포팅 메뉴얼

생성일 @2024년 9월 12일 오후 2:45

1. 개발 환경

1.1 Frontend

- Android
  - 사용 라이브러리

```
1. 주요 버전 정보

Kotlin 버전: 1.9.0
Compose 컴파일러 버전: 1.5.1
compileSdk: 34
minSdk: 24
targetSdk: 34

2. 빌드 도구 및 플러그인

Android Gradle Plugin: 8.2.2
Google Services Plugin: 4.4.2
Firebase Crashlytics Gradle: 3.0.2
Kotlin Serialization Plugin: 1.5.10
KSP (Kotlin Symbol Processing): 1.9.0-1.0.13

3. 결제 관련

Toss Payments SDK: 0.1.15

4. 네트워킹

Retrofit: 2.9.0
OkHttp: 4.9.1
Gson: 2.8.8

5. 데이터베이스

Room: 2.6.1

6. 이미지 처리

Glide: 4.12.0
Landscapist Glide: 1.4.6
Coil: 2.4.0

7. Firebase

Firebase BOM: 33.2.0
Firebase Analytics
Firebase Crashlytics: 19.1.0
Firebase Auth: 23.0.0

8. 지도 및 위치

Naver Maps SDK: 3.19.1
Google Play Services Location: 21.3.0

9. Jetpack Compose

Compose BOM: 2023.08.00
Compose UI
Compose Material: 1.7.0
Compose Material3
Compose Navigation: 2.8.0

10. 카메라

CameraX Core: 1.1.0
CameraX Camera2: 1.1.0
CameraX Lifecycle: 1.1.0
CameraX View: 1.1.0

11. 기타

AndroidX Core KTX: 1.13.1
AndroidX Lifecycle Runtime KTX: 2.8.5
AndroidX Activity Compose: 1.9.2
AndroidX AppCompat: 1.7.0
DataStore Preferences: 1.1.1
Android Billing Client: 7.1.1
Kotlin Serialization JSON: 1.5.0

12. 테스트 관련 라이브러리

JUnit: 4.13.2
AndroidX Test Ext JUnit: 1.2.1
Espresso Core: 3.6.1
Compose UI Test
```

- React
  - 사용 라이브러리

```
1. 핵심 React 라이브러리

react: ^18.3.1
react-dom: ^18.3.1

2. 라우팅

react-router-dom: ^6.26.2

3. UI 컴포넌트 및 스타일링

@emotion/react: ^11.13.3
@emotion/styled: ^11.13.0
@mui/material: ^6.1.2
react-datepicker: ^7.4.0
```

```
4. 상태 관리

zustand: ^5.0.0-rc.2

5. HTTP 클라이언트

axios: ^1.7.7

6. 데이터 시각화

chart.js: ^4.4.4
chartjs-plugin-datalabels: ^2.2.0
react-chartjs-2: ^5.2.0
recharts: ^2.12.7

7. 유틸리티 라이브러리

date-fns: ^4.1.0
jwt-decode: ^4.0.0

8. 알림 및 모달

sweetalert2: ^11.14.1

9. 개발 도구

@vitejs/plugin-react: ^4.3.1
vite: ^5.4.1

10. 코드 품질 및 린팅

@eslint/js: ^9.9.0
eslint: ^9.9.0
eslint-plugin-react: ^7.35.0
eslint-plugin-react-hooks: ^5.1.0-rc.0
eslint-plugin-react-refresh: ^0.4.9
globals: ^15.9.0

11. TypeScript 관련

@types/react: ^18.3.3
@types/react-dom: ^18.3.0
```

