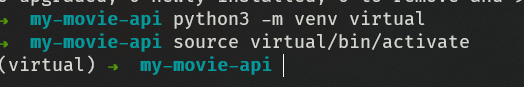
FAST API PATH OPERATIONS, VALIDACIONES Y AUTENTICACION

Instalar entorno virtual de manera global.



Instalar entorno virtual y activarlo.



Instalación de requirements y verificación.



Instalar fastapi



Instalar uvicorn



Actualizar requirement



Ejemplo de desarrollo

Main.py tiene el sgte código.

Importamos fastAPI

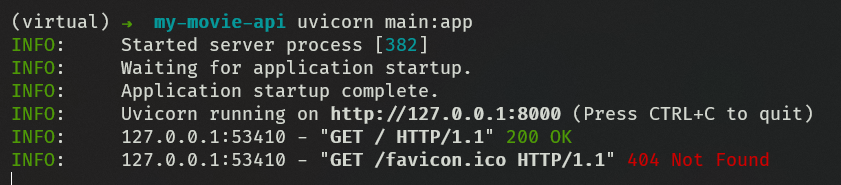
Inicializamos FastAPI y se almacena en la variable app

Se crea la función message que retorna un string

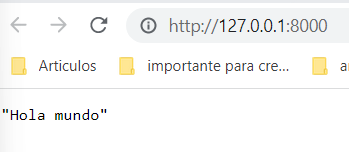
Se agrega un decorador @app y el método get donde se define que para



Para levantar el servidor se utiliza uvicorn, main se refiere al archivo main.py y app se refiere a la inicialización de fastAPI



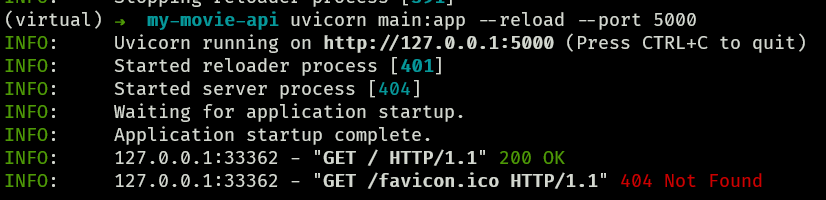
El resultado es el siguiente :



--reload recarga los cambios

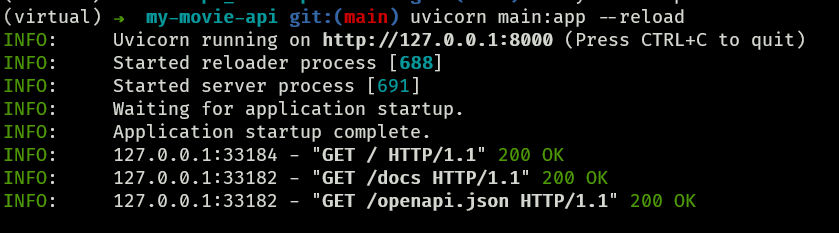


--port selecciona el port que deseas utilizar

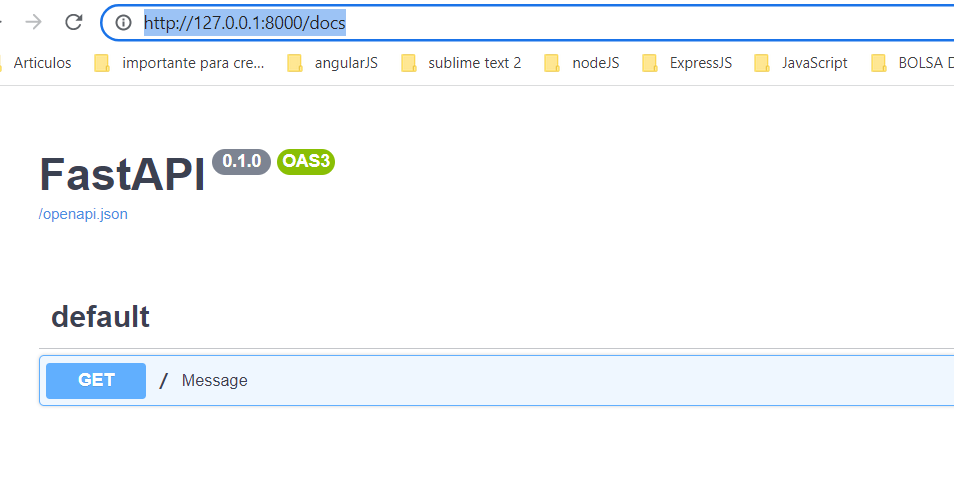


# DOCUMENTACION AUTOMATICA CON SWAGGER

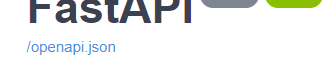
Cuando levantas el proyecto en el servidor uvicorn



