포팅 메뉴얼

소유자	김태영정 정웅 편 주영 이보규 보규 정 티모시 jiwon
태그	

1. 개발환경

1.1 Frontend

프로젝트 구조

- Monorepo 구조 (yarn workspaces)

    - my-code (메인 애플리케이션)
    - my-ide (회의실 애플리케이션)

핵심 기술 스택

- 빌드 도구 및 개발 환경
    - Vite v6.0.5 (my-code)
    - React Scripts v5.0.1 (my-ide)
    - TypeScript v4.4.2
    - ESLint v9.17.0

주요 라이브러리

- 공통
    - React v19.0.0
    - React DOM v19.0.0
    - OpenVidu Browser v2.31.0
    - Axios v1.7.9
    - TailwindCSS v3.4.17
- my-code (메인 앱)
    - Redux Toolkit v2.5.0 & Redux Persist v6.0.0
    - React Router DOM v7.1.1
    - Chart.js v4.4.7 & React-Chartjs-2 v5.3.0
    - DaisyUI v4.12.23
    - LiveKit Client v2.8.0
    - React Markdown v9.0.3
- my-ide (회의실)
    - CodeMirror 6 관련 패키지
        - 다양한 언어 지원 (@codemirror/lang-cpp v6.0.2)
        - 자동완성, 린트 지원 (@codemirror/autocomplete v6.18.4)
        - One Dark 테마 (@codemirror/theme-one-dark v6.1.2)

    - Yjs v13.6.23 & 관련 패키지
        - 실시간 협업 기능
        - WebRTC v10.3.0, WebSocket v2.1.0 지원
    - Konva v9.3.18 & React Konva v19.0.2
    - LiveBlocks Client v2.16.2
- 스타일링
    - TailwindCSS v3.4.17
    - PostCSS v8.4.49
    - Autoprefixer v10.4.20

개발 도구

- ESLint v9.17.0
- TypeScript v4.4.2
- Concurrently v9.1.2 (멀티 프로세스 실행)

1.2 Backend

기본 정보

- Java 17
- Spring Boot 3.4.2
- Spring Dependency Management 1.1.7

주요 프레임워크 및 라이브러리

- Spring Boot Starters
    - Spring Boot JDBC v3.4.2
    - Spring Boot Web v3.4.2
    - Spring Boot WebSocket v3.4.2
    - Spring Boot WebFlux v3.4.2
    - Spring Boot Data JPA v3.4.2
- 보안

    - Spring Boot Security v3.4.2
    - Spring Boot OAuth2 Client v3.4.2
    - Spring Security OAuth2 JOSE v3.4.2
    - JWT (JSON Web Token) v0.12.3
        - jjwt-api v0.12.3
        - jjwt-impl v0.12.3
        - jjwt-jackson v0.12.3
- 통신 및 미디어
    - Spring Boot Mail v3.4.2
    - OpenVidu Java Client v2.31.0
    - Spring Cloud AWS v2.2.6.RELEASE
- 데이터베이스
    - MySQL Connector/J v8.2.0

개발 도구

- Lombok

빌드 도구

- Gradle (Java Plugin)

1.4 Database

- MySQL 8.0

1.5 Self-host 서버

- Judge0 v1.13.0
- Openvidu v2.31.0

2. 환경변수

2.1 공통

```
ENVIRONMENT=prod
OPENAI_API_KEY=your-openai-api-key
```

2.2 Frontend

```
REACT_PORT=3000
WEBSOCKET_PORT=3001
FRONTEND_URL=your-frontend-server-url
BACKEND_URL=your-backend-server-url
REACT_APP_FRONTEND_URL=your-frontend-server-url
REACT_APP_BACKEND_URL=your-backend-server-url
REACT_APP_CONCURRENCY_BACKEND_URL=http://your-concurrency-server
REACT_APP_CONCURRENCY_BACKEND_WEBSOCKET_URL=ws://your-concu
```

