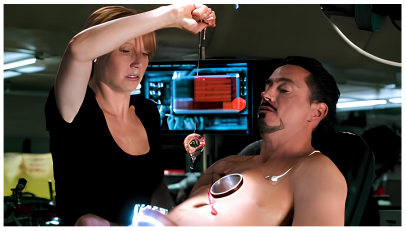

폴튕 메뉴월

1. 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost
- 디자인 : Figma
- CI/CD : Jenkins
- 모니터링 : Grafana v10.4.2(node-exporter, prometheus, cadvisor)

2. 개발 도구

- Visual Studio Code : 1.85.1
- Intellij : 2022.3.2 (Ultimate Edition)
- MobaXTerm
- Docker Desktop
- DBeaver 24.0.2
- Arduino IDE 2.3.2
- Autodesk Fusion 2.0.18961
- Autodest Inventor 2025

3. 개발 환경

- 상세 내용

Frontend

Node.js	`20.10.0`
npm	`10.2.3`
vite	`4.2.1`
type-script	`5.2.2`
React	`18.2.0`
Axios	`1.6.8`
zustand	`3.7.0`
tanstack/react-query	`5.29.2`

Backend

Java

Java	`azul-17 Azul Zulu version 17.0.11`
Spring Boot	`3.2.4`
gradle	`gradle-8.7`

Python

Python	`3.7.5`
paho-mqtt	`1.6.1`
PyAutoGUI	`0.9.54`
pyinstaller	`5.13.2`
pyinstaller-hooks-contrib	`2024.5`

Server

AWS S3	
AWS EC2	CPU : Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz /dev/root : 311G
RAM	15GB
OS	Ubuntu

Service

MariaDB	`10.3.23`

NginX	`1.18.0`
Jenkins	`2.449`
Docker	`26.1.1`
Ubuntu	`Ubuntu 20.04 LTS`

4. 환경변수 형태

- Frontend
  - package.json

```
{
  "name": "package",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
    "lint": "eslint . --ext ts,tsx --report-unused-di
sable-directives --max-warnings 0",
    "preview": "vite preview"
  },
  "dependencies": {
    "@emotion/react": "^11.11.4",
    "@emotion/styled": "^11.11.0",
    "@mui/icons-material": "^5.15.11",
    "@mui/material": "^5.15.12",
    "@types/d3": "^7.4.3",
    "@types/react-joyride": "^2.0.5",
    "@types/react-modal": "^3.16.3",
    "@types/swiper": "^6.0.0",
    "axios": "^1.6.8",
    "base-64": "^1.0.0",
    "d3": "^7.9.0",
    "jotai": "^2.7.1",
    "js-confetti": "^0.12.0",
    "query-string": "^9.0.0",
```

```
    "react": "^18.2.0",
    "react-cookie": "^7.1.0",
    "react-dom": "^18.2.0",
    "react-financial-charts": "^2.0.1",
    "react-joyride": "^2.8.0",
    "react-modal": "^3.16.1",
    "react-query": "^3.39.3",
    "react-router-dom": "^6.22.2",
    "react-spinners": "^0.13.8",
    "styled-components": "^6.1.8",
    "swiper": "^11.0.7"
  },
  "devDependencies": {
    "@types/base-64": "^1.0.2",
    "@types/node": "^20.11.30",
    "@types/react": "^18.2.56",
    "@types/react-dom": "^18.2.19",
    "@types/react-native-dotenv": "^0.2.2",
    "@typescript-eslint/eslint-plugin": "^7.0.2",
    "@typescript-eslint/parser": "^7.0.2",
    "@vitejs/plugin-react-swc": "^3.6.0",
    "dotenv": "^16.4.5",
    "eslint": "^8.56.0",
    "eslint-config-prettier": "^9.1.0",
    "eslint-plugin-prettier": "^5.1.3",
    "eslint-plugin-react-hooks": "^4.6.0",
    "eslint-plugin-react-refresh": "^0.4.5",
    "prettier": "^3.2.5",
    "react-native-dotenv": "^3.4.11",
    "typescript": "^5.2.2",
    "vite": "^5.1.4",
    "vite-tsconfig-paths": "^4.3.1"
  }
}
```

