📋 포팅 매뉴얼

🔎 서비스 개요

닌텐도 스위치 구매 전, 한 번 체험해 보고 싶으신 적 있으셨나요?

여행 갈 때만 쓰고 싶은데 평소엔 잘 쓰지 않아 구입을 망설인 제품이 있으신가요?

우리 동네 안심 물품 대여 서비스, `See You Again` 입니다.

🛠️ 주요 기능 소개

- 물품 대여 : 잠깐 필요한데 사기엔 부담스러운 물품을 대여
- 안전 구역 추천 : 경찰서, cctv, 가로등 위치 데이터를 기반으로 거래하기 안전한 장소 추천
- 실시간 위치 공유 : 물품 대여나 반납 시 서로 헤매지 않도록 실시간 위치 공유
- 실시간 채팅 : 대여 예약이나 장소 변경, 약속 시간 등을 정하기 쉽도록 실시간 채팅 기능 제공

🔧 기술 스택

- **FrontEnd**
  - Visual Studio Code
  - Node.js : 9.4.1
  - react : 18.2.0
  - react-dom : 18.2.0
  - react-chartjs-2 : ^5.2.0
  - react-hook-form : ^7.43.8
  - redux : ^4.2.1
  - husky : ^8.0.3
  - eslint : ^8.2.0
  - prettier : 2.8.4
  - tailwindcss : ^3.2.7
  - axios : ^1.3.4
  - firebase : 9.22.0
- **BackEnd**
  - IntelliJ
  - OpenJDK 11
  - Gradle : 7.6.1
  - SpringBoot v2.7.9
  - SpringCloud : 2021.0.6
    - Netflix Eureka Service (Discovery Service)
    - API Gateway
  - Spring Security

    - Kafka
    - Websocket : 2.3.3
    - firebase : 9.1.1
- **CI/CD**
    - AWS EC2
    - Docker
        - Bridge Network
    - Jenkins
        - Pipeline
- **협업 툴**
    - Git Lab
    - Jira
    - Mattermost
    - Discord
    - Notion
    - Postman
- **DB**
    - MySQL
    - Redis

⚙️ 환경 변수

- user-service application.yml

```
server:
  # port 번호가 0번이면 랜덤으로 배정된다.
  port: 0

spring:
  application:
    name: user-service
  servlet:
    multipart:
      maxFileSize: 50MB
      maxRequestSize: 50MB
  h2:
    console:
      enabled: true
      settings:
        web-allow-others: true
      path: /h2-console
  datasource:
#    url: jdbc:mysql://localhost:3306/seeyouagain?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC&characterEncoding=UTF-8
    url: jdbc:mysql://k8c101.p.ssafy.io:3307/seeyouagain?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=Asia/Seoul&character
    username: root
    password: seeyouagain1234
#    password: kp23156385@
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    database: mysql
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect # 추가 해준 부분
    hibernate:
      ddl-auto: update
    show-sql: true
    generate-ddl: true
    properties:
      hibernate:
        default_batch_fetch_size: 500
  redis:
    lettuce:
      pool:
        min-idle: 0
        max-idle: 8
        max-active: 8
    port: 6379
    host: localhost
  profiles:
    include: oauth

eureka:
  instance:
    prefer-ip-address: true
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.value}}
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://127.0.0.1:8761/eureka

logging:
  level:
    com.example.userservice.client: DEBUG

gateway:
#    ip: 192.168.100.51
#  ip: 192.168.100.120
  ip: 172.18.0.7/16
#  ip: 192.168.0.100
#  ip: 172.20.10.2

product-service:
  ip: 172.18.0.8/16

chatting-service:
  ip: 172.18.0.11/16

token:
  secret: dyAeHubOOc8KaOfYB6XEQoEj1QzRlVgtjNL8PYs1A1tymZvvqkcEU7L1imkKHeDa

cloud:
  aws:
    s3:
      bucket: seeyouagain-s3-bucket
    stack.auto: false
    region.static: ap-northeast-2
    credentials:
      accessKey: AKIAZDX46XDEUXV5APLQ
      secretKey: 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
```

