Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Итерация №1

По дисциплине «Рефакторинг баз данных и приложений»

Группа: P3412

Выполнили: Балин А.А., Кобелев Р. П.

Проверил: Логинов И.П.

Санкт-Петербург
2025г.

Оглавление

Логирование бэкенда

Логирование бекенда (добавление логов в ключевых местах, настройка уровней логирования). Поможет в отладке и мониторинге работы сервиса на следующих этапах.

Коммиты: 1, 2, 3, 4

Настройка метрик и дашбордов с помощью Prometheus/Micrometer и Grafana

Настройка метрик и дашбордов с помощью Prometheus/Micrometer и Grafana. Позволит отслеживать производительность и состояние системы в реальном времени. Сможем легко проверить, улучшилась ли производительность после оптимизаций на следующих этапах.

Коммиты: 1, 2, 3, 4

Вынесение задаваемых параметров в конфигурационные файлы

В этом пункте нашей задачей было вынести все магические числа и строки в конфигурационные файлы. Это облегчит поддержку и изменение настроек системы в будущем. Позволит задавать параметры для докера.

На этом этапе я взял все параметры, которые в будущем могут перенастраиваться, из классов spring и вынес их в файл application.properties. Также для некоторых параметров, как секретный ключ и порт, я значения убрал в .env файл для повышения безопасности.

Коммиты: 1

DIFF

git log --oneline 'ac73189220059f7880d62ab4f9cc091c635df818^..276c8eb735fc2193ad5a113f44ef00c19524a871'

```
276c8eb (HEAD -> main, origin/main, origin/HEAD) log fix + ignore logs in git
76445b3 service logging added
becdbef log config for different components
6185046 controller logginh
c075252 (origin/step1-refactoring, step1-refactoring) Merge branch 'grafana' into step1-refactoring
a7b2756 (origin/grafana, grafana) app config for micrometer
b7de8bd dashboard with metrics
dcbebdb configs for dashboards
a0d0014 readme and docker for grafana & prometheus
ac73189 add more configurable data of the project
```

git diff 'ac73189220059f7880d62ab4f9cc091c635df818^..276c8eb735fc2193ad5a113f44ef00c19524a871' –stat

```
.docker/docker-compose.yaml                                        | 46 ++++++++++++
.docker/grafana/dashboards/spring-boot-overview.json               | 113 ++++++++++++++++++++++++++++++
.docker/grafana/provisioning/dashboards/dashboards.yml             | 12 ++++
.docker/grafana/provisioning/datasources/datasource.yml            | 10 +++
.docker/prometheus/prometheus.yml                                  | 18 +++++
.gitignore                                                         |  4 ++
README.md                                                          | 54 ++++++++++++++
backend/pom.xml                                                    |  8 +++
backend/src/main/java/itmo/is/cw/GuitarMatchIS/config/WebSocketConfig.java        | 20 +++++-
backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ArticleController.java  |  9 +++
backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/AuthController.java     |  4 ++
backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/BrandController.java    |  4 ++
backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/FeedbackController.java |  6 ++
backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ForumPostController.java |  3 +
backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ForumTopicController.java |  9 +++
backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/MusicianController.java |  14 ++++
backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ProductController.java   |  11 +++
backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ShopController.java      |  5 ++
backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/UserController.java      | 12 ++++
backend/src/main/java/itmo/is/cw/GuitarMatchIS/dto/AuthResponseDTO.java            |  3 +-
backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/SecurityConfig.java        |  8 ++-
backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/jwt/JwtUtils.java          | 26 +++++--
backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ArticleService.java         | 65 ++++++++++++-----
backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/AuthService.java            | 19 +++--
backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/BrandService.java           |  9 ++-
backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/FeedbackService.java        | 253 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++---------------
backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ForumPostService.java       | 20 ++++--
backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ForumTopicService.java      | 50 ++++++++++---
backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/MusicianService.java        | 66 ++++++++++++-----
backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ProductService.java         | 33 +++++++--
backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ShopService.java            | 15 +++-
backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/UserService.java            | 52 ++++++++++----
backend/src/main/resources/application.properties                  | 32 ++++++++-
backend/src/main/resources/logback-spring.xml                      | 44 ++++++++++++
```

```
 34 files changed, 858 insertions(+), 199 deletions(-)
```

git show ac73189

```
commit ac73189220059f7880d62ab4f9cc091c635df818
Author: Romariok <ronya-ko@yandex.ru>
Date:   Fri Nov 7 21:18:59 2025 +0300

    add more configurable data of the project

diff --git a/.gitignore b/.gitignore
index c6f9a44..1c4dca5 100644
--- a/.gitignore
+++ b/.gitignore
@@ -1 +1,4 @@
 .vscode/settings.json
+*.env
+**/*.env
+.DS_Store
\ No newline at end of file
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/config/WebSocketConfig.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/config/WebSocketConfig.java
index 9faa403..a3eea5e 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/config/WebSocketConfig.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/config/WebSocketConfig.java
@@ -1,24 +1,38 @@
 package itmo.is.cw.GuitarMatchIS.config;

+import org.springframework.beans.factory.annotation.Value;
 import org.springframework.context.annotation.Configuration;
 import org.springframework.messaging.simp.config.MessageBrokerRegistry;
 import org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker;
 import org.springframework.web.socket.config.annotation.StompEndpointRegistry;
 import org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer;

+import java.util.List;
+
 @Configuration
 @EnableWebSocketMessageBroker
 public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {
+
+    @Value("#{'${app.websocket.broker-destinations:/topic,/feedbacks}'.split(',')}")
+    private List<String> brokerDestinations;
+
+    @Value("${app.websocket.application-destination-prefix:/app/**}")
+    private String applicationDestinationPrefix;
+
+
+    @Value("#{'${app.websocket.allowed-origins:http://localhost:5173}'.split(',')}")
+    private List<String> wsAllowedOrigins;
+
     @SuppressWarnings("null")
     @Override
     public void configureMessageBroker(MessageBrokerRegistry registry) {
-        registry.enableSimpleBroker("/topic", "/feedbacks");
```

```
-       registry.setApplicationDestinationPrefixes("/app/**");
+       registry.enableSimpleBroker(brokerDestinations.toArray(new String[0]));
+       registry.setApplicationDestinationPrefixes(applicationDestinationPrefix);
    }

    @SuppressWarnings("null")
    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
-       registry.addEndpoint("/ws").setAllowedOrigins("http://localhost:5173").withSockJS();
+       registry.addEndpoint("/ws").setAllowedOrigins(wsAllowedOrigins.toArray(new String[0])).withSockJS();
    }
}
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/dto/AuthResponseDTO.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/dto/AuthResponseDTO.java
index 7d9479f..520b98b 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/dto/AuthResponseDTO.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/dto/AuthResponseDTO.java
@@ -13,7 +13,8 @@ public class AuthResponseDTO {
    private String token;
    private final String tokenType = "Bearer ";

-   public AuthResponseDTO(String username, Boolean isAdmin, Integer subscriptions, LocalDateTime createdAt, String token) {
+   public AuthResponseDTO(String username, Boolean isAdmin, Integer subscriptions, LocalDateTime createdAt,
+           String token) {
        this.username = username;
        this.isAdmin = isAdmin;
        this.subscriptions = subscriptions;
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/SecurityConfig.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/SecurityConfig.java
index f1a256d..70a83c0 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/SecurityConfig.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/SecurityConfig.java
@@ -1,5 +1,6 @@
 package itmo.is.cw.GuitarMatchIS.security;

+import org.springframework.beans.factory.annotation.Value;
 import org.springframework.context.annotation.Bean;
 import org.springframework.context.annotation.Configuration;
 import org.springframework.http.HttpMethod;
@@ -32,12 +33,16 @@ public class SecurityConfig {

    private final AuthUserDetailsService userDetailsService;

+   @Value("#{'${app.cors.allowed-origins:http://localhost:5173}'.split(',')}")
+   private List<String> corsAllowedOrigins;
+
+
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        return http.csrf(AbstractHttpConfigurer::disable)
            .cors(cors -> cors.configurationSource(request -> {
                CorsConfiguration corsConfiguration = new CorsConfiguration();
-               corsConfiguration.setAllowedOrigins(List.of("http://localhost:5173"));
+               corsConfiguration.setAllowedOrigins(corsAllowedOrigins);
                corsConfiguration.setAllowedMethods(List.of("POST", "GET", "PUT", "PATCH","DELETE", "OPTIONS"));
                corsConfiguration.setAllowedHeaders(List.of("Content-Type", "Authorization", "X-Requested-With", "from", "size"));
```

```
            corsConfiguration.setAllowCredentials(true);
        diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/jwt/JwtUtils.java
b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/jwt/JwtUtils.java
        index f41a5fc..cab3b18 100644
        --- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/jwt/JwtUtils.java
        +++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/jwt/JwtUtils.java
        @@ -6,18 +6,24 @@ import jakarta.servlet.http.HttpServletRequest;

         import org.slf4j.Logger;
         import org.slf4j.LoggerFactory;
        +import org.springframework.beans.factory.annotation.Value;
         import org.springframework.stereotype.Component;
         import org.springframework.util.StringUtils;

        +import java.nio.charset.StandardCharsets;
         import java.security.Key;
        +import java.security.MessageDigest;
         import java.util.Date;

         @Component
         public class JwtUtils {
            private static final Logger logger = LoggerFactory.getLogger(JwtUtils.class);

        -   private static final Key secretKey = Keys.secretKeyFor(SignatureAlgorithm.HS256);
        -   private static final long expirationMs = 3600000; // Время действия токена: 1 час
        +   @Value("${app.security.jwt.secret}")
        +   private String jwtSecret;
        +
        +   @Value("${app.security.jwt.expiration-ms:3600000}")
        +   private long expirationMs;

            public String generateJwtToken(String username) {
                Date now = new Date();
        @@ -27,7 +33,7 @@ public class JwtUtils {
                        .setSubject(username)
                        .setIssuedAt(now)
                        .setExpiration(expiration)
        -               .signWith(secretKey)
        +               .signWith(getSecretKey())
                        .compact();
            }

        @@ -60,7 +66,19 @@ public class JwtUtils {
            }

            private Jws<Claims> getParsedToken(String authToken) {
        -       return Jwts.parserBuilder().setSigningKey(secretKey).build().parseClaimsJws(authToken);
        +       return Jwts.parserBuilder().setSigningKey(getSecretKey()).build().parseClaimsJws(authToken);
        +   }
        +
        +   private Key getSecretKey() {
        +       byte[] keyBytes = jwtSecret.getBytes(StandardCharsets.UTF_8);
        +       if (keyBytes.length < 32) {
        +           try {
        +               keyBytes = MessageDigest.getInstance("SHA-256").digest(keyBytes);
        +           } catch (Exception e) {
```

```
+                throw new IllegalStateException("Failed to initialize JWT secret key", e);
+            }
+        }
+        return Keys.hmacShaKeyFor(keyBytes);
    }

    public String parseJwt(HttpServletRequest request) {
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ArticleService.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ArticleService.java
index fe88a56..edb6437 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ArticleService.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ArticleService.java
@@ -7,6 +7,7 @@ import java.time.LocalDateTime;
 import java.util.Comparator;
 import java.util.List;

+import org.springframework.beans.factory.annotation.Value;
 import org.springframework.data.domain.PageRequest;
 import org.springframework.data.domain.Pageable;
-    List<Feedback> feedbacks = feedbackRepository.findByProductId(productId, pageable).getContent();
-
-    return feedbacks
-        .stream()
-        .map(this::convertToDTO)
-        .sorted(new Comparator<FeedbackDTO>() {
-          @Override
-          public int compare(FeedbackDTO o1, FeedbackDTO o2) {
-            return o1.getId().compareTo(o2.getId());
-          }
-        })
-        .toList();
-
- }
-
- public List<FeedbackDTO> getFeedbackByArticleId(Long articleId, int from, int size) {
-    Pageable pageable = Pagification.createPageTemplate(from, size);
-
-    List<Feedback> feedbacks = feedbackRepository.findByArticleId(articleId, pageable).getContent();
-
-    return feedbacks
-        .stream()
-        .map(this::convertToDTO)
-        .sorted(new Comparator<FeedbackDTO>() {
-          @Override
-          public int compare(FeedbackDTO o1, FeedbackDTO o2) {
-            return o1.getId().compareTo(o2.getId());
-          }
-        })
-        .toList();
- }
-
- @Transactional
- public Boolean addProductFeedback(CreateProductFeedbackDTO feedbackDTO, HttpServletRequest request) {
-    Product product = productRepository.findById(feedbackDTO.getProductId())
-        .orElseThrow(() -> new ProductNotFoundException(
-            String.format("Product with id %s not found", feedbackDTO.getProductId())));
```

```
-
-     User user = findUserByRequest(request);
-     feedbackRepository.addProductFeedback(user.getId(), product.getId(), feedbackDTO.getText(),
-         feedbackDTO.getStars());
-     simpMessagingTemplate.convertAndSend("/feedbacks", "New Feedback created for product");
-     return true;
- }
-
- @Transactional
- public Boolean addArticleFeedback(CreateArticleFeedbackDTO feedbackDTO, HttpServletRequest request) {
-     Article article = articleRepository.findById(feedbackDTO.getArticleId())
-         .orElseThrow(() -> new ArticleNotFoundException(
-             String.format("Article with id %s not found", feedbackDTO.getArticleId())));
-
-     User user = findUserByRequest(request);
-     feedbackRepository.addArticleFeedback(user.getId(), article.getId(), feedbackDTO.getText(),
-         feedbackDTO.getStars());
-     simpMessagingTemplate.convertAndSend("/feedbacks", "New Feedback created for article");
-     return true;
- }
-
- private User findUserByRequest(HttpServletRequest request) {
-     String username = jwtUtils.getUserNameFromJwtToken(jwtUtils.parseJwt(request));
-     return userRepository.findByUsername(username)
-         .orElseThrow(() -> new UsernameNotFoundException(
-             String.format("Username %s not found", username)));
- }
-
- private FeedbackDTO convertToDTO(Feedback feedback) {
-     return FeedbackDTO.builder()
-         .id(feedback.getId())
-         .author(UserInfoDTO.builder().id(feedback.getAuthor().getId()).username(feedback.getAuthor().getUsername()).build())
-         .product(feedback.getProduct() != null ? ProductDTO.builder()
-             .id(feedback.getProduct().getId())
-             .name(feedback.getProduct().getName())
-             .description(feedback.getProduct().getDescription())
-             .rate(feedback.getProduct().getRate())
-             .brand(BrandDTO.builder()
-                 .id(feedback.getProduct().getBrand().getId())
-                 .name(feedback.getProduct().getBrand().getName())
-                 .country(feedback.getProduct().getBrand().getCountry())
-                 .website(feedback.getProduct().getBrand().getWebsite())
-                 .email(feedback.getProduct().getBrand().getEmail())
-                 .build())
-             .guitarForm(feedback.getProduct().getGuitarForm())
-             .typeOfProduct(feedback.getProduct().getTypeOfProduct())
-             .lads(feedback.getProduct().getLads())
-             .avgPrice(feedback.getProduct().getAvgPrice())
-             .color(feedback.getProduct().getColor())
-             .strings(feedback.getProduct().getStrings())
-             .tipMaterial(feedback.getProduct().getTipMaterial())
-             .bodyMaterial(feedback.getProduct().getBodyMaterial())
-             .pickupConfiguration(feedback.getProduct().getPickupConfiguration())
-             .typeComboAmplifier(feedback.getProduct().getTypeComboAmplifier())
-             .build() : null)
-         .article(feedback.getArticle() != null ? ArticleDTO.builder()
```

```
-                .id(feedback.getArticle().getId())
-                .header(feedback.getArticle().getHeader())
-                .text(feedback.getArticle().getText())
-                .author(UserInfoDTO.builder().id(feedback.getArticle().getAuthor().getId())
-                    .username(feedback.getArticle().getAuthor().getUsername()).build())
-                .createdAt(feedback.getArticle().getCreatedAt())
-                .accepted(feedback.getArticle().getAccepted())
-                .build() : null)
-            .createdAt(feedback.getCreatedAt())
-            .text(feedback.getText())
-            .stars(feedback.getStars())
-            .build();
-    }
+    private final FeedbackRepository feedbackRepository;
+    private final JwtUtils jwtUtils;
+    private final UserRepository userRepository;
+    private final ProductRepository productRepository;
+    private final ArticleRepository articleRepository;
+    private final SimpMessagingTemplate simpMessagingTemplate;
+
+    @Value("${app.messaging.feedback-topic:/feedbacks}")
+    private String feedbackTopic;
+
+    public List<FeedbackDTO> getFeedbackByProductId(Long productId, int from, int size) {
+        Pageable pageable = Pagification.createPageTemplate(from, size);
+
+        List<Feedback> feedbacks = feedbackRepository.findByProductId(productId, pageable).getContent();
+
+        return feedbacks
+                .stream()
+                .map(this::convertToDTO)
+                .sorted(new Comparator<FeedbackDTO>() {
+                    @Override
+                    public int compare(FeedbackDTO o1, FeedbackDTO o2) {
+                        return o1.getId().compareTo(o2.getId());
+                    }
+                })
+                .toList();
+
+    }
+
+    public List<FeedbackDTO> getFeedbackByArticleId(Long articleId, int from, int size) {
+        Pageable pageable = Pagification.createPageTemplate(from, size);
+
+        List<Feedback> feedbacks = feedbackRepository.findByArticleId(articleId, pageable).getContent();
+
+        return feedbacks
+                .stream()
+                .map(this::convertToDTO)
+                .sorted(new Comparator<FeedbackDTO>() {
+                    @Override
+                    public int compare(FeedbackDTO o1, FeedbackDTO o2) {
+                        return o1.getId().compareTo(o2.getId());
+                    }
+                })
+                .toList();
```

```
+    }
+
+    @Transactional
+    public Boolean addProductFeedback(CreateProductFeedbackDTO feedbackDTO, HttpServletRequest request) {
+        Product product = productRepository.findById(feedbackDTO.getProductId())
+                .orElseThrow(() -> new ProductNotFoundException(
+                        String.format("Product with id %s not found", feedbackDTO.getProductId())));
+
+        User user = findUserByRequest(request);
+        feedbackRepository.addProductFeedback(user.getId(), product.getId(), feedbackDTO.getText(),
+                feedbackDTO.getStars());
+        simpMessagingTemplate.convertAndSend(feedbackTopic, "New Feedback created for product");
+        return true;
+    }
+
+    @Transactional
+    public Boolean addArticleFeedback(CreateArticleFeedbackDTO feedbackDTO, HttpServletRequest request) {
+        Article article = articleRepository.findById(feedbackDTO.getArticleId())
+                .orElseThrow(() -> new ArticleNotFoundException(
+                        String.format("Article with id %s not found", feedbackDTO.getArticleId())));
+
+        User user = findUserByRequest(request);
+        feedbackRepository.addArticleFeedback(user.getId(), article.getId(), feedbackDTO.getText(),
+                feedbackDTO.getStars());
+        simpMessagingTemplate.convertAndSend(feedbackTopic, "New Feedback created for article");
+        return true;
+    }
+
+    private User findUserByRequest(HttpServletRequest request) {
+        String username = jwtUtils.getUserNameFromJwtToken(jwtUtils.parseJwt(request));
+        return userRepository.findByUsername(username)
+                .orElseThrow(() -> new UsernameNotFoundException(
+                        String.format("Username %s not found", username)));
+    }
+
+    private FeedbackDTO convertToDTO(Feedback feedback) {
+        return FeedbackDTO.builder()
+                .id(feedback.getId())
+                .author(UserInfoDTO.builder().id(feedback.getAuthor().getId())
+                        .username(feedback.getAuthor().getUsername()).build())
+                .product(feedback.getProduct() != null ? ProductDTO.builder()
+                        .id(feedback.getProduct().getId())
+                        .name(feedback.getProduct().getName())
+                        .description(feedback.getProduct().getDescription())
+                        .rate(feedback.getProduct().getRate())
+                        .brand(BrandDTO.builder()
+                                .id(feedback.getProduct().getBrand().getId())
+                                .name(feedback.getProduct().getBrand().getName())
+                                .country(feedback.getProduct().getBrand().getCountry())
+                                .website(feedback.getProduct().getBrand().getWebsite())
+                                .email(feedback.getProduct().getBrand().getEmail())
+                                .build())
+                        .guitarForm(feedback.getProduct().getGuitarForm())
+                        .typeOfProduct(feedback.getProduct().getTypeOfProduct())
+                        .lads(feedback.getProduct().getLads())
+                        .avgPrice(feedback.getProduct().getAvgPrice())
```

```
+                           .color(feedback.getProduct().getColor())
+                           .strings(feedback.getProduct().getStrings())
+                           .tipMaterial(feedback.getProduct().getTipMaterial())
+                           .bodyMaterial(feedback.getProduct().getBodyMaterial())
+                           .pickupConfiguration(feedback.getProduct().getPickupConfiguration())
+                           .typeComboAmplifier(feedback.getProduct().getTypeComboAmplifier())
+                           .build() : null)
+                   .article(feedback.getArticle() != null ? ArticleDTO.builder()
+                           .id(feedback.getArticle().getId())
+                           .header(feedback.getArticle().getHeader())
+                           .text(feedback.getArticle().getText())
+                           .author(UserInfoDTO.builder().id(feedback.getArticle().getAuthor().getId())
+                                   .username(feedback.getArticle().getAuthor().getUsername()).build())
+                           .createdAt(feedback.getArticle().getCreatedAt())
+                           .accepted(feedback.getArticle().getAccepted())
+                           .build() : null)
+                   .createdAt(feedback.getCreatedAt())
+                   .text(feedback.getText())
+                   .stars(feedback.getStars())
+                   .build();
+    }

 }
diff --git a/backend/src/main/resources/application.properties b/backend/src/main/resources/application.properties
index 76d1f3e..125968b 100644
--- a/backend/src/main/resources/application.properties
+++ b/backend/src/main/resources/application.properties
@@ -9,4 +9,18 @@ spring.datasource.url=jdbc:postgresql://localhost:5432/${POSTGRES_NAME}
 spring.datasource.username=${POSTGRES_USER}
 spring.datasource.password=${POSTGRES_PASSWORD}

-server.port = ${PORT}
\ No newline at end of file
+server.port = ${PORT}
+
+app.security.jwt.secret=${JWT_SECRET}
+app.security.jwt.expiration-ms=${JWT_EXPIRATION_MS:3600000}
+
+app.cors.allowed-origins=http://localhost:5173
+
+app.websocket.broker-destinations=/topic,/feedbacks
+app.websocket.application-destination-prefix=/app/**
+app.websocket.allowed-origins=http://localhost:5173
+
+app.messaging.feedback-topic=/feedbacks
+app.messaging.articles-topic=/articles
+
+app.external.markdown-parser.path=${user.dir}/parser
\ No newline at end of file
```

