

PROEJCT 02 - 2조

휴먼클라우드 웹 - RestAPI

2조 - 김주혁, 박찬혁, 박선규, 송민경, 장유진



목차

1. 프로젝트 소개 및 목표
2. 목표
3. 협업전략
4. Base64
5. 리팩토링
6. AOP
7. JWT
8. Junit테스트
9. API 문서

프로젝트 소개 및 목표

달성률 : 100%

1차 프로젝트 휴먼클라우드 웹을 리팩토링하여 Rest API를 전환을 통해 백엔드의 역할과 json 확장성에 대해 이해하고 aws 배포하기

1. JPA Respository 리팩토링 & JPQL 문법
2. Service로 트랜잭션을 모으고 Controller 간소화
3. SSR 배포 설정완료
4. Service에서 Lazyloading 액세스 완료하고 ResponseDTO로 응답
5. Rest API 전환 & 포스트맨
6. AOP 유효성 검사적용
7. JWT(Json Web Token) 적용
8. Junit테스트
9. API 문서
10. EC2 AWS 배포

개발환경

Tools



POSTMAN

Language



VCS



Framework



DB



협업전략 - 스케줄 관리

스케줄

◀ 백링크 1개

☰ 보드 田 표 □ 캘린더 보기 1개 더 보기 필터 정렬 🔍 🏠 ... 새로 만들기 ▾

2차 스케줄 ...

팀 11 ... +

프로젝트 마감

● 시작 전

DTO응답하기

● 완료

Rest API 전환

● 완료

엔토링

● 완료

Rest API 배포

● 시작 전

Rest API시연영상 찍기

● 완료

SSR배포 및 시연

● 시작 전

api컨트롤러 주소 합치기

● 완료

API문서

● 완료

JUnit Test

● 진행 중

회원가입 ResquestDTO 나누기

● 완료

+ 새로 만들기

김주혁 10

주말 : skill

● 완료

이력서등록

● 완료

광고삭제

● 완료

더미 수정

● 완료

User

● 완료

User DTO(Guset,Comp)

