ALEJANDR.IA

Proyecto de una biblioteca de Inteligencia Artificial

Autor: Samuel Linares Fumero

Tokio.

Contenido

1.- Descripción general del proyecto

La programación junto con muchos otros avances tecnológicos nos ha permitido ahorrar el activo más preciado que tenemos y nunca vamos a recuperar, el tiempo. No obstante, el éxito de muchos desarrollos de software proviene de hacer resolver problemas relacionados con la duración. Y es por ello que decidí vincular mi proyecto final al ahorro de tiempo de los usuarios que usen la aplicación.

Con el avance de las nuevas tecnologías, disponemos de una cantidad inconmensurable de contenido formativo (libros, papers, estudios, investigaciones…). Es por ello, que he trabajado arduamente en buscar la forma de facilitar este trabajo resumiendo contenido con la ayuda de herramientas de inteligencia artificial.

Esa es la principal razón de ser de *Alejandr.ia*, una plataforma en la que se publicarán resúmenes animados de libros, estudios, informes e investigaciones, multitemáticas y con duraciones lo más reducidas posibles. Nuestra misión es democratizar el aprendizaje en la medida de lo posible, y hacer más fácil el camino a aprender nuevas habilidades con el contenido que hay en internet.

La web dispone de una base de datos robusta de obras organizadas tanto por categorías como por autores, con un buscador que hará mucho más sencilla la búsqueda del usuario.

Por otra parte, dispondrá de una membresía que estará habilitada cuando el producto mínimo viable se encuentre testado, y cuando la abundancia y la calidad del contenido nos permita cobrar por ello.

Por otra parte, en la parte inferior de la página se encuentran todos los apartados relacionados con los textos legales, las preguntas frecuentes o *FAQs*, el apartado *Sobre Nosotros* y el *Contacto*.

La web está programada también en base a un sistema de autenticación que permite al usuario tanto registrarse como iniciar sesión en su cuenta. Por motivos de seguridad, velamos por la protección de datos de acuerdo con la legislación vigente recogido en la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.

Una vez superado el sistema de autenticación, la usabilidad de la web es muy intuitiva. Las búsquedas son muy flexibles, para que encuentres exactamente la pieza de contenido que busca el usuario.

A continuación, voy a mencionar los objetivos de dicho proyecto además de su alcance.

2.- Objetivos y alcance del proyecto

Vamos a comenzar hablando de los objetivos del proyecto *Alejandr.ia* en el momento del desarrollo.

El proyecto de *Alejandr.ia* tiene unos objetivos claros, y es llegar a subir al menos 1.000 resúmenes en nuestra plataforma e ir añadiéndolos continuamente, para que cuando activemos el sistema de membresía nuestro usuario no se quiera ir de ella, así como lograr la incorporación de nuevos usuarios.

Por otra parte, la inversión es escasa, pero por muy pequeña que sea nuestro objetivo es superar el *break even* para poder hacer de *Alejandr.ia* un proyecto rentable y sostenible. En un principio, el método de pagos se realizaría a través de Stripe.

De momento nuestro objetivo es dirigirnos únicamente al mercado hispanohablante, y con esta infraestructura no nos planteamos por el momento expandirnos al mercado angloparlante, a no ser que el mercado nos obligue a hacerlo.

Es probable que dichos objetivos se alteren en cierta medida, o se modifiquen con el transcurso del tiempo. O incluso que se añadan nuevos objetivos relacionados con los perfiles de los clientes, así como análisis de tendencias sobre que libros son los más leídos por los clientes, así como la introducción de algoritmos que mejoren su experiencia en nuestra página.

Ahora vamos a hablar del alcance esperado de *Alejandr.ia*, lo cual es fundamental para no desviarnos de los objetivos principales del proyecto.

En este proyecto se incluirá un listado de resúmenes in crescendo en cuanto a cantidad. Es probable que a medida que transcurra el tiempo añadamos algunas categorías nuevas, al igual que mejoraremos la atención al cliente con alguna integración automatizada.

Todo el contenido que subamos a la plataforma será derivado de fuentes que citaremos y sin copyright. Además, si llegamos a algún acuerdo con algún autor de un libro o estudio, podremos añadir algún tipo de funcionalidad especial.

