포팅 메뉴얼

프론트 환경 변수

frontend/customer/.env

```
NEXT_PUBLIC_KAKAO_APP_JS_KEY=카카오맵API키
NEXT_PUBLIC_KAKAO_APP_RESTAPI_KEY=카카오맵RESTAPIKEY

NEXT_PUBLIC_FIREBASE_APIKEY=파이어베이스APIKEY
NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN=파이어베이스인증도메인
NEXT_PUBLIC_FIREBASE_PROJECTID=파이어베이스아이디
NEXT_PUBLIC_FIREBASE_STORAGEBUCKET=파이어베이스버킷
NEXT_PUBLIC_FIREBASE_MESSAGINGSENDERID=파이어베이스메세지
NEXT_PUBLIC_FIREBASE_APPID =파이어베이스앱ID
```

frontend/owner/.env

```
NEXT_PUBLIC_KAKAO_APP_JS_KEY=카카오맵API키
NEXT_PUBLIC_KAKAO_APP_RESTAPI_KEY=카카오맵RESTAPIKEY

NEXT_PUBLIC_FIREBASE_APIKEY=파이어베이스APIKEY
NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN=파이어베이스인증도메인
NEXT_PUBLIC_FIREBASE_PROJECTID=파이어베이스아이디
NEXT_PUBLIC_FIREBASE_STORAGEBUCKET=파이어베이스버킷
NEXT_PUBLIC_FIREBASE_MESSAGINGSENDERID=파이어베이스메세지
NEXT_PUBLIC_FIREBASE_APPID =파이어베이스앱ID
```

프론트 docker

customer

빌드

```
docker build -t sff-customer .
docker-compose up -d
```

docker-compose.yml

```
version: "3"
services:
  sff-customer:
    image: sff-customer
    environment:
      - NEXT_PUBLIC_API_URL=gatewayURL
      - NEXT_PUBLIC_KAKAO_APP_JS_KEY=카카오맵API키
      - NEXT_PUBLIC_KAKAO_APP_RESTAPI_KEY=카카오맵RESTAPIKEY
      - NEXT_PUBLIC_FIREBASE_APIKEY=파이어베이스APIKEY
      - NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN=파이어베이스인증도메인
      - NEXT_PUBLIC_FIREBASE_PROJECTID=파이어베이스아이디
      - NEXT_PUBLIC_FIREBASE_STORAGEBUCKET=파이어베이스버킷
      - NEXT_PUBLIC_FIREBASE_MESSAGINGSENDERID=파이어베이스메세지
      - NEXT_PUBLIC_FIREBASE_APPID =파이어베이스앱ID
    ports:
      - "3000:3000"
```

owner

빌드

```
docker build -t sff-owner .
docker-compose up -d
```

docker-compose.yml

```
version: "3"
services:
  sff-owner:
    image: sff-owner
    environment:
      - NEXT_PUBLIC_API_URL=gatewayURL
      - NEXT_PUBLIC_KAKAO_APP_JS_KEY=카카오맵API키
      - NEXT_PUBLIC_KAKAO_APP_RESTAPI_KEY=카카오맵RESTAPIKEY
      - NEXT_PUBLIC_FIREBASE_APIKEY=파이어베이스APIKEY
      - NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN=파이어베이스인증도메인
      - NEXT_PUBLIC_FIREBASE_PROJECTID=파이어베이스아이디
      - NEXT_PUBLIC_FIREBASE_STORAGEBUCKET=파이어베이스버킷
      - NEXT_PUBLIC_FIREBASE_MESSAGINGSENDERID=파이어베이스메세지
      - NEXT_PUBLIC_FIREBASE_APPID =파이어베이스앱ID
    ports:
      - "4000:3000"
```