Colocando docs al final de la url mostrara la documentación autogenerada swanger



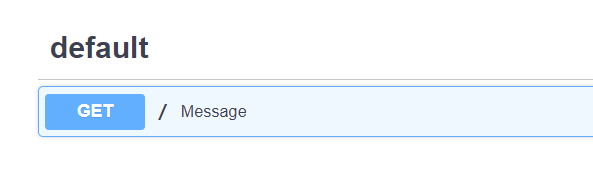
Si le das click a esa url ( openai.json)



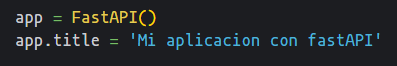
Te mandara este objeto que es toda la data que esta viniendo a la documentacion



Por ahora solo tenemos creado GET



Puedes editar el nombre con title



Se puede cambiar la version



Se puede agregar una nueva etiqueta



# METODOS HTTP

El protocolo HTTP es aquel que define un conjunto de métodos de petición que indican la acción que se desea realizar para un recurso determinado del servidor.

Los principales métodos soportados por HTTP y por ello usados por una API REST son:

POST: crear un recurso nuevo.  
PUT: modificar un recurso existente.  
GET: consultar información de un recurso.  
DELETE: eliminar un recurso.

Como te diste cuenta con estos métodos podemos empezar a crear un CRUD en nuestra aplicación.

¿De qué tratará nuestra API?  
El proyecto que estaremos construyendo a lo largo del curso será una API que nos brindará información relacionada con películas, por lo que tendremos lo siguiente:

Consulta de todas las películas  
Para lograrlo utilizaremos el método GET y solicitaremos todos los datos de nuestras películas.

Filtrado de películas  
También solicitaremos información de películas por su id y por la categoría a la que pertenecen, para ello utilizaremos el método GET y nos ayudaremos de los parámetros de ruta y los parámetros query.

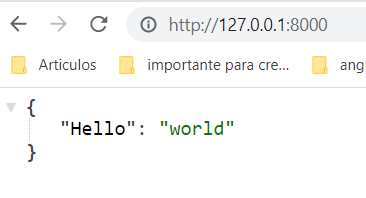
Registro de peliculas  
Usaremos el método POST para registrar los datos de nuestras películas y también nos ayudaremos de los esquemas de la librería pydantic para el manejo de los datos.

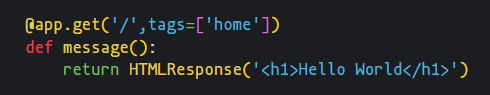
Modificación y eliminación  
Finalmente para completar nuestro CRUD realizaremos la modificación y eliminación de datos en nuestra aplicación, para lo cual usaremos los métodos PUT y DELETE respectivamente.

