포팅메뉴얼

1. 시스템 환경 및 소프트웨어 정보

1.1 백엔드 (Spring Boot)

- **프레임워크**: Spring Boot
- **버전**: 3.3.3
- **JVM 버전:** Azul Zulu 17.0.11
- **빌드 도구**: Gradle
- **IDE**: IntelliJ IDEA
- **IDE 버전**: 2024.1.4

1.2 프론트엔드 (React.js)

- **프레임워크**: React.js
- **버전**: 18.3.1
- **빌드 도구**: npm
- **IDE**: Visual Studio Code 1.90.2

1.2.1 라이브러리

- **emotion**: 11.13.0
- **axios**: 1.7.7
- **styled-components**: 6.1.13
- **chart.js**: 4.4.4
- **react-kakao-maps-sdk**: 1.1.27
- **jsbarcode**: 3.11.6

1.3 웹 서버

- **종류: Nginx**

- **역할:** Reverse Proxy로서 요청을 앤드포인트에 따라 백엔드, 프론트엔드로 구분하여 처리

1.4 WAS

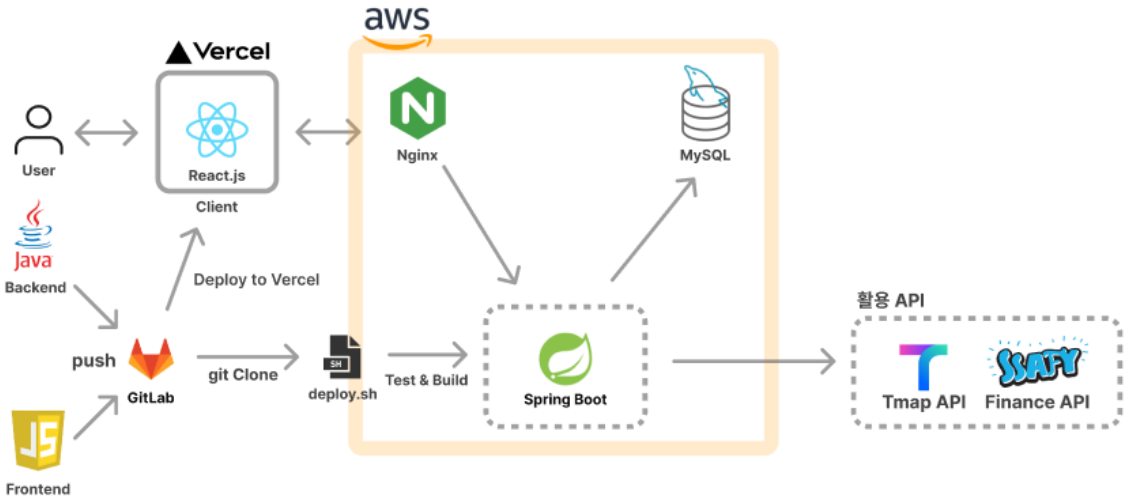
- **설정 파일:** `application.properties`, `info.properties`
  - info.properties 내용

```
spring.datasource.url=
spring.datasource.username=
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.D

api.key=
institution.code=
fintech.app.no=
fintech.api.url=


jwt.expiration=
spring.jwt.secret=
```

- **민감정보를 Info.properties로 관리하였습니다.**
- **종류: Spring Boot 3.0.0 내장 Tomcat**
- **포트:** 8080 (내장 Tomcat 기본 포트)

1.5 인프라 구성

1.5.1 구조



1.5.2 EC2 인스턴스

- **역할:** Nginx와 서비스 서버(Spring Boot,Next.js)의 실행
- **설치된 소프트웨어:** JDK17, Certbot, Nginx, Node

1.5.3 RDS (Relational Database Service)

- **DB 종류:** Mysql
- **버전:** 8.0.37
- **접속정보**
  - **URL:**
  - **Username:**
  - **Password:**

2. 빌드 및 배포

2.1 소스 클론

- GitLab에서 소스를 클론합니다

```
git clone https://lab.ssafy.com/s11-fintech-finance-sub
1/S11P21A107.git
```

- 프론트엔드와 백엔드의 root 폴더로 이동합니다

```
#frontend root folder
cd frontend

#backend root folder
cd backend
```

2.1.1 프론트엔드 추가 설정

- /frontend/cardmore로 이동합니다.

```
#frontend cardmore folder
cd frontend/cardmore
```

- .env 파일이 없다면 만들고 내부에 REACT_APP_KAKAO_API_KEY에 본인의 kakao api 키를 넣습니다.

```
/.env 파일

REACT_APP_KAKAO_API_KEY ="******************************
***"
```

2.1.2 백엔드 추가 설정

- /backend/src/main/resources로 이동합니다.
- info.properties 파일을 넣어줍니다. 파일에 대한 정보는 상단에 명시해두었습니다.

2.2 빌드 과정

- (Backend) Gradle을 사용하여 빌드를 수행합니다:

```
./gradlew clean build
```

- (Frontend) npm을 사용하여 빌드를 수행합니다.

```
./npm run build
```

2.3 배포 과정

백엔드

- 변경된 프로젝트를 master branch에 push 합니다.
- code push 후 Ec2내의 `deploy.sh` 파일을 실행합니다.

```
# 1. 기존 서버의 프로세스 ID 찾기
PID=$(lsof -t -i:8080)

set -e  # 명령어가 실패하면 스크립트를 종료

# 2. 기존 서버 종료
if [ -n "$PID" ]; then
    echo "Stopping existing server with PID: $PID"
    kill -9 $PID
else
    echo "No server is currently running."
fi

# 3. 최신 코드 가져오기
cd cardmore/backend/backend || { echo "Failed to navigate to
echo "Pulling latest code from git..."
git pull || { echo "Git pull failed"; exit 1; }

# 4. 프로젝트 빌드
echo "Building the project..."
sudo ./gradlew clean build || { echo "Build failed"; exit 1;

# 5. 새 서버 실행
echo "Starting new server..."
nohup java -jar build/libs/cardmore-0.0.1-SNAPSHOT.jar 1>outp
echo "New server started."
```

- 위 쉘 스크립트는 새로운 코드를 pull, 빌드 후 기존의 서버를 종료하고 서버를 새로 시작합니다.

프론트엔드

- 변경된 프로젝트를 master branch에 push 합니다.
- code push 후 Ec2내의 `frontend_deploy.sh` 파일을 실행합니다.

```
# 1. 기존 서버의 프로세스 ID 찾기
PID=$(lsof -t -i:8080)

set -e  # 명령어가 실패하면 스크립트를 종료

# 2. 기존 서버 종료
if [ -n "$PID" ]; then
    echo "Stopping existing server with PID: $PID"
    kill -9 $PID
else
    echo "No server is currently running."
fi

# 3. 최신 코드 가져오기
cd cardmore/backend/backend || { echo "Failed to navigate to
echo "Pulling latest code from git..."
git pull || { echo "Git pull failed"; exit 1; }

# 4. 프로젝트 빌드
echo "Building the project..."
sudo ./gradlew clean build || { echo "Build failed"; exit 1;

# 5. 새 서버 실행
echo "Starting new server..."
nohup java -jar build/libs/cardmore-0.0.1-SNAPSHOT.jar 1>outp
echo "New server started."
ubuntu@ip-172-26-4-252:~$ ^C
ubuntu@ip-172-26-4-252:~$ cat frontend_deploy.sh
#!/bin/bash
```

```
cd cardmore/frontend/frontend/cardmore

echo "Pulling latest code from git..."

git restore .

git pull origin frontend || { echo "Git pull failed"; exit 1;

npm run build
```