



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**TFG GII 23.23 Web de libros  
y sistema de clasificación**



Presentado por Daniel Fernández Fernández  
en Universidad de Burgos  
a 11 de junio de 2024  
Tutora: Ana Serrano Mamolar



## Resumen

La educación de los niños es fundamental para su correcto desarrollo y la posibilidad de un futuro brillante.

En este aspecto, la literatura infantil constituye uno de los recursos más adecuados para la aproximación al conocimiento de realidades sociales y culturales. En el campo de la prehistoria es a menudo bastante común la reproducción de estereotipos de género que sin embargo la ciencia ya ha descartado definitivamente. Es muy importante por tanto para docentes y familias descubrir literatura infantil que no contribuyan a perpetuar estos estereotipos sino que muestren una realidad más alineada con los datos avalados por la ciencia. Con este enfoque hay diferentes investigaciones que abordan la evaluación de los libros de prehistoria y su adecuación en la etapa infantil.

En este trabajo, se propone una plataforma que sirva de catálogo disponible para cualquier docente o familia a la hora de escoger qué libros de prehistoria ofrecer a las niñas y niños así como ofrecer mecanismos de evaluación derivados de investigaciones para la auto-evaluación de nuevos títulos.

Adicionalmente, esta plataforma web incluye un apartado de administración que proporciona todas las herramientas necesarias para un desarrollo eficiente y colaboración en la integración de nuevos títulos. Entre las funcionalidades se encuentran la gestión de cuentas, permisos, copias de seguridad del catálogo y la búsqueda automática de libros en tres fuentes diferentes.

La web está disponible ininterrumpidamente en el siguiente enlace: [Prehistoria en igualdad](#).

## Descriptores

Aplicación web, full-stack, web scraping, libros, flask, base de datos, angular, python

## Abstract

The education of children is essential for their correct development and the possibility of a bright future.

In this aspect, children's literature constitutes one of the most appropriate resources for approaching knowledge of social and cultural realities. In the field of prehistory, the reproduction of gender stereotypes is often quite common, although science has already definitively discarded them. It is therefore very important for teachers and families to discover children's literature that does not contribute to perpetuating these stereotypes but rather shows a reality more aligned with data supported by science. With this approach, there are different investigations that address the evaluation of prehistory books and their suitability in the childhood stage.

In this work, a platform is proposed that serves as a catalog available to any teacher or family when choosing which prehistory books to offer to girls and boys, as well as offering evaluation mechanisms derived from research for the self-assessment of new titles. .

Additionally, this web platform includes an administration section that provides all the necessary tools for efficient development and collaboration in the integration of new titles. Features include account management, permissions, catalog backups, and automatic search for books in three different sources.

The website is available continuously at the following link: [Pre-historia en igualdad](#).

## Keywords

Web app, full-stack, web-scraping, books, flask, database, angular, python

---

# Índice general

---

Índice general	iii
Índice de figuras	v
Índice de tablas	vi
1. Introducción	1
2. Objetivos del proyecto	4
3. Conceptos teóricos	6
3.1. Web scraping . . . . .	6
3.2. Json Web Token (JWT) . . . . .	8
3.3. Estimación de la adecuación didáctica de los libros . . . . .	12
4. Técnicas y herramientas	14
4.1. Metodologías . . . . .	14
4.2. Frameworks utilizado para el desarrollo del proyecto . . . . .	17
4.3. Desarrolllo de prototipo Web . . . . .	20
4.4. Hosting de los archivos . . . . .	20
4.5. Herramientas para la escritura de la memoria . . . . .	22
4.6. Otras herramientas usadas . . . . .	24
4.7. Análisis Comparativo entre Google Books API y Amazon Books API . . . . .	24
5. Aspectos relevantes del desarrollo del proyecto	27
5.1. Metodologías Aplicadas . . . . .	27
5.2. Formación . . . . .	28

5.3. Inicio del proyecto . . . . .	29
5.4. Desarrollo del <i>frontend</i> . . . . .	31
5.5. Desarrollo del <i>backend</i> . . . . .	35
5.6. Despliegue de la aplicación . . . . .	40
5.7. Testing . . . . .	42
5.8. Uso de Postman . . . . .	44
<b>6. Trabajos relacionados</b>	<b>46</b>
6.1. Atribuciones de género y construcción de identidades en la literatura infantil sobre prehistoria . . . . .	46
6.2. Discord . . . . .	47
<b>7. Conclusiones y Líneas de trabajo futuras</b>	<b>48</b>
7.1. Conclusiones . . . . .	48
7.2. Líneas de trabajo futuras . . . . .	49
<b>Bibliografía</b>	<b>50</b>

---

# Índice de figuras

---

3.1. Archivo robots de Agapea . . . . .	7
3.2. Header . . . . .	10
3.3. Payload . . . . .	10
3.4. Signature . . . . .	11
4.1. <i>milestone</i> con Issues incluidas . . . . .	16
4.2. Tablero de Github . . . . .	17
5.1. Tablero Kanban . . . . .	28
5.2. Logo de la web . . . . .	30
5.3. Prototipo del catálogo . . . . .	31
5.4. Administración antigua . . . . .	33
5.5. Panel de Administración . . . . .	34
5.6. Módulos del proyecto . . . . .	35
5.7. Funciones web scraping . . . . .	36
5.8. Leer Más Agapea . . . . .	37
5.9. Función Ajax Agapea . . . . .	38
5.10. Modal Usuario sin Permisos . . . . .	39
5.11. Interfaz Deploys Netlify . . . . .	41
5.12. Pantalla Deploys Render . . . . .	41
5.13. Variables de entorno Render . . . . .	42
5.14. Ejemplo llamada con Postman . . . . .	45
6.1. Permisos Discord . . . . .	47

---

# Índice de tablas

---

4.1. Herramientas y tecnologías utilizadas en cada parte del proyecto	26
---	----



---

# 1. Introducción

---

«Durante más de siglo y medio, las interpretaciones que se han hecho de los restos arqueológicos han contribuido en gran medida a invisibilizar a las mujeres prehistóricas, sobre todo al reducir su importancia en la economía», según Marylène Patou-Mathis [24].

En el contexto actual, donde la digitalización de la información juega un papel crucial en numerosos sectores, este trabajo de fin de grado introduce una aplicación web innovadora enfocada en el ámbito específico de la literatura infantil prehistórica. Esta aplicación está diseñada para facilitar el almacenamiento, la gestión y la clasificación de libros infantiles que abordan la temática prehistórica, con un enfoque particular en la representación de los roles de género conforme a los descubrimientos científicos más recientes.

La literatura infantil prehistórica ofrece una ventana única al pasado, permitiendo a los jóvenes lectores explorar cómo vivían, interactuaban y se organizaban las sociedades antiguas. Sin embargo, es crucial que estas representaciones sean fieles a los avances científicos actuales en cuanto a los roles de género, evitando perpetuar estereotipos desfasados o inexactitudes históricas.

Nuestra aplicación aborda esta necesidad al proporcionar una plataforma donde los libros infantiles sobre la prehistoria pueden ser catalogados y mostrados según su precisión y representación de los roles de género. Esto incluye una evaluación de cómo cada libro representa las dinámicas de género en contextos prehistóricos, asegurando que reflejen los conocimientos y descubrimientos científicos más recientes.

Una característica distintiva de esta herramienta es su capacidad para destacar libros que promueven una comprensión informada y actualizada de los roles de género en la prehistoria. Esto es especialmente valioso para el

personal de bibliotecas, docentes y familias que buscan ofrecer a los niños y niñas una perspectiva adecuada y didáctica, que fomente el pensamiento crítico y la comprensión de la diversidad y la igualdad de género desde una edad temprana.

Además, la aplicación no solo sirve como un repositorio de información, sino que también actúa como una guía para seleccionar material de lectura que esté alineado con los hallazgos científicos actuales, proporcionando así una base sólida para la educación y la sensibilización sobre los roles de género en diferentes épocas históricas, empezando por la prehistoria.

## Estructura de la memoria

Este proyecto consta de dos documentos, una memoria y un documento de anexos, los cuales se complementan para documentar completamente este proyecto. La estructura de la memoria es la siguiente:

**Introducción:** Descripción inicial del problema y la solución creada. Estructura de la memoria y los materiales adjuntos.

**Objetivos del proyecto:** Listado de los objetivos a cumplir durante la realización del proyecto.

**Conceptos teóricos:** Explicación de los conceptos teóricos clave implementados en la solución propuesta.

**Técnicas y herramientas:** Técnicas y herramientas utilizadas o consideradas para el desarrollo del proyecto.

**Aspectos relevantes del desarrollo:** Consideración de los aspectos destacables que tuvieron lugar durante la realización del proyecto.

**Trabajos relacionados:** Proyectos que he utilizado para poder tomar referencias y desarrollar la aplicación.

**Conclusiones y líneas de trabajo futuras:** Conclusiones obtenidas tras la finalización del proyecto y opciones de desarrollo a futuro disponibles.

Además de la estructura de la memoria, procedo a comentar la estructura de los anexos:

**Plan del proyecto:** Planificación temporal del desarrollo del proyecto y estudio de viabilidad económico y legal del proyecto.

**Requisitos:** Se contempla el listado de requisitos funcionales así como el diagrama de casos de uso explicado.

**Diseño:** Se mencionan las diferentes partes del diseño de la aplicación, como el diseño de los datos y el diseño arquitectónico.

**Manual del programador:** Incluye los aspectos más importantes del código fuente y una guía detallada de los pasos a realizar en caso de querer continuar con el desarrollo.

**Manual de usuario:** Manual de usuario para el manejo eficiente de la aplicación.

## Materiales Adjuntos

A continuación se muestra un listado con los materiales adjuntos a este documento:

- *Frontend* desarrollado en Angular, utilizado para la interfaz de usuario y se encarga de realizar las llamadas necesarias a la API.
- API REST en Python que contiene el *backend*. Contiene toda la lógica necesaria en base a los requisitos funcionales establecidos en los anexos.
- Aplicación web del proyecto Prehistoria en igualdad desplegada donde el equipo de administración ha comenzado con la carga de libros y datos reales.<sup>1</sup>
- Repositorio del proyecto que contiene todo el desarrollo del proyecto.<sup>2</sup>

---

<sup>1</sup>Acceso a la [Web desplegada](#)

<sup>2</sup>Acceso al [repositorio de GitHub](#)

---

## 2. Objetivos del proyecto

---

En este apartado se van a exponer los diferentes objetivos planteados inicialmente en el proyecto divididos en varios apartados.

### Objetivos generales

- Dar a conocer los resultados de investigaciones y metodologías desarrolladas por el Didáctica de la Historia y de las Ciencias Sociales de la Universidad de Burgos.
- Desarrollar una página web para la consulta de libros sobre la prehistoria aptos para todos los públicos en base a criterios científicamente fundamentados sobre roles de género.
- Proveer herramientas de gestión de la web para las personas que administren la página.
- Generar herramientas de seguridad para el control y limitación de las personas colaboradoras de la web.
- Almacenar todos los datos en una base de datos persistente y correctamente estructurada.
- Importar y exportar datos de la web con facilidad para asegurar su conservación, sincronización y uso de estudios externos.

