포팅 메뉴얼

1. 사용 도구

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

2. 개발 도구

Frontend

- 프레임워크 : Flutter
- 라이브러리
  - http: ^1.2.1
  - image_picker: ^1.1.0
  - image_cropper: ^5.0.1
  - cupertino_icons: ^1.0.6
  - dio: ^5.4.3+1
  - shared_preferences: ^2.2.3
  - flutter_secure_storage: ^5.0.2
  - flutter_native_splash: ^2.4.0
  - intl: ^0.19.0
  - flutter_colorpicker: ^1.0.3
  - flutter_riverpod: ^2.5.1
  - flutter_downloader: ^1.11.7
  - path_provider: ^2.1.3
  - mime: ^1.0.5

- firebase_core: ^2.31.0
- firebase_messaging: ^14.9.2
- flutter_local_notifications: ^17.1.2
- permission_handler: ^11.1.0
- android_id: ^0.3.6
- url_launcher: ^6.2.6
- auto_size_text: ^3.0.0
- eventflux: ^2.0.1

Backend

- IntelliJ 2024.1.1

Embedded

- 프레임워크 : Electron
- 라이브러리 :
- 언어:
- 라즈베리파이 세팅

  ▼ 라즈베리파이 requirements.txt

```
# requirements.txt
arandr==0.1.11
asgiref==3.6.0
astroid==2.14.2
asttokens==2.2.1
av==10.0.0
Babel==2.10.3
beautifulsoup4==4.11.2
blinker==1.5
certifi==2022.9.24
chardet==5.1.0
charset-normalizer==3.0.1
click==8.1.3
colorama==0.4.6
```

```
colorzero==2.0
cryptography==38.0.4
cupshelpers==1.0
dbus-python==1.3.2
dill==0.3.6
distro==1.8.0
docutils==0.19
Flask==2.2.2
gpiozero==2.0
html5lib==1.1
idna==3.3
importlib-metadata==4.12.0
isort==5.6.4
itsdangerous==2.1.2
jedi==0.18.2
Jinja2==3.1.2
lazy-object-proxy==1.9.0
lgpio==0.2.2.0
libevdev==0.5
logilab-common==1.9.8
lxml==4.9.2
MarkupSafe==2.1.2
mccabe==0.7.0
meson==1.0.1
more-itertools==8.10.0
mypy==1.0.1
mypy-extensions==0.4.3
numpy==1.24.2
oauthlib==3.2.2
olefile==0.46
parso==0.8.3
pexpect==4.8.0
pgzero==1.2
picamera2==0.3.18
pidng==4.0.9
piexif==1.1.3
pigpio==1.78
Pillow==9.4.0
```

```
platformdirs==2.6.0
psutil==5.9.4
ptyprocess==0.7.0
pycairo==1.20.1
pycryptodomex==3.11.0
pycups==2.0.1
pygame==2.1.2
Pygments==2.14.0
PyGObject==3.42.2
pyinotify==0.9.6
PyJWT==2.6.0
pylint==2.16.2
PyOpenGL==3.1.6
pyOpenSSL==23.0.0
PyQt5==5.15.9
PyQt5-sip==12.11.1
pyserial==3.5
pysmbc==1.0.23
python-apt==2.6.0
python-dotenv==0.21.0
python-prctl==1.8.1
pytz==2022.7.1
pyudev==0.24.0
reportlab==3.6.12
requests==2.28.1
requests-oauthlib==1.3.0
responses==0.18.0
roman==3.3
RPi.GPIO==0.7.1a4
RTIMULib==7.2.1
Send2Trash==1.8.1b0
sense-hat==2.6.0
simplejpeg==1.6.6
simplejson==3.18.3
six==1.16.0
smbus2==0.4.2
soupsieve==2.3.2
spidev==3.5
```

```
ssh-import-id==5.10
thonny==4.1.4
toml==0.10.2
tomlkit==0.11.7
twython==3.8.2
types-aiofiles==22.1
types-annoy==1.17
types-appdirs==1.4
types-aws-xray-sdk==2.10
types-babel==2.11
types-backports.ssl-match-hostname==3.7
types-beautifulsoup4==4.11
types-bleach==5.0
types-boto==2.49
types-braintree==4.17
types-cachetools==5.2
types-caldav==0.10
types-certifi==2021.10.8
types-cffi==1.15
types-chardet==5.0
types-chevron==0.14
types-click-spinner==0.1
types-colorama==0.4
types-commonmark==0.9
types-console-menu==0.7
types-contextvars==2.4
types-croniter==1.3
types-cryptography==3.3
