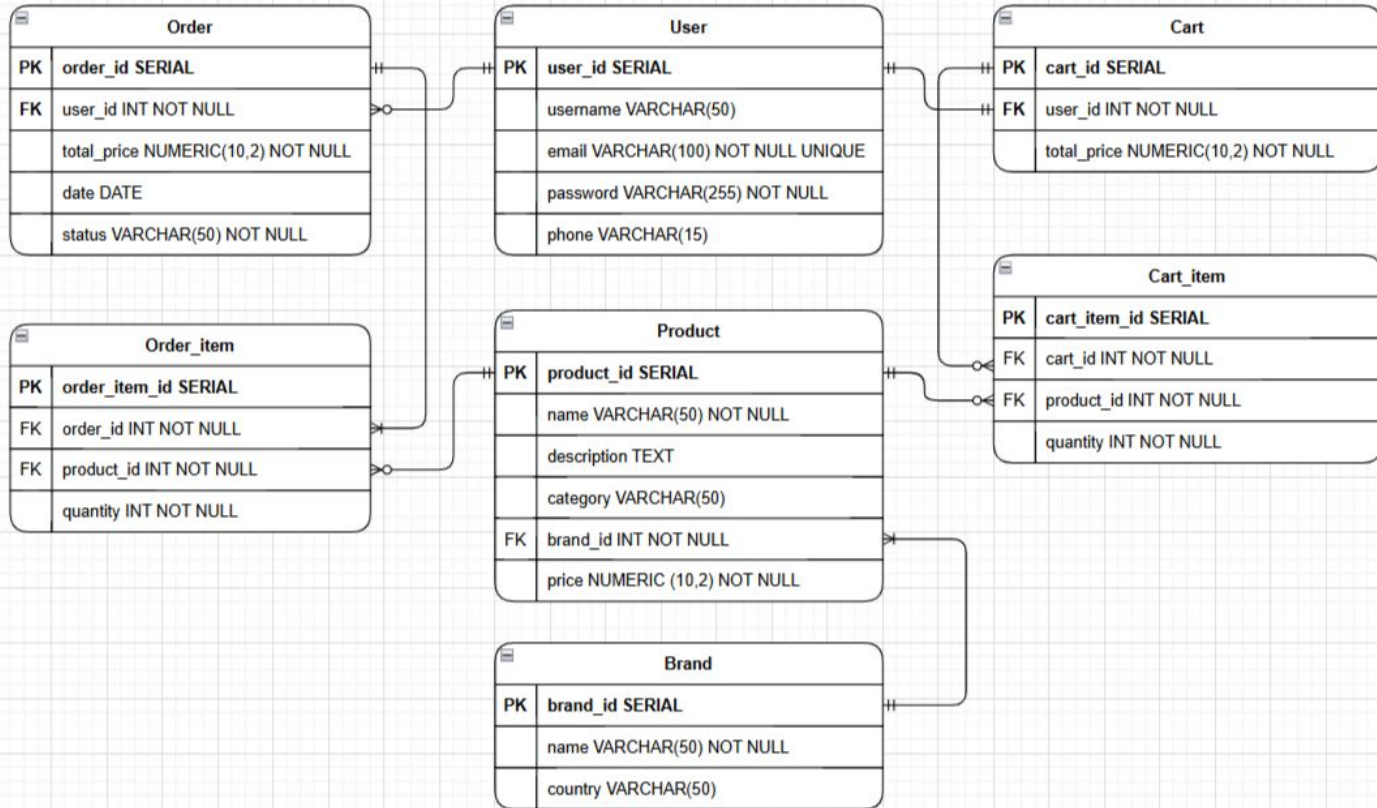




