INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

CARRERA:

INGENIERÍA EN SISTEMAS COMPUTACIONALES

UNIDAD DE APRENDIZAJE:

INGENIERIA DE SOFTWARE

PROFESOR:

AVILÉS HURTADO GABRIEL

PRACTICA:

PLAN DE PROYECTO CON BASE EN LOS FUNDAMENTOS DE INGENIERÍA DE SOFTWARE

INTEGRANTES:

SANDOVAL GARIBAY SALVADOR 2022630151

FECHA DE ENTREGA: 5/03/25

6CV3

Indice

Introducción

El proyecto consiste en un sistema de inicio de sesión basado en roles, específicamente Administrador y Usuario. Su propósito principal es gestionar el acceso de los usuarios de manera organizada, permitiendo el control de sesiones y preparando el sistema para futuras implementaciones de búsqueda y recomendaciones.

Para el desarrollo se utilizó Java, aprovechando su enfoque de programación orientada a objetos y su robustez en aplicaciones empresariales. Se optó por el framework Quarkus, que ofrece un alto rendimiento y optimización para entornos nativos y contenedorizados. Además, Quarkus facilita la integración con Docker, proporcionando configuraciones predeterminadas para su despliegue en contenedores.

El sistema está diseñado bajo una arquitectura hexagonal, lo que permite una separación clara entre la lógica de negocio y las capas externas, facilitando su mantenimiento, escalabilidad y futuras integraciones. Esta estructura mejora la modularidad del código y la adaptabilidad del sistema a nuevos requisitos.

La base de datos utilizada es MYSQL, ejecutándose en un contenedor de docker. Además, se han incorporado medidas de seguridad como el cifrado de contraseñas con BCrypt y la validación de tokens en cada petición protegida.

Endpoints

Los endpoints ejecutados para esta practica y los mas importantes son:

- /inicio/registrarse
- /inicio/iniciarsesion

Para el primer endpoint utilizamos un método http llamado POST el cual enviaremos como parámetros el nombre, el email y una contraseña. Por detrás del área de la lógica de negocio el email y la contraseña tienen expresiones regulares para conservar un estándar tanto que el correo tenga un @example.com como la contraseña tenga un carácter especial, una mayúscula y minúsculas esta al guardarse primero se codifica para cuidar la seguridad de la cuenta. A su vez cuando nosotros creamos un usuario por predeterminado insertamos el id 2 de roles, que es un rol de usuario.

```java
@Override
public Either<ErrorCodesEnum, Usuario> crearUsuario(Usuario entity) {

    String emailRegex = "^[a-zA-Z0-9]+@[a-z]+\\.[a-z.]{2,6}$";
    String passwordRegex = "^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)(?=.*[!@#$%^&*()_+\\-=\\[\\]{};':\"\\\\|,.<>/?]).{8,}$";

    if (entity.getNombre() == null || entity.getEmail() == null || entity.getPassword() == null) {
        return Either.left(ErrorCodesEnum.RNS001);
    }

    if (!entity.getEmail().matches(emailRegex)) {
        return Either.left(ErrorCodesEnum.RNS002);
    }
    if (!entity.getPassword().matches(passwordRegex)) {
        return Either.left(ErrorCodesEnum.RNS003);
    }

    String hashedPassword = BCrypt.withDefaults().hashToString(12, entity.getPassword().toCharArray());
    entity.setPassword(hashedPassword);

    Optional<Rol> rolUsuario = rolRepository.findById(2);
    if (rolUsuario.isPresent()) {
        entity.setRoles(new ArrayList<>());
        entity.getRoles().add(rolUsuario.get());
    } else {
        return Either.left(ErrorCodesEnum.RNS004);
    }

    return Either.right(usuarioRepository.save(entity));
}
```