2.3 Backend

```
SPRING_BOOT_PORT=8080
SPRING_DATASOURCE_DRIVER=com.mysql.cj.jdbc.Driver
SPRING_DATASOURCE_PASSWORD=
SPRING_DATASOURCE_URL=jdbc:mysql://mysql-db:3306/
SPRING_DATASOURCE_USERNAME=
SPRING_MAIL_PASSWORD=
SPRING_MAIL_USERNAME=
ACCESS_KEY=AWS-S3-Access-Key
SECRET_KEY=AWS-S3-Secret-Key
```

4. 빌드 및 배포

4.1 기본 포트 정보

1. BACKEND 서버 : 8080
2. FRONTEND 서버 : 3000
3. CONCURRENCY 서버 : 3001

⚠️ CONCURRENCY 서버는 FRONT END 내부의 /packages/my-ide/server/server.js임

4.2 Judge0 배포

judge0/CHANGELOG.md at master · judge0/judge0
🔥 The most advanced open-source online code execution system in the world. - judge0/judge0
https://github.com/judge0/judge0/blob/master/CHANGELOG.md#deployment-procedure


```
# ---------- (1) System Requirements ----------

# Judge0는 Linux에서만 테스트되었으며, 다른 시스템에서는 작동하지 않을 수 있음
# Ubuntu 22.04에서 사용을 권장

# GRUB 업데이트:
sudo nano /etc/default/grub
# GRUB_CMDLINE_LINUX 변수에 systemd.unified_cgroup_hierarchy=0 추가
# 예: GRUB_CMDLINE_LINUX="... systemd.unified_cgroup_hierarchy=0"
sudo update-grub
sudo reboot

# Docker 및 Docker Compose가 설치되어 있어야 함


# ---------- (2) Deployment Steps ----------

# Release 아카이브 다운로드 및 압축 해제
wget https://github.com/judge0/judge0/releases/download/v1.13.1/judge0-v1.1
unzip judge0-v1.13.1.zip

# REDIS_PASSWORD 업데이트:
# 아래 사이트에서 랜덤 비밀번호를 생성
# https://passwordsgenerator.net/
```

```
nano judge0.conf
# REDIS_PASSWORD 변수에 생성한 비밀번호 입력

# POSTGRES_PASSWORD 업데이트:
# 아래 사이트에서 랜덤 비밀번호를 생성
# https://passwordsgenerator.net/
nano judge0.conf
# POSTGRES_PASSWORD 변수에 생성한 비밀번호 입력


# ---------- (3) Docker 서비스 실행 ----------

cd judge0-v1.13.1
docker-compose up -d db redis
sleep 10s
docker-compose up -d
sleep 5s


# ---------- (4) Judge0 실행 확인 ----------

# Judge0 CE v1.13.1이 실행 중
# 다음 주소에서 문서를 확인할 수 있음
http://<서버의 IP 주소>:2358/docs
```

4.3 Openvidu v2.31.0 배포

On premises - OpenVidu Docs

OpenVidu is deployed in production as a set of Docker containers managed with Docker Compose. You can deploy OpenVidu in any modern Linux distribution.

https://docs.openvidu.io/en/stable/deployment/ce/on-premises/

```
# ---------- (1) Docker의 apt 저장소 설정 ----------

# Docker 공식 GPG 키 추가
sudo apt-get update
sudo apt-get install ca-certificates curl
```

```
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/key
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Docker 저장소를 Apt 소스에 추가
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update


# ---------- (2) Docker 패키지 설치 ----------

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plug


# ---------- (3) Docker Compose 설치 ----------

sudo apt-get update
curl -o ./docker-desktop-amd64.deb https://desktop.docker.com/linux/main/a
sudo apt-get install ./docker-desktop-amd64.deb
systemctl --user start docker-desktop


# ---------- (4) OpenVidu 배포 준비 ----------

sudo su
cd /opt


# ---------- (5) Openvidu 배포 과정 안내 메시지 ----------

========================================
OpenVidu Platform successfully installed.
========================================

1. Go to openvidu folder:
```

```
$ cd openvidu

2. Configure DOMAIN_OR_PUBLIC_IP and OPENVIDU_SECRET in .env file:
$ nano .env

3. Start OpenVidu
$ ./openvidu start

For more information, check:
https://docs.openvidu.io/en/stable/deployment/ce/on-premises/


# ---------- (6) .env 파일 설정 ----------

# OpenVidu configuration
# ----------------------
# Documentation: https://docs.openvidu.io/en/stable/reference-docs/openvid

# Domain name. If you do not have one, the public IP of the machine.
# For example: 198.51.100.1, or openvidu.example.com
DOMAIN_OR_PUBLIC_IP=

# OpenVidu SECRET used for apps to connect to OpenVidu server and users t
OPENVIDU_SECRET=

# Certificate type:
# - selfsigned:  Self signed certificate. Not recommended for production use.
#             Users will see an ERROR when connected to web page.
# - owncert:     Valid certificate purchased in a Internet services company.
#             Please put the certificates files inside folder ./owncert
#             with names certificate.key and certificate.cert
# - letsencrypt: Generate a new certificate using letsencrypt. Please set the
#             required contact email for Let's Encrypt in LETSENCRYPT_EMAIL
#             variable.
CERTIFICATE_TYPE=selfsigned

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for n
LETSENCRYPT_EMAIL=user@example.com
```

```
# ---------- (7) OpenVidu 실행 ----------

./openvidu start
```

