

Rbac - Seguridad en microservicios segundo acercamiento

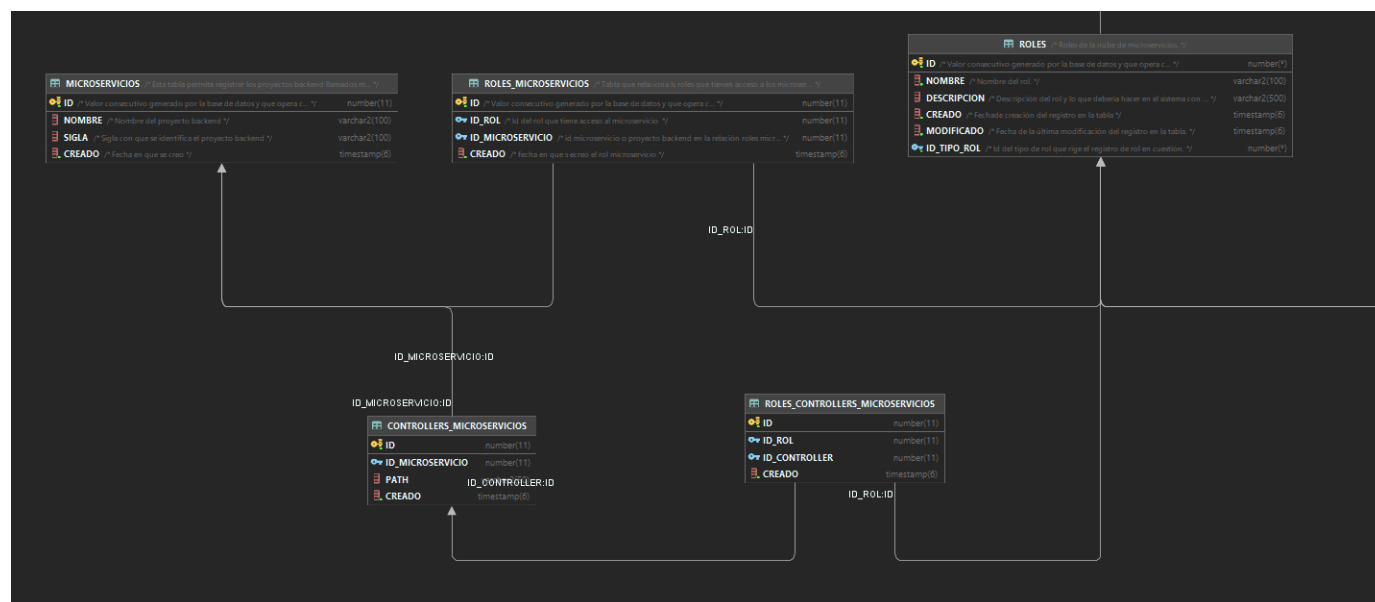
Seguridad sobre microservicios concentrando los esfuerzos en seguridad a nivel de controller y método.

RBAC - Segundo Enfoque - Seguridad En Microservicios

La seguridad Puede ser manejada a varios niveles, en el primer acercamiento se propuso una seguridad a nivel de **microservicio** y se plantearon las tablas **microservicios** y **microservicios roles**, todo esta documentado [aquí](#).

Para esta segunda aproximación se espera una seguridad más detallada (***Fine-grained*** access control) y para ello **spring security** ofrece desde configuraciones [httpSecurity](#) , y [globalMethodSecurity](#) principalmente y con estos se podría decir que es suficiente para manejar la autorización a cualquier nivel.

Teniendo claro entonces que el objetivo de este acercamiento es definir una seguridad a nivel de controller y otra a nivel de metodo se propone el siguiente esquema en base de datos.



En él podemos apreciar las tablas **controllers_microservicios** y **roles_controllers_microservicios**, y la lógica a implementar consiste en que en el **@PreAuthorize** se invoca un método que corrobore si el valor actual del **@RequestMapping** está declarado en la tabla **controllers_microservicios** y ademas si el usuario que realiza la petición tiene acceso a dicho **controller**, para de este modo permitir o denegar el acceso, esta sería la seguridad intermedia y en resumen para un usuario con los **n** roles disponibles, si alguno tiene acceso, se le permitirá, de lo contrario será denegado.