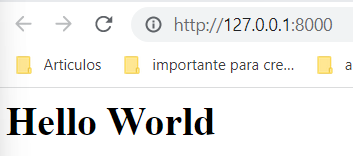
# METODO GET

Con get podemos retornar cualquier otra cosa

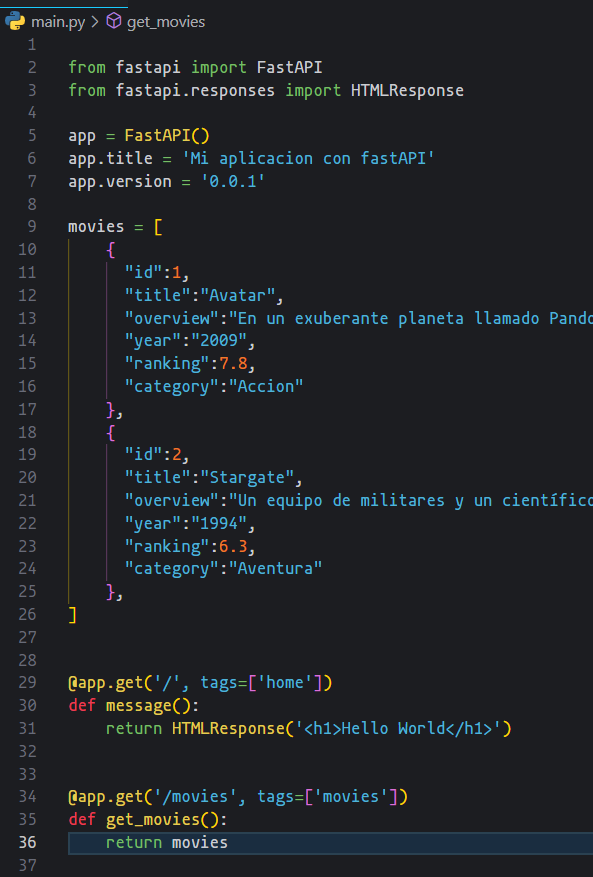




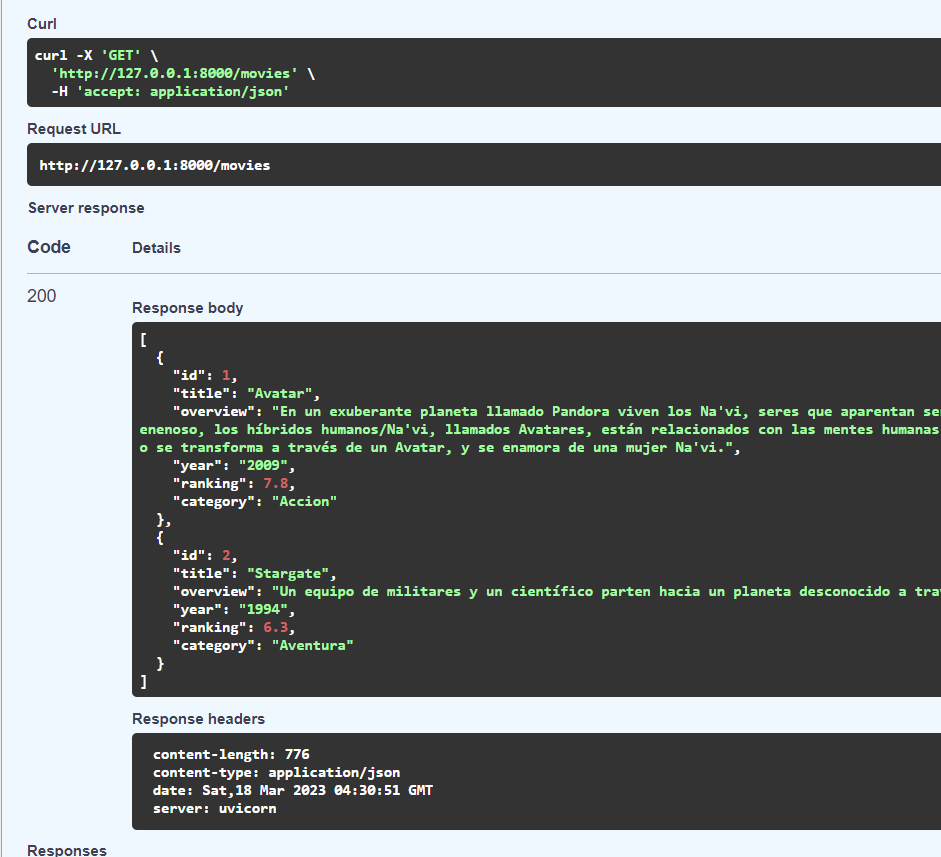




Agregamos una lista con 2 diccionarios, además la función get\_movies() retorna la lista.

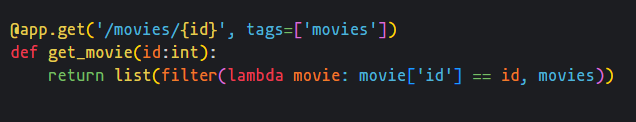


En la documentación de swagger se puede ver lo siguiente:



# PARAMETROS DE RUTA

Capturamos el id de la ruta. La función recibe el parámetro



# PARAMETROS QUERY