초급 프로젝트

HR Bank

Batch로 데이터를 관리하는 Open EMS

2조 (Part 2)

Codeit Sprint – Spring Backend 7기

목차

목차

1. 프로젝트 개요

1. 프로젝트 개요

프로젝트 내용

직원·부서 관리

- 직원·부서 정보 CRUD 관리
- 중복 검증
  - 부서 이름
  - 직원 이메일
- 직원 프로필 이미지 관리

데이터 백업 및 로그 관리

- 직원 데이터 백업
  - 변경 감지 기반 수행
- 1시간 주기 자동 백업
- 직원 수정 이력 로그 관리

조건 조회 시스템

- 정렬 및 범위/부분 일치 조회
- 다중 조건 조회
- 효율적인 데이터 조회
  - 커서 페이지네이션 기반

대시보드 기반 인사통계

- 직원 수 현황
- 최근 기간별 변화 추이
- 부서·직무별 직원 분포

1. 프로젝트 개요

개발 환경 및 사용 스택

개발 환경

- OS : Window 11 / mac OS
- IDE : Intellij IDEA
- JDK : Amazon Corretto 17.0.16
- 빌드 도구 : Gradle
- 버전 관리 : Git / GitHub

사용 스택

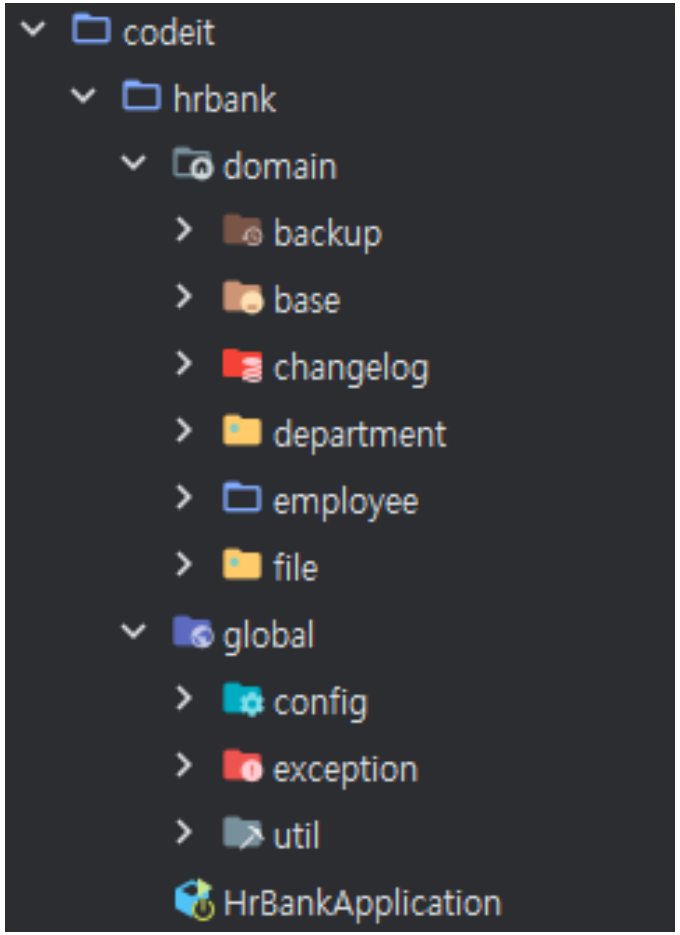
- Spring Boot Framework
- Spring Data JPA
- Spring Validation
- Lombok
- springdoc-openapi
- PostgreSQL
- Railway.io
- OpenCSV

1. 프로젝트 개요

프로젝트 구조

프로젝트 구조

- 도메인 중심 패키지 구조
- Global 공통 모듈 분리



HR bank 프로젝트 구조

[2. 프로젝트 팀 구성 및 역할, 협업 방식 소개]

