

Diseño y Construcción de Microservicios

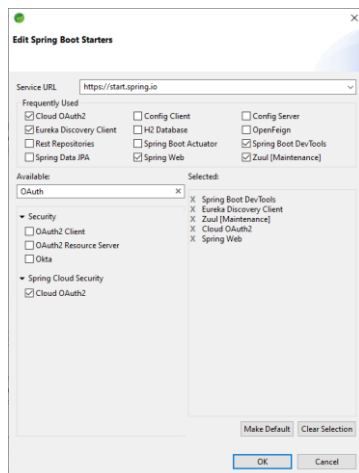
LAB Microservicios y Configuración Claims en Zuul

Objetivos

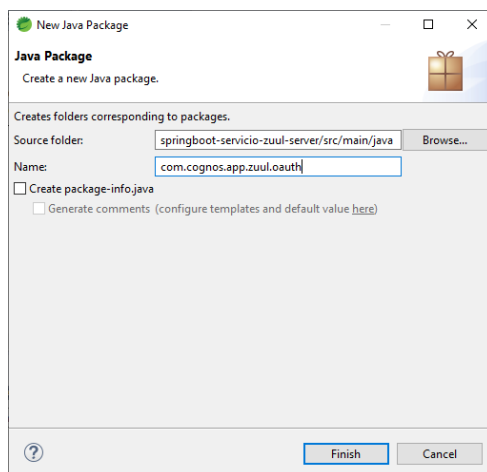
- Mostrar al participante el procedimiento para la configuración recursos protegidos en Zuul Server (claims).