git show a0d0014

```
commit a0d0014076323f03dea8d881b2d072c2a436aff8
Author: ta4ilka69 <artem_balin_2004@mail.ru>
Date:   Sun Nov 9 20:22:35 2025 +0300
```

readme and docker for grafana & prometheus

```
diff --git a/.docker/docker-compose.yaml b/.docker/docker-compose.yaml
index df8de9f..9a1e3f7 100644
--- a/.docker/docker-compose.yaml
+++ b/.docker/docker-compose.yaml
@@ -53,6 +53,52 @@ services:
       - "-c"
       - "mkdir -p /var/lib/pgadmin/sessions && chown -R 5050:5050 /var/lib/pgadmin && /entrypoint.sh"

+  prometheus:
+    container_name: prometheus
+    image: prom/prometheus:latest
+    volumes:
+      - ./prometheus/prometheus.yml:/etc/prometheus/prometheus.yml:ro
+    ports:
+      - "9090:9090"
+    restart: unless-stopped
+    networks:
+      - postgres
+
+  grafana:
+    container_name: grafana
+    image: grafana/grafana:latest
+    environment:
+      GF_SECURITY_ADMIN_USER: "admin"
+      GF_SECURITY_ADMIN_PASSWORD: "admin"
+      GF_USERS_DEFAULT_THEME: "dark"
+    volumes:
+      - ./grafana/provisioning/datasources:/etc/grafana/provisioning/datasources
+      - ./grafana/provisioning/dashboards:/etc/grafana/provisioning/dashboards
+      - ./grafana/dashboards:/var/lib/grafana/dashboards
+    ports:
+      - "3000:3000"
+    depends_on:
+      - prometheus
+    restart: unless-stopped
+    networks:
+      - postgres
+
+  cadvisor:
+    container_name: cadvisor
+    image: gcr.io/cadvisor/cadvisor:latest
+    ports:
+      - "8081:8080"
+    volumes:
+      - /:/rootfs:ro
+      - /var/run:/var/run:ro
+      - /sys:/sys:ro
+      - /var/lib/docker/:/var/lib/docker:ro
+      - /dev/disk/:/dev/disk:ro
+    privileged: true
+    restart: unless-stopped
+    networks:
+      - postgres
```

```
+

 networks:
   postgres:
diff --git a/README.md b/README.md
index c50dc8f..c91e5e0 100644
--- a/README.md
+++ b/README.md
@@ -89,3 +89,57 @@

 - Настроить https для безопасности передачи данных.
 - Настройки производительности приложения (пул соединений, оптимизация JVM, настройка postgreSQL).
+
+## Метрики и дашборды (Prometheus/Micrometer/Grafana)
+
+Добавлена интеграция метрик в бэкенд и инфраструктура мониторинга.
+
+- Бэкенд: `spring-boot-starter-actuator` + `micrometer-registry-prometheus`.
+- Эндпоинт метрик: `GET /actuator/prometheus`.
+- Prometheus и Grafana добавлены в `.docker/docker-compose.yaml`.
+- Grafana автоматически настраивает datasource и загружает дашборд `Spring Boot Overview`.
+
+### Как запустить локально
+
+1) Запустить бэкенд (по умолчанию порт 8080, можно задать `PORT`):
+
+```bash
+mvn -f backend/pom.xml spring-boot:run
+```
+
+2) Запустить Prometheus и Grafana:
+
+```bash
+cd .docker
+docker compose up -d prometheus grafana
+```
+
+3) Открыть:
+- Prometheus: `http://localhost:9090`
+- Grafana: `http://localhost:3000` (логин/пароль: `admin`/`admin`)
+- Дашборды в Grafana: папка `Spring`, дашборд `Spring Boot Overview`
+
+### Настройки
+
+- Точка скрейпа настроена в `.docker/prometheus/prometheus.yml` на `host.docker.internal:8080/actuator/prometheus`.
+  - Если бэкенд слушает другой порт — обновить `targets`.
+  - Если бэкенд будет запущен в Docker-сети — укажите имя сервиса и порт, например `backend:8080`.
+- Конфигурация Grafana:
+  - Datasource: `.docker/grafana/provisioning/datasources/datasource.yml`
+  - Дашборды: `.docker/grafana/dashboards/`
+
+### Полезные метрики
+
+- HTTP: `http_server_requests_seconds_count`, гистограмма `http_server_requests_seconds_bucket`
+- JVM: `jvm_memory_used_bytes`, `jvm_threads_live_threads`, `process_cpu_usage`
+- База данных (через Hikari/JPA): `hikaricp_connections_active`, `jdbc_connections_active`
```

```
+
+### Метрики контейнеров (CPU/RAM/Disk/Net)
+
+- Добавлен `cAdvisor` в `.docker/docker-compose.yaml`. Запуск вместе со стеком:
+  ```bash
+  cd .docker
+  docker compose up -d prometheus grafana cadvisor
+  ```
+- Prometheus-джоб `cadvisor` уже подключен.
+- На дашборде добавлены панели: Containers CPU/Memory/FS usage/Net IO.
```

git show dcbebdb

```
commit dcbebdb036e88f7bee32b4ca77a4b658a73b7929
Author: ta4ilka69 <artem_balin_2004@mail.ru>
Date:   Sun Nov 9 20:22:59 2025 +0300

    configs for dashboards

diff --git a/.docker/grafana/provisioning/dashboards/dashboards.yml b/.docker/grafana/provisioning/dashboards/dashboards.yml
new file mode 100644
index 0000000..a682d8d
--- /dev/null
+++ b/.docker/grafana/provisioning/dashboards/dashboards.yml
@@ -0,0 +1,12 @@
+apiVersion: 1
+
+providers:
+  - name: 'Spring Dashboards'
+    orgId: 1
+    folder: 'Spring'
+    type: file
+    disableDeletion: false
+    editable: true
+    options:
+      path: /var/lib/grafana/dashboards
+
diff --git a/.docker/grafana/provisioning/datasources/datasource.yml b/.docker/grafana/provisioning/datasources/datasource.yml
new file mode 100644
index 0000000..b357ba8
--- /dev/null
+++ b/.docker/grafana/provisioning/datasources/datasource.yml
@@ -0,0 +1,10 @@
+apiVersion: 1
+
+datasources:
+  - name: Prometheus
+    type: prometheus
+    access: proxy
+    url: http://prometheus:9090
+    isDefault: true
+    editable: false
+
diff --git a/.docker/prometheus/prometheus.yml b/.docker/prometheus/prometheus.yml
```

```
new file mode 100644
index 0000000..b072fe2
--- /dev/null
+++ b/.docker/prometheus/prometheus.yml
@@ -0,0 +1,18 @@
+global:
+  scrape_interval: 15s
+  evaluation_interval: 15s
+
+scrape_configs:
+  - job_name: 'prometheus'
+    static_configs:
+      - targets: ['localhost:9090']
+
+  - job_name: 'spring-boot'
+    metrics_path: '/actuator/prometheus'
+    scrape_timeout: 10s
+    static_configs:
+      - targets: ['host.docker.internal:5252']
+  - job_name: 'cadvisor'
+    scrape_interval: 15s
+    static_configs:
+      - targets: ['cadvisor:8080']
```

