



프로젝트 최종 보고서(포팅메뉴얼)

태그

보고서

1. 개요

나의 마음과 추억을 'NFC 키링'에 담아 선물하고, 꾸준히 쌓은 사진 및 메시지를 태깅으로 간편하게 확인할 수 있는 서비스

2. 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost
- 디자인 : Figma
- CI/CD : Jenkins, Docker

3. 개발 도구

- Intellij : 2024.3.1.1
- VisualStudioCode: 1.99.1

4. 개발 환경

4-1. React Native (모바일 앱)

항목	버전
React Native	0.78.2
React	19.0.0
Node.js	>=18

Dependencies

라이브러리	버전
Axios	^1.8.4
Lottie React Native	^7.2.2

react-native-device-info	^14.0.4
react-native-fs	^2.20.0
react-native-nfc-manager	^3.16.1
react-native-url-polyfill	^2.0.0
react-native-webview	^13.13.5

DevDependencies

라이브러리	버전
@babel/core	^7.25.2
@react-native-community/cli	15.0.1
@react-native/babel-preset	0.78.2
@react-native/metro-config	0.78.2
@types/react	^19.0.0
typescript	5.0.4
eslint	^8.19.0
jest	^29.6.3
prettier	2.8.8

4-2. React (웹 프론트엔드)

항목	버전
React	^18.2.0
React DOM	^18.2.0
Vite	^6.2.0
Node.js	>=18

Dependencies

라이브러리	버전
Axios	^1.8.3
Lottie React	^2.4.1
Recoil	^0.7.7
Styled Components	^6.1.15
React Router DOM	^7.3.0
Framer Motion	^12.5.0
Swiper	^11.2.6
react-datepicker	^8.2.1
lodash.debounce	^4.0.8
html2canvas	^1.4.1
browser-image-compression	^2.0.2
file-saver	^2.0.5

three / @react-three/fiber / drei	latest
-----------------------------------	--------

DevDependencies

라이브러리	버전
@vitejs/plugin-react	^4.3.4
eslint	^9.21.0
prettier	^3.5.3
eslint-plugin-react	^7.37.4
eslint-plugin-simple-import-sort	^12.1.1

4-3. Backend (Spring Boot)

항목	버전
Java (OpenJDK)	17
Spring Boot	3.3.9
Gradle	latest
Spring Dependency Management	1.1.7

Dependencies

라이브러리	버전
Spring Boot Starter Web	latest
Spring Boot Starter JPA	latest
QueryDSL	5.0.0 (jakarta)
MySQL Connector	8.x
Spring Cloud AWS	2.2.6.RELEASE
Swagger (SpringDoc OpenAPI)	2.3.0
Spring Boot Starter Security	latest
Spring Boot Starter OAuth2 Client	latest
WebClient (Spring WebFlux)	latest
Spring Boot Starter Mail	latest
Validation (Spring Boot Starter Validation)	latest
Thumbnailator	0.4.8
Metadata Extractor	2.18.0
Imgscalr	4.2
Lombok	latest
JUnit	latest

4-4. AI Backend (FastAPI)

항목	버전
----	----

Python	3.11 이상 권장
FastAPI	0.115.12
Uvicorn (ASGI 서버)	0.34.0
Starlette	0.46.1
Pydantic	2.11.0
SQLAlchemy	2.0.40
Redis (Python Client)	5.2.1

Dependencies

라이브러리	버전
Pydantic Core	2.33.0
Pydantic Settings	2.8.1
python-dotenv	1.1.0
python-multipart	0.0.20
Pillow	11.1.0
pillow-heif	0.22.0
aiohttp	3.11.12
PyMySQL	1.1.1
OpenAI	1.70.0
websockets	15.0.1

5. 외부 프로그램

- **FireBase**
 - 딥링크와 인증을 WebView 환경에서 안정적으로 구현하기 위해서 Android 시스템이 이 도메인과 앱이 연결되어 있다는 증거가 필요, 따라서 Firebase 이용
- **카카오**
 - OAuth
 - 카카오 디벨로퍼스에서 내 앱을 만들어 사용
 - Pay
 - 카카오페이 개발자센터에서 내 앱을 만들어 사용
- **Chat GPT**
 - FastApi에 키 등록 및 사요