Procedimiento

1. Agrega la dependencia **Cloud OAuth2** al proyecto **springboot-servicio-zuul-server**.



2. Crea el paquete **com.cognos.app.zuul.oauth** en el proyecto **springboot-servicio-zuul-server**



3. En el paquete **com.cognos.app.zuul.oauth** crea la clase **ResourceServerConfig.java**.

```
package com.cognos.app.zuul.oauth;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.oauth2.config.annotation.web.configuration.EnableResourceServer;
import org.springframework.security.oauth2.config.annotation.web.configuration.ResourceServerConfigurerAdapter;
import org.springframework.security.oauth2.config.annotation.web.configurers.ResourceServerSecurityConfigurer;
import org.springframework.security.oauth2.provider.token.store.JwtAccessTokenConverter;
import org.springframework.security.oauth2.provider.token.store.JwtTokenStore;

@Configuration
@EnableResourceServer
public class ResourceServerConfig extends ResourceServerConfigurerAdapter {

    @Override
    public void configure(ResourceServerSecurityConfigurer resources) throws Exception {
        resources.tokenStore(tokenStore());
    }

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests().antMatchers("/api/security/oauth/**").permitAll()
            .antMatchers(HttpMethod.GET, "/api/productos/listar", "/api/items/listar",
                "/api/usuarios/usuarios").permitAll()
            .antMatchers(HttpMethod.GET, "/api/productos/mostrar/{id}",
                "/api/items/mostrar/{id}/cantidad/{cantidad}",
                "/api/usuarios/usuarios/{id}").hasAnyRole("ADMIN", "USER")
            .antMatchers(HttpMethod.POST, "/api/productos/crear",
                "/api/items/crear").hasRole("ADMIN")
            .antMatchers(HttpMethod.PUT, "/api/productos/editar/{id}",
                "/api/items/editar/{id}", "/api/usuarios/usuarios/{id}").hasRole("ADMIN")
            .antMatchers(HttpMethod.DELETE, "/api/productos/eliminar/{id}",
                "/api/items/eliminar/{id}", "/api/usuarios/usuarios/{id}").hasRole("ADMIN")
            .anyRequest().authenticated();

        // tambien podria ser de forma genérica todos los métodos GET, POST DELETE
        // .antMatchers("/api/productos/**", "/api/items/**",
        //     "/api/items/**", "/api/usuarios/**").hasRole("ADMIN")
    }
}
```

```






@Bean
public JwtTokenStore tokenStore() {
    // Para almacenar el token necesitamos convertirlo
    return new JwtTokenStore(accessTokenConverter());
}

@Bean
public JwtAccessTokenConverter accessTokenConverter() {
    JwtAccessTokenConverter tokenConverter = new
                                                JwtAccessTokenConverter();

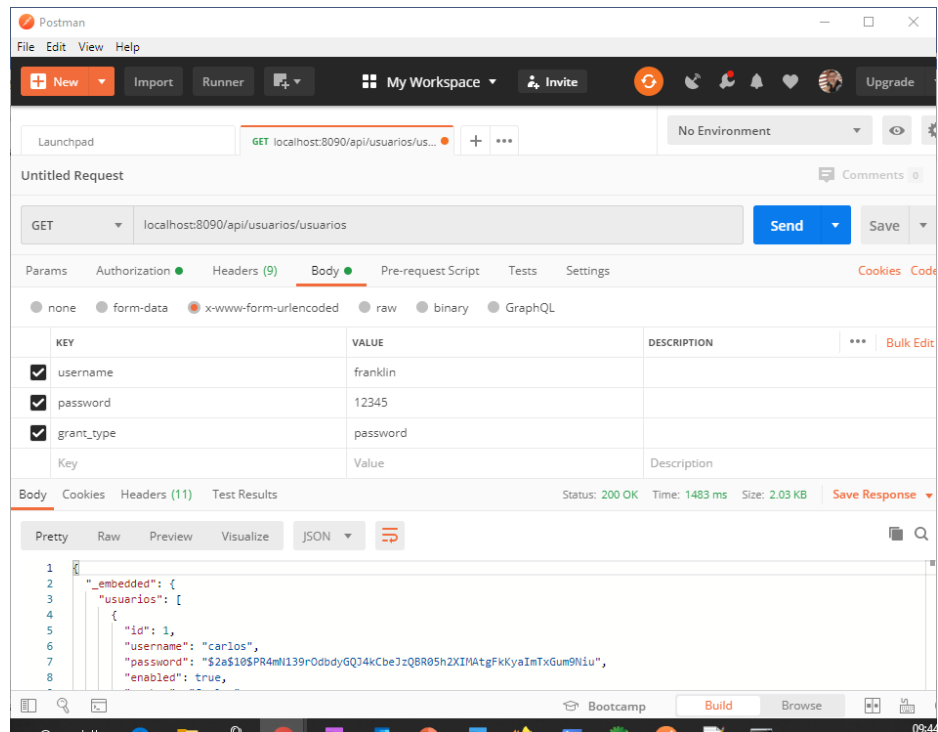
    String key = "algo_secreto";
    tokenConverter.setSigningKey(key);
    return tokenConverter;
}
}

