

Diseño y Construcción de Microservicios

LAB Microservicio de Gestión de Users

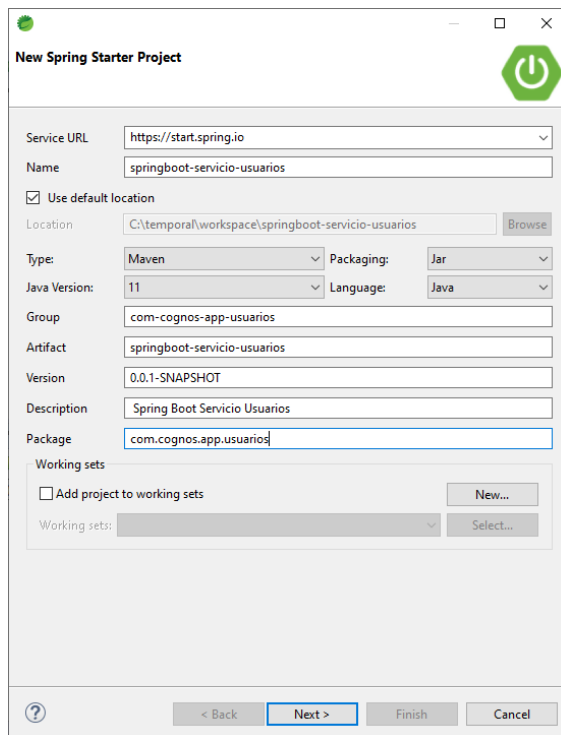
Objetivos

- Mostrar al participante el procedimiento para la creación del Microservicio de Gestión de Credenciales de Usuario.

Procedimiento

Creación de la Entidades

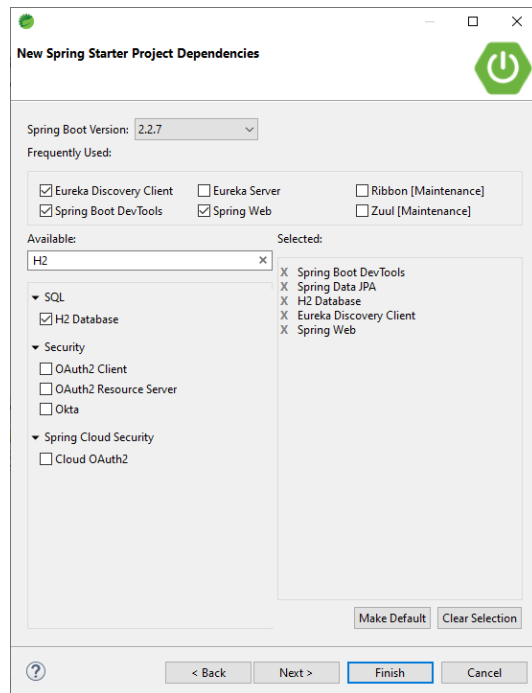
1. Crea el proyecto **springboot-servicio-usuarios**. Agrega las dependencias **Spring Boot Dev Tools**, **Spring Data JPA**, **H2 Database**, **Eureka Discovery Client** y **Spring Web**.



The screenshot shows the 'New Spring Starter Project' dialog box. The 'Service URL' is set to 'https://start.spring.io'. The 'Name' is 'springboot-servicio-usuarios'. The 'Location' is 'C:\temporal\workspace\springboot-servicio-usuarios'. The 'Type' is 'Maven', 'Packaging' is 'Jar', 'Java Version' is '11', and 'Language' is 'Java'. The 'Group' is 'com-cognos-app-usuarios', 'Artifact' is 'springboot-servicio-usuarios', 'Version' is '0.0.1-SNAPSHOT', and 'Description' is 'Spring Boot Servicio Usuarios'. The 'Package' is 'com.cognos.app.usuarios'. The 'Add project to working sets' checkbox is unchecked. The 'Next >' button is highlighted.

Agrega las dependencias indicadas.

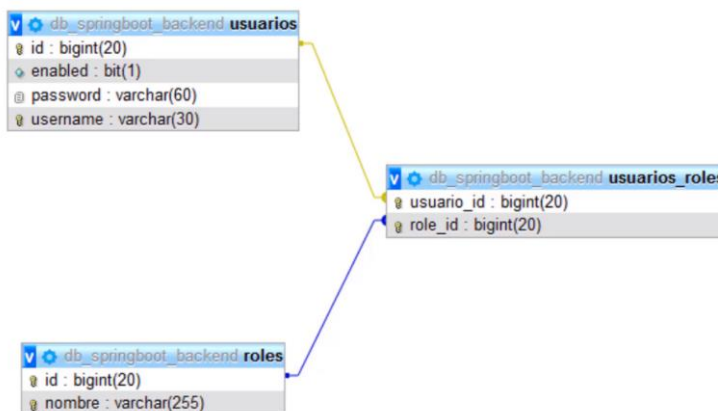
- **Spring Boot Dev Tools**
- **Spring Data JPA**,
- **H2 Database**
- **Eureka Discovery Client**
- **Spring Web**.



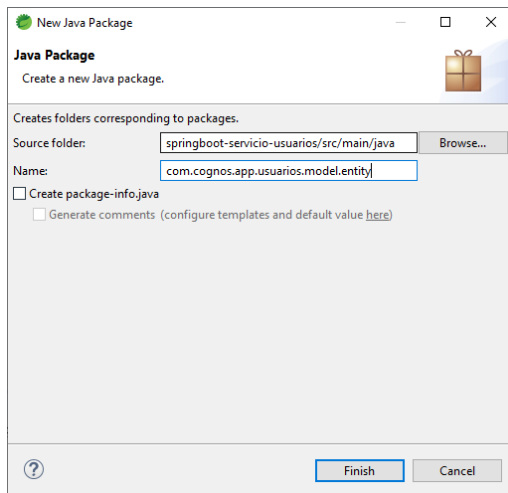
2. Edita el archivo **application.properties** del proyecto.

```
spring.application.name=servicio-usuarios
# Las siguientes lineas hacen que la asignacion del puerto del servicio sea
dinamica
server.port=${PORT:0}
eureka.instance.instance-
id=${spring.application.name}:${spring.application.instance_id:${random.value}}
# URL del servidor de registro
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
# Nivel de log en Debug para depuracion de errores.
logging.level.org.hibernate.SQL=debug
```

3. Creación de las entidades del modelo de datos.



- a) Crea el paquete **com.cognos.app.usuarios.model.entity** para las entidades.



- b) Crea la clase **Usuario** en el paquete **com.cognos.app.usuarios.model.entity**

```
package com.cognos.app.usuarios.model.entity;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "usuarios")
public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true, length = 20)
    private String username;
    @Column(length = 60)
    private String password;
    private Boolean enabled;
    private String nombre;
    private String apellido;
    @Column(unique = true, length = 64)
    private String email;

    public Long getId() {
        return id;
    }
}
```

```
    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public Boolean getEnabled() {
        return enabled;
    }

    public void setEnabled(Boolean enabled) {
        this.enabled = enabled;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

c) Crea la clase **Role** en el paquete `com.cognos.app.usuarios.model.entity`

```
package com.cognos.app.usuarios.model.entity;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "roles")
public class Role implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true, length = 30)
    private String nombre;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
}
```

d) Creando las relaciones entre Usuario y Role. En la clase **Usuario.java** agrega la siguiente relación.

```
...
@ManyToMany(fetch = FetchType.LAZY)
private List<Role> roles;
```

```

public List<Role> getRoles() {
    return roles;
}

public void setRoles(List<Role> roles) {
    this.roles = roles;
}

```

- e) Verificación de creación de tablas. Inicia los servicios primero Eurka server luego servicio-usuarios. En la consola verifica que se ejecutan los comandos DDL de creación de tablas.

