소콘소콘

소소한 우리 동네 기프티콘

소콘소콘

포팅 메뉴얼

광주 특화 C207

붕방붕방 C207

목차

I. 개요

1. 프로젝트 개요

현대 사회에서 기프티콘은 간편하고 실용적인 선물 옵션으로 자리 잡았습니다.
친구에게 기프티콘을 선물 받을 때 근처에 없는 가게의 기프티콘을 받은 적 있으신가요?
대부분의 기프티콘 서비스가 대형 프랜차이즈나 유명 브랜드에 집중되어 있어, 지역 소상공인이 운영하는 가게들은 이러한 디지털 선물 시장에서 소외되어 왔습니다.

이러한 배경에서, 우리는 지역 기반의 소상공인을 위한 기프티콘 플랫폼을 개발하게 되었습니다.

2. 프로젝트 사용 도구

이슈 관리 : JIRA
형상 관리 : Gitlab
커뮤니케이션 : Notion, Mattermost, Miro
디자인 : Figma
UCC : 모바비
CI/CD : Jenkins, Docker

3. 개발환경

VS Code : 1.64.2,
Flutter : 14.16.0
IntelliJ : 17.0.9+7-b1087.9 amd64
Android Studio
JVM : OpenJDK 64-Bit Server VM by JetBrains s.r.o (스프링은 17 로 빌드)
SERVER : AWS EC2 Ubuntu 20.04.3 LTS
RDS: AWS RDS
DB : MySQL 8.0.33

4. 외부 서비스

Google Cloud Storage : serviceAccountKey.json 에 해당 내용 있음
(과금이 발생할 수 있는 키입니다. 취급 주의)
Firebase Realtime DB (front) :
Firebase Storage

[Flutter package]
UI 구성 요소
cupertino_icons (^1.0.6): 플러터 애플 아이콘을 제공하는 패키지입니다.
flutter_svg (^2.0.10+1): SVG(확장 가능한 벡터 그래픽) 파일을 플러터 위젯으로 렌더링할 수 있습니다.
flutter_feather_icons (^2.0.0): 플러터 앱에서 사용할 수 있는 페더 아이콘을 제공합니다.
fluentui_emoji_icon (^0.0.2): 플러터 앱에서 사용할 수 있는 플루언트 UI 이모지 아이콘을 제공합니다.

네비게이션
go_router (^13.2.1): 플러터 애플리케이션에서 선언적 라우팅 및 네비게이션을 위한 패키지입니다.

네트워킹 및 데이터 처리
http (^1.2.1): HTTP 요청을 만들기 위한 함수를 제공합니다.
firebase_core (^2.27.1): 플러터 앱에서 Firebase 서비스를 사용할 수 있게 해주는 Firebase Core 용 플러그인입니다.
firebase_messaging (^14.7.20): Firebase Cloud Messaging(FCM)을 위한 플러터 플러그인입니다.
provider (^6.0.0): 플러터 앱의 상태 관리를 위한 패키지입니다.
shared_preferences (^2.2.2): 기기 저장소에 키-값 쌍을 저장하기 위한 플러그인입니다.

유틸리티
image_picker (^1.0.7): 이미지 갤러리 또는 카메라에서 이미지를 선택할 수 있습니다.
pretty_qr_code (^3.3.0): 시각적으로 매력적인 QR 코드를 생성합니다.
image_gallery_saver (^2.0.3): 이미지와 비디오를 기기 갤러리에 저장할 수 있습니다.
intl (^0.19.0): 국제화 및 로컬라이제이션 유틸리티를 제공합니다.
permission_handler (^11.3.1): 플러터 애플리케이션에서 권한을 처리하기 위한 플러그인입니다.

location (^5.0.3): 플러터 앱의 위치 정보 서비스를 제공합니다.
flutter_local_notifications (^17.0.0): 플러터 앱에서 로컬 알림을 표시하기 위한 플러그인입니다.
background_locator_2 (^2.0.6): 앱의 백그라운드에서 Dart 코드를 실행할 수 있습니다.
uni_links (^0.5.1): 딥 링크 및 URI 스키마 처리를 위한 플러그인입니다.
kpostal (^0.5.1): 플러터 앱용 한국 우편 번호 검색 라이브러리입니다.
geolocator (^11.0.0): 플러터 앱의 지리적 위치 서비스를 제공합니다.