6. 환경변수 형태

- **Backend**

- **application.properties**

```
spring.application.name=backend

spring.config.import=secrets.properties

logging.level.org.springframework.security=DEBUG
logging.level.org.springframework.web=DEBUG
logging.level.com.example.lettering=DEBUG

## API size
spring.servlet.multipart.max-file-size=60MB
spring.servlet.multipart.max-request-size=60MB
```

- **secrets.properties**

```
# server ip
domain.name={DomainName}

# keyring validation: apply = true, remove = false
validation.keyring.enabled=true

# pwa, web approve kakao
server.forward-headers-strategy=framework

# token
app.redirect-success-url: ${domain.name}/dear
app.redirect-fail-url: ${domain.name}/notag

# MYSQL
spring.datasource.url=jdbc:mysql://{DBip:port?schema}useSSL=false&allowPublicKeyRetrieval=true
spring.datasource.username={DBUser}
spring.datasource.password={DBUserPWD}
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto = none
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQLDialect
spring.jpa.properties.hibernate.jdbc.time_zone = Asia/Seoul

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,email
spring.security.oauth2.client.registration.kakao.redirect-uri=${domain.name}/login/oauth2callback
spring.security.oauth2.client.registration.kakao.client-name=KAKAO

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/
spring.security.oauth2.client.provider.kakao.user-info-authentication-method=header
spring.security.oauth2.client.provider.kakao.user-name-attribute=id

server.servlet.session.tracking-modes=cookie
server.servlet.session.cookie.http-only=true
server.servlet.session.cookie.secure=false
server.servlet.session.cookie.max-age=3600

# S3 DB Access
cloud.aws.s3.bucket={S3 Bucket name}
cloud.aws.credentials.access-key={AWS IAM Id}
cloud.aws.credentials.secret-key={AWS IAM Secret}
cloud.aws.region.static=ap-northeast-2

# KakaoPay
kakaopay.secret-key={Kakao Developer Secret Key}
kakaopay.api-url=https://open-api.kakaopay.com/online/v1
kakaopay.cid={Kakao Developers Cid}
kakaopay.approval-url=${domain.name}/payment/approve
kakaopay.cancel-url= ${domain.name}/payment/cancel
kakaopay.fail-url=${domain.name}/payment/fail

aes.secret-key={DecryptKey-Uuid16}

```

- React
 - .env

```

#서버 base URL
VITE_API_BASE_URL={DomainName}/api

#카카오 로그인
VITE_KAKAO_REDIRECT_URI={DomainName}/login/oauth2/code/kakao
VITE_KAKAO_REST_API_KEY=bef70af12e75fe9c6b2c17b647859897

#fast api 서버 URL
VITE_FAST_API_BASE_URL={DomainName}/ai

#websocket
VITE_WS_BASE_URL=wss://{DomainName}/ai/ws

```

- AI
 - .env

```

# app/.env
API_HOST=127.0.0.1
API_PORT=8001

```

```
# CORS
DOMAIN_URL={DomainName}
LOCAL_URL=http://localhost:5173
WS_URL=ws://localhost:5173

# Redis
REDIS_HOST=localhost
REDIS_PORT=6379
REDIS_DB=0
REDIS_PREFIX=lettering

# Database (MySQL)
DB_URL=mysql+pymysql://{DBid}:{DBpwd}@{DBIp:Port}/{DBSchema}

# 이미지 업로드 경로
UPLOAD_DIR=app/uploads

# GPT
OPENAI_API_KEY={GptApiKey}
```

7. CI/CD 구축

기본 설정

0. 기본적으로 위의 세팅을 참고하여 필요한 패키지, 프로그램을 EC에 다운로드 받았다는 가정 하에 진행

- Docker, Nginx, MySQL, Java 등

1. Docker Compose 설치

```
# 버전 지정 및 설치(2.32.4), uname -s: os이름 출력, uname -m: 하드웨어 이름 출력(x86_64 등)
sudo curl -L "https://github.com/docker/compose/releases/download/v2.32.4/docker-compose-$"

# 실행 권한 부여
sudo chmod +x /usr/local/bin/docker-compose
```