2. 프로젝트 팀 구성 및 역할, 협업 방식 소개

팀원 소개

김태언 | 팀장

https://github.com/Taeeon-kim

안대식

https://github.com/ian-i-an

황준영

https://github.com/OfficialHwempire

최태훈

https://github.com/Tae705

조성만

https://github.com/BetterCodings

최지혜

https://github.com/ChoiJiHye950

2. 프로젝트 팀 구성 및 역할, 협업 방식 소개

팀원 R&R

부서 – 김태언, 최지혜

- 부서 관리
- 검색 기능
- 커서 페이지네이션

회원 수정 로그 – 안대식

- 로그 관리
- 검색 기능
- 커서 페이지네이션

직원 – 조성만, 최태훈

- 직원 관리
- 검색 기능
- 커서 페이지네이션

백업 및 파일 – 황준영

- 파일과 메타 정보 관리
- CSV 다운
- 수동/자동 백업

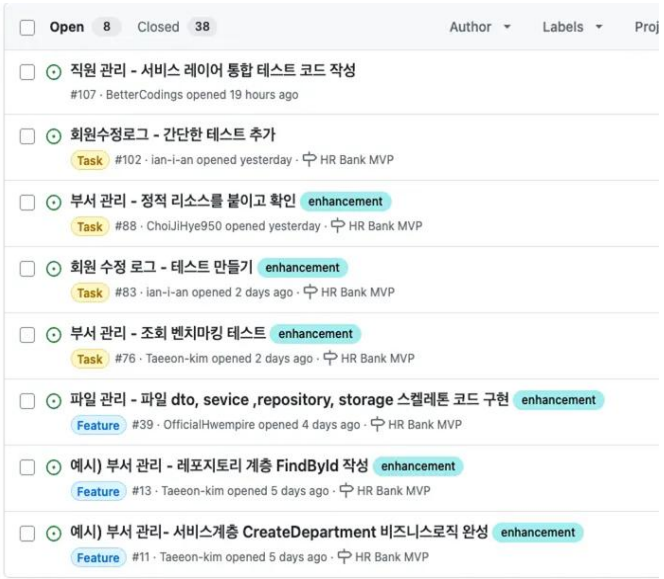
2. 프로젝트 팀 구성 및 역할, 협업 방식 소개

협업 방식

이슈 트래킹

- 사용 툴 : 노션, GitHub Issues

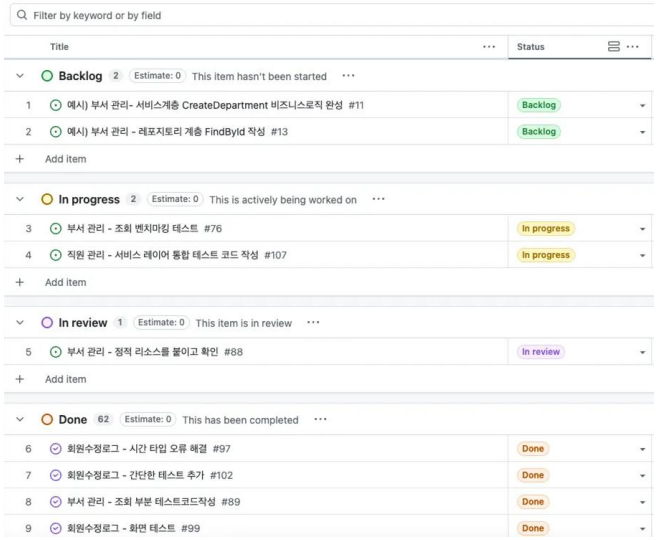
노션 스크럼 회의록



GitHub Issues

일정 관리

- 사용 툴 : GitHub Projects 칸반보드



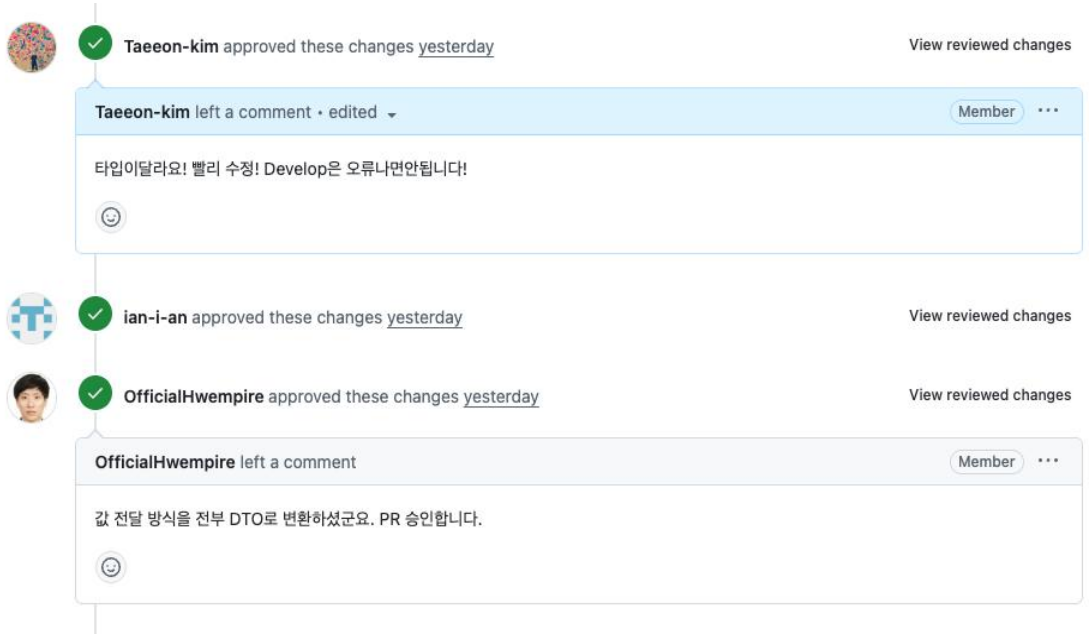
GitHub Projects 칸반보드

2. 프로젝트 팀 구성 및 역할, 협업 방식 소개

협업 방식

코드 리뷰

- 사용 툴 : GitHub PR(Pull Request)



GitHub PR

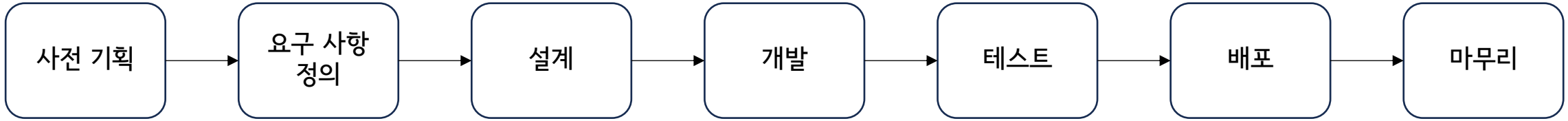
개발 컨벤션

- Java
  - Full Spelling 사용, 축약 금지
  - 변수명 : 카멜 케이스
  - 상수 : 대문자 + 스네이크 케이스
- GitHub
  - 브랜치
    - 브랜치 : main(배포용), develop(개발용)
    - 기능 개발 브랜치 : feature/도메인-기능-담당자이름
  - 커밋 메시지
    - type(타입) : title(제목), Body(본문)는 선택
- DB & API
  - 테이블/엔드포인트 이름 : 복수형 사용

