|  |
| --- |
|  |

|  |
| --- |
| **부산와바 ~VR Tour** |
| **포팅 메뉴얼** |
|  |
|  |
|  |
|  |
|  |

폰트, 그래픽, 로고, 상징이(가) 표시된 사진

자동 생성된 설명

목차

[I. 개요 2](#_Toc96072459)

[1. 프로젝트 개요 2](#_Toc96072460)

[2. 프로젝트 사용 도구 2](#_Toc96072461)

[3. 개발환경 2](#_Toc96072462)

[4. 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) Teachable Machine 7](#_Toc96072470)

[나) STT 8](#_Toc96072471)

# 개요

## 프로젝트 개요

부산에는 손에 꼽기도 힘들만큼 다양한 여행지들이 있습니다.

이렇게 다양한 여행지들을 직접 가지 않고도 실시간 VR 영상으로 즐길 수 있다면 얼마나 재미있는 경험일까요?

부산와Vr는 부산을 더 생생하고 재미있게 알려주기 위해 탄생한 실시간 VR 스트리밍 서비스입니다.

부산에 관심 있는 사람들이 부산와Vr를 이용하면서 직접 부산을 직접 방문한 것처럼 즐거운 경험을 가질 수 있습니다.

## 프로젝트 사용 도구

이슈 관리 : JIRA

형상 관리 : Gitlab

커뮤니케이션 : Notion, Mattermost

디자인 : Figma

Flow-chart : Miro

UCC : 모바비, 애프터이펙트, 프리미어

CI/CD : Jenkins

## 개발환경

VS Code : 1.64.2,

IntelliJ : 11.0.12+7-b1504.40 x86\_64

JVM : 16.0.1 (스프링은1.8로 빌드)

Node.js : 14.16.0

SERVER : AWS EC2 Ubuntu 20.04.3 LTS

DB : MySql, Redis

## Gitignore 처리한 핵심 키들

React : .node\_module (최상단 위치)

Spring : application.properties

(\src\main\resources)

# 빌드

## 환경변수 형태

.build gradle

buildscript {  
 ext {  
 queryDslVersion = "5.0.0"  
 }  
}  
  
plugins {  
 id 'java'  
 id 'org.springframework.boot' version '2.7.13'  
 id 'io.spring.dependency-management' version '1.0.15.RELEASE'  
}  
  
group = 'com.example'  
version = '0.0.1-SNAPSHOT'  
  
java {  
 sourceCompatibility = '1.8'  
}  
  
repositories {  
 mavenCentral()  
}  
  
dependencies {  
 implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
 implementation 'org.springframework.boot:spring-boot-starter-security'  
 implementation 'org.springframework.boot:spring-boot-starter-web'  
 compileOnly 'org.projectlombok:lombok'  
 runtimeOnly 'com.mysql:mysql-connector-j'  
 annotationProcessor 'org.projectlombok:lombok'  
 testImplementation 'org.springframework.boot:spring-boot-starter-test'  
 testImplementation 'org.springframework.security:spring-security-test'  
 implementation 'org.springframework.boot:spring-boot-starter-validation'  
  
 *// Jwt* implementation("io.jsonwebtoken:jjwt-api:0.11.2")  
 implementation("commons-codec:commons-codec:1.5")  
 implementation("io.jsonwebtoken:jjwt-impl:0.11.2")  
 implementation("io.jsonwebtoken:jjwt-jackson:0.11.2")  
 implementation("com.auth0:java-jwt:3.13.0")  
  
 *//swagger* implementation("org.springdoc:springdoc-openapi-ui:1.6.9")  
  
 *// aws* implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.2.6.RELEASE'  
  
 *// email* implementation 'org.springframework.boot:spring-boot-starter-mail'  
  
 *// redis* implementation 'org.springframework.boot:spring-boot-starter-data-redis'  
  
 *// health check* implementation 'org.springframework.boot:spring-boot-starter-actuator'  
  
 *// Gson* implementation 'com.google.code.gson:gson:2.9.0'  
  
 *//Socket* implementation("org.webjars:stomp-websocket:2.3.3-1")  
 implementation("org.webjars:sockjs-client:1.1.2")  
 implementation("org.springframework.boot:spring-boot-starter-websocket")  
  