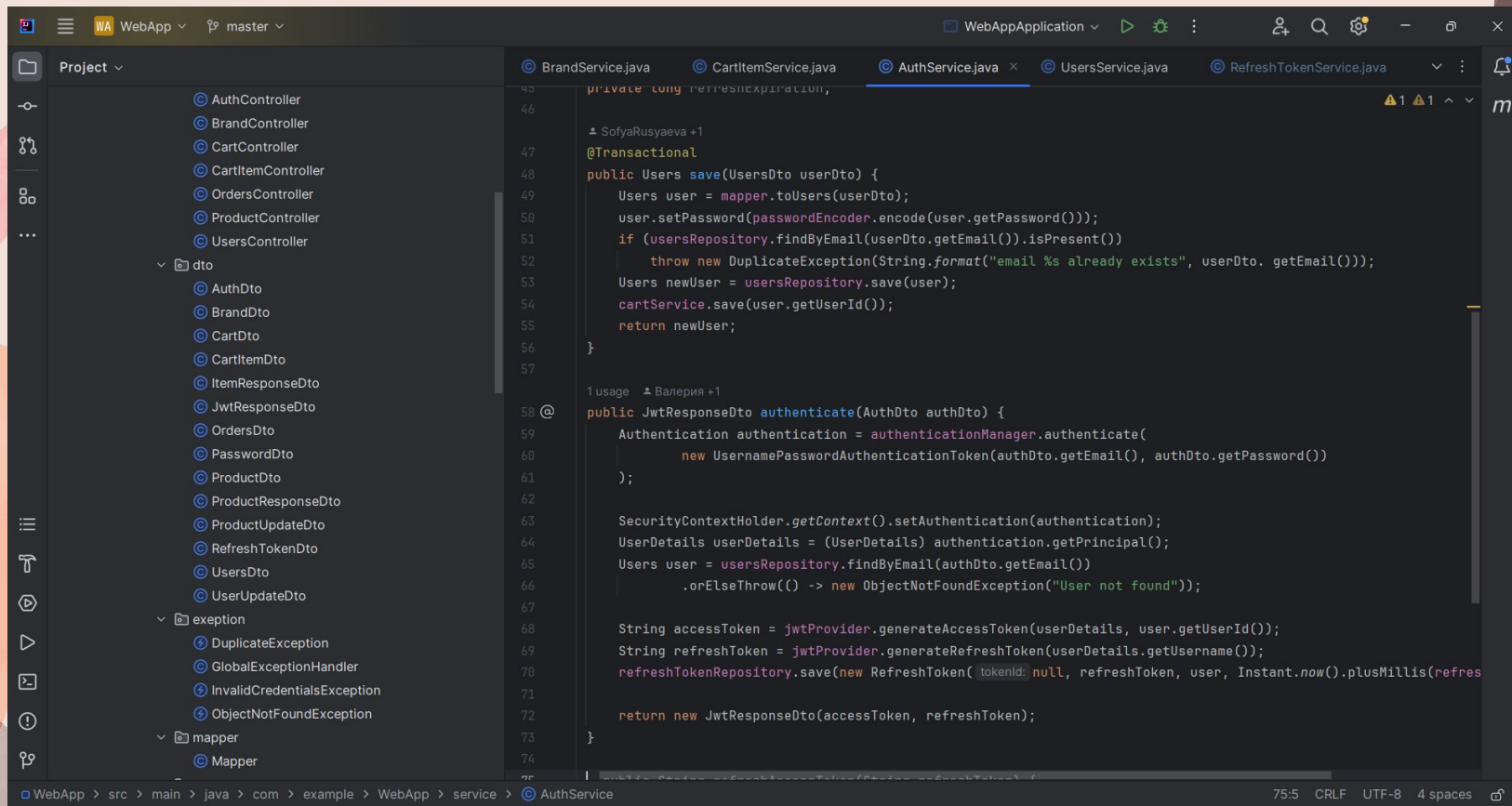
Интернет магазин косметики “Silver Pear”