git show b7de8bd

```
commit b7de8bd8ab3bfd1e4be1c076f8631a8322d23a48
Author: ta4ilka69 <artem_balin_2004@mail.ru>
Date:   Sun Nov 9 20:23:17 2025 +0300

    dashboard with metrics

diff --git a/.docker/grafana/dashboards/spring-boot-overview.json b/.docker/grafana/dashboards/spring-boot-overview.json
new file mode 100644
index 0000000..41e2f7d
--- /dev/null
+++ b/.docker/grafana/dashboards/spring-boot-overview.json
@@ -0,0 +1,113 @@
+{
+  "id": null,
+  "uid": "spring-overview",
+  "title": "Spring Boot Overview",
+  "timezone": "browser",
+  "schemaVersion": 36,
+  "version": 1,
+  "tags": ["micrometer", "spring"],
+  "time": {"from": "now-6h", "to": "now"},
+  "templating": {"list": []},
+  "panels": [
+    {
+      "type": "timeseries",
+      "title": "HTTP Requests Rate (req/s)",
+      "gridPos": {"h": 8, "w": 12, "x": 0, "y": 0},
+      "targets": [
```

```
+      {
+        "refId": "A",
+        "expr": "sum(rate(http_server_requests_seconds_count[1m]))"
+      }
+    ]
+  },
+  {
+    "type": "timeseries",
+    "title": "HTTP p95 Latency (s)",
+    "gridPos": {"h": 8, "w": 12, "x": 12, "y": 0},
+    "targets": [
+      {
+        "refId": "A",
+        "expr": "histogram_quantile(0.95, sum by (le) (rate(http_server_requests_seconds_bucket[5m])))"
+      }
+    ]
+  },
+  {
+    "type": "timeseries",
+    "title": "JVM Heap Used (bytes)",
+    "gridPos": {"h": 8, "w": 12, "x": 12, "y": 8},
+    "targets": [
+      {
+        "refId": "A",
+        "expr": "sum(jvm_memory_used_bytes{area=\"heap\"})"
+      }
+    ]
+  },
+  {
+    "type": "timeseries",
+    "title": "Process CPU (%)",
+    "gridPos": {"h": 8, "w": 12, "x": 0, "y": 16},
+    "targets": [
+      {
+        "refId": "A",
+        "expr": "avg(process_cpu_usage) * 100"
+      }
+    ]
+  },
+  {
+    "type": "timeseries",
+    "title": "System CPU (%)",
+    "gridPos": {"h": 8, "w": 12, "x": 12, "y": 16},
+    "targets": [
+      {
+        "refId": "A",
+        "expr": "avg(system_cpu_usage) * 100"
+      }
+    ]
+  },
+  {
+    "type": "timeseries",
+    "title": "JVM Heap Used (%)",
+    "gridPos": {"h": 8, "w": 12, "x": 0, "y": 24},
+    "targets": [
+      {
```

```
+        "refId": "A",
+        "expr": "100 * sum(jvm_memory_used_bytes{area=\"heap\"}) / sum(jvm_memory_max_bytes{area=\"heap\"})"
+      }
+    ]
+  },
+  {
+    "type": "timeseries",
+    "title": "JVM Threads Live",
+    "gridPos": {"h": 8, "w": 12, "x": 12, "y": 24},
+    "targets": [
+      {
+        "refId": "A",
+        "expr": "sum(jvm_threads_live_threads)"
+      }
+    ]
+  },
+  {
+    "type": "timeseries",
+    "title": "DB Connections Active",
+    "gridPos": {"h": 8, "w": 12, "x": 0, "y": 32},
+    "targets": [
+      {
+        "refId": "A",
+        "expr": "sum(hikaricp_connections_active)"
+      }
+    ]
+  },
+  {
+    "type": "timeseries",
+    "title": "Uptime (hours)",
+    "gridPos": {"h": 8, "w": 12, "x": 12, "y": 40},
+    "targets": [
+      {
+        "refId": "A",
+        "expr": "sum(process_uptime_seconds) / 3600"
+      }
+    ]
+  }
+ ]
+}
+
```

git show a7b2756

```
commit a7b27560bfe7ff6d9bb4554618c7ea746eb1b619
Author: ta4ilka69 <artem_balin_2004@mail.ru>
Date:   Sun Nov 9 20:23:31 2025 +0300

    app config for micrometer

diff --git a/backend/pom.xml b/backend/pom.xml
index 536022e..b0b02fc 100644
--- a/backend/pom.xml
+++ b/backend/pom.xml
@@ -23,6 +23,14 @@
```

```
                                        <groupId>org.springframework.boot</groupId>
                                        <artifactId>spring-boot-starter-web</artifactId>
                                </dependency>
+       <dependency>
+           <groupId>org.springframework.boot</groupId>
+           <artifactId>spring-boot-starter-actuator</artifactId>
+       </dependency>
+       <dependency>
+           <groupId>io.micrometer</groupId>
+           <artifactId>micrometer-registry-prometheus</artifactId>
+       </dependency>
                                <!-- <dependency>
                                        <groupId>org.hibernate</groupId>
                                        <artifactId>hibernate-core</artifactId>
diff                                        --git                                        a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/SecurityConfig.java
b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/SecurityConfig.java
index f1a256d..a5a0d3d 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/SecurityConfig.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/security/SecurityConfig.java
@@ -46,6 +46,7 @@ public class SecurityConfig {
      .authorizeHttpRequests(
          request -> request
              .requestMatchers("/index.html", "/favicon.ico", "/static/**").permitAll()
+              .requestMatchers("/actuator/**").permitAll()
              .requestMatchers("api/auth/**", "api/user/role/**", "/ws", "/api/").permitAll()
              .requestMatchers(HttpMethod.GET, "api/admin/**").hasRole("ADMIN")
              .requestMatchers(HttpMethod.POST, "api/moderate/**").hasRole("ADMIN")
diff --git a/backend/src/main/resources/application.properties b/backend/src/main/resources/application.properties
index 76d1f3e..7aa97f6 100644
--- a/backend/src/main/resources/application.properties
+++ b/backend/src/main/resources/application.properties
@@ -1,6 +1,6 @@
 #Database
 # spring.jpa.generate-ddl=true
-spring.jpa.show-sql=true
+spring.jpa.show-sql=false
 spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
 # spring.jpa.hibernate.ddl-auto=update
 spring.datasource.driver-class-name=org.postgresql.Driver
@@ -9,4 +9,15 @@ spring.datasource.username=${POSTGRES_USER}
 spring.datasource.password=${POSTGRES_PASSWORD}

-server.port = ${PORT}
\ No newline at end of file
+server.address=0.0.0.0
+server.port=5252
+#logging
+logging.level.org.springframework.web=DEBUG
+
+# Actuator & Micrometer Prometheus
+spring.application.name=GMatch-Backend
+management.endpoints.web.exposure.include=health,info,metrics,prometheus
+management.endpoint.health.show-details=always
+management.prometheus.metrics.export.enabled=true
+management.metrics.distribution.percentiles-histogram.http.server.requests=true
```

```
+management.metrics.tags.application=${spring.application.name}
\ No newline at end of file
```

git show c075252

```
commit c075252066c76ac62ab3b17da61502c626a0a4fb
Merge: ac73189 a7b2756
Author: Romariok <ronya-ko@yandex.ru>
Date:   Mon Nov 10 14:40:00 2025 +0300

    Merge branch 'grafana' into step1-refactoring

diff --cc backend/src/main/resources/application.properties
index 125968b,7aa97f6..83283ba
--- a/backend/src/main/resources/application.properties
+++ b/backend/src/main/resources/application.properties
@@@ -9,18 -9,15 +9,30 @@@ spring.datasource.url=jdbc:postgresql:/
  spring.datasource.username=${POSTGRES_USER}
  spring.datasource.password=${POSTGRES_PASSWORD}

 +server.port = ${PORT}
+ server.address=0.0.0.0
 -server.port=5252
 +
 +app.security.jwt.secret=${JWT_SECRET}
 +app.security.jwt.expiration-ms=${JWT_EXPIRATION_MS:3600000}
 +
 +app.cors.allowed-origins=http://localhost:5173
 +
 +app.websocket.broker-destinations=/topic,/feedbacks
 +app.websocket.application-destination-prefix=/app/**
 +app.websocket.allowed-origins=http://localhost:5173
 +
 +app.messaging.feedback-topic=/feedbacks
 +app.messaging.articles-topic=/articles
 +
- app.external.markdown-parser.path=${user.dir}/parser
++app.external.markdown-parser.path=${user.dir}/parser
++
+ #logging
+ logging.level.org.springframework.web=DEBUG
+
+ # Actuator & Micrometer Prometheus
+ spring.application.name=GMatch-Backend
+ management.endpoints.web.exposure.include=health,info,metrics,prometheus
+ management.endpoint.health.show-details=always
+ management.prometheus.metrics.export.enabled=true
+ management.metrics.distribution.percentiles-histogram.http.server.requests=true
 -management.metrics.tags.application=${spring.application.name}
++management.metrics.tags.application=${spring.application.name}
```

