포팅 메뉴얼

포팅 메뉴얼

목차

1. 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost
- 디자인 : Figma
- CI/CD : Jenkins, GitLab CICD(GitLab Runner)

2. 개발 도구

- Visual Studio Code : 1.85.1
- Intellij : 2023.3.2 (Ultimate Edition)

3. 개발 환경

Frontend

Node.js	20.11.0
React	18.2.0
Zustand	4.5.2
pnpm	8.15.6

Backend

Java	openjdk21.0.1(zulu 21.30.15)
Spring Boot	3.2.1

Server

AWS S3	Free Tier
AWS EC2	CPU : 4코어 RAM: 16GB 볼륨:320GB(SSD) ,6TB

Service

MongDB	5.0.24 Atlas
NginX	1.18.0
Jenkins	2.426.2
Docker	25.0.1
Ubuntu	"20.04.6 LTS (Focal Fossa)”

4. 환경변수 형태

Frontend

- package.json

```
{
  "name": "pairing-fe",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
    "lint": "eslint . --ext ts,tsx --report-unused-disable
    "preview": "vite preview"
  },
  "dependencies": {
    "@hookform/resolvers": "^3.3.4",
    "@radix-ui/react-alert-dialog": "^1.0.5",
    "@radix-ui/react-checkbox": "^1.0.4",
    "@radix-ui/react-dialog": "^1.0.5",
    "@radix-ui/react-hover-card": "^1.0.7",
    "@radix-ui/react-label": "^2.0.2",
    "@radix-ui/react-popover": "^1.0.7",
    "@radix-ui/react-progress": "^1.0.3",
    "@radix-ui/react-radio-group": "^1.1.3",
    "@radix-ui/react-select": "^2.0.0",
    "@radix-ui/react-slot": "^1.0.2",
    "@radix-ui/react-tabs": "^1.0.4",
    "@stomp/stompjs": "^7.0.0",
    "@types/node": "^20.11.25",
    "@types/sockjs-client": "^1.5.4",
    "autoprefixer": "^10.4.18",
    "axios": "^1.6.8",
    "class-variance-authority": "^0.7.0",
    "classnames": "^2.5.1",
    "clsx": "^2.1.0",
    "embla-carousel-react": "^8.0.0",
    "lucide-react": "^0.354.0",
    "moment": "^2.30.1",
    "postcss": "^8.4.35",
```

```
    "react": "^18.2.0",
    "react-daum-postcode": "^3.1.3",
    "react-dom": "^18.2.0",
    "react-hook-form": "^7.51.1",
    "react-icons": "^5.0.1",
    "react-intersection-observer": "^9.8.1",
    "react-multi-clamp": "^2.0.6",
    "react-number-format": "^5.3.4",
    "react-query": "^3.39.3",
    "react-router-dom": "6",
    "react-select": "^5.8.0",
    "sockjs-client": "^1.6.1",
    "sweetalert2": "^11.10.7",
    "tailwind-merge": "^2.2.1",
    "tailwindcss": "^3.4.1",
    "tailwindcss-animate": "^1.0.7",
    "vaul": "^0.9.0",
    "zod": "^3.22.4",
    "zustand": "^4.5.2"
  },
  "devDependencies": {
    "@tailwindcss/forms": "^0.5.7",
    "@types/react": "^18.2.56",
    "@types/react-dom": "^18.2.19",
    "@typescript-eslint/eslint-plugin": "^7.0.2",
    "@typescript-eslint/parser": "^7.0.2",
    "@vitejs/plugin-react": "^4.2.1",
    "eslint": "^8.56.0",
    "eslint-plugin-react-hooks": "^4.6.0",
    "eslint-plugin-react-refresh": "^0.4.5",
    "prettier": "3.2.5",
    "prettier-plugin-tailwindcss": "^0.5.12",
    "typescript": "^5.2.2",
    "vite": "^5.1.4"
  }
}
```

Backend

- bootstrap.yml (Spring Cloud Configuration을 통한 분산 환경 설정을 활용하였습니다.)

```
spring:
  profiles:
    active: main
  cloud:
    config:
      uri: https://www.ssafyhelper.shop
      name: application
```

