O2O

# O2O

## 포팅 메뉴얼

# 목차

**I. 개요 ..... 2**

**1. 프로젝트 개요 ..... 2**
**2. 프로젝트 사용 도구 ..... 2**
**3. 개발환경 ..... 2**
**4. 외부 서비스 ..... 2**
**5. GITGNORE 처리한 핵심 키들 ..... 3 ~ 11**

**II. 빌드 ..... 11**

**1. 환경변수 형태 ..... 11~13**
**2. 빌드하기 ..... 13~17**
**3. 배포하기 ..... 18~19**
**4. 서비스 이용 방법 ..... 20**

# I. 개요

## 1. 프로젝트 개요

여러분들은 물품을 대여 해보신적이 있나요? 대여 담당자에게 방문, 대여 물품 관련 서류 작성, 물품 확인, 사용, 반납을 위한 담당자 방문, 서류작성, 물품 확인, 반납 완료...
담당자 측면에서 확인해볼까요 ? 새로운 물품을 등록하는일, 사용자들의 요구사항(물품 구매 요청)등....

이렇게 복잡하고 번거로운 과정과 비효율적이다 라는 생각이 들지 않나요 ?
O2O 서비스를 통해 기업 및 개인 사용자가 필요할 때 쉽게 용품을 대여하고 관리까지 빠르고 간편하게 진행합니다!

## 2. 프로젝트 사용 도구

이슈 관리 : JIRA
형상 관리 : Gitlab
커뮤니케이션 : Notion, Mattermost
디자인 : Figma
UCC : 모바비(movavi)
CI/CD : Jenkins

## 3. 개발환경

VS Code : 1.64.2,
IntelliJ : 11.0.13+7-b1751.21 amd64
JVM : 17.0.1 (스프링 17 로 빌드)
Node.js : 20.01.0
SERVER : AWS EC2 Ubuntu 20.04.3 LTS
DB : MariaDB (azure), redis,

## 4. 외부 서비스

X

## 5. Gitgnore 처리한 핵심 키들

* FE 폴더 .ignore 파일

```
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
lerna-debug.log*
.pnpm-debug.log*

# Diagnostic reports (https://nodejs.org/api/report.html)
report.[0-9]*.[0-9]*.[0-9]*.[0-9]*.json

# Runtime data
pids
*.pid
*.seed
*.pid.lock

# Directory for instrumented libs generated by jscoverage/JSCover
lib-cov

# Coverage directory used by tools like istanbul
coverage
*.lcov

# nyc test coverage
.nyc_output

# Grunt intermediate storage (https://gruntjs.com/creating-plugins#storing-task-files)
.grunt

# Bower dependency directory (https://bower.io/)
bower_components

# node-waf configuration
.lock-wscript

# Compiled binary addons (https://nodejs.org/api/addons.html)
build/Release

# Dependency directories
node_modules/
jspm_packages/

# Snowpack dependency directory (https://snowpack.dev/)
web_modules/
```

```
# TypeScript cache
*.tsbuildinfo

# Optional npm cache directory
.npm

# Optional eslint cache
.eslintcache

# Optional stylelint cache
.stylelintcache

# Microbundle cache
.rpt2_cache/
.rts2_cache_cjs/
.rts2_cache_es/
.rts2_cache_umd/

# Optional REPL history
.node_repl_history

# Output of 'npm pack'
*.tgz

# Yarn Integrity file
.yarn-integrity

# dotenv environment variable files
.env
.env.development.local
.env.test.local
.env.production.local
.env.local

# parcel-bundler cache (https://parceljs.org/)
.cache
.parcel-cache

# Next.js build output
.next
out

# Nuxt.js build / generate output
.nuxt
dist

# Gatsby files
.cache/
```

```
# vuepress build output
.vuepress/dist

# vuepress v2.x temp and cache directory
.temp

# Docusaurus cache and generated files
.docusaurus

# Serverless directories
.serverless/

# FuseBox cache
.fusebox/

# DynamoDB Local files
.dynamodb/

# TernJS port file
.tern-port

# Stores VSCode versions used for testing VSCode extensions
.vscode-test

# yarn v2
.yarn/cache
.yarn/unplugged
.yarn/build-state.yml
.yarn/install-state.gz
.pnp.*

### Node Patch ###
.webpack/
.svelte-kit

### react ###
.DS_*
**/*.backup.*
**/*.back.*

node_modules

*.sublime*

psd
thumb
sketch
### VisualStudioCode ###
.vscode/*
```

```
!.vscode/settings.json
!.vscode/tasks.json
!.vscode/launch.json
!.vscode/extensions.json
!.vscode/*.code-snippets

# Local History for Visual Studio Code
.history/

# Built Visual Studio Code Extensions
*.vsix

### VisualStudioCode Patch ###
# Ignore all local history of files
.history
.ionide
```

