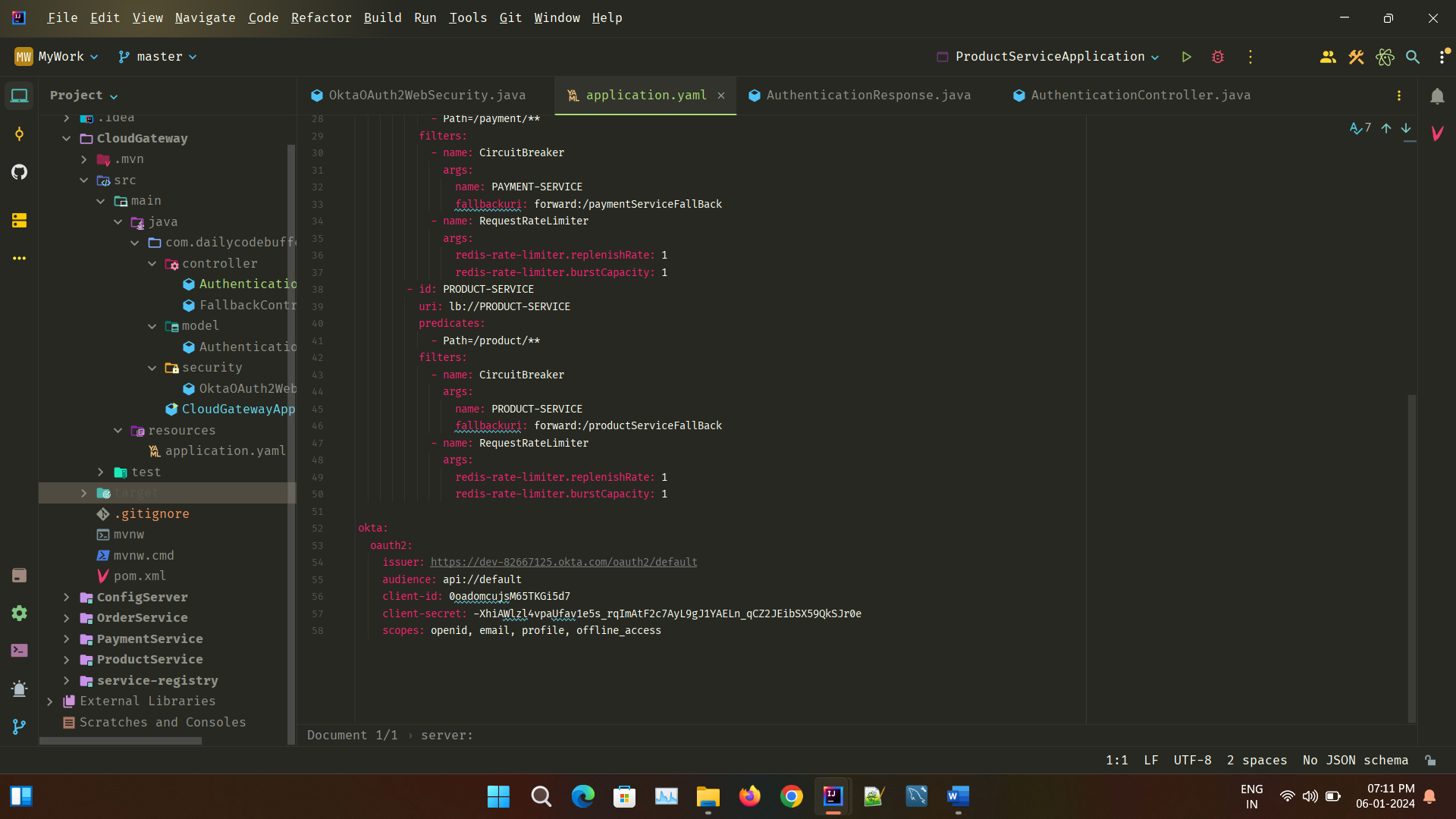
1. <https://start.spring.io> and add 2 dependencies in API Gateway:
   * 1. Spring Security
     2. OKTA
2. Application.yaml add these lines:

okta:  
 oauth2:  
 issuer: https://dev-82667125.okta.com/oauth2/default  
 audience: api://default  
 client-id: 0oadomcujsM65TKGi5d7  
 client-secret: -XhiAWlzl4vpaUfav1e5s\_rqImAtF2c7AyL9gJ1YAELn\_qCZ2JEibSX59QkSJr0e  
 scopes: openid, email, profile, offline\_access



1. Open okta developer-> login with google id-> CO LHS “Applications->Applications” tab.

Then CO My project i.e. “microservice”. Now take out these:

