GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

BIBLIOTRAIL: PLATAFORMA WEB MODULAR PARA LA GESTIÓN CENTRALIZADA DE BIBLIOTECAS

Autor: Guadalupe Martínez Blanco

Director: Luis Francisco Sánchez Merchante

Madrid, 5 de mayo de 2025

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título Aplicación Web para la gestión de bibliotecas en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2024/25 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Firmado por MARTINEZ BLANCO GUADALUPE - ***1285** el día 21/06/2025 con un certificado emitido por AC FNMT Usuarios

Fdo.: Guadalupe Martínez Blanco Fecha: ……/ ……/ ……

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Luis Fco Sánchez
2025.06.17
13:10:49 +02'00'

Fdo.: Luis Francisco Sánchez Merchante Fecha: ……/ ……/ ……

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

BIBLIOTRAIL: PLATAFORMA WEB MODULAR PARA LA GESTIÓN CENTRALIZADA DE BIBLIOTECAS

Autor: Guadalupe Martínez Blanco

Director: Luis Francisco Sánchez Merchante

Madrid, 5 de mayo de 2025

Agradecimientos

Me gustaría agradecerle este trabajo a Luis, mi director de proyecto, por toda su ayuda y consejos que me ha dado durante la elaboración de este Trabajo de Fin de Grado, y a mis familiares y amigos por su apoyo durante este curso.

APLICACIÓN WEB PARA LA GESTIÓN DE BIBLIOTECAS

Autor: Martínez Blanco, Guadalupe.
Director: Sánchez Merchante, Luis Francisco.
Entidad Colaboradora: ICAI – (Universidad Pontificia Comillas).

RESUMEN DEL PROYECTO

Este Trabajo de Fin de Grado ha consistido en el desarrollo de una plataforma web, llamada BiblioTrail, diseñada para la gestión bibliotecaria. Además, permite unificar el acceso a varias bibliotecas desde un mismo entorno, manteniendo la independencia de cada una de ellas. A pesar de esta autonomía, se ha logrado que haya una comunicación efectiva entre sistemas mediante APIs REST y peticiones HTTP. La plataforma permite a los usuarios realizar búsquedas de libros, hacer préstamos, inscribirse a eventos y reservar salas de trabajo desde un solo sitio web.

Palabras clave: Aplicación web, gestión bibliotecaria, acceso centralizado, APIs REST, Django, Django REST Framework, independencia, arquitectura modular, experiencia de usuario.

1. Introducción

Actualmente, muchas bibliotecas aún funcionan de forma aislada con sistemas que no favorecen el acceso centralizado a los recursos y obligan a los usuarios a navegar por plataformas distintas, enfrentarse a interfaces diferentes y a repetir procesos de búsqueda innecesarios en sitios web.

Este Trabajo de Fin de Grado propone una plataforma web llamada BiblioTrail que unifica el acceso a varias bibliotecas desde un mismo entorno. Esto permite que los usuarios puedan buscar libros, consultar ejemplares, gestionar préstamos, reservar salas e inscribirse a eventos a través de una interfaz sencilla y sin dificultades. Esta integración y unificación entre sistemas se realiza gracias a APIs externas.

El objetivo es mejorar la experiencia de usuario, facilitar la gestión bibliotecaria y aprovechar mejor los recursos disponibles, permitiendo que estudiantes, profesores, investigadores e interesados puedan tener un mejor acceso a la información.

2. Definición del proyecto

Este proyecto consiste en el desarrollo de una plataforma web que actúa como agregador para la gestión bibliotecaria. Mediante una única interfaz, permite a los usuarios que se dan de alta en ella consultar catálogos, gestionar préstamos, inscribirse en eventos de distintas bibliotecas independientes, conservando siempre la autonomía de cada una.

La arquitectura del sistema es distribuida y está conformada por tres proyectos independientes: un sistema central (plataforma agregadora) y una o varias bibliotecas

externas. Cada biblioteca expone su propia API REST, que es consultada por el sistema central para mostrar en el sitio web la información de forma unificada.

Para implementar cada uno de estos tres proyectos se ha utilizado Django y Django REST Framework. Estas herramientas han permitido construir un sistema escalable, seguro y dividido en módulos. Además, durante el desarrollo se ha priorizado la experiencia de usuario. Se ha diseñado una interfaz clara e intuitiva para el usuario, pensada en facilitar el uso tanto a estudiantes e investigadores como al personal bibliotecario.

3. Descripción del modelo/sistema/herramienta

El objetivo de este proyecto es unificar el acceso a los catálogos y servicios de distintas bibliotecas favoreciendo una mejor gestión de sus recursos. Con este trabajo se busca que cualquier usuario pueda consultar libros, reservar espacios o inscribirse a eventos de desde una única plataforma web.

El sistema se ha construido en Django [1] y sigue una arquitectura cliente-servidor. En la Figura 1 se muestra un diagrama detallado de cómo funciona la arquitectura del sistema implementado. El servidor central hace de intermediario entre el usuario (cliente) y las APIs de las bibliotecas. Cada biblioteca es independiente del resto, cuenta con su propia base de datos local y lógica interna. Estas instituciones se comunican con el sistema central a través de endpoints REST.

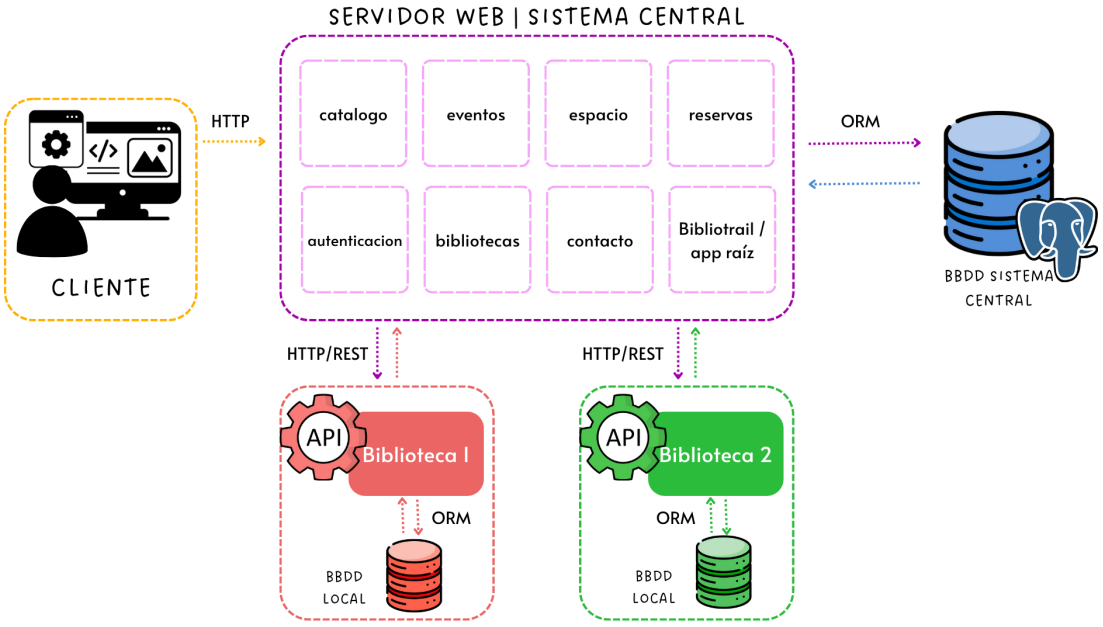


Figura 1.- Arquitectura del proyecto

Desde el inicio la estructura del proyecto ha sido modular. Esto ha permitido dividir mejor la lógica del sistema. Se ha separado en apps, distribuyendo de esta forma mejor las funcionalidades y permitiendo escalar el sistema en caso de que se quisieran añadir futuras ampliaciones o más bibliotecas. Las principales funcionalidades de esta herramienta web han sido: la búsqueda de libros, la gestión de reservas y la inscripción a eventos.

La interfaz gráfica desarrollada es clara e intuitiva. Para facilitar su diseño, al comenzar la etapa de planificación del proyecto se crearon mockups para guiar y poder ver de forma clara cómo organizar en secciones las funcionalidades del programa.

En cuanto a la gestión de usuarios, se han implementado mecanismos para autenticar a los usuarios, gestionar sus permisos y proteger su información. Django incorpora sistemas de seguridad [2] y proporciona un panel de administración muy completo para gestionar las cuentas de los usuarios y los modelos de cada aplicación. El sistema cuenta con permisos para diferenciar entre usuarios normales y bibliotecarios, dándoles permisos diferentes y limitando el acceso a determinadas partes de la página web en función del rol.

4. Resultados

El desarrollo de BiblioTrail ha cumplido con los objetivos propuestos. Se ha implementado una plataforma web intuitiva que permite centralizar los servicios bibliotecarios eficazmente.

La interfaz de usuario permite una navegación clara, accesible y adaptada a cada dispositivo (*responsive*), permitiendo al usuario consultar libros, realizar reservas y gestionar sus préstamos y eventos en un solo lugar.

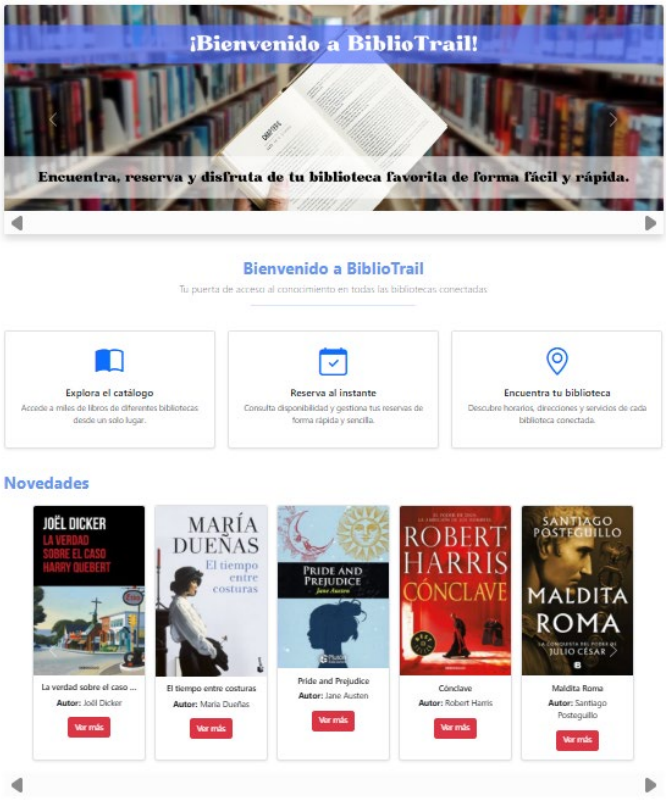


Figura 2.- Interfaz plataforma desarrollada

Además, se han implementado correctamente las funcionalidades propuestas como la autenticación de usuarios, la gestión de recursos bibliotecarios, la inscripción a eventos o la visualización de reservas activas.

Durante el proceso se han superado dificultades como familiarizarse con un entorno nuevo, los frameworks de Django y Django REST Framework [3], la migración de base de datos a PostgreSQL o la separación y comunicación entre proyectos. Sin embargo, a pesar de estos retos, se ha logrado implementar satisfactoriamente una plataforma adaptable, segura y que cumple con las funcionalidades propuestas.

El código completo del sistema desarrollado se ha gestionado desde un repositorio de GitHub. Esto ha permitido llevar un mejor control de versiones durante el proceso de desarrollo del proyecto y está accesible para su consulta [4].

5. Conclusiones

Como se ha mencionado en el apartado anterior, se han conseguido alcanzar los objetivos planteados al inicio del proyecto. Se ha construido una plataforma web funcional, escalable y que permite centralizar el acceso a servicios de distintas plataformas independientes en un único lugar.

Uno de los mayores logros ha sido conseguir una comunicación efectiva entre sistemas mediante la consulta de APIs REST. Esto ha permitido que las bibliotecas puedan compartir sus datos dinámicamente sin tener que adaptar su lógica a cada una de ellas y manteniendo una estructura común.

El sistema permite a los usuarios realizar préstamos, inscribirse a eventos y reservar espacios. Además, en todo momento, se ha priorizado una buena experiencia de usuario haciendo que la interfaz sea amigable y clara. Los frameworks más importantes para conseguir desarrolla el proyecto han sido Django y Django REST Framework. Estos programas han añadido mecanismos de seguridad y validación de usuarios, y favorecen la escalabilidad futura de la plataforma.

Para finalizar, BiblioTrail por tanto ofrece una solución eficiente para la gestión bibliotecaria, respetando la independencia y autonomía de cada proyecto, y facilitando el acceso a la información y gestión de recursos unificando todos sus servicios en un mismo sitio web.

6. Referencias

[1] Django Software Foundation, «Django,» 2005. [En línea]. Available: https://www.djangoproject.com/. [Último acceso: 8 mayo 2025].

[2] Django Software Foundation, «Security in Django,» [En línea]. Available: https://docs.djangoproject.com/es/5.2/topics/security/. [Último acceso: 8 mayo 2025].

[3] Django Software Foundation, «Django REST Framework,» 2013. [En línea]. Available: https://www.django-rest-framework.org/. [Último acceso: 8 mayo 2025].

[4] G. M. Blanco, «Repositorio GitHub BiblioTrail,» [En línea]. Available: https://github.com/gmblanco/BiblioTrail. [Último acceso: 9 junio 2025].

BIBLIOTRAIL: MODULAR WEB PLATFORM FOR CENTRALIZED LIBRARY MANAGEMENT

Author: Martínez Blanco, Guadalupe.
Supervisor: Sánchez Merchante, Luis Francisco.
Collaborating Entity: ICAI – (Universidad Pontificia Comillas)

ABSTRACT

This Final Degree Project consisted in the development of a web platform, called BiblioTrail, designed for library management. In addition, it allows unifying access to several libraries from the same environment, maintaining the independence of each one of them. Despite this autonomy, effective communication between systems has been achieved through REST APIs and HTTP requests. The platform allows users to search for books, make loans, register for events and reserve work rooms from a single website.

Keywords: Web application, library management, centralized access, REST APIs, Django, Django REST Framework, independence, modular architecture, user experience.

1. Introduction

Currently, many libraries still operate in isolation with systems that do not favor centralized access to resources and force users to navigate through different platforms, face different interfaces and repeat unnecessary search processes on websites.

This Final Degree Project proposes a web platform called BiblioTrail that unifies access to several libraries from a single environment. This allows users to search for books, consult copies, manage loans, reserve rooms and register for events through a simple and seamless interface. This integration and unification between systems is achieved through external APIs.

The goal is to improve the user experience, facilitate library management and make better use of available resources, allowing students, teachers, researchers and other interested parties to have better access to information.

2. Project definition

This project consists of the development of a web platform that acts as an aggregator for library management. Through a single interface, it allows users to consult catalogs, manage loans, register in events of different independent libraries, always preserving the autonomy of each one.

The system architecture is distributed and consists of three independent projects: a central system (aggregator platform) and one or more external libraries. Each library exposes its own REST API, which is consulted by the central system to display the information on the web site in a unified way.

Django and Django REST Framework have been used to implement each of these three projects. These tools have allowed us to build a scalable, secure and modular system. In addition, during the development, the user experience has been prioritized. A clear and intuitive user interface has been designed to facilitate the use by students, researchers or library staff.

3. Description of the model/system/tool

The objective of this project is to unify the access to the catalogs and services of different libraries, favoring better management of their resources. The aim of this work is to allow any user to consult books, reserve spaces or register for events from a single web platform.

The system has been built in Django [1] and follows a client-server architecture. Figure 1 shows a detailed diagram of how the architecture of the implemented system works. The central server acts as an intermediary between the user (client) and the APIs of the libraries. Each library is independent of the others, has its own local database and internal logic. These institutions communicate with the central system through REST endpoints.

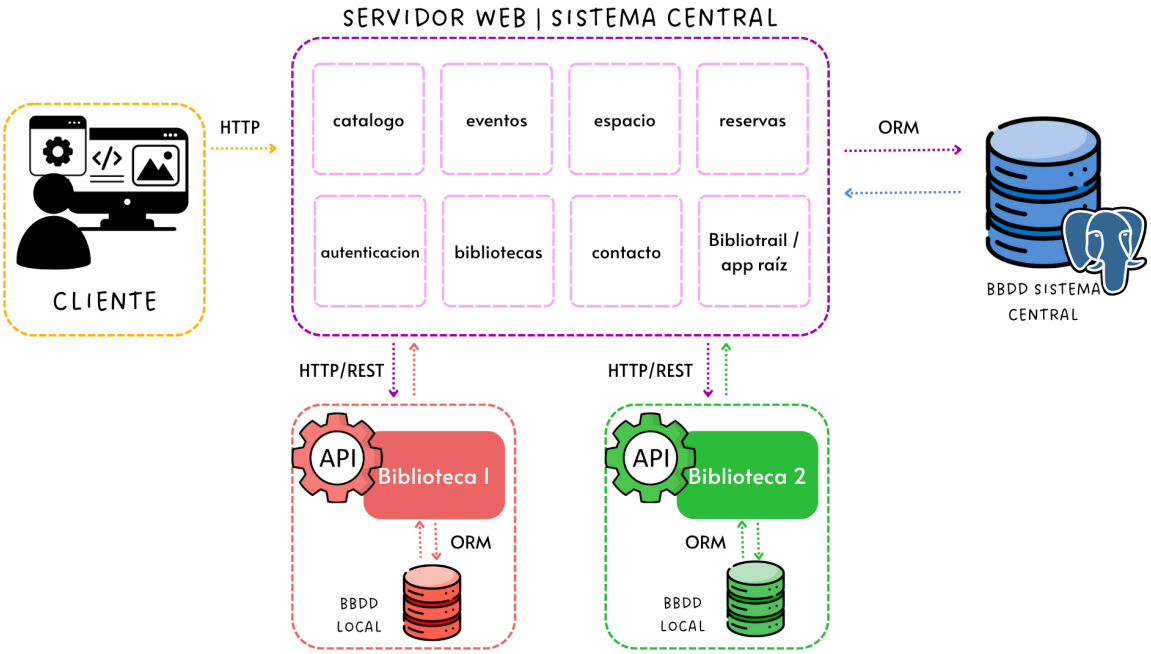


Figure 1.- Project architecture

From the beginning the structure of the project has been modular. This has allowed to divide the logic of the system better. It has been separated into apps, thus better distributing the functionalities and allowing a better scalability of the system in case we wanted to add future extensions or more libraries. The main functionalities of this web tool have been: book search, reservation management and event registration.

The graphical interface developed is clear and intuitive. To facilitate its design, mockups were created at the beginning of the planning stage of the project to guide and clearly show how to organize the program's functionalities in sections.

Regarding user management, mechanisms have been implemented to authenticate users, manage their permissions and protect their information. Django incorporates security

systems [2] and provides a very complete administration panel to manage user accounts and models for each application. The system has permissions to differentiate between normal users and librarians, giving them different permissions and limiting access to certain parts of the web page depending on the role.

4. Results

The development of BiblioTrail has met the proposed objectives. An intuitive web platform has been implemented to centralize library services effectively.

The user interface is clear, accessible and responsive, allowing users to consult books, make reservations and manage their loans and events in one place.

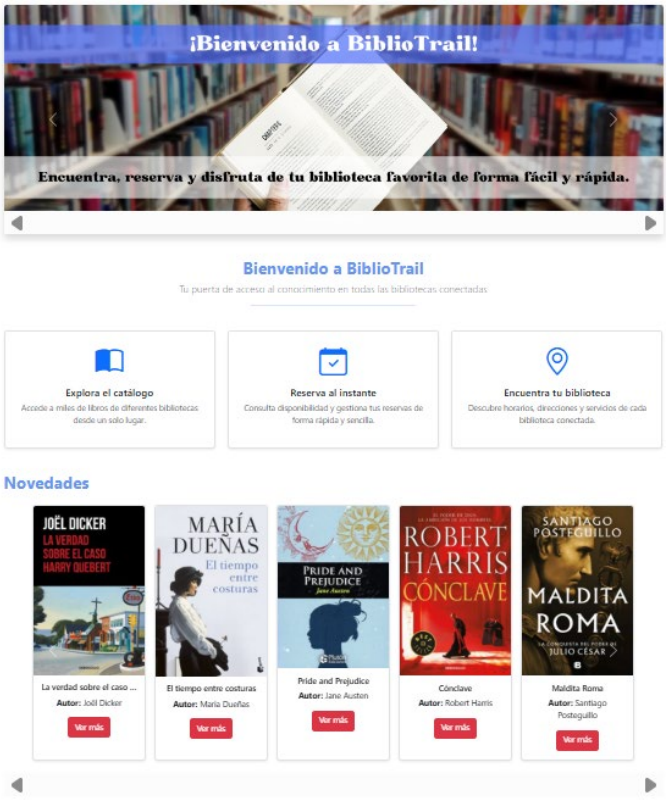


Figure 2.- Developed platform interface

In addition, the proposed functionalities such as user authentication, library resource management, event registration or display of active reservations have been correctly implemented.

During the process, difficulties have been overcome such as familiarization with a new environment, Django frameworks and Django REST Framework [3], database migration to PostgreSQL or the separation and communication between projects. However, despite these challenges, we have been able to successfully implement an adaptable, secure platform that meets the proposed functionalities.

The complete code of the developed system has been managed from a GitHub repository. This has allowed a better version control during the development process of the project and is accessible for consultation [4].

5. Conclusions

As mentioned in the previous section, the objectives set at the beginning of the project have been achieved. A functional, scalable and intuitive web platform has been built to centralize access to services from different independent platforms in a single place.

One of the greatest achievements has been to achieve effective communication between systems by querying REST APIs. This has allowed libraries to share their data dynamically without having to adapt their logic to each other and maintain a common structure.

The system allows users to make loans, register for events and reserve spaces. In addition, at all times, good user experience has been prioritized by making the interface friendly and clear. The most important frameworks to develop the project have been Django and Django REST Framework. These programs have added security and user validation mechanisms and favor the future scalability of the platform.

To conclude, BiblioTrail therefore offers an efficient solution for library management, respecting the independence and autonomy of each project and facilitating access to information and resource management by unifying all its services in a single website.

6. References

[1] Django Software Foundation, «Django,» 2005. [En línea]. Available: https://www.djangoproject.com/. [Último acceso: 8 mayo 2025].

[2] Django Software Foundation, «Security in Django,» [En línea]. Available: https://docs.djangoproject.com/es/5.2/topics/security/. [Último acceso: 8 mayo 2025].

[3] Django Software Foundation, «Django REST Framework,» 2013. [En línea]. Available: https://www.django-rest-framework.org/. [Último acceso: 8 mayo 2025].

[4] G. M. Blanco, «Repositorio GitHub BiblioTrail,» [En línea]. Available: https://github.com/gmblanco/BiblioTrail. [Último acceso: 9 junio 2025].

Índice de la memoria

Capítulo 1. Introducción 9
1.1 Motivación del proyecto 10
Capítulo 2. Descripción de las Tecnologías 11
2.1 Tecnologías de desarrollo web 11
2.1.1 Django 11
2.1.2 Django REST Framework 16
2.2 Bases de datos 18
2.2.1 SQLite 18
2.2.2 PostgreSQL 19
2.2.3 Análisis comparativo 20
2.3 Tecnologías REST y de integración 21
2.3.1 Protocolo HTTP 21
2.3.2 APIs REST 21
2.3.3 Librería HTTPX 23
2.4 Interfaz de usuario 23
2.4.1 Tecnologías base del front-end 23
2.4.2 Frameworks y herramientas de diseño 24
Capítulo 3. Estado de la Cuestión 25
3.1 Introducción 25
3.2 Evolución de los sistemas de gestión bibliotecaria 25
3.2.1 Orígenes: las bibliotecas como espacios físicos 26
3.2.2 Primera informatización: OPAC y aparición de sistemas locales 27
3.2.3 Sistemas ILS 28
3.2.4 Actualidad y digitalización 29
3.3 Análisis de competidores y plataformas existentes 29
3.3.1 Software propietario 29
3.3.2 Software libre 30
3.3.3 Plataformas de acceso remoto 31
3.4 Análisis comparativo entre plataformas existentes y la propuesta desarrollada 32
3.5 Desafíos actuales 34

Capítulo 4. Definición del Trabajo 35
4.1 Justificación 35
4.2 Objetivos 36
4.2.1 Objetivos principales 36
4.2.2 Objetivos secundarios 36
4.3 Metodología de trabajo y tecnologías utilizadas 37
4.3.1 Metodología de trabajo 37
4.3.2 Marco de desarrollo y lenguajes utilizados 37
4.4 Planificación y Estimación Económica 38
4.4.1 Planificación 38
4.4.2 Estimación económica 39

Capítulo 5. Sistema/Modelo Desarrollado 41
5.1 Análisis del Sistema 41
5.1.1 Contexto y alcance 41
5.1.2 Análisis de requisitos 42
5.1.3 Casos de uso 44
5.2 Diseño 49
5.2.1 Arquitectura 49
5.2.2 Modelo de datos 53
5.2.3 Diseño de la interfaz de usuario 56
5.2.4 Seguridad y gestión de usuarios 61
5.2.5 Comunicación entre sistemas 64
5.3 Implementación 67
5.3.1 Organización del proyecto 68
5.3.2 Configuración del entorno 69
5.3.3 Implementación de módulos principales 70

Capítulo 6. Análisis de Resultados 75
6.1 Evaluación del cumplimiento de objetivos 75
6.1.1 Desarrollo de plataforma web para la gestión bibliotecaria 75
6.1.2 Creación de una interfaz eficaz e intuitiva 76
6.1.3 Agregación de datos y conexión de bibliotecas externas 77
6.1.4 Implementación de funcionalidades 77

6.1.5 Seguridad, gestión de usuarios y escalabilidad *78*
6.2 Retos y limitaciones encontradas 78
6.2.1 Separación en proyectos y arquitectura distribuida *78*
6.2.2 Aprendizaje de Django *79*
6.2.3 Lógica de conexión con APIs externas *79*
6.2.4 Organización del tiempo y carga académica *79*
6.2.5 Migración de base de datos *80*
6.3 Resumen 80

Capítulo 7. Conclusiones y Trabajos Futuros 81
7.1 Conclusiones 81
7.2 Líneas de trabajo futuras 82
7.2.1 Mejoras funcionales *82*
7.2.2 Mejoras de infraestructura *83*
7.2.3 Mejoras de gestión y análisis *83*

Capítulo 8. Bibliografía 84

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS 92

ANEXO II: MOCKUP DE LA INTERFAZ 94

ANEXO III: CASOS DE USO 101

ANEXO IV: MANUAL DE USUARIO 104
1.- Cómo acceder al sitio web de BiblioTrail 104
Cómo registrarse 105
Cómo modificar el perfil de usuario 106
Cómo iniciar sesión 108
Cómo consultar el catálogo y reservar libros 109
Cómo realizar búsquedas en el catálogo *109*
Cómo realizar reservas de ejemplares *112*
Cómo consultar las bibliotecas asociadas 113
Cómo inscribirse a un evento 114
Cómo reservar espacios de trabajo 117

Cómo contactar con el soporte de la plataforma 119
Cómo gestionar las reservas del usuario 120
Gestión de préstamos 120
Gestión de eventos 121
Gestión de espacios 122
Cómo cerrar sesión 123
Cómo acceder al panel de administración 123
Cómo gestionar devoluciones de préstamos 125

Índice de figuras

Figura 1.- Arquitectura del proyecto 8
Figura 2.- Interfaz plataforma desarrollada 9
Figura 3.- Diagrama arquitectura MVT Django [62] 12
Figura 4.- Diagrama arquitectura REST 22
Figura 5.- Diagrama de casos de uso UML 48
Figura 6.- Diagrama de componentes de la arquitectura 50
Figura 7.- Diagrama de clases modelo central 54
Figura 8.- Diagrama de clases biblioteca 56
Figura 9.- Diagrama de flujo de navegación 60
Figura 10.- Panel de administración de Django 62
Figura 11.- Ejemplo validaciones formulario 64
Figura 12.- Endpoints REST relevantes 65
Figura 13.- Mensajes de fallo 66
Figura 14.- Estructura de carpetas 68
Figura 15.- Tabla resumen apps Bibliotrail 71
Figura 16.- DRF interfaz API 73
Figura 17.- Ejemplos serializers 74
Figura 18.- Ejemplo APIView 74
Figura 19.- Plataforma web desarrollada 75
Figura 20.- UX interfaz gráfica 76
Figura 21.- Objetivos de desarrollo sostenible 92
Figura 22.- Pantalla de inicio mockup 94
Figura 23.- Pantalla de registro mockup 95
Figura 24.-Pantalla de inicio de sesión mockup 95
Figura 25.- Pantalla de bibliotecas mockup 96
Figura 26.- Diseño de catálogo mockup 96
Figura 27.- Pantalla de novedades mockup 97
Figura 28.- Diseño de inscripción a eventos mockup 97

Figura 29.- Calendario de eventos mockup 98
Figura 30.- Pantalla de contacto mockup 98
Figura 31.- Diseño de gestión de reservas mockup 99
Figura 32.- Pantalla de gestión de préstamos mockup 99
Figura 33.- Pantalla de gestión de asientos mockup 100
Figura 34.- Pantalla de gestión de eventos mockup 100
Figura 35.- Página de inicio 104
Figura 36.- Formulario de registro 105
Figura 37.- Icono perfil usuario 106
Figura 38.- Editar perfil 106
Figura 39.- Formulario editar perfil 107
Figura 40.- Perfil editado 108
Figura 41.- Inicio de sesión 109
Figura 42.- Fallo de inicio de sesión 109
Figura 43.- Catálogo principal web 110
Figura 44.- Filtros de búsqueda de libros 111
Figura 45.- Búsqueda filtrada 111
Figura 46.- Reserva ficha libro 112
Figura 47.- Vista detallada libro 112
Figura 48.- Vista bibliotecas 114
Figura 49.- Vista de eventos 115
Figura 50.- Calendario de eventos 115
Figura 51.- Ventana emergente calendario eventos 116
Figura 52.- Ficha de evento 116
Figura 53.- Inscripción correcta evento 116
Figura 54.- Sistema de consulta de disponibilidad 117
Figura 55.-Tabla de disponibilidad por franja horaria 118
Figura 56.- Ventana emergente confirmación reserva 118
Figura 57.- Notificación de éxito 118
Figura 58.- Formulario de contacto 119

Figura 59.- Pantalla principal de mis reservas 120
Figura 60.- Sección de mis préstamos 121
Figura 61.- Mis eventos 122
Figura 62.- Reservas de espacios 122
Figura 63.- Inicio sesión panel administrador 123
Figura 64.- Panel de administración 124
Figura 65.- Gestión perfiles de usuarios 124
Figura 66.- Gestión de eventos 125
Figura 67.- Gestión de espacios de trabajo 125
Figura 68.- Gestionar devoluciones de préstamos 126
Figura 69.- Modificación préstamo devuelto 126

Índice de tablas

Tabla 1.- Análisis comparativo alternativas DRF (elaboración propia) 18
Tabla 2.- Comparativa gestor de base de datos (adaptado de diversas fuentes) 20
Tabla 3.- Tabla comparativa competidores (elaboración propia) 33
Tabla 4.- Diagrama de Gantt para planificación del proyecto 38
Tabla 5.- Estimación económica 40
Tabla 6.- Resumen casos de uso (elaboración propia) 47
Tabla 7.- Caso Uso 01 Reserva ejemplar 101
Tabla 8.- Caso Uso 02 Reserva espacio trabajo 102
Tabla 9.- Caso Uso 03 Inscripción evento 103

Capítulo 1. INTRODUCCIÓN

En la actualidad, la transformación digital está transformando el modo en que los usuarios obtienen información. Las bibliotecas, ya sean de carácter público, municipal o universitario, no están fuera de este proceso de transformación. Estos espacios, tradicionalmente vistos como lugares físicos dedicados a la consulta y préstamo de materiales, ahora cuentan con la posibilidad de avanzar hacia modelos más dinámicos y accesibles haciendo uso de las nuevas tecnologías digitales [5].