3. 프로젝트 수행 절차 및 방법

3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차



3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

사전 기획 → 요구 사항 정의 → 설계 → 개발 → 테스트 → 배포 → 마무리

1. 사전 기획

- 주제 선정 : HR Bank
- 팀 구성 및 역할
  - 도메인별 팀 구성
  - 각 담당 역할(R&R) 확정
- 협업 규칙 및 개발 컨벤션

4. 팀 에티켓 규칙
   1. 부재시 미리 팀에게 알리기!
   2. 누군가의 도움이 필요할때 서로 짜증내지않기
   3. 일정시간 이후에는 디스코드 2팀 채널방에 소통
   4. 팀프로젝트 관련 질문관련/피드백은 노션 Q&A 남기기 - 같은 문제 발생에 대한 질문이나 답변등 참고차
   5. 맡은 업무에 대한 지연에 대해서 만약 일정안에 못할거같을시 다른분들과 공유하고 분담할것
   6. 자신이 맡은 업무가 끝나면 다음 남은 업무 티케팅 공유후 작업 진행
   7. PR 요청자가 conflict시 스스로 해결하고 다시PR요청한다 만약 스스로 해결이안되거나 누구의 코드가 반영되야되는지 모르면 해당 코드의 팀원분 혹은 팀원 전체에게 알려서 수정후 다시 PR한다.
   8. PR 머지후 지속적으로 Pull 땡기기(develop에서 주기적인 업데이트가 이러나기때문에 conflict 을 최소화하기위해 미리 검사) → 추가내용 공유후 색다시 돌려놓기

7. PR 코드리뷰 방식
   PR시 서로에게 알리고 공유한다
   PR시 reviewers 최소인원 : 2명(모든팀원 reviews 태그, 자동설정 혹은 PR요청자가 달기)
   코드스타일이나 리펙토링 관점의 코드리뷰보단 유지보수/서비스 코드 이해를 위한 리뷰 관점으로 코드리뷰한다.
   PR 코드에대해서 궁금증이나 의견을 나누고싶을시 해당 PR에 comment남긴다.(의문이 풀릴때까지 merge없다)
   정규시간외에 PR요청시 디스코드에 PR 리뷰 요청

협업 규칙

10. 아키텍쳐 및 레이어드 컨벤션

    레이어드:
    서비스 레이어드는 순환 의존성 참조를 방지하기 위해 사용하지 말아야 하지만 중복되는 코드 발생했을 때 순환 의존성이 아닐 때는 사용 가능

14. 코드 컨벤션/커밋 컨벤션 :
    - 변수명, 클래스명, 폴더 명등 축약형을 쓰지말고 full spelling 사용 할것
      Ex) usernameForEmail (0), usrNameEm(x),
    - 변수명 : 카멜케이스 사용, `(_, -) 안됨, 파스칼 안됨` lastName, firstName
    - 상수 : 모든 대문자 및 _ 사용

    일관된 커밋 메세지
    https://sungwookoo.tistory.com/1

개발 컨벤션

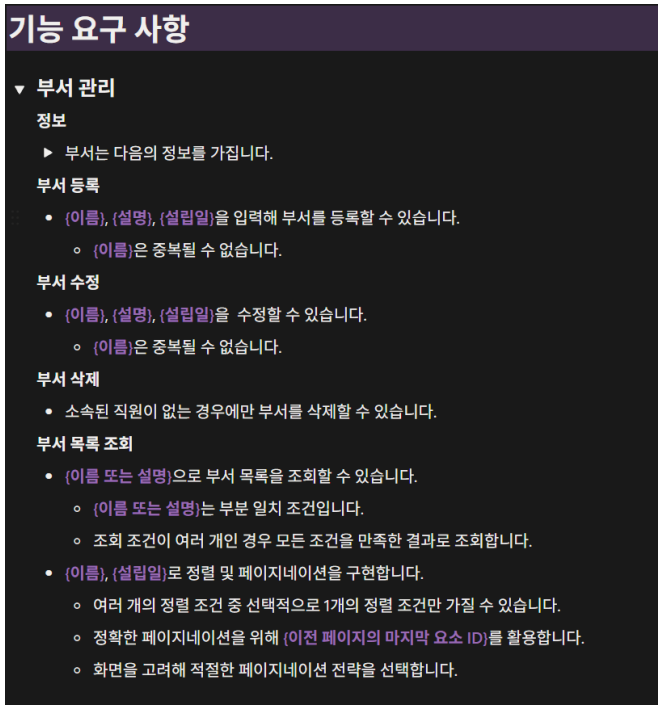
3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

사전 기획 → **요구 사항 정의** → 설계 → 개발 → 테스트 → 배포 → 마무리

2. 요구 사항 정의

- 사용자 요구사항 분석
- API 요구사항 명세 작성



기능 요구 사항

- **부서 관리**
  - **정보**
    - 부서는 다음의 정보를 가집니다.
  - **부서 등록**
    - {이름}, {설명}, {설립일}을 입력해 부서를 등록할 수 있습니다.
      - {이름}은 중복될 수 없습니다.
  - **부서 수정**
    - {이름}, {설명}, {설립일}을 수정할 수 있습니다.
      - {이름}은 중복될 수 없습니다.
  - **부서 삭제**
    - 소속된 직원이 없는 경우에만 부서를 삭제할 수 있습니다.
  - **부서 목록 조회**
    - {이름 또는 설명}으로 부서 목록을 조회할 수 있습니다.
      - {이름 또는 설명}는 부분 일치 조건입니다.
      - 조회 조건이 여러 개인 경우 모든 조건을 만족한 결과로 조회합니다.
    - {이름}, {설립일}로 정렬 및 페이지네이션을 구현합니다.
      - 여러 개의 정렬 조건 중 선택적으로 1개의 정렬 조건만 가질 수 있습니다.
      - 정확한 페이지네이션을 위해 {이전 페이지의 마지막 요소 ID}를 활용합니다.
      - 화면을 고려해 적절한 페이지네이션 전략을 선택합니다.

