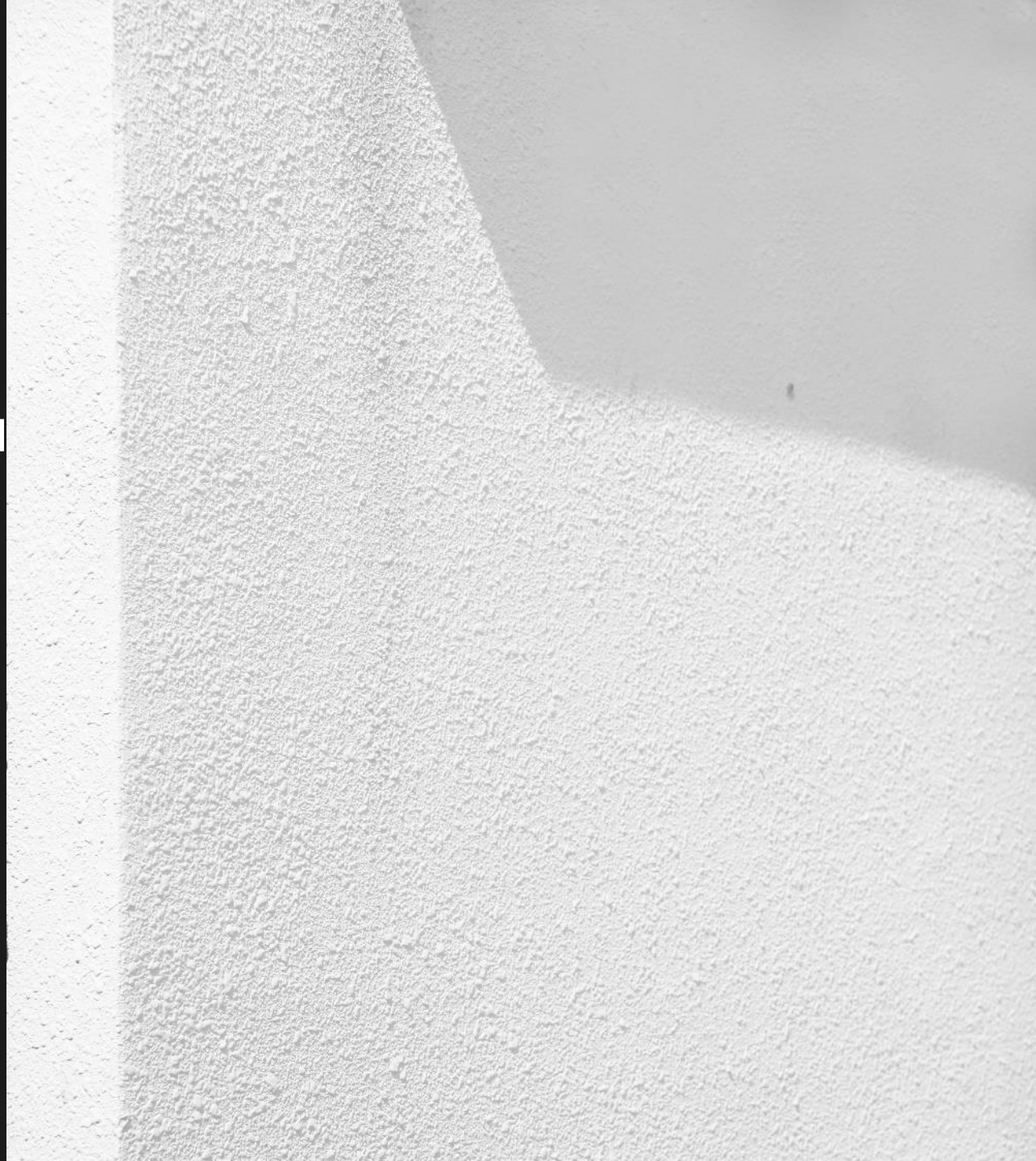
FLOW

목차

Part 1 프로젝트 소개



개요

"FLOW"는 Fashion Lifestyle Outfit Worker의 약자로 직장인을 대상으로 한 패션 Social Network Service입니다.