A pesar de los avances tecnológicos, muchas redes de bibliotecas continúan funcionando de forma aislada, con sistemas independientes que dificultan el acceso unificado a sus recursos. Para los usuarios, esto se traduce en la necesidad de consultar múltiples plataformas, repetir procesos de búsqueda o registro, y tener que enfrentarse a distintas interfaces, cada una diferente de las otras, lo que puede desincentivar el uso de los servicios bibliotecarios.

Por ello, surge la necesidad de una solución que permita centralizar y simplificar el acceso a los servicios de distintas bibliotecas. Ante este reto, este Trabajo de Fin de Grado presenta el desarrollo de una plataforma web agregadora que permite consultar catálogos de distintas bibliotecas a través de una única interfaz. La plataforma ofrece funcionalidades como la búsqueda unificada de libros, la gestión de reservas de ejemplares y espacios, y la inscripción en actividades organizadas por las bibliotecas participantes, mediante la integración de sus respectivas APIs (Application Programming Interface).

Con esta solución, se pretende mejorar la experiencia de usuario, optimizar la gestión de recursos y fomentar un uso eficiente de los servicios bibliotecarios.

1.1 MOTIVACIÓN DEL PROYECTO

Actualmente, a pesar de la creciente digitalización, los sistemas bibliotecarios siguen teniendo limitaciones como la falta de conexión entre las distintas bibliotecas o la ausencia de una interfaz común que facilite y simplifique la consulta del usuario.

Por ello, es necesario desarrollar una solución que permita a los usuarios acceder a los servicios de diferentes bibliotecas desde un único punto. El objetivo es ofrecer un acceso centralizado desde la experiencia del usuario, pero manteniendo los sistemas de cada biblioteca de forma independiente.

Además, resulta fundamental crear una plataforma web donde se priorice la experiencia de usuario. Muchos recursos actuales siguen presentando interfaces complejas y poco intuitivas [6], mientras que los usuarios buscan poder acceder e interactuar en los servicios digitales de forma sencilla y casi inmediata. Ante este escenario, es imprescindible mejorar la accesibilidad de los servicios bibliotecarios con el fin de aumentar su uso y su alcance.

Las tecnologías actuales ofrecen la oportunidad de avanzar hacia este objetivo, gracias al desarrollo de APIs REST y frameworks modernos que permiten diseñar una plataforma escalable, modular y ágil, capaz de interconectar distintos servicios de forma eficiente.

La motivación de este proyecto surge, por tanto, de la necesidad de construir una plataforma que actúe como puente entre las distintas bibliotecas y fomente la colaboración entre instituciones facilitando un acceso unificado y eficiente. Mediante este diseño, se busca mejorar el acceso a los recursos y, además, dar visibilidad a bibliotecas menos conocidas que podrían aprovechar de forma más eficiente de esta forma sus servicios.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Con el fin de mejorar la comprensión de este trabajo, a continuación, se explicarán las tecnologías, protocolos y frameworks elegidos para el desarrollo de este proyecto junto a la explicación de la causa por la que se eligieron y la comparativa entre otras opciones contempladas.

2.1 TECNOLOGÍAS DE DESARROLLO WEB

2.1.1 DJANGO

2.1.1.1 Definición y funcionamiento

Django es un software de desarrollo web de alto nivel que utiliza el lenguaje de programación Python. Esta herramienta facilita la creación de aplicaciones web de forma rápida, segura y eficiente. Su principal objetivo es simplificar el trabajo al desarrollador mediante la reutilización de código, la automatización de tareas comunes y el uso de componentes reutilizables [1].

La arquitectura y *work-flow* de Django se basan en un patrón de diseño llamado Model – View – Template (MTV), una variante del MVC (Modelo-Vista-Controlador) [7]. Esta arquitectura permite separar responsabilidades y facilitar el desarrollo y organización del proyecto. Cada uno de los componentes del patrón MTV cumplen una función desterminada:

- Modelo: El modelo define cómo es la estructura de los datos y la forma en la que se almacenan y consultan en la base de datos del proyecto. Cada modelo es representado

como una clase de Python y Django se encarga de traducirla a una tabla de base de datos haciendo uso de su ORM (Object-Relational Mapper).

- Vista: La vista es la lógica que se ejecuta cuando el servidor recibe una petición HTTP. A partir de la información recibida decide cómo gestionarla y qué respuesta devolverle al usuario, ya sea una página HTML, una redirección o un JSON.
- Plantilla (Template): Las plantillas son los componentes que definen la estructura de la respuesta visual que verá el usuario. Este componente se usa para generar el contenido que verá el usuario proveniente de los datos de las vistas o modelos.
- Enrutamiento (URL): Django tiene un sistema de mapeo de URLs que redirige las peticiones HTTP a la vista correspondiente.

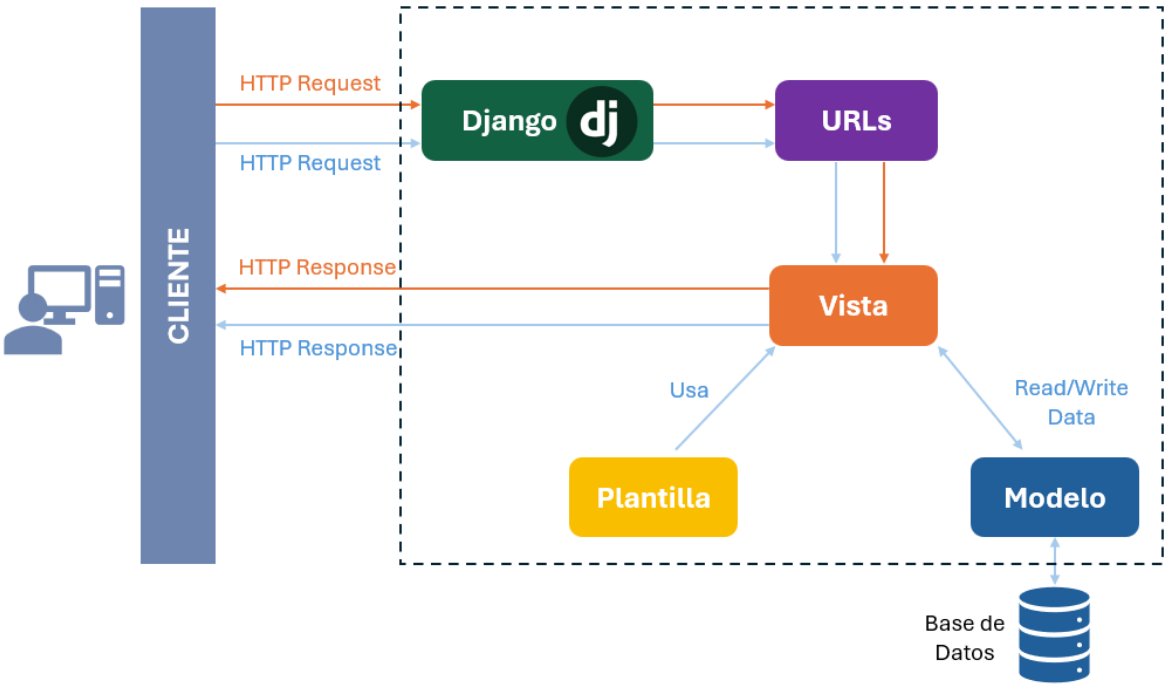


Figura 3.- Diagrama arquitectura MVT Django [62]

2.1.1.2 Ventajas y desventajas

Las ventajas de utilizar este framework de desarrollo web son las siguientes:

1. Seguridad: Django ofrece distintas funciones para proteger tanto las aplicaciones web desarrolladas como a los usuarios que las consultan [8]. Entre ellas se encuentran:
    - Protección de inyección de código SQL: Django emplea un ORM (Object-Relational Mapper) que separa el código SQL de los datos del usuario. Además, utiliza métodos de parametrización que construyen consultas SQL forma segura con el fin de evitar este tipo de ataques [2].
    - Prevención de ataques Cross-Site Scripting (XSS): Django protege a sus usuarios de los ataques XSS mediante su sistema de plantillas. Las plantillas HTML impiden que los usuarios puedan inyectar código JavaScript malicioso en las páginas web [2].
    - Protección contra Cross-Site Request Forgery (CSRF): Django cuenta con protección integrada contra los ataques CSRF como un sistema de tokens que verifican la autenticidad de los formularios POST y evitar que se suplanten las acciones del usuario [2].
    - Gestión segura de contraseñas: Django ofrece un sistema de almacenamiento de contraseñas utilizando algoritmos de hash como PBKDF2 y SHA256 [9]. Estos algoritmos aumentan su protección frente a intentos de adivinarlas o descifrarlas.
    - Autenticación y permisos integrados: Django cuenta con un sistema de permisos destinado a controlar fácilmente quién está autorizado para acceder a las distintas partes de la aplicación web.
    - Control de sesiones y cookies seguras: Este framework ofrece un sistema de sesiones anónimas automático que permite guardar y recuperar los datos de cada visitante del sitio web. De esta forma se evita exponer información personal en el navegador del usuario [10].
2. Eficiencia: La estructura modular de Django lo convierte en un framework óptimo tanto para proyectos de menor tamaño como para grandes aplicaciones web. Además, gracias a su estructura clara permite implementar funcionalidades complejas en poco

tiempo y automatizar funciones como la gestión de usuarios reduciendo el tiempo en la implementación y desarrollo.

3. Escalabilidad: Django está diseñado para crecer junto al proyecto. Este framework cuenta con una arquitectura modular, basada en aplicaciones independientes que permiten añadir funcionalidades sin afectar al resto del proyecto. Además, es compatible con bases de datos potentes y servidores de alto rendimiento lo que lo hacen ideal para aplicaciones que tienen que soportar un gran tráfico web o datos de forma eficiente.
4. ORM potente: Django incorpora un Object-Relational Mapper (ORM) que permite interactuar con las bases de datos utilizando el lenguaje de programación de Python sin necesidad de tener que escribir directamente las consultas SQL. Esto mejora la productividad y ayuda a reducir errores de los desarrolladores.
5. Compatibilidad con APIs REST: Este software permite con ayuda de Django REST Framework (DRF), crear de manera sencilla y estructurada APIs RESTful para las aplicaciones web. Además, permite convertir objetos de las bases de datos en otros formatos como JSON o XML fácilmente.
6. Comunidad y documentación: Esta herramienta de desarrollo web cuenta con una gran comunidad activa que contribuye con tutoriales web y soporte. Además, ofrece una documentación oficial muy completa lo que diferencia de otras herramientas de desarrollo web y hace que el proceso de aprendizaje sea mucho más rápido y eficiente.

A pesar de las múltiples ventajas de este framework, Django también presenta algunas limitaciones que hay que tener en cuenta a la hora de utilizarlo:

1. Monolítico y estructurado: Django tiene una estructura muy concreta e impone una forma específica de organizar el código (en forma de modelos, vistas y plantillas). Este aspecto limita la flexibilidad de la estructura del proyecto.
2. Poca flexibilidad con bases de datos no relacionales: Django es un framework pensado para bases de datos relacionales como SQLite (base de datos por

defecto), MySQL o PostgreSQL. Sin embargo, su uso para bases de datos NoSQL es limitado.

3. Curva de aprendizaje inicial: Aunque este software es muy completo puede resultar complejo al inicio, especialmente para los desarrolladores que no están familiarizados con el patrón MVT o el ORM. Además, para iniciarse en este framework es necesario tener un conocimiento previo de:
   - Lenguajes como Python para el backend o HTML & CSS para el frontend.
   - Estar familiarizado con la programación orientada a objetos (POO).
   - Comprender el protocolo HTTP, peticiones y respuestas.
   - Poseer conocimientos básicos de bases de datos y cómo funcionan.

2.1.1.3 Análisis comparativo con otras opciones contempladas

A pesar de haber escogido Django para el desarrollo del backend de la aplicación web, se consideraron como alternativa las siguientes tecnologías:

1. Flask [11]: Flask es un *micro-framework* que proporciona herramientas y funciones para el desarrollo de aplicaciones web. Es mucho más flexible y ligero que Django, no impone una estructura fija y es de gran utilidad para proyectos pequeños. Sin embargo, no cuenta con componentes como ORM y su integración y configuración es manual.
2. Ruby on Rails [12]: Ruby es un framework muy completo, que sigue el patrón MVC (Modelo-Vista-Controlador). Es un software muy flexible que permite modificar el código con facilidad y es ideal para proyectos que requieran de un desarrollo ágil (como prototipos). Además, cuenta con una comunidad activa y tiene gran popularidad en la actualidad. Sin embargo, Django destaca sobre Ruby en aspectos como la facilidad de aprendizaje, los mecanismos de seguridad, mayor documentación y comunidad. Además, es una mejor opción si se busca realizar una aplicación web con una base de datos grande o un gran tráfico de usuarios [13].

En definitiva, se acabó escogiendo Django frente al resto de tecnologías debido a su robustez (especialmente en cuanto a seguridad), estructura clara y herramientas integradas como

ORM. Django ofrece un entorno ideal para el desarrollo modular de aplicaciones web y permite realizar conexiones con APIs externas y gestionar de manera eficiente los usuarios y bases de datos. Además, su uso de Python, un lenguaje de programación conocido por su amplio uso y sintaxis clara, junto con su amplia documentación y comunidad resultaron factores claves para la elección de este entorno.

2.1.2 DJANGO REST FRAMEWORK

Django REST Framework (DRF) [3] es un framework basado en Django que permite crear APIs RESTful y desarrollar servicios web. Permite que otras aplicaciones realicen operaciones como consultar, crear, actualizar o eliminar datos (CRUD) e interactuar con una base de datos a través de peticiones HTTP de forma sencilla y eficaz. Proporciona herramientas para simplificar aspectos como la autenticación de usuarios, la serialización o el manejo de solicitudes y respuestas [14].

2.1.2.1 Ventajas y desventajas

Algunas de las ventajas de esta herramienta son:

1. Facilidad de uso: DRF permite crear de forma rápida y sencilla APIs gracias a las clases predefinidas que proporciona y a sus vistas genéricas.
2. Serialización: Esta herramienta tiene gran capacidad para convertir el formato de los modelos de Django en otros como JSON de forma sencilla e intuitiva.
3. Autenticación: El sistema de autenticación de DRF es muy potente y cuenta con métodos como las sesiones o autenticación por token que permiten identificar fácilmente al usuario y asegurar que únicamente él acceda a los recursos de la aplicación.
4. Interfaz interactiva: Django REST Framework crea automáticamente una interfaz visual intuitiva donde probar los endpoints y las distintas funciones desarrolladas.
5. Compatibilidad con Django: DRF aprovecha los recursos de Django como la estructura del proyecto y su sistema ORM. Esto permite que la integración sea fácil y se pueda utilizar ambos frameworks de forma conjunta sin necesidad de añadir complejidad al proyecto.

A pesar de que Django REST Framework sea una herramienta muy completa, también tiene algunas desventajas como:

1. Rendimiento limitado: DRF funciona peor para aplicaciones con baja latencia o muy concurridas. Además, no está inicialmente diseñado para operaciones asíncronas.
2. Menor flexibilidad: Esta herramienta tiene una estructura más rígida en comparación con otras alternativas como Flask o FastAPI.
3. Tiempo de respuesta alto: Al basarse en Django, puede responder más lento ante algunas situaciones, en especial en entornos donde hay que gestionar muchas solicitudes al mismo tiempo.

2.1.2.2 Análisis comparativo con otras opciones contempladas

A la hora de escoger la herramienta destinada a la integración de APIs RESTful se consideraron las siguientes tecnologías:

1. Flask- RESTful [15]: Flask-RESTful es una extensión de Flask para crear de manera rápida APIs REST. Esta herramienta se integra fácilmente con el ORM y está diseñada para simplificar el desarrollo de servicios. La curva de aprendizaje es baja, cuenta con una estructura muy flexible y es ideal para proyectos pequeños o prototipos.
2. FastAPI [16]: FastAPI es un framework web rápido basado en Python destaca por su rapidez, rendimiento y facilidad para construir APIs ligeras y modernas. Es especialmente útil en el diseño de endpoints asíncronos y cuenta con documentación clara e intuitiva [17].

Característica	Django REST Framework (DRF)	Flask-RESTful	FastAPI
Framework base	Django	Flask	Asíncrono
Facilidad de uso	Moderada	Muy fácil si se conoce Flask	Muy intuitivo
Curva de aprendizaje	Media	Baja	Baja-media
Rendimiento	Menor rendimiento en entornos de alta carga	Aceptable pero no síncrono	Alto rendimiento gracias al soporte asíncrono
Gestión de base de datos	Integra el ORM de Django	Flexible	No tiene ORM propio
Adecuado para	Proyectos grandes y complejos	APIs simples o prototipos	APIs rápidas y con mucha carga concurrente
Madurez	Muy estable	Estable, pero menor que DRF o FastAPI	En crecimiento

Tabla 1.- Análisis comparativo alternativas DRF (elaboración propia)

Después de analizar las distintas alternativas para el desarrollo de la API, se acabó escogiendo Django REST Framework (DRF) para esta tarea. Al tratarse de un proyecto complejo, se descartó la opción de utilizar Flask-RESTful ya que este framework estaba más orientado a proyectos de menor tamaño. Por otro lado, al estarse desarrollando el proyecto en Django resultaba lógico emplear este framework, ya que resultaba el más adecuado por su fácil integración. Al mismo tiempo, DRF permitía aprovechar la estructura diseñada en el proyecto de Django y acceder de forma sencilla a su base de datos sin tener que añadir complejidad adicional. Por estas razones, DRF se consideró como la herramienta que podía aportar más estabilidad con el proyecto y la que mejor se alineaba con sus necesidades.

2.2 BASES DE DATOS

2.2.1 SQLITE

SQLite [18] es un motor de base de datos relacional ligero, rápido y seguro basado en el lenguaje de programación C. A diferencia de otros gestores de bases de datos, SQLite no necesita un servidor independiente por lo que es ideal cuando se trabaja en aplicaciones

locales o de menor tamaño. Su estructura es muy versátil y es la base de datos configurada por defecto en Django.

Este gestor de bases de datos se caracteriza por su gran simplicidad de uso, su falta de configuración y, su alta velocidad y rendimiento. Sin embargo, presenta limitaciones como su baja escalabilidad y, además, no está diseñado para poder manejar muchas escrituras concurrentes.

2.2.2 POSTGRESQL

PostgreSQL [19] es un sistema de gestión de bases de datos relacional orientado a objetos. Este gestor es conocido por su gran potencia, versatilidad y fiabilidad. Está basado en el lenguaje SQL y su arquitectura es robusta y tiene una gran capacidad de ampliación. Es compatible con los principales sistemas operativos y entre sus ventajas destacan:

- Extensibilidad: PostgreSQL permite definir tipos de datos y funciones propios. Además, se adapta a todo tipo de proyectos y tiene capacidad de soportar grandes volúmenes de datos.
- Seguridad: PostgreSQL cuenta con distintos niveles de seguridad y de control de acceso lo que lo hace muy fiable en entornos donde la protección de los datos es clave.
- Fiabilidad y consistencia: Este gestor cumple con los principios ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) [20], lo que garantiza operaciones seguras, incluso si llega a ocurrir algún fallo en el sistema.
- Rendimiento: PostgreSQL realiza múltiples operaciones de forma simultánea sin bloquear la base de datos lo que mejora el rendimiento en entornos y aplicaciones con muchos usuarios.
- Soporte de tipos de datos: PostgreSQL puede trabajar con distintos tipos estructurados como arrays, UUIDs, rangos y documentos JSON o XML, lo que facilita mucho el trabajo con datos complejos.

2.2.3 ANÁLISIS COMPARATIVO

Durante el desarrollo de este proyecto se ha empleado el gestor de base de datos SQLite debido a su gran simplicidad, rapidez y por tratarse del gestor de base de datos de Django predeterminado. Sin embargo, a medida que el proyecto fue creciendo, se consideró migrar a una base de datos más potente, externa y que tuviese mayor fiabilidad, soporte y consistencia. Las opciones contempladas se valoran en la siguiente tabla comparativa:

Característica	SQLite	PostgreSQL	MySQL
Instalación y uso	Muy sencilla	Requiere configuración, orientado a producción	Fácil de instalar
Escalabilidad	Limitada	Alta	Alta
Concurrencia	Baja	Muy buena	Buena, aunque con alguna limitación
Complimiento ACID	Parcial	Total	Parcial/Total
Soporte de tipos avanzados	Básico	Amplio: JSON, Arrays...	Limitado
Rendimiento	Alto en consultas simples	Alto en operaciones complejas	Muy rápido en lectura
Extensibilidad	Muy limitada	Muy alta	Media
Compatibilidad con Django	Total	Recomendado oficialmente	Soportado oficialmente
Idoneidad para producción	No recomendada (solo en desarrollo)	Muy recomendable	Ampliamente utilizada

Tabla 2.- Comparativa gestor de base de datos (adaptado de diversas fuentes)

Como SQLite no poseía la escalabilidad necesaria para un proyecto donde se consultan múltiples APIs y se gestionan reservas a tiempo real, se optó por migrar a PostgreSQL, una base de datos relacional potente y con gran compatibilidad con Django. Aunque MySQL fue considerado como una posible alternativa para el entorno en producción, se descartó debido a varias razones. En primer lugar, PostgreSQL facilita una mejor gestión de los datos y proporciona un conjunto de funcionalidades más avanzadas que las de MySQL. Además, se valoró que PostgreSQL ofrecía una mayor robustez, seguridad y versatilidad en entornos de producción. Estas razones favorecieron su elección frente a otras opciones como MySQL o SQLite.

2.3 TECNOLOGÍAS REST Y DE INTEGRACIÓN

2.3.1 PROTOCOLO HTTP

El protocolo HTTP (*Hypertext Transfer Protocol*) [21] es un sistema de comunicación que favorece el intercambio de datos entre el cliente y el servidor de un sitio web. Este protocolo es ampliamente utilizado en los navegadores y supone la base para solicitar y transferir recursos. Este sistema de comunicación sigue un modelo *cliente-servidor*, donde el cliente realiza peticiones y el servidor le responde enviando los recursos que ha solicitado previamente.

Este sistema es un protocolo de capa de aplicación. Se transmite normalmente sobre TCP y se trata de un protocolo *sin estado*. Esto significa que cada petición se procesa de manera independiente y el servidor no recuerda ni lleva un registro de las solicitudes anteriores.

Este protocolo funciona mediante mensajes estructurados. Por un lado, las peticiones o solicitudes contienen el método HTTP (GET, POST, PUT o DELETE), una URL y cabeceras (opcional). Por otro lado, las respuestas a estas solicitudes incluyen el estado, metadatos y el recurso solicitado.

Gracias a su sencillez, gran uso y extensibilidad este protocolo es de los más utilizados en el desarrollo de aplicaciones y sitios web.

2.3.2 APIS REST

Una API REST [22], también llamada API RESTful [23], es una interfaz que permite a dos sistemas intercambiar información entre sí utilizando el protocolo HTTP de forma sencilla y eficiente. Se basa en la arquitectura REST (Transferencia de Estado Representacional) y se considera un estándar en el desarrollo de servicios web por su capacidad de adaptación y escalabilidad.

Está formado por un cliente que realiza solicitudes al servidor para acceder a unos datos o recursos determinados. Cada recurso tiene una URL única lo que lo identifica y las operaciones se realizan usando métodos HTTP como:

- GET: utilizado para consultar u obtener información.
- POST: empleado para crear información.
- PUT o PATCH: usado para actualizar o modificar datos.
- DELETE: método para eliminar información.

Las APIs REST pueden devolver información en distintos formatos, siendo JSON el más empleado debido a su claridad. Además, incluyen métodos de seguridad como tokens, claves o sistemas de autenticación como OAuth que permiten un acceso controlado a los recursos.

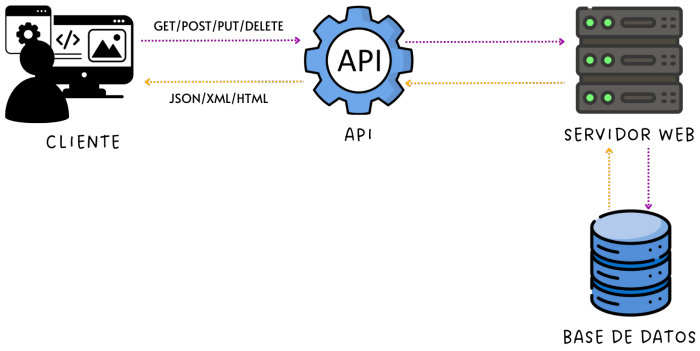


Figura 4.- Diagrama arquitectura REST

Sus características principales son:

- Independencia entre el cliente y el servidor: Cada uno de estos sistemas puede modificarse y evolucionar sin necesidad de conocer cómo funciona internamente el otro.

- Coherencia: Todas las solicitudes tienen una estructura uniforme y coherente con respecto a la interfaz.
- Sin estado: Cada solicitud es independiente y autónoma del resto y no se guarda información de las peticiones realizadas con anterioridad.
- Arquitectura por capas: Las APIs REST permiten incluir intermediarios sin que el funcionamiento quede afectado.

Además de las características mencionadas, su gran simplicidad y versatilidad hace que este tipo de APIs sean ampliamente utilizadas en arquitecturas con sistemas distribuidos o microservicios, donde se requiere que distintos sistemas se comuniquen de forma unificada.

2.3.3 LIBRERÍA HTTPX

Una librería empleada en el desarrollo de esta plataforma web ha sido la librería *httpx* [24]. Httpx es una librería de Python 3 que se puede utilizar con Django y permite realizar de forma sencilla peticiones HTTP. Esta herramienta ofrece una interfaz tanto síncrona como asíncrona lo que la hace muy útil para trabajar en servicios web. Además, es compatible con el protocolo HTTP y mantiene una estructura muy similar al paquete de Python *requests* [25].

2.4 INTERFAZ DE USUARIO

2.4.1 TECNOLOGÍAS BASE DEL FRONT-END

Para construir la interfaz de usuario se han empleado un conjunto de tecnologías destinadas al desarrollo web que definen la estructura, la estética y el comportamiento interactivo del proyecto.

2.4.1.1 HTML

HTML (*HyperText Markup Language*) [26] es un lenguaje de programación utilizado para definir el contenido y la estructura de las páginas web. A través de etiquetas y elementos

como formularios, enlaces y encabezados permite constituir la base a partir del cual se construye cualquier sitio web.

2.4.1.2 CSS

CSS (*Cascading Style Sheets*) [27] es un lenguaje de programación utilizado para definir la presentación visual y la estética de documentos HTML o XML. Este programa define cómo se deben mostrar los elementos que conforman la página web, incluyendo su posición y apariencia.

2.4.1.3 JavaScript

JavaScript [28] es un lenguaje de programación utilizado para dar interactividad a las páginas web. Este programa permite responder a eventos que realiza el usuario, modificar de forma dinámica el contenido del sitio web y permite comunicarse con servicios externos.

2.4.2 FRAMEWORKS Y HERRAMIENTAS DE DISEÑO

Además de las tecnologías base del front-end, se han empleado herramientas para acelerar el desarrollo y mejorar el estilo visual de la plataforma. En este proyecto se ha hecho uso de Bootstrap [29].

Bootstrap es un framework que facilita el desarrollo rápido de interfaces web. Este programa cuenta con una gran cantidad de estilos predefinidos de CSS y componentes interactivos en JavaScript, que permiten construir elementos gráficos sin tener que escribir el código desde cero. Boostrap promueve un diseño *responsive* y facilita una estructura coherente y ordenada requiriendo un esfuerzo mínimo. Además, su integración es muy sencilla y es perfectamente compatible con Django.

Capítulo 3. ESTADO DE LA CUESTIÓN

3.1 INTRODUCCIÓN

El objetivo de este capítulo es contextualizar el proyecto realizado dentro del panorama actual en el ámbito bibliotecario. Para ello, se explicarán brevemente los antecedentes y desarrollos relevantes en este campo desde una perspectiva histórica, tecnológica y competitiva.

Se comenzará realizando una breve explicación de cómo han evolucionado los sistemas de gestión bibliotecaria desde sus inicios hasta el momento actual, prestando atención a qué factores han sido relevantes y han favorecido la evolución hasta la tecnología actual. Posteriormente, se realizará el análisis de competidores, donde se evaluarán las distintas plataformas y herramientas presentes en el mercado que poseen características semejantes al proyecto propuesto. Por último, se realizará un análisis comparativo con el fin de identificar las principales diferencias entre estos sistemas.

3.2 EVOLUCIÓN DE LOS SISTEMAS DE GESTIÓN BIBLIOTECARIA

El sistema de gestión bibliotecaria ha experimentado numerosos cambios a los cargo del tiempo, siendo influenciado por los avances tecnológicos y las nuevas necesidades de acceso a la información. Tradicionalmente vistas como un simple lugar de conservación de libros, las bibliotecas han evolucionado con el tiempo hacia espacios educativos y culturales que no solo gestionan recursos físicos, sino que también integran recursos digitales, adaptándose a las nuevas tecnologías [30].

En esta sección se repasará brevemente esta evolución, desde los comienzos donde el proceso era manual hasta los sistemas actuales. Además, se analizarán los factores que han jugado un papel clave para este cambio y sus implicaciones en la forma de gestionar los recursos bibliotecarios.

3.2.1 ORÍGENES: LAS BIBLIOTECAS COMO ESPACIOS FÍSICOS

Las bibliotecas han sido concebidas desde sus orígenes como espacio de conservación y consulta de libros. En sus comienzos, estos espacios se trataban en su mayoría de colecciones privadas de las clases más privilegiadas. Rara vez estaban abiertas al público, y cuando lo estaban, era con el fin de que estudiosos pudiesen copiar y consultar los libros [31]. La gestión bibliotecaria se basaba en procesos completamente manuales, como el registro de inventarios o la clasificación de sus ejemplares. El préstamo bibliotecario, tal y como lo conocemos actualmente, no existía en las bibliotecas antiguas ya que la prioridad consistía en proteger y mantener los volúmenes disponibles.

Un intelectual que destacó en el ámbito de la biblioteconomía fue Gabriel Naudé. Naudé es reconocido como un gran bibliotecario y coleccionista. Además, ya en 1627 defendía en su obra "*Advis pour dresse une bibliothèque*" [32] que las bibliotecas debían ser de libre acceso, que su contenido debía ponerse al servicio del público y que el préstamo debía de formar parte como una de sus funciones [33].

A pesar de su propuesta, no es hasta finales del siglo XIX cuando surgió el concepto de biblioteca pública y el préstamo de libros se vuelve una funcionalidad habitual. Estas nuevas bibliotecas, financiadas públicamente se convirtieron en una nueva forma de acceder al conocimiento y democratizar la educación. En esta época, surgió también el concepto de préstamo interbibliotecario donde se crearon asociaciones como la *American Library Association* donde las bibliotecas pudieran colaborar entre sí y ayudarse mutuamente intercambiando sus recursos [30].

En sus inicios, la gestión interna de las bibliotecas se realizaba mediante fichas manuscritas [34], registros manuales de préstamos y devoluciones, y catálogos en fichas clasificadas según el autor, el título o el tema [35]. A pesar de que este sistema funcionó perfectamente durante décadas y supuso un control efectivo de los ejemplares, también presentaba algunas limitaciones como la lentitud que suponía la búsqueda manual de información, la posibilidad de cometer errores humanos en el sistema de fichas manuscritas y la dificultad de mantener actualizados los registros a medida que crecían las colecciones.

Ante estas limitaciones, surgieron más adelante gracias al avance de la tecnología, los primeros intentos de informatizar los catálogos bibliotecarios. Esto dio lugar al desarrollo de catálogos de acceso en línea, o OPAC (*Online Public Access Catalog*) [36].

3.2.2 PRIMERA INFORMATIZACIÓN: OPAC Y APARICIÓN DE SISTEMAS LOCALES

Como se mencionó en el apartado anterior, con el crecimiento de las colecciones y el número de usuarios, los métodos manuales dejaron de ser sostenibles para la gestión bibliotecaria. En especial, el control del préstamos y materiales suponía todo un reto, haciendo ineficiente el acceso a la información. Estas necesidades impulsaron los primeros intentos de automatización de la gestión bibliotecaria. A partir de la década de 1960 comenzaron a resolverse gracias a la incorporación de tecnologías informáticas y el inicio de los ordenadores [37].

