EXEC

1. Stacks

Frontend

Language | Javascript

Framework | Vue.js 3.3.11, Vue-router 4.2.5, pinia 2.1.7

Node | Node 20.10.0

Build Tool | Vite 5.0.10

IDE | VS Code 1.85.1

Backend

Language | Java 17

Framework | Spring Boot 3.2.1

Build Tool | Gradle 8.5

DB | MySQL 8.0.35 , Spring-Data-JPA, Redis

API Docs | Swagger

IDE | Intellij IDEA 2023.3.3

Infra

Infra | AWS EC2 (Ubuntu 20.04.6 LTS) , AWS S3, Nginx 1.18.0 (Ubuntu)

CI/CD | Git, Docker 25.0.0, Jenkins 2.426.2

Management Tool

Jira, Notion, Mattermost

2. Build & Distribute

Spring Boot

- dockerfile

```
FROM openjdk:17-alpine
CMD ["./gradlew", "clean", "bootJar"]
COPY build/libs/*.jar app.jar
ENTRYPOINT ["java", "-Dspring.profiles.active=dev", "-jar"
RUN mkdir -p /download/live
RUN mkdir -p /download/shortping
```

Vue

- dockerfile

```
FROM node:20.10.0 as build-stage

#폴더 위치
RUN mkdir -p /app
WORKDIR /app
ADD . .

#yarn 설치
RUN yarn install
RUN yarn run build

# production stage
FROM nginx:stable-alpine as production-stage
COPY  ./nginx/nginx.conf /etc/nginx/conf.d/default.conf

COPY --from=build-stage /app/dist /usr/share/nginx/html
```

```
EXPOSE 5173
CMD ["nginx", "-g", "daemon off;"]
```

3. Deployment Command

Jenkins를 이용하여 CI/CD 구축

Spring Boot

```
cd ./backend

# docker image build
docker build -t loverduck/pasila-backend:latest .

cd /home/ubuntu

# docker image build and container run
docker run -d -i --env-file env/.env -e TZ=Asia/Seoul --name
```

Vue

```
cd ./front

# docker image build
docker build -t loverduck/pasila-frontend:latest .

cd /home/ubuntu

# docker image build and container run
docker run -d -i --env-file env/.env -e TZ=Asia/Seoul --name
```

Etc

```
# jenkins
docker run -d -p 9090:8080 -v /home/ubuntu/jenkins-data:/var/

# redis
docker run -d -p 6379:6379 --name redis redis:latest

# mysql
docker run -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD=비밀번호 -v /

# ffmpeg-api
docker run -d -p 3000:3000 --name ffmpeg-api kazhar/ffmpeg-ap
```

4. MySQL WorkBench Connection

Spring Boot에서 연동

- application-dev.yml

```
spring:
  datasource:
    url: jdbc:mysql://${MYSQL_URL}:${MYSQL_PORT}/pasila?se
    username: ${MYSQL_USERNAME}
    password: ${MYSQL_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver
```

5. EC2 Setting

Port Setting

- frontend server: 5173
- backend server: 8080
- ffmpeg server : 3000
- opendvidu
  - https: 8443
  - http: 8442
  - STUN/TURN server client ips: 3478
  - kurento media server: 40000-57000
  - TURN server establish media connections: 57001 - 65535
  - 5442, 5443, 6379, 8888
- jenkins: 9090
- redis: 6379

EC2 Setting

- install docker
- install openvidu
- install nginx
- run container

Jenkins Setting

- jenkins 내 docker-ce, docker-compose 설치
- plugin install
  - Docker
  - Docker compose
  - Docker Pipeline

    - Docker API
    - NodeJS
    - SSH Agent
    - Generic Webhook Trigger
    - GitLab

- pipeline 설정

6. Nginx Default

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;