이미 다양한 패션 관련 Platform, SNS는 존재 하였으나 직장인을 타겟으로 한 서비스는 없었기 때문에 직장인이라는 주제에 집중하여 기획, 개발 하게 되었습니다.

같은 업종, 연차 등을 통한 필터링 기능 및 제품 구매에 의견을 구할 수 있는 투표 시스템에 주안점을 두고 프로젝트를 진행하였습니다.

프로젝트 소개

사용 기술 / 언어

Open API

NAVER OpenAPI

Kakao Developers

Google APIs

Front-End

- HTML 5
- CSS 3
- Javascript
- SCSS
- React
- Redux
- React-query

Back-End

- Java
- Spring-Boot
- MySql
- JPA

IDE/Tools

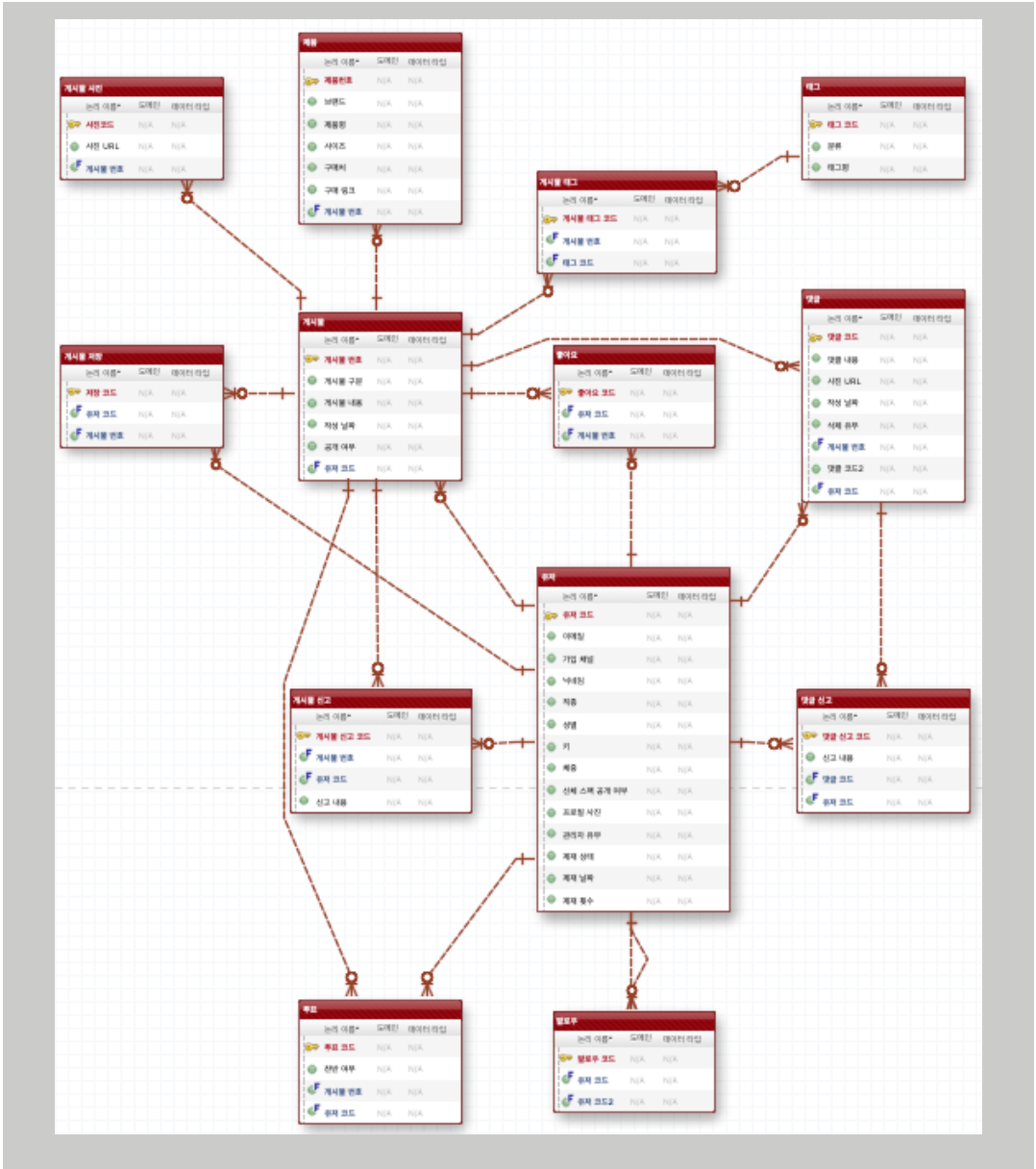
- Visual Studio Code
- IntelliJ
- PostMan
- GitHub