통합
webview_flutter (^4.7.0): 플러터 앱에 웹 뷰를 포함할 수 있습니다.
iamport_flutter (^0.10.0): IAMPORT 결제 서비스를 플러터에 통합하기 위한 패키지입니다.
iamport_webview_flutter (^3.0.6): IAMPORT 결제 서비스 통합을 위한 WebView 플러그인입니다.
google_maps_flutter (^2.1.12): 플러터 앱에 구글 지도를 포함할 수 있습니다.
google_maps_cluster_manager (^3.1.0): 플러터 앱에서 구글 지도의 클러스터링을 관리하는 패키지입니다.
google_maps_flutter_platform_interface (2.4.3): Google Maps 플러그인을 위한 플랫폼 인터페이스입니다.

UI 효과
animated_text_kit (^4.2.2): 플러터 앱에 애니메이션 텍스트 위젯을 제공합니다.
confetti (^0.7.0): 플러터 앱에 콘페티 효과를 생성합니다.

5.Gitgnore 처리한 핵심 키들

Spring : application.yml, gradle.properties
(₩src₩main₩resources, 또는 classPath 에 위치)
Flutter : google-services.json (android₩app 에 위치, firebase 통신을 위해 필요)
kakaoKey (location 를 통한 lat, lng 값 얻어오기 위해 필요)

II. 빌드

1. 환경변수 형태

1-1. .env

REDIS_PASSWORD=boongbang202403181403
CONFIG_USERNAME= 스프링 클라우드 컨피그 서버 관리자명
CONFIG_PASSWORD= 스프링 클라우드 컨피그 서버 관리자 비밀번호
GIT_USERNAME= 깃허브 사용자명
GIT_PASSWORD= 깃허브 사용자 토큰

1-2. application.yml – git submodule 통해 관리

1-2-1. release

1-2-1-1. config server

```
spring:
  application:
    name: config-service
  profiles:
    active: prod
  cloud:
    config:
      server:
        git:
          uri: https://github.com/boongbangsaurus/boongbang-settings.git
          skipSslValidation: true
          force-pull: true
          searchPaths:
            - back/**
          default-label: main
          username: ${GIT_USERNAME}
          password: ${GIT_PASSWORD}

server:
  port: 9000
```

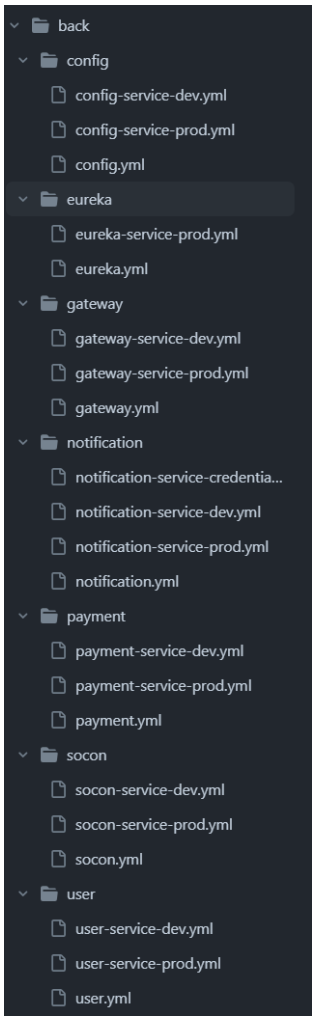
```
admin:
  username: ${CONFIG_USERNAME}
  password: ${CONFIG_PASSWORD}
```

1-2-1-2. config client

```
server:
  port: {서비스별 상이}

spring:
  application:
    name: gateway-service
  config:
    import:
optional:configserver:http://${CONFIG_USERNAME}:${CONFIG_PASSWORD}@j10c207.p.ssa
fy.io:9000
  profiles:
    active: prod
  cache:
    type: redis

  main:
    web-application-type: reactive


management:
  endpoints:
    enabled-by-default: false
    web:
      base-path: /actuator
      exposure:
        include: ['refresh','info', 'health']
    jmx:
      exposure:
        exclude: ['*']
```

1-2-2. main – spring cloud config server 를 통해 통신

파일 구조: {프로젝트명}-{프로필}.yml



2. 빌드하기

1) Front

Flutter build apk –release –target-platform=android-arm64

Flutter build apk –debug –target-platform=android-arm64

2) Back(Spring cloud)

