

Primer Acercamiento Seguridad en Microservicios

La seguridad en los microservicios, tanto autenticación y autorización, es una fase muy importante y necesita ser analizada y trabajada de manera cuidadosa. A continuación la primera capa de un modelo complejo de seguridad que se espera lograr

RBAC - Primer Enfoque - Seguridad En Microservicios

Propósito

“

La seguridad de nuestros servicios como empresa es de las cosas mas importantes y de las que deben ser tratadas con más cuidado, la seguridad se refiere a proteger la información y el procesamiento de la misma, evitar el acceso a datos a los cuales un usuario no debería tener acceso, la autorización de usuarios y la autenticación de los mismos.

Partiendo de este concepto debemos tener claro y hacer énfasis en el cuidado con el que se desarrolla el apartado de la seguridad , pues se espera tener desde el inicio conceptos y una idea clara que facilite el desarrollo a futuro de la seguridad en profundidad.

En este primer enfoque lo se busca es lograr una restricción a nivel de los microservicios, pues el proyecto RSI cuenta con una arquitectura de microservicios y aunque la teoría nos dice que deben ser piezas funcionales pequeñas en la práctica tenemos microservicios realmente grandes y con todo el desarrollo actual y los planes del mismo la seguridad se vuelve algo complicado de manejar.

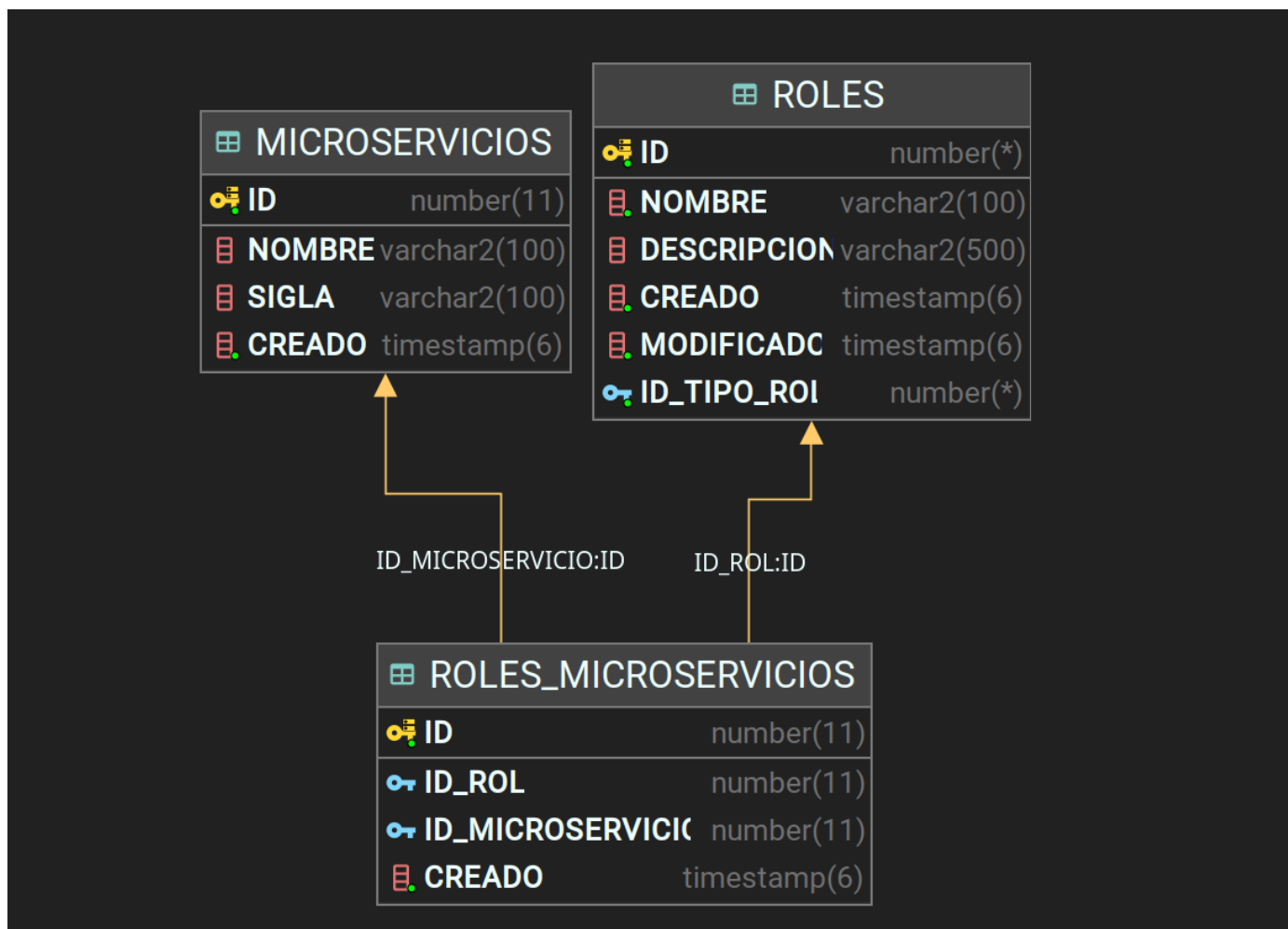
El sistema de almacenamiento de datos se maneja en una base de datos relacional (Oracle) sobre la cual se deberán hacer los ajustes para tener la arquitectura (llámese arquitectura a las tablas y sus relaciones para manejar la autorización) limpia y lista para la implementación en cuestión.

