

# Современная разработка на Spring Boot: JWT

Конспект по разработке веб-приложений

5 декабря 2025 г.

## Этап 1: Модель пользователя и репозиторий

Создайте модель User, реализующую UserDetails, и репозиторий для неё.

```
1 @Entity
2 @Table(name = "users")
3 @Getter
4 @Setter
5 @NoArgsConstructor
6 @AllArgsConstructor
7 @Builder
8 @ToString
9 @EqualsAndHashCode
10 public class User implements UserDetails {
11
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Long id;
15
16     @Column(nullable = false, length = 32)
17     private String username;
18
19     @Column(nullable = false, length = 100)
20     private String passwordHash;
21
22     @Override
23     public Collection<? extends GrantedAuthority> getAuthorities() {
24         return List.of();
25     }
26
27     @Override
28     public String getPassword() {
29         return passwordHash;
30     }
```

```

31
32     @Override
33     public String getUsername() {
34         return username;
35     }
36
37     @Override
38     public boolean isAccountNonExpired() { return true; }
39
40     @Override
41     public boolean isAccountNonLocked() { return true; }
42
43     @Override
44     public boolean isCredentialsNonExpired() { return true; }
45
46     @Override
47     public boolean isEnabled() { return true; }
48 }
```

Listing 1: User.java

```

1 @Repository
2 public interface UserRepository extends JpaRepository<User, Long> {
3     Optional<User> findByUsername(String username);
4 }
```

Listing 2: UserRepository.java

## Этап 2: DTO для регистрации и входа

Создайте DTO-классы для регистрации и входа.

```

1 public record UserRegistrationDto(
2     @NotBlank
3     @Size(max=32, message = "Слишком длинная строка")
4     @Size(min=3, message = "Слишком короткая строка")
5     String username,
6
7     @Size(max = 32, message = "Слишком длинный пароль")
8     @Size(min = 8, message = "Слишком короткий пароль")
9     @Pattern(regexp = "(?=.*[A-Za-z]).*\d.*", message = "Должна быть хотя бы
10    одна буква, хотя бы одна цифра")
11     String password
11 ) {}
```

Listing 3: UserRegistrationDto.java

```

1 public record AuthRequestDto(
2     @NotBlank
3     @Size(max=32, message = "Слишком длинная строка")
4     @Size(min=3, message = "Слишком короткая строка")
5     String username,
6
7     @Size(max = 32, message = "Слишком длинный пароль")
8     @Size(min = 8, message = "Слишком короткий пароль")
9     @Pattern(regexp = "(?=.*[A-Za-z]).*\d.*", message = "Должна быть хотя бы
одна буква, хотя бы одна цифра")
10    String password
11 ) {}

```

Listing 4: AuthRequestDto.java

```

1 public record TokenResponseDto(String token) {}

```

Listing 5: TokenResponseDto.java

## Этап 3: Контроллер аутентификации

Добавьте контроллер AuthController с методами регистрации и входа.

```

1 @RestController
2 @RequestMapping("/api/auth")
3 @RequiredArgsConstructor
4 public class AuthController {
5
6     private final UserRepository userRepository;
7     private final PasswordEncoder passwordEncoder;
8     private final AuthenticationManager authenticationManager;
9     private final JwtUtil jwtUtil;
10
11     @PostMapping("/register")
12     public ResponseEntity<Void> registerUser(@RequestBody @Valid
UserRegistrationDto dto) {
13         var user = User.builder()
14             .username(dto.username())
15             .passwordHash(passwordEncoder.encode(dto.password()))
16             .build();
17         userRepository.save(user);
18         return ResponseEntity.noContent().build();
19     }
20
21     @PostMapping("/login")
22     public ResponseEntity<TokenResponseDto> login(@RequestBody AuthRequestDto
request) {

```

```

23     authenticationManager.authenticate(
24         new UsernamePasswordAuthenticationToken(request.username(),
25             request.password())
26     );
27     String token = jwtUtil.generateAccessToken(request.username());
28     return ResponseEntity.ok(new TokenResponseDto(token));
29 }

```

Listing 6: AuthController.java

## Этап 4: Настройка безопасности (без JWT-фильтра)

На этом этапе включите Spring Security и настройте базовую безопасность без токенов (только логин/пароль в памяти или через UserDetailsService).

