstruir todo el proyecto para cada cambio, lo que puede hacer perder mucho tiempo a los desarrolladores [5]. Entre los generadores de sitios estáticos más conocidos se encuentran Astro, Next.js o Gatsby.

2.2.1.2 Monolítico vs. Microservicios

La arquitectura monolítica y la arquitectura de microservicios son dos enfoques diferentes para diseñar aplicaciones y sistemas de software. Las aplicaciones que utilizan una arquitectura monolítica se caracterizan por la agrupación del software en un solo componente o bloque de código. De esta manera, la aplicación se desarrolla como una entidad indivisible, todas las partes del sistema, incluyendo la interfaz de usuario, la lógica de negocio y la base de datos, están integradas en una sola unidad y están pensadas para ejecutar en una sola máquina. Este enfoque es conocido por su simplicidad y facilidad de implementación, pero es limitado en cuanto a escalabilidad y flexibilidad, ya que es complicada su adaptación a cambios o necesidades futuras.

Por el contrario, la arquitectura basada en microservicios se caracteriza por la división de la aplicación en pequeños servicios independientes, cada uno con su propia funcionalidad y ejecución por separado. La comunicación entre los servicios se realiza a través de interfaces bien definidas, generalmente a través de API REST. Como cada microservicio se desarrolla de manera independiente, el desarrollo de la aplicación es ágil y modular. Además, al poder diseñar cada módulo por separado, esta arquitectura facilita la

colaboración y permite a los equipos de desarrollo trabajar en paralelo en diferentes servicios. El inconveniente de utilizar esta arquitectura es la complejidad que requiere gestionar múltiples servicios y la necesidad de una infraestructura adecuada para su gestión y monitorización.

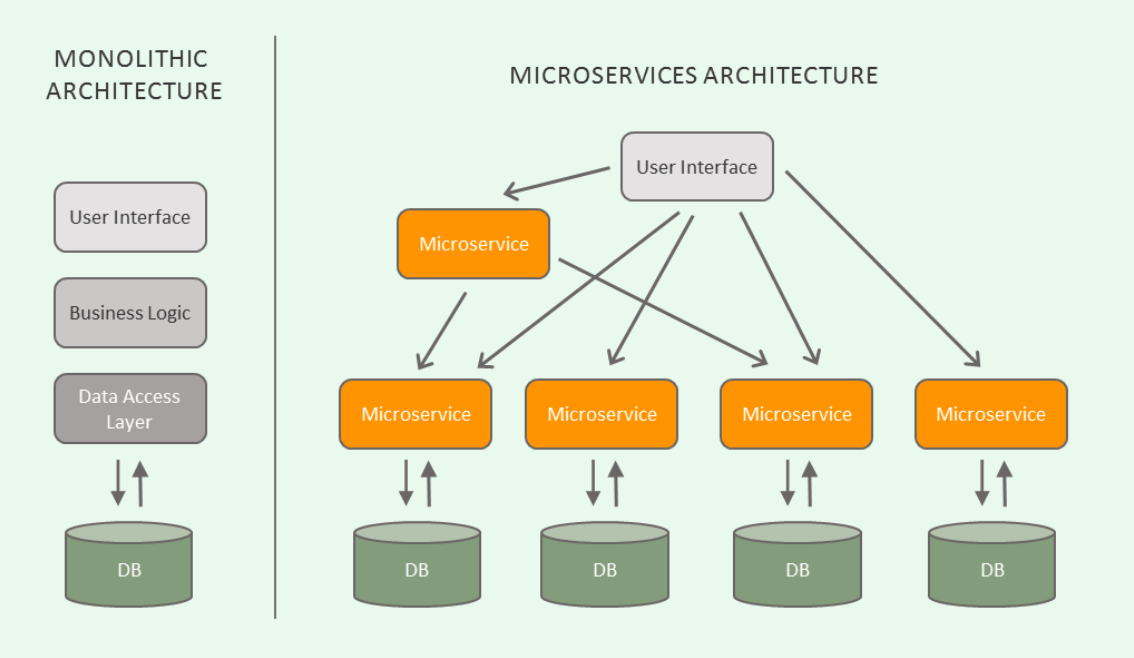


Fig. 2.5. Arquitectura monolítica y microservicios [6]

2.2.2 Stacks de back-end

Para desarrollar el back-end en una aplicación web existen actualmente muchas opciones a tener en cuenta, la elección dependerá de las necesidades y las características de la aplicación que se quiere diseñar. Entre los lenguajes más utilizados en la actualidad para el desarrollo web se encuentran Java, Python, JavaScript y PHP.

2.2.2.1 Java

Java es un lenguaje de programación utilizado para crear software compatible con una gran diversidad de sistemas operativos. Tiene la característica única de ser compilado e interpretado al mismo tiempo, convirtiéndolo en un lenguaje simplificado que transforma automáticamente el código en instrucciones ejecutables por la máquina [7].

Es un lenguaje de alto nivel y está orientado a objetos. Apareció por primera vez en 1995, cuando Sun Microsystems lo presentó públicamente, aunque en la actualidad es propiedad de Oracle. La versión más reciente de Java es la 17. Los frameworks basados en Java más comunes en el desarrollo web son JEE, Spring y Struts. Las diferencias entre estos frameworks se resumen en la Tabla 2.1.

TABLA 2.1. COMPARACIÓN ENTRE FRAMEWORKS DE JAVA [8], [9]

	JEE	Spring	Struts
Arquitectura	Basada en el modelo de componentes empresariales	Basada en MVC, integración de ORM	Basada en MVC
IoC (Inversión de Control)	Si	Si	No
Escalabilidad	Alta	Alta	Media
Flexibilidad	Alta	Alta	Media
Compatibilidad	Se pueden usar Spring y Struts dentro de aplicaciones JEE	Independiente de Struts	Independiente de Spring

2.2.2.2 Python

Python es un lenguaje de programación que se encuentra en multitud de servidores y sistemas operativos. Destaca por su simplicidad, legibilidad y facilidad de uso, convirtiéndolo así en una de las opciones preferidas tanto para desarrolladores experimentados como principiantes [10].

Igual que Java, es un lenguaje de alto nivel orientado a objetos. Seguramente sus mayores ventajas son su simplicidad, limpieza y rapidez. Ofrece muchas facilidades a los desarrolladores, y sobre todo es uno de los mejores lenguajes para empezar a programar, al ser su curva de aprendizaje muy positiva en comparación con sus competidores. Además, tiene una amplia variedad de frameworks poderosos entre los que poder elegir y cuenta con una gran comunidad activa y dispuesta a mejorar su desarrollo.

Uno de los frameworks más utilizados para el desarrollo web con Python es Django. Es muy completo y ofrece una amplia gama de características y funcionalidades para el desarrollo web que, a diferencia de Python, hace que su curva de aprendizaje inicial sea un poco más pronunciada, debiendo dedicar más tiempo a su aprendizaje. Al igual que los frameworks de Java vistos, sigue el patrón de diseño MVC y también tiene ORM incorporado que facilita la interacción con la base de datos. Por último y no menos importante, tiene muchos casos prácticos y documentos en línea que los desarrolladores pueden consultar en caso de tener cualquier problema.

Fig. 2.6. Django ORM [11]

2.2.2.3 JavaScript

JavaScript es un lenguaje de programación de alto nivel. La principal diferencia entre JavaScript y otros lenguajes como Java y Python es que JavaScript se encuentra en el lado del cliente, ejecutándose así en el navegador web del usuario, convirtiéndolo así en el único lenguaje de programación que entienden los navegadores. Los demás lenguajes se utilizan en el lado del servidor [12].

Por ello, JavaScript es considerado el lenguaje más versátil, al abarcar una amplia variedad de aplicaciones y usos actualmente. Permite a los desarrolladores escribir tanto código del servidor como del cliente en el mismo lenguaje y, además de en el navegador, puede ejecutarse en diferentes entornos fuera del navegador. Otra de las razones por las que es una opción propicia para principiantes es que es un lenguaje con tipado ligeramente flexible, por lo que los primeros pasos suelen ser relativamente sencillos. Por el contrario, es cierto que a medida que el programador profundiza en el lenguaje, sus características requieren un estudio detallado para avanzar con confianza y ser capaz de desarrollar aplicaciones más complejas.

En cuanto a los frameworks que ofrece JavaScript, destacan dos opciones importantes, Angular para el front-end y Express para el lado del servidor. Angular es un framework de JavaScript poderoso y muy adecuado para el desarrollo de aplicaciones front-end de complejidad media o alta. Por otro lado, Express es un framework minimalista que se utiliza para construir aplicaciones web en el lado del servidor y está basado en el módulo nativo de Node.js. En la siguiente tabla se diferencian las principales características de estos dos frameworks.

TABLA 2.2. CARACTERÍSTICAS ANGULAR Y EXPRESS

Característica	Angular	Express
Uso	Desarrollo de aplicaciones front-end complejas	Desarrollo de aplicaciones web y APIs en el lado del servidor
Tipo de aplicación que crea	SPA (Single Page Application) o PWA (Progressive Web App)	Aplicaciones web y APIs RESTful
Escalabilidad	Sólido y escalable	Flexible y escalable
Lenguaje principal	TypeScript	JavaScript
Convenciones	Estilo de codificación homogéneo y de gran modularidad	Enfoque modular y sintaxis clara

2.2.2.4 PHP

Actualmente, PHP es el lenguaje de programación más utilizado en la web, con él es posible realizar el back-end de aplicaciones web fácil y directamente [13]. Lo que diferencia a PHP de sus competidores es su facilidad de uso, permitiendo así desarrollar páginas web desde cero a personas con muy poco conocimiento.

Es un lenguaje de código abierto, y cuenta con una amplia comunidad de desarrolladores y también documentación disponible en la red para resolver cualquier tipo de duda y mejorar el lenguaje. Además, PHP es un lenguaje multiplataforma, al ser compatible tanto en Windows como en Linux y MacOS y también se puede ejecutar en la mayoría de los servidores web populares. En cuanto a seguridad, se ha señalado históricamente a PHP como un lenguaje inseguro, y no es del todo cierto. En las primeras versiones de PHP sí existían configuraciones que eran inseguras, pero en las versiones actuales del lenguaje estas configuraciones ya no existen.

Uno de los frameworks más importantes y populares de PHP es Laravel. Laravel ha ganado recientemente popularidad gracias a su elegante sintaxis, la legibilidad de su código y su amplio conjunto de características que permiten crear aplicaciones web totalmente personalizadas. Al igual que muchos frameworks ya analizados, sigue el patrón de diseño MVC.

Entre las características que ofrece Laravel destaca su sistema de enrutamiento, que permite crear diferentes tipos de URLs amigables para usuarios y buscadores, facilitando la navegación dentro de la aplicación; la abstracción de base de datos con ORM, que permite interactuar con una base de datos como si fuera un conjunto de objetos simples facilitando el manejo de datos; las colas de trabajo que hacen posible la ejecución de tareas en segundo plano, ayudando a mejorar el rendimiento de la aplicaciones y su alta seguridad, con frecuentes actualizaciones [14].

2.2.3 Bases de datos SQL y NoSQL

Para desarrollar una aplicación web, es imprescindible la elección de una base de datos, que puede ser relacional (SQL, Structured Query Language) o no relacional (NoSQL, Not Only Structured Query Language). La principal diferencia entre ellas radica en la forma en que se manejan los identificadores únicos de los registros. En SQL, cada registro debe tener un identificador único, la clave primaria, para garantizar su unicidad. En cambio, en NoSQL, no es necesario utilizar identificadores únicos para los datos almacenados [15]. Además, en SQL todas las tablas necesitan estar estructuradas, cada una requiere unos campos, pero no puede tener un número indefinido de estos. Por el contrario, con NoSQL es posible guardar un JSON sin conocer de antemano su estructura.

En la siguiente tabla se indican las diferencias entre bases de datos SQL y NoSQL:

TABLA 2.3. BASES DE DATOS SQL Y NOSQL [16]

Característica	SQL	NoSQL
Organización y estructura de los datos	Datos organizados en tablas, con filas, columnas y registros, relacionados a través de claves foráneas	Datos almacenados en documentos independientes, sin una estructura rígida, relacionados a través de embebido de documentos, referencias o modelado de gráficos
Lenguaje de consulta	Lenguaje SQL para acceder y manipular los datos	No se utiliza SQL, sino lenguajes o APIs específicas para cada base de datos
Complejidad	Más complejas, debido a su tiempo en el mercado y a su estructura	Más flexibles y menos complejas, debido a su capacidad para adaptarse a diferentes tipos de datos
Velocidad y rendimiento	Suelen ser más lentas y suelen consumir más recursos, por la atomicidad de sus operaciones y su estructura compleja	Más rápidas y consumen menos recursos, gracias a su capacidad para escalar horizontalmente
Herramientas y soporte	Amplia variedad de herramientas y soporte, debido a su largo tiempo en el mercado	Menos herramientas y soporte, pero en la actualidad están creciendo notablemente
Modificación de datos en tiempo real	Pueden requerir la detención de la base de datos para modificar los datos	Pueden modificar los datos sin necesidad de detener la base de datos, permitiendo cambios en tiempo real
Ejemplos	MySQL, MariaDB, Oracle, PostgreSQL, Microsoft SQL Server	MongoDB, Cassandra, Redis, CouchDB

2.3 Tecnologías utilizadas

En esta sección se especifican las tecnologías que se han utilizado a lo largo del proyecto, tanto para crear el módulo de valoraciones como para realizar las pruebas y llevar el seguimiento de la gestión del proyecto. Al trabajar sobre una aplicación ya existente, no ha habido ninguna posibilidad de elegir qué tecnologías usar para llevar a cabo el trabajo, ya que adaptarse al entorno de trabajo que tiene la empresa es imprescindible para participar en el proyecto.

Como lenguaje de programación, se utiliza Java junto a Spring, Maven y Thymeleaf para la codificación de la solución. Para almacenar los datos se utiliza MariaDB, junto a Hibernate y Spring Data JPA para la persistencia. Además, se utiliza Solr como motor de búsqueda para todos los productos de la biblioteca. Por último, se utilizan JaCoCo y Mockito para la realización de las pruebas y tests, Redmine para la gestión del proyecto y Subversion para el control de versiones.

2.3.1 Java

Hoy en día, Java es uno de los lenguajes de programación más utilizados para el desarrollo web, debido a sus múltiples ventajas, aunque también encontramos algún inconveniente al compararlo con otros lenguajes. En la Figura 2.7 se observan los principales lenguajes de programación, ordenados tanto por programadores como por su índice de popularidad, PYPL.

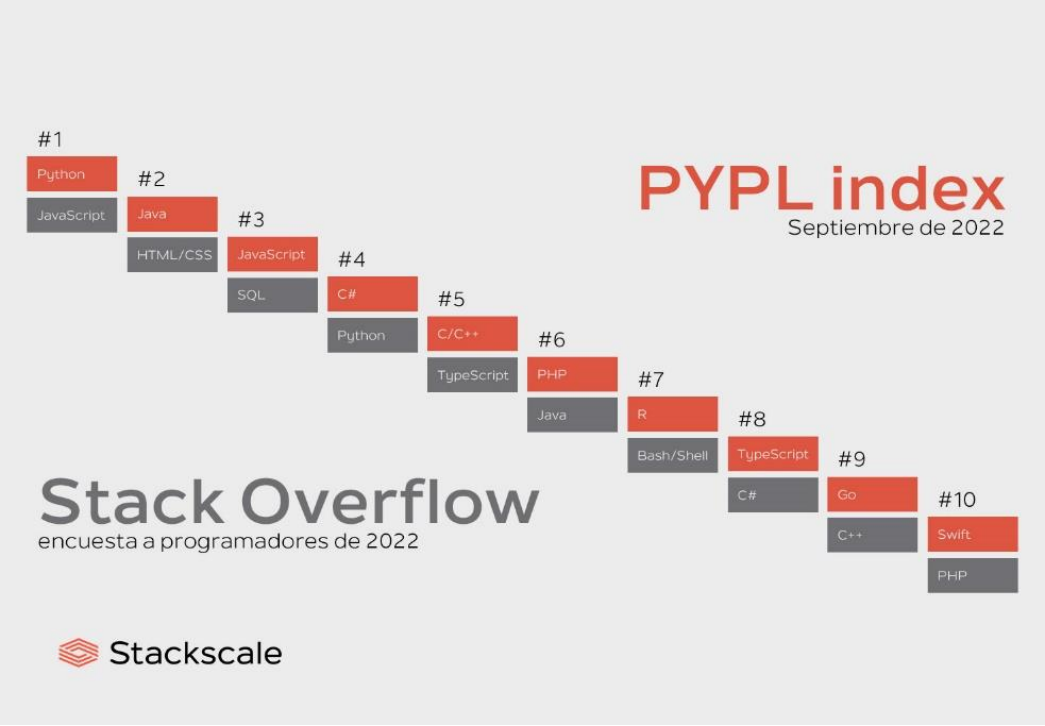


Fig. 2.7. Ránking de lenguajes de programación [17]

TABLA 2.4. VENTAJAS E INCONVENIENTES DE JAVA [18]

Ventajas	Inconvenientes
Lenguaje multiplataforma, no hay necesidad de escribir distintas versiones de la aplicación para cada plataforma	Rendimiento más lento que otros lenguajes, como C y C++
Posee una amplia gama de librerías y herramientas que ahorran tiempo a los programadores y reducen errores	Requiere más memoria que otros lenguajes. La JVM necesita disponer de una cantidad considerable de memoria disponible para poder trabajar
Sistema de seguridad incorporado, basado en políticas de seguridad, clasificadores de seguridad, certificados digitales y administrador de seguridad	La programación tiene sobrecarga de código, pudiendo ralentizar el desarrollo y hacer más difícil el mantenimiento del código
OOP, orientado a objetos, ahorro de tiempo y esfuerzo al crear código nuevo	Java ha tenido múltiples versiones a lo largo de los años, lo que conlleva a veces problemas de compatibilidad de las aplicaciones
Es un lenguaje de alto nivel, utiliza estructuras de datos complejas y abstracciones que permiten representar problemas y soluciones	Al ser un lenguaje de alto nivel, su curva de aprendizaje es una desventaja. No es un lenguaje fácil de aprender para la gente que empieza a programar

2.3.2 Spring Framework

Spring es un framework de código abierto que permite el desarrollo de aplicaciones Java [19]. Actualmente, es el framework más popular y utilizado de Java en el entorno empresarial, debido a su capacidad para el desarrollo de aplicaciones complejas de una forma rápida. Ofrece muchísimas posibilidades a los desarrolladores, gracias a todos los módulos que incluye, los principales se pueden observar en la Figura 2.8. Spring nace en 2002 de la mano de Ron Johnson y la versión más reciente es la 6, lanzada inicialmente en 2022.

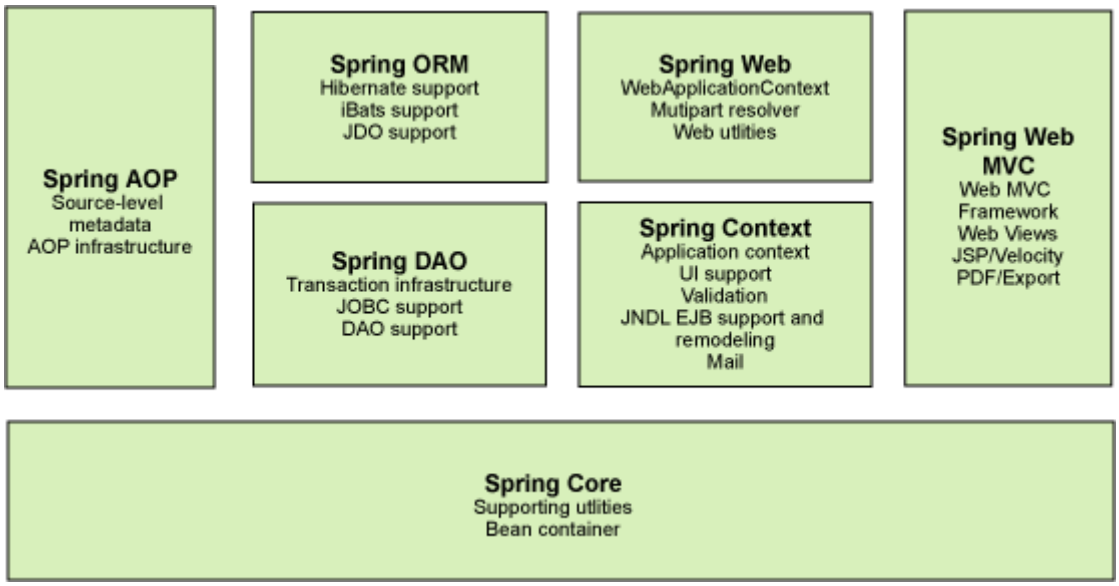


Fig. 2.8. Módulos de Spring [20]

2.3.2.1 Spring Boot

Spring Boot es un framework de desarrollo de aplicaciones de Java basado en Spring que se enfoca en la creación de aplicaciones Java stand-alone, rápidas y fáciles de configurar. Para crear una aplicación con Spring se distinguen tres pasos a seguir: seleccionar dependencias, crear la aplicación y desplegarla en un servidor. Spring Boot nace en 2014 con la idea de simplificar el primer y el tercer paso y permitir a los desarrolladores centrarse en crear la aplicación, al requerir una configuración mínima e incluir características listas para entornos de producción [21].

El primer paso se consigue con Maven, que se explica un poco más adelante. El despliegue de la aplicación en un servidor se realiza automáticamente y reduciendo al máximo la configuración del proyecto. Esto se debe a que Spring Boot incluye un servidor web Apache Tomcat (también puede ser Jetty o Undertow) embebido, por lo que ya no es necesario realizar el despliegue de archivos WAR, como ocurre al trabajar con JEE, por ejemplo.

Spring tiene muchos módulos que se tienen que configurar para poder integrarlos correctamente. Spring Boot facilita mucho esta tarea de configuración de los módulos de Spring, además de ofrecer una amplia gama de módulos y funcionalidades adicionales que son fáciles de incorporar en el proyecto.

2.3.2.2 Spring MVC

Spring MVC es un marco de Java para crear aplicaciones web y servicios RESTful de forma óptima y fácil, basado en el patrón de diseño Modelo-Vista-Controlador. Estos tres componentes trabajan juntos para gestionar la lógica de negocio, la presentación de la interfaz de usuario y la interacción con el usuario [22].

El modelo representa los datos de la aplicación. Se puede implementar de varias maneras, comúnmente se suele hacer con clases de Java o Beans de Spring. En el modelo la información se encapsula y se proporcionan métodos para acceder (getters) y manipular (setters) la información. Con la integración de Spring Data JPA que se explica más adelante, los modelos se relacionan directamente con tablas de una base de datos utilizando diferentes anotaciones que proporciona Spring.

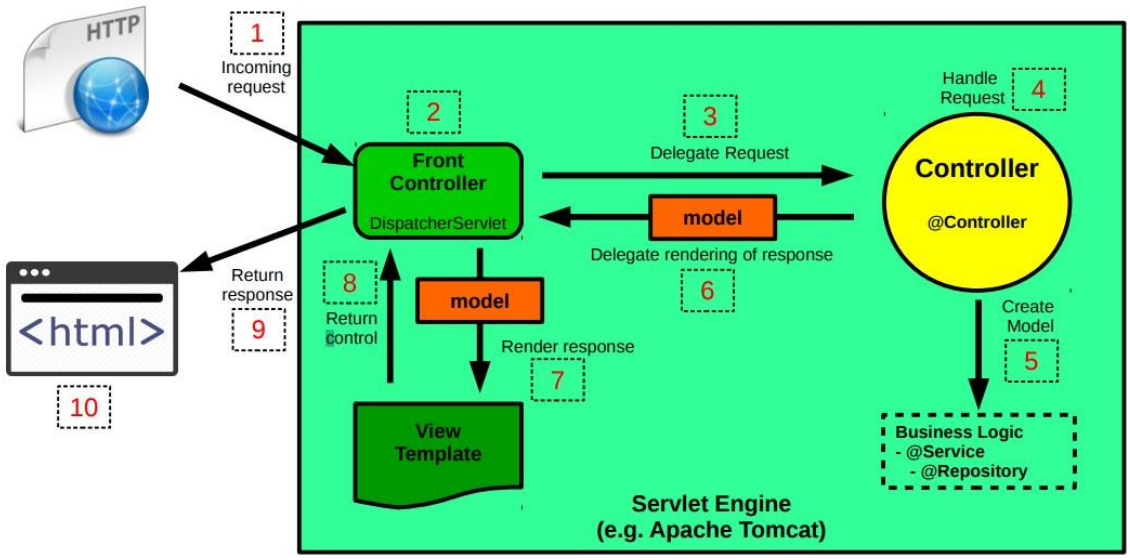


Fig. 2.9. Esquema de flujo Spring MVC [23]

La vista es la capa de presentación de la aplicación, lo que se le muestra al usuario, comúnmente conocido como Front-end. También existen varias formas de implementar las vistas, siendo HTML con Thymeleaf y JSP las más comunes. La vista se encarga tanto de la presentación como de la interacción con el usuario.