 *// Mongo DB* implementation 'org.springframework.boot:spring-boot-starter-data-mongodb'  
  
 *// QueryDSL  
 // queryDSL 설정* implementation "com.querydsl:querydsl-jpa"  
 implementation "com.querydsl:querydsl-core"  
 implementation "com.querydsl:querydsl-collections"  
 annotationProcessor "com.querydsl:querydsl-apt:${dependencyManagement.importedProperties['querydsl.version']}:jpa" *// querydsl JPAAnnotationProcessor 사용 지정* annotationProcessor "jakarta.annotation:jakarta.annotation-api" *// java.lang.NoClassDefFoundError (javax.annotation.Generated) 대응 코드* annotationProcessor "jakarta.persistence:jakarta.persistence-api" *// java.lang.NoClassDefFoundError (javax.annotation.Entity) 대응 코드*}  
  
*// Querydsl 설정부  
def* generated = 'src/main/generated'  
  
*// querydsl QClass 파일 생성 위치를 지정*tasks.withType(JavaCompile) {  
 options.getGeneratedSourceOutputDirectory().set(file(generated))  
}  
  
*// java source set 에 querydsl QClass 위치 추가*sourceSets {  
 main.java.srcDirs += [ generated ]  
}  
  
*// gradle clean 시에 QClass 디렉토리 삭제*clean {  
 delete file(generated)  
}  
tasks.named('test') {  
 useJUnitPlatform()  
}

.application.properties

*# mysql  
spring.datasource.driver-class-name*=com.mysql.cj.jdbc.Driver  
*spring.datasource.url*=jdbc:mysql://{DB IP주소}:3306/busan  
*spring.datasource.username*={디비 유저네임}   
*spring.datasource.password*={디비 비밀번호}   
*#jpa  
spring.jpa.hibernate.ddl-auto*=update  
*spring.jpa.database*=mysql  
*spring.jpa.database-platform*=org.hibernate.dialect.MySQL5InnoDBDialect  
*spring.jpa.show-sql*=true  
*spring.jpa.properties.hibernate.format\_sql*=true  
*spring.jpa.properties.hibernate.useSqlComments*=true  
  
*# secret  
spring.auth.secret.key*={JWT Token 발행을 위한 Secret Key}   
*#security  
spring.main.allow-circular-references*=true  
  
*cloud.aws.credentials.access-key*={S3 접근을 위한 access key}

*cloud.aws.credentials.secret-key*={S3 접근을 위한 secret key}   
*cloud.aws.region.static*=ap-northeast-2  
*cloud.aws.s3.buket*={버킷 이름}   
*cloud.aws.stack.auto*=false  
  
*# redis  
spring.redis.host*={접속할 redis IP}   
*spring.redis.port*=6379  
  
*# email sender  
mail.smtp.auth*=true  
*mail.smtp.starttls.required*=true  
*mail.smtp.starttls.enable*=true  
*mail.smtp.socketFactory.class*=javax.net.ssl.SSLSocketFactory  
*mail.smtp.socketFactory.fallback*=false  
*mail.smtp.port*=465  
*mail.smtp.socketFactory.port*=465  
  
*# admin ?? ??  
AdminMail.id*={SMTP를 위한 구글 아이디}   
*AdminMail.password*={SMTP를 위한 구글 비밀번호}   
  
*# MatterMost  
notification.mattermost.enabled*=true  
*notification.mattermost.webhook-url*={매터모스트 web hook url}   
  
*# MongoDB  
spring.data.mongodb.uri*=mongodb://localhost:27017/busan

## 빌드하기

1) Front

npm i

npm run build

2) Back-spring

Gradle 실행

쿼리DSL을 위한 compileJava 실행

Bootjar 실행

3) Back-flask

pip install -r requirements.txt

서버 구동 위치에서 python app.py

## 배포하기

Nginx 설정

server {

# listen 80 default\_server;

# listen [::]:80 default\_server;

# SSL configuration

#

listen 443 ssl default\_server;

listen [::]:443 ssl default\_server;

ssl on;

ssl\_certificate /etc/letsencrypt/live/busanwavrserver.store/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/busanwavrserver.store/privkey.pem;