Para el segundo endpoint justamente interactuamos con la base de datos para la obtención de a contraseña y desencriptarla y verificar que el usuario que se ingreso mediante el correo electrónico, sea el mismo que persiste en la base de datos, algunas verificaciones como tal son en caso de que el usuario no ingrese datos o no se encuentren los datos. Al momento de que la validación sea correcta nosotros generamos un token JWT el cual solo registra el id del usuario y su rol para así dar paso en la asignación de la siguiente url.

```java
@Override
public Either<ErrorCodesEnum, String> inicioSesion(Usuario entity) throws IOException, InvalidKeySpecException, NoSuchAlgorithmException {
    if (entity.getEmail() == null || entity.getPassword() == null) {
        return Either.left(ErrorCodesEnum.RNS001);
    }
    var usuario = buscarUsuarioPorCorreo(entity.getEmail());
    if (usuario == null) {
        return Either.left(ErrorCodesEnum.RNS005);
    }
    boolean contrasenaValida = BCrypt.verifyer().verify(entity.getPassword().toCharArray(), usuario.getPassword()).verified;
    if (!contrasenaValida) {
        return Either.left(ErrorCodesEnum.RNS007);
    }

    String token = generarTokenJWT(usuario);
    return Either.right(token);
}
```

Todos los manejos de errores bajo la denominación “regla de negocio del sistema” o “RNS” dirigen un flujo para una mayor visualización acerca de los problemas que llegaran a suscitarse eso combinado con “Eithers” que no son nada mas que tipo “TRY CATCH” que ajustamos la fluidez del sistema dependiendo el caso en donde nos encontremos.

```java
package mx.com.tarea3.util.error;

public enum ErrorCodesEnum {

    RNS001("Campo Obligatorio"),
    RNS002("Formato de correo no valido con el formato"),
    RNS003("Formato de contraseña no valido con el formato"),
    RNS004("Rol no encontrado"),
    RNS005("Usuario no encontrado"),
    RNS006("Correo no encontrado"),
    RNS007("Contraseña no valida"),

    RNN001("Unicidad de elementos"),
    ERROR("Error inesperado"),

    INVALID_LINK("Liga no vigente"),
    BAD_REQUEST("Error en la petición"),
    NOT_FOUND("Recurso no encontrado"),
    NEW_LINK("Nueva liga para registrar contraseña"),
    CAPA_PERSISTENCIA("Error en la capa de persistencia"),
    ERROR_EN_COMUNICACIONES("Error en la capa de comunicaciones");

    private final String detail;

    ErrorCodesEnum(String detail) {
        this.detail = detail;
    }

    public String getName() {
        return this.name();
    }

    public String getDetail() {
        return detail;
    }

}
```

Para los controladores, están divididos en 2 archivos, uno donde se tiene todas las funcionalidades de el administrador y otro donde es inicio de sesión y la vista de los usuarios.

```java
@Path("/inicio")
@RolesAllowed("ROLE_ADMIN")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
@Tag(name = "Operaciones del administrador")
public class AdministradorController {

    @Inject
    UsuarioService usuarioService;
    @GET
    @Path("/admin/usuarios")
    public Response listarUsuarios() {
        var usuarios= usuarioService.listarUsuarios().stream().map(UsuarioDto::fromEntity).collect(Collectors.toList());
        return Response.ok(usuarios).build();
    }
    @Path("/editar/{id}")
    @PUT
    public Response editar(@PathParam("id") Integer id, UsuarioDto usuarioDto) {
        return usuarioService.actualizarUsuario(id,usuarioDto.toEntity())
                .map(UsuarioDto::fromEntity).map(Response::ok).getOrElseGet(ErrorMapper::errorCodeToResponseBuilder).build();
    }

    @Path("/eliminar/{id}")
    @DELETE
    public Response eliminar(@PathParam("id") Integer id) {
        return usuarioService.eliminarUsuario(id).map(Response::ok).getOrElseGet(ErrorMapper::errorCodeToResponseBuilder).build();
    }

    @POST
    @Path("/registrarUsuario")
    public Response registrarUsuario(@Valid UsuarioDto usuarioDto) {
        return usuarioService.crearUsuario(usuarioDto.toEntity()).map(UsuarioDto::fromEntity).map(Response::ok)
                .getOrElseGet(ErrorMapper::errorCodeToResponseBuilder).build();

    }

}
```

```java
@Path("/inicio")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
@Tag(name = "Registro,inicio de sesion y vista de los de los usuarios")
public class RegistroController {
    @Inject
    UsuarioService usuarioService;

    @POST
    @Path("/registrarse")
    public Response registrarUsuario(@Valid UsuarioDto usuarioDto) {
        return usuarioService.crearUsuario(usuarioDto.toEntity()).map(UsuarioDto::fromEntity).map(Response::ok)
                .getOrElseGet(ErrorMapper::errorCodeToResponseBuilder).build();

    }

    @POST
    @Path("/iniciarsesion")
    public Response iniciarSesion(@Valid InicioSesionDto inicioSesionDto) throws IOException, InvalidKeySpecException, NoSuchAlgorithmException {
        var resultado = usuarioService.inicioSesion(inicioSesionDto.toEntity());

        return resultado.fold(
                error -> Response.status(Response.Status.UNAUTHORIZED)
                        .entity(error.getName())
                        .build(),
                token -> Response.ok(token).build()
        );
    }

    @GET
    @Path("/usuario/perfil")
    @RolesAllowed({"ROLE_USER", "ROLE_ADMIN"})
    public Response getUsuarioPerfil(@Context SecurityContext securityContext) {
        String userId = securityContext.getUserPrincipal().getName();
        Optional<Usuario> usuarioOpt = usuarioService.getUsuarioById(Integer.parseInt(userId));
        if (usuarioOpt.isEmpty()) {
            return Response.status(Response.Status.NOT_FOUND).build();
        }

        return Response.ok(usuarioOpt.get()).build();
    }
```