Por último, el controlador es el intermediario entre el modelo y la vista. Es el elemento más importante, ya que maneja las solicitudes del usuario, procesa la lógica de negocio y decide qué vista mostrar al usuario. Dentro de él, se implementan métodos que reciben las peticiones HTTP, invocan los componentes del modelo necesarios para procesar la solicitud del usuario y devuelven el nombre de la vista que se ha de mostrar al usuario en cada caso. En la siguiente tabla se muestran las anotaciones de Spring más importantes que se usan dentro de los controladores:

TABLA 2.5. ANOTACIONES DE SPRING EN LOS CONTROLADORES

Anotación	Función	Ejemplo

@Controller	Indica que la clase anotada es un controlador que incluye métodos que devuelven nombres de vistas	@Controller
		public class ValoracionesController{
		}

@ResponseBody	Indica que el método anotado no devuelve el nombre de una vista, sino que el resultado debe escribirse en el cuerpo de la respuesta	@GetMapping(“/texto”)
		@ResponseBody
		public String obtenerTexto(){
		}

@RestController	Combina las dos anteriores anotaciones, la clase anotada así se utiliza específicamente para construir servicios RESTful	@RestController
		public class Controller{
		}

@Autowired	Inyecta dependencias automáticamente, Spring se encarga de buscar una instancia con el nombre proporcionado después de la anotación	@Autowired private ValoracionesServ serviceVal; @Autowired private ValoracionesRepo repoVal;

@RequestMapping	Mapea una URL específica a un método de un controlador. Se pueden utilizar también las anotaciones @GetMapping, @PostMapping, @PutMapping y @DeleteMapping para indicar directamente la acción que lleva a cabo el método	@Controller
		@RequestMapping(“/valoracion”)
		public class ValoracionesController{
		}
		
		@PostMapping(“/guardar”)
		public String guardarValoracion{
		}

@PathVariable	Se utiliza para vincular un valor de la variable en la URL de la solicitud a un parámetro dentro del método, permitiendo acceder a partes dinámicas de la URL para poder utilizarlas en el controlador	@GetMapping(“/{id}”)
		public String
		eliminarVal(@PathVariable(“id”)
		int idVal){
		}

@RequestParam	Se utiliza para vincular los parámetros de una solicitud HTTP a los que	@PostMapping(“/guardar”)

	están definidos dentro del método	public String guardarValoracion(@RequestParam(“punt”) int puntuacion){ }

@RequestBody	Tiene un uso muy parecido a @RequestParam, pero en este caso se obtiene el cuerpo completo de la solicitud. Es útil cuando se necesita recibir un objeto completo y no un parámetro solo.	@GetMapping(“/crear”)
		public String crearValoracion(@RequestBody Valoracion valoracion){
		}

2.3.2.3 Spring Security

Spring Security es otro framework, muy potente y altamente personalizable, que permite aplicar seguridad a aplicaciones basadas en Spring [24].

Este framework aplica dos tipos de seguridad que son imprescindibles de entender y diferenciar para aplicarlo correctamente a las aplicaciones de Spring. Estos dos conceptos son autenticación y autorización.

La autenticación es el proceso que consiste en verificar y validar la identidad de un usuario de la aplicación. El usuario proporciona unas credenciales, normalmente nombre de usuario y contraseña, y se verifica si estas son válidas y si se corresponden con las credenciales de algún usuario registrado en el sistema o base de datos. Spring Security ofrece diversas formas de aplicar la autenticación a las aplicaciones Spring, como por ejemplo la autenticación basada en formularios, en tokens o en SAML, entre otros. Como he dicho antes, es posible también personalizar el proceso de autenticación para adaptarlo de la mejor manera a las necesidades de la aplicación.

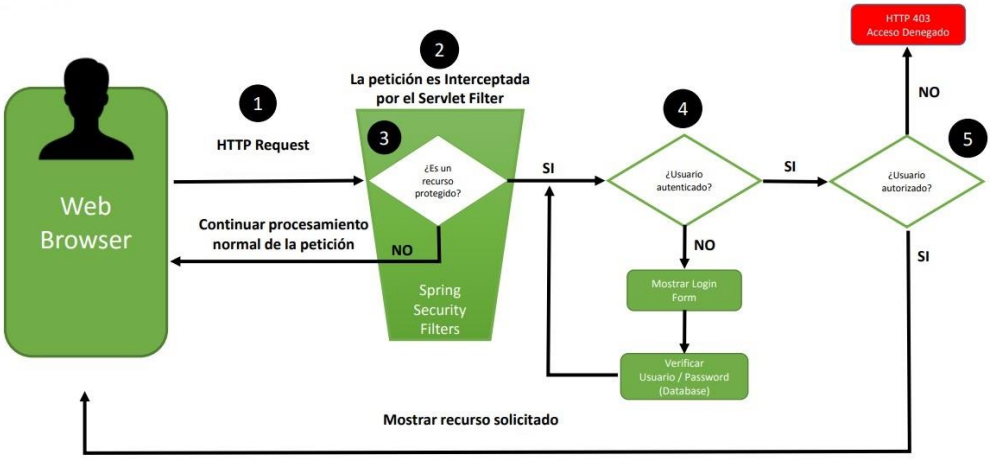


Fig. 2.10. Esquema de flujo Spring Security [23]

La autorización se lleva a cabo una vez el proceso de autenticación se ha completado, es decir, una vez que un usuario ha sido autenticado se procede a autorizarlo. Este proceso determina qué acciones puede realizar el usuario dentro de la aplicación o a qué recursos tiene permitido acceder. Para llevar a cabo este proceso, es importante diferenciar entre roles y permisos. Los roles son conjuntos de permisos y se asignan a los usuarios, y los permisos definen las acciones que un usuario puede realizar dentro de la aplicación. Por ejemplo, un rol puede ser el de administrador, incluyendo permisos exclusivos, como puede ser eliminar usuarios de la base de datos. Otro rol puede ser el de usuario, que no podrá eliminar otros usuarios de la aplicación, podrá como mucho darse de baja a sí mismo. Spring Security también proporciona distintos mecanismos para llevar a cabo el proceso de autorización, como pueden ser mediante anotaciones de seguridad en el código (*@Configuration*, *@Secured*, *@PreAuthorize*, etc.) o mediante configuraciones aplicadas en archivos de propiedades.

2.3.3 Maven

Maven es una de las herramientas más útiles para la gestión y construcción de software. Está estrechamente relacionado con Java, ya que los dos se utilizan juntos para el desarrollo de aplicaciones [25].

Maven simplifica en gran medida la gestión de proyectos Java, ofreciendo la posibilidad de administrar fácilmente las dependencias, compilar el código fuente, ejecutar pruebas, generar documentación y empaquetar la aplicación en distintas secciones. El archivo de configuración de Maven es el POM.xml (Project Object Model), en el que se define la estructura del proyecto y se inyectan las dependencias que se necesitan. Al declarar las dependencias en este archivo, Maven se encarga de descargarlas y agregarlas al proyecto durante su compilación y ejecución.

Es importante mencionar también tanto los plugins como los repositorios de Maven. Los plugins son los que realizan muchas de las tareas de Maven, como la compilación del código. Se pueden utilizar los que tiene Maven como predeterminados o se pueden añadir plugins personalizados al POM. Los repositorios es donde Maven busca las dependencias y los plugins que se añaden al POM para descargarlos y almacenarlos en un repositorio local en la máquina en la que se ejecuta el proyecto, pudiendo así estar disponibles sin necesidad de tener una conexión a Internet. El repositorio público más utilizado es Maven Central. Actúa como un almacén centralizado que contiene muchas bibliotecas y plugins de código abierto disponibles para los desarrolladores.

En las figuras 2.11 y 2.12 se muestran algunas de las dependencias más importantes que se incluyen en el POM para desarrollar una aplicación Java con Spring. La primera es fundamental para poder desarrollar aplicaciones web con Spring, ya que incluye entre otras cosas el servidor web y el contenedor de servlets.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Fig. 2.11. Dependencia Spring Boot

La segunda es también esencial si la aplicación web necesita acceder y manipular una base de datos relacional, proporcionando soporte para la capa de persistencia a través de JPA.

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Fig. 2.12. Dependencia Spring Data JPA

2.3.4 Hibernate y Spring Data JPA

JPA (Java Persistence API) es una especificación de Java EE que permite a los desarrolladores Java hacer un mapeo entre los objetos y las tablas de una base de datos para facilitar la administración de los datos relacionales en las aplicaciones [26].

Spring Data JPA es un módulo que forma parte del proyecto Spring Data y ayuda a simplificar el desarrollo de la persistencia de datos utilizando repositorios, similares al patrón de diseño DAO. En otras palabras, este módulo de Spring Data agrega una capa de abstracción al API de JPA para conseguir una forma más sencilla de trabajar con JPA. Al utilizar JPA en aplicaciones Java con Spring obtenemos muchas ventajas. La principal es la reducción de trabajo y tiempo al tener que escribir mucha menos cantidad de código. Permite realizar operaciones CRUD y consultas personalizadas con sólo declarar métodos en un repositorio, no es necesario escribir consultas SQL manualmente ni siquiera implementar las operaciones básicas de persistencia, reduciendo así también en gran medida la posibilidad de errores.

Para poder utilizar este módulo en un proyecto es necesario seguir una serie de pasos. Primero, se debe agregar la dependencia al archivo POM. Después, es necesario indicarle al proyecto dónde se encuentra la base de datos con la que se quiere trabajar. Para ello, en el archivo de propiedades hay que especificar la URL del datasource, el usuario y la contraseña para conectarse a la base de datos, pudiendo indicar alguna propiedad más si se necesita.

```
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/bbdd
spring.datasource.username=root
spring.datasource.password=admin
```

Fig. 2.13. Conexión a base de datos desde Spring

A continuación, se debe definir una entidad en el proyecto para relacionarla con una tabla en la base de datos que se tiene. En este apartado es importante que tanto el nombre con el que se llama a la tabla como el que se les pone a todos los atributos es exactamente el mismo en el proyecto que en la base de datos, de otra manera el mapeo no se hará correctamente. Para poder relacionar todo lo que incluye la entidad con la tabla se usan también las anotaciones de Spring.

TABLA 2.6. ANOTACIONES DE SPRING EN LAS ENTIDADES

Anotación	Función	Ejemplo
@Entity	Mapear la clase anotada a una tabla en la base de datos	@Entity public class Valoracion {}
@Table	Personalizar el nombre de la tabla	@Entity @Table (name=”valoracion”) public class Valoracion {}
@Id	Indicar que el atributo anotado es la clave primaria	@Id private Long idVal;
@GeneratedValue	Especificar cómo se genera la clave primaria, muy útil para generar un id automáticamente	@Id @GeneratedValue(strategy = GenerationType.AUTO) private Long idVal;
@Column	Especificar detalles sobre la columna anotada	@Column(name=”usuario”, nullable=false) private String usuario;
@OneToOne, @OneToMany, @ManyToMany	Definir relaciones entre tablas a través de claves primarias y foráneas	En la entidad “Usuario”: @Id private Long id; @OneToMany(mappedBy=”usuario”) private List<Valoracion> valoraciones; En la entidad “Valoracion”: @ManyToOne @JoinColumn(name=”usuario_id”) private Usuario usuario;

Una vez definidas las entidades necesarias, se deben crear los repositorios. Basta con declararlos, ya que Spring Data JPA incluye los métodos necesarios para realizar las operaciones CRUD. Para poder utilizar estos métodos y, si es necesario, declarar alguno nuevo, es necesario extender de algún repositorio que proporciona Spring, como por ejemplo *CrudRepository* o *JpaRepository*. Para crear algún método personalizado con una consulta SQL, basta con utilizar la anotación *@Query*.

Entre los repositorios y los controladores es necesario también crear los servicios y su implementación. Los servicios encapsulan la lógica de negocio de la aplicación y manejan las operaciones relacionadas con las entidades. Para indicar que una clase es un servicio, se debe utilizar la anotación *@Service*. Por último, se deben inyectar los servicios en los controladores, con la anotación *@Autowired*, para poder realizar las operaciones necesarias y devolver al cliente las respuestas que solicita.

En cuanto a Hibernate, es importante saber diferenciarlo de Spring Data JPA. Hibernate es una implementación JPA, mientras que Spring Data JPA es una abstracción de acceso a datos JPA. Por ello, no es correcto afirmar que hay que elegir entre usar una cosa u otra, sino que se complementan. Spring Data JPA no puede funcionar sin un proveedor de JPA, pudiendo ser este Hibernate, Eclipse Link o algún otro proveedor. Además de configurar el proyecto para usar JPA como se explica anteriormente, es imprescindible añadir las dependencias de Hibernate al POM y, si se quiere, añadir alguna propiedad de Hibernate al archivo de propiedades para poder utilizar Hibernate y Spring Data JPA conjuntamente.

2.3.5 Thymeleaf

Thymeleaf es un motor de plantillas Java en el lado del servidor que se utiliza para generar vistas HTML dinámicas. Es una herramienta muy popular en el desarrollo web con Java y se integra de manera sencilla con Spring [27].

Entre sus ventajas destacan tanto su alto rendimiento como su flexibilidad que, combinados con la facilidad para su aprendizaje, hacen de Thymeleaf una opción a tener muy en cuenta a la hora de diseñar el front-end de las aplicaciones. Para poder utilizarlo, se debe agregar su dependencia al archivo POM del proyecto. Una de las alternativas más comunes a Thymeleaf es JSP. Si se utilizan estas plantillas, la comunicación con los controladores o servlets se realiza a través de Beans de Java o a través de objetos de solicitud, *HttpServletRequest*.

Para cumplir con los objetivos del proyecto, como menciono en las primeras secciones, no es mi responsabilidad el diseño del Front-end de la aplicación, pero es imprescindible conocer cómo funciona Thymeleaf y cómo realizar la comunicación entre estas vistas y los controladores. Una vez construido el controlador, los métodos que contiene y los datos que se quieren mandar a las plantillas, como pueden ser variables o listas, se debe añadir un objeto *Model*, o *ModelAndView*, a los métodos deseados del controlador y añadir atributos a ese objeto. Thymeleaf buscará una vista con el mismo nombre que la que devuelve el método del controlador y en ella se tendrá acceso a los atributos creados. Como se puede ver en la Figura 2.14, se añade el objeto *Model* al método *visualizarValoracion*, se añaden al modelo los atributos *listaValoraciones*, una lista con objetos *Valoracion*, y *puntuacionMedia*, un doble con la media de las puntuaciones, y el nombre de la vista que devuelve es *detalle*.

```
@GetMapping("/valoracion")
public String visualizarValoracion(Model model) {
    List<Valoracion> valoraciones = serviceValoraciones.findAll();
    double media = serviceValoraciones.getAvgPuntuacion();
    model.addAttribute("listaValoraciones", valoraciones);
    model.addAttribute("puntuacionMedia", media);
    return "detalle";
```

Fig. 2.14. Comunicación vista-controlador

Por último, se accede a los atributos añadidos al modelo en la vista que devuelve el método. Para ello, en la vista *detalle.html*, se utilizan las expresiones de Thymeleaf. Existen muchas expresiones, que permiten acceder a variables, iterar sobre colecciones y también crear condicionales, entre otras cosas. En el ejemplo, para mostrar el contenido de la variable *puntuacionMedia*, habrá que utilizar la siguiente expresión en la vista: *th:text = "${puntuacionMedia}"*. Además, para iterar sobre cada valoración que hay en la lista que contiene el otro atributo, *listaValoraciones*, la expresión necesaria es: *th:each = "valoracion:${listaValoraciones}"*.

2.3.6 Solr

Solr es un motor de búsqueda basado en Apache. Está escrito en Java y basado en Lucene, una librería de Java que permite integrar motores de búsqueda verticales. Es una solución gratuita, escalable y de alto rendimiento para la búsqueda y recuperación de información en grandes volúmenes de datos, como webs o bases de datos [28].

El indexado es el concepto básico que hay que entender para saber cómo funciona Solr. Se trata de añadir las palabras clave de todos los documentos que se seleccionan para que Solr recorra a un índice. Este índice es capaz de tratar con datos en diversos formatos, como XML, CSV o archivos Word o PDF. Después, en vez de buscar en el texto, Solr busca las palabras clave en el índice y luego indica los documentos donde se encuentran las palabras clave. A este tipo de índice se le conoce como índice invertido, al basar la búsqueda en las palabras clave y no en la página.

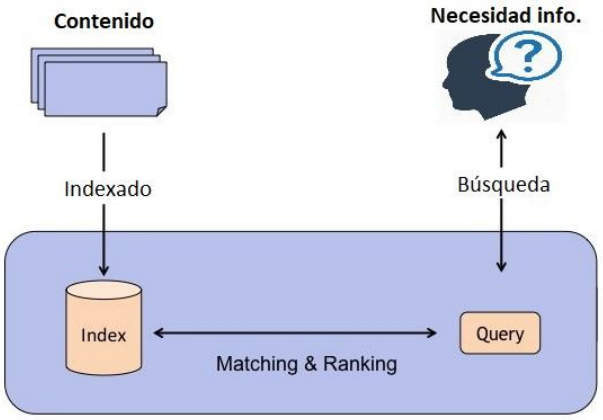


Fig. 2.15. Funcionamiento de Solr [28]

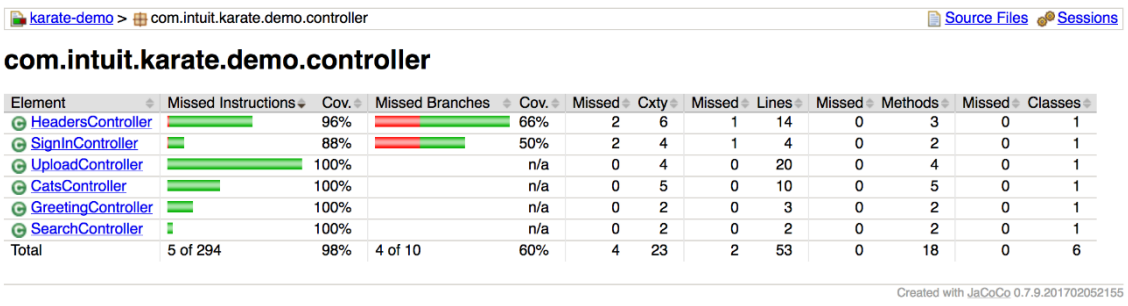
Para integrar Solr en un proyecto de Spring, es imprescindible agregar las dependencias de Solr y Spring Data Solr al archivo POM del proyecto y configurar la conexión a Solr en el archivo de propiedades del proyecto.

Una vez realizada la configuración, se crean entidades, repositorios y servicios muy parecidos a los de Spring Data JPA. En este caso, las entidades representan los documentos que se quieren indexar y buscar en Solr. En los repositorios se deben especificar las operaciones de búsqueda y almacenamiento que se quieren realizar y en los servicios se encapsula la lógica de negocio relacionada con la búsqueda y manipulación de documentos. Por último, se utilizan los métodos creados en los repositorios y servicios para realizar las operaciones de búsqueda que se necesiten en la aplicación, como pueden ser buscar, filtrar u ordenar documentos indexados. Spring Data Solr simplifica este proceso de búsqueda y recuperación de datos gracias a la capa de abstracción que proporciona, muy similar a la de Spring Data JPA también, que reduce la necesidad de escribir código repetitivo y facilita el uso de las funcionalidades que ofrece Solr.

2.3.7 JaCoCo y Mockito

JaCoCo (Java Code Coverage) es una herramienta de código abierto que se utiliza para medir la cobertura de código en aplicaciones Java. El diseño de pruebas o tests es una parte vital del código, ya que ayudan a los desarrolladores a estar seguros de que el código funciona como tiene que funcionar, teniendo en cuenta todos los casos que pueden surgir y todos los caminos que el código puede tomar en función de las acciones del usuario de la aplicación [29].

Especialmente cuando se tiene una aplicación muy grande, JaCoCo es una herramienta muy útil ya que hay muchísimo código y es difícil saber qué partes están bien cubiertas por pruebas y cuáles no. Además de analizar la cobertura en pruebas que existe en la aplicación, JaCoCo genera un informe que muestra estos resultados, como se ve en la Figura 2.16. En la primera columna se especifica el nombre de la clase o método, y para cada una JaCoCo muestra el porcentaje de cobertura y otras estadísticas. Además, al hacer clic en el nombre de la clase o el método se abre una pestaña con el archivo de código completo con las partes cubiertas en verde y las que no están cubierta en rojo.


karate-demo > com.intuit.karate.demo.controller

Source Files Sessions

com.intuit.karate.demo.controller

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
HeadersController		96%		66%	2	6	1	14	0	3	0	1
SignInController		88%		50%	2	4	1	4	0	2	0	1
UploadController		100%		n/a	0	4	0	20	0	4	0	1
CatsController		100%		n/a	0	5	0	10	0	5	0	1
GreetingController		100%		n/a	0	2	0	3	0	2	0	1
SearchController		100%		n/a	0	2	0	2	0	2	0	1
Total	5 of 294	98%	4 of 10	60%	4	23	2	53	0	18	0	6

Created with JaCoCo 0.7.9.201702052155

Fig. 2.16. Cobertura en JaCoCo

Por otra parte, Mockito es un framework también de código abierto que se utiliza para crear objetos simulados que se utilizan para realizar las pruebas unitarias en Java. Los objetos simulados se llaman mocks e imitan la interfaz de un módulo. La finalidad de su uso es engañar a la prueba unitaria, impidiendo que los tests se vayan a otras clases y asegurando que se ejecuta el test sobre la funcionalidad de la clase elegida de forma aislada [30].

Al crear los mocks, es posible especificar cómo deben comportarse y verificar las interacciones con ellos. Algunos de los métodos más conocidos de Mockito son: *Mockito.mock()*, con el que se crea el objeto simulado; *Mockito.when()*, que se utiliza para especificar el comportamiento del mock en una determinada situación y *Mockito.verify()*, que permite verificar si se ha llamado a un método en particular y con qué argumentos.

2.3.8 MariaDB

MariaDB es una base de datos relacional de código abierto muy popular actualmente, sobre todo entre los usuarios de Linux. Cuenta con la licencia GPL, por lo que cualquier usuario que distribuya un software bajo esta licencia debe proporcionar el código fuente del programa y permitir su modificación [31].

El servidor de MariaDB es una versión de MySQL, por lo tanto, tiene las mismas funcionalidades que esta base de datos hasta la versión 5.5, añadiendo además algunas nuevas. Entre las características comunes de estas bases de datos está el lenguaje utilizado para gestionarlas, SQL. MariaDB y MySQL tienen estructuras similares, aunque tienen características diferentes que se deben tener en cuenta a la hora de elegir una u otra para la aplicación. Por todo ello son altamente compatibles entre ellas, pudiendo cambiar una por otra directamente. En la siguiente tabla se muestran las principales diferencias entre una y otra.

TABLA 2.7. DIFERENCIAS ENTRE MARIADB Y MYSQL

Características	MariaDB	MySQL
Rendimiento	Mejor rendimiento y optimización de consultas	Buen rendimiento, pero menos que MariaDB
Replicación	Soporta replicación sincrónica y asincrónica	Sólo soporta replicación asincrónica
Seguridad	Igual que MySQL, añadiendo funciones de codificación diferenciadas y control de acceso basado en roles	Autenticación de usuarios, encriptación de datos, gestión de privilegios y auditoría

Compatibilidad	Amplia	Limitada con algunas herramientas y plugins de MariaDB
Motor de almacenamiento predeterminado	Aria	InnoDB
Asistencia	Bajo suscripción, con tiempo de reacción máximo de treinta minutos	Para clientes de pago

2.3.9 Redmine

Redmine es una herramienta multiplataforma con licencia GNU que se utiliza para la gestión y planificación de proyectos. Es altamente configurable, adaptándose así a todo tipo de empresa y proyecto [32].

Existen otras herramientas más populares entre los desarrolladores, como por ejemplo Jira o Trello, que son las herramientas más utilizadas para la gestión de proyectos. Redmine suele ser utilizado como una alternativa de software libre a Jira, al enfocarse más en proyectos más cortos y menos complejos. Aunque la interfaz de Redmine es mucho menos intuitiva que la de estas herramientas, su curva de aprendizaje es un poco más suave que la de Jira, lo que facilita y agiliza su uso para aquellos que lo utilizan por primera vez.

Redmine cuenta con muchas funciones interesantes para la gestión de proyectos. Entre ellas, se encuentra la Wiki. Con ella, los desarrolladores pueden escribir documentación sobre los distintos componentes del proyecto, evitando así en muchas ocasiones que los trabajadores estén constantemente preguntándose entre ellos cómo funcionan las partes nuevas que se crean. Se puede configurar y personalizar, además de trabajar de forma colaborativa. Al desarrollar un módulo nuevo, se crea una página nueva en la Wiki para que quien quiera informarse sobre él pueda hacerlo. También viene muy bien cuando gente nueva se incorpora a la empresa o al proyecto, pudiendo leerla para estar lo más actualizado posible.

En cuanto a la gestión de tareas y el control del tiempo, Redmine también permite llevar un seguimiento exhaustivo del proyecto. A las tareas creadas se le pueden asignar usuarios, se pueden modificar y se les puede añadir tiempo dedicado. Con todo ello, es posible llevar un control total del proyecto, algo fundamental para los jefes de proyecto. Además, Redmine cuenta también con la posibilidad de crear sprints, en los que añadir tareas, subtareas, casos de test y bugs encontrados. Por último, en función de las tareas registradas, el tiempo dedicado y los puntos de historia de cada tarea, entre otros, Redmine es capaz de generar diagramas de Gantt y gráficas burn-down, muy útiles en metodologías ágiles.

