**API REST gems\_backend**

Antes de iniciar el servidor, diríjase al repositorio creado en GitHub en la rama correspondiente y descargue los archivos **.env.example** y **orgconfig.env.example.** Una vez descargados, colóquelos fuera de la carpeta **src** del proyecto y renombre ambos archivos quitándoles el **.example**.

Los archivos con extensión **.env** sirven para guardar en variables los valores correspondientes a información personal de la maquina del usuario y requerida por la API para funcionar correctamente como por ejemplo: el **username** y **password** para acceder a su cliente de Bases de Datos como **phpmyadmin**.

Instale el programa **XAMPP** server o los programas: **Apache, MySQL, php y phpMyAdmin,** para administrar sus Bases de Datos.

En el archivo .env coloque su usuario y contraseña (si la posee) para ingresar a su cliente de Bases de datos en las variables **USERNAME** y **PASSWORD**. Por defecto el usuario es root y no tiene contraseña. De igual modo, agregue los mismos datos en el archivo ormconfig.env en las variables **TYPEORM\_USERNAME** y **TYPEORM\_PASSWORD.**

Cree una Base de datos con el nombre establecido en la variable **DATABASE** del archivo .env (la variable **TYPEORM\_DATABASE** del archivo ormconfig.env tiene el mismo valor).

Para ejecutar esta API, ingrese el comando:

**npm start**, si es en modo de producción

**npm run start:dev**, si es en modo de desarrollo y pruebas.

Posteriormente a la ejecución del servidor, ejecute en otra terminal el comando para generar una lista de países previamente programados en la API y cargarlos a la Base de Datos:

**npm run schema:drop && npm run seed:run**

1. **Entidades.**

**Usuario:**

* id: identificador autogenerado del Usuario.
* email: correo electrónico único. Es obligatorio.
* password: contraseña. Es obligatorio.
* first\_name: nombre. Es obligatorio.
* last\_name: apellido. Es obligatorio.

1. **Rutas.**

**Usuario:**

* **Método: GET, Ruta: /user**

No recibe valores, devuelve un arreglo formado por todos los registros de usuarios que existan en la Base de Datos.

* **Metodo: GET, Ruta: /user/:id**

Recibe el id del usuario y se devuelve el registro del usuario correspondiente al id que se envió.

* **Metodo: POST, Ruta: /user**

Recibe un objeto formado por los atributos obligatorios del usuario:**, email, password, first\_name, last\_name** .Devuelve un mensaje con el aviso de si pudo o no ser registrado el usuario.

* **Metodo: PATCH, Ruta: /user/:id**

Recibe el id del usuario y un objeto con la lista de atributos que se quieren modificar en el registro del usuario correspondiente. Devuelve un mensaje con el aviso de si se pudo o no realizar la actualización de datos.

* **Metodo: DELETE, Ruta: /user/:id**

Recibe el id del usuario cuyo registro se quiere eliminar. Devuelve un mensaje con el aviso de si se pudo o no eliminar el registro.

**Auth:**

* **Metodo POST, Ruta: /auth/login**

Recibe un objeto con el email y password del usuario. Devuelve un objeto formado por un mensaje de si pudo o no ser exitosa la operación y un token en caso de que los datos sean correctos.

* **Metodo POST, Ruta: /auth/change-password**

Esta ruta solo puede ser accedida por el usuario cuando haya iniciado sesión. Recibe un objeto formado por la contraseña actual y la nueva contraseña. Devuelve un mensaje con el aviso de si pudo o no ser exitosa la operación.