포팅 메뉴얼

📂 포팅 메뉴얼
├── 📄 01. 개발 도구
├── 📄 02. 개발 환경
├── 📄 03. 환경 변수
├── 📄 04. CI/CD 구축
└── 📄 05. 외부 서비스 사용

📄 01. 개발 도구

- **IDE**
  - IntelliJ 2023.3.8, Visual Studio Code 1.97.2
- **API 테스트**
  - Swagger
- **형상관리**
  - Git + GitLab
- **가상환경**
  - Pyenv
- **이슈관리**
  - Jira
- **문서/디자인 툴**
  - Notion, Figma, Figjam

📄 02. 개발 환경

OS

- Ubuntu : 24.04.2 LTS

BackEnd

- Java OpenJDK 17
- Spring Boot : 3.4.2
- Spring Security : 6.3.1.1
- Spring Data JPA : 3.4.2
- Gradle : 8.12.1
- JWT : 0.11.5
- Spring Websocket : 6.2.2
- Lombok : 1.18.36

BigData

- Python : 3.9.12
- FastAPI : 0.115.12
- Uvicorn : 0.34.0
- faiss-cpu : 1.7.4
- findspark : 2.0.1
- fpdf2 : 2.8.2
- google-cloud-vision : 3.10.1
- huggingface-hub : 0.29.3
- konlpy : 0.6.0
- numpy : 2.0.2
- scikit-learn : 1.6.1
- SQLAlchemy : 2.0.39

Android

- agp : 8.7.3
- kotlin : 1.9.24
- coreKtx : 1.15.0
- retrofit : 2.9.0
- okhttp : 4.9.0
- stomp : 1.6.6
- rxjava : 2.2.5
- rxandroid : 2.1.0

Database

- MySQL : 8.0
- AWS S3

Infra

- AWS EC2
- Gitlab Webhook
- Docker : 26.1.3
- Docker-compose : 2.24.1
- Jenkins : 2.479.3
- Nginx : 1.18.0
- Cerbot : 1.21.0

포트 정보

서비스	포트 번호
BackEnd	8080
FastAPI	8000
MySQL	3306
Jenkins	8090

📄 03. 환경 변수

application.yml - 개발용 (로컬에서 관리)

```yaml
# 서버 기본 설정
server:
  port: 8080

spring:
  # 애플리케이션 기본 정보
  application:
    name: mr-patent

  # 프로필 설정
  profiles:
    active: ${SPRING_PROFILES_ACTIVE:local}

  # JPA 설정
  jpa:
    open-in-view: false
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
        dialect: org.hibernate.dialect.MySQL8Dialect
        show-sql: false

  # 파일 업로드 설정
  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB

  # 스프링 시큐리티 설정
  security:
    user:
      name: ${SPRING_SECURITY_USER_NAME:admin}
```

```
    password: ${SPRING_SECURITY_USER_PASSWORD:admin}

# 개발 도구 설정
devtools:
  restart:
    enabled: false

# 데이터베이스 설정
datasource:
  url: jdbc:mysql://localhost:3306/mrpatent?useSSL=false&serverTimezone=
  username: root
  password: ssafy
  driver-class-name: com.mysql.cj.jdbc.Driver

# 이메일 설정
mail:
  host: smtp.gmail.com
  port: 587
  username: ssafymo786@gmail.com
  password: hmmyrgcgqffwqkjl
  properties:
    mail:
      smtp:
        auth: true
        starttls:
          enable: true
          required: true
        connectiontimeout: 5000
        timeout: 5000
        writetimeout: 5000
  auth-code-expiration-millis: 180000

# AWS 설정
cloud:
  aws:
    credentials:
      access-key: AKIA4QLIP4WZERSFOPWJ
      secret-key: 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
```

```
      region: ap-northeast-2
      s3:
        bucket: mr-patent

  #fcm
  firebase:
    key-path: firebase/mrpatent-b0c8a-firebase-adminsdk-fbsvc-7421e3218e.j

# JWT 설정
jwt:
  secret: 0428sldahfmf12ckwdktjwherepro22areyouningmo0514for78jwtdujmy
  access-token-expiration: 3600000  # 1시간
  refresh-token-expiration: 2592000000  # 30일

# 로깅 설정
logging:
  level:
    org:
      springframework:
        web: INFO

# Swagger 설정
springdoc:
  swagger-ui:
    path: /api/swagger-ui/index.html
  api-docs:
    path: /api/v3/api-docs
  packages-to-scan: com.d208.mr_patent_backend
```