2-1) config -> eureka -> gateway -> notification/socon/payment/user 순으로 빌드 및

2-2) 실행 방법

Gradle 실행

Bootjar 실행

3. 배포하기

3-1. Nginx 설정

nginx.conf

```
worker_processes auto;
events { worker_connections 8192; }

http {
    limit_req_zone $binary_remote_addr zone=limit_request_per_ip:10m rate=10r/s;

    map $http_user_agent $bad_bot {
        default 0;
        ~*(^MJ12bot|^MJ12bot/v1.4.5|SemrushBot|SemrushBot-SA|DomainCrawler|MegaIndex.ru|AlphaBot|Paros|ZmEu|nikto|dirbuster|sqlmap|openvas|w3af|Morfeus|Zollard|Arachni|Brutus|bsqlbf|Grendel-Scan|Havij|Hydra|N-Stealth|Netsparker|Pangolin|pmafind|webinspect) 1;
    }

    upstream jenkins {
        server jenkins:8080;
    }
    upstream nexus {
        server nexus:8081;
    }
    upstream gateway {
        server gateway:8000;
    }
    upstream eureka {
        server eureka:8761;
    }
    upstream config {
        server config:9000;
    }

    # 80 포트에 대한 설정 유지
```

```
    server {
        listen 80;
        listen 443;
        server_name j10c207.p.ssafy.io;
        return 403;
    }

    # SSL 설정 추가
    ssl_certificate /etc/letsencrypt/live/jenkins.socon-socon.site/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/jenkins.socon-socon.site/privkey.pem;

    # 80 포트 설정 포함
    include /etc/nginx/conf.d/nginx_80.conf;
    # 443 포트 설정 포함
    include /etc/nginx/conf.d/nginx_443.conf;
}
```

nginx_80.conf

```
server {
    listen 80;
    server_name jenkins.socon-socon.site;
    server_tokens off;

    location /.well-known/acme-challenge/ {
        allow all;
        root /var/www/certbot;
    }
    return 301 https://$host$request_uri;
}

server {
    listen 80;
    server_name nexus.socon-socon.site;
    server_tokens off;
```

```
    location /.well-known/acme-challenge/ {
        allow all;
        root /var/www/certbot;
    }
    return 301 https://$host$request_uri;
}

server {
    listen 80;
    server_name www.socon-socon.site;
    server_tokens off;

    location /.well-known/acme-challenge/ {
        allow all;
        root /var/www/certbot;
    }
    return 301 https://$host$request_uri;
}
```

nginx_443.conf

```
server {
    listen 443 ssl;
    server_name jenkins.socon-socon.site;
    server_tokens off;

    location /{
        # 나쁜 봇을 차단
        if ($bad_bot) {
        return 403;
        }

        # app 서비스로 라우팅
        proxy_pass http://jenkins/;
        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;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

server {
    listen 443 ssl;
    server_name nexus.socon-socon.site;
    server_tokens off;

    location /{
        # 나쁜 봇을 차단
        if ($bad_bot) {
        return 403;
        }

        # app 서비스로 라우팅
        proxy_pass http://nexus;
        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;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

server {
    listen 443 ssl;
    server_name www.socon-socon.site;
    server_tokens off;

    root /usr/share/nginx/html/policy;  # 정적 파일들의 루트 디렉토리
```

```
location / {
    index privacy_policy.html;        # 인덱스 파일 설정

    # 나쁜 봇을 차단
    if ($bad_bot) {
    return 403;
    }

    # 추가 보안 설정
    try_files $uri $uri/ =404;  # 요청된 URI 에 해당하는 파일이 없으면 404 에러 반환
}
location /api/ {
    # api 서비스로 라우팅
    proxy_pass http://gateway;
    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;
    proxy_set_header X-Forwarded-Proto $scheme;
}
location /eureka/ {
    # eureka 서비스로 라우팅
    rewrite ^/eureka(.*)$ $1 break;
    proxy_pass http://eureka;
    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;
    proxy_set_header X-Forwarded-Proto $scheme;
}

location /config/ {
```

```
        # config 서비스로 라우팅
        rewrite ^/config(.*)$ $1 break;
        proxy_pass http://config;
        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;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

3-2. Jenkins Jobs

be-config	5 days 4 hr #49	—	59 sec	
be-eureka-server	6 days 2 hr #13	—	52 sec	
be-gateway	5 days 21 hr #9	—	1 min 4 sec	
be-notification	1 day 4 hr #6	—	1 min 37 sec	
be-payment	15 hr #14	1 day 21 hr #10	1 min 8 sec	
be-socon	51 min #17	1 day 13 hr #9	1 min 13 sec	
be-user	3 hr 41 min #14	1 day 0 hr #6	1 min 6 sec	
be-utils	14 days #3	14 days #2	9.9 sec	
nginx	1 day 3 hr #83	—	10 sec	
policy	5 days 21 hr #17	—	6.1 sec	
redis	5 days 3 hr #17	5 days 3 hr #16	6.9 sec	

3-2-1. 젠킨스 파이프라인

3-2-1-1. 스프링부트 기반 전체 파이프라인 구조

