[Ain] 포팅 매뉴얼

목차

1. 개발 환경

A. 공통

i. Jira 9.12.8

ii. Notion 3.7.0

iii. Mattermost 5.8.0

iv. Git 2.45.1

v. GitLab 16.7.3

vi. Postman 11.1.0

vii. Figma

B. 프론트엔드

i. React 18.3.1

ii. Next js 14.2.2

iii. typescript 5.4.5

iv. Zustand 4.5.2

v. tailwind css 3.4.1

vi. styled-component 6.1.9

vii. scss 1.77.0

viii. HTML5

ix. CSS3

x. Visual Studio Code 1.85.1

C. 백엔드

i. Java 17.0.11

ii. Intellij Ultimate 2024.1

iii. Spring Boot 3.2.5

iv. Spring Data JPA 3.2.5

v. Spring Security 6.2.4

vi. JWT 0.12.3

vii. MySQL 8.3.0

viii. Redis 3.2.5

ix. Python 3.7.10

x. FastAPI 0.111.0

xi. Pycharm 2023.3.2

xii. OpenAI API GPT-4 Turbo

xiii. DALL E 3

D. 인프라

i. EC2

ii. S3

iii. Ubuntu 20.04.6

iv. Jenkins 2.456

v. Nginx 1.25.5

vi. Certbot

vii. Docker 26.1.0

2. 기술 스택

A. Development

i. Java & Spring Boot

ii. Python & FastAPI

iii. NextJs & React

B. Deployment

i. Gitlab & Jenkins Webhook

ii. Docker

iii. Nginx Reverse Proxy

iv. React Web Server

v. Spring Boot & Fast API Server

C. Sign up & Sign in

i. Java & Spring Boot

ii. NextJs & React

iii. Kakao Oauth

iv. Spring Security & JWT & Redis

D. EC2 Setting

i. Ubuntu

E. Nginx Setting

i. Docker

ii. Certbot & SSL/TLS & HTTPS

F. MySQL 접속 명령어

i. EC2 에 접속후 MySQL Docker Container에 접속

명령어: docker exec –it <컨테이너명> bash

ii. MySQL 계정 로그인 진행 (만들어둔 계정으로 로그인)

1. docker exec –it <컨테이너명> bash
2. mysql –u <유저명> -p
3. 패스워드 입력

G. Redis 접속 명령어

i. EC2에 접속 후 Redis Docker Container에 접속

명령어: docker exec –it <컨테이너명> bash

ii. EC2에서 Redis Client 접속 (패스워드 미적용 시)

명령어: docker exec –it <컨테이너명> redis-cli

iii. EC2에서 Redis Client 접속 (패스워드 적용 시)

명령어 docker exec –it <컨테이너명> redis-cli –a <패스워드>

3. 환경 변수 및 설정

A. Spring Boot

i. application.yml,

```yaml
spring:
  config:
    import:
      - classpath:/application-dev.yml
      - classpath:/application-s3.yml
      - classpath:/application-oauth.yml
      - classpath:/application-openai.yml
```

ii. application-dev.yml

```yaml
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://${EC2_ACCESS_IP}:3306/a1n?serverTimezone=Asia/Seoul&useUnicode=true&characterEncoding=utf8&useLegacyDatetimeCode=false
    username: ${MYSQL_USERNAME}
    password: ${MYSQL_PW}
  jpa:
    database: mysql
    database-platform: org.hibernate.dialect.MySQLDialect
    hibernate:
      ddl-auto: none
      naming:
        physical-strategy: org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
    properties:
      globally_quoted_identifiers: true
      hibernate:
        format_sql: true
        show_sql: true
  jwt:
    secret: ${JWT_SECRET_KEY}
    live:
      access: 3600000 # 1시간
      refresh: 1209600000 # 2주
  data:
    redis:
      host: ${EC2_ACCESS_IP}
      port: 6379

logging:
  level:
    com.ssafy.a1n: info
```