Por el contrario, no se incluirá ningún tipo de curso de ninguna de las tipologías. La razón de ser del proyecto es únicamente resumir contenido ya existente para que las personas no pierdan demasiado tiempo si lo único que buscan es entender los fundamentos. Tampoco incluiremos contenido deliberadamente falseado, o que dudemos de su fiabilidad y rigor.

En definitiva, incluiremos en la *landing page* del proyecto cualquier tipo de novedad que tenga lugar.

3.- Stack tecnológico

Para la realización del proyecto he utilizado en mi stack tecnológico diversas herramientas tanto de programación como de no code. Las definiré a continuación y luego explicaré por qué he elegido todas y cada una de ellas.

- Django
- MySQL Workbench
- Visual Studio Code
- Diagrams.net
- Vimeo
- Herramientas de IA no code
- Remove.bg
- Speechgen
- Adobe Express
- ChatGPT

Ahora vamos con la explicación de por qué he seleccionado dichas herramientas para el desarrollo del proyecto.

- Django.

Django ha sido el *framework* que he usado para construir el *backend* de la aplicación. Al ser una aplicación web, he descartado *frameworks* de aplicaciones de escritorio como *Tkinter*. Por otra parte, me he decantado por Django en lugar de otros *frameworks* de aplicaciones web como *Flask* porque es un paquete muy completo. Django ofrece muchas funcionalidades que están listas para usar, como es el caso de la autenticación de usuarios, por ejemplo. Django proporciona por defecto unas plantillas de registro de usuarios que hace más fácil la creación de *users* y *superusers*. Otra es la administración automática que proporciona Django, ya que desde el URL del administrador es muy sencillo crear, modificar y eliminar tanto usuarios como productos.

Además, Django posee un ORM propio, por lo que no hay que instalar otro externo como *SQL Alchemy*. Es muy sencillo crear modelos con su ORM crear, eliminar, editar y leer datos de la base de datos. A la par que sencillo es problemático si no concuerda lo establecido en Django con lo existente en la base de datos, pero realizando las migraciones adecuadas no tiene por qué haber problema.

Otro aspecto que debemos considerar es la seguridad. *Django* ofrece una protección contra ataques comunes (CSRF, XSS, etc.) y contra inyecciones SQL.

En cuanto a la organización, Django se lleva el primer premio, ya que su estructura *model-view-template* hace que sea más fácil organizar las diferentes aplicaciones que crees dentro de tu proyecto.

Por otra parte, era un *framework* que no había usado antes, y era una buena oportunidad para salir de la zona de confort y aprender a dar mis primeros pasos.

- MySQL Workbench

Tras analizar diversos tipos de bases de datos, me decidí por usar MySQL por las razones que mencionaré posteriormente en el apartado de modelos de datos.

Por ello, elegí MySQL Workbench, una herramienta visual de diseño y administración de datos que se conecta a bases de datos de MySQL.

- Visual Studio Code

Visual Studio Code es un editor de texto que tiene ciertas capacidades de algunos IDEs como PyCharm pero es más ligero y flexible. Este fue el principal motivo por el que lo elegí. Además,

es muy personalizable, por lo que puedes agregar extensiones para una gran diversidad de lenguajes, aunque en este proyecto usé principalmente Python para el backend y HTML y CSS para el frontend.

La rapidez de Visual Studio Code es superior a la de muchos editores e IDEs como es el ejemplo de Atom o Sublime Text, los cuales fueron una consideración al inicio del proyecto, pero rápido abandoné la idea.

- Diagrams.net

Para crear los diagramas de las bases de datos he acudido a la página de diagrams.net ya que me permite una gran flexibilidad

- Vimeo

Esta no es una herramienta de programación como tal, pero si es una página de alojamiento de contenido. En esta página estarán alojados los videos, ya que es mejor contar con un servicio externo, tanto por temas de seguridad como de rapidez de carga de la página. Subir los videos directamente a Django hubiera ocupado más almacenamiento y estaría usando muchos más recursos, por lo que sería más lenta.