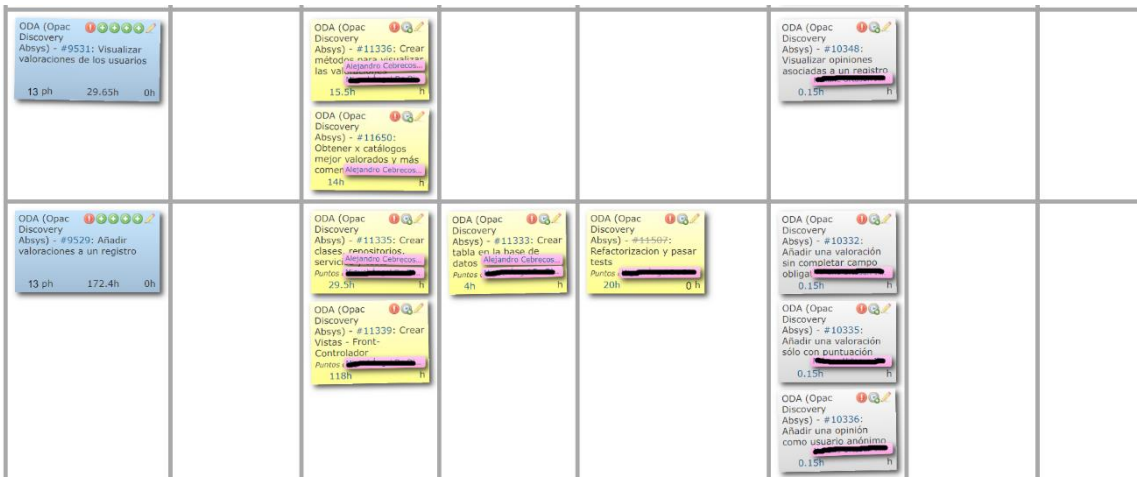


Fig. 2.17. Sprint en Redmine

2.3.10 Subversion

Subversion es un software para el control de versiones centralizado, en el que existe un repositorio que contiene todo el historial de las versiones del software. Al ser centralizado, los desarrolladores deben conectarse a un repositorio central para poder acceder a los archivos del proyecto y a todas sus versiones. Después, los desarrolladores pueden modificar los archivos al mismo tiempo en su repositorio local sin riesgo de borrar cambios que realice otro trabajador al mismo tiempo [33].

Algunas de las operaciones que se pueden realizar en Subversion son las siguientes: *checkout* del repositorio central, para obtener una copia local del proyecto; *update* de la copia de trabajo, que sincroniza la copia actual que el desarrollador tiene en el repositorio local con el estado actual del repositorio central; *commit* de la copia de trabajo, que permite guardar los cambios que se han realizado en local en el repositorio central y creación de ramas para que los desarrolladores puedan trabajar en paralelo sin riesgo de afectar a la rama principal, usando *merge* para fusionar los cambios entre dos ramas para combinar el trabajo realizado en ellas.

Además, Subversion se puede integrar perfectamente con proyectos de Spring, después de configurar un repositorio SVN (Subversion). Como se aprecia en la Figura 2.18, se puede trabajar con Spring y Subversion conjuntamente y así aprovechar todas las funciones disponibles de Subversion para manejar las versiones del proyecto.

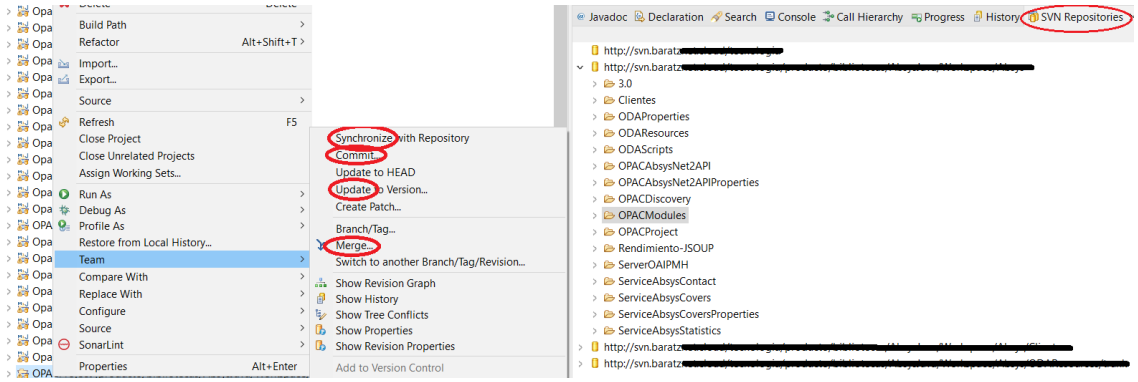


Fig. 2.18. Subversion en IDE de Spring

Es cierto que el gestor de versiones más conocido y utilizado mundialmente es Git. La principal diferencia entre Subversion y Git radica en que, como se explica anteriormente, Subversion es centralizado y Git es distribuido. Para realizar las distintas operaciones en Subversion es necesario estar conectado a la red para acceder al repositorio central, en cambio, en Git no se necesita esta conexión, salvo para algunas operaciones específicas, al disponer cada trabajador de una copia completa del repositorio localmente. En cuanto a rendimiento, al trabajar con grandes repositorios Subversion es más lento que Git, sobre todo en operaciones de *commit* y *update*.

3. CASOS DE USO Y REQUISITOS

La aplicación debe permitir a los usuarios valorar los productos de la biblioteca, además de acceder a las valoraciones que realizan los demás usuarios. Tanto los usuarios autenticados como los anónimos deben poder realizar estas acciones. La diferencia radica en que los usuarios registrados en la biblioteca pueden gestionar las valoraciones previamente realizadas. Además, se debe garantizar al usuario administrador de cada biblioteca la posibilidad de validar las valoraciones de los usuarios, pudiendo ocultarlas o eliminarlas definitivamente.

3.1 Diagrama de casos de uso

Para la realización del diagrama de casos de uso del módulo de valoraciones se tienen en cuenta los tres tipos de usuarios: usuario anónimo, registrado o autenticado y el administrador de la biblioteca. Cada caso de uso tiene un identificador, ubicado en la esquina inferior derecha.

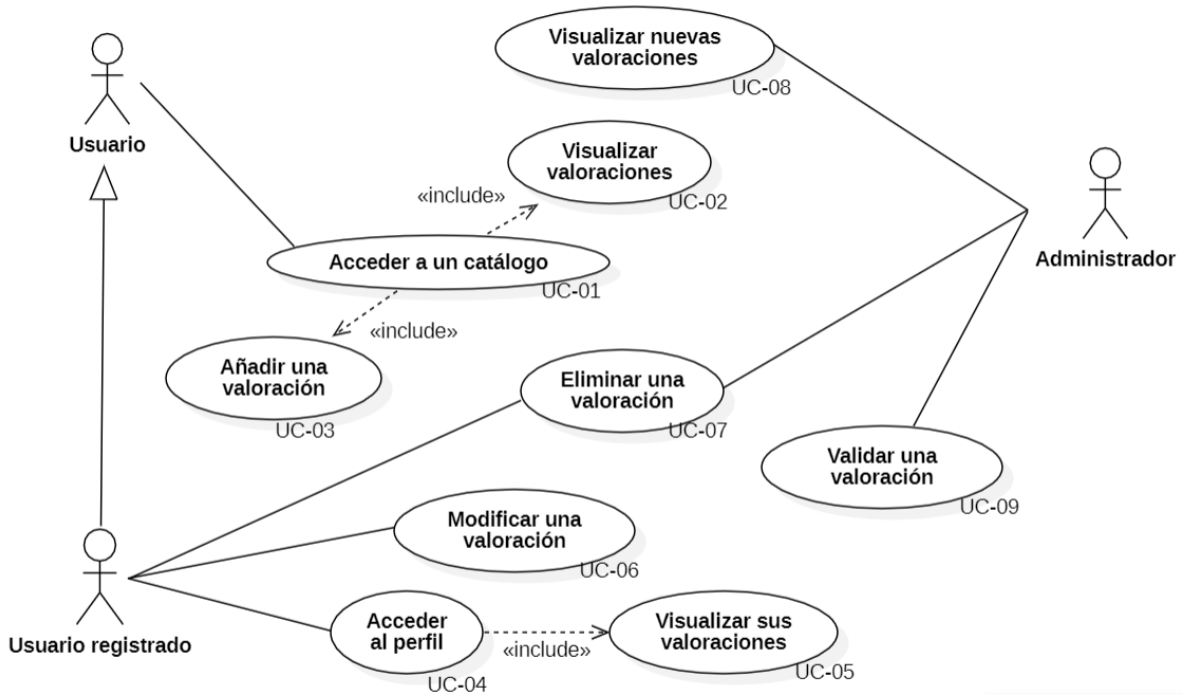


Fig. 3.1. Diagrama de casos de uso

3.2 Requisitos de usuario

Los requisitos de usuario se centran en las necesidades y expectativas de los usuarios de la aplicación. Se definen desde la perspectiva de las personas que utilizarán la aplicación, por ello, tienen la forma: “el usuario podrá…”.

En la Tabla 3.1 se observa el modelo a seguir para definir los requisitos de usuario:

TABLA 3.1. MODELO DEFINICIÓN DE REQUISITOS DE USUARIO

ID	**UR-XX**
Nombre	-
Prioridad	-
Prerrequisitos	-
Descripción	-

En el identificador del requisito, UR representa los requisitos de usuario. Todos ellos tienen un identificador que los hace únicos, un nombre, prioridad, algún prerrequisito para su cumplimiento, si es necesario, y una descripción que proporciona más detalle. La prioridad tiene tres posibles valores, alta, media o baja, que se designan teniendo en cuenta la urgencia del cumplimiento del requisito correspondiente en el tiempo de desarrollo del proyecto.

TABLA 3.2. UR-01 - VISUALIZACIÓN DESDE BÚSQUEDA

ID	**UR-01**
Nombre	Visualización desde búsqueda
Prioridad	Alta
Prerrequisitos	El usuario deberá haber realizado una búsqueda en la aplicación para obtener una lista de resultados.
Descripción	Una vez obtenida una lista de productos, el usuario podrá visualizar la puntuación media de valoraciones que tiene cada producto en la lista, así como el número de comentarios que tiene registrados.

TABLA 3.3. UR-02 - VISUALIZACIÓN DESDE DETALLE

ID	**UR-02**
Nombre	Visualización desde detalle
Prioridad	Alta
Prerrequisitos	El usuario deberá haber seleccionado un producto para acceder a su detalle.
Descripción	En el detalle de un producto, el usuario podrá visualizar la puntuación media de valoraciones que tiene ese producto y una lista de todos los comentarios asociados a ese producto.

TABLA 3.4. UR-03 - VISUALIZACIÓN DESDE EL PERFIL

ID	**UR-03**
Nombre	Visualización desde el perfil
Prioridad	Media
Prerrequisitos	El usuario deberá haber pinchado en la ventana “Mi perfil”.
Descripción	En el área del perfil del usuario, este podrá visualizar todas las valoraciones y comentarios que previamente haya realizado, así como los productos a los que pertenecen.

TABLA 3.5. UR-04 - PAGINACIÓN DE VALORACIONES

ID	**UR-04**
Nombre	Paginación de valoraciones
Prioridad	Media
Prerrequisitos	El usuario deberá haber accedido a un listado de valoraciones.
Descripción	En cualquier listado de valoraciones que el usuario pueda visualizar, si el número de estas excede el límite indicado por el administrador, la lista se dividirá en páginas y el usuario podrá acceder a las siguientes, y anteriores si no se encuentra en la primera.

TABLA 3.6. UR-05 - AÑADIR VALORACIÓN DESDE BÚSQUEDA

ID	UR-05
Nombre	Añadir valoración desde búsqueda
Prioridad	Alta
Prerrequisitos	El usuario deberá haber realizado una búsqueda en la aplicación para obtener una lista de resultados.
Descripción	Si existe algún producto que no tenga ninguna valoración asociada, el usuario podrá acceder al formulario de valoraciones desde la lista de productos para ser el primero en realizar una valoración a ese producto.

TABLA 3.7. UR-06 - AÑADIR VALORACIÓN DESDE DETALLE

ID	UR-06
Nombre	Añadir valoración desde detalle
Prioridad	Alta
Prerrequisitos	El usuario deberá haber seleccionado un producto para acceder a su detalle.
Descripción	En el detalle de un producto, el usuario podrá acceder al formulario de valoraciones para añadir una valoración a ese producto.

TABLA 3.8. UR-07 - AÑADIR VALORACIÓN SIN COMPLETAR FORMULARIO

ID	UR-07
Nombre	Añadir valoración sin completar formulario
Prioridad	Alta
Prerrequisitos	El usuario deberá haber accedido al formulario de valoraciones para añadir una valoración a un producto.
Descripción	Al completar el formulario de valoraciones, el usuario podrá crear una valoración añadiendo solamente la puntuación. Podrá registrarla sólo con puntuación, con puntuación y título y con puntuación, título y descripción.

TABLA 3.9. UR-08 - AÑADIR VALORACIÓN COMO USUARIO ANÓNIMO

ID	**UR-08**
Nombre	Añadir valoración como usuario anónimo
Prioridad	Alta
Prerrequisitos	El usuario deberá haber accedido al formulario de valoraciones para añadir una valoración a un producto.
Descripción	Cualquier usuario podrá añadir una valoración a un producto, sin necesidad de darse de alta con una cuenta en la aplicación. En ese caso, la valoración será contabilizada como anónima.

TABLA 3.10. UR-09 - ELIMINAR VALORACIÓN

ID	**UR-09**
Nombre	Eliminar valoración
Prioridad	Alta
Prerrequisitos	El usuario identificado deberá haber accedido a una valoración de la que sea autor.
Descripción	Una vez seleccionada una valoración como usuario identificado, el usuario podrá eliminarla. Los usuarios anónimos no podrán eliminar una valoración que hayan creado previamente.

TABLA 3.11. UR-10 - MODIFICAR VALORACIÓN

ID	**UR-10**
Nombre	Modificar valoración
Prioridad	Alta
Prerrequisitos	El usuario identificado deberá haber accedido a una valoración de la que sea autor.
Descripción	Una vez seleccionada una valoración como usuario identificado, el usuario podrá modificar cualquier campo, puntuación, título o descripción. Los usuarios anónimos no podrán modificar una valoración que hayan creado previamente.

TABLA 3.12. UR-11 - VISUALIZAR VALORACIÓN COMO ADMINISTRADOR

ID	UR-11
Nombre	Visualizar valoración como administrador
Prioridad	Alta
Prerrequisitos	El usuario administrador deberá haber accedido al listado de valoraciones registradas en su biblioteca.
Descripción	El usuario administrador de una biblioteca tendrá acceso a todas las valoraciones realizadas sobre productos de su biblioteca, y podrá comprobar cada una de ellas para decidir si son apropiadas o no.

TABLA 3.13. UR-12 - ELIMINAR VALORACIÓN COMO ADMINISTRADOR

ID	UR-12
Nombre	Eliminar valoración como administrador
Prioridad	Alta
Prerrequisitos	El usuario administrador deberá haber accedido al listado de valoraciones registradas en su biblioteca.
Descripción	El usuario administrador de una biblioteca tendrá acceso a todas las valoraciones realizadas sobre productos de su biblioteca, y podrá eliminar cualquier valoración si lo considera oportuno.

TABLA 3.14. UR-13 - VALIDAR VALORACIÓN COMO ADMINISTRADOR

ID	UR-13
Nombre	Validar valoración como administrador
Prioridad	Alta
Prerrequisitos	El usuario administrador deberá haber accedido al listado de valoraciones registradas en su biblioteca.
Descripción	El usuario administrador de una biblioteca tendrá acceso a todas las valoraciones realizadas sobre productos de su biblioteca, y podrá validar cualquier valoración si lo considera oportuno u ocultarla si lo desea.

3.3 Requisitos de sistema

Los requisitos de sistema se centran en las características y funcionalidades técnicas que deben implementarse para satisfacer los requisitos de usuario y cumplir con los objetivos del proyecto. Por ello, tienen la forma: “el sistema deberá…”.

En la Tabla 3.15 se observa el modelo a seguir para definir los requisitos de sistema:

TABLA 3.15. MODELO DEFINICIÓN DE REQUISITOS DE SISTEMA

ID	**SR-XX**
Nombre	-
Descripción	-

En el identificador del requisito, SR representa los requisitos de sistema. Cada identificador es único, de esta manera cada requisito de sistema se podrá localizar fácilmente. Cada uno tiene un nombre y una descripción explicando brevemente la necesidad correspondiente.

TABLA 3.16. SR-01 - VISUALIZACIÓN DE VALORACIONES EN LA BÚSQUEDA

ID	**SR-01**
Nombre	Visualización de valoraciones en la búsqueda
Descripción	Tras realizar una búsqueda y obtener productos, el sistema deberá mostrar en cada producto obtenido la media de valoraciones mediante cinco estrellas con o sin relleno y un link que ponga “Ver todas las opiniones”, acompañado del número de opiniones ya registradas en ese producto, que redirigirá al usuario a sus valoraciones. Si un producto no tiene valoraciones registradas, el sistema deberá mostrar un link en el que ponga “Sé el primero en opinar”, que redirigirá al usuario al formulario de creación de un comentario.

TABLA 3.17. SR-02. VISUALIZACIÓN DE VALORACIONES EN EL DETALLE

ID	**SR-02**
Nombre	Visualización de valoraciones en el detalle
Descripción	En el detalle de un producto, al seleccionarlo, el sistema deberá mostrar una lista de todas las valoraciones que tenga ese producto. Además, el sistema deberá mostrar en cada opinión el autor, la fecha y la hora, y un link que ponga “Añadir tu opinión”, que redirigirá al usuario al formulario de creación de una valoración. Si el autor de un comentario es un usuario no identificado en la aplicación o es un usuario identificado, pero sin alias asociado, se mostrará “Usuario anónimo”. En caso de que el usuario esté identificado y tenga un alias asociado, se mostrará su alias.

TABLA 3.18. SR-03 - VISUALIZACIÓN DE VALORACIONES EN EL PERFIL

ID	**SR-03**
Nombre	Visualización de valoraciones en el perfil
Descripción	Cuando el usuario acceda a la sección “Comentarios” en su perfil, el sistema deberá mostrar una lista con todas las valoraciones que el usuario haya realizado previamente, incluyendo todos sus detalles.

TABLA 3.19. SR-04 - PAGINACIÓN

ID	**SR-04**
Nombre	Paginación
Descripción	Cuando el usuario acceda a cualquier listado de valoraciones, si el número de estas excede el indicado por el administrador, el sistema deberá dividir la lista en páginas y deberá permitir al usuario acceder a las siguientes, y anteriores si no se encuentra en la primera, mediante los botones “Anterior” y “Siguiente”.

TABLA 3.20. SR-05 - CREACIÓN DE UNA VALORACIÓN

ID	**SR-05**
Nombre	Creación de una valoración
Descripción	Al acceder al formulario para añadir una valoración nueva, el sistema deberá mostrar al usuario la opción de añadir una puntuación , de 1 a 5 estrellas, y dos cuadros de texto que permitan al usuario escribir un título que resuma su reseña y el comentario que desee. Un usuario no podrá añadir más de una reseña en el mismo producto.

TABLA 3.21. SR-06 - ELIMINACIÓN DE UNA VALORACIÓN

ID	**SR-06**
Nombre	Eliminación de una valoración
Descripción	Si el usuario está identificado en la aplicación, al acceder a una valoración que haya realizado, el sistema deberá mostrar un botón con un icono de una basura que permitirá al usuario borrar su valoración. Esta opción deberá mostrarse tanto si el usuario accede a su valoración desde el detalle del producto o desde su perfil. Al pulsar el botón, el sistema eliminará la valoración.

TABLA 3.22. SR-07 - MODIFICACIÓN DE UNA VALORACIÓN

ID	**SR-07**
Nombre	Modificación de una valoración
Descripción	Si el usuario está identificado en la aplicación, al acceder a una valoración que haya realizado, el sistema deberá mostrar un botón con un icono de un lápiz que permitirá al usuario modificar su valoración, redirigiéndole al formulario con la información que previamente había añadido. Esta opción deberá mostrarse tanto si el usuario accede a su valoración desde el detalle del producto o desde su perfil. Al pulsar el botón de “Guardar”, el sistema modificará la valoración.

TABLA 3.23. SR-08 - VISUALIZACIÓN DESDE ADMINISTRADOR

ID	**SR-08**
Nombre	Visualización desde administrador
Descripción	Al acceder al perfil de administrador, el sistema deberá mostrar la opción "Validar comentarios", que redirigirá al administrador a la lista con todas las valoraciones. En la lista, el sistema mostrará los detalles de todas las valoraciones correspondientes.

TABLA 3.24. SR-09 - ELIMINACIÓN DESDE ADMINISTRADOR

ID	**SR-09**
Nombre	Eliminación desde administrador
Descripción	Una vez el administrador haya accedido a la lista con las valoraciones realizadas por los usuarios, el sistema deberá mostrarle un botón con un icono de una basura en cada valoración que permitirá al administrador eliminar la valoración.

TABLA 3.25. SR-10 - VALIDACIÓN DESDE ADMINISTRADOR

ID	**SR-10**
Nombre	Validación desde administrador
Descripción	Una vez el administrador haya accedido a la lista con las valoraciones realizadas por los usuarios, el sistema deberá mostrarle un botón para cada valoración, que puede contener el icono de un tick o el de una cruz. Si aparece la cruz, la valoración no está validada, y pulsando el botón el icono cambiará al tick y la valoración se validará. Si aparece el tick, la valoración está validada. Pulsando el botón, el icono cambiará a la cruz y la valoración no se mostrará en la aplicación, pero no será eliminada.

3.4 Matrices de trazabilidad

En esta sección se muestra la relación entre los requisitos de usuario y los de sistema y entre los casos de uso y los requisitos de usuario utilizando matrices de trazabilidad, diseñadas con la herramienta Microsoft Excel.

- Requisitos de usuario y requisitos de sistema:

REQUISITOS DE USUARIO Y DE SISTEMA		REQUISITOS DE SISTEMA									
		SR-01	SR-02	SR-03	SR-04	SR-05	SR-06	SR-07	SR-08	SR-09	SR-10
REQUISITOS DE USUARIO	UR-01	X									
	UR-02		X								
	UR-03			X							
	UR-04				X						
	UR-05					X					
	UR-06					X					
	UR-07					X					
	UR-08					X					
	UR-09						X				
	UR-10							X			
	UR-11								X		
	UR-12									X	
	UR-13										X

Fig. 3.2. Matriz de trazabilidad entre requisitos de usuario y requisitos de sistema

- Casos de uso y requisitos de usuario:

REQUISITOS DE USUARIO Y CASOS DE USO		CASOS DE USO								
		UC-01	UC-02	UC-03	UC-04	UC-05	UC-06	UC-07	UC-08	UC-09
REQUISITOS DE USUARIO	UR-01	X	X							
	UR-02		X							
	UR-03				X	X				
	UR-04		X			X			X	
	UR-05			X						
	UR-06			X						
	UR-07			X						
	UR-08			X						
	UR-09							X		
	UR-10						X			
	UR-11								X	
	UR-12							X		
	UR-13									X

Fig. 3.3. Matriz de trazabilidad entre requisitos de usuario y casos de uso

4. DISEÑO DEL SISTEMA

En este apartado, se explica de la mejor manera posible el diseño del sistema y las razones por las que se han tomado las principales decisiones, en la mayoría de los casos mostrando partes del código o scripts muy similares a los diseñados, al ser imposible mostrar el código real, ya que pertenece a la empresa.

4.1 Estructura del proyecto

Este módulo de valoraciones es un añadido más al proyecto ya existente, ODA. Por ello, la estructura del proyecto ya estaba creada antes de iniciar el desarrollo de este módulo. La estructura de la aplicación se divide en tres proyectos principales, *ODAProperties*, *ODAResources* y *OPACProject*.

4.1.1 Propiedades y recursos

ODAProperties contiene todos los entornos de ejecución de la aplicación. Estos entornos son los que permiten a los distintos grupos de trabajo de la empresa trabajar a la vez sobre la misma versión del proyecto y modificar lo que sea conveniente sin intervenir en el trabajo de los demás. Existen entornos para el grupo de desarrollo, el grupo de test y el grupo de producción, entre otros. Para cada uno de estos entornos, *ODAProperties* almacena todos los archivos de configuración. Además, uno de los entornos, local, se utiliza para que cada trabajador realice cambios y haga pruebas en su entorno local antes de liberar cualquier versión de la aplicación.

```
comment{
        #activar los comentarios
        #(si esta propiedad esta a false no se veran los comentarios
        #ni en la parte usuario ni en la manager)
        enabled=true
        #activar las valoraciones
        valoraciones.enabled=true
        #comentarios validados automaticamente
        auto.valid=true
        #estrellas de los comentarios 5,10 etc
        #(no se puede modificar si ya hay comentarios en la tabla)
        valoraciones.stars=5
        #n de comentarios por pagina
        num.page=10
        #el top x con mas media
        valoraciones.topMedia=10
        #el top x de los mas comentados
        valoraciones.topComment=10
        #aceptar comentarios anonimos
        anonymous=true
```

Fig. 4.1. Propiedades de configuración

Como se observa en la Figura 4.1, es necesario crear algunas propiedades en el archivo de configuración del proyecto para su correcto funcionamiento e integración en cada biblioteca. Una vez que las bibliotecas dispongan de la versión completa de la aplicación que incluya el módulo de valoraciones, estas propiedades las podrán modificar a su gusto los administradores de las bibliotecas.

La primera propiedad, *enabled*, se utiliza para permitir la aparición de los comentarios en la aplicación. De la misma manera, *valoraciones.enabled* se utiliza para permitir la aparición de las valoraciones. El administrador de una biblioteca puede cambiar el valor de estas propiedades a false si no quiere que en su biblioteca esté disponible el módulo de valoraciones para los usuarios.

La propiedad *auto.valid* es crucial en el funcionamiento de la aplicación. Si su valor es true, los comentarios nuevos se validan automáticamente y están disponibles en la aplicación. En cambio, si es false, la validación del administrador es necesaria para que estén disponibles.