● 완료

배포 실습

● 완료

뷰 통일하기

● 완료

사진 Base64 디코딩

● 완료

PPT 만들기

● 진행 중

+ 새로 만들기

박찬혁 9

이력서수정

● 완료

ApplyForm, PostionForm JPQL

● 완료

광고등록

● 완료

주말 : scrap

● 완료

Scrap DTO

● 완료

배포 실습

● 완료

업데이트 시 체크박스 활성화

● 완료

api컨트롤러 주소 합치기

● 완료

회원가입 ResquestDTO 나누기

● 완료

+ 새로 만들기

박선규 6

광고수정

● 완료

이력서보기

● 완료

주말 : board, reply

● 완료

Jobopen, Resume DTO

● 완료

배포 실습

● 완료

광고 DTO

● 완료

+ 새로 만들기

송민경 15

광고보기

● 완료

주말 : Pic

● 완료

뷰 분리

● 완료

이력서 삭제

● 완료

상세보기 수정

● 완료

Board, Reply DTO

● 완료

배포 실습

● 완료

최신이력서목록

● 완료

만나이로 변경하기

● 완료

사진 미리보기

● 완료

사진 null일 때 해결하기

● 완료

게시판, 댓글 전환 완료

● 완료

장유진 7

주말 : Apply

● 완료

Apply DTO

● 완료

배포 실습

● 완료

ScoutList

● 완료

Joinform

● 완료

User, Comp

● 완료

JWT

● 완료

+ 새로 만들기

스케줄

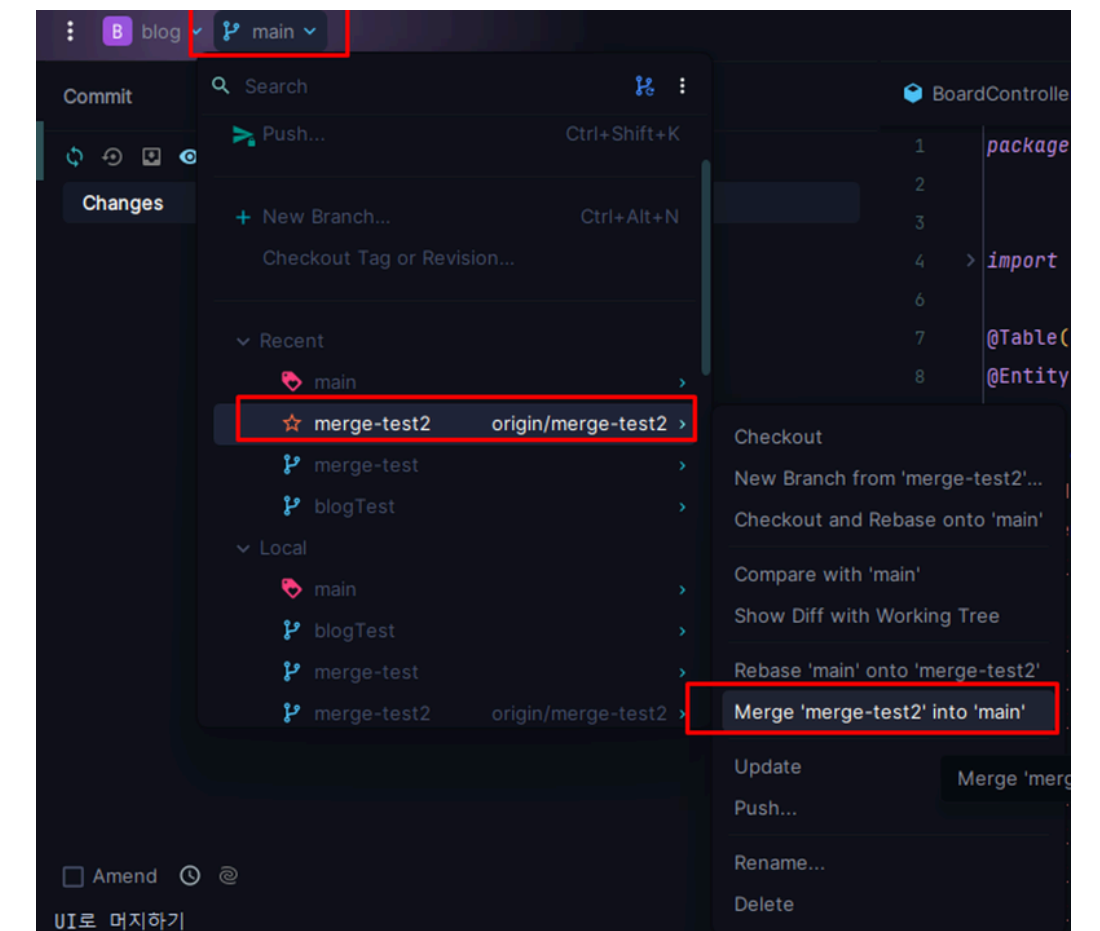
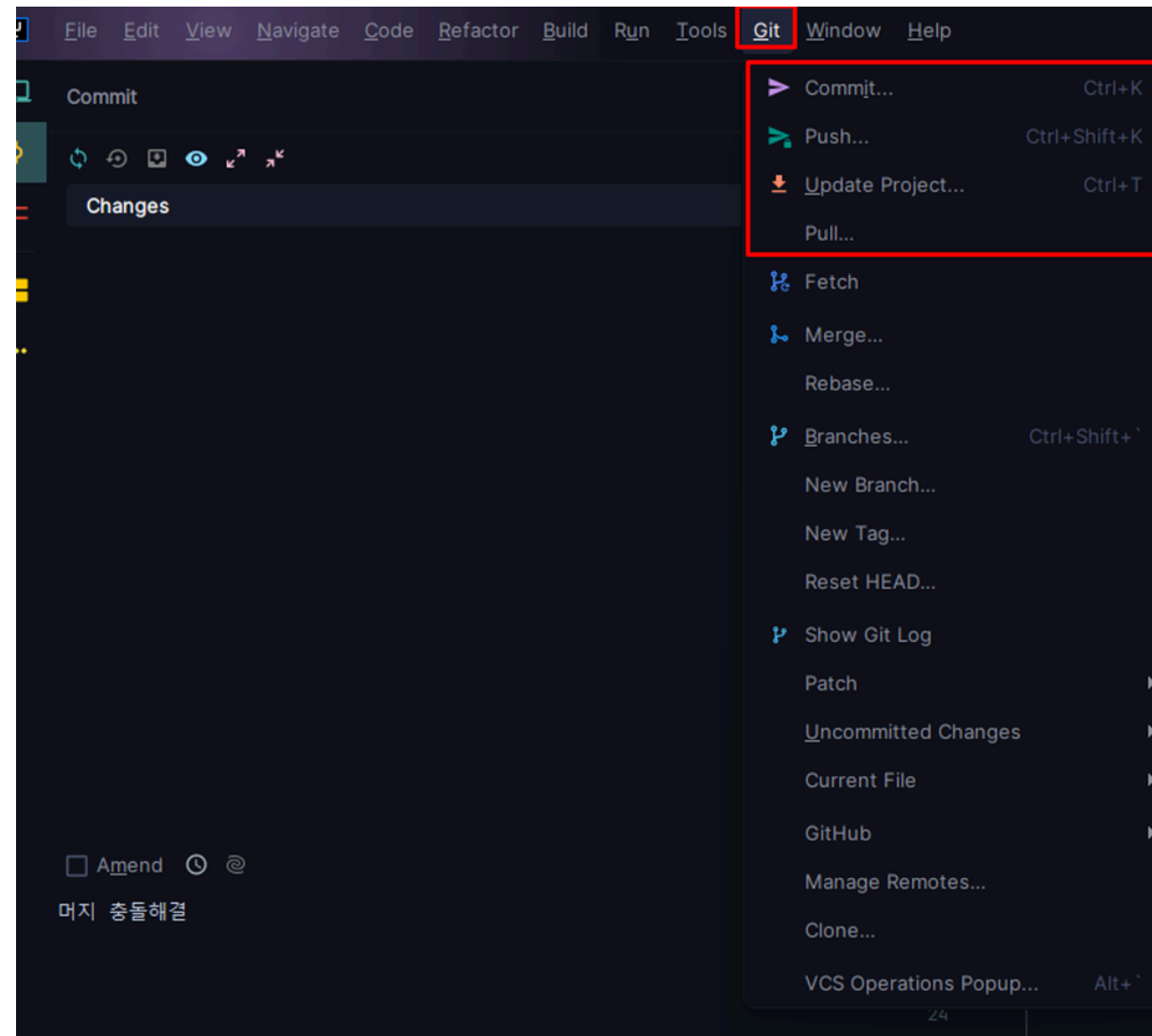
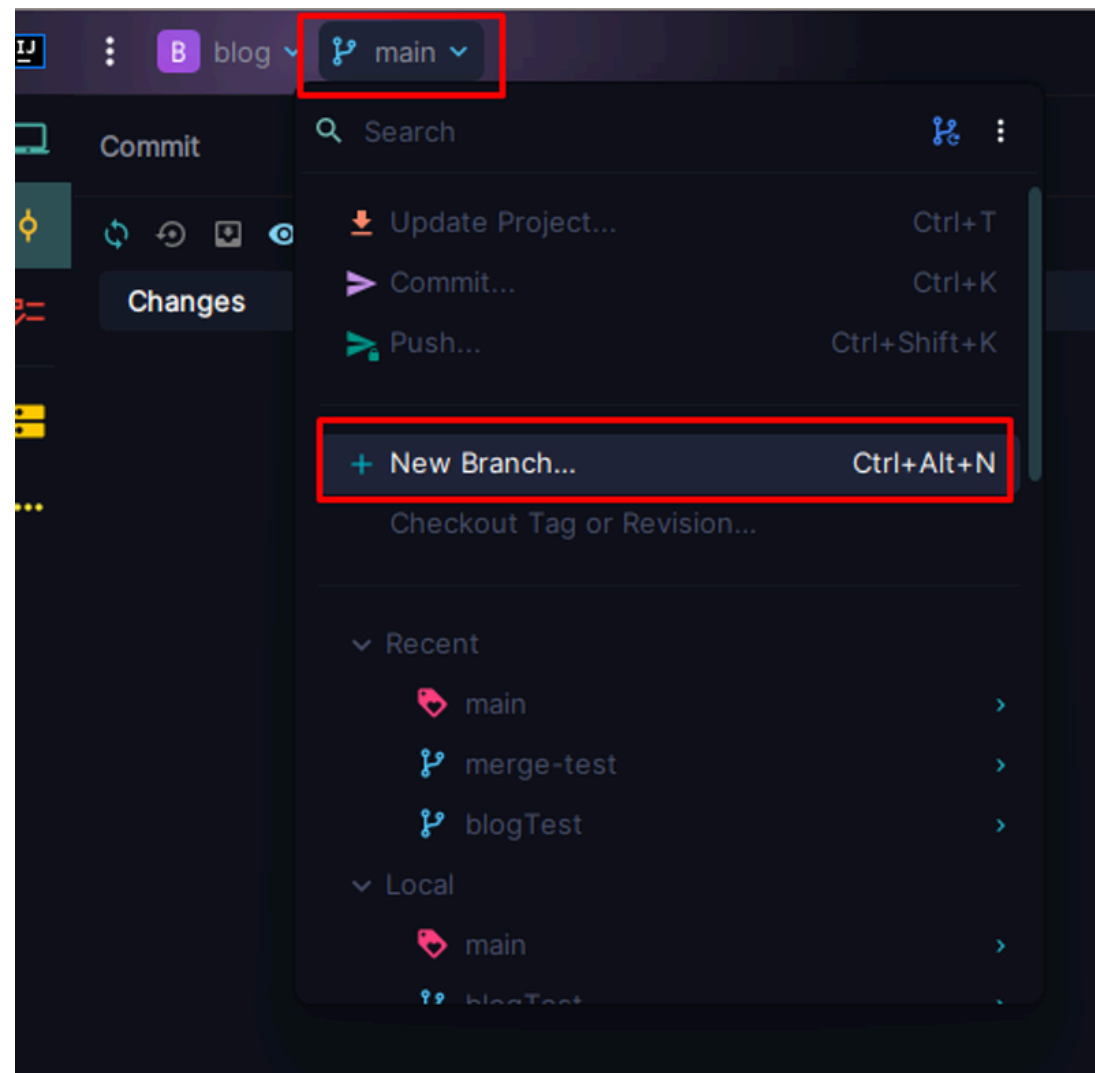
◀ 백링크 1개