git show 6185046

```
commit 6185046e0bd5e5f7357b765b20830162a19b2e3c
Author: ta4ilka69 <artem_balin_2004@mail.ru>
Date:   Mon Nov 10 23:50:15 2025 +0300

    controller logginh

diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ArticleController.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ArticleController.java
index 2f43410..6d857c5 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ArticleController.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ArticleController.java
@@ -13,6 +13,7 @@ import itmo.is.cw.GuitarMatchIS.service.ArticleService;
 import jakarta.servlet.http.HttpServletRequest;
 import jakarta.validation.Valid;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;
 import io.swagger.v3.oas.annotations.Operation;
 import io.swagger.v3.oas.annotations.Parameter;
 import io.swagger.v3.oas.annotations.tags.Tag;
@@ -20,6 +21,7 @@ import io.swagger.v3.oas.annotations.tags.Tag;
 @RestController
 @RequestMapping("/api/article")
 @RequiredArgsConstructor
+@Slf4j
 @Tag(name = "Статьи", description = "API для работы со статьями")
 public class ArticleController {
     private final ArticleService articleService;
@@ -32,6 +34,7 @@ public class ArticleController {
         @Parameter(description = "Направление сортировки") @RequestParam boolean ascending,
         @Parameter(description = "Начальная позиция") @RequestParam int from,
         @Parameter(description = "Количество элементов") @RequestParam int size) {
+        log.info("Request for accepted articles received. from={}, size={}, sortBy={}, ascending={}", from, size, sortBy, ascending);
         return articleService.getAcceptedArticles(from, size, sortBy, ascending);
     }

@@ -44,6 +47,7 @@ public class ArticleController {
         @Parameter(description = "Начальная позиция") @RequestParam int from,
         @Parameter(description = "Количество элементов") @RequestParam int size,
         HttpServletRequest request) {
+        log.info("Request for unaccepted articles received. from={}, size={}", from, size);
         return articleService.getStatusArticles(from, size, request);
     }

@@ -51,6 +55,7 @@ public class ArticleController {
             description = "Возвращает статью по её ID")
     @GetMapping("/id/{id}")
     public ArticleDTO getArticleById(@PathVariable Long id) {
+        log.info("Request for article with id {} received.", id);
         return articleService.getArticleById(id);
     }

@@ -61,6 +66,7 @@ public class ArticleController {
         @Parameter(description = "Текст для поиска в заголовке") @PathVariable String header,
         @Parameter(description = "Начальная позиция") @RequestParam int from,
         @Parameter(description = "Количество элементов") @RequestParam int size) {
+        log.info("Request for articles with header containing '{}' received. from={}, size={}", header, from, size);
```

```
    return articleService.getArticlesByHeaderContaining(header, from, size);
   }

@@ -71,6 +77,7 @@ public class ArticleController {
     @Parameter(description = "ID автора") @PathVariable Long authorId,
     @Parameter(description = "Начальная позиция") @RequestParam int from,
     @Parameter(description = "Количество элементов") @RequestParam int size) {
+    log.info("Request for articles by author {} received. from={}, size={}", authorId, from, size);
     return articleService.getArticlesByAuthorId(authorId, from, size);
   }

@@ -80,6 +87,7 @@ public class ArticleController {
   public boolean moderateArticle(
     @Parameter(description = "Данные для модерации") @RequestBody @Valid ModerateArticleDTO moderateArticleDTO,
     HttpServletRequest request) {
+    log.info("Request to moderate article with id {} received.", moderateArticleDTO.getArticleId());
     return articleService.moderateArticle(moderateArticleDTO, request);
   }

@@ -89,6 +97,7 @@ public class ArticleController {
   public ArticleDTO createArticle(
     @Parameter(description = "Данные новой статьи") @RequestBody @Valid CreateArticleDTO createArticleDTO,
     HttpServletRequest request) {
+    log.info("Request to create article with header '{}' received.", createArticleDTO.getHeader());
     return articleService.createArticle(createArticleDTO, request);
   }
 }
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/AuthController.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/AuthController.java
index 8d06c8d..a3aa8da 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/AuthController.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/AuthController.java
@@ -10,6 +10,7 @@ import itmo.is.cw.GuitarMatchIS.dto.UserDTO;
 import itmo.is.cw.GuitarMatchIS.service.AuthService;
 import jakarta.validation.Valid;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;
 import io.swagger.v3.oas.annotations.Operation;
 import io.swagger.v3.oas.annotations.tags.Tag;
 import io.swagger.v3.oas.annotations.responses.ApiResponse;
@@ -18,6 +19,7 @@ import io.swagger.v3.oas.annotations.responses.ApiResponses;
 @RestController
 @RequestMapping("/api/auth")
 @RequiredArgsConstructor
+@Slf4j
 @Tag(name = "Аутентификация", description = "API для регистрации и входа пользователей")
 public class AuthController {
   private final AuthService authService;
@@ -31,6 +33,7 @@ public class AuthController {
   })
   @PostMapping("/register")
   public AuthResponseDTO register(@RequestBody @Valid UserDTO registerUserDto) {
+    log.info("Received request to register user: {}", registerUserDto.getUsername());
     return authService.register(registerUserDto);
   }
```

```
@@ -43,6 +46,7 @@ public class AuthController {
    })
    @PostMapping("/login")
    public AuthResponseDTO login(@RequestBody @Valid UserDTO loginUserDto) {
+       log.info("Received request to login user: {}", loginUserDto.getUsername());
        return authService.login(loginUserDto);
    }
 }
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/BrandController.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/BrandController.java
index 6ebd349..4345063 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/BrandController.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/BrandController.java
@@ -7,6 +7,7 @@ import org.springframework.web.bind.annotation.*;
 import itmo.is.cw.GuitarMatchIS.dto.BrandDTO;
 import itmo.is.cw.GuitarMatchIS.service.BrandService;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;
 import io.swagger.v3.oas.annotations.Operation;
 import io.swagger.v3.oas.annotations.Parameter;
 import io.swagger.v3.oas.annotations.tags.Tag;
@@ -16,6 +17,7 @@ import io.swagger.v3.oas.annotations.responses.ApiResponses;
 @RestController
 @RequiredArgsConstructor
 @RequestMapping("/api/brand")
+@Slf4j
 @Tag(name = "Бренды", description = "API для работы с брендами музыкальных инструментов")
 public class BrandController {
    private final BrandService brandService;
@@ -30,6 +32,7 @@ public class BrandController {
    public List<BrandDTO> getBrands(
        @Parameter(description = "Начальная позиция") @RequestParam int from,
        @Parameter(description = "Количество элементов") @RequestParam int size) {
+       log.info("Request for brands received. from={}, size={}", from, size);
        return brandService.getBrands(from, size);
    }

@@ -37,6 +40,7 @@ public class BrandController {
            description = "Возвращает бренд по его ID")
    @GetMapping("/id/{id}")
    public BrandDTO getBrandById(@PathVariable Long id) {
+       log.info("Request for brand with id {} received.", id);
        return brandService.getBrandById(id);
    }
 }
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/FeedbackController.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/FeedbackController.java
index 34447db..e6018c9 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/FeedbackController.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/FeedbackController.java
@@ -22,10 +22,12 @@ import itmo.is.cw.GuitarMatchIS.service.FeedbackService;
 import jakarta.servlet.http.HttpServletRequest;
 import jakarta.validation.Valid;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;
```

```
 @RestController
 @RequiredArgsConstructor
 @RequestMapping("/api/feedback")
+@Slf4j
 @Tag(name = "Отзывы", description = "API для работы с отзывами на товары и статьи")
 public class FeedbackController {
    private final FeedbackService feedbackService;
@@ -41,6 +43,7 @@ public class FeedbackController {
        @Parameter(description = "ID товара") @PathVariable Long productId,
        @Parameter(description = "Начальная позиция") @RequestParam int from,
        @Parameter(description = "Количество элементов") @RequestParam int size) {
+      log.info("Request for feedback for product with id {} received. from={}, size={}", productId, from, size);
       return feedbackService.getFeedbackByProductId(productId, from, size);
    }

@@ -55,6 +58,7 @@ public class FeedbackController {
        @Parameter(description = "ID статьи") @PathVariable Long articleId,
        @Parameter(description = "Начальная позиция") @RequestParam int from,
        @Parameter(description = "Количество элементов") @RequestParam int size) {
+      log.info("Request for feedback for article with id {} received. from={}, size={}", articleId, from, size);
       return feedbackService.getFeedbackByArticleId(articleId, from, size);
    }

@@ -69,6 +73,7 @@ public class FeedbackController {
    public Boolean addProductFeedback(
        @Parameter(description = "Данные отзыва") @RequestBody @Valid CreateProductFeedbackDTO feedbackDTO,
        HttpServletRequest request) {
+      log.info("Request to add feedback for product with id {} received.", feedbackDTO.getProductId());
       return feedbackService.addProductFeedback(feedbackDTO, request);
    }

@@ -83,6 +88,7 @@ public class FeedbackController {
    public Boolean addArticleFeedback(
        @Parameter(description = "Данные отзыва") @RequestBody @Valid CreateArticleFeedbackDTO feedbackDTO,
        HttpServletRequest request) {
+      log.info("Request to add feedback for article with id {} received.", feedbackDTO.getArticleId());
       return feedbackService.addArticleFeedback(feedbackDTO, request);
    }
 }
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ForumPostController.java
b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ForumPostController.java
index c6c1200..20d6ebc 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ForumPostController.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ForumPostController.java
@@ -13,10 +13,12 @@ import itmo.is.cw.GuitarMatchIS.service.ForumPostService;
 import jakarta.servlet.http.HttpServletRequest;
 import jakarta.validation.Valid;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 @RestController
 @RequestMapping("/api/forum/post")
 @RequiredArgsConstructor
+@Slf4j
 @Tag(name = "Сообщения форума", description = "API для работы с сообщениями на форуме")
 public class ForumPostController {
```

```
   private final ForumPostService forumPostService;
@@ -31,6 +33,7 @@ public class ForumPostController {
   public ForumPostDTO createForumPost(
         @Parameter(description = "Данные нового сообщения") @RequestBody @Valid CreateForumPostDTO createForumPostDTO,
         HttpServletRequest request) {
+     log.info("Request to create forum post in topic with id {} received.", createForumPostDTO.getForumTopicId());
      return forumPostService.createForumPost(createForumPostDTO, request);
   }
 }
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ForumTopicController.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ForumTopicController.java
index d1ac36c..1500cd9 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ForumTopicController.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ForumTopicController.java
@@ -17,10 +17,12 @@ import itmo.is.cw.GuitarMatchIS.service.ForumTopicService;
 import jakarta.servlet.http.HttpServletRequest;
 import jakarta.validation.Valid;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 @RestController
 @RequestMapping("/api/forum/topic")
 @RequiredArgsConstructor
+@Slf4j
 @Tag(name = "Темы форума", description = "API для работы с темами на форуме")
 public class ForumTopicController {
   private final ForumTopicService forumTopicService;
@@ -35,6 +37,7 @@ public class ForumTopicController {
   public List<ForumTopicDTO> getForumTopics(
         @Parameter(description = "Начальная позиция") @RequestParam int from,
         @Parameter(description = "Количество элементов") @RequestParam int size) {
+     log.info("Request for forum topics received. from={}, size={}", from, size);
      return forumTopicService.getForumTopics(from, size);
   }

@@ -49,6 +52,7 @@ public class ForumTopicController {
         @Parameter(description = "ID автора") @PathVariable Long authorId,
         @Parameter(description = "Начальная позиция") @RequestParam int from,
         @Parameter(description = "Количество элементов") @RequestParam int size) {
+     log.info("Request for forum topics by author {} received. from={}, size={}", authorId, from, size);
      return forumTopicService.getForumTopicsByAuthor(authorId, from, size);
   }

@@ -62,6 +66,7 @@ public class ForumTopicController {
   public ForumTopicDTO createForumTopic(
         @Parameter(description = "Данные новой темы") @RequestBody @Valid CreateForumTopicDTO createForumTopicDTO,
         HttpServletRequest request) {
+     log.info("Request to create forum topic with title '{}' received.", createForumTopicDTO.getTitle());
      return forumTopicService.createTopic(createForumTopicDTO, request);
   }

@@ -76,6 +81,7 @@ public class ForumTopicController {
   public Boolean closeForumTopic(
         @Parameter(description = "ID темы") @PathVariable Long topicId,
         HttpServletRequest request) {
+     log.info("Request to close forum topic with id {} received.", topicId);
```

```
      return forumTopicService.closeTopic(topicId, request);
    }

@@ -90,6 +96,7 @@ public class ForumTopicController {
          @Parameter(description = "ID темы") @PathVariable Long topicId,
          @Parameter(description = "Начальная позиция") @RequestParam int from,
          @Parameter(description = "Количество элементов") @RequestParam int size) {
+      log.info("Request for posts in topic {} received. from={}, size={}", topicId, from, size);
      return forumPostService.getForumPostsByTopic(topicId, from, size);
    }

@@ -104,6 +111,7 @@ public class ForumTopicController {
    public Boolean isTopicOwner(
          @Parameter(description = "ID темы") @PathVariable Long topicId,
          HttpServletRequest request) {
+      log.info("Request to check ownership of topic {} received.", topicId);
      return forumTopicService.isTopicOwner(topicId, request);
    }

@@ -115,6 +123,7 @@ public class ForumTopicController {
    @GetMapping("/{topicId}")
    public ForumTopicDTO getForumTopicById(
          @Parameter(description = "ID темы") @PathVariable Long topicId) {
+      log.info("Request for forum topic with id {} received.", topicId);
      return forumTopicService.getForumTopicById(topicId);
    }

diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/MusicianController.java
b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/MusicianController.java
index 7dd9d16..1f83cda 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/MusicianController.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/MusicianController.java
@@ -16,6 +16,7 @@ import itmo.is.cw.GuitarMatchIS.service.MusicianService;
 import jakarta.servlet.http.HttpServletRequest;
 import jakarta.validation.Valid;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;
 import io.swagger.v3.oas.annotations.Operation;
 import io.swagger.v3.oas.annotations.Parameter;
 import io.swagger.v3.oas.annotations.tags.Tag;
@@ -25,6 +26,7 @@ import io.swagger.v3.oas.annotations.responses.ApiResponses;
 @RestController
 @RequestMapping("/api/musician")
 @RequiredArgsConstructor
+@Slf4j
 @Tag(name = "Музыканты", description = "API для работы с музыкантами")
 public class MusicianController {
    private final MusicianService musicianService;
@@ -37,6 +39,7 @@ public class MusicianController {
    })
    @GetMapping("/id/{id}")
    public MusicianInfoDTO getMusicianInfo(@PathVariable Long id) {
+      log.info("Request for musician info with id {} received.", id);
      return musicianService.getMusicianInfo(id);
    }
```

```
@@ -51,6 +54,7 @@ public class MusicianController {
         @Parameter(description = "Направление сортировки") @RequestParam boolean ascending,
         @Parameter(description = "Начальная позиция") @RequestParam int from,
         @Parameter(description = "Количество элементов") @RequestParam int size) {
+     log.info("Request for musicians received. from={}, size={}, sortBy={}, ascending={}", from, size, sortBy, ascending);
     return musicianService.getMusician(from, size, sortBy, ascending);
   }

@@ -61,6 +65,7 @@ public class MusicianController {
   })
   @GetMapping("/{musicianId}/subscription")
   public Boolean isSubscribed(@PathVariable Long musicianId, HttpServletRequest request) {
+     log.info("Request to check subscription for musician with id {} received.", musicianId);
     return musicianService.isSubscribed(musicianId, request);
   }

@@ -74,6 +79,7 @@ public class MusicianController {
         @Parameter(description = "Имя для поиска") @PathVariable String name,
         @Parameter(description = "Начальная позиция") @RequestParam int from,
         @Parameter(description = "Количество элементов") @RequestParam int size) {
+     log.info("Request to search musicians with name containing '{}' received. from={}, size={}", name, from, size);
     return musicianService.searchMusicians(name, from, size);
   }

@@ -85,6 +91,7 @@ public class MusicianController {
   @GetMapping("/{musicianId}/genres")
   public MusicianGenreDTO getGenresByMusician(
       @Parameter(description = "ID музыканта") @PathVariable Long musicianId) {
+     log.info("Request for genres of musician with id {} received.", musicianId);
     return musicianService.getMusiciansByGenre(musicianId);
   }

@@ -96,6 +103,7 @@ public class MusicianController {
   @GetMapping("/{musicianId}/types")
   public MusicianTypeOfMusicianDTO getMusiciansByTypeOfMusician(
       @Parameter(description = "ID музыканта") @PathVariable Long musicianId) {
+     log.info("Request for types of musician with id {} received.", musicianId);
     return musicianService.getMusiciansByTypeOfMusician(musicianId);
   }

@@ -109,6 +117,7 @@ public class MusicianController {
   public MusicianInfoDTO createMusician(
       @Parameter(description = "Данные музыканта") @RequestBody @Valid CreateMusicianDTO createMusicianDTO,
       HttpServletRequest request) {
+     log.info("Request to create musician with name '{}' received.", createMusicianDTO.getName());
     return musicianService.createMusician(createMusicianDTO, request);
   }

@@ -122,6 +131,7 @@ public class MusicianController {
   public Boolean subscribeToMusician(
       @Parameter(description = "Данные подписки") @RequestBody @Valid SubscribeDTO subscribeDTO,
       HttpServletRequest request) {
+     log.info("Request to subscribe to musician with id {} received.", subscribeDTO.getMusicianId());
     return musicianService.subscribeToMusician(subscribeDTO, request);
   }
```

```
@@ -134,6 +144,7 @@ public class MusicianController {
    public Boolean unsubscribeFromMusician(
        @Parameter(description = "Данные подписки") @RequestBody @Valid SubscribeDTO subscribeDTO,
        HttpServletRequest request) {
+     log.info("Request to unsubscribe from musician with id {} received.", subscribeDTO.getMusicianId());
      return musicianService.unsubscribeFromMusician(subscribeDTO, request);
    }

@@ -147,6 +158,7 @@ public class MusicianController {
    public Boolean addProductToMusician(
        @Parameter(description = "Данные связи продукта с музыкантом") @RequestBody @Valid AddProductMusicianDTO addProductMusicianDTO,
        HttpServletRequest request) {
+                log.info("Request to add product with id {} to musician '{}' received.", addProductMusicianDTO.getProductId(),
addProductMusicianDTO.getMusicianName());
      return musicianService.addProductToMusician(addProductMusicianDTO, request);
    }

@@ -159,6 +171,7 @@ public class MusicianController {
    public Boolean deleteProductFromMusician(
        @Parameter(description = "Данные связи продукта с музыкантом") @RequestBody @Valid AddProductMusicianDTO addProductMusicianDTO,
        HttpServletRequest request) {
+                log.info("Request to delete product with id {} from musician '{}' received.", addProductMusicianDTO.getProductId(),
addProductMusicianDTO.getMusicianName());
      return musicianService.deleteProductFromMusician(addProductMusicianDTO, request);
    }

@@ -170,6 +183,7 @@ public class MusicianController {
    @GetMapping("/{musicianName}/products")
    public MusicianProductDTO getMusicianProducts(
        @Parameter(description = "Имя музыканта") @PathVariable String musicianName) {
+     log.info("Request for products of musician with name '{}' received.", musicianName);
      return musicianService.getMusicianProducts(musicianName.replaceAll("_", " "));
    }
 }
diff                                    --git                                    a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ProductController.java
b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ProductController.java
index e972e2d..bb53f14 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ProductController.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ProductController.java
@@ -28,10 +28,12 @@ import itmo.is.cw.GuitarMatchIS.models.TypeComboAmplifier;
 import itmo.is.cw.GuitarMatchIS.models.TypeOfProduct;
 import itmo.is.cw.GuitarMatchIS.service.ProductService;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 @RestController
 @RequiredArgsConstructor
 @RequestMapping("/api/product")
+@Slf4j
 @Tag(name = "Товары", description = "API для работы с музыкальными инструментами и оборудованием")
 public class ProductController {
        private final ProductService productService;
@@ -46,6 +48,7 @@ public class ProductController {
                    @Parameter(description = "Название бренда") @RequestParam String brandName,
                    @Parameter(description = "Начальная позиция") @RequestParam int from,
                    @Parameter(description = "Количество элементов") @RequestParam int size) {
```

```
+            log.info("Request for products by brand '{}' received. from={}, size={}", brandName, from, size);
             return productService.getProductsByBrandName(brandName.replaceAll("_", " "), from, size);
         }

@@ -59,6 +62,7 @@ public class ProductController {
                     @Parameter(description = "ID товара") @PathVariable long id,
                     @Parameter(description = "Начальная позиция") @RequestParam int from,
                     @Parameter(description = "Количество элементов") @RequestParam int size) {
+            log.info("Request for musicians by product id {} received. from={}, size={}", id, from, size);
             return productService.getMusiciansByProductId(id, from, size);
         }

@@ -70,6 +74,7 @@ public class ProductController {
         public ProductGenreDTO getProductsById(
                     @Parameter(description = "id") @PathVariable long id) {
+            log.info("Request for product by id {} received.", id);
             return productService.getProductsById(id);
         }

@@ -81,6 +86,7 @@ public class ProductController {
         public ProductGenreDTO getGenresByProductName(
                     @Parameter(description = "Название товара") @PathVariable String name) {
+            log.info("Request for genres by product name '{}' received.", name);
             return productService.getGenresByProductName(name.replaceAll("_", " "));
         }

@@ -94,6 +100,7 @@ public class ProductController {
                     @Parameter(description = "ID товара") @PathVariable long id,
                     @Parameter(description = "Начальная позиция") @RequestParam int from,
                     @Parameter(description = "Количество элементов") @RequestParam int size) {
+            log.info("Request for articles for product id {} received. from={}, size={}", id, from, size);
             return productService.getProductArticles(id, from, size);
         }

@@ -107,6 +114,7 @@ public class ProductController {
                     @Parameter(description = "ID товара") @PathVariable long id,
                     @Parameter(description = "Начальная позиция") @RequestParam int from,
                     @Parameter(description = "Количество элементов") @RequestParam int size) {
+            log.info("Request for shops for product id {} received. from={}, size={}", id, from, size);
             return productService.getProductShops(id, from, size);
         }

@@ -120,6 +128,7 @@ public class ProductController {
                     @Parameter(description = "Тип товара") @PathVariable TypeOfProduct typeOfProduct,
                     @Parameter(description = "Начальная позиция") @RequestParam int from,
                     @Parameter(description = "Количество элементов") @RequestParam int size) {
+            log.info("Request for products by type '{}' received. from={}, size={}", typeOfProduct, from, size);
             return productService.getProductsByTypeOfProduct(typeOfProduct, from, size);
         }

@@ -133,6 +142,7 @@ public class ProductController {
                     @Parameter(description = "Строка для поиска") @PathVariable String name,
                     @Parameter(description = "Начальная позиция") @RequestParam int from,
                     @Parameter(description = "Количество элементов") @RequestParam int size) {
```

```
+            log.info("Request for products with name containing '{}' received. from={}, size={}", name, from, size);
             return productService.getProductsByNameContains(name, from, size);
         }

@@ -162,6 +172,7 @@ public class ProductController {
                 @Parameter(description = "Направление сортировки") @RequestParam boolean ascending,
                 @Parameter(description = "Начальная позиция") @RequestParam int from,
                 @Parameter(description = "Количество элементов") @RequestParam int size) {
+            log.info("Request to filter products received. from={}, size={}, sortBy={}, ascending={}", from, size, sortBy, ascending);
             return productService.getProductsByFilter(name, minRate, maxRate, brandId, guitarForm, typeOfProduct,
                     lads,
                     minPrice, maxPrice, color, strings, tipMaterial, bodyMaterial, pickupConfiguration,
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ShopController.java
b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ShopController.java
index 6a377d9..652f4bd 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ShopController.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/ShopController.java
@@ -13,10 +13,12 @@ import itmo.is.cw.GuitarMatchIS.dto.ShopDTO;
 import itmo.is.cw.GuitarMatchIS.dto.ShopProductDTO;
 import itmo.is.cw.GuitarMatchIS.service.ShopService;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 @RestController
 @RequiredArgsConstructor
 @RequestMapping("/api/shop")
+@Slf4j
 @Tag(name = "Магазины", description = "API для работы с музыкальными магазинами")
 public class ShopController {
     private final ShopService shopService;
@@ -26,6 +28,7 @@ public class ShopController {
             description = "Возвращает магазин по его ID")
     @GetMapping("/id/{id}")
     public ShopDTO getShopById(@PathVariable Long id) {
+        log.info("Request for shop with id {} received.", id);
         return shopService.getShopById(id);
     }

@@ -39,6 +42,7 @@ public class ShopController {
     public List<ShopDTO> getShops(
         @Parameter(description = "Начальная позиция") @RequestParam int from,
         @Parameter(description = "Количество элементов") @RequestParam int size) {
+        log.info("Request for shops received. from={}, size={}", from, size);
         return shopService.getShops(from, size);
     }

@@ -54,6 +58,7 @@ public class ShopController {
         @Parameter(description = "ID магазина") @PathVariable Long shopId,
         @Parameter(description = "Начальная позиция") @RequestParam int from,
         @Parameter(description = "Количество элементов") @RequestParam int size) {
+        log.info("Request for products in shop with id {} received. from={}, size={}", shopId, from, size);
         return shopService.getShopProducts(shopId, from, size);
     }
 }
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/UserController.java
b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/UserController.java
```

```
index 46ce526..766fb68 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/UserController.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/controller/UserController.java
@@ -21,10 +21,12 @@ import itmo.is.cw.GuitarMatchIS.models.TypeOfMusician;
 import itmo.is.cw.GuitarMatchIS.service.UserService;
 import jakarta.servlet.http.HttpServletRequest;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 @RestController
 @RequiredArgsConstructor
 @RequestMapping("/api/user")
+@Slf4j
 @Tag(name = "Пользователи", description = "API для работы с пользователями")
 public class UserController {
     private final UserService userService;
@@ -37,6 +39,7 @@ public class UserController {
     })
     @GetMapping("/id/{id}")
     public UserInfoDTO getUserInfoById(@PathVariable Long id) {
+        log.info("Request for user info with id {} received.", id);
         return userService.getUserInfoById(id);
     }

@@ -49,6 +52,7 @@ public class UserController {
     @GetMapping("/role/{username}")
     public Role getRoleByUsername(
             @Parameter(description = "Имя пользователя") @PathVariable String username) {
+        log.info("Request for role for username '{}' received.", username);
         return userService.getRoleByUsername(username);
     }

@@ -60,6 +64,7 @@ public class UserController {
     })
     @GetMapping("/products")
     public List<ProductDTO> getUserProducts(HttpServletRequest request) {
+        log.info("Request for user's products received.");
         return userService.getUserProducts(request);
     }

@@ -73,6 +78,7 @@ public class UserController {
     @PostMapping("/product")
     public Boolean addProductToUser(HttpServletRequest request,
             @Parameter(description = "Данные продукта") @RequestBody AddUserProductDTO addUserProductDTO) {
+        log.info("Request to add product with id {} to user received.", addUserProductDTO.getProductId());
         return userService.addProductToUser(addUserProductDTO, request);
     }

@@ -86,6 +92,7 @@ public class UserController {
     @DeleteMapping("/product")
     public Boolean deleteProductFromUser(HttpServletRequest request,
             @Parameter(description = "Данные продукта") @RequestBody AddUserProductDTO addUserProductDTO) {
+        log.info("Request to delete product with id {} from user received.", addUserProductDTO.getProductId());
         return userService.deleteProductFromUser(addUserProductDTO, request);
     }
```

```
@@ -97,6 +104,7 @@ public class UserController {
    })
    @GetMapping("/genres")
    public List<Genre> getGenresByUser(HttpServletRequest request) {
+       log.info("Request for user's genres received.");
        return userService.getGenresByUser(request);
    }

@@ -108,6 +116,7 @@ public class UserController {
    })
    @GetMapping("/types")
    public List<TypeOfMusician> getTypesOfMusiciansByUser(HttpServletRequest request) {
+       log.info("Request for user's types of musician received.");
        return userService.getTypesOfMusiciansByUser(request);
    }

@@ -121,6 +130,7 @@ public class UserController {
    @PostMapping("/genres")
    public Boolean setGenresToUser(HttpServletRequest request,
        @Parameter(description = "Список жанров") @RequestBody UserGenreDTO genres) {
+       log.info("Request to set genres for user received.");
        return userService.setGenresToUser(request, genres.getGenres());
    }

@@ -134,6 +144,7 @@ public class UserController {
    @PostMapping("/types")
    public Boolean setTypesOfMusiciansToUser(HttpServletRequest request,
        @Parameter(description = "Список типов музыканта") @RequestBody UserTypeOfMusicianDTO typesOfMusicians) {
+       log.info("Request to set types of musician for user received.");
        return userService.setTypesOfMusiciansToUser(request, typesOfMusicians.getTypesOfMusicians());
    }

@@ -145,6 +156,7 @@ public class UserController {
    })
    @GetMapping("/subscribed")
    public List<MusicianInfoDTO> getSubscribedMusicians(HttpServletRequest request) {
+       log.info("Request for user's subscribed musicians received.");
        return userService.getSubscribedMusicians(request);
    }
 }
```