```

springboot-servicio-usuarios - SpringbootServicioUsuariosApplication [Spring Boot App] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (16 may. 2020 19:52:46)
[ted@main] org.hibernate.Version : HHH0000412: Hibernate Core (5.3.17.Final)
[ted@main] org.hibernate.cfg.Environment : HHH000206: hibernate.properties not found
[ted@main] org.hibernate.annotations.common.Version : HCAN000001: Hibernate Commons Annotations (5.0.4.Final)
[ted@main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.H2Dialect
[ted@main] org.hibernate.SQL : drop table roles if exists
[ted@main] org.hibernate.SQL : drop table usuarios if exists
[ted@main] org.hibernate.SQL : drop table usuarios_roles if exists
[ted@main] org.hibernate.SQL : create table roles (id bigint generated by default as identity, nombre varchar(30), primary key (id))
[ted@main] org.hibernate.SQL : create table usuarios (id bigint generated by default as identity, apellido varchar(255), email varchar(64), enabled boolean, nom
[ted@main] org.hibernate.SQL : create table usuarios_roles (usuario_id bigint not null, roles_id bigint not null)
[ted@main] org.hibernate.SQL : alter table roles add constraint UK_ldv0v52e0udsh2hrs0r0gw1n unique (nombre)
[ted@main] org.hibernate.SQL : alter table usuarios add constraint UK_kfsg0s1t1mlcwlj8ldhqsad0 unique (email)
[ted@main] org.hibernate.SQL : alter table usuarios add constraint UK_m2dvbvfge29leumw6vkkocao unique (username)
[ted@main] org.hibernate.SQL : alter table usuarios_roles add constraint FK3say4n5o70d831842ncv2156x foreign key (roles_id) references roles
[ted@main] org.hibernate.SQL : alter table usuarios_roles add constraint FKqcxu02bq1pxr7cjy9dmhwec foreign key (usuario_id) references usuarios

```

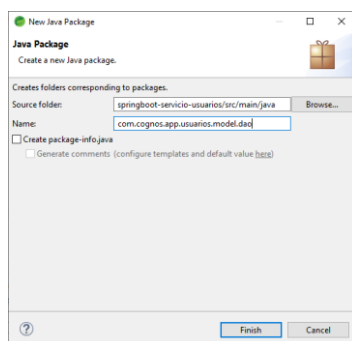
- f) Personaliza la tabla de la relación. Agrega a la clase **Usuario.java** la anotación **@JoinTable**. Reinicia el servicio servicio-usuarios para que recree las tablas.

```

...
@ManyToMany(fetch = FetchType.LAZY)
@JoinTable(name="usuarios_roles",
    joinColumns=@JoinColumn(name="usuario_id"),
    inverseJoinColumns = @JoinColumn(name="role_id"),
    uniqueConstraints= {@UniqueConstraint(columnNames=
        {"usuario_id", "role_id"})})
private List<Role> roles;
...

```

4. Creación de los métodos CRUD para la gestión de los usuarios. Crea el paquete **com.cognos.app.model.dao**.



5. En el paquete **com.cognos.app.model.dao**, crea la interface **UsuarioDao.java**.

```
package com.cognos.app.usuarios.model.dao;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.PagingAndSortingRepository;

import com.cognos.app.usuarios.model.entity.Usuario;

public interface UsuarioDao extends PagingAndSortingRepository<Usuario, Long> {

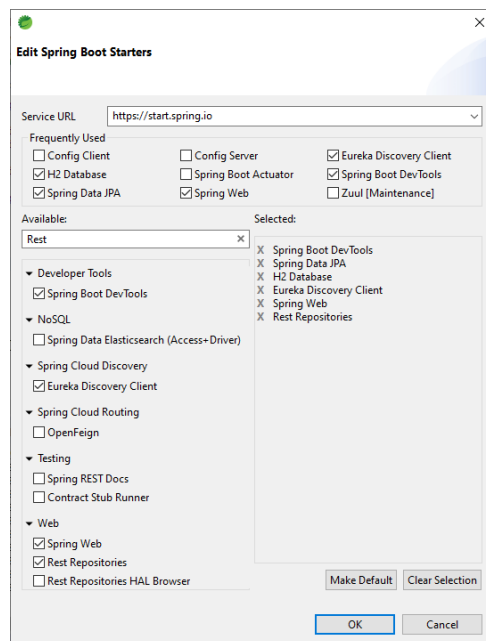
    // Usando palabras clave
    public Usuario findByUsername(String username);

    // JPA-QL
    @Query("Select u From Usuario u where u.username=?1")
    public Usuario obtenerPorUsername(String username);

    // Tambien se puede usar query nativos
    // @Query(value = "SELECT * FROM USERS WHERE EMAIL_ADDRESS = ?1",
    nativeQuery = true)
}
```

Exportando el CRUD al API Rest

6. Agrega la dependencia “**Rest Repositories**” al proyecto servicio-usuarios.



Verifica en el archivo pom.xml del proyecto que se haya agregado la siguiente dependencia.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
```

7. Agrega la anotación **@RepositoryRestResource** a la clase **UsuarioDao.java**

```
package com.cognos.app.usuarios.model.dao;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

import com.cognos.app.usuarios.model.entity.Usuario;

@RepositoryRestResource(path = "usuarios")
public interface UsuarioDao extends PagingAndSortingRepository<Usuario, Long> {

    // Usando palabras clave ver Spring Data JPA Reference
    public Usuario findByUsername(String username);

    // JPA-QL
    @Query("Select u From Usuario u where u.username=?1")
    public Usuario obtenerPorUsername(String username);

    // Tambien se puede usar query nativos
    // @Query(value = "SELECT * FROM USERS WHERE EMAIL_ADDRESS = ?1",
    nativeQuery =
        // true)
}
```

8. Actualiza el archivo **application.properties** del proyecto **servicio-zuul-server** o servidor Zuul.

```
spring.application.name=servicio-zuul-server
server.port=8090

eureka.client.service-url.defaultZone=http://localhost:8761/eureka

zuul.routes.usuarios.service-id=servicio-usuarios
zuul.routes.usuarios.path=/api/usuarios/**

zuul.routes.productos.service-id=servicio-productos
zuul.routes.productos.path=/api/productos/**

zuul.routes.items.service-id=servicio-items
```



