Backend 개발 명세서

1. 기술 스택

- **Framework**: Spring Boot 3.5.3
- 언어: Java 17
- 데이터베이스: MariaDB
- 캐시: Redis
- 빌드: Maven
- 보안: Spring Security + JWT
- **API** 문서화: SpringDoc OpenAPI (Swagger)

2. 프로젝트 구조

```
backend/
├── src/main/java/com/rookies/log2doc/
│   ├── controller/          # REST API 컨트롤러
│   │   ├── AuthController.java        # 인증 관련 API
│   │   ├── DocumentController.java    # 문서 관리 API
│   │   ├── ErrorReportController.java # 에러 리포트 API
│   │   └── UserController.java        # 사용자 관리 API
│   ├── service/             # 비즈니스 로직
│   │   ├── AuthService.java           # 인증 서비스
│   │   ├── DocumentService.java       # 문서 관리 서비스
│   │   ├── ErrorReportService.java    # 에러 리포트 서비스
│   │   ├── CeoUserService.java        # CEO 전용 사용자 관리
│   │   └── RefreshTokenService.java   # 토큰 관리 서비스
│   ├── repository/          # 데이터 액세스 레이어
│   │   ├── UserRepository.java
│   │   ├── ErrorReportRepository.java
│   │   ├── DocumentRepository.java
│   │   └── RoleRepository.java
│   ├── entity/              # JPA 엔티티
│   │   ├── User.java                  # 사용자 정보
│   │   ├── ErrorReport.java           # 에러 리포트 정보
│   │   ├── Document.java              # 문서 정보
│   │   ├── Role.java                  # 권한/직급 정보
│   │   └── RefreshToken.java          # 리프레시 토큰
│   ├── dto/                 # 데이터 전송 객체
│   │   ├── request/                   # 요청 DTO
│   │   └── response/                  # 응답 DTO
```

```
│   ├── config/            # 설정 클래스
│   │   ├── SecurityConfig.java        # Spring Security 설정
│   │   ├── DataInitializer.java       # 초기 데이터 설정
│   │   ├── WebMvcConfig.java          # MVC 설정
│   │   └── FileStorageConfig.java     # 파일 저장 설정
│   ├── security/          # 보안 관련
│   │   ├── jwt/                   # JWT 유틸리티
│   │   └── services/              # 인증 서비스
│   ├── exception/         # 예외 처리
│   └── log/               # 로깅 인터셉터
├── src/main/resources/
│   └── application.properties         # 설정 파일
├── Dockerfile                     # Docker 빌드 설정
└── pom.xml                        # Maven 의존성 설정
```

3. 주요 설정 (application.properties)

```
# 서버 설정
server.port=8080

# 데이터베이스 설정 (MariaDB)
spring.datasource.url=jdbc:mariadb://localhost:3306/log2doc
spring.datasource.username=admin
spring.datasource.password=admin
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver

# JPA/Hibernate 설정
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect

# Redis 설정
spring.data.redis.host=localhost
spring.data.redis.port=6379
spring.data.redis.password=1234

# JWT 설정
app.jwt.secret=mySecretKey12345678901234567890123456789012345678901234567890
app.jwt.expiration=86400000
app.jwt.refresh.expiration=604800000

# Flask AI 서버 연동 설정
flask.base.url=http://localhost:5001
```

```
flask.endpoint.analyze=/analyze
flask.endpoint.analyze-advanced=/analyze-advanced
flask.connection.timeout=30000

# 파일 업로드 설정
file.upload.path.nfs=/app/document
spring.servlet.multipart.max-file-size=50MB
spring.servlet.multipart.max-request-size=50MB
```

4. 주요 클래스

4.1 Entity

4.1.1 User (사용자)

```
@Entity
@Table(name = "users")
public class User {
    private Long id;
    private String username;
    private String email;
    private String password;
    private String phone;
    private Boolean isActive;
    private Boolean isEmailVerified;
    private LocalDateTime createdAt;
    private LocalDateTime updatedAt;

    @OneToOne(fetch = FetchType.LAZY)
    private Role role;  // 권한/직급 정보
}
```