## Objetivos técnicos

- Implementar *web scraping* y consultas a APIs públicas, junto a una lógica para trabajar con los datos en el *backend* de la web. Todo esto se utilizaría para la búsqueda en distintas fuentes bibliográficas.
- Crear una calculadora web que permita estimar en base a una metodología validada el porcentaje de realidad científica existente centrado en roles de género.
- Implementación de diferentes roles de uso de la aplicación para limitar el acceso, de manera dinámica, a ciertas acciones de administración de la página.
- Utilizar un sistema de control de versiones como GitHub.
- Utilizar metodologías ágiles como *SCRUM* y herramientas de organización de trabajo como *Kanban*.
- Implementar y validar el interfaz web utilizando el *framework* de Python Flask.
- Publicación del proyecto en un servicio como Netlify o Render para su acceso público.

## Objetivos personales

- Formarme en el desarrollo web.
- Formarme en el desarrollo de una aplicación full-stack.
- Brindar herramientas que ayuden a mejorar la educación.
- Aplicar los conocimientos adquiridos durante la carrera y las prácticas curriculares.

---

## 3. Conceptos teóricos

---

Dentro de este proyecto existen elementos o implementaciones que pueden llegar a tener una mayor complejidad teórica. Por ese mismo motivo, en este apartado se van a detallar en diferentes secciones los aspectos más complejos que han surgido durante el desarrollo.

### 3.1. Web scraping

El *web scraping* [27] es una técnica usualmente utilizada para poder obtener de forma automática y estructurada los contenidos de las páginas web para su utilización en diferentes fines. En el caso de este proyecto, para obtener los datos detallados de los libros deseados.

Esta técnica puede resultar muy beneficiosa debido a que se pueden obtener grandes volúmenes de datos minimizando las posibilidades de error frente a una búsqueda manual, la cuál es más lenta y costosa.

Aunque este sistema de obtención de datos sea muy útil y eficiente, antes de aplicarlo a una página web para la obtención de los datos, es imprescindible comprobar que esta página nos da la autorización para realizarlo.

Para realizar esta comprobación de manera legal y sencilla, y así evitar posibles problemas legales, es fundamental consultar el archivo *robots.txt* [11] de la página web. Este archivo es un protocolo de exclusión de robots que define a qué URLs de una página web pueden acceder los rastreadores o bots. Al acceder a la URL de la página web */robots.txt*, se puede verificar si la web permite o restringe el acceso a ciertos servicios y redireccionamientos para realizar *web scraping*.

El archivo *robots.txt* proporciona una lista de reglas que indican a los rastreadores web qué partes del sitio deben evitar, ayudando así a prevenir accesos no autorizados y proteger la integridad del sitio web. Este protocolo especifica, mediante directivas, qué áreas de la página web no deben ser rastreadas y puede incluir secciones como *'Disallow'* para bloquear acceso a determinadas rutas o archivos y *'Allow'* para permitir acceso a otras partes.



```
← → ↻ agapea.com/robots.txt

User-agent: Nutch
Disallow: /

User-agent: Spiderbot
Disallow: /

User-agent: Spiderbot/Nutch-1.7
Disallow: /

User-agent: MJ12bot
Disallow: /

User-agent: Mediapartners-Google
Disallow: /generarPedido.php

User-agent: ZoominfoBot
Disallow: /

User-agent: magpie-crawler
Disallow: /

User-agent: *
Crawl-delay: 5
Disallow: /registro-novedades.php
Disallow: /datosClientes/*
Disallow: /usuarios/*
Disallow: /usuariosV2/*
Disallow: /nuevoRegistro.php
Disallow: /entrarRegistro.php
Disallow: /recomendar.php
Disallow: /recordarContrasena.php
Disallow: /funcionesAjax.inc.php
Disallow: /ajax/funcionesAjaxResumen.inc.php
Disallow: /ajax/funcionesAjaxComentarios.inc.php
Disallow: /ajax/funcionesAjaxCesta.inc.php
Disallow: /buscar/buscador.php?IDSesion
Disallow: /catnew.php
Disallow: /bases-concurso.php
Disallow: /nota-legal.php
Disallow: /politica-cookies.php
Disallow: /condiciones-de-contratacion.php
Disallow: /confirmar-pedido.php
Disallow: /politica-privacidad.php
Noindex: *?px
```

Figura 3.1: Archivo robots de Agapea

En la Figura 3.1 podemos observar el archivo *robots.txt* de la web de la librería Agapea <sup>3</sup>, la cual hemos utilizado en el desarrollo de este proyecto. Tal como se muestra, las primeras líneas hacen referencia a distintos rastreadores y servicios específicos los cuales no pueden acceder a las URL marcadas en el apartado de "disallow" (En este caso, los servicios que aparecen no tienen permisos para acceder a ninguna URL perteneciente a esta web).

Debajo de estas primeras líneas observamos otro grupo de líneas que son las que indican cuáles son las URL a las que no está permitida la entrada o utilización de ningún tipo de servicio o rastreador, indistintamente del tipo que sea.

Otras formas de bloqueo existentes que podrían afectar a la utilización de esta herramienta son las siguientes:

- *CAPTCHAs*: Bloqueos simples para poder distinguir entre un robot y un humano.
- Limite de peticiones: Estableciendo un límite no se podrían obtener los datos de una forma tan rápida, ya que la propia web lo bloquea al superar el límite de peticiones.
- Bloqueo por IP: Las páginas web pueden contener una lista negra de usuarios que no puedan acceder a la web, esto se hace a través de listas de IPs.

Una vez se establecen las limitaciones y se tienen en cuenta en la aplicación del web scraping, se pueden utilizar diferentes herramientas para implementarlo, como bibliotecas o frameworks, que permiten un desarrollo correcto de análisis de los datos de las páginas web deseadas. En secciones posteriores se describen las utilizadas para este proyecto.

## 3.2. Json Web Token (JWT)

La tecnología JWT es una tecnología incluida en el estandar RFC 7519 [9] que permite enviar y recibir información de forma compacta y rápida dentro de un elemento JSON. Este elemento es muy seguro debido a que se encuentra firmado digitalmente.

---

<sup>3</sup>[Enlace a robots.txt](#)



## Estructura de un token JWT

Para poder realizar las firmas y generar las claves, se suele utilizar una clave privada y única a partir de la cual se generan el resto, comúnmente utilizando el algoritmo SHA256.

En el caso de este proyecto, esta tecnología se ha utilizado para poder limitar el acceso a personas no autorizadas a los recursos de la API que han de ser privados y solo gestionados por aquellas personas que tengan la autorización pertinente.

Un token de acceso JWT tiene un patrón de construcción, ya que se divide en 3 partes separados por ".", los cuales se describirán a continuación [20].

- Header
- Payload
- Signature

A continuación, se va a ir analizando un ejemplo de JWT, para ello utilizaremos un debugger de JWT [8]:

Header:

`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9`

En el header indica principalmente dos elementos, el tipo de algoritmo que se ha usado para generarlo, y que tipo de token es. En este caso se ha utilizado HS256.



Figura 3.2: Header

Payload:

```
eyJzdWIiOiIxMjMONTY3ODkwIiwibmFtZSI6IkhbmllbCBGZXJw6FuZGV6IiwiaWF0IjoxNTE2MjM5MDIyfQ
```

El apartado del payload contiene informaciones como el nombre de usuario, la fecha de la creación del token, y a quién se refiere el token.

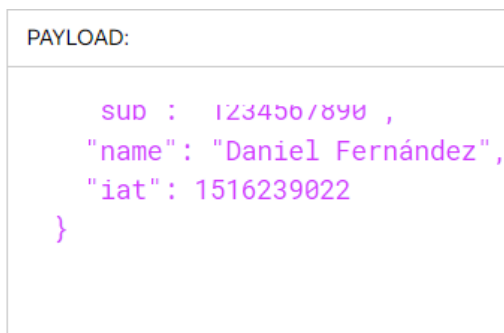


Figura 3.3: Payload

Signature:

```
RrvHMGQTYf15y5_WgdoH1HULZDRYukEmb5iMgAIw0k
```

Este último apartado es la firma del token, este apartado se forma con el algoritmo especificado en el apartado de *header* junto con una clave secreta que se defina por el administrador y los apartados anteriores codificados.



Figura 3.4: Signature

## Funcionamiento de un JWT

Cada vez que una persona usuaria desea acceder a un recurso protegido con JWT, tiene que realizar una petición al *backend* para que le retorne una clave nueva, si tenía una existente, no es necesario hacer la petición siempre y cuando ese token no expire. En el caso de este proyecto, el token está configurado para proteger información de los usuarios registrados, además de no permitir la modificación ni obtención del catálogo. El token para poder realizar estas acciones se obtiene de manera invisible al usuario cuando realiza la operación de inicio de sesión.

Al obtener este token, es almacenado en el local storage del navegador, por lo que el usuario no necesita en ningún momento manipularlo ni editarlo.

Una vez obtenido, al realizar una petición al *backend* será necesario incluir en la cabecera de la llamada el token generado. Esto se realiza internamente desde el *frontend*, obteniendo el token de la zona de almacenamiento e incluyéndolo. Es importante mencionar que no siempre es necesario mandar al *backend* el token, ya que en este proyecto no todas las llamadas lo requieren.

Tras este envío al servidor, el *backend* realiza dos comprobaciones:

- Fecha de expiración
- Validez del token

Si ambas comprobaciones resultan satisfactorias, el servidor responde a la petición correctamente.

### 3.3. Estimación de la adecuación didáctica de los libros

Este proyecto contiene un apartado que genera una estimación en un rango de 0 a 100 indicando al usuario la calidad de un libro introducido en términos de realidad científica en perspectiva de género. Este proceso de estimación se ha basado en la metodología publicada por Jesús Alberto San Martín Zapatero y Delfín Ortega Sánchez [28]. Para realizar estos cálculos, el *frontend* muestra al usuario un formulario donde tiene que rellenar los siguientes campos:

- Campo de selección si existe masculino genérico.
- Cuatro campos donde se deben de introducir el número de personas adultas y menores existentes en el libro.
- Por cada uno de los géneros, se seleccionan las actividades que realizan de entre las opciones de una lista.
- La ubicación en la que se ambienta el libro.

Dentro de este formulario todos los campos son obligatorios exceptuando el de los menores, ya que es posible que no aparezcan, en cuyo caso la estimación se reajusta para adaptarse. Más detalles de la metodología de evaluación pueden encontrarse en [28].

Una vez obtenidos estos datos se envían al *backend* que va realizando las siguientes comprobaciones para ir ajustando la estimación.

1. Obtiene el campo del masculino genérico y si la respuesta es negativa, se añaden 20 puntos a la nota final.
2. En el caso de los contadores de personas, se realiza una media de equilibrio entre géneros tanto para personas adultas como para menores de forma independiente para obtener cómo de igualados están los dos géneros. Idealmente tendrían que aparecer en la misma proporción o muy similar, lo que daría un resultado de 15 puntos para las personas adultas y 15 puntos para los menores. Si la proporción se encontrase desbalanceada, la nota sería menor ya sea en personas adultas o en menores.

En el caso de que el número total de menores existentes en el formulario sea el valor 0, la media se ajusta para no tener en cuenta este grupo y valorar sobre 30 puntos a las personas adultas.

3. Las actividades candidatas se encuentran contenidas en una tabla de la base de datos a las cuales los responsables de la administración pueden entrar y variar las actividades existentes y realizar una gestión completa. Por lo que, al llegar al *backend*, los elementos seleccionados se cruzan con todas las actividades existentes y se obtienen de las comunes si le corresponde dar una puntuación extra o no por realizar esa acción (Una acción poco común para ese género, por lo tanto se premia). En base a estas comprobaciones se asignan hasta 20 puntos de la nota final de estimación.
4. Finalmente, el apartado de ubicaciones es un selector con unos valores fijos que se asignan y se envían en base a la elección del usuario. Cuanta más diversidad exista en la ubicación del libro, más cerca estará la puntuación de los 30 puntos de este apartado.