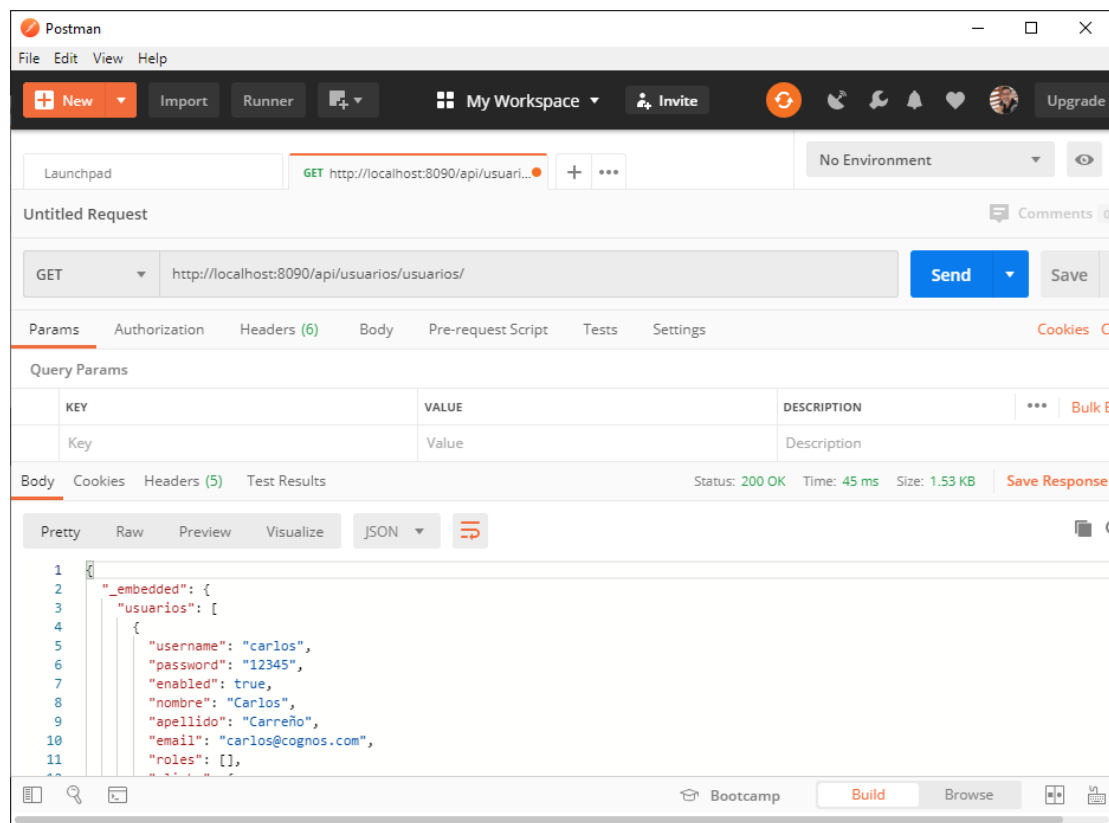
```
zuul.routes.items.path=/api/items/**
```

9. Agrega el archivo de recursos **import.sql** al proyecto **servicio-usuarios** con las inserciones indicadas (puede personalizarlas).

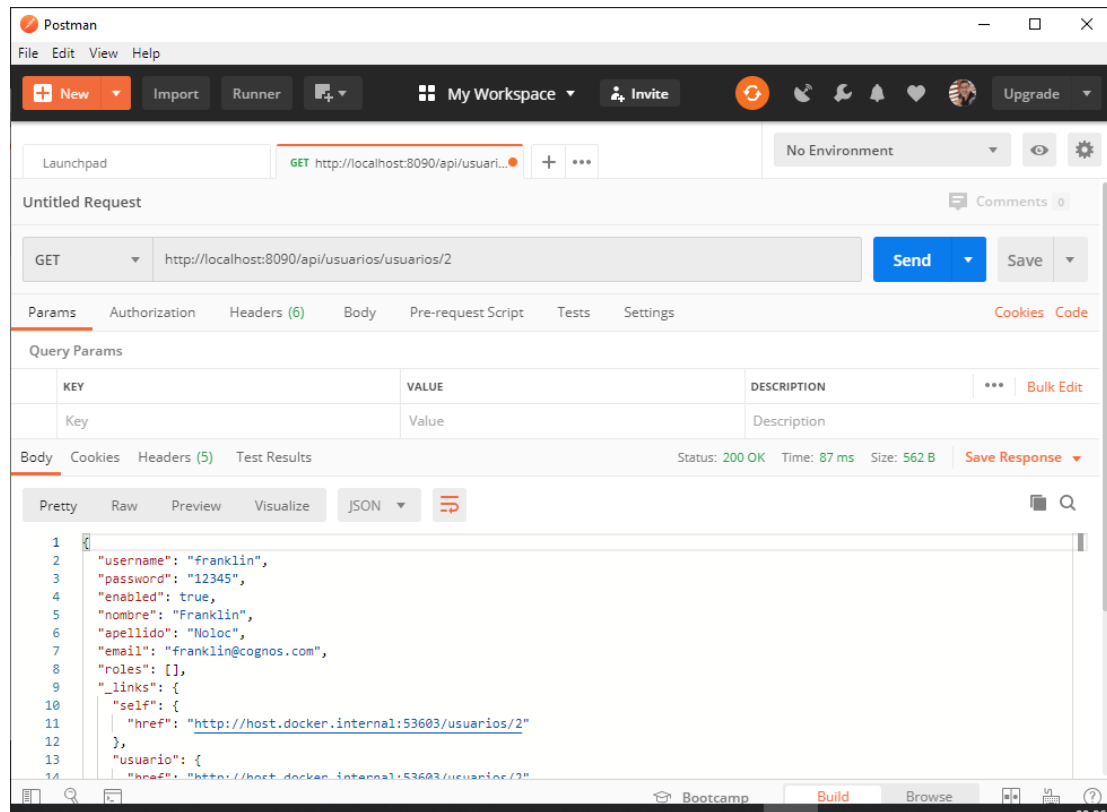
```
insert into usuarios (username,password,enabled,nombre,apellido,email)
values('carlos','12345',1,'Carlos','Carreño','carlos@cognos.com');
```

```
insert into usuarios (username,password,enabled,nombre,apellido,email)
values('franklin','12345',1,'Franklin','Noloc','franklin@cognos.com');
```

10. Reinicia los servicios, Eureka Server, servicio-usuarios y Zuul Server. Prueba con postman que el CRUD se haya exportado al API. <http://localhost:8090/api/usuarios/usuarios/>



Para visualizar un usuario en particular solo tienes que pasarle el id.