- configuration-server - application-main.yml

```
ymlchecker: main
server:
  port: 8080
  tomcat:
    max-swallow-size: 10MB
  servlet:
    session:
      cookie:
        http-only: true
        path: /
        secure: true
        same-site: none


cloud:
  aws:
    s3:
      bucket:  ssafyams3
```

```
    stack.auto: false
    region.static: ap-northeast-2
    credentials:
      accessKey: AKIATX7FZU6TBQVN34NB
      secretKey: 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

spring:
  datasource:
    driver-class-name: org.postgresql.Driver
    url: jdbc:postgresql://j10a709.p.ssafy.io:5432/pairing
    username: postgres
    password: root1432!!

  jpa:
    database: postgresql
    use-new-id-generator-mappings: true
    properties:
      hibernate:
        format_sql: true
        dialect: org.hibernate.dialect.PostgreSQLDialect
        default_batch_fetch_size: 100
    hibernate:
      ddl-auto: create
  show-sql: true

  security:
    jwt:
      secret-key: XsCdnHZMC3WzqQkxneSWIjyrOcDJm1Exodia
    oauth2:
      client:
        registration:
          kakao:
            client-id: 5a1bf993c63b329bf6aeb64d1d0de64b
            redirect-uri: "https://j10a709.p.ssafy.io/auth
            client-secret: 7n0svlhxcf1MwZbI3idSvfj8KRKK8p5
            authorization-grant-type: authorization_code
            scope: profile_nickname, email
          google:
```

```
            client-id: 578268866259-acovbgvbd9lgv45ptp72dc
            redirect-uri: https://j10a709.p.ssafy.io/auth/
            client-secret: GOCSPX---WiEjMfeb5gPOYebBIyyrGm
            authorization-grant-type: authorization_code
            scope: openid, profile, email
        provider:
          kakao:
            authorization_uri: https://kauth.kakao.com/oau
            token_uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/
          google:
            token-uri: https://www.googleapis.com/oauth2/v
            user-info-uri: https://www.googleapis.com/user
  kafka:
    bootstrap-servers: 3.38.252.18:9092
    consumer:
      auto-offset-reset: earliest
logging:
  level:
    org:
      hibernate:
        SQL: DEBUG
        orm:
          jdbc:
            bind: trace
      springframework:
        kafka: WARN
```

Configuration Server Dockerfile

이 도커파일은 Configuration server의 구동에 필요한 도커파일입니다. 위에 적힌 값들을 채운 후에 실행해주세요.

- Dockerfile

```
# OpenJDK 이미지를 기반으로 합니다.
FROM azul/zulu-openjdk:21
```

```
# 작업 디렉토리를 설정합니다.
WORKDIR /app

# 빌드 컨텍스트에서 JAR 파일을 작업 디렉토리로 복사합니다.
COPY I10A709BE-0.0.1-SNAPSHOT.jar /app/pairing-backend.jar

# 컨테이너가 시작될 때 실행될 명령어를 정의합니다.

# CMD ["java", "-jar", "pairing-backend.jar", "--spring.pr
CMD ["java", "-jar", "pairing-backend.jar", "--spring.prof
```

GitLab에 들어갈 설정 파일들

배포용

- application-main.yml

```
ymlchecker: main
server:
  port: 8080
  tomcat:
    max-swallow-size: 10MB
  servlet:
    session:
      cookie:
        http-only: true
        path: /
        secure: true
        same-site: none
```

```
cloud:
  aws:
    s3:
      bucket:  ssafyams3
    stack.auto: false
    region.static: ap-northeast-2
    credentials:
      accessKey: AKIATX7FZU6TBQVN34NB
      secretKey: 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

spring:
  datasource:
    driver-class-name: org.postgresql.Driver
    url: jdbc:postgresql://j10a709.p.ssafy.io:5432/pairing
    username: postgres
    password: root1432!!

  jpa:
    database: postgresql
    use-new-id-generator-mappings: true
    properties:
      hibernate:
        format_sql: true
        dialect: org.hibernate.dialect.PostgreSQLDialect
        default_batch_fetch_size: 100
    hibernate:
      ddl-auto: create
  show-sql: true

  security:
    jwt:
      secret-key: XsCdnHZMC3WzqQkxneSWIjyrOcDJm1Exodia
    oauth2:
      client:
        registration:
          kakao:
            client-id: 5a1bf993c63b329bf6aeb64d1d0de64b
            redirect-uri: "https://j10a709.p.ssafy.io/auth
```

```
            client-secret: 7n0svlhxcf1MwZbI3idSvfj8KRKK8p5
            authorization-grant-type: authorization_code
            scope: profile_nickname, email
          google:
            client-id: 578268866259-acovbgvbd9lgv45ptp72dc
            redirect-uri: https://j10a709.p.ssafy.io/auth/
            client-secret: GOCSPX---WiEjMfeb5gPOYebBIyyrGm
            authorization-grant-type: authorization_code
            scope: openid, profile, email
        provider:
          kakao:
            authorization_uri: https://kauth.kakao.com/oau
            token_uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/
          google:
            token-uri: https://www.googleapis.com/oauth2/v
            user-info-uri: https://www.googleapis.com/user
  kafka:
    bootstrap-servers: 3.38.252.18:9092
    consumer:
      auto-offset-reset: earliest
logging:
  level:
    org:
      hibernate:
        SQL: DEBUG
        orm:
          jdbc:
            bind: trace
      springframework:
        kafka: WARN
```

Nginx Conf

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
# include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
    # multi_accept on;
}

http {

    ##
    include /etc/nginx/mime.types;
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    client_max_body_size 50M;
    #  include /etc/nginx/conf.d/*.conf;
  #  include /etc/nginx/sites-enabled/*;

        server{
        server_name j10a709.p.ssafy.io;
        location /api/{
            proxy_pass http://localhost:8080/;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_for
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_buffering on;
            proxy_buffer_size          128k;
            proxy_buffers              4 256k;
            proxy_busy_buffers_size    256k;
            proxy_ssl_server_name on;
```