- Backend

- build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.2.4'
    id 'io.spring.dependency-management' version '1.1.4
    id "org.sonarqube" version "4.3.1.3277"
    id 'jacoco'
}

group = 'com.ssafy'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

sonarqube {
    properties {
        property "sonar.projectKey", "stellAR"
        property "sonar.host.url", "https://sonarqube.s
        property "sonar.login", {login_key}
        property "sonar.coverage.jacoco.xmlReportPaths"
        property "sonar.exclusions", "**/*Application*,
    }
}

dependencies {
```

```
    implementation 'org.springframework.boot:spring-boo
    implementation 'org.springframework.boot:spring-boo
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-bo
    runtimeOnly 'org.mariadb.jdbc:mariadb-java-client'
    annotationProcessor 'org.projectlombok:lombok'

    implementation 'org.springframework.boot:spring-boo
    implementation 'org.springframework.boot:spring-boo
    implementation 'org.apache.httpcomponents:httpclien
    implementation 'org.springdoc:springdoc-openapi-sta
    implementation 'com.google.code.gson:gson:2.8.9'
    implementation 'com.google.firebase:firebase-admin:
    implementation 'com.squareup.okhttp3:okhttp:4.9.3'
    implementation 'com.fasterxml.jackson.dataformat:ja

    testImplementation 'org.springframework.boot:spring
    testImplementation 'org.springframework.boot:spring
    testImplementation 'org.springframework.security:sp
    testImplementation 'com.h2database:h2'
}

tasks.named('test') {
    finalizedBy 'jacocoTestReport'
    useJUnitPlatform()
}

jacocoTestReport {
    dependsOn test
    reports {
        xml.required.set(true)
        xml.outputLocation.set(layout.buildDirectory.fi
        html.required.set(true)
        csv.required.set(false)
    }
    afterEvaluate {
        classDirectories.setFrom(
                files(classDirectories.files.collect {
```

```
                    fileTree(dir: it, excludes: [
                            "**/*Application*",
                            "**/settings/**",
                            "**/dto/**",
                            "**/utils/**"
                    ])
                })
        )
    }
}
```

- .env

```
SPRING_DATASOURCE_URL={MARIADB_URL}
SPRING_DATASOURCE_USERNAME={DB_USERNAME}
SPRING_DATASOURCE_PASSWORD={DB_PASSWORD}

kakaopay_admin-key={KAKAO_ADMIN_KEY}
kakaopay_cid=TC0ONETIME
kakaopay_ready-url=https://open-api.kakaopay.com/online/v1/payment/ready
kakaopay_aproval-url=https://open-api.kakaopay.com/online/v1/payment/approve HTTP/1.1

FIREBASE_API_KEY={FIREBASE_API_KEY}
```

5. CI/CD 구축

EC2 환경설정

EC2 인스턴스에 설치한 것

- Docker
- Jenkins

Docker

- Frontend

```
# Frontend 이미지 pull
$ docker pull habu2710/stellar-frontend:latest

# Frontend 컨테이너 서비스 실행
$ docker run -d --name Frontend -p 5173:5173 habu2710/stel
```

- Backend

```
# Backend 이미지 pull
$ docker pull habu2710/stellar-backend:latest

# Backend 컨테이너 서비스 실행 .env 파일을 해당경로에 저장해두고 실행
# .env는 상단의 Backend .env
$ docker run -d --name Backend -p 8080:8080 --env-file /ho
```

- mqtt-broker

```
# mqtt-broker 이미지 pull
$ sudo docker pull eclipse/mosquitto

# mqtt-broker 컨테이너 서비스 실행
$ sudo docker run -p 1883:1883 --name eclipse-mosquitto
```

- grafana

```
# 포트 열어주고
# 프로메테우스 포트 9090
sudo ufw allow 9090
# Node Exporter 포트 9100
sudo ufw allow 9100
# 그라파나 포트 3000
sudo ufw allow 3000
```