```
pipeline {
    agent any
    environment {
        REPO = "s10-blockchain-contract-sub2/S10P22C207"
    }
    stages {
        stage('Checkout Main Repo') {
```

```
            steps {
                checkout scm
            }
        }

        stage('Init & Update Submodules with Custom Credentials') {
            steps {
                script {
                    // 서브모듈 크레덴셜을 설정
                    withCredentials([usernamePassword(credentialsId: 'github-access-token',
usernameVariable: 'GIT_USERNAME', passwordVariable: 'GIT_PASSWORD')]) {
                        sh "git submodule init"
                        sh "git config submodule.secure-settings.url
https://${GIT_USERNAME}:${GIT_PASSWORD}@github.com/boongbangsaurus/boongbang
-settings.git"
                        sh "git submodule update --recursive"
                        sh "cd secure-settings && git checkout release "
                    }
                }
            }
        }
//환경 변수를 제외한 나머지 단계는 모두 동일
        stage('Move .env File') {
            steps {
                script{
                    sh 'cp ../environments/.env backend/config/.env'
                     // config 프로젝트를 제외한 프로젝트에서 주석 해제
                     // sh 'cp ../environments/gradle.properties
backend/user/gradle.properties'
                    sh "mkdir -p backend/config/src/main/resources"
                    // find 명령어를 사용하여 .yml 파일을  찾고, 각 파일에 대해 cp 명령어를
실행
                    sh "cp secure-settings/back/{컨테이너명}/{컨테이너명}-service-default.yml
backend/{컨테이너명}/src/main/resources/application.yml"
                }
```

```
            }
        }
        stage('Setup Environment') {
            steps {
                script{
                    sh "chmod +x ./backend/config/gradlew"
                }
            }
        }
        stage("Build & Run") {
            steps {
                dir("backend/{컨테이너명}"){
                    script {
                        try {
                            sh "docker-compose -p {컨테이너명}-project build"
                            sh "docker-compose -p {컨테이너명}-project up -d"
                        }catch (Exception e) {
                            sh "docker stop {컨테이너명}"
                            sh "docker-compose -p {컨테이너명}-project build"
                            sh "docker-compose -p {컨테이너명}-project up -d"
                        }
                    }
                }
            }
        }
        stage('Prune old images'){
//사용되지 않은지 1 시간이 지난 이미지 및 볼륨, 컨테이너 삭제
            steps{
                script{
                    sh "yes | docker system prune --filter until=1h"
                }
            }
        }
    }
//Mattermost 에 빌드 결과 전송
```

```
    post {
        always {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout:
true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout:
true).trim()
                mattermostSend (color:
currentBuild.currentResult=='SUCCESS'?'good':'danger',
                        message: "빌드 ${currentBuild.currentResult}: ${env.JOB_NAME}
#${env.BUILD_NUMBER} by
${Author_ID}(${Author_Name})₩n(<${env.BUILD_URL}|Details>)",
                        endpoint:
'https://meeting.ssafy.com/hooks/8656byf6wbn3mdns9ym35suhze',
                        channel: 'Jenkins'
                )
            }
        }
    }
}
```

3-2-2. 도커 설정

3-2-2-1. 도커 네트워크 연결 설정

