

# 마주교실 빌드 및 배포 매뉴얼

## 목차

- 1. [GitLab 소스 클론 이후 빌드 및 배포](#)
- 2. [외부 서비스 정보](#)
- 3. [프론트엔드 & 백엔드 환경변수](#)

## 1. GitLab 소스 클론 이후 빌드 및 배포

### 1.1 사용 제품 및 버전

#### 백엔드 (Backend)

항목	제품/버전
언어	Java 21
빌드 도구	Gradle 8.5
프레임워크	Spring Boot 3.5.6
WAS	내장 Tomcat (Spring Boot Embedded)
JVM	Eclipse Temurin 21-JRE
데이터베이스	MySQL 8.4
캐시	Redis 7-alpine

#### 주요 라이브러리:

- Spring Security (JWT 인증)
- Spring Data JPA (Hibernate)
- MySQL Connector J
- Spring Data Redis
- JWT: jjwt 0.12.6

- Swagger: springdoc-openapi 2.8.13
- WebClient (Spring WebFlux)

## 프론트엔드 (Frontend)

항목	제품/버전
언어	TypeScript 5.9.3
프레임워크	React 19.1.1
빌드 도구	Vite 7.1.7
패키지 매니저	npm (Node.js 20)
웹서버	Nginx stable-alpine

### 주요 라이브러리:

- React Router DOM 7.9.5
- Zustand 5.0.8 (상태관리)
- TanStack Query 5.90.7
- Tailwind CSS 3.4.18
- Axios 1.12.2
- Chart.js 4.5.1
- Lottie React 2.4.1

## AI 서비스 (AI)

항목	제품/버전
언어	Python 3.11
프레임워크	FastAPI 0.115.5
웹서버	Uvicorn 0.32.1 (ASGI)
데이터베이스	SQLAlchemy 2.0.36 (asyncio)

### 주요 라이브러리:

- PyTorch 2.5.1 (CPU only)

- LangChain 0.3.13
- OpenAI 1.58.1
- Sentence Transformers 3.3.1
- ChromaDB 0.5.23 (벡터 DB)
- AsyncMy 0.2.9 (MySQL 비동기 드라이버)

## 인프라

항목	제품/버전
컨테이너	Docker, Docker Compose
데이터베이스	MySQL 8.4
캐시	Redis 7-alpine
스토리지	AWS S3 (ap-northeast-2)

## 1.2 빌드 시 사용되는 환경 변수

프로젝트 루트에 `.env` 파일을 생성하여 다음 환경 변수를 설정해야 합니다.

### 필수 환경 변수

```
# =====
# 데이터베이스 설정
# =====
DB_HOST=db # Docker Compose 사용 시 서비스명, 로컬은
            local host
DB_PORT=3306
DB_NAME=your_database_name # 데이터베이스 이름
DB_USERNAME=your_db_user # MySQL 사용자명
DB_PASSWORD=your_db_password # MySQL 비밀번호
DB_ROOT_PASSWORD=your_root_password # MySQL root 비밀번호

# =====
# Redis 설정
# =====
REDIS_HOST=redis # Docker Compose 사용 시 서비스명, 로컬은
                 local host
REDIS_PORT=6379
```

```

REDIS_PASSWORD=your_redis_password # Redis 비밀번호

# =====
# JWT 설정
# =====
JWT_SECRET=your_jwt_secret_key # JWT 서명 시크릿 키 (최소 256bit)
JWT_ACCESS_TOKEN_EXPIRATION=3600000 # Access Token 만료 시간 (1시간, 밀리초)
JWT_REFRESH_TOKEN_EXPIRATION=604800000 # Refresh Token 만료 시간 (7일, 밀리초)

# =====
# AWS S3 설정
# =====
S3_PUBLIC_URL_BASE=https://ssafy-sai-project.s3.ap-northeast-2.amazonaws.com
LAMBDA_PRESIGNED_URL_API=https://fdmkozi a90.execute-api.ap-northeast-2.amazonaws.com/presigned-url

# =====
# 서비스 포트 설정
# =====
FASTAPI_PORT=8000 # AI 서비스 포트
SPRING_PORT=8080 # Spring Boot 포트

# =====
# AI - OpenAI API (GMS 프록시)
# =====
GMS_BASE=https://gms.ssafy.io/gmsapi/api.openai.com/v1
GMS_KEY=your_openai_api_key # OpenAI API Key

# =====
# AI - Naver Clova STT (선택사항)
# =====
CLOVA_CLIENT_ID=your_clova_client_id
CLOVA_CLIENT_SECRET=your_clova_client_secret

