|  |
| --- |
|  |

|  |
| --- |
| **PianoBear - 피아노 교육 웹 앱** |
| **포팅 매뉴얼** |
|  |
|  |
|  |
|  |
|  |

목차

[I. 개요 2](#_Toc96072459)

[1. 프로젝트 개요 2](#_Toc96072460)

[2. 프로젝트 사용 도구 2](#_Toc96072461)

[3. 개발환경 2](#_Toc96072462)

[4. 외부 서비스 2](#_Toc96072463)

[5. Gitgnore 처리한 핵심 키들 3](#_Toc96072464)

[II. 빌드 3](#_Toc96072465)

[1. 환경변수 형태 3](#_Toc96072466)

[2. 빌드하기 5](#_Toc96072467)

[3. 배포하기 5](#_Toc96072468)

[4. 서비스 이용 방법 7](#_Toc96072469)

[가) OpenAI GPT API 7](#_Toc96072470)

# 개요

## 프로젝트 개요

모두가 피아노 학원 한번씩 다녀보았을텐데 다들 피아노 배우는 것이 숙제 같다고 느껴졌을 거라고 생각합니다. 실제로 현업에 계시는 많은 음악 교육 전문가들은 현재 피아노 교육에 많은 문제점들이 있다고 말을 합니다. 배우는 어린이들 또한 "재미없다" "악보 보기 어렵다" "학원 가기 싫다" 같은 말들을 많이 합니다. 그래서 저희는 어린이들이 재밌고 쉽게 피아노를 배울 수 있도록 하는 피아노배어 서비스를 개발하였습니다. 피아노배어와 함께라면 계이름이 달린 악보와 자신이 원하는 음악을 악보로 만들어주는 기능, 자기주도적으로 연습하고 스스로를 채점할 수 있는 기능, 친구들과 함께 소통하며 연습할 수 있는 기능을 통해 어린이들이 더욱 더 피아노에 대한 흥미를 유지하며 재미있게 피아노를 배울 수 있습니다.

## 프로젝트 사용 도구

이슈 관리 : JIRA

형상 관리 : Gitlab

커뮤니케이션 : Notion, Mattermost, Discord

디자인 : Figma

UCC : 모바비

CI/CD : Jenkins, Docker

## 개발환경

VS Code : 1.91.0,

IntelliJ : 2024.1.4 amd64

JVM : eclipse-temurin:17-jre (빌드 환경: openjdk-17-jdk)

SERVER : AWS EC2 Ubuntu 20.04.3 LTS

DB : PostgreSQL 16

Docker version 27.1.1, build 6312585

## 외부 서비스

OpenAI API 키 : 개인별로 배포한 application-local.properties및 Jenkins내 파이프라인 스크립트에 해당 내용 있음(과금발생 가능성 존재하므로 취급에 주의할것)

## Gitgnore 처리한 핵심 키들

Vue : .env (최상단 위치)

Spring : application-local.properties, serviceAccountKey.json

(\src\main\resources, 또는 classPath에 위치)

전체 : .env (최상단 위치)

# 빌드

## 환경변수 형태

.env:

