

Manual de ejecución

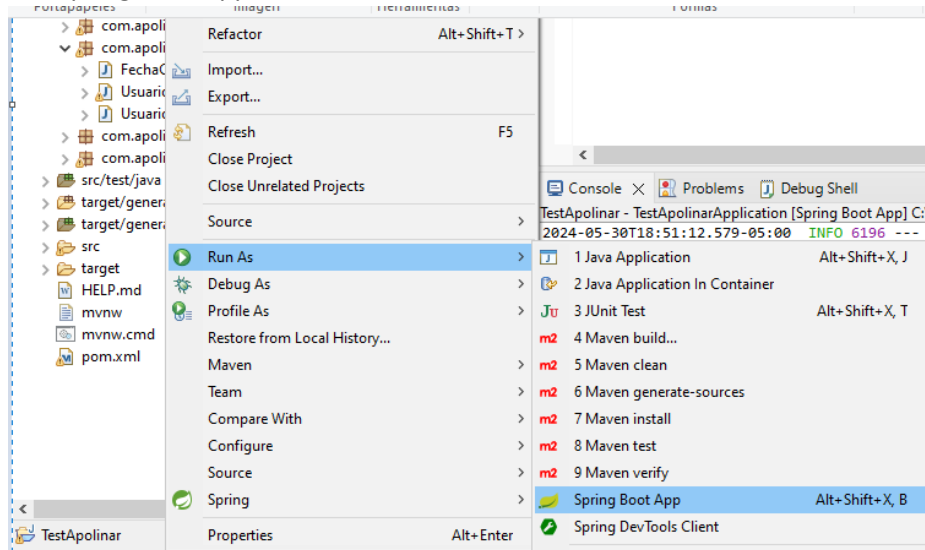
Test

Apolinar Solis Ordoñez

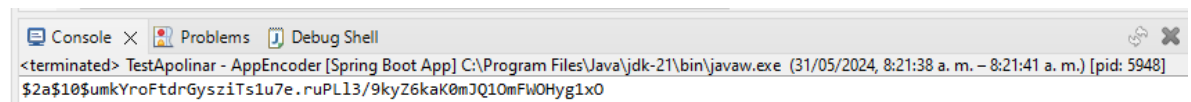
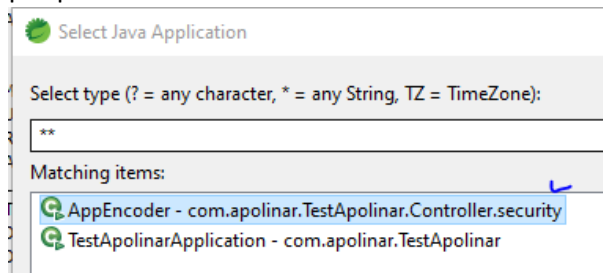
Readme

- Framework: Spring Boot.
- Banco de Datos: H2.
- Proceso de Build: Maven.
- Persistencia: JPA.
- JAVA 8+.
- JWT.

1. Se debe ejecutar el programa para obtener el TOKEN, damos
  - clic derecho sobre el proyecto.
  - Run As
  - Spring Boot App

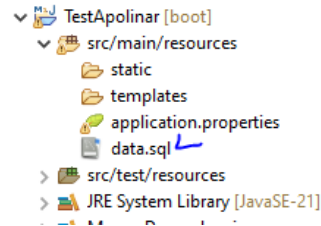


2. Después seleccionamos Controller.security para poder generar nuestro clave de acceso por primera vez.



Con la clave generada la pegamos en el archivo data.sql para que se inserte en la tabla (TBL\_USUARIO) cuando inicie el programa y de esta forma tener acceso al mismo.

Esta es la ubicación del archivo data.sql



```
1 DROP TABLE IF EXISTS TBL_USUARIO;
2 CREATE TABLE TBL_USUARIO(
3     ID INT AUTO_INCREMENT PRIMARY KEY,
4     NOMBRE VARCHAR (50) NOT NULL,
5     USUARIO VARCHAR (50) NOT NULL,
6     CORREO VARCHAR (50) NOT NULL,
7     CLAVE VARCHAR (100) NOT NULL,
8     TELEFONO VARCHAR (50) NOT NULL,
9     ESTADO CHAR (1) NOT NULL,
10    CODIGO_CIUDAD VARCHAR (10) NULL,
11    CODIGO_PAIS VARCHAR (10) NULL,
12    FECHA_CREACION Date not null,
13    FECHA_ULTIMA_ACTUALIZACION date not null,
14    FECHA_ULTIMO_INGRESO date NOT NULL
15
16
17 );
18
19 INSERT INTO TBL_USUARIO(ID, NOMBRE, USUARIO, CORREO, CLAVE, TELEFONO, ESTADO, CODIGO_CIUDAD, CODIGO_PAIS, FECHA_CREACION,
20 VALUES
21 (1, 'APOLINAR', 'asolis', 'apolina@gmail.com', '$2a$10$umkYroFtdrGysziTs1u7e.ruPL13/9kyZ6kaK0mJQ10mFW0Hyg1x0', '31541874
22
23
24
```

