**Documentación de Back\_RestAPI**

**Instalación**

Para poder hacer uso de este repositorio tendrás que cumplir los siguientes requerimientos:

* Tener instalado Node JS
* Verificar si está funcionando correctamente el comando npm para instalar las respectivas dependencias.

**Dependencias**

* Comando: npm install express
* URL: [express](https://www.npmjs.com/package/express).
* Comando: npm i bcrypt
* URL: [bcrypt](https://www.npmjs.com/package/bcrypt).
* Comando: npm install body-parser
* URL: [body-parser](https://www.npmjs.com/package/body-parser).
* Comando: npm i cors
* URL: [cors](https://www.npmjs.com/package/cors).
* Comando: npm i dotenv
* URL: [dotenv](https://www.npmjs.com/package/dotenv).
* Comando: npm i jsonwebtoken
* URL: [jsonwebtoken](https://www.npmjs.com/package/jsonwebtoken).
* Comando: npm i mongoose
* URL: [mongoose](https://www.npmjs.com/package/mongoose).
* Comando: npm i mongoose-unique-validator
* URL: [mongoose-unique-validator](https://www.npmjs.com/package/mongoose-unique-validator).
* Comando: npm i nodemon
* URL: [nodemon](https://www.npmjs.com/package/nodemon).
* Comando: npm i passport
* URL: [passport](https://www.npmjs.com/package/passport).
* Comando: npm i express-group-routes
* URL: [passport](https://www.npmjs.com/package/express-group-routes/v/1.1.0).

**¿Como usar la REST API?**

Usar en la consola el comando

* npm start Ahora podrás hacer uso de la API, existen dos maneras de hacer uso de ella.

1. **Opción 1**

* Ingresar en la carpeta Doc en donde podrás encontrar un archivo (Rest API bec.postman\_collection) que tendrás que importar a tu Postman.
* Al tenerlo ya importado solo quedaría configurar el “serverPath” y su configuración es la siguiente.
* VARIABLE: serverPath
* INITIAL VALUE: localhost:8090
* CURRENT VALUE: localhost:8090

1. **Opción 2** Para acceder a los respectivos endpoint deseados tendrás que hacer uso de la siguiente información.

**Health (Nos ayudara a verificar si el servidor está en línea y otros datos)**

* GET localhost:8090/apis/sps/helloworld/v1/health

**Agregar películas**

* POST localhost:8090/apis/sps/helloworld/v1/favorites
* Body tipo raw JSON: { "id": 181812, "original\_title": "Star Wars: The Rise of Skywalker", "title": "Star Wars: The Rise of Skywalker", "overview": "The surviving Resistance faces the First Order once again as the journey of Rey, Finn and Poe Dameron continues. With the power and knowledge of generations behind them, the final battle begins.", "popularity": 64.481, "poster\_path": "/db32LaOibwEliAmSL2jjDF6oDdj.jpg", "release\_date": "2019-12-18", "vote\_average": 6.6, "vote\_count": 5464 }

**Todas las películas**

* GET localhost:8090/apis/sps/helloworld/v1/favorites

**Eliminar una película**

* DELETE localhost:8090/apis/sps/helloworld/v1/favorites/\_id

**Crear una cuenta**

* POST localhost:8090/apis/sps/helloworld/v1/signup
* Body tipo raw JSON: { "email" : "[ejemplo@ejemplo.com](mailto:ejemplo@ejemplo.com)", "password" : "123456" }

**Iniciar sesión**

* POST localhost:8090/apis/sps/helloworld/v1/login
* Body tipo raw JSON: { "email" : "[ejemplo@ejemplo.com](mailto:ejemplo@ejemplo.com)", "password" : "123456" }

**Solicitar información del perfil**

* GET localhost:8090/apis/sps/helloworld/v1/profile?secret\_token=TokenGenerado

**Arquitectura del sistema**

* Descripción jerárquica:

En la siguiente imagen se podrá observar cual es la estructura básica del software.

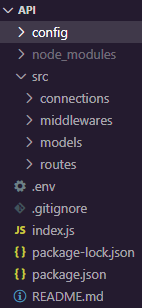


Ilustración 1estructura básica

**Descripción de la estructura jerárquica:**

**Config**: Se encarga de almacenar los archivos que gestiona la configuración de las variables del entorno que se van a utilizar y también se encarga de guardar los endpoints que se están usando.

**SRC**: Dentro de esta carpeta se encuentra todas las carpetas y archivos que se están usando en nuestro servicio. Esta divido en varios módulos.

1. **connections**:

En esta carpeta se van a almacenar todos los archivos que requieren tener una conexión, como a una base de datos.

* + orgMongo.js Guarda y exporta la conexión a nuestra base de datos.

1. **middlewares**
   * auth.js Es un Middleware que nos ayuda a verificar y obtener el token de acceso
2. **models**:

Se almacena todos los modelos que vallamos a usar y además también se almacena los schema de nuestra base de datos.

1. **routes**: Es el encargado de administrar todas las rutas de nuestro servicio.

**Carpeta raíz.**

**.env** En este archivo se definen todas la variables del entorno que se van a usar.

**Index.js** Es nuestro archivo fuente que nos ayuda a cargar todos los módulos de nuestro servicio.

**Package.json** Se guardan las dependencias que se van a usar en el servicio y scripts personalizados.