```
[
    {
        "Name": "jenkins-project_socon-net",
        "Id": "ef18bb4c68501230674cbf23362628751f9fb7e6a4b9a77fa6688eaa288d2e86",
        "Created": "2024-03-12T02:13:59.661430464Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
```

```
                "Subnet": "172.20.0.0/16",
                "Gateway": "172.20.0.1"
            }
        ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
        "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
        "19da9fc73435149e19990e918c04a70c7efb13fd56c8e3a765f015ba9450eacd": {
            "Name": "socon",
            "EndpointID":
"14d18267dc581ee453dfe19321acb38592188633d88d08ba6a1f61c971bd7946",
            "MacAddress": "02:42:ac:14:00:0f",
            "IPv4Address": "172.20.0.15/16",
            "IPv6Address": ""
        },
        "1ac5c4694c8ad1d68bf4f9a9d5735e2307ec226b6ce73f97d462508cd597750e": {
            "Name": "user",
            "EndpointID":
"a2f25b6f8cca28e6a445bdc2ac04d42009a5e13d9b7670fbe791543224da34b4",
            "MacAddress": "02:42:ac:14:00:0a",
            "IPv4Address": "172.20.0.10/16",
            "IPv6Address": ""
        },
        "4c945edb76c2990ff8d6f8294f067015d32307b3554a40d7154e10afea53a840": {
            "Name": "nginx",
            "EndpointID":
"eb1ed36910d442c01cd747cbfa24d2a8fbcd4da4c868d51254ae69227c6ddbf9",
            "MacAddress": "02:42:ac:14:00:03",
            "IPv4Address": "172.20.0.3/16",
```

```
            "IPv6Address": ""
        },
        "58fe1cad3221e1380d31b39a2affb1f4f8271acb5ed2d0d00ed1281032a90319": {
            "Name": "redis",
            "EndpointID":
"20a8c2b6434371bfa6df40fac58f5275c3189981f47d4098d4a28a758627ddcd",
            "MacAddress": "02:42:ac:14:00:06",
            "IPv4Address": "172.20.0.6/16",
            "IPv6Address": ""
        },
        "5a427840f37e9fe74d9048e3c10619585552857716fff042a1ff57776ec5dcfa": {
            "Name": "elasticsearch",
            "EndpointID":
"719abdcf70f47f339f660c32698064cfbd297957df845e0bf5e74757fb6fe87e",
            "MacAddress": "02:42:ac:14:00:0c",
            "IPv4Address": "172.20.0.12/16",
            "IPv6Address": ""
        },
        "5b3cb62bd0c88a0da1a5d10acaa70f07ea574f9cca19108488023c60e179e72f": {
            "Name": "nexus",
            "EndpointID":
"8962dc45b6b44ffd12f2d2808627a08a7de2c7bf8f0c82f9b6cd406ae9758c1d",
            "MacAddress": "02:42:ac:14:00:04",
            "IPv4Address": "172.20.0.4/16",
            "IPv6Address": ""
        },
        "6aa512df065af81ccc4b235f337cf6ba86446f38aad9172b52481e8fea48c5d7": {
            "Name": "config",
            "EndpointID":
"76a0397016c6a134ac8b5fec90f795ca9e51573edd0340718c947cd99d906667",
            "MacAddress": "02:42:ac:14:00:07",
            "IPv4Address": "172.20.0.7/16",
            "IPv6Address": ""
        },
        "6e23fe5ec11c55968b8a2c677f41e5551c8779c4cc3a9199418d0ff60f974f2e": {
```

"Name": "gateway",
"EndpointID":
"a6ef945b563d4cee4010e9c6b6ab56025f495815ebe303f7c0806aca3fbf0b82",
"MacAddress": "02:42:ac:14:00:09",
"IPv4Address": "172.20.0.9/16",
"IPv6Address": ""
},
"95726048cf13fabad3f31ff6312b548980c2c3c0b3e3076537dc421c12cf8d92": {
"Name": "jenkins",
"EndpointID":
"cd101b1b62d569e98cd7e7eba27f7cfb88c7d139d63fbb6a322391ec7bcf599e",
"MacAddress": "02:42:ac:14:00:02",
"IPv4Address": "172.20.0.2/16",
"IPv6Address": ""
},
"95f80f070b81dd86590def025143d529f2e8671771cb56b7763ab46cb9eb5774": {
"Name": "kibana",
"EndpointID":
"bb6f26f48ac99b0057052c6bb0ab22a842d50a5298369a3c1a1c4ee220bb00e8",
"MacAddress": "02:42:ac:14:00:0d",
"IPv4Address": "172.20.0.13/16",
"IPv6Address": ""
},
"99a3bcece5ab8e170274fb9f9dedb2535beb17fc5fd1397e5acdeb529f2ca967": {
"Name": "redis-key",
"EndpointID":
"82514cf3b1ffce5df199e1d19d097f0a717511c968981c616cbdc0c10efa6774",
"MacAddress": "02:42:ac:14:00:05",
"IPv4Address": "172.20.0.5/16",
"IPv6Address": ""
},
"c047de0569320f83a247bcf6a7e9af7d5f56c0a77afee55a0363e04b6e1db362": {
"Name": "notification",
"EndpointID":
"4e10e976c278868327ee8728f65c0fe2f185f21aa43841f38255ea7cf7469b2b",