+ :: ☰ 보드 田 표 □ 캘린더 보기 1개 더 보기 필터 정렬 🔍 🏠 ... 새로 만들기 ▾

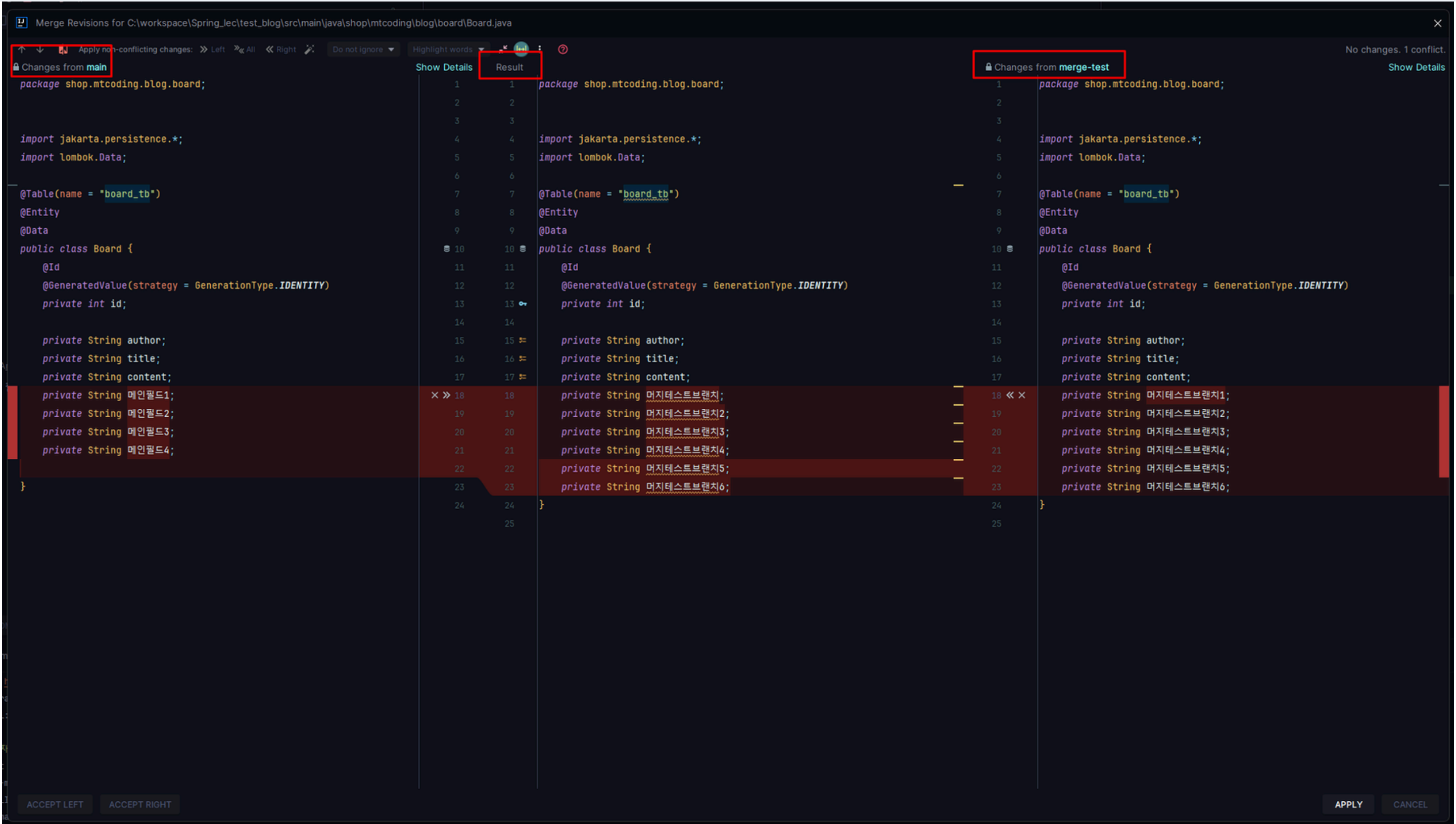
2차 스케줄 ...

