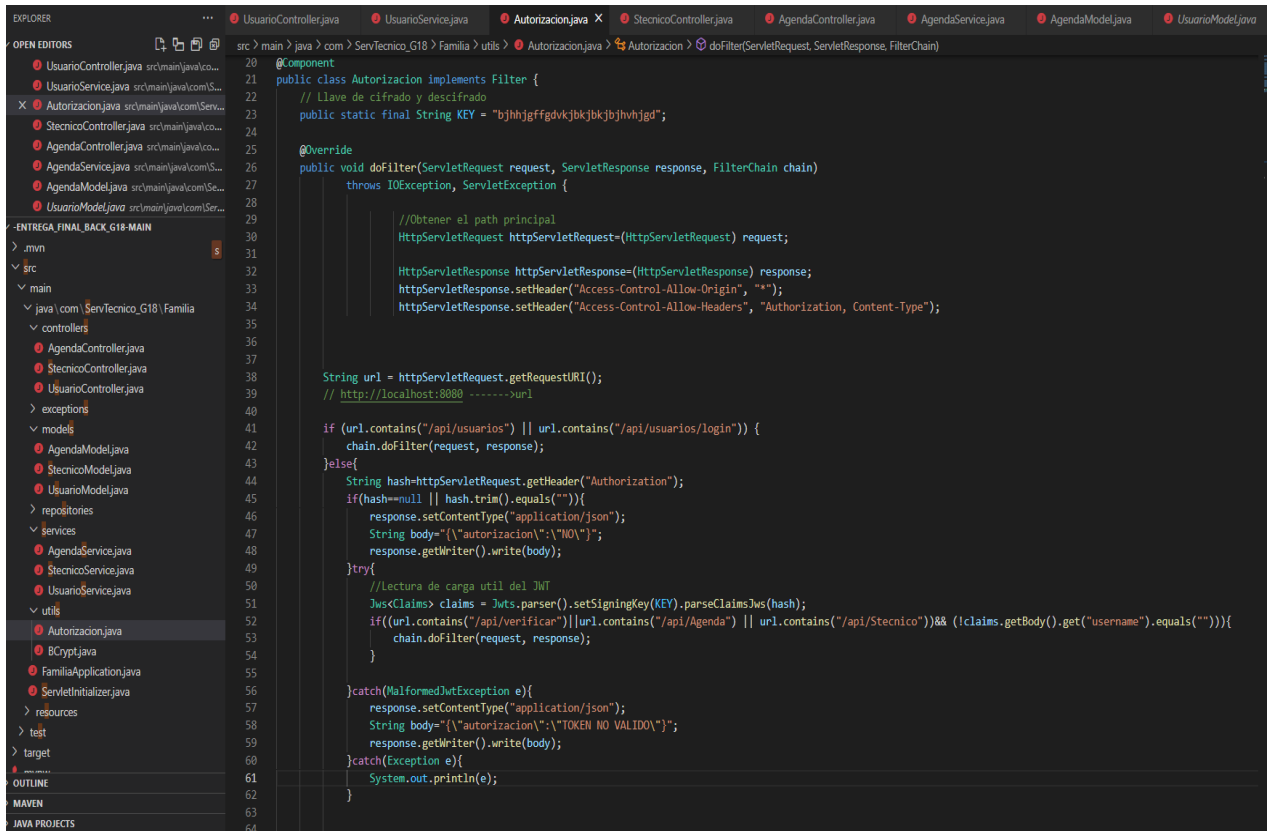


ANDREA MARCELA FERNANDEZ VARELA

ANDRIUN ESNAIDER SANTACRUZ NANDAR

## SPRINT 5

### 1. Clase o las clases Guard creadas para la validación de rutas



```
20 @Component
21 public class Autorizacion implements Filter {
22     // Llave de cifrado y descifrado
23     public static final String KEY = "bjhhjgffgdvkjbkjbkjbjhhvjgd";
24
25     @Override
26     public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
27         throws IOException, ServletException {
28
29         //Obtener el path principal
30         HttpServletRequest httpRequest=(HttpServletRequest) request;
31
32         HttpServletResponse httpResponse=(HttpServletResponse) response;
33         httpResponse.setHeader("Access-Control-Allow-Origin", "*");
34         httpResponse.setHeader("Access-Control-Allow-Headers", "Authorization, Content-Type");
35
36         String url = httpRequest.getRequestURI();
37         // http://localhost:8080 ----->url
38
39         if (url.contains("/api/usuarios") || url.contains("/api/usuarios/login")) {
40             chain.doFilter(request, response);
41         }else{
42             String hash=httpServletRequest.getHeader("Authorization");
43             if(hash==null || hash.trim().equals("")){
44                 response.setContentType("application/json");
45                 String body="{\"autorizacion\":\"NO\"}";
46                 response.getWriter().write(body);
47             }try{
48                 //Lectura de carga util del JWT
49                 Jws<Claims> claims = Jwts.parser().setSigningKey(KEY).parseClaimsJws(hash);
50                 if((url.contains("/api/verificar")||url.contains("/api/Agenda") || url.contains("/api/tecnico"))&& (!claims.getBody().get("username").equals(""))){
51                     chain.doFilter(request, response);
52                 }
53             }catch (MalformedJwtException e){
54                 response.setContentType("application/json");
55                 String body="{\"autorizacion\":\"TOKEN NO VALIDO\"}";
56                 response.getWriter().write(body);
57             }catch (Exception e){
58                 System.out.println(e);
59             }
60         }
61     }
62
63
64 }
```

```

const routes: Routes = [
  {
    path: '',
    component: InicioComponent,
  },
  {
    path: '/inicio',
    component: InicioComponent,
    canActivate: [AuthGuard],
    canLoad: [AuthGuard],
  },
  {
    path: '/servicio',
    component: ListarComponent,
