주마등

포팅 매뉴얼

구미 2반 7조 희멘

강교철, 김현진, 박희종, 임유정, 임지원, 하진우

목차

I. 개요

1. 프로젝트 개요

'주마등'은 '주식을 마주하는 등대'의 줄임말입니다. 등대가 어두운 바다에 빛을 밝혀 배가 안전하게 항해할 수 있도록 돕듯이, 이용자가 안전하게 주식 시장에 입문할 수 있도록 돕고자 하는 마음을 담았습니다.

2020년 COVID-19 팬데믹 이후 자산 가격이 급등하면서 국민 5명 중 한 명이 주식을 시작했으며, 그 중 절반이 2030 세대였다고 합니다. 당시 2030 투자자 사이에서는 '빚투'와 '영끌'이 유행했습니다. COVID-19가 종식되어 가고 있는 2023년 현재, 결과는 어떨까요? 한 기사에 따르면, '빚투'·'영끌' 했던 청년 4~5명 중 한 명은 소득의 3배에 달하는 부채를 안고 있다고 합니다.

저희는 주식을 처음 시작하는 2030 또래들이 쉽고, 안정적인 방법으로 주식 시장에 참여할 수 있는 방법을 안내합니다.

2. 주요 기술

Backend - Spring

IntelliJ IDE

Springboot Gradle 7.4

Java jdk corretto 11.0.17

Spring Data JPA

Springframework 2.7.9

Spring Security

Spring Validation

Frontend - React

Visual Studio Code IDE 1.74.2

Nodejs 18.12.1

React 18.2.0

zustand 4.3.6

typescript 4.9.5

TailwindCss 3.2.7

Spring Web

Swagger 3.0.0

Lombok

spark-core 2.12:3.3.2

spark-sql 2.12:3.2.3

jjwt 0.11.2

Backend - DB

MariaDB 10.11.2

CI/CD

AWS EC2

- Ubuntu 20.04

- Docker 23.0.1

Jenkins

NGINX

SSL

3. 사용한 외부 서비스 목록

- 소셜 로그인
  - Kakao : OAuth 기반 소셜 로그인 API 제공
  - https://developers.kakao.com/

II. 프론트 엔드 빌드

1. Git clone

1. Clone 시 Master로 Clone 하기

```
https://lab.ssafy.com/s08-bigdata-dist-sub2/S08P22D207.git
```

2. Clone 받은 폴더로 이동

```
cd S08P22D207
```

3. Branch를 front-dev으로 변경

```
git checkout –track origin/front-dev
```

2. Dockerfile, nginx.conf 파일 작성

Dockerfile

```
FROM nginx:stable-alpine

WORKDIR /app

RUN mkdir ./build

ADD ./build ./build

RUN rm /etc/nginx/conf.d/default.conf

COPY ./nginx.conf /etc/nginx/conf.d

EXPOSE 3000


CMD ["nginx", "-g", "daemon off;"]
```

nginx.conf

```
server {
    listen 3000;
    location / {
        root /app/build;
        index index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

3. 빌드

프로젝트 폴더에 있는 frontend 디렉토리의 루트 경로에서 다음과 같은 명령어를 입력합니다

```
npm i
npm run build
```

4. 배포

```
docker build -t [이미지명]:[태그명] .
docker run --name [컨테이너명] -d -p 3000:80 [이미지명]:[태그명]
```

III. 백엔드 빌드

1. Git clone

1. Clone 시 Master로 Clone 하기

```
2. https://lab.ssafy.com/s08-bigdata-dist-sub2/S08P22D207.git
```

3. Clone 받은 폴더로 이동

```
cd S08P22D207
```

4. Branch를 back-dev으로 변경

```
git checkout –track origin/back-dev
```

2. 카카오 API 관련 설정

1. Kakao Developers(https://developers.kakao.com/) 접속 및 로그인
2. [내 애플리케이션] → [애플리케이션 추가하기]
3. 'Client Id' 복사, *application.yml* 파일 *'client-id* 위치에 추가
4. [앱설정] → [플랫폼] → Web 플랫폼 등록 → 사이트 도메인 추가

```
https://{도메인}
```

5. Redirect URI 등록

```
https://{도메인}/login/oauth2/code/kakao
```

6. [앱설정] → [동의항목] → *닉네임, 프로필 사진, 카카오계정(이메일)*
7. [앱설정] → [보안] → Client Secret 코드 생성
8. 발급받은 Secret 코드를 *application.yml* 파일 *'client-secret'* 위치에 추가

3. application.yml 파일 작성