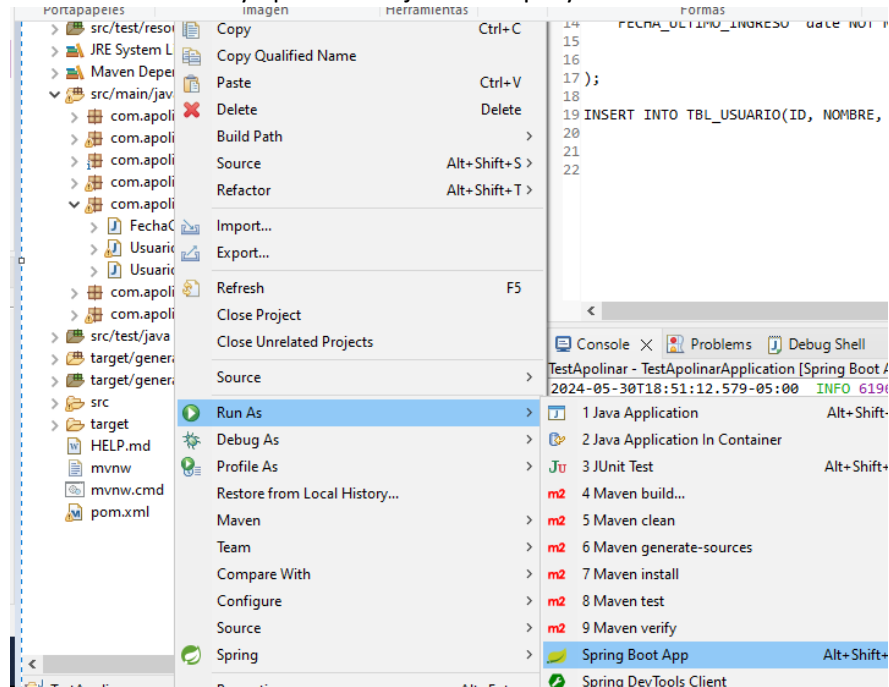
Console × Problems Debug Shell

<terminated> TestApolinar - AppEncoder [Spring Boot App] C:\Program Files\Java\jdk-21\bin\javaw.exe (31/05/2024, 8:21:38 a. m. – 8:21:41 a. m.) [pid: 5948]

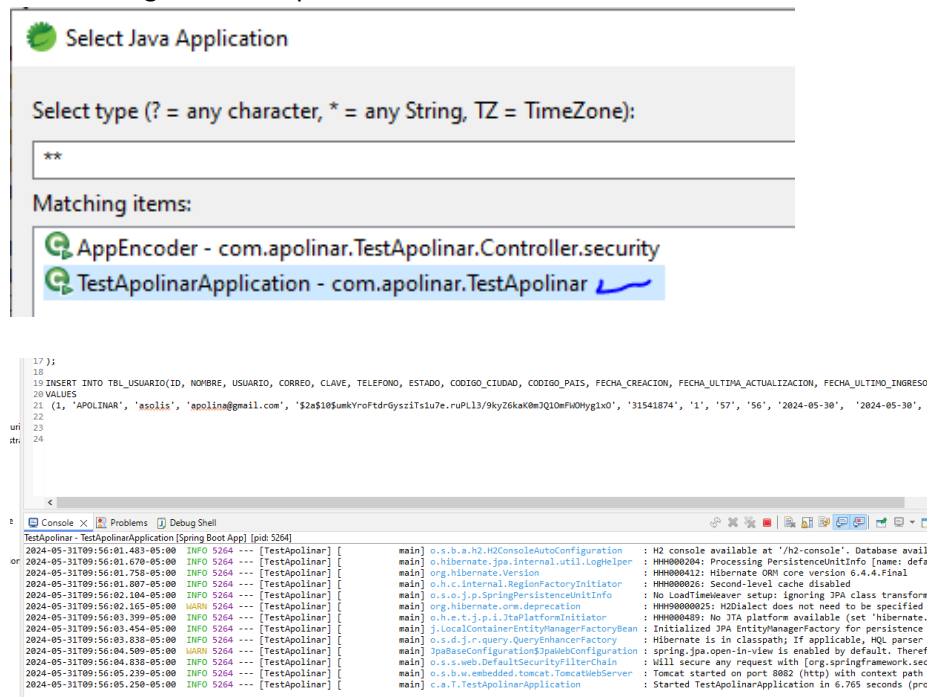
\$2a\$10\$umkYroFtdrGysziTs1u7e.ruPL13/9kyZ6kaK0mJQ10mFW0Hyg1x0

