제품 구성 및 배포 운영 절차

OSSProj 2조 Spring

1. 프로그램 구성

본 프로젝트는 Next.js 프레임워크와 Spring Boot를 기반으로 구축된 반응형 웹 애플리케이션으로, PC와 모바일 웹 환경을 지원합니다. 프로그램 구성은 크게 프론트엔드, 백엔드, 오픈소스 OCR 서버로 구성됩니다.

- **프론트엔드 (Next.js, React, TailwindCSS, TypeScript, Zustand, Framer motion)**

- frontend/public : 이미지 파일
- frontend/src/app/pages : 로그인, 착한 가게 탭, 커뮤니티 탭, 소비 인증, 지도 탭, 기부 탭 등 페이지 UI 구성
- frontend/src/components : 헤더, 모달, 버튼 등 공통 UI 컴포넌트
- frontend/src/store : 상태 관리를 위한 Zustand 구조 정의

- **백엔드 (Spring Boot, Spring Security, Spring JPA, MySQL)**

- controller / service / repository : RESTful API 제공 (회원, 가게, 기부 등)
- config / security: Spring Security 기반 인증 처리
- resources/static/images: 이미지 저장 및 제공 경로
- application.yml: 환경별 설정 파일 (MySQL 이미지, OCR 이미지, 내부 네트워크)
- Dockerfile: 백엔드 서버 컨테이너 구성
- docker-compose.yml: 백엔드+OCR 서버를 한 번에 실행할 수 있도록 정의

- **OCR 서버 (Flask, Tesseract OCR)**

- app.py: Flask 서버 엔트리포인트
- ocr_module.py : 이미지 내 텍스트 인식 및 필터링
- Dockerfile : 독립 OCR 서버 컨테이너 구성

2. 배포 대상 및 방법

- **프론트엔드**
  - 타겟: Netlify
  - 배포 주소 : https://donzzul.netlify.app
  - 방법

    Next.js의 동적 라우팅과 SSR(Server-Side Rendering) 기능을 Netlify에서 정상적으로 지원하기 위해 @netlify/plugin-nextjs 플러그인을 설치하였습니다.

    GitHub 저장소(2025-1-OSSProj-Spring-02)의 main 브랜치를 Netlify에 연동하여 자동으로 빌드 및 배포가 이루어지도록 설정하였습니다.

    백엔드와의 연결은 Netlify의 프록시(Proxy) 기능을 활용하여 EC2의 퍼블릭 IP를 API 경로에 매핑함으로써 통신이 가능하도록 구성하였습니다.

  - 환경 변수

    Netlify 환경 설정에서는 백엔드 서버(Spring Boot)와의 통신을 위해 NEXT_PUBLIC_API_URL 환경

변수를 등록하여 API 주소를 관리하였습니다.
또한, Kakao Maps API의 보안을 위해 NEXT_PUBLIC_KAKAO_API_KEY를 Netlify의 환경 변수로 설정하여 관리하였습니다.

- **백엔드**
  - 타겟: AWS EC2
  - 배포 주소 : http://13.125.255.84
  - 방법:
    AWS 콘솔에서 EC2 인스턴스를 생성한 뒤, 보안 그룹의 인바운드 규칙을 통해 80번 포트를 개방하였습니다. 백엔드 도커 이미지는 로컬 환경에서 빌드 후, docker tag 및 docker push 명령어를 사용하여 레지스트리에 푸시되었습니다. 이후 해당 EC2 인스턴스 내부에 리포지토리(repository)를 생성하고, docker-compose.yml 파일을 업로드하였습니다. 마지막으로, 업로드된 docker-compose.yml 파일에 명시된 이미지를 풀(pull)받아 컨테이너를 가동함으로써 배포를 완료하였습니다.
  - 환경 변수 설정: docker-compose.yml의 environment 항목

- **데이터베이스**
  - 타겟: AWS RDS (MySQL)
  - 설정: AWS RDS 인스턴스를 퍼블릭 액세스가 허용되도록 설정 후 생성하고, 네트워크 및 보안 설정으로 보안 그룹에서 3306포트를 허용했습니다. application-docker.yml에서 RDS 엔드포인트, DB이름과 사용자명, 비밀번호 등의 연결 정보를 입력했습니다.

- **한글 OCR 개발 환경**

본 프로젝트에서 사용되는 OCR 기능(영수증 인식 등)은 Tesseract OCR 기반으로 구현되었습니다.
해당 기능 테스트 및 사용을 위해 아래와 같은 환경 구성이 필요합니다.
1. Tesseract 설치
2. 한글 학습 데이터(kor.traineddata) 등록
3. 설치 경로:
- Windows: `C:\Program Files\Tesseract-OCR\tessdata`
- Ubuntu: `/usr/share/tesseract-ocr/4.00/tessdata`
4. 이용 방법: EC2에서 별도 container를 구동 중이므로 따로 설치 없이 이용이 가능합니다.

3. 운영 방법

- **프론트엔드**

Netlify는 GitHub 저장소와 연동된 CI/CD 파이프라인을 제공합니다.
본 프로젝트에서는 main 브랜치에 PR이 병합될 때마다 Netlify에서 자동으로 빌드 및 배포가 실행되도록 설정하여, 수동 개입 없이 안정적인 배포가 가능하도록 구성하였습니다.
최종 배포 주소는 https://donzzul.netlify.app 이며, 실제 사용자도 위 주소를 통해 별도 설치 없이 웹브라우저에서 서비스를 이용할 수 있습니다.
향후에도 추가 기능 개발 시 PR 및 Deploy Preview 기능을 활용해 검토 후 정식 배포할 예정입니다.

- **백엔드**

EC2 환경에서 docker-compose logs backend, docker-compose logs ocr을 통한 로그 확인을 통해서 서버 상태 모니터링이 가능합니다. 최종 배포 주소는 13.125.255.84이며, 서버에 이상이 있을 시 postman에서 매핑을 통해 어느 엔드포인트에 이상이 생겼는지 확인 가능합니다.
DB는 AWS RDS 콘솔을 통해 성능 모니터링, 자동 백업 확인이 가능합니다.