- ChatGPT

La herramienta de ChatGPT me ha sido de gran ayuda en la generación de textos, como puede ser el aviso legal o la política de privacidad. Además, gracias a la funcionalidad incorporada de DallE en la versión de pago de ChatGPT-4o pude crear imágenes hiperrealistas de la temática de la página web a mi medida. He preferido optar por ChatGPT en lugar de Midjourney porque con la nueva actualización considero que me va a dar resultados más exactos con el mismo prompt si ya sabe algo del proyecto por consultas que he hecho anteriormente.

- Herramientas de IA no-code

Para la realización de los videos he hecho uso de dos herramientas de inteligencia artificial no code. Primero he usado speechgen para la realización de los audios, ya que me permitía usar una voz lo más natural posible a un costo lo más bajo posible. Es decir, esta herramienta es la que encontré con mejor calidad precio para realizar transcripciones de texto a voz. Por otra parte, he

usado otra herramienta de Adobe para poner la animación a dichos videos. Esta ha sido Adobe Express, ya que es una de las pocas que existe en el mercado que te deja animar videos de más de 1 minuto.

4.- Modelo de datos

La base de datos que elegí para el proyecto es MySQL, una base de datos que pertenece a Oracle, y el motivo de mi elección te lo comentaré a continuación.

La base de datos del proyecto debía ser relacional, ya que algunas de las claves primarias en algunas tablas eran claves foráneas en otras. Es decir, las tablas debían estar relacionadas entre sí. Por consiguiente, no elegí ningún tipo de base de datos NoSQL como puede ser MongoDB.

Para crear este proyecto lo ideal es una base de datos fuerte y robusta, así que descarté SQLite pese a que era la que por defecto aparece en la configuración de Django. A diferencia de SQLite, MySQL es muy superior en escalabilidad y seguridad. No obstante, una biblioteca digital requiere de una base de datos que no quede obsoleta al instante.

MySQL no es una base de datos relacional con una complejidad demasiado alta, como pueden serlo otras como es el caso de SQLServer. Esto significa que ocupa menos almacenamiento en memoria.

Otro aspecto importante en mi elección de la base de datos de MySQL es la rapidez en consultas de lectura, frente a otras bases de datos como PostgresSQL. Mi objetivo era la creación de una base de datos versátil que respondiera bien a diversas consultas, dando por hecho que los usuarios iban a acceder en una gran cantidad de ocasiones al día, ya que cada resumen es un registro.

Los modelos de la base de datos en Django fueron creados usando el ORM incorporando en el paquete del framework. La aplicación raíz del proyecto, la cuál tiene el nombre de myproject contiene un archivo llamado settings.py con el que se vinculó la conexión a dicha base de datos de MySQL. Los nombres de los modelos creados tienen variables con nombre en español mientras que los que vinieron por defecto en django mantienen sus nombres originales en inglés. Esto se debe primordialmente a evitar discordancias en cuanto a las variables del proyecto y algunas de las palabras reservadas que Django utiliza. Como por ejemplo, el caso de *login* que me daba problemas en la ejecución, porque Django lo interpretaba de otra manera, y lo tuve que sustituir por *inicio sesión*.

El modelo que Django ha creado por defecto es concretamente el que maneja los clientes y administradores, dentro de la aplicación Accounts, concretamente en el archivo models.py.

Los demás modelos de la base de datos se encuentran dentro del archivo de models.py pero de la aplicación de resumenes. Aquí se definen las clases de

Libro donde se añadirán todos los resúmenes, Categoria que contiene las categorías en las que se van a dividir los libros y Autor donde se reflejará el autor que ha escrito ese libro. Cada categoría y autor son únicos, así que se han usado claves foráneas para evitar cualquier tipo de duplicidad.

5.- Explicación de los requisitos de la aplicación

Los requisitos de la aplicación están localizados en un archivo llamado requirements.txt en el archivo. Estos requisitos hacen referencia a todos los frameworks, librerías y dependencias que es necesario instalar como mínimo para poder ejecutar el proyecto.

Voy a explicar todos y cada uno de los requerimientos y adjunto una captura de pantalla final del archivo donde queda explicado.