git show becdbef

```
commit becdbef78f55b9b0f34502b4edef7ee50313485d
Author: ta4ilka69 <artem_balin_2004@mail.ru>
Date:   Mon Nov 10 23:50:33 2025 +0300

    log config for different components

diff --git a/backend/src/main/resources/application.properties b/backend/src/main/resources/application.properties
index 83283ba..7568c8e 100644
--- a/backend/src/main/resources/application.properties
+++ b/backend/src/main/resources/application.properties
@@ -27,7 +27,9 @@ app.messaging.articles-topic=/articles
 app.external.markdown-parser.path=${user.dir}/parser
```

```
 #logging
-logging.level.org.springframework.web=DEBUG
+logging.level.org.springframework.web=${LOGGING_LEVEL_SPRING_WEB:TRACE}
+logging.level.org.hibernate.SQL=${LOGGING_LEVEL_HIBERNATE_SQL:DEBUG}
+logging.level.org.hibernate.type.descriptor.sql.BasicBinder=${LOGGING_LEVEL_HIBERNATE_BINDER:TRACE}

 # Actuator & Micrometer Prometheus
 spring.application.name=GMatch-Backend
diff --git a/backend/src/main/resources/logback-spring.xml b/backend/src/main/resources/logback-spring.xml
new file mode 100644
index 0000000..836e2b8
--- /dev/null
+++ b/backend/src/main/resources/logback-spring.xml
@@ -0,0 +1,47 @@
+<?xml version="1.0" encoding="UTF-8"?>
+<configuration>
+
+    <property name="LOGS" value="./logs" />
+
+    <appender name="Console"
+        class="ch.qos.logback.core.ConsoleAppender">
+        <layout class="ch.qos.logback.classic.PatternLayout">
+            <Pattern>
+                %black(%d{ISO8601}) %highlight(%-5level) [%blue(%t)] %yellow(%C{1.}): %msg%n%throwable
+            </Pattern>
+        </layout>
+    </appender>
+
+    <appender name="RollingFile"
+        class="ch.qos.logback.core.rolling.RollingFileAppender">
+        <file>${LOGS}/spring-boot-logger.log</file>
+        <encoder
+            class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
+            <Pattern>%d %p %C{1.} [%t] %m%n</Pattern>
+        </encoder>
+
+        <rollingPolicy
+            class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
+            <!-- rollover daily and when the file reaches 10 MegaBytes -->
+            <fileNamePattern>${LOGS}/archived/spring-boot-logger-%d{yyyy-MM-dd}.%i.log
+            </fileNamePattern>
+            <timeBasedFileNamingAndTriggeringPolicy
+                class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
+                <maxFileSize>10MB</maxFileSize>
+            </timeBasedFileNamingAndTriggeringPolicy>
+        </rollingPolicy>
+    </appender>
+
+    <!-- LOG everything at INFO level -->
+    <root level="info">
+        <appender-ref ref="RollingFile" />
+        <appender-ref ref="Console" />
+    </root>
+
+    <!-- LOG "itmo.is.cw.*" at DEBUG level -->
```

```
+    <logger name="itmo.is.cw" level="debug" additivity="false">
+        <appender-ref ref="RollingFile" />
+        <appender-ref ref="Console" />
+    </logger>
+
+</configuration>
```