Добавьте в build.gradle.kts:

```
implementation("org.springframework.boot:spring-boot-starter-security")
```

Создайте конфигурацию безопасности:

```

1 @Configuration
2 @RequiredArgsConstructor
3 public class SecurityConfig {
4
5     private final JwtAuthenticationEntryPoint jwtAuthenticationEntryPoint;
6     private final PasswordEncoder passwordEncoder;
7
8     @Bean
9     public PasswordEncoder passwordEncoder() {
10         return new BCryptPasswordEncoder();
11     }
12
13     @Bean
14     public AuthenticationManager authenticationManager(
15         AuthenticationConfiguration config) throws Exception {
16         return config.getAuthenticationManager();
17     }
18
19     @Bean
20     public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
21     Exception {
22         http
23             .csrf(csrf -> csrf.disable())
24             .authorizeHttpRequests(auth -> auth
25                 .requestMatchers("/api/auth/**").permitAll()
26                 .anyRequest().authenticated())

```

```

25         )
26         .exceptionHandling(exc -> exc.authenticationEntryPoint(
27             jwtAuthenticationEntryPoint))
28         .sessionManagement(sess -> sess.sessionCreationPolicy(
29             SessionCreationPolicy.STATELESS));
30     return http.build();
31 }

```

Listing 7: SecurityConfig.java – базовая версия

Создайте точку входа при ошибке аутентификации:

```

1 @Component
2 public class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint {
3     @Override
4     public void commence(HttpServletRequest request, HttpServletResponse
5             response,
6                     AuthenticationException authException) throws
7             IOException {
8             response.sendError(HttpStatus.SC_UNAUTHORIZED);
9         }
10 }

```

Listing 8: JwtAuthenticationEntryPoint.java

Теперь у вас должна работать регистрация и вход (проверьте через Postman).

## Этап 5: JWT-утилиты и сервис пользователей

Создайте утилиту для работы с JWT:

```

1 @Component
2 public class JwtUtil {
3
4     @Value("${jwt.access.expiration}") private Long accessExpiration;
5     @Value("${jwt.access.password}") private String secret;
6
7     private SecretKey getSigningKey() {
8         return Keys.hmacShaKeyFor(secret.getBytes());
9     }
10
11     public String getUsername(String token) {
12         try {
13             var claims = Jwts.parser()
14                 .verifyWith(getSigningKey())
15                 .build()
16                 .parseSignedClaims(token)
17                 .getPayload();

```

```

18         var expiration = claims.getExpiration();
19         if (!expiration.before(new Date())) {
20             return claims.getSubject();
21         }
22     } catch (Exception e) {
23         return null;
24     }
25     return null;
26 }
27
28 public String generateAccessToken(String subject) {
29     return Jwts.builder()
30         .subject(subject)
31         .issuedAt(new Date())
32         .expiration(new Date(System.currentTimeMillis() + accessExpiration
33         ))
34         .signWith(getSigningKey())
35         .compact();
36 }
```

Listing 9: JwtUtil.java

Создайте сервис загрузки пользователя:

```

1 @Service
2 @RequiredArgsConstructor
3 public class CustomUserDetailsService implements UserDetailsService {
4
5     private final UserRepository userRepository;
6
7     @Override
8     public UserDetails loadUserByUsername(String username) throws
9         UsernameNotFoundException {
10         return userRepository.findByUsername(username)
11             .orElseThrow(() -> new UsernameNotFoundException("Пользователь не
12         найден"));
13     }
14 }
```

Listing 10: CustomUserDetailsService.java

## Этап 6: JWT-фильтр

Добавьте фильтр, который извлекает токен из заголовка и устанавливает контекст аутентификации:

```
1 @Component
```

```

2 @RequiredArgsConstructor
3 public class JwtRequestFilter extends OncePerRequestFilter {
4
5     private final JwtUtil jwtUtil;
6     private final CustomUserDetailsService userDetailsService;
7
8     @Override
9     protected void doFilterInternal(HttpServletRequest request,
HttpServletResponse response,
10                                         FilterChain filterChain) throws
ServletException, IOException {
11
12         if (!request.getRequestURI().startsWith("/api/auth/")) {
13             String header = request.getHeader("Authorization");
14             if (header != null && header.startsWith("Bearer ")) {
15                 String jwt = header.substring(7);
16                 String username = jwtUtil.getUsername(jwt);
17                 if (username != null) {
18                     UserDetails userDetails = userDetailsService.
loadUserByUsername(username);
19                     UsernamePasswordAuthenticationToken authToken =
new UsernamePasswordAuthenticationToken(userDetails,
20                         null, userDetails.getAuthorities());
21                     authToken.setDetails(new WebAuthenticationDetailsSource().
buildDetails(request));
22                     SecurityContextHolder.getContext().setAuthentication(
authToken);
23                 }
24             }
25         }
26         filterChain.doFilter(request, response);
27     }
28 }
```