```
            "MacAddress": "02:42:ac:14:00:0e",
            "IPv4Address": "172.20.0.14/16",
            "IPv6Address": ""
        },
        "ddd0b3a246f6e076c301eb40d002957a7c4b22be0ad606d9ef31e9f86b8e27d1": {
            "Name": "eureka",
            "EndpointID":
"a280e8597dbbd53982ae2231356f27aa27b5aa5d4da32339594530477a68216c",
            "MacAddress": "02:42:ac:14:00:08",
            "IPv4Address": "172.20.0.8/16",
            "IPv6Address": ""
        },
        "f78d80c7fac72c3297a09d8d6896374b1f150c0229c5c2236d8f50b636b24f2f": {
            "Name": "payment",
            "EndpointID":
"a8dcebd4a7d3865b8ed78ee38553d7eb5f62d6a45aa361214476947660e8e2e3",
            "MacAddress": "02:42:ac:14:00:0b",
            "IPv4Address": "172.20.0.11/16",
            "IPv6Address": ""
        }
    },
    "Options": {},
    "Labels": {
        "com.docker.compose.network": "socon-net",
        "com.docker.compose.project": "jenkins-project",
        "com.docker.compose.version": "2.24.7"
    }
  }
]
```


3-2-2-2. Spring boot

docker-compose.yml

```
version: '3.3'

services:
```

```
  socon:
    container_name: {컨테이너명}
    image: kimdahui/{컨테이너명}
    build:
      context: .
      dockerfile: Dockerfile
    restart: unless-stopped
    ports:
      - "{포트번호}:{포트번호}"
    environment:
      - CONFIG_PASSWORD=${CONFIG_PASSWORD}
      - CONFIG_USERNAME=${CONFIG_USERNAME}
    networks:
      - jenkins-project_socon-net

networks:
  jenkins-project_socon-net:
    external: true
```

Dockerfile – 기본 프로젝트

```
# 빌드 스테이지
FROM amazoncorretto:17.0.7-alpine AS builder
USER root
WORKDIR /config
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
# gradlew 실행 권한 부여
RUN chmod +x ./gradlew
RUN ./gradlew bootJar

# 실행 스테이지
FROM openjdk:17
```

```
WORKDIR /config
COPY --from=builder /config/build/libs/*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Dockerfile – utils

```
# 빌드 스테이지
FROM amazoncorretto:17.0.7-alpine AS builder
USER root
WORKDIR /utils
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY gradle.properties .
COPY settings.gradle .
COPY src src
# gradlew 실행 권한 부여
RUN chmod +x ./gradlew
RUN ./gradlew clean build
RUN ./gradlew publish
```

3-2-2-3. Elasticsearch

Docker-compose.yml

```
version: '3.7'
services:
  elasticsearch:
    container_name: elasticsearch
    build:
      context: .
      dockerfile: Dockerfile
    environment:
      - "discovery.type=single-node"
      - "xpack.security.enabled=false"
      - "node.name=single-node"
    ports:
      - 9200:9200
```

```
    - 9300:9300
  networks:
    - jenkins-project_socon-net

kibana:
  image: docker.elastic.co/kibana/kibana:8.13.0
  container_name: kibana
  environment:
    SERVER_NAME: kibana
    ELASTICSEARCH_HOSTS: http://elasticsearch:9200
  ports:
    - 5601:5601
  # Elasticsearch Start Dependency
  depends_on:
    - elasticsearch
  networks:
    - jenkins-project_socon-net

networks:
  jenkins-project_socon-net:
    external: true
```