1.2 Backend

- Spring boot
  - build.gradle

```
1. Spring Boot Starters

org.springframework.boot:spring-boot-starter-data-jpa
org.springframework.boot:spring-boot-starter-data-redis
org.springframework.boot:spring-boot-starter-mail
org.springframework.boot:spring-boot-starter-oauth2-client
org.springframework.boot:spring-boot-starter-security
org.springframework.boot:spring-boot-starter-web
org.springframework.boot:spring-boot-starter-webflux
org.springframework.boot:spring-boot-starter-validation
org.springframework.boot:spring-boot-starter-quartz

2. XML 바인딩 (JAXB)

javax.xml.bind:jaxb-api:2.3.1
org.glassfish.jaxb:jaxb-runtime:2.3.1

3. 개발 도구 및 유틸리티

org.projectlombok:lombok (compileOnly, annotationProcessor)
org.springframework.boot:spring-boot-devtools (developmentOnly)

4. 데이터베이스

com.mysql:mysql-connector-j (runtimeOnly)

5. 테스팅

org.springframework.boot:spring-boot-starter-test
io.projectreactor:reactor-test
org.springframework.security:spring-security-test
org.junit.platform:junit-platform-launcher (testRuntimeOnly)
org.junit.jupiter:junit-jupiter-api:5.8.2
org.junit.jupiter:junit-jupiter-engine:5.8.2

6. API 문서화

org.springdoc:springdoc-openapi-starter-webmvc-ui:2.6.0

7. 보안 및 인증

io.jsonwebtoken:jjwt-api:0.12.6
io.jsonwebtoken:jjwt-impl:0.12.6 (runtimeOnly)
io.jsonwebtoken:jjwt-jackson:0.12.6 (runtimeOnly)

8. 쿼리 도구

com.querydsl:querydsl-jpa:5.0.0:jakarta
com.querydsl:querydsl-apt:${dependencyManagement.importedProperties['querydsl.version']}:jakarta (annotationProcessor)
jakarta.annotation:jakarta.annotation-api (annotationProcessor)
jakarta.persistence:jakarta.persistence-api (annotationProcessor)

9. PDF 생성

com.itextpdf:itextpdf:5.5.13.4
com.itextpdf.tool:xmlworker:5.5.13.4

10. 클라우드 서비스

software.amazon.awssdk:s3:2.28.11
```

1.3 Server

- Ubuntu 20.04.6 LTS
- Nginx(1.18.0)

- Docker(27.2.1)
- Docker compose(v2.29.2)
- MySQL(8.0.38)
- Redis(7.0)
- Jenkins(jenkins/jenkins:jdk21)

1.4 CI/CD

- Gitlab
- Jenkins

---

2. EC2 인스턴스 설정

2.1 Docker, Docker-compose 설치

```
# 시스템 업데이트
sudo apt update
sudo apt upgrade -y

# 필요한 패키지 설치
sudo apt install apt-transport-https ca-certificates curl software-properties-common -y

# Docker의 공식 GPG 키 추가
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

# Docker 저장소 추가
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# 패키지 목록 업데이트
sudo apt update

# Docker 설치
sudo apt install docker-ce docker-ce-cli containerd.io -y

# Docker 서비스 시작 및 부팅 시 자동 실행 설정
sudo systemctl start docker
sudo systemctl enable docker

# 현재 사용자를 docker 그룹에 추가 (sudo 없이 Docker 명령어 실행 가능 하도록 하기 위해서)
sudo usermod -aG docker $USER

# 설치 확인
docker --version

# Docker compose 설치
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

# 실행 권한 부여
sudo chmod +x /usr/local/bin/docker-compose

# 설치 확인
docker-compose --version
```

2.2 Nginx 설치 및 설정

```
# Nginx 설치
sudo apt install nginx

# Nginx 서비스 시작 및 자동 시작 설정
sudo systemctl start nginx
sudo systemctl enable nginx

# 방화벽에서 HTTP, HTTPS 허용
sudo ufw allow 'Nginx HTTP'
sudo ufw allow 'Nginx HTTPS'

# Nginx 설정 파일 편집
sudo nano /etc/nginx/sites-available/default

# 설정 파일 마지막에 다음 내용 추가
"
server {
    listen 80;
    server_name j11e202.p.ssafy.io;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl;
    server_name j11e202.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/p.ssafy.io/privkey.pem;

    client_max_body_size 30M;

    # 다른 SSL 설정은 그대로 유지

    location / {
        proxy_pass http://127.0.0.1:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;

        proxy_buffer_size 128k;
        proxy_buffers 4 256k;
        proxy_busy_buffers_size 256k;

        proxy_read_timeout 300s;
    }

    location /api {
        proxy_pass http://localhost:8081;
        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 /download/app.apk {
        alias /home/ubuntu/build/frontend/app-debug.apk;
        add_header Content-Disposition "attachment; filename=givutake.apk";
        add_header Content-Type application/vnd.android.package-archive;
    }
}
"

# 저장 후 설정 파일 유효한 지 확인
sudo nginx -t

# Nginx reload
sudo systemctl reload nginx
```

