UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una Biblioteca Web de Dominio Público

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Atienza Rodrigo, Abel

Tutor/a: Valderas Aranda, Pedro José

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de una Biblioteca Web de Dominio Público

Trabajo Final de Grado

Grado en Ingeniería Informática

Autor: Abel Atienza Rodrigo

Tutor: Pedro José Valderas Aranda

Curso 2022-2023

Resumen

Este trabajo consiste en el desarrollo de una aplicación web con el fin de albergar una biblioteca de dominio público. A diferencia de los documentos que se encontrarían en una biblioteca convencional, los de esta biblioteca son creados por los propios usuarios que la utilizan, dejándolos accesibles para que cualquiera de los demás usuarios pueda disfrutar de los contenidos de su obra. El desarrollo de este trabajo se ha dividido en diferentes apartados, que van desde el análisis de mercado y de requisitos, pasando por el diseño de los diferentes componentes de la aplicación y la implementación de estos, para finalmente llegar a una etapa de validación para la solución resultante.

Palabras clave: Aplicación web, Desarrollo Web, Biblioteca, *Frontend*, *Backend*, React, Componente, *Framework*, .NET, Azure

Resum

Aquest treball consisteix en desenvolupar una aplicació web per tal d'allotjar una biblioteca de domini públic. A diferència dels documents que es trobarien a una biblioteca convencional, els d'aquesta biblioteca són creats pels mateixos usuaris que la utilitzen, deixant-los accessibles perquè qualsevol dels altres usuaris pugui gaudir dels continguts de la seva obra. El desenvolupament d'aquest treball s'ha dividit en diferents apartats, que van des de l'anàlisi de mercat i de requisits, passant pel disseny dels diferents components de l'aplicació i la implementació d'aquests, per finalment arribar a una etapa de validació per a la solució resultant.

Paraules clau: Aplicació web, Desenvolupament web, Biblioteca, *Frontend*, *Backend*, React, Component, *Framework*, .NET, Azure

Abstract

This work consists of the development of a web application in order to host a public domain library. Unlike the documents that would be found in a conventional library, those in this library are created by the users who use it, leaving them accessible so that any of the other users can enjoy the contents of their work. The development of this work has been divided into different sections, ranging from the analysis of the market and requirements, through the design of the different components of the application and their implementation, to finally reach a validation stage for the resulting solution.

Key words: Web Aplication, Web Development, Library, Frontend, Backend, React, Component, Framework, .NET, Azure

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

Las bibliotecas son lugares repletos de documentos que brindan gran cantidad de conocimiento. Desde relatos históricos de nuestro pasado hasta obras de literatura fantástica que hacen sumergir al lector en otros mundos, pasando por un libro de recetas con instrucciones de cómo preparar todo tipo de platos mediterráneos. Todos estos tienen la misma finalidad, la de brindar al objetivo interesado un contenido excepcional para su disfrute. La mayoría de estos documentos son escritos por grandes escritores u otras personas con un gran renombre detrás, sin embargo, no quiere decir que otros autores no conocidos no tengan él ni potencial ni la capacidad de crear documentos de semejante magnitud. Hoy en día, con el auge de la digitalización, muchas de estas bibliotecas son visitadas cada vez menos, por lo que cada vez se ven más esfuerzos por digitalizar estos medios tradiciones. Es aquí donde, aprovechando este auge, se presenta una plataforma, en la cual los documentos almacenados son proporcionados por las propias personas que visiten esta biblioteca.

Esta plataforma deja libertad a sus usuarios a publicar sus propios documentos, los cuales poblaran la biblioteca. Ya sea un escritor veterano con algo de experiencia en el mundo de la escritura o uno aficionado con poca o nula experiencia anterior, aquí podrá exponer sus creaciones con otros usuarios similares a ellos. Los usuarios que visiten la página como meros lectores, no solo podrán leer los diferentes documentos de esta, sino además dar su opinión sobre estos, proporcionando *feedback* a los propios escritores. La plataforma ha sido creada teniendo en cuanta la importancia e influencia de las redes sociales y de cómo estas han forjado un estándar en la manera de como interactuar con aplicaciones.

En este documento se muestra la creación de una aplicación web que traslada el concepto de biblioteca al ámbito digital, en la cual son los propios usuarios los que llenaran las estanterías digitales con documentos disponibles para todo aquel que la visite. Para poder desarrollar este proyecto, han sido necesarios los conocimientos adquiridos durante el grado, desde la planificación del trabajo a desarrollar, pasando por el diseño de la aplicación, hasta el desarrollo de esta; así como conocimientos adquiridos fuera del grado, como el de algunas de las tecnologías usadas para crear la aplicación.

1.1 Motivación

La principal motivación de este Trabajo Final de Grado ha sido la oportunidad de profundizar conocimientos y hacer uso de distintas herramientas y *frameworks* del ámbito del desarrollo de aplicaciones web, puesto que este campo sí que se profundiza en la rama de Tecnologías de la Información del Grado de Ingeniería Informática, pero haciendo uso de herramientas básicas. Además, en esta rama se profundiza mucho más la parte

frontend del desarrollo web, por lo que este proyecto ha servido también para aumentar mis conocimientos sobre las dos caras del desarrollo web.

Por otro lado, las aplicaciones web permiten una gran flexibilidad debido a su compatibilidad con múltiples tipos de dispositivos y sistemas operativos, por lo que los usuarios pueden acceder desde cualquier parte a ellas sin importar la compatibilidad con sus dispositivos.

Finalmente, con el auge de las redes sociales y la digitalización de ámbitos clásicos en los últimos años, la idea de la combinación de elementos de ambas sugería una propuesta interesante.

1.2 Objetivos

El objetivo principal de este proyecto consiste en el desarrollo de una aplicación web de una biblioteca de dominio público. En ella, los usuarios podrán leer distintos tipos documentos dentro de la página. Estos documentos serán proporcionados por los propios usuarios, los cuales podrán además valorar los documentos existentes dar su opinión respecto al contenido de estos, así como proporcionar *feedback* al propio escritor de la calidad de su obra.

Para ello, la aplicación debe dar soporte a las siguientes acciones:

- La aplicación debe permitir la filtración de documentos por categorías
- La aplicación debe permitir la ordenación de documentos según su antigüedad o valoración
- La aplicación debe permitir el inicio de sesión y el registro de usuarios
- La aplicación debe permitir a los usuarios registrados la visualización de documentos
- La aplicación debe permitir a los usuarios registrados la subida de documentos a la web
- La aplicación debe permitir a los usuarios registrados la valoración de los distintos documentos, así como el cambio de esta si el usuario cambia de opinión

1.3 Estructura del documento

Este proyecto se divide en una serie de capítulos, en los que en cada uno se explican y detallan los diferentes procedimientos que se han seguido para la realización de este. A continuación, se muestran las diferentes secciones junto a una breve descripción de lo que se relata en ellas:

1. Estado del arte: aquí se muestran aplicaciones cuyas funcionalidades o conceptos son similares a los de la propuesta, analizando dichas funcionalidades con las que contará la aplicación de este proyecto.
2. Metodología: en este apartado se explica la metodología aplicada a este proyecto, la metodología incremental, junto a los distintos *sprints* en los que se divide el proyecto.
3. Análisis y Especificación de Requisitos: en este apartado se recogen y se plantean los requisitos que la aplicación debe soportar, así como la explicación de los casos de uso que surgen de estos, los cuales determinan las diferentes funcionalidades y elementos que componen la aplicación a desarrollar.

4. Análisis Conceptual y Diseño: aquí se muestra el diagrama de clases planteado en base a los casos de uso, así como el modelo de base de datos, junto a los prototipos de las interfaces de la aplicación.
5. Desarrollo de la solución: en este capítulo se explica la arquitectura que sigue la aplicación, las diferentes herramientas utilizadas en el desarrollo de la aplicación, así como ejemplos del desarrollo de esta.
6. Producto desarrollado: en este capítulo se presenta el resultado la aplicación desarollada de una forma visual para poder observar con detalle el resultado obtenido.
7. Validación: aquí se muestran los métodos de validación que se han utilizado en el proyecto, así como se han realizado.
8. Conclusiones y trabajo futuro: en este capítulo se resume el trabajo realizado, así como una conclusión de este, además de mostrar una serie de planteamientos para mejorar la aplicación de cara a futuro.

CAPÍTULO 2
Estado del arte

Antes de empezar a plantar el desarrollo de la aplicación es importante realizar un estudio sobre otros servicios similares al nuestro que ya existen en Internet, para así nutrirse de ideas que se puedan plasmar en la aplicación, así como de otras que evitar. Estos servicios se pueden dividir en bibliotecas de ámbito tradicional y bibliotecas digitales. A continuación se van a mostrar las diferentes webs las cuales han servido de referencia para este proyecto.

2.1 Biblioteca tradicional

Un ejemplo de este tipo de web sería la web de la Generalitat Valenciana 'Catálogo de la Red Electrónica de Lectura Pública Valenciana', la cual sirve a modo de explorador de libros por biblitecas de toda la Comunidad Valenciana. En las Figuras 2.1 y 2.2 se puede observar como la web simplemente sirve a modo de buscador de libros, pero en ningun momento te permite la lectura de estos. Sin embargo, proporciona varias herramientas bastante útiles para buscar dichos libros, como una gran cantidad de etiquetas por las que filtrar o un buscador para filtrar por títulos o etiquetas.

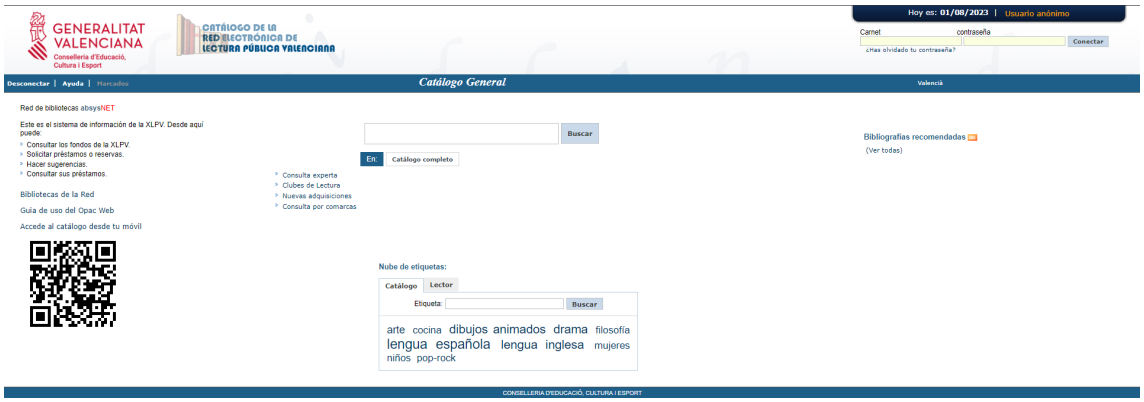

Figura 2.1: Ejemplo de biblioteca tradicional I

Figura 2.2: Ejemplo de biblioteca tradicional II

Dadas sus características, esta web no satisface las necesidades del usuario, el cual quiere acceder a diferentes documentos desde un dispositivo digital para consumirlos en su casa o en cualquier otro lado.

2.2 Bibliotecas digitales

Las webs mostradas en este apartado coinciden en que ambas son repositorios completamente digitales en los que poder leer diferentes tipos de documentos. La diferencia entre ambas es que una esta mas centralizada entorno al usuario que la otra. A continuación, se muestran ambos ejemplos.

2.2.1. Biblioteca Virtual Miguel de Cervantes

La web llamada 'Biblioteca Virtual Miguel de Cervantes' es una biblioteca digital española la cual reúne obras hispánicas como objetivo de la difusión de estas. En la Figuras 2.3 y 2.4 se puede observar como la web permite al usuario, al igual que en el ejmplo anterior, realizar una busqueda por filtros para encontrar lo que busca, además de esta vez si dejar al usuario leer los libros dentro de la propia web. Sin embargo, a pesar de la comodidad que proporciona la web a la hora de leer documentos, el usuario no puede proporcionar contenido a esta de forma sencilla e inmediata.

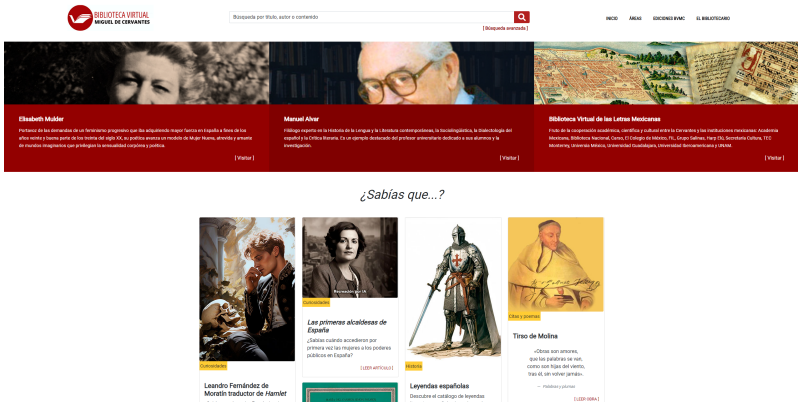

Figura 2.3: Ejemplo de biblioteca virtual I

Figura 2.4: Ejemplo de biblioteca virtual II

2.2.2. Wattpad

Por último esta la web 'Wattpad', la cual difiere un poco con las mostradas anterirormente, ya que es una plataforma online de lectura y escritura, también categorizada como red social. En ella los usuarios pueden publicar todo tipo de documentos narrativos, desde novelas hasta blogs, entre otros. En las Figuras 2.5 y 2.6 se puede observar que, como en los casos anteriores, la web cuenta con un sistema de filtrado para buscar los diferentes documentos, así como también permite al usuario leer los documentos en la propia web. Además cuenta con un enfoque más iteractivo para el usuario, permitiendo crearse una cuenta con la que poder subir sus propios documentos a la plataforma, así como guardar como favoritos los documentos de otros usuarios.

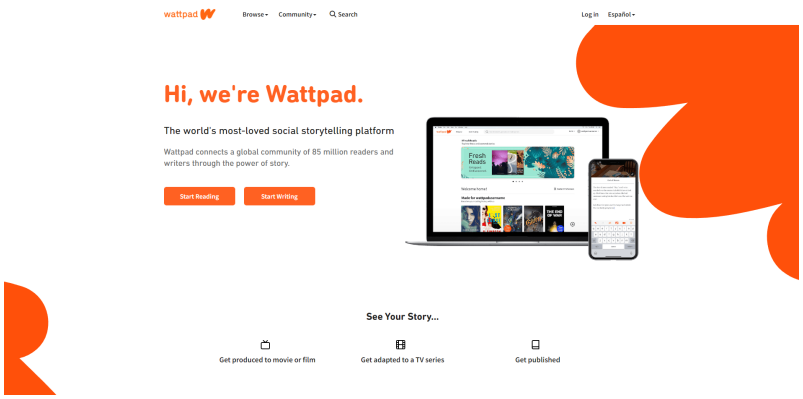

Figura 2.5: Ejemplo de wattpad I

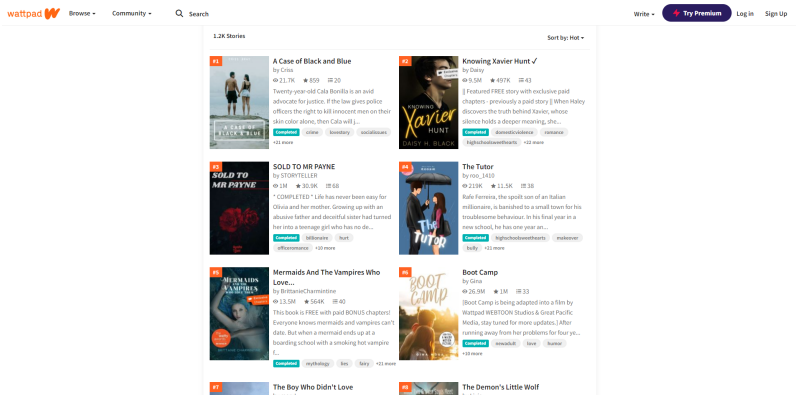

Figura 2.6: Ejemplo de wattpad II

Sin embargo, pese a ser una web de contenido gratuito, presenta algunos documentos para los cuales es necesario pagara si quieres acceder a ellos, privando al usuario de una parte del contenido tras un muro económico. Además, la web solo te permite añadir a favoritos los documentos, de forma que su valoración solo aumenta positivamente, dejando al usuario la imposibilidad de mostrar su desagrado con las obras.

2.3 Biblioteca propuesta

Por otro lado, la aplicación propuesta, como se ha mendiconado anteriormente, permite al usuario acceder a un amplio catalogo completamente gratuito y libre de derechos, el cual podra visualizar en la misma web, la cual proporciona una serie de herramientas de filtrado para hacer la busqueda mas amena y sencilla. Así mismo, el usuario, tras iniciar sesión, podrá mostrar su satisfacción con los documentos de la web, tanto si es buena como mala. Además, la web permite a los usuarios poblar su contenido con documentos subidos por ellos mismos.

CAPÍTULO 3
Metodología

Para este proyecto se ha utilizado un modelo de metodología incremental. Este modelo tiene como objetivo un crecimiento progresivo de la funcionalidad, de modo que las tareas, las cuales corresponden a diferentes funcionalidades de la aplicación, se dividen en iteraciones. Cada una de estas iteraciones corresponde a un incremento, en los cuales se desarrolla y se validan los resultados de la funcionalidad correspondiente. Estas iteraciones son dependientes unas de las otras, de forma que cada una suponga un avance respecto a la anterior [1]. En la Figura 3.1 se muestra un esquema del modelo.

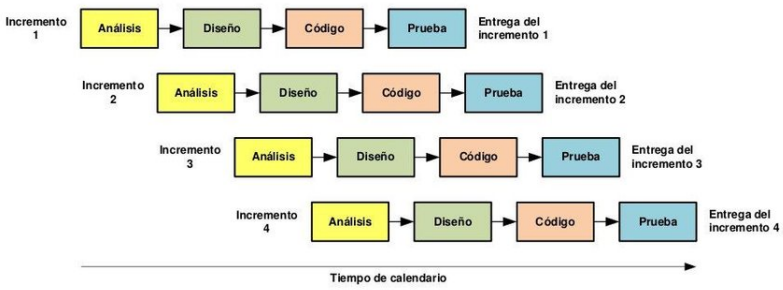

Figura 3.1: Esquema del modelo incremental

En cuanto a las funcionalidades respecta, estas se han determinado en base a los requisitos funcionales de la aplicación, junto también al diagrama de casos de uso. Estas funcionalidades, debido a la naturaleza de esta metodología, pueden ser validadas sin que sea necesario de que el conjunto completo de la aplicación haya finalizado.

Por poner un ejemplo, uno de los incrementos consiste en el diseño de la Base de Datos. Para cumplir este incremento hay que basarse en el diagrama de clases y los casos de uso, los cuales son incrementos anteriores, para determinar los objetos a persistir, los datos de las clases y las relaciones entre ellas.

Al finalizar este incremento, se puede avanzar hacia siguientes incrementos, como el de la creación e implementación de la API. Para cumplir este incremento hay que basarse en el diagrama de clases, los casos de uso y el modelo de base de datos, para determinar las relaciones entre entidades, las cuales poblarán más adelante la Base de Datos. Una vez creadas estas entidades, se definirán en Base de Datos mediante *code first*. A continuación, se crearán los diferentes endpoints y métodos de estos, los cuales serán los encargados de dar respuestas a las operaciones solicitadas por el *frontend*. Finalmente, se realizarán pruebas, haciendo uso de los *endpoints* mencionados anteriormente, enviando peticiones a la API para comprobar la correcta funcionalidad de los métodos creados.