Dockerfile – nori 한글 형태소 분석기 설치

```
FROM docker.elastic.co/elasticsearch/elasticsearch:8.13.0
RUN elasticsearch-plugin install analysis-nori
```

3-2-2-4. Nginx

docker-compose.yml

```
version: '3.3'

services:
  nginx:
    container_name: nginx
    image: kimdahui/nginx
    build:
```

```
    context: .
    dockerfile: Dockerfile
  restart: always
#   볼륨 매핑은 젠킨스 워크스페이스 기준
  volumes:
    - /etc/letsencrypt:/etc/letsencrypt
  ports:
    - "80:80"
    - "443:443"
  networks:
    - jenkins-project_socon-net

networks:
  jenkins-project_socon-net:
    external: true
```

Dockerfile

```
FROM nginx:latest

RUN rm -rf /etc/nginx/conf.d/default.conf
COPY nginx.conf /etc/nginx/nginx.conf
COPY nginx_80.conf /etc/nginx/conf.d/nginx_80.conf
COPY nginx_443.conf /etc/nginx/conf.d/nginx_443.conf
COPY policy/privacy_policy.html /usr/share/nginx/html/policy/privacy_policy.html
COPY policy/privacy_policy_history.html
/usr/share/nginx/html/policy/privacy_policy_history.html
COPY policy/history /usr/share/nginx/html/policy/history

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

3-2-2-4. Redis

docker-compose.yml

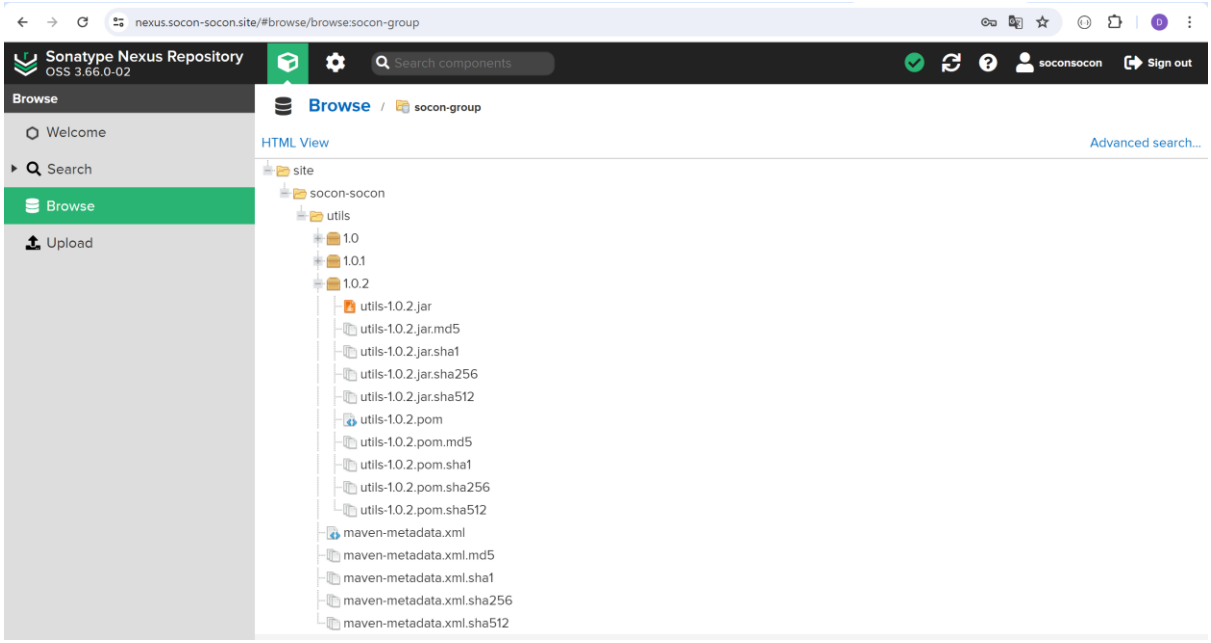
```
version: '3.8'