Выполнили Жданова Валерия и
Русяева Софья 6301-010302D

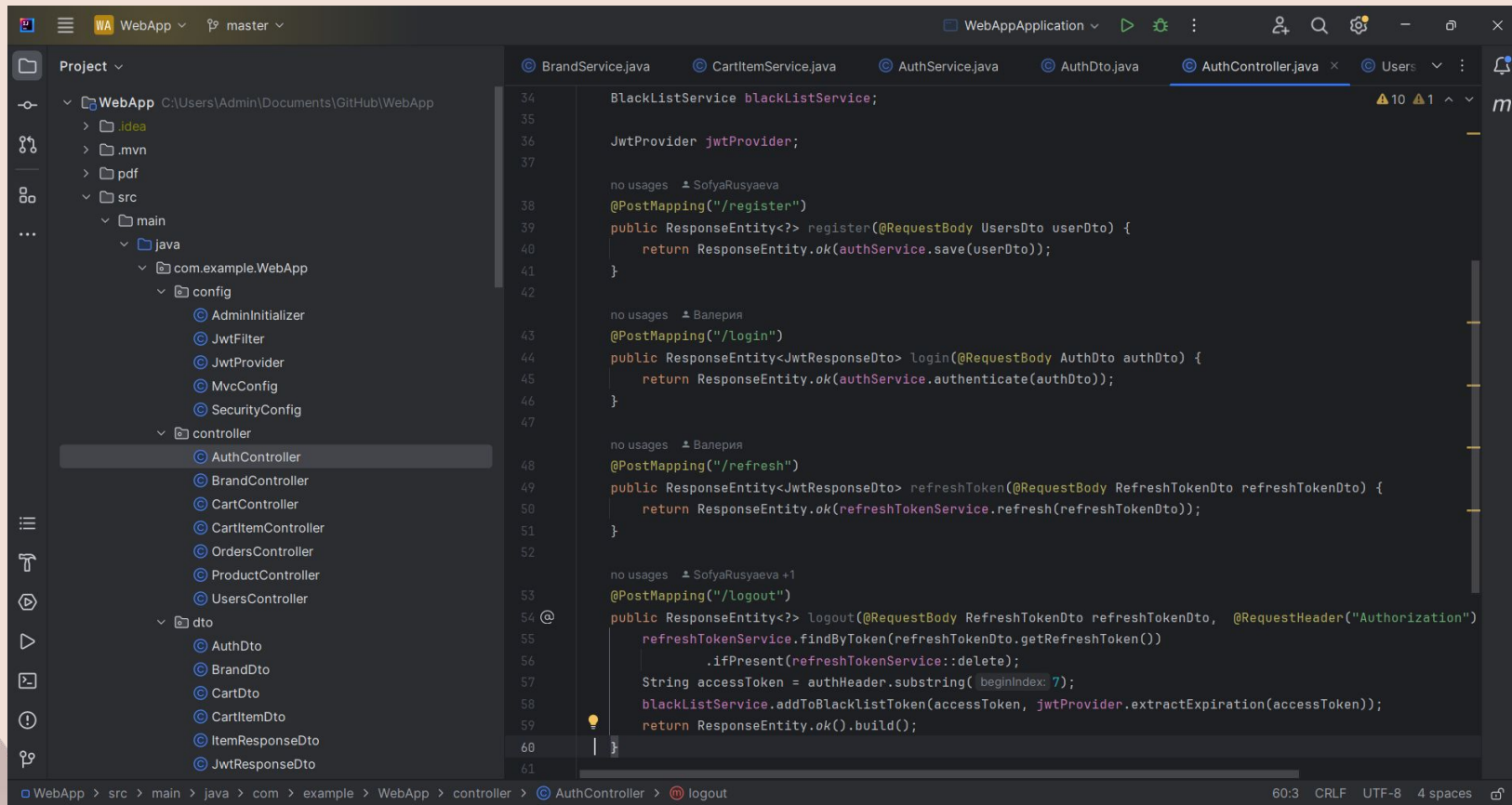
Схема БД



Реализация механизма



Реализация механизма



Генерация токена

The screenshot shows an IDE window with a project named 'WebApp' and a file explorer on the left. The file explorer shows the project structure, including 'src/main/java/com/example/WebApp/config'. The 'JwtProvider.java' file is selected, and its content is displayed in the editor. The code defines a 'JwtProvider' class with two methods: 'generateAccessToken' and 'generateRefreshToken'. The 'generateAccessToken' method takes 'UserDetails' and a 'Long' 'userId' as input and returns a 'String' token. It uses 'JwtBuilder' to create a token with claims, including the user's username, roles, and expiration time. The 'generateRefreshToken' method takes a 'String' 'username' as input and returns a 'String' token. It also uses 'JwtBuilder' to create a token with claims, including the user's username and expiration time. The code is annotated with '1 usage' and '2 usages' for the 'generateAccessToken' and 'generateRefreshToken' methods respectively. The status bar at the bottom shows the file path 'WebApp > src > main > java > com > example > WebApp > config > JwtProvider > accessExpiration' and the encoding 'UTF-8'.

```
31
32
33
34
35 @
36
37
38
39
40
41
42
43
44 //
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