```

## 환경 변수 설명

변수명	설명	사용 위치
DB_*	MySQL 데이터베이스 연결 정보	Backend, AI
REDIS_*	Redis 캐시 연결 정보	Backend
JWT_*	JWT 토큰 설정	Backend
S3_PUBLIC_URL_BASE	S3 버킷 공개 URL	Backend

LAMBDA_PRESIGNED_URL_API	Presigned URL 발급 Lambda API	Backend, AI
GMS_BASE	OpenAI API 프록시 Base URL	AI
GMS_KEY	OpenAI API Key	AI
CLOVA_*	Naver Clova STT API 인증 정보 (선택)	AI

## 1.3 배포 시 특이사항

Docker Compose를 이용한 배포

### 전체 스택 빌드 및 실행

```
# 프로젝트 루트 디렉토리에서 실행
docker compose up --build -d
```

### 개별 서비스 실행

```
# MySQL만 실행
docker compose --profile db up -d

# Spring Boot Backend만 실행
docker compose --profile spring up -d

# FastAPI AI 서비스만 실행
docker compose --profile fastapi up -d

# Redis만 실행
docker compose --profile redis up -d
```

### 로그 확인

```
# 전체 로그
docker compose logs -f

# 특정 서비스 로그
docker compose logs -f spring
docker compose logs -f fastapi
docker compose logs -f db
```

## 서비스 중지 및 제거

```
# 전체 중지
docker compose down

# 볼륨까지 삭제
docker compose down -v
```

---

## 백엔드 (Spring Boot) 빌드 프로세스

Dockerfile 위치: `backend/Dockerfile`

### 멀티 스테이지 빌드 구조

1. Build Stage (Gradle 8.5-jdk21)
  - Gradle 의존성 캐싱 최적화
  - `build.gradle`, `settings.gradle` 먼저 복사하여 의존성 다운로드
  - 소스 코드 복사 후 빌드 실행
  - 테스트 제외: `gradle build -x test --no-daemon`
2. Runtime Stage (Eclipse Temurin 21-JRE)
  - 빌드된 JAR 파일만 복사 (경량화)
  - 포트 8080 노출
  - 실행 명령: `java -jar app.jar`

### 로컬 빌드 (Docker 없이)

```
cd backend
./gradlew clean build -x test
java -jar build/libs/*.jar
```

### 주요 설정 파일

- 빌드 설정: `backend/build.gradle`
- 애플리케이션 설정: `backend/src/main/resources/application.yml`
  - 서버 포트: 8080

- Context Path: `/api`
  - JPA DDL-AUTO: `update` (자동 스키마 업데이트)
- 

## 프론트엔드 (React) 빌드 프로세스

Dockerfile 위치: `frontend/Dockerfile`

## 멀티 스테이지 빌드 구조

1. Build Stage (Node 20)
  - `package.json`, `package-lock.json` 먼저 복사
  - 의존성 설치: `npm ci` (캐시 활용)
  - TypeScript 컴파일 + Vite 빌드: `npm run build`
  - 빌드 산출물 위치: `/app/dist`
2. Runtime Stage (Nginx stable-alpine)
  - Nginx 기본 설정 제거
  - SPA 라우팅 설정 적용: `nginx/react.conf`
  - 빌드된 정적 파일 복사: `/usr/share/nginx/html`
  - 포트 80 노출
  - Health Check 포함

## Nginx SPA 라우팅 설정

파일 위치: `frontend/nginx/react.conf`

```
location / {  
    try_files $uri $uri /index.html;  
}
```