- product-service application.yml

```
server:
  port: 0

spring:
  application:
    name: product-service
  servlet:
    multipart:
      maxFileSize: 50MB
      maxRequestSize: 50MB
  h2:
    console:
      enabled: true
      settings:
        web-allow-others: true
      path: /h2-console
  datasource:
#    url: jdbc:mysql://localhost:3306/seeyouagain?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=Asia/Seoul
    url: jdbc:mysql://k8c101.p.ssafy.io:3308/seeyouagain?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=Asia/Seoul
    username: root
    password: seeyouagain1234
#    password: kp23156385@
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    database: mysql
    database-platform: org.hibernate.dialect.MySQL8Dialect
    hibernate:
      ddl-auto: update
    show-sql: true
    generate-ddl: true

eureka:
  instance:
    prefer-ip-address: true
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.value}}
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://127.0.0.1:8761/eureka

logging:
  level:
    com.example.productservice.client: DEBUG

cloud:
  aws:
    region:
      static: ap-northeast-2
    s3:
      bucket: seeyouagain-s3-bucket
    credentials:
      access-key: AKIAZDX46XDEUXV5APLQ
      secret-key: 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
    stack:
      auto: false
```

- chatting-service application.yml

```
server:
  port: 0

spring:
  application:
    name: chatting-service
  servlet:
    multipart:
      maxFileSize: 50MB
      maxRequestSize: 50MB
  h2:
    console:
      enabled: true
      settings:
        web-allow-others: true
      path: /h2-console
  datasource:
#    url: jdbc:mysql://localhost:3306/seeyouagain?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=Asia/Seoul
    url: jdbc:mysql://k8c101.p.ssafy.io:3309/seeyouagain?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=Asia/Seoul
    username: root
    password: seeyouagain1234
#    password: kp23156385@
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    database: mysql
    database-platform: org.hibernate.dialect.MySQL8Dialect
    hibernate:
      ddl-auto: update
    show-sql: true
    generate-ddl: true

eureka:
  instance:
    prefer-ip-address: true
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.value}}
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://127.0.0.1:8761/eureka

logging:
  level:
    com.example.chattingservice.client: DEBUG

cloud:
  aws:
    s3:
      bucket: seeyouagain-s3-bucket
    stack.auto: false
    region.static: ap-northeast-2
    credentials:
      accessKey: AKIAZDX46XDEUXV5APLQ
      secretKey: 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
```

⚒️ 빌드 및 배포

FrontEnd

- Docker 이미지 생성을 위한 Dockerfile (해당 파일은 frontend 폴더 내에 작성되어 있습니다.)

```
# Use an official Node runtime as a parent image
FROM node:16.19.0

# Set the working directory to /app
# Copy the package.json and package-lock.json files to the container
COPY package*.json ./

# Install dependencies
# RUN npm install --production 이건 지금 에러남
RUN npm install

# Copy the rest of the application files to the container
COPY . .

# Build the production version of the app
RUN npm run build

# Expose the port that the app will run on2332
EXPOSE 3000

CMD ["npm", "run", "start"]
```