En esta época de evolución hacia la informatización, aparecieron los primeros catálogos de acceso público en línea, conocidos como OPAC. Estos sistemas tenían una estructura modular y priorizaban tareas internas como la catalogación y la circulación de materiales. A partir de la década de 1980 el OPAC empezó a cobrar protagonismo y a reemplazar al catálogo de fichas tradicional. Este nuevo sistema mejoró aspectos como la velocidad de consulta o la accesibilidad. A pesar de estas mejoras, no aportó un gran avance con respecto a los usuarios finales ya que su diseño no estaba centrado en mejorar la experiencia de búsqueda [38].

A partir de la década de 1990, los OPACs comenzaron a incorporar mejoras orientadas a facilitar la interacción de los usuarios con el catálogo. Se empezó a prestar más atención al diseño de interfaces para hacerlos más amigables e intuitivos, adaptándolos a los usuarios. Aun así, a finales de los 2000, muchos OPACs seguían presentando limitaciones en comparación con otros servicios de búsqueda en internet [38].

De forma paralela al desarrollo de los catálogos OPAC, surgieron los primeros sistemas locales para la gestión, conocidos como *Integrated Library Systems* (ILS). Estos sistemas

impulsaron un mejor funcionamiento bibliotecario y permitieron automatizar no solo el catálogo sino también las principales funciones bibliotecarias como los préstamos, devoluciones, gestión de usuarios y adquisiciones [39]. Algunos ejemplos iniciales de estos sistemas locales fueron NOTIS (EE. UU), BALLOTS (Stanford) o DOBIS/LIBIS (IBM) [37].

3.2.3 SISTEMAS ILS

Como se comentó en el punto anterior, entre las décadas 1960-80, surgieron los Sistemas Integrados de Gestión Bibliotecaria, conocidos como ISL o SIGB. Estos sistemas aparecieron con el fin de integrar funciones que antes estaban separadas, y permitían integrar en un mismo entorno la mayoría de las operaciones necesarias para el funcionamiento básico de las bibliotecas.

Con los avances informáticos, los ILS pasaron de ser sistemas locales a convertirse en herramientas más flexibles y complejas. Empezaron a incluir arquitecturas cliente-servidor y servicios en la nube lo que permitió a las bibliotecas mejorar significativamente su sistema de gestión [40]. Además, incorporaron en una sola plataforma sistemas de adquisición, catalogación y control de usuarios lo que optimizó mucho los procesos internos. Al mismo tiempo, comenzaron nuevas formas de ofrecer estos sistemas como los *hosted systems* o los servicios basados en suscripción (*SaaS*) lo que permitió que las bibliotecas eligieran la opción que mejor se adaptase a sus recursos [40].

Actualmente, dentro del ámbito de los ISL, existen dos grandes líneas tecnológicas: el software propietario y el software libre. La primera línea se caracteriza por depender de licencias comerciales y tener un mayor control por parte del proveedor [38] [40], algunos ejemplos representativos de este sistema son AbsysNet, SPX, Aleph o Symphony [41]. La segunda, sin embargo, ofrece un acceso abierto y libre al código fuente lo que permite que sea más adaptable y reduce los costes asociados [38] [40]. Entre los sistemas libre más conocidos están Koha, Evergreen y PMB [41].

3.2.4 ACTUALIDAD Y DIGITALIZACIÓN

Las nuevas tecnologías digitales han influido enormemente en la forma de acceder a la información y han redefinido el papel de las bibliotecas en la sociedad actual. Las bibliotecas han tenido que adaptarse y evolucionar hacia modelos que integran recursos digitales, permiten el acceso remoto y fomentan la comunicación entre sistemas, respondiendo de esta forma a las nuevas necesidades de los usuarios [42].

Actualmente, el concepto de biblioteca tradicional ha evolucionado. Las bibliotecas modernas ya no solo ofrecen acceso a ejemplares físicos, sino que también incluyen recursos digitales y servicios en línea [38] [42]. Esta transformación ha favorecido el desarrollo de catálogos accesibles por internet, bases de datos y plataformas que interactúan con otros sistemas. Sin embargo, este proceso no ha sido homogéneo [41]. La fragmentación entre los distintos sistemas, la falta de una plataforma centralizadora de la oferta bibliotecaria y la presencia de interfaces poco amigables siguen representando algunos de los desafíos más importantes que quedan por solucionar.

3.3 ANÁLISIS DE COMPETIDORES Y PLATAFORMAS EXISTENTES

Una vez revisada la evolución y transformación de los sistemas de gestión bibliotecaria a lo largo del tiempo, se examinarán a continuación qué soluciones tecnológicas existen en el mercado. Este apartado tiene como objetivo identificar las principales herramientas empleadas en el ámbito bibliotecario y analizar sus características más relevantes.

3.3.1 SOFTWARE PROPIETARIO

3.3.1.1 AbsysNet

AbsysNet [43] es un software propietario muy utilizado en bibliotecas públicas y universitarias en España [44] destinado a facilitar su gestión. Este sistema cuenta con una arquitectura modular que permite gestionar las principales funciones bibliotecarias. Además, incorpora un OPAC que facilita la consulta remota por parte de los usuarios y es compatible

con estándares internacionales [45]. Al tratarse de un software propietario es necesario adquirir una licencia para su uso e implica costes adicionales.

3.3.1.2 Aleph

Aleph [46] es un sistema integrado de gestión bibliotecaria (ISL) de carácter propietario. Está orientado a bibliotecas de investigación, académicas y nacionales. Alguna de sus características más relevantes es su gran flexibilidad, su escalabilidad, interfaz gráfica intuitiva y su facilidad para adaptarse tanto a bibliotecas pequeñas como a algunas de gran tamaño. Además, está disponible para distintos idiomas y cuenta con herramientas de administración avanzadas.

3.3.1.3 Symphony

Symphony [47] es un ILS implementado en bibliotecas públicas, académicas y escolares de todo el mundo. Se caracteriza por su gran flexibilidad, su gran personalización de los distintos módulos que ofrece y su entorno de trabajo estable. Además, gracias a su acceso a través de APIs y servicios web puede integrar servicios externos y lo convierte en un software potente.

3.3.2 SOFTWARE LIBRE

3.3.2.1 Koha

Koha [48] es un sistema integrado de gestión bibliotecaria *opensource* muy extendido. Ofrece funcionalidades avanzadas para la catalogación y es compatible con los principales estándares bibliotecarios. Además, permite acceder a su interfaz desde cualquier navegador y al ser de código abierto no depende de ningún proveedor lo que hace que se pueda gestionar sus datos libremente sin restricciones.

3.3.2.2 Evergreen

Evergreen [49] es un ISL de software libre que destaca por su gran escalabilidad. Es ideal para bibliotecas pequeñas y ofrece módulos para gestionar las adquisiciones, catalogación y circulación de materiales. Es compatible con los estándares MARC21 [50] y Z39.50 y al

tratarse de una herramienta *opensource* permite a las bibliotecas adaptar el sistema a sus necesidades.

3.3.2.3 PMB

PMB [51] es un ISL de código abierto enfocado en la accesibilidad digital. Esa herramienta ofrece una plataforma completa para la gestión de recursos físicos y digitales integrando funciones de gestión bibliotecaria avanzadas. Está basado en tecnologías web como PHP y MySQL y cumple con los estándares internaciones. Además, está muy presente en Francia, aunque se ha ido extendiendo su uso a nivel internacional.

3.3.3 PLATAFORMAS DE ACCESO REMOTO

3.3.3.1 WorldCat

WorldCat [52] es una plataforma cooperativa que permite localizar libros, artículos y recursos bibliotecarios. Facilita tanto la búsqueda de materiales físicos como el acceso a contenidos digitales. Al igual que las herramientas mencionadas previamente, es compatible con los estándares MARC21 y Z39.50 y se basa en la colaboración entre instituciones.

3.3.3.2 eBiblio

eBiblio [53] es una plataforma de préstamo de libros electrónicos impulsada por el Ministerio de Cultura del Gobierno de España en colaboración con las comunidades autónomas. Esta plataforma permite a los usuarios de las redes de bibliotecas públicas acceder gratuitamente a una gran colección de contenidos digitales. Además, este servicio promueve el acceso a la cultura y la divulgación del conocimiento.

3.3.3.3 OverDrive

OverDrive [54] es una plataforma digital que da servicios de préstamo de ejemplares y libros electrónicos a través de bibliotecas públicas, escolares y universitarias. Permite acceder de forma remota a estos recursos y promueve la lectura digital y el acceso al conocimiento.

3.4 ANÁLISIS COMPARATIVO ENTRE PLATAFORMAS EXISTENTES Y LA PROPUESTA DESARROLLADA

En el apartado anterior se han descrito las principales herramientas disponibles en el ámbito de gestión y acceso a recursos bibliotecarios. En este apartado, se compararán las plataformas descritas anteriormente con la propuesta desarrollada en este proyecto. El objetivo será identificar las principales diferencias en términos de funcionalidades y capacidad para valorar en qué medida el sistema propuesto aporta mejoras al mercado y cubre las necesidades de los usuarios.

En la Tabla 3 se presenta un análisis comparativo entre las principales plataformas existentes y *BiblioTrail*, el proyecto propuesto.

La tabla comparativa realizada muestra cómo la propuesta desarrollada integra un único entorno para gestionar recursos tanto físicos como digitales. Además, cuenta con funcionalidades como la búsqueda unificada entre bibliotecas independientes accediendo a una web con una arquitectura moderna y abierta. A pesar de que plataformas potentes y consolidadas como WorldCat o OverDrive cuentan con mayores infraestructuras ninguna de ellas combina de forma completa los servicios de localización, reserva, préstamo físico y digital en un entorno centralizado y gratuito.

Tecnología base	Cliente-servidor tradicional, web limitada	Cliente-servidor modular	Cliente-servidor moderno, APIs disponibles	PHP + MySQL, completamente web	Perl + PostgreSQL	PHP + MySQL	Sistemas propietarios de OCLC	Plataforma propia de gestión digital	Plataforma privada para ebooks	Django (Python) + PostgreSQL APIs REST
Arquitectura	Local o en servidor propio	Modular local y red	Local o Symphony Web	Web pura, servidor LAMP	Web modular, API REST avanzada	Web modular	Infraestructura en nube centralizada	Web y app móvil	Web y app móvil	Web modular, API-first, microservicios
Software Libre/Propietario	Propietario	Propietario	Propietario	Libre	Libre	Libre	Propietario	Propietario	Propietario	Libre
Gestión de recursos físicos	Completa pero clásica	Completa y robusta	Completa	Completa	Completa y flexible	Completa	No gestiona, solo muestra registros	No aplica	No aplica	Completa, adaptada a catálogos de múltiples bibliotecas
Gestión de contenidos digitales	Limitada	Limitada	Limitada, necesita extensiones	Parcialmente soportada	Integrada	Integrada	No gestiona contenido	Solo contenido digital	Solo contenido digital	Integración física y digital en una única plataforma
Sistema propio de reservas	Local por biblioteca	Local por biblioteca	Local configurable	Integrado	Integrado	Integrado	No	Gestión de reservas digitales	Gestión en app móvil	Unificado para múltiples bibliotecas
Búsqueda centralizada entre bibliotecas	No soportada	No soportada	No soportada	No soportada	No soportada	No soportada	Sí	No	No	Sí, búsqueda única y agregada entre catálogos independientes
Integración multi-bilbioteca	No soportada	Parcial en consorcios	Parcial en consorcios	No nativo	Parcialmente mediante cooperación	No de forma nativa	Sí	No	No	Sí, agregación funcional con reservas y préstamos
Interfaz web intuitiva y modificada	Limitada, estructura clásica	Limitada, pensada para personal técnico	Mejorada	Moderna y adaptable	Moderna y optimizada	Moderna	Aceptable	Moderna y accesible en web y app	Moderna	Interfaz única, intuitiva accesible y responsive

Tabla 3.- Tabla comparativa competidores (elaboración propia)

3.5 DESAFÍOS ACTUALES

Aunque se han logrado grandes avances en las últimas décadas en la digitalización de los servicios bibliotecarios, aún se presentan retos que afectan a la calidad del acceso a la información y a la eficacia en la gestión de estas instituciones.

Desde el punto de vista de gestión de recursos electrónicos, uno de los principales retos es la restricción comercial. Las bibliotecas se enfrentan a altos costes de licencias y a limitaciones impuestas por los editores. Además, los sistemas de gestión de derechos digitales (DRM) restringen el control sobre los recursos adquiridos y generan una dependencia tecnológica que no existía en el modelo de préstamo tradicional [55].

Otro desafío importante es la fragmentación de los sistemas [56]. Muchas bibliotecas actuales siguen operando de manera aislada y no permiten la consulta de forma unificada ni la comunicación entre instituciones. A este problema se le suma la falta de interoperabilidad tecnológica, lo que complica aún más el intercambio de recursos y la construcción de redes bibliotecarias cooperativas.

Por otro lado, otro problema persistente es la ausencia de plataformas que no solo agreguen registros bibliográficos, sino que también integren funcionalidades como la reserva o el préstamo centralizado de documentos.

La accesibilidad de los sistemas también continúa siendo un área a mejorar. Muchos catálogos y servicios digitales siguen teniendo estructuras de navegación e interfaces complejas, lo que desincentiva su uso [42] [56].

Frente a estos desafíos, surge la necesidad de mejorar los modelos de gestión, sistemas de préstamo y arquitecturas actuales, y orientarlos hacia soluciones más abiertas, accesibles y eficientes que no dependan de proveedores cerrados y den prioridad a las necesidades de los usuarios.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

Como se explicó en el apartado 3.5 en la actualidad a pesar de haber logrado increíbles avances en la digitalización de los servicios bibliotecarios, el acceso a los catálogos sigue siendo un proceso poco centralizado. Cada biblioteca gestiona su catálogo de forma independiente, con su propia página web, base de datos y sistema de consulta, sin embargo, no existe una interfaz común que facilite la búsqueda de información en varias instituciones al mismo tiempo. Esto obliga a que los usuarios tengan que consultar manualmente cada plataforma para localizar un recurso, lo que dificulta el acceso rápido a la información.

BiblioTrail es una propuesta que surge para resolver este problema. Se trata de una plataforma web capaz de agregar los catálogos de múltiples instituciones independientes mediante consultas a APIs REST y unifica los resultados para el usuario en un mismo lugar sin tener que alterar la infraestructura de cada institución. De esta manera se ofrece al usuario una experiencia de consulta más eficiente, accesible y que al mismo tiempo respeta la independencia de cada biblioteca asociada.

Este proyecto aporta un enfoque innovador y facilita el acceso a la información bibliográfica. Además, se apoya en tecnologías como las APIs RESTful que favorecen la comunicación e interoperabilidad entre instituciones. Gracias a su estructura modular e independiente, no solo resulta aplicable al ámbito bibliotecario, sino que también podría escalarse a otros tipos de instituciones como archivos, centros documentales o a cualquier organismo que gestione recursos de información. Por tanto, esta propuesta no solo da respuesta a una necesidad real, sino que también puede ser adoptada en ámbitos educativos, públicos y privados.

4.2 OBJETIVOS

Como se ha señalado en los capítulos anteriores, el objetivo general de este proyecto consiste en desarrollar una plataforma web que permita a los usuarios consultar y utilizar los recursos de distintas bibliotecas de una manera centralizada, manteniendo la independencia entre instituciones, pero brindándole al usuario un único lugar desde el que realizar las consultas.

Inicialmente, en la propuesta del proyecto, se definieron unos objetivos preliminares orientados a la construcción de una plataforma web básica. Sin embargo, a lo largo del desarrollo del proyecto, se han ido ajustando estos objetivos y han evolucionado ligeramente para adaptarse mejor a las necesidades reales detectadas.

Para conseguirlo, se han definido objetivos principales y objetivos secundarios, que se explican con más detalle a continuación.

4.2.1 OBJETIVOS PRINCIPALES

Los objetivos principales representan los pilares fundamentales sobre los que se basa el proyecto.

- Diseño y desarrollo de una página web funcional y accesible, que facilite la gestión bibliotecaria y permita a los usuarios acceder a la información y los recursos de forma eficiente.
- Creación de un front-end moderno e intuitivo que facilite la interacción del usuario con la plataforma y garantice una experiencia de usuario satisfactoria.

4.2.2 OBJETIVOS SECUNDARIOS

Los objetivos secundarios se han establecido con el fin de complementar y fortalecer el cumplimiento de los objetivos principales.

- Implementar un sistema de reservas y gestiones de préstamos, así como de inscripciones a eventos y reserva de espacios.

- Unificar y presentar de manera integrada los datos provenientes de distintas instituciones en un mismo lugar mediante la consulta de APIs externas.
- Fomentar la interoperabilidad entre las distintas bibliotecas asociadas a través del uso de APIs RESTful.
- Garantizar la escalabilidad y modularidad de la plataforma.
- Gestionar de forma segura los usuarios, protegiendo sus datos personales y controlando a qué partes de la aplicación pueden acceder en función de los permisos establecidos.

Los objetivos definidos han servido de guía para desarrollar este proyecto y se verán reflejados en los distintos capítulos de la memoria, donde se detalla cómo se ha trabajado para conseguirlos.

4.3 METODOLOGÍA DE TRABAJO Y TECNOLOGÍAS UTILIZADAS

4.3.1 METODOLOGÍA DE TRABAJO

El desarrollo de este proyecto se ha basado en una metodología con enfoque ágil [57] adaptado a las características de un trabajo individual. Se llevaron a cabo reuniones semanales de revisión y planificación con el director del TFG. Además, el proyecto se ha desarrollado siguiendo principios de un trabajo iterativo e incremental, permitiendo que se realice una adaptación de manera progresiva de las tareas y objetivos en función de los avances que se han ido obteniendo en cada fase.

4.3.2 MARCO DE DESARROLLO Y LENGUAJES UTILIZADOS

Para el desarrollo de este proyecto se ha seguido un enfoque modular y se han utilizado las siguientes tecnologías:

- Para el front-end y diseño visual de la página se utilizaron lenguajes como HTML, CSS y JavaScript. Además, se hizo uso del framework Bootstrap para facilitar el diseño responsive y agilizar la creación de la interfaz.

- Para el back-end se utilizó Python como lenguaje principal, integrado ya en el framework de Django, en el que se ha desarrollado la lógica del trabajo, la estructura de las APIs de cada biblioteca y la gestión de datos del proyecto.
- Como gestor de base de datos se ha utilizado SQLite para el entorno de desarrollo local y posteriormente PostgreSQL.
- La ejecución y validación de la plataforma se realizó en un entorno de desarrollo local, simulando múltiples servidores de bibliotecas independientes, conectados a la plataforma agregadora a través de APIs REST.

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

4.4.1 PLANIFICACIÓN

Para planificar el proyecto, se elaboró el siguiente cronograma estimado al inicio del trabajo. Este cronograma, sirvió de guía orientativo para estructurar las fases del trabajo. En la tabla mostrada a continuación se puede observar la planificación inicial propuesta.

	Nov				Dic				Ene				Feb				Mar				Abr				May				Jun			
Número de semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Revisión de documentación																																
Definir requisitos del proyecto																																
Aprendizaje de Python y Django																																
Diseño de la arquitectura																																
Estructuración del diseño web																																
Desarrollo del backend																																
Implementación de la base de datos																																
Desarrollo del frontend																																
Desarrollo de objetivos secundarios																																
Corrección de errores																																
Ajustes finales y pruebas																																
Redacción del TFG																																

Tabla 4.- Diagrama de Gantt para planificación del proyecto

Como se puede observar en la Tabla 4, el primer mes se dedicó al aprendizaje del entorno de desarrollo, así como de los lenguajes de programación necesarios para llevar a cabo el proyecto. El diseño de la arquitectura y estructuración general del proyecto se realizaron en el mes de diciembre, momento a partir del cual se comenzó con el desarrollo del back-end.

Al comienzo del trabajo, se empezó desarrollando una *primera versión* de la plataforma web. Estas primeras etapas de desarrollo contaban con una base de datos sencilla de SQLite, adecuada para esta fase inicial. Más adelante, se comenzó el diseño del front-end y la implementación de las principales funcionalidades de la plataforma. Estas tareas ocuparon principalmente los meses de febrero, marzo y abril.

Por último, en las últimas semanas de abril, se llevó a cabo la migración de base de datos a PostgreSQL, un gestor mucho más potente y escalable. Posteriormente, se comenzó la etapa de ajustes y corrección de errores. La redacción del TFG se ha realizado de forma paralela al desarrollo de este trabajo.

4.4.2 ESTIMACIÓN ECONÓMICA

Aunque este trabajo no haya supuesto un desembolso económico directo, se ha realizado una valoración aproximada sobre los recursos empleados en su desarrollo. Este análisis considera aspectos como las herramientas de software utilizadas, el tiempo dedicado al proyecto y el equipamiento técnico necesario para llevarlo a cabo. De esta forma, se ofrece una estimación del esfuerzo y medios necesarios para implementar el proyecto y así, dimensionar el coste.

4.4.2.1 Herramientas y tecnologías

Para desarrollar este trabajo se han utilizado las siguientes plataformas y aplicaciones, todas ellas gratuitas:

- Django y DRF como herramientas base del back-end de la plataforma.
- Visual Studio Code como editor de código principal.
- PostgreSQL como gestor de la base de datos.
- PgAdmin como herramienta para administrar la base de datos.

4.4.2.2 Equipamiento

Para implementar el proyecto se ha utilizado el siguiente material informático:

- Ordenador portátil: modelo *HP laptop 15s-fq2xxx* con un precio de 699€

Considerando que tiene una vida útil de 6 años (72 meses) y un uso efectivo de 9 meses:

Primero, se ha calculado el coste mensual:

$$Coste\ mensual = \left(\frac{699\ €}{72\ meses}\right) = 9{,}71€$$

Por tanto, el coste imputable al proyecto por 9 meses es de:

$$9{,}71€ \times 9 = 87{,}39€$$

4.4.2.3 Estimación de horas de trabajo

El proyecto ha requerido una dedicación aproximada de 700 horas. Tomando como referencia el salario medio aproximado de un ingeniero junior en España (15€/hora) [58], el coste equivalente en términos de capital humano se estima en:

$$700 \times 15 = 10500€$$

4.4.2.4 Resumen de costes estimados

A continuación, se presenta un resumen de los principales costes estimados del proyecto, considerando tanto el equipamiento como las horas de trabajo dedicadas.

Concepto	***Valor estimado***
Amortización ordenador portátil (9 meses)	87,29 €
Valoración horas de trabajo (700h * 15€/h)	10.500,00 €
Total estimado	10.587,29 €

Tabla 5.- Estimación económica

Capítulo 5. SISTEMA/MODELO DESARROLLADO

A lo largo de este capítulo se explicará el funcionamiento del sistema desarrollado. El proyecto consiste en el desarrollo de una plataforma web agregadora, denominada BiblioTrail, que centraliza el acceso a los catálogos y servicios de múltiples bibliotecas en un mismo sitio web. Esta plataforma permite que los usuarios puedan consultar libros, realizar reservas de espacios de trabajo e inscribirse a eventos desde una única interfaz.

Este capítulo se dividirá en tres secciones. En primer lugar, se detallarán los requisitos propuestos (funcionales y no funcionales) que debe cumplir la plataforma. Más adelante, se explicará el diseño de la arquitectura, la organización del proyecto y separación en módulos, y el diseño de la interfaz de usuario. Por último, se detallará el proceso en el que se han ido implementando las distintas funcionalidades incluyendo la gestión de usuarios, la conexión con las API REST y la configuración del entorno.

5.1 ANÁLISIS DEL SISTEMA

5.1.1 CONTEXTO Y ALCANCE

Como se explicó en el Capítulo 3 y Capítulo 4 actualmente muchas bibliotecas funcionan de manera independiente, usando sus propios sistemas de reserva y gestión de usuarios. Esto puede suponer una gran barrera para los usuarios. Cuando necesitan acceder o buscar un recurso concreto, muchas veces se ven obligados a buscar en catálogos por separado o registrarse en bibliotecas de manera individual. Esto hace que el proceso de búsqueda y acceso a los recursos no sea eficiente y puede desincentivar a su uso.

BiblioTrail es una plataforma desarrollada justamente para hacer más eficiente este proceso y busca que, a través de una plataforma agregadora, se pueda centralizar el acceso a múltiples

bibliotecas desde un mismo lugar. Esto permite a los usuarios consultar catálogos, reservar ejemplares, inscribirse a eventos y gestionar sus préstamos en un mismo entorno.

El alcance del proyecto incluye:

- La construcción de la arquitectura de la plataforma agregadora y sus módulos.
- La integración de dos bibliotecas externas, simuladas en entornos independientes del sistema central.
- La implementación de funcionalidades como la búsqueda unificada de recursos.
- El diseño *responsive* e intuitivo de la página web.
- La gestión segura de usuarios y recursos.

Sin embargo, quedan fuera del alcance:

- La gestión directa de los contenidos internos de cada una de las bibliotecas asociadas (catálogos, eventos, etc.) que son responsabilidad de cada institución.
- Acceso al panel de administración de cada biblioteca.

5.1.2 ANÁLISIS DE REQUISITOS

Antes de empezar con el diseño, se realizó un análisis para tener claro qué se esperaba de él. Se detalló qué debía hacer y cómo debía comportarse. Para ello, en esta sección se recogen los requisitos que guiaron el desarrollo del proyecto, que se dividen en: requisitos funcionales, que describen qué debe de hacer el sistema, y, por otro lado, los no funcionales, que marcan cómo se debe de comportar el sistema en cuanto a seguridad, rendimiento y uso.

5.1.2.1 Requisitos funcionales

A continuación, se recogen los siguientes requisitos funcionales:

1. El sistema deber permitir a los usuarios buscar libros en los catálogos de distintas bibliotecas y presentarlos de una forma unificada.

2. El sistema debe mostrar los detalles de cada libro, junto con una lista de los ejemplares disponibles. Además, debe permitir reservar el ejemplar elegido, asignárselo al usuario y crear un préstamo bibliotecario de forma automática.
3. El sistema debe permitir visualizar en "Mis Reservas" en la sección de "Préstamos de libros" los préstamos activos del usuario, así como reflejar el estado actual del préstamo (activo, retrasado, reservas pasadas).
4. El sistema debe mostrar los eventos organizados por las bibliotecas asociadas y permitir al usuario acceder al calendario para ver una vista completa de los eventos.
5. El sistema debe permitirle al usuario inscribirse en un evento, siempre que haya plazas disponibles o no se haya inscrito ya en él.
6. El sistema debe mostrar la disponibilidad horaria de espacios de estudio, marcando en color rojo las franjas ocupadas y en verde las disponibles, así como permitir la reserva en función de la franja horaria escogida.
7. Además, en la sección de "Mis Reservas" el usuario autenticado podrá consultar las reservas asociadas con su perfil, con la posibilidad de cancelarlas si lo desea.
8. El sistema debe registrar correctamente a los nuevos usuarios que utilizan la plataforma por primera vez. Además, debe gestionar el inicio de sesión y la verificación de las credenciales del usuario ya registrado.
9. Las plataformas de cada biblioteca deben de ser independientes del sistema central y se comunicarán con él mediante solicitudes a sus propias APIs. De esta forma se conservará la independencia y seguridad de cada plataforma.
10. El sistema debe interactuar y obtener los datos de APIs externas de cada una de las bibliotecas asociadas. También tiene que ser capaz de sincronizar los datos de disponibilidad y reservas.
11. El usuario autenticado podrá consultar y editar su perfil directamente en la sección de "Mi perfil".
12. Dentro de los usuarios autenticados, el sistema debe ser capaz de diferenciar entre los distintos roles (usuario autenticado y bibliotecario), en función de sus permisos, que únicamente podrán darse por un administrador a través del panel de administración.

13. Los empleados o bibliotecarios poseen permisos adicionales, y podrán modificar datos de los usuarios y de la base de datos de la plataforma central.
14. En el sistema se debe poder acceder al panel de administración, capaz de manejar internamente los datos guardados en la plataforma, si el usuario posee los permisos correspondientes.
15. Cualquier tipo de usuario, autenticado o visitante, podrá contactar con el soporte técnico de la aplicación si necesita reportar alguna incidencia o manda algún tipo de pregunta o consulta.

5.1.2.2 Requisitos no funcionales

Por otro lado, a continuación, se recogen los siguientes requisitos no funcionales:

1. El sistema debe de ser accesible desde navegadores web y móviles, además, debe modificar la disposición de los elementos de la interfaz gráfica al tipo de dispositivo.
2. La interfaz gráfica debe ser intuitiva, *responsive* y clara.
3. La plataforma debe dividirse en módulos, lo más independientes posible, de forma que al añadir cambios no se tenga que modificar toda la lógica central.
4. Las contraseñas deben de almacenarse y cifrarse de forma segura en el sistema.
5. El sistema debe incluir controles de seguridad para los campos del formulario de registro e inicio de sesión.
6. El acceso a ciertas secciones de la aplicación estará limitado. Los usuarios necesitarán autenticarse con sus credenciales para poder acceder a las funcionalidades más importantes.
7. El sistema será escalable, y debe permitir futuras ampliaciones o añadir más funcionalidades (o bibliotecas).
8. El sistema debe manejar errores de red o fallos de las APIs externas.

5.1.3 CASOS DE USO

5.1.3.1 Actores

En este proyecto se han identificado tres actores principales:

- **Usuario no autenticado**: persona que aún no se ha identificado en la plataforma, ya sea por haberse conectado por primera vez o no haber iniciado sesión. Este usuario puede navegar y acceder únicamente a las vistas informativas de la aplicación. Sin embargo, si intenta realizar alguna reserva o acceder a secciones restringidas necesitará iniciar sesión o registrarse.

  Entre sus acciones dentro de la plataforma se encuentran:

  - Visualizar información general en la página de inicio.
  - Consultar el catálogo bibliotecario y realizar búsquedas (sin opción de realizar reservas de ejemplares, ni visualizar la vista detallada).
  - Consultar los datos de las bibliotecas asociadas.
  - Ver el listado de eventos programados (sin opción de inscribirse).

- **Usuario autenticado**: persona que se ha registrado previamente en la plataforma y ha iniciado sesión con sus credenciales correctamente. Una vez autenticado, puede navegar, realizar y modificar reservas o datos de su perfil.

  Este tipo de usuario puede:

  - Realizar búsquedas de libros y reservar ejemplares si están disponibles.
  - Inscribirse a eventos.
  - Enviar consultas o incidencias al soporte de la plataforma mediante el formulario de contacto.
  - Consultar la disponibilidad de los espacios de trabajo y realizar reservas.
  - Consultar y modificar su perfil de usuario, así como datos personales.
  - Cerrar sesión.
  - Consultar y gestionar sus propias reservas en la sección de “Mis Reservas”:
    - Revisar préstamos de libros.
    - Consultar o cancelar inscripciones a eventos.
    - Consultar o cancelar reservas de espacios de trabajo.

- **Bibliotecario**: empleado o administrador de la biblioteca con acceso al panel de administración. Cuenta con permisos para consultar y modificar préstamos bibliotecarios, gestionar devoluciones y modificar datos de los usuarios registrados en la plataforma.

Entre sus acciones dentro de la plataforma se encuentran:

- Mismos permisos que un usuario autenticado.
- Gestionar datos de los usuarios.
- Gestionar préstamos y devoluciones físicas.

5.1.3.2 Casos de uso

En esta sección se detallan los casos de uso que intervienen en el proyecto para los autores mencionados en el anterior apartado (usuario no autenticado, usuario autenticado y bibliotecario). Los casos de uso permiten entender mejor la funcionalidad del sistema diseñado.