Las siguientes cuatro propiedades creadas no son tan importantes porque no cambian el funcionamiento del módulo, sino que lo personalizan. La primera de esas cuatro sirve para definir el número máximo de estrellas con las que un usuario puede puntuar un producto, la segunda define el número máximo de comentarios que se muestran en cada página, y las últimas dos se utilizan en un servicio que se explica más adelante, que consiste en obtener los productos con mejor valoración y con el mayor número de comentarios. Este servicio no se exige como requisito, pero se implementa en caso de que los administradores de las bibliotecas quieran disponer de él en un futuro, ya que es un servicio muy común en los módulos de valoraciones similares al que se desarrolla.

Por último, la propiedad *anonymous* también es muy importante, ya que permite al desarrollador habilitar o deshabilitar los comentarios de los usuarios anónimos.

El proyecto para los recursos, *ODAResources*, contiene todas las plantillas de la aplicación para construir el Front-end. Sobre este proyecto trabaja el equipo de Front-end de la empresa, aunque es imprescindible conocer su funcionamiento para poder realizar satisfactoriamente la comunicación entre el Front-end y el Back-end de la aplicación. Esta comunicación se realiza mediante la implementación del patrón de diseño Spring MVC.

4.1.2 Proyecto principal

El proyecto principal de la aplicación se llama *OPACProject*, y contiene a todos los demás proyectos o módulos, por ello se le denomina proyecto padre. En el archivo POM de este proyecto se define la estructura del proyecto y se añaden todas las dependencias necesarias para su despliegue.

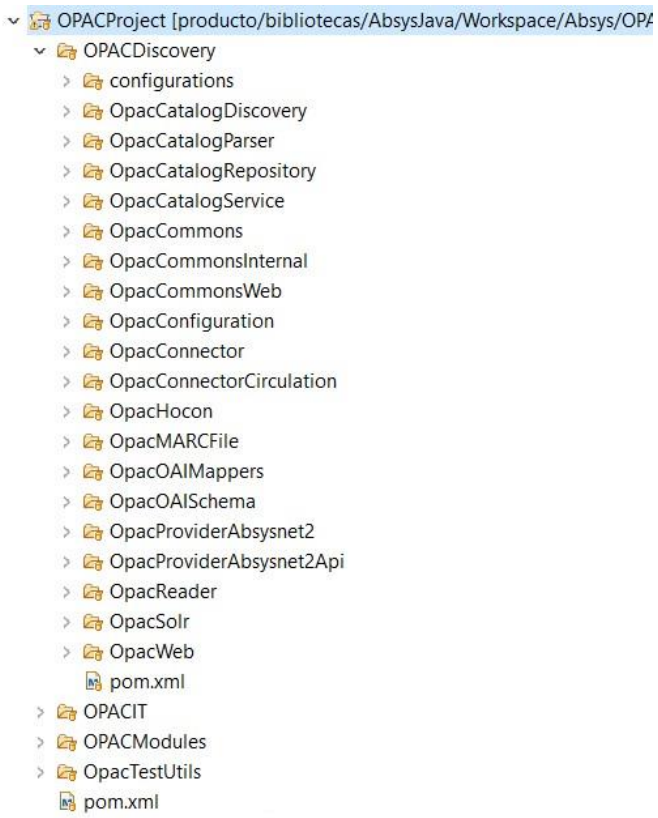


Fig. 4.2. Estructura del proyecto principal

Como se observa en la Figura 4.2, *OPACProject* contiene muchas carpetas dentro, que son más proyectos. Para desarrollar este módulo se ha trabajado en distintos proyectos que se encuentran dentro de la carpeta *OPACDiscovery*. Siguiendo con la política de la empresa, cada parte que se ha desarrollado para el módulo de valoraciones tiene que estar almacenada en el proyecto donde se almacenan esa misma parte, pero de otro módulo. Por ello, todos estos proyectos contienen código imprescindible para el desarrollo del módulo y se comunican entre ellos, añadiendo las clases necesarias en cada archivo con el comando import de Java.

En el proyecto *OpacCatalogRepository* se encuentran la definición de los objetos de la aplicación, llamados modelos, y los repositorios, en los que se crean los métodos de cada modelo que se necesitan. En *OpacCatalogService*, se encuentra la implementación de los servicios que utilizan los métodos creados en los repositorios. Por último, todos los controladores que se necesitan para el despliegue de la aplicación se encuentran en el proyecto *OpacWeb*, que es el proyecto que se despliega con el servidor de Spring. Todos los demás proyectos también contienen información vital para la aplicación, como, por ejemplo, *OpacHocon*, en el que se recogen los valores de las propiedades antes definidas y *OpacCommonsWeb*, en el que se crean las variables que contienen todas las rutas posibles a las que se puede acceder en la aplicación, para utilizarlas posteriormente en los controladores.

Todos estos proyectos que heredan la estructura del padre, *OPACProject*, tienen una estructura muy similar, compuesta por cuatro carpetas, como se observa en la Figura 4.3.

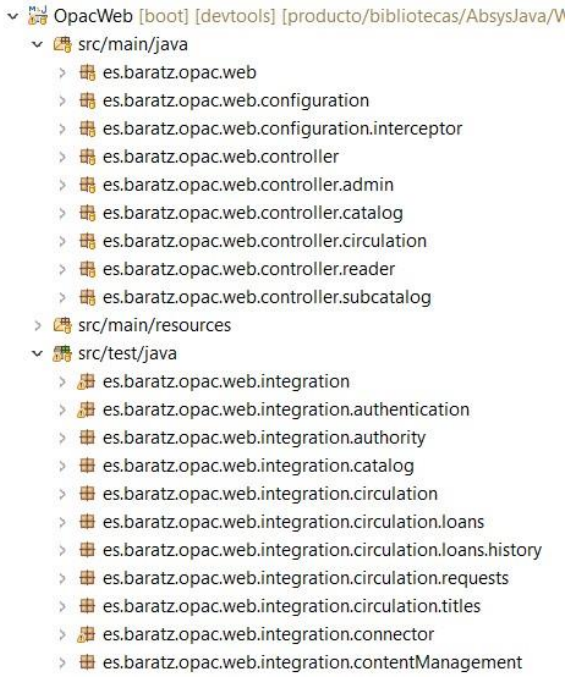


Fig. 4.3. Estructura de cada proyecto

La carpeta *src/main/java* contiene paquetes que almacenan los archivos java en los que está definido el código de cada proyecto. En *src/main/resources*, se guardan los archivos de configuración específicos de ese proyecto. Todos los archivos que contienen los tests y las pruebas de código se guardan en *src/test/java* y los archivos de configuración que se usan para realizar los tests se almacenan en *src/test/resources*. Dentro de las carpetas terminadas en */java*, los archivos se agrupan en paquetes. Los nombres de todos los paquetes de la aplicación empiezan por *es.baratz*, seguidos del nombre del proyecto correspondiente y del nombre que se utiliza para definir su contenido.

Todos estos proyectos son proyectos Maven, por lo que cada uno posee un archivo POM en el que se define la estructura y configuración del proyecto. En el archivo POM del proyecto padre se añaden todas las dependencias que necesitan todos los demás proyectos para su desarrollo. Luego, en el POM de cada proyecto específico se añaden más dependencias en caso de necesitarse solamente en ese proyecto. Algunas de las dependencias que se añaden en el POM padre son las siguientes, Figura 4.4:

- *Spring-boot-starter-aop*, que proporciona soporte para la programación orientada a aspectos en la aplicación.
- *Spring-boot-starter-data-jpa*, que proporciona soporte para la persistencia de datos utilizando Java Persistence API (JPA) con Hibernate como proveedor de persistencia, con el fin de poder interactuar con la base de datos.
- *Spring-security-test*, para realizar las pruebas de seguridad en la aplicación, como autenticación y autorización.
- *Spring-boot-starter-test*, que proporciona soporte para escribir las pruebas de código de la aplicación e incluye librerías como Mockito.

```
<!-- springboot AOP -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-aop</artifactId>
</dependency>

<!-- Spring data -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>

<!-- Testing -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

Fig. 4.4. Dependencias POM padre

Además de añadir las dependencias que sean necesarias, en los demás proyectos que heredan la configuración del proyecto padre se incluyen en el archivo POM dependencias con los nombres de otros proyectos de los que se necesita importar clases. Por ejemplo, en el proyecto *OpacWeb*, se incluye como dependencia el proyecto *OpacCommonsWeb*, ya que posee todas las rutas que se necesitan añadir en los controladores, Figura 4.5.

```
<dependency>
    <groupId>es.baratz.opac</groupId>
    <artifactId>OpacCommonsWeb</artifactId>
    <version>1.1.7-SNAPSHOT</version>
</dependency>
```

Fig. 4.5. Dependencia de otro proyecto

4.2 Administración de usuarios

El manejo de usuarios es un módulo que no pertenece a ODA. Toda la información relativa al inicio de sesión se almacena en otra base de datos en Oracle. Aun así, es imprescindible acceder a esta información desde la aplicación.

La autenticación de los usuarios se realiza vía petición HTTPs al API REST de otro producto de la empresa, AbsysNet. Para acceder a los datos sobre los usuarios y la sesión iniciada, es necesario utilizar los servicios ya creados en el proyecto de Spring que llaman a este API. Para ello, desde la aplicación se crea un proveedor de autenticación propio de Spring Security, *CustomAuthenticationProvider*, que realiza una petición XML con autenticación básica al API de AbsysNet. Este objeto crea una lógica personalizada para autenticar a los usuarios. Tras validar el usuario, AbsysNet devuelve un objeto usuario con un token. Para el mantenimiento de una sesión iniciada tras cerrar el navegador o reiniciar el dispositivo, Spring Security envía una cookie al navegador cuando el usuario inicia sesión en la aplicación. Esta cookie se almacena en el navegador durante un periodo de tiempo parametrizado, concretamente dos semanas. La próxima vez que el usuario inicia sesión en la aplicación, Spring Security verifica y valida la cookie almacenada y la sesión se inicia automáticamente. En las siguientes figuras se muestra el funcionamiento del inicio de sesión tanto para lectores como para el usuario administrador.

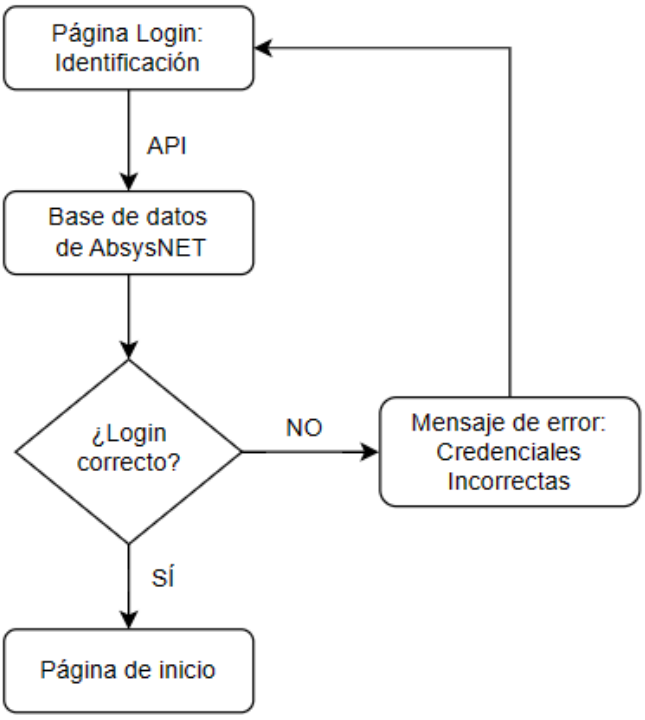


Fig. 4.6. Diagrama de flujo inicio de sesión lectores en ODA

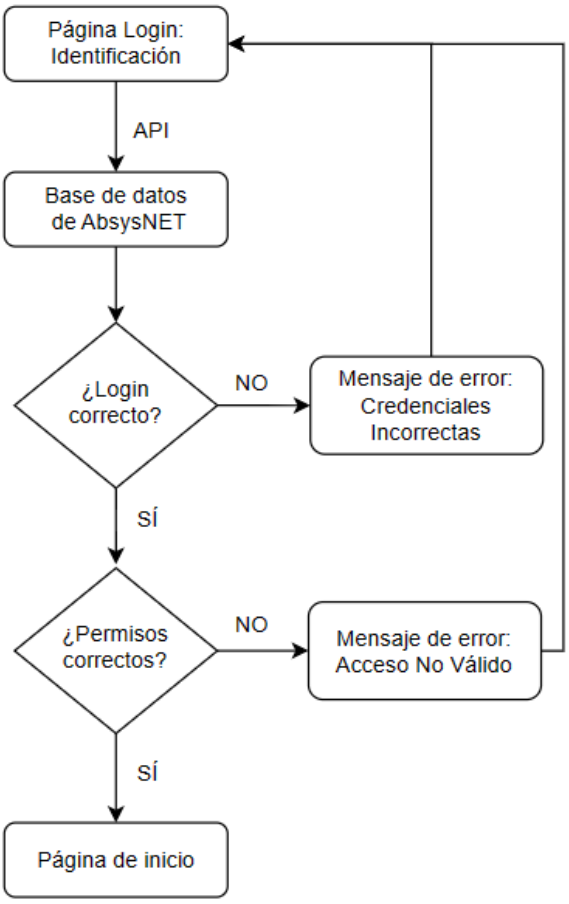


Fig. 4.7. Diagrama de flujo inicio de sesión administrador en ODA

Los atributos más importantes del usuario en la aplicación, que se utilizan en el desarrollo de este proyecto incluyen el identificador, un número de nueve dígitos, y un alias opcional, que es el atributo del lector que se muestra al resto de usuarios al realizar una valoración. Para acceder a esta información sobre el usuario, se utilizan los métodos disponibles en la aplicación que, recibiendo el objeto que guarda la información del usuario autenticado, devuelven el identificador y el alias.

En este módulo se contemplan tres tipos de usuario: el administrador de una biblioteca, el usuario autenticado y el usuario anónimo. Para la funcionalidad de las valoraciones, el usuario autenticado y el usuario anónimo pueden realizar prácticamente las mismas acciones. La única diferencia entre estos usuarios es que, si está autenticado, el usuario puede acceder a todas las valoraciones que ha realizado desde su perfil, pero a la hora de realizar un comentario tienen los mismo permisos que los usuarios anónimos.

En cambio, para los métodos que sólo pueden ser accedidos por el administrador de la biblioteca, se necesita comprobar que el rol del usuario autenticado es el de administrador. Para ello, es necesario utilizar anotaciones que proporciona Spring Security. En estos métodos, se añade *@PreAuthorize*, seguido de los métodos incluidos en los servicios ya creados en la aplicación que reciben la información del usuario autenticado y comprueban su rol. En caso de no ser administrador, el acceso a estos métodos es denegado.

4.3 Acceso y almacén de catálogos

El almacenamiento de toda la información sobre los catálogos también se realiza en una base de datos diferente a la que utiliza ODA. De la misma manera que con la información de inicio de sesión, es necesario utilizar los métodos y servicios ya creados en la aplicación para el desarrollo de este módulo.

En este caso, ODA utiliza un motor de búsqueda para indexar las búsquedas relacionadas con los catálogos. Este motor de búsqueda es Solr. La aplicación utiliza dos colecciones de Solr, una para las autoridades y otra para los catálogos, o productos. Las autoridades se refieren a toda la información sobre bibliotecas y autores. Los productos, en cambio, se refieren a cada ejemplar guardado en la aplicación, sea un libro, un libro electrónico o una película. Al realizar una consulta en Solr, el resultado se devuelve en formato JSON. Por ello, para acceder a los atributos que se necesitan para el desarrollo de este módulo, como son el identificador de un producto y su título, es necesario conocer cómo es la estructura de un JSON.

La estructura de un JSON consiste en pares clave-valor. Las claves son cadenas de texto y los valores pueden ser diferentes tipos de datos. Para acceder a los valores desde Spring, una vez se tiene el JSON resultado, es necesario seguir algunos pasos. En la Figura 4.8, se describe el proceso de obtención del nombre de un producto para mostrarlo en la lista de comentarios.

```
public String obtenerTitulo(String idCat) {

    //CREAR VARIABLE PARA GUARDAR EL NOMBRE DEL CATALOGO
    String titulo = "";

    //LLAMAR AL METODO DEL SERVICIO YA CREADO QUE DEVUELVE UN CATALOGO PROPORCIONANDO SU IDENTIFICADOR
    Catalogo catalogo = servicioCatalogo.findByIdCat(idCat);

    //SI EXISTE EL CATALOGO, OBTENER LA CLAVE EN LA QUE SE ALMACENA SU NOMBRE
    if (catalogo != null) {
        CampoCatalogo campoCatalogo = tfields.get("titulo").get(0);

        //SI EXISTE ESA CLAVE Y SU VALOR, AÑADIR EL VALOR A LA VARIABLE CREADA ANTERIORMENTE
        if ((campoCatalogo != null) && (campoCatalogo.getSubFields().get(0) != null)) {
            SubcampoCatalogo subcampoCatalogo = campoCatalogo.getSubFields().get(0);
            titulo = subcampoCatalogo.getSubfieldValue();
        }
    }
}
```

Fig. 4.8. Obtener nombre de un producto

4.4 Diseño físico de la tabla en base de datos y modelo

Para añadir este módulo a la aplicación se necesita crear otra tabla en la base de datos de ODA. Esta tabla se llama *comentario* y se crea con el siguiente script, Figura 4.9:

```
DROP TABLE IF EXISTS comentario;
CREATE TABLE comentario (
  id bigint(20) NOT NULL AUTO_INCREMENT,
  titulo varchar(25) NOT NULL,
  descripcion varchar(500) NOT NULL,
  idCat varchar(255) NOT NULL,
  puntuacion int(10),
  lector bigint(10),
  fecha datetime DEFAULT NULL,
  validado tinyint(1) NOT NULL DEFAULT 0,
  PRIMARY KEY (id),
  UNIQUE KEY comentario_uk_id_idCat_idx (id,idCat)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Fig. 4.9. Script de creación de la tabla en la base de datos

Cada comentario tiene un *id* que lo identifica. Este identificador es un número entero largo (bigint), para asegurar que no se acaban los números para identificar los comentarios en caso de existir gran cantidad de ellos. *Id* es la clave primaria, por ello no puede ser nulo y, además, se añade *AUTO_INCREMENT* para indicar que cada vez que se crea un nuevo registro de esta tabla, se le asigna automáticamente un valor incremental a esta columna.

Al añadir una valoración, el usuario deberá proporcionar una puntuación al producto y, opcionalmente, podrá escribir un breve título y una explicación de su reseña. Por ello, se añaden otras tres columnas a la tabla: *título*, *descripción* y *puntuación*. Tanto *título* como *descripción* son cadenas de texto con longitud limitada. El título del comentario está limitado a 25 caracteres y la descripción está limitada a 500 caracteres. La columna *puntuación* se refiere a las estrellas con las que el usuario valora un producto.

La columna *idCat* se refiere al identificador del producto sobre el que se realiza el comentario. Dependiendo del tipo de catálogo, la longitud del identificador puede variar, por ello se selecciona 255 como longitud de la cadena de texto. Además, esta columna se añade como clave única junto al identificador del comentario para asegurar que esta combinación de valores no se puede repetir en más de un comentario. La columna *lector* sirve para identificar al lector que realiza el comentario. Si el lector que realiza el comentario es anónimo, el valor de la columna será 0. Por lo tanto, ni *idCat* ni *lector* pueden ser claves primarias de esta tabla, ya que puede haber más de un comentario por producto, y también puede haber comentarios con el mismo valor para la columna *lector* para el mismo producto, 0, en caso de ser de lectores anónimos.

La columna *fecha* almacena la fecha y hora en la que se guarda un comentario en la base de datos, justo cuando un usuario lo realiza, por ello es de tipo datetime. Esta columna tampoco puede ser clave primaria de la tabla, ya que más de un comentario se puede haber

realizado en el mismo instante. Por último, la columna *validado* almacena un número, que será 0 si el comentario no está validado y 1 en caso de estarlo.

En la siguiente figura se muestra el diagrama de flujo que explica el proceso de recuperación de las valoraciones de un producto de la base de datos para mostrarla en la aplicación:

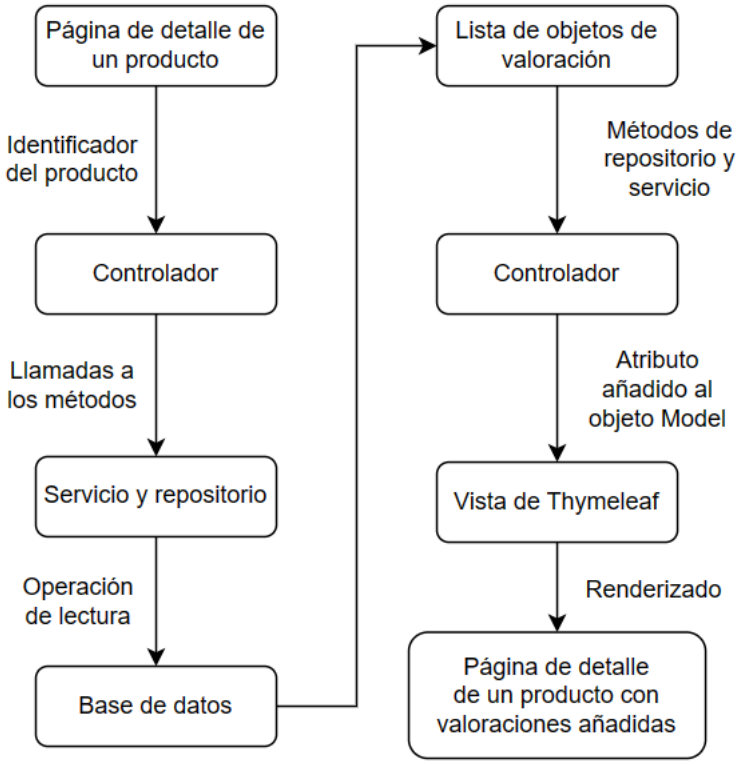


Fig. 4.10. Diagrama de flujo recuperación de valoraciones de la base de datos

Para relacionar esta tabla creada en la base de datos con el objeto que se utiliza en Spring MVC, es imprescindible crear el modelo. Para ello, se utilizan las anotaciones explicadas en la sección 2.3.4 de este documento. La clase de Java correspondiente al modelo se debe anotar con *@Entity* y con *@Table*, seguido del nombre de la tabla que se ha introducido en la base de datos. Además, para agilizar el proceso de modificación de un comentario, se utiliza la anotación *@DynamicUpdate*, que le indica a Hibernate que, al modificar un comentario existente, genere una sentencia de actualización que incluya solo los campos modificados en lugar de todos los campos para optimizar la operación.

Después, se declaran todas las columnas como variables, anotando la que sea clave primaria con *@Id*. En este caso, se utiliza también *@GeneratedValue* para generar automáticamente los valores de la clave primaria de la entidad. En este paso, es de vital importancia asignarles a las variables exactamente el mismo nombre con el que se han declarado las columnas en la base de datos, incluyendo mayúsculas y minúsculas, por el contrario, el mapeo entre la entidad y la tabla en la base de datos no se realiza correctamente. En caso de la fecha, como al crear la tabla automáticamente es nula, en el

modelo asignamos la fecha actual (*LocalDateTime.now()*) para almacenar la fecha y hora exacta en la que se realiza un nuevo comentario.

Por último, se crean los métodos getters y setters, que son importantes para utilizarse en los controladores y en las vistas de Thymeleaf, como se observa en la Figura 4.11. Además, se añade un constructor para poder crear en otros archivos nuevos objetos de comentario sin necesidad de utilizar los setters.

```
public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getTitulo() {
    return titulo;
}

public void setTitulo(String titulo) {
    this.titulo = titulo;
}

public String getDescripcion() {
    return descripcion;
}

public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}
```

Fig. 4.11. Métodos getters y setters

4.5 Repositorio y servicio

Tras crear el modelo para mapear la tabla *comentario* en la base de datos con los objetos de comentario que se utilizan en Spring MVC, se crea el repositorio y el servicio que implementan los métodos necesarios para trabajar con los objetos de comentario. El repositorio se debe anotar con *@Repository* y el servicio con *@Service*, para indicarle a Spring la función que cumplen esas interfaces o clases de Java.

El repositorio de comentarios extiende de dos repositorios de Java que tienen métodos ya creados e implementados, *JpaRepository* y *PagingAndSortingRepository*, además de incluir en él métodos personalizados que se necesitan para el desarrollo del módulo. Con el repositorio de JPA es posible disponer de los métodos necesarios para realizar las operaciones CRUD (crear, leer, actualizar y eliminar) que se necesitan para interactuar con la base de datos. El otro repositorio permite crear objetos de página para poder recuperar los datos divididos en páginas y así trabajar con ellos de manera más cómoda.

En el servicio de comentarios se incluye la lógica de negocio. Se encarga de dirigir las operaciones relacionadas con el modelo y utiliza los métodos definidos en el repositorio para acceder a los datos necesarios, complementándose con otros métodos creados para ajustar el desarrollo a las funcionalidades pedidas.

4.5.1 Métodos de lectura

4.5.1.1 Encontrar por atributo

En el repositorio JPA de Spring existe una convención para realizar un tipo de consultas personalizadas sin tener que escribir consultas SQL. Estos son los denominados métodos *findBy*. Para el desarrollo de este módulo se utiliza esta convención para encontrar valoraciones por el valor de una o más propiedades específicas. Siguiendo con la convención, se crean los métodos: *findById*, *findByIdCat*, *findByLector* y *findByIdCatAndLector*.

FindById obtiene el comentario que tiene por identificador el parámetro añadido al llamar al método. Para poder modificar o eliminar un comentario, es necesario disponer de su clave primaria, el identificador, por ello este método es de vital importancia.

FindByIdCat obtiene una página de objetos de comentario que tengan como atributo *idCat* la misma cadena de caracteres que el parámetro añadido al llamar al método. Es decir, este método sirve para obtener todos los comentarios de un mismo producto. Se utiliza al acceder al detalle de un producto, donde se esperan obtener todos los comentarios de ese producto.