Una vez más, al quedar este incremento finalizado, el proyecto puede avanzar al siguiente, como puede ser por ejemplo la creación de una versión primigenia de la interfaz de la aplicación en la que en la ventana principal se muestran los diferentes documentos que se encuentran almacenados en la Base de Datos mediante una comunicación con la API. Al igual que otros, este incremento también se apoya en anteriores, como el prototipado de la interfaz de usuario, la especificación de requisitos y los casos de uso.

Los incrementos realizados para el desarrollo de este proyecto son los siguientes:

1. Caso de estudio y especificación de requisitos: aquí se determina las acciones que la aplicación debe permitir realizar al usuario.
2. Casos de uso: en este incremento observan los diferentes escenarios y las acciones que pueden tomar los usuarios.
3. Diagrama de clases: aquí se definen los diferentes objetos con los que la aplicación y el usuario podrán interactuar.
4. Diseño de la Base de Datos: incremento en el cual se define la información que persistirá en el sistema. También se muestran las características de la Base de Datos, así como los objetos que se deben almacenar, definidos en el incremento anterior, así como sus tablas y relaciones.
5. Diseño y prototipado de las interfaces de usuario: aquí se diseñan los elementos visuales de la aplicación de la forma más apropiada para los usuarios.
6. Desarrollo e implementación de la solución: este es el incremento donde se desarrolla el código necesario para satisfacer los diferentes escenarios determinados por los casos de uso, así como la creación de las interfaces diseñadas en el incremento del diseño de estas, además del despliegue de la aplicación.
7. Validación y calidad: último incremento en el que se valida el uso de la aplicación a través de cuestionarios y comentarios sobre su uso y funcionamiento para poder sacar conclusiones de su calidad.

CAPÍTULO 4
Análisis y Especificacción de Requisitos

4.1 Requisitos iniciales

Para obtener los requisitos funcionales que la aplicación web debe proporcionar se ha de leer el caso de estudio del Apéndice A. Una vez realizada la lectura del caso de estudio, se pueden determinar los requisitos que debe cumplir la aplicación para los usuarios que utilicen la web. A continuación, se muestran dichos requisitos:

- El sistema debe permitir filtrar los documentos por categorías y por título.
- El sistema debe permitir iniciar sesión o registrarse al usuario.
- El sistema debe permitir subir documentos a los usuarios, pero solo a aquellos que hayan iniciado sesión.
- El sistema debe permitir leer los documentos a los usuarios, pero solo a aquellos que hayan iniciado sesión.
- El sistema debe permitir valorar a los usuarios que hayan iniciado sesión, positiva o negativamente, los documentos, pudiendo cambiar la valoración posteriormente.

4.2 Casos de uso

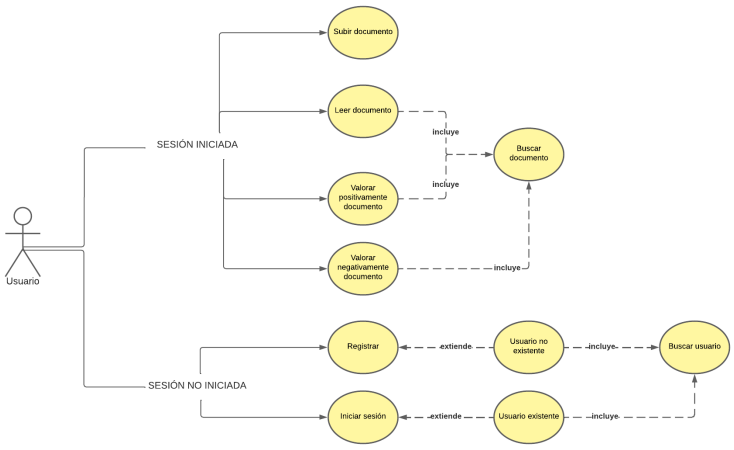

Figura 4.1: Diagrama de casos de uso

En la Figura 4.1 se muestra el diagrama de casos de uso de la aplicación, donde el único agente que participa es el usuario que utiliza la aplicación. El usuario podrá visualizar los diferentes documentos de la web tras que el sistema se los proporcione, por lo que esté deberá consultar la información almacenada en la Base de Datos.

Si el usuario no ha iniciado sesión, tan solo podrá observar los documentos disponibles, pero no podrá acceder a ellos. Lo que sí que podrá es filtrar la búsqueda de documentos según la categoría o el título de estos. De la misma forma, el usuario también podrá ordenar los documentos en base a su antigüedad o valoraciones. Para ello el sistema deberá consultar y filtrar los resultados de los documentos según los filtros aplicados, mostrándoselos al usuario.

En el caso de que quiera leerlos, el usuario deberá iniciar sesión, si ya tiene un usuario creado, proporcionando su nombre de usuario y su contraseña. El sistema iniciará la sesión del usuario, en caso de que exista un usuario ya registrado con esas credenciales. Por otro lado, si el usuario no está registrado en el sistema, deberá hacerlo proporcionando su nombre de usuario, correo electrónico y contraseña. El sistema almacenará la información del usuario en caso de que este no exista ningún usuario cuyos datos correspondan a los introducidos e iniciará sesión.

En el caso en el que el usuario quiera leer un documento, primero ha de haber pasado el caso de iniciar sesión. Una vez hecho, el sistema consultará el documento seleccionado y se lo mostrará al usuario para que este pueda leerlo.

Al igual que en el caso anterior, si el usuario desea valorar el documento primero ha de iniciar sesión. Tras hacerlo, a este se le presentará la opción de valorarlo positiva o negativamente. Si el usuario lo valora de una forma u otra, el sistema actualizará la valoración de ese documento en específico y mostrará al usuario el resultado de su acción, actualizando la información presentada ante él.

Pero si el usuario posteriormente se arrepiente de su valoración y desea cambiarla, este podrá cambiarla. En ese caso, el sistema nuevamente actualizará la valoración del documento y volverá a actualizar la información que se le presenta al usuario.

Por último, se contempla el escenario en el que el usuario quiere subir un documento, proporcionando los datos de este junto a una portada y el propio documento. El sistema deberá almacenar esos datos y persistirlos, de modo de que ese documento sea visible para todos los usuarios.

A continuación, se presentan una serie de escenarios en los cuales se encuentran contenidos los casos de uso comentados anteriormente, los cuales ejemplifican las acciones que pueden realizar los usuarios.

4.2.1. Escenario 1: El usuario se registra en la web

Guillermo, un joven apasionado por la lectura y la escritura, está buscando información que le sirva como fuente de inspiración para un nuevo proyecto que lleva entre manos. Tras un tiempo buscando, se topa con la web 'Biblioteca', en la que encuentra documentos de todo tipo, desde narraciones históricas hasta obras de ficción, como en la que está trabajando él. A Guillermo le parece que en la web hay el contenido suficiente para poder empaparse de ideas de las que luego podrá tomar inspiración para su trabajo, por lo que decide registrarse en la web, ya que ha comprobado que para poder leer los documentos hace falta una cuenta de usuario. Para ello, le dio al botón de registrarse e introdujo su nombre de usuario, su correo electrónico y su contraseña. Una vez hecho esto, su cuenta había sido creada.

4.2.2. Escenario 2: El usuario lee un documento

Durante algún tiempo, Guillermo ha estado ojeando los diferentes documentos que se encuentran alojados en la web, algunos de ellos despertando gran curiosidad en el. De repente, se topa con un documento cuya mera portada ya cautiva su atención, disparando su atención y haciendo que quiera saber más del contenido de este. Tras leer su descripción, se da cuenta que ha encontrado un documento que puede brindarle grandes cantidades de ideas las cuales podrá plasmar en la obra en la que trabaja. Además, las valoraciones de este documento son muy positivas, por lo que Guillermo se decide a leer su contenido. Para ello, pulsó en el botón de leer el documento, haciendo así que, al ya tener una sesión iniciada, la web le mostrará el contenido del documento, pudiendo así comenzar a empaparse de ideas.

4.2.3. Escenario 3: El usuario valora algunos documentos

Tras una ardua búsqueda de conocimiento, Guillermo ha leído ya varios documentos. Algunos de ellos le han servido de gran ayuda, proporcionándole gran cantidad de ideas, siendo, además, a su parecer de lector aficionado, obras de buena calidad, por lo que decide darles una valoración positiva. Para ello, Guillermo pulsó en el botón 'Upvote' de esas obras, viendo, así como la valoración de estas aumentaba. Por otro lado, también se topó con otros documentos los cuales no fueron tanto de su agrado. A su parecer, eran pobres narrativamente o no tenían mucha coherencia lo que decían. Por esos motivos, Guillermo decidió valorarlos negativamente. De forma similar a como ya hizo anteriormente, pulsó el botón 'Downvote' de estas obras, viendo como su valoración descendía. Sin embargo, y tras una revisión de algunos documentos, Guillermo se dio cuenta de que quizás su criterio había sido demasiado duro, y ya que reflexionar es de sabios, decidió cambiar las valoraciones de algunos documentos. Para ello, pulso nuevamente en el botón 'Upvote' de esas obras, viendo como su valoración cambiaba.

4.2.4. Escenario 4: El usuario realiza una búsqueda por filtros

Guillermo vuelve a la web después de un tiempo para volver a leer un documento de ficción que leyó tiempo atrás en el cual aparecían conceptos e ideas de las cuales podría inspirarse para su nueva obra. El problema es que no puede encontrar dicho documento a primera vista, puesto que el contenido de la web ha aumentado considerablemente desde la última vez que hizo uso de esta. Guillermo recuerda perfectamente que una de las categorías de este documento era ficción, por lo que mediante el selector de categorías filtra los documentos para que solo aparezcan los documentos de ficción. Aun así, sigue habiendo demasiados y no consigue encontrar el documento que busca. Entonces, Guillermo recuerda que el titulo contenía la palabra 'sombras', por lo que utiliza el buscador para filtrar los documentos que contengan esa palabra. Finalmente, tras reducir la búsqueda de documentos a un número más reducido, Guillermo reconoce la portada del documento que estaba buscando.

4.2.5. Escenario 5: El usuario sube un documento

Tras mucho tiempo y esfuerzo, Guillermo ha terminado por fin la obra literaria en la que ha estado trabajando los últimos meses. Tras pensar que hacer ahora con su nueva creación, se decide por publicarla en la biblioteca web que ha estado visitando recientemente, ya que piensa que es un buen lugar para que otros lectores aficionados como él puedan acceder a su obra. Guillermo entra a la web y se dirige al apartado 'Subir'. Una vez ahí,

Guillermo introduce el título de su obra, junto a una pequeña sinopsis de la trama de esta, selecciona la categoría de ficción, selecciona una portada que le hizo un amigo y por último selecciona el documento PDF que contiene su obra. Finalmente, Guillermo pulsa el botón de subir y tras unos segundos de espera, la página le notifica que su obra ha sido subida correctamente a la biblioteca.

CAPÍTULO 5
Analisis Conceptual y Diseño

5.1 Diagrama de clases

A continuación, se muestra el diagrama de clases, las cuales son las que darán soporte a los requisitos. Para describir el sistema, se ha hecho uso del lenguaje UML (*Unified Modeling Language*), el cual es un lenguaje que permite representar las funcionalidades de un sistema y la interactividad que tienen sus elementos entre ellos. Para lograr esto, hace uso de clases, las cuales se basan en dichos objetos, y las relaciones entre estas [2].

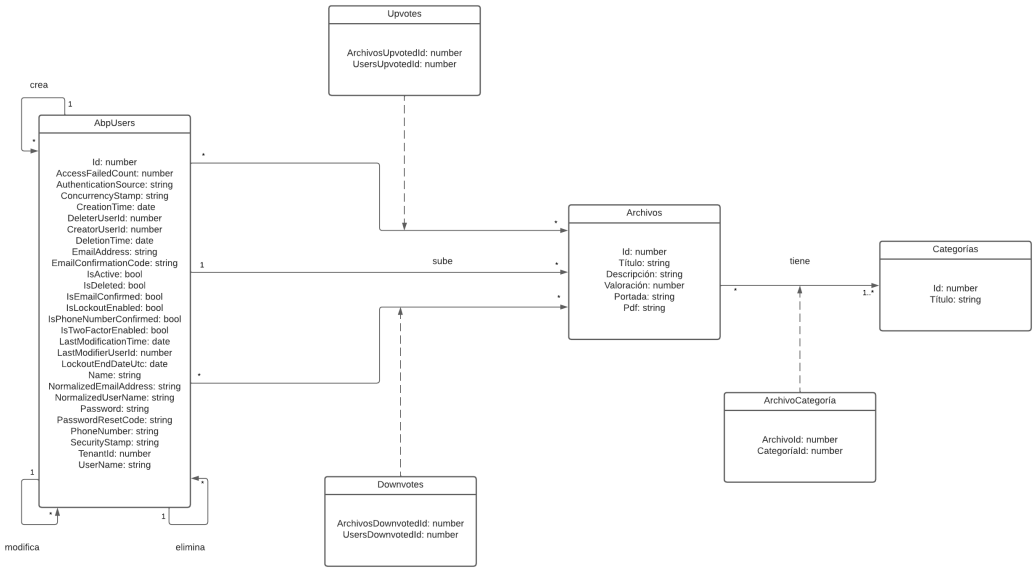

Figura 5.1: Diagrama de clases

En la Figura 5.1 se presenta el diagrama mencionado anteriormente. En esta se puede observar que para que el usuario pueda realizar las acciones descritas en los escenarios del Apartado 4.2, son necesarias las clases usuario, documento, categoría, además de un par de clases asociación para realizar las acciones de valoración, así como la asignación de categorías a documentos, ya que estas guardarán la información de las relaciones entre las clases. Con estas clases, el usuario será capaz de leer los documentos de la web, pudiendo haberlos filtrado por categorías anteriormente, así como valorar estos mismos. De este mismo modo, también podrá subir sus propios documentos a la web.

5.2 Modelo de Base de Datos

En cuanto a la Base de Datos que utiliza la aplicación, esta es una Base de Datos de tipo relacional, lo que significa que las estructuras de datos lógicas (tablas, vistas e índices) están separadas de las estructuras de almacenamiento físicas [3], y está implementada en SQL (*Structured Query Language*), el cual es un lenguaje diseñado para administrar y recuperar información de sistemas de gestión de bases de datos relacionales.

Antes de construir la Base de Datos, es necesario plantear que funcionalidadés proporcionará y que información deberá persistir. Para ello, se ha hecho uso del caso de estudio del Apartado 4.2 y del diagrama de clases del Apartado 5.1, de donde se puede determinar qué datos se deben guardar, así como todas las relaciones entre clases, ya que de estas puede aparecer nueva información que puede resultar clave para el funcionamiento de la aplicación. El resultado obtenido tras este análisis es el mostrado en la Figura 5.2.

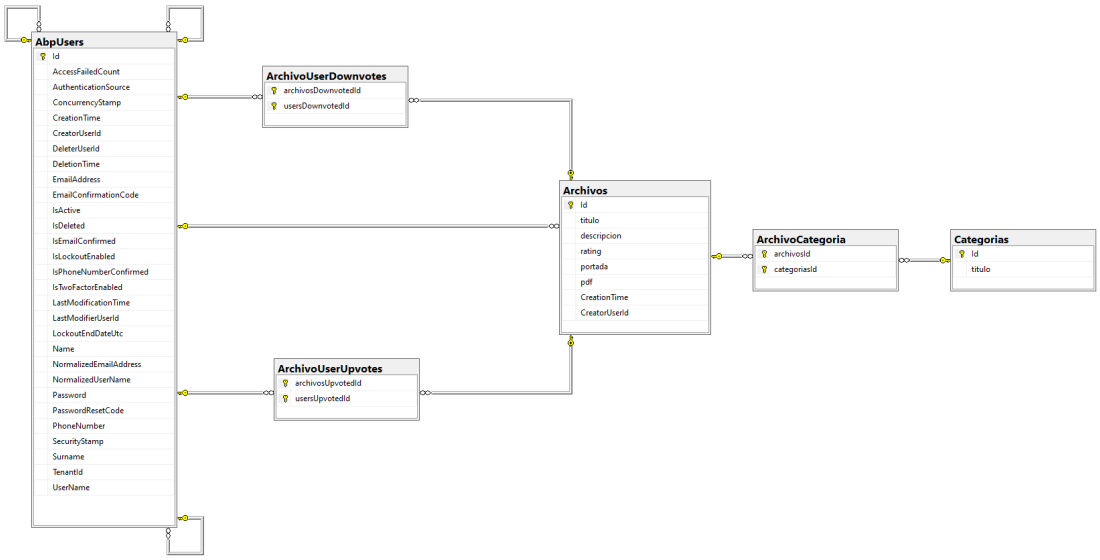

Figura 5.2: Modelo de Base de Datos

En cuanto a los usuarios, se puede ver que tiene una gran cantidad de datos a guardar. Esto es debido a que el objeto de usuario se basa en la entidad de usuario proporcionada por ASP.NET Boilerplate, por lo que cuenta con un gran número de propiedades para otras funcionalidades implementadas en el *framework*. Algunas de estás propiedades serán de utilidad para una futura apliación de las funcionalidades de la aplicación. Sin embargo, en el caso de este proyecto solo nos serán de utilidad el identificador, el nombre de usuario, el correo electrónico y la contraseña.

Por otro lado, en cuanto a los datos que la clase 'archivos' almacena, son los datos básicos de un documento, como el identificador, el título, la descripción, la valoración o *rating*, la portada y el PDF, así como la fecha en la que se creó y el usuario que ha subido el documento. Además, al existir una relación de muchos-a-muchos entre los archivos y los usuarios, se generan las tablas 'ArchivoUserDownvotes' y 'ArchivoUserUpvotes', las cuales almacenan la relación entre archivos y usuarios que los han valorado negativa o positivamente, respectivamente.

Por último, en cuanto a las categorías solo es necesario almacenar el identificador y el nombre de estas. Como se ha mencionado antes con la relación entre usuario y archivo, existe también una relación entre archivo y categoría del mismo tipo, de modo que se

genera la tabla 'ArchivoCategoría', en la cual se almacena la relación entre un archivo y sus categorías. En otras palabras, aquí se indica a que categorías pertenece un archivo.

5.3 Bocetos

Para la creación de prototipos se ha hecho uso de la herramienta *Lucidspark*, una pizarra virtual diseñada para facilitar la colaboración de desarrollo de proyectos en equipo. Esta herramienta te permite crear planes, mapas conceptuales, entre otros, y también permite la creación de *mockups*, razón por la cual se ha utilizado para realizar los prototipos. Los *mockups* permiten representar un concepto inicial del producto, el cual servirá como fundamento de su desarrollo. Como se ha mencionado, esta herramienta esta enfocada al uso en proyectos grupales, pero se puede utilizar perfecatmente en solitario de igual manera. La herramienta cuenta con una versión de pago, la cual incluye diferentes funcionalidades adicionales, pero para realizar los prototipos ha sido suficiente con la versión gratuita.

En cuanto a los prototipos, todas lás páginas contienen dos elementos principales. El primero de ellos se trata de la barra de navegación o *navbar*, la cual a tenido algunos cambios. En la versión final de esta se muesta el logo de la página, un menú desplegable con todas las categorías disponibles mediante las cuales el usuario puede filtrar los documentos, un buscador para que el usuario pueda buscar un documento por su nombre, así como los botónes de iniciar sesión o registrarse si el usuario no ha iniciado sesión o un menú desplegable con la opción de subir un documento. En el *mockup* que representa la Figura 5.3 se puede ver la existencia de un botón para subir documentos, el cual ha sido reubicado como se ha explicado anteriormente. El segundo de los elementos es el propio contenido de cada página, el cual sera diferente dependiendo de en que parte de la aplicación se encuentre el usuario. La Figura 5.3 representa la ventana principal de la aplicación, donde se muestran todos los documentos disponibles, con un menú desplegable arriba a la derecha por le que poder ordenar los documentos. Un pequeño cambio a destacar con este prototipo es que en la versión final no se puede ordenar alfabeticamente, solamente por antigüedad o valoraciones.