A continuación, en la siguiente tabla se muestra de forma resumida cada uno de los casos de uso identificados:

Caso de uso	Actor principal	Descripción
Navegar sitio	Usuario no autenticado / Usuario autenticado	Consultar y navegar por las secciones informativas del sitio web
Iniciar sesión / Registrarse	Usuario / Bibliotecario	Acceder al sistema con creden-ciales o registrarse como usuario.
Cerrar sesión	Usuario / Bibliotecario	Finalizar la sesión actual.
Consultar perfil	Usuario autenticado	Visualizar los datos de perfil del usuario.
Editar perfil	Usuario autenticado	Modificar los datos del perfil del usuario.
Buscar libros / Consultar catálogo	Todos	Filtrar y explorar el catálogo por título, autor, biblioteca o género.
Reservar ejemplar	Usuario autenticado	Solicitar el préstamo de un ejemplar disponible.
Filtrar disponibi-lidad de espacios	Todos	Consultar las salas disponibles por fecha, capacidad y biblioteca.
Reservar espacio de trabajo	Usuario autenticado	Realizar una reserva de sala en un horario específico.
Inscribirse a evento	Usuario autenticado	Registrarse en un evento programado con plazas disponibles.
Consultar "Mis Reservas"	Usuario autenticado	Gestionar los préstamos, reservas de salas e inscripciones a eventos.
Enviar mensaje de contacto	Todos	Rellenar un formulario para contactar con la administración o soporte.
Gestionar préstamos	Bibliotecario	Registrar y modificar el estado de los préstamos realizados a usuarios.
Gestionar devolu-ciones físicas	Bibliotecario	Procesar la devolución de ejemplares físicos en mostrador.
Gestionar panel de administrador	Bibliotecario	Acceder a funcionalidades avanzadas de administración y base de datos.

Tabla 6.- Resumen casos de uso (elaboración propia)

Para complementar la anterior tabla, en el Anexo III se detallan cada uno de los casos de uso más relevantes del sistema en fichas. Estas fichas incluyen información como los actores que intervienen en el caso de uso, una descripción detallada del caso, las precondiciones, las postcondiciones, la secuencia normal de ese caso y excepciones (si existen).

5.1.3.3 Diagrama de casos de uso

En la siguiente figura se muestra el diagrama de casos de uso:

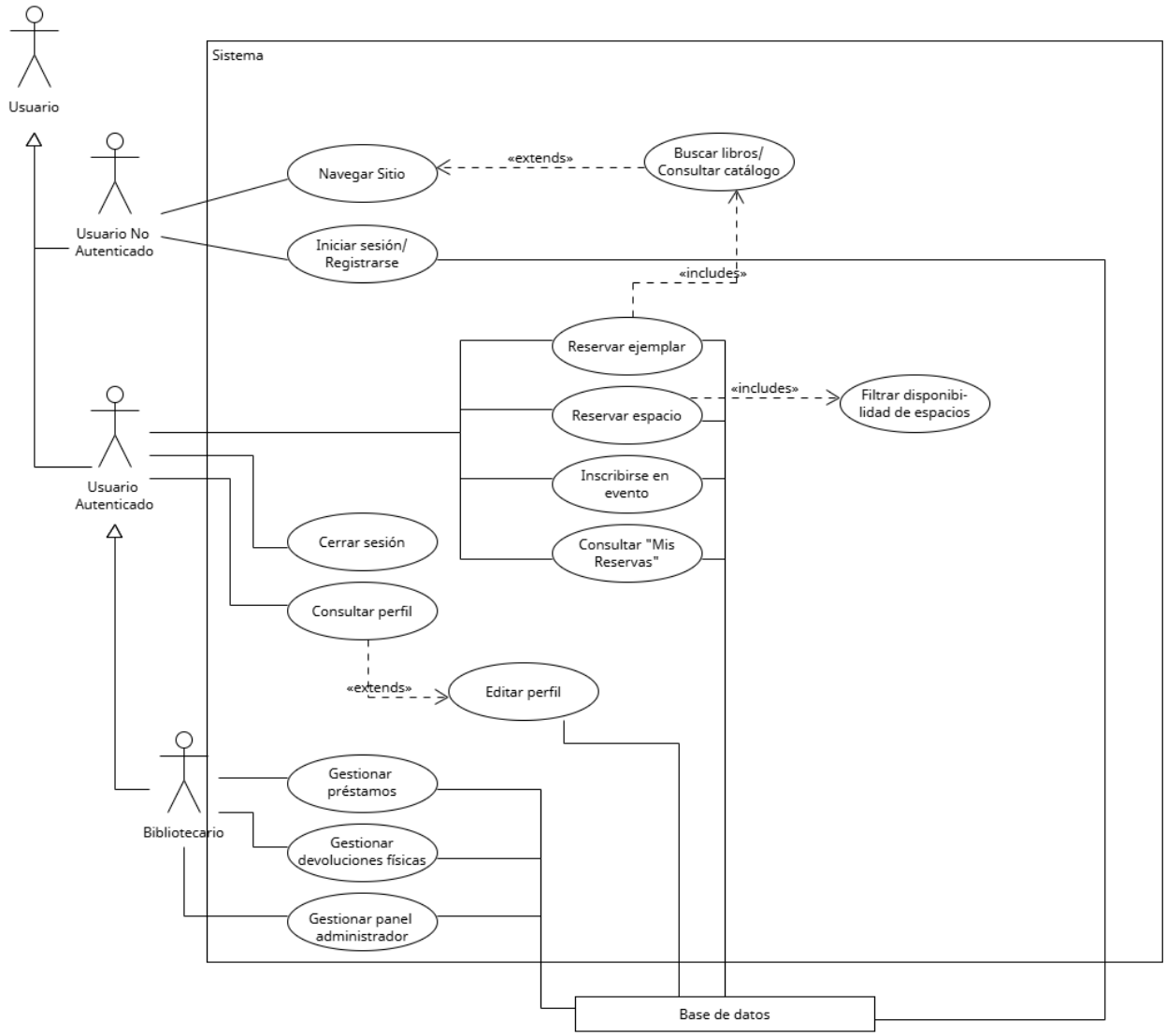


Figura 5.- Diagrama de casos de uso UML

En la Figura 5 representa los casos de uso descritos previamente y su relación con los actores que intervienen en el sistema. Cada óvalo representa un caso de uso y las líneas indican qué acciones puede realizar cada actor del sistema.

En el diagrama se distingue entre usuarios no autenticados y autenticados. Los primeros tienen un acceso más limitado a la plataforma. Pueden navegar, buscar libros, iniciar sesión o registrarse. Los segundos, al haberse autenticado, tienen más funcionalidades disponibles

como reservar ejemplares, espacios, inscribirse a eventos, gestionar reservas, consultar y editar el perfil o cerrar sesión.

Algunos usuarios autenticados tienen permisos de empleado o bibliotecario. Estos pueden acceder a más funcionalidades como la gestión de préstamos, devoluciones físicas y controlar el panel de administración.

Con este diagrama se consigue visualizar mejor las interacciones que pueden realizar los distintos actores con la plataforma, así como comprender mejor sus distintas funcionalidades.

5.2 DISEÑO

5.2.1 ARQUITECTURA

5.2.1.1 Arquitectura modular y de cliente-servidor

El proyecto tiene una arquitectura distribuida de tipo cliente-servidor. En este tipo de arquitectura intervienen: el cliente, que es quien hace las peticiones al sistema e interactúa desde el navegador web a la interfaz del sistema. Por otro lado, el servidor, desarrollado en Django, es el que recibe las peticiones realizadas por el cliente, las procesa, y devuelve una respuesta.

La separación entre el cliente y el servidor permite organizar mejor el sistema, facilita su escalabilidad y permite que tenga una estructura modular. En el proyecto intervienen:

- El usuario, que interactúa con la aplicación y realiza peticiones HTTP al sistema central.
- El servidor Django, que recibe las peticiones HTTP realizadas por el usuario, procesa la información, accede a la base de datos, y se comunica con las APIs de las bibliotecas externas.
- La base de datos local del sistema central, diseñada en PostgreSQL, guarda toda la información de la plataforma agregadora referente a los usuarios que se dan de alta

en el sitio web, reservas, inscripciones y préstamos bibliotecarios. Este componente no guarda información de las bibliotecas asociadas con la plataforma, sino que accede a sus datos a través de APIs REST que ellas exponen.

- Las bibliotecas externas son entidades independientes que intercambian información con el servidor Django de la plataforma agregadora, exponen sus catálogos, libros e información a través de sus endpoints REST. Además, cada una de ellas se comunica con su base de datos local, donde se guarda toda la información referente a la biblioteca.

En el siguiente diagrama se puede ver más en detalle cómo funciona la arquitectura del proyecto y la comunicación entre cada uno de sus componentes.

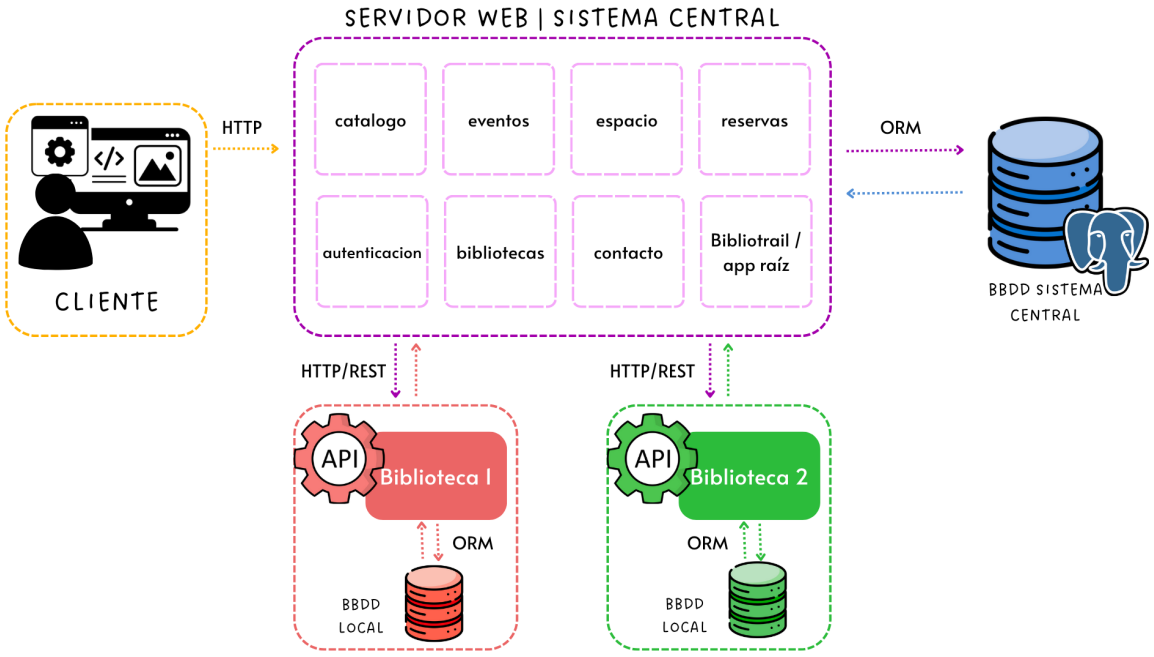


Figura 6.- Diagrama de componentes de la arquitectura

5.2.1.2 Flujo de comunicación entre componentes

Como se muestra en la Figura 6, los componentes se comunican entre sí intercambiando información entre cada uno de ellos. El cliente se comunica con la aplicación principal

mediante solicitudes HTTP y el servidor actúa como punto de entrada, comunicándose con el resto de los componentes.

Cuando un usuario realiza una acción (buscar un libro, ver evento o realizar una reserva), el servidor interpreta su solicitud. En función de ella existen dos opciones:

- La primera es que acceda a la base de datos central. Esto sucede si se trata de información que se gestione de forma local, es decir, todo lo que implique una intervención del usuario en la plataforma como, por ejemplo: reservar ejemplares, inscribirse a eventos, reservar salas de estudio, modificar datos del perfil del usuario, crear préstamos de libros asociados al usuario, etc.
- La segunda opción consiste en que en su lugar se lance una consulta a una API externa de la biblioteca. Esto sucede siempre que se necesite obtener información actualizada sobre los recursos, libros, disponibilidad o datos asociados con cada biblioteca externa.

Las bibliotecas externas son independientes del sistema central. Se comunican con él a través de APIs REST. Estas APIs se encargan de recibir solicitudes del agregador central y de responder con los datos solicitados, devolviéndoselos en formato JSON. Internamente, consultan únicamente sus propias bases de datos de forma local. Por tanto, el servidor de BiblioTrail no accede nunca directamente a estas bases de datos locales, sino que se comunica con unos endpoints REST definidos.

De esta forma, se consigue separar las lógicas internas de cada biblioteca y se asegura que los sistemas son independientes los unos de los otros, lo que facilita su mantenimiento. Además, esta estrategia ayuda a que si en un futuro se quieren añadir nuevas funcionalidades no se tenga que cambiar toda la lógica base.

5.2.1.3 Módulos del servidor central

El sistema central de BiblioTrail, al estar desarrollado en Django, se ha dividido en múltiples aplicaciones o apps. Cada una, está especializada en una parte concentra del sistema. La

ventaja de este diseño modular es que facilita en gran parte el desarrollo y escalabilidad del sistema.

Los principales módulos de este sistema principal son:

- Catálogo (`catalogo`): Esta app se encarga de realizar búsquedas en las bibliotecas externas. Para ello, consulta las APIs que exponen las bibliotecas y de esta forma obtiene los libros, autores, ejemplares, etc. Con esta información el servidor ya es capaz de procesar los datos y presentársela de vuelta al cliente.
- Eventos (`eventos`): Su función es gestionar la visualización de eventos ofrecidos por las bibliotecas en el sitio web. Además, registra las inscripciones de los usuarios y sincroniza los datos con las APIs remotas para que actualicen su información.
- Espacios (`espacios`): Esta aplicación muestra la disponibilidad de los espacios de trabajo en función de la franja horaria. Para ello, al igual que en los módulos anteriores, se accede a la API y obtiene la información de las bibliotecas. Además, para que la funcionalidad de reserva de salas funcione correctamente, trabaja conjuntamente con la app de reservas.
- Reservas (`reservas`): Este módulo guarda todas las acciones relacionadas con las reservas del usuario. Es decir, cuando el usuario pide un préstamo de un ejemplar, reserva un espacio o se inscribe en algún evento, la app es la encargada de guardarlo en la base de datos. Además, también se encarga de gestionar las cancelaciones.
- Autenticación (`autenticacion`): Esta app se encarga de gestionar la autenticación, el registro, el inicio de sesión, los perfiles y datos del usuario y el control de acceso de los usuarios.
- BibliotrailApp (`BibliotrailApp`): Esta aplicación centraliza las plantillas base y de inicio del sitio web del proyecto. De esta plantilla base es de la que heredan el resto de las plantillas para que el proyecto tenga la misma estructura y estilo visual.
- Bibliotrail: Es el módulo raíz del proyecto de Django. Se genera de forma automática al crear el proyecto. Contiene la configuración central (`settings.py`), rutas (`urls.py`) y archivos para que la aplicación funcione.

5.2.2 MODELO DE DATOS

El modelo de datos define cómo se guarda y organiza la información en el sistema. En este proyecto se han creado modelos que representan los distintos recursos de las bibliotecas, como los préstamos de libros, ejemplares, salas de trabajo, eventos, perfiles y datos de los usuarios, etc. Por tanto, los modelos se podrían entender como la estructura en la que se desean guardar los datos en la base de datos.

Django utiliza un sistema ORM (Mapeador Relacional de Objetos). Este sistema permite definir los modelos como clases de Python. Esto significa que en lugar de escribir directamente consultas SQL, se definen clases que representan las tablas. Cada clase es un modelo, y cada atributo, un campo dentro de la tabla. De esta forma, se consigue estructurar la información y manejar los datos de forma más sencilla.

Los campos de los modelos de datos pueden estar relacionados. Por ejemplo, un préstamo está vinculado con un ejemplar y con un usuario. Django permite establecer distintos tipos de relaciones:

- ForeignKey: representa una relación de muchos a uno.
- ManyToManyField: representa una relación de muchos a muchos.
- OneToOneField: representa una relación de uno a uno.

Django además permite tener la información de la base de datos actualizada. La manera de hacerlo es mediante las migraciones, que permiten aplicar cambios sin perder los datos. Además, permite añadir validaciones automáticas para asegurarse de que la estructura en la que se guardan los datos es correcta. Esto hace que sea un sistema sea más robusto y facilite futuras ampliaciones.

5.2.2.1 Modelos del sistema central

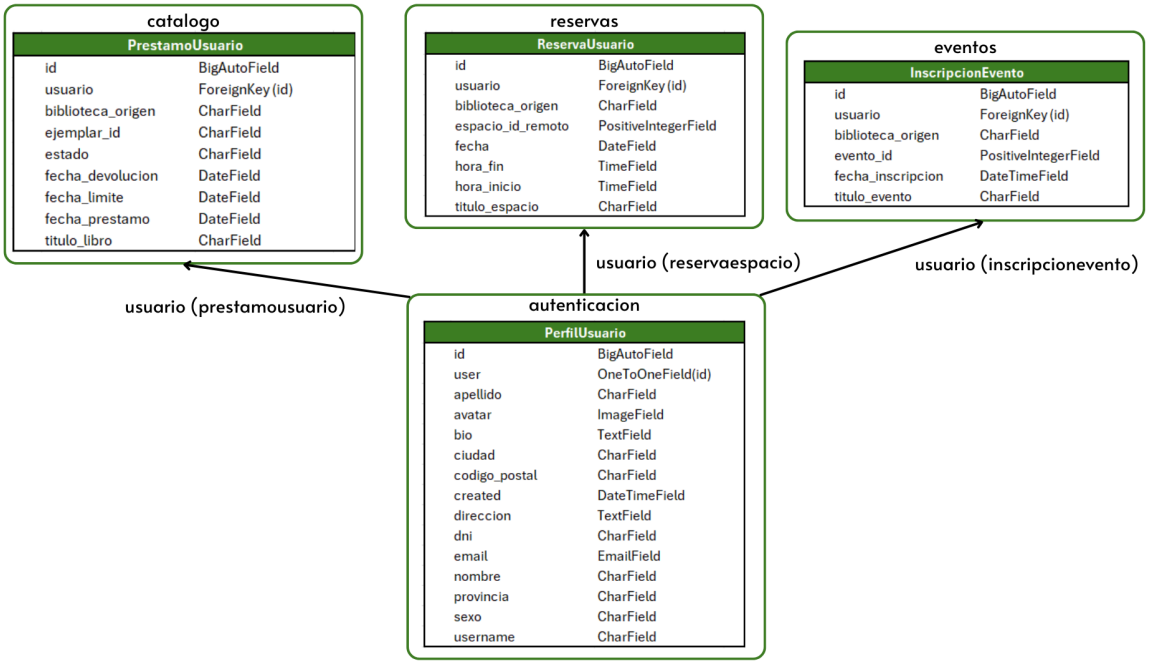


Figura 7.- Diagrama de clases modelo central

En la Figura 7 se puede ver el diagrama de clases que muestra las relaciones entre los modelos más relevantes de la plataforma central. A continuación, se enumeran y detallan los modelos que forman parte del sistema principal:

- PerfilUsuario: Este modelo es una ampliación del modelo de usuario por defecto de Django. Se creó con el fin de guardar datos adicionales como el nombre completo o los datos de contacto. Además, está relacionado con el catálogo, las reservas y los eventos.
- PrestamoUsuario: Se encarga de guardar los préstamos realizados por los usuarios a través del sitio web. Como los ejemplares están en las bibliotecas externas, no guarda los datos procedentes de ellas, sino la información asociada al préstamo como: el título del libro prestado, su identificador del ejemplar, la biblioteca en la que se encuentra y las fechas de préstamo y devolución.
- ReservaEspacio: Se encarga de guardar las reservas de los espacios de trabajo que realizan los usuarios en la plataforma. Contiene distintos campos como: el nombre

del espacio, la biblioteca origen, la fecha, la hora y el identificador. Este modelo vincula el usuario con su reserva para que pueda visualizarlo en la sección de "Mis asientos".

- InscripcionEvento: Este modelo guarda las inscripciones a eventos. Guarda datos como el título del evento, la localización, la fecha y el identificador del evento. Al inscribirse un usuario, el sistema actualiza el número de plazas llamando a la API de la biblioteca a la que corresponde.

5.2.2.2 Modelos de bibliotecas asociadas

Las bibliotecas externas (biblioteca1 y biblioteca2) tienen su propio sistema independiente y su propia base de datos local. Sin embargo, la estructura y lógica es idéntica, lo que hace que sea fácil de integrar en la plataforma.

En la Figura 8 se muestra el diagrama de clases de las bibliotecas independientes.

Cada biblioteca tiene los siguientes modelos:

- Libros y sus relaciones con idiomas, géneros literarios y autores.
- Ejemplares individuales de libros. Su estado puede ser disponible, prestado o en mantenimiento.
- Eventos, que incluyen una descripción, título, fecha, lugar y ocupación máxima.
- Espacios de estudio, con atributos como nombre, ubicación y capacidad.
- Datos de la biblioteca como la dirección en la que se encuentra, contacto y el horario.

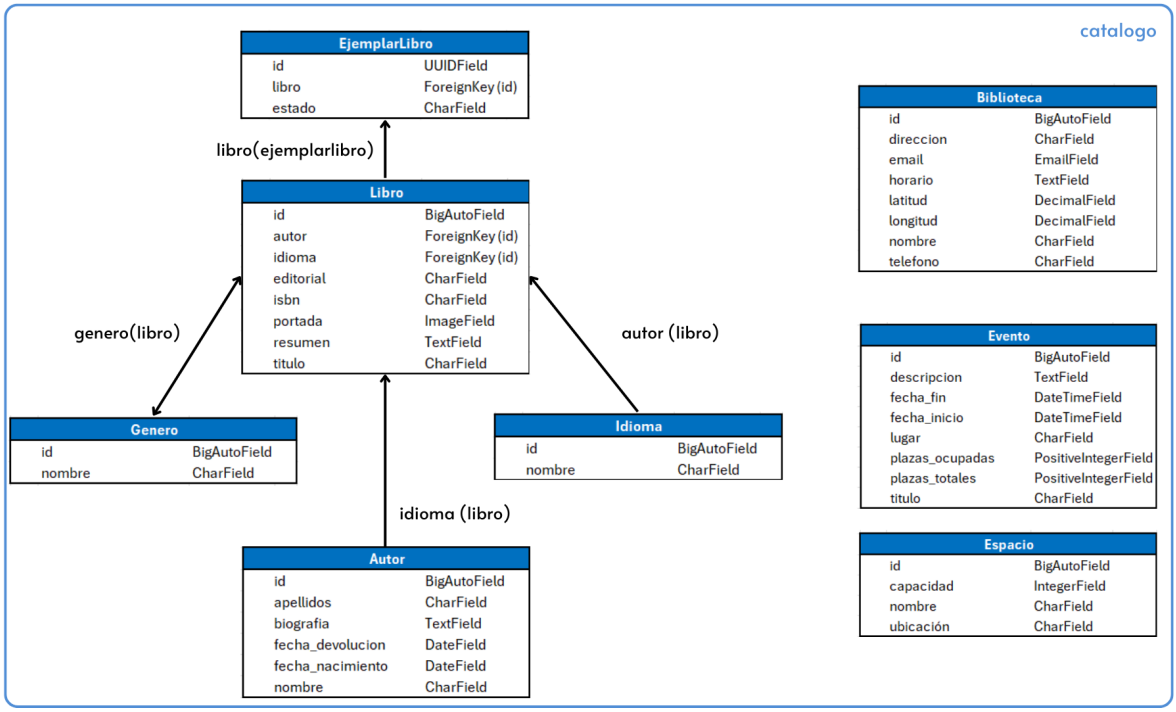


Figura 8.- Diagrama de clases biblioteca

5.2.3 DISEÑO DE LA INTERFAZ DE USUARIO

5.2.3.1 Mockup

El primer paso para realizar el diseño de la interfaz gráfica consistió en crear un *mockup* haciendo uso de la herramienta de Canva [59]. Este mockup se diseñó con el fin de tener un primer esbozo del aspecto gráfico de la aplicación web, así como de las funcionalidades que se querían incluir.

La estructura base de este primer diseño visual lo conforma una barra de navegación donde se incluyen las principales características de la plataforma: la página de inicio, el catálogo donde poder consultar y reservar libros, la información de las bibliotecas asociadas, la inscripción a eventos, la sección de contacto y el área donde el usuario puede consultar sus reservas activas. Además, en la parte superior izquierda se han incluido dos botones: uno de inicio de sesión y otro de registro. Estos tres elementos son comunes en todas las páginas de

la plataforma y le facilitan al usuario navegar y cambiar entre las distintas funcionalidades del sitio web. En el Anexo II recoge cada una de las pantallas del mockup diseñado.

5.2.3.1.1 Página de inicio

La pantalla principal incluye en la parte superior una barra de navegación. Justo debajo, se muestra un texto de bienvenida a la plataforma, así como un carrusel con las novedades destacadas de las bibliotecas y un botón para acceder a ellos directamente. En la Figura 22 se puede ver el mockup correspondiente con esta pantalla. En la versión final se incluyó una guía de uso de la plataforma junto con un listado de preguntas frecuentes para facilitarle al usuario la navegación por la página.

5.2.3.1.2 Páginas de registro e inicio de sesión

Las Figura 23 y 24 muestran las pantallas de registro e inicio de sesión de la plataforma. A través de estas páginas, el usuario puede crear una cuenta, si es la primera vez que se conecta, o iniciar sesión a través de sus credenciales si ya se ha registrado previamente. Para navegar por la plataforma y acceder a funcionalidades de reserva, consultar el historial de préstamos o manejar el perfil del usuario, es necesario haberse autenticado antes.

5.2.3.1.3 Página de consulta de bibliotecas

La Figura 25 muestra el diseño de la pantalla donde se detalla la información correspondiente con las bibliotecas asociadas a la plataforma. Esta primera versión se diseñó como una lista de tarjetas verticales que detallan información sobre la localización, número de asientos y ejemplares de la institución correspondiente. En la versión final se separó la lógica de reserva de espacios de trabajo con la consulta de datos de la biblioteca, favoreciendo de esta forma una navegación más intuitiva y una mejor organización.

5.2.3.1.4 Diseño del catálogo

La pantalla del catálogo muestra en una disposición vertical cada uno de los libros que forman parte del catálogo bibliotecario. En el lateral izquierdo, se dedicó una sección para

el filtrado de ejemplares en función del género literario. De esta forma se le permite al usuario obtener un resultado que se alinee más con sus intereses. La Figura 26 muestra el resultado de este diseño. Posteriormente, cuando se desarrolló la plataforma se añadió en la parte superior una barra de búsqueda de libros incluyendo justo debajo filtros en función de la biblioteca, el género y el autor. Además, se incluyó en la tarjeta un botón para acceder a la reserva del libro y a su información detallada.

5.2.3.1.5 Diseño de eventos

Las Figura 28 y 29 muestran el diseño de la pantalla de inscripción a eventos. En estas visualizaciones el usuario puede registrarse en eventos organizados por las distintas instituciones y consultar su calendario para tener una visión más completa.

El diseño se centró en mostrar de forma clara la información relevante de cada evento (título, biblioteca, localización, aforo disponible y fecha/hora). Además, se incluyó un botón para facilitar el proceso de inscripción.

5.2.3.1.6 Página de contacto

En la Figura 30 se incluye el mockup de la pantalla de contacto. Esta página incluye un formulario claro con información que el usuario debe rellenar para poder realizar una pregunta, queja, solicitud o consulta y mandarla por correo a los administradores de la plataforma web.

De esta forma, los usuarios cuentan con una vía directa para contactar con el equipo responsable de la plataforma. Esto les permite resolver dudas, hacer sugerencias o comunicar cualquier problema de forma sencilla.

5.2.3.1.7 Gestión de reservas

Las Figura 31, 32, 33 y 34 se corresponden con el diseño de la gestión de reservas que ha realizado el usuario en la plataforma. La primera figura se corresponde con el menú principal

a través del cual el usuario puede acceder directamente a la gestión de ejemplares reservados, eventos inscritos y reservas de asientos.

En cada una de estas pantallas se presenta la información más relevante de cada reserva realizada. Además, se le ofrece al usuario autenticado la opción de cancelar sus reservas de forma sencilla si así lo desea.

5.2.3.2 Flujo de navegación

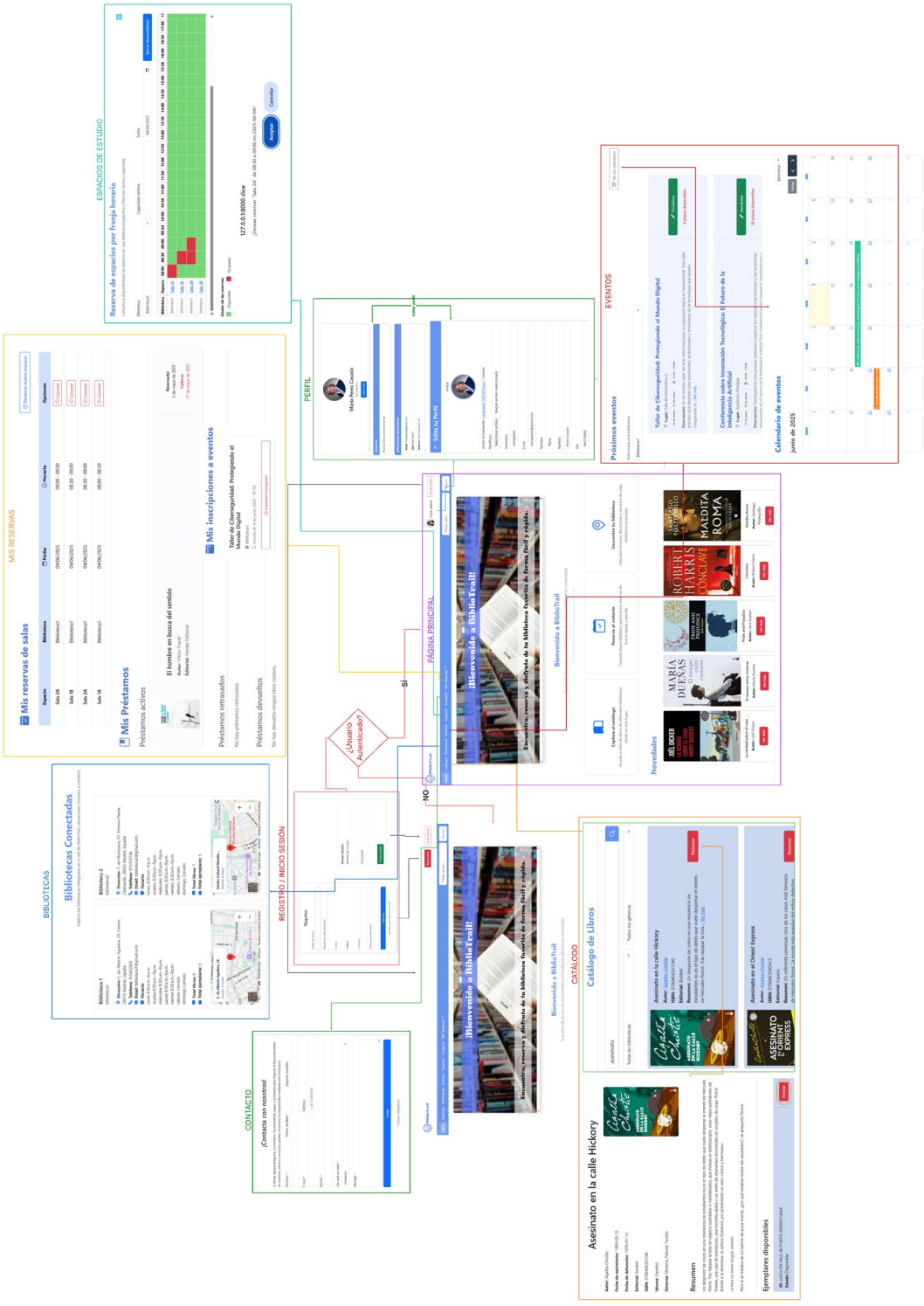


Figura 9.- Diagrama de flujo de navegación

En la Figura 9 se muestra el flujo de navegación de la página web. Gracias a este diagrama se puede comprender mejor desde dónde se accede a cada vista del sitio web y cómo se conectan entre sí las distintas funcionalidades.