- **asgiref==3.8.1**

Esta librería es parte del soporte de Django para *ASGI (Asynchronous Server Gateway Interface)*. ASGI es una especificación que permite manejar peticiones asíncronas en Django, necesaria para la ejecución de aplicaciones en tiempo real como WebSockets. Si tu proyecto no utiliza funcionalidades asíncronas, aún así es recomendable que la instales para un mejor funcionamiento del framework.

```
9
10  import os
11
12  from django.core.asgi import get_asgi_application
13
14  os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'myproject.settings')
15
16  application = get_asgi_application()
17
```

- **Django==5.1**

Este es el framework principal que usé para desarrollar la aplicación web. Django proporciona las herramientas para construir la aplicación, gestionar bases de datos, crear plantillas y manejar rutas, entre otras cosas. Un framework completo que explicaré como instalar en el siguiente punto.

- **mysql==0.0.3**

Este paquete es un cliente básico para MySQL, aunque es bastante limitado. Fue el primero que utilicé, pero posteriormente tuve que descargar

otra librería que es más completa como *mysqlclient* para interacciones robustas con bases de datos MySQL. Este último lo comentaremos a continuación.

- **mysqlclient==2.2.4**

Es un cliente para MySQL que permite que tu proyecto de Django se comunique con la base de datos MySQL. Esta librería es esencial si estás utilizando MySQL como sistema de gestión de bases de datos.

- **python-decouple==3.8**

Te permite separar las configuraciones sensibles (como claves secretas, credenciales de bases de datos, etc.) en un archivo *.env*. Esto es importante para mantener segura la información confidencial y evitar exponerla en el código fuente.

```
96
97  DATABASES = {
98      'default': {
99          'ENGINE': 'django.db.backends.mysql',
100         'NAME': config('DB_NAME'),
101         'USER': config('DB_USER'),
102         'PASSWORD': config('DB_PASSWORD'),
103         'HOST': config('DB_HOST', default='localhost'),
104         'PORT': config('DB_PORT', default='3306'),
105     }
106 }
107
```

- **sqlparse==0.5.1**

Es una librería de análisis de SQL que Django utiliza internamente para procesar y formatear consultas SQL. No la he usado directamente, pero aparece en el archivo de requerimientos porque es necesaria para el funcionamiento de Django con bases de datos.

- **tzdata==2024.1**

Este paquete contiene información sobre zonas horarias, lo cual es utilizado por Django para manejar de manera correcta fechas y horas en diferentes regiones geográficas. Aparece en este archivo porque en los modelos de *accounts* aparecen algunas variables que son de tipo DateField.

```python
class Account(AbstractBaseUser):
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)
    email = models.EmailField(max_length=100, unique=True)
    username = models.CharField(max_length=100, unique=True)
    phone_number = models.CharField(max_length=100, blank=True, null=True)
    date_birth = models.DateField(blank=True, null=True)

    # Campos de membresía y administración
    is_member = models.BooleanField(default=False)
    membership_date = models.DateTimeField(null=True, blank=True)
    date_joined = models.DateTimeField(auto_now_add=True)
    last_login = models.DateTimeField(auto_now=True)
    is_active = models.BooleanField(default=True)
    is_staff = models.BooleanField(default=False)
    is_admin = models.BooleanField(default=False)
    is_superuser = models.BooleanField(default=False)
```

Y este sería el archivo requirements que desglosamos previamente:

requirements.txt

```
asgiref==3.8.1
Django==5.1
mysql==0.0.3
mysqlclient==2.2.4
mysqldump==0.0.10
python-decouple==3.8
sqlparse==0.5.1
tzdata==2024.1

```

6.- Manual de instalación

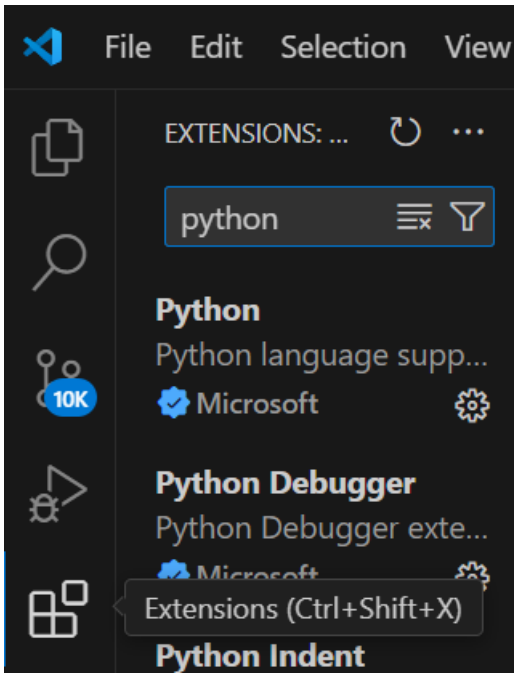
Vamos a comenzar con el manual de instalación de librerías y dependencias del proyecto desde cero. Lo primero de todo es descargar alguna de las versiones de Python 3 en el dispositivo. En este caso accederemos al link oficial que verás a continuación. Ahí puedes elegir la versión que prefieras según tu sistema operativo. Yo te recomiendo instalar la última versión, aunque cualquiera es viable si comienza con el número *3*.