git show 76445b3

```
commit 76445b32b9c416e6a15d16134d46a9862c78be1c
Author: ta4ilka69 <artem_balin_2004@mail.ru>
Date:   Mon Nov 10 23:50:49 2025 +0300

    service logging added

diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ArticleService.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ArticleService.java
index edb6437..25e830d 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ArticleService.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ArticleService.java
@@ -40,9 +40,11 @@ import itmo.is.cw.GuitarMatchIS.utils.exceptions.ForbiddenException;
 import itmo.is.cw.GuitarMatchIS.utils.exceptions.ProductNotFoundException;
 import jakarta.servlet.http.HttpServletRequest;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 @Service
 @RequiredArgsConstructor
+@Slf4j
 public class ArticleService {
     private final ArticleRepository articleRepository;
     private final JwtUtils jwtUtils;
@@ -58,6 +60,7 @@ public class ArticleService {
     private String markdownParserPath;

     public List<ArticleDTO> getAcceptedArticles(int from, int size, ArticleSort sortBy, boolean ascending) {
+        log.info("Fetching accepted articles from: {}, size: {}, sortBy: {}, ascending: {}", from, size, sortBy, ascending);
         Sort sort = Sort.by(ascending ? Sort.Direction.ASC : Sort.Direction.DESC,
                 sortBy.getFieldName());
         Pageable page = PageRequest.of(from / size, size, sort);
@@ -70,11 +73,16 @@ public class ArticleService {
     }

     public ArticleDTO getArticleById(Long id) {
+        log.info("Fetching article by id: {}", id);
         return convertToDTO(articleRepository.findById(id)
-                .orElseThrow(() -> new ArticleNotFoundException(String.format("Article with id %s not found", id))));
+                .orElseThrow(() -> {
+                    log.warn("Article with id {} not found", id);
+                    return new ArticleNotFoundException(String.format("Article with id %s not found", id));
+                }));
     }

     public List<ArticleDTO> getArticlesByHeaderContaining(String header, int from, int size) {
+        log.info("Fetching articles with header containing: {}", header);
```

```
     Pageable page = Pagification.createPageTemplate(from, size);

     List<Article> articles = articleRepository.findByHeaderContainingAndAccepted(header, true, page).getContent();
@@ -92,6 +100,7 @@ public class ArticleService {
   }

   public List<ArticleDTO> getArticlesByAuthorId(Long authorId, int from, int size) {
+     log.info("Fetching articles by author id: {}", authorId);
     Pageable page = Pagification.createPageTemplate(from, size);

     List<Article> articles = articleRepository.findByAuthorIdAndAccepted(authorId, true, page).getContent();
@@ -110,37 +119,46 @@ public class ArticleService {

   public boolean moderateArticle(ModerateArticleDTO moderateArticleDTO, HttpServletRequest request) {
     User moderator = findUserByRequest(request);
-    if (!moderator.getIsAdmin())
+     log.info("Moderating article with id: {} by user: {}", moderateArticleDTO.getArticleId(), moderator.getUsername());
+     if (!moderator.getIsAdmin()) {
+       log.warn("User {} has no rights to moderate article", moderator.getUsername());
       throw new ForbiddenException("User have no rights to moderate article");
-
+     }
     Long articleId = moderateArticleDTO.getArticleId();
-    if (!articleRepository.existsById(articleId))
+     if (!articleRepository.existsById(articleId)) {
+       log.warn("Article with id {} not found for moderation", articleId);
       throw new ArticleNotFoundException(String.format("Article with id %s not found", articleId));
-
+     }
     if (!moderateArticleDTO.isAccepted()) {
       // Delete the article if rejected
+       log.info("Article with id {} was rejected and will be deleted", articleId);
       articleRepository.deleteById(articleId);
       simpMessagingTemplate.convertAndSend(articlesTopic, "Article was rejected and deleted");
       return true;
     }

     // Accept the article if not rejected
+     log.info("Article with id {} was approved", articleId);
     simpMessagingTemplate.convertAndSend(articlesTopic, "Article was approved");
     return articleRepository.moderateArticle(articleId, true, moderator.getId());
   }

   @Transactional
   public ArticleDTO createArticle(CreateArticleDTO createArticleDTO, HttpServletRequest request) {
-    if (!productRepository.existsByName(createArticleDTO.getProductName()))
+     log.info("Creating article with header: {}", createArticleDTO.getHeader());
+     if (!productRepository.existsByName(createArticleDTO.getProductName())) {
+       log.warn("Product {} not found for article creation", createArticleDTO.getProductName());
       throw new ProductNotFoundException(String.format("Product %s not found", createArticleDTO.getProductName()));
-
+     }
     Product product = productRepository.findByName(createArticleDTO.getProductName());

-    if (articleRepository.existsByHeader(createArticleDTO.getHeader()))
+     if (articleRepository.existsByHeader(createArticleDTO.getHeader())) {
```

```
+        log.warn("Article with header {} already exists", createArticleDTO.getHeader());
         throw new ArticleAlreadyExistsException(String.format("Article with header %s already exists",
             createArticleDTO.getHeader()));
-
+     }
     User author = findUserByRequest(request);
+     log.info("Article author is {}", author.getUsername());

     Article article = Article.builder()
         .header(createArticleDTO.getHeader())
@@ -150,6 +168,7 @@ public class ArticleService {
         .accepted(false)
         .build();
     articleRepository.save(article);
+     log.info("Article with header {} saved with id {}", article.getHeader(), article.getId());

     ProductArticle productArticle = new ProductArticle();
     productArticle.setArticleId(article.getId());
@@ -157,15 +176,17 @@ public class ArticleService {
     productArticleRepository.save(productArticle);

     simpMessagingTemplate.convertAndSend("/articles", "New Article created for product " + product.getName());
-
+     log.info("Article creation message sent to websocket");
     return convertToDTO(article);
   }

   public List<StatusArticlesDTO> getStatusArticles(int from, int size, HttpServletRequest request) {
     User user = findUserByRequest(request);
-     if (!user.getIsAdmin())
+     log.info("User {} is requesting status articles", user.getUsername());
+     if (!user.getIsAdmin()) {
+       log.warn("User {} has no rights to get moderate articles", user.getUsername());
       throw new ForbiddenException("User have no rights to get moderate articles");
-
+     }
     Pageable page = Pagification.createPageTemplate(from, size);

     List<ProductArticle> productArticles = productArticleRepository.findByAccepted(false, page).getContent();
@@ -184,6 +205,7 @@ public class ArticleService {

   private User findUserByRequest(HttpServletRequest request) {
     String username = jwtUtils.getUserNameFromJwtToken(jwtUtils.parseJwt(request));
+     log.debug("Finding user by username: {}", username);
     return userRepository.findByUsername(username)
         .orElseThrow(() -> new UsernameNotFoundException(
             String.format("Username %s not found", username)));
@@ -244,6 +266,7 @@ public class ArticleService {
       // Get parser path
       String parserPath = markdownParserPath;
       // Build the command - passing text directly
+       log.debug("Converting markdown to html using parser at {}", parserPath);
       ProcessBuilder processBuilder = new ProcessBuilder(
           parserPath, "-c", markdownText);
       processBuilder.redirectErrorStream(true);
@@ -262,12 +285,13 @@ public class ArticleService {
```

```
        int exitCode = process.waitFor();

        if (exitCode != 0) {
-           System.err.println("Parser failed with output: " + output.toString());
+            log.error("Markdown parser failed with exit code {} and output: {}", exitCode, output.toString());
            return markdownText; // Fallback to original text
        }
+       log.debug("Markdown successfully converted to HTML");
        return output.toString();
      } catch (Exception e) {
-       e.printStackTrace();
+       log.error("Exception while converting markdown to html", e);
        return markdownText; // Fallback to original text
      }
    }
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/AuthService.java
b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/AuthService.java
index 4825ddc..5a97741 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/AuthService.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/AuthService.java
@@ -16,9 +16,11 @@ import itmo.is.cw.GuitarMatchIS.security.jwt.JwtUtils;
 import itmo.is.cw.GuitarMatchIS.utils.exceptions.UserAlreadyExistException;
 import itmo.is.cw.GuitarMatchIS.utils.exceptions.UserNotFoundException;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 @Service
 @RequiredArgsConstructor
+@Slf4j
 public class AuthService {

     private final JwtUtils jwtUtils;
@@ -27,10 +29,12 @@ public class AuthService {
     private final AuthenticationManager authenticationManager;

     public AuthResponseDTO register(UserDTO registerUserDto) {
-        if (userRepository.existsByUsername(registerUserDto.getUsername()))
+        log.info("Registering user with username: {}", registerUserDto.getUsername());
+        if (userRepository.existsByUsername(registerUserDto.getUsername())) {
+            log.warn("User with username {} already exists", registerUserDto.getUsername());
             throw new UserAlreadyExistException(
                     String.format("Username %s already exists", registerUserDto.getUsername()));
-
+        }
         User user = User
                 .builder()
                 .username(registerUserDto.getUsername())
@@ -41,7 +45,7 @@ public class AuthService {
                 .build();

         user = userRepository.save(user);
-
+        log.info("User with username {} successfully registered", user.getUsername());
         String token = jwtUtils.generateJwtToken(user.getUsername());
         return new AuthResponseDTO(
                 user.getUsername(),
```

```
@@ -52,15 +56,20 @@ public class AuthService {
     }

     public AuthResponseDTO login(UserDTO loginUserDto) {
+         log.info("Logging in user with username: {}", loginUserDto.getUsername());
         User user = userRepository.findByUsername(loginUserDto.getUsername())
-                .orElseThrow(() -> new UserNotFoundException(
-                        String.format("Username %s not found", loginUserDto.getUsername())));
+                .orElseThrow(() -> {
+                    log.warn("User with username {} not found", loginUserDto.getUsername());
+                    return new UserNotFoundException(
+                            String.format("Username %s not found", loginUserDto.getUsername()));
+                });

         authenticationManager.authenticate(
                 new UsernamePasswordAuthenticationToken(loginUserDto.getUsername(),
                         loginUserDto.getPassword()));

         String token = jwtUtils.generateJwtToken(user.getUsername());
+         log.info("User with username {} successfully logged in", user.getUsername());
         return new AuthResponseDTO(
                 user.getUsername(),
                 user.getIsAdmin(),
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/BrandService.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/BrandService.java
index da6db32..74af02e 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/BrandService.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/BrandService.java
@@ -12,13 +12,16 @@ import itmo.is.cw.GuitarMatchIS.models.Brand;
 import itmo.is.cw.GuitarMatchIS.repository.BrandRepository;
 import itmo.is.cw.GuitarMatchIS.utils.exceptions.BrandNotFoundException;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 @Service
 @RequiredArgsConstructor
+@Slf4j
 public class BrandService {
     private final BrandRepository brandRepository;

     public List<BrandDTO> getBrands(int from, int size) {
+        log.info("Fetching brands from: {} to: {}", from, size);
         Pageable page = Pagification.createPageTemplate(from, size);

         List<Brand> brands = brandRepository.findAll(page).getContent();
@@ -41,7 +44,11 @@ public class BrandService {
     }

     public BrandDTO getBrandById(Long id) {
-        return convertToDTO(brandRepository.findById(id).orElseThrow(() -> new BrandNotFoundException(String.format("Brand with id %s not found", id))));
+        log.info("Fetching brand by id: {}", id);
+        return convertToDTO(brandRepository.findById(id).orElseThrow(() -> {
+            log.warn("Brand with id {} not found", id);
+            return new BrandNotFoundException(String.format("Brand with id %s not found", id));
+        }));
     }
```

```
   private BrandDTO convertToDTO(Brand brand) {
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/FeedbackService.java
b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/FeedbackService.java
index 0026122..c17d7e2 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/FeedbackService.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/FeedbackService.java
@@ -25,9 +25,11 @@ import itmo.is.cw.GuitarMatchIS.security.jwt.JwtUtils;
 import itmo.is.cw.GuitarMatchIS.utils.exceptions.ArticleNotFoundException;
 import itmo.is.cw.GuitarMatchIS.utils.exceptions.ProductNotFoundException;
 import jakarta.servlet.http.HttpServletRequest;
+import lombok.extern.slf4j.Slf4j;

 @Service
 @RequiredArgsConstructor
+@Slf4j
 public class FeedbackService {
     private final FeedbackRepository feedbackRepository;
     private final JwtUtils jwtUtils;
@@ -40,6 +42,7 @@ public class FeedbackService {
     private String feedbackTopic;

     public List<FeedbackDTO> getFeedbackByProductId(Long productId, int from, int size) {
+        log.info("Fetching feedback for product with id: {}", productId);
         Pageable pageable = Pagification.createPageTemplate(from, size);

         List<Feedback> feedbacks = feedbackRepository.findByProductId(productId, pageable).getContent();
@@ -58,6 +61,7 @@ public class FeedbackService {
     }

     public List<FeedbackDTO> getFeedbackByArticleId(Long articleId, int from, int size) {
+        log.info("Fetching feedback for article with id: {}", articleId);
         Pageable pageable = Pagification.createPageTemplate(from, size);

         List<Feedback> feedbacks = feedbackRepository.findByArticleId(articleId, pageable).getContent();
@@ -76,27 +80,39 @@ public class FeedbackService {

     @Transactional
     public Boolean addProductFeedback(CreateProductFeedbackDTO feedbackDTO, HttpServletRequest request) {
+        log.info("Adding feedback for product with id: {}", feedbackDTO.getProductId());
         Product product = productRepository.findById(feedbackDTO.getProductId())
-                .orElseThrow(() -> new ProductNotFoundException(
-                        String.format("Product with id %s not found", feedbackDTO.getProductId())));
+                .orElseThrow(() -> {
+                    log.warn("Product with id {} not found while adding feedback", feedbackDTO.getProductId());
+                    return new ProductNotFoundException(
+                            String.format("Product with id %s not found", feedbackDTO.getProductId()));
+                });

         User user = findUserByRequest(request);
+        log.info("Feedback author is {}", user.getUsername());
         feedbackRepository.addProductFeedback(user.getId(), product.getId(), feedbackDTO.getText(),
                 feedbackDTO.getStars());
         simpMessagingTemplate.convertAndSend(feedbackTopic, "New Feedback created for product");
+        log.info("Feedback for product with id {} successfully added", feedbackDTO.getProductId());
         return true;
```

```
    }

    @Transactional
    public Boolean addArticleFeedback(CreateArticleFeedbackDTO feedbackDTO, HttpServletRequest request) {
+       log.info("Adding feedback for article with id: {}", feedbackDTO.getArticleId());
        Article article = articleRepository.findById(feedbackDTO.getArticleId())
-               .orElseThrow(() -> new ArticleNotFoundException(
-                       String.format("Article with id %s not found", feedbackDTO.getArticleId())));
+               .orElseThrow(() -> {
+                   log.warn("Article with id {} not found while adding feedback", feedbackDTO.getArticleId());
+                   return new ArticleNotFoundException(
+                           String.format("Article with id %s not found", feedbackDTO.getArticleId()));
+               });

        User user = findUserByRequest(request);
+       log.info("Feedback author is {}", user.getUsername());
        feedbackRepository.addArticleFeedback(user.getId(), article.getId(), feedbackDTO.getText(),
                feedbackDTO.getStars());
        simpMessagingTemplate.convertAndSend(feedbackTopic, "New Feedback created for article");
+       log.info("Feedback for article with id {} successfully added", feedbackDTO.getArticleId());
        return true;
    }

diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ForumPostService.java
b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ForumPostService.java
index bdb79f4..0a52f7a 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ForumPostService.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ForumPostService.java
@@ -23,9 +23,11 @@ import itmo.is.cw.GuitarMatchIS.security.jwt.JwtUtils;
 import itmo.is.cw.GuitarMatchIS.utils.exceptions.ForumTopicNotFoundException;
 import jakarta.servlet.http.HttpServletRequest;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 @Service
 @RequiredArgsConstructor
+@Slf4j
 public class ForumPostService {
    private final ForumPostRepository forumPostRepository;
    private final ForumTopicRepository forumTopicRepository;
@@ -34,10 +36,14 @@ public class ForumPostService {
    private final SimpMessagingTemplate simpMessagingTemplate;

    public List<ForumPostDTO> getForumPostsByTopic(Long topicId, int from, int size) {
+       log.info("Fetching forum posts for topic with id: {}", topicId);
        Pageable pageable = Pagification.createPageTemplate(from, size);

        ForumTopic topic = forumTopicRepository.findById(topicId)
-               .orElseThrow(() -> new ForumTopicNotFoundException("Topic with id " + topicId + " not found"));
+               .orElseThrow(() -> {
+                   log.warn("Forum topic with id {} not found", topicId);
+                   return new ForumTopicNotFoundException("Topic with id " + topicId + " not found");
+               });
        List<ForumPost> posts = forumPostRepository.findAllByTopic(topic, pageable).getContent();

        return posts
```

```
@@ -53,12 +59,17 @@ public class ForumPostService {
     }

     public ForumPostDTO createForumPost(CreateForumPostDTO forumPostDTO, HttpServletRequest request) {
+         log.info("Creating forum post for topic with id: {}", forumPostDTO.getForumTopicId());
         ForumTopic topic = forumTopicRepository.findById(forumPostDTO.getForumTopicId())
-                .orElseThrow(() -> new ForumTopicNotFoundException(
-                        String.format("Forum topic with id %s not found", forumPostDTO.getForumTopicId())));
-
+                .orElseThrow(() -> {
+                    log.warn("Forum topic with id {} not found while creating post", forumPostDTO.getForumTopicId());
+                    return new ForumTopicNotFoundException(
+                        String.format("Forum topic with id %s not found", forumPostDTO.getForumTopicId()));
+                });
         User author = findUserByRequest(request);
+         log.info("Forum post author is {}", author.getUsername());
         if (topic.getIsClosed()) {
+             log.warn("Attempt to create post in closed topic with id {}", topic.getId());
             throw new ForumTopicNotFoundException("This topic is closed");
         }
         ForumPost post = ForumPost.builder()
@@ -68,6 +79,7 @@ public class ForumPostService {
                 .content(forumPostDTO.getContent())
                 .build();
         forumPostRepository.save(post);
+         log.info("Forum post with id {} successfully created", post.getId());
         simpMessagingTemplate.convertAndSend("/forum/posts", "Forum post created");

         return convertToDTO(post);
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ForumTopicService.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ForumTopicService.java
index 104d9f8..2cfd11d 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ForumTopicService.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ForumTopicService.java
@@ -25,9 +25,11 @@ import itmo.is.cw.GuitarMatchIS.utils.exceptions.ForumTopicNotFoundException;
 import itmo.is.cw.GuitarMatchIS.utils.exceptions.UserNotFoundException;
 import jakarta.servlet.http.HttpServletRequest;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 @Service
 @RequiredArgsConstructor
+@Slf4j
 public class ForumTopicService {
     private final ForumTopicRepository forumTopicRepository;
     private final JwtUtils jwtUtils;
@@ -35,6 +37,7 @@ public class ForumTopicService {
     private final SimpMessagingTemplate simpMessagingTemplate;

     public List<ForumTopicDTO> getForumTopics(int from, int size) {
+         log.info("Fetching forum topics from: {} to: {}", from, size);
         Pageable pageable = Pagification.createPageTemplate(from, size);

         List<ForumTopic> topics = forumTopicRepository.findAll(pageable).getContent();
@@ -52,9 +55,13 @@ public class ForumTopicService {
     }
```

```
    public List<ForumTopicDTO> getForumTopicsByAuthor(Long authorId, int from, int size) {
+       log.info("Fetching forum topics by author with id: {}", authorId);
        User user = userRepository.findById(authorId)
-               .orElseThrow(() -> new UserNotFoundException(
-                       String.format("User with id %s not found", authorId)));
+               .orElseThrow(() -> {
+                   log.warn("User with id {} not found while fetching forum topics", authorId);
+                   return new UserNotFoundException(
+                           String.format("User with id %s not found", authorId));
+               });

        Pageable pageable = Pagification.createPageTemplate(from, size);
        List<ForumTopic> topics = forumTopicRepository.findAllByAuthor(user, pageable).getContent();
@@ -73,27 +80,38 @@ public class ForumTopicService {

    @Transactional
    public Boolean closeTopic(Long topicId, HttpServletRequest request) {
+       log.info("Closing forum topic with id: {}", topicId);
        ForumTopic topic = forumTopicRepository.findById(topicId)
-               .orElseThrow(() -> new ForumTopicNotFoundException(
-                       String.format("Forum topic with id %s not found", topicId)));
+               .orElseThrow(() -> {
+                   log.warn("Forum topic with id {} not found while closing", topicId);
+                   return new ForumTopicNotFoundException(
+                           String.format("Forum topic with id %s not found", topicId));
+               });

        User user = findUserByRequest(request);
+       log.info("User {} is attempting to close topic {}", user.getUsername(), topicId);
        if (!user.getIsAdmin() && !user.getId().equals(topic.getAuthor().getId())) {
+           log.warn("User {} is not authorized to close topic {}", user.getUsername(), topicId);
            throw new ForbiddenException("You are not authorized to close this topic");
        }

        forumTopicRepository.closeTopic(topic.getId());
        simpMessagingTemplate.convertAndSend("/forum/topics", "Forum topic closed");
+       log.info("Forum topic with id {} successfully closed", topicId);
        return true;
    }

    @Transactional
    public ForumTopicDTO createTopic(CreateForumTopicDTO createForumTopicDTO, HttpServletRequest request) {
-       if (forumTopicRepository.existsByTitle(createForumTopicDTO.getTitle()))
+       log.info("Creating forum topic with title: {}", createForumTopicDTO.getTitle());
+       if (forumTopicRepository.existsByTitle(createForumTopicDTO.getTitle())) {
+           log.warn("Forum topic with title {} already exists", createForumTopicDTO.getTitle());
            throw new ForumTopicAlreadyExistsException(String.format("Forum topic with title %s already exists",
                    createForumTopicDTO.getTitle()));
+       }

        User author = findUserByRequest(request);
+       log.info("Forum topic author is {}", author.getUsername());

        ForumTopic topic = ForumTopic.builder()
                .title(createForumTopicDTO.getTitle())
```

```
@@ -104,7 +122,7 @@ public class ForumTopicService {
             .build();
     forumTopicRepository.save(topic);
     simpMessagingTemplate.convertAndSend("/forum/topics", "Forum topic created");
-
+     log.info("Forum topic with title {} successfully created with id {}", topic.getTitle(), topic.getId());
     return convertToDTO(topic);
   }

@@ -128,18 +146,28 @@ public class ForumTopicService {
   }

   public Boolean isTopicOwner(Long topicId, HttpServletRequest request) {
+     log.info("Checking if user is owner of topic with id: {}", topicId);
     ForumTopic topic = forumTopicRepository.findById(topicId)
-         .orElseThrow(() -> new ForumTopicNotFoundException(
-             String.format("Forum topic with id %s not found", topicId)));
+         .orElseThrow(() -> {
+           log.warn("Forum topic with id {} not found while checking ownership", topicId);
+           return new ForumTopicNotFoundException(
+             String.format("Forum topic with id %s not found", topicId));
+         });

     User user = findUserByRequest(request);
-    return user.getIsAdmin() || user.getId().equals(topic.getAuthor().getId());
+     boolean isOwner = user.getIsAdmin() || user.getId().equals(topic.getAuthor().getId());
+     log.info("User {} is {}owner of topic {}", user.getUsername(), isOwner ? "" : "not ", topicId);
+     return isOwner;
   }

   public ForumTopicDTO getForumTopicById(Long topicId) {
+     log.info("Fetching forum topic by id: {}", topicId);
     ForumTopic topic = forumTopicRepository.findById(topicId)
-         .orElseThrow(() -> new ForumTopicNotFoundException(
-             String.format("Forum topic with id %s not found", topicId)));
+         .orElseThrow(() -> {
+           log.warn("Forum topic with id {} not found", topicId);
+           return new ForumTopicNotFoundException(
+             String.format("Forum topic with id %s not found", topicId));
+         });

     return convertToDTO(topic);
   }
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/MusicianService.java
b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/MusicianService.java
index bce8896..0924e69 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/MusicianService.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/MusicianService.java
@@ -45,6 +45,7 @@ import itmo.is.cw.GuitarMatchIS.utils.exceptions.SubscriptionAlreadyExistsExcept
 import itmo.is.cw.GuitarMatchIS.utils.exceptions.SubscriptionNotFoundException;
 import jakarta.servlet.http.HttpServletRequest;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 import java.util.List;
 import java.util.ArrayList;
```

```
@@ -52,6 +53,7 @@ import java.util.Comparator;

 @Service
 @RequiredArgsConstructor
+@Slf4j
 public class MusicianService {
    private final MusicianRepository musicianRepository;
    private final MusicianGenreRepository musicianGenreRepository;
@@ -64,6 +66,7 @@ public class MusicianService {
    private final ProductRepository productRepository;

    public List<MusicianInfoDTO> getMusician(int from, int size, MusicianSort sortBy, boolean ascending) {
