За обеспечение безопасности системы отвечает сервис авторизации. Авторизации в системе основана на JWT-токенах, которые передаются между backend-сервером и frontend-сервером. JWT (JSON Web Token) - это стандарт для передачи информации в зашифрованном виде в виде объекта JSON (JavaScript Object Notation). JWT используется для авторизации и аутентификации пользователей на веб-страницах и приложениях. JWT состоит из трех частей: заголовка, полезной нагрузки (payload) и подписи. Логика шифрования содержится в классе JwtTokenProvider. Шифрование происходит с помощью алгоритма HS256.

@Component

public class JwtTokenProvider {

@Value("${jwt.token.secret}")

private String secret;

@Value("${jwt.token.expired}")

private long validityInMilliseconds;

@Autowired

private UserDetailsService userDetailsService;

@Bean

public BCryptPasswordEncoder passwordEncoder(){

BCryptPasswordEncoder bCryptPasswordEncoder = new BCryptPasswordEncoder();

return bCryptPasswordEncoder;

}

@PostConstruct

protected void init(){secret = Base64.getEncoder().encodeToString((secret.getBytes()));}

public String createToken(String username){

Claims claims = Jwts.claims().setSubject(username);

Date now = new Date();

Date validity = new Date(now.getTime() + validityInMilliseconds);

return Jwts.builder()

.setClaims(claims)

.setIssuedAt(now)

.setExpiration(validity)

.signWith(SignatureAlgorithm.HS256,secret)

.compact();

}

public Authentication getAuthentication(String token){

UserDetails userDetails = this.userDetailsService.loadUserByUsername(getUsername(token));

return new UsernamePasswordAuthenticationToken(userDetails,"",userDetails.getAuthorities());

}

public String getUsername(String token){

return Jwts.parser().setSigningKey(secret).parseClaimsJws(token).getBody().getSubject();

}

public String resolveToken(HttpServletRequest req){

String bearerToken = req.getHeader(("Authorization"));

if (bearerToken != null && bearerToken.startsWith("Bearer\_")){

return bearerToken.substring(7,bearerToken.length());

}

return null;

}

public boolean validateToken(String token) throws JwtAuthenticationException{

try {

Jws<Claims> claims = Jwts.parser().setSigningKey(secret).parseClaimsJws(token);

if(claims.getBody().getExpiration().before(new Date())){

return false;

}

return true;

} catch (JwtException | IllegalArgumentException e){

throw new JwtAuthenticationException("JWT token is expired or invalid");

}

}

}