몰¿룸?: 온라인 심리 상담 서비스

포팅 메뉴얼

목차

I. 개요

1. 프로젝트 개요

방탈출이 최근 취미로 떠오르고 있는 지금 당신도 방탈출에 관심이 있나요?
근데 테마가 너무 많아서 뭘 해야할 지 모르겠다고요?
테마는 정했는데 같이 갈 사람이 없다고요?

몰룸은 그러한 불편사항을 완화시켜주기 위해 탄생한 빅데이터 기반 방탈출 테마 추천 및 파티 매칭 서비스입니다. 몰룸과 함께라면, 당신의 방탈출 취미를 보다 재밌게 즐길 수 있습니다.

2. 프로젝트 사용 도구

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

3. 개발환경

Frontend

Node	**20.15.0**
TypeScript	4.9.5
React	18.3.1
Emotion	11.13.3
Storybook	8.2.9
Tanstack-Query	5.55.4
Zustand	4.5.5
stompjs	7.0.0
axios-mock-adapter	2.0.0

Backend

Java	**openjdk version "17.0.11" 2024-04-16 LTS**
Spring Boot	3.3.2.
Hibernate	6.5.2
MySQL	8.0.32
Redis	7.4.0
MongoDB	7.0.14

Infra

AWS EC2	**Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1063-aws x86_64)**
Docker	27.2.0
Jenkins	2.475
NginX	nginx/1.27.2

4. 외부 서비스

Google mail service : application.yml 에 해당 내용 있음
Firebase Realtime DB (back) : serviceAccountKey.json 에 해당 내용 있음
Redis cloud : application.yml 에 해당 내용 있음
Firebase Realtime DB (front) :

5. Gitignore 처리한 핵심 키들

React : .env (moreroom 폴더 하위)
Spring : application.yml, application-secret.yml
S11P21D206/.env
S11P21D206/bigdata/fastapi/.env
(₩src₩main₩resources, 또는 classPath 에 위치)

II. 빌드

1. 환경변수 형태

.application.yml
spring:

```
  application:
    name: moreroom

  datasource:
    url: ${MYSQL_URL}
    username: ${MYSQL_ROOT_USERNAME}
    password: ${MYSQL_ROOT_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver

  jpa:
    database-platform:
      org.hibernate.dialect.MySQLDialect
    hibernate:
      naming:
        physical-strategy:
org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl

  mail:
    host: smtp.gmail.com
    port: 587
    username: ${MAIL_USERNAME}
    password: ${MAIL_PASSWORD}
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            enable: true
            required: true
          connectiontimeout: 5000
          timeout: 5000
          writetimeout: 5000
      auth-code-expriation-millis: 1800000
```

```
  data:
    redis:
      host: ${REDIS_HOST}
      port: ${REDIS_PORT}
      password: ${REDIS_PASSWORD}
    mongodb:
      uri: ${MONGODB_URL}

  rabbitmq:
    username: ${RABBITMQ_DEFAULT_USER}
    password: ${RABBITMQ_DEFAULT_PASS}
    port: 5672
    host: rabbitmq

  fastAPI:
    URL: ${FAST_API_URL}
    ONE_URL: ${FAST_API_ONE_URL}

server:
  servlet:
    session:
      cookie:
        http-only: true
        path: /
        secure: true
        same-site: none
        max-age: 86400 # 하루
        domain: ${SERVER_DOMAIN}
      timeout: 90m
    context-path: /${CONTEXT_PATH}
  port: ${BACKEND_PORT}
```