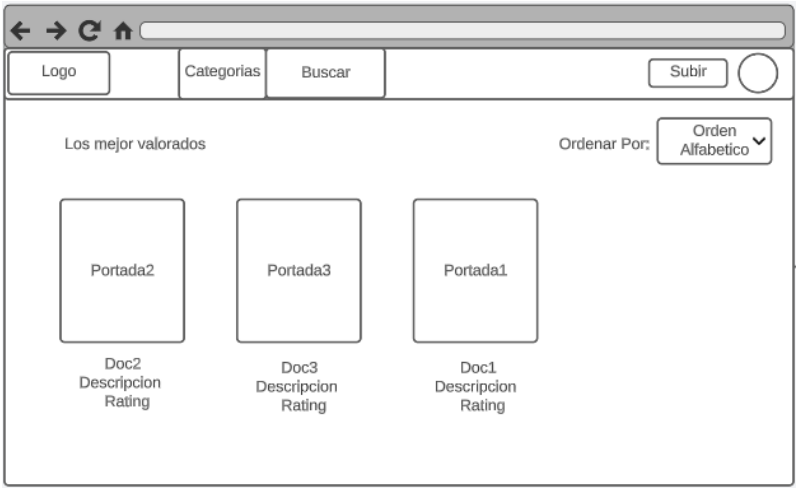

Figura 5.3: Prototipo de la ventana principal

En este prototipo no aparecen los botones de inciar sesión y registrarse debido a un cambio de diseño. En un principio, la página principal que se le mostraría al usuario era una página en la que se mostraba información sobre la funcionalidad de la web y sobre sus

contenidos, donde el usuario podria inciar sesión o registrarse. Una vez efectuada la autenticación, el usuario llegaría a la página mostrada en la Figura 5.3. Sin embargo, en la versión final se decidió que la primera ventana que el usuario viese fuera la ventana donde se muestran todos los documentos. Este cambio surge debido a la idea de que en una biblioteca física, una persona que entre lo primero que verá seran todos los libros que tiene a su disposición, categorizados en sus respectivas estanterías, a los cuales puede observar brevemente para saber si su contenido sera de su agrado y despues solicitarlo o leerlo. Al hacer este cambio, el diseño de la aplicación se acerca más a esta idea.

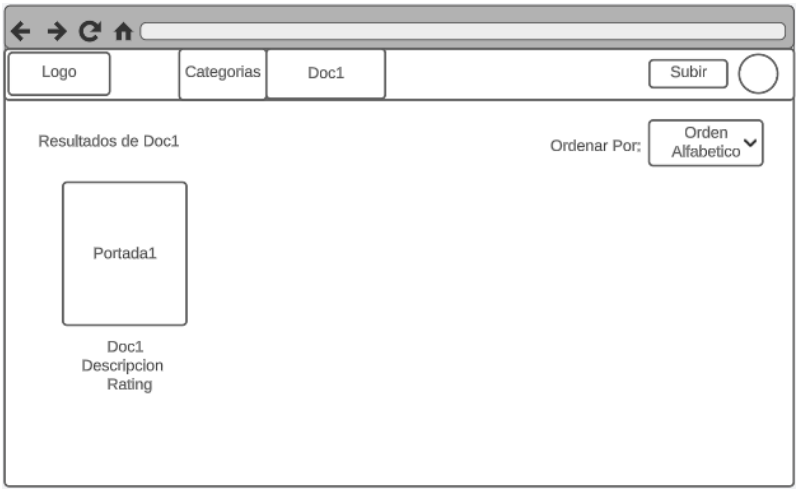

Figura 5.4: Prototipo de la ventana principal con filtro por título

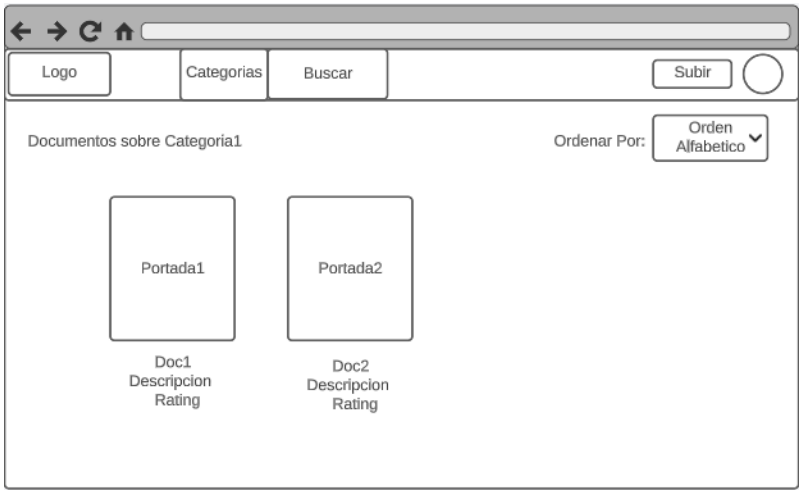

Figura 5.5: Prototipo de la ventana principal con filtro por categoría

En la Figura 5.4 se muestra el conenido de la ventana principal al aplicar un filtrado por título. El usuario busca el título del documento que quiere encontrar, haciendo uso del buscador de la barra de navegación, y obtiene los resultados que coincidan con ese título. En cuanto a la Figura 5.5, en ella se muestra el contenido de la ventana principal al aplicar un filtrado por categoría. El usuario selecciona una de las categorías por las que quiere filtrar los documentos a través del menú desplegable de la barra de navegación y se le mostrarán solo los documentos que pertenecen a esa categoría.

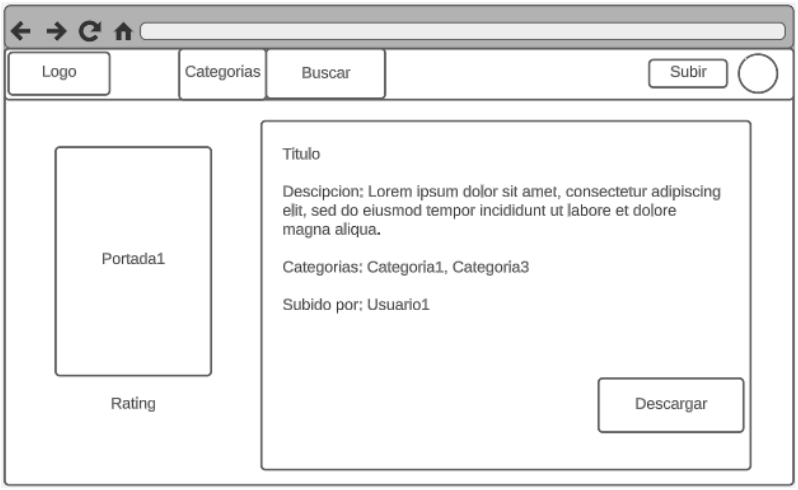

Figura 5.6: Prototipo de la ventana de previsualización del documento

En la Figura 7.3, se muestra la página que corresponde a la previsualización de un documento. En ella se muestran las características del documento, como el título, la descripción, las categorías a las que pertenece, el usuario por el cual ha sido subido el documento, la portada y la valoración de este. Abajo a la derecha de la ventana se encuentra un botón mediante el cual el usuario podría descargar el documento. Esta ventana también ha sufrido algunos cambios respecto a la versión final. En primer lugar, el botón de descargar ha sido sustituido por otro botón para visualizar el documento. Esto es debido a que en la versión final los documentos se visualizan en la propia página en vez de descargarse. Por otro lado, la valoración del documento aparece junto a la demás información de este, debido a que así la información del documento queda más centralizada. Por último, en la versión final aparecerá una opción para valorar positiva o negativamente el documento si el usuario ha iniciado sesión.

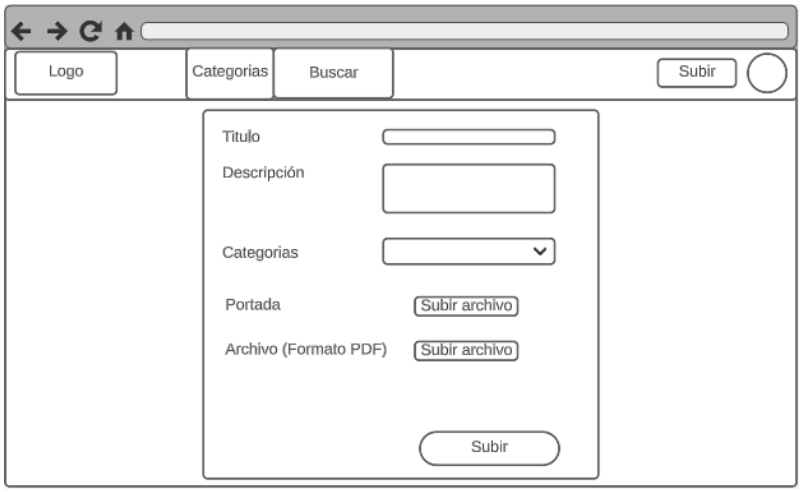

Figura 5.7: Prototipo de la ventana de subir archivo

En la Figura 5.7 se muestra la página donde el usuario puede subir documentos a la aplicación. En ella el usuario debe rellenar los campos con la información del documento. En estos campos el usuario debe especificar el título, la descripción y las categorías a las que pertenece el documento, así como seleccionar una imagen como portada y el propio documento. En la versión final se muestra un listado con todas las categorías para que le sea al usuario más sencillo visualizar cuales a seleccionado.

CAPÍTULO 6
Desarrollo de la solución

6.1 Arquitectura

Hoy en día, las páginas web modernas siguen un modelo de arquitectura en el cual se diferencia la parte del cliente de la parte del servidor. La aplicación web de este proyecto es una de ellas, siguiendo un modelo de tres. Este modelo divide la aplicación en tres capas conectadas entre sí de una forma estructurada, donde cada una realiza una función determinada. En la Figura 6.1 se muestra una representación gráfica de este modelo. Las capas de este modelo son las siguientes:

- **Capa de presentación**. Esta es la capa más externa y es la responsable de manejar la interfaz de usuario y las interacciones de este con la aplicación. Su función es mostrarle la información al usuario, la cual obtiene al comunicarse con la capa de aplicación, y registrar la información introducida por el usuario del mismo modo. Típicamente, esta capa suele constituirla el navegador web, como es el caso en este proyecto, junto a React. React es una biblioteca de Javascript de código abierto, la cual tiene como objetivo facilitar el desarrollo de aplicaciones en una sola página.
- **Capa de aplicación**. Esta capa funciona a modo de intermediario entre la capa de presentación y la capa de datos. En ella está contenida toda la lógica de negocio y todas las funcionalidades que conforman el comportamiento de la aplicación. Su funcionalidad es recibir peticiones de la capa de presentación, procesarlas y devolver la información correspondiente a dicha solicitud. La capa de aplicación de este proyecto está constituida por una API basada en ASP.NET y hospedada en Azure. ASP.NET es un entorno para aplicaciones web, el cual fue desarrollado y distribuido por Microsoft, el cual es usado para, entre otras funciones, construir aplicaciones y servicios web. Por otro lado, Azure es una plataforma de computación en la nube creada también por Microsoft, utilizada para construir, probar, desplegar y administrar aplicaciones y servicios mediante el uso de sus centros de datos.
- **Capa de datos**. Esta capa es la responsable de almacenar y manejar los datos de la aplicación. Su funcionalidad es, como se ha mencionado, almacenar los datos que le son enviados desde la capa de aplicación, mantener la integridad de estos y proporcionarlos cuando la capa de aplicación los requiera. Esta capa esta compuesta por la base de datos, que en el caso de este proyecto se ha utilizado Microsoft SQL Server, el cual es un sistema de gestión de base de datos relacional de Microsoft.

En cuanto a la comunicación entre estas capas, la comunicación es establecida mediante un servicio REST. Un servicio REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles,

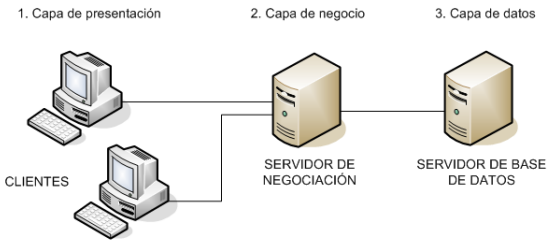

Figura 6.1: Esquema de un modelo de arquitectura de tres capas

en el caso de esta aplicación, en formato JSON [4]. Las principales características de un servicio REST son:

1. **Protocolo cliente/servidor sin estado**: cada petición HTTP contiene toda la información necesaria para ejecutarla, por lo que el cliente y el servidor no necesitan recordar ningún estado previo para satisfacer dicha petición.
2. **Interfaz uniforme**: los servicios REST tienen una interfaz estandarizada y uniforme, la cual consiste en métodos CRUD básicos de HTTP (GET, POST, PUT, DELETE), que usan para realizar operaciones en los distintos recursos, siempre y cuando estén identificados con una URI (Uniform Resoruce Identifier).
3. **Orientado a recursos**: los recursos representan entidades de datos con las que el cliente quiere interactuar. Estos recursos se identificas mediante el uso de URIs, y el servidor proporciona representaciones de estos recursos en diferentes formatos. En el caso de este proyecto, en formato JSON.
4. **Sistema de capas**: la arquitectura de un servicio REST puede contener diversas capas, como balanceadores de carga, proxies, incluso firewalls entre el cliente y el servidor. Cada capa puede aportar funcionalidad, escalabilidad o seguridad adicional.

6.2 Contexto tecnológico

Como ya se ha mencionado anteriormente, en la aplicación web se pueden distinguir dos partes principales: el Frontend y el Backend. La parte de Frontend constituye a la parte del cliente y se ha implementado haciendo uso de React. La parte del Backend constituye a la API encargada de conectar el cliente con la base de datos, la cual se ha implementado haciendo uso de ASP.NET Boilerplate.

6.2.1. Frontend

React es una de las librerías más populares de JavaScript para el desarrollo de aplicaciones móviles y web. Fue creada por Facebook en 2013 y su propósito es facilitar el desarrollo de aplicaciones en una sola página. Para hacer esto posible, React esta compuesto por componentes, reutilizables fragmentos de código JavaScript que se usan para crear interfaces de usuario. Esta librería hace uso de la extensión de sintaxis de JavaScrip JSX, la cual es usada para crear los elementos de React. Es empleada para incrustar código HTML en los objetos JavaScript. JSX también acepta expresiones válidas de JavaScript y funciones, por lo que puede simplificar estructuras de datos complejas [5].

React Bootstrap sustituye completamente al Bootstrap clásico. Bootstrap es uno de los *frameworks* Frontend más utilizados. Compuesto por HTML, CSS y JavaScript, su finalidad es servir como estructura de inicio en la producción de aplicaciones web. Entre sus

principales características podemos destacar que ofrece un diseño *responsive* al adaptar la interfaz de usuario a las limitaciones físicas de la pantalla en la que se muestra, así como su fácil uso apoyado por un soporte continuo y una amplia documentación donde encontrar proyectos de ejemplo. React Bootstrap se basa en esos principios para crear su *framework*. Los componentes que lo forman están construidos desde cero basándose en la hoja de estilos propia de Bootstrap, sin ninguna dependencia innecesaria. Su simple uso de componentes personalizados lo hacen una opción muy accesible desde el principio del desarrollo. Al ser una de las librerías más antiguas de React, este proyecto ha ido evolucionando junto a React por mucho tiempo, haciéndolo una de las mejores opciones en cuanto a construir las bases de una interfaz de usuario [6].

React Router es una librería de React la cual proporciona un conjunto de componentes navegacionales y otras utilidades que permiten manejar el enrutamiento o *routing*, proceso el cual determina que componentes se deben renderizar en base a la dirección actual, de las aplicaciones en React. Permite enlazar componentes a diferentes rutas de una manera sencilla y intuitiva, permitiendo crear una estructura de enrutamiento declarativa. Esto permite crear aplicaciones de una página (*single-page applications* o SPAs) con multiples páginas donde la direccion URL cambia dinámicamente sin tener que actualizar la página [7].

HTML5 + CSS3 + JavaScript. Estas son las tres tecnologías que predominan en el ámbito del desarrollo web. *HyperText Markup Lenguaje*, de su siglas HTML, es el componente más básico de la Web, en el cual se define el significado y la estructura del contenido estático o dinámico de esta. Este lenguaje se compone de etiquetas que definen los diferentes componentes de la web, ya sea texto, imágenes, listas, enlaces, etc.

Por otra parte, CSS (*Cascading Style Sheets*) es un lenguaje que permite definir el estilo de presentación de un documento estructurado mediante el uso de etiquetas, en este caso HTML. Su objetivo es presentar el contenido de la interfaz de manera atractiva y agradable a la vista del usuario. Esto lo consigue mediante la modificación de colores, fondos, márgenes, bordes, tipografías..., y cualquier otra cualidad de la interfaz.

Finalmente, JavaScript es un lenguaje de programación ligero, interpretado, o compilado *just-in-time* con funciones de primera clase, el cual es el lenguaje predominante en cuanto a lenguaje de *scripting* en desarrollo web. Se encarga de dar funcionalidad a los demás componentes de la web para obtener interactividad con esta.

PDF.JS es una biblioteca de JavaScript creada por Mozilla Corporation originalemten como una extensión del navegador Firefox, la cual es utilizada para transcribir y renderizar documentos PDF dentro de una página web.

6.2.2. Backend

ASP.NET Boilerplate es un *framework* el cual sigue los principios del *Domain-Driven Design*, de sus siglas *DDD*, el cual proporciona una forma estructurada de construir aplicaciones sostenibles, escalables y modulares en ASP.NET [8]. En cuanto a sus principales sus principales características caben destacar:

1. El uso de ***Domain-Driven Desing***, haciendo uso de entidades, repositorios, servicios de dominio, etc., lo cual ayuda a diseñar la lógica de negocio de una aplicación de forma estructurada.
2. La integración de **Entity Framework**, lo cual proporciona capa de acceso a datos robusta y flexible, además de simplificar la persistencia de datos, el uso de **querys** y la administración de transacciones.

3. La implementación de un sistema de **Autenticación y Autorización**, el cual proporciona una forma unificada de manejar la autenticación y autorización dentro de la aplicación.
4. La integración de otras librerías y *frameworks*, como **AutoMapper**, para realizar *mapping* de objeto-a-objeto.

En cuanto a **ASP.NET**, se trata de un *framework* desarrollado por Microsoft como parte de la plataforma **.NET**. Su función es ayudar a los desarrolladores a crear aplicaciones web y servicios dinámicos haciendo uso de las tecnologías de .NET. Proporciona un entorno para crear la parte de Backend de una aplicación web, manejar peticiones HTTP y la organización del estado de la aplicación [9].

Por lo que respecta a **.NET**, es un *framework* gratuito, de código abierto y multiplataforma desarrollado por Microsoft. Su finalidad es proporcionar una plataforma donde crear y ejecutar aplicaciones en diversos sistemas operativos y dispositivos [10]. Las principales características de este *framework* son las siguientes:

1. ***Common Language Runtime*** (CLR), es el entorno de ejecución de .NET, el cual es el en cargado de administrar la memoria, manejar las excepciones, realizar la *garbage collection* y proporcionar varios servicios necesarios para ejecutar las aplicaciones.
2. ***Base Class Libraries*** (BCL), una colección de clases reutilizables, tipos y APIs proporcionadas por .NET las cuales cubren un amplio rango de funcionalidades.
3. **Interoperabilidad del lenguaje**, ya que soporta diferentes lenguajes de programación, como **C#** Visual Basic .NET y F#, entre otros.
4. Desarrollo **Multi-Plataforma** (*Cross-Platform*), gracias a la variante de este *framework* .NET Core.
5. **Librerías y *frameworks*** proporcionadas por el *framework* que extienden sus capacidades, como ASP.NET para desarrollo web y Entity Framework para acceso a datos.