프로젝트 소개

DB Modeling

각 기능 별 구현이 가능하도록 테이블을 구성하였습니다.

주로 게시물 테이블과 유저 테이블에서 파생되어 생긴 테이블의 수가 많습니다.



요구사항 목록

분류	요구사항 ID	요구사항 명	세부항목	선행 요구사항	TODO
회원 관리	LO-001	회원가입	네이버 계정으로 가입		구현완
			카카오 계정으로 가입		구현완
			구글 계정으로 가입		구현완
	LO-002	로그인	네이버 계정으로 로그인		구현완
			카카오 계정으로 로그인		구현완
			구글 계정으로 로그인		구현완
	LO-003	로그아웃	사이트에서 로그아웃	LO-002	구현완
	LO-004	회원수정	회원정보 수정	LO-001	구현완
	LO-005	회원탈퇴	사이트에서 회원 탈퇴	LO-001	구현완
마이페이지	MY-001	북마크한 게시물 조회	내가 북마크한 게시물 목록 조회	SH-007	구현완
			북마크한 게시물의 수 조회	"	구현완
	MY-002	좋아요한 게시물 조회	내가 좋아요한 게시물 목록 조회	SH-006	구현완
			좋아요한 게시물의 수 조회	"	구현완
	MY-003	작성한 게시물 조회	내가 작성한 게시물 목록 조회		구현완
			작성한 게시물의 수 조회		구현완
	MY-004	팔로잉 조회	내가 팔로우한 회원 수 조회	FL-001	구현완
			내가 팔로우한 회원 목록 조회	"	구현완
	MY-005	팔로워 조회	나를 팔로우한 회원 수 조회	"	구현완
			나를 팔로우한 회원 목록 조회	"	구현완
	MY-006	내가 작성한 투표현황 조회	내가 작성한 투표현황/결과 조회	TP-004	구현완
팔로우	FL-001	다른 회원 팔로우	다른회원의 계정을 팔로우		구현완
	FL-002	팔로우 취소	팔로우한 계정 팔로우 취소	FL-001	구현완
게시물 (업로드)	UP-001	게시물 업로드	게시물 추가		구현완
			게시물에 제품 추가	UP-001-게시물 추가	구현완
			게시물에 태그 추가	"	구현완
	UP-002	게시물 수정	게시물 내용/공개여부 수정	UP-001	구현완
			게시물 사진 삭제	"	구현완
			게시물 제품 삭제	"	구현완
			게시물 제품 추가	"	구현완
			게시물 태그 삭제	"	구현완
			게시물 태그 추가	"	구현완
	UP-003	게시물 삭제	작성한 게시물 삭제	"	구현완
게시물 조회	SH-001	게시물 목록 조회	전체 게시물 목록 조회	SH-001	구현완
	SH-002	게시물 좋아요 순 조회	게시물 좋아요순으로 조회	"	구현완
	SH-003	내가 팔로우한 유저 게시글 조회	내가 팔로우한 유저 게시글 조회	SH-002	구현완
	SH-004	카테고리별 게시물 조회	직종별/복장규정 게시물 조회	SH-001	구현완
	SH-005		성별/키/체중 등 필터링 하여 조회	"	구현완
	SH-006	게시물 태그로 조회	검색한 내용에 포함된 태그 조회	"	구현완
	SH-006	게시물 좋아요	게시물 좋아요		구현완
			좋아요 취소		구현완
	SH-007	게시물 북마크	게시물 북마크		구현완
			북마크 삭제		구현완
	SH-008	게시물 신고	게시물 신고		구현완
	SH-009	게시물 상세 페이지 조회	게시물 상세 페이지 조회		구현완
댓글	CM-001	댓글 작성	댓글 작성		구현완
	CM-003	댓글 수정	작성자만 댓글 수정	"	구현완
	CM-004	댓글 삭제	작성자만 댓글 삭제	"	구현완
	CM-005	댓글 신고	부적절한 댓글 신고		구현완
투표	TP-001	투표 작성	투표 작성	LO-002	구현완
	TP-002	찬성 / 반대 선택	찬성 혹은 반대 선택	LO-002	구현완
	TP-004	투표 삭제	투표 삭제	LO-002	구현완
	TP-005	투표목록 조회	전체 투표 목록 조회		구현완
	TP-006	투표현황 조회	현재 투표의 현황 조회		구현완
관리자	AD-001	게시물 삭제	신고한 게시물 삭제	SH-008	구현완
	AD-002	댓글 삭제	신고한 댓글 삭제	CM-005	구현완
	AD-003	회원 제재	신고받은 회원을 일정기간(7/15/30) 접속 불가하도록 제재	LO-002	구현완