Seguridad a nivel de controller

Para esta seguridad debemos considerar los siguientes factores, `@PreAuthorize`, el componente `@Component("RouteGuard")`, java Reflection y el identificador `#root`, en esta seguridad es importante manejar estos elementos y tener clara la función de cada uno, el `@PreAuthorize` se encarga de ejecutar una validación anterior a la ejecución del método o clase que engloba, el **component RouteGuard** es una clase que maneja los métodos que controlan el acceso y es un componente para poder ser invocado desde el `@PreAuthorize` o desde el `httpSecurity`, también con el uso de **java reflection** se logra obtener el valor de la anotación `@RequestMapping` y de este modo realizar la comparación o contrastar dicho valor con base de datos y el identificador `#root` que nos permite fácilmente acceder al contexto y manejar el uso de parámetros como en este caso el `this` con el cual podemos luego obtener la anotación.

A continuación el ejemplo de la anotación a nivel de clase y el identificador `#root`.

```
1  @RestController
2  @RequestMapping("/api/auditor")
3  @PreAuthorize("@RouteGuard.checkAccessController(#root.this.getClass().getName()
4  public class AuditoriaBackendController {
5      ...
6  }
```

A continuación los métodos en el componente con el uso de java reflection.

```
1  @Component("RouteGuard")
2  @Slf4j
3  public class RouteGuard {
4      ...
5
6      public boolean checkAccessController(String className) throws ClassNotFoundException {
7          var reqMappingValue = Class.forName(className).getAnnotation(RequestMapping.class);
8          var controllerMicroservicio = controllerMicroservicioRepository
9              .findFirstByPathAndInitialsMicroservice(reqMappingValue, initialPath);
10
11          if(controllerMicroservicio == null){
12              log.info("No se ha encontrado un registro para el valor del requestMapping");
13              return false;
14          }
15
16          return this.rollLibraryRepository.checkAccessToControllerByAuthorities(controllerMicroservicio.getId());
17      }
18  }
```

19 | }

A continuación los métodos de los repositorio para poder comparar y decidir el acceso.

```

1  @Query("from ControllerMicroservicio c " +
2      "where c.path = :path " +
3      "and c.idMicroservicio = " +
4      "(select m.id from Microservicio m where m.sigla = :siglaMicroservicio)"
5  ControllerMicroservicio findFirstByPathAndInitialsMicroservice(String path, Str:

```

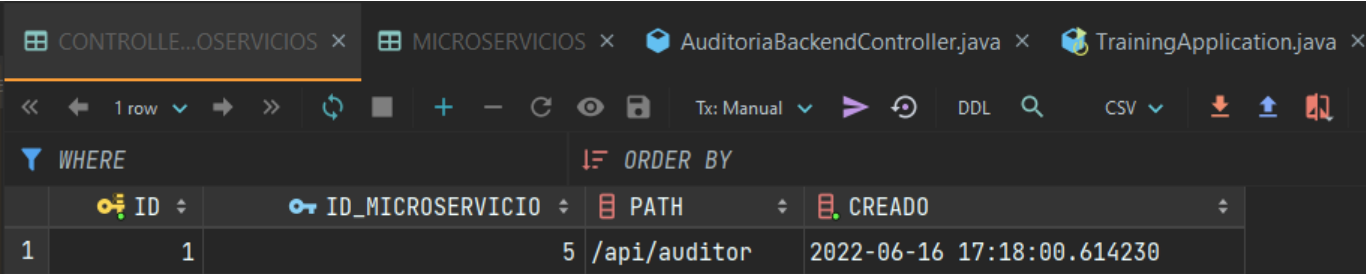
```

1  @Query("SELECT CASE WHEN COUNT (rcm) > 0 THEN TRUE ELSE FALSE END " +
2      "From RolControllerMicroservicio rcm where rcm.idController = :idConti
3      "and rcm.idRol in " +
4      "(select u.idRol from UsuarioRolLibrary u where u.idUsuario = :idUsuar
5  Boolean checkAccessToControllerByAuthorities (Long idUsuario , Long idControlle