Lógica de autorización

Se decidió definir una lógica de autorización basada en roles y microservicios a los cuales pueden tener acceso, para esto se crearon dos tablas, una llamada MICROSERVICIOS y la otra ROLES_MICROSERVICIOS que relaciona los roles y microservicios existentes, esto con el fin de otorgar o no acceso a un microservicio.

La tabla microservicios consta de dos campos principales; el nombre que hace referencia al nombre del microservicio y la sigla que es un parámetro que con el tiempo no debería ser modificado, ya que es usado en los servicios de autorización en back para comparar que roles tienen o no acceso.

Esquema autorización



Implementación

A nivel de autorización usando Spring Security definimos la clase de seguridad en cada microservicio, esto no se ubica en la librería porque cada microservicio es diferente y a futuro puede requerir cambios específicos que al implementarlo de manera genérica en la librería complicaría las cosas. A continuación el método principal de dicha clase

```

1  @Configuration
2  @EnableWebSecurity
3  public class SecurityConfig extends WebSecurityConfigurerAdapter {
4      ...
5      @Override
6      protected void configure(HttpSecurity http) throws Exception {
7          //habilitamos cors y desabilitamos la protección csrf de modo que cualq
8          if (securityEnabled) {
9              http
10                 .cors().and().csrf().disable()
11                 .authorizeRequests()
12                 .antMatchers("/api/**").access("@RouteGuard.checkAccess()")
13                 .anyRequest().authenticated().and().addFilterBefore(filterF
    
```

```

14         .sessionManagement().sessionCreationPolicy(SessionCreationP
15     }else{
16         http.cors().and().csrf().disable().sessionManagement().sessionCrea
17     }
18 }
19
20 }
```

Y la seguridad o el **decider** lo designamos al Guard implementado por medio del método **access**, esto permite consumir la base de datos y manejar una lógica un poco más completa, este Guard se ve de la siguiente manera.

```

1  @Component("RouteGuard")
2  public class RouteGuard {
3
4      private HttpServletRequest httpServletRequest;
5      private IRolRepository repository;
6
7      public boolean checkAccess(){
8          return this.repository.checkAccessToMicroserviceByAuthorities(Long.parseLong(idRol),
9              IConstantesTraining.INITIALS_MICROSERVICE);
10     }
11
12     ...
13
14 }
```

Y el método **checkAccessToMicroserviceByAuthorities** que se encarga de comprobar si el usuario de la petición por medio de sus roles asignados tiene acceso a dicho microservicio luce de la siguiente manera.

```

1  @Query("SELECT CASE WHEN COUNT (t) > 0 THEN TRUE ELSE FALSE END " +
2      "From Microservicio t where t.sigla = :sigla and t.id in " +
3      "(select r.idMicroservicio from RolMicroservicio r where r.idRol in " +
4      "(select u.idRol from UsuarioRol u where u.idUsuario = :idUsuario)");
5  Boolean checkAccessToMicroserviceByAuthorities (Long idUsuario, String sigla);
```