Aa 이름	☀ 상태	📅 날짜	👤 태그	📄 텍스트	+ ...
사진 Base64 디코딩	● 완료	2024년 4월 1일	김주혁		
PPT 만들기	● 진행 중	2024년 4월 2일 → 2024년 4월 :	김주혁		
뷰 통일하기	● 완료	2024년 3월 29일 → 2024년 3월	김주혁		
배포 실습	● 완료	2024년 3월 27일	김주혁 박찬혁 박선규		
📄 User DTO(Guset,Comp)	● 완료	2024년 3월 25일 → 2024년 3월	김주혁		
📄 더미 수정	● 완료	2024년 3월 25일	김주혁		
📄 주말 : skill	● 완료	2024년 3월 22일 → 2024년 3월	김주혁		
📄 이력서등록	● 완료	2024년 3월 22일	김주혁		
📄 광고삭제	● 완료	2024년 3월 22일	김주혁		
📄 User	● 완료	2024년 3월 21일	김주혁		
회원가입 ResquestDTO 나누기	● 완료		박찬혁		
📄 api컨트롤러 주소 합치기	● 완료		박찬혁		
ApplyForm, PostionForm JPQL	● 완료		박찬혁		
📄 업데이트 시 체크박스 활성화	● 완료	2024년 3월 29일 → 2024년 3월	박찬혁		
📄 Scrap DTO	● 완료	2024년 3월 25일 → 2024년 3월	박찬혁		
📄 주말 : scrap	● 완료	2024년 3월 22일 → 2024년 3월	박찬혁		
📄 이력서수정	● 완료	2024년 3월 22일 → 2024년 3월	박찬혁		
📄 광고등록	● 완료	2024년 3월 22일	박찬혁		
광고 DTO	● 완료		박선규		
📄 Jobopen, Resume DTO	● 완료	2024년 3월 25일 → 2024년 3월	박선규		
📄 주말 : board, reply	● 완료	2024년 3월 22일 → 2024년 3월	박선규		
📄 이력서보기	● 완료	2024년 3월 22일 → 2024년 3월	박선규		
📄 광고수정	● 완료	2024년 3월 22일	박선규		
AOP	● 완료		송민경		
게시판, 댓글 응답DTO 전환	● 완료	2024년 3월 31일	송민경		
게시판, 댓글 유효성 추가	● 완료	2024년 4월 1일	송민경		
게시판, 댓글 전환 완료	● 완료		송민경		
사진 null일 때 해결하기	● 완료		송민경	사진 빼고 업데이트할 때 기존	
최신이력서목록	● 완료	2024년 3월 29일	송민경	인재채용에 각 유저별 최신 이력	
만나이로 변경하기	● 완료	2024년 3월 29일	송민경	생년월일로 받은 데이터를 만 나	

협업전략 - Git



협업전략 - Git



Base 64

data:이미지 이름/확장자:base64, 문자열...

data:image/png;base64, 문자열...

협업전략 - 프로젝트 블로그

프로젝트 블로그

🔔 동기화를 해도 되고 링크를 공유해서 링크를 넣어놔도 됩니다~ 한게 있으면 뭐든 블로그 해주세요~! 이거 보고 코드리뷰나 의견 나누겠습니다.

🔗 링크로 공유하기

🔗 코드리뷰(아래 중 택1)

1. 비즈니스 로직 설명 및 어떤식으로 구현했는지 설명

2. 코드를 리팩토링 하면서 전 후로 비교하여 어떤 부분을 어떻게 사용하면 좋은지 설명

3. 막힌 부분을 어떻게 진행하면 좋을지 의견구하기

🔗 상세보기 (응답 DTO만들기 예시) 각 진행사항 안에 넣으면 됩니다.

🔗 민경 진행상황

🔗 찬혁 진행상황

🔗 선규 진행 상황

🔗 유진 진행상황

🔗 주혁 진행사항

🔗 DTO 총정리

Project 정리

🔗 백링크 1개

Project1. SSL 구축사이트

- 🔗 1. 이력서 사진 업로드
- 🔗 2. 이력서 사진 수정하기
- 🔗 3. 채용공고 사진 업로드
- 🔗 4. 채용공고 사진 수정하기
- 🔗 5. 메인 페이지에 사진 매칭하기

📖 :: Project2. CSL 구축사이트

- SERVICE 레이어 만들어 CONTROLLER가 의존하기

- 🔗 1. 공고 상세보기
- 🔗 2. 이력서 삭제하기
- 🔗 3. 개인 프로필사진 업데이트
- 🔗 4. 기업 프로필사진 업데이트
- DTO 그려보기
- 🔗 5. 게시판, 댓글 응답 DTO 그려보기
- 🔗 게시판 목록보기 DTO 그려보기
- 🔗 게시물 쓰기 DTO 그려보기
- 🔗 게시물 수정하기 DTO 그려보기
- 수정하기
- 🔗 6. 이력서 상세보기 페이지 수정하기
- 🔗 7. 채용공고 상세보기 수정하기
- service 점검하기
- 🔗 service 현황
- 사진
- 🔗 Preview 되는 로직 필요
- 🔗 null일때 수정하지 않는 로직 필요
- 이력서
- 🔗 생년월일을 만나일로 변환하기
- 🔗 최신 이력서만 조회하기
- RestAPI 전환
- 🔗 게시판 전환하기

- :: 🔗 공고 등록 Skill 테이블 삭제 후 List로 받기
- :: 🔗 의미 있는 set만들기
- 🔗 strem으로 배열파싱
- 🔗 응답DTO만들기
- 🔗 모달 창 응답 DTO
- 🔗 페이징/ 유틸에 만들고 DI로 쓰기
- 🔗 임포트 및 컴파일 안될때
- 🔗 base64 변환하기

Scrap

- 🔗 회사가 스크랩
- 🔗 게스트가 스크랩
- 🔗 회사가 스크랩 한 목록 보기
- 🔗 게스트가 스크랩 한 공고 목록 보기
- 🔗 ResponseDTO 생성

필요한 기능

- 🔗 기업과 게스트의 상태 업데이트
- 🔗 지원하기 기능
- 🔗 지원정보 상세조회

- 🔗 기존 전체 코드
- 🔗 기존 뷰
- 🔗 테스트 모음

🔗 findApplyCompById , 🔗 findById , 🔗 findJopOpenById 를 유지하는 이유!