Como ya se ha mencionado anteriormente, estos *frameworks* integran consigo ***Entity Framework***, el cual es un *framework* ORM (*Object-Relational Mapping*) desarrollado por Microsoft, el cual proporciona un conjunto de herramientas y librerías para poder crear capas de datos de alto nivel en .NET sobre diferentes bases de datos. [11]

Por último, **Azure Blob Storage** es un sistema de almacenamiento de objetos en la nube, el cual está diseñado para el almacenamiento de cantidades masivas de datos no estructurados. Estos son datos que no se ciñen a ningún modelo de datos concreto, como texto o datos binarios. Su uso proporciona una manera simple y rápida de almacenar y recuperar dichos objetos [12]. Este sistema de almacenamiento ha sido utilizado para almacenar las imágenes y los documentos PDF de la aplicación.

6.3 Ejemplos de código

En este apartado se van a mostrar algunos ejemplos del código desarrollado para la implementación de la aplicación web. Esta fase del proyecto ha sido la más duradera, ya que se partía con la premisa de aprender React, por lo que el conocimiento sobre este era nulo, además de que ser la parte con más contenido a realizar.

Como ya se ha mencionado en el Capítulo 5, se realizaron los prototipos iniciales del aspecto de la interfaz de la aplicación a modo de poder previsualizar el aspecto que tendía y las funcionalidades que iba a implementar. En base a estos prototipos y tras un poco de investigación respecto al funcionamiento de React, se empezó a desarrollar un primer

diseño de la aplicación. Una vez desarrollado, se comenzó a desarrollar el modelo de base de datos, junto a la API y los *endpoints* necesarios para que el *frontend* pudiera acceder a los datos necesarios. Finalmente, se añadieron todas las funcionalidades definidas y se realizaron algunos cambios para mejorar la optimización.

Sin duda, el apartado con más cambios ha sido el almacenamiento de imágenes y documentos PDF, ya que en una primera instancia y sin mucho conocimiento al respecto, se almacenaban en la base de datos como cadenas de texto binario. Esto suponía varios problemas en cuanto a capacidad de almacenamiento, ya que estas cadenas de texto pueden llegar fácilmente a los millones de caracteres si se trataba de un documento medianamente extenso; y en cuanto al rendimiento, ya que enviar datos de tales magnitudes suponía un crecimiento exponencial en el tiempo de comunicación entre las dos partes de la aplicación. Por lo tanto, y después de buscar diferentes alternativas, se optó por el uso de Azure Blob Storage, que, como ya se mencionado anteriormente en el apartado 6.2, tiene como función almacenar, entre otros, objetos binarios en la nube. Así en la base de datos solo se guarda la dirección de cada objeto almacenado.

En primer lugar, se va a explicar brevemente la estructura de una aplicación en React. Como en la mayoría de aplicaciones web, su ventana principal es el archivo llamado *index.html*, compuesto por sus respectivas cabeceras y un único * * en el *body* con un identificador, como se puede ver en la Figura 6.2

```
<body>
  <div id="root"></div>
</body>
```

Figura 6.2: Elemento *div* con identificador *root*

Esta página tiene asociada un *script* el cual tiene la estructura del código de la Figura 6.3.

```
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Figura 6.3: Código de *index.js*

En este código se está creando un nuevo objeto raíz mediante el método *createRoot* de la librería *ReactDOM* sobre el elemento con identificador *root* que hemos visto anteriormente. A continuación, se llama al método *render* sobre el objeto creado, el cual inicia el proceso de renderizado de la aplicación. Dentro de este método se encuentra el componente *App*, el cual es el componente principal de la aplicación, del cual hablaremos a continuación, encapsulado en el componente *React.StrictMode*, el cual proporciona avisos en tiempo de ejecución para identificar potenciales problemas.

Una vez explicado esto, podemos pasar al componente principal de la aplicación, del cual se ha hablado anteriormente. Como se ha mencionado en el apartado anterior, el componente *App* es el componente principal de cualquier aplicación en React. En este caso, el componente implementa un sistema de *routing* proporcionado por la librería React Router, la cual se descrito anteriormente en el apartado 6.2, el cual permitirá renderizar diferentes componentes dependiendo en la dirección que nos encontremos.

```
function App() {
  const { token, setToken } = useToken();
  const { sessionToken, setSessionToken } = useSessionToken();
  const { userId, setUserId } = useUserId();
  const { sessionUserId, setSessionUserId } = useSessionUserId();
  const [selectedCategory, setSelectedCategory] = useState(null);
  const [searchTitle, setSearchTitle] = useState("");

  return (
    <BrowserRouter>
      <Routes>
        <Route
          path="/"
          element={
            <Layout
              token={token}
              sessionToken={sessionToken}
              userId={userId}
              sessionUserId={sessionUserId}
              setSelectedCategory={setSelectedCategory}
              setSearchTitle={setSearchTitle}
            />
          }
        >
          <Route
            index
            element={
              <Home
                selectedCategory={selectedCategory}
                searchTitle={searchTitle}
              />
            }
          />
          <Route path="upload" element={<Upload />} />
          <Route
            path="preview/:id"
            element={<Preview token={token} sessionToken={sessionToken} />}
          />
```

Figura 6.4: Código del componente *App* I

```
          <Route path="view/:id" element={<View />} />
          <Route
            path="login"
            element={
              <Login
                setToken={setToken}
                setSessionToken={setSessionToken}
                setUserId={setUserId}
                setSessionUserId={setSessionUserId}
              />
            }
          />
          <Route
            path="register"
            element={<Register setToken={setToken} setUserId={setUserId} />}
          />
          <Route
            path="profile"
            element={<Profile token={token} sessionToken={sessionToken} />}
          />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}
```

Figura 6.5: Código del componente *App* II

El código de las Figuras 6.4 y 6.5 representan el componente App. En este se definen diferentes variables globales mediante el uso de *hooks* para después pasarlos como parámetros a los diferentes componentes, de las cuales se hablarán más adelante. El componente devuelve la estructura típica de una arquitectura de enrutamiento de una aplicación en React. El componente *BrowserRouter* almacena la dirección actual en la barra de direcciones del navegador y maneja la navegación a través de la diferentes rutas que se establezcan haciendo uso del *history stack* del propio navegador. Dentro de este componente, esta encapsulado el componente *Routes*, en el cual se definen las diferentes rutas de la aplicación. Cada uno de los elementos *Route* corresponde a una dirección URL dentro de la aplicación, en la cual se renderizará el componente asignado a esa ruta. Caben destacar dos componentes:

1. El componente con *path=/"*, el cual renderizará el componente *Layout* siempre en cualquier lugar de la aplicación.
2. El componente con la etiqueta *index* indica que el componente *Home* será el componente renderizado en la página principal de la aplicación.

El siguiente componenente que se va a mostrar corresponde al componente *Layout*, el cual actúa como la cabecera o *Header* de la aplicación.

```
<Navbar bg="light" expand="lg" fixed="top">
  <Container fluid="xl">
    <Navbar.Brand
      as={Link}
      to="/"
      onClick={() => setSelectedCategory(null)}
    >
      <img src="../Book_icon.png" width="30" height="30" />
      Biblioteca
    </Navbar.Brand>
    <Navbar.Toggle aria-controls="basic-navbar-nav" />
    <Navbar.Collapse id="basic-navbar-nav">
      <Nav className="me-auto">
        <NavDropdown title="Categorias" id="basic-nav-dropdown">
          {categorias.map((categoria) => (
            <NavDropdown.Item
              key={categoria.id}
              onClick={() => setSelectedCategory(categoria)}
            >
              {categoria.titulo}
            </NavDropdown.Item>
          ))}
        </NavDropdown>
        <Form className="d-flex" onSubmit={() => setSearchTitle("")}>
          <Form.Control
            type="search"
            placeholder="Buscar"
            className="me-2"
            aria-label="Search"
            onChange={(e) => setSearchTitle(e.target.value)}
          />
        </Form>
      </Nav>
      {token || sessionToken ? <Loged userId={uID} /> : <NotLoged />}
    </Navbar.Collapse>
  </Container>
</Navbar>
```

Figura 6.6: Código del componente *Layout* I

Como podemos ver en la Figura 6.6, este componente hace uso del componente *Navbar*, proporcionado por React Bootstrap, para su estructura. Este está compuesto por varios

componentes: un icono de la aplicación junto al nombre de esta, el cual funciona a modo de ruta para volver al inicio de la aplicación al ser pulsado; un elemento desplegable, el cual muestra todas las categorías por las cuales puedes filtrar los documentos; un buscador mediante el cual puedes filtrar documentos por título; y un elemento que depende del estado de la aplicación, renderizará un componente u otro. En el elemento desplegable se puede observar como este es poblado con los elementos de una lista llamada *categorias*. Esta lista es el resultado de una llamada a la API, la cual devuelve un listado con todas las categorías existentes. En la Figura 6.7 se muestra el código de la función para obtener el listado de categorías.

```
const getCategorias = () => {
  fetch(
    "https://tfgproyectwebhost20230629150639.azurewebsites.net/api/services/app/Categorias/GetAll"
  )
    .then((response) => response.json())
    .then((data) => {
      setCategorias(data.result.items);
    })
    .catch((err) => {});
};
```

Figura 6.7: Código del componente *Layout* II

Esta función hace una petición GET al metodo *GetAll* del *endpoint* Categorias de la API. Este método es uno de los métodos proporcionados por la clase *AsyncCrudAppService*, la cual proporciona métodos para realizar las operaciones CRUD básicas, como se puede observar en la Figura 6.8. Como resultado se obtiene la lista de todas las categorías, la cual es almacenada en la variable çategorias".

```
public class CategoriasAppService : AsyncCrudAppService<Categoria, CategoriaDto>, ICategoriasAppService
```

Figura 6.8: Código de la clase *CategoriasAppService* de la API

En cuanto al elemento que depende del estado mencionado anteriormente, dependiendo de si el usuario ha iniciado sesión o no, estado el cual se comprueba mediante las variables *token* y *sessionToken*, hará aparecer el componente *Loged*, el cual mostrará un icono de usuario junto a un elemento desplegable mediante el cual el usuario podrá acceder a diferentes funcionalidades, o el componente *NotLoged*, el cual mostrará los botones para que el usuario pueda iniciar sesión o registrarse.

Al final del componente hay un componente proporcionado por la librería React Router llamado *Outlet*, el cual se utiliza para renderizar los componentes anteriormente asignados en las rutas en el componente *App*.

En cuanto al componente de la página principal, el cual podemos apreciar en la Figura 6.9, este se puede dividir en dos partes. La primera parte se compone de un componente desplegable el cual sirve para ordenar los documentos por antigüedad y valoración y por un texto el cual indica la categoría por la cual se está filtrando actualmente. La segunda parte se compone de un conjunto de componentes *Card* proporcionados por React Bootstrap, cada uno de los cuales representa un documento en concreto, mostrando algunas de las características de este. Si se pulsa en uno de estos elementos, el usuario podrá ver una previsualización del documento.

```
<Container fluid="xl" className="content bottom-padding">
  <ErrorAlert show={show} />
  <Row className="align-vertical">
    <Col className="align-vertical">{categoryName}</Col>
    <Col xl="auto">
      Ordenar por:
      <DropdownButton id="dropdown-basic-button" title={order}>
        <Dropdown.Item onClick={() => setOrder("Mas recientes")}>
          Mas recientes
        </Dropdown.Item>
        <Dropdown.Item onClick={() => setOrder("Mas antiguos")}>
          Mas antiguos
        </Dropdown.Item>
        <Dropdown.Item onClick={() => setOrder("Mejor valorados")}>
          Mejor valorados
        </Dropdown.Item>
      </DropdownButton>
    </Col>
  </Row>
  <Row className="justify-content-center">
    {filteredArchivosByTitle.map((archivo) => (
      <Card key={archivo.id} className="border col-div col-xl-4">
        <Link to={`/preview/${archivo.id}`}>
          <Card.Img
            variant="top"
            src={archivo.portada}
            fluid="xl"
            className="card-image"
          />
          <Card.Body>
            <Card.Title>{archivo.titulo}</Card.Title>
            <Card.Text>
              {archivo.descripcion.length < 30
                ? archivo.descripcion.slice(0, 50)
                : archivo.descripcion.slice(0, 50) + "..."}
            </Card.Text>
            <Card.Text>Valoración: {archivo.rating}</Card.Text>
          </Card.Body>
        </Link>
      </Card>
    ))}
  </Row>
</Container>
```

Figura 6.9: Código del componente *Home* I

La forma de obtener el listado de documentos es la misma que la de obtener categorías que se ha mostrado en la Figura 6.7, simplemente cambiando el contenido de la URL para acceder al *endpoint* correspondiente. Sin embargo, a diferencia del ejemplo anterior donde se utilizaba un método proporcionado por una clase heredada, el método utilizado en este caso es un método propio, como se puede ver en la Figura 6.10 el cual devuelve toda la información del documento exceptuando el documento PDF y las listas de usuarios que han valorado ese documento. Esto es debido a que en esta parte de la aplicación no se utiliza para nada esta información, por lo cual no se envía para mejorar la optimización, ya que estos datos pueden llegar a ser extensos y suponer una carga innecesaria en la comunicación.

```
public async Task<List<ArchivoDto>> GetAllBasicDataAsync()
{
    CheckGetAllPermission();

    using (var unitOfWork = UnitOfWorkManager.Begin())
    {
        var archivos = await _archivoRepository.GetAllIncluding(x => x.categorias).Select(a => new ArchivoDto
        {
            Id = a.Id,
            titulo = a.titulo,
            descripcion = a.descripcion,
            rating = a.rating,
            portada = a.portada,
            categorias = ObjectMapper.Map<ICollection<CategoriaDto>>(a.categorias)
        })
        .ToListAsync();

        await unitOfWork.CompleteAsync();
        return archivos;
    }
}
```

Figura 6.10: Código de la clase *GetAllBasicDataAsync* de la API

El método recupera todos los archivos del repositorio y crea para cada uno de ellos una nueva entidad DTO de archivo en el cual se incluyen todos los datos menos los relacionados con el documento PDF y las listas de usuarios lo hayan valorado, para finalmente almacenarlos en una lista y devolverlos. Las entidades DTO (*Data Transfer Object*) son contenedores de datos simples que se utilizan como modelo para recuperar o enviar datos de una entidad, evitando así la exposición de algunos datos.

Volviendo al componente *Home*, los documentos que se muestran en él pueden ser filtrados de tres formas: por categoría, por título y ordenados por antigüedad o valoración. Para filtrar por categoría o por título se le pasa por parámetros al componente la categoría o título seleccionados en el componente *Layout*, filtrando la lísta que se muestra. Para filtrar por antigüedad o valoración se hace uso del componente desplegable antes mencionado, mediante el cual ordenará ordenarán los elementos de la lista de componentes filtrados en función de la característica seleccionada. Para la antigüedad simplemente consta de revertir el orden de la lista, ya que ya vienen ordenados inicialmente, y para la valoración se hace uso de la propiedad *rating* de los propios documentos. En la Figura 6.11 se muestra el código de estos filtros. Todos ellos se ejecutan dentro del *hook* **UseEffect**, el cual está compuesto dos partes: la primera parte es el código que se va a ejecutar, mientras que la segunda parte es una lista de dependencias, mediante las cuales, si alguna de ellas cambia, se ejecutará el código de la primera parte. De esta manera, solo se ejecutarán cuando renderice el componente, ya q ue es cuando carga los archivos, y cada vez que la categoría seleccionada o el texto en el buscador cambien.

```
useEffect(() => {
  const filtered = selectedCategory
    ? archivos.filter((archivo) =>
        archivo.categorias.some(
          (categoria) => categoria.id === selectedCategory.id
        )
      )
    : archivos;
  setFilteredArchivosByCategory(filtered);
}, [selectedCategory, archivos]);

useEffect(() => {
  const filteredTitle = searchTitle
    ? filteredArchivosByCategory.filter((archivo) =>
        archivo.titulo.toLowerCase().includes(searchTitle.toLowerCase())
      )
    : filteredArchivosByCategory;
  setFilteredArchivosByTitle(filteredTitle);
}, [searchTitle, selectedCategory, filteredArchivosByCategory, archivos]);

useEffect(() => {
  let sortedArray = [...filteredArchivosByTitle];
  if (order === "Mas recientes") {
    sortedArray.sort((a, b) => b.id - a.id);
  } else if (order === "Mas antiguos") {
    sortedArray.sort((a, b) => a.id - b.id);
  } else if (order === "Mejor valorados") {
    sortedArray.sort((a, b) => b.rating - a.rating);
  }
  setFilteredArchivosByTitle(sortedArray);
}, [order]);
```

Figura 6.11: Código del componente *Home* II

Pasando al componente que controla el *Log in* de la aplicación, el cuerpo de este esta compuesto por un formulario en el cual hay dos cajas de texto donde introduce el nombre de usuario y la contraseña, una casilla para marcar si quieres que se mantenga sesión y un botón para enviar el formulario. Una vez se envía el formulario, se envía una petición POST al *endpoint* correspondiente de la API con los datos del usuario para realizar la autenticación. Si la autenticación falla, aparecerá un mensaje indicándole al usuario que sus credenciales son incorrectas. Por otra parte, si la autenticación es correcta, se obtendrá el usuario junto a un *token* de autenticación. En la Figura 6.12 se puede apreciar cómo, dependiendo si se ha marcado la casilla de recordar sesión o no, se guarda el identificador del usuario y el *token* en dos sitios diferentes. Aquí entra en juego un elemento que ya se ha visto en ejemplos anteriores de código: el uso de los *hooks* ***useToken, useUserid, useSessionToken y useSessionUserid***. Estos se tratan de unos *hooks* personalizados que se encargan de almacenar y extraer el *token* de autenticación y el identificador de usuario en el *localStorage* o *sessionStorage* del navegador. Estos dos se tratan de propiedades que permiten acceder al almacenamiento del navegador. Respectivamente, el primero te permite acceder al almacenamiento local, en el cual los datos se mantienen a través de diferentes instancias del navegador, mientras que el segundo te permite acceder al almacenamiento

de sesión, en el cual los datos se mantienen en esa instancia del navegador, por lo que una vez se cierre esos datos desaparecerán. En la Figura 6.13 se muestra el código de uno de estos *hooks*, concretamente el de *useToken*, el cual se encarga de almacenar y recuperar el *token* y el identificador del *localStorage*. Como se puede ver en el código, se hace uso del *hook useState*, el cual permite añadir variables de estado a los componentes [13]. El valor que acepta como parámetro es el valor inicial que tendrá la variable *token*, en este caso, el valor que este guardado en el *localStorage*, si existe. Por otro lado, a la función *setToken*, se le asigna la función *saveToken*, lo cual permite guardar el *token* en el almacenamiento. El funcionamiento de los demás *hooks* es el mismo, variando el contenido que guardan y el lugar donde lo hacen. Una vez almacenados los datos en sus correspondientes almacenes, la aplicación redirige al usuario a la ventana principal con su sesión iniciada.