```
# 프로메테우스 설정 파일
# sudo vi /tmp/prometheus.yml
global:
  scrape_interval: 15s
  evaluation_interval: 15s

scrape_configs:
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['ec2서버url:9100']

# 프로메테우스 설치 및 마운트
sudo docker run -d --name prometheus -p 9090:9090 -v /tmp/



# Node Exporter 설치
sudo docker pull prom/node-exporter
sudo docker run -d --name node_exporter -p 9100:9100 prom/




# 그라파나 설치
sudo docker run -d --name grafana -p 3000:3000 grafana/gra
# 초기 비밀번호 admin / admin
# http://ec2서버url:3000 에 접속
# DataSource에 프로메테우스 추가
# Data Sources 설정 중 URL 에 http://<ec2 서버 url>:9090 을 적
# New Dashboad & Add new Panel
```

- Jenkins

```
# Jenkins 전용 포트 개방
$ sudo ufw allow 20001

# Jenkins 컨테이너 실행
$ sudo docker run -d -p 20001:8080 --name jenkins -v /home
```

```
# 젠킨스 환경설정
$ cd /home/ubuntu/jenkins-data

$ mkdir update-center-rootCAs

$ wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-c
update-center.crt \
  -O ./update-center-rootCAs/update-center.crt

$ sudo sed -i \
  's#https://updates.jenkins.io/update-center.json'\
  '#https://raw.githubusercontent.com/' \
  'lework/jenkins-update-center/master/updates/tencent/upd
  ./hudson.model.UpdateCenter.xml

# 젠킨스 이미지 안에 도커 설치
$ docker exec -it -u root jenkins bash
$ apt-get update
$ apt-get install -y apt-transport-https ca-certificates c
$ curl -fsSL https://download.docker.com/linux/debian/gpg
$ add-apt-repository "deb [arch=amd64] https://download.do

$ apt-get update
$ apt-get install -y docker-ce-cli

$ apt-get update && apt-get install -y vim

$ docker --version

$ sudo docker restart jenkins
```

Docker File

- Frontend

```
# 가져올 이미지를 정의
FROM node:20
# 경로 설정하기
WORKDIR /app
# package.json 워킹 디렉토리에 복사 (.은 설정한 워킹 디렉토리를
COPY package.json .
# 명령어 실행 (의존성 설치)
RUN npm install
# 현재 디렉토리의 모든 파일을 도커 컨테이너의 워킹 디렉토리에 복사한
COPY . .

# 5173번 포트 노출
EXPOSE 5173

# npm start 스크립트 실행
CMD ["npm", "run" , "dev"]
```

- Backend

```
FROM gradle:8.5-jdk17 AS builder
WORKDIR /build

# 그래들 파일이 변경되었을 때만 새롭게 의존패키지 다운로드 받게함.
COPY /build.gradle /settings.gradle /build/
RUN gradle build -x test --parallel --continue > /dev/n

# 빌더 이미지에서 애플리케이션 빌드
COPY ./ /build
RUN gradle build -x test --parallel

# APP
FROM openjdk:17
WORKDIR /app

# 빌더 이미지에서 jar 파일만 복사
# COPY --from=builder /build/build/libs/*-SNAPSHOT.jar
```

```
COPY --from=builder /build/build/libs/StellAR-0.0.1-SNA

EXPOSE 8080

# 로그 파일 폴더 만들어주고, 사용권한 넣기
RUN mkdir -p /app/logs && chown nobody:nobody /app/logs

# root 대신 nobody 권한으로 실행
USER nobody
ENTRYPOINT [
   "java",
   "-jar",
   "StellAR-0.0.1-SNAPSHOT.jar"             \
]
```

Jenkins File

```
pipeline {
    agent any

    stages {

        stage('rm directory') {
            steps {
                sh "echo rm directory"
                sh "ls -al"
                sh "pwd"
                sh "rm -rf *"
                sh "echo rm directory2"
                sh "ls -al"
            }
        }

        stage('Checkout') {
            steps {
                // 소스 코드를 클론하는 Git 명령어
```