- FrontEnd Jenkins Pipeline Script

```
pipeline {
    agent any

    environment {
        FRONTEND_PROJECT='frontend-service'
    }

    stages {
        // 깃랩 프로젝트 코드를 클론해오는 코드
        stage('github clone') {
            steps {
                git branch: 'release',
                // Global Credentials ID
                credentialsId: 'seeyouagainglobalkey',
                url: 'https://lab.ssafy.com/s08-final/S08P31C101'
            }
        }

        // 프로젝트 빌드
        stage('Build') {
            // parallel - 프로젝트 병렬 처리
            parallel {
                stage('build-frontend'){
                    when {
                        changeset "frontend/**"
                    }
                    steps{
                        dir('frontend') {
                            sh 'docker build -t ssafyseeyouagain/${FRONTEND_PROJECT} .'
                          sh 'docker push ssafyseeyouagain/${FRONTEND_PROJECT}'
                        }
                    }
                }
            }
        }

        // 배포 명령어
        stage('Deploy'){
            // 병렬 처리
            parallel{
                stage('deploy-frontend-service'){
                    // 해당 프로젝트에서 changeset이 발생했을 때 동작
                    when{
                        changeset "frontend/**"
                    }
                    steps{
                        sh 'docker stop ${FRONTEND_PROJECT} || true && docker rm ${FRONTEND_PROJECT} || true'
                        sh 'docker run -d -p 3000:3000 --network seeyouagain-network --name ${FRONTEND_PROJECT}  ssafyseeyouagain/${FR
                    }
                }
            }
        }
    }
}
```

BackEnd

- eureka-service Dockerfile

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY build/libs/eureka-service-0.0.1-SNAPSHOT.jar EurekaService.jar
ENTRYPOINT ["java", "-jar", "EurekaService.jar"]
```

- apigateway-service Dockerfile

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY build/libs/apigateway-service-0.0.1-SNAPSHOT.jar ApigatewayService.jar
ENTRYPOINT ["java", "-jar", "ApigatewayService.jar"]
```

- user-service Dockerfile

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY build/libs/user-service-0.0.1-SNAPSHOT.jar UserService.jar
RUN ln -sf /usr/share/zoneinfo/Asia/Seoul /etc/localtime
ENTRYPOINT ["java", "-jar","UserService.jar"]
```

- product-service Dockerfile

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY build/libs/product-service-0.0.1-SNAPSHOT.jar ProductService.jar
RUN ln -sf /usr/share/zoneinfo/Asia/Seoul /etc/localtime
ENTRYPOINT ["java", "-jar","ProductService.jar"]
```

- chatting-service Dockerfile

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY build/libs/chatting-service-0.0.1-SNAPSHOT.jar ChattingService.jar
RUN ln -sf /usr/share/zoneinfo/Asia/Seoul /etc/localtime
ENTRYPOINT ["java", "-jar","ChattingService.jar"]
```

- BackEnd Jenkins Pipeline Script

```
pipeline {
    agent any

    tools {
        gradle 'gradle7.6'
    }

    // 빌드 커맨드와 마이크로 서비스 선언
    environment {
        BUILD_COMMAND = ' ./gradlew clean build -x test'
        PRODUCT_PROJECT='product-service'
        USER_PROJECT='user-service'
        CHATTING_PROJECT='chatting-service'
        EUREKA_PROJECT='eureka-service'
        GATEWAY_PROJECT='api-gateway-service'
    }

    stages {
        // 깃랩 프로젝트 코드를 클론해오는 코드
        stage('github clone') {
            steps {
                git branch: 'release',
                // Global Credentials ID
                credentialsId: 'seeyouagainglobalkey',
                url: 'https://lab.ssafy.com/s08-final/S08P31C101'
            }
        }