SPRING\_PROFILES\_ACTIVE=prod

SPRING\_DB\_USERNAME=포스트그레스 DB 사용자명

SPRING\_DB\_PASSWORD=포스트그레스 DB 비밀번호

SPRING\_MAIL\_USERNAME=이메일 관련기능 메일 사용자명

SPRING\_MAIL\_PASSWORD= 이메일 관련기능 메일 비밀번호

JWT\_ACCESS\_EXP=216000

JWT\_REFRESH\_EXP=36288000

JWT\_SECRET=JWT 토큰 생성위한 고유 암호

APP\_SERVICE\_URL=애플리케이션 서비스 URL

APP\_ALLOWED\_ORIGINS=교차출처접근 허용 URL

OPENVIDU\_URL=Openvidu 서버 URL

OPENVIDU\_SECRET= Openvidu 서버 비밀번호

OPENAI\_KEY=OpenAI API Key

.application-prod.properties

# Database

spring.datasource.driver-class-name=org.postgresql.Driver

spring.datasource.url=jdbc:postgresql://postgres-server:5432/pianobear

spring.datasource.username=${ENV\_SPRING\_DB\_USERNAME}

spring.datasource.password=${ENV\_SPRING\_DB\_PASSWORD}

# JPA

spring.jpa.database=postgresql

spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

spring.jpa.properties.hibernate.format\_sql=false

spring.jpa.properties.hibernate.show\_sql=false

# Redis

spring.data.redis.host=redis-server

spring.data.redis.port=6379

# Email

spring.mail.host=smtp.gmail.com

spring.mail.port=587

spring.mail.username=${ENV\_SPRING\_MAIL\_USERNAME}

spring.mail.password=${ENV\_SPRING\_MAIL\_PASSWORD}

spring.mail.properties.mail.smtp.auth=true

spring.mail.properties.mail.smtp.starttls.enable=true

# JWT (time = seconds)

jwt.access-expiration-time=${ENV\_JWT\_ACCESS\_EXP}

jwt.refresh-expiration-time=${ENV\_JWT\_REFRESH\_EXP}

jwt.secret=${ENV\_JWT\_SECRET}

# File Upload

file.save-path=/app/data/uploads/

spring.servlet.multipart.max-file-size=100MB

spring.servlet.multipart.max-request-size=100MB

#OpenVidu

application.openvidu-url=${ENV\_OPENVIDU\_URL}

application.openvidu-secret=${ENV\_OPENVIDU\_SECRET}

# Global

application.service-url=${ENV\_APP\_SERVICE\_URL}

application.allowed-origins=${ENV\_APP\_ALLOWED\_ORIGINS}

application.gpu-server-url=http://222.107.238.124:8888

#OpenAI

openai.api.key=${ENV\_OPENAI\_KEY}

#MusicTest

application.compare-server-url=http://audio-compare-server:7003/

## 빌드하기

1) Front

npm install

npm run build

2) Back-spring

Gradle 실행

Build task 실행

3) grading module

도커 이미지화 하였으므로 audiocompare 폴더에서 docker build하여 실행환경 구축

서버 구동 위치에서 python main.py

4) transcribe module

transcriber 폴더에서 setup.sh 스크립트 파일을 실행하고 start\_server.sh 스크립트 파일 실행하여 서버 시작

## 배포하기

Nginx 설정

upstream backend {

server localhost:7000;

}

upstream frontend {

server localhost:7001;

}

upstream openviduserver {

server localhost:5443;

}

server {

server\_name pianobear.kr;

# Proxy

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-Forwarded-Proto https;

# Websockets

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection "upgrade";

location / {

proxy\_pass http://frontend/;

}

location /swagger-ui/ {

proxy\_pass http://backend/swagger-ui/;

}

location /swagger-resources/ {

proxy\_pass http://backend/swagger-resources/;

}

location /v3/api-docs/ {

proxy\_pass http://backend/v3/api-docs/;

}

location /api/ {

proxy\_pass http://backend/api/;

}

########################

# OpenVidu Locations #

########################

#################################

# Common rules CE #

#################################

# Dashboard rule

location /dashboard {

allow all;

deny all;

proxy\_pass http://openviduserver;

}

# Websocket rule

location ~ /openvidu$ {

proxy\_pass http://openviduserver;

}

#################################

# New API #

#################################

location /openvidu/layouts {

rewrite ^/openvidu/layouts/(.\*)$ /custom-layout/$1 break;

root /opt/openvidu;

}

location /openvidu/recordings {

proxy\_pass http://openviduserver;

}

location /openvidu/api {

allow all;

deny all;

proxy\_pass http://openviduserver;

}

location /openvidu/info {

allow all;

deny all;

proxy\_pass http://openviduserver;

}

location /openvidu/accept-certificate {

proxy\_pass http://openviduserver;

}

location /openvidu/cdr {

allow all;

deny all;

proxy\_pass http://openviduserver;

}

listen 443 ssl; # managed by Certbot

ssl\_certificate /etc/letsencrypt/live/pianobear.kr/fullchain.pem; # managed by Certbot

ssl\_certificate\_key /etc/letsencrypt/live/pianobear.kr/privkey.pem; # managed by Certbot

include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot

ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}

server {

if ($host = pianobear.kr) {

return 301 https://$host$request\_uri;