2. docker-compose.yml 구성, Docker file 디렉토리 구성

- docker-compose.yml

```
services:
  db:
```

```

container_name: db
image: mysql:8.0
ports:
  - "3306:3306"
environment:
  TZ: "Asia/Seoul"
  MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
  MYSQL_DATABASE: ${MYSQL_DATABASE}
  MYSQL_USER: ${MYSQL_USER}
  MYSQL_PASSWORD: ${MYSQL_PASSWORD}
volumes:
  - db_data:/var/lib/mysql
networks:
  - app_network
healthcheck:
  test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
  interval: 10s
  timeout: 5s
  retries: 3

backend:
  container_name: backend
  build:
    context: ../backend
    dockerfile: Dockerfile
  ports:
    - "8080:8080"
  environment:
    TZ: "Asia/Seoul"
    SPRING_DATASOURCE_URL: ${SPRING_DATASOURCE_URL}
    SPRING_DATASOURCE_USERNAME: ${MYSQL_USER}
    SPRING_DATASOURCE_PASSWORD: ${MYSQL_PASSWORD}
    KAKAO_CLIENT_ID: ${KAKAO_CLIENT_ID}
    KAKAO_CLIENT_SECRET: ${KAKAO_CLIENT_SECRET}
    SPRING_CONFIG_ADDITIONAL_LOCATION: /app/resources/secrets.properties
    ALLOWED_ORIGINS: ${ALLOWED_ORIGINS}
    DOMAIN_NAME: ${DOMAIN_NAME}
  volumes:
    - ../backend/secrets.properties:/app/resources/secrets.properties
  depends_on:
    db:
      condition: service_healthy
  networks:
    - app_network

nginx:
  build:
    context: ../frontend

```



```

    dockerfile: Dockerfile
    container_name: nginx
    ports:
      - "443:443"
    environment:
      TZ: "Asia/Seoul"
    volumes:
      - /home/ubuntu/nginx/conf.d/default.conf:/etc/nginx/conf.d/default.conf
      - /home/ubuntu/nginx/letsencrypt:/etc/nginx/letsencrypt
    depends_on:
      - backend
    networks:
      - app_network

volumes:
  db_data:
    external: true
    name: s12p21a203_db_data

networks:
  app_network:
    driver: bridge

```

- docker-compose.fastapi.yml

```
cd /home/ubuntu/fastapi/S12P21A203
```

```

version: "3.8"

services:
  redis:
    image: redis:latest
    container_name: redis
    ports:
      - "6379:6379"
    environment:
      TZ: "Asia/Seoul"
    command: redis-server
    networks:
      - infra_app_network
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 10s
      retries: 5
      start_period: 5s
      timeout: 3s

```

```

fastapi:
  build:
    context: ./ai
    dockerfile: Dockerfile
  container_name: fastapi
  ports:
    - "8001:8001"
  environment:
    TZ: "Asia/Seoul"
    API_HOST: 0.0.0.0
    API_PORT: 8001
    REDIS_HOST: redis
    REDIS_PORT: 6379
    REDIS_DB: 0
    REDIS_PREFIX: lettering
    DB_URL: mysql+pymysql://{DBId}:{DBpwd}@43.201.84.89:3306/{SchemaName}
    UPLOAD_DIR: uploads
  depends_on:
    redis:
      condition: service_healthy
  networks:
    - infra_app_network

networks:
  infra_app_network:
    external: true

```

- Dockerfile

```
cd /fastapi/S12P21A203/ai
```


```
FROM python:3.10-slim
```

```
WORKDIR /app
```

```
COPY . /app
```

```
RUN pip install --no-cache-dir --upgrade pip \
&& pip install --no-cache-dir -r requirements.txt
```

```
COPY .env.docker .env
```

```
#  uploads 디렉토리 생성
```

```
RUN mkdir -p app/uploads
```

```
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8001"]
```

- Dockefile

- backend

```
FROM openjdk:17-jdk-slim

WORKDIR /app

COPY build/libs/lettering-0.0.1-SNAPSHOT.jar /app/app.jar

COPY src/main/resources/secrets.properties /app/secrets.properties

EXPOSE 8080
ENTRYPOINT ["java", "-Dspring.config.additional-location=/app/secrets.properties", "-jar"]
```

