Shape, rectangle

Description automatically generated

Text

Description automatically generated

**목차**

1. 개요

2. 프로젝트 개발 환경

2.1. 협업 및 프로젝트 관리 도구

2.2. 프로그래밍 환경

3. 빌드 및 배포

3.1. 빌드 및 배포 개요

3.2. 배포에 필요한 파일 설정

3.3. 어플리케이션 환경 변수

**1. 개요**

코로나 19 이후 거점 오피스를 이용하거나 자율 좌석제를 시행하는 회사가 늘어나고 있습니다. 회사가 고정된 자리를 의미하던 이전과는 달리 이제 내가 일을 하는 곳이 곧 회사가 된 것인데요.

거점 오피스는 좌율 좌석제가 비교적 편리한 공간에서 일할 수 있다는 장점을 갖고 있지만 이는 곧 자신만의 자리가 없다는 말과도 같기 때문에 출근할 때마다 자리를 나에 맞춰서 세팅을 해줘야 한다는 이전에 없던 새로운 불편함이 생겨났습니다.

나만의 똑똑한 데스크, 똑디는 데스크 세팅을 로그인이나 사원증 태그 한 번으로 유저가 미리 설정한 세팅으로 바꾸어줍니다. 또한 회사에서 오늘 할 일을 기록하고 일정 시간 이전에 알림을 띄워주는 캘린더 기능으로 팀원 간 서로 떨어져 일하는 게 일상화되는 요즘의 근무 환경에서 중요한 일정을 잊지 않도록 도와줍니다. 이렇게 똑디는 유저가 어느 자리에서든 자신의 자리처럼 똑똑하게 일할 수 있도록 도와줍니다.

**2. 프로젝트 개발 환경**

**2.1. 협업 및 프로젝트 관리 도구**

-이슈 관리**: JIRA** (https://jira.ssafy.com/secure/RapidBoard.jspa?rapidView=12581&

projectKey=S07P12A102&selectedIssue=S07P12A102-441&quickFilter=26238)

- 버젼 관리: **GITLAB** (https://lab.ssafy.com/s07-webmobile3-sub2/S07P12A102)

- 디자인 및 와이어프레임: **Figma**

- UCC 제작 : **Movavi**

**2.2. 프로그래밍 환경**

- 데스크탑 프레임워크

- **Electron** v.20.0.1

- 서버 및 CI/CD

- 클라우딩 컴퓨팅 서비스: AWS EC2

- EC2 OS: **Ubuntu** 22.04.1 LTS

- 리버스 프록시 서버: **Nginx**

- 컨테이너기반 가상화 : **Docker**

- CI/CD 자동배포 : **Jenkins**

- 데이터베이스

- **MySQL**

- 통합개발환경(IDE)

- **Intellij** v.21.3.3

- **Vs Code** v.1.70.1

- 백엔드 개발 환경

- **OpenJDK** v.11.0.11

- **SpringBoot** v.2.7.2

- **gradle** v.7.5

- 프론트엔드 개발 환경

- **Node.js** v.16.16.0

- **React** v.18.2.0

**3. 빌드 및 배포**

**1. 빌드 및 배포 개요**

**Diagram, schematic

Description automatically generated**

빌드 및 배포는 깃랩 레포지토리(https://lab.ssafy.com/s07-webmobile3-sub2/S07P12A102)에 젠킨스의 웹훅이 걸려 있어 각각 FE/BE 브랜치에 깃 푸시가 발생하면 서버에 자동으로 배포되도록 설정되어 있습니다.

프론트엔드의 경우 키오스크 앱은 FE브랜치에 깃 푸시가 발생한 직후 젠킨스를 통해서 우분투의 지정된 프론트엔드 디렉토리에 깃 페치가 발생합니다. 그리고 미리 설정된 Dockerfile을 사용해서 젠킨스가 **Node.js** 이미지를 기반으로 리액트 소스 코드를 포함하는 컨테이너를 빌드한 후, 깃에서의 새로운 변경사항을 토대로 리액트 서버를 다시 빌드하고 노드 컨테이너 안에 띄웁니다. 반면 태블릿 앱의 경우 각각의 데스크/모니터 모터와의 연동을 위해서 해당 화면을 EC2 클라우딩 컴퓨팅에 띄울 필요가 없어 로컬호스트에서 서버와 통신하는 형식을 취합니다.

백엔드의 경우 BE 브랜치에 깃 푸시가 발생한 직후 젠킨스를 통해서 우분투의 지정된 백엔드 디렉토리에 깃 페치가 발생합니다. 그리고 미리 설정된 Dockerfile을 사용해서 젠킨스가 JVM 이미지를 기반으로 스프링부트 소스 코드를 포함하여 새로운 컨테이너를 빌드한 후 컨테이너 안에서 스프링부트 프로젝트의 빌드를 진행하고 JVM 컨테이너 안에 띄웁니다.

**2. 배포에 필요한 파일 설정**

**Dockerfile**

**(1) 젠킨스 컨테이너 (Docker out of docker)**

**Text

Description automatically generated**

**(2) Node.js 프론트엔드 컨테이너**

Text

Description automatically generated

\* WORKDIR의 경우 프론트엔드 키오스크 루트 폴더의 파일 트리대로 지정하지 않아도 됨.

**(3) JVM 백엔드 컨테이너**

**Graphical user interface, text

Description automatically generated**

\* WORKDIR의 경우 백엔드 루트 폴더의 파일 트리대로 지정하지 않아도 됨.

도커 컨테이너의 경우 프론트엔드의 컨테이너는 "ddokdi\_fe\_container", 백엔드는 "ddokdi\_spring\_container"라는 이름으로 개설도되록 젠킨스에서 설정되어 있음. 따라서 해당 컨테이너의 현재 로그를 보고 싶다면 다음의 명령어를 우분투 서버에서 입력하면 된다.

(1) 리액트 앱 도커 로그 확인 : docker logs -f ddokdi\_fe\_container

(2) 백엔드 앱 도커 로그 확인 : docker logs -f ddokdi\_spring\_container   
\* logs의 -f 옵션은 실시간으로 갱신되는 로그를 반영하는 옵션

**3. 어플리케이션 환경 변수**

BackEnd/src/main/resources/application.properties

# MySQL DB

spring.datasource.ulr : 우분투 내의 DB주소

spring.datasource.username : DB 접근 유저

spring.datasource.password : DB 접근 비밀번호

spring.datasource.driver-class-name : com.mysql.cj.jdbc.Driver :DB 드라이버

spring.jpa.database = mysql

spring.jpa.database-platform = org.hibernate.dialect.MySQL5InnoDBDialect

# Spring Security

spring.security.user.name = 스프링 시큐리티 디폴트 유저

spring.security.user.password = 스프링 시큐리티 디폴트 패스워드

# jwt

jwt.header : jwt토큰 키 네임

jwt.secret : jwt 키