httpSecurity.csrf().disable()  
 .sessionManagement()  
 .sessionCreationPolicy(SessionCreationPolicy.*STATELESS*).and()  
  
 **.cors().configurationSource(new CorsConfigurationSource() {  
 @Override  
 public CorsConfiguration getCorsConfiguration(HttpServletRequest request) {  
 CorsConfiguration config = new CorsConfiguration();  
config.setAllowedOrigins(Collections.*singletonList*("http://localhost:4200"));  
config.setAllowedMethods(Collections.*singletonList*("\*"));  
 config.setAllowCredentials(true);  
config.setAllowedHeaders(Collections.*singletonList*("\*"));  
config.setExposedHeaders(Arrays.*asList*("Authorization"));  
 config.setMaxAge(3600L);  
 return config;  
 }  
 }).and()**  
 .authorizeHttpRequests()  
 .requestMatchers("/login").permitAll()

1. Ajouter une méthode dans **SecurityConfiguration.ts**

**@Bean("corsConfigurationSource")  
public CorsConfigurationSource getCorsConfigurationSource() {  
 CorsConfiguration configuration = new CorsConfiguration();  
 configuration.setAllowedMethods(Arrays.*asList*("GET","POST","PUT"));  
 configuration.applyPermitDefaultValues();  
 configuration.addAllowedOrigin("http://localhost:4200");  
 configuration.setAllowedHeaders(List.*of*("\*"));  
 UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();  
 source.registerCorsConfiguration("/\*\*", configuration);  
 return source;  
}**

1. Ajouter l’appel dans **securityFilterChain.ts**

.sessionManagement()  
.sessionCreationPolicy(SessionCreationPolicy.*STATELESS*).and()  
  
**.cors().configurationSource(getCorsConfigurationSource())**.and()