application.yml - 배포용 (서버에서 관리)

```
# 서버 기본 설정
server:
  port: 8080
```

```yaml
spring:
  # 애플리케이션 기본 정보
  application:
    name: mr-patent

  # 프로필 설정
  profiles:
    active: ${SPRING_PROFILES_ACTIVE:local}

  # JPA 설정
  jpa:
    open-in-view: false
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
        dialect: org.hibernate.dialect.MySQL8Dialect
        show-sql: false

  # 파일 업로드 설정
  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB

  # 스프링 시큐리티 설정
  security:
    user:
      name: ${SPRING_SECURITY_USER_NAME:admin}
      password: ${SPRING_SECURITY_USER_PASSWORD:admin}

  # 개발 도구 설정
  devtools:
    restart:
      enabled: false
```

```yaml
# 데이터베이스 설정
datasource:
  url: ${DB_URL}
  username: ${DB_USERNAME}
  password: ${DB_PASSWORD}
  driver-class-name: com.mysql.cj.jdbc.Driver

# 이메일 설정
mail:
  host: smtp.gmail.com
  port: 587
  username: ${MAIL_USERNAME}
  password: ${MAIL_PASSWORD}
  properties:
    mail:
      smtp:
        auth: true
        starttls:
          enable: true
          required: true
        connectiontimeout: 5000
        timeout: 5000
        writetimeout: 5000
  auth-code-expiration-millis: 180000

# AWS 설정
cloud:
  aws:
    credentials:
      access-key: ${AWS_ACCESS_KEY_ID}
      secret-key: ${AWS_SECRET_ACCESS_KEY}
    region: ${AWS_REGION}
    s3:
      bucket: ${S3_BUCKET_NAME}

#fcm
firebase:
  key-path: ${FIREBASE_KEY_PATH}
```

```yaml
# JWT 설정
jwt:
  secret: ${JWT_SECRET}
  access-token-expiration: 86400000  # 1일
  refresh-token-expiration: 2592000000  # 30일

# 로깅 설정
logging:
  level:
    org:
      springframework:
        web: INFO

# Swagger 설정
springdoc:
  swagger-ui:
    path: /api/swagger-ui/index.html
  api-docs:
    path: /api/v3/api-docs
  packages-to-scan: com.d208.mr_patent_backend
```

mr_patent-backend/.env - 서버내 관리

```
AWS_ACCESS_KEY_ID=AKIA4QLIP4WZERSFOPWJ
AWS_SECRET_ACCESS_KEY=GwI5Pv5YWtsDzExn4H1Z/xkZt0qnt1jN85WikMo
AWS_REGION=ap-northeast-2

S3_BUCKET_NAME=mr-patent
MAIL_USERNAME=ssafymo786@gmail.com
MAIL_PASSWORD=hmmyrgcgqffwqkjl

JWT_SECRET=wherearеyouningmoforjwtdujmyssjsmyjmrpatentwithsldahgkdl

DB_URL=jdbc:mysql://j12d208.p.ssafy.io:3306/mr_patent?useSSL=false&serv
DB_USERNAME=mr_patent
```

```
DB_PASSWORD=ssafy

SPRING_PROFILES_ACTIVE=production
FIREBASE_KEY_PATH=/app/config/firebase/firebase-service-account.json

SPRING_DATASOURCE_URL=jdbc:mysql://j12d208.p.ssafy.io:3306/mr_patent?
SPRING_DATASOURCE_USERNAME=mr_patent
SPRING_DATASOURCE_PASSWORD=ssafy
SPRING_DATASOURCE_DRIVER_CLASS_NAME=com.mysql.cj.jdbc.Driver
```

mr_patent-fastapi/.env - 서버내 관리

```
DATABASE_URL=mysql+pymysql://mr_patent:ssafy@j12d208.p.ssafy.io:3306/
GOOGLE_CREDENTIALS_PATH=./credentials/d208-mr-patent-ab1793e56fe1.js
KIPRIS_SERVICE_KEY=gCw4LP6MJu7w/C6332Er=0IjwYHwqna3rK56S0J9XS
```