types-D3DShot==0.1
types-dateparser==1.1
types-DateTimeRange==1.2
types-decorator==5.1
types-Deprecated==1.2
types-dj-database-url==1.0
types-docopt==0.6
types-docutils==0.19
types-editdistance==0.6
types-emoji==2.1
```

```
types-entrypoints==0.4
types-first==2.0
types-flake8-2020==1.7
types-flake8-bugbear==22.10.27
types-flake8-builtins==2.0
types-flake8-docstrings==1.6
types-flake8-plugin-utils==1.3
types-flake8-rst-docstrings==0.2
types-flake8-simplify==0.19
types-flake8-typing-imports==1.14
types-Flask-Cors==3.0
types-Flask-SQLAlchemy==2.5
types-fpdf2==2.5
types-gdb==12.1
types-google-cloud-ndb==1.11
types-hdbcli==2.14
types-html5lib==1.1
types-httplib2==0.21
types-humanfriendly==10.0
types-invoke==1.7
types-JACK-Client==0.5
types-jmespath==1.0
types-jsonschema==4.17
types-keyboard==0.13
types-ldap3==2.9
types-Markdown==3.4
types-mock==4.0
types-mypy-extensions==0.4
types-mysqlclient==2.1
types-oauthlib==3.2
types-openpyxl==3.0
types-opentracing==2.4
types-paho-mqtt==1.6
types-paramiko==2.11
types-parsimonious==0.10
types-passlib==1.7
types-passpy==1.0
types-peewee==3.15
```

```
types-pep8-naming==0.13
types-Pillow==9.3
types-playsound==1.3
types-polib==1.1
types-prettytable==3.4
types-protobuf==3.20
types-psutil==5.9
types-psycopg2==2.9
types-pyaudio==0.2
types-PyAutoGUI==0.9
types-pycurl==7.45
types-pyfarmhash==0.3
types-pyflakes==2.5
types-Pygments==2.13
types-pyinstaller==5.6
types-PyMySQL==1.0
types-pynput==1.7
types-pyOpenSSL==22.1
types-pyRFC3339==1.1
types-PyScreeze==0.1
types-pysftp==0.2
types-pytest-lazy-fixture==0.6
types-python-crontab==2.6
types-python-dateutil==2.8
types-python-gflags==3.1
types-python-jose==3.3
types-python-nmap==0.7
types-python-slugify==6.1
types-pytz==2022.6
types-pyvmomi==7.0
types-pywin32==304
types-PyYAML==6.0
types-redis==4.3
types-regex==2022.10.31
types-requests==2.28
types-retry==0.9
types-Send2Trash==1.8
types-setuptools==65.5
```

```
types-simplejson==3.17
types-singledispatch==3.7
types-six==1.16
types-slumber==0.7
types-SQLAlchemy==1.4.43
types-stdlib-list==0.8
types-stripe==3.5
types-tabulate==0.9
types-termcolor==1.1
types-toml==0.10
types-toposort==1.7
types-tqdm==4.64
types-tree-sitter==0.20
types-tree-sitter-languages==1.5
types-ttkthemes==3.2
types-typed-ast==1.5
types-tzlocal==4.2
types-ujson==5.5
types-urllib3==1.26
types-vobject==0.9
types-waitress==2.1
types-whatthepatch==1.0
types-xmltodict==0.13
types-xxhash==3.0
types-zxcvbn==4.4
typing_extensions==4.4.0
ufw==0.36.2
urllib3==1.26.12
v4l2-python3==0.3.3
webencodings==0.5.1
Werkzeug==2.2.2
wrapt==1.14.1
zipp==1.0.0
```

▼ 라즈베리파이 가상 환경 설정

```
python3 -m venv --system-site-packages myenv
```

3. 개발 환경

Frontend

Dart	3.3.4
Flutter	3.19.6

Backend

Java	17
Spring Boot	3.2.5
MySql	
MongoDB	

Embedded

Python	3.11.2
Javascript	
Node.js	20.13.1
Electron	28.2.0
React	18.2.0

Infra

Docker	26.0.2
Nginx	1.18.0 (Ubuntu)