11. Edita el archivo de recursos import.sql del proyecto servicio-usuarios, para agregar algunos roles.

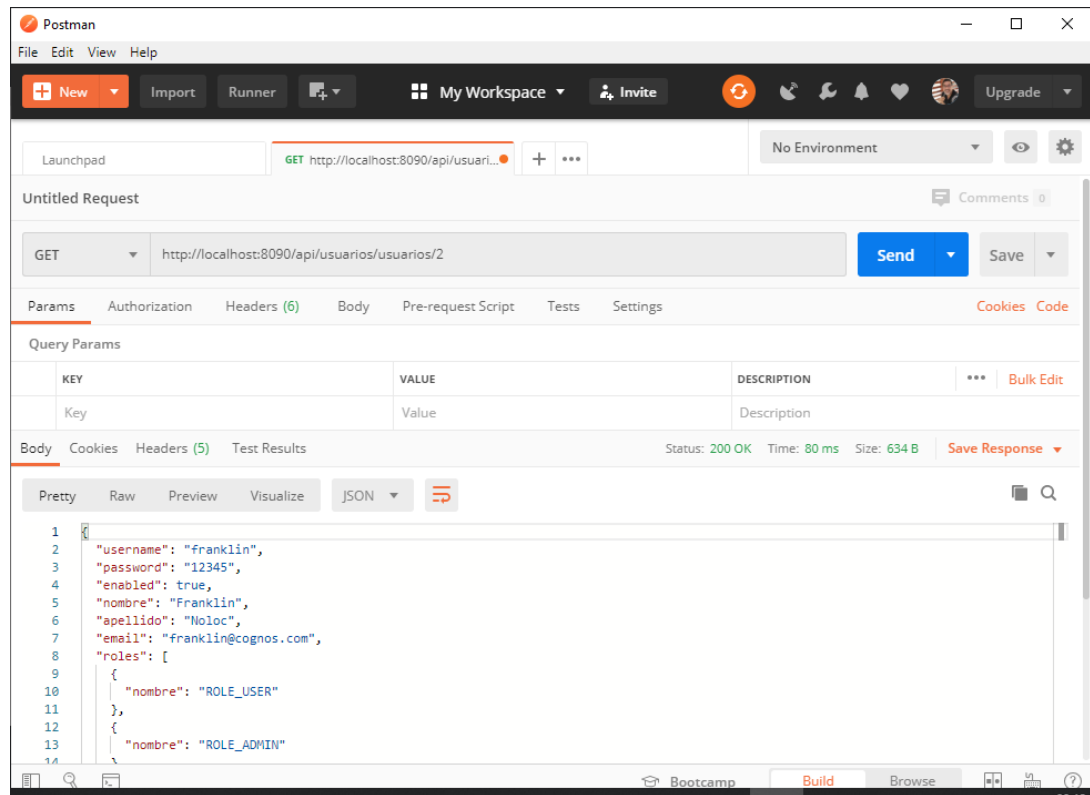
```
insert into usuarios (username,password,enabled,nombre,apellido,email)
values('carlos','12345',1,'Carlos','Carreño','carlos@cognos.com');
insert into usuarios (username,password,enabled,nombre,apellido,email)
values('franklin','12345',1,'Franklin','Noloc','franklin@cognos.com');

insert into roles (nombre) values('ROLE_USER');
insert into roles (nombre) values('ROLE_ADMIN');

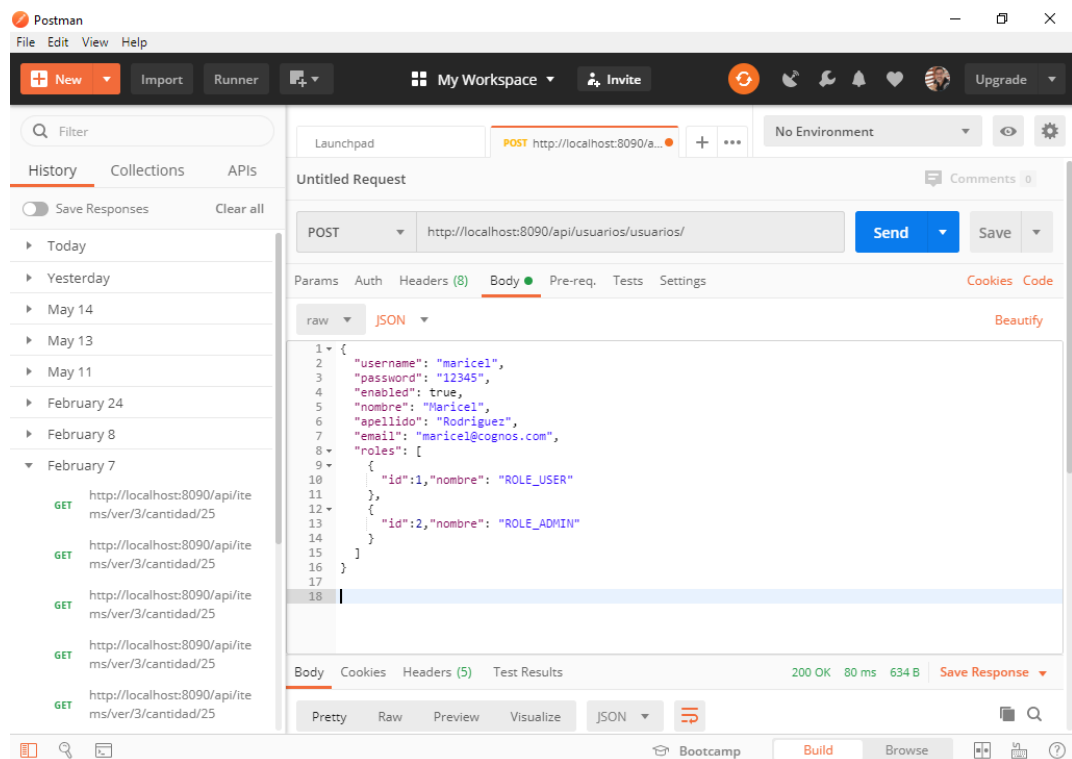
insert into usuarios_rols (usuario_id, role_id) values (1,1);
insert into usuarios_rols (usuario_id, role_id) values (2,2);
insert into usuarios_rols (usuario_id, role_id) values (2,1);
```

Nota: En Spring Security los nombres de los roles tienen que ser escritos en mayúsculas y el nombre debe iniciar con **ROLE_**

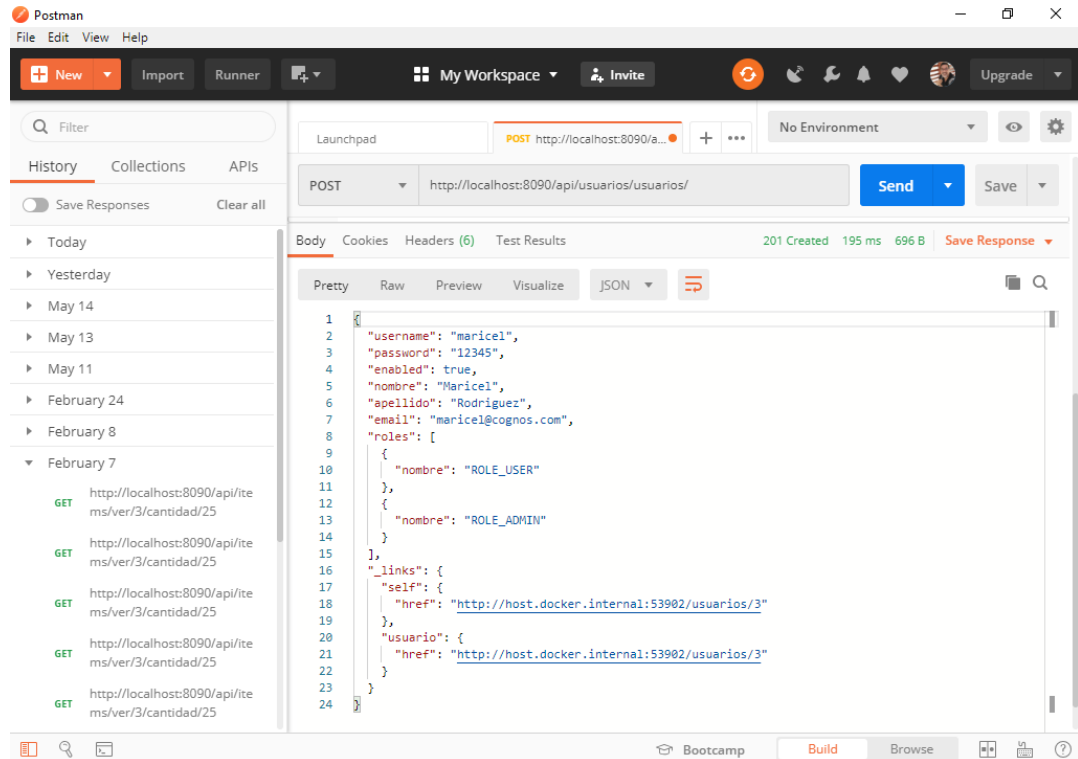
12. Verifica con Postman que los roles se han asignado correctamente.