DB

docker-compose.yml

```
version: "3.1"

services:
  sff_db:
    image: mysql:8.1
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: k9e205
      MYSQL_USER: sff
      MYSQL_PASSWORD: k9e205
      # TZ 환경 변수로 시간대 설정
      TZ: Asia/Seoul
      MYSQL_CHARSET: utf8mb4
      MYSQL_COLLATION: utf8mb4_unicode_ci
    ports:
      - "3306:3306"
    volumes:
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
```

init.sql

```
CREATE DATABASE IF NOT EXISTS `pay`;
CREATE DATABASE IF NOT EXISTS `member`;
CREATE DATABASE IF NOT EXISTS `orders`;
CREATE DATABASE IF NOT EXISTS `noti`;
CREATE DATABASE IF NOT EXISTS `store`;
CREATE DATABASE IF NOT EXISTS `owner`;

GRANT ALL PRIVILEGES ON `pay`.* TO 'sff'@'%';
GRANT ALL PRIVILEGES ON `member`.* TO 'sff'@'%';
GRANT ALL PRIVILEGES ON `orders`.* TO 'sff'@'%';
GRANT ALL PRIVILEGES ON `noti`.* TO 'sff'@'%';
GRANT ALL PRIVILEGES ON `store`.* TO 'sff'@'%';
GRANT ALL PRIVILEGES ON `owner`.* TO 'sff'@'%';

FLUSH PRIVILEGES;
```

Backend

config-server

docker 빌드 명령어

```
docker build -t sff-config-server .
```

docker-compose.yml

```
version: "3"

services:
  # 서비스 명
  config-server:
    image: sff-config-server
    ports:
      - "9000:9000"
    environment:
      - SPRING_PROFILES_ACTIVE=dev
```

eureka-server

docker 빌드 명령어

```
docker build -t sff-eureka-server .
```

docker-compose.yml

```
version: "3"

services:
  # 서비스 명
  eureka-server:
    image: sff-eureka-server
    ports:
      - "8761:8761"
    environment:
      - SPRING_PROFILES_ACTIVE=dev
```

gateway-server

docker 빌드 명령어

```
docker build -t sff-gateway-server .
```

docker-compose.yml

```
# docker-compose.yml
version: "3"

services:
  gateway-server:
    image: sff-gateway-server
    ports:
      - "8000:8000"
    environment:
      - SPRING_PROFILES_ACTIVE=dev
      - SECURITY_ID=admin
      - SECURITY_PASSWORD=1234
      - CONFIG_SERVER=http://admin:1234@서버아이피:9000
    volumes:
      - ./logs:/app/logs
```

application-prd.yml

```
#spring:
#  profiles:
#    active: dev
#  config:
#    import: optional:configserver:${CONFIG_SERVER}
#  application:
#    name: gateway

server:
  port: 8000

eureka:
  instance:
    prefer-ip-address: true
    ip-address: k9e205.p.ssafy.io
    non-secure-port: 8000
  client:
    enabled: true
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://admin:1234@k9e205.p.ssafy.io:8761/eureka/

spring:
  application:
    name: apigateway
  cloud:
```

```
    config:
      enabled: false
    gateway:
      globalcors:
        cors-configurations:
          '[/**]': # 이 패턴은 모든 경로에 CORS 설정을 적용합니다.
            allowed-origins: "*" # 모든 오리진 허용
            allowed-methods: # 허용할 HTTP 메소드 목록
              - GET
              - POST
              - PUT
              - DELETE
              - OPTIONS
            allowed-headers: "*" # 모든 헤더 허용
            allow-credentials: false # 쿠키 및 인증 정보 포함 허용
            exposedHeaders:
              - Authorization
              - Authorization-Refresh
            max-age: 3600 # 브라우저가 preflight 응답을 캐싱하는 시간 (초 단위)

      default-filters:
        - DedupeResponseHeader=Access-Control-Allow-Origin Access-Control-Allow-Creden
tials
        - name: GlobalFilter
          args:
            baseMessage: Spring Cloud Gateway Global Filter
            preLogger: true
            postLogger: true
      routes:
        - id: user-service
          uri: lb://USERSERVER
          predicates:
            - Path=/api/user-server/**
        - id: order-service
          uri: lb://ORDERSERVER
          predicates:
            - Path=/api/order-server/**
        - id: owner-service
          uri: lb://OWNERSERVER
          predicates:
            - Path=/api/owner-server/**
        - id: store-service
          uri: lb://STORESERVER
          predicates:
            - Path=/api/store-service/**
        - id: pay-service
          uri: lb://PAYSERVER
          predicates:
            - Path=/api/payment-server/**
        - id: noti-service
          uri: lb://NOTISERVER
          predicates:
            - Path=/api/noti-server/**
        - id: second-service
          uri: lb://MY-SECOND-SERVICE
          predicates:
```

```
            - Path=/second-service/**
          filters:
            - name: CustomFilter
            - name: LoggingFilter
              args:
                baseMessage: Hi, there.
                preLogger: true
                postLogger: true

custom:
  ignore-urls:
    - "/api/user-server/sign-up"
    - "/api/user-server/login"
    - "/api/user-server/jwt"
    - "/api/owner-server/sign-up"
    - "/api/owner-server/login"
    - "/api/owner-server/jwt"

app:
  jwtSecret: lk2jelk32rjfo23ijrf091u013jnfoi431jt01h08ryh0193u45091fj09ew8rjodsajflkas
dj120834uj309u450938509djfslkdfasjlakfdsjalks
  jwtExpirationMs: 100000

logging:
  config: classpath:logback.xml
  level:
    root: info
```