Prueba entorno local

Para las pruebas de entorno local, solo basta con ejecutar el Dockerfile y el docker compose, montando **docker build -t tarea3 .** Y por consiguiente **docker compose up** , en general **se debe de tener libres los puertos 3306 y el 8080 para la prueba local**, por predeterminado las configuraciones del sistema empezaran a levantarse, solo quedara acceder a la url **http://localhost:8080/inicio/login#** para poder inicializar el programa, de igual forma el administrador registrador por predeterminado es : **ssandovalgaribay@gmail.com** Y **Salvador@123** **.**

Manejo de sesiones

En el área de manejo de sesiones, primero en el entorno del back se utilizo el framework quarkus, en donde este una de sus opciones muy parecidas a spring boot es el uso de @RolesAllowed el cual mediante el token que se genera al iniciar sesión se descompone para obtener al usuario por su id y así saber que opciones tendrá por su token, de igual forma en el front cuando se decodifica se abre la opción dependiendo su usuario si es admin o usuario.

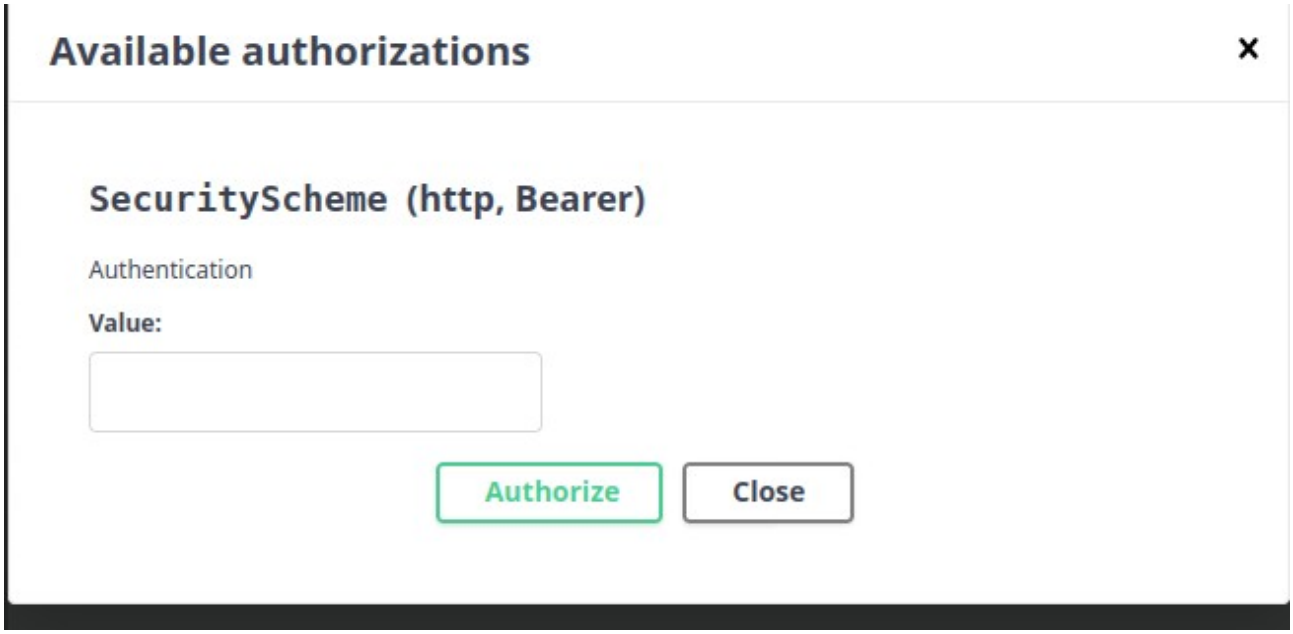
```
if (roles && roles.includes('ROLE_ADMIN')) {
    window.location.href = 'http://localhost:8080/inicio/admin';
} else {
    window.location.href = 'http://localhost:8080/inicio/usuario';
}
```