services:
```

```
  redis:
    container_name: redis
    hostname: redis
    image: kimdahui/redis
    build:
      context: no-key
      dockerfile: Dockerfile
    restart: unless-stopped
    networks:
      - jenkins-project_socon-net
    volumes:
      - redis-data:/data
    user: root
    command: redis-server /usr/local/etc/redis/redis.conf --requirepass
${REDIS_PASSWORD}
    ports:
      - "6379:6379"
networks:
  jenkins-project_socon-net:
    external: true

volumes:
  redis-data:
```

Dockerfile

```
FROM redis:7.2.4
COPY redis.conf /usr/local/etc/redis/redis.conf
```

3-3. 공통 라이브러리 넥서스 사설 레포지토리 배포



3-3-1. 도커 설정

docker-compose.yml

```
version: '3.3'

services:
  nexus:
    container_name: nexus
    image: sonatype/nexus3
    restart: always
#    볼륨 매핑은 젠킨스 워크스페이스 기준
    volumes:
      - nexus:/sonatype-work
    ports:
      - "8081:8081"
    networks:
      - jenkins-project_socon-net

volumes:
  nexus:

networks:
  jenkins-project_socon-net:
    external: true
```

3-3-2. 스프링 부트 설정

3-3-2-1. gradle.properties

```
nexusUrl=https://nexus.socon-socon.site/repository
nexusUsername=
nexusPassword=

project.name=
version=
queryDslVersion=
```

3-3-2-2. build.gradle

publish – 주요 설정

```
plugins {
    id 'java'
    id 'maven-publish'
    id "io.github.gradle-nexus.publish-plugin" version "2.0.0-rc-2" //넥서스 플러그인
    ...
}
//그룹명과 버전 지정
group = 'site.socon-socon'
version = '1.0.2'

...

bootJar.enabled = false // 실행가능한 아카이브로 main-class 가 있는 모듈인경우
jar.enabled = true // PLAIN 으로 생성 실행이 불가능한 일반 아카이브

repositories {
    mavenCentral()
     //nexus 레포지토리 연결
    maven {
        credentials(PasswordCredentials) {
            username = "${nexusUsername}"
            password = "${nexusPassword}"
        }
```

```
        url "${nexusUrl}/socon-group/"
        allowInsecureProtocol true
        // 가능한 경우 allowInsecureProtocol 제거하고 HTTPS 사용
    }
}

jar {
    archiveFileName = "${project.name}-${version}.jar"
}

//배포 단계
publishing {
    publications {
        mavenJava(MavenPublication) {
            groupId group
            artifactId project.name
            version version
            artifact("build/libs/$project.name-$version" + ".jar") {
                extension 'jar'
            }
        }
    }
    repositories {
        maven {
            credentials(PasswordCredentials) {
                username = "${nexusUsername}"
                password = "${nexusPassword}"
            }
            def releasesRepoUrl = "${nexusUrl}/socon-release"
            def snapshotsRepoUrl = "${nexusUrl}/socon-snapshot"
            url = version.endsWith('SNAPSHOT') ? snapshotsRepoUrl : releasesRepoUrl
            allowInsecureProtocol true
            // 가능한 경우 allowInsecureProtocol 제거하고 HTTPS 사용
        }
    }
```

```
}

dependencies {
    ...
}
...
```

client 주요 설정

```
...

repositories {
  mavenCentral()
  maven {
    credentials(PasswordCredentials) {
      username = "${nexusUsername}"
      password = "${nexusPassword}"
    }
    url "${nexusUrl}/socon-group/"
    allowInsecureProtocol true
    // 가능한 경우 allowInsecureProtocol 제거하고 HTTPS 사용
  }
}
...

dependencies {
  implementation 'site.socon-socon:utils:1.0.2' //배포한 라이브러리 추가
  ...
}
...
```

4.서비스 이용 방법

가) 포트원

준비 : 포트원 API 등록

- 채널 추가 → 테스트 → 결제 대행사 선택
- 결제 연동 → 식별코드 API Keys
- V1 API → 고객사 식별코드(프론트엔드), REST API Key, REST API Secret → 보안을 위해 yml 파일에 추가(gitgnore 했는지 확인!)

나) Firebase 실시간 알림

1. Firebase 콘솔 프로젝트 만들기
2. 설정 → 프로젝트 설정 → 서비스 계정 → 새 비공개 키 생성