FindByLector funciona de la misma manera que el método anterior, la diferencia es que el parámetro introducido es el identificador de un lector. Se utiliza al acceder al perfil de un usuario, en la sección de comentarios, donde se muestran todos los comentarios que ha realizado.

Por último, el método *findByIdCatAndLector* se utiliza en un método auxiliar para comprobar una condición importante que se exige en los requisitos, que un usuario no pueda añadir más de una reseña por producto. El método auxiliar llama a este método, que devuelve un identificador en caso de que exista un comentario con los identificadores de producto y lector añadidos como parámetros. En caso de que el método devuelva un identificador, se declara como verdadera una variable de tipo booleana que impide al usuario realizar otro comentario en ese producto.

4.5.1.2 Obtener resultado de una consulta SQL

En caso de que se desee añadir una consulta SQL en Spring para obtener un resultado personalizado, se utiliza la anotación *@Query*, seguida de la consulta en cuestión. Algunos ejemplos de estas consultas se detallan en la Figura 4.12. En la primera consulta de la figura se obtiene el número de comentarios de un producto y en la segunda se obtiene la media de puntuaciones de un producto. Estos dos métodos se utilizan para la parte de administrador porque tienen en cuenta todos los comentarios, estén validados o no. Por ello, es necesario crear también los mismos métodos que tengan en cuenta sólo los comentarios validados, que son los que se muestran en la aplicación. Las dos últimas consultas se deciden crear después de pensar que serán de gran utilidad para las bibliotecas. Estos dos últimos métodos devuelven una página con los productos mejor valorados y más comentados de una biblioteca. El número de productos que devuelven estos dos métodos se configura en las propiedades del módulo, como se explica en la sección 4.1.1, que por defecto son 5.

```
@Query("select count(*) from Comentario c where c.idCat = ?1")
public long contarPorCatalogo(String idCat);

@Query("select round(coalesce(avg(c.puntuacion), 0), 1) from Comentario c where c.idCat = ?1")
public double mediaValoracion(String idCat);

@Query("select idCat from Comentario Group By idCat Order By round(coalesce(avg(puntuacion), 0), 1) desc")
public Page<String> topValoraciones(int x, Pageable pageable);

@Query("select c.oaiIdentifierB64 from Comentario c Group By c.idCat Order By count(c.id) desc")
public Page<String> topReseñas(int x, Pageable pageable);
```

Fig. 4.12. Consultas SQL en el repositorio

En las dos primeras consultas, se utiliza al final la expresión *c.idCat = ?1* para indicar que el identificador del producto que tiene que se le proporciona a la consulta es el parámetro de tipo String que se le añade al método. La primera consulta devuelve el número de comentarios que tiene registrados un producto específico, del que se proporciona su identificador. Su uso es clave para mostrar el número de comentarios que tiene un producto al visualizarlo en la búsqueda o al entrar en su detalle y, además, se utiliza también para comprobar si mostrar en pantalla “Sé el primero en comentar”, en caso de que este método devuelva 0. La segunda consulta se emplea para conseguir un método que devuelva la media de las puntuaciones que los usuarios le han otorgado a un producto, para mostrar esa media también en la búsqueda o al entrar en el detalle de un producto. Se utiliza *avg* para calcular esta media, *coalesce* para manejar los casos en los que un producto no tenga ninguna puntuación asociada, en ese caso la consulta devolverá 0, y *round* para redondear el resultado a un decimal.

En las dos últimas consultas, utilizadas para los métodos que devuelven los productos mejor valorados y más comentados, se utilizan las expresiones *Order By* y *desc* para que la consulta devuelva los resultados en orden descendente, es decir, los que más puntuación media o número de comentarios tengan, primero. Además, se utiliza *Group By* para agrupar los resultados según el identificador del producto, para obtener un resultado por

cada producto encontrado. En la primera de estas dos, se utiliza la misma estructura que en la consulta diseñada para calcular la media de puntuaciones de un producto. En la segunda, se cuentan los comentarios que tiene cada producto buscando todos los identificadores de comentarios que existen. Por último, a ambos métodos se les añade como parámetros un número entero *x*, que en el servicio se obtiene consultando el valor de la propiedad explicada anteriormente, y un objeto de tipo *Pageable*, para poder devolver el resultado de la consulta como una página.

4.5.2 Métodos para añadir, modificar y eliminar valoraciones

Las otras tres operaciones CRUD se realizan con los métodos que proporciona el repositorio JPA de Spring. Tanto para añadir como para modificar un comentario, se utiliza el método *save()* del repositorio. Spring comprueba automáticamente si el comentario añadido como parámetro al método tiene un identificador asignado. En ese caso, Spring Data JPA asume que el comentario ya existe y ejecuta una consulta para buscar la entidad existente en la base de datos, utilizando el identificador del comentario proporcionado, y actualiza los valores de sus campos con los valores que posee el comentario proporcionado. Por el contrario, si el comentario no tiene un identificador asignado, considera que es una nueva entidad y la inserta en la base de datos. Por defecto, este método devuelve la entidad actualizada o insertada en la base de datos.

Para eliminar un comentario, el método que se utiliza es *deleteById()*. Este método requiere de un identificador como argumento, con el que Spring Data JPA busca una entidad en la base de datos. Si se encuentra el identificador en la base de datos, automáticamente se realiza una consulta para borrar esa entidad. En caso contrario, el método no devuelve ningún error, simplemente no se realiza ninguna operación. En ambos casos, el método no devuelve ningún valor.

4.6 Controlador

Por último, en el controlador se manejan las solicitudes HTTP del cliente y se utilizan los métodos creados en el servicio para formular una respuesta acorde a la solicitud, que se traslada a las diferentes vistas para visualizarla en la web. Para indicarle a Spring que una clase es un controlador, se utiliza la anotación *@RestController*. En la Figura 4.13, se muestra el modelo a seguir de los métodos creados en el controlador.

```
@GetMapping("Ruta desde la que se accede a este método")
public ModelAndView metodoGuardar(/*Parámetros a añadir*/ ModelAndView model) {

    //LLAMAR A ALGÚN MÉTODO AUXILIAR SI ES NECESARIO

    //LLAMAR A LOS MÉTODOS CREADOS EN EL SERVICIO
    Comment com = new Comment("Muy bueno", "El mejor libro que he leído desde hace mucho",
            "identificadorCatalogo", 5, 1L, null);
    Comment nuevo = commentService.save(nuevo);

    //AGREGAR OBJETO AL MODELO PARA QUE ESTÉ DISPONIBLE EN LA VISTA
    model.addObject("comentario", nuevo);

    //ESTABLECER NOMBRE DE LA VISTA A RENDERIZAR
    model.setViewName("guardar_comentario");

    //DEVOLVER EL OBJETO DE MODELO
    return model;
}
```

Fig. 4.13. Modelo de controlador utilizado

Existen tres páginas distintas desde las que visualizar una lista de comentarios y, dependiendo de la página accedida, se necesita una información u otra. Si se accede a la lista de comentarios desde el usuario administrador es necesario disponer de toda la información sobre las valoraciones, pero no estará disponible la opción de modificarlas. En caso de acceder desde el detalle de un producto, no es necesario mostrar en la tabla el título del producto, y si se accede desde el perfil del usuario, no es necesario mostrar en la tabla el nombre del usuario que ha realizado cada comentario. Como cada una de estas tres páginas tienen rutas distintas, es necesario crear métodos distintos en el controlador, aunque las operaciones que se realicen en dichos métodos sean las mismas.

A continuación, se explican brevemente los métodos creados en el controlador. Si el método realiza una operación de búsqueda de comentarios, se anota con *@GetMapping*, en cambio, si la operación es de actualización, inserción o eliminación, se anota con *@PostMapping*. Los parámetros más frecuentes que se introducen en los métodos son los siguientes: un objeto *ModelAndView*, con el que se añadirán objetos a la vista correspondiente; un objeto *HttpServletRequest*, para manejar la información relacionada con la solicitud HTTP recibida en el controlador; un objeto *Authentication*, que contiene la información sobre el usuario autenticado y variables (utilizando *@PathVariable*) o parámetros (utilizando *@RequestParam*) que se obtengan de la solicitud, como pueden ser los identificadores de producto y usuario.

Para mostrar una lista de comentarios, lo primero de todo se llama a un método creado en el servicio para añadir al modelo el valor de las propiedades que permiten la inclusión de los comentarios. En la vista, si estas propiedades tienen "false" como valor, no se pintará nada relacionado con los comentarios. Después, se comprueba con un servicio que ya existía en la aplicación si en la sesión hay un usuario autenticado. En ese caso, se añade al modelo el identificador de ese usuario. Antes de recibir todos los comentarios, se crea un objeto de tipo *Pageable* con el tamaño que se indique en la propiedad, para crear páginas con la longitud que el administrador haya elegido en las propiedades. Si se trata de los comentarios de un producto en concreto, se llama al método que encuentra todos los comentarios validados de ese producto, recibiendo su identificador. En caso de que se acceda a los comentarios de un usuario, se llama al método que encuentra todos los comentarios validados de un lector, recibiendo su identificador. Por último, en caso de acceder desde el perfil del administrador, se buscan todos los comentarios de la biblioteca, estén o no validados.

Para crear o modificar un comentario, los métodos correspondientes del controlador son accedidos tras enviar el formulario para añadir o modificar una reseña. Cuando se envía el formulario, se comprueban dos supuestos utilizando métodos auxiliares creados en el servicio. Primero, se verifica que no pueda suceder que, a la vez, la puntuación sea 0 y tanto el título como el desarrollo del comentario estén vacíos, ya que esto quiere decir que el usuario ha enviado un formulario vacío. En ese caso se añade un mensaje de error al modelo y no se inserta el comentario en la base de datos. Si la puntuación es 0 pero alguna de las otras entradas contiene texto, se considera que la puntuación del comentario es 0. Después, se comprueba si el usuario autenticado ya tiene un comentario en ese producto, y en ese caso también se añade un mensaje de error al modelo y no se inserta nada. Una vez realizadas estas comprobaciones, el método puede tomar dos caminos distintos. Si ya existe un comentario con ese identificador, se llama al método para modificarlo, que lo actualiza con los datos recibidos en el formulario. Si no, se añade un objeto de comentario nuevo al modelo con los valores correspondientes, es decir, el identificador del producto que se valora y el identificador del usuario que realiza la operación, en caso de no ser anónimo.

Finalmente, los métodos que se encargan de eliminar un comentario son accedidos tras pulsar el botón de eliminar que tiene un comentario. En ellos, se elimina el comentario correspondiente de la base de datos, llamando al método que elimina un comentario al proporcionar su identificador. El modelo que devuelven estos métodos es el mismo que reciben, pero ya no cuentan con la información del comentario eliminado.

4.7 Vistas

Las vistas son el último elemento del patrón de diseño Spring MVC. Se encargan de generar la respuesta final que se envía al cliente tras haber procesado su solicitud. En ellas, se muestran los datos del modelo que les llega del controlador. El desarrollo de estas vistas es responsabilidad del equipo de Front-end de la empresa, pero la obtención de los datos del modelo es un trabajo conjunto, ya que para la elaboración de las vistas se utiliza Thymeleaf, que incluye código Java.

Para este módulo, modifican ligeramente algunas vistas que ya estaban creadas en la aplicación y se añaden otras nuevas, que son insertadas como fragmentos en las vistas pertinentes. Las ligeras modificaciones incluyen el añadido de la media de las valoraciones y el número de comentarios en cada producto al obtener resultados de una búsqueda y las opciones de ventanas nuevas tanto en el perfil de usuario, para visualizar los comentarios realizados, como en el perfil de administrador, para validar los nuevos comentarios.

En total, las nuevas vistas creadas son cuatro. Tres de ellas recogen los datos sobre los comentarios y muestran una tabla con ellos. Estas tres vistas son prácticamente iguales, pero tienen alguna diferencia, ya que, como se comenta anteriormente, la lista de comentarios que se muestra varía un poco dependiendo de si es accedida desde el detalle de un producto, desde el perfil de un usuario o desde el perfil de administrador. Para recoger los datos sobre los comentarios que están guardados en el modelo, se utilizan las siguientes expresiones de Thymeleaf:

- Para recorrer la página de comentarios guardada en el modelo, se utiliza *th:each=”comentario : ${paginaComentarios.content}”*. Esta expresión equivale al bucle *for* utilizado en tantos lenguajes de programación.

- Una vez se tiene localizado un comentario, se obtienen sus atributos con la expresión *th:text=”${comentario.atributo}”*.

- En caso de necesitar comprobar una condición, como cuando se requiere comprobar el valor de una propiedad o se muestra un texto u otro dependiendo de si un producto tiene comentarios registrados o no, se utiliza la expresión *th:if=*, seguida de la condición correspondiente.

Por último, la vista restante se corresponde con el formulario de creación o modificación de un comentario que, dependiendo del método desde el que sea accedida, muestra el formulario vacío para crear un nuevo comentario o lo muestra relleno con los datos existentes del comentario que se quiere modificar.

5. GESTIÓN DEL PROYECTO

5.1 Metodología utilizada

Para el desarrollo de este proyecto se sigue una metodología ágil. Esta metodología se caracteriza por la división del proyecto en fases, la colaboración entre los miembros que participan en él y las continuas mejoras.

Para seguir esta metodología existen distintos marcos ágiles de gestión de proyectos, como son scrum, kanban o XP (Programación Extrema) [34]. En este caso, se utiliza Scrum. Lo que distingue a scrum de sus similares es el ánimo de aprender a base de experiencia, la autoorganización para abordar un problema y la reflexión posterior para garantizar una mejora continua. Para facilitar la gestión y organización de todas las tareas, se ha utilizado la herramienta Redmine.

A continuación, se describen los protocolos de scrum realizados a lo largo del desarrollo del proyecto:

- Backlog del producto: junto al jefe del producto de la empresa, se identifican las necesidades del usuario relacionadas con la valoración de productos bibliotecarios y se diseña una lista con las tareas a realizar. Posteriormente, se escriben las historias de usuario para capturar y describir los requisitos de la aplicación.

- Planificación de sprint: se explican las historias de usuario diseñadas y se planifica el trabajo que se va a realizar en el sprint. Después, se añaden las historias creadas desde el backlog al sprint.

- Sprint: el sprint es el periodo efectivo en el que se trabaja para conseguir los objetivos iniciales. En Redmine, se crea un nuevo sprint en el que los desarrolladores pueden añadir tareas, subtareas y errores incluyendo su estado (en progreso, terminadas o resuelto en caso de ser un error), horas estimadas y dedicadas y el desarrollador o desarrolladores a los que se les asignan.

- Reunión diaria y seguimiento: todos los días a primera hora de la mañana se realiza una reunión en la que participa el equipo de desarrollo. La duración de esta reunión no suele superar los 15 minutos, ya que en ella los miembros del equipo suelen compartir las tareas realizadas el día de antes, las que se van a realizar en el día y, si procede, alguna duda con respecto a alguna tarea del sprint. Además, cada dos semanas, se realiza una reunión de seguimiento en la que también participa el jefe de producto para estar informado del estado del desarrollo y de los problemas que han surgido en él.

- Colaboración y comunicación: la comunicación entre el equipo de desarrollo se realiza a través del chat de Google Gmail. Esta herramienta además permite crear videollamadas con Google Meet en caso de necesitar una reunión rápida para clarificar cualquier aspecto del desarrollo.

- Gestión de cambios: la gestión de cambios en el desarrollo del proyecto se ha llevado a cabo con Subversion. Esta herramienta permite la creación de distintas ramas del desarrollo para permitir a los desarrolladores trabajar sobre la misma versión del proyecto sin modificar la rama principal hasta que el desarrollo sea definitivo y trabajar a la vez sin interrumpirse en el desarrollo. Las operaciones que proporciona Subversion permiten a los desarrolladores juntar dos versiones de una misma rama para unir su trabajo, bajarse los cambios nuevos de la rama principal, añadir nuevos cambios a esta o revisar el historial de versiones del proyecto en caso de necesitar volver atrás, entre otras. Además, Subversion se integra a la perfección con proyectos Spring, lo que facilita la gestión de cambios de este proyecto.

- Revisión de sprint: al final del sprint, se produce una reunión en la que también participa el jefe de producto para analizar y comprobar los resultados del desarrollo. Una vez acabada esta reunión, se decide el lanzamiento del producto.

- Retrospectiva: en esta última parte del proyecto, el equipo se reúne para analizar y compartir las experiencias positivas y negativas que cada miembro obtiene tras la realización del trabajo, además de compartir aspectos a modificar en futuros desarrollos. Además, se documenta el trabajo realizado para que cualquier miembro de la empresa entienda cómo funciona este módulo de valoraciones sin necesitar la explicación de ningún trabajador. Esta documentación se realiza utilizando la Wiki de Redmine, una herramienta colaborativa que permite a los usuarios crear, editar y organizar páginas web con contenido textual, imágenes, enlaces y otros elementos.

5.2 Evaluación

La evaluación de este módulo se divide en dos partes: la realización de tests del código utilizando las herramientas JaCoCo y Mockito para garantizar la cobertura y el correcto funcionamiento del código y la realización de pruebas del sistema para comprobar que la aplicación funciona como debería.

5.2.1 Tests y cobertura del código

Para todos los métodos nuevos creados en este proyecto, se realizan pruebas unitarias que prueban el correcto funcionamiento y los distintos caminos que pueden tomar los métodos diseñados. Todos los archivos que contienen estas pruebas se guardan en la carpeta *src/test/java* del proyecto de Spring correspondiente. Estos tests prueban el funcionamiento de los métodos creados en el repositorio, en el servicio y en el controlador. En relación con la cobertura de código, la empresa exige superar el 80% para asegurar que lo desarrollado se ha probado adecuadamente.

Para indicar que una clase de Java se utiliza para realizar estas pruebas, se utiliza la anotación *@SpringBootTest* de Spring. Con ella se carga el contexto completo de la aplicación para utilizar y acceder a todas las clases para poder probarlas. En la mayoría de las ocasiones, se acompaña a esta anotación con el nombre de las clases necesarias en el test correspondiente, para no cargar clases que no se utilizan en ese archivo. Es necesario también anotar cada método de esta clase con *@Test*. Además, se crea un objeto de comentario antes de ejecutar cada test utilizando la anotación *@BeforeEach*, como se observa en la Figura 5.1. Con ella, se prepara un comentario con atributos específicos para luego comprobar que los métodos trabajan correctamente y ejecutan las acciones indicadas con este comentario.

```
@BeforeEach
public void init() {
    coment = new Comentario();
    coment.setId(99L);
    coment.setIdCat("FDAGHAYRAGACVX");
    coment.setPuntuacion(3);
    LocalDateTime localDateTime= LocalDateTime.now();
    coment.setFecha(localDateTime);
    coment.setLector(123456789);

    String opinion="Teodoro, un joven ingeniero español";
    coment.setTitulo(opinion);
    coment.setDescripcion(opinion);
}
```

Fig. 5.1. Inicializar un comentario para realizar los tests

En los tests para los métodos del repositorio, se crean distintos casos para comprobar que los métodos que proporciona el repositorio de Jpa y los que se crean con consultas personalizadas producen el resultado esperado. Para la comprobación de este resultado, se utilizan las *assertions* de Java, que incluyen métodos para comprobar si un resultado existe o no, o para comparar dos resultados, entre otros. A continuación, se describen algunos tests creados:

- Para el método que elimina un comentario, se utiliza el identificador elegido en el método *init()* para encontrar el comentario que se quiere eliminar. Después, se llama al método del repositorio que lo elimina y se comprueba que ya no existe el comentario al volver a buscarlo.

- De igual forma, se encuentra el comentario creado al principio para probar el método que lo modifica. Después, se cambian algunos atributos del comentario, como la puntuación o el texto de opinión y se llama al método *save()* del repositorio, que modifica el comentario porque ya existe en la base de datos. Para verificar que la operación se realiza correctamente, se comprueba que los atributos actuales del comentario se corresponden con los nuevos atributos que se le han proporcionado.

- Para el método que encuentra los comentarios de un producto, se crea un test que utiliza el identificador del producto creado en el método *init()* y se comprueba que su resultado es una página de comentarios de longitud 1, ya que solo se ha creado un comentario. Además, se crea otro test que comprueba que el método devuelve una página vacía si se introduce como parámetro al método un identificador inexistente.

- Para la consulta personalizada que calcula la media de valoraciones que tiene un producto se crea otro test. En él, además de contar con el comentario creado al principio, se crean otros dos nuevos, pertenecientes al mismo producto. El del principio tiene una puntuación de 3 estrellas, y los nuevos se crean con puntuaciones 1 y 5. Tras llamar al método que utiliza la consulta creada para calcular la puntuación media, se comprueba que el resultado es 3.

En los tests del servicio, se comprueba el funcionamiento tanto de los métodos que utilizan el repositorio como de los métodos auxiliares creados, estos últimos son muy similares a los tests del repositorio. Como el correcto funcionamiento de los métodos del repositorio ya está probado con los tests del repositorio, en los test del servicio se utilizan objetos simulados (los objetos *Mock* de Mockito) para simular el comportamiento de los métodos del servicio. Simplemente, en estos tests se utiliza la función *Mockito.verify()*, acompañada del método que se quiere probar, para verificar que los métodos de los objetos simulados se llaman con los argumentos esperados. Estos tests pueden parecer un poco tontos, pero son de gran utilidad, porque en cuanto se utilice un argumento que no es exactamente el indicado en un método, el test falla.

Utilizando el informe que crea JaCoCo, observamos que con los tests realizados se tiene una muy buena cobertura de todos los métodos creados en el servicio, Figura 5.2.

CommentService

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
commentExists(CommentVO, Authentication)		95%		62%	3	5	1	13	0	1
getTitle(String)		100%		62%	3	5	0	10	0	1
getCommentsReaderHome(Authentication, Integer)		100%		100%	0	2	0	5	0	1
addTitlesHome(Page)		100%		100%	0	2	0	4	0	1
commentsFind(String, Pageable)		100%		n/a	0	1	0	3	0	1
commentFindAllReader(long, Pageable)		100%		n/a	0	1	0	3	0	1
commentFindAll(Pageable)		100%		n/a	0	1	0	3	0	1
validateCommentVO(CommentVO)		100%		66%	2	4	0	4	0	1
mejoresValorados()		100%		n/a	0	1	0	3	0	1
mejoresComentados()		100%		n/a	0	1	0	3	0	1
commentBreadcrumb(String, long)		100%		n/a	0	1	0	4	0	1
commentSaveUpdate(CommentVO)		100%		n/a	0	1	0	2	0	1
commentFindId(String)		100%		n/a	0	1	0	2	0	1
commentDelete(String)		100%		n/a	0	1	0	2	0	1
averageRatings(String)		100%		n/a	0	1	0	1	0	1
countComment(String)		100%		n/a	0	1	0	1	0	1
getTitleCount(String)		100%		n/a	0	1	0	1	0	1
getPunctuationAvg(String)		100%		n/a	0	1	0	1	0	1
commentDelete(Long)		100%		n/a	0	1	0	2	0	1
CommentAppService()		100%		n/a	0	1	0	1	0	1
Total	2 of 316	99%	8 of 26	69%	8	33	1	68	0	20

Fig. 5.2. Cobertura del servicio

Por último, los tests para los métodos del controlador. En estos tests se utilizan también objetos simulados y, además, se define el comportamiento que deben tener cuando se llaman a ciertos métodos. Esto se consigue utilizando la función de Mockito *.when().thenReturn()*. Esta función permite centrarse en probar el comportamiento específico de la clase que se está probando y aislarla de sus dependencias externas.

Al haber comprobado ya el funcionamiento de los métodos del repositorio y del servicio, queda por comprobar el comportamiento esencial del controlador, que es dirigir cada método a la vista correcta. Para ello, se utilizan los objetos simulados y también las aserciones de Java para verificar que, al llamar a los métodos del controlador, el modelo que se devuelve existe y no es nulo y que el nombre de la vista que se guarda en ese modelo es igual al nombre de la vista a la que se espera acceder al llamar a ese método del controlador. De igual manera que con los tests del servicio, se comprueba con el informe de JaCoCo que la cobertura de los tests es buena, en este caso se obtiene un 93%.

5.2.2 Pruebas del sistema

En las pruebas del sistema, se comprueba que la aplicación responde a las peticiones del usuario como se espera, comparando su funcionamiento con los requisitos de sistema diseñados antes de comenzar el desarrollo. Para detallar las pruebas realizadas, se utiliza un modelo de tabla como el siguiente:

TABLA 5.1. MODELO DEFINICIÓN DE PRUEBAS DEL SISTEMA

ID	**TC-XX**
Nombre	-
Descripción	-
Resultado esperado	-
Resultado obtenido	Satisfactorio / No satisfactorio

Cada prueba realizada tiene un identificador y un nombre. Además, se describe la funcionalidad que se quiere comprobar, el resultado que se espera obtener y el resultado obtenido, siendo satisfactorio si la prueba devuelve el resultado esperado y no satisfactorio en caso contrario.