1 usage
@Value("${jwt.refresh-expiration}")
long refreshExpiration;

2 usages 1 SofiaRusyaeva +1
public String generateAccessToken(UserDetails userDetails, Long userId) {
    Claims claims = Jwts.claims().setSubject(userDetails.getUsername());
    claims.put("roles", userDetails.getAuthorities().stream().map(GrantedAuthority::getAuthority).collect(Collectors.toList()));
    claims.put("userId", userId);

    return Jwts.builder()
        .setClaims(claims)
        .setSubject(userDetails.getUsername())
        .setIssuedAt(new Date())
        .setExpiration(new Date(System.currentTimeMillis() + accessExpiration))
        .signWith(Keys.hmacShaKeyFor(secret.getBytes()), SignatureAlgorithm.HS256)
        .compact();
}

2 usages 1 Валерия +1
public String generateRefreshToken(String username) {
    return Jwts.builder()
        .setSubject(username)
        .setIssuedAt(new Date())
        .setExpiration(new Date(System.currentTimeMillis() + refreshExpiration))
        .signWith(Keys.hmacShaKeyFor(secret.getBytes()), SignatureAlgorithm.HS256)
        .compact();
}
```

Содержимое токена:

- Subject (email пользователя)
- User ID
- Роль (USER/ADMIN)
- Время выдачи и expiration

Реализована через проверка валидности токена через JwtFilter.

Проверяется:

- Наличие токена в заголовке Authorization
- Валидность подписи
- Срок действия
- Наличие в черном списке (для logout)