사용자 요구사항

3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

2. 요구 사항 정의

- 사용자 요구사항 분석
- API 요구사항 명세 작성

index	기능	HTTP m...	API Path	request	response
Department	부서 목록 조회	GET	/api/departments	JSON : CursorPageRequestDepartmentDto	200(ok) : CursorPageResponseDepartmentDto
	부서 등록	POST	/api/departments	JSON : DepartmentCreateRequest	200(ok) : DepartmentDto
	부서 상세 조회	GET	/api/departments/{id}	Path: id	200(ok) : DepartmentDto
	부서 수정	PATCH	/api/departments/{id}	JSON: DepartmentUpdateRequest	200(ok) : DepartmentDto
	부서 삭제	DELETE	/api/departments/{id}	Path: id	204(no content)
Employees	직원 목록 조회	GET	/api/employees	JSON : CursorPageRequestEmployeeDto	200(ok) : EmployeeDto[ ]
	직원 등록	POST	/api/employees	Form-Data: employee(JSON), profile(file)	201(created) : EmployeeDto
	직원 상세 조회	GET	/api/employees/{id}	Path: id	200(ok) : EmployeeDto
	직원 수정	PATCH	/api/employees/{id}	Form-Data: employee(JSON), profile(file)	200(ok) : EmployeeDto
	직원 삭제	DELETE	/api/employees/{id}	Path: id	204(no content)
	직원 증감 추이	GET	/api/employees/stats/trend	JSON : EmployeeTrendRequest	200(ok) : EmployeeTrendDto
	직원 분포 조회	GET	/api/employees/stats/distribution	RequestParam : groupBy, status	200(ok) : EmployeeDistributionDto
	직원 수 조회	GET	/api/employees/count	RequestParam : status, fromDate, toDate	200(ok) : number

API 명세

(Department, Employees)

3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

2. 요구 사항 정의

- 사용자 요구사항 분석
- API 요구사항 명세 작성

Change-logs	직원 수정 이력 목록 조회	GET	/api/change-logs	JSON : ChangeLogFilter	200(ok) : CursorPageResponseChangeLogDto
	직원 수정 이력 상세 조회	GET	/api/change-logs/{id}/diffs	Path: id	200(ok) : DiffDto[ ]
	수정 이력 건수 조회	GET	/api/change-logs/count	RequestParam : fromDate, toDate	200(ok) : number
Backups	백업 목록 조회	GET	/api/backups	RequestParam : worker, status, startedAtFrom, startedAtTo, sortDirection, sortField, size, cursor, idAfter	200(ok) : CursorPageResponseBackupDto
	백업 생성	POST	/api/backups		200(ok) : BackupDto
	최근 백업 정보 조회	GET	/api/backups/latest		200(ok) : BackupDto
Files	파일 다운로드	GET	/api/files/{id}/download	Path:id	200(ok) : Binary(파일 데이터)

API 명세

(Change-logs, Backups, Files)

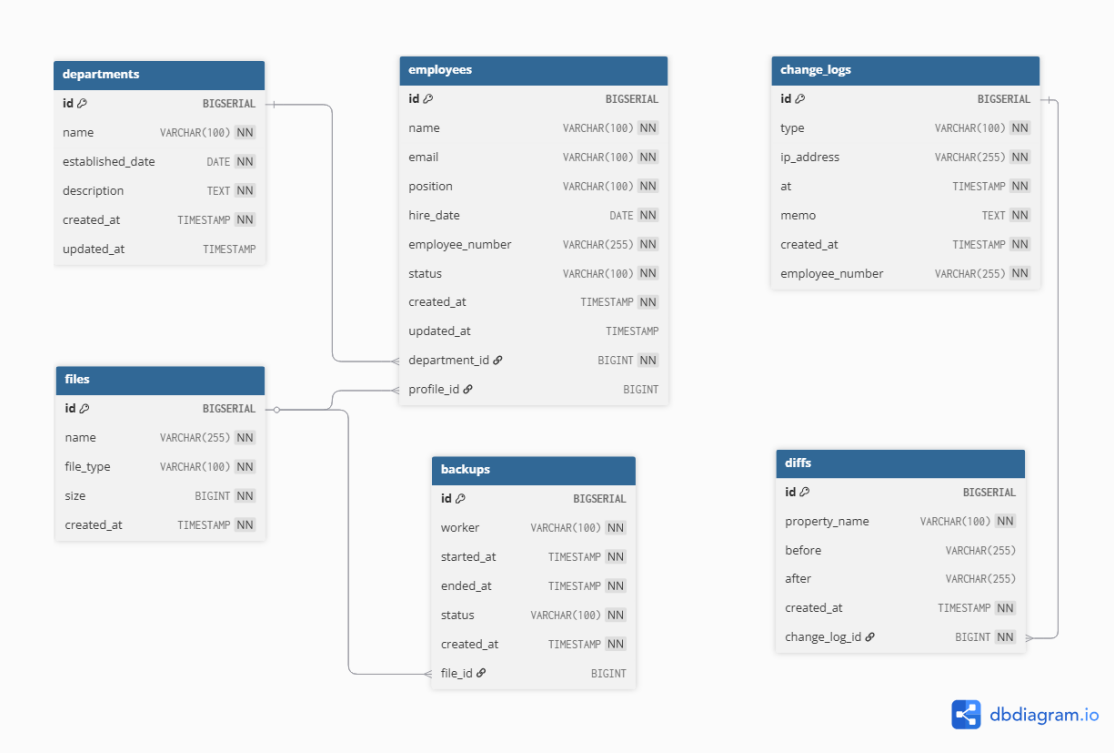
3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

