|  |
| --- |
|  |



|  |
| --- |
| **Flea : ON 초대형 라이브 커머스** |
| **포팅 메뉴얼** |
|  |
|  |
|  |
|  |
|  |

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)

# 개요

## 프로젝트 개요

**프로젝트명 -** Flea:ON

**목표:** WebRTC 기술을 접목하여 일반인 누구나 쉽게 라이브 방식으로 중고 물품을 판매하고 구매할 수 있는 플랫폼 구축

→ 대량의 물품을 한번에, 쉽게 판매하고 구매할 수 있는 방법은 없을까? WebRTC 기술을 접목하여 개인이 호스트가 되는 초대형 중고 라이브 커머스 플랫폼 구축

**핵심 가치:**

* **간편함:** 쇼호스트 없이도 누구나 쉽게 라이브 판매 가능
* **신속함:** 라이브 커머스 방식으로 대량의 물품을 빠른 판매 및 구매
* **저렴함:** 마감 세일 효과로 저렴한 가격에 상품 구매 가능
* **편리함:** 판매 도우미 챗봇 지원, 거래 장소 및 시간 설정 가능
* **재미:** 숏폼 콘텐츠 제작 및 다음 상품 버튼으로 재미있는 쇼핑 경험 제공

**목적과 배경:**

* **목적:** 판매자는 적은 이동 비용으로 쉽게 판매할 수 있도록 도와주고 구매자는 싼 값에 구매할 수 있게 하겠다.
* **배경:** 중고거래의 가장 큰 불편함이 대면거래와 거래 준비 과정이 가장 많은 비율을 차지했다. 이점을 해소하기 위한 방법이 필요했다. → “많은 물건을 한번에” 에 좀 더 포커스
* 판매자
  + 구매자와의 연락, 가격 조율 ⇒ 일정 시간 안 팔리면 지나면 자동 가격 할인
  + 게시글 작성 등 거래 준비과정 ⇒ 한번에 여러 물품 판매 및 방송 보조
  + 개인정보 노출 ⇒ 거래 장소 설정 가능
  + 일방적인 거래 취소 ⇒ 구매자가 거래 취소해도 다음 거래자 자동 컨택
* 구매자
  + 거래물품 품질 상태 확인 ⇒ 직접 영상을 통해 물품 상태 확인 가능, 채팅을 통해 궁금한 부분 질문 가능

### ****타겟 페르소나****

1. **이사하는 가정**
   * **이름:** 김민수
   * **나이:** 40세
   * **직업:** 회사원
   * **목표:** 이사 전 집 정리를 하며 불필요한 물건을 처분하고 이사 비용을 절감
   * **고민:** 중고 물품을 효율적으로 팔기 어려움, 여러 플랫폼에 물건을 등록하고 관리하는 번거로움
2. **중고 물품 구매자**
   * **이름:** 박지현
   * **나이:** 28세
   * **직업:** 대학원생
   * **목표:** 저렴하고 유용한 중고 물품을 구매해 생활비 절약
   * **고민:** 중고 물품의 상태를 확인하기 어려움, 거래 과정에서 사기 당할 우려

### 주요 기능 및 기술

1. **라이브 판매**
   * 일반 사용자가 쇼핑 앱을 통해 간편하게 라이브 판매 방송 가능
   * 예약제로 구매 우선권 제공
   * 판매 도움 봇을 통한 예약자 관리 및 상품 정보 제공
     + 상품 추가 구매, 구매 취소, 날짜 및 시간 변경
   * 불건전한 방송 콘텐츠 감지 및 욕설, 비방어 채팅 금지
   * 이벤트성 무료 나눔 진행
2. **숏폼 콘텐츠**
   * 상품별 숏폼 콘텐츠로 변환
   * 숏폼 스크랩 기능
   * 숏폼에 대한 AI 요약 제공
3. **사용자 개인화**
   * 관심사 등록한 물품(숏폼)과 라이브의 알림 기능
   * 회원 등급 시스템 (애기손, 작은손, 중간손, 큰손)
   * 사용자의 관심 지역 위주 숏폼 추천
4. **기타**
   * 상품 카테고리 정보 자동 입력 기능 (네이버 쇼핑 API 사용)
   * 구매자측에서 특별히 채팅을 신청하지 않는 한, 단순 버튼 선택으로 거래일정 조율 및 취소, 추가 가능 ⇒ 사용자의 채팅 부담 완화