5.2.3.3 Manual de usuario

En el Anexo IV recoge una guía de uso detallada que ayuda y facilita la navegación al usuario por la plataforma web creada.

5.2.4 SEGURIDAD Y GESTIÓN DE USUARIOS

Uno de los objetivos principales de este proyecto, mencionado en el Capítulo 4 fue la implementación de un sistema que garantice la gestión segura de los datos de los usuarios y contenga medidas de validación y seguridad. El sistema desarrollado, cuenta con distintos mecanismos para acceder a ciertas funcionalidades, validar la información, restringir permisos y autenticar a los usuarios que se registran en la plataforma. En esta sección se recogen los más relevantes.

5.2.4.1 Autenticación

Django ofrece varias medidas de seguridad de base y mecanismos de protección [2]. Además, como se comentó en el Capítulo 2, este software ayuda a evitar ataques de seguridad comunes como las inyecciones SQL, CSRF o XSS.

En el proyecto desarrollado se ha utilizado el sistema de autenticación por defecto de Django. Este sistema incluye mecanismos como el almacenamiento de forma segura de las contraseñas de los usuarios, que se realiza mediante hashing (PBKDF y stretching), o validaciones de sesiones. Para el uso de la plataforma, cada persona debe registrarse con un nombre de usuario y una contraseña. Al iniciar sesión se crea una sesión segura que permite acceder a funcionalidades restringidas de la plataforma.

En el diseño de los formularios de registro e inicio de sesión se ha incluido un token CSRF. Esto evita que otra página web pueda enviar peticiones en nombre del usuario, sin que este sea consciente de ello. Además, la interfaz de estas dos funcionalidades se ha construido con

Crispy Forms [60]. Este paquete de Django permite mostrar de forma más clara los campos de los formularios y trae consigo validaciones adicionales.

5.2.4.2 Control de acceso y permisos

Para el uso completo de las funcionalidades de BiblioTrail los usuarios deben autenticarse. Solo los usuarios que estén autenticados podrán acceder a las vistas restringidas. Para limitar el permiso a estas vistas, se utilizó el decorador `@login_required` de forma que cada vez que el usuario quiera acceder a alguna de estas páginas se comprobará su estado de autenticación. Todo el proceso se realiza bajo HTTPs lo que asegura que la transmisión de credenciales esté cifrada y se empleen cookies de sesión.

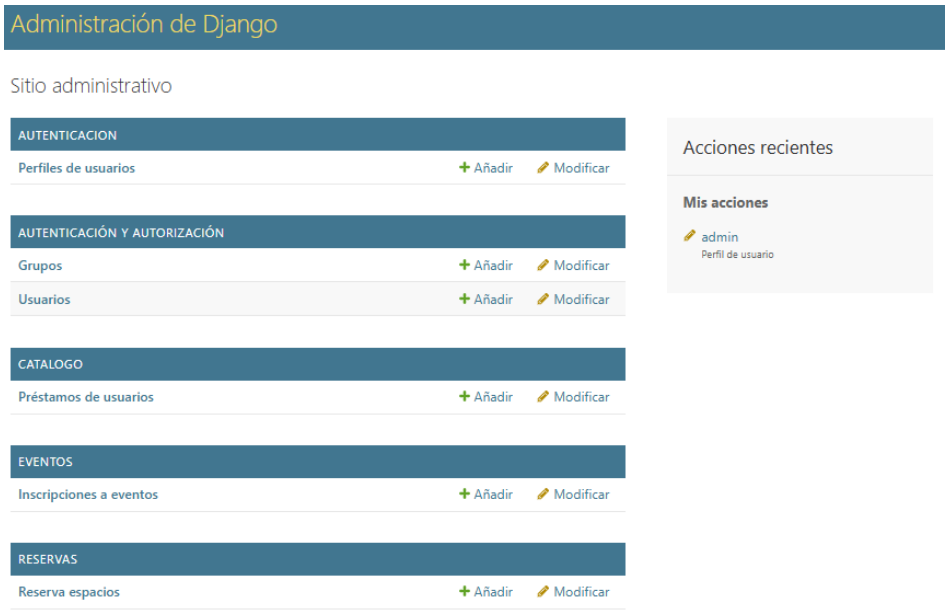


Figura 10.- Panel de administración de Django

Django también incluye un panel de administración. Sin embargo, únicamente pueden acceder a él los usuarios que cuenten con permisos especiales. El proyecto está configurado de forma que solo los usuarios con permisos de staff (`is_staff`) o superusuarios (`is_superuser`) puedan acceder. Esto permite a los administradores gestionar fácilmente la plataforma desde una interfaz intuitiva y visual.

5.2.4.3 Seguridad y cifrado

Django aplica varias capas de seguridad en el proyecto para proteger tanto los datos guardados en la base de datos como los que se transmiten entre el cliente y el servidor. Las contraseñas se cifran de forma automática con algoritmos como PBKDF2. Este algoritmo toma la contraseña original creada por el usuario, le añade un numero aleatorio que se mezcla con ella y aplica múltiples veces una función hash antes de ser guardada en la base de datos. De esta forma se garantiza la seguridad de las credenciales del usuario.

Además, en la configuración del proyecto se ha configurado el parámetro `ALLOWED_HOSTS` para limitar los dominios válidos desde los que acceder al sistema. Por otro lado, Django utiliza ORM (Mapeo de Objetos Relacional), que protege frente a inyecciones SQL.

```
ALLOWED_HOSTS = [
    '127.0.0.1',
    '127.0.0.2',
    'localhost',
]
```

5.2.4.4 Validaciones

En el proyecto se incorporan validaciones en varios niveles. En los formularios se definen campos obligatorios, formatos fijos (como campos de verificación de correo electrónico o número de teléfono) o validaciones personalizadas.

Además, se generan mensajes de error específicos cuando el usuario escribe algo incorrecto. De esta forma, este sabe qué debe corregir y porqué. El sistema aplica validaciones mediante campos específicos de Django (`EmailField`, `CharField`, `unique=True`, `blank=False`, etc.).

Esto evita inconsistencias, asegura la integridad de las tablas de datos y evita datos incorrectos tanto en el frontend como en el backend.

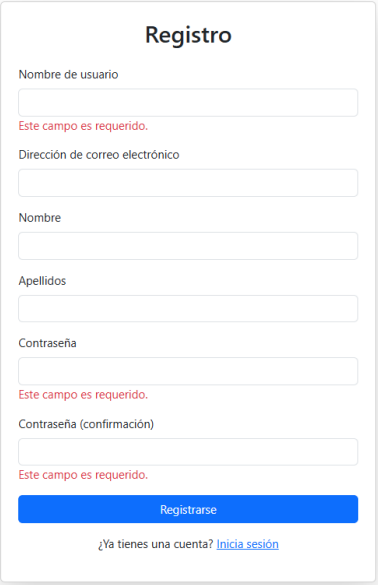


Figura 11.- Ejemplo validaciones formulario

5.2.5 COMUNICACIÓN ENTRE SISTEMAS

Uno de los aspectos más importantes del sistema es su capacidad de comunicarse automáticamente con múltiples bibliotecas externas, intercambiando información entre sí. BiblioTrail actúa como un agregador: no almacena todos los recursos (como libros, ejemplares, datos de las bibliotecas, etc.) sino que los consulta y gestiona accediendo a las APIs REST que proporcionan cada una de las bibliotecas asociadas.

Esto permite acceder a catálogos, eventos, o consultar la disponibilidad sin necesidad de tener que guardar internamente los datos. A continuación, se detallará cómo ha sido posible realizar esta conexión entre sistemas y qué mecanismos de consulta y sincronización se han implementado para que la plataforma funcione.

5.2.5.1 Arquitectura de comunicación

Como se explicó al principio de este capítulo, la arquitectura es de tipo cliente-servidor. El servidor principal (BiblioTrail) hace de intermediario entre el cliente (navegador del usuario) y las bibliotecas externas. Cada una de ellas expone su API REST y le ofrece al servidor puntos específicos por los que acceder a sus datos, llamados *endpoints*.

Cada una de las conexiones entre ambos componentes se realiza con el cliente HTTP *httpx*. Este cliente permite realizar peticiones asíncronas y devolver las respuestas en formato JSON. Cada petición es dinámica y se construye a partir de la URL base de la biblioteca. Esta URL se recupera con la función `cargar_bibliotecas()` (que se explicará más adelante). Este diseño permite evitar que haya una única fuente de datos de la que dependa todo el sistema. Además, favorece el diseño modular y la escalabilidad del sistema de forma que añadir bibliotecas no suponga un problema.

5.2.5.2 Consulta de información

Cuando un usuario realiza una búsqueda, el sistema principal lanza una petición a cada biblioteca usando su endpoint específico y unificando los resultados. Por ejemplo, en el caso particular de la búsqueda de libros, se emplea el endpoint `/api/libros/` y la función `buscar_libros()`.

A continuación, la siguiente tabla muestra cuales son los endpoints REST más relevantes junto con una breve explicación de a qué información se está accediendo con su uso.

Endpoint	Método	Descripción
/api/libros/	GET	Devuelve el listado de libros disponibles en la biblioteca.
/api/libros/<id>/	GET	Devuelve la información detallada de un libro concreto.
/api/libros/<id>/ejemplares-disponibles/	GET	Lista los ejemplares disponibles para préstamo de un libro específico.
/api/espacios/	GET	Devuelve una lista de todos los espacios de las bibliotecas.
/api/disponibilidad-general/?fecha=YYYY-MM-DD	GET	Devuelve la disponibilidad completa de todos los espacios en esa fecha.
/api/reservas/	POST	Crea una reserva de un espacio o ejemplar.
/api/reservas/<id>/	DELETE	Cancela una reserva concreta.
/api/eventos/	GET	Devuelve todos los eventos disponibles en la biblioteca.
/api/eventos/<id>/inscripcion/	PATCH	Aumenta el número de inscritos en un evento concreto.

Figura 12.- Endpoints REST relevantes

5.2.5.3 Sincronización

Además de consultar los datos, el sistema envía información a las bibliotecas cuando el usuario realiza acciones importantes y modifica el estado de algunos campos en sus bases de datos. Por ejemplo:

- Cuando el usuario se inscribe en un evento, se lanza un PATCH a la API de la biblioteca para modificar la ocupación.
- Al reservar un ejemplar, se actualiza el estado (disponible, reservado).

5.2.5.4 Manejo de errores

Como las bibliotecas se encuentran en servidores independientes, podrían ocurrir errores en la red, caídas o tiempos de respuesta muy largos. Para evitar que esto afecte a la experiencia de usuario:

- Cada llamada a las APIs se envuelve en bloques try-except. Esta estructura permite que, si ocurre un error o excepción en la ejecución, se maneje directamente sin detener el programa.
- Además, se comprueba que las respuestas sean válidas y tengan el formato adecuado.
- Si se produce algún fallo, el sistema informa al usuario sin romper la interfaz. Además, en la interfaz gráfica se le manda un mensaje al usuario notificándole del problema:

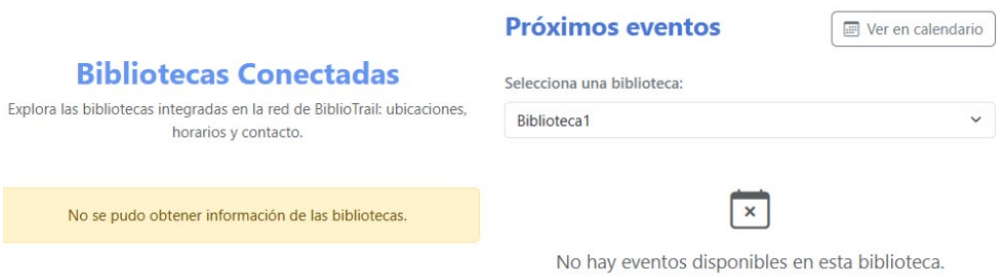


Figura 13.- Mensajes de fallo

Esto garantiza que el sistema siga funcionando, aunque una biblioteca no esté disponible en ese momento.

5.2.5.5 Escalabilidad

Para facilitar que el sistema siga creciendo, se ha diseñado una arquitectura modular. La función `cargar_bibliotecas()` actúa como punto de entrada para saber qué bibliotecas están conectadas y acceder a su información. Añadir una biblioteca adicional resulta un proceso simple, teniendo únicamente que agregar su estructura en el proyecto y registrarla en el diccionario. Esto permite escalar el sistema sin modificar la lógica principal.

```
def cargar_bibliotecas():
    bibliotecas = {}
    with open("bibliotecas.txt", "r") as f:
        for linea in f:
            if ',' in linea:
                nombre, url = linea.strip().split(',', 1)
                bibliotecas[nombre.strip()] = url.strip()
    return bibliotecas
```

Además, los endpoints están estandarizados. Esto significa que el agregador no necesita adaptar su lógica a cada biblioteca por separado. Todas las APIs comparten la misma estructura, por lo que el sistema puede reutilizar las mismas funciones para acceder a cualquier biblioteca. Esto permite que el código se adapte de forma dinámica en función del origen de los datos, haciendo que el sistema sea mucho más eficaz y escalable.

5.3 IMPLEMENTACIÓN

La implementación del proyecto se realizó de forma incremental. Primero, se realizó una página web básica que tan solo contenía la lógica del catálogo, la funcionalidad de buscar libros y acceder al formulario de contacto. Más adelante, conforme fue creciendo el proyecto se separó la lógica de la plataforma base en distintos módulos. Se dividió en tres proyectos con el fin de conseguir que fuesen completamente autónomos. Por un lado, se implementó la lógica de las bibliotecas, y por otro lado la del sistema central.

5.3.1 ORGANIZACIÓN DEL PROYECTO

El proyecto se desarrolló en Django y se estructuró dentro de una misma carpeta que agrupa los tres proyectos principales: el sistema principal (Bibliotrail) donde se ha desarrollado toda la lógica de la plataforma web, y, por otro lado, cada uno de los proyectos de las bibliotecas (biblioteca1 y biblioteca2). Aun habiéndose implementado en la misma carpeta de trabajo por hacer más cómodo el desarrollo del proyecto, cada uno de ellos funciona como un proyecto de Django completamente autónomo. Esto significa que cada uno cuenta con su propia configuración, base de datos local y servidor. Además, se pueden ejecutar de forma aislada sin que afecte a su funcionamiento.

En el esquema de directorios se puede observar la organización y estructura de carpetas cada proyecto:

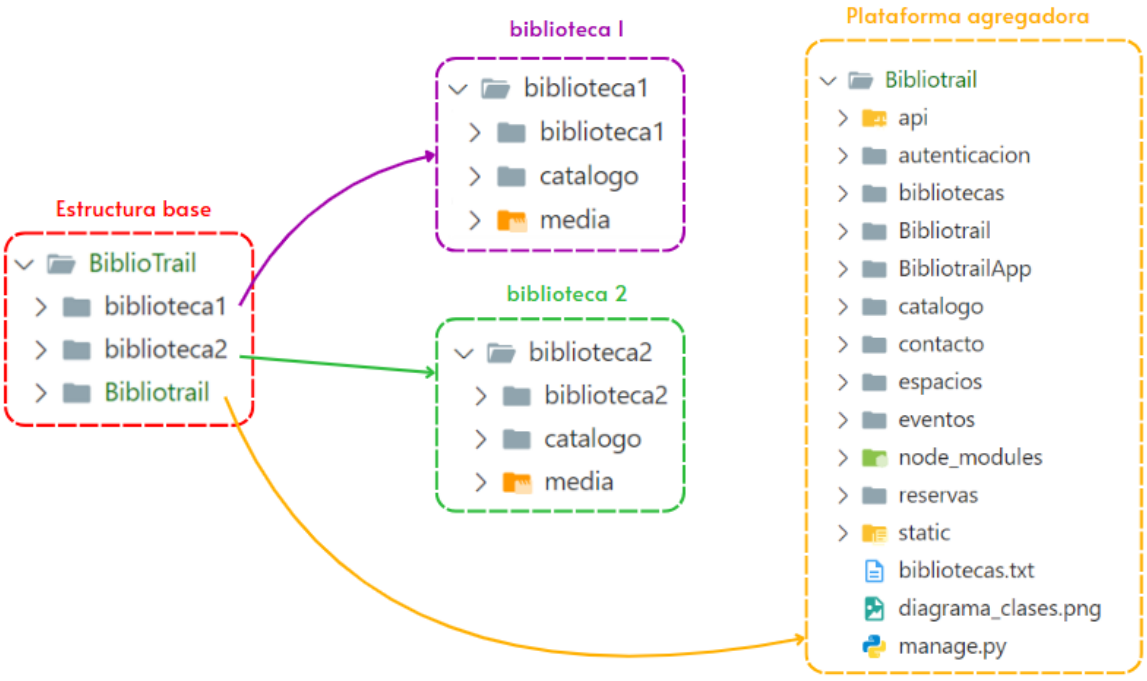


Figura 14.- Estructura de carpetas

5.3.2 CONFIGURACIÓN DEL ENTORNO

Para el desarrollo del sistema y la construcción de la aplicación web se utilizó la versión de Python 3.10.11 junto con el framework Django. Cada uno de los tres proyectos (BiblioTrail, biblioteca1 y biblioteca2) son entornos independientes y su configuración es autónoma.

Las dependencias necesarias para cada proyecto se guardaron en un archivo de texto (`requirements.txt`). Esto permitió poder instalar los paquetes necesarios para el despliegue de la aplicación haciendo uso del comando `pip install -r requirements.txt`.

En cuanto a la base de datos, se comenzó utilizando para el desarrollo SQLite por su facilidad de uso y simplicidad. Sin embargo, más adelante se cambió a PostgreSQL. Cada proyecto cuenta con su propia base de datos local, que son gestionadas a través del sistema de migraciones de Django (usando los comandos `makemigrations` y `migrate`). Estas bases de datos son independientes las unas de las otras y cada proyecto únicamente tiene acceso a la suya.

Para levantar los servidores de cada proyecto se utilizaron los siguientes puertos:

- 8000 para el sistema central
- 8001 para la biblioteca 1
- 8002 para la biblioteca 2

Esto permitió simular un entorno distribuido, donde cada una de las bibliotecas actúa como un servicio independiente con su propia API.

La información referente a los servidores también se recoge en el archivo `bibliotecas.txt` que contiene la siguiente información:

```
biblioteca1,127.0.0.1:8001
biblioteca2,127.0.0.1:8002
```

Si se quisieran añadir bibliotecas adicionales al sistema sería necesario incluir una nueva entrada en este archivo con el nombre de la biblioteca a la que desea acceder y la dirección IP y puerto en el que se levanta su servidor.

En el proyecto también se instalaron algunas librerías complementarias como httpx para realizar llamadas HTTP entre servicios. También incorpora librerías externas como DRF para las APIs.

Como se mencionó en la sección 5.2.4.3 se le permite el acceso al sistema desde *localhost* y a direcciones locales definidas en el parámetro `ALLOWED_HOSTS`. Además, se habilitó `CorsMiddleware` para que el agregador y las APIs de las bibliotecas pudiesen comunicarse.

Los archivos estáticos e imágenes de los usuarios (imagen de perfil) se almacenan en carpetas separadas (/static/). Por otro lado, el sistema de envío de correos electrónicos empleado, en la sección de "Contacto" para permitir la transmisión de la información recogida en el formulario, se configuró mediante un servidor SMTP de Gmail.

En cuanto a la configuración de las bibliotecas externas, está principalmente diseñada para poder exponer sus recursos a través de una API REST. Para ello se utilizan fundamentalmente *Django Rest Framework* (DRF) y `django_filters` para poder hacer búsquedas y filtros de forma más sencilla en los endpoints. Además, están configurados para permitir peticiones que vengan del agregador mediante `corsheaders`.

5.3.3 IMPLEMENTACIÓN DE MÓDULOS PRINCIPALES

A continuación, se explicarán cada uno de los módulos principales de cada proyecto y se mencionarán las funciones más relevantes de cada uno de ellos.

5.3.3.1 Sistema principal: BiblioTrail

Este sistema actúa como la plataforma central del proyecto. Desde él se pueden consultar, reservar y gestionar los recursos que ofrecen las bibliotecas externas. Como se mencionó anteriormente, se ha desarrollado en Django e integra múltiples funcionalidades. Cada funcionalidad se separó en un módulo distinto (apps). Cada una de estas apps se encarga de una parte concreta del sistema y esto permite tener una arquitectura modular y escalable.

Como cada uno de los módulos se explicaron en el apartado 5.2.2.1 de este capítulo, a continuación, se incluye una tabla resumen:

Resumen de las aplicaciones principales:

Aplicaciones principales	Descripción
catalogo	Gestina la búsqueda de libros y reserva de ejemplares. Incluye la lógica para consultar las APIs externas de cada biblioteca y combianar las respuestas en una vista única.
reservas	Controla el historial de préstamos y las reservas del usuario. En esta app se encuentra la lógica de la funcionalidad de "Mis Reservas" y accede a la información de los préstamosrealizados por el usuario.
eventos	Permite consultar eventos organizados por las bibliotecas y realizar inscripciones.
espacios	Muestra la disponibilidad horaria de salas o espacios de estudio. Permite al usuario reservar directamente desde la interfaz gráfica.
autenticacion	Se encarga del inicio de sesión, el registro de usuarios y la gestión de sesiones.
bibliotecas	Contiene la lógica para acceder a la información más relevante de cada biblioteca asociada. Se comunica con las APIs de las bibliotecas asociadas.
BibliotrailApp	Centraliza las vistas generales del sitio, como la página de inicio y el diseño base.
contacto	Implementa un formulario de contacto que permite al usuario enviar consultas o sugerencias al equipo de administración.

Figura 15.- Tabla resumen apps Bibliotrail

Funciones destacadas:

Una de las funciones más importantes es `buscar_libros(request)`, que se encuentra en la app de `catalogo`. Esta función se encarga de consultar simultáneamente en cada una de las APIs de las bibliotecas y unifica los resultados en una sola lista.

Otra función destacada es `crear_reserva_espacio`. Esta función se activa cuando el usuario hace clic en una franja disponible (para una sala de estudio y una hora determinada) y, primero crea una reserva en la base de datos y luego lanza una petición a la API de la biblioteca para que se actualice la ocupación de esa sala.

En cuanto a los modelos principales del sistema, se encuentran: `PrestamoUsuario`, `ReservaEspacio` e `InscripcionEvento`. Sus atributos principales se pueden ver en la Figura 6 que representa el diagrama de clases de estos modelos. Estas estructuras permiten registrar las acciones del usuario sin tener que modificar las bases de datos de las bibliotecas

asociadas. Por tanto, estos modelos son la "*unión*" entre la información mostrada al usuario y las operaciones de las APIs.

5.3.3.2 Bibliotecas independientes: biblioteca 1 y biblioteca2

Cada biblioteca independiente se encuentra en un proyecto Django distinto, que comparten la misma estructura. Los componentes que la conforman son:

- Catálogo: Es la app principal de cada biblioteca. Aquí se definen los modelos internos como: `Libro`, `EjemplarLibro`, `Autor`, `Genero` e `Idioma`. También gestiona los endpoints REST de cada una de sus APIs para consultar libros y ejemplares.
- Api: Contiene las vistas implementadas con Django Rest Framework. En esta sección se define la lógica para compartir los endpoints con el sistema principal, permitiéndole acceder y consultar los datos de las bibliotecas.
- Admin: Gestiona el panel de administración de Django. Esto permite a los responsables de cada biblioteca añadir ejemplares, modificar libros y gestionar internamente el catálogo de libros que se compartirá a través de la API con la plataforma principal.

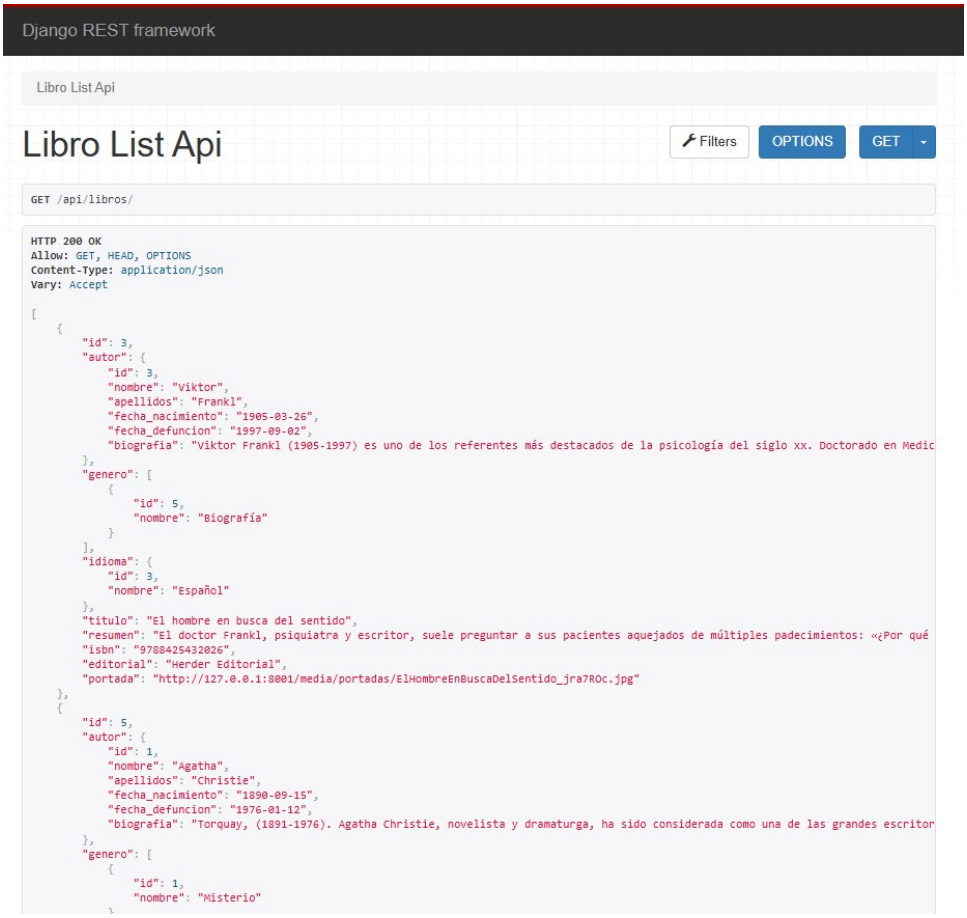


Figura 16.- DRF interfaz API

Para comprender mejor cómo funciona Django Rest Framework (DRF), es necesario entender alguno de sus componentes principales:

- Serializers: Se encargan de traducir los models de Django (como los libros y ejemplares) a formatos como JSON.

```python
class AutorSerializer(ModelSerializer):
    class Meta:
        model = Autor
        fields = '__all__'

class LibroSerializer(ModelSerializer):
    autor = AutorSerializer()
    genero = GeneroSerializer(many=True)
    idioma = IdiomaSerializer()
    class Meta:
        model = Libro
        fields = '__all__'
```

Figura 17.- Ejemplos serializers

- ViewSets y APIViews: Permiten agrupar la lógica en una misma clase. Esto reduce mucho código repetitivo y permite que el programa se centre en las rutas y respuestas.

```python
class LibroListAPIView(generics.ListAPIView):
    queryset = Libro.objects.all()
    serializer_class = LibroSerializer
    filter_backends = [SearchFilter, DjangoFilterBackend]
    search_fields = ['titulo', 'autor__nombre', 'autor__apellidos']
    filterset_class = LibroFilter
```

Figura 18.- Ejemplo APIView

- Routers: Los routers definen las URLs asociadas a los ViewSets. Indican qué URL debe tener cada recurso de la API.

Todas las bibliotecas comparten esta misma estructura de API. Esto permite que el sistema central pueda enviar solicitudes a cualquiera de ellas sin tener que adaptar el código a cada biblioteca concreta.

Capítulo 6. ANÁLISIS DE RESULTADOS

6.1 EVALUACIÓN DEL CUMPLIMIENTO DE OBJETIVOS

6.1.1 DESARROLLO DE PLATAFORMA WEB PARA LA GESTIÓN BIBLIOTECARIA

Uno de los objetivos principales del proyecto consistía en desarrollar una plataforma web destinada a la gestión de recursos bibliotecarios. Este objetivo se ha cumplido haciendo uso de Django [1]. Sobre este framework se construyó tanto el sistema central como las bibliotecas independientes. El sistema se estructura en tres proyectos independientes: uno central y dos bibliotecas asociadas.

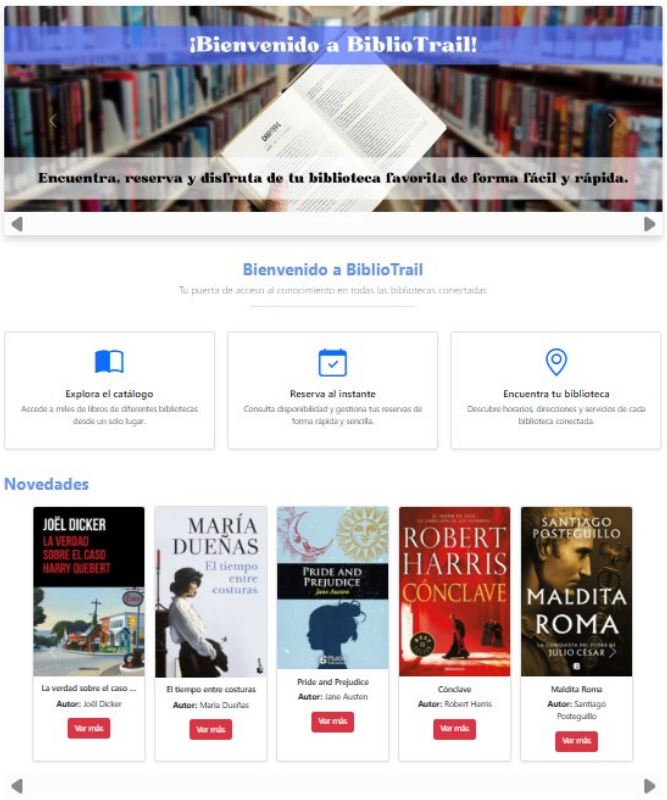


Figura 19.- Plataforma web desarrollada

A pesar de desplegarse en servidores distintos y tener independencia los unos de los otros, los sistemas se logran comunicar entre sí mediante peticiones HTTP y APIs REST.

La implementación del sistema principal se detalla en el Capítulo 5, en especial, en los apartados 5.2.1.3, 5.2.2.1 y 5.3.3.1, mientras que la de las bibliotecas se explican en los apartados 5.2.2.2, 5.2.5 y 5.3.3.2.

6.1.2 CREACIÓN DE UNA INTERFAZ EFICAZ E INTUITIVA

Otro de los objetivos principales descrito en el Capítulo 4 fue garantizar una buena experiencia de usuario en la plataforma, haciendo que sea sencilla, intuitiva y fácil de usar. Para lograrlo, se creó una interfaz web que permite a los usuarios navegar por la plataforma y acceder fácilmente a las principales funcionalidades como acceder al catálogo, reservar espacios, realizar préstamos o inscribirse a eventos.

Esta interfaz se diseñó con lenguajes de programación como HTML, CSS y Java Script. Además, se utilizó Bootstrap para construir páginas *responsive* que tuviesen un buen diseño visual en cualquier tipo de dispositivo.

La organización lógica del menú de navegación y el sistema de rutas fue algo a lo que se le dedicó especial atención, permitiendo que la navegación fuese sencilla e intuitiva para el usuario. En la Figura 20 se puede ver un ejemplo del diseño de la plataforma.

Figura 20.- UX interfaz gráfica

6.1.3 AGREGACIÓN DE DATOS Y CONEXIÓN DE BIBLIOTECAS EXTERNAS

Uno de los objetivos propuestos en el Capítulo 4, en especial detallado en el apartado 4.2.2, fue lograr una conexión efectiva entre el sistema agregador y las bibliotecas independientes. Este objetivo se ha cumplido gracias a la implementación de APIs REST en cada uno de los proyectos de las bibliotecas externas.