Una vez terminado el análisis completo de los datos introducidos, se realiza una suma de todas las notas de cada apartado y se envía en escala 0-100, donde 0 es un libro muy poco adecuado y 100 es un libro que se corresponde a la perfección con la evidencia científica. Tras esto, el *frontend* muestra al usuario un botón donde puede registrar su respuestas para que posteriormente los personas colaboradoras y responsables de la administración de la web puedan analizarlo y considerar agregar ese libro de cuya estimación ha sido realizada.

---

## 4. Técnicas y herramientas

---

### 4.1. Metodologías

#### Scrum

La metodología Scrum [21] es un marco de trabajo que es comúnmente usado para la gestión del desarrollo de un proyecto software y permite realizar entregas incrementales de valor en vez de realizar una única entrega en un periodo mayor de tiempo. Este marco de trabajo se fundamenta en separar los proyectos en entregas más pequeñas llamadas *sprints*. En el caso de los *sprints*, no existe una duración estándar, si no que al iniciar el proyecto, se decide esa duración. Para este proyecto los *sprints* han sido de 2 semanas de manera general, y de manera excepcional 3 semanas.

Todo sprint se inicia realizando una planificación de las tareas del proyecto que se van a realizar durante su duración y, al terminar el sprint, se realiza una nueva reunión para revisar el estado de las tareas previstas para el sprint y una retrospectiva que permite detectar ineficiencias y errores para su posterior solución y mejora en el rendimiento del proyecto.

#### Kanban

Kanban [13] es un sistema de apoyo visual cuyo objetivo es mejorar la eficiencia de los procesos a realizar mediante una gestión rápida del estado de las tareas. Esta metodología originaria de la marca de vehículos Toyota <sup>4</sup> consiste en una tabla visual en la que aparecen distintas columnas, siendo cada una de ellas un estado decidido por el equipo y no hay un estándar fijo. Comúnmente son las siguientes:

---

<sup>4</sup>Enlace a la [historia de kanban en Toyota](#)

- Pendiente de realizar
- En progreso
- Completado

Dentro de cada una de las columnas se encuentran todas las tareas pertenecientes al sprint actual, repartidas según su estado establecido por el equipo.

Una gran ventaja de este sistema es que permite al equipo de trabajo ver dónde se encuentran los cuellos de botella de las tareas ya que se van acumulando en esa columna. Al visualizar esto, se pueden tomar medidas rápidas para evitar así una pérdida de eficiencia.

Por otra parte, otra gran ventaja de esta metodología es su facilidad de uso ya que el flujo de acciones a realizar para la mantenibilidad del tablero es escaso.

1. Se abre una nueva incidencia o desarrollo en el sprint, por lo que se coloca la tarea en la columna por hacer.
2. Cuando el responsable de realizar esta tarea comienza a desarrollar la solución al problema propuesto, la tarea se pasa a la columna En Progreso.
3. Una vez el desarrollador de la tarea termine, la pasa a la columna completado, indicando así al equipo que no es necesario invertir más tiempo y recursos a ese elemento.

## Scrum y Kanban en Github

Para poder aplicar estas metodologías a este proyecto, se han utilizado diferentes herramientas gratuitas de Github, las cuales nos permiten simular los *sprints* y tener una batería de tareas que podemos gestionar en un tablero visual.

1. **Tareas y *sprints*:** Las tareas de este proyecto se han generado en el apartado de Issues de Github. Este apartado nos permite establecer un título y una descripción, así como realizar diferentes asignaciones tales como la persona encargada de resolverla, en que proyecto se encuentra, y en que *milestone* ha de agruparse.

Los *milestones* mencionadas en la figura 4.1 son una herramienta que nos brinda Github y que nos permite simular los *Sprints*. Dentro de esta herramienta se establece un título y descripción de lo que se va a tratar, así como una fecha de fin en la cual se asocian las tareas a realizar. Es importante destacar que uno de los puntos donde se diferencia una *milestone* de un sprint es en la posibilidad de modificar la fecha de fin si es necesario, opción que en un sprint no se puede realizar.



Figura 4.1: *milestone* con Issues incluidas

2. **Projects:** Los proyectos de Github son una representación kanban de las tareas que se crean para poder llevar una visión rápida de las tareas. En este caso, estas dos herramientas son muy similares exceptuando pequeños matices, como el límite de trabajo en progreso, el cual en Github se puede gestionar de manera manual pero no es tan estricto como Kanban. En la figura 4.2 se puede observar el tablero de Github utilizado durante este proyecto.



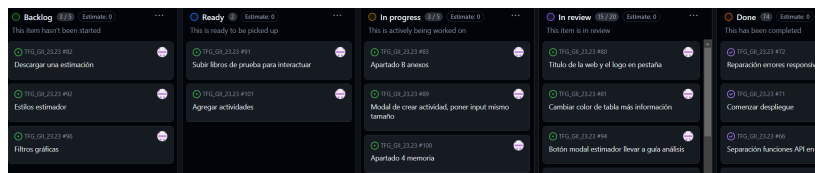


Figura 4.2: Tablero de Github

## 4.2. Frameworks utilizado para el desarrollo del proyecto

### Herramienta *Backend*: Flask

Flask [10] es un microframework para Python que ha sido diseñado para facilitar el desarrollo de aplicaciones web. A diferencia de otros frameworks más complejos y rígidos, Flask proporciona la flexibilidad necesaria para adaptar la estructura del proyecto a las necesidades específicas de este. Ofrece un conjunto de herramientas que simplifican procesos como el enrutamiento, la gestión de sesiones y la integración con bases de datos. Inicialmente en el desarrollo del proyecto se ha utilizado esta herramienta tanto para el *backend* como para el *frontend*, sin embargo, se presentó la oportunidad de cambiar el sistema cuando ya se encontraba muy avanzado y migrar el *frontend* a Angular y dejar este framework para la parte de *backend*. Esto ocurrió debido a la toma de contacto con esta nueva tecnología en mis prácticas curriculares y la posibilidad de desarrollar elementos más complejos gracias a este *framework*.

### Funcionalidades y Ventajas de Flask

La elección de Flask se debe a su capacidad para manejar tanto aspectos básicos como avanzados del desarrollo web:

- **Desarrollo Ágil:** Flask permite un rápido desarrollo y prototipado, lo que es ideal para los ciclos de iteración del TFG.
- **Simplicidad y Flexibilidad:** Su simplicidad facilita la comprensión del código, lo que es fundamental para un TFG, donde la claridad y la documentación son clave.
- **Ecosistema Completo:** Existe una amplia variedad de extensiones disponibles que permiten añadir funcionalidades adicionales según sea necesario, sin sobrecargar el núcleo de la aplicación.

- **Licencia:** Flask está disponible bajo la licencia BSD, una licencia de software libre que permite la reutilización y distribución del código con pocas restricciones.

### Bibliotecas utilizadas:

- Flask y sus derivados: Estas bibliotecas de Flask son las básicas para tener la importación completa del framework y poder utilizar todas sus funcionalidades.
- **beautifulSoup4**: Es una librería que se encarga de permitir obtener información de un archivo HTML, así como métodos para realizar búsquedas o modificaciones en ese archivo. En este proyecto se utiliza para obtener los datos de un libro durante la búsqueda automática de libros.
- **bcrypt**: Proporciona funciones para la gestión de contraseñas, como el cifrado y la verificación de una contraseña. Esto es realmente útil ya que no va a ser nunca necesario descifrar una contraseña para poder comparar si es la correcta o no.
- **unicorn**: Es un servidor WSGI que permite a Flask manejar llamadas de manera concurrente y así permitir la simultaneidad.
- **openpyxl**: Esta librería permite crear y leer archivos excel, lo cual es bastante útil para que los administradores puedan gestionar diferentes aspectos de la web.
- **requests**: Esta librería permite realizar llamadas de distintos tipos a una web remota. Esta librería se utiliza durante el web scraping
- **SQLAlchemy**: Permite tener una interfaz de alto nivel para poder tratar con los datos de una base de datos relacional, donde se encuentran recogidos todos los datos de la aplicación web.

### Herramienta *Frontend*: Angular

Angular [16] es un framework de desarrollo de aplicaciones web creado y mantenido por Google. Este framework está especialmente diseñado para poder crear webs robustas y muy rápidas y para este proyecto se ha usado en la parte *frontend* después de la decisión de refactorizar desde Flask.

## Características y ventajas de Angular

- Estilo de la aplicación y arquitectura:

Angular tiene un concepto de SPA (Single Page Application) y una arquitectura basada en componentes. Esto nos permite mantener una modularidad absoluta de los elementos de la web y poder reutilizarlos siempre que se quiera. Además, la ventaja principal de un SPA es que todos los elementos, si no se encuentra activo el Lazy Loading, se cargan en una misma página, pero se ocultan y se muestran en base a la redirección necesaria, por lo que los tiempos son mucho menores entre el cambio de páginas.

- Lenguaje utilizado en este framework:

Angular utiliza TypeScript para su programación lógica y para la parte visible al usuario se utiliza HTML y CSS. Typescript es un superconjunto de JavaScript que permite tener más características y funcionalidades que si se utilizase únicamente JavaScript

- Inyección de dependencias: Una gran ventaja de Angular es la facilidad para realizar inyecciones de dependencias en componentes y servicios para poder interconectarse de una manera rápida, evitar posible código duplicado y poder mantener una organización clara y simple que puede fomentar la eficiencia en el desarrollo

## Bibliotecas utilizadas:

- PrimeNG [3]/PrimeIcons [2]: Es una librería de componentes de interfaz de usuario que proporciona un amplio catálogo de componentes personalizables, como tablas, formularios, menús y gráficos mientras que PrimeIcons es un conjunto de iconos utilizados con PrimeNG para mejorar la apariencia de la aplicación.
- SweetAlerts2 [4]: Es una librería para crear alertas personalizables en aplicaciones web. Permite mostrar alertas con diferentes estilos, iconos y botones, mejorando la experiencia de usuario y la sencillez de la mantenibilidad.

## 4.3. Desarrollo de prototipo Web

### Justinmind

Justinmind [19] es una herramienta de prototipado interactiva utilizada en este proyecto para diseñar la interfaz de la web de libros de la prehistoria. Fue elegida por su facilidad de uso, y amplia biblioteca de widgets.

#### Ventajas de Justinmind

- Interactividad: Simulación realista de la interfaz final de usuario
- Versatilidad: Adecuada para prototipos de baja y alta fidelidad.

Justinmind ha sido fundamental para definir y validar la experiencia del usuario de manera inicial, realizar pruebas de usabilidad y facilitar la comunicación del diseño entre los desarrolladores y el cliente.

## 4.4. Hosting de los archivos

### GitHub (Repositorio)

GitHub [14] es una plataforma que utiliza el sistema de control de versiones Git para almacenar y permitir gestionar los proyectos creados de software de manera colaborativa. Además contiene una serie de herramientas muy variadas para comprobar el estado actual de las tareas relacionadas con el proyecto. Estas herramientas son las mencionadas anteriormente en el apartado de Kanban y Scrum.

#### Ventajas de la utilización de Github

- **Sistema de versionado:** Esta herramienta permite mantener un sistema de versiones y poder consultar que cambios se han realizado y el momento en el que se han realizado. Esto es de especial relevancia en caso de que una nueva versión genere errores o incompatibilidades, ya que podemos revertir los cambios y restaurar a una versión anterior.
- **Existencia de Issues:** Tal como se ha expuesto antes, esta herramienta permite la gestión de tareas a través de las Issues, lo cual nos permite indicar y ver de manera rápida los desarrollos o incidencias existentes sin necesidad de utilizar ninguna herramienta de terceros.