## 프로젝트 사용 도구

- 이슈 관리 : JIRA

- 형상 관리 : Gitlab

- 커뮤니케이션 : Notion, Mattermost

- 디자인 : Figma

- UCC : 모바비, VLLO

- CI/CD : Ubuntu 20.04, Docker, Docker hub, Docker compose, Jenkins, nginx

## 개발환경

## [ Front ]

- React 10.8.1

- HTML5, CSS3

- Node.js 20.15.0

- PWA

**[ Back ]**

- SpringBoot 3.3.2 (JDK 17.0.10)

- SpringSecurity

- Swagger

**[ DB ]**

- MySQL 8.0.37, Redis, Firebase

**[ Infra ]**

**-** Docker

- Jenkins

- Nginx

## 외부 서비스

- Google OAuth : application.properties파일에 해당 내용 있음

- Naver OAuth : application.properties파일에 해당 내용 있음

- KaKao OAuth : application.properties파일에 해당 내용 있음

- OpenAi API : .env 파일에 키가 있음

- react-speech-kit

- badwords-ko

## Gitgnore 처리한 핵심 키들

**[ Back ]**

``` HELP.md  
.gradle  
build/  
!gradle/wrapper/gradle-wrapper.jar  
!\*\*/src/main/\*\*/build/  
!\*\*/src/test/\*\*/build/  
  
### STS ###  
.apt\_generated  
.classpath  
.factorypath  
.project  
.settings  
.springBeans  
.sts4-cache  
bin/  
!\*\*/src/main/\*\*/bin/  
!\*\*/src/test/\*\*/bin/  
  
### IntelliJ IDEA ###  
.idea  
\*.iws  
\*.iml  
\*.ipr  
out/  
!\*\*/src/main/\*\*/out/  
!\*\*/src/test/\*\*/out/  
.env  
  
### NetBeans ###  
/nbproject/private/  
/nbbuild/  
/dist/  
/nbdist/  
/.nb-gradle/  
  
### VS Code ###  
.vscode/  
  
# json.key  
src/main/resources/firebase/firebasePushAlarm.json

**[ Front ]**

# See https://help.github.com/articles/ignoring-files/ for more about ignoring files. # dependencies /node\_modules /.pnp .pnp.js # testing /coverage # production /build # misc .DS\_Store .env.local .env.development.local .env.test.local .env.production.local npm-debug.log\* yarn-debug.log\* yarn-error.log\* .env env

# 빌드

## 환경변수 형태

**[ .application.properties ]**

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
spring.datasource.url=jdbc:mysql://i11b202.p.ssafy.io:3306/ssafy\_web\_db?serverTimezone=Asia/Seoul&useUnicode=true&characterEncoding=UTF-8  
spring.datasource.username=${DB\_PASSWORD}  
spring.datasource.password=${DB\_PASSWORD}  
  
spring.config.import=optional:file:.env[.properties]  
  
server.port=${SERVER\_PORT}  
server.ssl.enabled=false  
openvidu.url=${OPENVIDU\_URL}  
openvidu.secret=${OPENVIDU\_SECRET}  
server.domain=${SERVER\_DOMAIN}  
spring.mvc.converters.preferred-json-mapper=gson  
  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl  
  
  
spring.jwt.secret=${SPRING\_JWT\_SECRET}  
  
spring.servlet.multipart.max-file-size=10MB  
spring.servlet.multipart.max-request-size=10MB  
  
# JVM ??? ?? (?: Asia/Seoul)  
 spring.jpa.properties.hibernate.jdbc.time\_zone=Asia/Seoul  
  
  
#redis  
spring.data.redis.host=i11b202.p.ssafy.io  
spring.data.redis.port=6379  
spring.data.redis.password=${DB\_PASSWORD}  
spring.data.redis.timeout=2000  
  
logging.level.org.springframework.security=DEBUG  
  
  
#registration  
spring.security.oauth2.client.registration.naver.client-name=naver  
spring.security.oauth2.client.registration.naver.client-id=${NAVER\_CLIENT\_ID}  
spring.security.oauth2.client.registration.naver.client-secret=${NAVER\_CLINET\_SECRET}  
spring.security.oauth2.client.registration.naver.redirect-uri=${SERVER\_DOMAIN}/login/oauth2/code/naver  