```
fetch(
  "https://tfgproyectwebhost20230629150639.azurewebsites.net/api/TokenAuth/Authenticate",
  {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
      userNameOrEmailAddress: usuario,
      password: contraseña,
      rememberClient: false,
    }),
  }
)
  .then((response) => response.json())
  .then((data) => {
    if (data.success) {
      if (remember) {
        setToken(data.result.accessToken);
        setUserId(data.result.userId);
      } else {
        setSessionToken(data.result.accessToken);
        setSessionUserId(data.result.userId);
      }
      navigate("/");
      window.location.reload();
    } else {
      setAlert(true);
    }      You, 4 days ago • Added blob storage connection, some css & E.handle
  })
  .catch(() => {
    setShow(true);
  });
```

Figura 6.12: Código del componente *Login* I

```
export default function useToken() {
  const getToken = () => {
    const tokenString = localStorage.getItem("token");
    const userToken = JSON.parse(tokenString);
    return userToken;
  };

  const [token, setToken] = useState(getToken());

  const saveToken = (userToken) => {
    localStorage.setItem("token", JSON.stringify(userToken));
    setToken(userToken);
  };

  return {
    setToken: saveToken,
    token,
  };
}
```

Figura 6.13: Código del *hook useToken*

El componente *Register* funciona de forma similar al componente *Login*. Esta compuesto por un formulario en el cual el usuario debe introducir un nombre de usuario, un correo electrónico y una contraseña que debe introducir dos veces para verificar que es correcta. Al igual que en el *Log in*, se envía una petición POST al *enpoint* correspondiente para registrar al usuario con los datos introducidos. Si el registro es efectuado, la API devuelve los datos del usuario creado, los cuales son utilizados para hacer una petición de *Log in* temporal, así el usuario queda con sesión iniciada nada más registrarse.

Por lo que respecta al componente *Upload*, este tiene como función almacenar los documentos que los usuarios pretenden publicar en la aplicación. Esta compuesto por un formulario en el cual el usuario debe de especificar el nombre del documento, una descripción del mismo, seleccionar las categorías a las que pertenece, una portada, y el propio documento. La portada solo admite archivos de formato imagen de cualquier tipo, mientras que para el documento solo se permiten archivos PDF. Las categorías se obtienen del mismo modo que en el componente *Layout*. Al enviar el formulario, primero se guarda la portada y el archivo PDF mediante la función *uploadBlob* de la Figura 6.14. Esta función acepta como parámetros un el archivo que se quiere almacenar y el nombre del contenedor donde se alojará. Para evitar problemas con los nombres, el archivo se almacenará con un identificador único delante de su nombre. Una vez el archivo se haya sido almacenado con éxito, se recupera la dirección URL de este, la cual será almacenada en la base de datos para futuras referencias al objeto. De esta forma, no es necesario volver a realizar una conexión con el almacén de Blobs para obtener las portadas o los archivos PDF. A continuación, se hace una petición POST al *endpoint* correspondiente enviando el documento con los datos proporcionados por el usuario. Una vez completada la operación, la página redirige al usuario a la previsualización del documento que acaba de subir.

```
async function uploadBlob(file, containerName) {
  const containerClient = blobServiceClient.getContainerClient(containerName);
  const blobName = `${uuidv4()}-${file.name}`;
  const blockBlobClient = containerClient.getBlockBlobClient(blobName);

  await blockBlobClient.upload(file, file.size, {
    blobHTTPHeaders: { blobContentType: file.type },
  });

  const blobUrl = blockBlobClient.url;
  return blobUrl;
}
```

Figura 6.14: Código del componente *Upload* I

La previsualización de un documento está definida por el componente *preview*, en el cual se muestra toda la información sobre el documento, es decir, el título de este, su descripción completa, las categorías a las que pertenece, la portada y sus valoraciones, así como un botón para poder leer el documento. Si el usuario no ha iniciado sesión, al pulsar en el botón será redirigido a la página de iniciar sesión ya mencionada anteriormente. Por el caso contrario, al pulsar ahora el botón se mostrará el documento para que el usuario pueda leerlo. Además, al haber iniciado sesión, ahora aparecerá la opción de valorar, tanto positivamente como negativamente, el documento. El sistema de valoraciones funciona mediante una relación entre la entidad de los documentos y la entidad de los usuarios. Cada documento tiene una lista de usuarios que lo han valorado positivamente y otra de los que lo han hecho negativamente, al mismo tiempo que cada usuario tiene una

lista con los documentos que ha valorado positivamente y una para los que ha valorado negativamente. Esto establece una relación muchos-a-muchos entre las dos entidades. Dependiendo de si la valoración es positiva o negativa, se hará una petición PUT al *endpoint* correspondiente. En esta petición se añade la cabecera *Authorization*, la cual contiene el *token* de autenticación del usuario, como se muestra en la Figura 6.15. En la Figura 6.16 se muestra la lógica para manejar una valoración positiva, en la cual se obtiene el archivo y el usuario de sus respectivos repositorios. A continuación, se comprueba si el usuario se encuentra en la lista de valoraciones negativas. Si es así, significa que el usuario está cambiando su valoración de negativa a positiva, por lo que habrá que deshacer su anterior valoración, eliminándolo de la lista de valoraciones negativas, añadirlo a la lista de valoraciones positivas y aumentar la valoración del documento en dos. Por el contrario, si no se encontraba en la lista, significa que es la primera vez que valora el documento, por lo que se agrega al usuario a la lista de valoraciones positivas y se aumenta la valoración en uno. Una vez efectuado este cambio, se devuelve el documento con la información actualizada y se actualiza la interfaz para mostrar los cambios efectuados.

```
headers: {
  "Content-Type": "application/json",
  Authorization: `Bearer ${prevToken}`,
},
```

Figura 6.15: Código del componente *Preview* I

```
public async Task<ArchivoDto> UpdateUpvoteAsync(int archivoId, long userId)
{
    var archivo = await _archivoRepository.GetAllIncluding(x => x.usersDownvoted, x => x.usersUpvoted)
.FirstOrDefaultAsync(x => x.Id == archivoId);

    var userLoged = await _userRepository.FirstOrDefaultAsync(x => x.Id == userId);
    var userInDownvoted = archivo.usersDownvoted.FirstOrDefault(x => x.Id == userId);

    if(userInDownvoted != null) {
        archivo.usersDownvoted = archivo.usersDownvoted.Where(u => u.Id != userId).ToHashSet();
        archivo.usersUpvoted.Add(userLoged);
        archivo.rating = archivo.rating + 2;
    }
    else {
        archivo.usersUpvoted.Add(userLoged);
        archivo.rating++;
    }

    await CurrentUnitOfWork.SaveChangesAsync();

    var archivoDto = ObjectMapper.Map<ArchivoDto>(archivo);
    return archivoDto;
}
```

Figura 6.16: Código del método *UpdateUpvote* del endpoint Archivos

Por último, el componente *View* es el responsable de mostrar el contenido del documento. Primero se obtiene la información del documento haciendo una solicitud a la API al igual que se ha hecho en los componentes anteriores. Una vez obtenido, se hace uso de la librería PDF.js, la cual permite renderizar archivos PDF en un objecto *canvas* HTML. Para ello, la librería lee el documento mediante la URL proporcionada y lo prepara para ser renderizado. Una vez todo listo, se itera cada una de las páginas del documento para renderizarlas en un objeto *canvas*.

6.4 Despliegue

El despliegue de la aplicación se ha realizado en **Azure**, una plataforma de computación en la nube de Microsoft, la cual ofrece un amplio rango de herramientas y servicios para construir, desplegar y administrar apliaciones y sevicios. Al ser un servicio *cloud*, permite el *hosting* de aplicaciones sin necesidad de una infraestructura física [14]. Se ha elegido Azure para realizar el despliegue de la aplicación debido a la gran compatibilidad que tiene con otros productos ofrecidos por Microsoft, que en el caso de este proyecto son el *backend* de la aplicación, ya que esta hecho con ASP.NET, y la base de datos, que está hecha con Microsoft SQL Server; además de proporcionar herramientas y servicios que facilitan el despliegue de aplicaciones como la de este projecto.

La base de datos se desplegó en una base de datos en la nube mediante las herramientas proporcionadas por SQL Server Management Studio, creando una base de datos idéntica a la local en la nube.

Para el *backend* y *frontend* se ha hecho uso de dos **Azure App Service**, un *platform-as-a-service* que permite hospedar aplicaciones web y RESTful APIs, entre otros.

Para el *backend* se ha hecho uso de las herramientas incorporadas que ofrece Visual Studio. Estas te permiten indicar en el lugar donde vaya a ser desplegado, en este caso el *AppService* mencionado anteriormente. Finalmente queda configurar las dependencias del servicio, en este caso la base de datos, para lo cual se selecciona la base de datos que se ha creado en la nube anteriormente, además de configurar la cadena de conexión. Una vez configuradas las dependencias, el servicio esta listo para ser desplegado. En cuanto al despliegue del *frontend*, primero se debe ejecutar el comando *npm run build* para crear la versión de producción, la cual es almacenada en la carpeta *build*.

La parte *frontend* del proyecto ha sido desarrollada en Visual Studio Code, a diferencia de Visual Studio, no cuenta de primeras con las herramientas necesarias para realizar el despliegue. Para ello, se ha hecho uso de la extensión **Azure Account**, la cual permite conectar tu cuenta de Azure a VS Code. Una vez instalada y configurada con la cuenta de Azure correspondiente, ya se puede comenzar el despliegue, pero no sin antes realizar un pequeño ajuste. El *App Service* donde se va a desplegar el *frontend* hace uso del IIS (*Internet Information Services*, el cual por defecto interpreta las rutas de la aplicación como rutas dentro del servidor, por lo que si se intentase recargar la página o ocurriese alguna acción que redirigiera a otra página distinta de la principal, el servidor devolvería un error 404 al no encontrar nada en la ruta del servidor. Para hacer posible que sea la propia aplicación quien maneje las rutas y no el servidor, es necesario configurar el servidor, creando una regla *rewrite*. Para ello, se ha creado en la aplicación un archivo 'web.config', el cual el servidor leerá y reescribirá su configuración en base al contenido de este. En la Figura 6.17, se muestra la parte del código del 'web.config' donde se realiza el *rewrite*. La función de esta parte del código es crear una regla que, si la dirección solicitada no es ni un archivo, un directorio o un *endpoint* de una API dentro del servidor, la ruta debe de ser manejada por React Router.