```

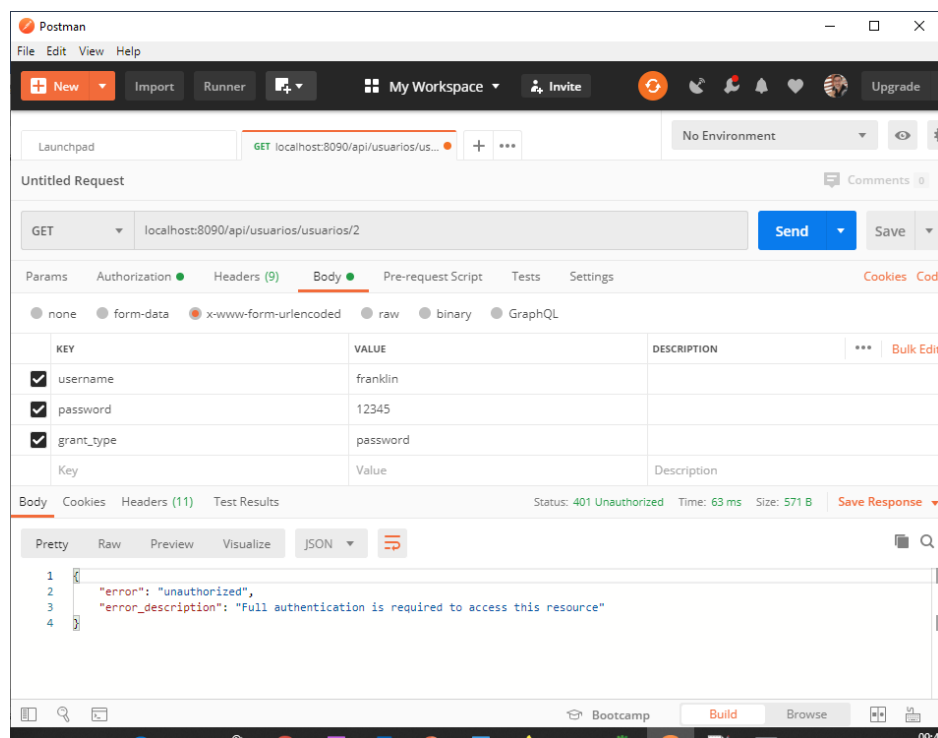
4. Inicia los servicios.

 **local**
[Install local cloud services](#)
 ● springboot-servicio-config-server [devtools]
 springboot-servicio-eureka-server [devtools] [:8761]
 ● springboot-servicio-items [devtools]
 springboot-servicio-oauth [devtools] [:9100]
 ● springboot-servicio-productos [devtools]
 springboot-servicio-usuarios [devtools] [:65327]
 ● springboot-servicio-usuarios-commons
 springboot-servicio-zuul-server [devtools] [:8090]