TABLA 5.2. TC-01 - VISUALIZACIÓN DESDE BÚSQUEDA

ID	**TC-01**
Nombre	Visualización desde búsqueda
Descripción	Tras realizar una búsqueda, aparece una lista de resultados. En cada uno de los productos se especifican algunos datos, entre los que debe estar la información básica sobre sus valoraciones.
Resultado esperado	En cada producto que aparece, se muestra el número de comentarios que tiene registrados junto al texto “Ver todas las opiniones” y cinco estrellas que se rellenan en función de la media de sus valoraciones, estando completa si es un 5 y vacía si es un 0. Al pinchar en el texto, se abre el formulario de creación de un comentario.
Resultado obtenido	Satisfactorio

TABLA 5.3. TC-02 - VISUALIZACIÓN DESDE BÚSQUEDA SIN VALORACIONES

ID	**TC-02**
Nombre	Visualización desde búsqueda sin valoraciones
Descripción	Tras realizar una búsqueda, aparece una lista de resultados. En los productos que aparezcan como resultado que no tengan valoraciones registradas, se debe informar al usuario de ello, dándole la opción de añadir el primer comentario.
Resultado esperado	En los productos que no tengan comentarios, se muestra el texto “Sé el primero en opinar”, que abre el formulario de creación de un comentario en caso de pinchar en él.
Resultado obtenido	Satisfactorio

TABLA 5.4. TC-03 - VISUALIZACIÓN DESDE DETALLE

ID	TC-03
Nombre	Visualización desde detalle
Descripción	Al acceder al detalle de un producto, se deben mostrar al usuario todas las valoraciones que tenga ese producto, permitiéndole navegar entre las páginas anterior y siguiente.
Resultado esperado	En la tabla de valoraciones de ese producto, aparecen los siguientes datos para cada una: autor, fecha, título y descripción del comentario y las estrellas correspondientes a la puntuación, sin estrellas en caso de que la puntuación sea 0. Se muestra al usuario la posibilidad de navegar entre las páginas de valoraciones con los botones “Anterior” y “Siguiente”, ubicados debajo de la tabla.
Resultado obtenido	Satisfactorio

TABLA 5.5. TC-04 - VISUALIZACIÓN DESDE DETALLE SIN COMENTARIOS

ID	TC-04
Nombre	Visualización desde detalle sin comentarios
Descripción	Al acceder al detalle de un producto que no tenga valoraciones registradas, no se debe mostrar nada relacionado con ellas, excepto la posibilidad de añadir un comentario nuevo.
Resultado esperado	En el detalle de un producto sin valoraciones, solamente aparece la opción de añadir el primer comentario, con el mismo texto que en la búsqueda: “Sé el primero en opinar”, que abre el formulario de creación de un comentario.
Resultado obtenido	Satisfactorio

TABLA 5.6. TC-05 - OPCIÓN DE AÑADIR UNA VALORACIÓN MÁS A UN PRODUCTO

ID	**TC-05**
Nombre	Opción de añadir una valoración más a un producto
Descripción	Al acceder al detalle de un producto que tenga valoraciones, se debe dar al usuario la opción de añadir una opinión más al producto.
Resultado esperado	En el detalle de un producto con valoraciones registradas, se muestra encima de la tabla de estas el texto “Añadir tu opinión”. Al pinchar en el texto, se abre el formulario de creación de una valoración.
Resultado obtenido	Satisfactorio

TABLA 5.7. TC-06 - OPCIÓN DE MODIFICAR O ELIMINAR UNA VALORACIÓN DESDE EL DETALLE

ID	**TC-06**
Nombre	Opción de modificar o eliminar una valoración desde el detalle
Descripción	Al acceder al detalle de un producto que tenga valoraciones como usuario autenticado, se deben mostrar al usuario las opciones de modificar y eliminar una valoración de la tabla en caso de ser el autor.
Resultado esperado	Si en la tabla de valoraciones que un usuario autenticado visualiza en el detalle de un producto aparece un comentario realizado por él mismo, aparecen dos iconos a la derecha del todo de la tabla: un lápiz, para modificarlo, y una basura, para eliminarlo.
Resultado obtenido	Satisfactorio

TABLA 5.8. TC-07 - VISUALIZACIÓN DESDE EL PERFIL DE USUARIO

ID	TC-07
Nombre	Visualización desde el perfil de usuario
Descripción	Tras autenticarse, en la pantalla del perfil del usuario se debe mostrar la opción de visualizar todas sus valoraciones realizadas, permitiendo al usuario navegar entre las páginas anterior y siguiente de la tabla de valoraciones. Además, se le debe ofrecer la posibilidad de eliminar o modificar cualquiera de ellas.
Resultado esperado	Al acceder a las valoraciones de un usuario en su perfil se muestra una tabla con todas ellas. Para cada una, se muestra un enlace al producto correspondiente, la fecha en la que se registró, título y desarrollo del comentario y las estrellas correspondientes a la puntuación, sin estrellas en caso de que la puntuación sea 0. Se muestra al usuario la posibilidad de navegar entre las páginas de comentarios con los botones “Anterior” y “Siguiente”, ubicados debajo de la tabla. Además, para cada valoración aparecen dos iconos a la derecha del todo de la tabla: un lápiz, para modificarla, y una basura, para eliminarla.
Resultado obtenido	Satisfactorio

TABLA 5.9. TC-08 - FORMULARIO PARA AÑADIR UNA VALORACIÓN

ID	TC-08
Nombre	Formulario para añadir una valoración
Descripción	Al acceder al formulario para añadir una valoración, ya sea desde un producto con comentarios o sin ellos, se deberá mostrar al usuario la opción de escribir una reseña nueva y valorar un producto. Se debe permitir añadir una valoración solo con puntuación, o rellenando título o descripción, debiendo guardar una puntuación de 0 estrellas.
Resultado esperado	Si el usuario accede al formulario desde las opciones “Añadir tu opinión” o “Sé el primero en comentar”, este se muestra vacío, y ofrece al usuario la posibilidad de añadir un texto que resuma su opinión y otro en el que la desarrolle, además de cinco estrellas sin colorear para que el usuario puntúe el producto. Al final del formulario, se muestra el botón “Guardar” para registrarlo y el botón “Cancelar”, que devuelve al usuario al detalle del producto. Si el formulario se envía vacío, la valoración no se registra y se informa al usuario. En caso de rellenar uno de los dos cuadros de texto y no añadir puntuación, se guarda con una puntuación de 0 estrellas.
Resultado obtenido	Satisfactorio

TABLA 5.10. TC-09 - FORMULARIO PARA MODIFICAR UNA VALORACIÓN

ID	TC-09
Nombre	Formulario para modificar una valoración
Descripción	Cuando un usuario autenticado accede al formulario para modificar una valoración, desde el detalle de un producto o desde el perfil del usuario, este debe aparecer relleno con el título, el desarrollo y las estrellas del comentario, en caso de que el usuario quiera modificar ligeramente su contenido.
Resultado esperado	Tras acceder al formulario de modificación de una valoración, después de pinchar en el icono del lápiz, este se muestra relleno con los datos que ya tenía la valoración. Se permite al usuario la modificación del título, el desarrollo y la puntuación. Al final del formulario, se muestra el botón "Guardar" para confirmar la modificación y el botón "Cancelar", que devuelve al usuario a la página anterior sin guardar ningún cambio. Si se modifica, la fecha de realización de la valoración se actualiza.
Resultado obtenido	Satisfactorio

TABLA 5.11. TC-10 - VISUALIZACIÓN DESDE EL PERFIL DE ADMINISTRADOR

ID	TC-10
Nombre	Visualización desde el perfil de administrador
Descripción	Tras acceder al perfil de administrador, se debe mostrar una opción para acceder a las valoraciones realizadas. En esa sección, debe aparecer una tabla con todas ellas, ofreciendo la posibilidad de validar u ocultar y eliminar cualquiera de ellas. El administrador debe poder también navegar entre las páginas anterior y siguiente de la tabla de valoraciones.
Resultado esperado	Al acceder a las nuevas valoraciones de una biblioteca en el perfil de administrador, se muestra una tabla con todas ellas. Para cada una, se muestra un enlace al producto correspondiente, la fecha en la que se registró, su autor, título y desarrollo del comentario y las estrellas correspondientes a la puntuación, sin estrellas en caso de que la puntuación sea 0. Se muestra al administrador la posibilidad de navegar entre las páginas de valoraciones con los botones "Anterior" y "Siguiente", ubicados debajo de la tabla. Además, para cada valoración aparecen dos iconos a la derecha del todo de la tabla: un tick o una cruz, para mostrar que está validad u oculta, y una basura, para eliminarla.
Resultado obtenido	Satisfactorio

TABLA 5.12. TC-11 - ELIMINAR UNA VALORACIÓN

ID	TC-11
Nombre	Eliminar una valoración
Descripción	Los usuarios autenticados deben poder eliminar cualquiera de sus valoraciones realizadas, desde su perfil o desde el detalle de un producto. El usuario administrador debe poder eliminar cualquier valoración y el usuario anónimo no debe tener la posibilidad de eliminar ninguna valoración.
Resultado esperado	Si se accede a la tabla de valoraciones desde el perfil del administrador, se da la posibilidad de eliminar cualquiera de ellas. En caso de autenticarse, el usuario puede eliminar cualquiera de sus valoraciones desde su perfil o desde el detalle de un producto, en caso de tener registrada alguna en ese producto. Para eliminar una valoración, se pulsa el icono de una basura a la derecha del todo de la tabla. En ningún caso se ofrece la posibilidad de eliminar una valoración a un usuario anónimo.
Resultado obtenido	Satisfactorio

TABLA 5.13. TC-12 - VALIDAR UNA VALORACIÓN

ID	TC-12
Nombre	Validar una valoración
Descripción	Cuando se visualizan las valoraciones realizadas en el perfil del administrador, se debe ofrecer la posibilidad de validar u ocultar cualquiera de ellas para así mostrarlas a todos los usuarios.
Resultado esperado	Para cada valoración nueva que se muestra en el perfil de administrador, aparece un botón a la derecha del todo de la tabla con un icono de un tick o el de una cruz. Si aparece la cruz, la valoración no está validada. Si se pulsa el botón, el icono cambia a un tick y la valoración aparece en la aplicación. Si aparece el tick, la valoración está validada. Pulsando el botón, el icono cambia a la cruz y la valoración no se muestra en la aplicación.
Resultado obtenido	Satisfactorio

Matriz de trazabilidad entre los requisitos de sistema y las pruebas realizadas:

REQUISITOS DE SISTEMA Y PRUEBAS DE SISTEMA		REQUISITOS DE SISTEMA									
		SR-01	SR-02	SR-03	SR-04	SR-05	SR-06	SR-07	SR-08	SR-09	SR-10
PRUEBAS DE SISTEMA	TC-01	X									
	TC-02	X									
	TC-03		X		X						
	TC-04		X								
	TC-05					X					
	TC-06						X	X			
	TC-07			X	X						
	TC-08					X					
	TC-09							X			
	TC-10				X				X		
	TC-11						X			X	
	TC-12										X

Fig. 5.3. Matriz de trazabilidad entre requisitos de sistema y pruebas de sistema

5.3 Planificación y presupuesto

5.3.1 Tareas

La duración total del proyecto ha sido de 4 meses. En ellos, se identifican tres tareas principales: planificación, desarrollo y despliegue. A continuación, se detalla el trabajo realizado en cada una de las tareas. Además, en la siguiente sección se muestra gráficamente el desarrollo del proyecto con un diagrama de Gantt, Figura 5.4.

- Planificación:

    - Análisis de mercado: esta fase consiste en el análisis de módulos semejantes existentes en el mercado para obtener ideas para el proyecto. El mejor ejemplo de un módulo parecido es el que tiene Amazon. Esta tarea se desarrolla en 3 días.

    - Requisitos: en estos 23 días se diseñan los requisitos del proyecto, tanto de usuario como de sistema, pensando en las necesidades de los usuarios de las bibliotecas, tanto lectores como administrador.

    - Historias de usuario: el diseño de las historias de usuario es muy similar al de los requisitos, por ello esta tarea tiene una duración de 5 días. La diferencia es que estas historias de usuario se tienen que diseñar en Redmine para añadirlas al sprint, y así poder asignar responsables y tiempo dedicado en la fase de desarrollo.

    - Definición de arquitectura: en este tiempo se define como va a ser la arquitectura de la aplicación. Como el patrón de desarrollo seguido es el mismo que con el resto de los proyectos ya realizados en la empresa, la duración de esta tarea no supera los 3 días.

- Desarrollo:

    - Análisis de historias y requisitos: esta tarea es muy importante para empapar a los desarrolladores de las ideas principales del proyecto y tener una idea clara de su misión, objetivo y funcionamiento, por ello tiene una duración de 5 días.

    - Diseño: en esta fase que dura 10 días, se define la estructura de la tabla en la base de datos, así como una primera estructura de los métodos que se van a necesitar en el desarrollo. Además, se estudian todos los servicios de la aplicación ya creados que se van a necesitar en el desarrollo del proyecto, como el manejo de las sesiones y de los catálogos.

    - Implementación: la duración de esta tarea es, con diferencia, la más extensa. En total son 30 días laborables, más de un mes real de trabajo. En ella, se incluye todo el desarrollo del diseño principal, además de la inclusión de los servicios ya existentes en el proyecto para lograr la funcionalidad exigida en los requisitos.

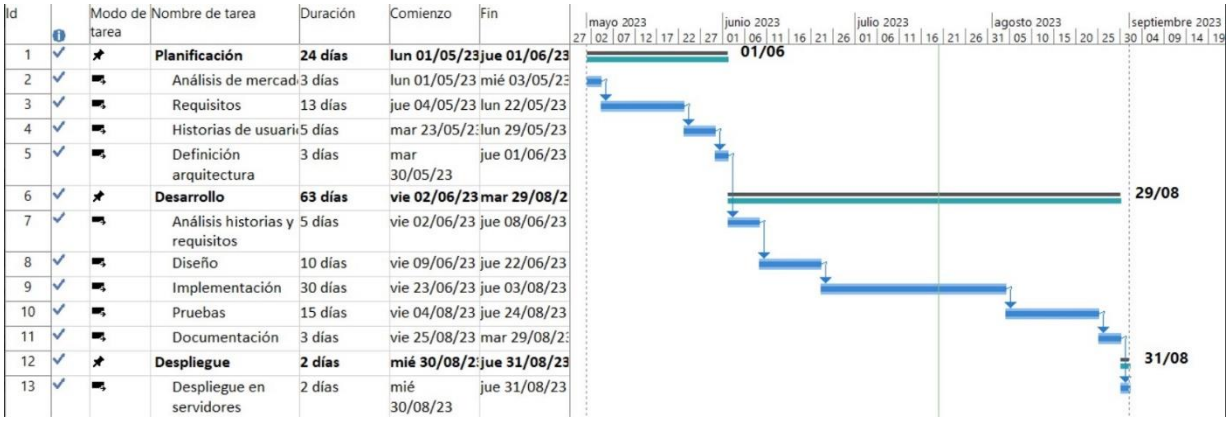
    - Pruebas: en esta fase se llevan a cabo todas las pruebas realizadas, incluyendo las de código y las del sistema. Además, de encontrar algún fallo se debe arreglar, por ello la duración de esta tarea es de 15 días.

    - Documentación: por último, en la fase de desarrollo se incluyen 3 días en los que se documenta todo lo realizado en el proyecto, utilizando la Wiki de Redmine.

- Despliegue: en esta última fase del proyecto, se despliegan los cambios realizados en todos los servidores y entornos de ejecución de la empresa. Tiene una duración de únicamente 2 días.

5.3.2 Diagrama de Gantt

El diseño de este diagrama se ha realizado con la herramienta Microsoft Project. Los días que se incluyen en la columna "Duración" se contabilizan como días laborables, con lo cual la suma de todos los días en los que se ha llevado a cabo el proyecto es de 89, lejos de los 123 días comprendidos entre mayo y agosto, ambos incluidos.



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		**Planificación**	**24 días**	**lun 01/05/23**	**jue 01/06/23**
2		Análisis de mercad	3 días	lun 01/05/23	mié 03/05/23
3		Requisitos	13 días	jue 04/05/23	lun 22/05/23
4		Historias de usuari	5 días	mar 23/05/23	lun 29/05/23
5		Definición arquitectura	3 días	mar 30/05/23	jue 01/06/23
6		**Desarrollo**	**63 días**	**vie 02/06/23**	**mar 29/08/23**
7		Análisis historias y requisitos	5 días	vie 02/06/23	jue 08/06/23
8		Diseño	10 días	vie 09/06/23	jue 22/06/23
9		Implementación	30 días	vie 23/06/23	jue 03/08/23
10		Pruebas	15 días	vie 04/08/23	jue 24/08/23
11		Documentación	3 días	vie 25/08/23	mar 29/08/23
12		**Despliegue**	**2 días**	**mié 30/08/23**	**jue 31/08/23**
13		Despliegue en servidores	2 días	mié 30/08/23	jue 31/08/23

Fig. 5.4. Diagrama de Gantt

5.3.3 Presupuesto

Los costes del proyecto se dividen en tres bloques: recursos humanos, hardware y software. En el bloque de recursos humanos se tienen en cuenta los gastos en salario de los trabajadores que participan en el proyecto, el jefe de producto, el desarrollador jefe y el desarrollador. El jefe de producto participa principalmente en la fase de planificación, además de ayudar en los primeros pasos de la fase de desarrollo. El desarrollador jefe dirige al desarrollador, además de orientarle y ayudarle todo lo posible.

La duración del proyecto es de 534 horas, como se observa en el anterior diagrama de Gantt. Teniendo en cuenta un tiempo de trabajo de 6 horas al día y los salarios de los trabajadores, los gastos en recursos humanos son los siguientes:

TABLA 5.14. COSTES DE RECURSOS HUMANOS

RECURSOS HUMANOS		
Jefe de producto	115 horas * 35 € / hora	4025 €
Desarrollador jefe	110 horas * 20 € / hora	2200 €
Desarrollador	309 horas * 6 € / hora	1854 €
TOTAL	**8079 €**	

En cuanto a los gastos de hardware, se deben tener en cuenta las herramientas físicas utilizadas por los trabajadores a lo largo del proyecto. Entre ellas se encuentran los portátiles y otros servicios utilizados en la oficina, como puede ser el mantenimiento de la instalación de los servidores, o el aire acondicionado. A los portátiles se les resta el porcentaje de amortización, que por defecto es del 33% como nivel máximo anual. Esto ha sido decidido en base a lo establecido por la Agencia Tributaria española para "Sistemas y programas informáticos" [35].

TABLA 5.15. COSTES DE HARDWARE

HARDWARE		
Portátiles	3 * 800€, porcentaje de amortización del 33%	2400 € - 792 € = 1608 €
Otros	Mantenimiento y servicios	500 €
TOTAL	**2108 €**	

Por último, en los gastos de software se contabiliza el dinero invertido en las herramientas de desarrollo utilizadas y en los servidores y seguridad que emplea la empresa.

TABLA 5.16. COSTES DE SOFTWARE

SOFTWARE		
IDE de Spring	Gratuito	0€
Lenguaje de programación	Gratuito	0€
Base de datos	Gratuita	0€
Servidores web y hosting	100 € / mes * 4 meses.	400 €
Seguridad	Gratuita	0€
TOTAL	**400 €**	

En total, el coste del proyecto es de 8079 € + 2108 € + 400 € = 10587 €. A este coste total hay que añadirle el impuesto aplicado, que es el 21% de IVA (Impuesto sobre el Valor Añadido). Por tanto, en total, el coste asciende a 10587 € + 2223.27 € = 12810.27

6. MARCO REGULADOR

6.1 Leyes

Para la realización de este módulo de valoraciones y, en general, para la aplicación de ODA, es imprescindible cumplir la Ley de Protección de Datos para proteger los derechos y privacidad de las personas respecto al tratamiento de sus datos personales. Esta aplicación cumple con la Ley de Protección de Datos y se preocupa por la privacidad y seguridad de los usuarios que la utilizan. Los reglamentos mencionados son los siguientes:

- REGLAMENTO (UE) 2016/679 DEL PARLAMENTO EUROPEO Y DEL CONSEJO de 27 de abril de 2016 (Reglamento general de protección de datos) [36].

- Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales (LOPDGDD) [37].

6.2 Licencias

En este apartado se enumeran las licencias utilizadas para el desarrollo de este proyecto, además de las herramientas que las requieren.

- Office 365: esta licencia la proporciona la Universidad Carlos III de Madrid a todos sus alumnos. Con ella, se utilizan las herramientas Microsoft Word, para la redacción de este documento, y Microsoft Excel, para el diseño de las matrices de trazabilidad.

- GNU (General Public License): esta licencia es de software libre y de código abierto. Garantiza libertades fundamentales a los desarrolladores de software, que incluyen la posibilidad de usar el programa con cualquier propósito, de estudiarlo y modificarlo en base a sus necesidades y de distribuir copias a otros usuarios. Las herramientas que se utilizan bajo esta licencia son Java, como lenguaje de programación de la aplicación; MariaDB, para almacenar la base de datos de la aplicación; DBeaver, para la administración de la base de datos y StarUML, utilizada para el diseño del diagrama de los casos de uso.

- LGPL (GNU Lesser General Public License): esta licencia también es de código abierto y de software libre y garantiza las mismas libertades que GNU. Está diseñada específicamente para las bibliotecas de software y componentes reutilizables, no para aplicaciones completas. La herramienta que se utiliza bajo LGPL es Hibernate, para la comunicación con la base de datos.

- Apache 2.0: esta licencia fue desarrollada por la Apache Software Foundation, y también es de software libre y de código abierto. Las herramientas que se utilizan bajo ella son Spring, para el desarrollo de la aplicación; Maven, para la estructuración de los proyectos de Spring; Thymeleaf, para generar las vistas HTML; Solr, para la búsqueda de catálogos; Subversion, para el control de versiones del proyecto y Draw.io, para la realización de los diagramas de flujo que se muestran en este documento.

- MIT (Massachusetts Institute of Technology License): esta licencia es muy permisiva, también de código abierto y de software libre. Mockito, que se utiliza en el proyecto para la realización de las pruebas de código, se distribuye bajo esta licencia.

- EPL (Eclipse Public License): esta licencia fue desarrollada por la Fundación Eclipse y está diseñada para utilizarla en proyectos de software desarrollados bajo el marco de trabajo Eclipse. Es también de software libre y de código abierto. Tanto JaCoCo, utilizado para observar la cobertura de las pruebas de código, como STS (Spring Tool Suite), utilizado como entorno de desarrollo de la aplicación, se distribuyen bajo la licencia EPL.

7. ENTORNO SOCIOECONÓMICO

La realización de este módulo de valoraciones y su incorporación a ODA, la aplicación web ya existente que tiene la empresa, puede tener diversas repercusiones positivas, tanto en la sociedad como en la economía.

Al añadir una sección de valoraciones a la aplicación se estimula la participación de los usuarios, lo cual lleva al crecimiento de la comunidad de lectores de cada biblioteca, ya que los lectores pueden compartir entre ellos su experiencia y sus opiniones sobre todos los libros o películas que conocen. Este añadido a la aplicación sin duda incentivará a más personas a explorar nuevas obras, ya que, en la actualidad, las recomendaciones personales y las críticas de otros usuarios se tienen muy en cuenta a la hora de adquirir un producto, en este caso un préstamo de un libro o película. Esta interacción entre los usuarios puede enriquecer notablemente su experiencia al usar la aplicación.

Esta mejora de la aplicación permite también a los usuarios ahorrar tiempo, al evitar la necesidad de consultar otras fuentes para obtener opiniones o recomendaciones sobre un producto. De esta manera, podrán acceder a diversas perspectivas y puntos de vista directamente desde la plataforma. De igual manera, esta sección puede tener un impacto significativo en las decisiones de los usuarios al obtener un préstamo de un producto. Por ejemplo, en situaciones en las que un usuario tenía la intención de adquirir un producto porque tenía una referencia muy positiva sobre él, la presencia de comentarios negativos en la aplicación puede hacer que se eche atrás en su decisión, o simplemente le puede hacer ver que el producto no es tan bueno como pensaba, rebajando así sus expectativas.

En cuanto a los beneficios que pueden obtener las bibliotecas, destaca la recopilación de datos. La información y reseñas que añaden los usuarios proporcionan datos muy valiosos a las bibliotecas sobre las preferencias de los usuarios. Todos estos datos pueden ser utilizados para mejorar la oferta de productos bibliotecarios en función de los gustos de sus usuarios. Si estas mejoras son implementadas eventualmente, las bibliotecas podrían obtener importantes beneficios económicos. Además, las bibliotecas también pueden tener en cuenta los comentarios de los usuarios para identificar diferentes áreas de mejora en su servicio y en la aplicación.

Por último, la empresa también puede obtener un gran beneficio tras la inclusión de este módulo en la aplicación. En primer lugar, las bibliotecas mostrarán mayor satisfacción y contento con su contratación, ya que la plataforma mejora la interacción con los usuarios y fomenta una comunidad de usuarios más activa. Esto podría contribuir a la fidelización de las bibliotecas existentes, construyendo relaciones más sólidas con ellas.

Además, el aumento en la satisfacción de las bibliotecas puede generar un efecto positivo en la adquisición de nuevos clientes. Esto puede tener un impacto directo en el crecimiento de los ingresos de la empresa al aumentar las suscripciones de la plataforma, consolidando su posición en el mercado.

8. CONCLUSIONES

En este último apartado, quiero destacar los objetivos que se han cumplido, analizando brevemente el proceso y haciendo retrospectiva para reflexionar sobre el proyecto. Además, se añaden posibles mejoras para que el módulo de valoraciones crezca en el futuro.

Por experiencia creo que, en el momento de adquirir un producto, sea un libro, una película o un juego, lo primero en lo que nos fijamos actualmente es en la opinión que tiene de él la gente que ya lo ha adquirido. Por ello, creo que este proyecto ha sido muy importante para que ODA evolucione como producto y la empresa crezca más todavía. El módulo desarrollado permite a los usuarios de las bibliotecas la creación de valoraciones, con el principal objetivo de disponer de opiniones de usuarios con conocimiento para estar más informado de las características de los productos. Se permiten incluso los comentarios de usuarios anónimos, lo que diferencia a este módulo de las páginas de reseñas habituales en sitios de compra o alquiler de productos. El control de todas las valoraciones se ha logrado con la implementación de las validaciones por parte del usuario administrador de cada biblioteca, que permiten mantener un orden y evitar comentarios que no aportan nada positivo a la biblioteca.

