------------------------------------------------------------------------------- --------------------------------------------------------------------------------

**NOUVEAU CHANGEMENT 2024 – SECURITY CONFIG**

------------------------------------------------------------------------------- --------------------------------------------------------------------------------

1. Ajouter et modifier dans  **AddProductComponent.ts**

@Configuration  
@EnableWebSecurity  
public class SecurityConfig {  
 @Autowired  
 AuthenticationManager authenticationManager;  
 @Bean  
 public AuthenticationManager authenticationManager(  
 HttpSecurity httpSecurity,  
 BCryptPasswordEncoder bCryptPasswordEncoder,  
 UserDetailsService userDetailsService  
 ) throws Exception {  
 AuthenticationManagerBuilder authManagerBuilder = httpSecurity.getSharedObject(AuthenticationManagerBuilder.class);  
 authManagerBuilder.userDetailsService(userDetailsService).passwordEncoder(bCryptPasswordEncoder);  
 return authManagerBuilder.build();  
 }  
 **@Bean("corsConfigurationSource")  
 public CorsConfigurationSource getCorsConfigurationSource() {  
 CorsConfiguration configuration = new CorsConfiguration();  
 configuration.setAllowedMethods(Arrays.*asList*("GET","POST","PUT","DELETE","OPTIONS"));  
 configuration.applyPermitDefaultValues();  
 configuration.addAllowedOrigin("http://localhost:4200");  
 configuration.setAllowedHeaders(List.*of*("\*"));  
 configuration.setExposedHeaders(Arrays.*asList*("Authorization"));  
 UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();  
 source.registerCorsConfiguration("/\*\*", configuration);  
 return source;  
 }**  
 @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception{  
 **httpSecurity  
 .csrf(csrf -> csrf.disable())  
 .cors(cors -> cors.configurationSource(getCorsConfigurationSource()))  
  
 .sessionManagement(management -> management.sessionCreationPolicy(SessionCreationPolicy.*STATELESS*))  
 .authorizeHttpRequests(  
 requests -> requests  
 .requestMatchers("/login").permitAll()  
 .requestMatchers(HttpMethod.*GET*, "/api/users/all").hasAuthority("ADMIN")  
 .requestMatchers(HttpMethod.*GET*, "/api/products/all").hasAnyAuthority("ADMIN", "CASHIER", "USER")  
 .requestMatchers(HttpMethod.*GET*, "/api/products/\*\*").hasAnyAuthority("ADMIN", "CASHIER", "USER")  
 .requestMatchers(HttpMethod.*GET*, "/api/categories/all").hasAnyAuthority("ADMIN", "CASHIER", "USER")  
 .requestMatchers(HttpMethod.*GET*, "/api/categories/\*\*").hasAnyAuthority("ADMIN", "CASHIER", "USER")  
 .requestMatchers(HttpMethod.*POST*, "/api/products/save").hasAuthority("CREATE")  
 .requestMatchers(HttpMethod.*DELETE*, "/api/products/delete/\*\*").hasAuthority("ADMIN")  
 .requestMatchers(HttpMethod.*PUT*, "/api/products/update").hasAuthority("ADMIN")  
 .anyRequest().authenticated()  
 )  
 .addFilterBefore(new JWTAuthenticationFilter(authenticationManager),UsernamePasswordAuthenticationFilter.class)  
 .addFilterBefore(new JWTAuthorizationFilter(), UsernamePasswordAuthenticationFilter.class);  
 return httpSecurity.build();**  
 }  
}