5. Prueba el acceso a los recursos con Postman

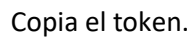


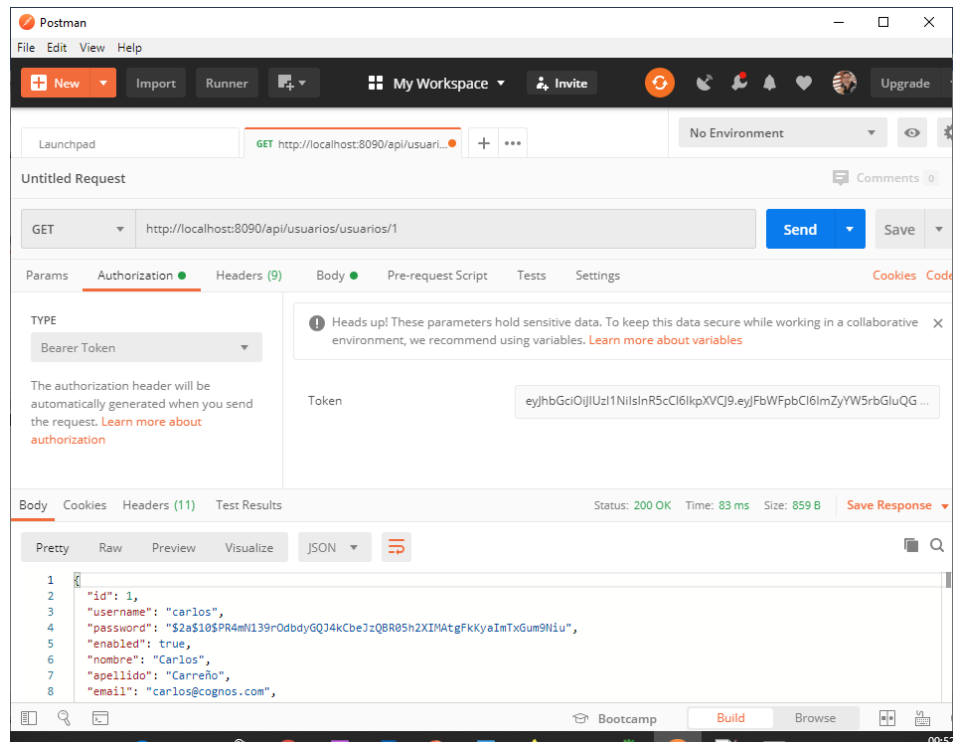
Muestra el usuario con id:2



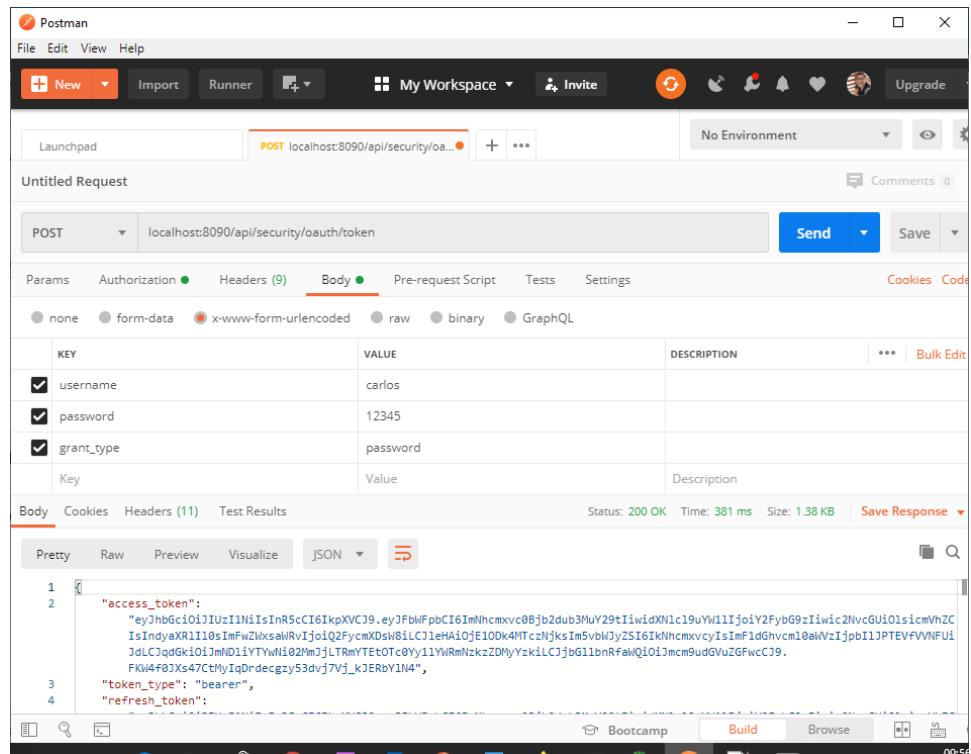
Observa que la operación requiere autenticación.

Genera el token.