iii. application-s3.yml

```yaml
cloud:
  aws:
    s3:
      bucket: ain-bucket
    credentials:
      accessKey: ${BUCKET_ACCESSKEY}
      secretKey: ${BUCKET_SECRETKEY}
    region:
      static: ap-northeast-2
      auto: false
    stack:
      auto: false
spring:
  servlet:
    multipart:
      enabled: true
      max-file-size: 3MB
      max-request-size: 3MB
```

iv. application-oauth.yml

```yaml
spring:
  security:
    oauth2:
      client:
        registration:
          kakao:
            client-name: kakao
            client-id: ${KAKAO_OAUTH_CLIENT_ID}
            client-secret: ${KAKAO_OAUTH_CLIENT_SECRET}
            redirect-uri: https://myain.co.kr/api/login/oauth2/code/kakao
            authorization-grant-type: authorization_code
            client-authentication-method: client_secret_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
```

v. application-openai.yml

```
openai:
  chatgpt:
    ideal-person:
      assistant-id: ${IDEAL_PERSON_ASSISTANT_ID}
```

B. FastAPI

i. requirements.txt

```
openai==1.23.3
uvicorn~=0.29.0
starlette~=0.37.2
python-dotenv~=1.0.1
fastapi~=0.111.0
pydantic~=2.7.1
pillow~=10.3.0
requests~=2.31.0
rembg~=2.0.56
```

4. 배포 시 특이사항

A. EC2 서버에 Docker를 활용하여 MySQL을 설치

i. Docker hub 사이트에 존재하는 Mysql image pull 진행

명령어: (sudo) docker pull mysql

ii. docker run –name <컨테이너명> -d –p <로컬 포트>:<도커 포트> -e TZ=<지역명> -v <볼륨명>:<컨테이너 디렉토리 경로> --network <네트워크명> <이미지명>

B. EC2 서버에 Docker를 활용하여 Redis 설치

i. Docker hub 사이트에 존재하는 Redis image pull 진행

명령어: (sudo) docker pull redis

ii. docker run –name <컨테이너명> -d -p <로컬 포트>:<도커 포트> -e TZ=<지역명> -v <볼륨명>:<컨테이너 디렉토리 경로> --network <네트워크명> <이미지명>

C. EC2 서버에 Docker로 Spring Boot 배포

i. Dockerfile 작성

ii. docker run –name <컨테이너명> -d –network <네트워크명> -e TZ=<지역명> <이미지명>

D. EC2 서버에 Docker로 FastAPI 배포

i. Dockerfile 작성

ii. docker run --name <컨테이너명> -d --network <네트워크명> -e TZ=<지역명> <이미지명>

E. EC2 서버에 Docker로 React 설치

i. Dockerfile 작성

ii. docker run –name <컨테이너명> -d -v <볼륨명>:<컨테이너 디렉토리 경로> -e TZ=<지역명> --network <네트워크명> <이미지명>

F. EC2 서버에 Docker로 Jenkins 설치

i. Dockerfile 작성

ii. docker run --name <컨테이너명> -d -p <로컬 포트>:<도커 포트> -v <볼륨명>:<컨테이너 디렉토리 경로> -e TZ=<지역명> -u root <이미지명>

G. EC2 서버에 Docker로 Nginx 배포

i. Docker hub 사이트에 존재하고 있는 Nginx image pull 진행

(sudo) docker pull nginx

ii. docker run --name <컨테이너명> -d -p <로컬 포트>:<도커 포트> -v <볼륨명>:<컨테이너 디렉토리 경로> -e TZ=<지역명> --network <네트워크명> <이미지명>

H. Nginx 컨테이너에 HTTPS 설치

i. SSL 인증서 발급 진행

(sudo) apt-get update

(sudo) apt-get install vim

(sudo) apt-get install python3-certbot-nginx

ii. HTTPS 설정

1. /etc/nginx/conf.d 경로 내의 default.conf 파일 수정하여 도메인에 대한 https 설정 완료하기
2. 수정 후 nginx –s reload를 통해 nginx 재시작

5. 외부 서비스 정보

A. 카카오 소셜 로그인

i. 카카오 로그인 API 문서

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

ii. 카카오 개발자

https://developers.kakao.com/