## 로컬 개발 서버 실행

```
cd frontend  
npm install  
npm run dev # http://localhost:5173
```

## 프로덕션 빌드

```
cd frontend
npm run build
# 빌드 결과: dist/ 디렉토리
```

## AI 서비스 (FastAPI) 빌드 프로세스

Dockerfile 위치: `ai/Dockerfile`

### 4단계 멀티 스테이지 빌드 구조

3. Base Stage (Python 3.11-slim)
  - 시스템 의존성 설치 (build-essential, curl 등)
  - 환경 변수 설정 (PYTHONDONTWRITEBYTECODE, PYTHONUNBUFFERED)
4. Builder Stage
  - Python 패키지 wheel 빌드
  - pip 캐시 마운트로 빌드 속도 향상
5. Model Downloader Stage
  - 한국어 Sentence Transformer 모델 사전 다운로드
  - 모델: `snunlp/KR-SBERT-V40K-k1ueNLI-augSTS`
  - 컨테이너 이미지에 모델 캐시 포함 → 초기 시작 시간 단축
6. Production Stage
  - 비루트 사용자(`appuser`) 생성 및 실행 (보안)
  - ChromaDB 데이터 디렉토리 생성: `/app/data/chroma_db`
  - Health Check 엔드포인트: `/health`
  - 실행 명령: `uvicorn app.main:app --host 0.0.0.0 --port 8000 --workers 1`

### 로컬 실행 (Docker 없이)

```
cd ai
pip install -r requirements.txt
uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload
```

주요 특이사항

- 모델 사전 다운로드: 첫 실행 시 모델 다운로드 대기 시간 없음
- ChromaDB 볼륨: 벡터 DB 데이터 영구 저장을 위해 볼륨 마운트 필수
- Worker 수: `--workers 1` (ML 모델 메모리 중복 로딩 방지)

데이터베이스 초기화

DB 덤프 파일 위치: `exec/DB_덤프파일/`

포함된 SQL 파일

파일명	설명
<code>users.sql</code>	사용자 (교사) 정보
<code>school s.sql</code>	학교 정보
<code>students.sql</code>	학생 정보
<code>scenari o_categori es.sql</code>	시나리오 카테고리
<code>scenari os.sql</code>	시나리오
<code>scenari o_sequences.sql</code>	시나리오 질문 시퀀스
<code>seq_opti ons.sql</code>	시퀀스 옵션 (선택지)
<code>scenari o_sessi ons.sql</code>	시뮬레이션 세션
<code>sessi on_answers.sql</code>	세션 답변
<code>sessi on_stt_answers.sql</code>	STT 음성 답변

덤프 파일 적용 방법

```
# MySQL 컨테이너에 접속
docker exec -it a202-mysql bash

# 데이터베이스 선택 후 덤프 파일 실행
mysql -u root -p your_database_name < /path/to/dump.sql
```

```
# 또는 호스트에서 직접 실행
docker exec -i a202-mysql mysql -u root -p your_database_name <
exec/DB_덤프파일/users.sql
```

---

## 1.4 프로젝트에 활용되는 주요 계정 및 프로퍼티 파일

### Backend 프로퍼티 파일

파일 위치: `backend/src/main/resources/application.yml`

### 주요 설정 내용

```
server:
  port: 8080
  servlet:
    context-path: /api

spring:
  datasource:
    url: jdbc:mysql://${DB_HOST}:${DB_PORT}/${DB_NAME}
    username: ${DB_USERNAME}
    password: ${DB_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: update          # 스키마 자동 업데이트
    show-sql: false
    open-in-view: false
    properties:
      hibernate:
        jdbc:
          time_zone: Asia/Seoul

  data:
    redis:
      host: ${REDIS_HOST}
      port: ${REDIS_PORT}
      password: ${REDIS_PASSWORD}

  jwt:
    secret: ${JWT_SECRET}
    access-token-expiration: ${JWT_ACCESS_TOKEN_EXPIRATION}
    refresh-token-expiration: ${JWT_REFRESH_TOKEN_EXPIRATION}

  aws:
    s3:
      public-url-base: ${S3_PUBLIC_URL_BASE}
```

```
lambda:
  presigned-url -api : ${LAMBDA_PRESIGNED_URL_API}
```

## 설정 참조

- API Swagger 문서: <http://localhost:8080/api/swagger-ui.html>
- Context Path: 모든 API 요청은 `/api` 접두사 필요
- 타임존: `Asia/Seoul` (한국 시간대)

---

## Frontend 설정 파일

파일 위치: `frontend/vite.config.ts`

```
export default defineConfig({
  plugins: [react()],
  server: {
    // allowedHosts: ["..."], // 테스트용 서버 주소
  },
})
```