Ahora ya podemos ejecutar nuestro proyecto para que pueda crear la tabla e insertar el registro con nuestra clave de acceso.

Ahora con nuestro ya podemos ejecutar el proyecto

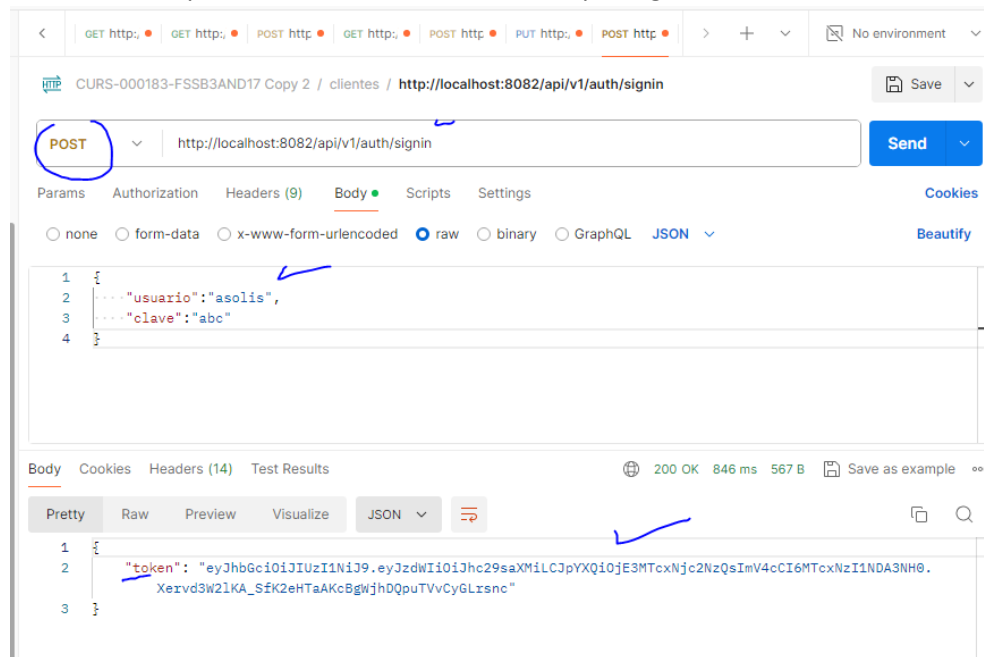


Ahora escogemos TestApolar



Ahora que nuestro proyecto está corriendo con la seguridad de Spring Security podemos realizar la consulta a la BD con Postman de autenticación para que el sistema nos devuelva el token para el acceso.

Procedemos a poner las credenciales de acceso para generada el token



Post → <http://localhost:8082/api/v1/auth/signin>

```
{  "usuario": "asolis",
  "clave": "abc"
}
```



Ya con el Token generado podemos ingresar a nuestro Backend

Ahora en nuestro [Postman](#) podemos pegar el token y realizar la consulta de los usuarios, para realizar la consulta escogemos;

Auth Token → Bearer Token y pegamos el Token como se ilustra en la siguiente imagen.



PUT http://localhost:8082/v1/Usuarios/1

Params Authorization Headers (10) Body Scripts Settings

Auth Type ✓  
Bearer Token

① Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborator, use variables. Learn more about [variables](#).