```
                sh "ls -al"
                git branch: 'release2', credentialsId:
'GIT_CREDENTIALS_ID', url: 'https://lab.ssafy.com/s10-fi
nal/S10P31C105.git'
                sh "ls -al"
            }
        }

        stage('Build and Push Docker Images') {
            steps {
                script {
                    // 각 프로젝트 폴더에서 Docker 이미지를 빌
드하고 Docker Hub에 푸시
                    sh "pwd"
                    sh "ls -al"
                    buildAndPushImage('frontend')
                    sh "docker image ls"
                    buildAndPushImage('BackEnd')
                    sh "docker image ls"
                }
            }
        }

        stage('Deploy with Docker') {
            steps {
                script {
                    // 1. 기존 작동중인 컨테이너 삭제 및 중지
                    sh 'docker stop Frontend Backend ||
true'
                    sh 'docker rm Frontend Backend || tr
ue'

                    // 2. 기존 이미지를 Docker Hub에서 최신
버전으로 갱신
                    sh 'docker pull habu2710/stellar-fro
ntend:latest'
                    sh 'docker pull habu2710/stellar-bac
kend:latest'
```

```
                    // 3. 새로 갱신된 이미지를 기반으로 컨테이
너 실행
                    sh 'docker run -d --name Frontend -p
5173:5173 habu2710/stellar-frontend:latest'
                    sh 'docker run -d --name Backend -p
8080:8080 --env-file /.env habu2710/stellar-backend:late
st'
                }
            }
        }
    }
}

def buildAndPushImage(projectName) {
    def lowercaseProjectName = projectName.toLowerCase()
    if("${projectName}" == "BackEnd") {
        dir("${projectName}/StellAR") {
            sh "ls -al"
            sh "docker build -t habu2710/stellar-${lower
caseProjectName}:latest ."
            withCredentials([usernamePassword(credential
sId: 'DOCKER_HUB', passwordVariable: 'DOCKER_PASSWORD',
usernameVariable: 'DOCKER_ID')]) {
                sh "docker login -u $DOCKER_ID -p $DOCKE
R_PASSWORD"
                sh "docker push habu2710/stellar-${lower
caseProjectName}:latest"
            }
        }
    } else {
        dir("${projectName}") {
            sh "docker build -t habu2710/stellar-${lower
caseProjectName}:latest ."
            withCredentials([usernamePassword(credential
sId: 'DOCKER_HUB', passwordVariable: 'DOCKER_PASSWORD',
usernameVariable: 'DOCKER_ID')]) {
                sh "docker login -u $DOCKER_ID -p $DOCKE
```

```
R_PASSWORD"
                sh "docker push habu2710/stellar-${lower
caseProjectName}:latest"
            }
        }
    }

}
```

Nginx site-available/default

```
server {
    listen 80 ;
    server_name k10c105.p.ssafy.io;
#    server_name shining-stellar.kro.kr;
    return 301 https://$host$request_uri;


#    if ($host = k10c105.p.ssafy.io) {
#        return 301 https://$host$request_uri;
#    } # managed by Certbot

}

server {
    listen 443 ssl;
 #   listen [::]:443 ssl ipv6only=on;
    server_name k10c105.p.ssafy.io;
    ssl_certificate /etc/letsencrypt/live/k10c105.p.ssafy.
    ssl_certificate_key /etc/letsencrypt/live/k10c105.p.ss
    include /etc/letsencrypt/options-ssl-nginx.conf; # SSL
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # DH 파

    location /jenkins {
        proxy_pass http://k10c105.p.ssafy.io:20001; # Jenk
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
```

```
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /api {
        # proxy_pass http://backend;
        proxy_pass http://k10c105.p.ssafy.io:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_set_header X-Forwarded-Proto $scheme;