2.빌드하기

CI/CD

Docker-compose.yml

```
version: "3.3"

services:

    nginx:
        image: ${DOCKER_IMAGE}:${DOCKER_TAG_FE}-latest
        container_name: ${DOCKER_TAG_FE}
        ports:
            - "80:80"
            - "443:443"
        environment:
            - DOCKER_TAG_BE=${DOCKER_TAG_BE}
            - BACKEND_PORT=${BACKEND_PORT}
            - SERVER_NAME=${SERVER_NAME}
            - CONTEXT_PATH=${CONTEXT_PATH}
            - REACT_APP_API_BASE_URL=${REACT_APP_API_BASE_URL}
            - REACT_APP_KAKAOMAP_KEY=${REACT_APP_KAKAOMAP_KEY}
            - REACT_APP_CHAT_SOCKET=${REACT_APP_CHAT_SOCKET}
            - REACT_APP_CHAT_DEST=${REACT_APP_CHAT_DEST}
            - DOCKER_TAG_FASTAPI=${DOCKER_TAG_FASTAPI}
            - TZ=Asia/Seoul
        volumes:
            - /home/ubuntu/data/certbot/conf:/etc/letsencrypt
            - /home/ubuntu/data/certbot/www:/var/www/certbot
        networks:
            - backend-network
        depends_on:
            - ${DOCKER_TAG_BE}
            - ${DOCKER_TAG_FASTAPI}

    certbot:
        image: certbot/certbot
        container_name: certbot
        volumes:
            - /home/ubuntu/data/certbot/conf:/etc/letsencrypt
```

```
      - /home/ubuntu/data/certbot/www:/var/www/certbot
    depends_on:
      - nginx
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew --webroot -w
/var/www/certbot sleep 60d & wait ${!}; done;'"

  springboot:
    image: ${DOCKER_IMAGE}:${DOCKER_TAG_BE}-latest
    container_name: ${DOCKER_TAG_BE}
    environment:
      - PROFILES_ACTIVE=${PROFILES_ACTIVE}
      - BACKEND_PORT=${BACKEND_PORT}
      - MYSQL_URL=${MYSQL_URL}
      - MYSQL_ROOT_USERNAME=${MYSQL_ROOT_USERNAME}
      - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
      - REDIS_HOST=${REDIS_HOST}
      - REDIS_PORT=${REDIS_PORT}
      - REDIS_PASSWORD=${REDIS_PASSWORD}
      - MONGODB_URL=${MONGODB_URL}
      - MAIL_USERNAME=${MAIL_USERNAME}
      - MAIL_PASSWORD=${MAIL_PASSWORD}
      - CONTEXT_PATH=${CONTEXT_PATH}
      - RABBITMQ_DEFAULT_USER=${RABBITMQ_DEFAULT_USER}
      - RABBITMQ_DEFAULT_PASS=${RABBITMQ_DEFAULT_PASS}
      - MONGO_INITDB_ROOT_USERNAME=${MONGO_INITDB_ROOT_USERNAME}
      - MONGO_INITDB_ROOT_PASSWORD=${MONGO_INITDB_ROOT_PASSWORD}
      - FAST_API_URL=${FAST_API_URL}
      - SERVER_DOMAIN=${SERVER_DOMAIN}
      - FAST_API_ONE_URL=${FAST_API_ONE_URL}
      - TZ=Asia/Seoul
    expose:
      - ${BACKEND_PORT}
    networks:
      - backend-network
    depends_on:
```

```
        - mysql
        - redis
        - mongodb
        - rabbitmq

fastapi:
    image: ${DOCKER_IMAGE}:${DOCKER_TAG_FASTAPI}-latest
    container_name: ${DOCKER_TAG_FASTAPI}
    environment:
        - TZ=Asia/Seoul
        - MYSQL_HOST=${MYSQL_HOST}
        - MYSQL_USER=${MYSQL_USER}
        - MYSQL_PW=${MYSQL_PW}
        - MYSQL_DB=${MYSQL_DB}
        - MYSQL_PORT=${MYSQL_PORT}
        - MONGO_HOST=${MONGO_HOST}
        - MONGO_PORT=${MONGO_PORT}
        - MONGO_USER=${MONGO_USER}
        - MONGO_PW=${MONGO_PW}
        - HOST=${HOST}
        - USER=${USER}
        - PASSWORD=${PASSWORD}
        - DATABASE=${DATABASE}
        - PORT=${PORT}
    expose:
        - "5000:5000"
    networks:
        - backend-network
    depends_on:
        - mysql
        - mongodb

rabbitmq:
    container_name: rabbitmq
    image: rabbitmq:3-management-alpine
```

```
    user: "1001:1001"
    init: true
    volumes:
      - /home/ubuntu/rabbitmq/data:/var/lib/rabbitmq
      - /home/ubuntu/rabbitmq/etc:/etc/rabbitmq
      - /home/ubuntu/rabbitmq/logs:/var/log/rabbitmq
    ports:
      - "5672:5672" # AMQP 프로토콜
      - "15672:15672" # 관리 UI
      - "61613:61613" # STOMP 프로토콜
      - "15674:15674" # Web STOMP (웹소켓)
    environment:
      - RABBITMQ_DEFAULT_USER=${RABBITMQ_DEFAULT_USER}
      - RABBITMQ_DEFAULT_PASS=${RABBITMQ_DEFAULT_PASS}
    networks:
      - backend-network

mysql:
    image: mysql:8.0.32
    container_name: mysql-con
    environment:
      - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
      - TZ=Asia/Seoul
    volumes:
      - mysql-vol:/var/lib/mysql
    ports:
      - "${MYSQL_BINDING_PORT}:3306"
    networks:
      - backend-network
    command: --lower_case_table_names=1

redis:
    image: redis:latest
    container_name: my-redis
    environment:
```

```
            - TZ=Asia/Seoul
        volumes:
            - redis_data:/data
            - ${REDIS_DEFAULT_CONFIG_FILE}:/usr/local/etc/redis/redis.conf
        ports:
            - "${REDIS_BINDING_PORT}:6379"
        command: redis-server /usr/local/etc/redis/redis.conf
        networks:
            - backend-network

    mongodb:
        image: mongo:latest
        container_name: mongodb-con
        volumes:
            - mongo-vol:/data/db
        environment:
            - TZ=Asia/Seoul
            - MONGO_INITDB_ROOT_USERNAME=${MONGO_INITDB_ROOT_USERNAME}
            - MONGO_INITDB_ROOT_PASSWORD=${MONGO_INITDB_ROOT_PASSWORD}
        ports:
            - "${MONGO_BINDING_PORT}:27017"
        networks:
            - backend-network
volumes:
    mysql-vol:
        external: true
    redis_data:
        external: true
    mongo-vol:
        external: true

networks:
    backend-network:
        name: backend-network
        driver: bridge
```