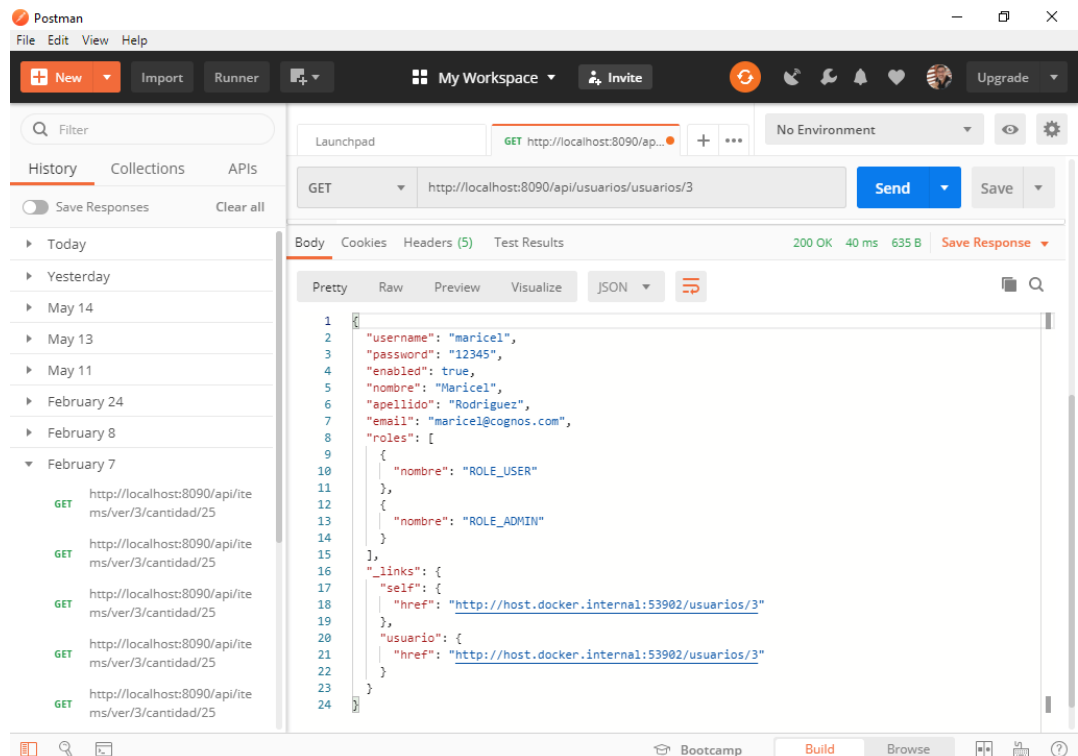
13. Creación de un nuevo usuario con el API Rest exportada.



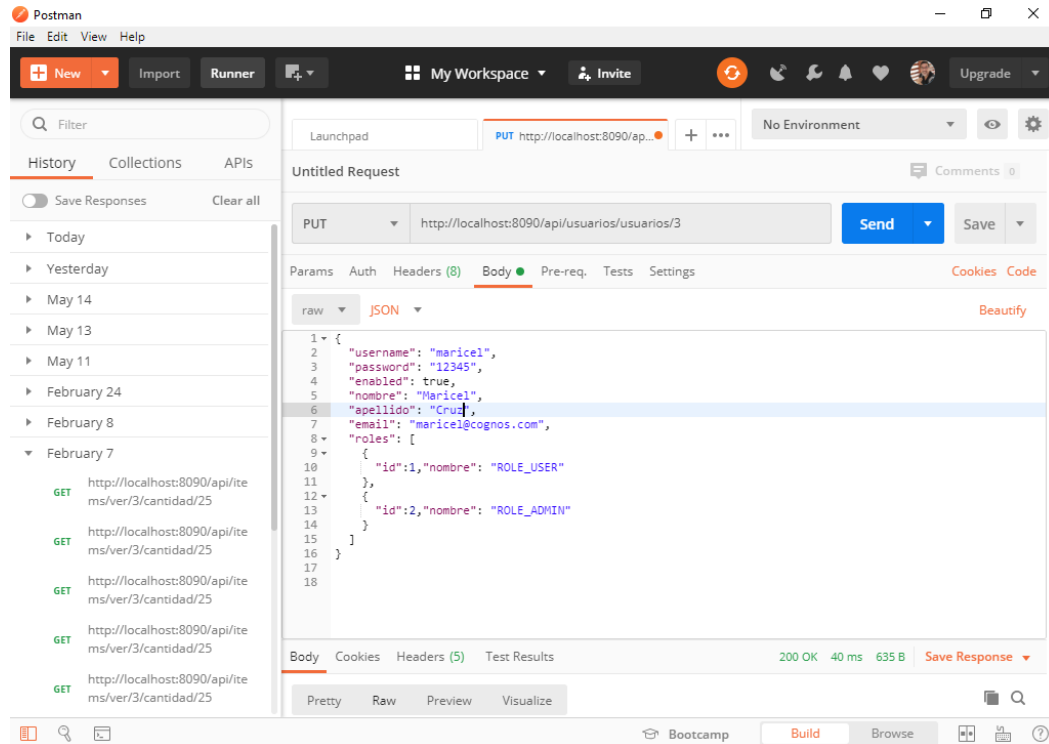
Haz Clic en **Send**, y observa la respuesta de creación del nuevo usuario.



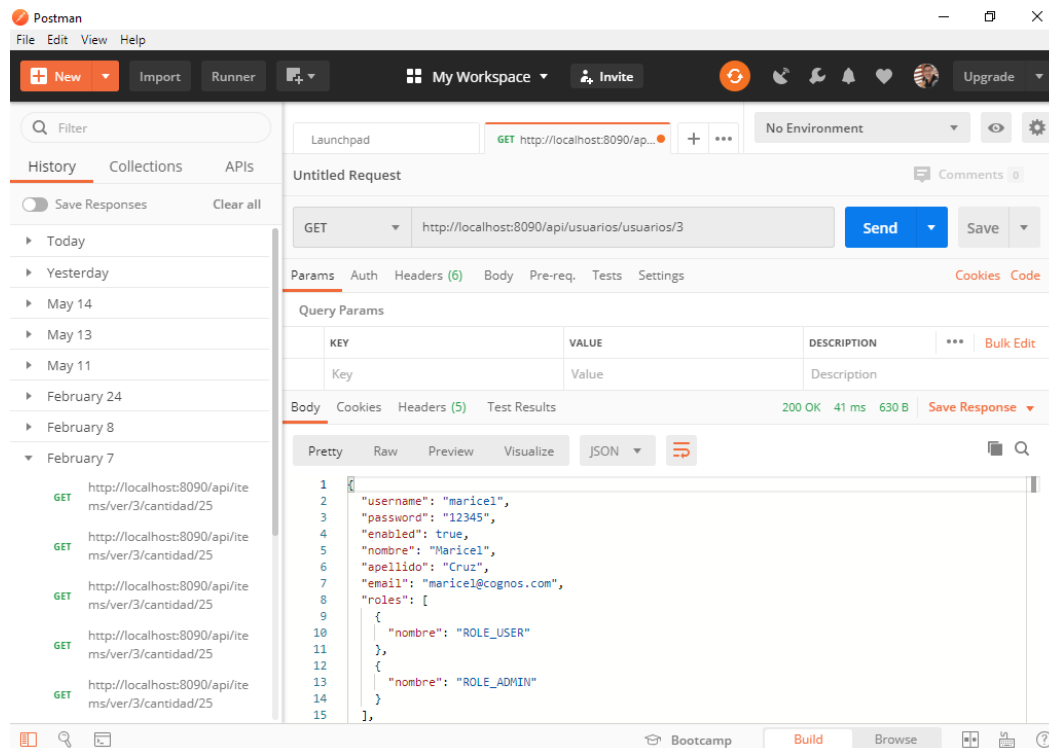
Consulta el nuevo usuario creado.



Edita el usuario



Consulta el usuario y verifica la actualización de los datos.

