포팅메뉴얼

1. 개발환경

1.1 Frontend

Next.js 및 관련 라이브러리

- next 14.2.13
- react ^18
- react-dom ^18
- zustand ^5.0.0-rc.2

지도 및 위치 서비스

- react-kakao-maps-sdk ^1.1.27

스타일링 및 최적화 라이브러리

- critters ^0.0.24

유틸리티 및 데이터 라이브러리

- axios ^1.7.7
- @tanstack/react-query ^5.56.2

아이콘 라이브러리

- react-icons ^5.3.0

PWA 관련 라이브러리

- next-pwa ^5.6.0

Firebase

- firebase ^10.14.0

SVG 처리 라이브러리

- @svgr/webpack ^8.1.0

디지털 잉크 처리 라이브러리

- iink-ts ^1.0.5

1.2 Backend

자바

- Java OpenJDK 17
- Spring Boot 3.3.1
- Spring Dependency Management 1.1.5
  - Spring Data JPA 3.3.1
  - Spring Security 3.3.1
  - Websocket 3.3.1
  - Validation 3.3.1
  - Lombok 1.18.34
- Gradle 8.8

JSON 및 XML 처리

- jsoup 1.15.3
- org.json 20231013

JWT

- io.jsonwebtoken 0.11.5

데이터베이스

- MySQL 8.0.33

Amazon S3

- com.amazonaws 1.12.765

SpringDoc

- org.springdoc 2.6.0

1.3 Server

- Ubuntu 20.04.6 LTS
- Nginx 1.27.0
- Docker 27.1.1

- Docker Compose 2.29.1
- Jenkins 2.471

1.4 Database

- MySQL 9.0.1

1.5 IDE

- Visual Studio Code 1.90.2
- IntelliJ IDEA 2024.1.4

1.6 형상 / 이슈관리

- Gitlab
- Jira

1.7 기타 툴

- Postman 11.8.0

2. 환경변수

2.1 Frontend

> **NEXT_PUBLIC_KAKAO_KEY**

2.2 Backend

가독성이 좋은 application.yml 을 작성하여 환경변수를 관리

```
spring:
  config:
    import: optional:file:.env[.properties]
  application:
    name: missing_back
  datasource:
    url: ${SPRING_DATASOURCE_URL}
    username: ${SPRING_DATASOURCE_USERNAME}
    password: ${SPRING_DATASOURCE_PASSWORD}
    driver-class-name: ${SPRING_DATASOURCE_DRIVER_CLASS_NAME}
    hikari:
      connection-timeout: 15000
      maximum-pool-size: 10
      max-lifetime: 240000
      leak-detection-threshold: 10000
  jpa:
    database-platform: org.hibernate.dialect.MySQLDialect
    hibernate:
```

```
      ddl-auto: update
      jdbc:
        time-zone: Asia/Seoul
  servlet:
    multipart:
      max-file-size: 20MB # 파일당 최대 크기 설정
      max-request-size: 20MB # 전체 요청 크기 설정

logging:
  level:
    org:
      springframework: DEBUG

cloud:
  aws:
    credentials:
      access-key: ${S3_ACCESS_KEY_ID}
      secret-key: ${S3_SECRET_ACCESS_KEY}
    region:
      static: us-east-1
    stack:
      auto: false

jwt:
  secret: ${JWT_SECRET}
  access-token-ms: ${JWT_ACCESS_TOKEN_MS}
  refresh-token-ms: ${JWT_REFRESH_TOKEN_MS}

springdoc:
  swagger-ui:
    path: /swagger-ui.html
```

3. EC2 세팅

3.1 Docker 설치

```
# Add Docker's official GPG key:
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/ubuntu/gpg -o /etc/apt/keyr
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.
```

```
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

# Docker 패키지 설치(최신 버전)
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plug

# Docker Engine 설치 성공 확인
sudo docker run hello-world
```