각 기능 별 요구사항을 정리 후 역할 분배를 하고,
세부 기능을 재 정립했습니다.

그 후 세부기능 중 선행으로 개발되어야 하는 기능이
있는 경우 따로 표기를 해서 팀 프로젝트 중에 혼선이
생기지 않도록 방지 했습니다.

Part 2 담당 업무

Part 2 >>

담당 업무

- Open API 사용하여 로그인 / 회원가입 기능 구현
  - API는 Naver, Kakao, Google 로그인 API 활용
  - 개발 문서 참고하여 개발 진행
- Context 사용하여 로그인 정보 전역 설정
- 회원 수정 기능 구현
- 회원 탈퇴 기능 구현
- 프로젝트 취합 및 총괄
- 일부 페이지 퍼블리싱 작업
  - 마이페이지 퍼블리싱 수행
  - 회원 가입 페이지 퍼블리싱 작업 수행
  - 회원 수정 페이지 퍼블리싱 작업 수행
  - 상세페이지 퍼블리싱 작업 수행

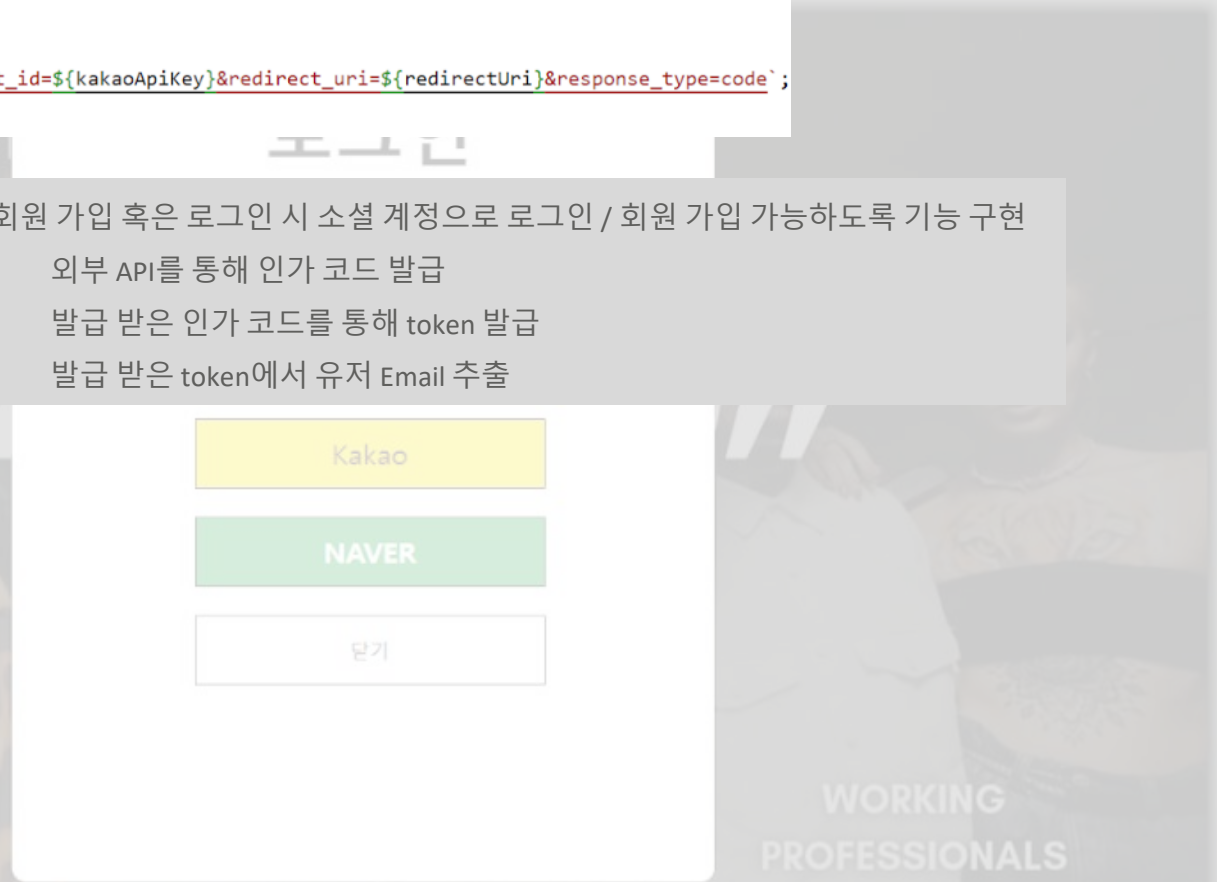
Open API 로그인 / 회원가입

```
// 인증 코드 발급
export const getKakaoCode = (type) => {
  let redirectUri;

  if (type === "register") redirectUri = redirectRegisterUri;
  // 회원가입
  else redirectUri = redirectLoginUri; // 로그인
  window.location.href = `https://kauth.kakao.com/oauth/authorize?client_id=${kakaoApiKey}&redirect_uri=${redirectUri}&response_type=code`;
};
```

```
// 토큰 발급 후 유저 이메일 추출
export const getKakaoToken = async (code, type) => {
  let redirectUri;

  if (type === "register") redirectUri = redirectRegisterUri;
  // 회원가입
  else redirectUri = redirectLoginUri; // 로그인

  const res = await axios.post(
    "https://kauth.kakao.com/oauth/token",
    {
      grant_type: "authorization_code",
      client_id: kakaoApiKey,
      redirectUri: redirectUri,
      code: code,
      client_secret: kakaoSecret,
    },
    {
      headers: {
        "Content-type": "application/x-www-form-urlencoded;charset=utf-8",
      },
    }
  );

  // 토큰 발급
  const token = res.data.access_token;

  const userData = await axios.get(`https://kapi.kakao.com/v2/user/me`, {
    headers: {
      Authorization: `Bearer ${token}`,
      "Content-type": "application/x-www-form-urlencoded;charset=utf-8",
    },
  });

  // 유저 이메일
  return userData.data.kakao_account.email;
};
```