        // 프로젝트 빌드
        stage('Build') {
            // parallel - 프로젝트 병렬 처리
            parallel {
                stage('build-eureka-service'){
                    when {
                        changeset "backend/eureka-service/**"
                    }
                    steps{
                        dir('backend/eureka-service') {
                            sh 'chmod +x ./gradlew'
                            sh "$BUILD_COMMAND"
                        }
                    }
                }
                stage('build-api-gateway-service'){
                    when {
                        changeset "backend/apigateway-service/**"
                    }
                    steps{
                        dir('backend/apigateway-service') {
                            sh 'chmod +x ./gradlew'
                            sh "$BUILD_COMMAND"
                        }
                    }
                }
                stage('build-product-service'){
                    // 해당 프로젝트에 changeset이 발생했을때 동작
                    when {
                        changeset "backend/product-service/**"
                    }
                    steps{
                        dir('backend/product-service') {
                            sh 'chmod +x ./gradlew'
                          // environment 에서 선언한 빌드 커맨드
                            sh "$BUILD_COMMAND"
                        }
                    }
                }
                stage('build-user-service'){
                    when {
                        changeset "backend/user-service/**"
                    }
                    steps{
                        dir('backend/user-service') {
                            sh 'chmod +x ./gradlew'
                            sh "$BUILD_COMMAND"
                        }
                    }
                }
                stage('build-chatting-service'){
                    when {
                        changeset "backend/chatting-service/**"
                    }
                    steps{
                        dir('backend/chatting-service') {
                            sh 'chmod +x ./gradlew'
                            sh "$BUILD_COMMAND"
                        }
                    }
                }
            }
        }

        // 빌드된 도커 이미지를 도커 허브에 push 하기 위한 작업
        stage('Backup & Copy'){
            // 병렬 처리
            parallel{
                stage('backup-copy-eureka-service'){
                    when{
                        changeset "backend/eureka-service/**"
                    }
                    steps{
                        dir('backend/eureka-service') {
                            sh 'docker build -t ssafyseeyouagain/${EUREKA_PROJECT} .'
                            sh 'docker push ssafyseeyouagain/${EUREKA_PROJECT}'
                        }
                    }
                }
                stage('backup-copy-api-gateway-service'){
```

```
                when{
                    changeset "backend/apigateway-service/**"
                }
                steps{
                    dir('backend/apigateway-service') {
                        sh 'docker build -t ssafyseeyouagain/${GATEWAY_PROJECT} .'
                        sh 'docker push ssafyseeyouagain/${GATEWAY_PROJECT}'
                    }
                }
            }
            stage('backup-copy-product-service'){
                // 해당 프로젝트에 changeset이 발생했을 때 동작
                when{
                    changeset "backend/product-service/**"
                }
                steps{
                    dir('backend/product-service') {
                        sh 'docker build -t ssafyseeyouagain/${PRODUCT_PROJECT} .'
                        sh 'docker push ssafyseeyouagain/${PRODUCT_PROJECT}'
                    }
                }
            }
            stage('backup-copy-user-service'){
                when{
                    changeset "backend/user-service/**"
                }
                steps{
                    dir('backend/user-service') {
                        sh 'docker build -t ssafyseeyouagain/${USER_PROJECT} .'
                        sh 'docker push ssafyseeyouagain/${USER_PROJECT}'
                    }
                }
            }
            stage('backup-copy-chatting-service'){
                when{
                    changeset "backend/chatting-service/**"
                }
                steps{
                    dir('backend/chatting-service') {
                        sh 'docker build -t ssafyseeyouagain/${CHATTING_PROJECT} .'
                        sh 'docker push ssafyseeyouagain/${CHATTING_PROJECT}'
                    }
                }
            }
        }
    }