사전 기획 → 요구 사항 정의 → **설계** → 개발 → 테스트 → 배포 → 마무리

3. 설계

- ERD 및 DDL 설계
- 프로젝트 구조와 브랜치 컨벤션 확정



ERD - diagram

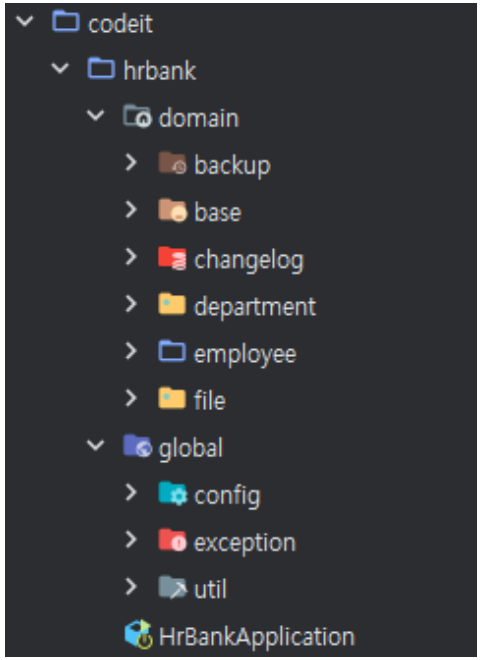
3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

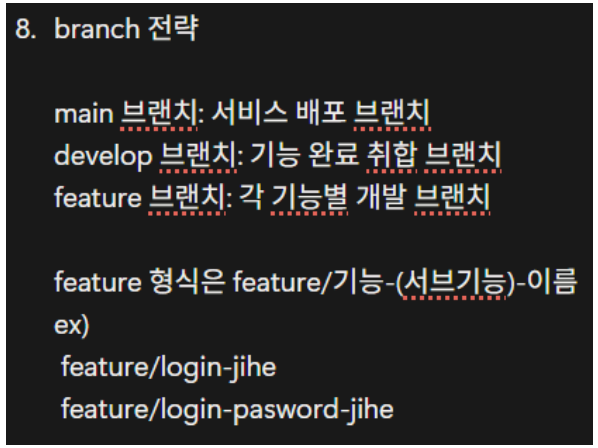
사전 기획 → 요구 사항 정의 → **설계** → 개발 → 테스트 → 배포 → 마무리

3. 설계

- ERD 및 DDL 설계
- 프로젝트 구조와 브랜치 컨벤션 확정



프로젝트 구조
(도메인 중심)



브랜치 컨벤션

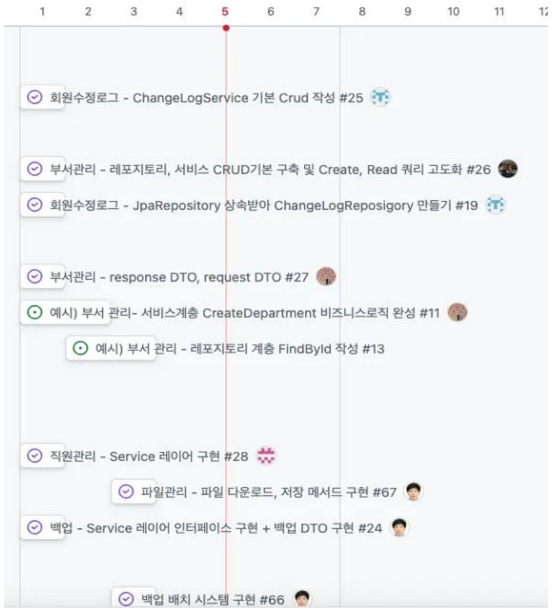
3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

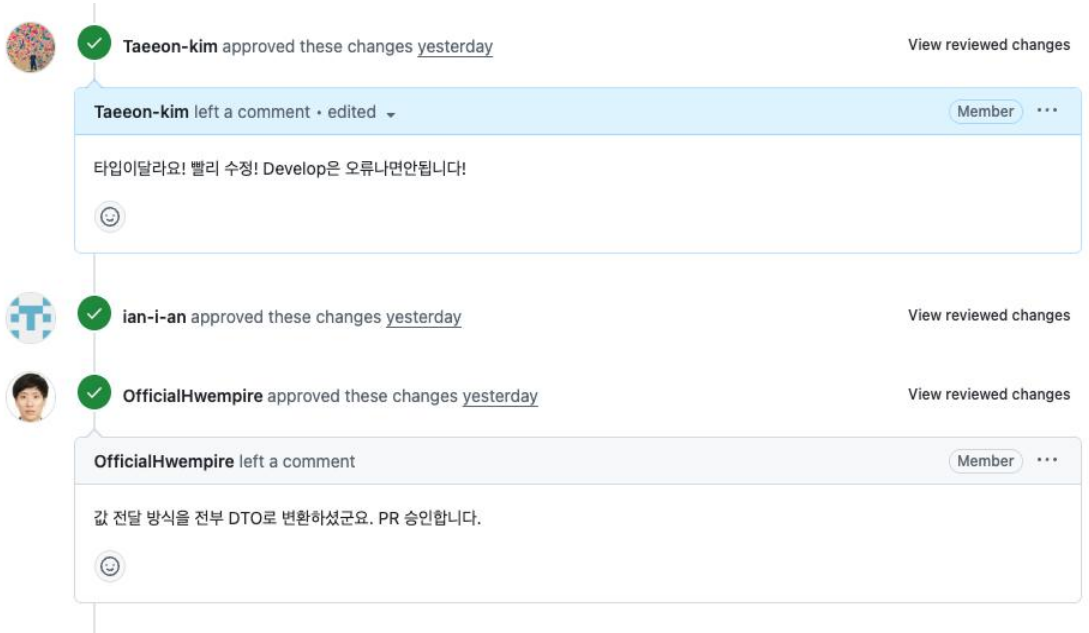
사전 기획 → 요구 사항 정의 → 설계 → **개발** → 테스트 → 배포 → 마무리