회원 가입 혹은 로그인 시 소셜 계정으로 로그인 / 회원 가입 가능하도록 기능 구현

- 외부 API를 통해 인가 코드 발급
- 발급 받은 인가 코드를 통해 token 발급
- 발급 받은 token에서 유저 Email 추출



Open API 로그인 / 회원가입

```
// 4. 해당 회원이 가입되어 있는지 조회
const loginCheck = async () => {
  const token = await userCheck({
    userEmail: userEmail,
    userPlatform: "kakao",
    type: "login",
  });
  if (token !== null) login(token);
  navigate("/");
};
```

```
// 로그인
@PostMapping("/login")  no usages  Dong Yeop-Lee +1
public ResponseEntity login(@RequestBody User vo){
    Map<String, String> userMap = userService.login(vo);
    if(userMap.get("ban").equals("Y")){
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(userMap.get("token"));
    }
    return ResponseEntity.status(HttpStatus.OK).body(userMap.get("token"));
}
```

```
// 로그인
public Map<String, String> login(User vo){  1 usage  Dong Yeop-Lee
    User user = userDao.duplicateCheck(vo.getUserEmail(), vo.getUserPlatform()).get();

    Map<String, String> response = new HashMap<>();
    String token = tokenProvider.create(user);
    response.put("token", token);

    int banCount = user.getUserBanCount();

      밴 상태일때 로그인 금지 (4회 이상은 그냥 금지)
    if(user.getUserBanStatus().equals("Y")) {
        if(banCount == 1) {
            Duration duration = Duration.between(LocalDateTime.now(), user.getUserBanDate().plusDays(7));
            if(duration.isPositive()) response.put("ban", "Y");
            else if(duration.isNegative()) response.put("ban", "N");
        } else if(banCount == 2) {
            Duration duration = Duration.between(LocalDateTime.now(), user.getUserBanDate().plusDays(15));
            if(duration.isPositive()) response.put("ban", "Y");
            else if(duration.isNegative()) response.put("ban", "N");
        } else if(banCount == 3) {
            Duration duration = Duration.between(LocalDateTime.now(), user.getUserBanDate().plusMinutes(1));
            if(duration.isPositive()) response.put("ban", "Y");
            else if(duration.isNegative()) response.put("ban", "N");
        } else if(banCount >= 4) response.put("ban", "Y");
    } else if(user.getUserBanStatus().equals("N")) response.put("ban", "N");

    if(response.get("ban").equals("N")){
        user.setUserBanStatus("N");
        user.setUserBanDate(null);
        userDao.save(user);
    }

    return response;
}
```

```
// 회원 로그인 or 회원가입
export const userCheck = async (user) => {
  const result = await instance.get(
    `duplicateCheck?userEmail=${user.userEmail}&userPlatform=${
      user.userPlatform
    }`
  );

  if (!result.data) {
    if (user.type === "login") {
      try {
        const response = await instance.post("login", {
          userEmail: user.userEmail,
          userPlatform: user.userPlatform,
        });
        return response.data;
      } catch (error) {
        alert("계정이 일시정지 되었습니다.");
      }
      return null;
    } else {
      alert("이미 가입한 회원입니다.");
      window.location.href = "/";
    }
  } else {
    if (user.type === "login") {
      alert("회원가입 후 이용해 주세요.");
      return null;
    } else {
      window.location.href = `/registerUser?userEmail=${
        user.userEmail
      }&userPlatform=${user.userPlatform}`;
    }
  }
};
```

- 유저 Email과 소셜 계정 구분자를 통해 Server에 Token 요청
- Client를 통해 받은 유저정보를 토대로 회원 정보 조회
- Token Provider를 통해 Token 생성
- 일시정지 된 계정일 경우 정지 횟수에 따라 계정 정지
- HashMap 사용하여 Token 정보와 정지 유무 구분자를 Controller에 전달
- Controller는 계정정지 유무에 따라 Client에 각기 다른 응답을 시행

FLOW 회원정보 수정

```
// 회원 수정
const updateSubmit = async () => {
  const formData = new FormData();
  formData.append("userCode", user.userCode);
  formData.append("userEmail", user.userEmail);
  formData.append("userPlatform", user.userPlatform);
  formData.append("userJob", user.userJob);
  formData.append("userHeight", user.userHeight);
  formData.append("userWeight", user.userWeight);
  formData.append("userBodySpecYn", user.userBodySpecYn);
  formData.append("userProfileUrl", user.userProfileUrl);
  formData.append("userNickname", user.userNickname);
  formData.append("userGender", user.userGender);
  if (user.imgFile != null) formData.append("imgFile", user.imgFile);
  await updateUser(formData);
  navigate(`/mypage/${user.userCode}`);
};
```

- 회원 이미지 / 닉네임 / 직종 / 신장 / 체중 / 신체정보 공개 여부 수정
  - 이미지 업로드로 인해 FormData 사용하여 Server로 요청