Link para la descarga de Python: https://www.python.org/downloads/

Lo siguiente es descargar el ide, en este caso el editor de texto Visual Studio Code, que es muy similar a un IDE. Para ello, accedemos a la página oficial de Visual Studio Code y repetimos el proceso.

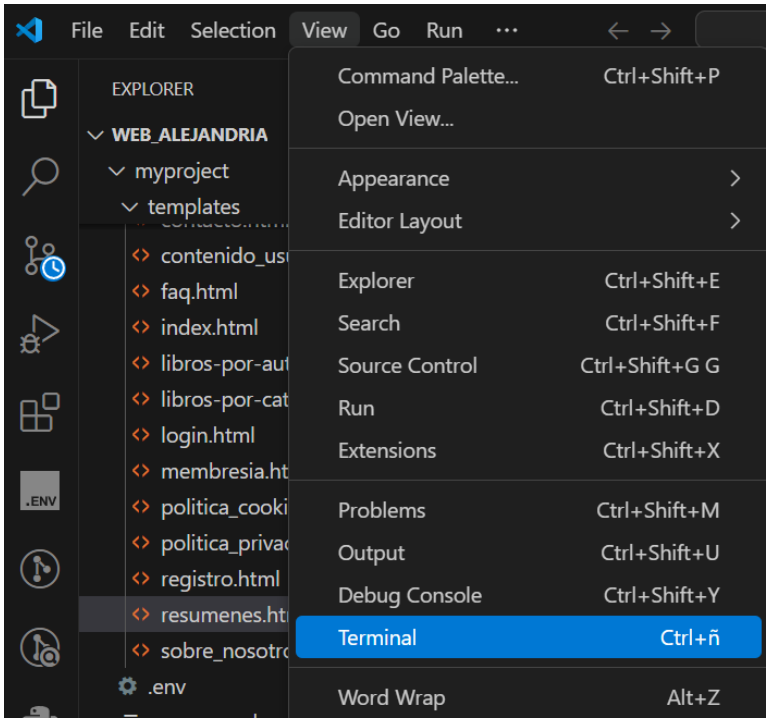
Link para la descarga de Visual Studio Code:
https://code.visualstudio.com/download

Una vez instalado deberás agregar la extensión de Python desde la pestaña de extensiones.



Tienes dos opciones para ejecutar el código, usar la terminal de Visual Studio Code o usar Git Bash, yo usé la propia terminal de Visual Studio Code por comodidad y compatibilidad.

Para ello, dirígete a la esquina superior izquierda “View>Terminal”, tal y como se muestra en la siguiente foto.



Para comprobar tu versión de Python desde la terminal, puedes ejecutar el siguiente código:

```
(env) PS C:\Users\Samuel\Side proyects\web_alejandria> python --version
Python 3.11.3
```

Vamos ahora con la descarga del pip, que es lo que nos permitirá instalar muchos paquetes, librerías y frameworks de una forma óptima. Lo primero es comprobar si lo tenemos o no instalado en nuestro sistema. Ejecutamos el siguiente código en la terminal de Visual Studio Code o en Git Bash el siguiente código:

```
pip --version
```

Si no lo tenemos instalado, nos dirigimos a la siguiente página para descargar el script de instalación de pip:

https://bootstrap.pypa.io/get-pip.py

Luego vamos a la terminal y volvemos a ejecutar este código para instalar pip mediante el script que descargamos:

```
python get-pip.py
```

Ahora lo siguiente es instalar Django en el terminal para poder ejecutar el código. Django es el framework que he utilizado para el desarrollo de la web, y se puede instalar directamente desde la terminal. Para ello utilizamos el comando *pip install Django.* De esta forma se instalará la versión *Django 5.1*, que fue la que yo utilicé para el proyecto.