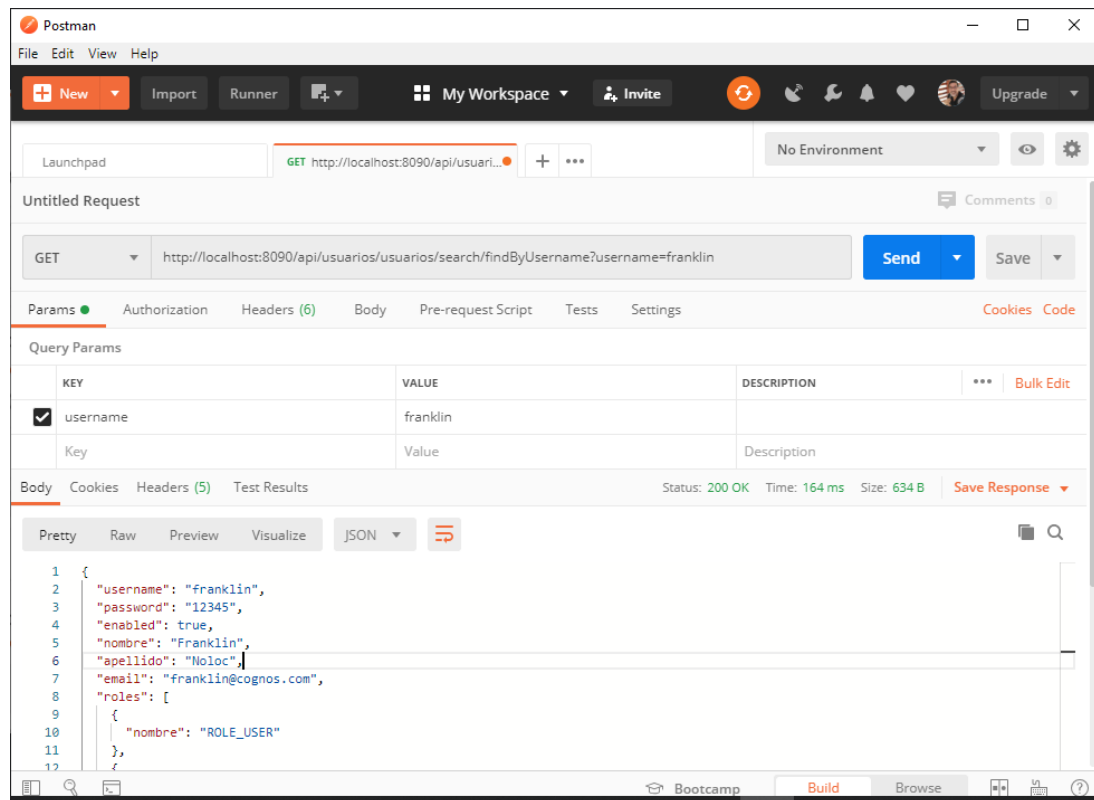


(Opcional) Elimina el usuario con id:1

14. Exporta al API Rest los métodos personalizados de la clase UsuariosDao.java

Para acceder a los métodos personalizados usamos el path search así para llamar al método **findByUsername** el endpoint será:

<http://localhost:8090/api/usuarios/usuarios/search/findByUsername?username=franklin>



15. Exponiendo el método con un path personalizado. En la clase **UsuarioDao.java** anota el método **findByUsername** con la anotación **@RestResource**

```
package com.cognos.app.usuarios.model.dao;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;
import org.springframework.data.rest.core.annotation.RestResource;

import com.cognos.app.usuarios.model.entity.Usuario;

@RepositoryRestResource(path = "usuarios")
public interface UsuarioDao extends PagingAndSortingRepository<Usuario, Long> {
```

```

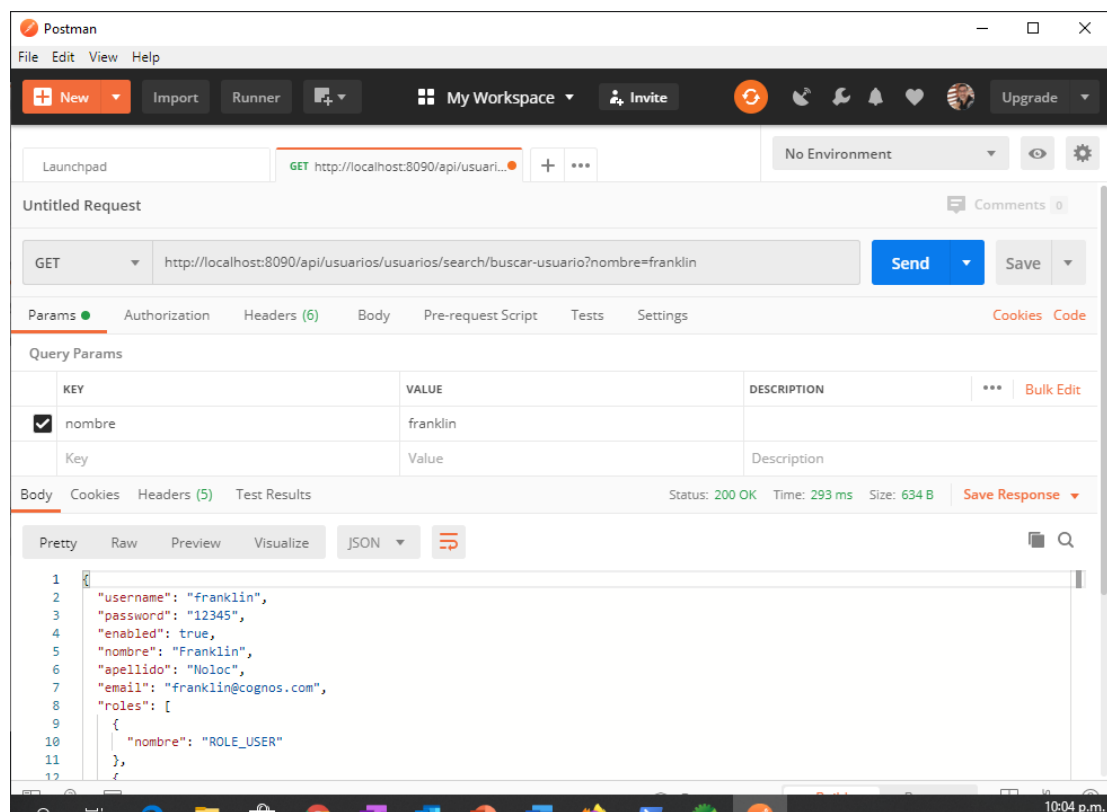
@RestResource(path = "buscar-usuario")
public Usuario findByUsername(@Param("nombre") String username);

@Query("Select u From Usuario u where u.username=?1")
public Usuario obtenerPorUsername(String username);

}

```

16. Reinicia los servicios servicio-usuarios, servicio-eureka-server y servicio-zuul-server y consulta el usuario con nombre 'franklin' usando postman.