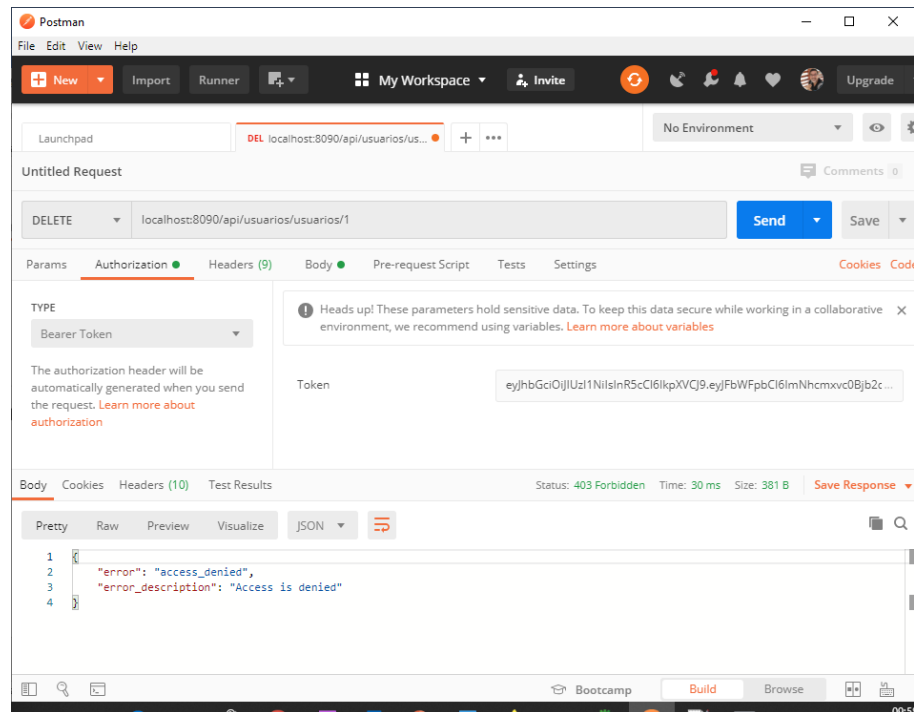




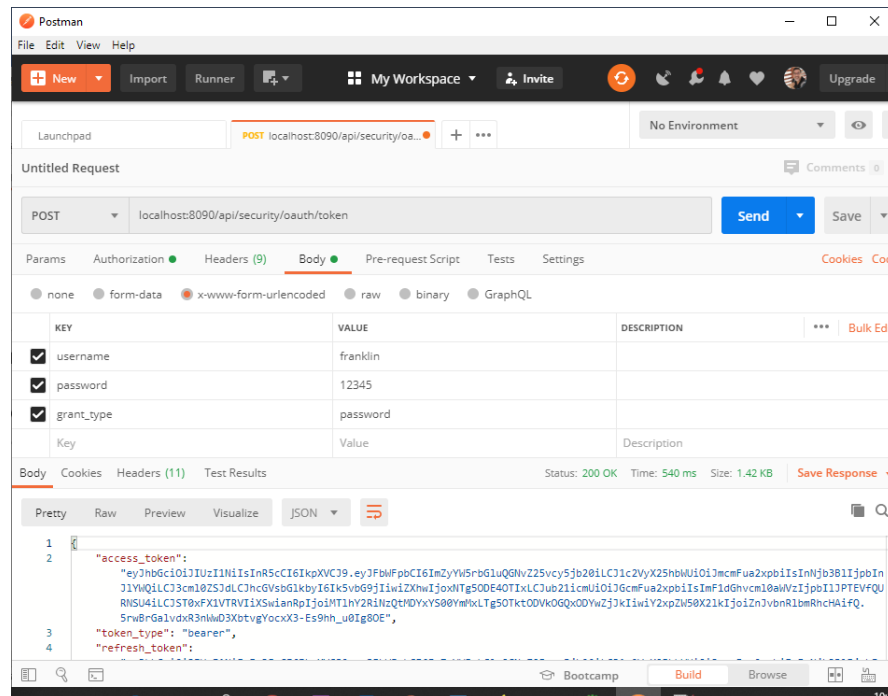
Vamos a intentar eliminar un usuario con el token del usuario Carlos. Generamos el token primero.