2.3 Jenkins에서 SSH 접속을 위한 SSH 키 생성

```
# ec2에서 다음 명령으로 ssh 키를 생성한다
ssh-keygen -t rsa -b 4096 -C "for jenkins" -N ""

# 생성된 키의 위치로 이동
cd ~/.ssh

# jenkins에서 ec2 인스턴스로의 접근을 위해 공개 키를 authorized_keys 파일에 등록한다
cat id_rsa.pub >> authorized_keys
```

2.4 Jenkins 설치 및 설정

```
# Jenkins 설치할 경로로 이동
mkdir -p ~/docker/jenkins && cd $_

# dockerfile 생성
nano dockerfile

# 다음 내용을 입력
'
# Jenkins JDK 21 이미지를 기반으로 함
FROM jenkins/jenkins:jdk21

# root 권한으로 전환
USER root

# 필요한 패키지 설치 및 Adoptium 저장소 추가
RUN apt-get update && \
    apt-get install -y wget apt-transport-https gnupg && \
    wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | apt-key add - && \
    echo "deb https://packages.adoptium.net/artifactory/deb $(awk -F= '/^VERSION_CODENAME/{print$2}' /etc/os-release) main" | tee /etc/apt/sources.list.d/adoptium.list

# Adoptium OpenJDK 21 설치
RUN apt-get update && \
    apt-get install -y temurin-17-jdk

# 기타 필요한 패키지 설치
RUN apt-get install -y \
    unzip \
    lib32stdc++6 \
    lib32z1

# Android SDK 설치
ENV ANDROID_HOME /opt/android-sdk
ENV ANDROID_SDK_ROOT $ANDROID_HOME
RUN mkdir -p ${ANDROID_SDK_ROOT} && cd ${ANDROID_SDK_ROOT} && \
    wget https://dl.google.com/android/repository/commandlinetools-linux-11076708_latest.zip && \
    unzip commandlinetools-linux-*_latest.zip && \
    rm commandlinetools-linux-*_latest.zip && \
    mv cmdline-tools latest && \
    mkdir cmdline-tools && \
    mv latest cmdline-tools/

# Gradle 설치
ENV GRADLE_HOME /opt/gradle
ENV GRADLE_VERSION 8.10.1
RUN wget https://services.gradle.org/distributions/gradle-${GRADLE_VERSION}-bin.zip -P /tmp && \
    unzip -d /opt/gradle /tmp/gradle-${GRADLE_VERSION}-bin.zip && \
    ln -s /opt/gradle/gradle-${GRADLE_VERSION} /opt/gradle/latest && \
    rm /tmp/gradle-${GRADLE_VERSION}-bin.zip

# Node.js와 npm 설치
ENV NODE_VERSION 20.x
RUN curl -fsSL https://deb.nodesource.com/setup_${NODE_VERSION} | bash - && \
    apt-get install -y nodejs

# 환경 변수 설정
ENV PATH ${PATH}:${ANDROID_SDK_ROOT}/cmdline-tools/latest/bin:${ANDROID_SDK_ROOT}/platform-tools:${GRADLE_HOME}/latest/bin

# Android SDK 컴포넌트 설치
RUN yes | sdkmanager --licenses && \
    sdkmanager "platform-tools" "platforms;android-34" "build-tools;34.0.0"

# Jenkins 사용자로 다시 전환
USER jenkins
'

# docker-compose.yml 파일 생성
nano docker-compose.yml

# 다음 내용을 입력
'
services:
  jenkins:
    build:
      context: .
      dockerfile: dockerfile
    container_name: jenkins
    user: root
    ports:
      - "7777:8080" # Jenkins 웹 인터페이스
      - "50000:50000" # Jenkins 에이전트 통신
    volumes:
      - ./jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    restart: always
    environment:
      TZ: "Asia/Seoul"
'
```

```
# 방화벽에서 7777번 포트 허용(Jenkins 웹 인터페이스 접속을 위해)
sudo ufw allow 7777/tcp

# jenkins 실행
docker-compose up -d

# jenkins 초기 비밀번호 확인
docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword

# jenkins 웹 인터페이스(https://j11e202.p.ssafy.io:7777)로 접속 후 로그인 한다
# Jenkins 관리 - Plugins 에 들어가 다음 플러그인을 설치한다
```
Docker plugin
Generic Webhook Trigger Plugin
GitLab Authentication plugin
GitLab Plugin
Gradle Plugin
Pipeline
Pipeline: Stage View Plugin
SSH Agent Plugin
Workspace Cleanup Plugin
```

# jenkins 메인 화면에서 '새로운 Item' - 'Item 이름 입력' - 'Pipeline' - 'OK' 순서대로 하여 Item을 생성한다
# Item의 Configure 탭의 General의 Build Triggers에서 'Build when a change is pushed to GitLab. GitLab webhook URL: http://j11e202.p.ssafy.io:7777/project/{Item 이름}" 을 체크한다
# Push Events를 클릭한다
# 아래의 고급 탭을 누르면 Secret token 탭이 나온다. 우측 하단에 Generate를 눌러 Secret token을 생성한다(이는 gitlab에서 사용한다)
# 아래의 Pipeline 탭에서 Definition으로 Pipeline script를 선택한 후 다음 내용을 입력한다
'
pipeline {
    agent any

    stages {
        stage('Checkout') {
            steps {
                script {
                    checkout([
                        $class: 'GitSCM',
                        branches: [[name: "develop"]],
                        userRemoteConfigs: [[