Cada una de ellas exponen sus datos guardados en su base de datos local y permiten al sistema principal acceder a ellos, consultarlos y modificarlos. Esto se consiguió gracias al sistema de llamadas HTTP de la librería httpx. Además, toda esta funcionalidad se realiza de forma dinámica gracias a la arquitectura modular del proyecto.

La implementación de este objetivo se detalla en el Capítulo 5 en las secciones 5.2.1.1, 5.2.1.2 y 5.2.5.

6.1.4 IMPLEMENTACIÓN DE FUNCIONALIDADES

Otro objetivo clave fue implementar las funcionalidades descritas en la sección 4.2. Entre las más destacadas están:

- La búsqueda y consulta de libros.
- La reserva de ejemplares.
- La gestión de eventos e inscripción de los usuarios a las actividades ofrecidas por las bibliotecas.
- La reserva de espacios de trabajo y búsqueda de disponibilidad por hora.
- La visualización de los préstamos, reservas y cancelaciones realizadas por el usuario (en la sección de “Mis reservas”).
- La gestión de usuarios y su autenticación. Incluyendo un sistema de registro e inicio de sesión, y, además, limitando el acceso de las vistas de la plataforma a los permisos del usuario.

Por tanto, este objetivo se ha cumplido satisfactoriamente, desarrollando múltiples módulos que cumplan una funcionalidad concreta y, permitan a los usuarios interactuar con el sistema y acceder fácilmente a los recursos bibliotecarios que quieran consultar.

6.1.5 SEGURIDAD, GESTIÓN DE USUARIOS Y ESCALABILIDAD

El sistema implementado cumple el objetivo propuesto en el Capítulo 4 , sección 4.2.2, pues cuenta con un modelo de autenticación para identificar a los usuarios que se dan de alta en la plataforma, validaciones, cifrado de contraseñas y protección contra ataques de ciberseguridad. La explicación de estas medidas de seguridad se detalla en el apartado 5.2.4.

Además, al tratarse de una arquitectura modular, el proyecto favorece la escalabilidad del sistema. De esta forma, cada biblioteca puede operar de forma independiente y su funcionamiento no se ve alterado si se agregasen más bibliotecas. Esto permite que el programa y sus funcionalidades puedan seguir creciendo fácilmente.

6.2 RETOS Y LIMITACIONES ENCONTRADAS

Los principales retos encontrados durante el desarrollo de este Trabajo Final de Grado fueron los siguientes:

6.2.1 SEPARACIÓN EN PROYECTOS Y ARQUITECTURA DISTRIBUIDA

Uno de los retos más importantes que se encontraron al realizar el proyecto fue la separación en proyectos independientes. Al comenzar la implementación del sistema, se tenía toda la lógica en un mismo proyecto. Al intentar agregar las funcionalidades de las bibliotecas, primero se intentó realizar dentro del mismo proyecto, sin embargo, al comprender que el funcionamiento debía de estar aislado del sistema central, se vio que era necesario separar completamente los sistemas en tres proyectos completamente independientes. Además, también se tuvieron que levantar los servidores en puertos distintos para conseguir la comunicación entre plataformas a través de APIs REST.

Asimilar este enfoque y reorganizar la arquitectura fue un proceso que llevó tiempo. Sin embargo, gracias a él, se logró conseguir sistemas autónomos e independientes y una arquitectura distribuida.

6.2.2 APRENDIZAJE DE DJANGO

Al comenzar la etapa de desarrollo, no se tenía experiencia previa con Django. Entender este framework (gestión de plantillas, vistas, modelos, enrutamientos, etc.) y aprender a utilizarlo no supuso una tarea trivial, sin embargo, fue fundamental para el desarrollo de la plataforma web. Al tratarse de un software tan completo, con su uso no solo se fue capaz de realizar las funcionalidades previstas, sino que también, gracias a sus librerías y sistemas incluidos, reforzó la seguridad de la plataforma y la gestión de datos y usuarios.

6.2.3 LÓGICA DE CONEXIÓN CON APIS EXTERNAS

Otro reto importante fue diseñar un sistema que permitiera comunicar a cada uno de los proyectos independientes entre sí. Esto se logró gracias a las APIs REST y a Django REST Framework. Con estas implementaciones, se consiguió que el sistema central pudiera consultar y modificar la información de las bases de datos externas manteniendo la arquitectura distribuida.

Fue necesario que cada biblioteca expusiera su propia API para permitir la comunicación entre sistema, pero con ayuda de endpoints estandarizados se consiguió que el agregador se adaptara dinámicamente sin necesidad de tener que personalizar el código para cada una de las bibliotecas agregadas. Esto supuso un punto clave para que el proyecto sea escalable, de forma que, si se quisieran añadir más instituciones a la plataforma, no supusiera un reto adicional.

6.2.4 ORGANIZACIÓN DEL TIEMPO Y CARGA ACADÉMICA

Otro reto fue la gestión del tiempo. El desarrollo del proyecto coincidió con otros compromisos académicos y un curso demandante. Por tanto, se tuvo que compaginar tareas, asignaturas y exámenes con el desarrollo y avance del TFG. Fue necesario una buena

organización, establecer prioridades y ajustarse a la planificación propuesta en el Anexo B, presentado a principio de curso.

6.2.5 MIGRACIÓN DE BASE DE DATOS

En un inicio el proyecto utilizaba el gestor de base de datos por defecto, SQLite, que incorpora Django al crear un nuevo proyecto. Durante gran parte del desarrollo se trabajó con él. Sin embargo, a medida que el proyecto crecía, se decidió migrar a PostgreSQL, un sistema mucho más robusto y potente que permitiese suportar una mayor carga de datos.

Esta migración no resultó trivial, ya que en el sistema ya se habían guardado datos de usuarios y recursos bibliotecarios durante el desarrollo. Se tuvo que realizar esta migración de forma cuidadosa para no perder los datos almacenados. Esto implicó cambiar las configuraciones del proyecto, adaptar el entorno y comprobar que el proyecto fuese compatible con PostgreSQL.

6.3 RESUMEN

En resumen, a lo largo de este capítulo se han evaluado los objetivos propuestos en el Capítulo 4 y se puede afirmar que se han cumplido de forma satisfactoria. Se ha logrado diseñar una plataforma web accesible, modular e intuitiva que permita gestionar eficazmente los recursos bibliotecarios centralizándolos en una misma página web y conectándose con bibliotecas externas independientes mediante APIs. Además, este Trabajo de Fin de Grado, favorece una arquitectura modular y escalable para futuras ampliaciones o líneas de trabajo.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1 CONCLUSIONES

Tras haber desarrollado el proyecto y analizado los resultados, se puede concluir que se han alcanzado los objetivos propuestos de forma satisfactoria. BiblioTrail es una plataforma web capaz de gestionar eficazmente los recursos bibliotecarios permitiendo a los usuarios que se inscriben en ella hacer préstamos, inscribirse a eventos y reservar salas de trabajo. Además, permite centralizar el acceso de múltiples bibliotecas independientes en un mismo entorno mediante una estructura modular y escalable.

A pesar de esta independencia y autonomía entre sistemas, se ha logrado que haya una comunicación efectiva entre cada uno de ellos mediante la consulta de APIs REST. Como se explicó en el Capítulo 5, cada biblioteca expone sus propia API (libros, ejemplares, eventos, salas, capacidad) y el sistema accede a esos datos a través de peticiones HTTP. Para conseguir esta comunicación, las bibliotecas externas comparten una misma estructura. Esto permite que el sistema central no tenga que adaptarse a cada una de ellas, sino que pueda conectarse dinámicamente utilizando su URL base.

Durante todo el desarrollo, se ha buscado que la plataforma no solo funcionara correctamente, sino que también fuera accesible y clara para los usuarios que la utilizan. Esto se ha logrado utilizando Django [1] y Django Rest Framework [3], frameworks que han facilitado que el proyecto tenga una arquitectura distribuida, segura y robusta.

En cuanto a la planificación, se ha cumplido la programación prevista. La primera etapa de familiarización con el entorno web duró algo más de lo pensado originalmente. Sin embargo, esto no afectó al resultado final y cada tarea se desarrolló en el tiempo previsto.

Otro aspecto a destacar es que gracias a la arquitectura modular del proyecto, resulta sencillo escalar el sistema actual. Si en un futuro se desean añadir nuevas bibliotecas o funcionalidades, se puede conseguir sin necesidad de modificar la lógica y el funcionamiento base del programa.

En definitiva, este Trabajo de Fin de Grado cumple con los objetivos, ofreciendo una plataforma web capaz de gestionar efectivamente distintas bibliotecas bajo un mismo entorno. Se ha priorizado una buena experiencia de usuario, diseñando una interfaz gráfica intuitiva y fácil de usar. Gracias a su arquitectura distribuida y modular, cada institución cuenta con autonomía y el sistema está preparado para seguir creciendo.

7.2 LÍNEAS DE TRABAJO FUTURAS

Aunque se ha cumplido con los objetivos propuestos y el alcance del proyecto ha sido extenso, se han dejado fuera algunos aspectos que podrían añadirse en el futuro para seguir mejorando el sistema. Estas mejoras se han dividido en los siguientes bloques:

7.2.1 MEJORAS FUNCIONALES

Para mejorar la funcionalidad de la plataforma se podrían añadir:

- Sistema de reseñas y valoraciones tanto de los libros prestados, como de los eventos a los que se ha asistido.
- Filtros de búsqueda más avanzados, como poder filtrar por fechas, popularidad, “ejemplares más prestados”, etc.
- Incorporación de nuevas bibliotecas a la plataforma. Por simplicidad, en este proyecto solo se incluyen dos. Pero gracias a la arquitectura modular del sistema y al uso de APIs, añadir más resultaría un proceso relativamente sencillo. Bastaría que la nueva institución siguiese el mismo formato que los anteriores y expusiera su API al sistema principal.

7.2.2 MEJORAS DE INFRAESTRUCTURA

En cuanto a la infraestructura, otras posibilidades que ayudarían a hacer el sistema más completo y escalable serían:

- Despliegue real de los distintos proyectos en servidores independientes o en la nube.
- Dockerización de cada proyecto para facilitar el despliegue.
- Implementación de sistemas de backup automáticos.
- Ajustes y mejoras de configuración para la seguridad de la plataforma.

7.2.3 MEJORAS DE GESTIÓN Y ANÁLISIS

Por último, se podrían añadir funcionalidades más profundas de gestión y análisis, como:

- Análisis de patrones de uso, como franjas horarias para poder ver cuando existe una mayor demanda de reservas o préstamos.
- Detección de libros o eventos de mayor interés.
- Dashboards de administración para visualizar estadísticas de préstamos, reservas, inscripciones o uso de los recursos bibliotecarios.

Capítulo 8. BIBLIOGRAFÍA

[1] Django Software Foundation, «Django,» 2005. [En línea]. Available: https://www.djangoproject.com/. [Último acceso: 8 mayo 2025].

[2] Django Software Foundation, «Security in Django,» [En línea]. Available: https://docs.djangoproject.com/es/5.2/topics/security/. [Último acceso: 8 mayo 2025].

[3] Django Software Foundation, «Django REST Framework,» 2013. [En línea]. Available: https://www.django-rest-framework.org/. [Último acceso: 8 mayo 2025].

[4] G. M. Blanco, «Repositorio GitHub BiblioTrail,» [En línea]. Available: https://github.com/gmblanco/BiblioTrail. [Último acceso: 9 junio 2025].

[5] A. F. Prósper, «La biblioteca híbrida como oportunidad para situar a la biblioteca pública en el centro de la sociedad,» *El Profesional de La Información,* pp. 126-130, 2004.

[6] O. Y. C. Mora, S. A. D. d. l. Santos, M. C. O. Chico y E. C. Medina, «Factores que incentivan el uso de la biblioteca virtual en los estudiantes universitarios: un estudio de caso de la Universidad de Gómez Palacio de Durango,» *Biblios Journal of Librarianship and Information Science,* nº 66, pp. 98-111, 2017.

[7] Mozilla Corporation, «Introducción a Django - Aprende desarrollo web | MDN,» [En línea]. Available: https://developer.mozilla.org/es/docs/Learn_web_development/Extensions/Server-side/Django/Introduction. [Último acceso: 8 mayo 2025].

[8] Amazon Web Services, Inc., «¿Qué es Django? - Django - AWS,» [En línea]. Available: https://aws.amazon.com/what-is/django/. [Último acceso: 8 mayo 2025].

[9] Django Software Foundation, «Password management in Django,» [En línea]. Available: https://docs.djangoproject.com/en/5.2/topics/auth/passwords/. [Último acceso: 8 mayo 2025].

[10] Django Software Foundation, «How to use sessions,» [En línea]. Available: https://docs.djangoproject.com/en/5.1/topics/http/sessions/. [Último acceso: 8 mayo 2025].

[11] Flask, «Welcome to Flask - Flask Documentation (3.1.x),» [En línea]. Available: https://flask.palletsprojects.com/en/stable/. [Último acceso: 8 mayo 2025].

[12] Ruby on Rails, «Ruby on Rails,» [En línea]. Available: https://rubyonrails.org/. [Último acceso: 8 mayo 2025].

[13] «Ruby on Rails Vs Django: Which Is Right for You? | BairesDev,» 11 octubre 2023. [En línea]. Available: https://www.bairesdev.com/blog/ruby-on-rails-vs-django/. [Último acceso: 8 mayo 2025].

[14] «The Power of Django Rest Framework: Streamline Your API Development,» 18 agosto 2024. [En línea]. Available: https://medium.com/@devsumitg/the-power-of-django-rest-framework-streamline-your-api-development-387be93a8acb. [Último acceso: 8 mayo 2025].

[15] Flask, «Flask RESTful,» [En línea]. Available: https://flask-restful.readthedocs.io/en/latest/. [Último acceso: 8 mayo 2025].

[16] Tiangolo.com, «FastAPI,» 2018. [En línea]. Available: https://fastapi.tiangolo.com/. [Último acceso: 8 mayo 2025].

[17] A. Singh, «Fastapi vs. Django Rest Framework: A comprehensive comparison,» 04 noviembre 2023. [En línea]. Available: https://medium.com/@abhisheksingh_50175/fastapi-vs-django-rest-framework-a-comprehensive-comparison-c55139254b7d. [Último acceso: 8 mayo 2025].

[18] SQLite, «SQLite,» 2000. [En línea]. Available: https://sqlite.org/. [Último acceso: 8 mayo 2025].

[19] PostgreSQL Global Development Group, «PostgreSQL,» 1996. [En línea]. Available: https://www.postgresql.org/. [Último acceso: 8 mayo 2025].

[20] ScyllaDB, «ACID Database,» 01 07 2024. [En línea]. Available: https://www.scylladb.com/glossary/acid-database/. [Último acceso: 8 mayo 2025].

[21] MDN Web Docs, «Generalidades del protocolo HTTP - HTTP: MDN,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/Guides/Overview. [Último acceso: 8 mayo 2025].

[22] IBM, «¿Qué es una API REST?,» 15 abril 2025. [En línea]. Available: https://www.ibm.com/mx-es/think/topics/rest-apis. [Último acceso: 8 mayo 2025].

[23] AWS, «What is RESTful API? - RESTful API Explained - AWS,» [En línea]. Available: https://aws.amazon.com/what-is/restful-api/. [Último acceso: 8 mayo 2025].

[24] HTTPX, «HTTPX,» [En línea]. [Último acceso: 8 mayo 2025].

[25] Python Software Foundation, «requests,» 29 mayo 2024. [En línea]. Available: https://www.python-httpx.org/. [Último acceso: 8 mayo 2025].

[26] Mozilla Developer Network. (s.f.), «HTML: HyperText Markup Language: MDN,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML. [Último acceso: 8 mayo 2025].

[27] Mozilla Developer Network. (s.f), «CSS: Cascading Style Sheets: MDN,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS. [Último acceso: 8 mayo 2025].

[28] Mozilla Developer Network. (s.f), «JavaScript: MDN,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Último acceso: 8 mayo 2025].

[29] Bootstrap, «Bootstrap · The most popular HTML, CSS, and JS library in the world.,» [En línea]. Available: https://getbootstrap.com/. [Último acceso: 8 mayo 2025].

[30] J. A. G. Hernández, «Gestión de bibliotecas,» *Universidad de Murcia,* 2002.

[31] M. Cartwright, «Las bibliotecas de la antigüedad,» World History Encyclopedia, 23 julio 2019. [En línea]. Available: https://www.worldhistory.org/trans/es/2-1428/las-bibliotecas-de-la-antiguedad/. [Último acceso: 09 mayo 2025].

[32] G. Naudé, de *Advis pour dresser une bibliothèque: présenté à Monseigneur le Président de Mesmes*, Paris, Chez François Targa, 1627.

[33] R. Linares Columbie y M. A. Romero Quesada, «Gabriel Naudé y los orígenes,» *Bibliotecas. Anales de Investigación,* vol. 13, nº 1, pp. 96-102, 2017.

[34] K. Jensen, «A History of Library Hand,» 13 septiembre 2021. [En línea]. Available: https://bookriot.com/library-hand. [Último acceso: 9 mayo 2025].

[35] A. Mihelich, «A Brief History of the Library Catalog,» 10 noviembre 2021. [En línea]. Available: https://www.wccls.org/news/brief-history-library-catalog. [Último acceso: 9 mayo 2025].

[36] National Archives and Records Administration, «Online Public Access Catalog,» U.S. National Archives, [En línea]. Available: https://www.archives.gov/research/alic/tools/online-public-access-catalog. [Último acceso: 9 mayo 2025].

[37] S. -. U. d. Salamanca, «Automatización de Bibliotecas,» [En línea]. Available: https://sabus.usal.es/docu/pdf/Automat.PDF. [Último acceso: 9 mayo 2025].

[38] J. Tramullas Saz y P. Garrido Picazo, «Informatización y digitalización de bibliotecas: factores de desarrollo,» *Boletín de la ANABAD,* vol. 59, nº 4, pp. 187-200, 2009.

[39] O. Arriola Navarrete, G. Tecuatl Quechol y G. González Herrera, «Software propietario vs software libre: una evaluación de sistemas integrales para la automatización de bibliotecas,» *Investigación bibliotecológica,* vol. 25, nº 54, p. 37–70, 2024.

[40] D. Webber y A. Peters, Integrated Library Systems: Planning, Selecting, and Implementing, Santa Barbara: Libraries Unlimited, 2010.

[41] S. Picazo Cháfer, «Panorámica general del uso de Sistemas Integrados de Gestión Bibliotecaria en España,» *Métodos de información,* vol. 6, nº 10, pp. 43-56, 2015.

[42] M. d. C. Agustín Lacruz, «Bibliotecas digitales y sociedad de la información,» *Scire: Representación y Organización del Conocimiento,* vol. 2, nº 47-62, p. 4, 1998.

[43] Baratz, «Ecosistema de soluciones Absys,» 4 abril 2024. [En línea]. Available: https://www.baratz.es/software-para-bibliotecas/. [Último acceso: 10 mayo 2025].

[44] C. Baratz, «AbsysNet es el SIGB en el cual confía la mayoría de bibliotecas en España,» 18 septiembre 2019. [En línea]. Available: https://www.comunidadbaratz.com/blog/absysnet-es-el-sigb-en-el-cual-confia-la-mayoria-de-bibliotecas-en-espana/. [Último acceso: 10 mayo 2025].

[45] Baratz, «AbsysNet 2.3, evolución y revolución del SIGB más utilizado por las bibliotecas en España,» 6 mayo 2020. [En línea]. Available: https://www.baratz.es/absysnet-2-3-evolucion-y-revolucion-del-sigb-mas-utilizado-por-las-bibliotecas-en-espana/. [Último acceso: 10 mayo 2025].

[46] Ex Libris, «Integrated Library System : Aleph,» 14 noviembre 2022. [En línea]. Available: https://exlibrisgroup.com/products/aleph-integrated-library-system/. [Último acceso: 10 mayo 2025].

[47] SirsiDynix, «Symphony,» [En línea]. Available: https://www.sirsidynix.com/symphony/. [Último acceso: 10 mayo 2025].

[48] Koha-community.org, «Official Website of Koha Library Software,» 2019. [En línea]. Available: https://koha-community.org/. [Último acceso: 7 mayo 2025].

[49] Evergreen ILS, «Evergreen ILS,» 16 abril 2025. [En línea]. Available: https://evergreen-ils.org/. [Último acceso: 10 mayo 2025].

[50] L. o. Congress, «MARC 21 Format for Bibliographic Data: Table of Contents (Network Development and MARC Standards Office, Library of Congress),» 6 diciembre 2024. [En línea]. Available: https://www.loc.gov/marc/bibliographic/. [Último acceso: 10 mayo 2025].

[51] PMB Services, «PMB,» [En línea]. Available: https://www.sigb.net/index.php?lvl=cmspage&pageid=12&id_rubrique=37&opac_view=1. [Último acceso: 10 mayo 2025].

[52] WorldCat.org, «WorldCat,» [En línea]. Available: https://search.worldcat.org/es. [Último acceso: 10 mayo 2025].

[53] eBiblio Ministerio de Cultura, «Proyecto eBiblio,» [En línea]. Available: https://ebiblio.es/proyecto-ebiblio.html. [Último acceso: 10 mayo 2025].

[54] OverDrive, «Search and borrow from OverDrive's huge catalogue of eBooks, audiobooks and videos,» 2012. [En línea]. Available: https://www.overdrive.com/. [Último acceso: 7 mayo 2025].

[55] J. Alonso Arévalo y J. A. Cordón García, «Retos en torno al préstamo de libros digitales en bibliotecas.,» *Anales de Documentación,* vol. 18, nº 1, 2015.

[56] T. Giappiconi, V. Pirsich y R. Hapel, Servicios de futuro basados en Internet en las bibliotecas públicas, Fundación Bertelsmann.

[57] B. NOTICIAS, «Metodología 'Agile'. La revolución de las formas de trabajo,» 17 enero 2025. [En línea]. Available: https://www.bbva.com/es/innovacion/metodologia-agile-la-revolucion-las-formas-trabajo/. [Último acceso: 11 mayo 2025].

[58] Glassdoor, «Sueldo: Ingeniero Junior en España en Junio, 2023 | Glassdoor,» [En línea]. Available: https://www.glassdoor.es/Sueldos/ingeniero-junior-sueldo-SRCH_KO0,16.htm. [Último acceso: 11 mayo 2025].

[59] Canva, «Canva,» [En línea]. Available: https://www.canva.com/. [Último acceso: 1 junio 2025].

[60] Django Crispy Forms, «Django Crispy Forms,» [En línea]. Available: https://django-crispy-forms.readthedocs.io/en/latest/. [Último acceso: 6 junio 2025].

[61] UNDP, «Objetivos de Desarrollo Sostenible,» [En línea]. Available: https://www.undp.org/es/sustainable-development-goals. [Último acceso: 11 mayo 2025].

[62] sekhar srinivas, «Model View Controller (MVC) vs Model View Template (MVT) Difference in Process Model,» 2020. [En línea]. Available: https://www.youtube.com/watch?v=zhrLVCjNbyk. [Último acceso: 14 junio 2025].

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

Los Objetivos de Desarrollo Sostenible, también conocidos como Objetivos Globales se adoptaron en el año 2015 por la Asamblea General de las Naciones Unidas representando una llamada universal a la acción para proteger el planeta, erradicar la pobreza y asegurar la prosperidad de la población [61].

Figura 21.- Objetivos de desarrollo sostenible

Este trabajo se alinea con varios de estos objetivos, destacando entre ellos:

- **ODS 4: Educación de calidad**

  Este objetivo es el que está más relacionado con el proyecto. La plataforma web diseñada tiene como objetivo centralizar y facilitar el acceso a catálogos, eventos y recursos bibliotecarios con el fin de ampliar las oportunidades educativas. Permite a mayor número de personas y usuarios acceder a libros, recursos digitales y actividades formativas, favoreciendo el aprendizaje continuo, la alfabetización y el acceso equitativo al conocimiento.

- **ODS 5: Igualdad de género**

  Al democratizar el acceso a la información y a espacios culturales, este proyecto elimina barreras de entrada que afectan de manera desigual a mujeres y niñas, especialmente en contextos donde su acceso a recursos educativos es limitado.

- **ODS 10: Reducción de desigualdades**

  El proyecto contribuye a reducir brechas territoriales y socioeconómicas, ya que usuarios de diferentes zonas pueden acceder a servicios bibliotecarios integrados. Esto ayuda a igualar el acceso a la información, el conocimiento y los espacios educativos, independientemente del nivel socioeconómico o la localización geográfica.

- **ODS 9: Industria, innovación e infraestructura**

  Desde una perspectiva tecnológica, este trabajo impulsa la digitalización de servicios públicos y la interoperabilidad entre plataformas. Esto contribuye a la infraestructura digital de servicios culturales y educativos, y a la innovación en el acceso al conocimiento.

- **ODS 17: Alianzas para lograr los objetivos**

  El sistema parte de la idea de cooperación entre bibliotecas independientes, lo que supone una alianza institucional orientada a ampliar el alcance y la eficiencia de los servicios bibliotecarios. El intercambio de datos mediante APIs y la integración de sistemas refuerzan la colaboración interinstitucional y contribuyen a la sostenibilidad del ecosistema bibliotecario. Además, el desarrollo de esta plataforma busca fomentar alianzas con instituciones educativas, universidades y comunidades locales, facilitando la creación de una herramienta tecnológica más robusta que promueva el acceso equitativo a la información. Al integrar múltiples bibliotecas bajo un mismo sistema, se da a conocer a centros menos conocidos, permitiendo un mejor aprovechamiento de sus recursos disponibles.

ANEXO II: MOCKUP DE LA INTERFAZ

En este anexo se incluyen los mockups diseñados durante la fase inicial. Estos prototipos sirvieron de guía para el diseño posterior de la plataforma y ayudaron a conseguir una navegación clara entre página y página del sitio web.

A continuación, se muestran las páginas que forman este primer diseño, ordenadas según el flujo de navegación previsto.

