Examen Parcial Ingeniería Web

Memoria de examen

Jaime Ezequiel Rodríguez Rodríguez

Diseño de la base de datos

La base de datos ha sido creada en MongoDB, estará diseñada para acceder desde cualquier IP. La URI y las credenciales son las siguientes:

```
MONGO_URI=mongodb+srv://jaezro:examen@clusterexamenbackend.nonxk.mongodb
.net/?retryWrites=true&w=majority&appName=ClusterExamenBackend
```

La URI ha de quedar como indica en la variable MONGO_URI en un archivo “.env”, dentro del proyecto. Dentro de la entrega debería venir incluido el “.env” en caso de problema únicamente es crear dicho archivo en la dirección especificada e incluir la línea de código superior.

La base de datos no está poblada más allá de lo usado para las pruebas del examen ya que esta se poblará conforme los usuarios interactúen.

La estructura de la base de datos mantiene el siguiente formato definido por el enunciado.

Tareas:

Atributos:

- _id: Identificador generado automáticamente por MongoDB y que es de utilidad para el gestor de base de datos para hacer operaciones internas, por la misma razón lo conservamos. En esta entidad lo usaremos como su identificador
- responsable: Un atributo del tipo string que es el email que usaremos como identificador de un usuario o también llamado colaborador
- descripción: Un campo tipo string que es la descripción de la tarea a realizar con un máximo definido de 50 caracteres.
- Habilidades: Representado en MongoDB como un Array de elementos tipo string. En este caso son las habilidades adecuadas para participar en la tarea.
- Segmentos: Un valor numérico que es el número de horas que se dedicarán de forma estimada a la tarea.
- Colaboradores: Un array de strings que serán los emails que identifican a los colaboradores de la tarea

Relaciones:

- Se relaciona con colaboradores mediante de uno a muchos mediante el atributo colaboradores.

Colaboradores:

Atributos:

- _id: Identificador generado automáticamente por MongoDB y que es de utilidad para el gestor de base de datos para hacer operaciones internas, por la misma razón lo conservamos.
- Email: Un atributo del tipo string que es el email del usuario. En este caso lo usamos como identificador único y nos aseguramos que así sea en la base de datos.
- Nombre: Un atributo del tipo string representando el nombre del usuario.
- Representado en MongoDB como un Array de elementos tipo string. En este caso son las habilidades que posee el usuario.

Tecnologías utilizadas

Algunas de las tecnologías usadas para el examen realizado:

- **FastAPI**: Un framework de Python moderno y de alto rendimiento para construir APIs rápidamente. Permite el desarrollo de aplicaciones web y APIs RESTful con tipado automático y fácil de usar.
- **OpenAPI**: Una especificación estándar para diseñar, documentar y consumir APIs. Define de forma estructurada cómo debe funcionar una API, facilitando la interoperabilidad y documentación.
- **Motor**: Un cliente asíncrono para MongoDB en Python que permite realizar operaciones eficientes y no bloqueantes en la base de datos, ideal para aplicaciones de alto rendimiento.
- **MongoDB**: Base de datos NoSQL orientada a documentos, diseñada para manejar grandes volúmenes de datos y escalar horizontalmente. Almacena datos en formato BSON (similar a JSON).
- **Python**: Un lenguaje de programación versátil y fácil de aprender, utilizado en una amplia variedad de aplicaciones, desde desarrollo web hasta ciencia de datos e inteligencia artificial.
- **Swagger UI**: Interfaz gráfica que permite visualizar y probar de manera interactiva APIs basadas en OpenAPI, facilitando a los desarrolladores explorar las funcionalidades de la API sin necesidad de escribir código adicional.

Para justificar el archivo `openapi.json` adjunto hemos de comprender que Swagger UI genera una interfaz interactiva que se basa en el archivo `openapi.json` de la API, el cual describe su estructura y operaciones. Al trabajar en conjunto con FastAPI, Swagger UI toma esta especificación y la usa para generar documentación visual y un entorno de pruebas. Esto garantiza que el `openapi.json` refleje fielmente los endpoints, parámetros y respuestas de la API, ya que Swagger UI se basa en este archivo para representar toda la funcionalidad disponible, asegurando la validez y consistencia del diseño de la API.

Despliegue con Docker

Para el despliegue de la aplicación haremos uso de Docker, para ello necesitamos una terminal de Linux en la que ejecutar el proceso de despliegue. En caso de tener un sistema Windows podremos hacer uso de WSL.

Como recomendación personal recomiendo crear una carpeta en el directorio deseado para mantener nuestro sistema organizado (mkdir <nombre de la carpeta nueva>).

Podemos hacer un despliegue desde el archivo .zip entregado de forma que el comando de “git clone” es sustituido por el proceso de copiar el contenido del zip, el resto del proceso no varía.

Para el despliegue mediante github primero descargamos en nuestra máquina el repositorio de github (o incluimos en la carpeta en la que nos encontramos en la terminal el zip adjunto en la entrega).

```
git clone https://github.com/jaiRodRod/Parcial02.git
```

Accedemos a la carpeta donde se encuentra nuestro proyecto.

```
cd Parcial02_JaimeRodRod/
```

Por seguridad NO se incluye el archivo .env en el repositorio GITHUB, se puede incluir manualmente desde el zip entregado o haciendo el siguiente conjunto de comandos dentro de la carpeta del proyecto a la misma altura que encontramos nuestro Dockerfile:

```
nano .env
```

Y en el .env incluimos el contenido expuesto anteriormente (también repetido en la siguiente figura):

```
MONGO_URI=mongodb+srv://jaezro:examen@clusterexamenbackend.nonxk.mongodb
.net/?retryWrites=true&w=majority&appName=ClusterExamenBackend
```

Finalmente desplegamos el compose que hay preparado que construye el contenedor y los lanza localmente.

```
docker compose up -build
```

El servicio de la API se desplegará en el **localhost:8000**, se puede acceder a una documentación sencilla mediante Swagger accediendo a la ruta **localhost:8000/docs#**.

Limitaciones y problemáticas

A lo largo del proyecto siendo consciente se han aplicado ciertas malas prácticas debido a la brevedad del examen en comparación a todo el trabajo requerido.

El uso de “sub-urls“ para evitar problemas con el filtrado como por ejemplo en el caso de “/tareas/habilidad”. Esto podría haberse hecho mejor con un filtrado mucho más sofisticado en la url “/tareas/”. En un proyecto mayor esto sería refactorizado.

El proyecto podría contar con una mayor comprobación de errores y manejo de los mismos pero de nuevo debido al tiempo esto no ha sido posible.