다음과 같은 경우에 원래의 네이티브 SQL 쿼리를 유지하는 것이 더 낫습니다:

1. 특정 컬럼 선택: 네이티브 SQL 쿼리는 특정 컬럼을 선택하여 반환하는 데 더 유연합니다. JPQL도 `SELECT new` 구문을 사용하여 비슷한 작업을 할 수 있지만, 모든 반환 필드에 대해 생성자 매개변수가 정확히 일치하는 DTO가 필요합니다.
2. 복잡한 조인 조건: 네이티브 SQL 쿼리는 복잡한 조인 조건과 서브쿼리를 처리하는 데 더 강력합니다. JPQL은 엔티티 간의 정의된 관계를 사용하여 조인을 수행하지만, 일부 복잡한 SQL 쿼리를 표현하는 데는 한계가 있습니다.
3. 성능 최적화: 특정 데이터베이스에 특화된 최적화를 적용할 수 있는 네이티브 SQL 쿼리를 통해, 때때로 더 나은 성능을 얻을 수 있습니다. JPQL은 데이터베이스에 독립적이기 때문에, 이런 최적화를 직접 적용하기 어렵습니다.

- 🔗 ApplyDTO
- 🔗 구글로그인
- 🔗 도로명주소

Apply restapi

- 🔗 restapi 전환

- 🔗 메모

리팩토링 - 컨트롤러 주소설계

```
// 이력서 관리 페이지
@GetMapping("/{quest/mngForm}")
public String mngForm(HttpServletRequest req) {
    User sessionUser = (User) session.getAttribute( s: "sessionUser");
    List<Resume> resumeList = guestRepository.findResumeById(sessionUser.getId());
    req.setAttribute( s: "resumeList", resumeList);
    return "quest/_myPage/mngForm";
}

@GetMapping("/{quest/profileForm}")
public String profileForm(HttpServletRequest req) {
    User sessionUser = (User) session.getAttribute( s: "sessionUser");
    User guestProfile = userService.guestInfo(sessionUser.getId());
    req.setAttribute( s: "guestProfile", guestProfile);
    return "quest/_myPage/profileForm";
}

// 마이페이지 - 공고 관리
@GetMapping("/{comp/mngForm}")
public String mngForm(HttpServletRequest req) {
    User sessionUser = (User) req.getSession().getAttribute( s: "sessionUser");
    List<JobopenResponse.DTO> jobopenList = compService.searchJobopenList(sessionUser.getId());
    req.setAttribute( s: "jobopenList", jobopenList);
    return "comp/_myPage/mngForm";
}

// 마이페이지 - 프로필관리
@GetMapping("/{comp/profileForm}")
public String profileForm(HttpServletRequest req) {
    User sessionUser = (User) session.getAttribute( s: "sessionUser");
    User compProfile = compService.getCompanyProfile(sessionUser.getId());
    req.setAttribute( s: "compProfile", compProfile);
    return "comp/_myPage/profileForm";
}

// 마이페이지 프로필 업데이트
@PostMapping("/{comp/updateProfile}") // 주소 수정 필요!
public String updateProfile(CompRequest.CompProfileUpdateDTO reqDTO) {
    User sessionUser = (User) session.getAttribute( s: "sessionUser");
    compService.compUpdateProfile(reqDTO, sessionUser);
    return "redirect:/comp/profileForm";
}
```

```
// 마이페이지 - 이력서, 공고 관리
@GetMapping("/{api/mngForm}")
public ResponseEntity<?> mngForm(@RequestParam(defaultValue = "0") int page) {
    SessionUser sessionUser = (SessionUser) session.getAttribute( s: "sessionUser");
    if (sessionUser.getRole() == 0) { // 개인 마이페이지
        ResumeResponse.MngDTO respDTO = guestService.guestResumesMng(page, sessionUser);
        return ResponseEntity.ok(new ApiUtil(respDTO));
    } else { // 기업 마이페이지
        List<JobopenResponse.MngDTO> respDTO = compService.compJobopenMng(sessionUser);
        return ResponseEntity.ok(new ApiUtil(respDTO));
    }
}

// 마이페이지 - 프로필
@GetMapping("/{api/profile}")
public ResponseEntity<?> profileForm() {
    SessionUser sessionUser = (SessionUser) session.getAttribute( s: "sessionUser");
    if (sessionUser.getRole() == 0) { // 개인 프로필
        UserResponse.GuestProfile respDTO = guestService.guestProgile(sessionUser);
        return ResponseEntity.ok(new ApiUtil(respDTO));
    } else { // 기업 프로필
        UserResponse.GuestProfile respSTO = compService.compProfile(sessionUser);
        return ResponseEntity.ok(new ApiUtil(respSTO));
    }
}
```