* BE

```
### Eclipse ###
.metadata
bin/
tmp/
*.tmp
*.bak
*.swp
*~.nib
local.properties
.settings/
.loadpath
.recommenders

# External tool builders
.externalToolBuilders/

# Locally stored "Eclipse launch configurations"
*.launch

# PyDev specific (Python IDE for Eclipse)
*.pydevproject

# CDT-specific (C/C++ Development Tooling)
.cproject

# CDT- autotools
.autotools
```

```
# Java annotation processor (APT)
.factorypath

# PDT-specific (PHP Development Tools)
.buildpath

# sbteclipse plugin
.target

# Tern plugin
.tern-project

# TeXlipse plugin
.texlipse

# STS (Spring Tool Suite)
.springBeans

# Code Recommenders
.recommenders/

# Annotation Processing
.apt_generated/
.apt_generated_test/

# Scala IDE specific (Scala & Java development for Eclipse)
.cache-main
.scala_dependencies
.worksheet

### Eclipse Patch ###
# Spring Boot Tooling
.sts4-cache/

### Git ###
# Created by git for backups. To disable backups in Git:
*.orig

# Created by git when using merge tools for conflicts
*.BACKUP.*
*.BASE.*
*.LOCAL.*
*.REMOTE.*
*_BACKUP_*.txt
*_BASE_*.txt
*_LOCAL_*.txt
*_REMOTE_*.txt
```

```
### Intellij ###

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml

# Generated files
.idea/**/contentModel.xml

# Sensitive or high-churn files
.idea/**/dataSources/
.idea/**/dataSources.ids
.idea/**/dataSources.local.xml
.idea/**/sqlDataSources.xml
.idea/**/dynamic.xml
.idea/**/uiDesigner.xml
.idea/**/dbnavigator.xml

# Gradle
.idea/**/gradle.xml
.idea/**/libraries

# CMake
cmake-build-*/

# Mongo Explorer plugin
.idea/**/mongoSettings.xml

# File-based project format
*.iws

# IntelliJ
out/

# mpeltonen/sbt-idea plugin
.idea_modules/

# JIRA plugin
atlassian-ide-plugin.xml

# Cursive Clojure plugin
.idea/replstate.xml
```

```
# SonarLint plugin
.idea/sonarlint/

# Crashlytics plugin (for Android Studio and IntelliJ)
com_crashlytics_export_strings.xml
crashlytics.properties
crashlytics-build.properties
fabric.properties

# Editor-based Rest Client
.idea/httpRequests

# Android studio 3.1+ serialized cache file
.idea/caches/build_file_checksums.ser
.idea/**/sonarlint/

# SonarQube Plugin
# https://plugins.jetbrains.com/plugin/7238-sonarqube-community-plugin
.idea/**/sonarIssues.xml

# Markdown Navigator plugin
# https://plugins.jetbrains.com/plugin/7896-markdown-navigator-enhanced
.idea/**/markdown-navigator.xml
.idea/**/markdown-navigator-enh.xml
.idea/**/markdown-navigator/

# Cache file creation bug
# See https://youtrack.jetbrains.com/issue/JBR-2257
.idea/$CACHE_FILE$

# CodeStream plugin
# https://plugins.jetbrains.com/plugin/12206-codestream
.idea/codestream.xml

# Azure Toolkit for IntelliJ plugin
# https://plugins.jetbrains.com/plugin/8053-azure-toolkit-for-intellij
.idea/**/azureSettings.xml
.idea/*

!.idea/codeStyles
!.idea/runConfigurations

### Java ###
# Compiled class file
*.class

# Log file
*.log
```

```
# BlueJ files
*.ctxt

# Mobile Tools for Java (J2ME)
.mtj.tmp/

# Package Files #
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar

# virtual machine crash logs, see http://www.java.com/en/download/help/error_hotspot.xml
hs_err_pid*
replay_pid*

### Maven ###
target/
pom.xml.tag
pom.xml.releaseBackup
pom.xml.versionsBackup
pom.xml.next
release.properties
dependency-reduced-pom.xml
buildNumber.properties
.mvn/timing.properties
# https://github.com/takari/maven-wrapper#usage-without-binary-jar
.mvn/wrapper/maven-wrapper.jar

# Eclipse m2e generated files
# Eclipse Core
.project
# JDT-specific (Eclipse Java Development Tools)
.classpath

### VisualStudioCode ###
.vscode/*
!.vscode/settings.json
!.vscode/tasks.json
!.vscode/launch.json
!.vscode/extensions.json
!.vscode/*.code-snippets

# Local History for Visual Studio Code
.history/
```

```
# Built Visual Studio Code Extensions
*.vsix

### VisualStudioCode Patch ###
# Ignore all local history of files
.history
.ionide

### Gradle ###
.gradle
**/build/
!src/**/build/

# Ignore Gradle GUI config
gradle-app.setting

# Avoid ignoring Gradle wrapper jar file (.jar files are usually ignored)
!gradle-wrapper.jar

# Avoid ignore Gradle wrappper properties
!gradle-wrapper.properties

# Cache of project
.gradletasknamecache

# Eclipse Gradle plugin generated files
# Eclipse Core
# JDT-specific (Eclipse Java Development Tools)

### Gradle Patch ###
# Java heap dump
*.hprof

# End of https://www.toptal.com/developers/gitignore/api/java,gradle,maven,eclipse,intellij+all,intellij,visualstudiocode,git
```