Funcionamiento endpoints

Para probar estos endpoints, basta con entrar a la url http://localhost:8080/q/swagger-ui/#/ , en esta url nos da la vista de los endpoints de los cuales estuvimos hablando, en cierta forma si desea probar algunas operaciones del administrador ingrese con el usuario que se dejo en el área de **prueba entorno local ,** así como visualizar el perfil de la persona con la cual inicio sesión. Deberá meter el token que se proporcione en el área que dice "**Authorize"** para poder tener acceso sea el caso de admin o usuario.

Obteniendo eso, solo basta con ejecutar cada uno de los endpoints para su visualización, en el caso de los últimos controladores llamados **"Renderizacion html's"** no van a funcionar, debido a que estos son solo plantillas html que son usadas para la visualización del login y las demás funcionalidades del sistema.


Operaciones del administrador
GET /inicio/admin/usuarios get_inicio_admin_usuarios
PUT /inicio/editar/{id} put_inicio_editar__id_
DELETE /inicio/eliminar/{id} delete_inicio_eliminar__id_
POST /inicio/registrarUsuario post_inicio_registrarUsuario
Registro,inicio de sesion y vista de los de los usuarios
POST /inicio/iniciarsesion post_inicio_iniciarsesion
POST /inicio/registrarse post_inicio_registrarse
GET /inicio/usuario/perfil get_inicio_usuario_perfil
Renderización de html's
GET /inicio/admin get_inicio_admin
GET /inicio/admin/lista-usuarios get_inicio_admin_lista_usuarios
GET /inicio/admin/registrar get_inicio_admin_registrar
GET /inicio/login get_inicio_login
GET /inicio/usuario get_inicio_usuario


Available authorizations
SecurityScheme (http, Bearer)
Authentication
Value:
Authorize
Close

POST /inicio/iniciarsesion post_inicio_iniciarsesion

Parameters Cancel Reset

No parameters

Request body application/json

```
{
  "email": "juan@gmail.com",
  "password": "Juan@123"
}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8080/inicio/iniciarsesion' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "email": "juan@gmail.com",
  "password": "Juan@123"
}'
```

Request URL

```
http://localhost:8080/inicio/iniciarsesion
```

Server response

Code	Details
200	Response body

```
can't parse JSON.  Raw result:

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJ0dV9hcGxpY2FjaW9uIiwic3ViIjoiMiIsImdyb3VwcyI6WyJST0xFX1VTRVIiXSwiZXhwIjoxNzQwODE0ODQ5LCJpYXQiOjE3NDA4MTEyNDksImp0aSI6IjExZDg3ZTI4LTQxZWQtNDE2Zi04NjI1LTQxMGM5ZmMxYjJlYyJ9.aPq22yAuzZbrnjBFXJAG5PmeltRffg13UyRfzPWKdHgdxp0Hgp7IxNpIP3s2tdYzsftR3CbSKSFAPlmqN5AXyreSRFwP3U7DKkHhwZ-hQSnCmzEr9vZZKYZ2t_O31XG8eoxLw0TuGRhpUs9w--8U0UaSWe108zz4s6yHGuvCojZrorZt0wnoSAojowArHjSgffEBVTkb4dN4Stcu5r6id5DDPW80vwAlYkIHsUXB3IAiVjK591ZOh3CVWBzzyBl44DUzx3_WyGJukt49ymdWzLEDgAzHws8ZnYjKZeDa...
v0TsZWiH6NtEfhi88Hhd0TLqWKnbQ
```