spring.security.oauth2.client.registration.naver.authorization-grant-type=authorization\_code  
spring.security.oauth2.client.registration.naver.scope=name,email,nickname,profile\_image  
  
#provider  
spring.security.oauth2.client.provider.naver.authorization-uri=https://nid.naver.com/oauth2.0/authorize  
spring.security.oauth2.client.provider.naver.token-uri=https://nid.naver.com/oauth2.0/token  
spring.security.oauth2.client.provider.naver.user-info-uri=https://openapi.naver.com/v1/nid/me  
spring.security.oauth2.client.provider.naver.user-name-attribute=response  
  
  
#registration  
spring.security.oauth2.client.registration.google.client-name=google  
spring.security.oauth2.client.registration.google.client-id=${GOOGLE\_CLIENT\_ID}  
spring.security.oauth2.client.registration.google.client-secret=${GOOGLE\_CLIENT\_SECRET}  
spring.security.oauth2.client.registration.google.redirect-uri=${SERVER\_DOMAIN}/login/oauth2/code/google  
spring.security.oauth2.client.registration.google.authorization-grant-type=authorization\_code  
spring.security.oauth2.client.registration.google.scope=profile,email  
  
#provider  
spring.security.oauth2.client.provider.kakao.authorization-uri=https://kauth.kakao.com/oauth/authorize  
spring.security.oauth2.client.provider.kakao.token-uri=https://kauth.kakao.com/oauth/token  
spring.security.oauth2.client.provider.kakao.user-info-uri=https://kapi.kakao.com/v2/user/me  
spring.security.oauth2.client.provider.kakao.user-info-authentication-method=header  
spring.security.oauth2.client.provider.kakao.user-name-attribute=id  
  
#registration  
spring.security.oauth2.client.registration.kakao.client-id=${KAKAO\_CLIENT\_ID}  
spring.security.oauth2.client.registration.kakao.client-secret=${KAKAO\_CLIENT\_SECRET}  
spring.security.oauth2.client.registration.kakao.client-authentication-method=client\_secret\_post  
spring.security.oauth2.client.registration.kakao.authorization-grant-type=authorization\_code  
spring.security.oauth2.client.registration.kakao.scope=profile\_nickname,profile\_image, account\_email  
spring.security.oauth2.client.registration.kakao.redirect-uri=${SERVER\_DOMAIN}/login/oauth2/code/kakao  
spring.security.oauth2.client.registration.kakao.client-name=Kakao

## 빌드하기

**[ Front ]**

- npm i : 의존성 설치

- npm run build : 프로젝트 빌드

**[ Back ]**

## - /gradlew build : Gradle 실행

- ./gradlew bootRun : 빌드된 jar 파일 실행

## 배포하기

## [ Jenkins 파이프라인 ]

app: image: qsc753969/fleaon:latest restart: on-failure network\_mode: host volumes: - /opt/openvidu/recordings/live:/opt/openvidu/recordings/live environment: - SERVER\_PORT=5442 - OPENVIDU\_URL=http://localhost:5443 - OPENVIDU\_SECRET=${OPENVIDU\_SECRET} - CALL\_OPENVIDU\_CERTTYPE=${CERTIFICATE\_TYPE} - CALL\_PRIVATE\_ACCESS=${CALL\_PRIVATE\_ACCESS:-} - CALL\_USER=${CALL\_USER:-} - CALL\_SECRET=${CALL\_SECRET:-} - CALL\_ADMIN\_SECRET=${CALL\_ADMIN\_SECRET:-} - CALL\_RECORDING=${CALL\_RECORDING:-} - KAKAO\_CLIENT\_ID=fe4fd71c9de673c8302ca519ded2aa58 - KAKAO\_CLIENT\_SECRET=ekax6Kcl2OhQovyqnk10CEF2J3Ty6LmC - NAVER\_CLIENT\_ID=ya1SxGAelvqiuXj2oWqS - NAVER\_CLINET\_SECRET=DpLuioa5f5 - GOOGLE\_CLIENT\_ID=326298975697-uckaiqbdsc56cgf06m249n8eua5vr3n0.apps.googleusercontent.com