4. 개발

- 기능 구현
  - 도메인별 R&R 기반
- 일정 관리
  - GitHub 칸반보드 기반
- 코드 리뷰
  - GitHub PR 기반



GitHub Projects 칸반보드



GitHub PR

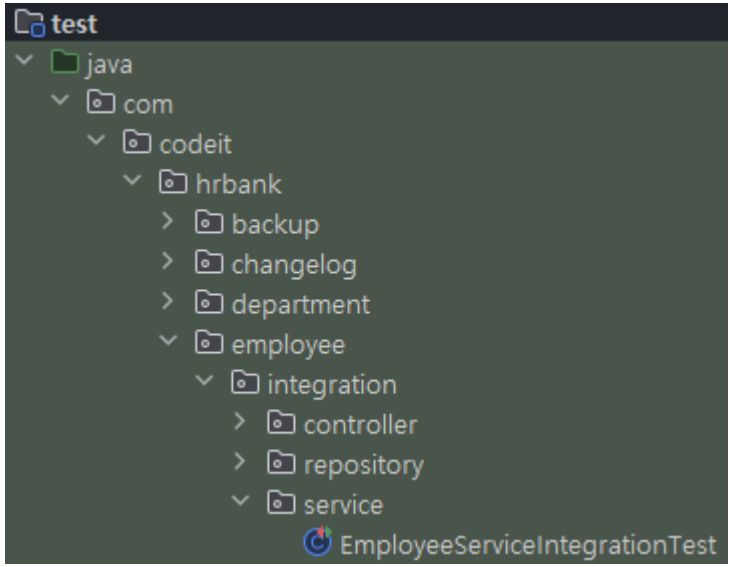
3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

사전 기획 → 요구 사항 정의 → 설계 → 개발 → **테스트** → 배포 → 마무리

5. 테스트

- 기능 구현 테스트
  - 통합 테스트
- 버그/성능 개선



도메인 별 테스트

```java
@DisplayName("직원 등록 테스트") seongman Joe +1
@Nested
class CreateEmployee {
    // 성공 케이스
    @Test seongman Joe +1
    @DisplayName("직원 등록 - 성공")
    void createEmployee() throws IOException {
        // given
        EmployeeCreateRequest request =
                new EmployeeCreateRequest(
                        name: "김춘식",
                        email: "chunsik@kakao.com",
                        departmentId: 5L,
                        position: "신입",
                        LocalDate.now(),
                        memo: "직원 생성"
                );

        // when
        EmployeeDto employee = employeeService.createEmployee(request, mockFile(), clientIp: "127.0.0.1");

        // then
        assertEquals(request.name(), employee.name());
        assertEquals(request.email(), employee.email());
        assertEquals(request.departmentId(), employee.departmentId());
        assertEquals(request.position(), employee.position());
        assertEquals(request.hireDate(), employee.hireDate());
    }
```

직원 등록 테스트 코드

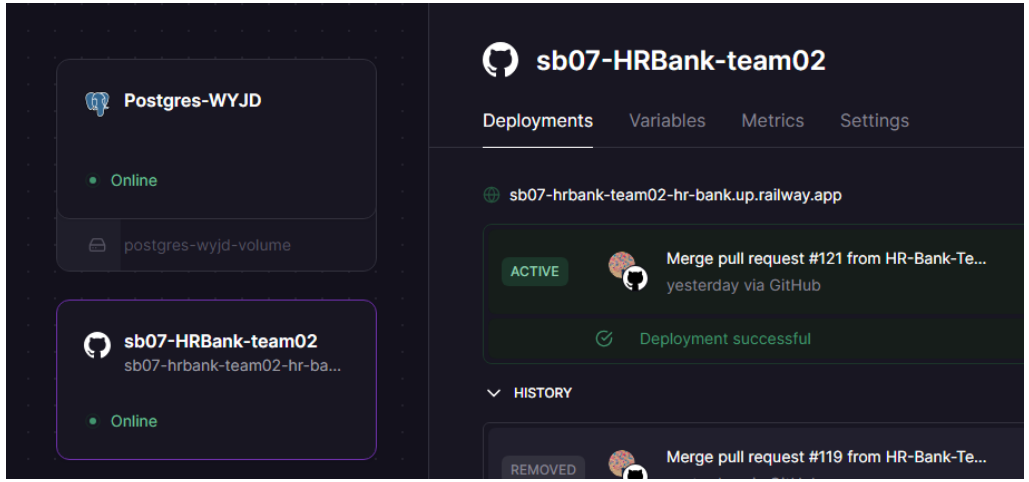
3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

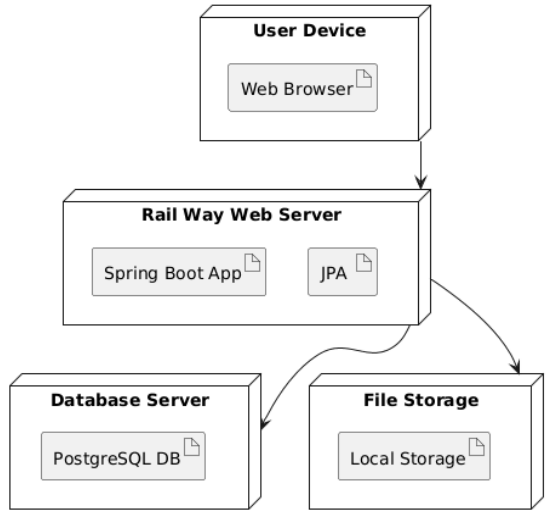
사전 기획 → 요구 사항 정의 → 설계 → 개발 → 테스트 → **배포** → 마무리