- **Sistema colaborativo:** Al permitir un sistema colaborativo a tiempo real, Github da la oportunidad a que varios integrantes del equipo puedan realizar modificaciones y poder ser más eficientes en términos temporales. En el caso del desarrollo de este proyecto, esta opción de colaboración se ha utilizado para la creación de incidencias durante la fase de despliegue.

## Render (*Backend*)

Render [26] es una plataforma de servicios que se ha utilizado para desplegar la parte de la aplicación escrita en Python, consistente en la API que contiene toda la lógica de la aplicación web. Esta decisión se ha tomado principalmente por el bajo costo del despliegue en las condiciones necesarias, la sencillez y la compatibilidad con el lenguaje generado.

Además de eso, la balanza se ha decantado por esta herramienta por los siguientes motivos:

- **Despliegue automático:** Esta herramienta permite tener una conexión activa con el repositorio de Github y al detectar un nuevo cambio en el repositorio, automáticamente lanza un nuevo despliegue para que el servidor siempre se encuentre en la última versión disponible.
- **Configuración del despliegue simple:** Al utilizar la herramienta incorporada de crear la API es necesario rellenar campos de configuración para establecer el lugar de los archivos y como se lanza cada uno de los necesarios. A diferencia de otros servicios, Render facilita todo este trabajo con formularios muy simples incluyendo un ejemplo e información muy completa relativo a cada campo. Además de ese formulario, de una manera rápida y sencilla se pueden modificar las variables de entorno necesarias para el correcto funcionamiento de la aplicación.
- **Existencia de discos de guardado de datos:** Tal como se menciona en el título del apartado, Render permite tener a disposición del proyecto un disco para poder guardar los datos deseados. En este proyecto se ha utilizado para poder gestionar los datos recogidos en la base de datos que usa el API. La gran ventaja de esto es que cada 24 horas se realiza una copia de seguridad que se puede restablecer en caso de necesidad.

## Netlify (*Frontend*)

Netlify [22], al igual que Render, es una plataforma de alojamiento para proyectos software, pero está especialmente orientada hacia el *Frontend* y Angular, por lo que, al estar optimizada para este framework, en el caso de este proyecto se convierte en una opción muy interesante teniendo en cuenta que contiene una opción de despliegue gratuito. Para consolidar esta herramienta como la elegida para poder realizar este despliegue, se tuvieron en cuenta las siguientes ventajas:

- **Despliegue continuo:** Al igual que Render para el *backend*, Netlify ofrece la posibilidad de tras realizar una conexión exitosa con la cuenta de Github donde se encuentra el repositorio, cargar y realizar las actualizaciones necesarias en el sistema de despliegue después de que se haya detectado un cambio en el repositorio.
- **Configuración simple:** Esta característica es importante ya que, aunque para Angular se necesiten hacer pasos adicionales ya que es un SPA y puedan aumentar la complejidad del despliegue, con unos pocos pasos se puede realizar la conexión inicial sin ningún problema. En el caso de que fallase algo, Netlify contiene una zona de Logs donde se puede observar todos los pasos que realiza durante el despliegue y nos puede dar indicaciones del problema que ha surgido.
- **Gestión de versiones:** Existe un apartado dentro de Netlify que permite volver a un despliegue anterior en caso de que se produzcan errores o incompatibilidades con la versión más reciente. Como gran punto a favor, es importante mencionar que, a diferencia de Render, mientras se realiza un nuevo despliegue, este se realiza en segundo plano, manteniendo así la disponibilidad de la web el máximo tiempo posible.

## 4.5. Herramientas para la escritura de la memoria

### Overleaf

Overleaf [23] es una plataforma diseñada para redactar documentos en L<sup>A</sup>T<sub>E</sub>X. Overleaf es una herramienta pensada principalmente para investigadores ya que permite colaboración en tiempo real y una organización de páginas mucho más limpia que otras herramientas.

A continuación se muestran algunas ventajas por las que se usa este editor de texto.

- **Compilación automática:** A diferencia de otras herramientas que trabajan con  $\text{\LaTeX}$ , Overleaf permite compilar en tiempo real el documento para ir llevando una depuración de fallos mientras se está redactando.
- **Historial de versiones:** Esta herramienta permite la posibilidad de poder revisar los cambios que se han ido realizando con el tiempo, al igual que ofrece Github para sus archivos. Esto es muy beneficioso ya que de esta manera podemos recuperar datos eliminados por error.
- **Acceso Online:** Overleaf al ser una plataforma web, permite acceder desde cualquier parte a los documentos guardados y así no tener que realizar transferencias de archivos a otros dispositivos, lo cual puede llegar a ser tedioso en el caso de que sea necesario en repetidas ocasiones.
- **Adaptación al usuario:** Cuenta con dos tipos de vista para poder redactar el documento, una vista de código, donde se puede aplicar código de programación  $\text{\LaTeX}$  y comandos directamente por el usuario, o la vista de editor, una vista mucho más simple para usuarios que no están familiarizados con el lenguaje.

## Draw.io

Draw.io [12] es una herramienta en línea con opción a descarga que permite realizar diagramas de diferentes tipos para mostrar de una manera gráfica procesos o un esquema un diagrama de flujo. Dentro de las opciones disponibles en el mercado, esta herramienta tiene varias ventajas:

- **Personalización:** Esta herramienta permite una gran personalización de todos los elementos que la componen, permitiendo así realizar los diagramas de la manera óptima y visual para el usuario final.
- **Exportación:** Para poder tener más libertad durante la manipulación de los diagramas, Drawio permite exportar en diferentes extensiones para poder generar el archivo más adecuado para el posterior uso de ese diagrama. En el caso de este proyecto se ha utilizado la exportación como PNG para poder incluir el archivo dentro de los Anexos.

- **Precio:** A diferencia de muchas herramientas, todos estos beneficios se pueden obtener de forma totalmente gratuita y sin necesidad de registros, por lo que es una buena opción para reducir gastos de desarrollo del proyecto.

## 4.6. Otras herramientas usadas

### Postman

Postman [25] es una herramienta que permite realizar llamadas a APIs durante las fases de desarrollo y testing. A través de Postman se pueden realizar peticiones HTTP de diferentes tipos y mostrar los datos enviados por la API sin tener limitaciones de formato. A continuación se muestran algunas de las ventajas clave de esta herramienta:

- **Organización de carpetas:** Postman permite tener una organización de carpetas fijas con llamadas preestablecidas para que no sea necesario escribir de nuevo la petición entera, lo que es muy eficiente en términos temporales.
- **Múltiples tipos de solicitudes HTTP:** Este servicio permite no solo realizar llamadas de obtención o envío de datos si no que permite realizar además llamadas de modificación de datos (PUT) y de borrado (DELETE). Esto da la oportunidad de usando un único programa poder comprobar si los desarrollos que se van realizando se completan satisfactoriamente, tanto en tiempo como el contenido resultante.
- **Interfaz y precio:** La interfaz de esta herramienta es muy sencilla y gráfica por lo que sin necesidad de una extensa documentación se puede comenzar a utilizar el programa. Además, el software es gratuito y no es necesario iniciar sesión, por lo que el proceso de instalación y puesta en marcha es muy sencillo.

## 4.7. Análisis Comparativo entre Google Books API y Amazon Books API

Debido a la necesidad de integrar una API de libros para poder realizar llamadas, se realiza una comparativa inicial entre Google Books API [15] y Amazon Books API [7] para obtener el mejor candidato para implantar en el trabajo.



## Accesibilidad y Documentación

- **Google Books API:** Ofrece accesibilidad superior y documentación detallada. Proporciona una clave de API gratuita con un límite de 1,000 solicitudes diarias.
- **Amazon Books API:** Requiere afiliación a Amazon Advertising API y está orientada hacia usuarios con propósitos comerciales. La documentación es robusta pero más compleja.

## Amplitud de Datos Disponibles

- **Google Books API:** Acceso a más de 25 millones de libros con información extensa, ideal para proyectos educativos o bibliotecarios.
- **Amazon Books API:** Proporciona datos orientados a ventas y reseñas, incluyendo rankings y precios, útil para análisis de mercado, pero no tanto para nuestra intencionalidad.

## Facilidad de Integración y Uso

- **Google Books API:** Fácil integración gracias a su estructura basada en REST y compatibilidad con múltiples lenguajes de programación.
- **Amazon Books API:** Requiere comprensión avanzada de las API de Amazon y sus requisitos de autenticación, generando una curva de aprendizaje más grande.

## Restricciones de Uso y Limitaciones

- **Google Books API:** Tiene limitaciones en el número de solicitudes diarias, pero de manera general es aceptable para la escala de este proyecto.
- **Amazon Books API:** Limitaciones más estrictas en cuanto a la frecuencia de las solicitudes y acceso a algunos datos.

La elección de la API de Google Books se justifica por su accesibilidad, amplia gama de datos bibliográficos, facilidad de integración, y una cuota de solicitudes gratuitas mayor. Esto la hace ideal para este proyecto ya que nos brinda más facilidades y más datos que nos resulten relevantes.

Herramientas	App	Angular	API REST	BD	Memoria
HTML5		X			
CSS3		X			
PrimeNG		X			
TypeScript		X			
AngularJS		X			
SweetAlert2		X			
Netlify		X			
Python			X		
Flask framework			X		
BeautifulSoup4			X		
Bcrypt			X		
Google Books API			X		
JSON		X	X		
Gunicorn			X		
Openpyxl			X		
Requests			X		
Postman			X		
Render			X	X	
SQLite				X	
SQLAlchemy			X	X	
Github		X	X	X	X
Overleaf					X
Drawio					X
Justinmind					X

Tabla 4.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

---

## 5. Aspectos relevantes del desarrollo del proyecto

---

### 5.1. Metodologías Aplicadas

Desde el inicio del proyecto se tenía muy claro que se iba a proceder de la manera más ordenada y profesional posible. Para poder realizar exitosamente este proceso recurrimos a diferentes herramientas que nos permitiesen realizar un correcto desarrollo de este proyecto.

A continuación se comentan las herramientas utilizadas:

- Kanban:

La metodología Kanban consiste en poder informarte del estado de las tareas del proyecto de una forma visual y rápida, por lo que, con una rápida visualización, podemos saber el estado de cada una de las tareas existentes en el tablero.