- GOOGLE\_CLIENT\_SECRET=GOCSPX-qlgNTNuEotu4AwqJe-rD\_RiixZbf - SPRING\_JWT\_SECRET=dyddnjsdksjtkrlekdgksrjdirmaldydtlfektlrkwlakfpwjsem69dyddnjsehlaaaaaaaakkkkkkkk - SERVER\_DOMAIN=https://i11b202.p.ssafy.io - DB\_PASSWORD=ssafy - OPENAI\_API\_KEY=sk-3anYys5uiHGo-iiK2ihldv7Co9hZRUgZk\_rx97\_DBnT3BlbkFJh0YRSRPHbu-bvUcVA78EauR9Jj5Uhi3YHx-gRlKRkA - TZ=Asia/Seoul #- SERVER\_DOMAIN = http://localhost:5442 logging: options: max-size: "${DOCKER\_LOGS\_MAX\_SIZE:-100M}"

pipeline { agent any environment { DOCKER\_HUB\_CREDENTIALS = credentials('docker-hub-credentials') // Docker Hub 자격 증명 ID } stages { stage('Clone') { steps { // Git backend 리포지토리 클론 git credentialsId: 'ssafy', url: 'https://gitlab+deploy-token-6221:gldt-eEXEg9U7BwZoyhU1Y7C7@lab.ssafy.com/s11-webmobile1-sub2/S11P12B202.git', branch: 'frontend' } } stage('Set Permissions') { steps { script { sh 'ls' sh 'cd frontend/docker && chmod +x create\_image.sh' } } } stage('docker-frontend') { steps { withCredentials([usernamePassword(credentialsId: 'docker-hub-credentials', usernameVariable: 'DOCKER\_USERNAME', passwordVariable: 'DOCKER\_PASSWORD')]) { sh 'echo $DOCKER\_PASSWORD | docker login -u $DOCKER\_USERNAME --password-stdin' } // sh 'docker build . -t qsc753969/fleaon\_front:latest' sh 'cd frontend/docker && ./create\_image.sh' sh 'docker images' sh 'docker push qsc753969/fleaon\_front:latest' } } // stage('react-restart') { // steps { // sh 'docker run -p 3000:3000 qsc753969/fleaon\_front:latest' // } // } } }

pipeline { agent any environment { DOCKER\_HUB\_CREDENTIALS = credentials('docker-hub-credentials') // Docker Hub 자격 증명 ID } stages { stage('Clone') { steps { // Git backend 리포지토리 클론 git credentialsId: 'ssafy', url: 'https://gitlab+deploy-token-6221:gldt-eEXEg9U7BwZoyhU1Y7C7@lab.ssafy.com/s11-webmobile1-sub2/S11P12B202.git', branch: 'backend' } } stage('Set Permissions') { steps { script { sh 'ls' sh 'cd backend/docker && chmod +x create\_image.sh' } } } stage('docker-backend') { steps { withCredentials([usernamePassword(credentialsId: 'docker-hub-credentials', usernameVariable: 'DOCKER\_USERNAME', passwordVariable: 'DOCKER\_PASSWORD')]) { sh 'echo $DOCKER\_PASSWORD | docker login -u $DOCKER\_USERNAME --password-stdin' } sh 'cd backend/docker && ./create\_image.sh qsc753969/fleaon:latest' sh 'docker images' sh 'docker push qsc753969/fleaon:latest' } } // stage('openvidu-restart') { // steps { // sh 'JENKINS\_NODE\_COOKIE=dontKillMe cd /opt/openvidu && ./openvidu restart &' // } // } } }

**[ Nginx 설정 ]**

# Your App upstream yourapp { server localhost:5442; } upstream openviduserver { server localhost:5443; } upstream front { server localhost:3000; } server { listen 80; listen [::]:80; server\_name i11b202.p.ssafy.io; location /videos/ { alias /opt/openvidu/recordings/; } # Redirect to https location / { rewrite ^(.