6. 배포

- 프론트 연동
- 서버 배포
  - RailWay 서비스
- 배포 환경 검증



RailWay.io 배포



배포 다이어그램

3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

7. 마무리

- 주요 트러블 슈팅 기록
- 프로젝트 산출물 정리
- 발표 준비

3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

사전 기획 → 요구 사항 정의 → 설계 → 개발 → 테스트 → 배포 → 마무리

7. 마무리 – 트러블 슈팅(부서)

문제

부서 별
직원 수 조회 시
속도 저하

원인

LAZY 조회 방식으로
부서마다
COUNT(*)를 호출

N+1 문제 발생

해결

JOIN + GROUP BY
쿼리로 한 번에 조회

N+1 문제 해결

배운점

1. 쿼리 튜닝이
   성능에 큰 차이를
   만든다는 것을
   깨달음
2. DB 튜닝뿐만 아니라
   조회 방식도 중요한
   요소임을 배움

3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

사전 기획 → 요구 사항 정의 → 설계 → 개발 → 테스트 → 배포 → 마무리

7. 마무리 – 트러블 슈팅(부서)

문제	원인	해결	배운점
코드 리팩토링 시 조회 기능 (검색·정렬·페이지네이션) 오동작	Repository 단위 테스트 및 통합 테스트 부재	테스트 코드 작성/적용하여 오동작 방지 자동화	테스트 코드 작성이 기능 리팩토링 시 중요함을 깨달음

3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

사전 기획 → 요구 사항 정의 → 설계 → 개발 → 테스트 → 배포 → 마무리

7. 마무리 – 트러블 슈팅(직원)

문제	원인	해결	배운점
프론트에서 전달하는 시간 정보를 수신하지 못하는 문제 발생	시간 타입의 파라미터 타입 불일치	파라미터를 프론트 엔드가 요구하는 타입으로 변경	시간 타입도 여러 타입이 존재하여 요청 형식에 맞는 타입을 결정해야 함

3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

사전 기획 → 요구 사항 정의 → 설계 → 개발 → 테스트 → 배포 → 마무리

7. 마무리 – 트러블 슈팅(직원 & 백업)

문제	원인	해결	배운점
배포 환경에서 실제 IP가 아닌 프록시 IP를 저장	IP를 가져오는 getRemoteAddr()은 프록시를 거치면 프록시 IP를 저장	HTTP Servlet 요청의 X-Forwarded-For 헤더에서 실제 IP를 가져오도록 로직을 변경	동일한 코드가 로컬 환경에서는 잘 동작하더라도 배포 환경에서는 다를 수 있음을 배움

3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

사전 기획 → 요구 사항 정의 → 설계 → 개발 → 테스트 → 배포 → 마무리

7. 마무리 – 트러블 슈팅(수정 이력 로그)

문제	원인	해결	배운점
JPQL만으로는 효율적인 조회 쿼리 작성이 불가능	다양한 조회 조건으로 인한 복잡해지는 쿼리	커스텀 레포지토리를 활용하여 수동 동적 쿼리를 자바코드로 필터링	동적 쿼리 작성시 JPQL의 한계를 느끼고 QueryDSL을 활용하여 리팩토링이 필요함을 느낌

3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

7. 마무리 – 트러블 슈팅(백업)

문제	원인	해결	배운점
API 명세의 요구사항에 맞추어 Backup 파일 생성 불가능	Backup의 상태 중 “진행중”은 트랜잭션을 활용한 로직 상 구현에 부적합	다른 팀원과 강사님과 검토 결과 구현에 부적합하다고 판단하여 미구현 후 결과 공유	요구사항이 현재 서비스에 부적합한 경우 팀원과 의사소통을 통해 해결해야 함을 배움

3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

사전 기획 → 요구 사항 정의 → 설계 → 개발 → 테스트 → 배포 → 마무리

7. 마무리 – 트러블 슈팅(파일)

문제	원인	해결	배운점
백업 파일 작성 시 느린 로컬 디스크 저장 속도	저장 시 전체 직원 객체를 하나씩 write()를 진행	BufferedWriter를 활용하여 파일 작성 속도를 기존 대비 10배 향상	다수 객체의 파일 저장 시 버퍼를 활용하는 것이 중요함을 깨달음