- frontend

```
FROM nginx:latest

COPY dist/ /usr/share/nginx/html/

RUN chmod -R 755 /usr/share/nginx/html
```

3. Jenkins 설치 및 구성

- docker-compose-jenkins.yml

```
cd jenkins
```

```
services:
  jenkins:
    image: jenkins/jenkins:its
    container_name: jenkins
    ports:
      - "8081:8080"
    environment:
      TZ: "Asia/Seoul"
    volumes:
      - jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock # 🔥 추가
    user: root # 🔥 권한 문제 방지
    restart: always
```

```
volumes:
jenkins_home:
```

- Jenkinsfile

```
pipeline {
  agent any

  environment {
    COMPOSE_PROJECT_DIR = "${WORKSPACE}/infra"
  }

  stages {
    stage('Git 변경 확인') {
      steps {
        script {
          def changedFiles = sh(script: "git diff --name-only HEAD~1", returnStdout: true).trim()

          // 기본 변경 감지
          env.DEFAULT_CONF_CHANGED = changedFiles.contains("infra/nginx/conf.d/default.conf")
          env.FRONTEND_CHANGED = changedFiles.contains("frontend/") ? "true" : "false"
          env.BACKEND_CHANGED = changedFiles.contains("backend/") ? "true" : "false"
          env.FASTAPI_CHANGED = changedFiles.contains("ai/") ? "true" : "false"

          // Jenkinsfile 변경 시 전체 빌드 강제
          if (changedFiles.contains("infra/Jenkinsfile")) {
            echo "🔧 Jenkinsfile 변경 감지 → 전체 서비스 재빌드 강제 실행한다."
            env.DEFAULT_CONF_CHANGED = "true"
            env.FRONTEND_CHANGED = "true"
            env.BACKEND_CHANGED = "true"
            env.FASTAPI_CHANGED = "true"
          }

          echo "🔍 변경된 파일 목록:\n${changedFiles}"
          echo "🟡 Nginx default.conf 변경됨: ${env.DEFAULT_CONF_CHANGED}"
          echo "🍷 프론트엔드 변경됨: ${env.FRONTEND_CHANGED}"
          echo "🔧 백엔드 변경됨: ${env.BACKEND_CHANGED}"
          echo "⚡ FastAPI 변경됨: ${env.FASTAPI_CHANGED}"
        }
      }
    }

    stage('Secret 파일 다운로드') {
      steps {
        withCredentials([
          file(credentialsId: 'env-docker-compose', variable: 'ENV_FILE'),
        ])
      }
    }
  }
}
```

```

    file(credentialsId: 'spring-secrets', variable: 'SECRETS_FILE'),
    file(credentialsId: 'env-frontend', variable: 'FRONTEND_ENV_FILE')
  }) {
    sh '''
      cp "$ENV_FILE" infra/.env
      cp "$SECRETS_FILE" backend/src/main/resources/secrets.properties
      cp "$FRONTEND_ENV_FILE" frontend/.env
    '''
  }
}
}

stage('백엔드 JAR 빌드') {
  when {
    expression { env.BACKEND_CHANGED == "true" }
  }
  steps {
    echo "⚙️ 백엔드 Gradle 빌드 시작"
    dir('backend') {
      sh './gradlew build -x test --no-daemon'
    }
  }
}

stage('프론트엔드 빌드') {
  when {
    expression { env.FRONTEND_CHANGED == "true" }
  }
  steps {
    echo "🔧 프론트엔드 빌드 시작"
    dir('frontend') {
      sh '''
        echo "🧹 dist, node_modules 정리"
        rm -rf dist node_modules package-lock.json

        echo "📦 의존성 설치"
        export PATH=/usr/local/lib/nodejs/node-v22.13.1/bin:$PATH
        npm install --legacy-peer-deps || exit 1

        echo "🛠️ Vite 빌드"
        npm run build || {
          echo "❌ 빌드 실패"
          exit 1
        }

        [ -d dist ] || {
          echo "❌ dist 디렉토리 없음, 빌드 실패"
          exit 1
        }
      '''
    }
  }
}