\*) https://i11b202.p.ssafy.io:443$1 permanent; } # letsencrypt location /.well-known/acme-challenge/ { root /var/www/certbot; } location /nginx\_status { stub\_status; allow 127.0.0.1; #only allow requests from localhost deny all; #deny all other hosts } }

ver { listen 443 ssl; listen [::]:443 ssl; server\_name i11b202.p.ssafy.io; # SSL Config ssl\_certificate /etc/letsencrypt/live/i11b202.p.ssafy.io/fullchain.pem; ssl\_certificate\_key /etc/letsencrypt/live/i11b202.p.ssafy.io/privkey.pem; ssl\_trusted\_certificate /etc/letsencrypt/live/i11b202.p.ssafy.io/fullchain.pem; ssl\_session\_cache shared:SSL:50m; ssl\_session\_timeout 5m; ssl\_stapling on; ssl\_stapling\_verify on; ssl\_protocols TLSv1.2 TLSv1.3; ssl\_ciphers "ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-C ssl\_prefer\_server\_ciphers off; add\_header Strict-Transport-Security "max-age=63072000" always; # 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; proxy\_headers\_hash\_bucket\_size 512; proxy\_redirect off; # Websockets proxy\_http\_version 1.1; proxy\_set\_header Upgrade $http\_upgrade; proxy\_set\_header Connection "upgrade"; # Your App location /front/ { proxy\_pass http://front/; # Openvidu call by default allow all; deny all;

# add\_header 'Access-Control-Allow-Origin' '\*'; } ######################## # 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; } location / { allow all; deny all; proxy\_pass http://yourapp; add\_header 'Access-Control-Allow-Methods' 'GET, POST, PUT, DELETE, OPTIONS'; } ################################# # LetsEncrypt # ################################# location /.well-known/acme-challenge { root /var/www/certbot; try\_files $uri $uri/ =404; } location /videos/ { alias /opt/openvidu/recordings/; } }

**[ Front DockerFile ]**

**FROM** node:20:15.0-alpine

**WORKDIR** /front

**COPY** package.json .

**RUN** npm install

**COPY** . .

**EXPOSE** 3000

**CMD** ["npm", "start"]

**[ Back DokerFile ]**

*# Stage 1: Build*

**FROM** gradle:8.9-jdk17 as build

**WORKDIR** /openvidu-recording-java

**COPY** ./build.gradle build.gradle

**COPY** ./settings.gradle settings.gradle

**COPY** ./src/main src/main

**RUN** gradle clean build

*# Stage 2: Package*

**FROM** alpine:3.15

*# Install OpenJDK 17*

**RUN** apk update && \

    apk add openjdk17 && \

    rm -rf /var/cache/apk/\*

*# Install basic-webinar*

**RUN** mkdir -p /opt/openvidu-recording-java

**COPY** --from=build /openvidu-recording-java/build/libs/\*.jar /opt/openvidu-recording-java/openvidu-recording-java.jar

*# Entrypoint*

**COPY** ./docker/entrypoint.sh /usr/local/bin

**RUN** chmod +x /usr/local/bin/entrypoint.sh

**CMD** ["/usr/local/bin/entrypoint.sh"]

## 서비스 이용 방법

### 가) Google OAuth

* 구글 클라우드에서 프로젝트 생성하기
* OAuth 동의 화면 만들기
* OAuth 클라이언트의 ID와 비밀번호 만들기

### 나) Naver OAuth

* 네이버 개발자에서 프로젝트 등록하기
* 내 애플리케이션 등록 후 API 신청하기
* 네이버 로그인 URL 만들어서 인가 코드 받기

### 다) Kaokao OAuth

* Kakao Developers에서 개발자 계정 생성
* 애플리케이션 추가
* REST\_API 키 발급받기
* 카카오 로그인 활성화하기
* Redirect URI 등록하기

### OpenAI API

* OenAI에 들어가서 계정 생성하기
* API 신청하기 -> 과금이 발생할 수도 있음
* SECRET\_KEY 발급받기

## 시연 시나리오

## 로그인

## 소셜 로그인 3개 중에 하나 선택

## 로그인 진행 후, 추가 정보 입력

## 로그인 완료

## 라이브 생성

## 메인 페이지에서 “+”을 눌러서 라이브를 생성함

## 라이브 시작

## 구매자들의 채팅

## 구매 버튼이 활성화가 되면 구매 버튼 클릭

## 구매자가 되지 못했을 때 줄서기 버튼 활성화

## 라이브 완료

## 채팅

## 구매자들과 챗봇간의 채팅

## 추가 문의 사항이 있는 경우 판매자와 직접 채팅

## 마이페이지

## 마이페이지 달력에서 거래 내역을 확인

## 구매내역, 판매내역을 확인

## 개인 정보를 조회하고 수정

## 메인 페이지 검색

## 메인 페이지에서 물품의 이름으로 검색 가능

## 해당 물품이 판매되고 있는 라이브, 쇼츠 목록 확인 가능