Diseño de la página de inicio

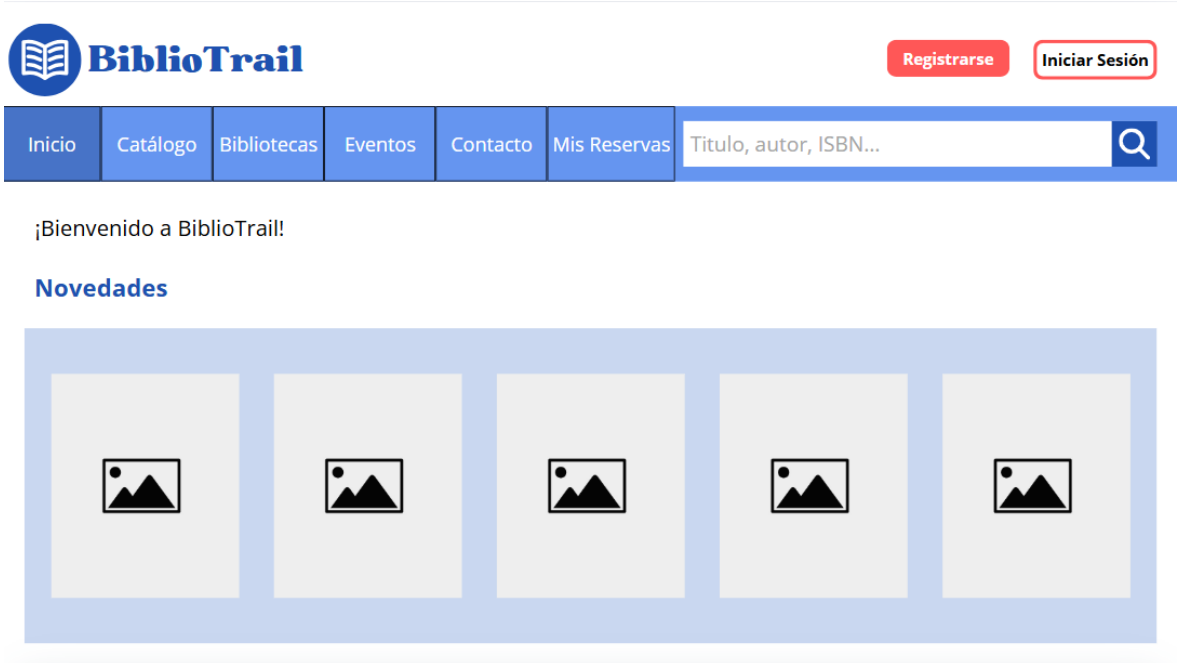


Figura 22.- Pantalla de inicio mockup

Pantalla de registro

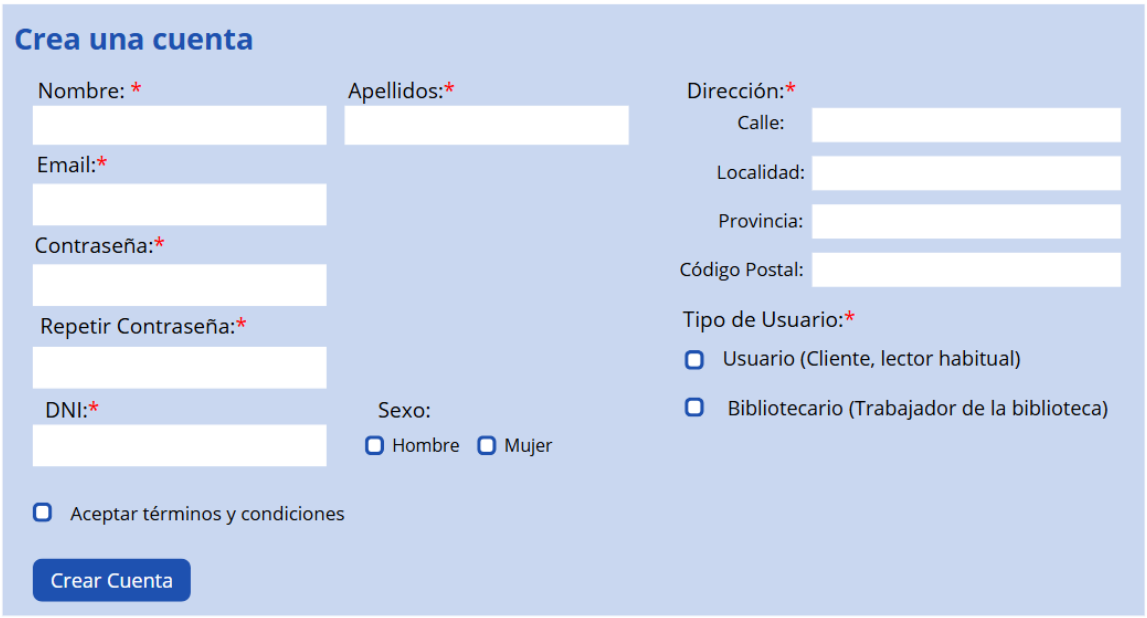


Figura 23.- Pantalla de registro mockup

Pantalla de inicio de sesión

Figura 24.-Pantalla de inicio de sesión mockup

Diseño de la página de bibliotecas

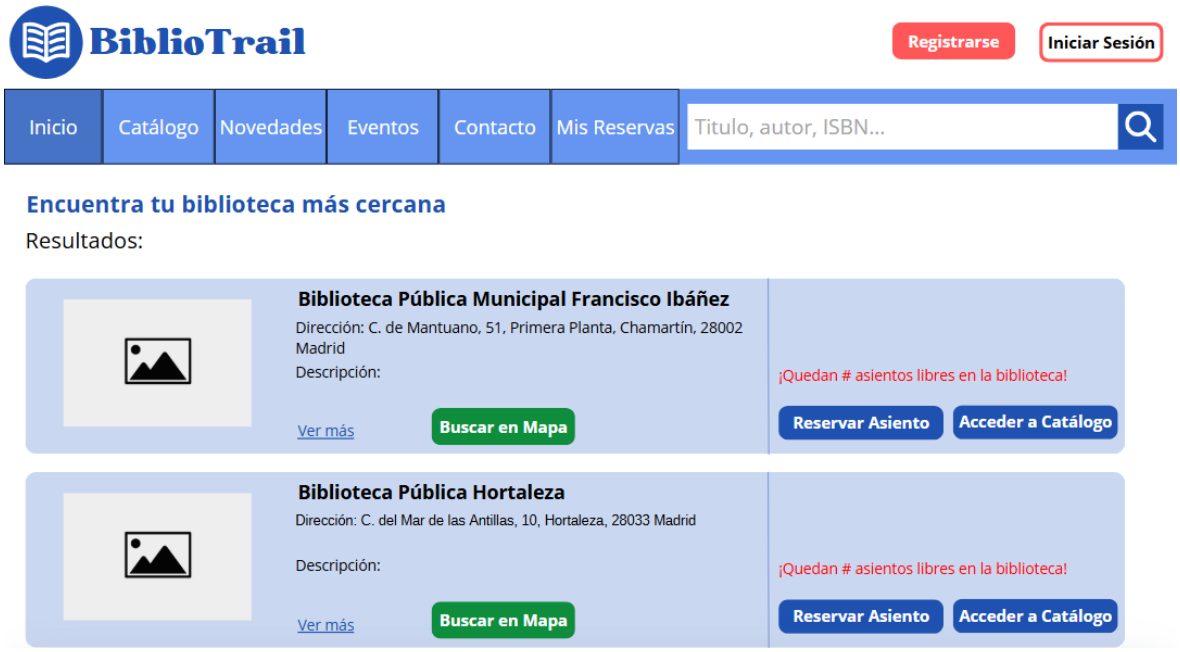


Figura 25.- Pantalla de bibliotecas mockup

Diseño del catálogo

Figura 26.- Diseño de catálogo mockup

Diseño de la sección de novedades

Figura 27.- Pantalla de novedades mockup

Diseño de página de eventos

Figura 28.- Diseño de inscripción a eventos mockup

Figura 29.- Calendario de eventos mockup

Diseño de página de contacto

Figura 30.- Pantalla de contacto mockup

Diseño de gestión de reservas

Figura 31.- Diseño de gestión de reservas mockup

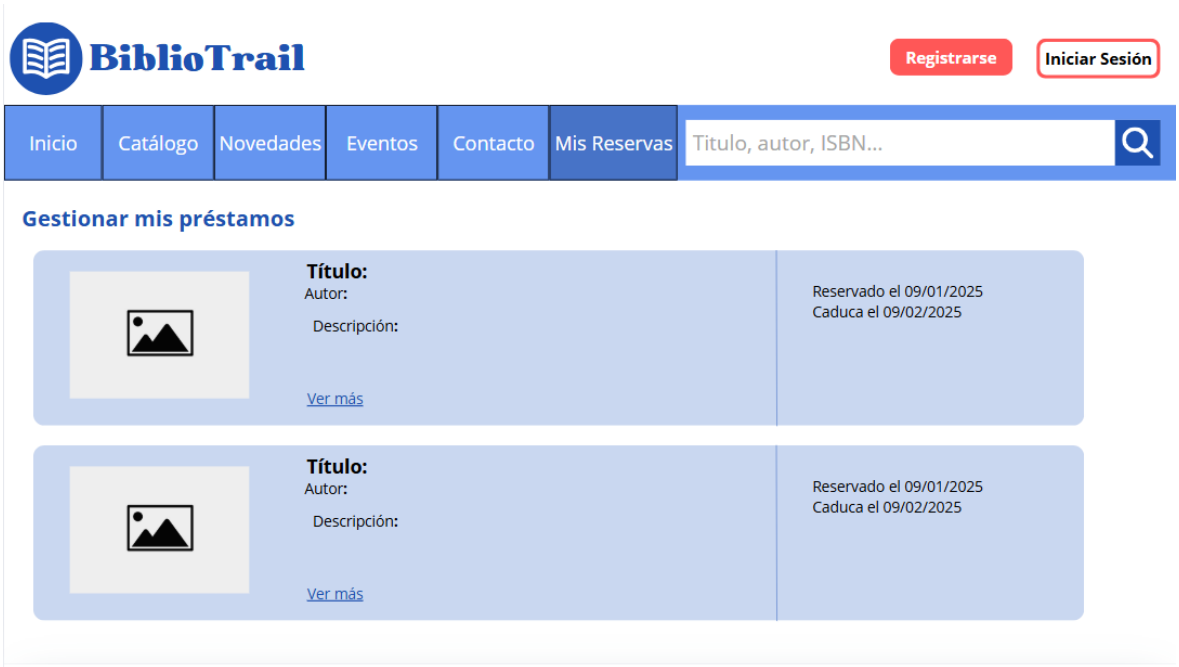


Figura 32.- Pantalla de gestión de préstamos mockup

Figura 33.- Pantalla de gestión de asientos mockup

Figura 34.- Pantalla de gestión de eventos mockup

ANEXO III: CASOS DE USO

Caso de uso 1 (CU01): Reserva de un ejemplar de un libro

CU01	**Reservar un ejemplar**		
Descripción	Un usuario autenticado accede a la vista detallada de un libro y, si hay ejemplares disponibles, solicita el préstamo de uno. El objetivo es asociar el ejemplar al usuario y registrar la reserva.		
Actores	Usuario autenticado		
Pre condiciones	Sesión iniciada Ejemplar disponible		
Post condiciones	Ejemplar bajo préstamo. Registro de préstramo creado. Nueva entrada en "Mis reservas"		
Secuencia Normal	**nº**	**Actor**	**Sistema**
	1	Realiza búsqueda en catálogo	Muestra en formato de lista los resultados de la búsqueda.
	2	Pulsa "Reservar" en la tarjeta de un libro	Verifica que el usuario esté autenticado y abre la vista detallada del libro.
	3	Abre detalle del libro	Muestra información completa del libro junto a una lista de ejempleres disponibles.
	4	Pulsa "Prestar" de un ejemplar disponible	Verifica disponibilidad del ejemplar y autenticación del usuario. Asigna el ejemplar y crea el préstamo. Actualiza "Mis Reservas".
Excepciones	nº		
	1	Si el usuario no está autenticado deberá iniciarla o registrarse para poder acceder a la vista detallada del libro y realizar la reserva.	
	2	Si no hay ejemplares disponibles, el sistema mostrará un mensaje de aviso indicándolo en la parte inferior de la vista detallada del libro que se desa reservar.	

Tabla 7.- Caso Uso 01 Reserva ejemplar

Caso de uso 2 (CU02): Reserva de espacios de trabajo

CU02	**Reservar espacio de trabajo**		
Descripción	Un usuario autenticado consulta la disponibilidad de espacios y realiza una reserva para una sala y una franja horaria concreta.		
Actores	Usuario autenticado		
Pre condiciones	Sesión iniciada Existe disponibilidad para la sala/fecha/hora seleccionada.		
Post condiciones	Sala reservada. Reserva registrada y visible en “Mis Reservas”.		
Secuencia Normal	**nº**	**Actor**	**Sistema**
	1	Selecciona biblioteca, fecha, capacidad	Filtra y muestra tabla de disponibilidad.
	2	Pulsa en una franja disponible	Muestra detalles y botón de reserva.
	3	Confirma la reserva	Guarda reserva y bloquea franja. Actualiza "Mis Reservas".
Excepciones	nº		
	1	Si hay un problema con la API externa, se mostrará un error en el sistema y habrá posibilidad de reintentar o volver atrás.	
	2	Si no hay espacios disponibles o la sala esta completa se mostrará en rojo como ocupada o ya reservada.	

Tabla 8.- Caso Uso 02 Reserva espacio trabajo

Caso de uso 3 (CU03): Inscripción a evento

CU03	**Inscripción a evento**		
Descripción	Un usuario autenticado se inscribe en un evento de una biblioteca asociada, siempre que haya plazas disponibles.		
Actores	Usuario autenticado		
Pre condiciones	Sesión iniciada Evento con plazas disponibles.		
Post condiciones	Usuario inscrito. Plazas disponibles actualizadas. Actualización de "Mis Reservas" y base de datos.		
Secuencia Normal	**nº**	**Actor**	**Sistema**
	1	Filtra eventos por biblioteca	Muestra lista de eventos.
	2	Pulsa “Inscribirse” en un evento	Verifica autenticación y plazas. Crea inscripción y actualiza plazas.
	3	Confirma la inscipción	Muestra mensaje de inscrito y actualiza “Mis Reservas”.
Excepciones	nº		
	1	Si no hay plazas disponibles o el usuario ya se ha inscrito en ese mismo evento se deshabilita el botón de la inscripción y el ususario no podrá incribirse en el evento.	
	2	Si existe algún fallo con la API externa, muestra error técnico y permite reintentar.	

Tabla 9.- Caso Uso 03 Inscripción evento

ANEXO IV: MANUAL DE USUARIO

1.- CÓMO ACCEDER AL SITIO WEB DE BIBLIOTRAIL

Para acceder a la plataforma web, el usuario debe acceder en su navegador a la URL http://127.0.0.1:8000/ o http://localhost:8000/. Una vez accedido, el usuario se encontrará en la página de inicio:

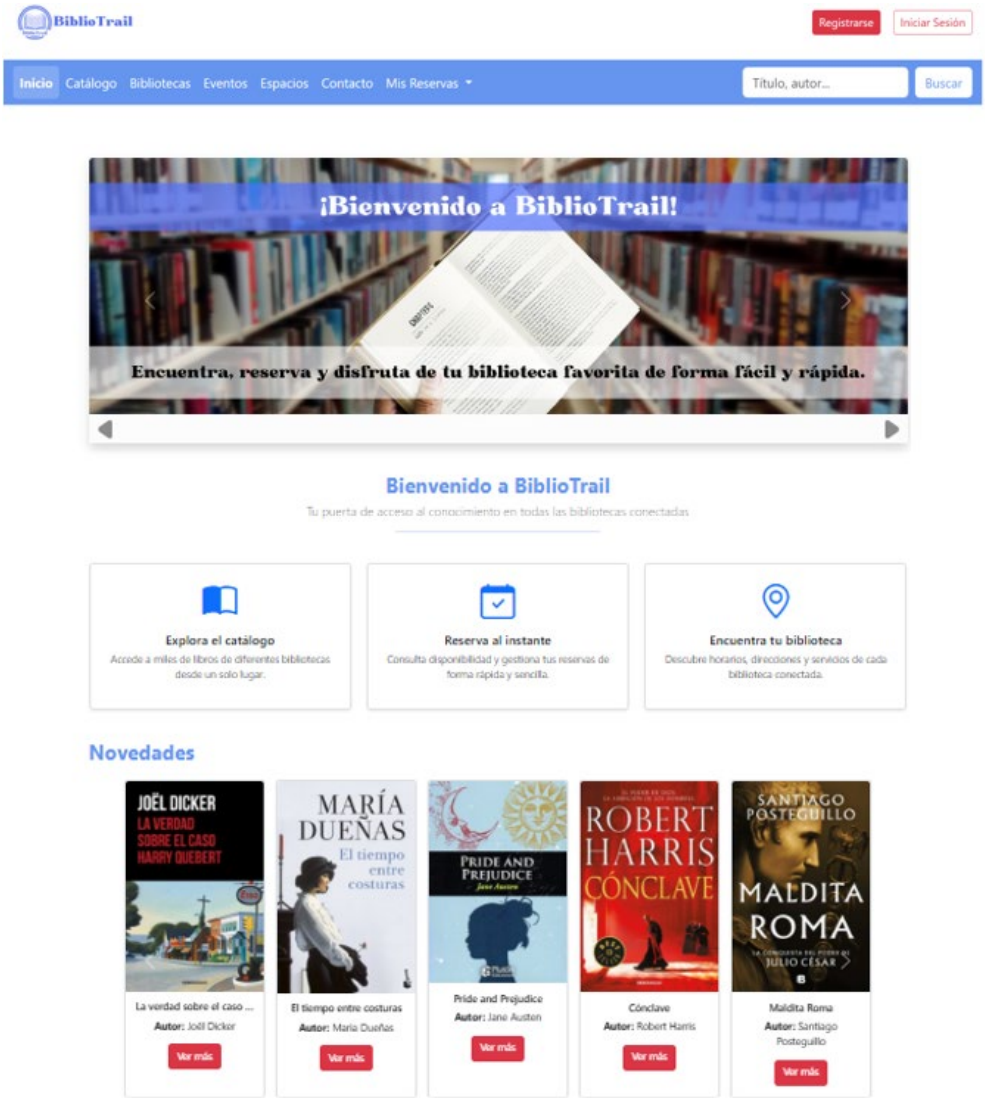
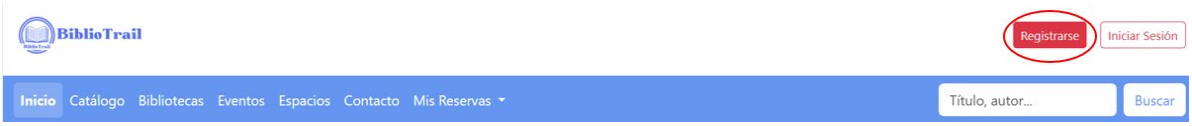


Figura 35.- Página de inicio

CÓMO REGISTRARSE

Para registrarse en la página web es necesario pulsar el botón de "Registrarse", situado en la esquina superior derecha de la página.



Una vez pulsado el usuario deberá rellenar el siguiente formulario de registro y presionar el botón inferior para registrarse en el sistema.

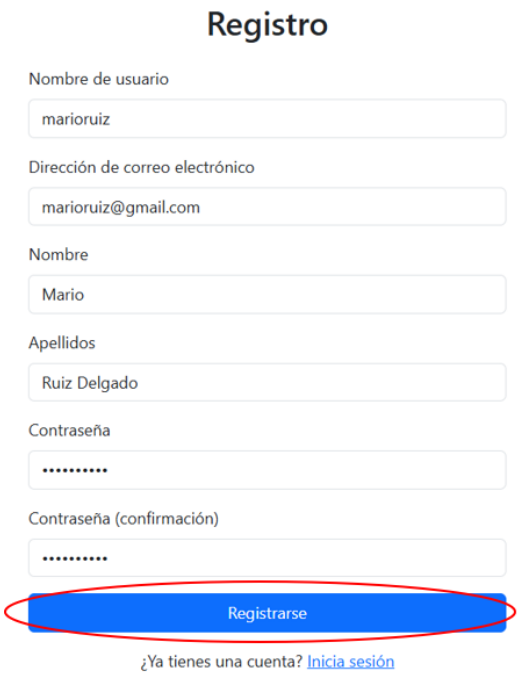


Figura 36.- Formulario de registro

Si el registro se ha realizado con éxito, el usuario será enviado a la página de inicio y podrá ver que en la esquina superior derecha, se ha creado automáticamente un perfil con su nombre de usuario:

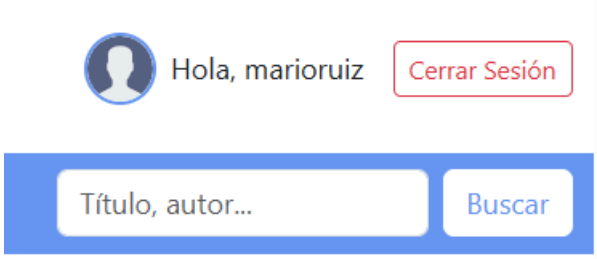


Figura 37.- Icono perfil usuario

Al presionar el icono el usuario podrá modificar los datos de su perfil de usuario.

CÓMO MODIFICAR EL PERFIL DE USUARIO

Para modificar el perfil, el usuario debe presionar el icono situado en la esquina superior derecha, tal y como muestra la Figura 37.

Al presionarlo podrá ver la siguiente imagen:

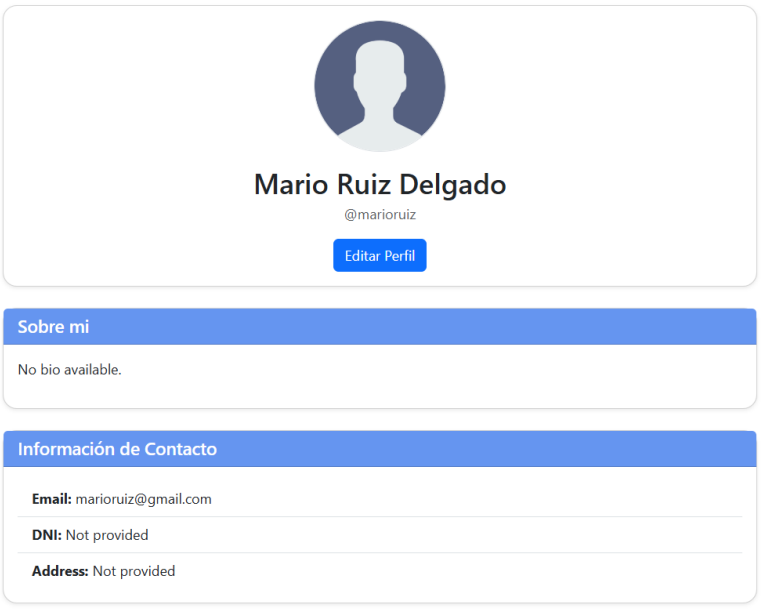


Figura 38.- Editar perfil

Para editar los datos del usuario, se debe presionar el botón "Editar Perfil". Al presionarlo, el usuario podrá modificar datos como su nombre de usuario, correo electrónico, nombre, apellido, DNI, dirección y cambiar su foto de perfil, pudiendo añadir una de su galería.

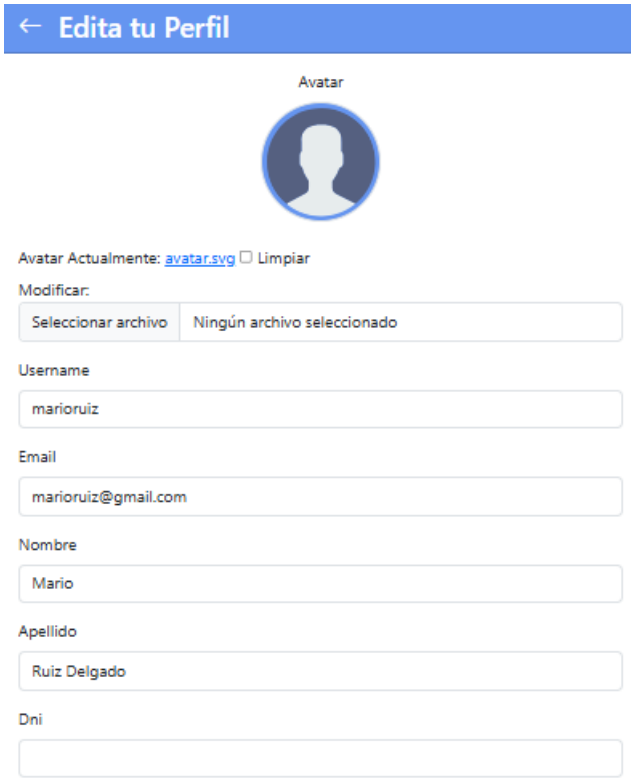


Figura 39.- Formulario editar perfil

Una vez modificada la información y presionado el botón de "Actualizar" se guardarán automáticamente en la base de datos la información del usuario. Si los campos se han rellenado correctamente, se retornará al usuario a la vista anterior.

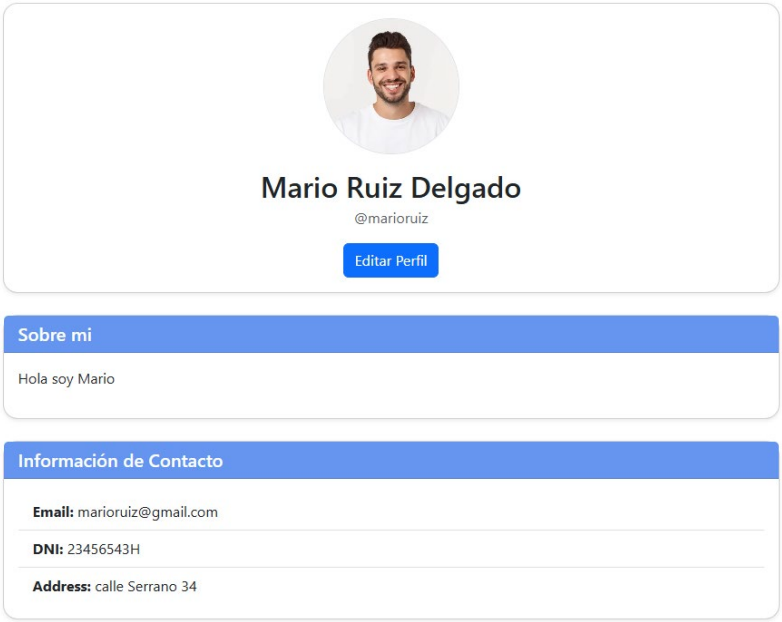
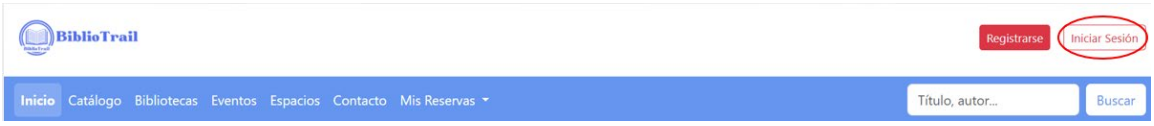


Figura 40.- Perfil editado

CÓMO INICIAR SESIÓN

Para iniciar sesión, el usuario debe presionar el botón de "Iniciar sesión", situado en la esquina superior derecha junto al botón de registro.



Una vez presionado el usuario deberá rellenar el formulario mostrado en la con su nombre de usuario y contraseña.

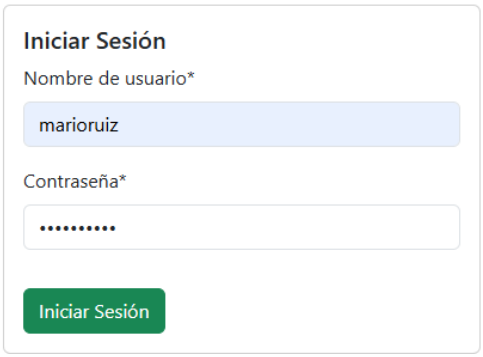


Figura 41.- Inicio de sesión

Si las credenciales del usuario son incorrectas, se le comunicará con un mensaje de error el campo en el que se ha producido el fallo de inicio de sesión.

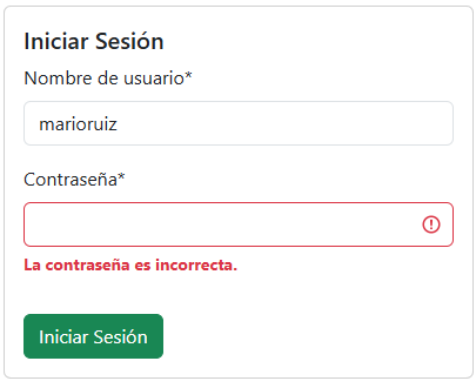


Figura 42.- Fallo de inicio de sesión

Por el contrario, si las credenciales del usuario son correctas, el usuario quedará autenticado y aparecerá en la vista del inicio de la página web.

CÓMO CONSULTAR EL CATÁLOGO Y RESERVAR LIBROS

CÓMO REALIZAR BÚSQUEDAS EN EL CATÁLOGO

Para acceder al catálogo se debe presionar la segunda pestaña del menú de navegación, la pestaña “Catálogo”.

Al hacer clic sobre ella, se accederá a una página que muestra en formato de lista todos los libros ofrecidos por el agregador. En la parte superior hay un buscador, donde el usuario podrá realizar búsquedas en función del título del libro que desea reservar o su autor. Además, justo debajo se le permite filtrar la búsqueda por biblioteca y por género literario.

Figura 43.- Catálogo principal web

Una vez que el usuario ya haya rellenado el campo de búsqueda y los filtros, deberá presionar el icono azul de buscar. Adicionalmente, el usuario podrá buscar también en el buscador situado en el menú de navegación.

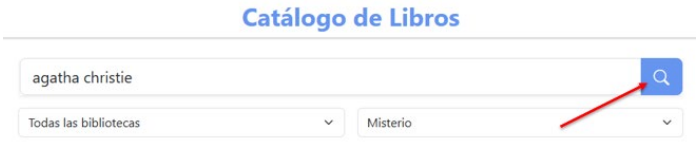


Figura 44.- Filtros de búsqueda de libros

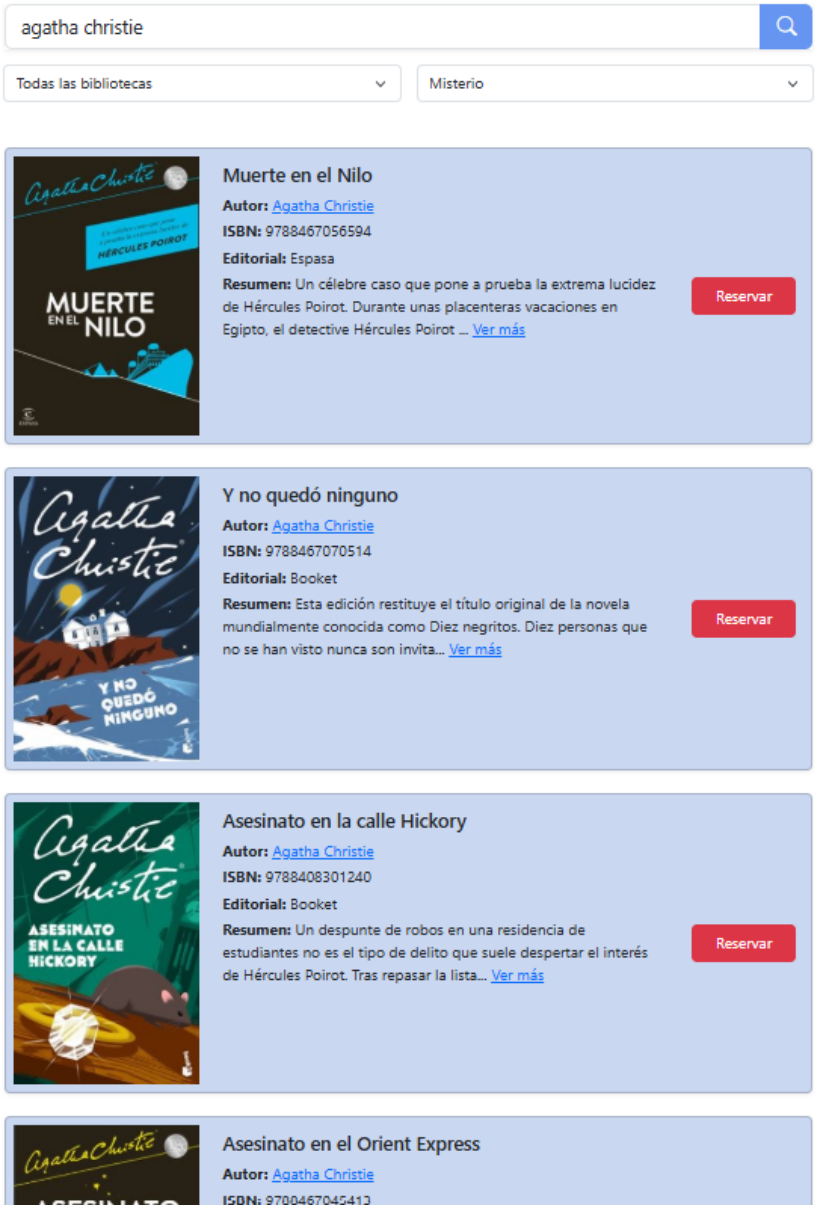


Figura 45.- Búsqueda filtrada

Al hacerlo, se mostrarán los resultados coincidentes con la búsqueda realizada.

CÓMO REALIZAR RESERVAS DE EJEMPLARES

Para reservar un ejemplar, el usuario debe presionar sobre el botón de "Reservar", situado dentro de la ficha del libro correspondiente.

Figura 46.- Reserva ficha libro

Una vez presionado, se mostrará una vista detallada del libro que se desea tomar bajo préstamo.

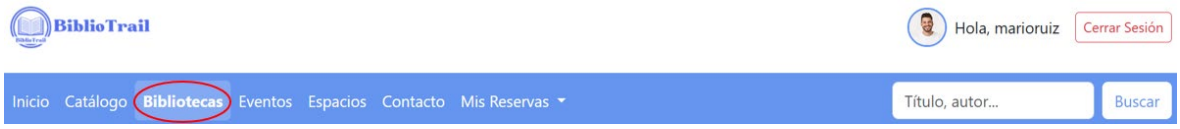
Figura 47.- Vista detallada libro

Esta vista recoge en la parte superior información relacionada con el autor y el libro. Justo debajo, se incluye un resumen o sinopsis del libro. Por último, debajo aparece una lista de los ejemplares disponibles para ese libro. Si el usuario desea tomar prestado alguno de ellos, debe pulsar el botón de "Prestar".

Una vez pulsado el usuario volverá automáticamente a la vista del catálogo y podrá comprobar si su reserva se ha tramitado correctamente en la pestaña de "Mis Reservas" (situada en el menú de navegación), en la sección de "Préstamos de libros".

CÓMO CONSULTAR LAS BIBLIOTECAS ASOCIADAS

Para acceder a la información más relevante de cada biblioteca asociada con *BiblioTrail* el usuario debe presionar la pestaña de "Bibliotecas" del menú de navegación.



Una vez presionada, se mostrará una página con los datos de cada biblioteca. Entre ellos se encuentran:

- Dirección
- Teléfono
- Email
- Horario
- Total de libros de la biblioteca
- Total de ejemplares
- Localización en "Google Maps" de la biblioteca

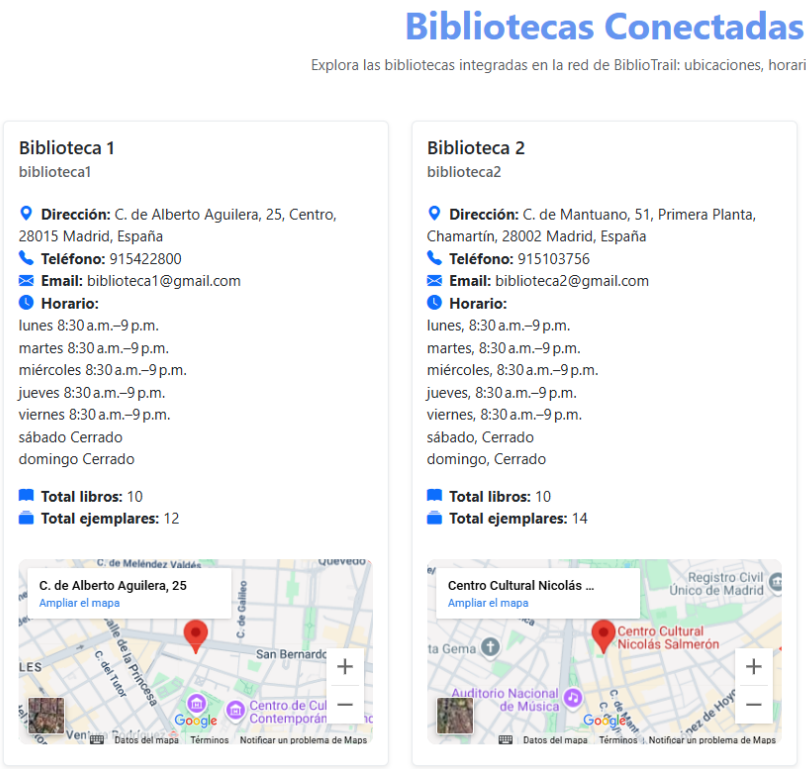
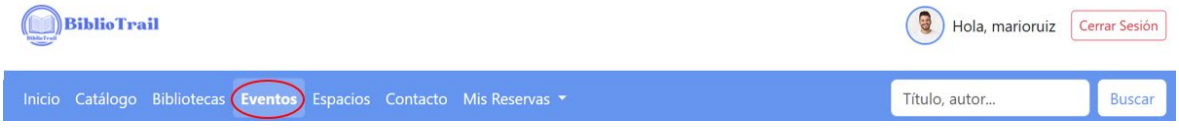


Figura 48.- Vista bibliotecas

CÓMO INSCRIBIRSE A UN EVENTO

Para inscribirse a un evento el usuario debe presionar la pestaña de "Eventos" del menú de navegación.