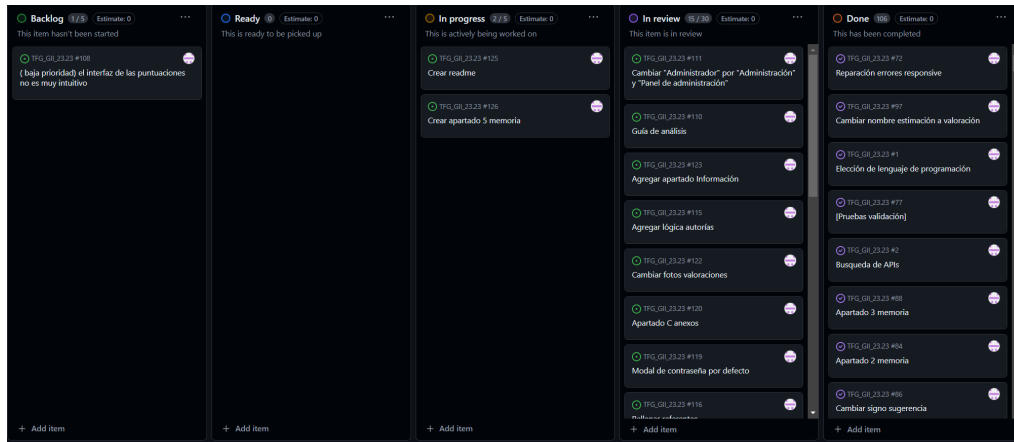


Figura 5.1: Tablero Kanban

## ■ Scrum

Esta metodología principalmente se posiciona en realizar ciclos de trabajos (Sprints) con una duración fija, en la que al finalizar se realiza una entrega del proyecto. Esto permite tener una mayor flexibilidad durante el proyecto y una correcta entrega continua que nos permite la máxima colaboración con el cliente, en este caso el profesor de la UBU Jesús Alberto San Martín Zapatero [18]. En este caso se han realizado Sprints de 2 semanas con una reunión al finalizar el Sprint para poder debatir acerca de la entrega realizada y realizar las propuestas de trabajo para el siguiente Sprint. Una vez generadas esas propuestas se transformaban en tareas que se incluían en el tablero Kanban para realizar un seguimiento a tiempo real del progreso del Sprint.

## 5.2. Formación

Para la realización del proyecto han sido necesarios diferentes conocimientos técnicos para poder plasmar las ideas y los prototipos en una aplicación web real y completa. Dentro de todos estos conocimientos requeridos, existían algunos de ellos que no se disponían al iniciar el proyecto, por ese motivo, se dedicó tiempo a realizar formaciones sobre los siguientes elementos:

- Flask: Aunque Flask sea un Framework de Python, contiene elementos y bibliotecas compatibles que agregan funcionalidades muy distintas a lo existente en una versión nativa de Python. Para poder hacer

frente a la necesidad de estos nuevos conocimientos, se ha utilizado la documentación de Flask [10] de manera principal para poder tener un punto de partida sólido y unas directrices básicas. Tras tener los puntos esenciales, a través de diferentes videos y pequeños cursos se realizaron pequeños proyectos de prueba para ir desarrollando las habilidades necesarias.

- Angular: Angular es una herramienta que de manera inicial no se encontraba contemplada en este proyecto, sin embargo, tras mi experiencia en las prácticas curriculares y la posibilidad de realizar un curso profesional de Angular [17], se presentó como una opción muy interesante para aumentar la complejidad de la aplicación web y disponer de más herramientas para el desarrollo del *Frontend*, así como numerosas ventajas para la parte de diseño del proyecto, permitiendo crear una web más rápida y fácilmente reutilizable y escalable. Como Angular es un Framework, este curso aportó conocimientos de TypeScript, HTML y CSS.

### 5.3. Inicio del proyecto

Esta idea comenzó por parte del profesor de la UBU Jesús Alberto San Martín Zapatero con el afán de aportar una nueva herramienta que permitiese a docentes y familias realizar consultas acerca de la rigurosidad científica existente en términos de perspectiva de género de los libros de la prehistoria.

Este proyecto tiene como nombre PREHISTORIA EN IGUALDAD.



Figura 5.2: Logo de la web

Para poder plasmar esa idea en un proyecto real, se puso en contacto con la facultad de Informática y comenzamos con el desarrollo de esa aplicación web.

Una de las partes positivas de este proyecto ha sido la posibilidad de realizarlo tratando con el docente mencionado como si fuera un cliente al que entregar un producto, lo cuál también ha sido un reto añadido en este proyecto. Esto ha aportado valor al desarrollo ya que a lo largo de cada pequeña entrega se han tenido que realizar demostraciones del programa y trasladar los requisitos necesarios propuestos por él a un producto viable.

Una vez decidido el comienzo del proyecto, se llegó al acuerdo de utilizar el Framework de python Flask como *frontend* y *backend* de la aplicación, y se desarrollaron prototipos básicos como la imagen de debajo, permitiendo dar una idea de cómo podría quedar la aplicación una vez desarrollada.

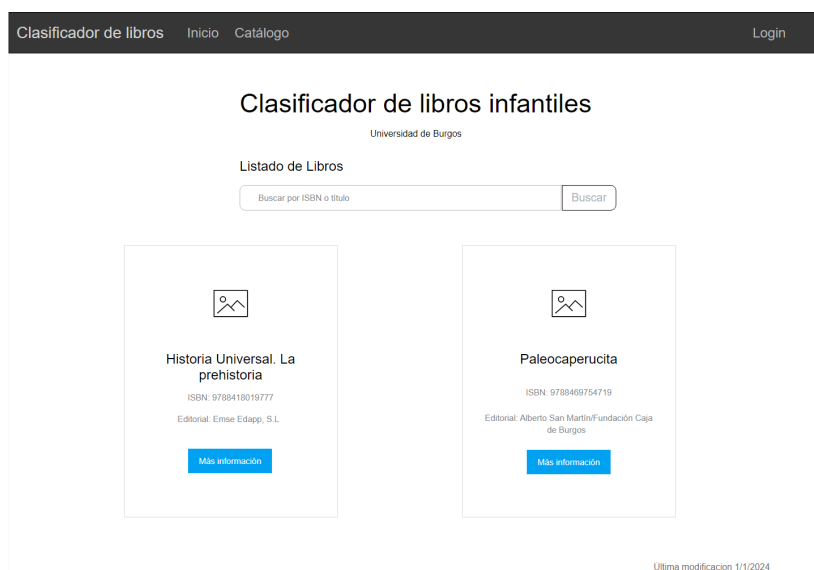


Figura 5.3: Prototipo del catálogo

## 5.4. Desarrollo del *frontend*

Al comienzo del desarrollo, como ya se ha mencionado, se ha utilizado el *framework* Flask, el cuál permite realizar conexiones directas entre el *frontend* y el *backend* mediante funciones muy simples. Además de este *framework*, se utilizó bootstrap [1], el cuál es un *toolkit* disponible para numerosos lenguajes que permite obtener de una manera rápida componentes prediseñados que facilitan el desarrollo en HTML y CSS. Tras realizar diversos desarrollos entre los que se encontraba el catálogo con sus opciones de edición protegido por un inicio de sesión, se fueron incluyendo más ideas de desarrollo para poder ampliar la aplicación y permitir a los usuarios tener su propia estimación de un libro.

Uno de los mayores problemas de este desarrollo consistió en la dependencia que existía entre *front* y *back*. Esto podía generar un problema en el caso de que una de las dos partes fallase, impidiendo al usuario utilizar la funcionalidad que funcionase correctamente. La solución adoptada consistió en separar ambas partes para brindar total independencia. Esto se traduce en que por ejemplo, si se diese el caso de que el *backend* fallase, la interfaz del usuario podría seguir mostrando al usuario la información que desease pero de forma limitada, reduciendo así las posibilidades de que todo el conjunto falle y de ninguna forma un usuario pueda interactuar con la aplicación.

Para aplicar la solución encontrada, se decidió refactorizar el proyecto entero a un sistema cliente-servidor utilizando los *frameworks* Angular y Flask. Esta decisión se consolidó debido a la posibilidad de realizar un curso de pago y el conocimiento adquirido durante las prácticas curriculares.

Esta refactorización supuso un antes y un después en el diseño del proyecto y su rendimiento, ya que Angular, al ser un SPA, no tiene que cargar página a página, sino que tiene todos sus componentes cargados y los va mostrando u ocultando en función de la interacción del usuario. Por lo tanto, mereció la pena retrasar un poco la planificación inicial dados los resultados obtenidos.

Además, se dejó de aplicar Bootstrap 5 [1] y se comenzó a utilizar PrimeNG [3] y PrimeIcons [2] para la generación de componentes con responsive incluido y sweetAlert2 para las ventanas modales de confirmación u avisos.

Estas nuevas herramientas han permitido desarrollar una aplicación web mucho mas compleja y dinámica para que tanto usuarios como el personal de administración puedan interactuar con una gran variedad de opciones a su disposición.

La tarea para realizar esto fue de gran complejidad ya que se rediseñó toda la estructura de los ficheros y la gran mayoría del HTML no se podía reutilizar por el uso de Bootstrap, por lo que se reprogramó todo lo hecho hasta ese desarrollo mientras, paralelamente, se continuaba en Flask por si el resultado de la refactorización no era satisfactorio.

Una vez realizada completamente la refactorización, se rediseñó la interfaz del personal de administración, el cual en la anterior versión únicamente permitía hacer modificaciones del catálogo y realizar importaciones o exportaciones desde el propio catálogo. El sistema de gestión de usuarios se realizaba a nivel de código, por lo que no era útil para la administración.



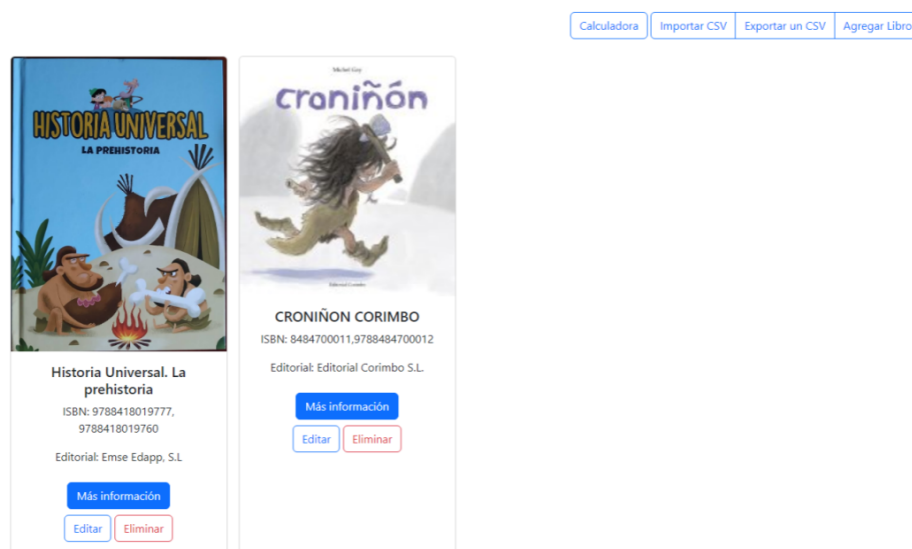


Figura 5.4: Administración antigua

Tras los desarrollos pedidos por el cliente, los cuales se centraban en mostrar información al usuario sobre los libros estimados por el personal de administración, las estimaciones de los propios usuarios y recomendaciones de recursos para facilitar el aprendizaje, se iban desarrollando paralelamente elementos los cuales permitían automatizar y personalizar la aplicación web de manera gráfica y sencilla desde una panel de administración completo.