4.1.2 ErrorReport (에러 리포트)

```
@Entity
@Table(name = "error_report")
public class ErrorReport {
    private Long id;
    private String reportTitle;        // 리포트 제목
    private String reportPreview;      // 리포트 간략 설명
    private ReportCategory reportCategory;  // ATTACK, VALID, INVALID
    private String reportPath;         // 리포트 파일 경로
```

```
    private ReportStatus reportStatus; // NOT_STARTED, IN_PROGRESS, COMPLETED
    private String reportComment;      // 관리자 코멘트
    private Boolean isDeleted;
    private LocalDateTime createdDt;
    private LocalDateTime deletedDt;
}
```

4.1.3 Role (권한/직급)

```
@Entity
@Table(name = "roles")
public class Role {
    private Long id;
    private RoleName name;  // INTERN(1) ~ CEO(11)
    private String description;

    public enum RoleName {
        INTERN(1, "인턴"),
        STAFF(2, "사원"),
        SENIOR_STAFF(3, "주임"),
        ASSISTANT_MANAGER(4, "대리"),
        MANAGER(5, "과장"),
        SENIOR_MANAGER(6, "차장"),
        DIRECTOR(7, "부장"),
        VICE_PRESIDENT(8, "이사"),
        PRESIDENT(9, "상무"),
        EXECUTIVE_VICE_PRESIDENT(10, "전무"),
        CEO(11, "대표이사");
    }
}
```

4.1.4 Document (문서)

```
@Entity
@Table(name = "documents")
public class Document {
    private Long id;
    private String title;
    private String content;
    private String originalFilename;
    private String storedFilename;
    private String filePath;
    private String author;
    private LocalDateTime uploadDate;
```

```
    @ManyToOne
    private Role readRole;  // 읽기 권한 제한
}
```

4.2 Controller

4.2.1 AuthController (인증)

- POST /api/v1/auth/signin - 로그인
- POST /api/v1/auth/refreshtoken - 토큰 갱신
- POST /api/v1/auth/signout - 로그아웃

4.2.2 ErrorReportController (에러 리포트)

- GET /api/v1/error-reports/analytics/daily-count - 일별 에러 카운트
- GET /api/v1/error-reports/analytics/statistics - 리포트 통계
- GET /api/v1/error-reports/list/latest - 최신 리포트 목록
- GET /api/v1/error-reports/list/attacks - 공격 탐지 리포트
- GET /api/v1/error-reports/{id} - 리포트 상세 조회
- PATCH /api/v1/error-reports/{id}/status/in-progress - 상태 변경
- DELETE /api/v1/error-reports/{id} - 리포트 삭제

4.2.3 DocumentController (문서)

- POST /documents/upload - 문서 업로드
- GET /documents - 문서 목록 조회
- GET /documents/{id}/download - 문서 다운로드

4.3 Service (비즈니스 로직)

4.3.1 ErrorReportService

- Flask AI 서버와 연동하여 보안 로그 분석 워크플로우 관리
- 분석 결과에 따른 리포트 생성 및 상태 변경 처리
- 공격 탐지 시 긴급 알림 및 대응 절차 실행

4.3.2 DocumentService

- 파일 업로드부터 다운로드까지 전체 문서 생명주기 관리
- 사용자 직급에 따른 문서 접근 권한 검증
- 문서 버전 관리 및 검색 엔진 기능 제공
- 파일 무결성 검사 및 보안 스캔 수행

4.3.3 RefreshTokenService

- 다중 디바이스 로그인 지원을 위한 토큰 관리
- 토큰 보안 강화 및 성능 최적화 전략 구현
- 토큰 만료 및 갱신 주기 관리

4.3.4 AuthService

- 통합 인증 처리 및 보안 정책 적용
- 사용자 권한과 직급 매핑 관리
- 인증 실패 시 보안 이벤트 로깅 및 대응

4.4 Repository

4.4.1 ErrorReportRepository

- 대용량 보안 로그 데이터의 효율적인 조회 및 통계 생성
- 실시간 모니터링을 위한 최적화된 쿼리 제공
- 일별/주별/월별 에러 발생 추세 분석 데이터 제공