Sin ninguna duda, la parte más complicada del proyecto ha sido la integración del módulo de valoraciones en la aplicación web ya existente. Al final, lo que he aprendido en la universidad me ha servido de gran ayuda cuando tenía que programar. En cambio, me ha llevado mucho tiempo y esfuerzo entender bien cómo funcionaban todos los servicios de la aplicación para poder integrar correctamente este proyecto en ODA. Concretamente, la parte más complicada ha sido para mí la gestión de los catálogos, ya que la empresa utiliza varias herramientas que yo al principio del proyecto no conocía, como por ejemplo Solr. Cuando aparece algo nuevo que tengo que controlar y no sé cómo hacerlo, personalmente he aprendido que, para futuros proyectos, tengo que intentar no sentirme tan agobiado si me encuentro con alguna situación parecida porque, al final, repercute en mi capacidad para gestionar los problemas.

Personalmente, creo que mis conocimientos sobre el desarrollo de aplicaciones web han crecido exponencialmente estos meses durante mi estancia en la empresa. Además de aplicar todo lo aprendido durante el grado, la empresa me brindó la oportunidad de realizar un curso sobre aplicaciones web y APIs REST de más de 40 horas que aumentó notablemente mis conocimientos para poder aplicarlos también en el desarrollo de este proyecto. En cuanto al trabajo en equipo, considero que he ganado mucha experiencia participando en un entorno real de trabajo, aplicando también lo que he aprendido a lo largo del grado sobre gestión de proyectos y, más concretamente, sobre el desarrollo ágil y scrum. Desde mi punto de vista, es distinto trabajar en el desarrollo ágil con los compañeros de clase que hacerlo con los compañeros de trabajo. Aun así, los conocimientos que he aprendido en la universidad me han ayudado mucho, sobre todo al principio del proyecto.

Por último, creo que el módulo de valoraciones creado puede evolucionar en el futuro para mejorar todo lo posible la experiencia de los usuarios de ODA. Algunas de las mejoras que se proponen son las siguientes:

- Ofrecer a los usuarios una serie de productos recomendados que, en función de las valoraciones que ha realizado anteriormente, se cree que pueden ser de su gusto, con el fin de mejorar su experiencia y ahorrarles el tiempo que conlleva buscar un nuevo libro o película que sea de su interés.

- Ofrecer a los usuarios la posibilidad de responder comentarios de otros usuarios en caso de tener cualquier tipo de duda sobre un producto. Estas conversaciones serían publicadas también, de manera que, si más de un usuario se pregunta lo mismo, le bastaría con leer todos los comentarios. Esta mejora fomentaría más aún la participación de los usuarios.

- Una mejora muy similar a la característica que ofrece Amazon, explicada en el punto 2.1.1 de este documento. Cuando un usuario valora un producto que se le ha prestado desde la biblioteca, en la reseña aparece un icono, o un texto que dé a entender al resto de usuarios que esa opinión es fundada ya que ese usuario tiene experiencia con ese producto.

BIBLIOGRAFÍA

[1] "Amazon.es: compra online de electrónica, libros, deporte, hogar, moda y mucho más." https://www.amazon.es/ (accessed Jun. 23, 2023).

[2] "Libros e eBooks, descuentos y envíos gratis | Casa del Libro." https://www.casadellibro.com/ (accessed Jul. 06, 2023).

[3] "Conoce todas las Técnicas de Renderizado de Sitios Web." https://felixicaza.com/blog/conoce-todas-las-tecnicas-de-renderizado-de-sitios-web (accessed Jun. 19, 2023).

[4] "Rendering on the Web." https://web.dev/rendering-on-the-web/ (accessed Aug. 06, 2023).

[5] "Server-side website programming - Learn web development | MDN." https://developer.mozilla.org/en-US/docs/Learn/Server-side (accessed Aug. 06, 2023).

[6] "Migration from Monolithic Application to Microservices — Part 1: Introduction | by Thi Tran | Medium." https://medium.com/@thi.tran.fi/migration-from-monolithic-application-to-microservices-introduction-1c289def193d (accessed Jun. 19, 2023).

[7] "Qué es Java, para qué sirve, características e historia." https://blog.hubspot.es/website/que-es-java (accessed Jun. 09, 2023).

[8] "Java Platform, Enterprise Edition (Java EE) | Oracle Technology Network | Oracle." https://www.oracle.com/java/technologies/java-ee-glance.html (accessed Aug. 03, 2023).

[9] "Welcome to the Apache Struts project." https://struts.apache.org/ (accessed Aug. 03, 2023).

[10] "Qué es Python: Características, evolución y futuro | OpenWebinars." https://openwebinars.net/blog/que-es-python/ (accessed Jun. 16, 2023).

[11] "Ventanas Modales con Django usando Bootstrap y Vistas Basadas en Clases ~ DEBS Consultores." https://debsconsultores.blogspot.com/2019/04/ventanas-modales-con-django-usando.html (accessed Jun. 16, 2023).

[12] "Javascript." https://desarrolloweb.com/home/javascript (accessed Jun. 18, 2023).

[13] "PHP." https://desarrolloweb.com/home/php (accessed Jun. 18, 2023).

[14] "Laravel - The PHP Framework For Web Artisans." https://laravel.com/ (accessed Aug. 03, 2023).

[15] "Bases de datos relacionales y no relacionales – conceptos y diferencias." https://codespaceacademy.com/blog/bases-datos-relacionales-y-no-relacionales/ (accessed Jun. 19, 2023).

[16] “NoSQL vs SQL: diferencias y ventajas de cada sistema.” https://www.unir.net/ingenieria/revista/nosql-vs-sql/ (accessed Aug. 03, 2023).

[17] “Los lenguajes de programación más populares de 2022 [Ranking].” https://www.stackscale.com/es/blog/lenguajes-programacion-mas-populares/ (accessed Jun. 12, 2023).

[18] “Ventajas y desventajas de Java que debes conocer.” https://blog.hubspot.es/website/ventajas-desventajas-java (accessed Jun. 12, 2023).

[19] “Spring Framework.” https://spring.io/projects/spring-framework (accessed Jun. 06, 2023).

[20] “Spring Interview Questions.” http://www.javaiq.net/interview/questions/spring-interview-questions.htm (accessed Jun. 14, 2023).

[21] “¿Qué es Spring Boot? - Arquitectura Java.” https://www.arquitecturajava.com/que-es-spring-boot/ (accessed Jun. 06, 2023).

[22] “¿Qué es Spring MVC? | CNAC IT.” https://www.cnac.es/noticias/que-es-spring-mvc/ (accessed Jun. 06, 2023).

[23] “Spring Boot 3. Aplicaciones web y REST APIs con Spring MVC | Udemy.” https://www.udemy.com/course/spring-framework-desarrollo-web-spring-mvc/ (accessed Jun. 06, 2023).

[24] “Spring Security.” https://spring.io/projects/spring-security (accessed Jun. 09, 2023).

[25] “¿Qué es Maven y para qué se utiliza? - Panama Hitek.” https://panamahitek.com/que-es-maven-y-para-que-se-utiliza/?utm_content=cmp-true (accessed Jun. 12, 2023).

[26] “¿Qué es Spring Data JPA? – Barcelona Geeks.” https://barcelonageeks.com/que-es-spring-data-jpa/ (accessed Jun. 12, 2023).

[27] “Thymeleaf.” https://www.thymeleaf.org/index.html (accessed Jun. 13, 2023).

[28] “Qué es solr. Qué es Solr y para qué sirve.” https://solrtutorial.es/que-es-solr.html (accessed Jun. 13, 2023).

[29] “JaCoCo y la cobertura de pruebas en el código.” https://platzi.com/tutoriales/1503-testing-java/3841-jacoco-y-la-cobertura-de-pruebas-en-el-codigo/ (accessed Jul. 05, 2023).

[30] “Tutorial de Mockito Java: qué es, ejemplos, static...” https://es.ccm.net/ordenadores/programacion/2780-mockito/ (accessed Jul. 05, 2023).

[31] “¿Qué es MariaDB? | KeepCoding Bootcamps.” https://keepcoding.io/blog/que-es-mariadb/ (accessed Jun. 15, 2023).

[32] "| Redmine como herramienta de gestión de proyectos." https://www.ilimit.com/blog/redmine-como-herramienta-de-gestion-de-proyectos/ (accessed Jun. 15, 2023).

[33] "Subversión: La alternativa para el control de versiones | Arsys." https://www.arsys.es/blog/subversion (accessed Jun. 16, 2023).

[34] "¿Qué es ágil? | Atlassian." https://www.atlassian.com/es/agile (accessed Jul. 18, 2023).

[35] "Agencia Tributaria: Amortizaciones." https://sede.agenciatributaria.gob.es/Sede/impuesto-sobre-sociedades/que-base-imponible-se-determina-sociedades/amortizaciones.html?faqId=42c3904421205710VgnVCM100000dc381e0aRCRD (accessed Aug. 03, 2023).

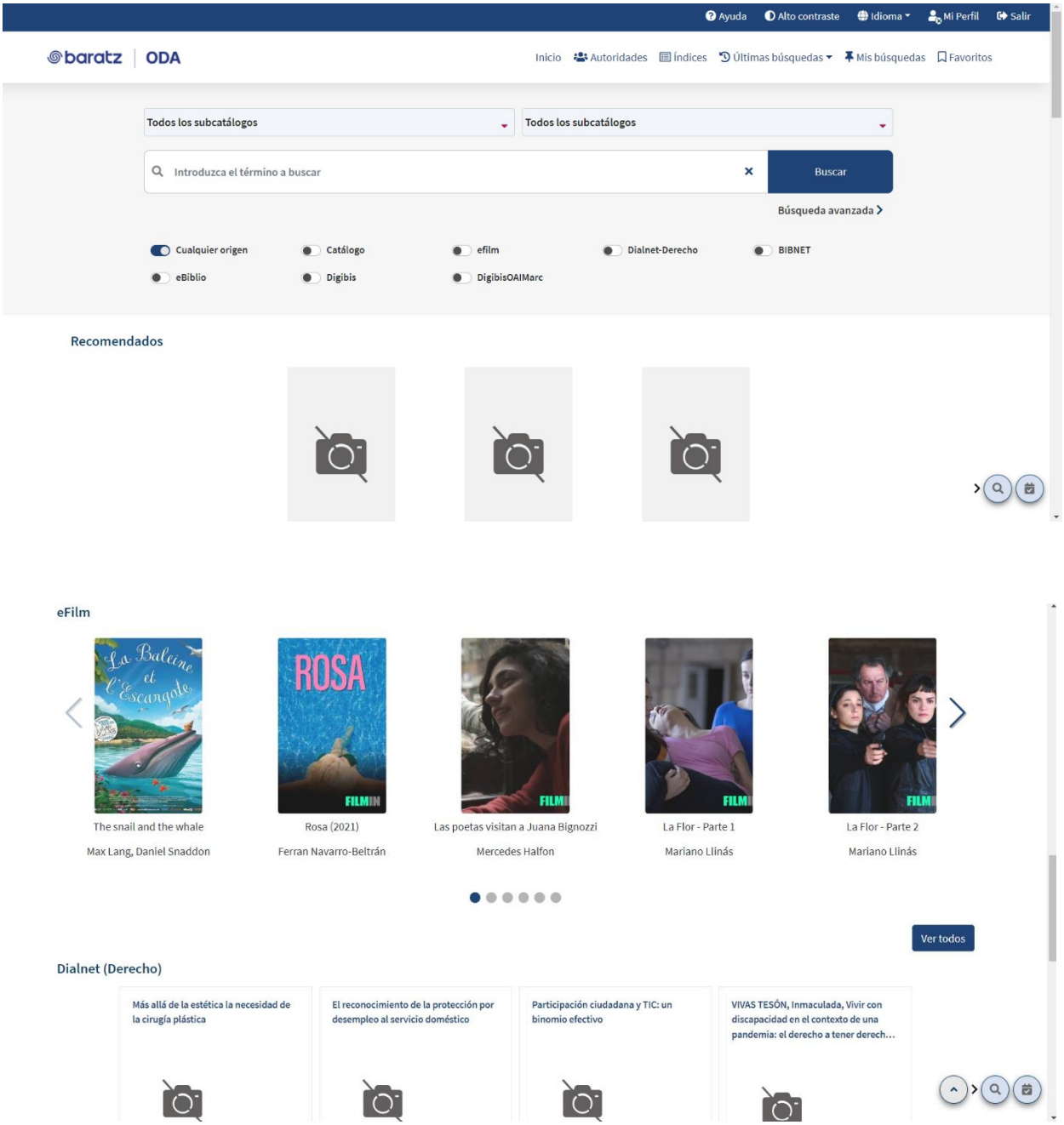
[36] "BOE.es - DOUE-L-2016-80807 Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos)." https://www.boe.es/buscar/doc.php?id=DOUE-L-2016-80807 (accessed Aug. 06, 2023).

[37] "BOE-A-2018-16673 Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales." https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16673 (accessed Aug. 06, 2023).

ANEXO A. MANUAL USUARIO ANÓNIMO

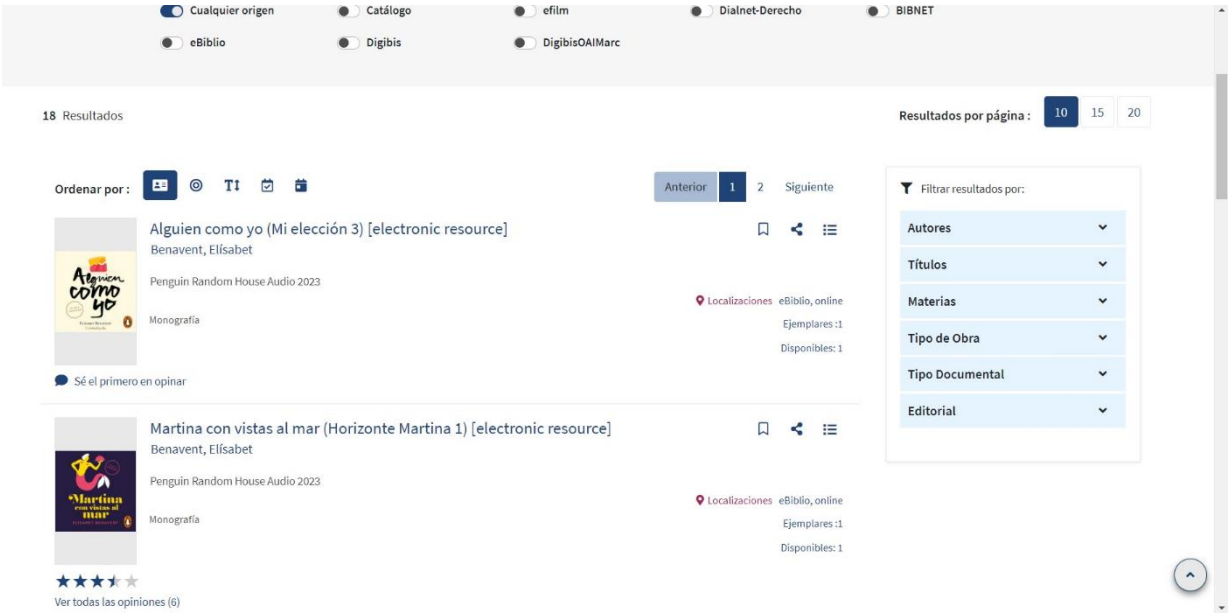
Página de inicio

Al iniciar la aplicación, la página de inicio contiene un buscador amplio con el que se puede buscar por catálogos, subcatálogos y por términos concretos. Más abajo, aparece la sección de “Recomendados” y otras secciones que contienen distintos productos a los que acceder.



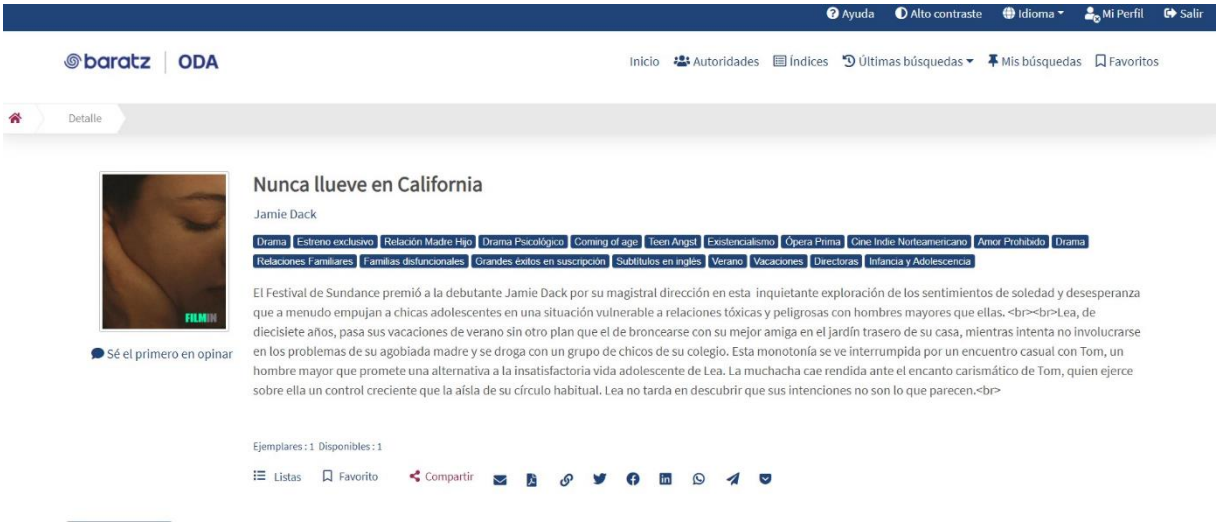
Resultados de una búsqueda

Tras realizar una búsqueda, los resultados aparecen en forma de lista. Si los productos no tienen valoraciones registradas, aparece el texto “Sé el primero en opinar, que da acceso al formulario de creación de una valoración. En cambio, si el producto tiene valoraciones, aparecerá la media de las puntuaciones representada con estrellas coloreadas y el número de valoraciones que tiene, que da acceso al detalle del producto.



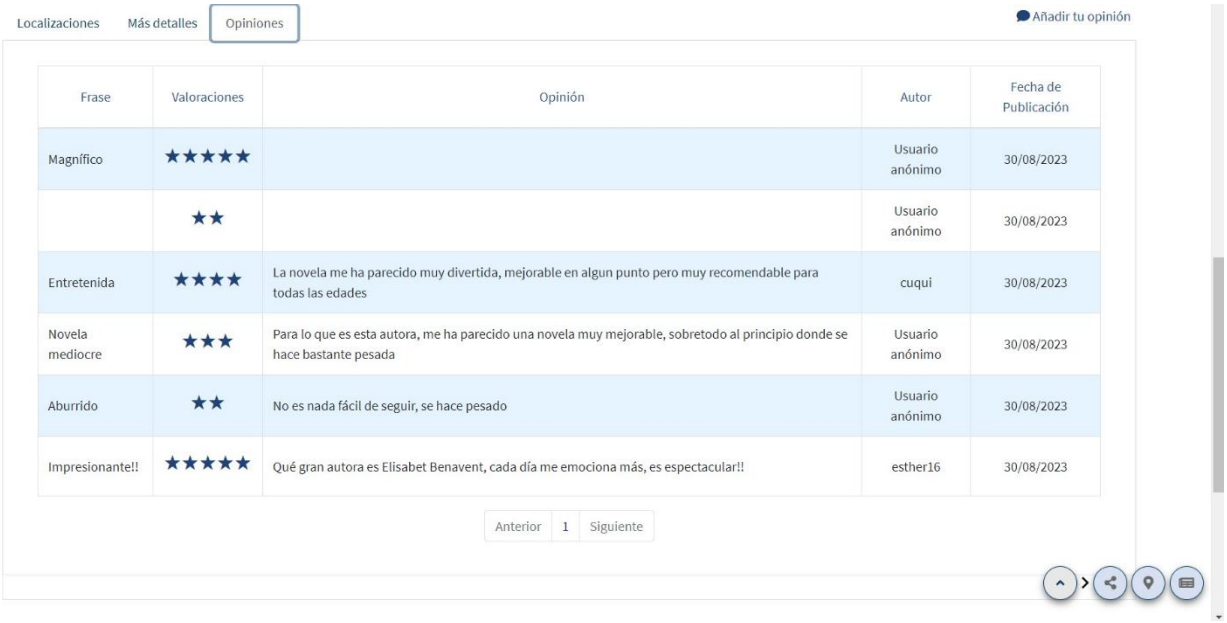
Detalle de un producto

Al acceder al detalle de un producto, aparece más información sobre él. Si no tiene valoraciones, el resultado será el mismo que en la búsqueda, se ofrece al usuario la posibilidad de ser el primero en comentar.



En cambio, si el producto tiene valoraciones registradas, se incluye la misma información que en la búsqueda, además de la pestaña de “Opiniones”, en la que se pueden ver todas las valoraciones que tiene ese producto.

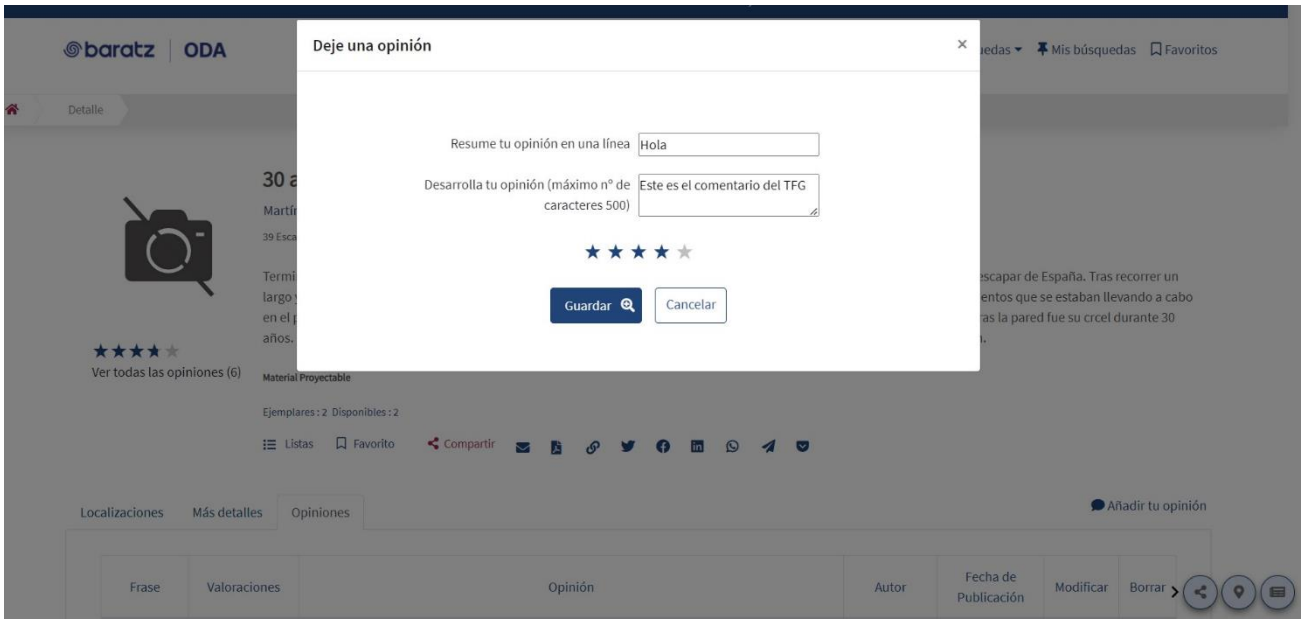
Como se observa en la tabla, si el autor del comentario es anónimo no se detalla información sobre él, en cambio si es un usuario registrado aparece su alias.



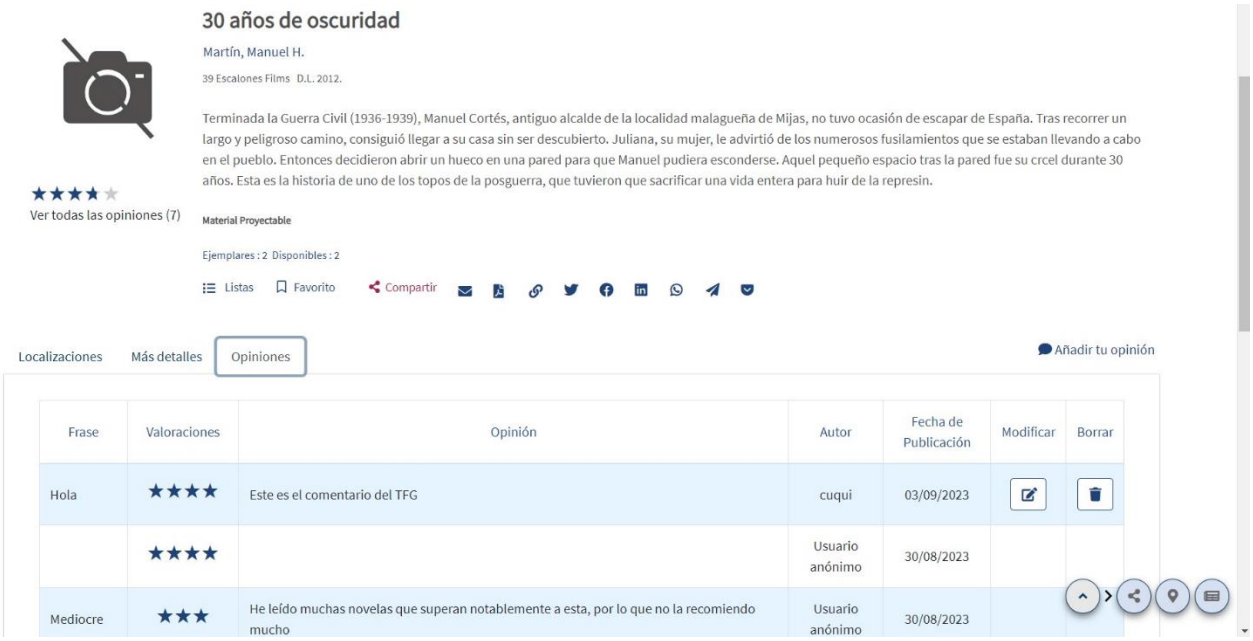
Frase	Valoraciones	Opinión	Autor	Fecha de Publicación
Magnífico	★★★★★		Usuario anónimo	30/08/2023
	★★		Usuario anónimo	30/08/2023
Entretenida	★★★★	La novela me ha parecido muy divertida, mejorable en algun punto pero muy recomendable para todas las edades	cuqui	30/08/2023
Novela mediocre	★★★	Para lo que es esta autora, me ha parecido una novela muy mejorable, sobretodo al principio donde se hace bastante pesada	Usuario anónimo	30/08/2023
Aburrido	★★	No es nada fácil de seguir, se hace pesado	Usuario anónimo	30/08/2023
Impresionante!!	★★★★★	Qué gran autora es Elisabet Benavent, cada día me emociona más, es espectacular!!	esther16	30/08/2023

Añadir una valoración

Tras acceder al formulario para añadir una nueva valoración, este se completa, debiendo elegir obligatoriamente una puntuación. Escribir un título y una descripción del comentario es opcional. Este comentario se crea con un perfil autenticado, por ello en la segunda imagen aparece un alias y la opción de modificar y eliminar, pero los usuarios anónimos siguen exactamente los mismos pasos para crear una valoración.