# II. 빌드

## 1. 환경변수 형태

**.application.properties**

spring.application.name=application.이름

server.port=서버포트

spring.profiles.include=aws, redis

spring.jpa.hibernate.ddl-auto=DB 속성

spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQLDialect
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.use_sql_comments=true

spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB

spring.mvc.static-path-pattern=/static/**

**.application-aws.properties** (비밀번호, 암호화가 필요한 구조 )

spring.datasource.password=서버 password

#jwt 토큰 secretekey
jwt.secret.key=~~

#레디스 비밀번호 및 host, port 번호
spring.data.redis.password=비밀번호
spring.data.redis.host=host 주소
spring.data.redis.port=포트번호

```
# nginx images path
upload.path.emp=./nginx/html/images/employees/
upload.path.products=./nginx/html/images/products/
upload.path.empTemp =./nginx/html/images/empTemp/
upload.path.report.products=./nginx/html/images/report/products/
```

## 2. 빌드하기

* docker-compose.yml 파일 이용

```
services:
  kisok:
    build:
      context: ./FE/kiosk
    image: react_container  # 이미지 이름 설정
    container_name: react_container  # 컨테이너 이름 설정
    ports:
      - "3000:3000"  # kiosk React 앱 포트
    networks:
      - my_network #나만의 network 설정!

  web:
    build:
      context: ./FE/web
    image: react_container_web  # 이미지 이름 설정
    container_name: react_container_web  # 컨테이너 이름 설정
    ports:
       - "3001:3001"  # web React 앱 포트
    networks:
      - my_network #나만의 network 설정!

  spring-boot-app:
    build:
      context: ./BE/o2o
    image: tem  # 이미지 이름 설정
```

```
    container_name: my_container  # 컨테이너 이름 설정
    ports:
      - "8000:8000"  # 스프링 부트 포트
    networks:
      - my_network #나만의 network 설정!
  nginx:
    build:
      context: ./nginx  # Dockerfile 이 위치한 경로
    image: nginx:alpine
    container_name: nginx_container  # 컨테이너 이름 설정
    ports:
      - "80:80"
      - "443:443"  # HTTPS 포트


    #volumes:

      #-
/etc/letsencrypt/live/i11d101.p.ssafy.io/fullchain.pem:/etc/letsencrypt/live/i11d101.p.ssafy.i
o/fullchain.pem  # SSL 인증서 경로
      #-
/etc/letsencrypt/live/i11d101.p.ssafy.io/privkey.pem:/etc/letsencrypt/live/i11d101.p.ssafy.io
/privkey.pem  # SSL 개인 키 경로

    depends_on:
      - kisok
      - web
      - spring-boot-app
    networks:
      - my_network #나만의 network 설정!

networks:
  my_network: #네트워크 정의!
```