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

        server_name _;

        location / {
                try_files $uri $uri/ =404;
        }
}

server {
                root /var/www/html;

        index index.html index.htm index.nginx-debian.html;
        server_name i10a402.p.ssafy.io;

        location / {
                proxy_pass ${pasila-frontend url};

                add_header 'Cross-Origin-Embedder-Policy' 'cr
                add_header 'Cross-Origin-Opener-Policy' 'same
                add_header 'Cross-Origin-Resource-Policy' 'cr
        }

        location /video/extract/download {
                proxy_pass ${ffmpeg-api url}/video/extract/do
        }

        location /download {
                proxy_pass ${pasila-backend url}/download;
        }

        location /api {
                proxy_pass ${pasila-backend url}/api;
        }

        location /api/real-time/subscribe {
                proxy_http_version 1.1;
                proxy_set_header Connection '';
                proxy_set_header X-Accel-Buffering no;
                proxy_set_header Content-Type 'text/event-str
                proxy_buffering off;
                chunked_transfer_encoding on;

                proxy_pass ${pasila-backend url}/api/real-tim
        }


                location /stomp/pasila {
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_set_header Host $host;
                proxy_hide_header X-Frame-Options;
                proxy_pass ${pasila-backend url}/stomp/pasila
        }

        listen [::]:443 ssl ipv6only=on; # managed by Certbot
        listen 443 ssl; # managed by Certbot

        ssl_certificate /etc/letsencrypt/live/i10a402.p.ssafy
        ssl_certificate_key /etc/letsencrypt/live/i10a402.p.s
        ssl_trusted_certificate /etc/letsencrypt/live/i10a402

        # Websockets
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";

        include /etc/letsencrypt/options-ssl-nginx.conf; # ma
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # mana


        location /.well-known/acme-challenge {
                root /var/www/certbot;
                try_files $uri $uri/ =404;
        }

}
server {
    if ($host = i10a402.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
        listen 80 ;
        listen [::]:80 ;
    server_name i10a402.p.ssafy.io;
    return 404; # managed by Certbot
}
```

7. Files ignore

Environment variable

- env/.env

  EC2 내 경로에 존재하는 환경변수 파일입니다.

```
MYSQL_URL="MySQL 접속 url"
MYSQL_PORT="MySQL 접속 port"
MYSQL_USERNAME="MySQL 사용자 아이디"
MYSQL_PASSWORD="MySQL 사용자 비밀번호"

OPEN_AI_STYLE_MODEL="말투 데이터셋 fine-tuning한 gpt-3.5-turb
OPEN_AI_KEY="OpenAI api key"

AWS_ACCESSKEY="AWS access key"
AWS_SECRETKEY="AWS secret key"

COOLSMS_APIKEY="coolsms api key"
COOLSMS_APISECRET="coolsms api secret"
COOLSMS_FROMNUMBER="coolsms에서 사용할 전화번호"

FFMPEG_URL="FFMPEG API 주소"

REDIS_URL="redis 컨테이너 주소"
REDIS_PORT="redis 컨테이너 접속 port"

OPENVIDU_URL="openvidu server 주소"
OPENVIDU_SECRET="openvidu secret"

MAIL_URL="gmali smtp 주소"
MAIL_PORT="smtp port"
USER_EMAIL="이메일 전송에 사용할 이메일 주소"
USER_PASSWORD="이메일 계정 비밀번호"

JWT_SECRET="JWT 토큰 생성시 사용되는 secret"

AES_SECRET="암호화 secret"
AES_SALT="암호화 salt"
```

```
DDL_AUTO="ddl auto 사용여부"
```

- application-dev.yml

```
spring:
  datasource:
    url: jdbc:mysql://${MYSQL_URL}:${MYSQL_PORT}/pasila?se
    username: ${MYSQL_USERNAME}
    password: ${MYSQL_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver

  servlet:
    multipart:
      max-file-size: 1000MB
      max-request-size: 1000MB
      enabled: true
      location: /download/

  jpa:
    hibernate:
      ddl-auto: ${DDL_AUTO}
    properties:
      hibernate:
        format_sql: true
        default_batch_fetch_size: 100

logging:
  level:
    org.hibernate.SQL: debug
  file:
    path: logs

#chatGpt
openai:
  model: gpt-3.5-turbo
  style-model: ${OPEN_AI_STYLE_MODEL}
  api:
    url: https://api.openai.com/v1
    key: ${OPEN_AI_KEY}