} # managed by Certbot

server\_name pianobear.kr;

listen 80;

return 404; # managed by Certbot

}

이후 sudo service nginx start

Docker Compose 활용한 컨테이너 실행

docker-compose.yml 내용

services:

  application-server:

    build:

      context: ./backend/application

      dockerfile: Dockerfile

    ports:

      - "7000:8080"

    environment:

      SPRING\_PROFILES\_ACTIVE: ${SPRING\_PROFILES\_ACTIVE}

      ENV\_SPRING\_DB\_USERNAME: ${SPRING\_DB\_USERNAME}

      ENV\_SPRING\_DB\_PASSWORD: ${SPRING\_DB\_PASSWORD}

      ENV\_SPRING\_MAIL\_USERNAME: ${SPRING\_MAIL\_USERNAME}

      ENV\_SPRING\_MAIL\_PASSWORD: ${SPRING\_MAIL\_PASSWORD}

      ENV\_JWT\_ACCESS\_EXP: ${JWT\_ACCESS\_EXP}

      ENV\_JWT\_REFRESH\_EXP: ${JWT\_REFRESH\_EXP}

      ENV\_JWT\_SECRET: ${JWT\_SECRET}

      ENV\_APP\_SERVICE\_URL: ${APP\_SERVICE\_URL}

      ENV\_APP\_ALLOWED\_ORIGINS: ${APP\_ALLOWED\_ORIGINS}

      ENV\_OPENVIDU\_URL: ${OPENVIDU\_URL}

      ENV\_OPENVIDU\_SECRET: ${OPENVIDU\_SECRET}

      ENV\_OPENAI\_KEY: ${OPENAI\_KEY}

      TZ: Asia/Seoul

    depends\_on:

      - postgres-server

      - redis-server

    volumes:

      - pianobear-app-data:/app/data

    networks:

      - pianobear-network

  frontend-server:

    build:

      context: ./frontend/vue-project

      dockerfile: Dockerfile

    ports:

      - "7001:80"

    networks:

      - pianobear-network

  redis-server:

    image: redis:latest

    ports:

      - 6379:6379

    networks:

      - pianobear-network

  postgres-server:

    image: postgres:latest

    environment:

      POSTGRES\_USER: ${SPRING\_DB\_USERNAME}

      POSTGRES\_PASSWORD: ${SPRING\_DB\_PASSWORD}

      POSTGRES\_DB: pianobear

    ports:

      - "5432:5432"

    volumes:

      - pianobear-db-data:/var/lib/postgresql/data

    networks:

      - pianobear-network

  audio-compare-server:

    build:

      context: ./backend/audiocompare

      dockerfile: Dockerfile

    networks:

      - pianobear-network

networks:

  pianobear-network:

    driver: bridge

volumes:

  pianobear-app-data:

  pianobear-db-data:

프로젝트 루트에 존재하는 buildNrun.sh를 실행하면 도커 스택으로 프로젝트가 실행된다.

Transcriber 서버는 setup.sh 실행후 start\_server.sh 실행하면 실행된다.

## 외부 서비스 이용 방법

가) OpenAI API 설정

OpenAI API 키 생성

OpenAI 웹사이트에서 계정 생성 후 로그인

API 키를 생성하고, 보안을 위해 yml 파일에 추가 (gitignore 확인)

나) API 요청 코드 작성

이미지 생성 API 호출

사용 중인 언어에 맞는 OpenAI 클라이언트 라이브러리를 설치

API 키를 설정하고, 이미지 생성을 위한 코드를 작성

다) 웹 프로젝트와 통합

프론트엔드와 백엔드 통합

프론트엔드에서 사용자 입력을 받아 API 요청을 백엔드에 전송

백엔드에서 OpenAI API를 호출하여 생성된 이미지 URL을 반환

프론트엔드에서 이미지를 표시

라) 비용 및 사용량 관리

API 사용량 모니터링

OpenAI 대시보드에서 API 사용량과 비용을 정기적으로 확인

예산을 초과하지 않도록 모니터링하고, 필요시 호출 최적화 적용