Nginx 설정 파일: `frontend/nginx/react.conf`

```
server {
  listen 80;
  server_name _;

  root /usr/share/nginx/html;
  index index.html;

  # SPA 라우팅: 파일이 없으면 index.html 로
  location / {
    try_files $uri $uri /index.html;
  }

  # 업로드 용량 제한
  client_max_body_size 10m;
  sendfile on;
}
```

---

## AI 서비스 환경 설정

AI 서비스는 `.env` 파일에서 다음 환경변수를 로드합니다:

환경변수	용도
------	----

<code>GMS_BASE</code>	OpenAI API 프록시 Base URL
<code>GMS_KEY</code>	OpenAI API Key (GPT, Whisper)
<code>CLOVA_CLIENT_ID</code>	Naver Clova STT Client ID (선택)
<code>CLOVA_CLIENT_SECRET</code>	Naver Clova STT Secret (선택)
<code>LAMBDA_PRE_SIGNED_URL_API</code>	S3 Presigned URL 발급 Lambda API
<code>DB_HOST</code> , <code>DB_PORT</code> , <code>DB_USER</code> , <code>DB_PASSWORD</code> , <code>DB_NAME</code>	MySQL 연결 정보

## 데이터베이스 ERD 및 스키마

### 주요 테이블

- users: 교사 계정 정보
- schools: 학교 정보
- students: 학생 정보 (교사와 연결)
- scenario\_categories: 시나리오 카테고리 (예: 카페 주문, 영화 예매)
- scenarios: 시나리오 정보
- scenario\_sequences: 시나리오 내 질문 시퀀스
- seq\_options: 각 질문의 선택지 옵션
- scenario\_sessions: 학생의 시뮬레이션 세션
- session\_answers: 세션 내 답변 (EASY, NORMAL 난이도)
- session\_stt\_answers: STT 음성 답변 (HARD 난이도)

## 1.5 배포 전체 프로세스 요약

### 1단계: 환경 설정

```
# 1. 저장소 클론
git clone <repository-url>
```

```
cd <project-directory>

# 2. .env 파일 생성 및 환경 변수 설정
cp .env.example .env # 또는 직접 생성
vi .env               # 환경 변수 입력
```

## 2단계: Docker Compose로 빌드 및 실행

```
# 전체 스택 빌드 및 실행
docker compose up --build -d

# 로그 확인
docker compose logs -f
```

## 3단계: 데이터베이스 초기화 (선택)

```
# DB 덤프 파일 적용
docker exec -i a202-mysql mysql -u root -p${DB_ROOT_PASSWORD} ${DB_NAME} <
exec/DB_덤프파일/users.sql
# 나머지 SQL 파일도 동일하게 적용
```

## 4단계: 서비스 확인

- Frontend: <http://localhost> (포트 80)
- Backend API: <http://localhost:8080/api>
- Backend Swagger: <http://localhost:8080/api/swagger-ui.html>
- AI Service: <http://localhost:8000>
- AI Service Docs: <http://localhost:8000/docs>

---

## 2. 외부 서비스 정보

### 2.1 OpenAI API (GPT, Whisper)

#### 서비스 개요

- **제공업체:** OpenAI
- **사용 모델:**
  - GPT-4: AI 시나리오 자동 생성, 음성 유사도 평가 피드백

- Whisper: 음성을 텍스트로 변환 (STT)

## 접근 방법

프로젝트에서는 SSAFY GMS 프록시를 통해 OpenAI API를 사용합니다.

### 환경 변수 설정:

```
GMS_BASE=https://gms.ssafy.io/gmsapi/api.openai.com/v1  
GMS_KEY=your_openai_api_key
```

## 가입 및 API Key 발급

1. [OpenAI Platform](#) 가입
2. API Keys 메뉴에서 새 API Key 생성
3. 생성된 Key를 **GMS\_KEY** 환경 변수에 설정

## 사용되는 API 엔드포인트

- Chat Completions (GPT-4):
  - URL: **{GMS\_BASE}/chat/completions**
  - 용도: 시나리오 자동 생성, 피드백 생성
- Audio Transcriptions (Whisper):
  - URL: **{GMS\_BASE}/audio/transcriptions**
  - 용도: 음성을 텍스트로 변환