Dockerfile (nginx)

```
FROM nginx:alpine
LABEL authors="LEE JIHYE"

COPY ./frontend/moreroom/build /usr/share/nginx/html
COPY nginx.conf.template /etc/nginx/nginx.conf.template

EXPOSE 80
CMD ["/bin/sh", "-c", "envsubst '$DOCKER_TAG_BE $BACKEND_PORT $SERVER_NAME
$CONTEXT_PATH' < /etc/nginx/nginx.conf.template > /etc/nginx/nginx.conf && nginx -g
'daemon off;'"]
```

Dockerfile (fastAPI)

```
FROM python:3.10.8-slim

WORKDIR /code

COPY requirements.txt .
RUN pip install --no-cache-dir --upgrade -r requirements.txt
COPY . .

EXPOSE 5000
CMD ["uvicorn", "app:app", "--host", "0.0.0.0", "--port", "5000"]
```

Nginx.comf.template

```
events {
    worker_connections 1024;
}

http  {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
```

```
    sendfile on; # 로컬에 저장된 파일 전송

    gzip on;
    gzip_comp_level 5;
    gzip_types text/plain text/css application/json application/javascript text/xml
application/xml application/xml+rss text/javascript;

    server {
        listen 80;
        server_name ${SERVER_NAME};

        # Let's Encrypt 인증서 발급을 위한 설정
        location /.well-known/acme-challenge/ {
            allow all;
            root /var/www/certbot;
        }

        # HTTP 를 HTTPS 로 리다이렉트
        location / {
            return 301 https://$host$request_uri;
        }
    }

    server {
        listen 443 ssl;
        server_name ${SERVER_NAME};
        server_tokens off;

        ssl_certificate /etc/letsencrypt/live/${SERVER_NAME}/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/${SERVER_NAME}/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        # 보안 헤더
```

```
        add_header Strict-Transport-Security "max-age=31536000; includeSubDomains"
always;
        add_header X-Content-Type-Options nosniff;
        add_header X-Frame-Options DENY;
        add_header X-XSS-Protection "1; mode=block";
        # add_header Content-Security-Policy "default-src 'self'; script-src 'self' 'unsafe-
inline'";
        add_header Referrer-Policy "no-referrer-when-downgrade";

        # Spring Boot 외부 연결
        location /${CONTEXT_PATH}/ {
            proxy_pass <http://$>{DOCKER_TAG_BE}:${BACKEND_PORT};
            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;
        }

        # FAST API 외부 연결
        location /${CONTEXT_PATH}/fastapi/ {
            proxy_pass <http://fastapi:5000>;
            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;
        }

        # 소켓 설정
        location /${CONTEXT_PATH}/ws {
                proxy_pass <http://$>{DOCKER_TAG_BE}:${BACKEND_PORT};
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                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;
            proxy_read_timeout 20m;
        }

        location / {
            root /usr/share/nginx/html;
            index index.html index.htm;
            try_files $uri $uri/ /index.html;
        }
    }
}
```