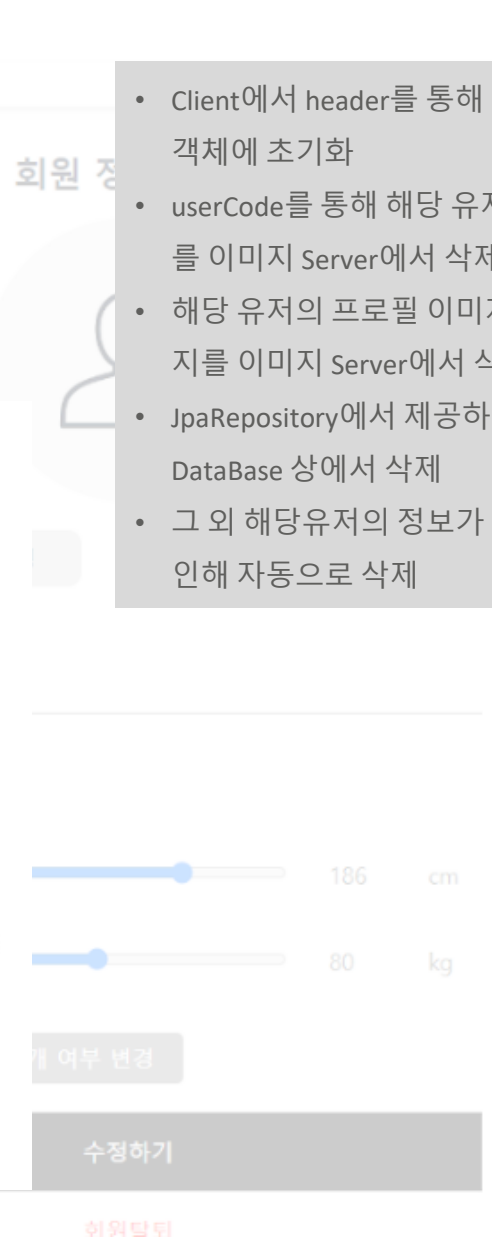
- DTO를 통해 회원 Code(고유 값)을 추출하여 수정 전의 회원정보를 가지고 있는 User 객체 생성
- 프로필 이미지 변경 시 적용 중이던 이미지가 Default 이미지인지 확인 후 그에 맞게 비즈니스 로직 적용
- 변경된 정보들은 user 객체에 setter 메서드를 통해 변경된 값 대입
- JpaRepository에서 제공하는 save 메서드를 통해 변경된 정보를 DataBase에 Update

```
// 유저 정보 수정
public void updateUser(UserUpdateDTO dto) throws IOException {  1 usage   Dong Yeop-Lee

    User user = userDao.findById(dto.getUserCode()).get();

    // 이미지 변경을 했으면 실행
    if(dto.getImgFile() != null){
        if(!user.getUserProfileUrl().equals("http://192.168.10.51:8081/userImg/defaultUser.png")){
            File deleteFile =  new File( pathname: userImgDir + new File(user.getUserProfileUrl()).getName());
            deleteFile.delete();
        }
        // 파일이름 랜덤으로 생성
        UUID uuid = UUID.randomUUID();
        String fileName = uuid.toString() + "_" + dto.getImgFile().getOriginalFilename();
        File copyFile = new File( pathname: userImgDir + fileName);
        // 이미지 서버에 이미지 저장
        dto.getImgFile().transferTo(copyFile);

        user.setUserProfileUrl("http://192.168.10.51:8081/userImg/" + fileName);
    }else{
        // 프로필 이미지가 기본이미지로 바뀜
        if(!dto.getUserProfileUrl().equals(user.getUserProfileUrl()) && !user.getUserProfileUrl().equals("http://192.168.10.51:8081/userImg/defaultUser.png")){
            File deleteFile =  new File( pathname: userImgDir + new File(user.getUserProfileUrl()).getName());
            deleteFile.delete();

            user.setUserProfileUrl("http://192.168.10.51:8081/userImg/defaultUser.png");
        }
    }
    user.setUserNickname(dto.getUserNickname());
    user.setUserJob(dto.getUserJob());
    user.setUserHeight(dto.getUserHeight());
    user.setUserWeight(dto.getUserWeight());
    user.setUserBodySpecYn(dto.getUserBodySpecYn());

    userDao.save(user);
}
```