+     log.info("Fetching musicians from: {}, size: {}, sortBy: {}, ascending: {}", from, size, sortBy, ascending);
      Sort sort = Sort.by(ascending ? Sort.Direction.ASC : Sort.Direction.DESC,
                  sortBy.getFieldName());
      Pageable page = PageRequest.of(from / size, size, sort);
@@ -80,21 +83,28 @@ public class MusicianService {

    public Boolean isSubscribed(Long musicianId, HttpServletRequest request) {
      User user = findUserByRequest(request);
-                 return userMusicianRepository.existsByUserAndMusician(user, musicianRepository.findById(musicianId).orElseThrow(() -> new
MusicianNotFoundException("Musician with id %s not found".formatted(musicianId))));
+     log.info("Checking if user {} is subscribed to musician with id: {}", user.getUsername(), musicianId);
+     return userMusicianRepository.existsByUserAndMusician(user, musicianRepository.findById(musicianId).orElseThrow(() -> {
+       log.warn("Musician with id {} not found while checking subscription", musicianId);
+       return new MusicianNotFoundException("Musician with id %s not found".formatted(musicianId));
+     }));
    }

    @Transactional
    public MusicianInfoDTO createMusician(CreateMusicianDTO createMusicianDTO, HttpServletRequest request) {
-     findUserByRequest(request);
-     if (musicianRepository.existsByName(createMusicianDTO.getName()))
+     User user = findUserByRequest(request);
+     log.info("User {} is creating musician with name: {}", user.getUsername(), createMusicianDTO.getName());
+     if (musicianRepository.existsByName(createMusicianDTO.getName())) {
+       log.warn("Musician with name {} already exists", createMusicianDTO.getName());
        throw new MusicianAlreadyExistsException("Musician %s already exists".formatted(createMusicianDTO.getName()));
-
+     }
      Musician musician = Musician.builder()
          .name(createMusicianDTO.getName())
          .subscribers(0)
          .build();

      musician = musicianRepository.save(musician);
+     log.info("Musician with name {} successfully created with id {}", musician.getName(), musician.getId());

      for (Genre genre : createMusicianDTO.getGenres()) {
        musicianGenreRepository.saveByMusicianIdAndGenre(musician.getId(), genre.toString());
@@ -105,42 +115,54 @@ public class MusicianService {
      }

      simpMessagingTemplate.convertAndSend("/musicians", "New musician added");
+     log.info("Musician creation message sent to websocket");
```

```
    return convertToDTOLists(musician, createMusicianDTO.getGenres(), createMusicianDTO.getTypesOfMusician(), new ArrayList<>());
  }

  public Boolean subscribeToMusician(SubscribeDTO subscribeDTO, HttpServletRequest request) {
    User user = findUserByRequest(request);
+    log.info("User {} is subscribing to musician with id: {}", user.getUsername(), subscribeDTO.getMusicianId());

    Musician musician = musicianRepository.findById(subscribeDTO.getMusicianId())
-        .orElseThrow(() -> new MusicianNotFoundException(
-            "Musician with id %s not found".formatted(subscribeDTO.getMusicianId())));
-
+        .orElseThrow(() -> {
+          log.warn("Musician with id {} not found while subscribing", subscribeDTO.getMusicianId());
+          return new MusicianNotFoundException(
+              "Musician with id %s not found".formatted(subscribeDTO.getMusicianId()));
+        });
    if (userMusicianRepository.existsByUserAndMusician(user, musician)) {
+      log.warn("User {} is already subscribed to musician {}", user.getUsername(), musician.getName());
      throw new SubscriptionAlreadyExistsException("You are already subscribed to this musician");
    }
    simpMessagingTemplate.convertAndSend("/musicians", "New subscriber to musician");
    userMusicianRepository.subscribeToMusician(user.getId(), subscribeDTO.getMusicianId());
+    log.info("User {} successfully subscribed to musician {}", user.getUsername(), musician.getName());
    return true;
  }

  @Transactional
  public Boolean unsubscribeFromMusician(SubscribeDTO subscribeDTO, HttpServletRequest request) {
    User user = findUserByRequest(request);
+    log.info("User {} is unsubscribing from musician with id: {}", user.getUsername(), subscribeDTO.getMusicianId());

    Musician musician = musicianRepository.findById(subscribeDTO.getMusicianId())
-        .orElseThrow(() -> new MusicianNotFoundException(
-            "Musician with id %s not found".formatted(subscribeDTO.getMusicianId())));
-
+        .orElseThrow(() -> {
+          log.warn("Musician with id {} not found while unsubscribing", subscribeDTO.getMusicianId());
+          return new MusicianNotFoundException(
+              "Musician with id %s not found".formatted(subscribeDTO.getMusicianId()));
+        });
    if (!userMusicianRepository.existsByUserAndMusician(user, musician)) {
+      log.warn("User {} is not subscribed to musician {}", user.getUsername(), musician.getName());
      throw new SubscriptionNotFoundException("You have no subscription to this musician");
    }
    simpMessagingTemplate.convertAndSend("/musicians", "Deleted subscription to musician");
    userMusicianRepository.deleteByUserAndMusician(user, musician);
+    log.info("User {} successfully unsubscribed from musician {}", user.getUsername(), musician.getName());
    return true;
  }

  public List<MusicianInfoDTO> searchMusicians(String name, int from, int size) {
+    log.info("Searching for musicians with name containing: {}", name);
    Pageable page = Pagification.createPageTemplate(from, size);
    List<Musician> musicians = musicianRepository.findAllByNameContains(name, page).getContent();

@@ -160,6 +182,7 @@ public class MusicianService {
```

```
    }

    public MusicianGenreDTO getMusiciansByGenre(Long musicianId) {
+       log.info("Fetching genres for musician with id: {}", musicianId);
        Musician musician = musicianRepository.findById(musicianId)
            .orElseThrow(() -> new MusicianNotFoundException(
                "Musician with id %s not found".formatted(musicianId)));
@@ -173,6 +196,7 @@ public class MusicianService {
    }

    public MusicianTypeOfMusicianDTO getMusiciansByTypeOfMusician(Long musicianId) {
+       log.info("Fetching types for musician with id: {}", musicianId);
        Musician musician = musicianRepository.findById(musicianId)
            .orElseThrow(() -> new MusicianNotFoundException(
                "Musician with id %s not found".formatted(musicianId)));
@@ -186,6 +210,7 @@ public class MusicianService {
    }

    public MusicianProductDTO getMusicianProducts(String name) {
+       log.info("Fetching products for musician with name: {}", name);
        if (!musicianRepository.existsByName(name))
            throw new MusicianNotFoundException("Musician with name %s not found".formatted(name));

@@ -204,6 +229,7 @@ public class MusicianService {
    }

    public MusicianInfoDTO getMusicianInfo(Long musicianId) {
+       log.info("Fetching info for musician with id: {}", musicianId);
        Musician musician = musicianRepository.findById(musicianId)
            .orElseThrow(() -> new MusicianNotFoundException(
                "Musician with id %s not found".formatted(musicianId)));
@@ -216,6 +242,7 @@ public class MusicianService {

    @Transactional
    public Boolean addProductToMusician(AddProductMusicianDTO addProductMusicianDTO, HttpServletRequest request) {
+       log.info("Adding product with id {} to musician with name {}", addProductMusicianDTO.getProductId(), addProductMusicianDTO.getMusicianName());
        if (!musicianRepository.existsByName(addProductMusicianDTO.getMusicianName()))
            throw new MusicianNotFoundException(
                "Musician with name %s not found".formatted(addProductMusicianDTO.getMusicianName()));
@@ -224,21 +251,25 @@ public class MusicianService {
            throw new ProductNotFoundException(
                "Product with id %s not found".formatted(addProductMusicianDTO.getProductId()));

-       findUserByRequest(request);
+       User user = findUserByRequest(request);
+       log.info("User {} is performing this action", user.getUsername());
        Musician musician = musicianRepository.findByName(addProductMusicianDTO.getMusicianName());
        Product product = productRepository.findById(addProductMusicianDTO.getProductId()).get();

-       if (musicianProductRepository.existsByMusicianAndProduct(musician, product))
+       if (musicianProductRepository.existsByMusicianAndProduct(musician, product)) {
+           log.warn("Product {} already exists in musician {}", product.getName(), musician.getName());
            throw new ProductMusicianAlreadyExists("Product %s already exists in musician %s".formatted(product.getName(),
                musician.getName()));
-
+       }
```

```
      musicianProductRepository
          .save(MusicianProduct.builder().musicianId(musician.getId()).productId(product.getId()).build());
+     log.info("Product {} successfully added to musician {}", product.getName(), musician.getName());
      return true;
    }

    @Transactional
    public Boolean deleteProductFromMusician(AddProductMusicianDTO addProductMusicianDTO, HttpServletRequest request) {
+                       log.info("Deleting product with id {} from musician with name {}", addProductMusicianDTO.getProductId(),
addProductMusicianDTO.getMusicianName());
      if (!musicianRepository.existsByName(addProductMusicianDTO.getMusicianName()))
        throw new MusicianNotFoundException(
            "Musician with name %s not found".formatted(addProductMusicianDTO.getMusicianName()));
@@ -246,15 +277,18 @@ public class MusicianService {
      if (!productRepository.existsById(addProductMusicianDTO.getProductId()))
        throw new ProductNotFoundException(
            "Product with id %s not found".formatted(addProductMusicianDTO.getProductId()));
-     findUserByRequest(request);
+     User user = findUserByRequest(request);
+     log.info("User {} is performing this action", user.getUsername());
      Musician musician = musicianRepository.findByName(addProductMusicianDTO.getMusicianName());
      Product product = productRepository.findById(addProductMusicianDTO.getProductId()).get();

-     if (!musicianProductRepository.existsByMusicianAndProduct(musician, product))
+     if (!musicianProductRepository.existsByMusicianAndProduct(musician, product)) {
+       log.warn("Product {} not found in musician {}", product.getName(), musician.getName());
        throw new ProductMusicianNotFoundException("Product %s not found in musician %s".formatted(product.getName(),
            musician.getName()));
-
+     }
      musicianProductRepository.deleteByMusicianAndProduct(musician, product);
+     log.info("Product {} successfully deleted from musician {}", product.getName(), musician.getName());
      return true;
    }

diff                                --git                                a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ProductService.java
b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ProductService.java
index 16920e6..5871954 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ProductService.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ProductService.java
@@ -28,12 +28,14 @@ import itmo.is.cw.GuitarMatchIS.repository.ProductGenreRepository;
 import itmo.is.cw.GuitarMatchIS.repository.ProductRepository;
 import itmo.is.cw.GuitarMatchIS.repository.ShopProductRepository;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 import java.util.List;
 import java.util.Comparator;

 @Service
 @RequiredArgsConstructor
+@Slf4j
 public class ProductService {
    private final ProductRepository productRepository;
    private final BrandRepository brandRepository;
@@ -45,9 +47,11 @@ public class ProductService {
```

```
    private final ShopProductRepository shopProductRepository;

    public List<ProductGenreDTO> getProductsByBrandName(String brandName, int from, int size) {
+       log.info("Fetching products for brand: {} from: {} to: {}", brandName, from, size);
        Pageable page = Pagification.createPageTemplate(from, size);

        if (!brandRepository.existsByName(brandName)) {
+           log.warn("Brand with name {} not found", brandName);
            throw new BrandNotFoundException("Brand %s not found".formatted(brandName));
        }

@@ -65,6 +69,7 @@ public class ProductService {
    }

    public List<ProductGenreDTO> getProductsByTypeOfProduct(TypeOfProduct typeOfProduct, int from, int size) {
+       log.info("Fetching products by type: {} from: {} to: {}", typeOfProduct, from, size);
        Pageable page = Pagification.createPageTemplate(from, size);

        return productRepository.findAllByTypeOfProduct(typeOfProduct, page).getContent().stream()
@@ -79,7 +84,9 @@ public class ProductService {
    }

    public ProductGenreDTO getGenresByProductName(String productName) {
+       log.info("Fetching genres for product: {}", productName);
        if (!productRepository.existsByName(productName)) {
+           log.warn("Product with name {} not found", productName);
            throw new ProductNotFoundException("Product %s not found".formatted(productName));
        }

@@ -94,7 +101,11 @@ public class ProductService {
    }

    public ProductGenreDTO getProductsById(long id){
-       Product p = productRepository.findById(id).orElseThrow(() -> new ProductNotFoundException("Product id = %d not found".formatted(id)));
+       log.info("Fetching product by id: {}", id);
+       Product p = productRepository.findById(id).orElseThrow(() -> {
+           log.warn("Product with id {} not found", id);
+           return new ProductNotFoundException("Product id = %d not found".formatted(id));
+       });
        return ProductGenreDTO.builder()
                .product(convertToDTO(p))
                .genres(productGenreRepository.findByProduct(p).stream()
@@ -104,6 +115,7 @@ public class ProductService {
    }

    public List<ProductGenreDTO> getProductsByNameContains(String name, int from, int size) {
+       log.info("Fetching products with name containing: {}", name);
        Pageable page = Pagification.createPageTemplate(from, size);

        return productRepository.findAllByNameContains(name, page).getContent().stream()
@@ -118,9 +130,13 @@ public class ProductService {
    }

    public ProductArticleDTO getProductArticles(long productId, int from, int size) {
+       log.info("Fetching articles for product with id: {}", productId);
        Pageable page = Pagification.createPageTemplate(from, size);
```

```
-                Product product = productRepository.findById(productId).orElseThrow(() -> new ProductNotFoundException("Product id = %d not found".formatted(productId)));
+      Product product = productRepository.findById(productId).orElseThrow(() -> {
+        log.warn("Product with id {} not found while fetching articles", productId);
+        return new ProductNotFoundException("Product id = %d not found".formatted(productId));
+      });

       List<ProductArticle> productArticles = productArticleRepository
           .findByProductIdAndAccepted(productId, true, page)
@@ -141,8 +157,12 @@ public class ProductService {
    }

    public ProductShopsDTO getProductShops(long productId, int from, int size) {
+      log.info("Fetching shops for product with id: {}", productId);
       Pageable page = Pagification.createPageTemplate(from, size);
-                Product product = productRepository.findById(productId).orElseThrow(() -> new ProductNotFoundException("Product id = %d not found".formatted(productId)));
+      Product product = productRepository.findById(productId).orElseThrow(() -> {
+        log.warn("Product with id {} not found while fetching shops", productId);
+        return new ProductNotFoundException("Product id = %d not found".formatted(productId));
+      });
       List<ShopProduct> shopProducts = shopProductRepository.findAllByProduct(product, page).getContent();
       List<Shop> shops = shopProducts.stream().map(shopProduct -> shopProduct.getShop()).toList();

@@ -177,6 +197,7 @@ public class ProductService {
         boolean ascending,
         int from, int size) {

+    log.info("Fetching products by filter");
     Specification<Product> specification = Specification.where(ProductSpecification.hasBrand(brandId))
         .and(ProductSpecification.hasName(name))
         .and(ProductSpecification.hasRateBetween(minRate, maxRate))
@@ -202,8 +223,12 @@ public class ProductService {
 }

    public List<MusicianInfoDTO> getMusiciansByProductId(long productId, int from, int size) {
+      log.info("Fetching musicians for product with id: {}", productId);
       Pageable page = Pagification.createPageTemplate(from, size);
-                Product product = productRepository.findById(productId).orElseThrow(() -> new ProductNotFoundException("Product id = %d not found".formatted(productId)));
+      Product product = productRepository.findById(productId).orElseThrow(() -> {
+        log.warn("Product with id {} not found while fetching musicians", productId);
+        return new ProductNotFoundException("Product id = %d not found".formatted(productId));
+      });
       List<MusicianProduct> musicianProducts = musicianProductRepository.findByProduct(product, page).getContent();
       List<Musician> musicians = musicianProducts.stream().map(musicianProduct -> musicianProduct.getMusician()).toList();
       return musicians.stream().map(musician -> MusicianInfoDTO.builder()
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ShopService.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ShopService.java
index 3e1ffef..c7f2da8 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ShopService.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/ShopService.java
@@ -16,14 +16,17 @@ import itmo.is.cw.GuitarMatchIS.repository.ShopProductRepository;
 import itmo.is.cw.GuitarMatchIS.repository.ShopRepository;
 import itmo.is.cw.GuitarMatchIS.utils.exceptions.ShopNotFoundException;
```

```
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 @Service
 @RequiredArgsConstructor
+@Slf4j
 public class ShopService {
    private final ShopRepository shopRepository;
    private final ShopProductRepository shopProductRepository;

    public List<ShopDTO> getShops(int from, int size) {
+       log.info("Fetching shops from: {} to: {}", from, size);
        Pageable page = Pagification.createPageTemplate(from, size);

        List<Shop> shops = shopRepository.findAll(page).getContent();
@@ -47,10 +50,14 @@ public class ShopService {
    }

    public ShopProductDTO getShopProducts(Long shopId, int from, int size) {
+       log.info("Fetching products for shop with id: {}", shopId);
        Pageable page = Pagification.createPageTemplate(from, size);

        Shop shop = shopRepository.findById(shopId)
-           .orElseThrow(() -> new ShopNotFoundException("Shop with id " + shopId + " not found"));
+           .orElseThrow(() -> {
+               log.warn("Shop with id {} not found while fetching products", shopId);
+               return new ShopNotFoundException("Shop with id " + shopId + " not found");
+           });

        List<ShopProduct> shopProducts = shopProductRepository.findAllByShop(shop, page).getContent();

@@ -61,7 +68,11 @@ public class ShopService {
    }

    public ShopDTO getShopById(Long id) {
-       return convertToDTO(shopRepository.findById(id).orElseThrow(() -> new ShopNotFoundException(String.format("Shop with id %s not found", id))));
+       log.info("Fetching shop by id: {}", id);
+       return convertToDTO(shopRepository.findById(id).orElseThrow(() -> {
+           log.warn("Shop with id {} not found", id);
+           return new ShopNotFoundException(String.format("Shop with id %s not found", id));
+       }));
    }

    private ProductOfShopDTO convertToDTO(ShopProduct shopProduct) {
diff --git a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/UserService.java b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/UserService.java
index 7f48e65..10106d8 100644
--- a/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/UserService.java
+++ b/backend/src/main/java/itmo/is/cw/GuitarMatchIS/service/UserService.java
@@ -38,9 +38,11 @@ import itmo.is.cw.GuitarMatchIS.utils.exceptions.ProductMusicianAlreadyExists;
 import itmo.is.cw.GuitarMatchIS.utils.exceptions.ProductNotFoundException;
 import jakarta.servlet.http.HttpServletRequest;
 import lombok.RequiredArgsConstructor;
+import lombok.extern.slf4j.Slf4j;

 @Service
```

```
 @RequiredArgsConstructor
+@Slf4j
 public class UserService {
     private final UserRepository userRepository;
     private final UserGenreRepository userGenreRepository;
@@ -54,42 +56,53 @@ public class UserService {
     private final MusicianProductRepository musicianProductRepository;

     public Role getRoleByUsername(String username) {
+         log.info("Getting role for username: {}", username);
         User user = userRepository.findByUsername(username)
-                .orElseThrow(() -> new UsernameNotFoundException(
-                        String.format("Username %s not found", username)));
+                 .orElseThrow(() -> {
+                     log.warn("User with username {} not found while getting role", username);
+                     return new UsernameNotFoundException(
+                             String.format("Username %s not found", username));
+                 });
         return user.getIsAdmin() ? Role.ADMIN : Role.USER;
     }

     public List<Genre> getGenresByUser(HttpServletRequest request) {
         User user = findUserByRequest(request);
+         log.info("Getting genres for user: {}", user.getUsername());
         return userGenreRepository.findByUser(user).stream().map(UserGenre::getGenre).toList();
     }

     @Transactional
     public Boolean setGenresToUser(HttpServletRequest request, List<Genre> genres) {
         User user = findUserByRequest(request);
+         log.info("Setting genres for user: {}", user.getUsername());
         userGenreRepository.deleteAllByUser(user.getId());
         genres.forEach(genre -> userGenreRepository.saveByUserIdAndGenre(user.getId(), genre.getCapsValue()));
+         log.info("Successfully set genres for user: {}", user.getUsername());
         return true;
     }

     @Transactional
     public Boolean setTypesOfMusiciansToUser(HttpServletRequest request, List<TypeOfMusician> typesOfMusicians) {
         User user = findUserByRequest(request);
+         log.info("Setting types of musicians for user: {}", user.getUsername());
         userTypeOfMusicianRepository.deleteAllByUser(user.getId());
         typesOfMusicians.forEach(typeOfMusician -> userTypeOfMusicianRepository
                 .saveByUserIdAndTypeOfMusician(user.getId(), typeOfMusician.getCapsValue()));
+         log.info("Successfully set types of musicians for user: {}", user.getUsername());
         return true;
     }

     public List<TypeOfMusician> getTypesOfMusiciansByUser(HttpServletRequest request) {
         User user = findUserByRequest(request);
+         log.info("Getting types of musicians for user: {}", user.getUsername());
         return userTypeOfMusicianRepository.findByUser(user).stream().map(UserTypeOfMusician::getTypeOfMusician)
                 .toList();
     }

     public List<ProductDTO> getUserProducts(HttpServletRequest request) {
```

```
        User user = findUserByRequest(request);
+        log.info("Getting products for user: {}", user.getUsername());

        List<UserProduct> userProducts = userProductRepository.findByUser(user);

@@ -98,6 +111,7 @@ public class UserService {

    public List<MusicianInfoDTO> getSubscribedMusicians(HttpServletRequest request) {
        User user = findUserByRequest(request);
+        log.info("Getting subscribed musicians for user: {}", user.getUsername());
        List<Musician> musicians = userMusicianRepository.findByUser(user).stream()
                .map(UserMusician::getMusician).toList();

@@ -112,48 +126,62 @@ public class UserService {

    @Transactional
    public Boolean addProductToUser(AddUserProductDTO addUserProductDTO, HttpServletRequest request) {
-        if (!productRepository.existsById(addUserProductDTO.getProductId()))
+        log.info("Adding product with id {} to user", addUserProductDTO.getProductId());
+        if (!productRepository.existsById(addUserProductDTO.getProductId())) {
+            log.warn("Product with id {} not found while adding to user", addUserProductDTO.getProductId());
            throw new ProductNotFoundException(
                    "Product with id %s not found".formatted(addUserProductDTO.getProductId()));
-
+        }
        User user = findUserByRequest(request);
+        log.info("User is {}", user.getUsername());

        Product product = productRepository.findById(addUserProductDTO.getProductId()).get();

-        if (userProductRepository.existsByUserAndProduct(user, product))
+        if (userProductRepository.existsByUserAndProduct(user, product)) {
+            log.warn("Product {} already exists for user {}", product.getName(), user.getUsername());
            throw new ProductMusicianAlreadyExists(
                    "Product %s already exists in user %s".formatted(product.getName(),
                            user.getUsername()));
-
+        }
        userProductRepository
                .save(UserProduct.builder().userId(user.getId()).productId(product.getId())
                        .build());
+        log.info("Successfully added product {} to user {}", product.getName(), user.getUsername());
        return true;
    }

    @Transactional
    public Boolean deleteProductFromUser(AddUserProductDTO addUserProductDTO, HttpServletRequest request) {
-        if (!productRepository.existsById(addUserProductDTO.getProductId()))
+        log.info("Deleting product with id {} from user", addUserProductDTO.getProductId());
+        if (!productRepository.existsById(addUserProductDTO.getProductId())) {
+            log.warn("Product with id {} not found while deleting from user", addUserProductDTO.getProductId());
            throw new ProductNotFoundException(
                    "Product with id %s not found".formatted(addUserProductDTO.getProductId()));
-
+        }
        User user = findUserByRequest(request);
```

```
+            log.info("User is {}", user.getUsername());

            Product product = productRepository.findById(addUserProductDTO.getProductId()).get();

-           if (!userProductRepository.existsByUserAndProduct(user, product))
+           if (!userProductRepository.existsByUserAndProduct(user, product)) {
+               log.warn("Product {} not found for user {}", product.getName(), user.getUsername());
                throw new ProductNotFoundException(
                        "Product %s not found in user %s".formatted(product.getName(),
                                user.getUsername()));
-
+           }
            userProductRepository.deleteByUserAndProduct(user, product);
+           log.info("Successfully deleted product {} from user {}", product.getName(), user.getUsername());
            return true;
        }

        public UserInfoDTO getUserInfoById(Long id) {
+           log.info("Getting user info for id: {}", id);
            User user = userRepository.findById(id)
-                   .orElseThrow(() -> new UsernameNotFoundException(
-                           String.format("User with id %s not found", id)));
+                   .orElseThrow(() -> {
+                       log.warn("User with id {} not found", id);
+                       return new UsernameNotFoundException(
+                               String.format("User with id %s not found", id));
+                   });
            return UserInfoDTO.builder().id(user.getId()).username(user.getUsername()).build();
        }
```