#

# Note: You should disable gzip for SSL traffic.

# See: https://bugs.debian.org/773332

#

# Read up on ssl\_ciphers to ensure a secure configuration.

# See: https://bugs.debian.org/765782

#

# Self signed certs generated by the ssl-cert package

# Don't use them in a production server!

#

# include snippets/snakeoil.conf;

root /var/www/html;

# Add index.php to the list if you are using PHP

index index.html index.htm index.nginx-debian.html;

server\_name busanwaverserver.store;

include /etc/nginx/conf.d/service-url.inc;

location / {

# First attempt to serve request as file, then

# as directory, then fall back to displaying a 404.

proxy\_pass $service\_url;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection "Upgrade";

proxy\_set\_header Host $host;

}

# pass PHP scripts to FastCGI server

#

#location ~ \.php$ {

# include snippets/fastcgi-php.conf;

#

# # With php-fpm (or other unix sockets):

# fastcgi\_pass unix:/var/run/php/php7.4-fpm.sock;

# # With php-cgi (or other tcp sockets):

# fastcgi\_pass 127.0.0.1:9000;

#}

# deny access to .htaccess files, if Apache's document root

# concurs with nginx's one

#

#location ~ /\.ht {

# deny all;

#}

}

이후 sudo service nginx start

Nginx Front deploy 설정

server {

listen 443;

ssl on;

server\_name busanwavr.store;

ssl\_certificate /etc/letsencrypt/live/busanwavr.store/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/busanwavr.store/privkey.pem;

location / {

root /home/ubuntu/S09P12E102/FrontEnd/dist;

index index.html index.htm;

try\_files $uri $uri/ /index.html;

error\_page 405 = $uri;

}

}

이후 sudo service nginx start

Nginx 설정 – openvidu

server {

#listen 80 default\_server;

#listen [::]:80 default\_server;

# SSL configuration

#

# listen 443 ssl default\_server;

# listen [::]:443 ssl default\_server;

#

# Note: You should disable gzip for SSL traffic.

# See: https://bugs.debian.org/773332

#

# Read up on ssl\_ciphers to ensure a secure configuration.

# See: https://bugs.debian.org/765782

#

# Self signed certs generated by the ssl-cert package

# Don't use them in a production server!

#

# include snippets/snakeoil.conf;

listen 443 ssl default\_server;

listen [::]:443 ssl default\_server;

ssl on;

ssl\_certificate /etc/letsencrypt/live/busanopenvidu.store/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/busanopenvidu.store/privkey.pem;

root /var/www/html;

# Add index.php to the list if you are using PHP

index index.html index.htm index.nginx-debian.html;

server\_name busanopenvidu.store;

location / {

# First attempt to serve request as file, then

# as directory, then fall back to displaying a 404.

proxy\_pass http://127.0.0.1:8080;

}

# pass PHP scripts to FastCGI server

#

#location ~ \.php$ {

# include snippets/fastcgi-php.conf;

#

# # With php-fpm (or other unix sockets):

# fastcgi\_pass unix:/run/php/php7.4-fpm.sock;

# # With php-cgi (or other tcp sockets):

# fastcgi\_pass 127.0.0.1:9000;

#}

# deny access to .htaccess files, if Apache's document root

# concurs with nginx's one

#

#location ~ /\.ht {

# deny all;

#}

}

이후 sudo service nginx start

## 서비스 이용 방법

### Teachable Machine

<https://teachablemachine.withgoogle.com/train> 에 접속

- 포즈 프로젝트를 선택

- 3개의 class 를 만들기

- 1 class 는 왼손을 들어 학습

- 2 class 는 오른손을 들어 학습

- 3 class 는 가만히 있는 모습을 학습

- 학습 시작하기

- 프런트 코드에서 원격 machine url 등록

### STT

<https://learn.microsoft.com/ko-kr/azure/ai-services/speech-service/how-to-recognize-speech?pivots=programming-language-javascript> 에 접속

- Microsoft Azure 계정 생성

- 새로운 구독 설정

- 사용자 키 및 지역 설정 확인 후 소스 코드에 입력

- 소스 코드 내 원본 언어가 ko-KR로 설정되어 있는지 확인