                            url: 'https://lab.ssafy.com/s11-fintech-finance-sub1/S11P21E202.git',
                            credentialsId: '23b7058f-83e8-4a5e-8cea-82df7e0de713'
                        ]]
                    ])
                }
            }
        }

        stage('Build Backend') {
            steps {
                dir('backend/givutake') {
                    sh '''
                        chmod +x ./gradlew
                        export GRADLE_OPTS="-Xmx4g -XX:MaxMetaspaceSize=1g"
                        ./gradlew clean build -x test --no-daemon --max-workers 4
                    '''
                }
            }
        }

        stage('Build Android') {
            steps {
                dir('frontend/android') {
                    sh '''
                        chmod +x ./gradlew
                        export GRADLE_OPTS="-Xmx4g -XX:MaxMetaspaceSize=1g"
                        ./gradlew clean assembleDebug --no-daemon --max-workers 4 -x test
                    '''
                }
            }
        }

        stage('Archive Artifacts') {
            steps {
                archiveArtifacts artifacts: 'backend/givutake/build/libs/*.jar', fingerprint: true
                archiveArtifacts artifacts: 'frontend/android/app/build/outputs/apk/debug/*.apk', fingerprint: true
            }
        }

        stage('Deploy to EC2') {
            steps {
                sshagent(credentials: ['ec2-deploy-key']) {
                    sh '''
                        # Backend JAR 파일 전송
                        scp -o StrictHostKeyChecking=no backend/givutake/build/libs/*.jar ubuntu@j11e202.p.ssafy.io:~/build/backend

                        # Android APK 파일 전송
                        scp -o StrictHostKeyChecking=no frontend/android/app/build/outputs/apk/debug/*.apk ubuntu@j11e202.p.ssafy.io:~/build/frontend

                        # 프론트엔드 프로젝트 폴더 전체 전송
                        scp -o StrictHostKeyChecking=no -r frontend/react ubuntu@j11e202.p.ssafy.io:~/build/frontend
                    '''
                }
            }
        }

        stage('Deploy and Run on EC2') {
            steps {
                sshagent(credentials: ['ec2-deploy-key']) {
                    sh '''
                        ssh ubuntu@j11e202.p.ssafy.io '

                        cd ~/build/frontend

                        docker build -t react-app .

                        docker-compose up -d

                        cd ~/build/backend

                        # Docker 이미지 빌드
                        docker build -t givutake-app:latest .

                        # 이전 컨테이너 중지 및 제거
                        docker-compose down
```

```
                        # 새 컨테이너 실행
                        docker-compose up -d

                        # 사용하지 않는 이미지 정리
                        docker image prune -f
                        '
                    '''
                }
            }
        }
    }

    post {
        always {
            cleanWs()
        }
        success {
            echo 'Build successful!'
        }
        failure {
            echo 'Build failed!'
        }
    }
}
'

# 그리고 저장 버튼을 눌러 jenkins item 설정을 저장한다

# GitLab의 프로젝트로 들어간다
# Settings - Webhooks로 들어가 Add new webhook을 누른다
# URL에는 jenkins의 build triggers 설정할 때 있었던 url을 입력한다
# Secret token에도 build triggers 에서 생성했던 secret token을 입력한다
# Trigger는 Push events를 체크한 뒤, Regular expression 에서 '(master|develop)' 을 입력한 뒤 Add webhook을 눌러 webhook을 생성한다

# jenkins로 돌아와서 'Jenkins 관리'에 들어간다
# 'Credentials'에 들어와서 Domain (global) 을 클릭하고 Add credentials 를 선택한다
# Kind는 'SSH Username with private key', ID는 'ec2-deploy-key', Username은 'ubuntu', Private Key는 ec2에서 생성했던 ssh 키 중, id_rsa 의 내용을 입력한다
# Create를 눌러 credential을 생성한다

# GitLab에서 webhook 테스트를 통해 jenkins가 정상적으로 동작하고 빌드되는지 확인한다
```