- **로컬과 서버의 보안관리 방식**
  - **개발 환경 (로컬)**
    - application.yml 파일을 소스코드 내부에서 직접 관리
    - 개발자 개인의 환경에 맞춰 수정할 수 있도록 설정
  - **배포 환경 (서버)**
    - application.yml 파일에서는 민감 정보를 직접 작성하지 않고, 환경변수를 참조하도록 설정
    - 환경 변수는 서버 내에서 .env 파일로 관리

📄 04. CI/CD 구축

> 자동화된 배포 프로세스 구축 완료 → 추후 적용 예정

AWS EC2 접속

- pem키가 있는 디렉토리에서 터미널 실행

```
# ssh -i [pem키] [접속 계정]@[접속 도메인]
ssh -i j12D208T.pem
ubuntu@j12D208.p.ssafy.io
```

Docker-compose.yml

```
version: '3.8'

services:
  backend:
    build:
      context: ../S12P21D208/mr_patent_backend
      dockerfile: Dockerfile
    container_name: mr_patent_backend
    command: ["java", "-jar", "app.jar", "--spring.config.location=file:/app/confi
    volumes:
      - ./config:/app/config
    ports:
      - "8080:8080"  # 오직 백엔드만 외부에 노출
    depends_on:
      - mysql
      - fastapi
    env_file:
      - .env
    environment:
      - SPRING_DATASOURCE_URL=jdbc:mysql://mysql:3306/mr_patent?useSS
      - SPRING_DATASOURCE_USERNAME=mr_patent
      - SPRING_DATASOURCE_PASSWORD=ssafy
      - SPRING_DATASOURCE_DRIVER_CLASS_NAME=com.mysql.cj.jdbc.Driver
      - TZ=Asia/Seoul
    networks:
      - app-network
    logging:
      driver: "json-file"
      options:
        max-size: "10m"
```

```
      max-file: "3"

  fastapi:
    build:
      context: ../S12P21D208/mr_patent_bigdata
      dockerfile: Dockerfile
    container_name: mr_patent_fastapi
    # 개발
    # volumes:
    # - ./fastapi:/app
    ports:
      - "8000:8000"  # 외부에서 직접 접근 가능하도록 추가 (테스트용)
    environment:
      - ELASTICSEARCH_HOST=elasticsearch
      - ELASTICSEARCH_PORT=9200
      - TZ=Asia/Seoul
    env_file:
      - ../S12P21D208/mr_patent_bigdata/.env
    networks:
      - app-network
    logging:
      driver: "json-file"
      options:
        max-size: "10m"
        max-file: "3"

  mysql:
    image: mysql:8.0
    container_name: mr_patent_mysql
    ports:
      - "3306:3306"  # 외부에서 접근 가능하도록 포트 오픈
    environment:
      - MYSQL_ROOT_PASSWORD=ssafy
      - MYSQL_DATABASE=mr_patent
      - MYSQL_USER=mr_patent
      - MYSQL_PASSWORD=ssafy
      - TZ=Asia/Seoul
    volumes:
```

```
      - mysql_data:/var/lib/mysql
    restart: unless-stopped
    networks:
      - app-network
    logging:
      driver: "json-file"
      options:
        max-size: "10m"
        max-file: "3"

networks:
  app-network:
    driver: bridge

volumes:
  mysql_data:
```

Dockerfile

- **Backend**

```
FROM openjdk:17-jdk-slim

WORKDIR /app

COPY build/libs/*.jar app.jar

EXPOSE 8080

CMD ["java", "-Dfirebase.config.path=file:/app/config/firebase/firebase-s
```