3. 프로젝트 수행 절차 및 방법

프로젝트 수행 절차

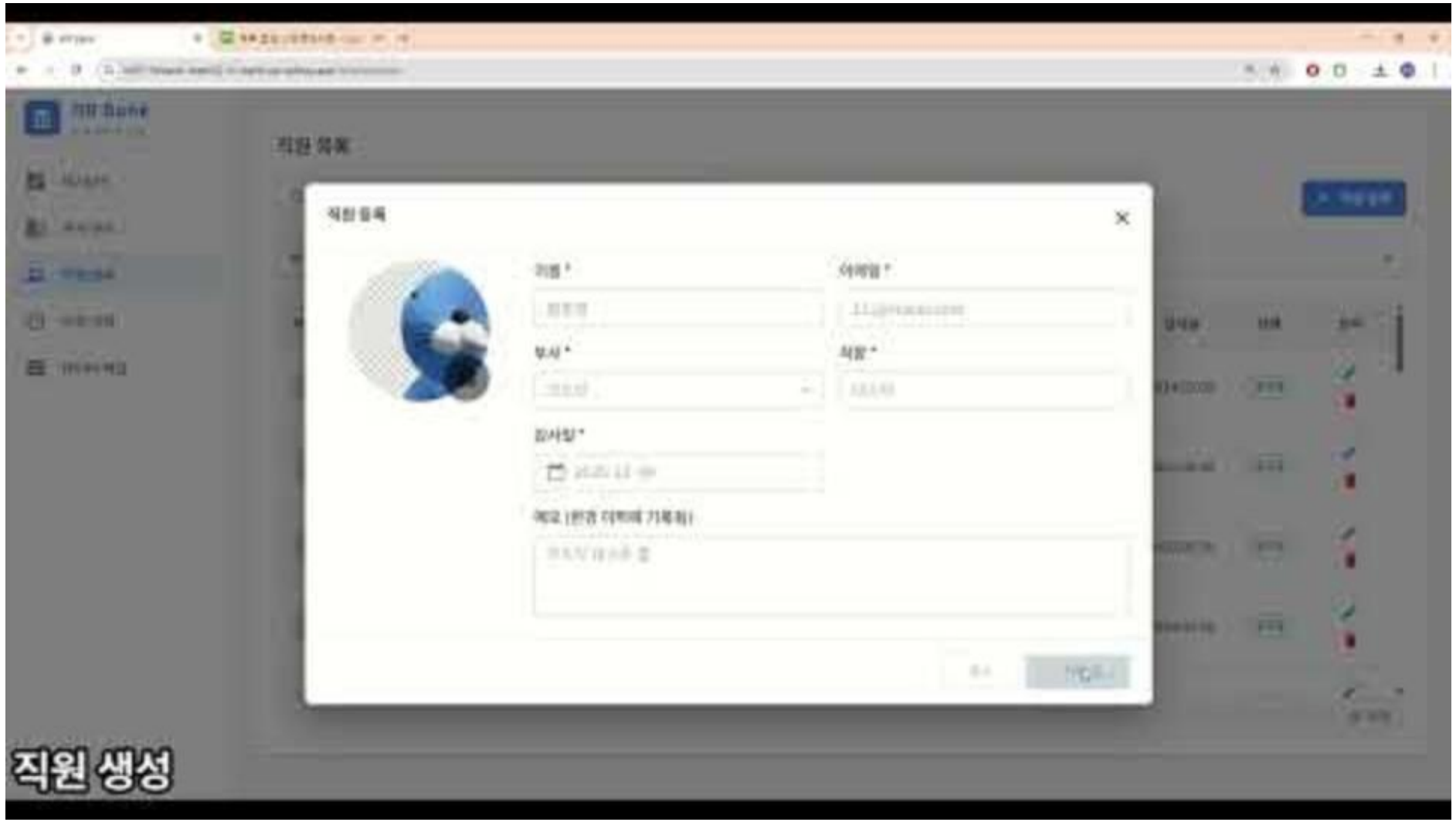
7. 마무리

- 주요 트러블 슈팅 기록
- 프로젝트 산출물 정리
- 발표 준비

4. 프로젝트 수행 결과

4. 프로젝트 수행 결과

시연 영상



4. 프로젝트 수행 결과

서비스 배포 링크

서비스 배포 링크

https://sb07-hrbank-team02-hr-bank.up.railway.app

서비스 배포 API Swagger

https://sb07-hrbank-team02-hr-bank.up.railway.app/swagger-ui/index.html

5. 팀 자체 평가 의견

5. 팀 자체 평가 의견

아쉬운 점

1. JPQL 기반 동적 쿼리 구현의 어려움
    - QueryDSL을 효과적으로 활용하지 못한 점이 아쉬웠다.

2. ERD 작성 실수로 인한 전체 애플리케이션 수정
    - ERD 작성 시 부서 설립일과 직원 입사일 속성의 타입을 다르게 작성
    - 이후 개발 시 DDL을 수정하게 되어, 전체 애플리케이션을 함께 수정

3. 유지 보수성 및 확장성 고려 부족 (기한 맞추기 우선)

4. 도메인 간 이해 부족
    - 맡은 도메인 구현에 집중하면서 다른 도메인에 대한 이해도가 부족했다.

5. 팀 자체 평가 의견

잘한 점

1. 사전 기획 및 요구 사항 분석을 철저하게 진행하여 일관성과 완성도를 높임
2. 원활한 팀 내 의사소통으로 병목 없이 작업 진행
3. Git 브랜치 컨벤션을 통해 충돌 최소화, 합병 과정 원활
4. 코드 리뷰를 통해 코드 품질 및 도메인 이해도 향상
5. 일정 지연 없이 주요 기능을 통합하여 계획대로 마무리

5. 팀 자체 평가 의견

느낀 점

1. 소통의 가치
   - 다양한 작업 방식과 의견을 조율하는 과정에서 서로를 이해하고 협력하는 능력을 키웠다
   - 초반에 공통 작업을 소통하며 함께 진행한 덕분에 작업 충돌 없이 원활하게 진행했다고 생각한다.
2. 도메인 간 협력과 상호 이해
   - 각자 맡은 도메인에 책임감을 가지고 진행하면서, 다른 도메인과의 의존성으로 인해 맡지 않은 도메인에 대해서도 이해해야 한다는 점을 깊이 깨달았다.
3. 프로젝트 초기 단계의 중요성
   - 초기 기획, 요구사항 분석, 설계 단계를 잘 진행하는 것이 프로젝트의 품질과 진행에 큰 영향을 주는 것을 깨달았다.
4. 테스트와 트러블 슈팅
   - 성능, 유지보수, 운영환경까지 고려한 테스트와 트러블슈팅이 중요한 요소임을 체감하였다.
5. 문서화의 중요성
   - 팀 지식을 문서화하며 " 나만 아는 지식 " 이 아닌 팀의 공동 지식으로 쌓아가는 경험을 하였다.
   - 이는 후속 작업이나 다른 팀원에게 큰 도움이 될 것이라 생각한다.

QnA

Appendix

참고자료

시연 영상 링크

https://youtu.be/nq_mY2DS1T0