#s3
cloud:
  aws:
    credentials:
      accessKey: ${AWS_ACCESSKEY}
      secretKey: ${AWS_SECRETKEY}
    s3:
      bucket: pasila
    region:
      static: ap-northeast-2
    stack:
      auto: false

#server
server:
  port: 80

#swagger
springdoc:
  swagger-ui:
    # swagger-ui 접근 경로. default 값은 /swagger-ui.html이다.
    path: /swagger-pasila-ui.html

    # 각 API의 그룹 표시 순서
    # path, query, body, response 순으로 출력
    groups-order: DESC

    # 태그 정렬 순서.
    # alpha: 알파벳 순 정렬
    # method: OpenAPI specification file에 원하는 태그 정렬 방
    tags-sorter: alpha

    # 컨트롤러 정렬 순서.
    # method는 delete - get - patch - post - put 순으로 정렬
```

```
      # alpha를 사용해 알파벳 순으로 정렬할 수 있다.
      operations-sorter: method

      # swagger-ui default url인 petstore html의 비활성화 설정
      disable-swagger-default-url: true

      # swagger-ui에서 try 했을 때 request duration을 알려주는 설
      display-request-duration: true

    # openAPI 접근 경로. default 값은 /v3/api-docs 이다.
    api-docs:
      path: /api-docs

    # Spring Actuator의 endpoint까지 보여줄 것인지?
    show-actuator: true

    # request media type 의 기본 값
    default-consumes-media-type: application/json

    # response media type 의 기본 값
    default-produces-media-type: application/json

    # 해당 패턴에 매칭되는 controller만 swagger-ui에 노출한다.
    paths-to-match:
      - /api/**

  # coolsms
  coolsms:
    apiKey: ${COOLSMS_APIKEY}
    apiSecret: ${COOLSMS_APISECRET}
    fromNumber: ${COOLSMS_FROMNUMBER}

  # redis
  redis:
    host: ${REDIS_URL}
    port: ${REDIS_PORT}

  # ffmpeg
```

```
ffmpeg:
  url: ${FFMPEG_URL}

# openvidu
openvidu:
  openvidu_url: ${OPENVIDU_URL}
  openvidu_secret: ${OPENVIDU_SECRET}

# google SMTP
mail:
  protocol: smtp
  host: ${MAIL_URL}
  port: ${MAIL_PORT}
  username: ${USER_EMAIL}
  password: ${USER_PASSWORD}
  properties:
    mail:
      smtp:
        auth: true
        timeout: 5000
        starttls:
          enable: true
          required: true

jwt:
  expiration_time: 86400000
  secret: ${JWT_SECRET}

aes:
  secret: ${AES_SECRET}
  salt: ${AES_SALT}
```

외부 서비스

OpenAI API

자연어 처리를 비롯한 다양한 ai 기술들을 활용하여 다양한 기능을 제공하는 API

https://platform.openai.com/

- GPT-3.5-Turbo 모델을 사용한 Chat Completions
- Whisper 모델을 사용한 Speech-to-text

OpenVidu

웹 또는 모바일 환경에서 화상 회의 기능을 쉽게 추가할 수 있도록 해주는 오픈소스 멀티 플랫폼

version: 2.29.0

FFmpeg

영상 및 음성과 같은 멀티미디어의 인코딩/디코딩을 제공하는 오픈소스 라이브러리

영상 편집 및 음성 추출에 사용하였습니다.