Por si acaso, dejo la documentación oficial de Django en el siguiente enlace.

Link de la documentación de Django: https://docs.djangoproject.com/en/5.1/

Lo ideal es crear un nuevo entorno de desarrollo para poder ejecutar el código de forma aislada. Es decir, puedes aislar las dependencias para que no interfieran con paquetes del sistema. Además es fundamental para compartirlo y que todos puedan trabajar con las mismas versiones y dependencias.

Para crearlo ejecutaremos este código. Es ideal por buenas prácticas llamarlo venv, ya que es un concepto que todos asocian al virtual environment (entorno virtual en inglés).

```
python -m venv venv
```

Luego para activarlo deberás ejecutar en la terminal el siguiente código:

```
venv\Scripts\activate
```

Para poder ejecutar este proyecto hay que instalar algún software de bases de datos MySQL y replicar la base de datos. En este caso, te recomiendo instalar *MySQL Workbench 8.0 CE*, que es la versión de la comunidad y es totalmente gratuita.

Link para descargar MySQL Workbench Community:

Las claves de mi base de datos están dentro de un archivo *.env* protegidas con variables. Pero puedes crear tus propias claves a partir de mi archivo .env si ejecutas el siguiente código en tu terminal y presionas *enter*. El código a ejecutar es el siguiente:

```
cp .env.example .env
```

Luego debes completarlo con unas credenciales que recuerdes.

Para entrar a MySQL desde la terminal debes ejecutar el siguiente comando:

```
mysql -u root -p
```

Lo siguiente es crear una base de datos completamente nueva, que luego se quedará vinculada al proyecto de Django. Para ello, ejecuta el siguiente comando en la terminal:

```
CREATE DATABASE nombre_de_la_base_de_datos;
CREATE USER 'nombre_usuario'@'localhost' IDENTIFIED BY 'contraseña';
GRANT ALL PRIVILEGES ON nombre_de_la_base_de_datos.* TO 'nombre_usuario'@'localhost';
FLUSH PRIVILEGES;
```

Posteriormente, debes hacer las migraciones para que los datos de la base de datos se repliquen en la que acabas de crear. Y lo harás de nuevo en la terminal con el siguiente comando:

```
python manage.py migrate
```

Lo siguiente es crear un superuser, es decir un administrador. Esto lo podemos ejecutar con el siguiente código:

```
python manage.py createsuperuser
```

Para acceder desde otro dispositivo, se podría con una correcta configuración del acceso remoto. Para ello hay que editar el archivo llamado my.cnf

bind-address = 0.0.0.0

Luego hay que reiniciar el servidor de MySQL y asegurarse de que el usuario esté habilitado para conexiones remotas.

```
GRANT ALL PRIVILEGES ON nombre_de_la_base_de_datos.* TO 'usuario'@'%' IDENTIFIED BY 'contraseña';
```

De esta forma puedes manipular el proyecto con base de datos propia.

7.- Conclusiones

He sacado en claro muchas conclusiones de este proyecto que iré desglosando una por una a continuación.

La principal conclusión que saco del proyecto, y por lo cuál he decidido hacer este en concreto, es que el tiempo es uno de los activos más preciados que puedes ahorrar programando una página web que resuelva una necesidad real. Al final, en el desarrollo web, y en el desarrollo software en general, un buen programador es aquel que resuelve problemas reales mediante el código. En cuanto a la metodología de la realización del proyecto saco como conclusión que la metodología lean de crear un producto mínimo viable para testearlo y luego mejorarlo o pivotar es la clave en desarrollos de este tipo.