noti-server

docker 빌드 명령어

```
docker build -t sff-noti-server .
```

docker-compose.yml

```
# docker-compose.yml
version: "3"

services:
  noti-server:
    image: sff-noti-server
    ports:
      - "8120:8120"
    environment:
      - SPRING_PROFILES_ACTIVE=prd
      - SECURITY_ID=admin
      - SECURITY_PASSWORD=1234
```

```
      - CONFIG_SERVER=http://admin:1234@서버아이피:9000
      - DB_SERVER=디비서버아이피:3306
      - DB_NAME=noti
      - DB_ID=sff
      - DB_PASSWORD=k9e205
```

application-prd.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://${DB_SERVER}/${DB_NAME}?useSSL=false&allowPublicKeyRetrieval=tru
e&serverTimezone=Asia/Seoul&characterEncoding=UTF-8
    username: ${DB_ID}
    password: ${DB_PASSWORD}

  jpa:
    hibernate:
      ddl-auto: update
logging:
  config: classpath:logback.xml
  level:
    root: info
server:
  port: 8120
eureka:
  instance:
    prefer-ip-address: true
    ip-address: 서버아이피
    non-secure-port: 8120
  client:
    enabled: true
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://admin:1234@서버아이피:8761/eureka/
```

owner-server

docker 빌드 명령어

```
docker build -t sff-owner-server .
```

docker-compose.yml

```
# docker-compose.yml
version: "3"

services:
  owner-server:
    image: sff-owner-server
    ports:
      - "8140:8140"
    environment:
      - SPRING_PROFILES_ACTIVE=prd
      - SECURITY_ID=admin
      - SECURITY_PASSWORD=1234
      - CONFIG_SERVER=http://admin:1234@서버아이피:9000
      - DB_SERVER=디비서버아이피:3306
      - DB_NAME=owner
      - DB_ID=sff
      - DB_PASSWORD=k9e205
```

application-prd.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://${DB_SERVER}/${DB_NAME}?useSSL=false&allowPublicKeyRetrieval=tru
e&serverTimezone=Asia/Seoul&characterEncoding=UTF-8
    username: ${DB_ID}
    password: ${DB_PASSWORD}

  jpa:
    hibernate:
      ddl-auto: update

logging:
  config: classpath:logback.xml
  level:
    root: info
server:
  port: 8140
eureka:
  instance:
    prefer-ip-address: true
    ip-address: 서버아이피
    non-secure-port: 8140
  client:
    enabled: true
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://admin:1234@서버아이피:8761/eureka/
jwt:
  secretKey: lk2jelk32rjfo23ijrf091u013jnfoi431jt01h08ryh0193u45091fj09ew8rjodsajflkas
dj120834uj309u450938509djfslkdfasjlakfdsjalks
```

```
  access:
    expiration: 1209600000
    header: Authorization
  refresh:
    expiration: 1209600000
    header: Authorization-Refresh
```

pay-server

docker 빌드 명령어

```
docker build -t sff-pay-server .
```

docker-compose.yml

```
# docker-compose.yml
version: "3"

services:
  pay-server:
    image: sff-pay-server
    ports:
      - "8110:8110"
    environment:
      - SPRING_PROFILES_ACTIVE=prd
      - SECURITY_ID=admin
      - SECURITY_PASSWORD=1234
      - CONFIG_SERVER=http://admin:1234@서버아이피:9000
      - DB_SERVER=DB서버아이피:3306
      - DB_NAME=pay
      - DB_ID=sff
      - DB_PASSWORD=k9e205
```

application-prd.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://${DB_SERVER}/${DB_NAME}?useSSL=false&allowPublicKeyRetrieval=tru
e&serverTimezone=Asia/Seoul&characterEncoding=UTF-8
    username: ${DB_ID}
    password: ${DB_PASSWORD}
  jpa:
    hibernate:
      ddl-auto: update
```

