Donkeeper

포팅메뉴얼

목차

1. 개요

1. 프로젝트 개요

Don-Keeper 는 소비문화 개선을 위한 가계부 서비스입니다. 사용자의 소득구간을 설정하여 동일한 소득구간에 있는 다른 사용자들과 비교해 자신이 얼마나 소비하는지 확인할 수 있습니다. 카드 추천 기능과 제공하여 효율적으로 돈을 관리할 수 있도록 도와줍니다. 챌린지 서비스를 제공하여 사용자가 현명한 소비 습관을 만들도록 돕습니다.

2. 프로젝트 사용 도구

이슈 관리: JIRA

형상관리: Gitlab

커뮤니케이션: Notion, MatterMost

디자인: Figma

CI/CD: Jenkins

3. 개발 환경

VS CODE

INTELLIJ

JDK11

Spring Boot

Vue.js

SERVER: AWS EC2 Ubuntu 20.04.3 LTS

DB: MariaDB(azure)

4. 외부 서비스

Kakao OAuth: application.yml

2. 빌드

1. 환경변수

.env.local

VUE_APP_API_URL = 'http://localhost:8080/api'

VUE_APP_URL = 'http://localhost:3000'

.env

VUE_APP_API_URL = 'http://donkeeper.com/api'

VUE_APP_URL = 'http://donkeeper.com'

appication.yml

```
spring:
  profiles:
    active: dev
  devtools:
    livereload:
      enabled: true
  datasource:
    url: 마리아 DB 주소
    username: 유저 이름
    password: 유저 비밀번호
    driver-class-name: com.mariadb.jdbc.Driver
    hikari:
      maximum-pool-size: 10
      idle-timeout: 30000
      pool-name: MyHikariCP
  jpa:
    database: mysql
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    properties:
      hibernate:
        show_sql: true
        format_sql: true
        use_sql_comments: true
    hibernate:
      ddl-auto: update
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher
```

```
jwt:
  header: Authorization
  secret: 시크릿 키
  token-validity-in-milliseconds:
  refresh-token-validity-in-milliseconds:

kakao:
  rest-api-key: 카카오 api 키
  redirect-uri: http://donkeeper.com/kakaoCallback

logging:
  level:
    org:
      springframework:
        security: INFO

file:
  image:
    path: /app

server:
  servlet:
    session:
      timeout:
```

2. 빌드하기

1) spring-boot

Gradle 실행

2) vue.js

npm i

npm run serve

3. 배포하기

Nginx 설정

```
upstream backend{
        ip_hash;
        server 172.26.2.70:8080;
}

server {
    listen          80;
    listen  [::]:80;
    server_name  k8c209.p.ssafy.io;

    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        root    /usr/share/nginx/html;
        index  index.html index.htm;
        try_files $uri $uri/ /index.html;
    }

        location /api {
        proxy_pass http://backend;
        proxy_redirect      off;
        proxy_set_header    Host $host;
        proxy_set_header    X-Real-IP $remote_addr;
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
```

```
        root    /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    #location ~ ₩.php$ {
    #    proxy_pass    http://127.0.0.1;
    #}

    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    #location ~ ₩.php$ {
    #    root           html;
    #    fastcgi_pass   127.0.0.1:9000;
    #    fastcgi_index  index.php;
    #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
    #    include        fastcgi_params;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /₩.ht {
    #    deny  all;
    #}
}
```

Jenkins Excute shell

```
# back 폴더 빌드 및 실행
docker build -t backimg ./moneykeeperbackend
if (docker ps | grep "backimg"); then docker stop backimg; fi
docker run -it -d --rm -p 8080:8080 --name backimg backimg
echo "Run moneykeeperbackend"

# front 폴더 빌드 및 실행
docker build -t frontimg ./frontend
if (docker ps | grep "frontimg"); then docker stop frontimg; fi
docker run -it -d --rm -p 80:80 --name frontimg frontimg
echo "Run frontend"
```