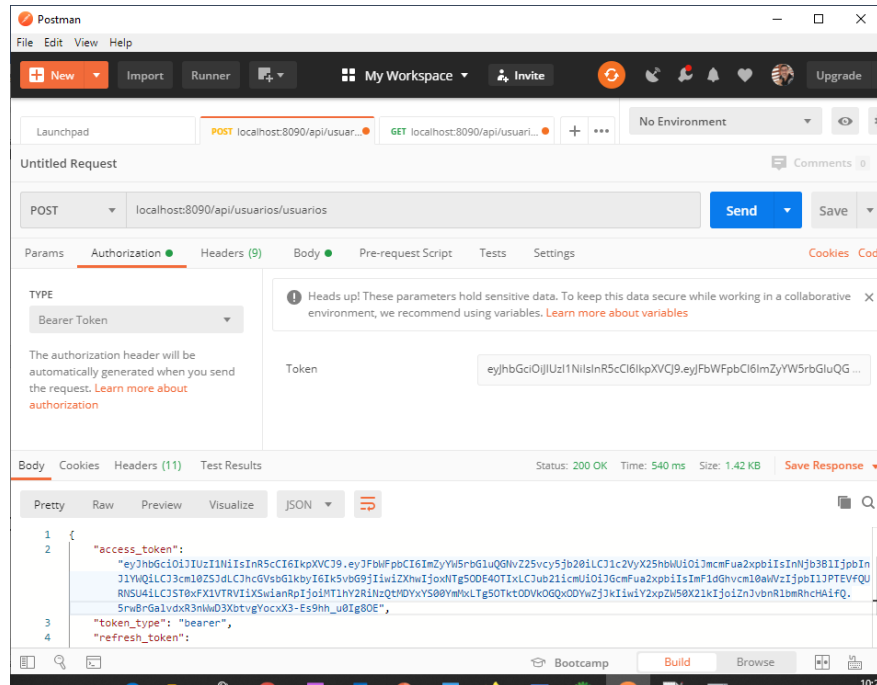
Observa que la respuesta indica un error del tipo **403 Forbidden**, es decir que esta operación esta prohibida con esta credencial pues **"carlos"** no tiene el rol ADMIN.



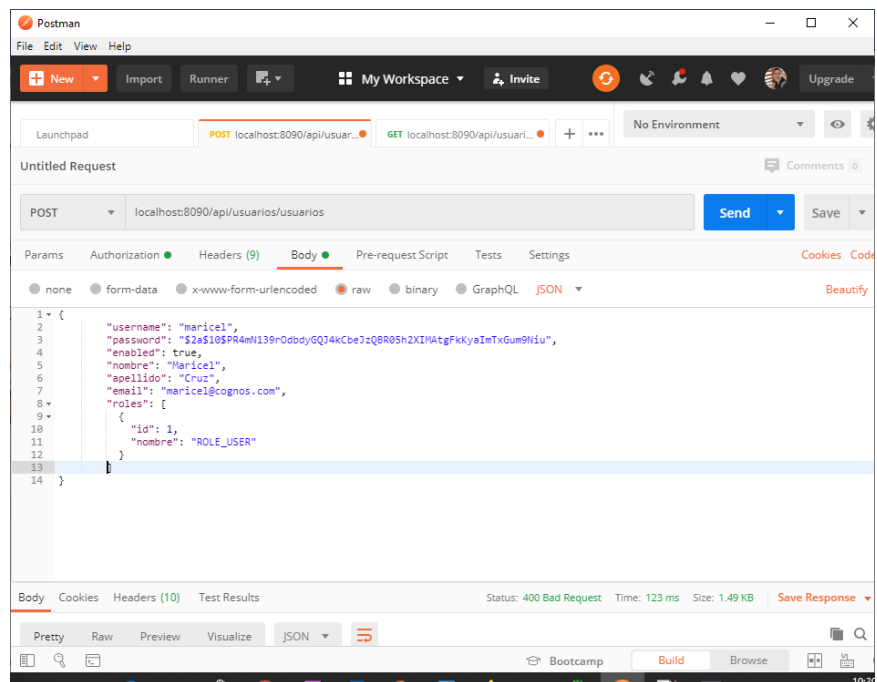
Vamos a crear un nuevo usuario, genera el token del usuario **“franklin”**, es el usuario que tiene el rol **ADMIN**.



Copia el token generado.



Creamos el **body** del nuevo usuario.



Observa la respuesta.