```
<rewrite>
    <rules>
        <rule name="React Routes" stopProcessing="true">
            <match url=".*" />
            <conditions logicalGrouping="MatchAll">
                <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" />
                <add input="{REQUEST_FILENAME}" matchType="IsDirectory" negate="true" />
                <add input="{REQUEST_URI}" pattern="^/(api)" negate="true" />
            </conditions>
            <action type="Rewrite" url="/index.html" />
        </rule>
    </rules>
</rewrite>
```

Figura 6.17: Código del archivo web.config

Una vez realizada esta configuración, la aplicación ya está lista para el despliegue. Para ello, se ha desplegado la versión de producción de la carpeta *build* en el *AppService* creado para el *frontend* mencionado anteriormente.

CAPÍTULO 7
Producto desarrollado

Una vez realizado todo el trabajo de los apartados anteriores, la aplicación se ha consolidado en un estado en el que los usuarios pueden ya hacer uso de las funcionalidades mencionadas en anteriores ocasiones. Para mostrar el resultado de la aplicación web se van a mostrar a continuación, mediante una serie de capturas de pantalla explicadas, las diferentes funcionalidades que la aplicación proporciona, las cuales corresponden además a los escenarios descritos en el Apartado 4.2.

Lo primero con lo que el usuario se encuentra al abrir la página por primera vez es la ventana principal que se muestra en la Figura 7.1. En ella se presentan todos los documentos disponibles actualmente en la biblioteca.

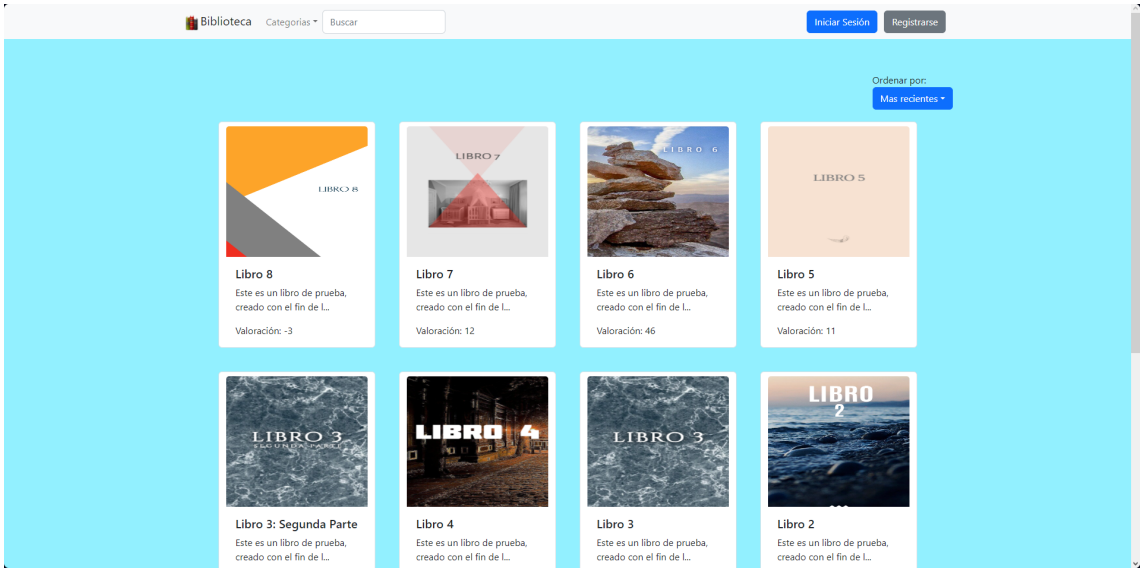

Figura 7.1: Página principal de la aplicación web

En esta ventana el usuario puede filtrar tanto por categorías o por título, como se mostrará más adelante, como también puede ordenar los documentos por antigüedad o por valoración. Por defecto, los documentos se muestran ordenados para que se muestren primero los más recientes, como se podía ver en la Figura 7.1. En la Figura 7.2 se muestra el contenido de la ventana principal una vez el usuario ha decidido ordenarlos por valoración.

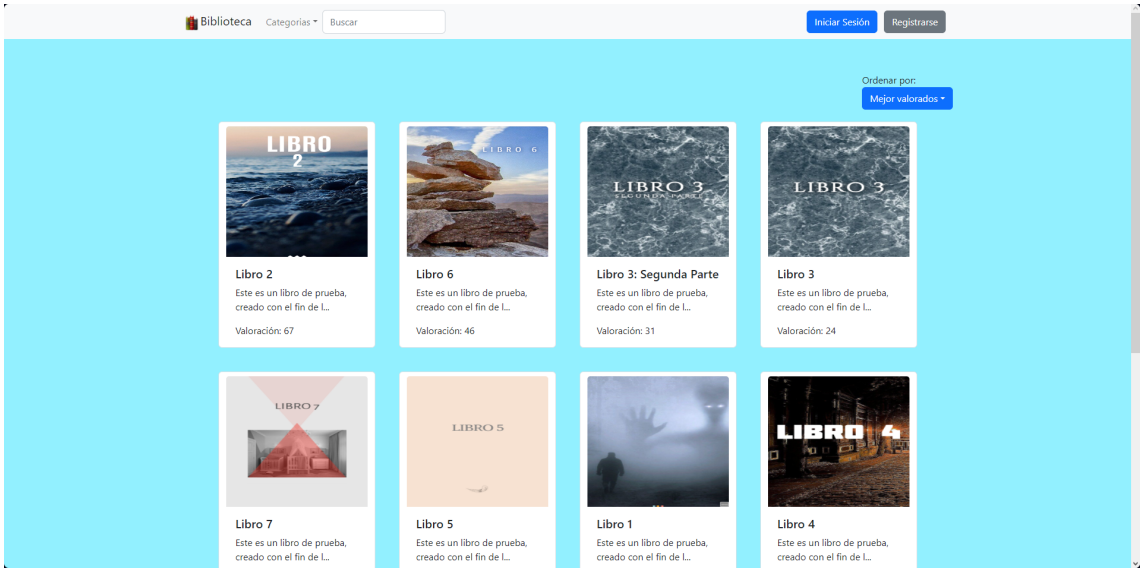

Figura 7.2: Página principal ordenada por 'Mejor Valorados'

Si el usuario pulsa en cualquiera de los documentos se le mostrará una ventana de previsualización, como se muestra en la Figura 7.3. En ella se muestran las características del documento, como su título, la descripción completa, las categorías a las que pertenece, el usuario por el cual ha sido subido a la web y su valoración, así como una mejor representación de la portada de este.

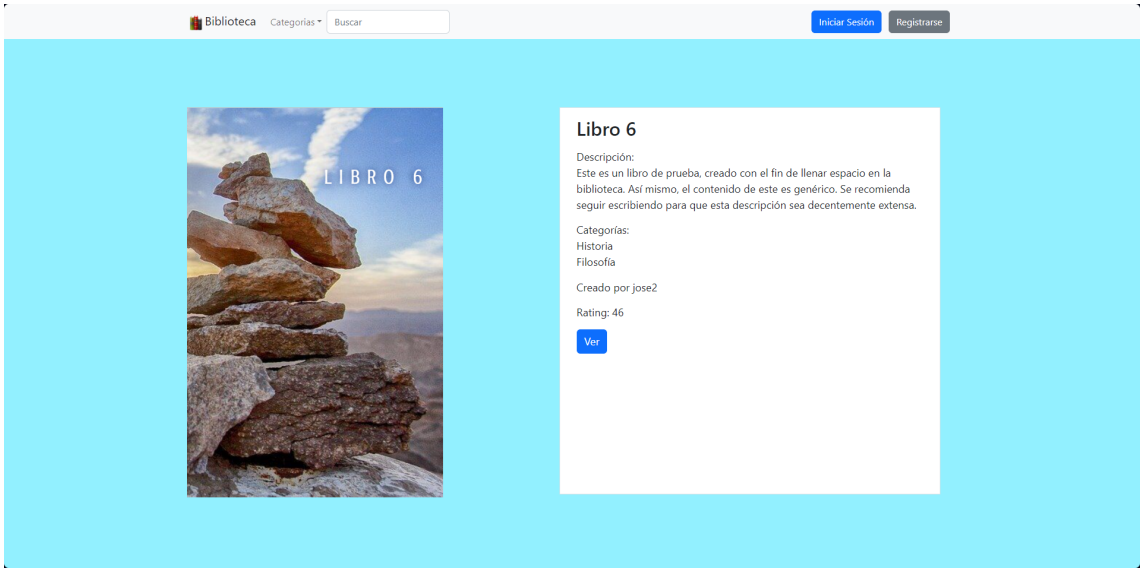

Figura 7.3: Vista de la página de previsualización de un documento

Si el usuario pulsa en el botón 'Ver' para visualizar el contenido del documento, al no haber iniciado sesión, la página le redirigirá a la ventana de iniciar sesión, que se muestra en la Figura 7.4. Aquí el usuario deberá introducir su nombre de usuario y contraseña para poder iniciar sesión. Así mismo, el usuario puede pulsar en la casilla 'Recuerdame' si desea que su sesión se mantenga tras cerrar el navegador. Si el usuario no está registrado en la página web, deberá de pulsar el botón de registrarse que aparece en la barra de navegación, lo que le mostrará la ventana de registro de usuario que se muestra en la Figura 7.5. Aquí el usuario deberá introducir un nombre de usuario, un correo y una contraseña, que por seguridad debe deberá de introducir dos veces, para registrarse en

la web. Tras introducir los datos y la validación de estos, se le iniciará sesión automáticamente. Cabe destacar que el usuario puede acceder en cualquier momento a la ventana de iniciar sesión o de registro desde los botones que aparecen arriba a la derecha en la barra de navegación.

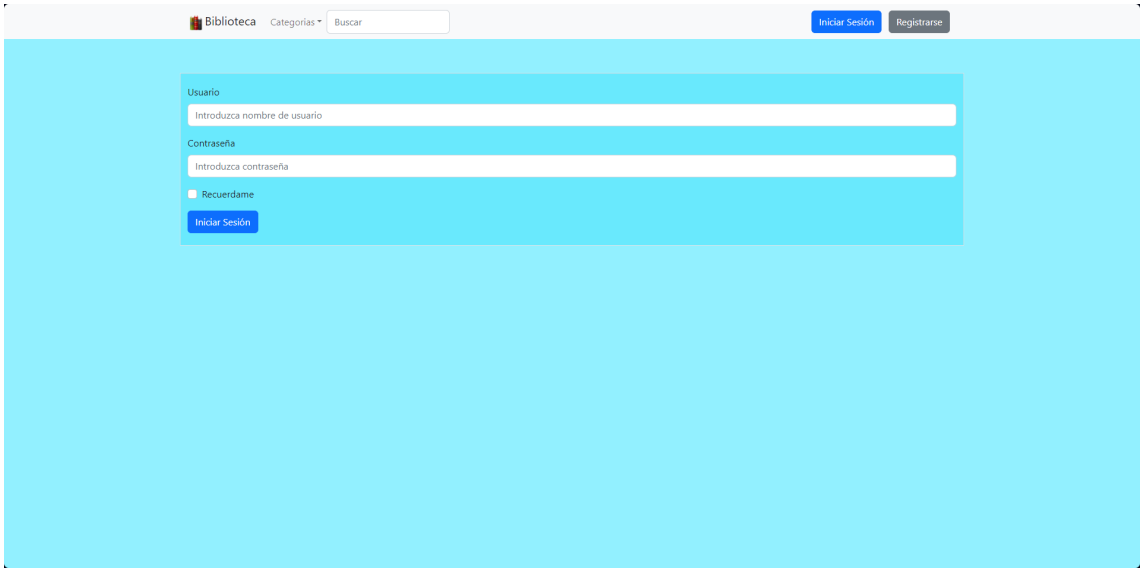

Figura 7.4: Vista de la página de iniciar sesión

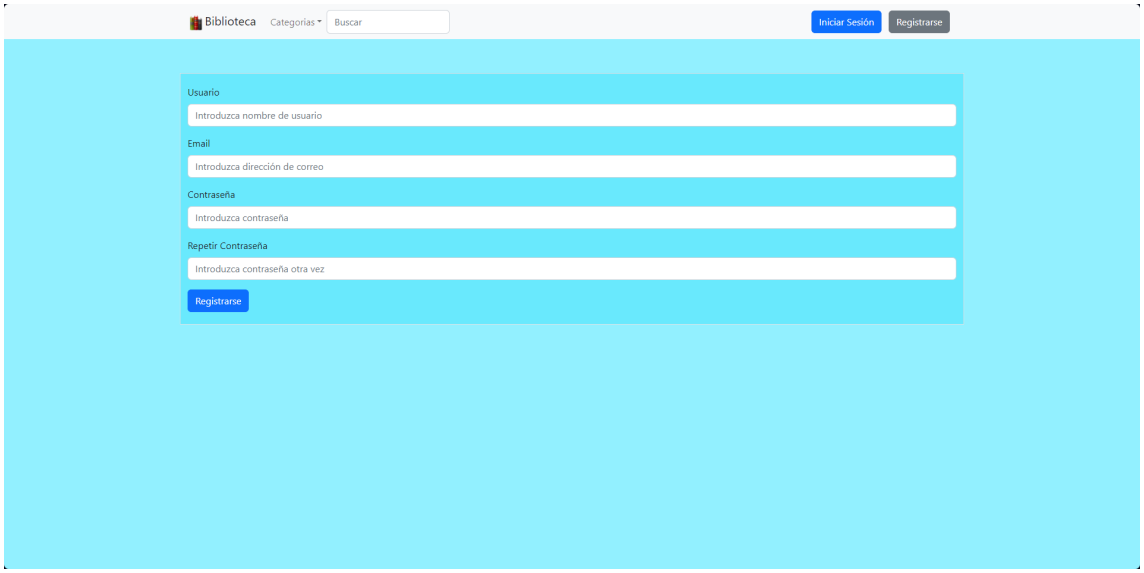

Figura 7.5: Vista de la página de registro de usuario

Una vez iniciada sesión, tanto si lo ha hecho mediante el inicio de sesión o el inicio de sesión automático tras un registro de usuario exitoso, el usuario será redirigido a la página principal con su sesión iniciada. Como se puede ver en la Figura 7.6, el contenido que se muestra es idéntico al de la Figura 7.1, excepto por que ahora, en vez de aparecer los botones de iniciar sesión o registrarse aparece un icono de usuario, indicando que la sesión ha sido realizada con éxito.

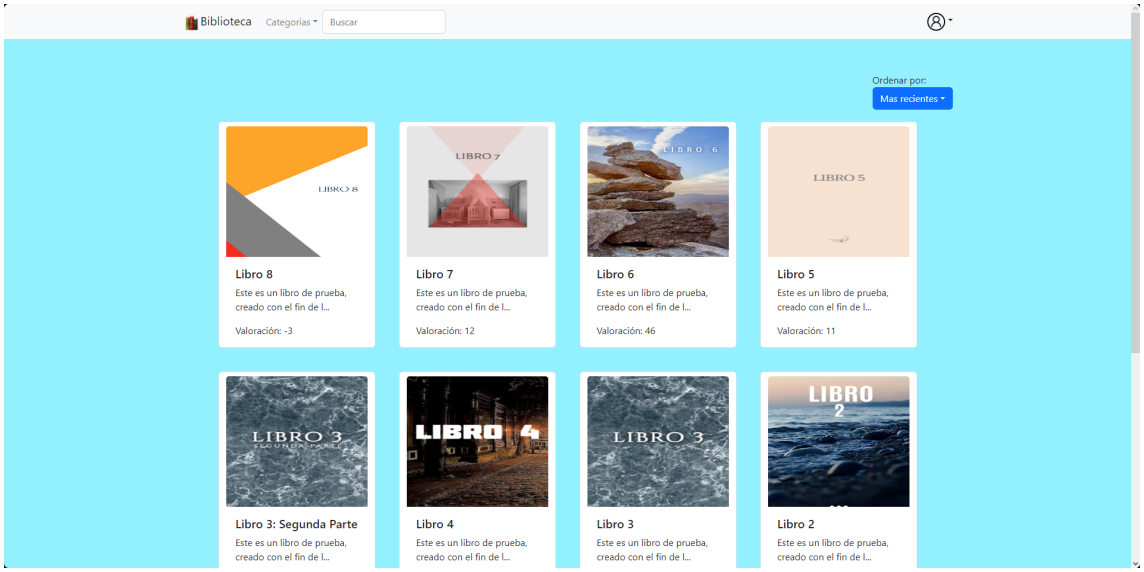

Figura 7.6: Vista de la página principal tras iniciar sesión

Como ya se ha mencionado anteriormente, en esta página el usuario puede filtrar los documentos existentes por categorías o por título, así como puede ordenarlos por antigüedad o valoraciones, como se ha mostrado anteriormente en la Figura 7.2. Si el usuario desea filtrar los documentos por categorías, debe pulsar en el menú desplegable 'Categorias' que aparece en la barra de navegación. Al hacerlo, el menú se abrirá, mostrando todas las categorías, como se muestra en la Figura 7.7. Si el usuario pulsa en una de las categorías, la web solo mostrará los documentos que pertenecen a esa categoría, como se muestra en la Figura 7.8. Si se desea de buscar por otra categoría, el proceso que el usuario debe realizar es el mismo. Si por otro lado quiere retirar los filtros por categoría y volver a ver todos los documentos, el usuario debe de pulsar en el icono de la web.

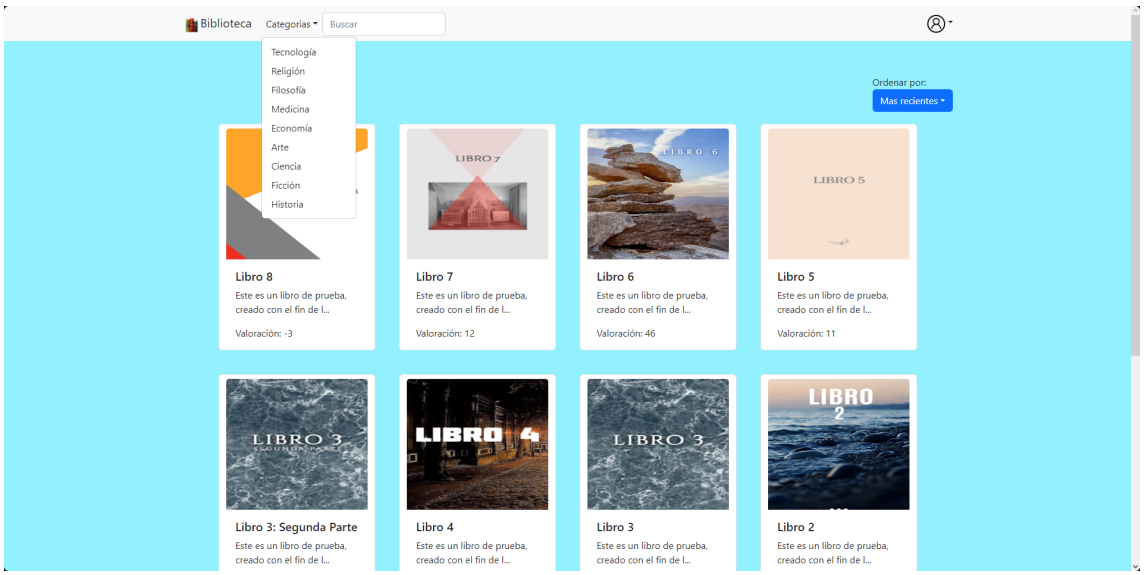

Figura 7.7: Vista del menú desplegable de categorías

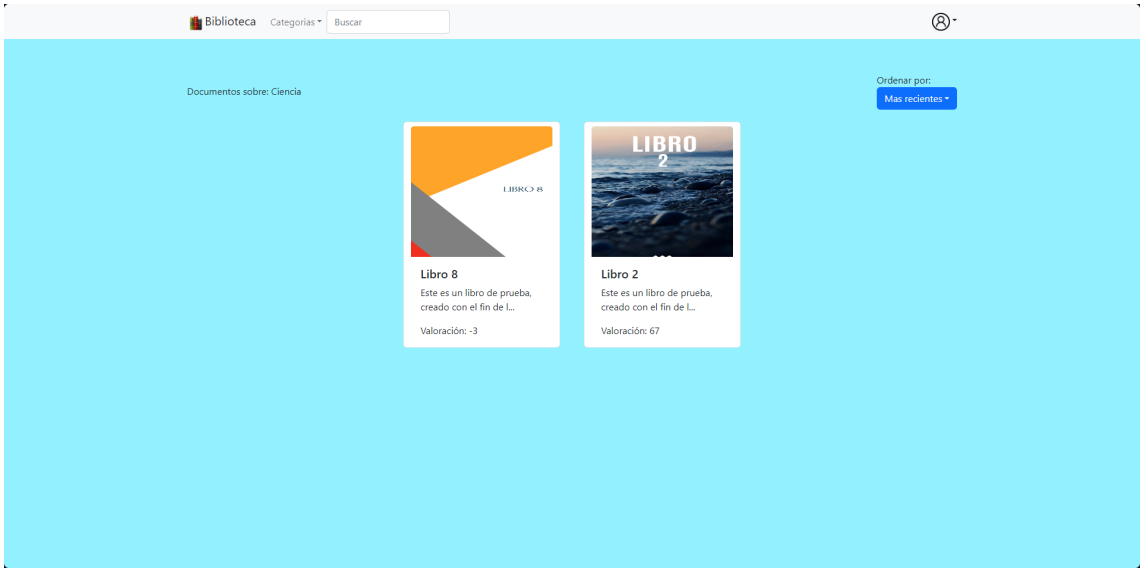

Figura 7.8: Vista de la página principal tras filtrar por la categoría 'Ciencia'

Si además el usuario quiere filtrar los documentos por título, debe de introducir el título por el que quiere filtrar en el buscador en la barra de navegación, como se muestra en la Figura 7.10. Cabe destacar que cada una de estas opciones de filtrado pueden ser utilizadas independientemente de los demás filtros, o pueden utilizarse varios a la vez, como se muestra en la Figura 7.10.

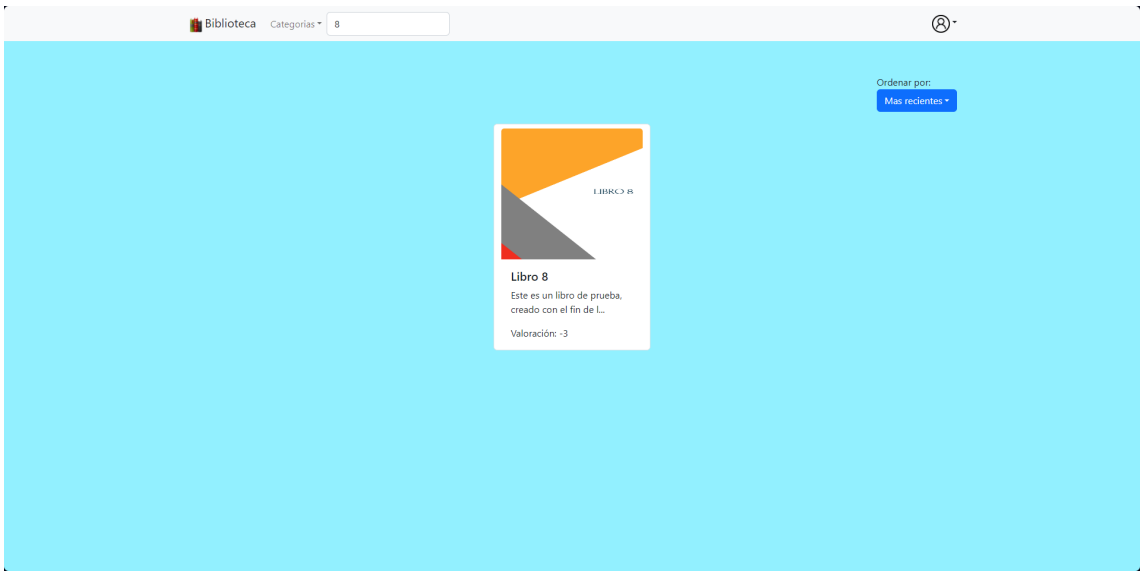

Figura 7.9: Vista de la página principal tras filtrar por título

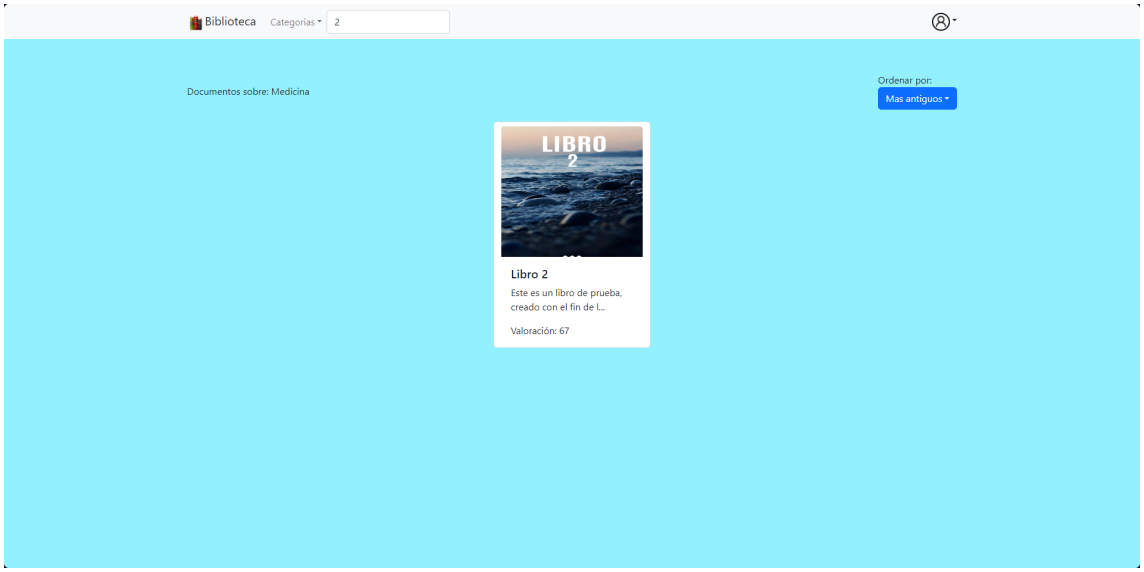

Figura 7.10: Vista de la página principal tras utilizar varias opciones de filtrado

Si a continuación el usuario pulsa en uno de los documentos, se le mostrará nuevamente una previsualización de este como se mostraba en la Figura 7.3. Sin embargo, ahora hay una pequeña diferencia, y es que tras haberse iniciado sesión, ahora aparecen además dos selectores llamados '*Upvote*' y '*Downvote*', como se muestra en la Figura 7.11, los cuales sirven para valorar el documento.

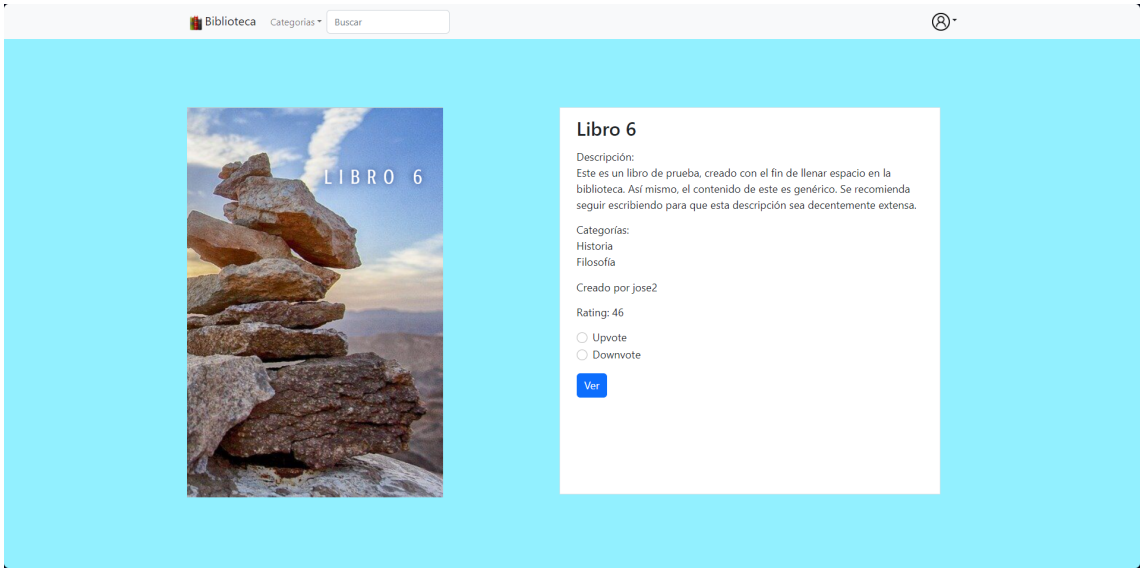

Figura 7.11: Vista de la página de previsualización de un documento tras iniciar sesión

Si ahora el usuario decide valorar positivamente el documento, debe seleccionar la opción '*Upvote*'. Al hacerlo, el usuario verá en tiempo real como la valoración del documento ha aumentado, además de ver como la opción que ha seleccionado ahora ha quedado marcada, como se puede ver en la Figura 7.12. Si el usuario decide cambiar de opinión y valorarlo negativamente, este debe seleccionar la opción '*Downvote*'. Esta vez el usuario verá como la valoración del documento disminuye respecto del valor que esta tenía inicialmente antes de valorarla anteriormente, además de ver como la opción que había seleccionado anteriormente ha quedado desmarcada y ahora la que está marcada es la que acaba de seleccionar, como se muestra en la Figura 7.13.

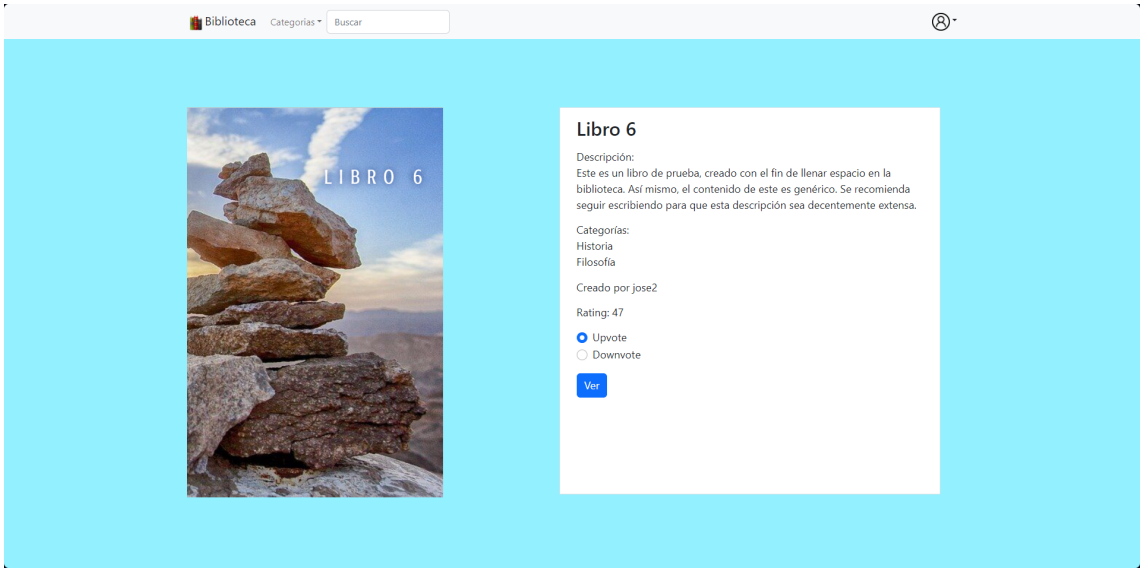

Figura 7.12: Vista de la página de previsualización tras una valoración positiva del usuario

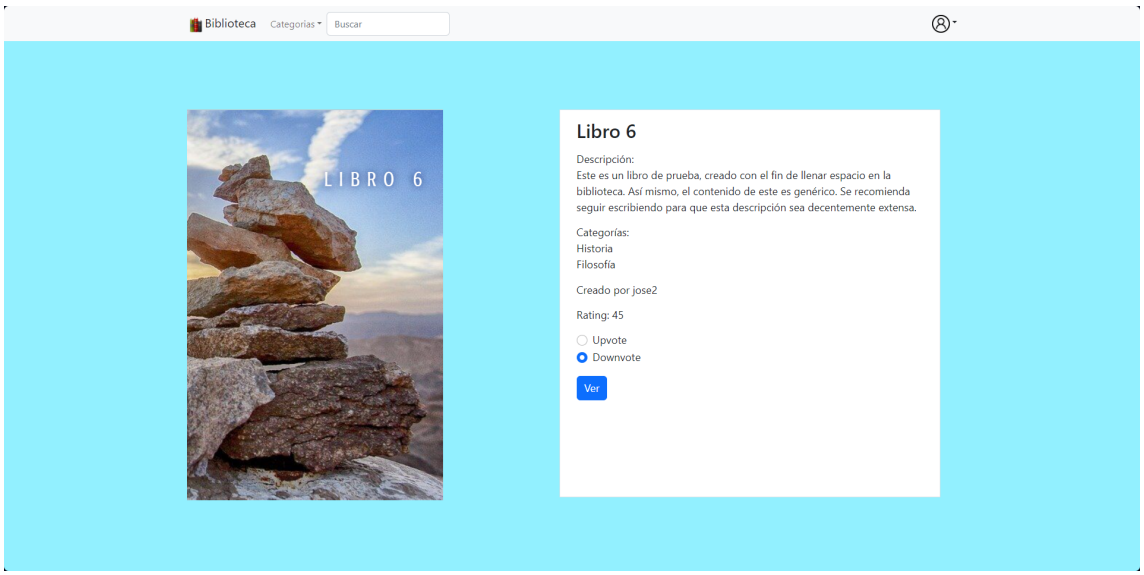

Figura 7.13: Vista de la página de previsualización tras una valoración negativa del usuario

Al haber realizado el inicio de sesión, si el usuario ahora pulsa en el botón 'Ver' para visualizar el documento, esta vez sí que se le mostrará dicho documento, pudiendo ahora si acceder y disfrutar de su contenido, como se muestra en la Figura 7.14. Si el usuario desea volver a la página principal, puede hacerlo pulsando en el icono de la web.

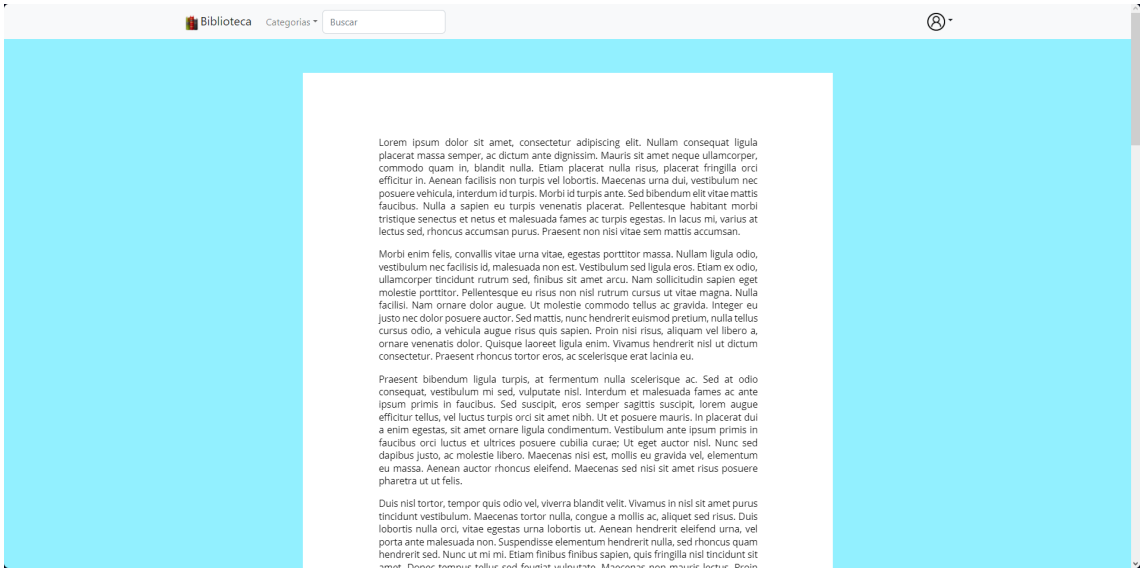

Figura 7.14: Vista de la página del contenido del documento

Para subir un documento a la web, el usuario, una vez iniciada sesión, debe pulsar en el icono de arriba a la derecha en la barra de navegación, lo cual desplegará un menú con diferentes opciones, siendo una de estas la opción 'Subir Archivo', como se puede observar en la Figura 7.15.

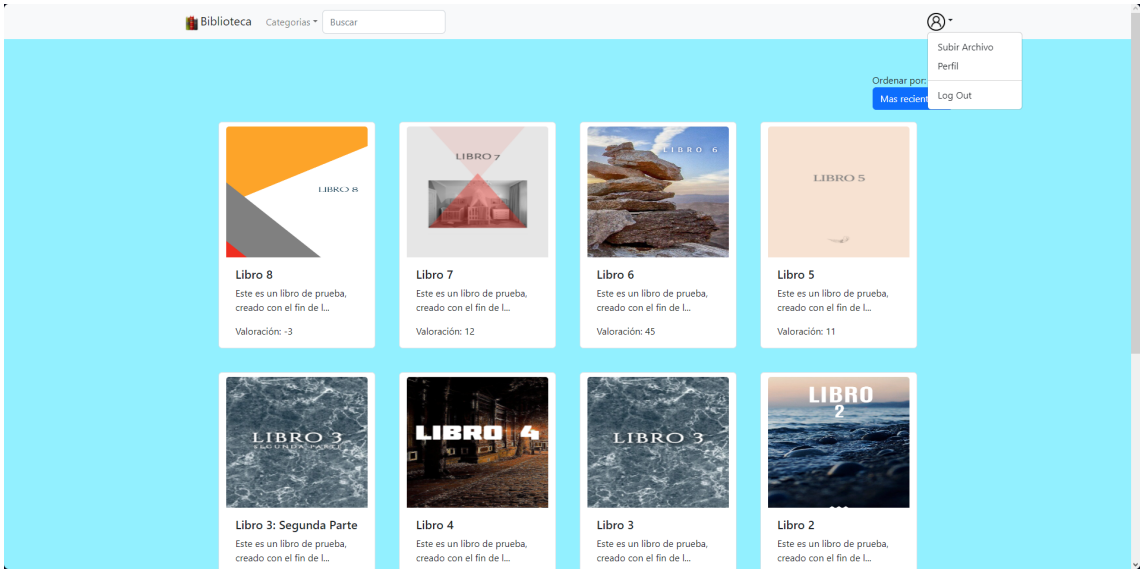

Figura 7.15: Vista del menú desplegable de opciones de usuario

Una vez que el usuario pulse la opción 'Subir Archivo', se le mostrará una página con información a rellenar sobre el documento, como se muestra en la Figura 7.16. El usuario debe proporcionar un título, una descripción, seleccionar las categorías a las que el documento pertenece, una portada y el propio documento. Cabe destacar que el usuario puede proporcionar cualquier formato de imagen para la portada. Sin embargo, para el documento solo puede proporcionar un archivo en formato PDF.

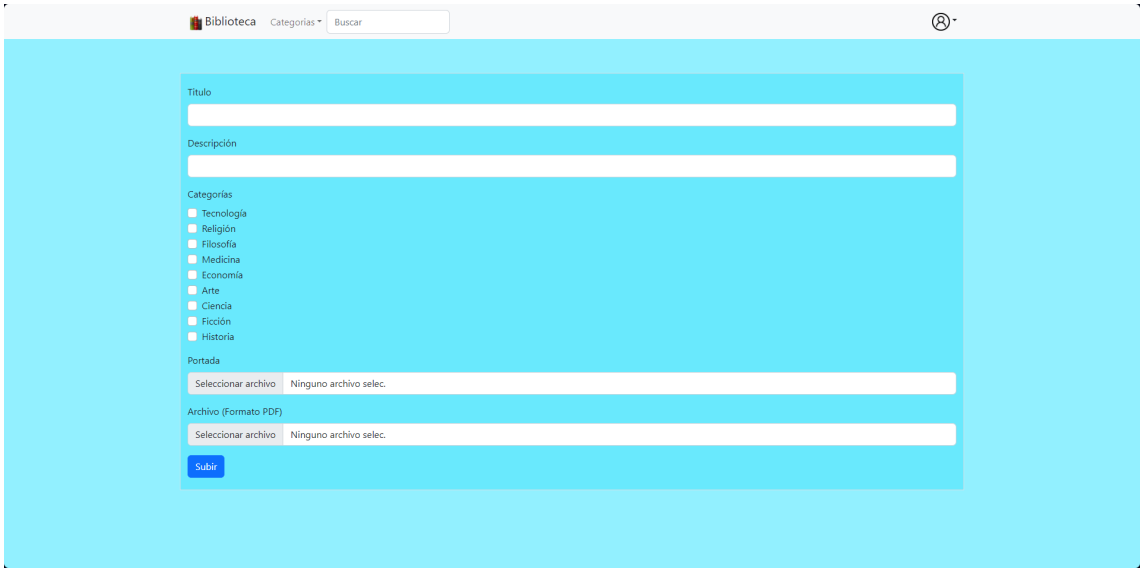

Figura 7.16: Vista de la página de subir archivos

Tras rellenar todos los campos con la información pertinente del documento, el usuario debe pulsar en el botón 'Subir' para subir el documento. Tras unos instantes de espera, dependiendo del tamaño del documento, se redirigirá al usuario a una previsualización del mismo, como la que se mostraba en la Figura 7.3, indicando así que el documento ha sido subido con éxito.

Por otro lado, si el usuario accede a la opción del menú desplegable que se mostraba en la Figura 7.15 'Perfil', este accederá a su perfil de usuario. Este apartado de la aplicación todavía no se ha finalizado del todo, por lo en esta versión cuenta con unas pocas funcionalidades limitadas. De momento, en esta sección al usuario se le mostrará su nombre de usuario y su correo electrónico, así pues como los documentos subidos por él a la plataforma, como se puede ver en la Figura 7.17. Si el usuario pulsa en uno de sus documentos, se le mostrará la previsualización de este mismo, como ya se ha mostrado anteriormente.

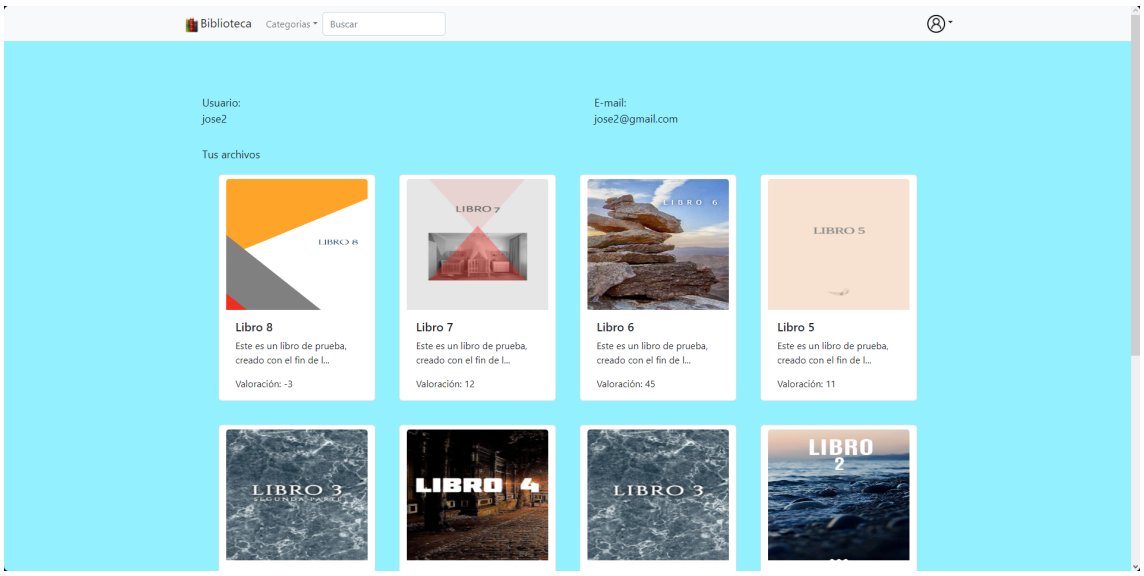

Figura 7.17: Vista de la página de perfil de usuario

Por último, si en el menú desplegable el usuario pulsa en la opción '*Log Out*', el usuario cerrará sesión y la aplicación quedará en el mismo estado que se mostraba en la Figura 7.1.

CAPÍTULO 8
Validación