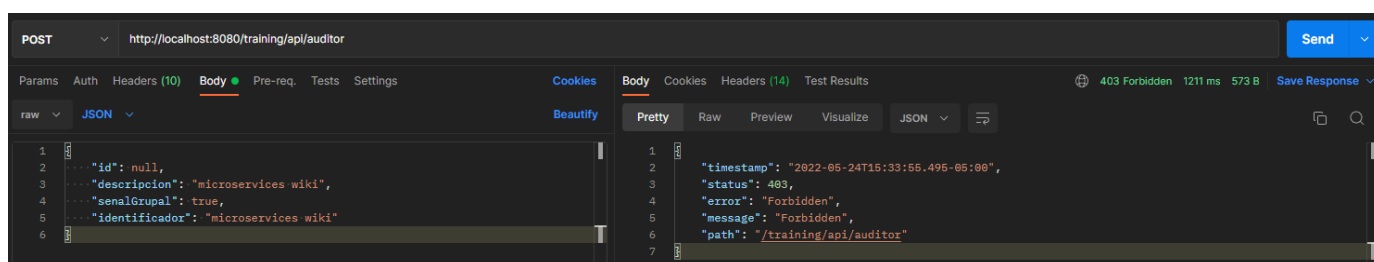
Es una consulta que relaciona el esquema previamente explicado y maneja el acceso a los microservicios representándolo como un booleano que habilita o deshabilita el acceso comparando si hay un registro en base de datos o no.

Pruebas

Si no tenemos registro definido para ningún rol de los que tiene asignados mi usuario, entonces el backend responderá con un error de autorización.

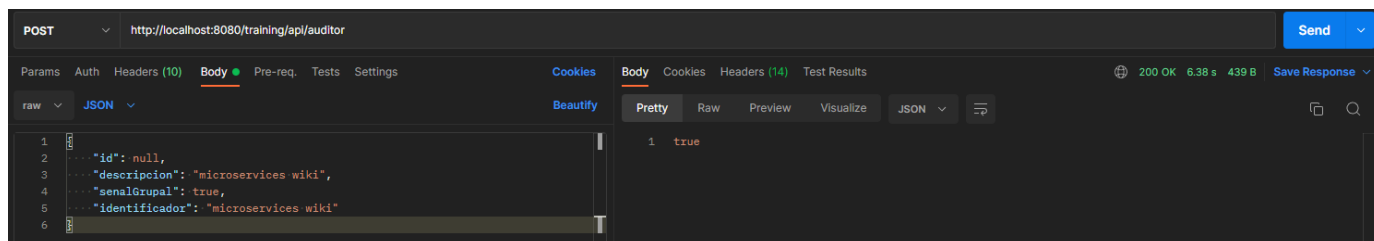


De esta manera la petición es denegada.



Por su parte si añadimos un registro para la sigla **TRAINING** que es la sigla que identifica al microservicio de training y la cual se encuentra en el archivo **IConstantesTraining** y será similar en cada microservicio entonces vemos como la petición es resuelta, esto libera el proyecto de estar modificando las versiones en backend y que sea una lógica de simple acceso para no complicarlo a base de datos.

A continuación la demostración de la petición resuelta.



Consideraciones

Notamos que hay una consulta en cada petición para comparar el **idUsuario** que viene en los **headers** con el **idUsuario** que está en el token, esto se podría refactorizar de tal manera que el **idUsuario** pueda venir codificado en el **token**, optimizando de esta manera el tiempo en las peticiones.

Sugerimos que se añada información adicional en el **token**, como por ejemplo los roles que tenga el usuario y el **username**.



Las siguientes aproximaciones en el apartado de seguridad tratarán los aspectos de seguridad por módulos y por métodos. Para esto a nivel de base de datos tenemos las tablas **roles** y **tipos roles** que los categorizan, las cuales serán tomadas en cuenta y en conjunto con las anotaciones previamente estudiadas **@PreAuthorize** permitirán definir el resto de **RBAC** sobre los proyectos.

Esto implica no solo un monitoreo sobre la evolución y el comportamiento en los proyectos sino también un esfuerzo tanto de líderes como analistas para definir previamente a cada nuevo desarrollo el nivel de acceso o seguridad que sea requerido.

Para las siguientes fases tenemos en base de datos las tablas que asocian los roles y los tipos roles con lo cuál una segunda fase agrupará dichos datos para mantener una seguridad a un segundo nivel.

En la tercer fase se espera un control más detallado con el cuál a nivel de métodos se restringe el acceso, para esto en base de datos está planteada una lógica de endpoints y acciones que puede llegar a ser útil, a este punto aún no está definido pero en su momento se adaptará y se encontrará la manera de manejarlo.