3.2 MySQL(Docker) 설치

```
# MySQL Docker 이미지 다운로드
## 버전 명시하지 않으면 최신버전으로 다운로드
$sudo docker pull mysql

# MySQL Docker 컨테이너 생성 및 실행
$sudo docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=<password> -d
```

3.3 Nginx 설치 및 Reverse Proxy 세팅

3.3.1 nginx 이미지 다운로드

```
$docker pull nginx:lates
```

3.3.2 nginx.conf 파일 작성

```
* ec2 인스턴스 /home/ubuntu/nginx.conf 경로에 존재

events { }

http {
    upstream backend {
        server backend:8080;
    }

    upstream frontend {
        server frontend:5000;
    }

    server {
        listen 80;
        server_name j11a202.p.ssafy.io;


        location /.well-known/acme-challenge/ {
        allow all;
            root /var/www/certbot;
```

```
        }
        location / {
            return 301 https://$host$request_uri;
        }
    }

    server {
        listen 443 ssl;
        server_name j11a202.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/j11a202.p.ssafy.io/fullchain.pe
        ssl_certificate_key /etc/letsencrypt/live/j11a202.p.ssafy.io/privkey.
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        # 최대 요청 크기 설정 (20MB)
        client_max_body_size 20M;

        location / {
            proxy_pass http://frontend/;
            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;
        }

        location /api/ {
            proxy_pass http://backend/api/;
            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;

        }
    }
}
```

3.3.3 docker-compose-prod.yml 파일 작성

```
version: '3.3'
services:
  nginx:
    image: nginx:latest
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
```

```
      - ./certbot/conf:/etc/letsencrypt
      - ./certbot/www:/var/www/certbot
    networks:
      - wish

  backend:
    image: kodongyeon/missing_back:8.1
    ports:
      - "8081:8080"
    env_file:
      - ./.env
    networks:
      - wish

  frontend:
    image: kodongyeon/missing_front:2.5
    ports:
      - "5000:5000"
    env_file:
      - ./.front_env
    networks:
      - wish

  certbot:
    image: certbot/certbot
    volumes:
      - ./certbot/conf:/etc/letsencrypt
      - ./certbot/www:/var/www/certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep

networks:
  wish:
```

3.3.4 SSL 인증서 받기

```
# ec2 인스턴스

# ssl 인증서를 저장할 경로를 만들어 준다.
$ mkdir -p certbot/conf
$ mkdir -p certbot/www

$ curl -L https://raw.githubusercontent.com/wmnnd/nginx-certbot/master/init-l
$ chmod +x init-letsencrypt.sh
$ vi init-letsencrypt.sh // 도메인, 이메일, 디렉토리 수정

domains=(j11a202.p.ssafy.io)
rsa_key_size=4096
data_path="./certbot"
```

```
email="" # Adding a valid address is strongly recommended
staging=0 # Set to 1 if you're testing your setup to avoid hitting request li
-> 위 부분 처럼 도메인, 경로 지정 후 저장

$ sudo ./init-letsencrypt.sh // script를 실행하여 인증서 발급
```

3.5 EC2 Port

Port 번호	내용
22	SSH
80	HTTP (HTTPS로 redirect)
443	HTTPS
3306	MySQL
5000	Frontend
8080	Backend

4. Front, Back 배포 Dockerfile

4.1 Frontend

4.1.1 frontend 루트 경로 파일구조

```
frontend/
│
├── .next/                      # Next.js에서 빌드된 파일들이 저장되는 디렉토리
│
├── node_modules/               # 프로젝트의 NPM 패키지들이 저장되는 디렉토리
│
├── public/                     # 정적 파일(이미지, 폰트 등)이 위치한 디렉토리
│
├── src/                        # 소스 코드가 위치한 디렉토리
│
├── .dockerignore               # Docker 빌드 시 제외할 파일 및 디렉토리 목록
├── .eslintrc.json              # ESLint 설정 파일
├── .gitignore                  # Git에서 추적하지 않을 파일 및 디렉토리 목록
├── .prettierrc                 # Prettier 코드 스타일 설정 파일
├── Dockerfile                  # Docker 이미지를 빌드하기 위한 설정 파일
├── next.config.mjs             # Next.js 설정 파일
├── next-env.d.ts               # Next.js에서 환경 변수를 정의하는 TypeScript 파일
├── package.json                # 프로젝트 메타정보 및 의존성을 정의하는 파일
├── package-lock.json           # NPM 패키지의 의존성 트리를 잠그는 파일
├── postcss.config.mjs          # PostCSS 설정 파일
├── README.md                   # 프로젝트에 대한 설명과 정보를 담은 파일
├── tailwind.config.ts          # Tailwind CSS 설정 파일
└── tsconfig.json               # TypeScript 컴파일러 설정 파일
```