En esta sección se muestra un listado de los eventos organizados por todas las bibliotecas. En la parte superior izquierda de la pantalla se le permite al usuario filtrar en función de la biblioteca en la que se realizará el evento.

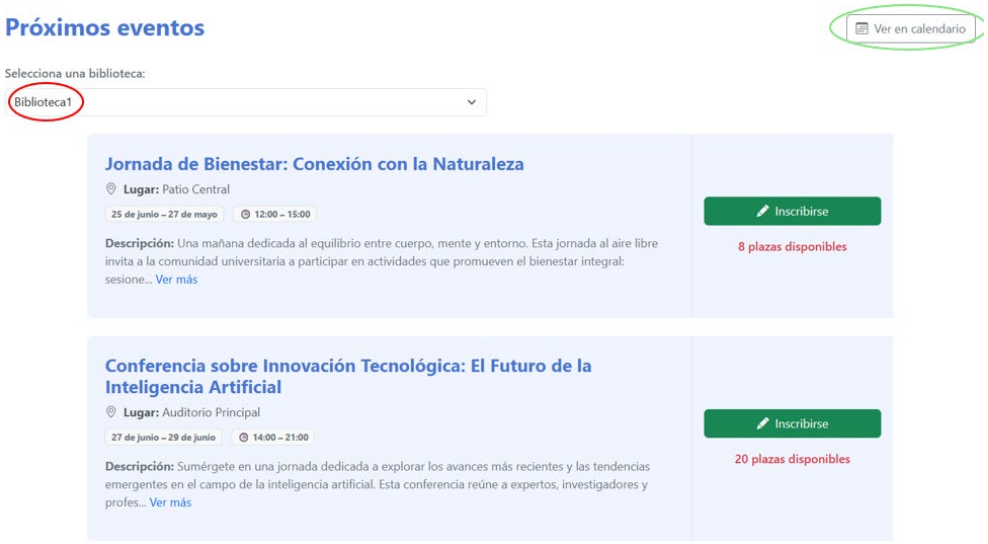


Figura 49.- Vista de eventos

En cada ficha del evento se muestra el título del evento, el lugar en el que se realizará, la fecha y hora, y una breve descripción.

En la parte superior derecha se le permite al usuario acceder a una vista en modo de calendario de los eventos organizados por la biblioteca seleccionada. Para acceder a él se debe presionar el botón “Ver en calendario”.

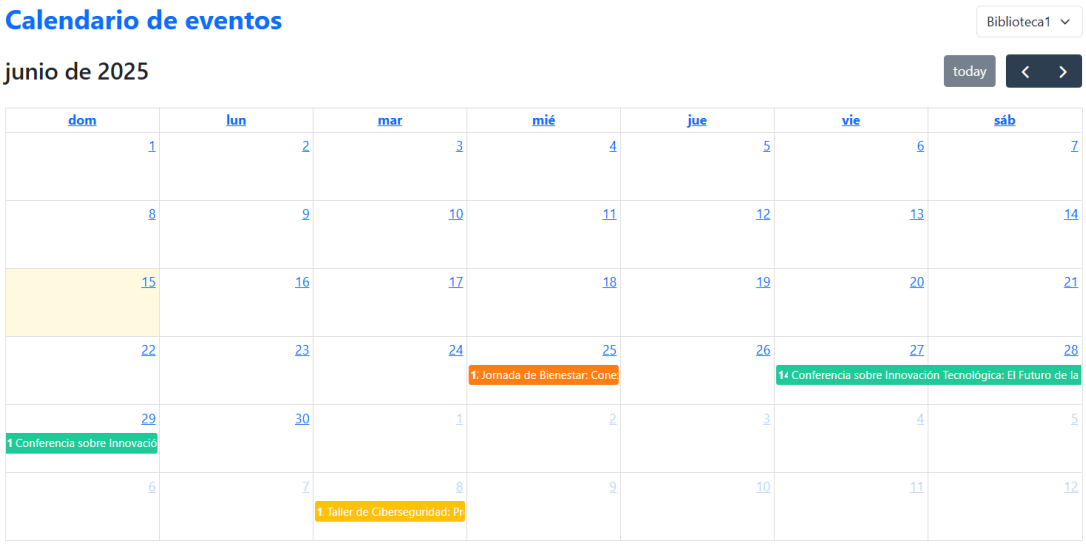


Figura 50.- Calendario de eventos

Al igual que en la pantalla anterior, en esta vista se le permite al usuario filtrar por biblioteca. Además, al presionar en un evento se le mostrará en una ventana emergente la información más relevante del evento seleccionado.

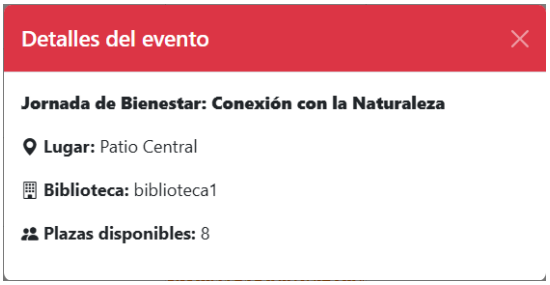


Figura 51.- Ventana emergente calendario eventos

Para inscribirse a un evento, el usuario debe retroceder a la pantalla anterior y presionar el botón de "Inscribirse".

Figura 52.- Ficha de evento

Una vez presionado el usuario quedará correctamente inscrito a ese evento. Para consultar su inscripción el usuario podrá acceder a la sección de "Inscripciones a eventos" dentro de "Mis Reservas" (del menú de navegación).

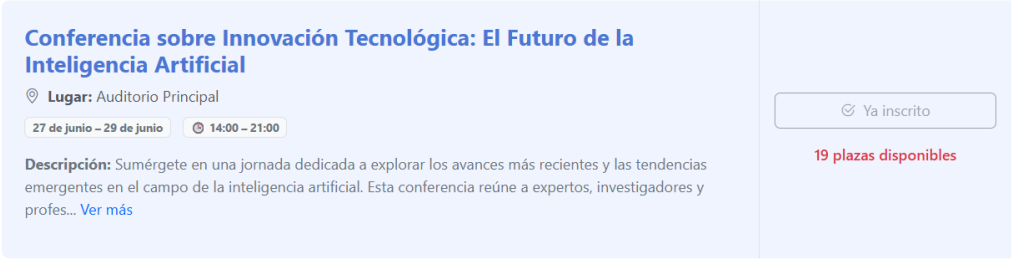
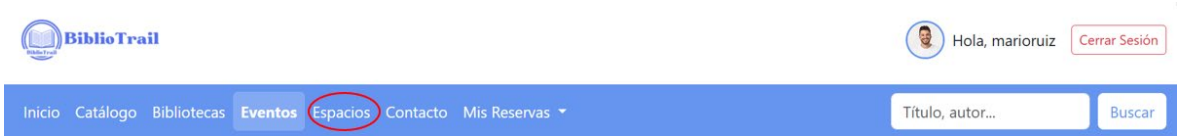


Figura 53.- Inscripción correcta evento

CÓMO RESERVAR ESPACIOS DE TRABAJO

Para reservar espacios de trabajo el usuario debe presionar la pestaña de "Espacios" situada en el menú de navegación.



Una vez en ella, se muestra un sistema de consulta de disponibilidad de salas de trabajo donde el usuario podrá filtrar en función de la biblioteca, la capacidad mínima de la sala que quiere reservar y la fecha. Una vez rellenados esos campos deberá presionar el botón de "Buscar disponibilidad".

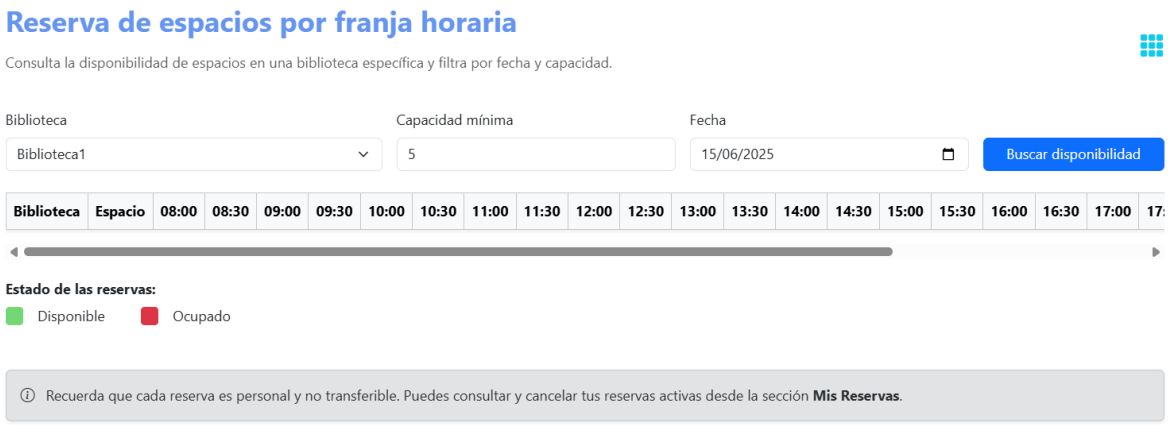


Figura 54.- Sistema de consulta de disponibilidad

Al hacerlo aparecerá una tabla con la disponibilidad horaria de cada sala de trabajo de la biblioteca escogida. En esta tabla se muestran en color rojo las salas reservadas para esa franja horaria determinada y en verde las disponibles.

Figura 55.-Tabla de disponibilidad por franja horaria

Para reservar una sala el usuario deberá presionar sobre la celda correspondiente a la hora y sala que desee. Una vez presionada la casilla, aparecerá una ventana emergente para confirmar la reserva del espacio de trabajo.

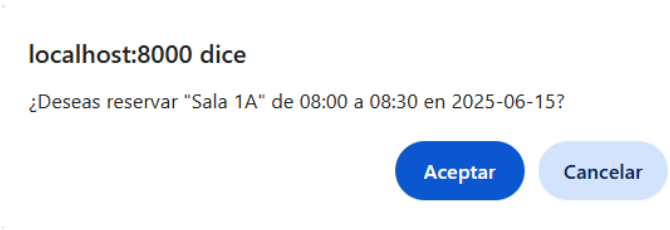


Figura 56.- Ventana emergente confirmación reserva

Al presionar "Aceptar" la reserva se realizará de forma automática y quedará guardada en la base de datos.

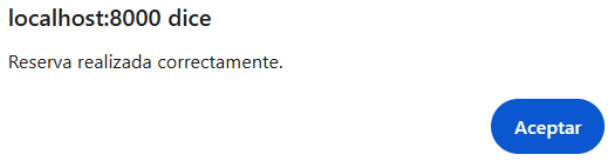
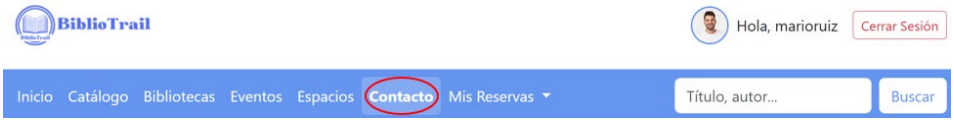


Figura 57.- Notificación de éxito

Además, se cambiará el color de la celda a rojo para indicar el cambio de estado. Para consultar la reserva el usuario podrá hacerlo en la sección de "Espacios de trabajo", dentro de la pestaña de "Mis Reservas".

CÓMO CONTACTAR CON EL SOPORTE DE LA PLATAFORMA

Para enviar cualquier consulta, queja, reclamación o pregunta al soporte técnico de la plataforma el usuario deberá presionar la pestaña de "Contacto".



En ella se mostrará un formulario de contacto que el usuario podrá enviar al equipo de la plataforma.

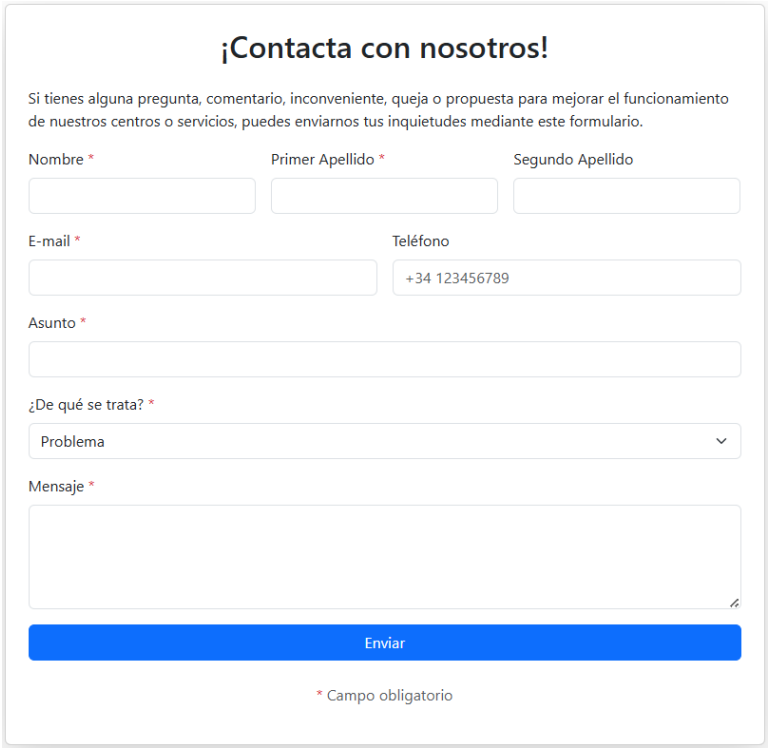
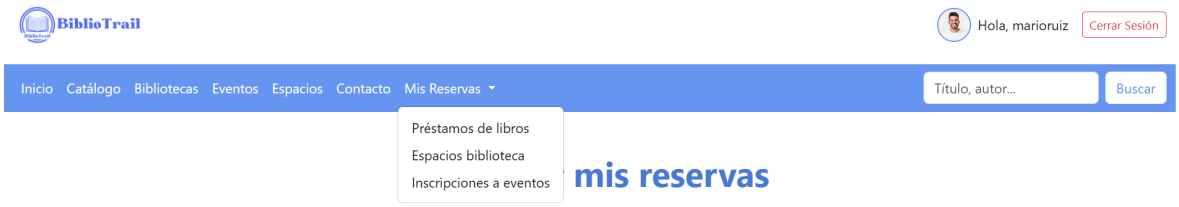


Figura 58.- Formulario de contacto

CÓMO GESTIONAR LAS RESERVAS DEL USUARIO

Para gestionar las reservas del usuario hay que acceder a través de la pestaña de "Mis Reservas" del menú de navegación.



En ella se muestra un menú desplegable para cada una de las secciones de gestión del usuario. Otra forma de acceder a cada una de ellas será mediante la pantalla principal.

Figura 59.- Pantalla principal de mis reservas

GESTIÓN DE PRÉSTAMOS

Para gestionar los préstamos del usuario es necesario acceder a la pestaña de "Préstamos de libros" o bien, a la sección de "Mis préstamos".

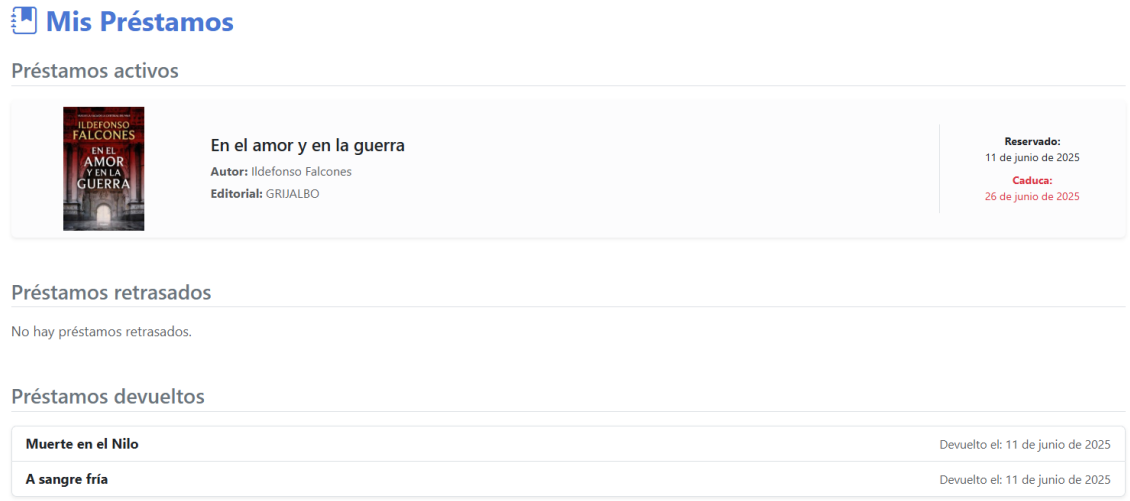


Figura 60.- Sección de mis préstamos

En ella se puede observar los préstamos activos por el usuario, así como información importante como el día en el que se realizó la reserva y la fecha en la que caduca el préstamo.

Si se supera la fecha límite, el préstamo se mostrará en la sección de "Préstamos retrasados". Además, en la sección inferior se puede ver también el historial de préstamos realizados por el usuario autenticado.

GESTIÓN DE EVENTOS

Para gestionar las inscripciones del usuario es necesario acceder a la pestaña de "Inscripciones a eventos" o bien, a la sección de "Mis eventos" de la página principal.

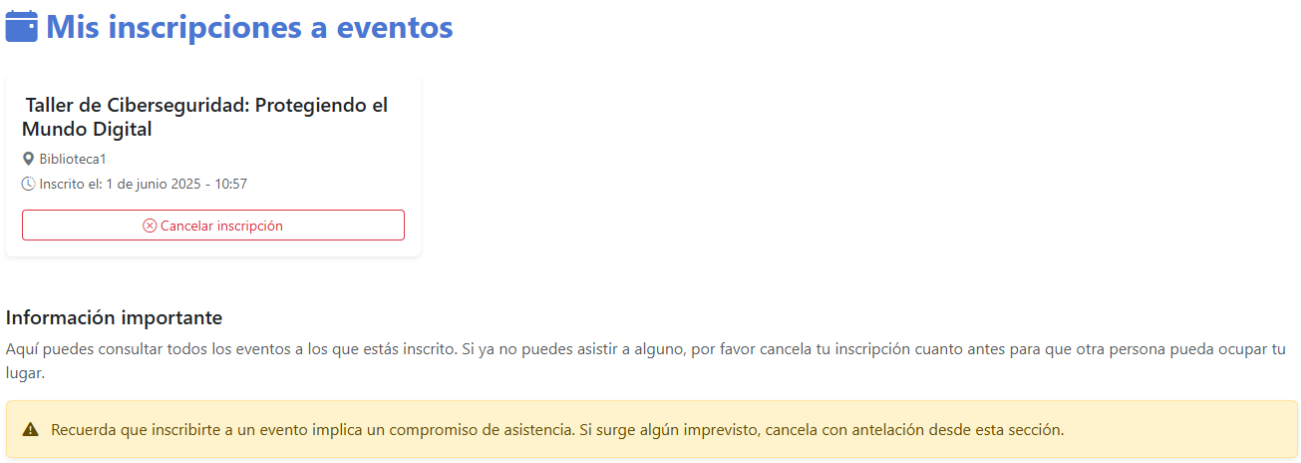


Figura 61.- Mis eventos

En esta sección se pueden gestionar las inscripciones a eventos del usuario, apareciendo cada reserva en modo de tarjeta con información sobre la localización, el título del evento y la fecha de inscripción. Además, se le permite al usuario la opción de cancelación. Para ello hay que pulsar el botón de "Cancelar inscripción".

GESTIÓN DE ESPACIOS

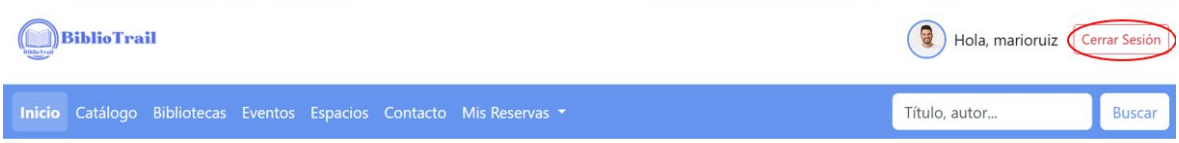
Para gestionar las reservas de espacios de trabajo es necesario acceder a la pestaña de "Espacios biblioteca" o bien, a la sección de "Mis espacios".

Figura 62.- Reservas de espacios

Al acceder a esta sección se puede observar una tabla con todas las reservas de salar realizadas por el usuario. En ella aparece información sobre la sala, biblioteca, fecha de reserva, franja horaria y un botón para cancelar la reserva.

CÓMO CERRAR SESIÓN

Para cerrar sesión en la plataforma únicamente hay que pulsar el botón de "Cerrar Sesión", situado en la esquina superior derecha.



CÓMO ACCEDER AL PANEL DE ADMINISTRACIÓN

Para acceder al panel de administración el usuario debe acceder en su navegador a la URL http://127.0.0.1:8000/admin

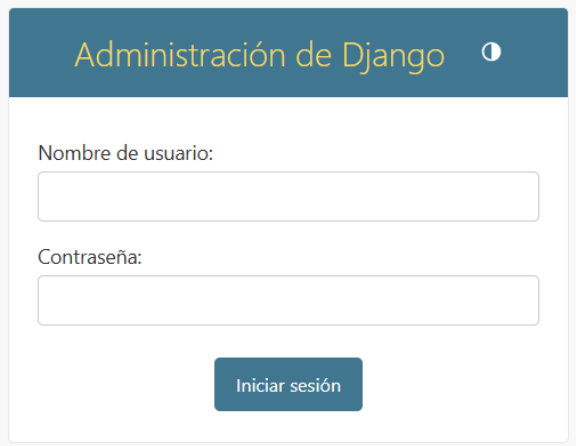


Figura 63.- Inicio sesión panel administrador

Una vez en él, el usuario debe iniciar sesión con su nombre de usuario y contraseña. Para acceder al panel únicamente podrán hacerlo los usuarios que cuentan con los permisos de administrador o empleado (`is_staff`).

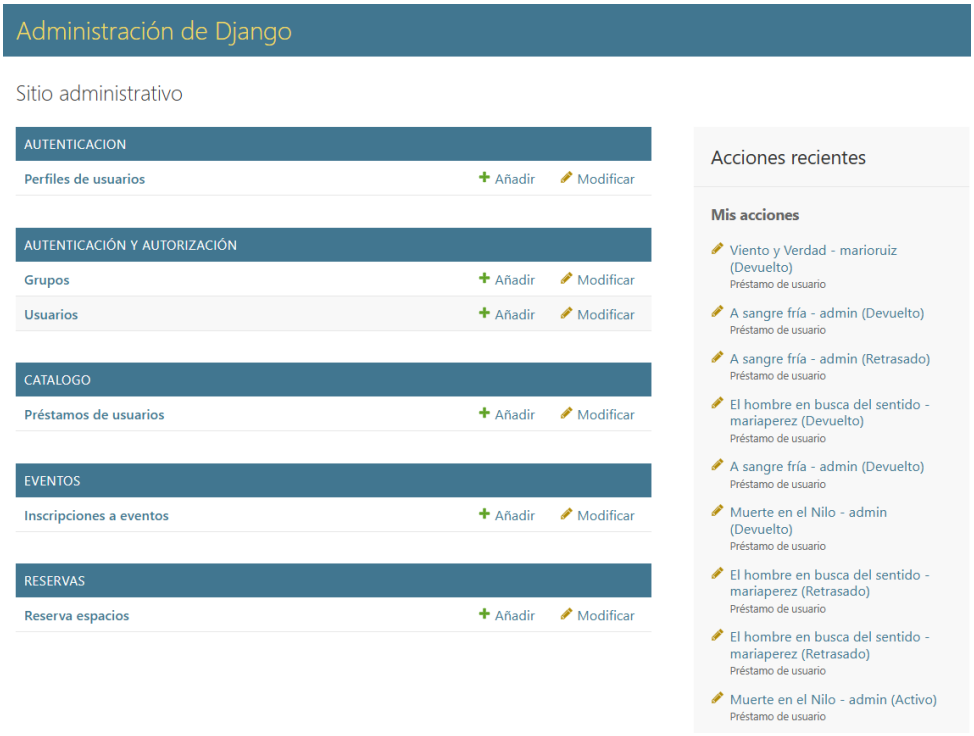


Figura 64.- Panel de administración

Una vez autenticado, el administrador podrá modificar datos relacionados con las aplicaciones de autenticación, usuarios, catálogo, eventos y reservas.

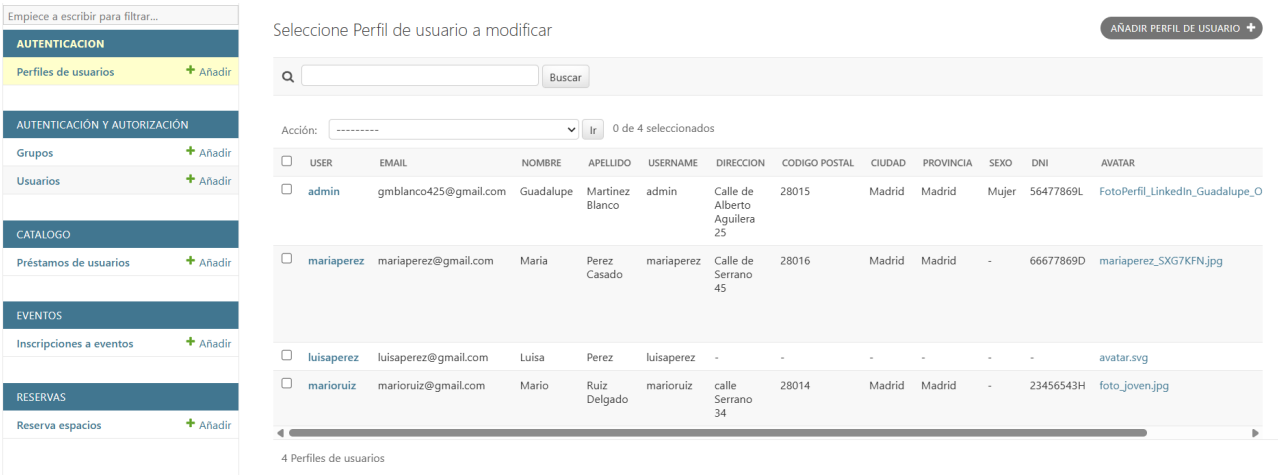


Figura 65.- Gestión perfiles de usuarios

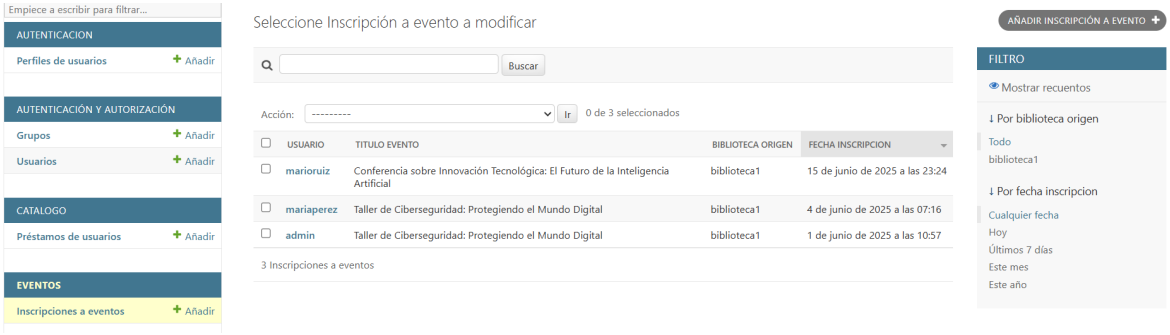


Figura 66.- Gestión de eventos

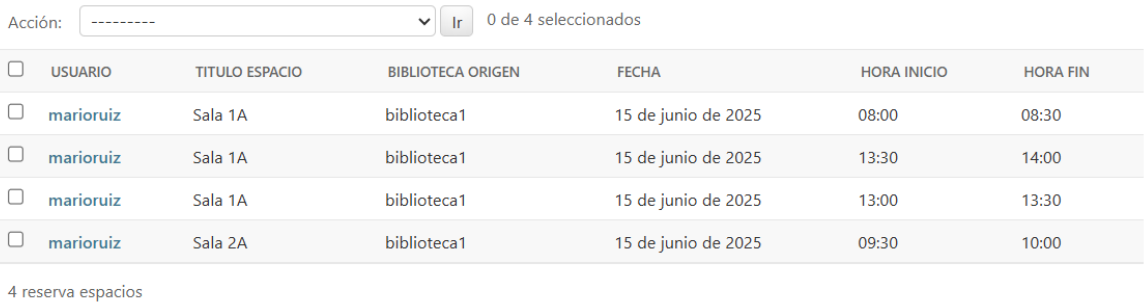


Figura 67.- Gestión de espacios de trabajo

En cada una de estas secciones se le permite al administrador añadir manualmente o modificar reservas del usuario, préstamos bibliotecarios, inscripciones a eventos, información personal o permisos y espacios de trabajo.

CÓMO GESTIONAR DEVOLUCIONES DE PRÉSTAMOS

Al producirse una reserva física de un ejemplar, el usuario podrá devolverlo a la biblioteca donde se ha tomado bajo préstamo el libro correspondiente.

Figura 68.- Gestionar devoluciones de préstamos

Una vez allí, el bibliotecario al recibir el ejemplar de vuelta en la institución deberá modificar el préstamo activo del usuario. Para hacerlo deberá hacer clic en el ejemplar correspondiente y añadir la fecha de devolución del préstamo en la casilla del formulario correspondiente.

Por último, deberá guardar los cambios. De esta forma el estado cambiará automáticamente de "Activo" o "Retrasado" a "Devuelto", quedando registrado en la base de datos y en la plataforma web.

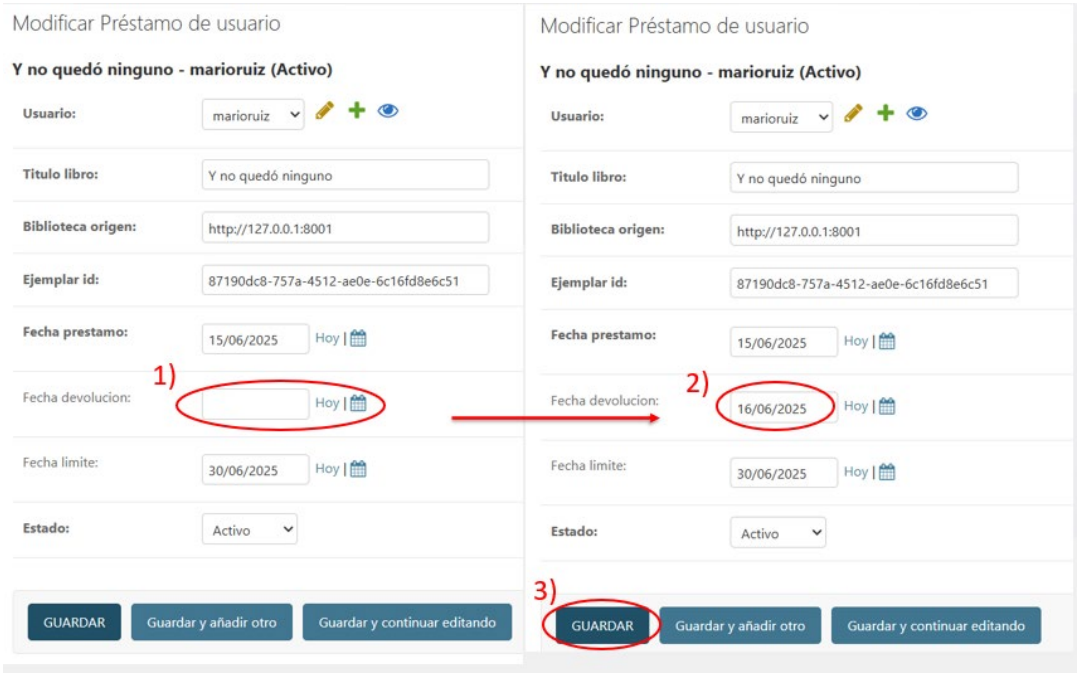


Figura 69.- Modificación préstamo devuelto