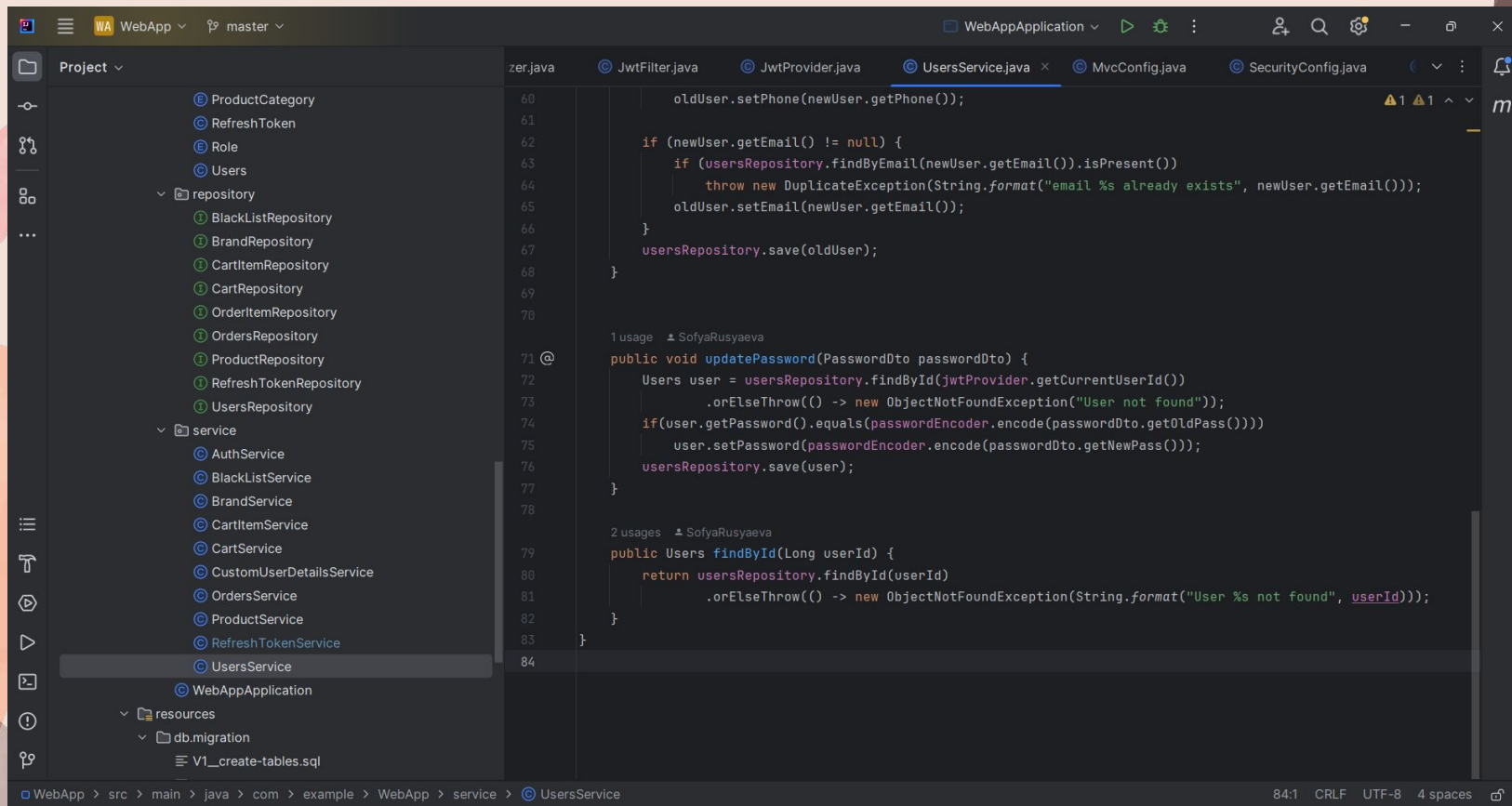
Валидация токена

The screenshot shows an IDE with a project named "WebApp" and a file named "JwtFilter.java" selected. The code in "JwtFilter.java" is as follows:

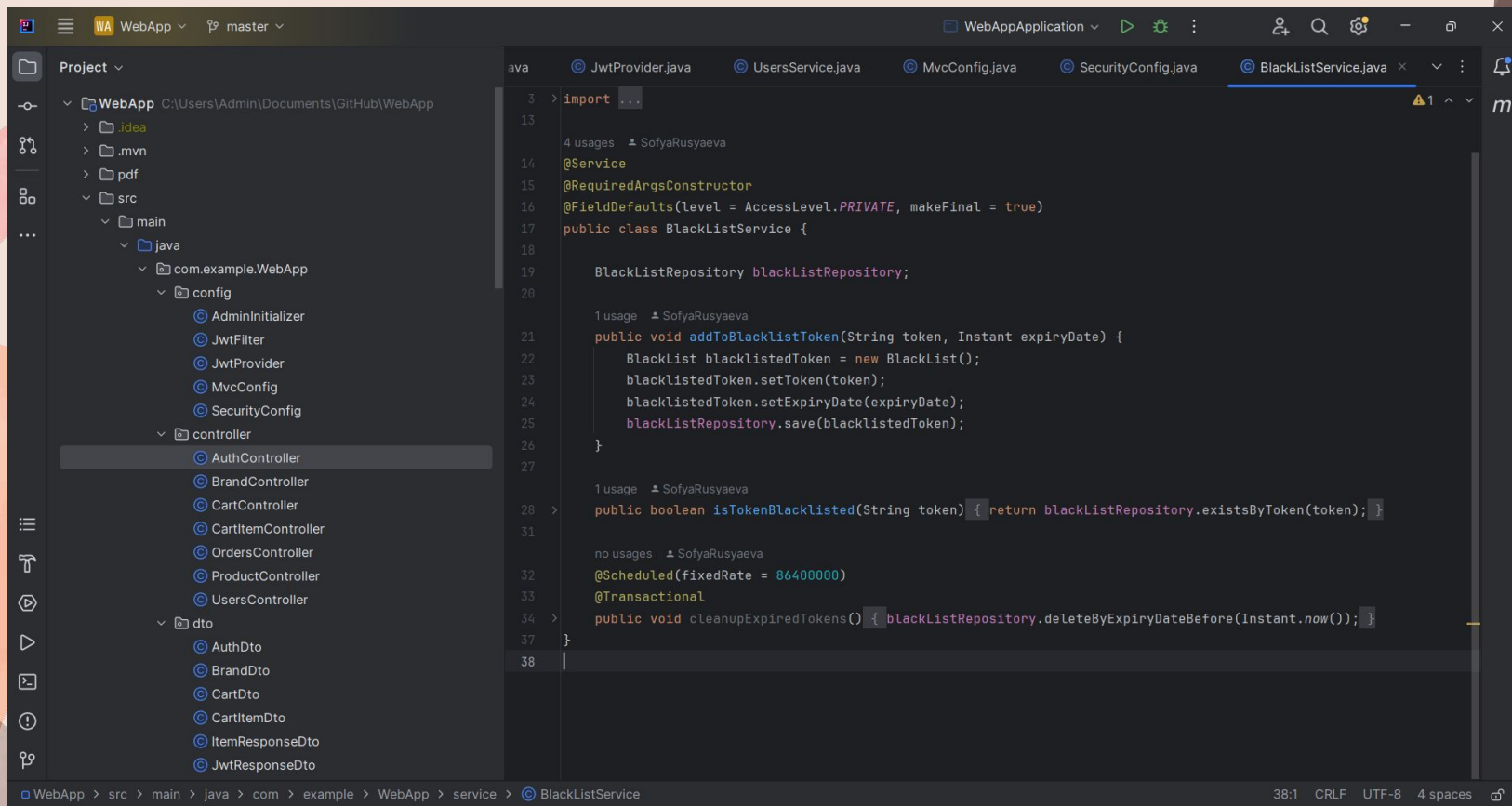
```
21
22 1 usage  ▲ SofiaRusyaeva +1
23 @Component
24 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
25 @RequiredArgsConstructor
26 public class JwtFilter extends OncePerRequestFilter {
27     JwtProvider jwtProvider;
28     CustomUserDetailsService userDetailsService;
29     BlackListService blackListService;
30
31     no usages  ▲ SofiaRusyaeva +1
32     @Override
33     protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,
34                                     FilterChain filterChain) throws ServletException, IOException {
35         final String authHeader = request.getHeader("Authorization");
36         if (authHeader != null && authHeader.startsWith("Bearer ")) {
37             String jwt = authHeader.substring(beginIndex: 7);
38             String email = jwtProvider.extractUsername(jwt);
39
40             if (email != null && SecurityContextHolder.getContext().getAuthentication() == null) {
41                 UserDetails userDetails = userDetailsService.loadUserByUsername(email);
42                 if (jwtProvider.validateToken(jwt)) {
43                     if (blackListService.isTokenBlacklisted(jwt))
44                         throw new AuthenticationException("You're authorised");
45                     UsernamePasswordAuthenticationToken authToken =
46                         new UsernamePasswordAuthenticationToken(userDetails, jwt, userDetails.getAuthorities());
47                     SecurityContextHolder.getContext().setAuthentication(authToken);
48                 }
49             }
50             filterChain.doFilter(request, response);
51         }
52     }
53 }
```

The IDE interface includes a project explorer on the left showing the file structure, a central editor window with the code, and a status bar at the bottom indicating the file path and encoding.