17. Exponiendo los ID en el API Rest. En el proyecto **springboot-servicio-usuarios**, crea la clase **RepositoryConfig.java** en el paquete base **com.cognos.app.usuarios**.

```

package com.cognos.app.usuarios;

import org.springframework.context.annotation.Configuration;
import org.springframework.data.rest.core.config.RepositoryRestConfiguration;
import org.springframework.data.rest.webmvc.config.RepositoryRestConfigurer;

import com.cognos.app.usuarios.model.entity.Role;

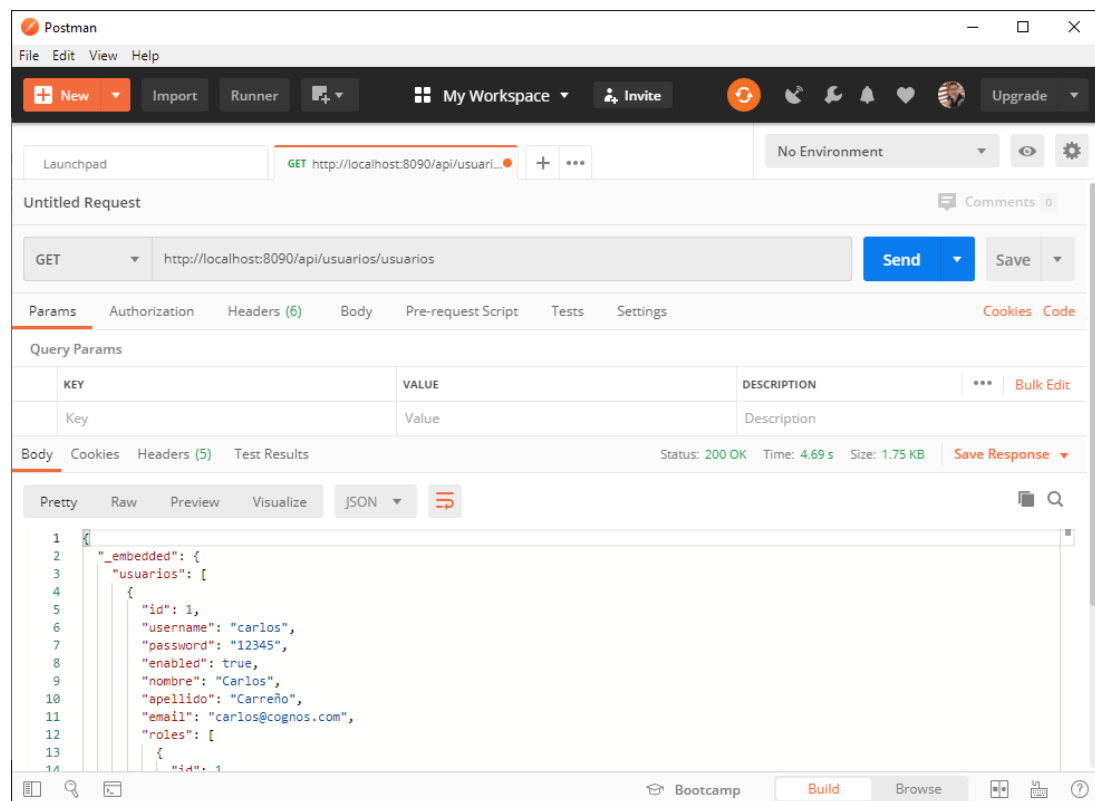
```

```
import com.cognos.app.usuarios.model.entity.Usuario;

@Configuration
public class RepositoryConfig implements RepositoryRestConfigurer {

    @Override
    public void configureRepositoryRestConfiguration(RepositoryRestConfiguration config) {
        config.exposeIdsFor(Usuario.class, Role.class);
    }
}
```

Consulta los usuarios con Postman y verifica que ahora se muestra el id de los usuarios.



Creando la librería Commons de Usuarios

18. Crea un nuevo proyecto llamado `springboot-servicio-usuarios-commons`. Agrega la dependencia Spring Data JPA.

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

☐ Add project to working sets

Working sets:

Agrega la dependencia Spring Data JPA.

New Spring Starter Project Dependencies

Spring Boot Version:

Frequently Used:

<input type="checkbox"/> Config Client	<input type="checkbox"/> Config Server	<input type="checkbox"/> Eureka Discovery Client
<input type="checkbox"/> H2 Database	<input type="checkbox"/> Spring Boot Actuator	<input type="checkbox"/> Spring Boot DevTools
<input checked="" type="checkbox"/> Spring Data JPA	<input type="checkbox"/> Spring Web	<input type="checkbox"/> Zuul [Maintenance]

Available:

Type to search dependencies

- Microsoft Azure
- NoSQL
- Observability
- Ops
- Pivotal Cloud Foundry
- SQL
 - ☐ JDBC API
 - ☒ Spring Data JPA
 - ☐ Spring Data JDBC
 - ☐ Spring Data R2DBC
 - ☐ MyBatis Framework
 - ☐ Liquibase Migration
 - ☐ Flyway Migration
 - ☐ JOOQ Access Layer
 - ☐ IBM DB2 Driver

Selected:

X Spring Data JPA

19. En el archivo pom.xml del proyecto springboot-servicio-usuarios-commons elimina los plugins.

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

20. En el proyecto **springboot-servicio-usuarios-commons** quita el método main() y agrega la anotación **@EnableAutoConfiguration** a la clase **SpringbootServicioUsuariosCommonsApplication.java**

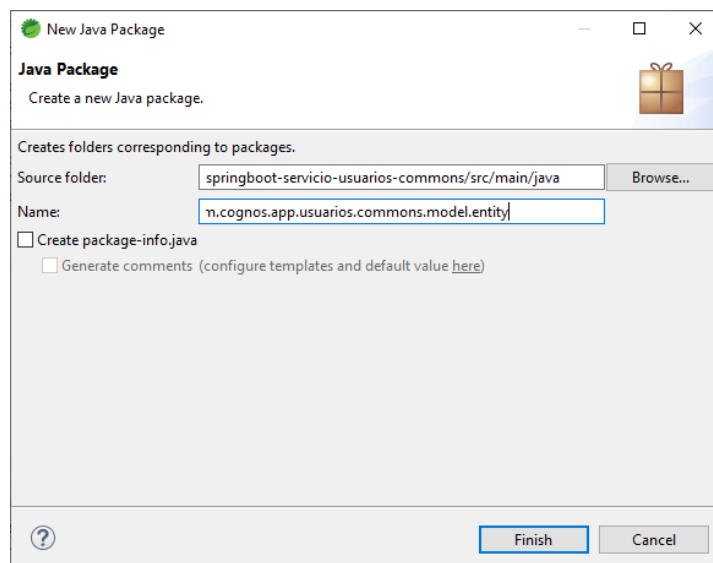
```
package com.cognos.app.usuarios.common;

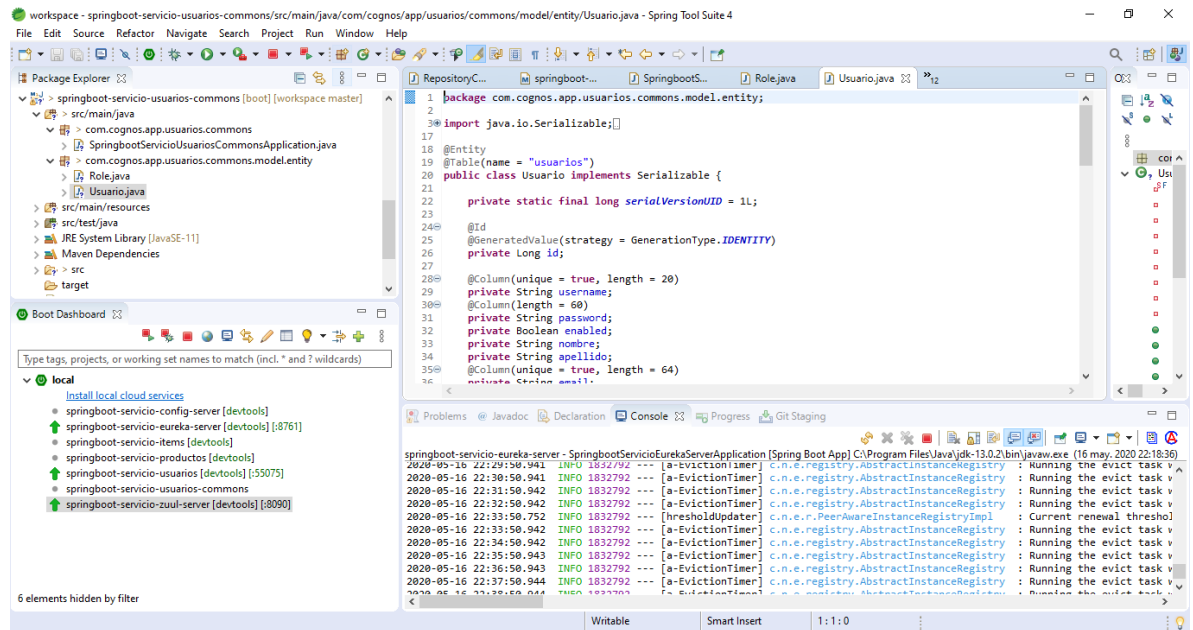
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

@SpringBootApplication
@EnableAutoConfiguration(exclude = {DataSourceAutoConfiguration.class})
public class SpringbootServicioUsuariosCommonsApplication {

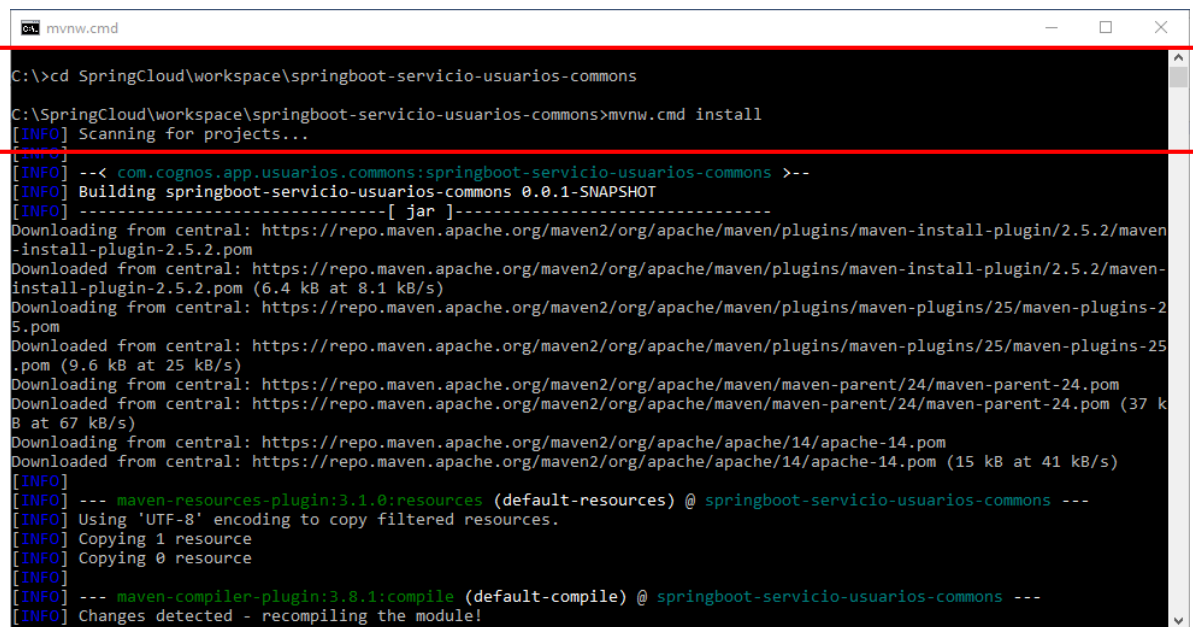
}
```