```
        proxy_connect_timeout 1600;
        proxy_send_timeout 1600;
        proxy_read_timeout 1600;
        add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Allow-Methods' 'GET, P
        add_header 'Access-Control-Allow-Headers' 'Conten
        add_header 'Access-Control-Max-Age' 86400;
    }
    location /kafka/{
        proxy_pass http://localhost:9092/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_for
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_buffering on;
        proxy_buffer_size          128k;
        proxy_buffers              4 256k;
        proxy_busy_buffers_size    256k;
        proxy_ssl_server_name on;

        proxy_connect_timeout 1600;
        proxy_send_timeout 1600;
        proxy_read_timeout 1600;


    }



    location / {
        root        /home/ubuntu/deploy/FE/dist;
        index       index.html index.htm;
        try_files   $uri $uri/ /index.html;
```

```
            }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/j10a709.p.ssafy.io/
    ssl_certificate_key /etc/letsencrypt/live/j10a709.p.ssafy
    include /etc/letsencrypt/options-ssl-nginx.conf; # manage
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed

}



        server{
    if ($host = j10a709.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot



        listen 80;
        server_name j10a709.p.ssafy.io;
    return 404; # managed by Certbot


}}
```

5. 배포 환경 구축

기본 설정

- aws ec2
  - 도커

1. Docker 설치

```
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubu
ntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/
etc/apt/keyrings/docker.asc] https://download.docker.
com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") st
able" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /de
v/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containe
rd.io docker-buildx-plugin docker-compose-plugin
```

Check

```
docker --version
```

2. 자바 21 설치

```
sudo apt install gnupg ca-certificates curl
curl -s https://repos.azul.com/azul-repo.key | sudo gpg
echo "deb [signed-by=/usr/share/keyrings/azul.gpg] http


sudo apt update

sudo apt install zulu21-jdk -y
```

```
java -version
```

3. Git 설치 및 Repo 받아오기

```
sudo apt-get install git
git clone https://lab.ssafy.com/s10-blockchain-contract
```

4. nginx 설치

```
sudo apt update
sudo apt install nginx
```

4-1. certbot (secure cookie 때문에 https 적용을 하거나 혹은 FE/.env의 https → http로 바꿔주세요.)

```
sudo apt-get install python3-certbot-nginx
sudo certbot --nginx -d <<domain>>
```

5. Frontend 띄우기

```
# node 설치 20.11
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo
sudo apt-get install -y nodejs

cd S10P22A709/FE
```

```
# pnpm 설치
sudo npm install -g pnpm

# 패키지 설치
sudo pnpm i

# 요청 api 엔드포인트 설정
sudo vim .env

# 프로젝트 빌드
pnpm run build

cd dist
pwd

# 출력 위치 복사후 위의 nginx.conf에 넣기
sudo vim /etc/nginx/nginx.conf

#dist 폴더 붙여넣기
mv ./dist /home/ubuntu/deploy/FE/dist
sudo systemctl restart nginx
```

KAFKA 및 기반 환경 세팅

```
version: '3'
services:
  zookeeper:
    # 사용할 이미지
    image: wurstmeister/zookeeper
    # 컨테이너명 설정
    container_name: zookeeper
    # 접근 포트 설정 (컨테이너 외부:컨테이너 내부)
    ports:
      - "2181:2181"
```

```
# 서비스 명
kafka:
  # 사용할 이미지
  image: wurstmeister/kafka
  # 컨테이너명 설정
  container_name: kafka
  # 접근 포트 설정 (컨테이너 외부:컨테이너 내부)
  ports:
    - "9092:9092"
  # 환경 변수 설정
  environment:
    KAFKA_ADVERTISED_HOST_NAME: j10a709.p.ssafy.io
    KAFKA_ADVERTISED_PORT: 9092
    KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
    KAFKA_MESSAGE_MAX_BYTES: 2000000
    KAFKA_REPLICA_FETCH_MAX_BYTES: 2000000
  # 볼륨 설정
  volumes:
    - /var/run/docker.sock
  # 의존 관계 설정
  depends_on:
    - zookeeper
kafka-ui:
  image: provectuslabs/kafka-ui
  container_name: kafka-ui
  ports:
    - "8987:8080"
  restart: always
  environment:
    - KAFKA_CLUSTERS_0_NAME=local
    - KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS=kafka:9092
    - KAFKA_CLUSTERS_0_ZOOKEEPER=zookeeper:2181
  depends_on:
    - zookeeper
    - kafka
postgres:
  image: postgres
```

```
    container_name: postgres
    ports:
      - "5432:5432"
    environment:
      POSTGRES_DB: pairing
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: root1432!!

  pairing-backend:
    build:
      context: ../
      dockerfile: ./backend/dockerFile/springDockerFi
le
    ports:
      - "8080:8080"
    depends_on:
      - zookeeper
      - kafka
```