```
logging:
  config: classpath:logback.xml
  level:
    root: info
server:
  port: 8110
eureka:
  instance:
    prefer-ip-address: true
    ip-address: 서버아이피
    non-secure-port: 8110
  client:
    enabled: true
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://admin:1234@서버아이피:8761/eureka/

feign:
  storeserver:
    url: 서버아이피:8090
  userserver:
    url: 서버아이피:8100
  payserver:
    url: 서버아이피:8110
  notiserver:
    url: 서버아이피:8120
  orderserver:
    url: 서버아이피:8130
  ownerserver:
    url: 서버아이피:8140
```

store-server

docker 빌드 명령어

```
docker build -t sff-store-server .
```

docker-compose.yml

```
# docker-compose.yml
version: "3"

services:
  store-server:
    image: slbin/sff-store-server
    ports:
```

```
      - "8090:8090"
    environment:
      - SPRING_PROFILES_ACTIVE=prd
      - SECURITY_ID=admin
      - SECURITY_PASSWORD=1234
      - CONFIG_SERVER=http://admin:1234@서버아이피:9000
      - DB_SERVER=DB서버아이피:3306
      - DB_NAME=store
      - DB_ID=sff
      - DB_PASSWORD=k9e205
```

application-prd.yml

```
spring:
  config:
    import: optional:configserver:${CONFIG_SERVER}
  application:
    name: storeserver
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://${DB_SERVER}/${DB_NAME}?useSSL=false&allowPublicKeyRetrieval=tru
e&serverTimezone=Asia/Seoul&characterEncoding=UTF-8
    username: ${DB_ID}
    password: ${DB_PASSWORD}

  jpa:
    hibernate:
      ddl-auto: update
logging:
  config: classpath:logback.xml
  level:
    root: info
server:
  port: 8090
eureka:
  instance:
    prefer-ip-address: true
    ip-address: 서버아이피
    non-secure-port: 8090
  client:
    enabled: true
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://admin:1234@서버아이피:8761/eureka/
```

user-server

docker 빌드 명령어

```
docker build -t sff-user-server .
```

docker-compose.yml

```
# docker-compose.yml
version: "3"

services:
  user-server:
    image: slbin/sff-user-server
    ports:
      - "8100:8100"
    environment:
      - SPRING_PROFILES_ACTIVE=prd
      - SECURITY_ID=admin
      - SECURITY_PASSWORD=1234
      - CONFIG_SERVER=http://admin:1234@서버아이피:9000
      - DB_SERVER=DB서버아이피:3306
      - DB_NAME=member
      - DB_ID=sff
      - DB_PASSWORD=k9e205
```

application-prd.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://${DB_SERVER}/${DB_NAME}?useSSL=false&allowPublicKeyRetrieval=tru
e&serverTimezone=Asia/Seoul&characterEncoding=UTF-8
    username: ${DB_ID}
    password: ${DB_PASSWORD}

  jpa:
    hibernate:
      ddl-auto: update

  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: 카카오 클라이언트id
            client-secret: 클라이언트 시크릿키
            redirect-uri: 리다이렉트 rui
            client-authentication-method: client_secret_post
            authorization-grant-type: authorization_code
            scope: profile_nickname, profile_image
            client-name: Kakao
```

```
        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

logging:
  config: classpath:logback.xml
  level:
    root: info
server:
  port: 8100
eureka:
  instance:
    prefer-ip-address: true
    ip-address: 서버주소
    non-secure-port: 8100
  client:
    enabled: true
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://admin:1234@서버주소:8761/eureka/
jwt:
  secretKey: lk2jelk32rjfo23ijrf091u013jnfoi431jt01h08ryh0193u45091fj09ew8rjodsajflkas
dj120834uj309u450938509djfslkdfasjlakfdsjalks
  access:
    expiration: 1209600000
    header: Authorization
  refresh:
    expiration: 1209600000
    header: Authorization-Refresh
```