    canActivate: [AuthGuard],
    canLoad: [AuthGuard],
  },
  {
    path: '/nuevo-servicio',
    component: AgregarComponent,
    canActivate: [AuthGuard],
    canLoad: [AuthGuard],
  },
  {
    path: '/nueva-agenda',
    component: CrearAgendaComponent,
    canActivate: [AuthGuard],
    canLoad: [AuthGuard],
  },
  {
    path: '/registro',
    component: RegistrarComponent,
  },
  {
    path: '**',
    redirectTo: ''
  }
];

```

## 2. Método que verifica la veracidad del Token

```

// Método para checar autenticidad
@PostMapping("/usuarios/login")
public ResponseEntity<Map<String, String>> login(@RequestBody UsuarioModel usuario){
    UsuarioModel usuarioAux=this.usuarioService.buscarUsername(usuario.getUsername());
    Map<String, String> respuesta = new HashMap<>();
    if(usuarioAux.getUsername()==null){
        respuesta.put("mensaje","usuario o contraseña incorrectos");
    }else{
        if(!BCrypt.checkpw(usuario.getPassword(), usuarioAux.getPassword())){
            respuesta.put("mensaje","usuario o contraseña incorrectos");
        }else{
            String hash = "";
            long tiempo = System.currentTimeMillis();
            if(usuarioAux.getId()!=null){
                hash=Jwts.builder()
                    .signWith(SignatureAlgorithm.HS256, Autorizacion.KEY)
                    .setSubject(usuarioAux.getNombre())
                    .setIssuedAt(new Date(tiempo))
                    .setExpiration(new Date(tiempo+900000))
                    .claim("username", usuarioAux.getUsername())
                    .claim("correo", usuarioAux.getCorreo())
                    .compact();
            }
            usuarioAux.setHash(hash);
            respuesta.put("mensaje","Se accedió correctamente");
            respuesta.put("token",hash);
            respuesta.put("id",usuarioAux.getId());
            respuesta.put("nombre",usuarioAux.getNombre());
            respuesta.put("correo",usuarioAux.getCorreo());
            respuesta.put("username",usuarioAux.getUsername());
        }
    }
    return ResponseEntity.ok(respuesta);
}

```

**3. Enlace al repositorio Github**

[https://github.com/AndreFer13/Entrega\\_Final\\_Front\\_Back](https://github.com/AndreFer13/Entrega_Final_Front_Back)

**4. Enlace a un video explicando el funcionamiento**

[https://github.com/AndreFer13/Entrega\\_Final\\_Front\\_Back/Video\\_Explicativo\\_Funcionalidad](https://github.com/AndreFer13/Entrega_Final_Front_Back/Video_Explicativo_Funcionalidad)