4. 환경 변수

Backend

```
spring.application.name=moass-api-server
spring.profiles.active=${ACTIVE}

# R2DBC mariaDB
spring.r2dbc.url=${MARIADB_URL}
spring.r2dbc.username=${MARIADB_USERNAME}
spring.r2dbc.password=${MARIADB_PASSWORD}


# create db table
# spring.sql.init.mode=always

spring.task.scheduling.pool.size=2

spring.r2dbc.pool.initial-size=15
spring.r2dbc.pool.max-size=50
spring.r2dbc.pool.max-idle-time=30m

#MongoDB
spring.data.mongodb.uri=${MONGODB_URL}
spring.data.mongodb.database=${MONGODB_DBNAME}

spring.webflux.base-path=/api
server.port=8080



# Default log level

logging.level.root=INFO

# Spring Framework Web log level
logging.level.org.springframework.web=INFO

# Spring Data log level
logging.level.org.springframework.data=INFO

# Spring Security log level
```

```
logging.level.org.springframework.security=ERROR

# R2DBC db log level
logging.level.org.springframework.r2dbc.core.DefaultDatabaseC

# R2DBC pool log level
logging.level.io.r2dbc.pool=ERROR

# Custom application log level
logging.level.com.moass.api=INFO

logging.level.reactor.netty.http.client=INFO
# Specific packages can be further adjusted if necessary
# logging.level.your.specific.package=INFO

value.access-key=${JWT.ACCESS_KEY}
value.refresh-key=${JWT.REFRESH_KEY}

value.jira-client-id=${JIRA.CLIENT_ID}
value.jira-client-secret=${JIRA.CLIENT_SECRET}
value.jira-redirect-uri=${JIRA.REDIRECT_URI}

value.gitlab-client-id=${GITLAB.CLIENT_ID}
value.gitlab-client-secret=${GITLAB.CLIENT_SECRET}
value.gitlab-redirect-uri=${GITLAB.REDIRECT_URI}

value.mm-base-uri=${MM.BASE_URI}
value.mm-redirect-uri=${MM.REDIRECT_URI}
value.mm-webhook-uri1=${MM.WEBHOOK_URI1}
value.mm-webhook-uri2=${MM.WEBHOOK_URI2}

#AWS S3
aws.s3.region=${S3.REGION}
aws.s3.access-key-id=${S3.ACCESS_KEY_ID}
aws.s3.secret-access-key=${S3.SECRET_ACCESS_KEY}
aws.s3.bucket=${S3.BUCKET}

aws.s3.image-url=${S3.URL}
```

```
# Edussafy
edussafy.id=${EDUSSAFY.USER_ID}
edussafy.pwd=${EDUSSAFY.USER_PWD}
```

5. CI/CD

AWS

- 포트 번호

MariaDB	3306
Jenkins	8081
Backend	8080
Nginx	80/443
Canvas	3000

Docker

```
# 도커 삭제
$ sudo apt-get remove docker docker-engine docker.io containe

$ sudo apt-get update

// 의존성 패키지 설치
$ sudo apt-get install apt-transport-https ca-certificates cu

// Docker 패키지 인증 키 추가
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | s

// 인증키 확인
$ sudo apt-key fingerprint 0EBFCD88

// Docker 저장소 추가
$ sudo add-apt-repository "deb [arch=amd64] https://download.

// 저장소 업데이트
```

```
$ sudo apt-get update

// 도커 CE 최신 버전 설치
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

MariaDB

```
// 도커 MariaDB 이미지 다운
$ sudo docker pull mariadb

// 도커 이미지 확인
$ sudo docker images

// 3306 포트 열기
$ sudo ufw allow 3306

// 컨테이너 MariaDB 적재
$ sudo docker run -d --name mariadb -e MYSQL_ROOT_PASSWORD=[파

// 컨테이너 확인
$ sudo docker ps

// 컨테이너에 접속
$ sudo docker exec -it mariadb bash

// 관리자로 접속
$ mariadb -u root -p

// 사용자 추가
use mariadb

CREATE USER '사용자명'@'%' IDENTIFIED BY '비밀번호';

GRANT ALL PRIVILEGES ON *.* TO '사용자명';

FLUSH PRIVILEGES;
```

Jenkins

```
// 도커 jenkins 이미지 pull
$ sudo docker pull jenkins/jenkins:lts

//8081 포트 열기
$ sudo ufw allow 8081

// jenkins 이미지 컨테이너로 적재
$ sudo docker run --name jenkins -d -p 8080:8080 -p 50000:500

// 초기 비밀번호 확인
$ sudo docker logs jenkins

// jenkins - gitlab 연동
```

Nginx

```
cd /etc/nginx/sites-available
sudo vim default
```

- default

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

        server_name k10e203.p.ssafy.io;

        return 308 https://k10e203.p.ssafy.io$request_uri;
}

server {
        root /var/www/html;

        index index.html index.htm index.nginx-debian.html
            server_name k10e203.p.ssafy.io; # managed by C
```