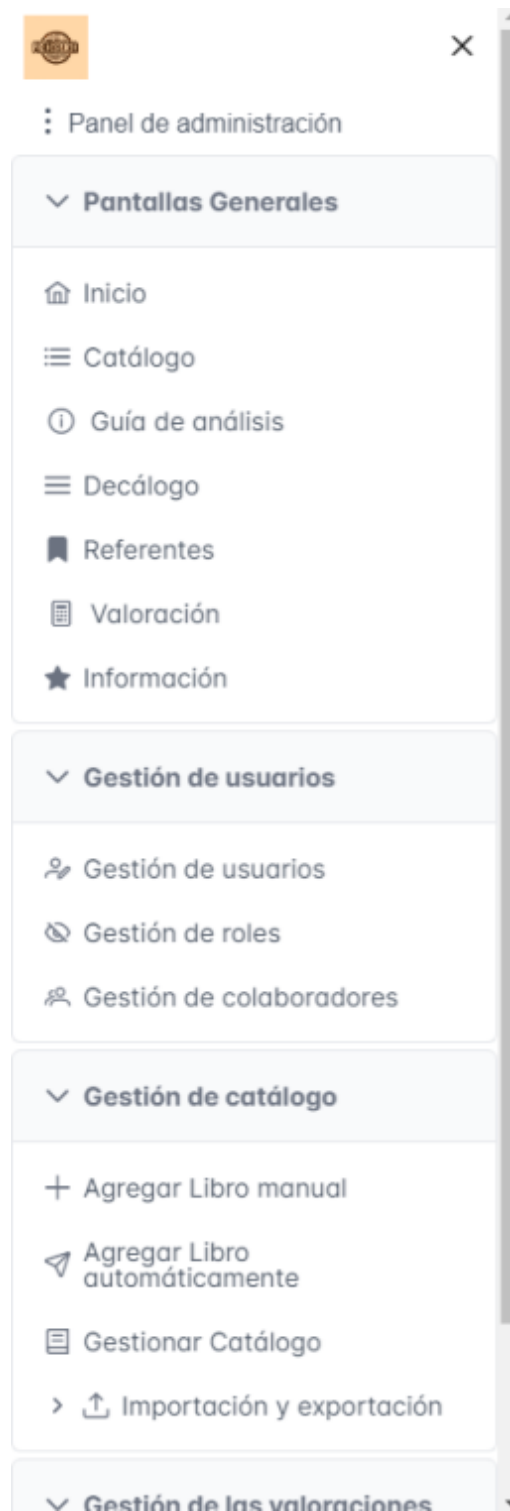


Figura 5.5: Panel de Administración

## 5.5. Desarrollo del *backend*

Esta parte de la aplicación, a diferencia del *frontend*, no ha sufrido un cambio de tecnología, simplemente se ha cambiado el esquema de los archivos y se han modificado las funciones para poder funcionar como una API y poder conectarse satisfactoriamente con el Angular. De manera adicional, al generar esta separación, se pueden realizar llamadas a este servicio sin la necesidad de Angular, a través de aplicaciones como Postman [25]. Esto deja una puerta abierta a la posibilidad de distribución de la API para que puedan trabajar con ellos terceros e implementarlo en sus sistemas.

Durante el diseño de esta API se decidió separar por grupos funcionales todas las funciones de la aplicación, pudiendo así incorporar nuevas utilidades de forma sencilla sin tener que modificar otras funciones.

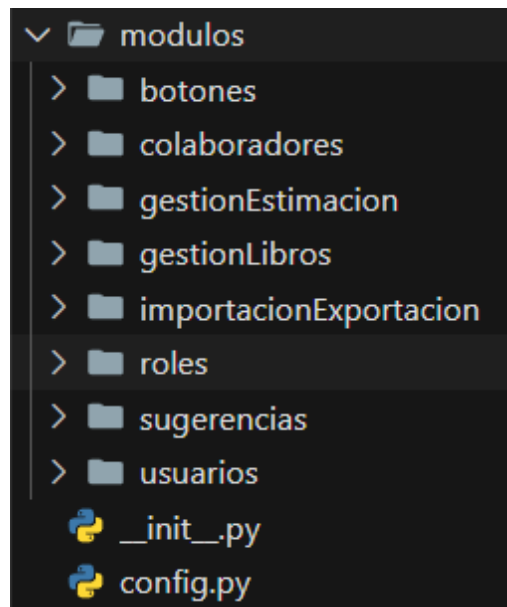


Figura 5.6: Módulos del proyecto

Dentro de esta API se encuentran todas las funcionalidades existentes para un correcto funcionamiento de todos los requisitos funcionales mencionados en los anexos. Estas funciones dependen de una base de datos donde se guarda toda la información utilizada por la aplicación web.

## Web scraping

El web sraping de esta aplicación tiene como origen la propuesta de facilitar la búsqueda de libros utilizando diferentes fuentes bibliográficas, utilizando la API de Google Books y las otras dos son páginas web:

- Amazon
- Agapea

El inicio de este requisito comienza al recibir en el backend una petición adjuntando un identificador del libro que que desea (Título, ISBN10 o ISBN13).

Una vez estos datos llegan al *backend*, se llama a una función auxiliar encargada de realizar las llamadas a las fuentes. Como dependiendo del parámetro pasado inicialmente por el equipo de administración puede influir en los pasos a seguir en la fuente de Agapea y en la fuente de Google Books, como primera tarea a realizar es identificar mediante expresiones regulares si se trata de un ISBN o de un título. Para esto se utiliza la función `buscar_libro`.

```
1  import requests
2  from bs4 import BeautifulSoup
3  import re
4
5  parser = "html.parser"
6  descripcion = "No se ha encontrado descripción"
7  userAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36"
8  lenguaje = "es; q=0.5"
9  sintitulo = "No se ha encontrado título"
10 sineditorial = "No se ha encontrado editorial"
11
12
13 > def obtener_resumen_extendido(isbn): ...
41
42
43 > def buscar_libro_amazon(elemento): ...
197
198
199 > def buscar_libro_agapea_libro(info): ...
230
231
232 > def buscar_libro_agapea_isbn(info): ...
252
253
254 > def buscar_libro_agapea_generico(instancia_libro): ...
329
330
331 > def obtener_info_libro_google(isbn_o_titulo, filtro): ...
360
361 ✨
362 > def buscar_libro(isbn_o_titulo): ...
372
```

Figura 5.7: Funciones web scraping

## Web scraping Agapea

Para esta primera fuente, la manera de buscar los libros varía significativamente, ya que si el usuario introduce el ISBN, si se incluye el ISBN en la URL de la web nos va a llevar directamente a la ficha del libro, mientras que si se busca por título, primero es necesario acceder a un listado de libros con resultados que podrían equivaler al texto introducido y después entrar en la ficha del libro.

Para realizar esos pasos diferentes se crearon dos funciones las cuales cada una se encargaba de un tipo de búsqueda. Al finalizar la ejecución, ambas funciones llaman a una función común que se encarga de recopilar la información disponible en la ficha del libro.

Uno de los mayores problemas que se generaron al trabajar con esta fuente es que el contenido del Resumen en la ficha del libro está limitado por un Leer Más tal como aparece en la figura 5.8.

### Resumen

Biblioteca para mentes curiosas presenta tres libros de conocimiento que tratan de acercar diversos saberes (históricos, científicos, deportivos...), apoyándose en unas cuidadas...

[Leer más](#)

Figura 5.8: Leer Más Agapea

Para poder solucionar este limitante, se estudió la arquitectura de la página y se descubrió que realizando una solicitud Ajax [6] al servidor de Agapea se podía indicar a la web que se deseaba leer toda la información y posteriormente recoger los datos. La llamada se puede detectar en la web al inspeccionarla en el apartado network.

Una solicitud Ajax es una técnica que se utiliza para enviar y recibir datos de un servidor sin necesidad de recargar la web utilizando mayoritariamente Javascript y XML de manera asíncrona.

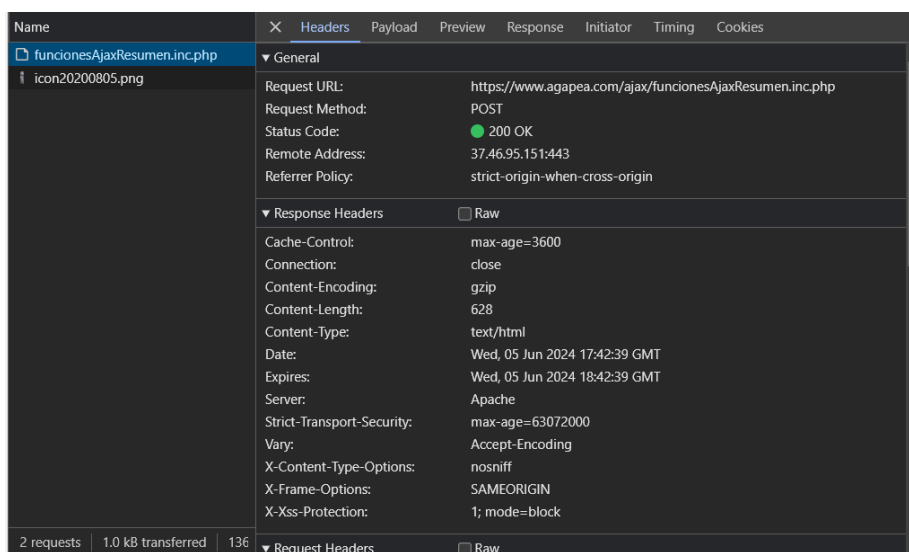


Figura 5.9: Función Ajax Agapea

## Web scraping Amazon

Para esta segunda fuente no es necesaria esa distinción entre ISBN y título, ya que ambos elementos redirigen a un menú donde se puede seleccionar el producto deseado.

Una vez obtenido la ficha del libro, se van haciendo comprobaciones para consolidar la existencia de los datos y poder almacenarlos. En el caso de que no se encuentre alguno de los datos, se incluye en el resultado como "No se ha encontrado el elemento X".

## API Google Books

Esta tercera y última fuente se obtiene directamente realizando llamadas a una API propiedad de Google. Esta API nos pide una distinción entre ISBN y título, por lo que es necesario pasárselo por parámetro para realizar la llamada. Debido al tipo de llamada que vamos a realizar, el cuál apunta al *endpoint* correspondiente al apartado volúmenes, no es necesario registrarse ni tener un token de autenticación, esto nos permite realizar llamadas sin un limite establecido a cualquier tipo de libro. Tras realizar la llamada al *endpoint* indicándole el título o el ISBN, la API devuelve la información que contiene en un JSON, por lo que se traduce y se comprueba la existencia de datos válidos en esa respuesta.

Como esta es la última función en ejecutarse, al finalizar devuelve el control a la función principal que recoge el resultado de todas las fuentes y las guarda en una base de datos temporal para que el equipo de administración decida si utiliza esos datos obtenidos y de que manera (Escogiendo una única fuente o combinando varias de ellas).

## Gestión de permisos

Tras ir haciendo desarrollos se presentó la idea de no permitir a todos los miembros del equipo de administración realizar todas las gestiones disponibles en el *frontend*, sino que dependiendo del usuario se le bloquee o desbloquee las funcionalidades dinámicamente. Para la resolución de ese problema, además de existir una cuenta por usuario, se creó un sistema de roles dinámicos con permisos. Es decir, los miembros del equipo de administración que tengan los permisos necesarios, pueden crear, editar y eliminar todos los roles que necesiten tener en la aplicación, y establecer a cada uno de esos roles una serie de permisos activados y desactivados.

Esto significa que cuando el *frontend* quiere cambiar de pestaña, manda una solicitud de permisos al *backend* preguntando si ese usuario tiene los permisos necesarios para poder acceder a ese recurso, la función responsable de esa parte revisa sus credenciales y devuelve una respuesta afirmativa o negativa para que le muestre la pestaña si es admitido, o una modal indicándole que no tiene permisos para realizar esa acción.



### Acción denegada

No tiene permisos para acceder a esta página.  
Contacte con el administrador.



Figura 5.10: Modal Usuario sin Permisos

## 5.6. Despliegue de la aplicación

Para desplegar esta aplicación, se utilizaron dos plataformas diferentes:

- Netlify para el *frontend*.
- Render para el *backend*.