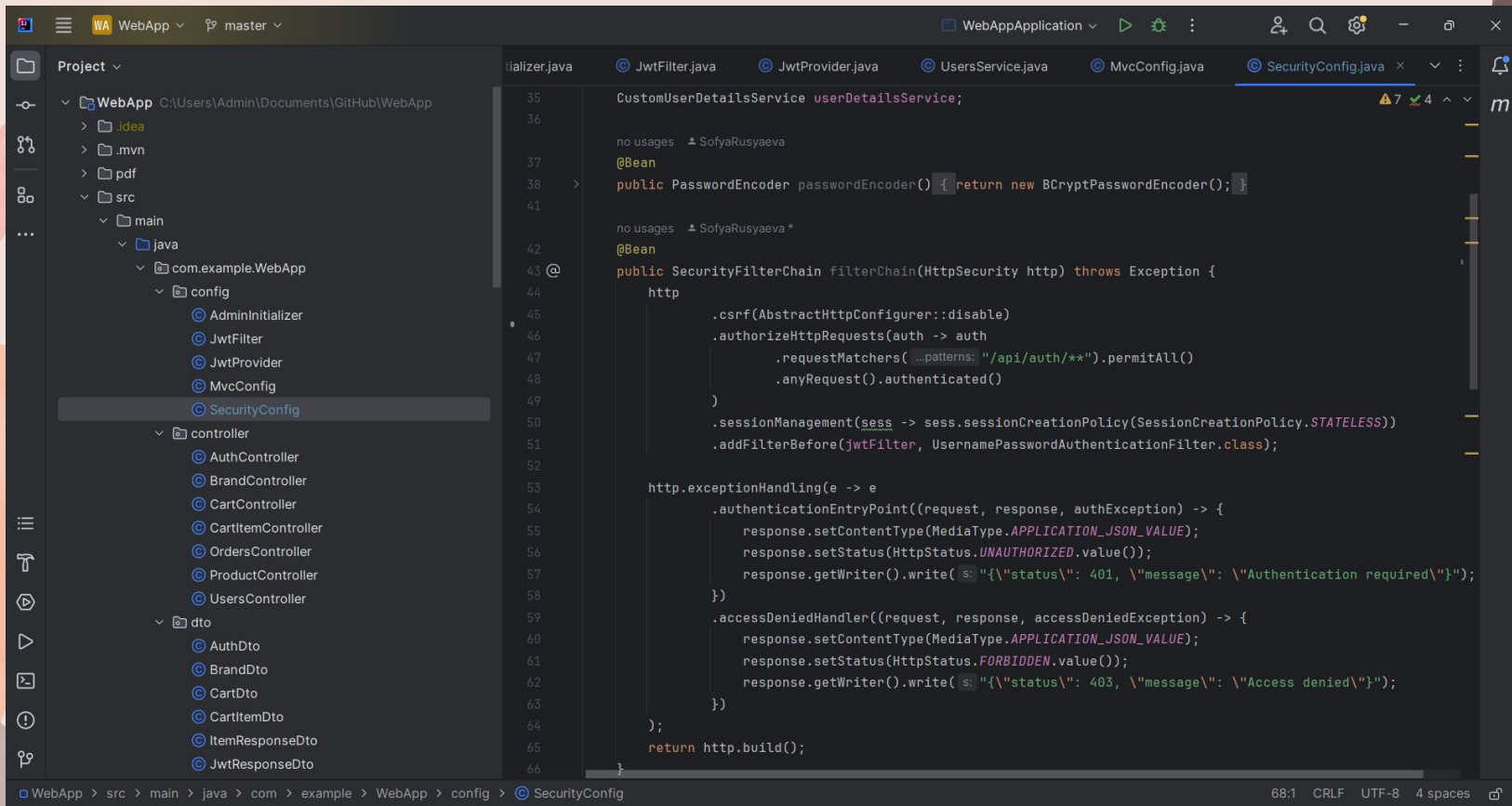
Изменение пароля



Выход из системы (отзыв токена)



Огр. доступа к опред. эндпоинтам



Проверка работы аутентификации

The screenshot displays the Postman interface with a workspace named "Shop_SilverPear". The left sidebar shows a "Collections" list with items like "Intro to Writing Tests", "Mock Data Generation", "New Collection", "Performance Testing", "Regression Testing", and "REST API basics: CRUD, test & variable". The main area shows a POST request to `http://localhost:8080/api/auth/login`. The request body is a JSON object with `email` and `password` fields. The response is a 200 OK status with a JSON body containing `accessToken` and `refreshToken`. The bottom status bar shows "Online", "Find and replace", "Console", "Postbot", "Runner", "Start Proxy", "Cookies", "Vault", "Trash", and a help icon.