4.4 백엔드 배포

```
# ================================
# Spring Boot (Backend) 배포
# ================================

# ---------- 1. 환경 변수 설정 ----------
export ENVIRONMENT=prod
export SPRING_BOOT_PORT=8080
export BACKEND_URL=your-backend-server-url

echo "ENVIRONMENT 설정: $ENVIRONMENT"
echo "SPRING_BOOT_PORT 설정: $SPRING_BOOT_PORT"
echo "BACKEND_URL 설정: $BACKEND_URL"


# ---------- 2. Docker Compose 설치 확인 ----------
if ! command -v docker-compose &> /dev/null; then
  echo "Docker Compose가 설치되어 있지 않음. 설치 시작."
  curl -L "https://github.com/docker/compose/releases/latest/download/docke
    -o /usr/local/bin/docker-compose
  chmod +x /usr/local/bin/docker-compose
  ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
fi

docker-compose version


# ---------- 3. Spring Boot 빌드 ----------
echo "Spring Boot CI/CD 빌드 시작 (프로덕션 서버용)"
cd backend/codemaestro || { echo "backend/codemaestro 폴더를 찾을 수 없음"
```

```
chmod +x gradlew
./gradlew clean build || { echo "Spring Boot 빌드 실패"; exit 1; }
cd ../../


# ---------- 4. 기존 백엔드 컨테이너 종료 ----------
PROJECT_NAME="codemaestro-prod"

echo "기존 백엔드 컨테이너 종료 중 (프로젝트: $PROJECT_NAME)..."
docker-compose -p "$PROJECT_NAME" stop backend || true
docker-compose -p "$PROJECT_NAME" rm -f backend || true


# ---------- 5. Docker Compose로 백엔드 실행 ----------
echo "Docker Compose로 백엔드 컨테이너 실행 (프로젝트: $PROJECT_NAME)..."
docker-compose -p "$PROJECT_NAME" up --build -d backend

BACKEND_CONTAINER="codemaestro-prod-backend"

echo "백엔드 컨테이너 상태 확인..."
if ! docker ps --format "{{.Names}}" | grep -q "$BACKEND_CONTAINER"; then
  echo "백엔드 컨테이너($BACKEND_CONTAINER)가 실행되지 않음. 실행 시도..."
  docker start "$BACKEND_CONTAINER"
else
  echo "백엔드 컨테이너 실행 확인 완료: $BACKEND_CONTAINER"
fi


# ---------- 6. MySQL 네트워크 연결 ----------
MYSQL_NETWORK="jenkins_default"

echo "백엔드 컨테이너($BACKEND_CONTAINER)를 MySQL 네트워크($MYSQL_NI
docker network connect "$MYSQL_NETWORK" "$BACKEND_CONTAINER" || t


# ---------- 7. 배포 완료 ----------
echo "Spring Boot 프로덕션 서버 실행 완료 (프로젝트: $PROJECT_NAME)"
```