4.4.2 UserRepository

- 사용자 정보의 안전한 CRUD 작업 처리
- 사용자 활동 이력 추적 및 감사 기능 지원
- 계정 보안 이벤트 로깅 및 모니터링

4.4.3 DocumentRepository

- 문서 메타데이터의 체계적인 관리 및 고급 검색 기능
- 문서 접근 통계 및 사용 패턴 분석
- 대용량 파일 처리를 위한 성능 최적화

4.4.4 RoleRepository

- 복잡한 조직 직급 체계의 권한 구조 관리
- 직급별 권한 상속 및 위임 기능 구현
- 권한 변경 이력 추적 및 감사

4.4.5 RefreshTokenRepository

- 토큰의 안전한 저장 및 관리
- 토큰 만료 및 정리 작업 자동화
- 보안 강화를 위한 토큰 암호화 저장

5. 보안 구현

5.1 JWT 토큰 인증

- Access Token (24시간) + Refresh Token (7일) 구조
- Redis를 통한 토큰 캐싱 및 블랙리스트 관리
- 토큰 자동 갱신 지원

5.2 Spring Security 설정

- JWT 기반 Stateless 인증
- 권한별 API 접근 제어
- CORS 설정 포함

5.3 권한 기반 접근 제어

- 11단계 직급 시스템 (인턴 ~ CEO)
- 상위 직급이 하위 직급 데이터 접근 가능
- 문서별 읽기 권한 제한

6. AI 모듈 연동 (Flask)

6.1 Flask 서버 통신

- HTTP 클라이언트를 통한 비동기 통신
- 로그 분석 요청 전송
- 생성된 리포트 수신 및 저장

6.2 API 엔드포인트

- /analyze - 기본 로그 분석
- /analyze-advanced - 고급 로그 분석
- /receive-report - 리포트 수신

7. 파일 처리

7.1 문서 업로드

- MultipartFile을 통한 파일 업로드
- UUID 기반 파일명 생성
- 지원 형식: PDF, DOCX, TXT, 이미지
- 최대 파일 크기: 50MB

7.2 파일 저장 관리

- NFS 기반 파일 저장 (/app/document)
- 원본 파일명과 저장 파일명 분리 관리
- 파일 메타데이터 DB 저장

8. 예외 처리

8.1 Custom Exception 클래스

- PermissionDeniedException - 권한 부족
- TokenRefreshException - 토큰 갱신 실패

8.2 GlobalExceptionHandler

- 전역 예외 처리
- 통일된 에러 응답 형식
- 적절한 HTTP 상태 코드 반환

9. 로깅 및 모니터링

9.1 LoggingInterceptor

- 모든 API 요청/응답 로깅
- 에러 리포트 관련 특별 로깅
- 사용자 활동 추적

9.2 Spring Boot Actuator

- 애플리케이션 헬스 체크
- 메트릭 수집
- DB 연결 상태 모니터링

10. 빌드 및 배포

10.1 Maven 빌드

```
<!-- 주요 의존성 -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
```

```
</dependency>
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
<dependency>
   <groupId>io.jsonwebtoken</groupId>
   <artifactId>jjwt-api</artifactId>
</dependency>
<dependency>
   <groupId>org.mariadb.jdbc</groupId>
   <artifactId>mariadb-java-client</artifactId>
</dependency>
```

10.2 Docker 설정

```
FROM openjdk:17-alpine AS builder
WORKDIR /app
COPY pom.xml ./
RUN ./mvnw dependency:go-offline -B
COPY src ./src
RUN ./mvnw clean package -DskipTests

FROM openjdk:17-alpine
COPY --from=builder /app/target/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```

11. 데이터베이스 스키마

11.1 주요 테이블

- users - 사용자 정보
- roles - 권한/직급 정보
- error_report - AI 생성 에러 리포트
- documents - 업로드 문서 정보
- refresh_tokens - 리프레시 토큰 관리
- category_types - 문서 카테고리

11.2 관계 설정

- User ↔ Role (1:1)
- User ↔ RefreshToken (1:N)
- Document ↔ Role (N:1, 읽기 권한)
- ErrorReport (독립적 관리)

12. 개발 및 운영 환경

12.1 개발 환경

- Java 17
- Maven 3.6+
- MariaDB 10.5+
- Redis 6.0+
- IDE: IntelliJ IDEA / Eclipse

12.2 운영 환경

- Docker 컨테이너 기반 배포
- MariaDB 클러스터
- Redis 클러스터
- Nginx 리버스 프록시