|  |
| --- |
|  |

|  |
| --- |
| **체리씨** |
| **포팅 매뉴얼** |
|  |
|  |
|  |
|  |
|  |



목차

[I. 개요 2](#_Toc158738222)

[1. 사용 도구 2](#_Toc158738223)

[2. 개발환경 2](#_Toc158738224)

[A. 프론트 엔드 2](#_Toc158738225)

[B. 백엔드 2](#_Toc158738226)

[C. 서버 2](#_Toc158738227)

[3. 외부 서비스 2](#_Toc158738228)

[A. 프론트 엔드 2](#_Toc158738229)

[B. 백엔드 2](#_Toc158738230)

[4. 환경변수 2](#_Toc158738231)

[A. 프론트 엔드 2](#_Toc158738232)

[B. 백엔드 2](#_Toc158738233)

[II. 빌드 2](#_Toc158738234)

[1. 환경변수 형태 2](#_Toc158738235)

[2. 빌드하기 4](#_Toc158738236)

[3. 배포하기 4](#_Toc158738237)

[4. 서비스 이용 방법 6](#_Toc158738238)

[가) 카카오 페이 6](#_Toc158738239)

[나) Firebase 실시간 알림 6](#_Toc158738240)

[다) Google Cloud Platform 7](#_Toc158738241)

# 사용 도구

**이슈 관리** : JIRA

**형상 관리** : Gitlab

**커뮤니케이션 및 문서화** : Notion, Mattermost

**디자인** : Figma

**CI/CD** : Docker, Jenkins

# 개발환경

* 1. 프론트 엔드

Node.js : 20.10.0

React : 18.2.0  
Visual Studio Code : 1.85.1

* 1. 백엔드

Java : 17.0.9

**MySql** : 8.0.36

**Intellij** : 2023.3.2

**ffmpeg** : 6.1.1

* 1. 서버

**OS** : AWS EC2 Ubuntu 20.04.6 LTS

**MySql** : 8.0.36

**Docker** : 25.0.1

# 외부 서비스

* 1. 프론트 엔드

OpenAI API : GPT API를 사용하였습니다.

.env.local에 키값을 저장하였고 .env.local은 gitignore로 공유저장소에 올리지 않고 따로 저장하고 있습니다.

Kakao API : 카카오 로그인과 카카오 알림 기능을 사용하였습니다.

.env.local에 키값을 저장하였고 .env.local은 gitignore로 공유저장소에 올리지 않고 따로 저장하고 있습니다.

* 1. 백엔드

S3 : 동영상 클립들을 저장하고 불러오는 저장소로 사용하였습니다.

application-secrets.yml에 접속 링크를 저장하였고 application-secrets.yml은gitignore로 공유저장소에 올리지 않고 따로 저장하고 있습니다.

# 환경변수

* 1. 프론트 엔드

.env.local : 노출되면 안되는 중요정보들을 아래와 같이 작성하여 gitignore로 공유 저장소에 올리지 않고 따로 개발환경에서 불러와 사용하였습니다

VITE\_APP\_SOCKET\_URL=

VITE\_APP\_KAKAO\_JAVASCRIPT\_KEY=

VITE\_APP\_KAKAO\_REST\_API\_KEY=

VITE\_APP\_BACKEND\_URL=

VITE\_APP\_GPT\_API\_KEY=

* 1. 백엔드

application.yml : 노출되어도 괜찮은 정보들은 아래와 같이 작성하여 저장하였습니다.

spring:  
 profiles:  
 include:  
 - secrets  
 datasource:  
 *#hikari Datasource 설정* hikari:  
 driver-class-name: com.mysql.cj.jdbc.Driver  
 jdbc-url: jdbc:mysql://${MYSQL\_HOST:localhost}:3306/cherish?serverTimezone=UTC&useUniCode=yes&characterEncoding=UTF-8  
 username: ${db.username}  
 password: ${db.password}  
 pool-name: Hikari Connection Pool  
 *# 타임리프 설정* thymeleaf:  
 prefix: classpath:templates/  
 check-template-location: true  
 suffix: .html  
 mode: HTML5  
 cache: false  
 order: 0  
 *# 동영상 blob 파일 저장 용량 설정* servlet:  
 multipart:  
 maxFileSize: 100MB *# 파일 하나당 용량* maxRequestSize: 200MB *# 요청 하나당 용량  
  
