MANUAL TÉCNICO PROYECTO 2 BACKEND 202003654

1. DESCIPCION GENERAL DE LA SOLUCION

A. Para el funcionamiento del backend se utilizó Python+FastApi para conectar a una base de datos no relacional de mongo, utilizando queries y aggregates para el manejo de los datos del sistema.

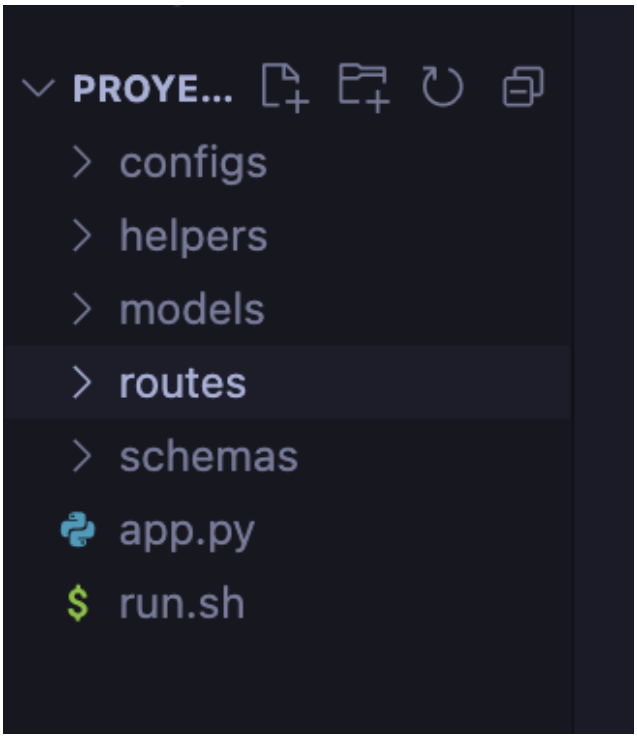
2. REQUERIMIENTOS DEL SISTEMA

A. MongoDB v6.0.6

B. Python 3

C. Pip 3

3. ESTRUCTURA DEL PROYECTO



A. Configs
    i. Configuraciones generales del proyecto, como la conexión a la base de datos.
B. helpers
    i. Funciones útiles en diferentes archivos del proyecto.
C. Models
    i. Define las estructuras de los objetos.
D. Routes
    i. Almacena las rutas y endpoints del proyecto y su comportamiento con la base de datos, así como las validaciones respectivas.
E. Schemas
    i. Estructura de los datos de forma legible para MongoDB
F. app.py
    i. Archivo principal de la aplicación que encapsula todo el sistema
G. run.sh
    i. Almacena los comandos necesarios para inicalizar el proyecto

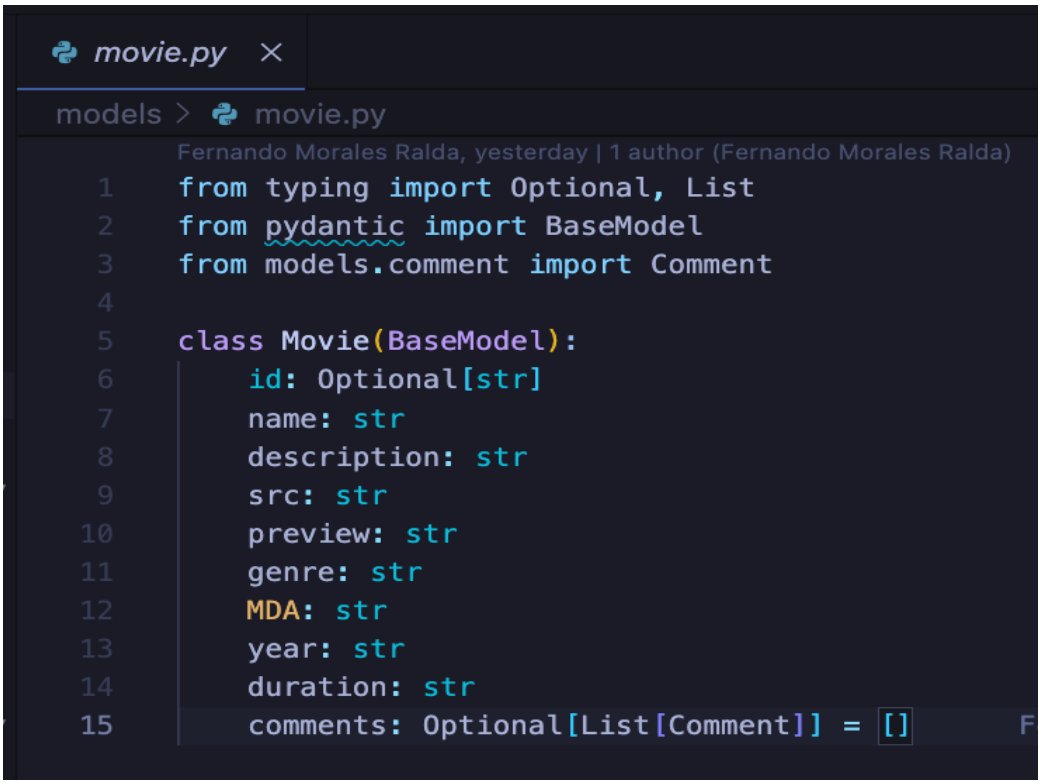
4. INICIALIZAR EL PROYECTO

A. Instalar las librerías necesarias para iniciar el proyecto con Pip3.
B. Iniciar el servicio de mongo para levantar la base de datos.
C. Ejecutar el archivo run.sh para iniciar el sistema.

5. LIBRERÍAS UTILIZADAS

A. Pydantic para validar los datos.
B. Para conversión de los datos.
C. Fast api: framework para la estructura del api.
D. ObjectId: para convertir los id recibidos y enviados a los documentos.
E. Pymongo: para controlar la base de datos.
F. Datetime: para el manejo de fechas

6. ESTRUCTURA DE MODELOS



A. Los id son de tipo opcional para no interferir en la creación de un nuevo documento.

B. Se pueden importar otros modelos de ser necesarios para referenciar otro documento según sea el caso

7. ESTRUCTURA DE ESQUEMAS

A. Se espera como un string el id desde _id del documento.

B. Es recomendable crear una entidad de múltiples para poder ser utilizada directamente en donde sea necesario.

8. ESTRUCTURA DE RUTAS

```python
@movie_router.put("/movies/update/{movie_id}")
def update_movie(movie_id: str, movie: Movie):
    movie_id = ObjectId(movie_id)
    update_fields = {}

    if movie.name is not None:
        update_fields["name"] = movie.name
    if movie.description is not None:
        update_fields["description"] = movie.description
    if movie.genre is not None:
        update_fields["genre"] = movie.genre
    if movie.MDA is not None:
        update_fields["MDA"] = movie.MDA
    if movie.year is not None:
        update_fields["year"] = movie.year
    if movie.duration is not None:
        update_fields["duration"] = movie.duration
    if movie.src is not None:
        update_fields["src"] = movie.src
    if movie.preview is not None:
        update_fields["preview"] = movie.preview

    if update_fields:
        db.movies.update_one({"_id": movie_id}, {"$set": update_fields})

    return movieEntity(db.movies.find_one({"_id": movie_id}))
```

A. Para crear una ruta se debe crear la ruta con los parámetros de ser utilizables, seguido de su función en la cual recibirá los parámetros y el body de la ruta.

B. Si es necesario validar que el parámetro del objeto exista antes de agregarlo al query.

C. Ejecutar el query y preferiblemente devolver como respuesta el documento con su debida modificación.