

MANUAL TÉCNICO PROYECTO 2

BACKEND

202003654

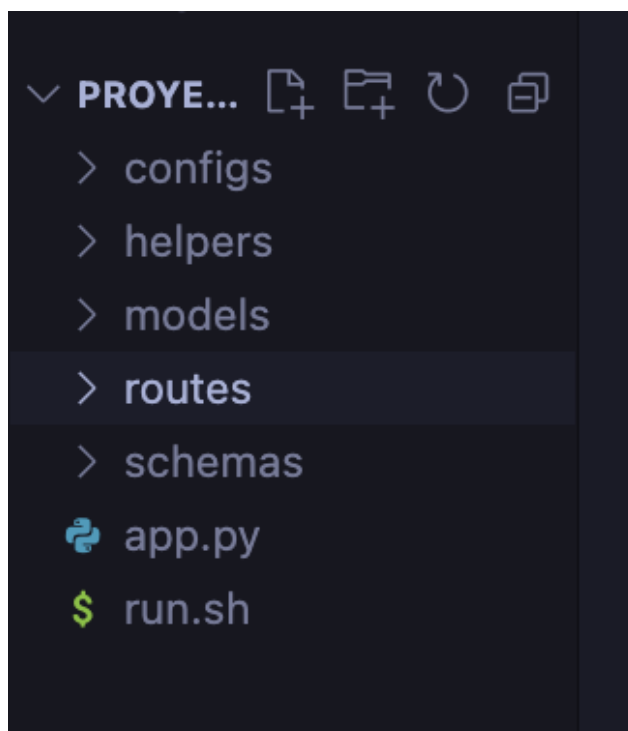
1. DESCRIPCION GENERAL DE LA SOLUCION

- 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 inicializar 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

```
movie.py ×
models > movie.py
Fernando Morales Ralda, yesterday | 1 author (Fernando Morales Ralda)
1  from typing import Optional, List
2  from pydantic import BaseModel
3  from models.comment import Comment
4
5  class Movie(BaseModel):
6      id: Optional[str]
7      name: str
8      description: str
9      src: str
10     preview: str
11     genre: str
12     MDA: str
13     year: str
14     duration: str
15     comments: Optional[List[Comment]] = []
```

- 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

```
movie.py ×
schemas > movie.py
Fernando Morales Ralda, yesterday | 1 author (Fernando Morales Ralda)
1  from schemas.comment import commentsEntity
2
3  def movieEntity(item) -> dict:
4      return {
5          "id": str(item["_id"]),
6          "name": item["name"],
7          "description": item["description"],
8          "src": item["src"],
9          "preview": item["preview"],
10         "genre": item["genre"],
11         "MDA": item["MDA"],
12         "year": item["year"],
13         "duration": item["duration"],
14         "comments": commentsEntity(item.get("comments", []))
15     }
16
17  def moviesEntity(entity) -> list:
18      return [movieEntity(item) for item in entity]
```

- 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

```
81
82 @movie_router.put("/movies/update/{movie_id}")
83 def update_movie(movie_id: str, movie: Movie):
84     movie_id = ObjectId(movie_id)
85     update_fields = {}
86
87     if movie.name is not None:
88         update_fields["name"] = movie.name
89     if movie.description is not None:
90         update_fields["description"] = movie.description
91     if movie.genre is not None:
92         update_fields["genre"] = movie.genre
93     if movie.MDA is not None:
94         update_fields["MDA"] = movie.MDA
95     if movie.year is not None:
96         update_fields["year"] = movie.year
97     if movie.duration is not None:
98         update_fields["duration"] = movie.duration
99     if movie.src is not None:
100         update_fields["src"] = movie.src
101     if movie.preview is not None:
102         update_fields["preview"] = movie.preview
103
104     if update_fields:
105         db.movies.update_one({"_id": movie_id}, {"$set": update_fields})
106
107     return movieEntity(db.movies.find_one({"_id": movie_id}))
108
```

- 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.