리팩토링 - JPA Repository & JPQL

```
private final EntityManager em;

@Transactional
public void statusUpdate(Integer applyId, String status) {

    Query query = em.createNativeQuery("UPDATE apply_tb SET state = ? WHERE");
    query.setParameter(1, status);
    query.setParameter(2, applyId);
    query.executeUpdate();
}
```

```
public List<ApplyResponse.ApplyDTO> findApplyCompByUserId(int compId, String sta

String q = ""

SELECT at.id, jot.jobopen_title, rt.resume_title, rt.name, rt.edu, jot.endTime, at.state, rt.id
FROM apply_tb at
INNER JOIN jobopen_tb jot ON at.jobopen_id = jot.id
INNER JOIN resume_tb rt ON rt.id = at.resume_id
WHERE jot.user_id = ? and at.role = 0 and at.state = ?;

Query query = em.createNativeQuery(q);
query.setParameter(1, compId);
query.setParameter(2, state);
```

```
public List<ApplyResponse.ApplyDTO2> findJobOpenByUserId(int userId, String state) { // 로그인한

String q = ""

SELECT at.id, jot.jobopen_title, rt.resume_title, jot.compname, rt.edu, jot.endTime
FROM apply_tb at
INNER JOIN jobopen_tb jot ON at.jobopen_id = jot.id
INNER JOIN resume_tb rt ON rt.id = at.resume_id
WHERE rt.user_id = ? and at.role = 1 and at.state = ?;

Query query = em.createNativeQuery(q);
query.setParameter(1, userId);
query.setParameter(2, state);

// qLrm 사용하기
JpaResultMapper mapper = new JpaResultMapper();
List<ApplyResponse.ApplyDTO2> responseDTO = mapper.list(query, ApplyResponse.ApplyDTO2.class);
return responseDTO;
}
```

```
public List<ApplyResponse.ApplyDTO> findByUserId(int userId) { // 로그인한 User ID

String q = ""

SELECT at.id, jot.jobopen_title, rt.resume_title, rt.name, rt.edu, jot.endTime, at.state, rt.id
FROM apply_tb at
INNER JOIN jobopen_tb jot ON at.jobopen_id = jot.id
INNER JOIN resume_tb rt ON rt.id = at.resume_id
WHERE at.user_id = ? order by id desc;

Query query = em.createNativeQuery(q);
query.setParameter(1, userId);

// qLrm 사용하기
JpaResultMapper mapper = new JpaResultMapper();
List<ApplyResponse.ApplyDTO> responseDTO = mapper.list(query, ApplyResponse.ApplyDTO.class);
return responseDTO;
}
```

```
public List<ApplyResponse.HireDTO> hfindAllByUserId(int compId) { // 로그인한 기업 ID

String q = ""

SELECT at.id, jot.jobopen_title, rt.resume_title, rt.name, at.state
FROM apply_tb at
INNER JOIN jobopen_tb jot ON at.jobopen_id = jot.id
INNER JOIN resume_tb rt ON rt.id = at.resume_id
WHERE at.user_id = ?;

Query query = em.createNativeQuery(q);
query.setParameter(1, compId);

// qLrm 사용하기
JpaResultMapper mapper = new JpaResultMapper();
List<ApplyResponse.HireDTO> responseDTO2 = mapper.list(query, ApplyResponse.HireDTO.class);
return responseDTO2;
}
```

```
public void findAll() {
}
```

```
import ...

public interface ApplyJpaRepository extends JpaRepository<Apply, Integer> {

    // 개인이 지원한 이력서 현황
    @Query("select new com.example.jobala.apply.ApplyResponse$GuestApplyDTO(a) from Apply a where a.user.id = :userId")
    List<ApplyResponse.GuestApplyDTO> findApplyGuestByUserId(@Param("userId") int userId);

    // 기업이 포지션 제안한 현황
    @Query("select new com.example.jobala.apply.ApplyResponse$CompPositionDTO(a) from Apply a where a.user.id = :userId")
    List<ApplyResponse.CompPositionDTO> findPositionCompByUserId(@Param("userId") int userId);

    // 기업이 보는 지원자 현황
    @Query("select new com.example.jobala.apply.ApplyResponse$CompApplyDTO(a) from Apply a where a.jobopen.user.id = :userId and a.role = 0")
    List<ApplyResponse.CompApplyDTO> findApplyCompByUserId(@Param("userId") int userId);

    // 개인이 보는 포지션 제안 현황
    @Query("select new com.example.jobala.apply.ApplyResponse$GuestPositionDTO(a) from Apply a where a.resume.user.id = :userId and a.role = 1")
    List<ApplyResponse.GuestPositionDTO> findPositionGuestByUserId(@Param("userId") int userId);

    // 지원한 공고의 수 세기
    @Query("SELECT COUNT(a) FROM Apply a WHERE a.jobopen.id = :jobopenId AND a.role = 0 AND a.state = '길트중'")
    int countJobopenApplyById(@Param("jobopenId") int jobopenId);
}
```

리팩토링 - DTO 그려보기

JSON

```
usercompname(쿠팡)  
userimgFilename(쿠팡사진)  
jobOpenTitle  
  
scrap  
jobopenid  
userid  
  
jobOpenId  
jobOpenCarrer  
jobOpenEdu  
jobOpenJobType  
jobOpenSalary  
jobOpenCompLocation  
  
jobOpenHopeJob  
인원은 직접 적음 뷰에서 이거 해결해야됨  
jobOpenSkills  
  
입사지원하기  
resumeid  
resumetitle  
resumecareer  
resumeedu  
  
접수 방법  
mustache에서 직접 적음
```

JSON

복사 캡션

```
{  
  "usercompname": "쿠팡",  
  "userimgfilename": "/image/파일명",  
  "scrap": [  
    {  
      "jobopenid": 1,  
      "userid": 1,  
    }  
  ],  
  "Title": "직무 제목",  
  "Id": 1,  
  "carrer": "경력 요구사항",  
  "edu": "학력 요구사항",  
  "JobType": "고용 형태",  
  "Salary": "연봉",  
  "CompLocation": "회사 위치",  
  "HopeJob": "희망 직무",  
  "Skills": ["필요 기술1", "필요 기술2", "필요 기술3"],  
  "resumeList2": [  
    {  
      "id": "이력서 ID",  
      "title": "이력서 제목",  
      "career": "이력서 경력",  
      "edu": "이력서 학력"  
    }  
  ],  
  "career": "경력",  
  "edu": "학력",  
  "hopeJob": "희망 직무",  
  "skills": ["java", "javascript", "spring", "html", "jquery", "Mysql"],  
  "license": ["자격증1", "자격증2", "자격증3"],  
  "content": "이력서 내용"  
}
```


리팩토링 - DTO

```
@AllArgsConstructor
@Data
public static class DetailDTO {
    private Integer id;
    private String jobopenTitle;
    private String career;
    private String edu;
    private String jobType;
    private String salary;
    private String compLocation;
    private String hopeJob;
    private String skills;
    private Date endTime; // 마감일
    private boolean isScrap;
    private boolean isGuestScrap;
    private UserDTO userDTO;
    private List<ResumeDTO> applyResumeList = new ArrayList<>();

    public DetailDTO(Jobopen jobopen, User sessionUser, List<Resume> resumeList) {
        this.id = jobopen.getId();
        this.jobopenTitle = jobopen.getJobopenTitle();
        this.career = jobopen.getCareer();
        this.edu = jobopen.getEdu();
        this.jobType = jobopen.getJobType();
        this.salary = jobopen.getSalary();
        this.compLocation = jobopen.getCompLocation();
        this.hopeJob = jobopen.getHopeJob();
        this.endTime = jobopen.getEndTime();
        this.skills = jobopen.getSkills();
        this.isScrap = false;
```