- **FastAPI**

```
FROM python:3.9-slim-buster

# 필요한 시스템 의존성 설치 (Java 제외)
RUN apt-get update && apt-get install -y \
```

```
    wget \
    build-essential \
    default-libmysqlclient-dev \
    pkg-config \
    && rm -rf /var/lib/apt/lists/*

# Java 별도 설치
RUN apt-get update && \
    apt-get install -y --no-install-recommends ca-certificates && \
    apt-get install -y --no-install-recommends openjdk-11-jre-headless && \
    rm -rf /var/lib/apt/lists/*

# 나머지 Dockerfile 내용은 기존과 동일
# Spark 다운로드 및 설치
ENV SPARK_VERSION=3.3.2
ENV HADOOP_VERSION=3
ENV SPARK_HOME=/opt/spark
ENV PATH=$PATH:$SPARK_HOME/bin

RUN wget https://archive.apache.org/dist/spark/spark-${SPARK_VERSION
    && tar -xzf spark-${SPARK_VERSION}-bin-hadoop${HADOOP_VERSION
    && mv spark-${SPARK_VERSION}-bin-hadoop${HADOOP_VERSION} $S
    && rm spark-${SPARK_VERSION}-bin-hadoop${HADOOP_VERSION}.tgz

WORKDIR /app

# Python 의존성 설치
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt \
    && pip install pyspark findspark

# 프로젝트 파일 복사
COPY . .

# 환경 변수 설정
ENV PYTHONUNBUFFERED=1
ENV PYSPARK_PYTHON=python3
ENV PYSPARK_DRIVER_PYTHON=python3
```

```
ENV SPARK_HOME=/opt/spark

# 포트 노출
EXPOSE 8000

# 애플리케이션 실행
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Jenlinsfile

```
pipeline {
    agent any

    environment {
        DOCKER_COMPOSE_DIR = "/var/jenkins_shared/mr_patent"
        BACKEND_IMAGE = 'mr_patent-backend'
        BRANCH_NAME = "${env.BRANCH_NAME}"
        DOCKER_COMPOSE = '$HOME/bin/docker-compose'
    }

    stages {
        stage('Setup') {
            steps {
                echo '====== 환경 설정 시작 ======'
                // 도커 컴포즈 설치 확인 또는 설치
                sh '''
                    if ! command -v docker-compose &> /dev/null; then
                        echo "Docker Compose not found, installing..."
                        mkdir -p $HOME/bin
                        curl -L "https://github.com/docker/compose/releases/downloac
                        chmod +x $HOME/bin/docker-compose
                        export PATH=$HOME/bin:$PATH
                    else
                        echo "Docker Compose already installed"
                    fi
                    docker-compose --version || $HOME/bin/docker-compose --vers
```

```
        '''
        echo '====== 환경 설정 완료 ======'
    }
}

stage('Checkout') {
    steps {
        checkout scm
        // 빌드 시작 시 커밋 정보 저장
        script {
            env.GIT_AUTHOR = sh(script: "git show -s --pretty=%an", returnS
            env.GIT_EMAIL = sh(script: "git show -s --pretty=%ae", returnStdo
        }
    }
}

stage('Build') {
    steps {
        echo '====== 백엔드 빌드 시작 ======'
        dir('mr_patent_backend') {
            sh 'chmod +x ./gradlew || true'
            sh './gradlew clean build -x test'
        }
        echo '====== 백엔드 빌드 완료 ======'
    }
}

stage('Test') {
    steps {
        echo '====== 백엔드 테스트 시작 ======'
        dir('mr_patent_backend') {
            sh './gradlew test || true'
        }
        echo '====== 백엔드 테스트 완료 ======'
    }
}

stage('Deploy') {
```

```
        steps {
            echo '====== 백엔드 배포 시작 ======'

            // .env 복사
            sh 'cp ${DOCKER_COMPOSE_DIR}/.env ${DOCKER_COMPOSE_DIR}

            // 빌드 결과 복사
            sh 'mkdir -p ${DOCKER_COMPOSE_DIR}/build/libs/'
            sh 'cp -f mr_patent_backend/build/libs/*.jar ${DOCKER_COMPOSE_D