git show 276c8eb

```
commit 276c8eb735fc2193ad5a113f44ef00c19524a871
Author: ta4ilka69 <artem_balin_2004@mail.ru>
Date:   Mon Nov 10 23:57:43 2025 +0300

    log fix + ignore logs in git

diff --git a/.gitignore b/.gitignore
index 1c4dca5..00f2881 100644
--- a/.gitignore
+++ b/.gitignore
@@ -1,4 +1,5 @@
 .vscode/settings.json
 *.env
 **/*.env
-.DS_Store
\ No newline at end of file
+.DS_Store
+backend/logs/spring-boot-logger.log
diff --git a/backend/src/main/resources/logback-spring.xml b/backend/src/main/resources/logback-spring.xml
index 836e2b8..73fbddb 100644
--- a/backend/src/main/resources/logback-spring.xml
+++ b/backend/src/main/resources/logback-spring.xml
```

```
@@ -7,7 +7,7 @@
        class="ch.qos.logback.core.ConsoleAppender">
        <layout class="ch.qos.logback.classic.PatternLayout">
          <Pattern>
-             %black(%d{ISO8601}) %highlight(%-5level) [%blue(%t)] %yellow(%C{1.}): %msg%n%throwable
+              %black(%d{ISO8601}) %highlight(%-5level) [%blue(%t)] %yellow(%C{1}): %msg%n%throwable
          </Pattern>
        </layout>
    </appender>
@@ -17,18 +17,15 @@
        <file>${LOGS}/spring-boot-logger.log</file>
        <encoder
          class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
-         <Pattern>%d %p %C{1.} [%t] %m%n</Pattern>
+          <Pattern>%d %p %C{1} [%t] %m%n</Pattern>
        </encoder>

-       <rollingPolicy
-         class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
+       <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
          <!-- rollover daily and when the file reaches 10 MegaBytes -->
-         <fileNamePattern>${LOGS}/archived/spring-boot-logger-%d{yyyy-MM-dd}.%i.log
-         </fileNamePattern>
-         <timeBasedFileNamingAndTriggeringPolicy
-           class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
-           <maxFileSize>10MB</maxFileSize>
-         </timeBasedFileNamingAndTriggeringPolicy>
+          <fileNamePattern>${LOGS}/archived/spring-boot-logger-%d{yyyy-MM-dd}.%i.log</fileNamePattern>
+          <maxFileSize>10MB</maxFileSize>
+          <totalSizeCap>1GB</totalSizeCap>
+          <maxHistory>30</maxHistory>
        </rollingPolicy>
    </appender>
```

Заключение

Теперь статистику работы информационной системы можно отслеживать с помощью Grafana и Prometheus дэшбордов. Также было добавлено логирование компонентов нашей системы.