```yaml
server:
  port: 8888
spring:
  profiles:
    active: dev
  mvc:
    pathmatch:
      matching-strategy: ant-path-matcher
  servlet:
    multipart:
      maxFileSize: 40MB
      maxRequestSize: 100MB
  data:
    web:
      pageable:
        one-indexed-parameters: true
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: {DB주소}
    username: {DB username}
    password: {DB password}
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        show_sql: true #System.out.println \uC744 \uD1B5\uD574
\uC2E4\uD589\uB418\uB294 \uCFFC\uB9AC \uCD9C\uB825
        format_sql: true # \uCFFC\uB9AC\uAC00 \uD55C\uC904\uB85C
\uB098\uC624\uB294\uAC8C \uC544\uB2C8\uB77C \uBCF4\uAE30
\uD3B8\uD558\uB3C4\uB85D \uD3EC\uB9E4\uD305 \uD574\uC90C
```

```
        default_batch_fetch_size: 100
  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: {카카오 client id}
            client-secret:{카카오 client secret}
            redirect-uri: https://{도메인}/login/oauth2/code/kakao
            authorization-grant-type: authorization_code
            client-authentication-method: POST
            client-name: Kakao
            scope:
              - profile_nickname
              - profile_image
              - account_email
        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
JWT:
  SECRET: {JWT secret_key}

logging:
  level:
    org.hibernate.SQL: trace
    org.hibernate.type: trace
```

4. Dockerfile 작성

Dockerfile

```
FROM openjdk:11-jdk-slim

ARG JAR_FILE=build/libs/*.jar

COPY ${JAR_FILE} app.jar

EXPOSE 8888


ENTRYPOINT ["java", "-jar", "/app.jar","-
Dspring.profiles.active=server"]
```

5. 빌드

1. GUI 이용시 (IntelliJ IDEA 2022.3.1 Ultimate Edition 기준)
   1) *Gradle* 선택
   2) *BeconOfStock/Tasks/build/Clean* 더블 클릭
   3) *BeconOfStock/Tasks/build/BootJar* 더블 클릭
2. Command 사용시

   프로젝트 폴더 내에 있는 *backend* 디렉토리의 루트 경로에서 다음의 명령어를 실행합니다.

```
gradle clean build
```

6. 배포

```
docker build -t [이미지명]:[태그명] .

docker run --name [컨테이너명] -d -p 8080:8080 [이미지명]:[태그명]
```

IV. 서버 세팅

1. EC2 세팅

1. TimeZone 설정

```
SET GLOBAL time_zone='ASIA/SEOUL';

SET time_zone='+09:00';

flush privileges;

date
```

2. EC2 서버에 docker 설치

1. apt를 이용하여 docker를 설치할 예정이라 apt를 update 합니다.

```
sudo apt update
```

2. docker 설치에 필요한 패키지들을 설치합니다.

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

3. curl을 이용, 도커를 설치하기 위한 내용을 다운로드 받고, apt 기능을 위한 리스트에 추가합니다.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

4. ubuntu 18.04 버전에 맞는 docker를 다운로드 할 수 있도록 repository 리스트에 추가합니다.

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
```

5. apt update를 실행합니다.

```
sudo apt update
```

→ apt list에 도커를 다운로드 할 경로가 업데이트 되었습니다.

6. docker-ce를 설치합니다.(커뮤니티 버전)

```
apt-cache policy docker-ce

sudo apt install docker-ce
```

7. docker가 설치되면, 자동으로 시스템 서비스로서 등록이 됩니다.

   systemctl 명령어를 통해 docker 서비스 상태를 확인해 보면, 도커엔진이 구동중인 상태 확인

```
sudo systemctl status docker
```

3. EC2 서버에 MariaDB 설치

1. MariaDB 설치

```
sudo docker pull mariadb
```

2. 3306 포트로 실행

```
sudo docker run -p 3306:3306 --name mariadb -e
MARIADB_ROOT_PASSWORD=[비밀번호] -d mariadb
```

- *-name* : mariadb 라는 컨테이너 이름을 부여
- *p 3306:3306* : host port number:container port number
- *e* : -e는 환경 변수 옵션이다.

  e MARIADB_ROOT_PASSWORD=비밀번호: 비밀번호를 지정
- *d* : detached 모드에서 실행되어 백그라운드로 실행한다.
- *mariadb* : 앞서 받은 이미지 이름

4. DB bash 접속, 계정 생성

1. bash 접속

```
sudo docker exec -it mariadb bash
```

2. DB 버전확인

```
mysql --version
```

3. DB 접속

```
mysql -u root -p
```

4. 사용자 추가 : localhost에서만 접속 가능한 계정 생성

```
# mysql>

use mysql;


# CREATE USER 'YOUR_SYSTEM_USER'@'localhost' IDENTIFIED BY
'YOUR_PASSWD';

create user 'localhost'@'localhost' identified by '비밀번호';


# GRANT ALL PRIVILEGES ON *.* TO 'YOUR_SYSTEM_USER'@'localhost';

grant all privileges on *.* to 'localhost'@'localhost';


flush privileges;
```

