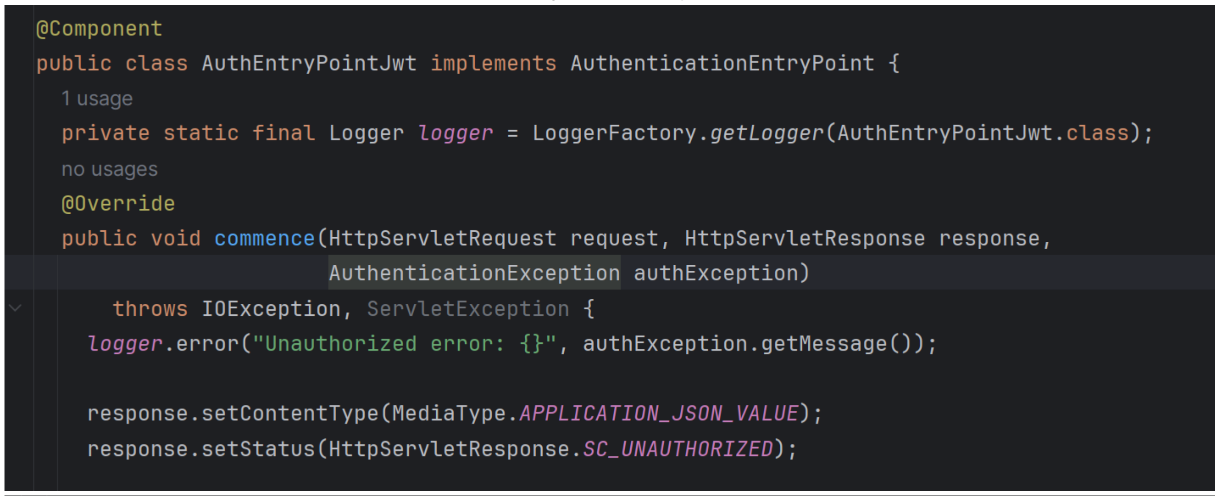
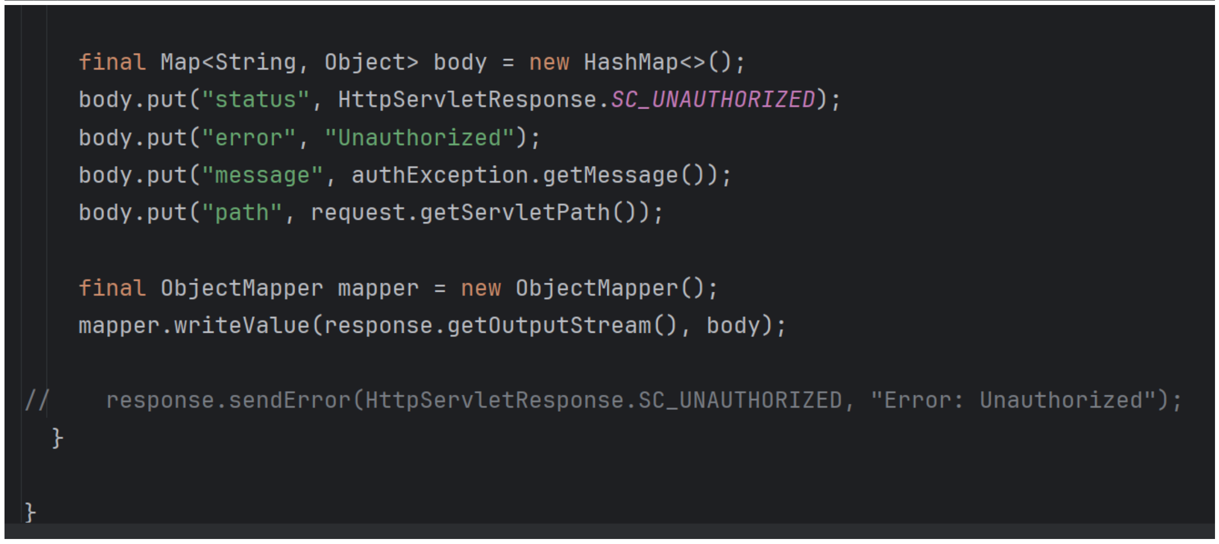
TP Sécurité

Partie 2:

1) Créer une Classe « **AuthEntryPointJwt »**

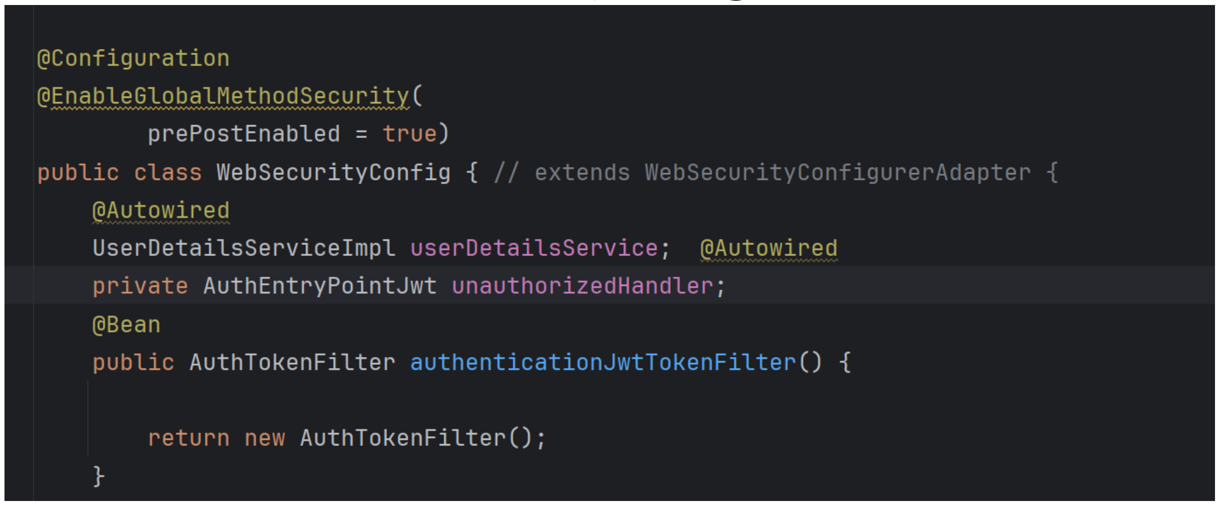


```
@Component
public class AuthEntryPointJwt implements AuthenticationEntryPoint {
  1 usage
  private static final Logger logger = LoggerFactory.getLogger(AuthEntryPointJwt.class);
  no usages
  @Override
  public void commence(HttpServletRequest request, HttpServletResponse response,
                       AuthenticationException authException)
      throws IOException, ServletException {
    logger.error("Unauthorized error: {}", authException.getMessage());

    response.setContentType(MediaType.APPLICATION_JSON_VALUE);
    response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
```

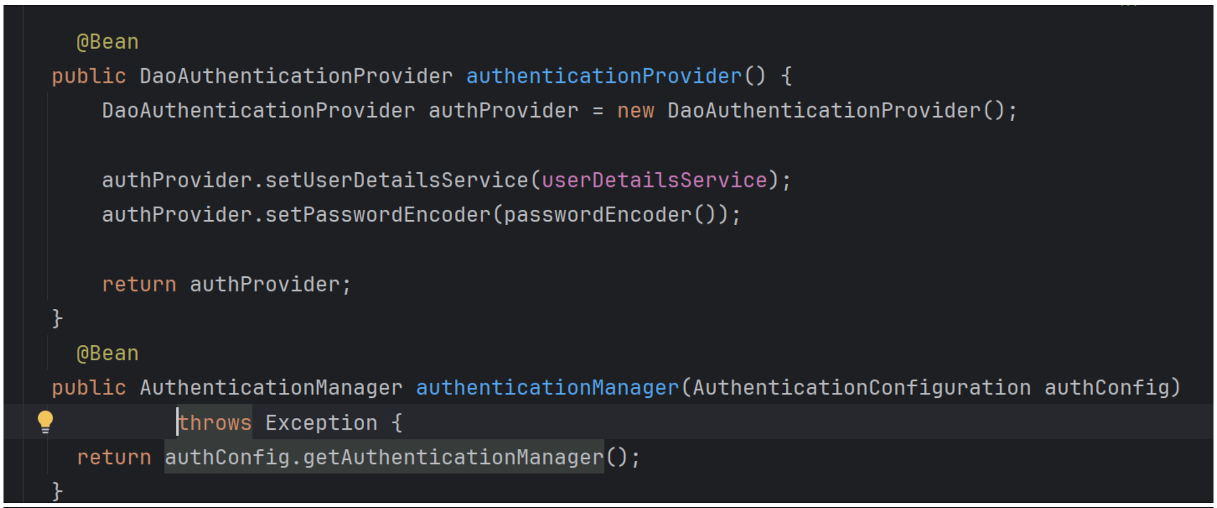


```

    final Map<String, Object> body = new HashMap<>();
    body.put("status", HttpServletResponse.SC_UNAUTHORIZED);
    body.put("error", "Unauthorized");
    body.put("message", authException.getMessage());
    body.put("path", request.getServletPath());

    final ObjectMapper mapper = new ObjectMapper();
    mapper.writeValue(response.getOutputStream(), body);

//    response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Error: Unauthorized");
  }

}
```