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


        location /api {
                proxy_pass http://localhost:8080;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_ad
                proxy_set_header Host $host;
                proxy_connect_timeout 120;
                proxy_send_timeout 120;
                proxy_read_timeout 120;
                send_timeout 120;
        }

        location /api/stream {
                proxy_pass http://localhost:8080;
                proxy_set_header Connection '';
                proxy_set_header Cache-Control 'no-cache';
                proxy_set_header X-Accel-Buffering 'no';
                proxy_set_header Content-Type 'text/event-
                proxy_buffering off;
                chunked_transfer_encoding on;
                proxy_read_timeout 300;
        }

        location /ws {
                proxy_pass http://localhost:9090/ws;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "Upgrade";
                proxy_set_header Host $host;
        }

        location /v1 {
                proxy_pass http://localhost:9090;
```

```
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "Upgrade";
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_ad
                proxy_set_header Host $host;
                # proxy_set_header Origin "";
        }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl http2; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/k10e203.p.ssafy.
    ssl_certificate_key /etc/letsencrypt/live/k10e203.p.ss
    include /etc/letsencrypt/options-ssl-nginx.conf; # man
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # manag

}

server {
    if ($host = k10e203.p.ssafy.io) {
        return 308 https://$host$request_uri;
    } # managed by Certbot

        listen 80 ;
        listen [::]:80 ;
    server_name k10e203.p.ssafy.io;
    return 404; # managed by Certbot
}
```

6. 배포 단계

Backend

- Http server
    - Dockerfile

```
# 첫 번째 스테이지: Gradle 빌드
FROM gradle:8.6 AS builder

# 작업 디렉토리 설정
WORKDIR /home/gradle/project

# Gradle 빌드에 필요한 파일 복사
COPY build.gradle .
COPY settings.gradle .
# COPY gradle.properties .
COPY src src

# 의존성 다운로드 및 애플리케이션 빌드
RUN gradle build --no-daemon -x test

# 두 번째 스테이지: 빌드된 JAR를 실행할 OpenJDK 기반의 JRE 이미
FROM openjdk:17-alpine

ENV JAVA_OPTS "-XX:+AllowRedefinitionToAddDeleteMethods

# 작업 디렉토리 설정
WORKDIR /app

# 첫 번째 스테이지에서 빌드된 JAR를 두 번째 스테이지로 복사
COPY --from=builder /home/gradle/project/build/libs/*.j

# 애플리케이션 실행
ENTRYPOINT ["sh", "-c", "java ${JAVA_OPTS} -jar -Dsprin
```

  - Docker Container 적재

```
docker run -d -p 8080:8080 --name backend backend
```

- WebSocket server
  - Dockerfile

```
# 첫 번째 스테이지: Gradle 빌드
FROM gradle:8.6 AS builder

# 작업 디렉토리 설정
WORKDIR /home/gradle/project

# Gradle 빌드에 필요한 파일 복사
COPY build.gradle .
COPY settings.gradle .
COPY src src

# 의존성 다운로드 및 애플리케이션 빌드
RUN gradle build --no-daemon -x test

# 두 번째 스테이지: 빌드된 JAR를 실행할 OpenJDK 기반의 JRE 이미
FROM openjdk:17-oracle

# 작업 디렉토리 설정
WORKDIR /app

# 첫 번째 스테이지에서 빌드된 JAR를 두 번째 스테이지로 복사
COPY --from=builder /home/gradle/project/build/libs/*.j

# 애플리케이션 실행
ENTRYPOINT ["sh", "-c", "java -jar -Dspring.profiles.ac
```

- Docker container 적재

```
docker run -d -p 9090:9090 backend-ws backend-ws
```

Embedded

- Dockerfile

```
# 베이스 이미지 설정
FROM electronuserland/builder:wine AS builder

# 작업 디렉토리 생성 및 설정
```

```
WORKDIR /app

# 필요한 파일 복사
COPY . .

# 애플리케이션 의존성 설치
RUN npm install

# Windows용 애플리케이션 빌드
RUN npm run build:win

# # Linux용 애플리케이션 빌드
RUN npm run build:pi

# # 최종 빌드 디렉토리 구조 확인
RUN ls -l /app/dist
```

- Docker Container 적재

```
docker run --name embedded dongho416/embedded

// Container 속 .exe 파일 추출
```