        # CORS headers
        add_header 'Access-Control-Allow-Origin' 'https://
        add_header 'Access-Control-Allow-Methods' 'GET, PO
        add_header 'Access-Control-Allow-Headers' 'DNT,Use
        add_header 'Access-Control-Expose-Headers' 'Conten
        if ($request_method = 'OPTIONS') {
        # Tell client that this pre-flight info is valid f
                add_header 'Access-Control-Allow-Origin' '
                add_header 'Access-Control-Allow-Methods'
                add_header 'Access-Control-Allow-Headers'
                add_header 'Access-Control-Max-Age' 172800
                add_header 'Content-Type' 'text/plain; cha
                add_header 'Content-Length' 0;
                return 204;
        }
    }

    location / {
       # proxy_pass http://frontend;
       proxy_pass http://k10c105.p.ssafy.io:5173;
       proxy_set_header Host $host;
       proxy_set_header X-Real-IP $remote_addr;
       proxy_set_header X-Forwarded-For $proxy_add_x_forwa
       proxy_set_header X-Forwarded-Proto $scheme;
       proxy_ssl_server_name on;
       proxy_set_header Upgrade $http_upgrade;
```

```
        proxy_set_header Connection "upgrade";
    }
}
```

6. 임베디드 마우스 제어 실행파일

실행파일 제작

사용 라이브러리 명령어 및 파일

- pyinstaller

```
# mqtt_sub_v2.py가 있는 경로로 이동
S10P31C105/Embedded/mqtt_sub_v2.py(gitLab 기준)

# 단일 파일로 제작
pyinstaller -F mqtt_sub_v2.py

# spec 파일로 단일 실행파일 제작
pyinstaller mqtt_sub_v2.spec
```

- .spec

```
# -*- mode: python ; coding: utf-8 -*-


block_cipher = None


a = Analysis(
    ['mqtt_sub_v2.py'],
    pathex=[],
    binaries=[],
    datas=[],
    # python script에 사용되는 라이브러리
    hiddenimports=['paho.mqtt.client','json','pyautogui','
```

```
    hookspath=[],
    hooksconfig={},
    runtime_hooks=[],
    excludes=[],
    win_no_prefer_redirects=False,
    win_private_assemblies=False,
    cipher=block_cipher,
    noarchive=False,
)
pyz = PYZ(a.pure, a.zipped_data, cipher=block_cipher)

exe = EXE(
    pyz,
    a.scripts,
    a.binaries,
    a.zipfiles,
    a.datas,
    [],
    # 실행파일 이름
    name='StellAR',
    debug=False,
    bootloader_ignore_signals=False,
    strip=False,
    upx=True,
    upx_exclude=[],
    runtime_tmpdir=None,
    console=True,
    disable_windowed_traceback=False,
    argv_emulation=False,
    target_arch=None,
    codesign_identity=None,
    entitlements_file=None,
    # 실행파일 아이콘 생성
    icon='favicon.ico'
)
```

7. 라즈베리파이 wifi 연결

sd 카드 conf 파일 설정을 통한 wifi 자동 연결

처음 부팅 시 사용자 지정 wifi에 자동 연결하기 위해 라즈베리파이에 포함된 sd 카드를 이용해 wifi를 설정할 수 있다.

- conf 파일 작성

1. 라즈베리파이에 동봉된 sd 카드를 리더기를 사용해 컴퓨터로 인식한다.
2. `/Boot` 파티션으로 이동한다.
3. 폴더 내 `SSH`, `wpa_supplicant.conf` 빈 파일을 생성한다.
4. 아래 내용을 작성하여 저장한다.

```
# wpa_supplicant.conf
# ssid, psk, key_mgmt를 지정한다.

country=US # Your 2-digit country code, 변경하지 않는다.
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
network={
    ssid="YOUR_NETWORK_NAME" #네트워크 이름
    psk="YOUR_PASSWORD" #네트워크 비밀번호
    key_mgmt=WPA-PSK
}
```

비밀번호가 있는 와이파이의 경우
key_mgmt=WPA-PSK

비밀번호가 없는 와이파이의 경우
key_mgmt=NONE