Listing 11: JwtRequestFilter.java

Добавьте фильтр в SecurityConfig:

```

1 private final JwtRequestFilter jwtRequestFilter;
2
3 // ...
4
5 http.addFilterBefore(jwtRequestFilter, UsernamePasswordAuthenticationFilter.
class);
```

Listing 12: SecurityConfig.java – обновлённая версия

## Этап 7: Обновление зависимостей

Добавьте в `build.gradle.kts` все JWT-зависимости и Resilience4j:

```
implementation("org.springframework.boot:spring-boot-starter-security")
implementation("io.jsonwebtoken:jjwt-api:0.13.0")
implementation("io.jsonwebtoken:jjwt-impl:0.13.0")
implementation("io.jsonwebtoken:jjwt-jackson:0.13.0")
```

## Этап 8: Привязка заметок к пользователю

Измените модель `Note`:

```
1 @ManyToOne
2 @JoinColumn(name = "user_id", nullable = false)
3 private User user;
```

Listing 13: Note.java

Обновите репозиторий:

```
1 List<Note> findByUser(User user);
```

Listing 14: NoteRepository.java

Создайте DTO для заметок:

```
1 public record NoteDto(Long id, String title, String content) {}
```

Listing 15: NoteDto.java

Измените `NoteController`: замените маппинг на `/api/notes` и добавьте получение пользователя из `SecurityContextHolder`:

```
1 var auth = SecurityContextHolder.getContext().getAuthentication();
2 var user = (User) auth.getPrincipal();
3
4 // В createNote:
5 note.setUser(user);
6
7 // В getAllNotes:
8 return noteRepository.findByUser(user).stream()
9     .map(note -> new NoteDto(note.getId(), note.getTitle(), note.getContent()))
10    .toList();
11
12
13 @PutMapping("/{id}")
14 public ResponseEntity<NoteDto> updateNote(@PathVariable Long id,
```

```

15     @RequestBody @Valid NoteCreateDto noteCreateDto) {
16         var auth = SecurityContextHolder.getContext().getAuthentication();
17         var user = (User)auth.getPrincipal();
18
19         Optional<Note> res = noteRepository.findById(id);
20         res = res.flatMap(note -> (note.getUser().equals(user))?Optional.of(
21             note):Optional.empty());
22
23         return res.map(existingNote -> {
24             existingNote.setTitle(noteCreateDto.title());
25             existingNote.setContent(noteCreateDto.content());
26             var updatedNote = noteRepository.save(existingNote);
27
28             return ResponseEntity.ok(new NoteDto(updatedNote.getId(),
29                 updatedNote.getTitle(),updatedNote.getContent()));
30         }).orElseGet(()->ResponseEntity.notFound().build());
31     }
32
33     @DeleteMapping("/{id}")
34     public ResponseEntity<Void> deleteNote(@PathVariable Long id) {
35         ResponseEntity<Void> response ;
36         var auth = SecurityContextHolder.getContext().getAuthentication();
37         var user = (User)auth.getPrincipal();
38
39         var res = noteRepository.findById(id);
40         res = res.flatMap(note -> (note.getUser().equals(user))?Optional.of(
41             note):Optional.empty());
42
43         if (res.isPresent()) {
44             noteRepository.deleteById(id);
45             response = ResponseEntity.noContent().build();
46         } else {
47             response = ResponseEntity.notFound().build();
48         }
49         return response;
50     }

```

Listing 16: NoteController.java – ключевые фрагменты

## Этап 9: Настройка application.properties

Добавьте в application.properties:

```

jwt.access.expiration=3600000
jwt.access.password=CnwyutT5nP424AS7jg5rJteZYJuY5f5rAsHUXuRJ
logging.level.org.springframework.security=DEBUG

```