// 공고보기

```
public JobopenResponse.DetailDTO findJobopenById(Integer jobopenId, User sessionUser) {
    Jobopen jobopen = jobopenJpaRepository.findByJobopenIdWithUser(jobopenId)
        .orElseThrow(() -> new Exception404( msg: "공고를 찾을 수 없습니다"));
    List<String> skills = Arrays.stream(jobopen.getSkills().replaceAll( regex: "[\\[\\]\\"]", replacement: "").split( regex: ","))
        .toList();
    String skillsString = String.join( delimiter: ", ", skills);

    // isScrap
    if (sessionUser != null){
        // 모달 공고 목록 조회
        List<Resume> applyResumeList = resumeJpaRepository.findByUserId(sessionUser.getId());
        // 스크랩 했는지 안했는지 조회
        Optional<Scrap> scrap = scrapJpaRepository.findGuestScrapByJobopenIdAndUserId(jobopenId, sessionUser.getId());

        JobopenResponse.DetailDTO respDTO = new JobopenResponse.DetailDTO(jobopen, sessionUser, applyResumeList);
        respDTO.setScrap(scrap.isPresent());
        respDTO.setSkills(skillsString);
        return respDTO;
    }

    JobopenResponse.DetailDTO respDTO = new JobopenResponse.DetailDTO(jobopen, sessionUser);
    respDTO.setSkills(skillsString);
    return respDTO;
}
```

AOP(Aspect-Oriented Programming)

```
@NotEmpty(message = "유저네임이 공백일 수 없습니다")
private String username;

>Email(message = "올바른 이메일 형식이어야 합니다")
@NotEmpty(message = "email이 공백일 수 없습니다")
private String email;

@NotEmpty(message = "비밀번호가 공백일 수 없습니다")
@Size(min = 4, max = 20, message = "비밀번호는 최소 4자 이상 20자 이하여야 합니다")
private String password;

@NotEmpty(message = "이름이 공백일 수 없습니다")
@Size(min = 1, max = 20, message = "이름은 1자 이상 20자 이하여야 합니다")
private String name;

@AllArgsConstructor
@Data
@NotEmpty(message = "이름이 공백일 수 없습니다")
@Pattern(regexp = "^(신입/경력)$", message = "경력은 비어있을 수 없습니다")
private String career;

@Pattern(regexp = "^(고등학교 졸업/대학교 졸업)$", message = "학력은 비어있을 수 없습니다")
private String edu;

@Pattern(regexp = "^(백엔드/프론트엔드)$", message = "희망직종은 비어있을 수 없습니다")
private String hopeJob;
}
```

```
@Aspect // AOP 등록
@Component // IoC 등록
public class MyValidationHandler {

    // Advice (부가 로직 메서드)
    // Advice가 수행될 위치 = PointCut
    @Before("@annotation(org.springframework.web.bind.annotation.PostMapping) || @annotation(org.springframework.web.bind.annotation.PutMapping)") // PostMapping, PutMapping에 적용
    public void hello(JoinPoint jp) {
        Object[] args = jp.getArgs(); // Args: 파라미터 → object를 리턴
        System.out.println("크기 : " + args.length);

        for (Object arg : args) {
            if (arg instanceof Errors) {
                Errors errors = (Errors) arg; // 에러스타일의 arg를 다룬다
                if (errors.hasErrors()) {
                    for (FieldError error : errors.getFieldErrors()) {
                        System.out.println(error.getField());
                        System.out.println(error.getDefaultMessage());
                        throw new ApiException400(msg: error.getDefaultMessage() + " : " + error.getField());
                    }
                }
            }
        }
    }

    System.out.println("MyValidationHandler: hello_____");
}
```


JWT(JSON Web Token)

```
public class JwtUtil {
    public static String create(User user) {
        String jwt = JWT.create()
            .withSubject("blog")
            .withExpiresAt(new Date(System.currentTimeMillis() + 1000 * 60 * 60 * 24))
            .withClaim( name: "id", user.getId())
            .withClaim( name: "username", user.getUsername())
            .withClaim( name: "role", user.getRole())
            .sign(Algorithm.HMAC512( secret: "jobala"));
        return jwt;
    }

    public static SessionUser verify(String jwt) {
        System.out.println(jwt);

        DecodedJWT decodedJWT = JWT.require(Algorithm.HMAC512( secret: "jobala")).build().verify(jwt);
        System.out.println("decodedJWT = " + decodedJWT);
        int id = decodedJWT.getClaim( s: "id").asInt();
        System.out.println("id = " + id);
        String username = decodedJWT.getClaim( s: "username").asString();
        int role = decodedJWT.getClaim( s: "role").asInt();

        return SessionUser.builder()
            .id(id)
            .username(username)
            .role(role)
            .build();
    }
}
```

```
@Data
public class SessionUser {
    private Integer id;
    private String username;
    private String email;
    private Timestamp createdAt;
    private Integer role;

    @Builder
    public SessionUser(Integer id, String username, String email, Timestamp createdAt, Integer role) {
        this.id = id;
        this.username = username;
        this.email = email;
        this.createdAt = createdAt;
        this.role = role;
    }

    public SessionUser(User user) {
        this.id = user.getId();
        this.username = user.getUsername();
        this.email = user.getEmail();
        this.createdAt = user.getCreatedAt();
        this.role = user.getRole();
    }
}
```

JuitTest

```
// 공고 보기
@Test
void detailForm_jwt_test() throws Exception {
    // given
    User user = User.builder()
        .id(1)
        .username("cos1")
        .role(0)
        .build();


    // when
    String jwt = JwtUtil.create(user);

    int id = 1;

    ResultActions resultActions = mvc
        .perform(get( uriTemplate: "/api/jobopens/" + id).header( name: "Authorization", _values: "Bearer " + jwt));

    String responseBody = resultActions.andReturn().getResponse().getContentAsString();
    System.out.println("테스트 : " + responseBody);

    // then
    resultActions.andExpect(status().isOk())
        .andExpect(jsonPath( expression: "$.msg").value( expectedValue: "성공")) // 응답 메시지가 "성공" 인지 확인
        .andExpect(jsonPath( expression: "$.body").exists()) // 응답 본문이 존재하는지 확인
        .andExpect(jsonPath( expression: "$.body.id").value(id)) // 응답 본문에 공고의 ID가 있는지 및 값이 올바른지 확인
        .andExpect(jsonPath( expression: "$.body.jobopenTitle").exists()) // 공고의 제목이 응답 본문에 존재하는지 확인
        .andExpect(jsonPath( expression: "$.body.career").exists());
}
```