2.5 amazon S3, amazon Cloudfront 설정

```
# aws 사이트에 접속하여 AWS Management Console에 로그인한다
# S3 서비스로 이동한다
# 버킷 만들기를 클릭한다
# 버킷 이름을 givutakebucket 으로 설정하고, 다른 설정은 유지한 채 버킷 만들기를 클릭하여 버킷을 생성한다
# 생성된 버킷을 클릭한 후, 권한 탭으로 들어간다
# 버킷 정책의 우측에 있는 편집 버튼을 클릭하고 다음 내용을 입력한다
'
{
    "Version": "2008-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Sid": "AllowCloudFrontServicePrincipal",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::givutakebucket/public/*",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceArn": "arn:aws:cloudfront::600627332627:distribution/E2KP7GKHG5TWSF"
                }
            }
        }
    ]
}
'

# IAM 서비스로 이동한다
# 사용자 탭에서 '사용자 추가' 버튼을 클릭한다
# 사용자 이름은 's3-access-user' 로 설정한다
# 다음 버튼을 누르고 권한 옵션은 '직접 정책 연결'을 선택한다
# 정책 생성 버튼을 누른다
# 정책 편집기에서 '시각적'을 'JSON' 으로 변경한다
# 다음 내용을 붙여넣는다
'
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowS3Access",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::givutakebucket",
                "arn:aws:s3:::givutakebucket/*"
            ]
        }
    ]
}
'
# 정책 이름은 'givutakebucket-access' 로 설정한다
# '정책 생성' 버튼을 눌러 정책을 생성한다
# 다시 사용자 권한 설정으로 돌아왔다면 방금 생성한 정책을 선택하고 다음 버튼을 누른다
# 권한 요약에서 권한 정보를 확인한 후, '사용자 생성' 버튼을 눌러 사용자를 생성한다


# Cloudfront 서비스로 이동한다
# 우측의 '배포 생성' 버튼을 누른다
# Origin domain의 Choose origin domain을 클릭하여 생성한 버킷을 선택한다
# 원본 액세스는 '원본 액세스 제어 설정(권장)'을 선택한다
# 'Create new OAC' 버튼을 클릭하고 create 버튼을 눌러 OAC를 생성한다
# 좌측에서 생성한 OAC를 선택하고 '원본 생성' 버튼을 눌러 원본을 생성한다
```