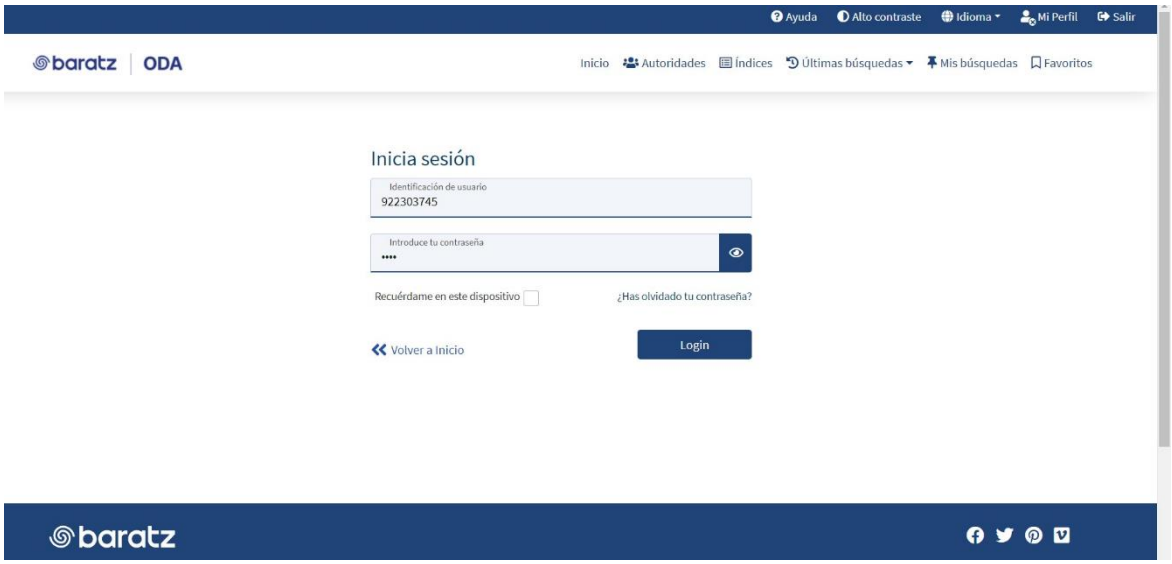
Como se observa en las dos imágenes, los datos añadidos en el formulario se ven reflejados en la nueva valoración que se ha creado.



ANEXO B. MANUAL USUARIO REGISTRADO

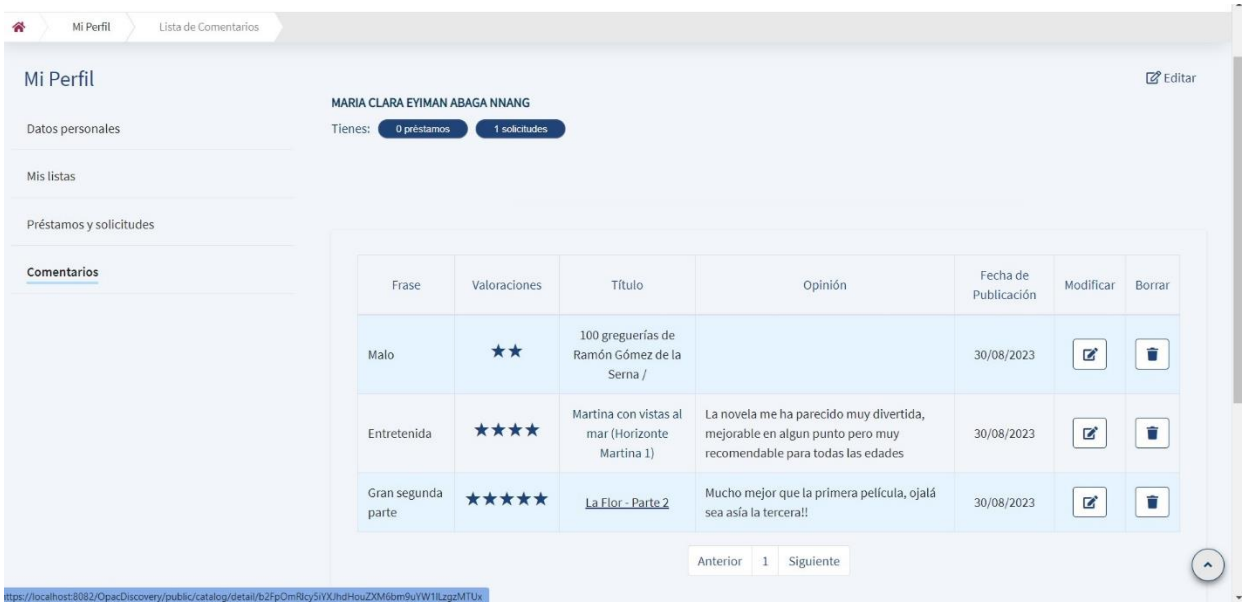
Inicio de sesión

Al pinchar en “Mi Perfil”, se redirige al usuario al formulario de inicio de sesión, en el que necesita su número de lector.



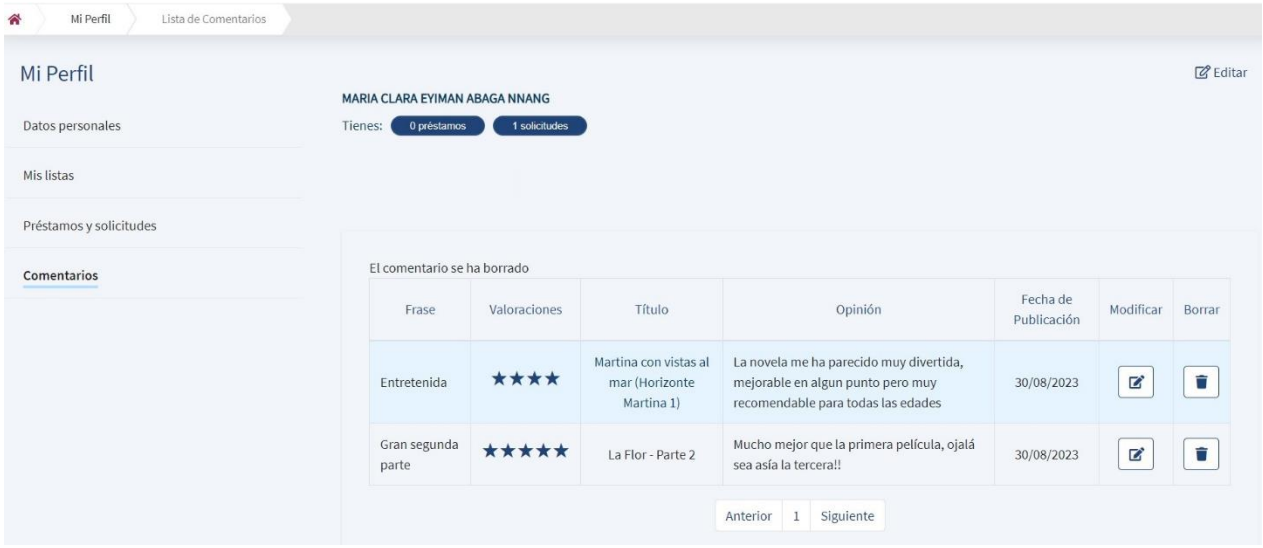
Sección de comentarios en el perfil

En la parte izquierda de la página del perfil, se puede acceder a la sección de comentarios, en los que el usuario autenticado puede observar las valoraciones que ha realizado, además de poder modificarlas y borrarlas. Como se ve en la parte inferior izquierda de la imagen, al colocar el puntero sobre el título del producto, se redirige al usuario al detalle del producto.



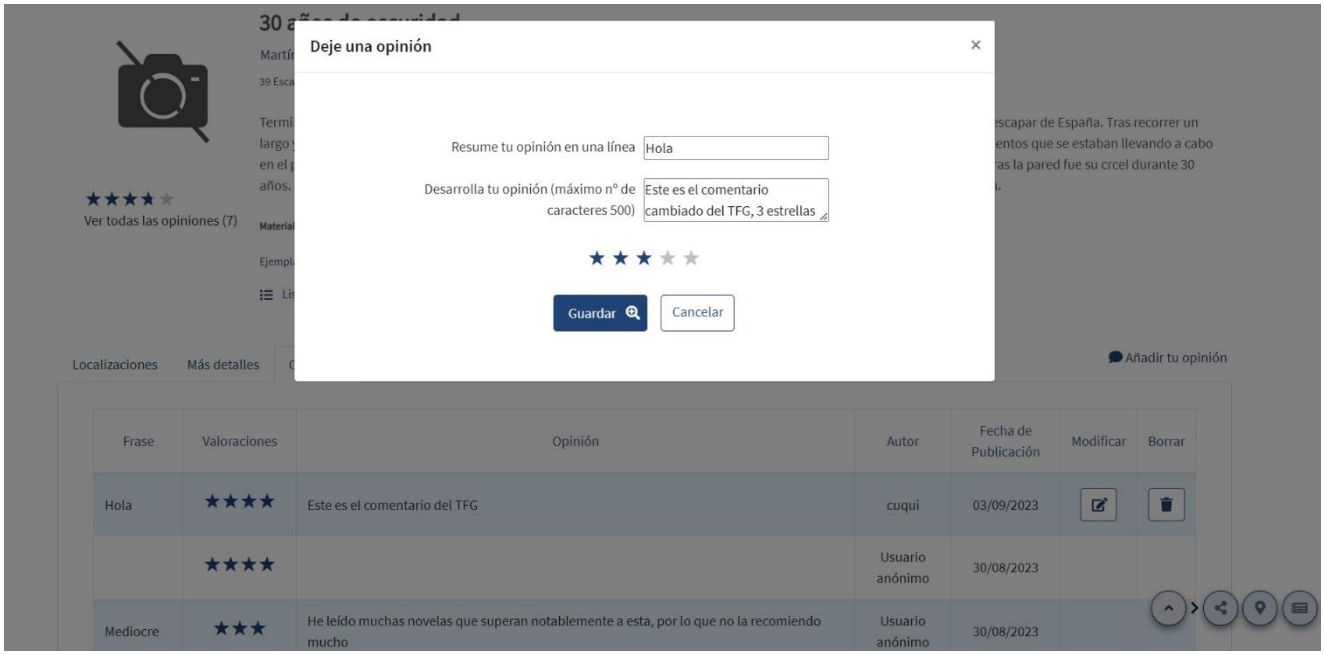
Borrar un comentario

Tras pulsar en el botón de borrado de la primera valoración, esta se elimina y la página se recarga, mostrando la misma tabla, pero sin esa valoración. Además, se muestra el mensaje “El comentario se ha borrado” en pantalla.

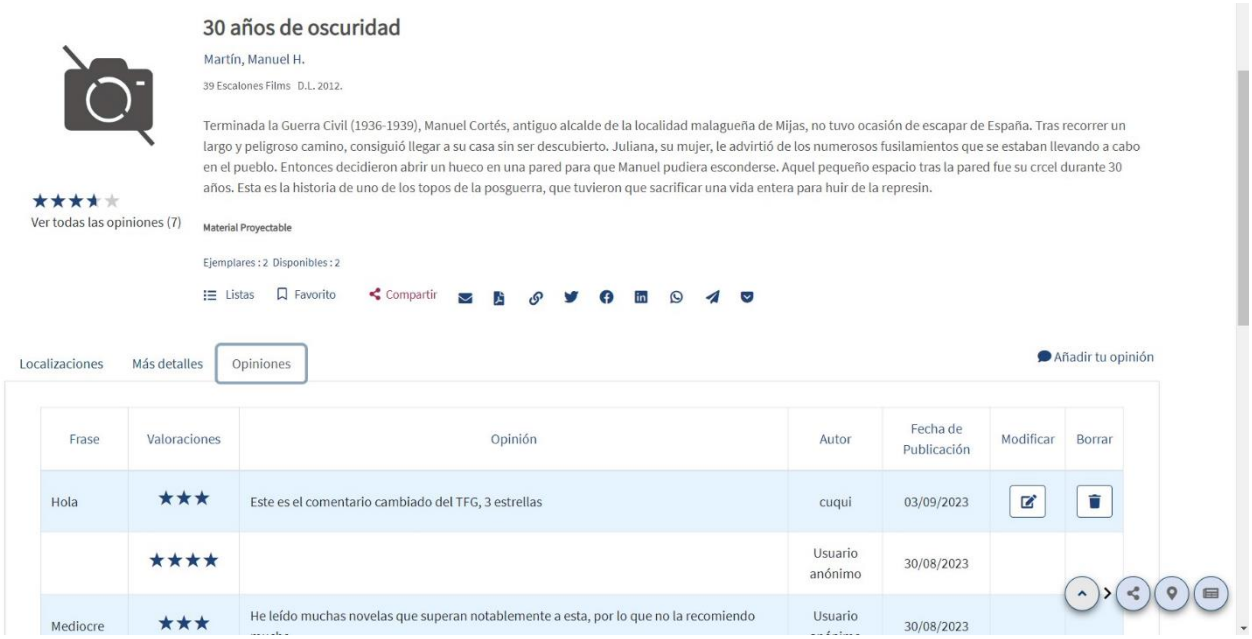


Modificación de un comentario

A continuación, se modifica la valoración creada en el anexo anterior. Pulsando en el botón con el icono del lápiz, se accede a un formulario muy similar al de creación, solo que ahora la información del comentario a modificar aparece al abrir el formulario. Una vez abierto el formulario, se cambia la puntuación a 3 estrellas y la descripción respecto a la anterior valoración creada.

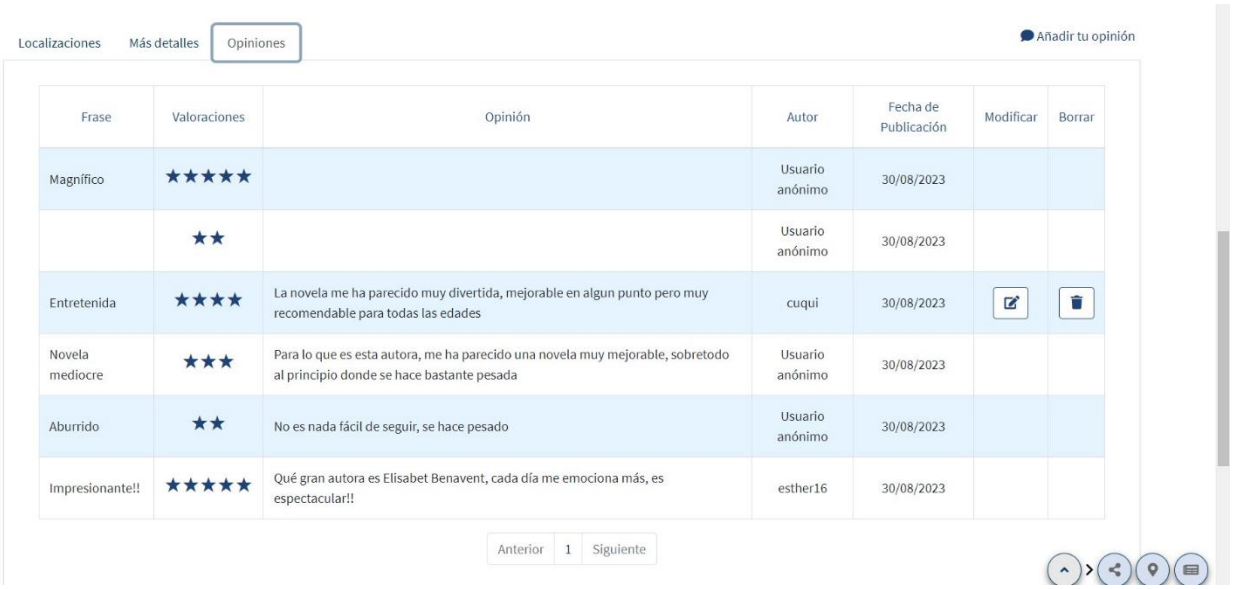


Como se observa en la imagen de debajo, la valoración del usuario “cuqui” se modifica correctamente.



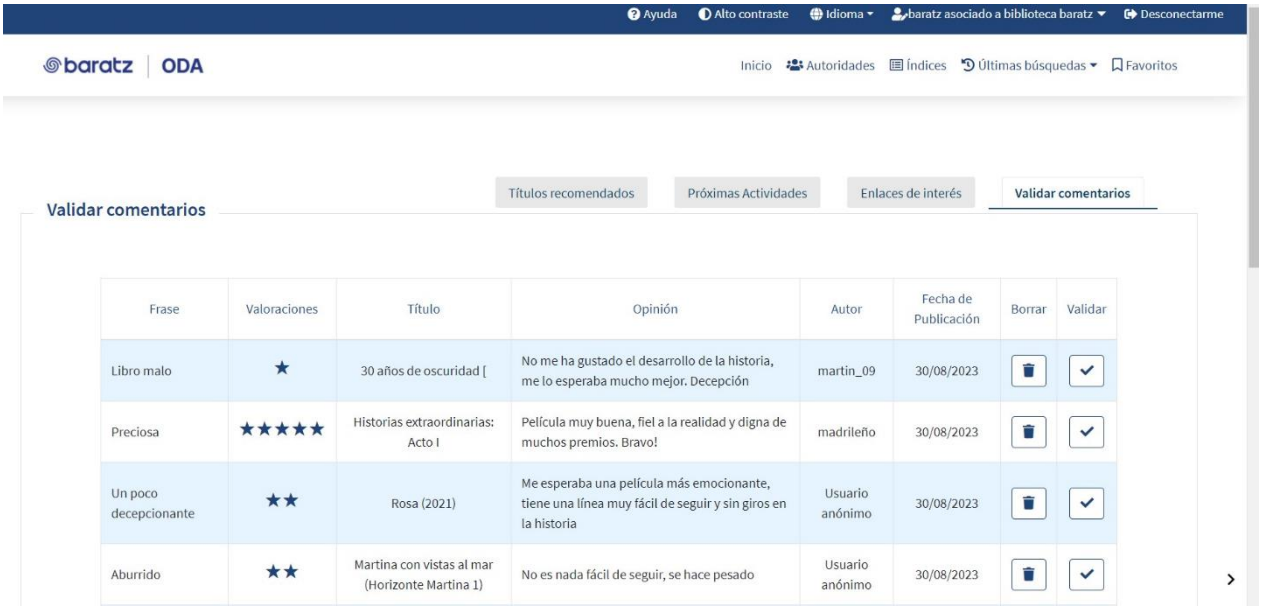
Borrado y modificación desde el detalle de un producto

Si el usuario accede al detalle de un producto del que ha hecho una valoración, podrá modificarla o borrarla desde esa página, sin necesidad de acceder a la sección de comentarios de su perfil.

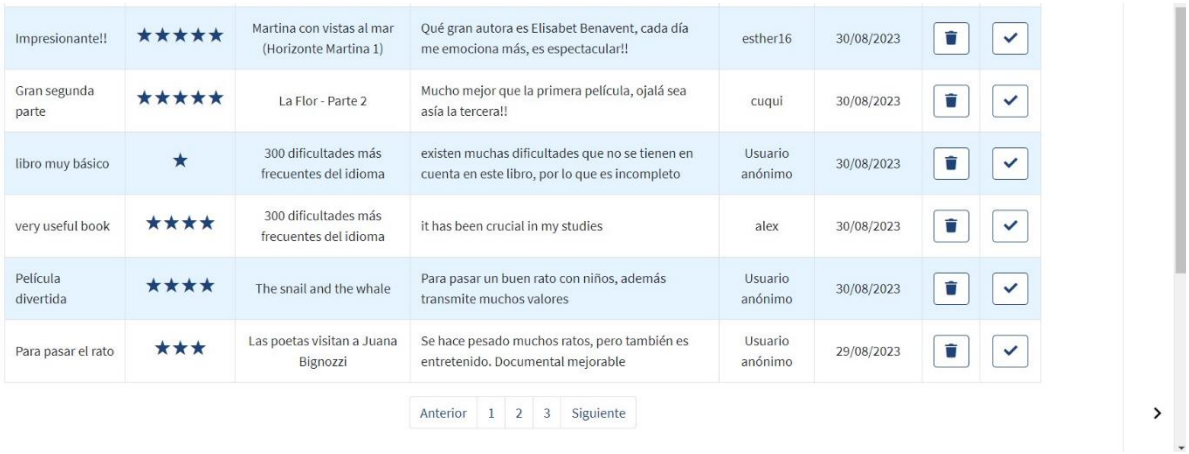


ANEXO C. MANUAL USUARIO ADMINISTRADOR

Tras iniciar sesión con las credenciales del usuario administrador de una biblioteca, se puede acceder a la sección “Validar comentarios”, en la que el usuario podrá validar u ocultar y eliminar las valoraciones que están registradas en su biblioteca.

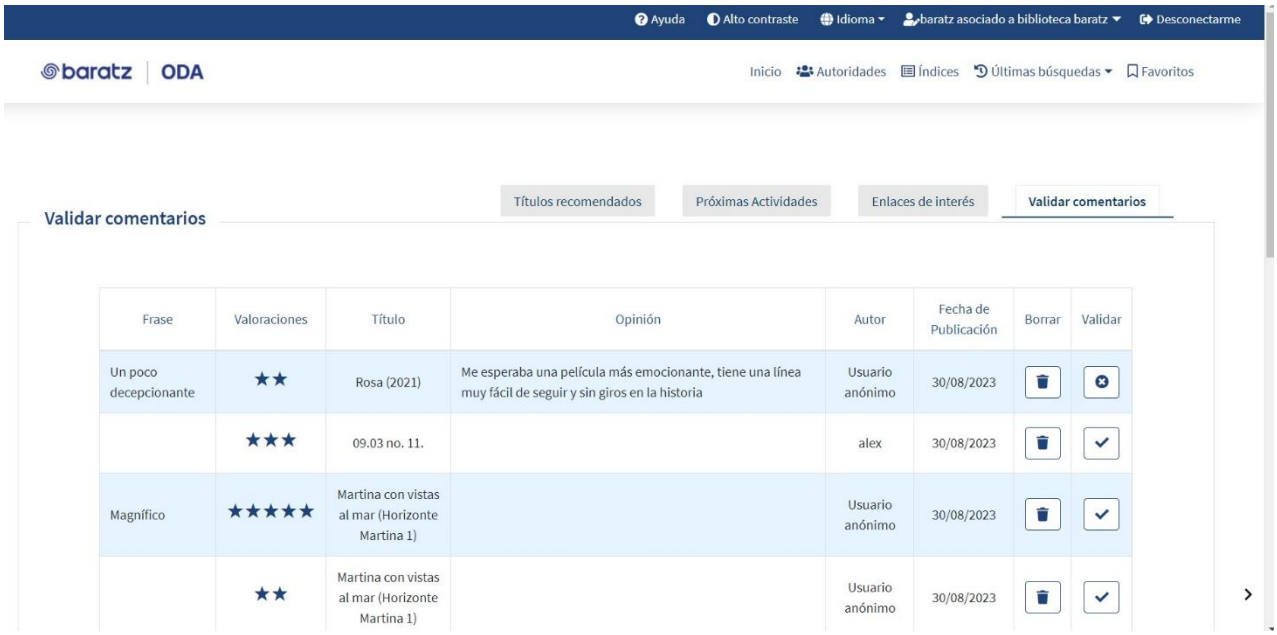


Como se observa al final de la imagen de debajo, se navega entre las páginas de valoraciones con los botones “Anterior” y “Siguiente”.



Validación de comentarios

En las imágenes de la página anterior, todas las valoraciones están validadas, por ello aparece un tick en la sección “Validar” de la derecha del todo. Si se pulsa sobre el tick de alguna valoración, esta se ocultará y no estará disponible en la aplicación, solo la podrá visualizar el administrador. En la imagen de debajo, se oculta la valoración de la película “Rosa”.



Si se accede al detalle de esa película desde cualquier usuario que no sea el administrador, no aparecerá la valoración que el administrador ha ocultado, como se observa en la siguiente imagen. Ahora esa película aparece sin ninguna valoración.