4.5 프론트엔드 배포

```
# ================================
# React (Frontend) 배포
# ================================

# ---------- 1. 환경 변수 설정 ----------
export ENVIRONMENT=prod
export REACT_PORT=3000
export FRONTEND_URL=your-frontend-server-url
export REACT_APP_FRONTEND_URL=your-frontend-server-url
export REACT_APP_BACKEND_URL=your-backend-server-url
export REACT_APP_CONCURRENCY_BACKEND_URL=http://your-concurrency
export REACT_APP_CONCURRENCY_BACKEND_WEBSOCKET_URL=ws://your-

echo "ENVIRONMENT 설정: $ENVIRONMENT"
echo "REACT_PORT 설정: $REACT_PORT"
echo "FRONTEND_URL 설정: $FRONTEND_URL"
echo "REACT_APP_FRONTEND_URL 설정: $REACT_APP_FRONTEND_URL"
echo "REACT_APP_BACKEND_URL 설정: $REACT_APP_BACKEND_URL"
echo "REACT_APP_CONCURRENCY_BACKEND_URL 설정: $REACT_APP_CONC
echo "REACT_APP_CONCURRENCY_BACKEND_WEBSOCKET_URL 설정: $REAC



# ---------- 2. Node.js 설치 및 업그레이드 ----------
if ! command -v node &> /dev/null; then
  echo "Node.js가 설치되어 있지 않음. Node.js 20 설치 시작."
  curl -fsSL https://deb.nodesource.com/setup_20.x | bash -
  apt-get install -y nodejs
else
  NODE_VERSION=$(node --version)
  NODE_MAJOR=$(echo "$NODE_VERSION" | sed 's/^v//' | cut -d. -f1)
  echo "현재 Node.js 버전: $NODE_VERSION"

  if [ "$NODE_MAJOR" -lt 20 ]; then
    echo "Node.js 버전이 20 미만. 업그레이드 진행."
    curl -fsSL https://deb.nodesource.com/setup_20.x | bash -
    apt-get install -y nodejs
```

```
    echo "업그레이드 후 Node.js 버전: $(node --version)"
  else
    echo "Node.js 20 이상 설치 완료."
  fi
fi


# ---------- 3. Yarn 설치 확인 ----------
if ! command -v yarn &> /dev/null; then
  echo "Yarn이 설치되어 있지 않음. 설치 진행."
  npm install -g yarn
else
  echo "Yarn 설치 확인 완료: $(yarn --version)"
fi


# ---------- 4. 프론트엔드 의존성 설치 ----------
echo "프론트엔드 의존성 설치 시작 (frontend/codemaestro)"
cd frontend/codemaestro || { echo "frontend/codemaestro 폴더를 찾을 수 없음"

if command -v yarn &> /dev/null; then
  echo "Yarn을 사용하여 의존성 설치 중..."
  yarn install
else
  echo "npm을 사용하여 의존성 설치 중..."
  npm install
fi

cd - > /dev/null


# ---------- 5. 기존 프론트엔드 컨테이너 종료 ----------
PROJECT_NAME="codemaestro-prod"

echo "기존 프론트엔드 컨테이너 종료 중 (프로젝트: $PROJECT_NAME)..."
docker-compose -p "$PROJECT_NAME" stop frontend || true
docker-compose -p "$PROJECT_NAME" rm -f frontend || true
```

```
# ---------- 6. Docker Compose로 프론트엔드 실행 ----------
echo "Docker Compose로 프론트엔드 컨테이너 실행 (프로젝트: $PROJECT_NAME
docker-compose -p "$PROJECT_NAME" up --build -d frontend

FRONTEND_CONTAINER="codemaestro-prod-frontend"

echo "프론트엔드 컨테이너 상태 확인..."
if ! docker ps --format "{{.Names}}" | grep -q "$FRONTEND_CONTAINER"; the
  echo "프론트엔드 컨테이너($FRONTEND_CONTAINER)가 실행되지 않음. 실행 시도
  docker start "$FRONTEND_CONTAINER"
else
  echo "프론트엔드 컨테이너 실행 확인 완료: $FRONTEND_CONTAINER"
fi
```