## 관련 코드 파일

- **ai/app/domains/scenario/services/llm\_service.py** (GPT 시나리오 생성)
  - **ai/app/domains/speech\_to\_text/services/stt\_service.py** (Whisper STT)
  - **ai/app/domains/speech\_to\_text/services/speech\_analysis\_service.py** (음성 유사도 평가)
-

## 2.2 Naver Clova STT (선택사항)

### 서비스 개요

- **제공업체:** Naver Cloud Platform
- **용도:** 음성을 텍스트로 변환 (Whisper 대안)

### 가입 및 설정

1. [Naver Cloud Platform](#) 가입
2. Console > AI Services > Clova Speech Recognition (CSR/Premium) 신청
3. 애플리케이션 등록
4. Client ID 및 Client Secret 발급

### 환경 변수 설정:

```
CLOVA_CLIENT_ID=your_client_id  
CLOVA_CLIENT_SECRET=your_client_secret
```

### API 정보

- API 엔드포인트: <https://naveropenapi.apigw.ntruss.com/recog/v1/stt>
- 지원 언어: 한국어(Kor), 영어(Eng), 일본어(Jpn), 중국어(Chn)
- 인증 방식: 헤더에 Client ID/Secret 포함

### 관련 코드 파일

- [ai/app/domains/speech\\_to\\_text/services/stt\\_service.py](#) (ClovaSTTService 클래스)

---

## 2.3 AWS S3 및 Lambda

### 서비스 개요

- S3: 시나리오 이미지 저장 (썸네일, 배경, 픽토그램)
- Lambda: Presigned URL 발급을 통한 보안 업로드

## AWS S3 설정 정보

항목	값
버킷 이름	ssafy-sai-project
리전	ap-northeast-2 (서울)
Public URL	https://ssafy-sai-project.s3.ap-northeast-2.amazonaws.com

### 환경 변수 설정:

```
S3_PUBLIC_URL_BASE=https://ssafy-sai-project.s3.ap-northeast-2.amazonaws.com
```

## AWS Lambda (Presigned URL API)

항목	값
API Gateway 엔드포인트	https://fdmkozi-a90.execute-api.ap-northeast-2.amazonaws.com/presigned-url
메서드	POST
리전	ap-northeast-2

### 환경 변수 설정:

```
LAMBDA_PRE_SIGNED_URL_API=https://fdmkozi-a90.execute-api.ap-northeast-2.amazonaws.com/presigned-url
```

## Presigned URL API 사용법

### 요청 형식:

```
{
  "fileName": "scenarios/thumbnails/uiid.png",
  "operation": "putObject",
  "contentType": "image/png"
}
```

### 응답 형식:

```
{
  "url": "https://s3.amazonaws.com/...",
  "fileName": "scenarios/thumbnails/uiid.png"
}
```

## S3 디렉토리 구조

- scenarios/thumbnails/ - 시나리오 썸네일 이미지

- `scenarios/backgrounds/` - 시나리오 배경 이미지
- `scenarios/options/` - 시나리오 선택지 픽토그램 이미지

## AWS 계정 설정

1. [AWS Console](#) 로그인
2. S3 버킷 생성 및 CORS 설정
3. Lambda 함수 생성 (Presigned URL 발급)
4. API Gateway 설정 및 Lambda 연결

## 관련 코드 파일

- `backend/src/main/resources/application.yml` (S3, Lambda 설정)
- `ai/app/domains/scenarios/services/s3_service.py` (S3 업로드)
- `ai/app/domains/speech_to_text/utils/presigned_url_service.py` (Presigned URL 발급)

---

## 2.4 ChromaDB (로컬 벡터 데이터베이스)

### 서비스 개요

- **제공업체:** Chroma (오픈소스)
- **용도:** RAG (Retrieval-Augmented Generation) 기반 시나리오 생성
- **저장 방식:** 로컬 파일 시스템 (Docker 볼륨)

### 설정 정보

항목	값
저장 위치	<code>/app/data/chroma_db</code> (컨테이너 내부)
Docker 볼륨	<code>chroma_data</code>
임베딩 모델	<code>snunlp/KR-SBERT-V40K-k1ueNLI-augSTS</code>