Netlify [22] es una plataforma de despliegue para sitios web compatible con Angular. Esta herramienta cuenta con un plan gratuito con el que se puede subir el proyecto desde Github. Aunque esta herramienta esta pensada para esta tecnología, durante el proceso de despliegue aparecieron problemas relativos a las redirecciones de la aplicación. Esto se debe a que Netlify intenta cargar esa URL pero no encuentra nada. Esto se debe a que Angular es un SPA, por lo que no tienen que existir redirecciones con la carga de nuevo de la página. Para poder solucionar este problema correctamente, al crear los ficheros de producción del *frontend*, se necesitó añadir un archivo que indica a Netlify que las redirecciones siempre lleven a la pantalla principal aunque cambie de URL.

Una gran ventaja que aporta esta herramienta es la realización de actualizaciones automáticas de la web. Es decir, cada vez que detecta un cambio en el repositorio de Github automáticamente comienza a construir la nueva versión mientras mantiene activa la antigua, una vez la más actualizada está preparada, sustituye una por otra, permitiendo así el máximo tiempo de disponibilidad posible en la aplicación web. En el caso de que detecte un cambio pero no pertenezca al *frontend*, no realiza esta acción.

La interfaz gráfica de Netlify es muy visual y sencilla, teniendo un apartado de *deploys* donde marca el estado de construcción del sitio al incorporar nuevos cambios.

En la figura 5.11 se puede observar los últimos despliegues realizados en este servicio, en los cuales se puede entrar para ver la información relativa a ese despliegue y la posibilidad de forzar esa versión como la más actualizada.



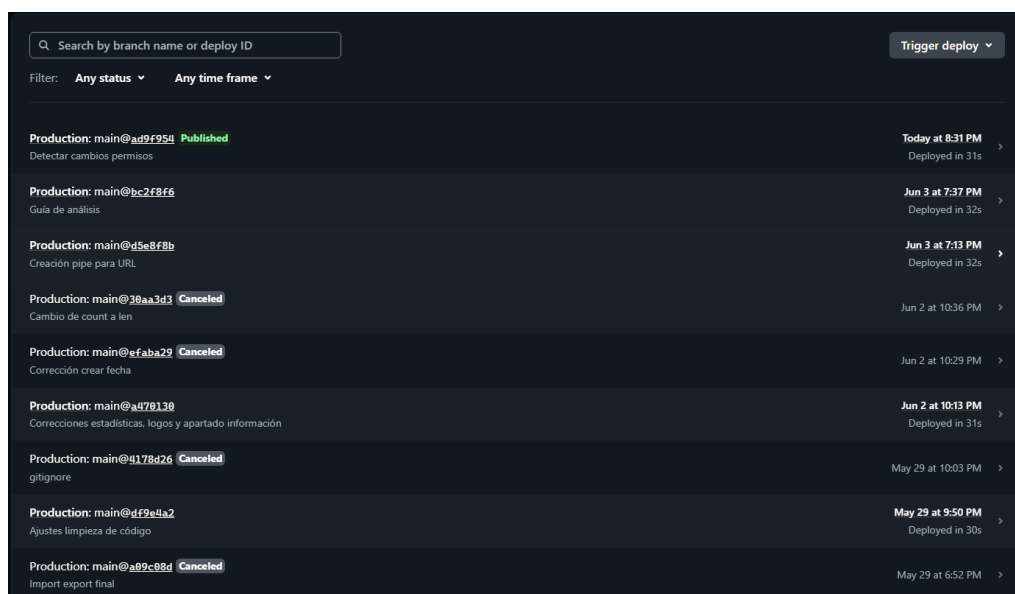


Figura 5.11: Interfaz Deploys Netlify

Por otra parte, Render ha sido la herramienta elegida para realizar el despliegue del *backend* Flask. Render, a diferencia de Netlify es más global y acepta diferentes tipos de despliegues. Esta herramienta se ha usado para el despliegue utilizando la opción de web service inicialmente con una cuenta gratuita. Una vez conectado con el repositorio de Github y establecidas las carpetas y comandos de arranque se desplegó la aplicación con las mismas características de autodespliegue que Netlify.

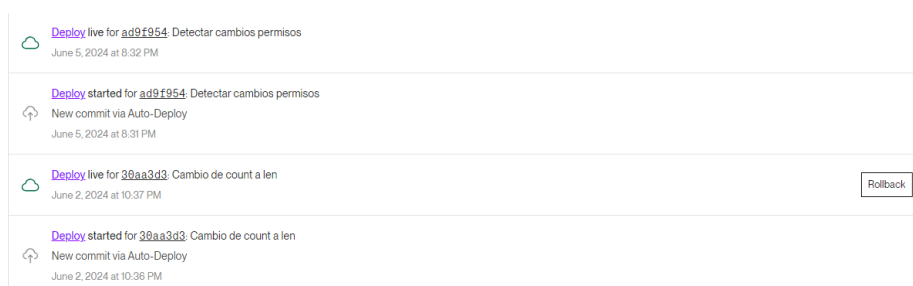
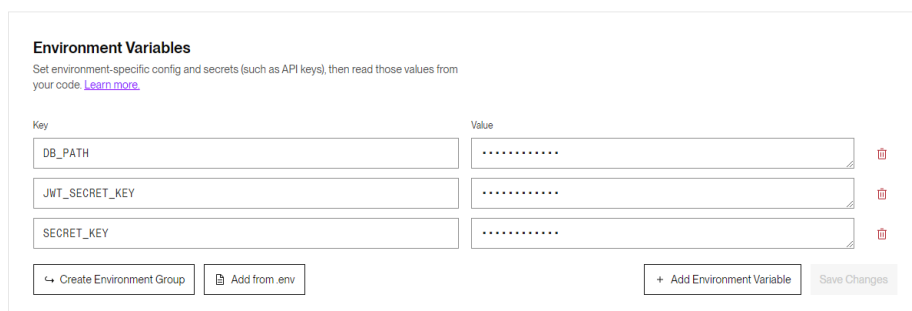


Figura 5.12: Pantalla Deploys Render

Para los ficheros de configuración que pueden contener contenido sensible, se migraron esos datos al apartado de variables de entorno para aumentar la seguridad cifrándolas y guardándolas de forma aislada.



The screenshot shows the 'Environment Variables' configuration page in the Render dashboard. At the top, there is a title 'Environment Variables' and a brief instruction: 'Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more.](#)'. Below this, there is a table with two columns: 'Key' and 'Value'. The table contains three rows of variables: 'DB\_PATH', 'JWT\_SECRET\_KEY', and 'SECRET\_KEY'. Each row has a text input field for the key and a masked text input field for the value, represented by dots. To the right of each value field is a red trash icon for deletion. At the bottom of the table, there are three buttons: '← Create Environment Group', 'Add from env', and '+ Add Environment Variable'. A 'Save Changes' button is located at the bottom right of the form.

Figura 5.13: Variables de entorno Render

Tras el despliegue y el inicio de realización de pruebas, se detectó que la base de datos al encontrarse integrada, al cerrarse el servidor se borraban todos los datos que no se encontrasen añadidos desde el *commit*. Para solucionar esto ha sido necesario pagar la versión más barata de este servicio para poder tener un disco dedicado a la base de datos y un servidor 24 horas encendido.

Para subir la base de datos al disco de Render, se necesitó realizar una conexión SSH utilizando unas claves dadas por Render para poder acceder remotamente al sistema de ficheros y poder enviar a esa dirección la base de datos que contenía los datos mínimos para funcionar correctamente.

## 5.7. Testing

El proceso de testing es una parte fundamental del desarrollo de software que asegura la calidad y fiabilidad del producto final. En este proyecto, se han realizado pruebas unitarias para verificar el correcto funcionamiento de los diferentes componentes de la aplicación web.

### Pruebas Unitarias

Las pruebas unitarias se implementaron utilizando el framework `unittest` [5] de Python. Este framework permite definir y ejecutar pruebas para asegurar que las funciones y métodos individuales se comporten como se espera. Cada prueba verifica una pequeña unidad de funcionalidad de forma aislada, permitiendo identificar y corregir errores de manera temprana en el ciclo de desarrollo.

Las pruebas unitarias son cruciales por varias razones:

- **Aseguran la calidad del código:** Verifican que cada función y método produce los resultados esperados para los *inputs* establecidos. Esto ayuda a identificar errores en la lógica antes de que se integren en la versión de producción.
- **Facilitan el *Refactoring*:** Permiten realizar cambios en el código con la confianza de que no se introducirán errores. Si una prueba unitaria falla después de un cambio, el equipo de desarrollo sabe exactamente qué parte del código fue afectada.
- **Documentación del Código:** Las pruebas unitarias sirven como una forma de documentación que explica cómo se espera que el código se comporte. Esto es especialmente útil para nuevos el equipo de desarrollo que se integran al proyecto.
- **Detección Temprana de Errores:** Al identificar problemas en las primeras etapas del desarrollo, se reduce el coste y tiempo de corrección. Es más económico y rápido corregir errores encontrados durante el desarrollo que aquellos encontrados después del despliegue.

## Proceso de Testing

El proceso de testing en este proyecto sigue una metodología sistemática para asegurar que todas las funcionalidades sean probadas de manera exhaustiva. Los pasos incluyen:

- **Implementación de Pruebas:** Utilizando `unittest`, se implementan las pruebas correspondientes para cada caso de prueba que se considere crítico.
- **Ejecución de Pruebas:** Las pruebas se ejecutan regularmente durante el desarrollo para asegurar que cualquier cambio en el código no introduce nuevos errores. Esto se logra mediante la ejecución de todas las pruebas unitarias como parte del ciclo de desarrollo continuo.
- **Análisis de Resultados:** Los resultados de las pruebas son analizados para identificar y corregir errores. Cualquier falla en una prueba unitaria indica un problema que debe ser investigado y resuelto.

## Beneficios Generales del Testing

El testing en general ofrece varios beneficios clave en el desarrollo de software:

- **Mejora de la Calidad del Código:** A través de pruebas rigurosas, se garantiza que el código cumple con los estándares de calidad y se comporta como se espera.
- **Reducción de Costos:** Identificar y corregir errores durante el desarrollo es mucho menos costoso que hacerlo después de que el software ha sido desplegado.
- **Mantenimiento Sostenible:** El código bien probado es más fácil de mantener y actualizar, ya que las pruebas ayudan a identificar rápidamente los impactos de los cambios.

## 5.8. Uso de Postman

Postman es una herramienta para el desarrollo de APIs que facilita la creación, el envío y el análisis de solicitudes HTTP. En este proyecto, se utilizó Postman para realizar pruebas manuales de la API desarrollada.

### Pruebas Manuales

Las pruebas manuales con Postman implican el envío de solicitudes HTTP a los endpoints de la API y la verificación de las respuestas. Esto es útil para:

- **Verificación Rápida:** Permite al equipo de desarrollo verificar rápidamente la funcionalidad de los endpoints durante el desarrollo. Se pueden enviar solicitudes GET, POST, PUT, DELETE, entre otras, para comprobar las respuestas de la API.
- **Depuración:** Ayuda a identificar y resolver problemas en la API de manera interactiva. El equipo de desarrollo puede ajustar los parámetros de las solicitudes y observar cómo cambian las respuestas.
- **Exploración:** Facilita la exploración de las funcionalidades de la API y el ensayo de diferentes casos de uso. Esto es especialmente útil durante la fase de desarrollo para entender cómo interactúan diferentes partes de la API.