Podría haberme complicado y hacer un código mucho más complejo e inútil, pero en el largo plazo es más difícil de mantener, y de desarrollar. Como dice una frase que me gusta mucho repetir una y otra vez en este rubro, tu stack tecnológico es como un martillo, y por muy grande y potente que sea tu martillo no todos los objetos son clavos. Es decir, muchas veces las soluciones son más simples de lo que pensamos. Llegué a la conclusión de esto cuando en lugar de usar el extends para reciclar código de html, lo que hice en un principio fue copiar y pegar el código en todas las plantillas. Esta práctica me quitaba mucha energía y paciencia ya que cuando quería modificar alguna parte del código, tenía que hacerlo en todas las plantillas, y si se me olvidaba hacerlo en alguna descuadraba todo el código.

Por otra parte, he descubierto la gran cantidad de paquetes que tiene django internamente, y aunque siempre se puede saber más, he aprendido mucho. A decir verdad, el desarrollo web en Python da muchas posibilidades que nunca imaginé que pudiera llegar a hacer en este lenguaje. Y también me sorprendo de mi evolución desde el principio del curso hasta ahora, desarrollando código que nunca imaginé hacer por mi mismo.

Es cierto que muchas veces tuve dudas de si este proyecto valdría lo suficiente tanto para ser un entregable decente como para sacarlo al mercado próximamente, pero poco a poco cuando las implementaciones y mejoras han ido cogiendo forma, me he ido convenciendo poco a poco de ello.

Me gustaría leer todas y cada una de las correcciones que podáis hacerme, para aprender de mis errores, y en aplicarlas en los próximos proyectos. Así como consejos de implementaciones que pueda hacer en el largo plazo para agregar un valor añadido.

En definitiva, ha sido un proyecto muy enriquecedor que me motiva a seguir mejorándolo con el tiempo.

8.- Evolutivos del proyecto

Como he mencionado anteriormente, el proyecto en sí es un producto mínimo viable. A medida que siga trabajando en el proyecto agregaré mejoras y funcionalidades nuevas. Además, he aprendido nuevas mejoras que me servirán para proyectos futuros. No obstante, este apartado lo dividiré en dos: primero comentaré los aspectos que puedo implementar o mejorar en este proyecto y luego que tengo pensado para futuros desarrollos.

En este proyecto se me ocurren muchas mejoras que puedo integrar en un futuro, como por ejemplo la integración de chatbots en el área de atención al cliente, para resolver respuestas repetitivas que puedan ser programadas.

Una de las mejoras que iré implementando en el futuro es la integración de APIs que aporten nuevas funcionalidades a la página y mejoren la eficiencia, por ejemplo, alguna API de alguna herramienta IA que me ayude con los resúmenes de muchos libros. Esto no solo haría que la página sea más interactiva, sino que la convertiría en una herramienta mucho más útil para los usuarios que buscan recomendaciones personalizadas.

En las nuevas especializaciones de Tokio School de Inteligencia Artificial y Machine Learning puedo aprender sobre nuevos frameworks que me permitan añadir funcionalidades extra, ya sea en el estudio de patrones de visualizaciones de resúmenes, para mejorar la usabilidad y experiencia del usuario, como de funcionalidades extra que puedan añadir un valor al producto inicial. De este modo, podría implementar mejoras que no solo aumenten la usabilidad de la plataforma, sino que también la conviertan en un recurso indispensable para aquellos que buscan información de manera rápida y efectiva.

Se me ocurren algunas otras mejoras que puedo hacer como implementar gamificación para mejorar la experiencia del usuario. Y de esa forma aprovechar para que lea más resúmenes.

Por otra parte, hay muchas ideas que se me ocurren para futuros desarrollos, como, por ejemplo, comenzar a aprender sobre algunas otros frameworks como puede ser Streamlit para la creación de aplicaciones nativas usando Python. Es una idea que me da curiosidad probar de cara a futuros desarrollos.

La ventaja de usar Python en futuros desarrollos es que otorga la posibilidad de integrar muchas APIs, desde las de Google como pueden ser maps, sheets o calendar, hasta incluso la de idealista y demás páginas que tengan un apartado para desarrolladores.

En conclusión, el proyecto actual es solo el primer paso hacia un desarrollo continuo. Las mejoras y funcionalidades que tengo en mente no solo enriquecerán esta plataforma, sino que me permitirán seguir creciendo como desarrollador y explorar nuevas áreas de la tecnología. Estoy convencido de que el camino que me queda por recorrer está lleno de oportunidades para seguir aprendiendo e innovando.