## 임베딩 모델 상세

- **모델명:** snunlp/KR-SBERT-V40K-klueNLI-augSTS
- **언어:** 한국어 특화
- **용도:** 시나리오 텍스트 임베딩 및 유사도 검색
- **다운로드:** Dockerfile 빌드 시 자동 다운로드

## 데이터 영구 저장

Docker Compose 볼륨 마운트:

```
volumes:  
- chroma_data: /app/data/chroma_db
```

## 관련 코드 파일

- `ai/app/domains/scenarios/services/embedding_service.py` (임베딩 서비스)
  - `ai/scripts/ingest_scenarios.py` (시나리오 데이터 수집)
- 

## 2.5 사용하지 않는 서비스

프로젝트에서는 다음 서비스를 사용하지 않습니다:

### 소셜 인증 (OAuth)

- Google, Kakao, Naver 등의 소셜 로그인 미사용
- JWT 기반 자체 인증만 사용

### Photon Cloud

- 실시간 멀티플레이어 기능 없음
- 단일 학생 시뮬레이션만 지원

### 코드 컴파일 서비스

- Judge0, Wandbox 등의 코드 실행 서비스 미사용
  - 시나리오 기반 시뮬레이션만 제공
-

### 3. 외부 서비스 요약

#### 필수 외부 서비스

서비스	제공업체	용도	환경 변수
OpenAI API (GPT-4)	OpenAI	시나리오 자동 생성	GMS_BASE, GMS_KEY
OpenAI API (Whisper)	OpenAI	음성을 텍스트로 변환	GMS_BASE, GMS_KEY
AWS S3	AWS	이미지 파일 저장	S3_PUBLIC_URL_BASE
AWS Lambda	AWS	Presigned URL 발급	LAMBDA_PRESIGNED_URL_API

#### 선택사항 외부 서비스

서비스	제공업체	용도	환경 변수
Naver Clova STT	Naver Cloud	음성을 텍스트로 변환 (Whisper 대안)	CLOVA_CLIENT_ID, CLOVA_CLIENT_SECRET

#### 내부 서비스 (자체 호스팅)

서비스	버전	용도
MySQL	8.4	주 데이터베이스
Redis	7-alpine	캐시 및 JWT 블랙리스트
ChromaDB	0.5.23	벡터 데이터베이스 (RAG)

### 3 프론트엔드 백엔드 환경변수

#### 프론트엔드

```
VITE_BASE_API_URL=https://www.majuclass.com/api /  
VITE_BASE_AI_URL=https://www.majuclass.com/ai /
```

#### 백엔드 & AI

```
# SpringBoot  
DB_HOST=local host  
DB_PORT=3306
```

```
DB_NAME=a202
DB_USERNAME=maj_u_user
DB_PASSWORD=db_secret_password_for_maj_u_ssafy_2025
DB_ROOT_PASSWORD=db_secret_password_for_maj_u_very_strong

# JWT Configuration
JWT_SECRET=Maj_uJWTSecretKeyForSSAFY2025byA202Whi chI sVeryLongAndStrong
JWT_ACCESS_TOKEN_EXPIRATION=86400000
JWT_REFRESH_TOKEN_EXPIRATION=604800000

# AWS S3 & Lambda Configuration
S3_PUBLIC_URL_BASE=https://ssafy-sai-project.s3.ap-northeast-2.amazonaws.com
LAMBDA_PRESIGNED_URL_API=https://fdmkozi a90.execute-api.ap-northeast-2.amazonaws.com/presigned-url

# Redis Configuration
REDIS_HOST=localhost
REDIS_PORT=6379
REDIS_PASSWORD=redis_password_for_maj_u_2025_ssafy

##FastAPI
GMS_KEY=S13P32A202-b7da93b0-d74e-46c5-9f72-504a493859f7
GMS_IMAGE_BASE=https://gms.ssafy.io/gmsapi/api.openai.com/v1/images/generations
GMS_BASE=https://gms.ssafy.io/gmsapi/api.openai.com/v1

# CLOVA
CLOVA_CLIENT_ID="6gl3dav1dy"
CLOVA_CLIENT_SECRET="sJXb0AaCbuQBMPjYIn1bRLbEi8pUF4dGRtVEIEcX"

FASTAPI_PORT=8001
SPRING_PORT=8080
COMPOSE_PROFILES=db, fastapi, spring, redis
```