5. 모든 DB, 테이블에 접속 가능한 계정 생성

```
USE mysql;

CREATE USER 'localhost'@'%' IDENTIFIED BY 'd110';

GRANT ALL PRIVILEGES ON *.* TO 'localhost'@'%';
```

```
FLUSH PRIVILEGES;



USE mysql;

CREATE USER 'user'@'%' IDENTIFIED BY 'd110';

GRANT ALL PRIVILEGES ON *.* TO 'user'@'%';

FLUSH PRIVILEGES;
```

6. 데이터베이스 생성

```
create database [데이터베이스명]
```

7. SSH접속과 외부 접속 허용

```
vi /etc/mysql/mariadb.conf.d/50-server.cnf

# bind-address = 0.0.0.0

# 추가해주기 :wq 로 저장 후 나가기
```

8. MariaDB characterset 변경

```
# apt 업데이트

apt-get update



# vim 설치

apt-get install vim



vi /etc/mysql/my.cnf
```

9. *my.cnf*에 아래 내용 추가

```
[client]

default-character-set=utf8mb4
```

```
[mysql]
default-character-set=utf8mb4

[mysqld]
character-set-server=utf8mb4
collation-server=utf8mb4_unicode_ci
skip-character-set-client-handshake
```

5. Nginx Default 값

1. 서버 Default 값 설정

Server

```
server {
        location /{
                proxy_pass http://localhost:3000;
                proxy_connect_timeout 600;
                proxy_send_timeout 600;
                proxy_read_timeout 600;
                send_timeout 600;
        }

        location /api {
                proxy_pass http://localhost:8888;
```

```
                proxy_connect_timeout 600;
                proxy_send_timeout 600;
                proxy_read_timeout 600;
                send_timeout 600;
        }

        location /login{
                proxy_pass http://localhost:8888;
        }

        location ~ ^/(swagger¦webjars¦configuration¦swagger-
resources¦v2¦csrf) {
                proxy_pass http://localhost:8888;
                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;
        }

    listen 443 ssl; # managed by Certbot
    ssl_certificate
/etc/letsencrypt/live/j8d207.p.ssafy.io/fullchain.pem; # managed by
Certbot
```

```
    ssl_certificate_key
/etc/letsencrypt/live/j8d207.p.ssafy.io/privkey.pem; # managed by
Certbot

    # include /etc/letsencrypt/options-ssl-nginx.conf; # managed by
Certbot

    # ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by
Certbot

}


server {

    if ($host = j8d207.p.ssafy.io) {

        return 301 https://$host$request_uri;

    } # managed by Certbot


        listen 80;

        server_name j8d207.p.ssafy.io;

    return 404; # managed by Certbot

}
```

2. Nginx 실행

```
sudo systemctl start nginx
```

3. Front Nginx

```
server {

  listen 80;

  location / {

    root /app/build;
```

```
    index index.html;
    try_files $uri $uri/ /index.html;
  }
}
```

V. 자동 배포 : Jenkins

1. Jenkins 플러그인 설치

Jenkins 관리 → 플러그인 관리 → Available plugins

1. SSH
   - Publish Over SSH
   - SSH Agent Plugin
2. GitLab
   - GitLab
   - Gitlab API Plugin
   - GitLab Authentication plugin
   - Generic Webhook Trigger Plugin
3. Docker
   - Docker API Plugin
   - Docker Commons Plugin
   - Docker Pipeline
   - Docker plugin

4. NodeJS
    - NodeJS Plugin

2. Jenkins 프로젝트 생성

1. 젠킨스 메인 페이지 → 새로운 Item → Freestyle project
2. Jenkins 관리 → Credentials → add credentials
    - gitlab
    - Docker hub
    - Ssh
3. webhook 설정

3. Backend 배포

```
cd /var/jenkins_home/S08P22D207

git checkout back-dev

git submodule update --remote

git pull origin back-dev

cd /var/jenkins_home/S08P22D207/backend/BeconOfStock

chmod +x gradlew


#./gradlew clean build

./gradlew build

docker build -t gyocheol/backend .
```

```
docker push gyocheol/backend

docker pull gyocheol/backend

docker rm -f backend ¦¦ true

docker run -d -p 8888:8888 -v
/jenkins/S08P22D207/backend/BeconOfStock:/var/jenkins_home/S08P22D207/b
ackend/BeconOfStock --name backend gyocheol/backend
```

4. Frontend 배포

```
cd /var/jenkins_home/front/S08P22D207

git checkout front-dev

git pull origin front-dev

cd /var/jenkins_home/front/S08P22D207/frontend/beacon-of-stock

rm -rf /var/jenkins_home/front/S08P22D207/frontend/beacon-of-
stock/build

npm run build
```