FLOW 회원 탈퇴

```java
// 사용자 정보 가져오는 메서드
private User getUser(){  4 usages  Dong Yeop-Lee
    try{
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        if(auth != null && auth.isAuthenticated()){
            return (User) auth.getPrincipal();
        }
    }catch (SignatureException e){
        return null;
    }
    return null;
}

// 유저 탈퇴
public void deleteUser(){  1 usage  Dong Yeop-Lee
    User user = userDao.findById(getUser().getUserCode()).get();
    // 유저가 생성한 게시글 조회
    List<Post> post = postDAO.findByUser_UserCode(user.getUserCode());

    // 이미지 서버에 존재하는 유저의 게시물 이미지들 삭제
    for(Post p : post){
        // 게시물 마다의 이미지 객체 받아옴
        List<PostImg> imgs = postImgDAO.findByPost_PostCode(p.getPostCode());
        for(PostImg postImg : imgs){
            String url = postImg.getPostImgUrl();
            String fileNmae = url.substring( beginIndex: url.lastIndexOf( str: "\\") + 1);
            File file = new File( pathname: postImgpath + fileNmae);
            file.delete();
        }
    }

    if(!user.getUserProfileUrl().equals("http://192.168.10.51:8081/userImg/defaultUser.png")){
        String url = user.getUserProfileUrl();
        String fileName = url.substring( beginIndex: url.lastIndexOf( str: "/") + 1);
        File file = new File( pathname: userImgDir + fileName);
        file.delete();
    }

    userDao.deleteById(getUser().getUserCode());
}
```

- Client에서 header를 통해 넘겨준 Token 정보를 통해 User 정보를 추출 후 User 객체에 초기화
- userCode를 통해 해당 유저가 생성한 게시물 조회 후 게시물에 포함된 이미지를 이미지 Server에서 삭제
- 해당 유저의 프로필 이미지가 Default 이미지가 아닐 경우 유저의 프로필 이미지를 이미지 Server에서 삭제
- JpaRepository에서 제공하는 deleteById 메서드를 사용해 해당 유저의 정보를 DataBase 상에서 삭제
- 그 외 해당유저의 정보가 있는 테이블의 값들은 DataBase 내 casecade 설정으로 인해 자동으로 삭제



Part 3 트러블 슈팅

트러블 슈팅

- 문제 발생
  - Naver API 적용 중 회원가입 / 로그인 시도 중 Cors 에러가 발생하는 현상
- 사실 수집
  - Client 내 Log Message 내용을 통해 오류 발생 시점 확인
- 원인 추론
  - Naver API는 보안상의 이유로 Client에서 REST 방식으로 API 사용을 할 시 Cors 에러가 발생
- 해결 과정
  1. Project 내에 Proxy 설정을 할 시 token 발급 가능
  2. token 발급 후 token 내 정보 요청 시 요청 주소가 지정 되어있음에도 주소를 찾지 못하는 현상 발생
  3. Proxy 설정에 scope 추가 시도를 해봤지만 같은 현상 반복
- 문제 해결
  - Token과 token 정보 요청을 Server에서 요청하는 방식으로 해결

Part 4 개선 점

Part 4 >>

개선 점

많은 인원이 투입되는 팀 프로젝트인 만큼 기능적으로 더 많은 걸 넣고 싶었지만 인원 대다수가 React 라이브러리에 익숙하지 않은 상태라 필수 기능 구현에 더욱 집중하게 되었습니다.

여러 인원이 참여한 프로젝트라 코드가 정리 되지 않고 작성되는 경우도 많았고 코드를 취합하는 과정에서 코드 리팩토링에 시간을 많이 투자를 했습니다.

그럼에도 프로젝트 내 코드 정리가 되어있지 않다고 느껴져서 전체적으로 코드들을 수정하는 시간을 가져야 할 것 같습니다.

추후 다른 프로젝트 진행 시에도 이런 점을 유의해서 코드 작성을 할 것 입니다.

Fin.