```

Ejemplos de uso y respuesta

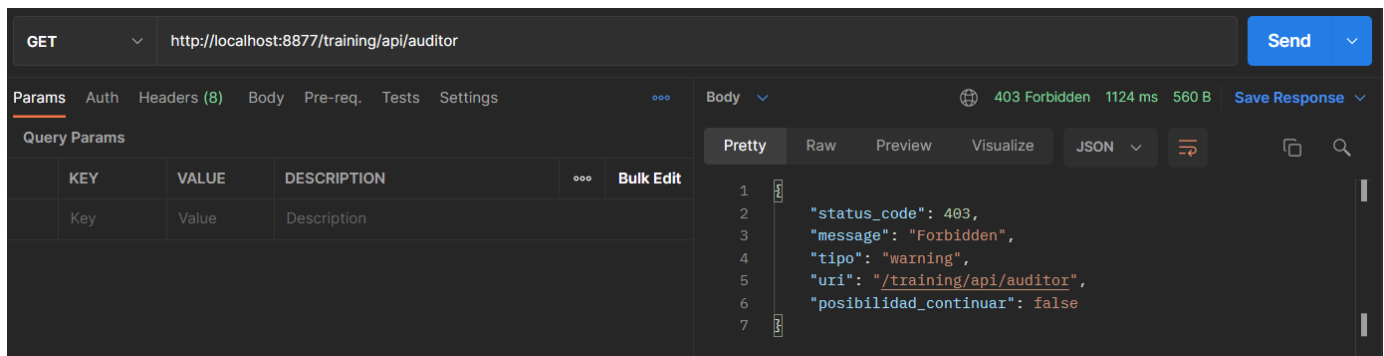
Una vez se ha establecido el `@PreAuthorize` a nivel de clase controladora solo queda invocar cualquier método de la misma clase para comprobar el funcionamiento, pero antes debemos agregar en base de datos información con la cuál tener el control de acceso, para ello debemos agregar la información del controller en la tabla `controller_microservicios`, para el servicio auditor, se añade lo siguiente:



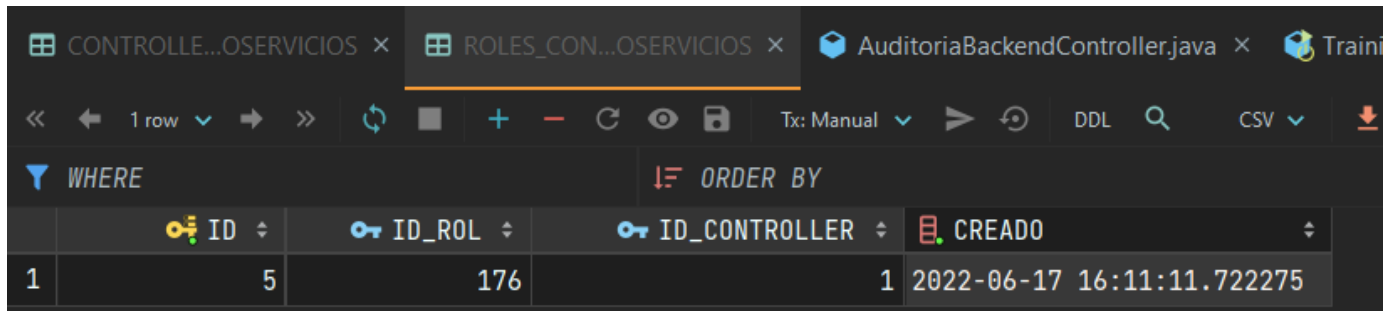
The screenshot shows a database interface with a table named 'CONTROLLER...OSERVICIOS'. The table has four columns: ID, ID_MICROSERVICIO, PATH, and CREADO. There is one row of data with the following values: ID=1, ID_MICROSERVICIO=5, PATH=/api/auditor, and CREADO=2022-06-16 17:18:00.614230.