            // Firebase 키 복사
            withCredentials([file(credentialsId: 'firebase_key', variable: 'FIREBAS
                sh 'mkdir -p ${DOCKER_COMPOSE_DIR}/config/firebase'
                sh 'cp -f ${FIREBASE_KEY_FILE} ${DOCKER_COMPOSE_DIR}/con
                sh 'chmod 600 ${DOCKER_COMPOSE_DIR}/config/firebase/fireba
            }

            // 디버깅 정보 출력
            sh 'echo "현재 작업 디렉토리 확인:" && pwd'
            sh 'echo ".env 파일 있는지 확인:" && ls -al ${DOCKER_COMPOSE_DIR}
            sh 'echo "docker-compose.yml 위치 확인:" && ls -al ${DOCKER_COM

            // 도커 재배포
            sh '''
                cd ${DOCKER_COMPOSE_DIR}
                $HOME/bin/docker-compose -f docker-compose.yml stop backer
                $HOME/bin/docker-compose -f docker-compose.yml rm -f backe
                $HOME/bin/docker-compose -f docker-compose.yml build --no-c
                $HOME/bin/docker-compose -f docker-compose.yml up -d --no-
                docker image prune -f || true
            '''
            echo '====== 백엔드 배포 완료 ======'
        }
    }

    stage('Notification') {
        steps {
            echo 'jenkins notification!'
```

```
            }
        }
    }

    post {
        success {
            echo '====== 파이프라인 성공 ======'
            mattermostSend(
                color: 'good',
                message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by
                endpoint: 'https://meeting.ssafy.com/hooks/hgafhbr6n7fe7japbi7n5t
                channel: 'D208-GitLab-Build'
            )
        }
        failure {
            echo '====== 파이프라인 실패 ======'
            mattermostSend(
                color: 'danger',
                message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by
                endpoint: 'https://meeting.ssafy.com/hooks/hgafhbr6n7fe7japbi7n5t
                channel: 'D208-GitLab-Build'
            )
        }
        always {
            echo '====== 파이프라인 종료 ======'
            cleanWs()
        }
    }
}
```

Jenlinsfile-python

```
pipeline {
    agent any

    environment {
        DOCKER_COMPOSE_DIR = "/var/jenkins_shared/mr_patent"
        DOCKER_COMPOSE = "$HOME/bin/docker-compose"
```

```
        BRANCH_NAME = "${env.BRANCH_NAME}"
        PYTHON_VERSION = "3.9"
        PATH = "/usr/local/bin:/usr/bin:/bin:$HOME/.local/bin"
    }

    stages {
        stage('Setup Python Environment') {
            steps {
                sh '''
                    apt-get update
                    apt-get install -y python3 python3-pip python3-venv
                    python3 --version
                    python3 -m pip --version
                '''
            }
        }

        stage('Checkout') {
            steps {
                checkout scm
                script {
                    env.GIT_AUTHOR = sh(script: "git show -s --pretty=%an", returnS
                    env.GIT_EMAIL = sh(script: "git show -s --pretty=%ae", returnStdo
                }
            }
        }

        stage('Build FastAPI') {
            steps {
                echo '====== FastAPI 빌드 시작 ======'
                dir('mr_patent_bigdata') {
                    sh '''
                        python3 -m venv venv
                        . venv/bin/activate
                        pip install --upgrade pip
                        pip install pipenv
                        pipenv install --dev || pipenv install
                        ls -la
```

```
                ls -la app
                cat Pipfile
            '''
            echo '====== FastAPI 빌드 완료 ======'
        }
    }
}

stage('Deploy FastAPI') {
    steps {
        echo '====== FastAPI 배포 시작 ======'
        // 환경 설정 파일 복사
        withCredentials([
            file(credentialsId: 'fastapi-env', variable: 'ENV_FILE'),
            file(credentialsId: 'application-prod.yml', variable: 'APP_YML_FILE
        ]) {
            sh '''
                mkdir -p ${DOCKER_COMPOSE_DIR}/fastapi/config
                cp -f ${ENV_FILE} ${DOCKER_COMPOSE_DIR}/fastapi/config/.e
                cp -f ${APP_YML_FILE} ${DOCKER_COMPOSE_DIR}/fastapi/con
                chmod 600 ${DOCKER_COMPOSE_DIR}/fastapi/config/.env
                chmod 600 ${DOCKER_COMPOSE_DIR}/fastapi/config/applicat
            '''
        }