    // 배포 명령어
    stage('Deploy'){
        // 병렬 처리
        parallel{
            stage('deploy-eureka-service'){
                when{
                    changeset "backend/eureka-service/**"
                }
                steps{
                    sh 'docker stop ${EUREKA_PROJECT} || true && docker rm ${EUREKA_PROJECT} || true'
                    sh 'docker run -d -p 8761:8761 --network seeyouagain-network --name ${EUREKA_PROJECT} ssafyseeyouagain/${EUREK
                }
            }
            stage('deploy-api-gateway-service'){
                when{
                    changeset "backend/apigateway-service/**"
                }
                steps{
                    sh 'docker stop ${GATEWAY_PROJECT} || true && docker rm ${GATEWAY_PROJECT} || true'
                    sh 'docker run -d -p 8000:8000 --network seeyouagain-network --name ${GATEWAY_PROJECT} -e "eureka.client.servi
                }
            }
            stage('deploy-product-service'){
                // 해당 프로젝트에서 changeset이 발생했을 때 동작
                when{
                    changeset "backend/product-service/**"
                }
                steps{
                    // 기존에 해당 프로젝트 이름으로 만들어진 컨테이너가 실행중이라면 정지 후 삭제
                    sh 'docker stop ${PRODUCT_PROJECT} || true && docker rm ${PRODUCT_PROJECT} || true'
                    // docker hub에 재 업로드된 이미지를 받아서 컨테이너 재실행
                    sh 'docker run -d --network seeyouagain-network --name ${PRODUCT_PROJECT} -e "eureka.client.serviceUrl.default
                }
            }
            stage('deploy-user-service'){
                when{
                    changeset "backend/user-service/**"
                }
                steps{
                    sh 'docker stop ${USER_PROJECT} || true && docker rm ${USER_PROJECT} || true'
```

```
                        sh 'docker run -d --network seeyouagain-network --name ${USER_PROJECT} -e "eureka.client.serviceUrl.defaultZon
                    }
                }
                stage('deploy-chatting-service'){
                    when{
                        changeset "backend/chatting-service/**"
                    }
                    steps{
                        sh 'docker stop ${CHATTING_PROJECT} || true && docker rm ${CHATTING_PROJECT} || true'
                        sh 'docker run -d --network seeyouagain-network --name ${CHATTING_PROJECT} -e "eureka.client.serviceUrl.defaul
                    }
                }
            }
        }
    }
}
```

Nginx

- 상태 확인

💡 sudo service nginx status

- 재실행

💡 sudo service nginx restart

- 환경설정 - etc/nginx/sites-available/custom.conf

```
upstream frontend {
    server localhost:3000;
}

upstream backend {
    server localhost:8000;
}

server {
    listen 80;
    server_name k8c101.p.ssafy.io;

    location /{
        proxy_pass http://frontend;
        proxy_set_header X-Forwarded-Proto $scheme;
        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 /user-service {
       proxy_pass http://backend/user-service;
        proxy_set_header X-Forwarded-Proto $scheme;
        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 /product-service {
        proxy_pass http://backend/product-service;
        proxy_set_header X-Forwarded-Proto $scheme;
        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 /chatting-service {
        proxy_pass http://backend/chatting-service;
        proxy_set_header X-Forwarded-Proto $scheme;
        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 Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    if ($scheme != "https") {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name k8c101.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/k8c101.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k8c101.p.ssafy.io/privkey.pem;

    location /{
        proxy_pass http://frontend;
        proxy_set_header X-Forwarded-Proto $scheme;
        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 /_next/webpack-hmr {
        proxy_pass http://frontend/_next/webpack-hmr;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $host;
    }

    location /user-service {
        proxy_pass http://backend/user-service;
        proxy_set_header X-Forwarded-Proto $scheme;
        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 /product-service {
        proxy_pass http://backend/product-service;
        proxy_set_header X-Forwarded-Proto $scheme;
        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 /chatting-service {
        proxy_pass http://backend/chatting-service;
        proxy_set_header X-Forwarded-Proto $scheme;
        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 Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

💬 외부 서비스

Kakao OAuth : application-oauth.yml에 해당 내용 있음

OAuth 기반 소셜 로그인 API 제공

https://developers.kakao.com/docs/latest/ko/getting-started/rest-api

```
spring:
  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: 5c2af632e5eb943eadbf20d0c4006bdb
            client-secret: x7TGPwsREQ44ndn3YoVQOberBwoPFbRE
            client-name: Kakao
            scope:
              - profile_nickname
              - profile_image
              - account_email
            authorization-grant-type: authorization_code
            redirect-uri: http://k8c101.p.ssafy.io:8000/user-service/login/oauth2/code/kakao
            client-authentication-method: POST
        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id
```