Download

GET /inicio/usuario/perfil get_inicio_usuario_perfil

Parameters Cancel

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/inicio/usuario/perfil' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJ0dV9hcGxpY2FjaW9uIiwic3ViIjoiMiIsImdyb3VwcyI6WyJST0xFX1VTRVIiXSwiZXhwIjoxNzQwODE0ODQ5LCJpYXQiOjE3NDA4MTEyNDksI
```

Request URL

```
http://localhost:8080/inicio/usuario/perfil
```

Server response

Code	Details
200	Response body

```
{
  "id": null,
  "nombre": "Juan Antonio Marquez",
  "email": "juan@gmail.com",
  "password": "$2a$12$p3xfkxtHfg0VrwBdRi7zGONIP2G0ZLn3Uc.L6t8f3yT6OrvP.bchS",
  "roles": null
}
```

Download

Response headers

```
content-length: 155
content-type: application/json
```

Responses

Code	Description	Links
200	OK	*No links*
401		*No links*

GET /inicio/admin/usuarios get_inicio_admin_usuarios

Parameters Cancel

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/inicio/admin/usuarios' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJ0dV9hcGxpY2FjaW9uIiwic3ViIjoiMSIsImdyb3VwcyI6WyJST0xFX0FETUlOIl0sImV4cCI6MTc0MDgxNTAyNSwiaWF0IjoxNzQwODExNDI1L
```

Request URL

```
http://localhost:8080/inicio/admin/usuarios
```

Server response

Code	Details
200	Response body

```
[
  {
    "id": 1,
    "nombre": "Salvador Sandoval Garibay",
    "email": "ssandovalgaribay@gmail.com",
    "password": "$2a$12$J6Fo0kIvQnRs7I97Xm5HSeGQ8gUWPxdexvsJC40PTwvIZc2tK8ZKm",
    "roles": [
      {
        "id": 1,
        "nombre": "ROLE_ADMIN"
      }
    ]
  },
  {
    "id": 2,
    "nombre": "Juan Antonio Marquez",
    "email": "juan@gmail.com",
    "password": "$2a$12$p3xfkxtHfg0VrwBdRi7zGONIP2G0ZLn3Uc.L6t8f3yT6OrvP.bchS",
    "roles": [
      {
        "id": 2,
        "nombre": "ROLE_USER"
      }
    ]
  }
]
```

Download

Response headers

PUT /inicio/editar/{id} put_inicio_editar__id_

Parameters Cancel Reset

Name	Description
id * required integer($int32) (path)	2

Request body application/json

```
{
  "nombre": "Jose Mendoza",
  "email": "jose@gmail.com",
  "password": "Jose@123"
}
```