Jenkins Pipeline

```
pipeline {
    agent any
    environment {
        DOCKERHUB_CREDENTIALS_ID = 'dockerhub-jenkins'
        DOCKERHUB_CREDENTIALS = credentials('dockerhub-jenkins')
        DOCKER_IMAGE = 'jihye9807/more-room'

        DOCKER_TAG_FE = "nginx"
        DOCKER_TAG_BE = "springboot"
        DOCKER_TAG_FASTAPI = "fastapi"
        backDockerImage = ''
        frontDockerImage = ''
        fastapiDockerImage = ''

        FIREBASE_CONFIG = credentials('firebase-config')
    }
    tools {
        jdk 'JDK 17'
        nodejs 'nodejs-20.17.0'
    }
```

```
    stages {
        stage('GitLab-Clone') {
            steps {
                git branch: 'develop', credentialsId: '4bb2afb3-351a-41f4-aaa9-dbbf0756fe91',
url: '<https://lab.ssafy.com/s11-bigdata-recom-sub1/S11P21D206>'
            }
        }
        stage('Build') {
            parallel {
                stage('BE-Build') {
                    steps {
                        echo "BE-Build stage: 백엔드 빌드"
                        dir("./backend") {
                            sh "chmod +x ./gradlew"
                            sh "./gradlew clean build -x test --stacktrace"
                        }
                    }
                }
                stage('FE-Build') {
                    steps {
                        echo "FE-Build stage: 프론트엔드 빌드"
                        dir("./frontend/moreroom") {
                            sh 'npm install'
                            sh 'CI=false npm run build'
                        }
                    }
                }
            }
        }
        stage('Build-Docker-Images') {
            parallel {
                stage('BE-Docker-Build') {
                    steps {
                        echo "BE Docker Build"
                        dir('./backend') {
```

```
                    script {
                        backDockerImage =
docker.build("${DOCKER_IMAGE}:${DOCKER_TAG_BE}-latest")
                    }
                }
            }
        }
        stage('FE-Docker-Build') {
            steps {
                echo "FE Docker Build"
                script {
                    frontDockerImage =
docker.build("${DOCKER_IMAGE}:${DOCKER_TAG_FE}-latest")
                }
            }
        }
        stage('FastAPI-Docker-Build') {
            steps {
                echo "FastAPI Docker Build"
                dir('./bigdata/fastapi') {
                    script {
                        fastapiDockerImage =
docker.build("${DOCKER_IMAGE}:${DOCKER_TAG_FASTAPI}-latest")
                    }
                }
            }
        }
    }
    stage('Push-Docker-Images') {
        steps {
            script {
                docker.withRegistry("", env.DOCKERHUB_CREDENTIALS_ID) {
                    backDockerImage.push()
                    frontDockerImage.push()
```

```
                    fastapiDockerImage.push()
                    backDockerImage.push("${DOCKER_TAG_BE}-${env.BUILD_NUMBER}")
                    frontDockerImage.push("${DOCKER_TAG_FE}-${env.BUILD_NUMBER}")
                    fastapiDockerImage.push("${DOCKER_TAG_FASTAPI}-
${env.BUILD_NUMBER}")
                }
            }
        }
    }
    stage('Deploy') {
        steps {
            echo 'Deploy stage'

            withCredentials([file(credentialsId: 'docker-env-file', variable:
'DOCKER_ENV_FILE')]) {
                sh '''
                    # Secret File 을 .env 로 복사
                    # .env 파일의 권한을 600 으로 설정
                    cp ${DOCKER_ENV_FILE} .env
                    chmod 600 .env
                    # 컨테이너 재시작, 기존 데이터 유지
                    docker compose up -d
                '''
            }
        }
    }
    stage('Cleanup') {
        steps {
            echo 'cleanup docker image'
            script {
                sh "docker image prune -f"
                [env.DOCKER_TAG_BE, env.DOCKER_TAG_FE,
env.DOCKER_TAG_FASTAPI].each { service ->
                    sh "docker rmi ${DOCKER_IMAGE}:${service}-${env.BUILD_NUMBER} ||
true"
```

```
                }
            }
        }
    }
    post {
        always {
            echo 'I complete CI/CD'
        }
        success{
            echo 'I success CI/CD'
            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: 'good',
                message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by
${Author_ID}(${Author_Name})₩₩n(<${env.BUILD_URL}|Details>)",
                endpoint:
'<https://meeting.ssafy.com/hooks/6jwxhq8653b19pewoahacg1hwy>',
                channel: 'cicd'
                )
            }
        }
        failure{
            echo 'I fail CI/CD'
            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: 'danger',
                message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by
${Author_ID}(${Author_Name})₩₩n(<${env.BUILD_URL}|Details>)",
```

```
                endpoint:
'<https://meeting.ssafy.com/hooks/6jwxhq8653b19pewoahacg1hwy>',
                channel: 'cicd'
                )
            }
        }
    }
}
```

3.6. DB 접속 정보 등 프로젝트(ERD) 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록

- .env