Memoria Parcial III

Ingeniería web

https://mi-mapa-frontend.vercel.app/

Antonio Cañete Baena

Contenido

Diseño de base de datos

Para el diseño de la base de datos, se ha utilizado **MongoDB Atlas** como sistema de almacenamiento en la nube. El diseño de la base de datos ha sido modelado en torno a dos colecciones principales: **Chincheta** y **Log**, que representan las principales entidades de la aplicación y almacenan la información relevante de manera estructurada.

Las colecciones y sus atributos son los siguientes:

- **Chincheta**
    - CreadorEmail
    - Lugar
    - Lat
    - Lon
    - imagenURI
    - fecha_creacion
- **Log**
    - Uid (token de oaut)
    - emailVisiatante
    - emailPropietario
    - visitTime

Acceso a la Base de Datos

El acceso a la base de datos se realiza mediante una conexión desde Python, utilizando un usuario que tiene permisos de **lectura y escritura**, sin privilegios de administración. Este usuario cuenta con las siguientes credenciales:

- **Nombre de usuario**: user
- **Contraseña**: ant22031

El clúster en MongoDB Atlas donde se almacena esta base de datos se denomina **examen3wb** y la base de datos donde se encuentra los datos es **MiMapa**.

Para acceder a la base de datos desde **MongoDB Compass**, la URI de conexión es la siguiente (reemplazando <db_username> y <db_password> con el usuario y la contraseña correspondientes):

mongodb+srv://<db_username>:<db_password>@examen3web.hya9n.mongodb.net

/

Tecnología usada

Backend

Para el desarrollo de la API se ha elegido el lenguaje de programación Python por su simplicidad y claridad, que permiten una implementación limpia y eficiente. En combinación con FastAPI, se ha logrado construir una API estable, robusta y estructurada con una integración directa de OpenAPI, que facilita la generación de documentación detallada y fácil de interpretar.

- Lenguaje: Python
- Framework: FastAPI

Además, para el despliegue del servicio web, se utiliza Docker para crear y gestionar contenedores, lo que permite una replicación sencilla del servicio en cualquier entorno compatible.

Estructura del Código

La estructura principal del código se compone de tres archivos esenciales:

1. db_connection.py: Contiene la lógica de conexión y las operaciones (consultas, actualizaciones y eliminaciones) en la base de datos.
2. app.py: Principal archivo de la API, donde se definen todos los endpoints y la lógica central del servicio.
3. XXXX_models.py: Define los modelos de datos utilizados por OpenAPI para describir y validar las estructuras de datos en las solicitudes y respuestas de la API.
4. XXXX_endpoints: Define los endpoints característicos de cada colección de la base de datos.

Frontend

Para el diseño frontend se ha optado por el uso de React, una biblioteca que resulta muy útil debido a su flexibilidad y capacidad para crear interfaces de usuario dinámicas y reutilizables.

A continuación, se destacan algunos aspectos clave de la aplicación:

- Contextos: Los contextos en React son herramientas poderosas que permiten compartir datos entre componentes sin necesidad de pasar props manualmente en cada nivel. En esta aplicación, se han implementado dos contextos principales:
    - APIContext: Gestiona las llamadas a la API, centralizando y simplificando la comunicación con los servicios externos.
    - AuthContext: Se encarga de todas las operaciones relacionadas con la autenticación, incluyendo el inicio de sesión mediante OAuth.
-

- Componentes: React facilita la creación de componentes independientes y reutilizables. En esta aplicación, se han desarrollado varios componentes, como los mapas, que pueden integrarse fácilmente en diferentes partes del proyecto.

Para la implementación de OAuth se ha utilizado Firebase, una solución que permite una integración completa con un alto nivel de funcionalidad y una configuración sencilla.

Instrucciones de instalación.

A continuación, se detallan los pasos necesarios para instalar y ejecutar la aplicación:

1. Descarga de archivos

Es necesario obtener los archivos del repositorio, lo cual puede realizarse de dos formas:

Clonando el repositorio mediante Git.

Descargando el archivo ZIP y descomprimiéndolo en la ubicación deseada.

Nota importante: Es imprescindible contar con los archivos .env correspondientes, que incluyen las claves y configuraciones necesarias para que la aplicación funcione correctamente.

2. Configuración del Frontend

El desarrollo del cliente, implementado con React y Node.js, se encuentra en la carpeta denominada frontend. Para configurarlo, siga los pasos a continuación:

Acceda a la carpeta frontend.

Ejecute el comando **npm install** para instalar las dependencias requeridas.

Posteriormente, utilice el comando **npm run dev** para iniciar el servidor de desarrollo.

Una vez completados estos pasos, podrá acceder a la parte del cliente desde el navegador.

3. Configuración del Backend

El backend de la aplicación, desarrollado con FastAPI, se encuentra configurado para ejecutarse mediante Uvicorn. Existen dos métodos para iniciar el backend:

Ejecutando el archivo **run_local.bat**, ubicado en el directorio raíz del proyecto, el cual inicia automáticamente Uvicorn con la API ya montada.

Iniciándolo manualmente mediante el archivo app.py, utilizando el siguiente comando:

uvicorn app:app --reload

Con cualquiera de estas opciones, el backend estará listo para recibir solicitudes.

Despliegue

La aplicación ha sido desplegada utilizando Vercel, separando el frontend y el backend en dos proyectos independientes. Este enfoque permite una gestión más modular y

simplificada de cada componente. Además, Vercel permite guardar todas la variables de entorno de forma segura en su página.

- Frontend:
  El frontend, desarrollado con React, ha sido desplegado en el proyecto de Vercel correspondiente. La URL pública para acceder al frontend es la siguiente:
  https://mi-mapa-frontend.vercel.app/

- Backend:
  El backend, implementado con FastAPI, también se encuentra desplegado como un proyecto independiente en Vercel. La URL pública para acceder al backend es:

  https://mi-mapa-backend.vercel.app/

Actualización Automática

Ambos proyectos están configurados para realizar actualizaciones automáticas. Cada vez que se realiza un push a la rama main del repositorio https://github.com/antbaena/mimapa, Vercel detecta los cambios y realiza un nuevo despliegue, asegurando que la versión más reciente del código esté siempre activa en los entornos de producción.

Este flujo automatizado garantiza una integración continua eficiente y una rápida implementación de mejoras o correcciones.

Consideraciones y limitaciones

Chincheta es la colección que contiene los datos, y se busca por el correo electrónico del usuario que la crea.

Al crear la chincheta, el campo de correo electrónico no se puede modificar, ya que se toma del usuario que ha iniciado sesión. Además, la latitud y la longitud solo se pueden obtener mediante geocoding, evitando así la introducción de ubicaciones con nombres poco precisos.

Antes de crear una chincheta, se realiza una llamada a Cloudinary para subir la imagen y obtener la URL segura.

Al visitar, primero se verifica que el usuario tenga chinchetas. Luego se genera un log con los datos del UID de OAuth, ya que no es una buena idea almacenar el token (pues es volátil y poco seguro), mientras que el UID es un identificador único y más confiable.

Esta nueva versión está correctamente implementada y corrige errores previos.