4.1.2 frontend Dockerfile

1. 경량화하여 이미지를 만들도록 설정함.
2. “next.config” 파일 수정

```
const nextConfig={
....
output: 'standalone',
....
}
```

```
# Dockerfile

FROM node:18-alpine AS base

# Install dependencies only when needed
FROM base AS deps
RUN apk add --no-cache libc6-compat
WORKDIR /app

# Install dependencies based on the preferred package manager
COPY package*.json ./
RUN npm install --force
RUN npm install sharp
RUN rm -rf ./.next/cache

# Rebuild the source code only when needed
FROM base AS builder
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY . .
RUN npm run build

# Production image, copy all the files and run next
FROM base AS runner
WORKDIR /app

ENV NODE_ENV=production
ENV PORT 5000

RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nextjs

COPY --from=builder /app/public ./public/
COPY --from=builder --chown=nextjs:nodejs /app/.next/standalone ./
COPY --from=builder --chown=nextjs:nodejs /app/.next/static ./.next/static

USER nextjs
```

```
EXPOSE 5000

CMD ["node", "server.js"]
```

4.1.3 .dockerignore 파일

```
node_modules
```

4.2 Backend

4.2.1 backend 루트 경로 파일구조

```
backend/
└── koala_back/
    │
    ├── .gradle/                # Gradle 빌드 도구와 관련된 캐시 및 파일이 저장되는 디
    │
    ├── .idea/                  # IDE(IntelliJ 등) 설정 디렉토리
    │
    ├── build/                  # 빌드된 파일들이 저장되는 디렉토리
    │
    ├── gradle/                 # Gradle Wrapper 관련 파일들이 위치한 디렉토리
    │
    ├── sql/                    # SQL 스크립트 파일들이 저장된 디렉토리
    │
    ├── src/                    # 애플리케이션의 소스 코드가 위치한 디렉토리
    │
    ├── .env                    # 환경 변수를 정의한 파일
    │
    ├── .gitignore              # Git에서 추적하지 않을 파일 및 디렉토리 목록
    │
    ├── build.gradle            # Gradle 빌드 스크립트 파일
    │
    ├── Dockerfile              # Docker 이미지를 빌드하기 위한 설정 파일
    │
    ├── gradlew                 # Gradle Wrapper 실행 파일 (Unix/Linux/Mac용)
    │
    ├── gradlew.bat             # Gradle Wrapper 실행 파일 (Windows용)
    │
    ├── passportKey.txt         # Passport 키를 저장하는 파일
    │
    ├── settings.gradle         # Gradle 설정 파일
    │
    └── README.md               # 프로젝트에 대한 설명과 정보를 담은 파일
```

4.2.2 backend Dockerfile

```
# Gradle 버전을 8.x로 변경
FROM gradle:8.2-jdk17 AS build
WORKDIR /app

# 프로젝트 전체 복사
COPY . .

# Gradlew 파일에 실행 권한 추가
RUN chmod +x ./gradlew

# Gradle 빌드 실행
RUN ./gradlew clean build

FROM bellsoft/liberica-openjdk-alpine:17
LABEL authors="SSAFY"

ARG JAR_FILE=build/libs/*.jar

# 빌드된 JAR 파일 복사
COPY --from=build /app/${JAR_FILE} app.jar

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "/app.jar"]
```