1초 500ms  테스트 통과: 1 / 1개 테스트 - 1초 500ms

```
> 테스트
```

```
FROM  
    scrap_tb s1_0  
left join  
    jobopen_tb j1_0  
        on j1_0.id=s1_0.jobopen_id  
left join  
    user_tb u1_0  
        on u1_0.id=s1_0.user_id  
where  
    j1_0.id=?  
    and u1_0.id=?
```

```
테스트 : {"status":200,"msg":"성공","body":{"id":1,"jobopenTitle":"?남?뵈?듕?삿?뽕 媛샹킷?엣 梨풀众","career":"?뽕?엣","edu":"怨준븃?빌媛?"}
```






```
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
```

```
2024-04-03T22:35:04.575+09:00 INFO 14504 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory  
2024-04-03T22:35:04.579+09:00 INFO 14504 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown  
2024-04-03T22:35:04.584+09:00 INFO 14504 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown
```

```
> Task :test
```

API 문서(바로가기)

+ User

- +   [메인\(HOME\)](#)
-  [회원가입](#)
-  [로그인](#)
-  [로그아웃](#)
-  [채용공고 목록](#)
-  [채용공고 검색필터](#)
-  [개인 -마이페이지-이력서 관리](#)
-  [기업 - 마이페이지- 공고 관리](#)
-  [개인 - 마이페이지- 프로필](#)
-  [기업 - 마이페이지 - 프로필](#)
-  [개인 - 마이페이지- 프로필 수정](#)
-  [개인 - 마이페이지- 프로필 수정](#)


Comp

-  [기업 - 인재 이력서 검색](#)
-   [기업 - 인재 명단 목록](#)









Jobopen

-  [공고 삭제](#)
-  [공고 수정](#)
-  [공고 등록](#)
-  [공고 보기](#)







Reply

-  [댓글 쓰기](#)
-  [댓글 삭제](#)

Apply

-  [기업- 내 공고 지원 현황](#)
-  [기업- 포지션 제안 현황](#)
-  [기업- 포지션 제안](#)
-  [기업, 지원 상태 업데이트\(지원 결과 합격 불합격\)](#)
-  [개인 - 포지션 제안 받은 현황](#)
-  [개인- 공고 지원 현황](#)
-  [개인- 공고 지원하기](#)
-  [개인 - 지원 상태 업데이트\(포지션 제안 수락, 거절\)](#)

board

-   [글 상세보기](#)
-  [글 목록 보기](#)
-  [글쓰기](#)
-  [글 수정](#)
-  [글 삭제](#)

Resume

-   [이력서 수정](#)
-  [이력서 상세보기](#)
-  [이력서 등록](#)
-  [이력서 삭제](#)

Scrap

-  [스크랩 목록 보기](#)
-  [스크랩](#)

API 문서(바로가기)

```
Java ▾
요청 주소 (GET)
- http://localhost:8080/api/mngForm

요청 헤더
Authorization : Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJjb3PthqDtgbAiLCJpZC...
```

응답 바디

```
- application/json

{
  "status": 200,
  "msg": "성공",
  "body": {
    "userId": 1,
    "resume": [
      {
        "id": 10,
        "resumeTitle": "열정적인 프론트엔드 엔지니어 자기소개서"
      },
      {
        "id": 1,
        "resumeTitle": "도전적인 프론트엔드 개발자 지원서"
      }
    ]
  }
}
```

```
Java ▾
요청 주소 (POST)
- http://localhost:8080/join?role=0

요청 바디
- application/json
{
  "name": "설동훈",
  "username": "cos14",
  "email": "yeeun0118s@nate.com",
  "password": "1234",
  "address": "부산광역시 금정구",
  "age": "2000-04-25",
  "phone": "010-1234-5678",
  "role": 0,
  "created_at": "현재 날짜 및 시간"
}
```

요청 헤더

```
Authorization : Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJjb3PthqDtgbAiLCJpZC...
```

응답 바디

```
- application/json

{
  "status": 200,
  "msg": "성공",
  "body": {
    "userId": 22,
    "username": "cos14",
    "name": "설동훈",
    "email": "yeeun0118s@nate.com",
    "password": "1234",
    "address": "부산광역시 금정구",
    "age": "2000-04-25",
    "phone": "010-1234-5678",
    "role": 0,
    "created_at": "현재 날짜 및 시간"
  }
}
```

후기

민경

처음에는 핵심로직만 구현하면 된다고 생각하다가 점점 욕심이 생겨서 뷰도 고치고싶고 만나이 설정이나 도로명 주소, 간편 로그인도 구현해보고 싶은 욕심이 생기고 더 많이 하고 싶었지만 정해진 기간이 있어 못한게 너무 아쉬웠고 좋은 팀원들을 만나서 같이 소통하고 서로 도우면서 많이 발전할 수 있었던 것 같아 너무 좋았습니다.

선규

이번에 2조 팀을 하면서 부족한부분이 많았는데 한가지 기능만 맡아서 하는것이 아닌 여러 기능들을 구현하고 검색 필터 페이징을 하면서 구체적으로 조인을 하고 막히는 부분들은 검색을 통해 다른 사람 코드도 보면서 실력이 늘은부분이많은거 같다. 파이널 전까지 기초를 탄탄히 준비하여 일처리하는 시간을 줄여야겠다

후기

찬혁

저희가 직접 짠 코드를 다시 리팩토링하는 과정이 매우 복잡하였습니다. 이를통해 초반 설계가 중요하다는 것을 깨달았으며, 설계부분을 수정하고 보완하면서 한단계 성장한 것 같습니다.

유진

훌륭한 팀원들과 함께 저번 프로젝트보다 더욱 완성도 있는 결과물을 만들어 내서 좋았고 앞으로도 이번 프로젝트에서의 경험을 살려 더 많은 도전을 하고 싶습니다

주혁

팀원들과 협업하며 프로젝트를 하면서 같이 만들어가는 코딩문화를 더욱 많이 겪어봐야 겠다고 생각합니다. 혼자서는 안되는 일도 같이 생각하게 되면 더욱 쉽고 빠르게 나아갈 수 있다는 걸 이번 프로젝트로 인해서 느꼈습니다. 코딩을 잘하는사람, 적극적으로 프로젝트를 이끌어주는 사람, 포기하지 않고 끝까지 해보는 사람, 새로운 로직을 적용시켜 퀄리티를 높여주는 사람 다양한 사람들이 모여서 만들어지는 프로젝트는 코딩이 다가 아닌 협력, 해주었습니다.



감사합니다

Thank you