Tras concluir las fases de diseño, desarrollo e implementación de la aplicación, se ha realizado una prueba de usabilidad con usuarios reales. Este método de validación se centra en observar y analizar el comportamiento de un usuario real haciendo uso de la página web, tras lo cual se obtendrán conclusiones sobre el funcionamiento de la aplicación.

El procedimiento que se ha llevado a cabo consiste en asignar un conjunto de tareas, las cuales abarquen las diferentes funcionalidades de la aplicación, que usuario debe de realizar. Tras la finalización de estas tareas, el usuario deberá de rellenar un formulario de valoración de la web, en el que indicarán como ha sido su experiencia al realizar las diferentes tareas en la web, así como dando su opinión sobre esta mediante el uso de comentarios.

Las tareas que un usuario debe realizar dentro de la aplicación son las siguientes:

1. Filtrar documentos por categoría y/o título
2. Ordenar documentos por antigüedad o valoración
3. Iniciar sesión o registrarse en la web
4. Leer algunos documentos
5. Valorar algunos documentos
6. Cambiar algunas de tus valoraciones
7. Subir algún documento
8. Cerrar sesión

Tras la realización de las tareas propuestas, el usuario realizó un cuestionario, el cual, como ya se ha mencionado, está centrado en la usabilidad, además de contar con la posibilidad de proporcionar su opinión acerca de la aplicación. A continuación, se presenta dicho cuestionario.

Como ya se ha mencionado, el cuestionario intenta realizar una evaluación de la usabilidad de aplicación para el usuario. Las diferentes preguntas que forman este cuestionario se pueden encontrar en el Apéndice B. La encuesta ha sido realizada por 7 usuarios con diversos perfiles, algunos de ellos con mayores conocimientos y experiencia sobre aplicaciones web, pero en general todos presentan conocimientos y familiaridad en ámbitos tecnológicos referentes a aplicaciones de este tipo. En cuanto a los resultados de la prueba, las siete primeras cuestiones, las cuales tienen como finalidad comprobar si el usuario

ha sido capaz de realizar las tareas designadas, han sido realizadas por todos los usuarios, salvo la tarea de leer los documentos de la aplicación, la cual un pequeño porcentaje de usuarios no han podido completar, como se muestra en la Figura 8.1.

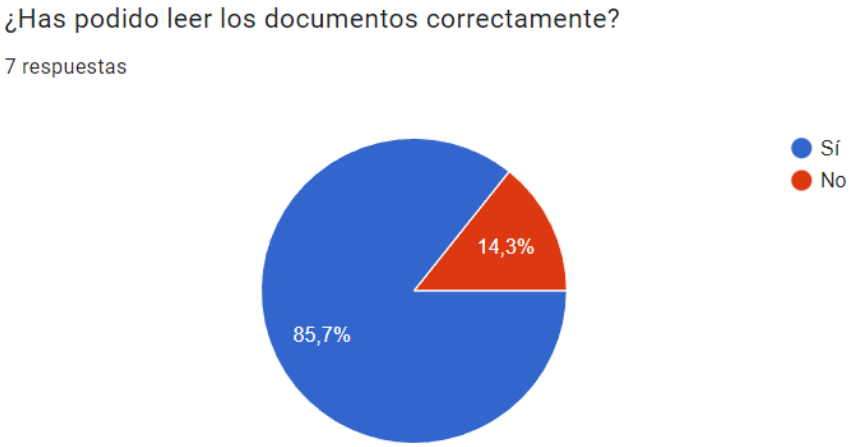

Figura 8.1: Encuesta de usuario I

Tras la comunicación con los usuarios y gracias a los detalles proporcionados sobre esta imposibilidad de realizar la tarea, se ha llegado al problema. El problema reside en que algunos navegadores interpretan de forma diferente algunas restricciones de los diferentes objetos HTML, por lo que sí que es cierto que el documento que subes a la aplicación está restringido a que solo permita seleccionar archivos PDF, algunos navegadores permiten la opción de subir todo tipo de archivos. Esto imposibilita la tarea de leer algunos documentos, ya que la aplicación no podrá renderizar archivos cuya extensión no sea PDF. Además, como la aplicación no comprueba una vez seleccionado el documento su extensión, no hay problemas con la subida de archivos, por lo que tiene sentido que no haya habido problemas con la subida de documentos.

Por otro lado, las cuestiones restantes tienen como finalidad averiguar el grado de conformidad de los usuarios respecto a la realización de las tareas y el uso de la aplicación en general. Las Figuras 8.2, 8.3 y 8.4 muestran los resultados de estas cuestiones.

Figura 8.2: Encuesta de usuario II

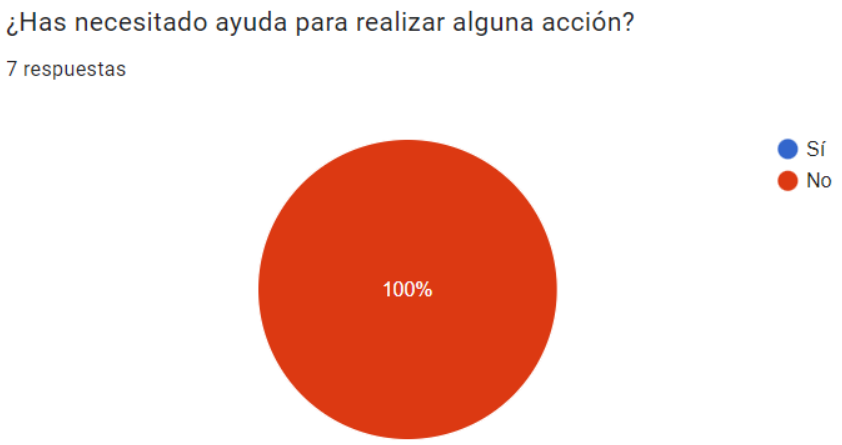

Figura 8.3: Encuesta de usuario III

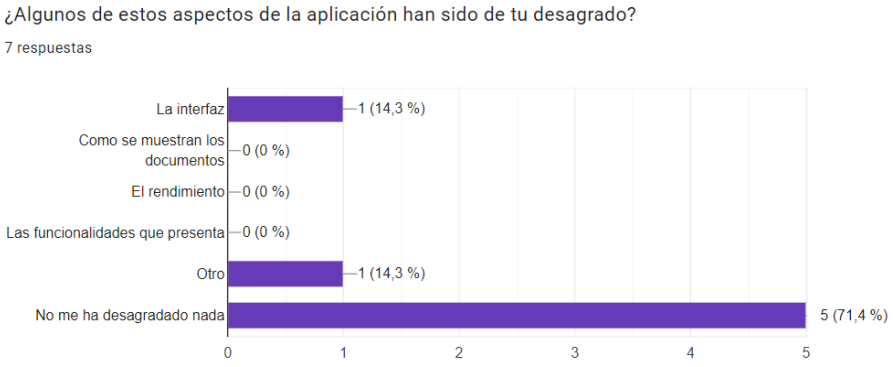

Figura 8.4: Encuesta de usuario IV

Los resultados de estas cuestiones indican una eficacia de la aplicación mayormente positiva, aunque hay que tener en cuenta que un grupo muy reducido de usuarios se ha encontrado con dificultades a la hora de navegar por la aplicación, además de que algunos han tenido malas experiencias con la interfaz de la aplicación. Además, los usuarios que seleccionaran la opción 'Otro' en la última cuestión tenían la posibilidad de comentar cuál es ese otro elemento el cual no ha sido de su agrado. Estas respuestas se podrían categorizar como casos más específicos de la interfaz, por lo que se puede concluir que la totalidad de las quejas son hacia la interfaz, por lo que habría que realizar algunos cambios para no dejar de lado a ningún usuario.

CAPÍTULO 9
Relación del trabajo desarrollado con los estudios cursados

Durante el Grado de Ingeniería Informática se han cursado diferentes asignaturas cuyos conocimientos impartidos han servido para el desarrollo de este trabajo, tanto directa como indirectamente. Dichas asignaturas se muestran a continuación:

- **Introducción a la Informática y a la Programación y Programación**. Estas dos asignaturas son las encargadas de formar las bases de los conocimientos sobre programación, gracias a los cuales ha sido posible continuar expandiendo los conocimientos respecto a los diferentes lenguajes utilizados.
- **Desarrollo Web**. En esta asignatura se presentaron los conceptos de aplicación web y la estructura que estas siguen. Estos conocimientos han permitido un correcto desarrollo en la arquitectura de la aplicación de este proyecto en cuanto a sus diferentes partes y las comunicaciones entre ellas.
- **Ingeniería del Software**. Gracias a esta asignatura, en la cual se presentaron métodos, técnicas y herramientas actuales para desarrollar *software* de calidad, el proceso de desarrollo del trabajo ha resultado mucho más sencillo al haber realizado un proyecto de *software* completo pasando por todas sus diferentes fases.
- **Bases de Datos y Sistemas de Información y Tecnología de Bases de Datos**. En estas asignaturas se adquieren los conocimientos sobre gestión y administración de bases de datos, lo que ha servido en este proyecto para la creación, la administración y el mantenimiento de la base de datos donde se almacenan los datos de las diferentes entidades que componen la aplicación (usuarios, documentos, etc).
- **Integración de Aplicaciónes**. En esta asignatura se estudia el uso de tecnologías que facilitan la comunicación e integración de sistemas informaticos, asi como arquitecturas orientadas a servicios y técnicas y herramientas para el intercambio de datos. Estos conocimientos han sido de gran utilidad a la hora del desarrollo e implementación de la API de la aplicación y la comunicación con esta.

CAPÍTULO 10
Conclusiones y trabajos futuros

Este proyecto ha comenzado con el análisis de diferentes plataformas con funcionalidades similares a las propuestas de la aplicación, detectando de que todas incluyen por lo menos alguna de estas características, pero ninguna incluye todas las especificadas.

El proyecto se ha llevado a cabo siguiendo la metodología de modelo incremental, la cual ha permitido dividir el proyecto en pequeñas etapas, en las cuales se podido observar y analizar el estado actual del trabajo, permitiendo así una progresión gradual eficiente debido a resultados de los *sprints* anteriores.

A continuación, se han extraído los requisitos iniciales de la aplicación en base a un caso de estudio, el cual ha permitido condensar todas las ideas planteadas para la aplicación. En base a estos requisitos iniciales, se han especificado los casos de uso, permitiendo un análisis más preciso de las posibles interacciones del usuario con la aplicación y las acciones que este podrá llevar a cabo. Más adelante, se han especificado las diferentes clases de la aplicación mediante un diagrama de clases utilizando el lenguaje UML, brindando un primer vistazo las propiedades con las que estas contarán y las diferentes relaciones que se darán acabo. En base a las clases especificadas anteriormente, se ha definido el modelo de Base de Datos, poniendo en práctica las clases definidas, sus propiedades y sus distintas relaciones.

Por otro lado, se han diseñado los prototipos de la página web en base a los requisitos especificados anteriormente, permitiendo contemplar visualmente una primera idea de la estructura y el contenido la interfaz de la aplicación. Gracias a esta parte del diseño, el desarrollo de la interfaz de la aplicación ha sido mucho más sencillo, debido a que al partir de estos bocetos los objetivos a alcanzar eran mucho más claros.

En cuanto al desarrollo de la aplicación, se han hecho uso de diferentes herramientas y *frameworks*, tanto para la parte *frontend* como *backend*. Esta parte del trabajo ha sido la más costosa, por el mismo motivo que también es la principal motivación de este proyecto: el aprendizaje de nuevas herramientas, tecnologías y lenguajes. Gracias a este trabajo he tenido la oportunidad de aprender nuevos lenguajes y *frameworks*, como React y .NET, y hacer uso de tecnologías que nunca había utilizado, como las tecnologías utilizadas en la nube de Azure, así también como expandir los conocimientos existentes de lenguajes ya conocidos, como Javascript.

Finalmente, se ha mostrado el resultado final de la aplicación, pudiendo observar que se cumplen todos los objetivos especificados. Además, se ha realizado la validación de la aplicación mediante una prueba de usabilidad llevada a cabo por usuarios reales, dando como resultado el correcto funcionamiento de esta.

Sin embargo, debido al tiempo limitado que se ha presentado para realizar este proyecto, esta primera versión de la aplicación cuenta con funcionalidades limitadas, por lo que se

ha realizado un listado de mejoras y ampliaciones a implementar de cara al futuro. Estas incluyen funcionalidades que la aplicación debería tener para obtener un rendimiento más completo, así como también mejoras para solucionar problemas identificados por los usuarios. Los cambios por realizar son los siguientes:

1. Permitir al usuario editar sus documentos y sus datos personales. Como se ha mostrado antes, actualmente existe un apartado personal para el usuario en el que se le muestran algunos datos personales, así como los documentos que ha subido. La funcionalidad por implementar sería permitir al usuario editar ambos, ya que actualmente el usuario no puede realizar ningún cambio en estos. En cuanto a los datos del usuario, esta función es más sencilla de implementar, debido a que como se mostraba anteriormente, el objeto usuario es proporcionado por el *framework* ASP.NET Boilerplate y ya cuenta con algunas funcionalidades integradas, entre ellas la de modificación de usuarios.