* BE 폴더내 Dockerfile

```
FROM openjdk:17-jdk-alpine

# 작업 디렉토리 설정
WORKDIR /app

COPY . .


RUN ls -al

RUN pwd
# Gradle 빌드 함!
#RUN rm -rf .gradle


RUN chmod +x ./gradlew && ./gradlew clean build
RUN ls -al ./build/libs/

#ARG JAR_FILE=o2o-0.0.1-SNAPSHOT.jar

ARG JAR_FILE=o2o-0.0.1-SNAPSHOT.jar
RUN mv ./build/libs/${JAR_FILE} ./build/libs/app.jar
RUN ls -al ./build/libs/

RUN cp ./build/libs/app.jar ./app.jar

#ENV TZ=Asia/Seoul
ENTRYPOINT ["java", "-jar", "./app.jar"]
```

* FE 폴더 내 Dockerfile (kiosk)

```
# 가져올 이미지를 정의
FROM node:20 AS builder
# 경로 설정하기
WORKDIR /app
# package.json 워킹 디렉토리에 복사 (.은 설정한 워킹 디렉토리를 뜻함)
#RUN ls -al
COPY package.json package-lock.json ./
# 명령어 실행 (의존성 설치)
RUN ls -al
RUN npm install
RUN ls -al
COPY . .
RUN npm run build

# /app/build 의 내용 확인
RUN ls -al /app/build

FROM nginx:alpine
# Nginx 가 수신 대기할 포트 노출
#EXPOSE 80
# 필요에 따라 Node.js 애플리케이션의 포트를 노출
#EXPOSE 3000
RUN ls -al


COPY --from=builder /app/build /usr/share/nginx/html
#RUN ls -al /usr/share/nginx/html/kiosk

#RUN ls -al /usr/share/nginx/html
CMD ["nginx", "-g","daemon off;"]
```

* FE 폴더 내 Dockerfile (web)

```
# 가져올 이미지를 정의
FROM node:20 AS builder
# 경로 설정하기
WORKDIR /app
# package.json 워킹 디렉토리에 복사 (.은 설정한 워킹 디렉토리를 뜻함)
#RUN ls -al
COPY package.json package-lock.json ./
# 명령어 실행 (의존성 설치)
#RUN ls -al
RUN npm install
#RUN ls -al
COPY . .
#RUN ls -al
RUN npm run build

# /app/build 의 내용 확인
#RUN ls -al /app/build

FROM nginx:alpine
# Nginx 가 수신 대기할 포트 노출
#EXPOSE 80
# 필요에 따라 Node.js 애플리케이션의 포트를 노출
#EXPOSE 3001
#RUN ls -al
COPY --from=builder /app/build /usr/share/nginx/html

#RUN ls -al /usr/share/nginx/html/web

#RUN ls -al /usr/share/nginx/html

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

### 3. 배포하기

Nginx 설정

```
upstream backend {
        #server i11d101.p.ssafy.io:8000;  # 스프링 부트 백엔드
        server spring-boot-app:8000;  # 스프링 부트 백엔드의 컨테이너 이름
    }

    server {
        listen 80;
        server_name i11d101.p.ssafy.io;

        # CORS 헤더 추가
        add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
        add_header 'Access-Control-Allow-Headers' 'Content-Type, Authorization';

        # /api 경로는 API 서비스
        location /ty/ty2/ {
            proxy_pass http://backend/;   # 스프링 부트 백엔드로 프록시
            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;
        }

        # 잘못된 요청에 대한 처리
        location / {
            return 404;  # 잘못된 요청에 대한 처리
        }
    }


server {
    listen 3000;
    server_name i11d101.p.ssafy.io;
```

```
    root /usr/share/nginx/html;
    location / {
        index index.html;
         try_files $uri $uri/ /index.html;
    }

    location ~ .(static)/(jslcsslmedia)/(.+)$ {
         try_files $uri $uri/ /$1/$2/$3;

    }

}


server {
    listen 3001;
    server_name i11d101.p.ssafy.io;

    root /usr/share/nginx/html;
    location / {
        index index.html;
        try_files $uri $uri/ /index.html;
    }

    location ~ .(static)/(jslcsslmedia)/(.+)$ {
         try_files $uri $uri/ /$1/$2/$3;

    }

}
```

이후 sudo service nginx start

## 4. 서비스 이용하기

- Jenkins 를 통한 aws 서버내 도커 컨테이너 build
- IOT 기기(Orin Jetson 내 fastapi 서버 구축 및 백그라운드 실행)
- 사용자는 웹 페이지를 통한 물품 현황 확인, 예약, 물품 신청진행
- 관리자는 웹 페이지를 통해 물품 현황 확인, 연체, 파손 물품 확인
- 키오스크를 통한 물품 대출, 반납 진행