# log 수준 debug로 설정*logging:  
 level:  
 root:  
 zaxxer:  
 hikari: DEBUG  
 com:  
 ssafy:  
 cherish: DEBUG  
  
*# 로컬 스토리지 저장을 위한 자체 경로 정보  
# EC2에 맞춰서 수정 필요*custom:  
 path:  
*# mac 기준  
# String ffmpegPath=System.getProperty("user.home")+"/ffmpeg/6.1.1\_3/bin";* ffmpeg: "C:\\Program Files\\ffmpeg-6.1.1-essentials\_build\\bin\\"  
 monthly-video: "C:\\Users\\SSAFY\\Documents\\cherish\_video\\monthly-video\\"  
 clip: "C:\\Users\\SSAFY\\Documents\\cherish\_video\\clip\\"  
 answer: "C:\\Users\\SSAFY\\Documents\\cherish\_video\\answer\\"  
  
server:  
 servlet:  
 context-path: /api

application-secrets.yml : 노출되면 안되는 중요정보들을 아래와 같이 작성하여 gitignore로 공유 저장소에 올리지 않고 따로 개발환경에서 불러와 사용하였습니다

db:  
 username:   
 password:   
  
cloud:  
 aws:  
 credentials:  
 access-key:   
 secret-key:

s3:  
 bucket:   
 region:  
 static:   
 stack:  
 auto:

# 빌드

## 빌드하기

1) Front

npm i

npm run build

2) Back-spring

Gradle 실행

Bootjar 실행

3) Back-flask

pip install -r requirements.txt

서버 구동 위치에서 python app.py

## 배포하기

Nginx 설정

upstream jenkins {

server 127.0.0.1:4000;

keepalive 32;

}

map $http\_upgrade $connection\_upgrade {

default upgrade;

'' close;

}

server {

listen 80 default\_server;

listen [::]:80 default\_server;

server\_name tupli.kr;

return 301 https://$server\_name$request\_uri;

}

server {

listen 443 ssl;

listen [::]:443 ssl;

server\_name tupli.kr;

ssl\_certificate /etc/letsencrypt/live/tupli.kr/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/tupli.kr/privkey.pem;

root /var/www/html/dist;

index index.html;

location / {

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

}

location /api/v1 {

proxy\_pass http://localhost:8080;

proxy\_redirect off;

charset utf-8;

rewrite /api/v1/(.\*) /$1 break;

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-NginX-Proxy true;

}

location /api/v1/ws-stomp {

proxy\_pass http://localhost:8080;

rewrite /api/v1/(.\*) /$1 break;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection "Upgrade";

proxy\_set\_header Host $host;

}

location /flask {

proxy\_pass http://localhost:5000;

proxy\_redirect off;

charset utf-8;

rewrite /flask/(.\*) /$1 break;

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-NginX-Proxy true;

}

}

이후 sudo service nginx start

## 서비스 이용 방법

### 카카오 페이

준비 ; 카카오페이 API 등록

- 내 애플리케이션 → 애플리케이션 추가하기

- 들어갈 경우 Admin키가 있음 → 보안을 위해 yml 파일에 추가(gitgnore 했는지 확인!)

- 플랫폼 등록 ⇒ 웹 (적는것은 프론트!)

### Firebase 실시간 알림

1. Firebase 콘솔 프로젝트 만들기 (GCP에 프로젝트 있을 경우 그거 선택)

2. 설정 → 프로젝트 설정 → 서비스 계정 → 새 비공개 키 생성

3. realtimedatabase 생성 후 주소 확인

### Google Cloud Platform

1. Google Cloud Platform 접속 후 새 프로젝트 만들기

→ 좌측 햄버거 메뉴에서 저장소 Cloud Storage 클릭

2. 브라우저로 이동될텐데 버킷 만들기 선택 후 버킷 이름 지정하고 만들기

일단은 tupli-profile로 생성 (이 버킷 이름은 나중에 spring에서 저장 위치로 사용됨)

3. springboot 접속용 access key를 만들어야 함.

좌측 햄버거 메뉴 다시 열고 IAM 및 관리자 > 서비스 계정 으로 이동후

서비스 계정 만들기 클릭

4. 서비스 계정이름을 만들고 (tupli-profile-access),

권한 부여에 저장소 관리자, 저장소 개체 관리자 부여

5. 서비스 계정 만들고 나올 경우, 속성 등으로 키로 이동

키 추가 선택하여 json 다운로드