Shop_SilverPear New Import

POST http://localhost:8080/api/auth/login

POST http://localhost:8080/api/auth/login

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "email": "kate@gmail.com",
3   "password": "123"
4 }
```

Body Cookies Headers (11) Test Results

200 OK · 634 ms · 700 B

{ } JSON Preview Visualize

```
1 {
2   "accessToken": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJrYXJlQGdtYWlsLmNvbSIsInJybGVzIjpbIlJPTeVfVWVhbnVlZjJkLzJ1c2V5SWQ1Ij01YsIm1hdCI6MTc0NDk4Nzc3NCwiZXhwIjoxNzQ0OTg4Njc0fQ.w6i2u9SyT87xmVhwCaxX6Z58GIxBTE_FH0482LctIeE",
3   "refreshToken": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJrYXJlQGdtYWlsLmNvbSIsIm1hdCI6MTc0NDk4Nzc3NSwiZXhwIjoxNzQ1MDc0MTc1fQ.MXHJzTQfvqL48dKXbCqzmpunjQj68RQ0dRovgV1TRLK"
4 }
```

Online Find and replace Console Postbot Runner Start Proxy Cookies Vault Trash

Проверка работы аутентификации

Shop_SilverPear

GET http://localhost:8080/api/shop/users/me

Send

Headers (9)

Key	Value	Description
<input checked="" type="checkbox"/> Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJrYXRIQGUdtYWwLmNvbSIsInJvbGVzIjpbIiJPTeVfVFNfUiJdLCJ1c2VySWQiOiJYsmlhdC16MTc0NDk4Nzc3NCwiZXBwIjoxNzQ0OTg4NjcwOQ.w6r2u95yT87xmVhwCaxX6Z5BGlxBTE_FHO482LctleE	
Key		Description

Body

200 OK • 181 ms • 503 B

```
{
  "userId": 6,
  "userName": "Kate",
  "email": "kate@gmail.com",
  "password": "$2a$10$LedOpAcJnr2Kv.ZR4r1N9eu4N0ZuUPPnbfs8fhLftA8DsQCELnfXw",
  "phone": "1234567890",
  "role": "ROLE_USER"
}
```

Online Find and replace Console Postbot Runner Start Proxy Cookies Vault Trash

Проверка работы аутентификации

The screenshot shows the Postman interface with a workspace named "Shop_SilverPear". The left sidebar contains a "Collections" section with a plus icon and a list of environments: "Intro to Writing Tests", "Mock Data Generation", "New Collection", "Performance Testing", "Regression Testing", and "REST API basics: CRUD, test & variable". The main panel displays an API test for the endpoint `http://localhost:8080/api/shop/users/me`. The request method is `GET`. The "Headers" tab is selected, showing a table with 9 headers. The "Body" tab is also visible, showing a JSON response.

Headers (9)

Key	Value	Description
Authorization	Bearer	
Key	Value	Description

Body

```
{
  "status": 401,
  "message": "Authentication required"
}
```

401 Unauthorized • 16 ms • 408 B

Проверка работы аутентификации

The screenshot shows the Postman interface with a workspace named "Shop_SilverPear". The active collection is "Intro to Writing Tests". The selected environment is "No environment". The API endpoint being tested is `http://localhost:8080/api/shop/users/6`. The request method is `GET`. The request headers are:

Key	Value	Description
<input checked="" type="checkbox"/> Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWwiOiJrYXRIQGdtYW...	
Key	Value	Description

The response status is `403 Forbidden` with a response time of 115 ms and a body size of 392 B. The response body is:

```
{
  "status": 403,
  "message": "You can't view this page"
}
```

The bottom status bar shows "Online", "Find and replace", "Console", "Postbot", "Runner", "Start Proxy", "Cookies", "Vault", "Trash", and "Help".