        sh '''
            mkdir -p ${DOCKER_COMPOSE_DIR}/fastapi
            cp -rf mr_patent_bigdata/app ${DOCKER_COMPOSE_DIR}/fastapi/
            cp -rf mr_patent_bigdata/models ${DOCKER_COMPOSE_DIR}/fast
            cp -f mr_patent_bigdata/Pipfile ${DOCKER_COMPOSE_DIR}/fastap
            cp -f mr_patent_bigdata/Pipfile.lock ${DOCKER_COMPOSE_DIR}/f
            cd mr_patent_bigdata
            pipenv requirements --dev > ${DOCKER_COMPOSE_DIR}/fastapi/


            cd ${DOCKER_COMPOSE_DIR}
            $HOME/bin/docker-compose -f docker-compose.yml stop fastapi
            $HOME/bin/docker-compose -f docker-compose.yml rm -f fastap
```

```
            $HOME/bin/docker-compose -f docker-compose.yml build --no-c
            $HOME/bin/docker-compose -f docker-compose.yml up -d fastap
        '''
        echo '====== FastAPI 배포 완료 ======'
    }
}


stage('Notification') {
    steps {
        echo 'jenkins notification!'
    }
}
}

post {
    success {
        echo '====== 파이프라인 성공 ======'
        mattermostSend(
            color: 'good',
            message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by
            endpoint: 'https://meeting.ssafy.com/hooks/hgafhbr6n7fe7japbi7n5t
            channel: 'D208-GitLab-Build'
        )
    }
    failure {
        echo '====== 파이프라인 실패 ======'
        mattermostSend(
            color: 'danger',
            message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by
            endpoint: 'https://meeting.ssafy.com/hooks/hgafhbr6n7fe7japbi7n5t
            channel: 'D208-GitLab-Build'
        )
    }
    always {
        echo '====== 파이프라인 종료 ======'
    }
```

```
    }
}
```

Nginx.config

```
server {
    listen 80;
    listen [::]:80;
    server_name j12d208.p.ssafy.io;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name j12d208.p.ssafy.io;
    ssl_certificate /etc/letsencrypt/live/j12d208.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j12d208.p.ssafy.io/privkey.pem;

    # 보안 헤더 설정
    add_header Strict-Transport-Security "max-age=31536000; includeSubDor
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header X-Frame-Options "SAMEORIGIN" always;

    # sse 연결 설정
    location ~* /api/chat/rooms/subscribe/ {
        proxy_pass http://localhost:8080;

        proxy_http_version 1.1;
        proxy_set_header Connection 'keep-alive';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_buffering off;
```

```
        proxy_cache off;
        proxy_read_timeout 600s;
        keepalive_timeout 600s;

        add_header Cache-Control no-cache;
        add_header Content-Type text/event-stream always;
    }

    # 백엔드 API 라우팅
    location /api {
        proxy_pass http://localhost:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # FastAPI 문서 접근
    location /docs {
        proxy_pass http://localhost:8000/docs;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /openapi.json {
        proxy_pass http://localhost:8000/openapi.json;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # 웹소켓 라우팅
    location /ws/ {
        proxy_pass http://localhost:8080;
```

```
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # Jenkins 프록시 설정 (루트 경로로)
    location /jenkins/ { proxy_pass http://localhost:8090/jenkins/;

        proxy_http_version 1.1;
        proxy_request_buffering off;
        proxy_buffering off;
        proxy_max_temp_file_size 0;
        proxy_connect_timeout 150;
        proxy_send_timeout 100;
        proxy_read_timeout 100;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Jenkins-Value "";
    }

    # 상태 체크용
    location = /status {
        return 200 '{"status":"running","service":"mr_patent_api"}';
        add_header Content-Type application/json;
    }
```

📄 05. 외부 서비스 사용

외부 API

- KIPRIS API - 특허 데이터 원문 추출
- Firebase Auth
- Google OAuth
- Mattermost 알림 설정