Execute Clear

Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:8080/inicio/editar/2' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJ0dV9hcGxpY2FjaW9uIiwic3ViIjoiMSIsImdyb3VwcyI6WyJST0xFX0FETUlOIl0sImV4cCI6MTc0MDgxNTAyNSwiaWF0IjoxNzQwODExNDI1LCJqd
  -H 'Content-Type: application/json' \
  -d '{
  "nombre": "Jose Mendoza",
  "email": "jose@gmail.com",
  "password": "Jose@123"
}'
```

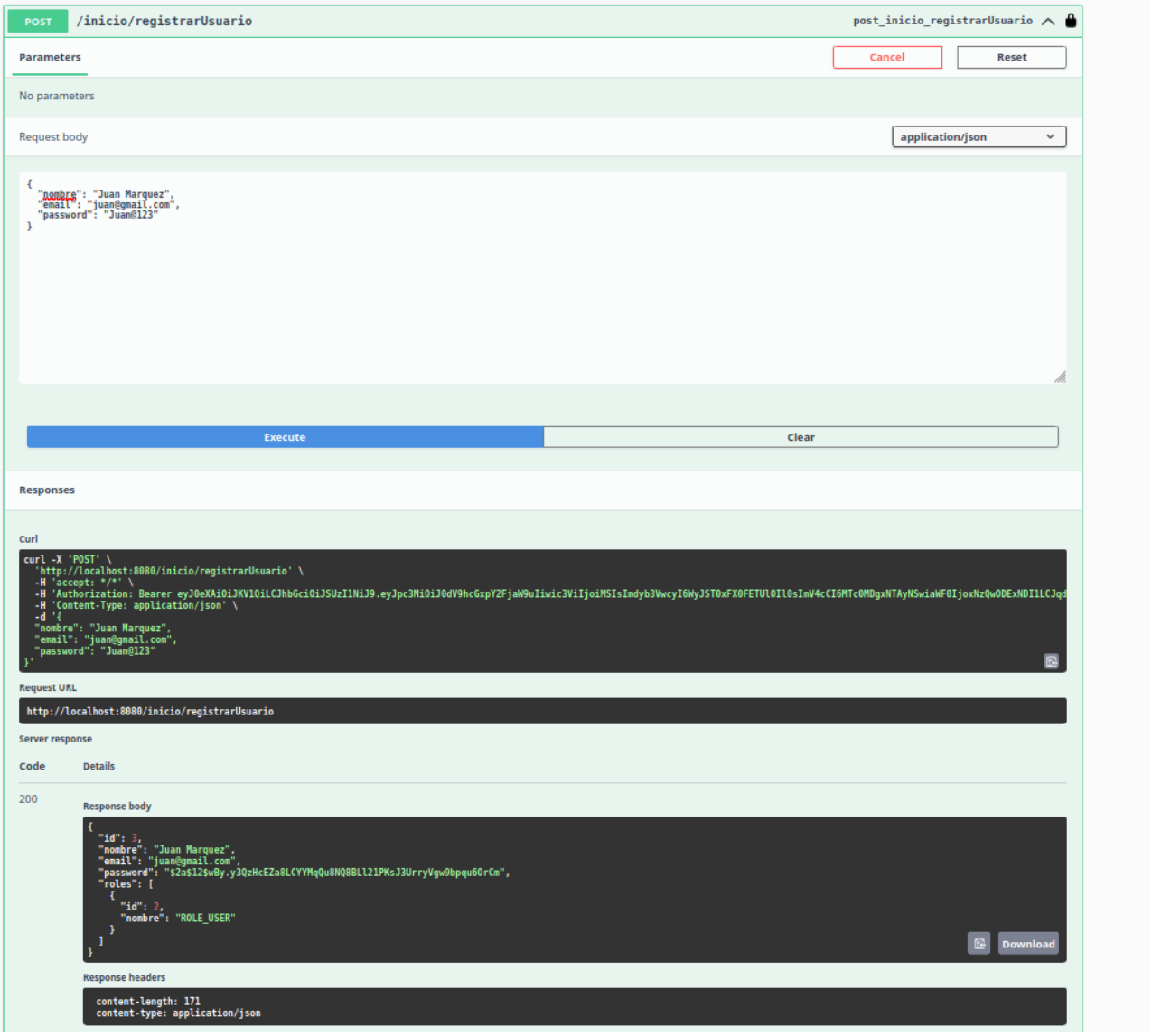
Request URL

```
http://localhost:8080/inicio/editar/2
```

Server response

Code	Details
200	Response body

```
{
  "id": 2,
  "nombre": "Jose Mendoza",
  "email": "jose@gmail.com",
  "password": "$2a$12$eAdlCAeuAvyWptHZlCMP2OSp0lhjrrPuKupePl15Z6cGcukjXXw9e",
  "roles": [
    {
      "id": 2,
      "nombre": "ROLE_USER"
    }
```


POST
/inicio/registrarUsuario
post_inicio_registrarUsuario
Parameters
Cancel
Reset
No parameters
Request body
application/json
{
"nombre": "Juan Marquez",
"email": "juan@gmail.com",
"password": "Juan@123"
}
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://localhost:8080/inicio/registrarUsuario' \
-H 'accept: */*' \
-H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJ0dV9hcGxpY2FjaW9uIiwic3ViIjoiMSIsImdyb3VwcyI6WyJST0xFX0FETUlOIl0sImV4cCI6MTc0MDgxNTAyNSwiaWF0IjoxNzQwODExNDI1LCJqd
-H 'Content-Type: application/json' \
-d '{
"nombre": "Juan Marquez",
"email": "juan@gmail.com",
"password": "Juan@123"
}'
Request URL
http://localhost:8080/inicio/registrarUsuario
Server response
Code
Details
200
Response body
{
"id": 3,
"nombre": "Juan Marquez",
"email": "juan@gmail.com",
"password": "$2a$12$wBy.y3QzHcEZa8LCYYMqQu8NQ8BLl21PKsJ3UrryVgw9bpqu6OrCm",
"roles": [
{
"id": 2,
"nombre": "ROLE_USER"
}
]
}
Download
Response headers
content-length: 171
content-type: application/json