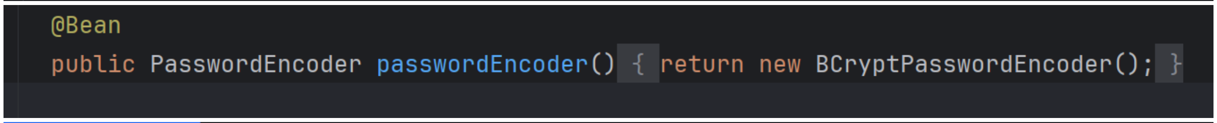
2) Créer une Classe « **WebSecurityConfig** »



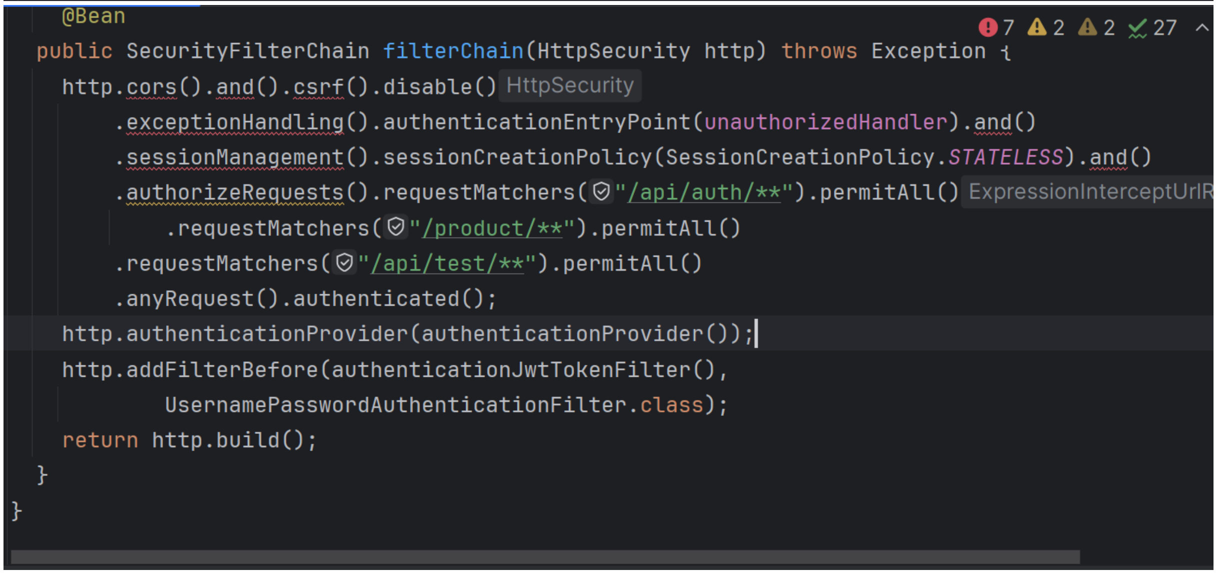
```
@Configuration
@EnableGlobalMethodSecurity(
        prePostEnabled = true)
public class WebSecurityConfig { // extends WebSecurityConfigurerAdapter {
    @Autowired
    UserDetailsServiceImpl userDetailsService;  @Autowired
    private AuthEntryPointJwt unauthorizedHandler;
    @Bean
    public AuthTokenFilter authenticationJwtTokenFilter() {

        return new AuthTokenFilter();
    }
```



```java
  @Bean
public DaoAuthenticationProvider authenticationProvider() {
    DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();

    authProvider.setUserDetailsService(userDetailsService);
    authProvider.setPasswordEncoder(passwordEncoder());

    return authProvider;
}
  @Bean
public AuthenticationManager authenticationManager(AuthenticationConfiguration authConfig)
          throws Exception {
  return authConfig.getAuthenticationManager();
}
```



```java
@Bean
public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }
```



```java
  @Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
  http.cors().and().csrf().disable() HttpSecurity
      .exceptionHandling().authenticationEntryPoint(unauthorizedHandler).and()
      .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()
      .authorizeRequests().requestMatchers("/api/auth/**").permitAll() ExpressionInterceptUrlR
          .requestMatchers("/product/**").permitAll()
      .requestMatchers("/api/test/**").permitAll()
      .anyRequest().authenticated();
  http.authenticationProvider(authenticationProvider());
  http.addFilterBefore(authenticationJwtTokenFilter(),
          UsernamePasswordAuthenticationFilter.class);
  return http.build();
}
}
```