```

```

    }
    ""
  }
}
}

stage('도커 이미지 빌드 및 변경 서비스 재시작') {
  steps {
    script {
      def buildFlags = ""
      if (env.DEFAULT_CONF_CHANGED == "true" || env.FASTAPI_CHANGED == "true") {
        echo "🔄 default.conf 또는 FastAPI 변경 → --no-cache 적용"
        buildFlags = "--no-cache"
      }

      // dist 없을 경우 대비
      sh '''
        if [ ! -d frontend/dist ]; then
          echo "⚠️ dist 폴더가 없어서 빈 폴더 생성"
          mkdir -p frontend/dist
        fi
      '''

      sh """
        echo "🚀 Docker 이미지 빌드"
        docker compose -f infra/docker-compose.yml build ${buildFlags}
        docker compose -f infra/docker-compose.yml up -d
      """
    }
  }
}

post {
  failure {
    echo '❌ 빌드 실패! 에러 로그를 확인해주세요!'
  }
  success {
    echo '🎉 배포 성공! Lettering 서비스가 자동으로 반영되었습니다!'
  }
}
}

```

- Jenkins에 secrets.properties 및 env 관련된 key내용 저장(Dashboard - Jenkins 관리 - Credentials)
 - Add credentials클릭시 유형 선택 가능
 - 보통은 secret text
 - key.pem과 서버 user등록은 SSH Username with private key

- secrets.properties는 secret file형태로 file 업로드

gitlab_gitops_token	[REDACTED] (ken)	Username with password
gitlab_api_token	[REDACTED]	GitLab API token
server-ip	[REDACTED]	Secret text
server-name	[REDACTED]	Secret text
server-ssh	[REDACTED]	SSH Username with private key
secret-properties	[REDACTED]	Secret file

- gitlab과 Jenkins 연결
 - gitlab에서 Jenkins에 등록할 Access Token발급(프로필-prefernece - Access tokens - Add new token)
 - Jenkins 관리 - system에 gitlab관련 내용 Key 작성
 - Jenkins 파이프라인 설정(Dashboard - item 생성)
 - 프로젝트 주소 작성
 - push event source 위치 설정
 - Credentials 등록
 - Jenkins에서 플러그인 설치 및 재시작
 - GitLab, Docker pipeline, Docker API, Genrice Webhook Trigger, SSH Agent

4. nginx 세팅

```
# HTTP 요청 → HTTPS 리다이렉트
# qusru
server {
    listen 80;
    server_name lettering.shop;

    client_max_body_size 50M;

    location /.well-known/acme-challenge/ {
        root /var/www/html;
        allow all;
    }

    return 301 https://$host$request_uri;
}

# HTTPS 서버
server {
    listen 443 ssl;
    server_name lettering.shop;

    client_max_body_size 50M;
```

```

# 🗝️ 인증서 절대 경로
ssl_certificate /etc/nginx/letsencrypt/live/letterring.shop/fullchain.pem;
ssl_certificate_key /etc/nginx/letsencrypt/live/letterring.shop/privkey.pem;

# 🛠️ 정적 파일
root /usr/share/nginx/html;
index index.html;

# 🔄 FastAPI 서버
location ^~ /ai/ {
    client_max_body_size 50M;
    proxy_pass http://fastapi:8001/ai/;

    # ✅ WebSocket을 위한 설정
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

location / {
    # 🛠️ 정적 파일
    root /usr/share/nginx/html;
    index index.html;
    try_files $uri $uri/ /index.html;
}

# 🔄 API
location /api/ {
    proxy_pass http://backend:8080;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    add_header 'Access-Control-Allow-Origin' '*' always;
    add_header 'Access-Control-Allow-Methods' 'GET, POST, PUT, DELETE, OPTIONS' always;
    add_header 'Access-Control-Allow-Headers' 'Authorization, Content-Type' always;
    add_header 'Access-Control-Allow-Credentials' 'true' always;
}

# 📄 Swagger UI
location /swagger-ui/ {
    proxy_pass http://backend:8080/swagger-ui/;
}

```



```

    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

# 📄 Swagger API 문서
location /v3/api-docs {
    proxy_pass http://backend:8080/v3/api-docs;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

location /oauth2/ {
    proxy_pass http://backend:8080;
    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 https;
}

location /login/oauth2/ {
    proxy_pass http://backend:8080;
    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 https;
}
}

```