2. Optimización de la aplicación en dispositivos móviles. Debido a que el desarrollo se ha centrado en la aplicación para navegadores en pantallas grandes, la interfaz de la aplicación no está optimizada para dispositivos móviles, aunque sigue contando con todas las funcionalidades que proporciona la aplicación. Actualmente, la aplicación puede ser usada en dispositivos móviles, sin embargo, la experiencia para el usuario no será nada agradable debido a la interfaz des optimizada. Haciendo uso de los correctos estilos y con la ayuda de React Bootstrap, este problema tiene fácil solución.
3. Mejorar la interfaz de la aplicación. La interfaz en algunas partes de la aplicación requiere de algunas mejoras para dar una mejor experiencia a los usuarios, como la ventana de previsualización de documentos, la cual necesita un retoque en cuanto a algunos aspectos en ella, así como la de visualización de documentos, donde algunos usuarios echaban en falta un botón para volver a la ventana de previsualización.
4. Implementar un sistema de paginación en la visualización de documentos. Si el documento a leer es poco extenso, actualmente no hay ningún problema pasar por las pocas páginas que contenga, pero en caso contrario resulta tedioso el navegar por un documento con centenares de páginas. Por ello, es necesario la implementación de un sistema de paginado para poder navegar hasta la página deseada.
5. Implementar un reconocimiento del tipo de documentos que se intentan subir. Como ya se ha comentado anteriormente en los resultados de las encuestas, algunos navegadores permiten saltarse la restricción de extensión de archivos a la hora de subir las portadas y los documentos. Por lo que es necesario implementar un sistema que verifique que la extensión del archivo que se pretende subir es la correcta.
6. Aumentar el número de categorías disponibles. En esta primera versión se cuenta con un pequeño número de categorías con las cuales los usuarios pueden indicar a que géneros pertenecen sus documentos, por lo que estaría bien aumentar el número de estas para que los usuarios puedan expresar con mayor precisión la temática de sus obras.
7. Implementar un sistema para detectar documentos con derechos. Actualmente, cualquier documento puede ser subido a la biblioteca. Esto provoca que sea posible la distribución de documentos con derechos de autor sin el consentimiento de este, por lo que es necesario implementar un sistema que detecte si los documentos a subir son libres de derechos.

Bibliografía

[1] Anna Pérez. Características y fases del modelo incremental. Disponible en https://www.obsbusiness.school/blog/caracteristicas-y-fases-del-modelo-incremental, consultada el 21/07/2023.

[2] A. Weitzenfeld. *Ingeniería del Software OO con UML. Java e Internet*. Thomson, 2005.

[3] Oracle. ¿qué es una base de datos relacional (sistema de gestión de bases de datos relacionales)? Disponible en https://www.oracle.com/es/database/what-is-a-relational-database/, consultada el 27/07/2023.

[4] BBVA. Api rest: qué es y cuáles son sus ventajas en el desarrollo de proyectos, 2016. Disponible en https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/#:~:text=Buscando%20una%20definici%C3%B3n%20sencilla%2C%20REST,posibles%2C%20como%20XML%20y%20JSON, consultada el 9/07/2023.

[5] Deyimar A. Qué es react: definición, características y funcionamiento, 2023. Disponible en https://www.hostinger.es/tutoriales/que-es-react, consultada el 9/07/2023.

[6] React Bootstrap. React bootstrap. Disponible en https://react-bootstrap.netlify.app/, consultada el 9/07/2023.

[7] React Router. Main concepts. Disponible en https://reactrouter.com/en/main/start/concepts, consultada el 17/07/2023.

[8] Microsoft. Asp.net boilerplate. Disponible en https://aspnetboilerplate.com, consultada el 10/07/2023.

[9] Microsoft. Asp.net. Disponible en https://dotnet.microsoft.com/es-es/apps/aspnet, consultada el 10/07/2023.

[10] Microsoft. ¿qué es .net? Disponible en https://dotnet.microsoft.com/es-es/learn/dotnet/what-is-dotnet, consultada el 10/07/2023.

[11] Microsoft. Entity framework documentation. Disponible en https://learn.microsoft.com/en-us/ef/, consultada el 11/07/2023.

[12] Microsoft. Introducción a azure blob storage. Disponible en https://learn.microsoft.com/es-es/azure/storage/blobs/storage-blobs-introduction, consultada el 17/07/2023.

[13] React. usestate. Disponible en https://react.dev/reference/react/useState, consultada el 18/07/2023.

[14] Microsoft. What is azure. Disponible en https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/, consultada el 19/07/2023.

APÉNDICE A
Caso de estudio

Se pretende crear una biblioteca de dominio público virtual, en la que los propios usuarios que participarán serán los consumidores y los productores del contenido de esta.

La biblioteca es un espacio donde se publican y se leen documentos de diferentes relacionados con diferentes ámbitos, como historia, arte, ciencia, etc. los cuales contienen el título del documento, una breve descripción del mismo, las categorías a las que este asociado, una portada y el propio documento. La biblioteca muestra todos los documentos, los cuales pueden ser filtrados por categorías o por título. Además, estos pueden ser ordenados mediante diferentes pautas, como su antigüedad o popularidad. Todos los documentos han de ser subidos a la biblioteca por un usuario, el cual deberá haberse registrado en la web primero, proporcionando un nombre de usuario, un correo electrónico y una contraseña.

Los usuarios que no hayan iniciado sesión podrán navegar por la biblioteca, pudiendo solamente observar las características de los documentos de esta. Por otro lado, los usuarios que ya hayan iniciado sesión podrán leer cualquiera de los documentos dentro de la biblioteca. Además, ahora también tendrán la opción de valorar, tanto positiva como negativamente, los documentos que quiera. El usuario tendrá la opción de cambiar su valoración posteriormente.

APÉNDICE B
Formulario usuario

Opciones: sí y no.

1. ¿Has podido filtrar los documentos por categorías y/o por título correctamente?
2. ¿Has podido ordenar los documentos por antigüedad o por valoración correctamente?
3. ¿Has podido iniciar sesión o registrarte correctamente?
4. ¿Has podido leer los documentos correctamente?
5. ¿Has podido valorar los documentos positiva o negativamente?
6. ¿Has podido cambiar tu valoración en documentos que ya habías valorado?
7. ¿Has podido subir tus propios documentos correctamente?
8. ¿Has tenido alguna dificultad a la hora de navegar por los menús de la aplicación para encontrar lo que buscabas?
9. ¿Has necesitado ayuda para realizar alguna acción?

Opción múltiple: ¿Algunos de estos aspectos de la aplicación han sido de tu desagrado?

- La interfaz
- Como se muestran los documentos
- El rendimiento
- Las funcionalidades que presenta
- Otro
- No me ha desagradado nada

Pregunta adicional respecto a la opción múltiple: Si en la pregunta anterior has seleccionado la opción 'Otro', por favor, especifique el aspecto de la aplicación que no le ha gustado.

APÉNDICE C
Objetivos de Desarrollo Sostenible

La Organización de las Naciones Unidas aprobó en 2015 la Agenda 2030 sobre el Desarrollo Sostenible, proponiendo como objetivo el mejorar el bienestar de todas las personas. Dicha Agenda define un total de 17 Objetivos de Desarrollo Sostenible (ODS), los cuales abarcan tareas como el crecimiento económico, el compromiso con las necesidades sociales y la protección del medio ambiente. La Universitat Politècnica de València se compromete socialmente como institución a la consecución de estos objetivos desde sus distintas facetas. A continuación, se muestran los diferentes objetivos:

1. Fin de la pobreza
2. Hambre Cero
3. Salud y Bienestar
4. Educacion de Calidad
5. Igualdad de género
6. Agua limpia y saneamiento
7. Energia asequible y no contaminante
8. Trabajo decente y crecimiento económico
9. Industria innovación e infraestructura
10. Reduccion de las desigualdades
11. Ciudades y comunidades sostenibles
12. Produccion y consumos responsables
13. Accion por el clima
14. Vida submarina
15. Vida de ecosistemas terrestres
16. Paz, justicia e instituciones
17. Alianzas para lograr objetivos

Seguidamente, se muestra el grado de relación entre este proyecto y los Objetivos de Desarrollo Sostenible mencionados anteriormente mediante la siguiente tabla:

ODS	Alto	Medio	Bajo	No procede
1 Fin de la pobreza				x
2 Hambre Cero				x
3 Salud y Bienestar			x	
4 Educación de Calidad	x			
5 Igualdad de género				x
6 Agua limpia y saneamiento				x
7 Energía asequible y no contaminante				x
8 Trabajo decente y crecimiento económico				x
9 Industria innovación e infraestructura			x	
10 Reducción de las desigualdades				x
11 Ciudades y comunidades sostenibles			x	
12 Producción y consumos responsables				x
13 Acción por el clima				x
14 Vida submarina				x
15 Vida de ecosistemas terrestres				x
16 Paz, justicia e instituciones	x			
17 Alianzas para lograr objetivos	x			

Entre los distintos objetivos, se han podido lograr algunos de ellos en mayor o menor medida. Dichos objetivos se presentan a continuación:

1. Educación de Calidad, ya que tanto el trabajo de una biblioteca como la influencia que esta ejerce benefician positivamente al enriquecimiento cultural de la población. Esto es debido al acceso a bastas cantidades de conocimientos de todo tipo de forma completamente gratuita, así como la distribución de este, sin el impedimento de este debido a cuestiones políticas, ideológicas, sociales, culturales o religiosas.
2. Paz, Justicia e Instituciones, como se ha mencionado anteriormente, gracias a los conocimientos que son proporcionados por la biblioteca, las personas pueden nutrirse de los mismos, lo que resulta en una población más informada sobre el tema, haciendo así posible que esta no cause daños al sector, como discriminación o prejuicios.
3. Alianzas para lograr objetivos, puesto que, dada la naturaleza de la biblioteca, sin colaboración de los usuarios, los cuales son los encargados de nutrir de conocimientos la aplicación, los contenidos de esta serían inexistentes. Por lo tanto, la biblioteca fomenta la cooperación de individuos para lograr la expansión de conocimientos a más individuos.

En cuanto al resto de objetivos, considero que no proceden analizar ya que no tienen relación con este proyecto dadas las características de este, el cual pretende proporcionar bastas cantidades de conocimientos en forma de documentos a la población, dejando a su vez que sea esta misma la que expanda dichos conocimientos. Por otro lado, sí que es posible que, de forma indirecta, se cumplan algunos de estos objetivos, como por ejemplo el de 'Salud y Bienestar', puesto que al enriquecer a la población culturalmente esta tomase un rumbo de vida mejor.