Para llevar a cabo las pruebas manuales con Postman, se siguen los siguientes pasos:

1. **Configuración de la Solicitud:** Se selecciona el tipo de solicitud HTTP (GET, POST, PUT, DELETE) y se especifica la URL del endpoint que se desea probar.
2. **Configuración de los Parámetros:** Si la solicitud requiere parámetros, estos se configuran en la sección correspondiente de Postman. Esto puede incluir parámetros de URL, headers, y cuerpo de la solicitud en formato JSON.
3. **Envío de la Solicitud:** Se envía la solicitud y se espera la respuesta del servidor. Postman muestra la respuesta en su interfaz, incluyendo el código de estado HTTP, los headers de la respuesta y el cuerpo de la respuesta.
4. **Verificación de la Respuesta:** Se verifica que la respuesta del servidor sea la esperada. Esto incluye comprobar que el código de estado HTTP es correcto y que el cuerpo de la respuesta contiene los datos esperados.

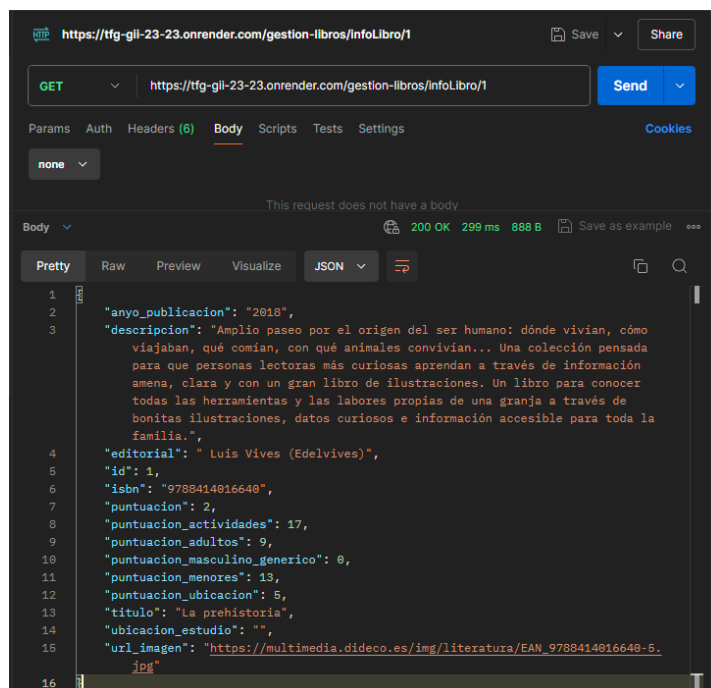


Figura 5.14: Ejemplo llamada con Postman

---

## 6. Trabajos relacionados

---

Durante la realización de este proyecto, se han tomado referencias de otros trabajos para poder tomar ideas y moldearlas de manera que se genere un producto completo y de calidad. A continuación, se detallan los trabajos y programas de los que se han tomado referencias o que se encuentran relacionados en este ámbito.

### 6.1. Atribuciones de género y construcción de identidades en la literatura infantil sobre prehistoria

El estudio de Alberto San Martín Zapatero y Delfín Ortega-Sánchez [28], titulado 'Atribuciones de género y construcción de identidades en la literatura infantil sobre prehistoria', examina cómo los libros infantiles dedicados a la prehistoria presentan estereotipos de género. Publicado en el *Bellaterra Journal of Teaching & Learning Language & Literature*, este trabajo utiliza técnicas de análisis de contenido para evaluar textos e ilustraciones, revelando que a menudo se distorsionan los roles de género y se alejan de los últimos avances científicos.

Este proyecto se inspira en este estudio para desarrollar su metodología de análisis. La herramienta del estimador del proyecto evalúa de manera objetiva la presencia de sesgos de género en los libros infantiles, adaptando y ampliando las categorías de análisis propuestas en el estudio de San Martín Zapatero y Ortega-Sánchez.

Ambos trabajos comparten el objetivo de alcanzar un discurso inclusivo en la literatura infantil sobre prehistoria y basado en evidencias científicas,

proporcionando recursos educativos que reflejen una visión más justa de la prehistoria. Esta colaboración y adaptación de metodologías refuerza la importancia de revisar y actualizar los materiales educativos para eliminar los estereotipos de género.

## 6.2. Discord

Discord<sup>5</sup> es una red social de comunicación que ha sido una gran referencia para la parte lógica, ya que se ha tomado como referencia para la lógica de permisos del personal de administración. Esta parte ha sido desarrollada de manera similar, donde en un grupo el administrador puede crear los roles que se necesiten y aplicar unos permisos personalizados para los usuarios con esos roles.

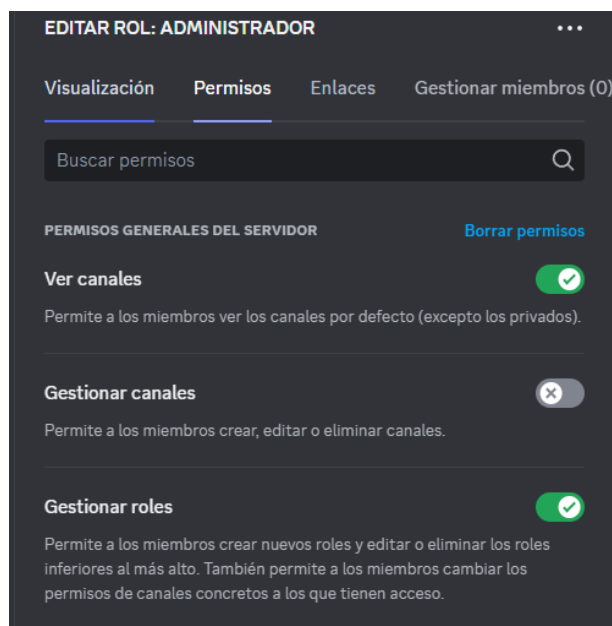


Figura 6.1: Permisos Discord

---

<sup>5</sup>Web de [Discord](#)

---

## 7. Conclusiones y Líneas de trabajo futuras

---

En este apartado se van a tratar las conclusiones obtenidas de este trabajo junto a posibles líneas de trabajo futuras por donde se puede continuar el proyecto.

### 7.1. Conclusiones

A continuación se detallan las conclusiones de este proyecto:

- El objetivo inicial se ha cumplido de manera satisfactoria e incluso pudiendo completar la aplicación con elementos adicionales. Al tener este proyecto finalmente desplegado, tanto docentes como familias tienen la oportunidad de obtener información que beneficie a la educación y poder participar descubriendo cómo de adecuado según los estudios científicos es un libro en lo relativo a sesgos de género.
- Las tecnologías propuestas inicialmente no han sido las que se han utilizado en todos los casos. Durante la realización del proyecto se decidió realizar un cambio de rumbo y pasar el *frontend* de Flask a Angular, permitiendo así desarrollar el proyecto en un entorno con más herramientas y posibilidades de escalar en el futuro.
- Gracias a la parte de investigación y necesidad de enfrentarme a diferentes retos completamente desconocidos, se ha aprendido a realizar investigaciones exhaustivas de una herramienta y comenzar a tener un criterio en relación a estudios de viabilidad.



- Ha sido complejo intentar realizar estimaciones acertadas, ya que han existido ocasiones en los que la tecnología ha sido completamente desconocida y no se podían realizar estimaciones fiables. Para estos casos, la organización en *sprints* ha sido muy beneficiosa para adaptar la carga de trabajo a las circunstancias.
- Se ha podido obtener mucho conocimiento acerca del trato con el cliente, ya que al tener reuniones y definir las especificaciones que debía de tener el producto, se generaban dos visiones totalmente distintas, las cuales había que unificar para encontrar un resultado acorde a la petición del cliente.

## 7.2. Líneas de trabajo futuras

A continuación se muestran posibles líneas de trabajo por las que se puede desarrollar este proyecto:

- Una línea de trabajo interesante sería el desarrollo de filtros para ordenar los libros y guardar la configuración para guardar esa decisión.
- Otra opción que podría ser interesante implementar, sería mejorar el *web scraping* mostrando un catálogo completo de opciones por fuente detectando sólo los elementos que sean libros.
- En un futuro se podría perfeccionar la API en relación a los permisos para poder permitir a terceros utilizar esa API para realizar sus propias integraciones.

---

## Bibliografía

---

- [1] Bootstrap 5. <https://getbootstrap.com/docs/5.0/getting-started/introduction/>, 2024.
- [2] Documentación de primeicons. <https://primeng.org/icons>, 2024.
- [3] Documentación de primeng. <https://primeng.org/>, 2024.
- [4] Documentación de sweetalert2. <https://sweetalert2.github.io/>, 2024.
- [5] Framework unittest. <https://docs.python.org/3/library/unittest.html>, 2024.
- [6] Introducción a Ajax. [https://www.w3schools.com/xml/ajax\\_intro.asp](https://www.w3schools.com/xml/ajax_intro.asp), 2024.
- [7] Amazon. Api amazon books. <https://webservices.amazon.com/paapi5/documentation/search-items.html>, 2024.
- [8] Auth0. Jwt debugger. <https://jwt.io/>.
- [9] John Bradley. Jwt estandar. <https://datatracker.ietf.org/doc/html/rfc7519>, 2015.
- [10] Desconocido. Documentación de Flask. <https://flask.palletsprojects.com/en/3.0.x/>, 2010.
- [11] Desconocido. Archivos robots.txt. [https://es.wikipedia.org/wiki/Est%C3%A1ndar\\_de\\_exclusi%C3%B3n\\_de\\_robots](https://es.wikipedia.org/wiki/Est%C3%A1ndar_de_exclusi%C3%B3n_de_robots), 2023.
- [12] Drawio. Funciones de drawio. <https://www.drawio.com/features>, 2024.

- [13] Laia Gilibets. Definición de kanban. <https://www.iebschool.com/blog/metodologia-kanban-agile-scrum/>, 2023.
- [14] Github. Inicio de github. <https://github.com/>, 2024.
- [15] Google. Api google books. <https://developers.google.com/books?hl=es-419>, 2024.
- [16] Google. Documentación de Angular. <https://angular.io/docs>, 2024.
- [17] Fernando herrera. Curso de angular. <https://www.udemy.com/course/angular-fernando-herrera/>, 2024.
- [18] <https://investigacion.ubu.es/investigadores/35522/detalle>.
- [19] Justinmind. Inicio de justinmind. <https://www.justinmind.com/>, 2024.
- [20] Fede J.Álvarez. Información jwt. <https://bcube.bitban.com/blog/sabias-que-json-web-token>, 2023.
- [21] Jeff Sutherland Ken Schwaber. Definición de scrum. <https://scrumguides.org/scrum-guide.html>, 2020.
- [22] Netlify. Documentación de Netlify. <https://www.netlify.com/>, 2024.
- [23] Overleaf. Página de inicio de Overleaf. <https://es.overleaf.com/project>, 2024.
- [24] Marylene Patou-Mathis. *El hombre prehistórico es también una mujer. Una historia de la invisibilidad de las mujeres*. Colección Lumen Ensayo, Penguin Random House. Barcelona, 2021.
- [25] Postman. Inicio de postman. <https://www.postman.com/product/what-is-postman/>, 2024.
- [26] Render. Documentación de render. <https://docs.render.com/>, 2024.
- [27] Margaret Rouse. Que es el web scraping. <https://www.techopedia.com/definition/5212/web-scraping>, 2023.
- [28] Jesús Alberto San Martín Zapatero, Delfín Ortega Sánchez, et al. Atribuciones de género y construcción de identidades en la literatura infantil sobre prehistoria. *Bellaterra Journal of Teaching & Learning Language & Literature*, 15(3):e973, 2022.