Client ID- 0oadomcujsM65TKGi5d7

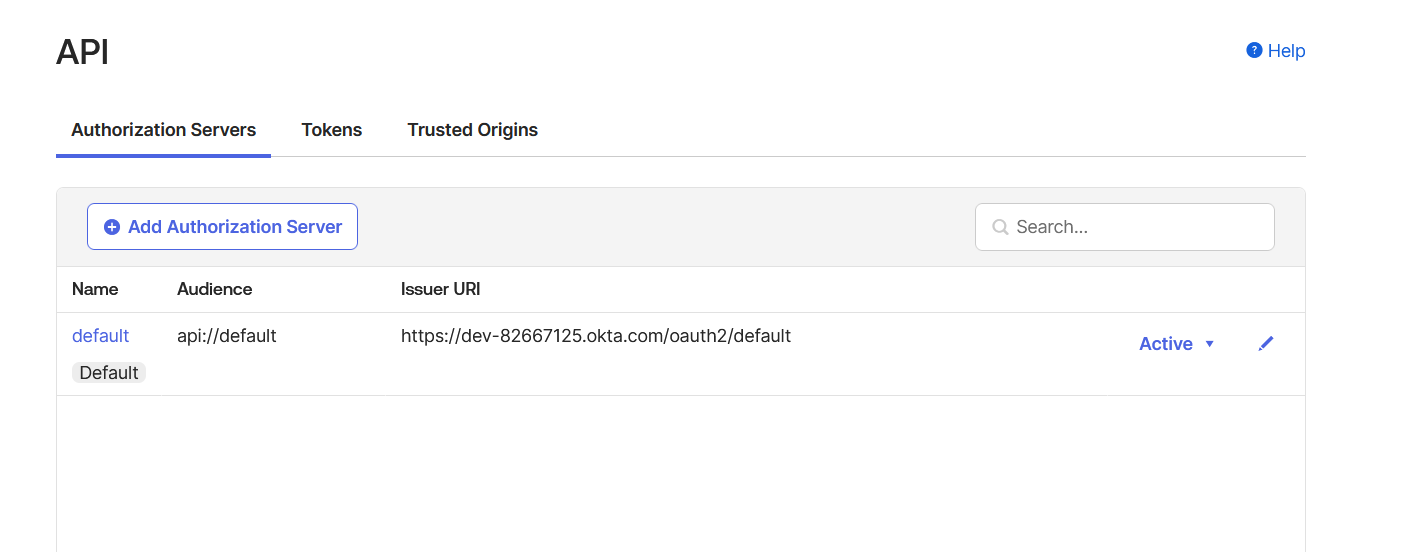
Client Secret- -XhiAWlzl4vpaUfav1e5s\_rqImAtF2c7AyL9gJ1YAELn\_qCZ2JEibSX59QkSJr0e

1. CO LHS “Security->API” then CO “default”. We get

Audience- api://default

Issuer- <https://dev-82667125.okta.com/oauth2/default>

scopes: openid, email, profile, offline\_access (Manually) {we can check all scopes in “API”}



1. Create a class in package security “OktaOAuth2WebSecurity.java”

*@Configuration  
@EnableWebFluxSecurity*public class *OktaOAuth2WebSecurity* {  
  
 *@Bean* public *SecurityWebFilterChain* securityFilterChain(*ServerHttpSecurity http*) {  
 *http* .authorizeExchange()  
 .anyExchange().authenticated()  
 .and()  
 .oauth2Login()  
 .and()  
 .oauth2ResourceServer()  
 .jwt();  
 return *http*.build();  
 }  
}

1. Create one Login Page with One API
2. After login into that page we will get:
   1. Refresh Token (Not that useful)
   2. **Access Token**
3. Create controller.AuthencationController.java

*@RestController  
@RequestMapping*("/authenticate")  
public class *AuthenticationController* {  
  
 *@GetMapping*("/login")  
 public *ResponseEntity*<*AuthenticationResponse*> login(  
 *@AuthenticationPrincipal OidcUser oidcUser*,   
 *Model model*,   
 *@RegisteredOAuth2AuthorizedClient*("okta") *OAuth2AuthorizedClient client*) {  
 *AuthenticationResponse* authenticationResponse = *AuthenticationResponse*.*builder*()  
 .userId(*oidcUser*.getEmail())  
 .accessToken(*client*.getAccessToken().getTokenValue())  
 .refreshToken(*client*.getRefreshToken().getTokenValue())  
 .expiresAt(*client*.getAccessToken().getExpiresAt().getEpochSecond())  
 .authorityList(*oidcUser*.getAuthorities()  
 .stream()  
 .map(*GrantedAuthority*::getAuthority)  
 .collect(*Collectors*.*toList*()))  
 .build();  
 return new ResponseEntity<>(authenticationResponse, *HttpStatus*.OK);  
 }  
}

1. Create a model class for AuthenticationResponse.java

*@Data  
@AllArgsConstructor  
@NoArgsConstructor  
@Builder*public class *AuthenticationResponse* {  
  
 private *String* userId;  
 private *String* accessToken;  
 private *String* refreshToken;  
 private long expiresAt;  
 private *Collection*<*String*> authorityList;  
}