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Token

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpzZW50L3M4IiwiaXNjaWkiOiJkaW50L3M4IiwiaWF0IjoiMjAyNC01MC0wMCJ9
```

HTTP PUT http://localhost:8082/v1/Usuarios/4

Params Authorization Headers (10) Body Scripts Settings

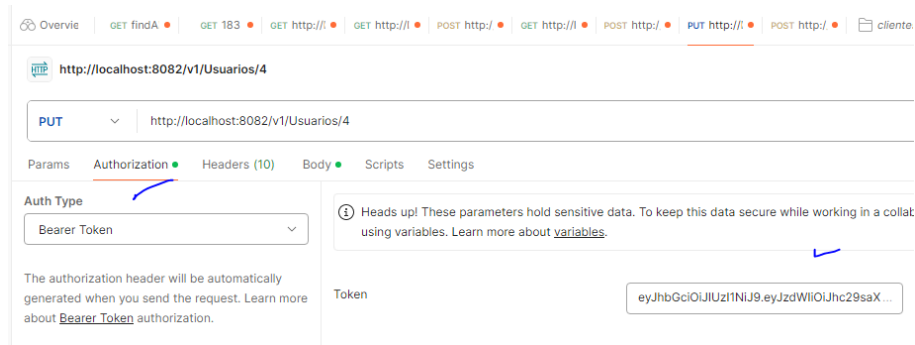
☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ✓

```
1 {  
2   ...  
3   "nombre": "Luis Solis",  
4   "usuario": "lsis",  
5   "correo": "luis@gmail.com",  
6   "clave": "aced123",  
7   "telefono": "310454111",  
8   "estado": "0",  
9   "codigoCiudad": "57",  
10  "codigoPais": "56",  
11  "fechaCreacion": "2024-05-30"  
12 }
```

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize JSON ✓

```
1 {  
2   "id": 4,  
3   "nombre": "Luis Solis",  
4   "usuario": "lsis",  
5   "correo": "luis@gmail.com",  
6   "clave": "aced123",  
7   "telefono": "310454111",  
8   "estado": "0",  
9   "codigoCiudad": "57",  
10  "codigoPais": "56",  
11  "fechaCreacion": "2024-05-29",  
12  "fechaUltimaActualizacion": "2024-05-31T10:36:19.548373",  
13  "fechaUltimoIngreso": "2024-05-31T10:36:19.548373"  
14 }
```



Ahora validaremos si el correo ya fue registrado desde el Post  
POST→ <http://localhost:8082/v1/Usuarios/post>

```
{  
  
  "nombre": "Michael Jackson",  
  "usuario": "mica",  
  "correo": "micha@gmail.com",  
  "clave": "abcde",  
  "telefono": "31541874",  
  "estado": "1",  
  "codigoCiudad": "57",  
  "codigoPais": "56",  
  "fechaCreacion": "2024-05-31"  
}
```



Pruebas / http://localhost:8082/v1/Usuarios

POST http://localhost:8082/v1/Usuarios/post

Params Authorization Headers (10) Body Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   ...
3   ... "nombre": "Olegario",
4   ... "usuario": "oleg",
5   ... "correo": "olegario@gmail.com",
6   ... "clave": "abcde",
7   ... "telefono": "31541874",
8   ... "estado": "1",
9   ... "codigoCiudad": "57",
10  ... "codigoPais": "56",
11  ... "fechaCreacion": "2024-05-31"
12 }
```

Body Cookies Headers (13) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "mensaje": "El correo ya esta Registrado"
3 }
```

El sistema nos valida si el correo ya fue registrado

También podemos ver como registra la fecha de última actualización

PUT→ <http://localhost:8082/v1/Usuarios/1>

Body

```
{
  "nombre": "Faustino",
  "usuario": "faus",
  "correo": "faus@gmail.com",
  "clave": "awsda",
  "telefono": "3104524111",
  "estado": "0",
  "codigoCiudad": "57",
  "codigoPais": "56",
  "fechaCreacion": "2024-05-30"
}
```


PUT ▼ http://localhost:8082/v1/Usuarios/1

Params Authorization ● Headers (10) **Body ●** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **J**

```
1 {
2   "nombre": "Faustino",
3   "usuario": "faus",
4   "correo": "faus@gmail.com",
5   "clave": "awsda",
6   "telefono": "3104524111",
7   "estado": "0",
8   "codigoCiudad": "57",
9   "codigoPais": "56",
10  "fechaCreacion": "2024-05-30"
11 }
```

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize JSON ▼ 

```
1 {
2   "id": 1,
3   "nombre": "Faustino",
4   "usuario": "faus",
5   "correo": "faus@gmail.com",
6   "clave": "awsda",
7   "telefono": "3104524111",
8   "estado": "0",
9   "codigoCiudad": "57",
10  "codigoPais": "56",
11  "fechaCreacion": "2024-05-29",
12  "fechaUltimaActualizacion": "2024-05-31T11:46:19.0280094",
13  "fechaUltimoIngreso": "2024-05-31T11:46:19.0280094"
14 }
```