ID	ID_MICROSERVICIO	PATH	CREADO
1	5	/api/auditor	2022-06-16 17:18:00.614230

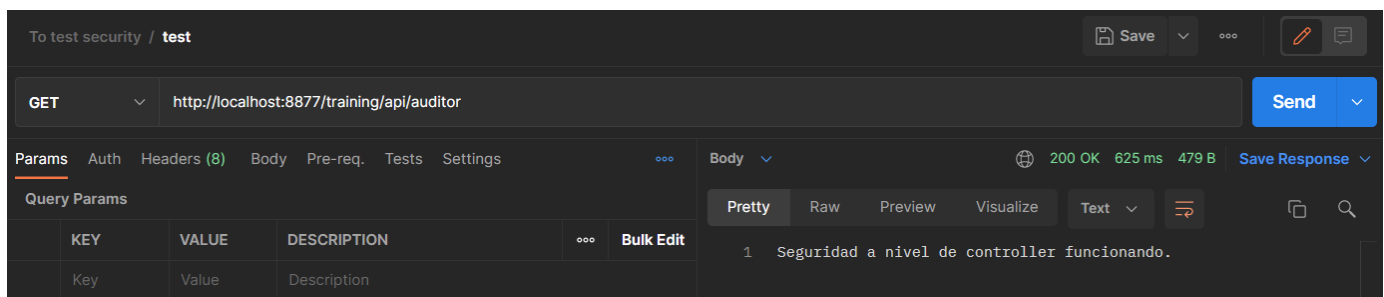
Si ahora realizamos la petición deberá salir un mensaje de error puesto que aún no hay ningún rol asignado a dicho controller y el acceso será denegado, y la excepción es manejada por su respectivo método en el `@RestControllerAdvice`.



Por lo tanto para asignarle acceso a un usuario por su rol debemos agregar dicho rol en la tabla.



Con esto queda enlazado el rol para el controller y la petición será resuelta exitosamente como se aprecia a continuación.



“

Como conclusión en este acercamiento medio se propone una seguridad donde la clase controladora que necesita seguridad extra se anota con `@Preauthorize` y se registra en base de datos el valor del `@RequestMapping` y con esto se corrobora el acceso, este acceso garantiza la ejecución de todos los métodos de la clase, y en caso de que algún método requiera una seguridad adicional se debe utilizar el `@PreAuthorize` a nivel de método el cual sería un añadido y se aplicaría como una seguridad en cascada, a continuación dicho acercamiento a la seguridad a nivel de métodos.

Falta la parte de métodos arli acá abajo

paso para liberar de aquí en adelante un proyecto en producción que tenga seguridad RBAC etapa b

En la librería se encuentra tanto el [Guard](#) como la configuración base para el uso de **spring security**, esto permite utilizar la lógica **pre/post authorize** además de la seguridad por medio de **httpSecurity** y **web security**(security filter chain que procesa cada solicitud y autentica el jwt y autoriza los accesos), estas dos características se utilizan en conjunto para lograr la seguridad de los microservicios, cuando se inicia un nuevo microservicio o proyecto se debe tener en cuenta los schemas mencionados a lo largo de la guía de seguridad (microservicios, y sus roles, controllers y sus roles y tipos endpoint y sus roles), pues en ellos se registrarán las autorizaciones.

“

- La seguridad es heredada por lo tanto actuará como operadores '**AND**'.
- Además como consideración se especifica una seguridad que en caso de no haber anotaciones **@PreAuthorize** los accesos serán para cualquier usuario con acceso al **microservicio**.
- Así como el **@PreAuthorize** se puede ejecutar a nivel de clase en la capa **controller**, puede ser ejecutado a nivel de método en la clase **controller** y en cualquier otra capa como la capa del servicio, sin embargo para garantizar la no ejecución de lógica y transacciones se sugiere utilizarse a nivel de **controller**.
- Para securizar un proyecto el ideal sería registrar primero el acceso a nivel de microservicio, luego registrar los controllers que requieren seguridad y que roles tendrán el acceso y en caso de necesitar un permiso específico para algún método registrar la acción o el permiso para el usuario deseado (lógicamente para una acción a nivel de método también requiere el acceso al controller de estar especificado y el acceso al microservicio por la naturaleza de la seguridad heredada).