21. Crea el paquete **com.cognos.app.usuarios.common.model.entity** y copia las clases **Usuario.java** y **Role.java** desde el proyecto **springboot-servicio-usuarios** al proyecto **springboot-servicio-usuarios-commons**.

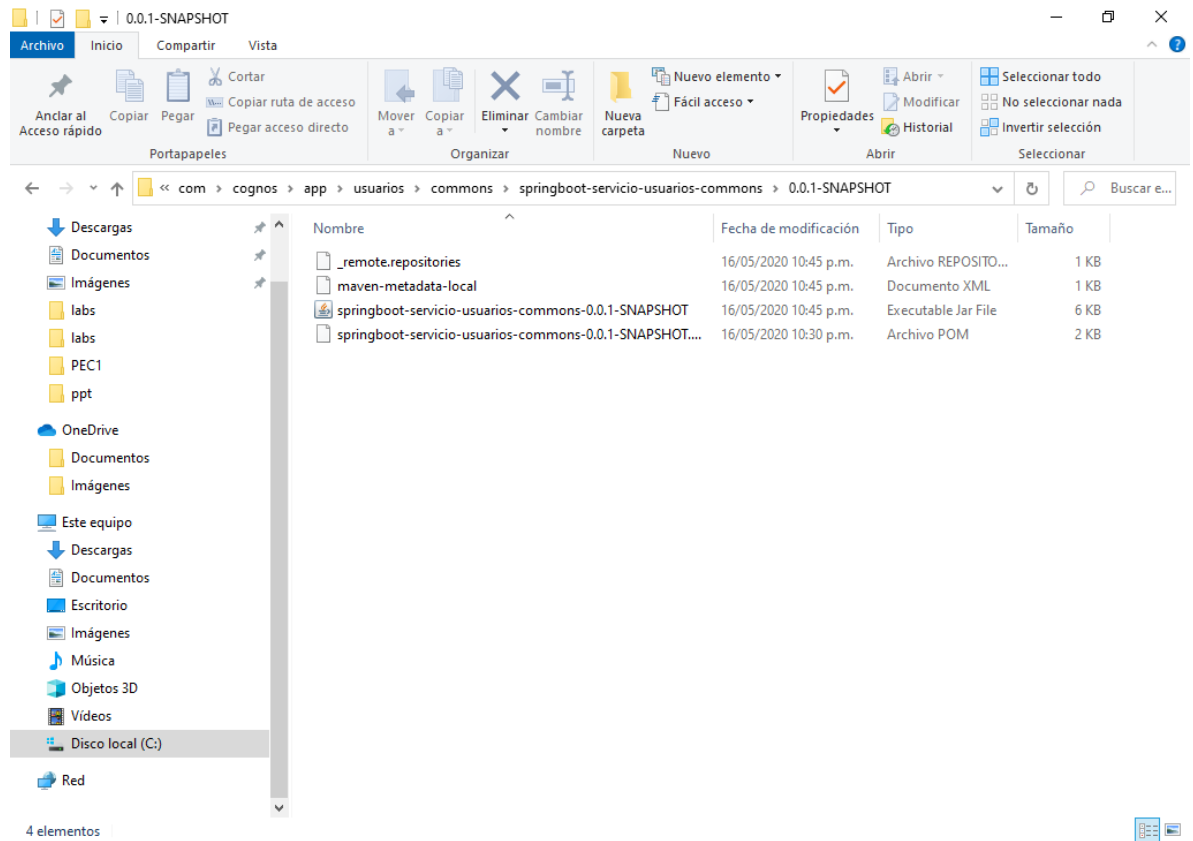




22. Genera el archivo jar del proyecto **springboot-servicio-usuarios-commons**. Abre un terminal en el sistema operativo (En Windows no usar PowerShell), activa el directorio del proyecto y ejecuta el comando: `mvnw.cmd install`



23. Verifica que se haya creado el archivo jar en el repositorio local de Maven.



24. Modifica el archivo pom.xml del proyecto springboot-servicio-usuarios y agrega como dependencia la librería springboot-servicio-usuarios-commons.

```
<dependency>
  <groupId>com.cognos.app.usuarios.common</groupId>
  <artifactId>springboot-servicio-usuarios-commons</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</dependency>
```

25. En el proyecto **springboot-servicio-usuarios** agrega la anotación **@EntityScan** a la clase **SpringbootServicioUsuariosApplication.java**

```
package com.cognos.app.usuarios;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.domain.EntityScan;

@EntityScan({"com.cognos.app.usuarios.common"})
@SpringBootApplication
public class SpringbootServicioUsuariosApplication {

    public static void main(String[] args) {
```

```
SpringApplication.run(SpringbootServicioUsuariosApplication.class, args);
}
}
```

26. En el proyecto springboot-servicio-usuarios, elimina las clases Usuario.java y Role.java y actualiza las importaciones de estas clases en todas las clases del proyecto para que importen las clases desde el paquete: **com.cognos.app.usuarios.common**s. Inicia los servicios y consulta la lista de usuarios con Postman.

