Devster

# NaverCloud 4기
# Final project 4조
# Devster


개발자를 꿈꾸는 학생들과
주니어 개발자들이 모인
개발 학원 커뮤니티 웹 사이트 DevSter

Devster

# Devster

# 발표 목차



Hello World !!

개발자를 꿈꾸는 학생들과
주니어 개발자들이 모인
개발 학원 커뮤니티 웹 사이트 DevSter

Devster

# 01.
# 프로젝트 개요

발표자 - 장수현


팀 소개

팀원 소개

주제 선정 이유

리팩토링 성과


# TFT 팀 소개

- 세미 프로젝트에서 파이널 프로젝트로 이어지는 Devster 사이트 리팩토링

- 반응형 모바일 사이트 Devster QR코드 한번씩 찍어주세요!

# TFT
# 팀원 소개

Devster 사이트를 함께 개발한 6명의 팀원을 소개합니다.
팀장 김동규 / 프론트 총괄 김규헌 / 백엔드 권현오 /
백엔드 김애리 / 프론트엔드 장수현 / 프론트엔드 정우영

jWT, Spring Security 로직 구현
로그인, 회원가입 API 구현
SSL 인증서 적용
프로토타입 프로젝트 생성

김동규

채용 게시판 백엔드&프론트엔드
공지게시판 백엔드
메인페이지 백엔드&프론트엔드

권현오

리뷰게시판 백엔드&프론트엔드
댓글 대댓글 구현
좋아요 싫어요 구현

김애리

리액트, 프론트 총괄
채팅 (웹소켓) 구현
유효성 검사 API

김규헌

자유게시판 백엔드&프론트엔드
사진 업로드 & 수정 로직 및 프론트
페이지 반응형 CSS 적용

장수현

마이페이지 백엔드&프론트엔드
이력서 서비스 구현

정우영

# 프로젝트 개요

## 주제 선정 이유

Devster는 개발자를 꿈꾸는 IT 학원생들을 위한 커뮤니티입니다.
애브리타임, OKKY 등 다양한 커뮤니티들이 존재하지만 IT 취업준비생들만을 위한 커뮤니티의 필요성을 느끼고 Devster를 개발하기로 했습니다.

## 리팩토링


같은 교집합을 가진
인증된 사용자들간의
정보공유 사이트

Devster

# 프로젝트 개요

### 리팩토링 #1

html 삽입 공격 등 취약했던 보안 강화

### 리팩토링 #2

REST API 강화를 통해 코드 표현 구조 개선

### 리팩토링 #3

컨트롤러의 복잡했던 로직을

서비스단으로 이동시켜

유지보수에 용이하게 프로젝트 구조를 개선

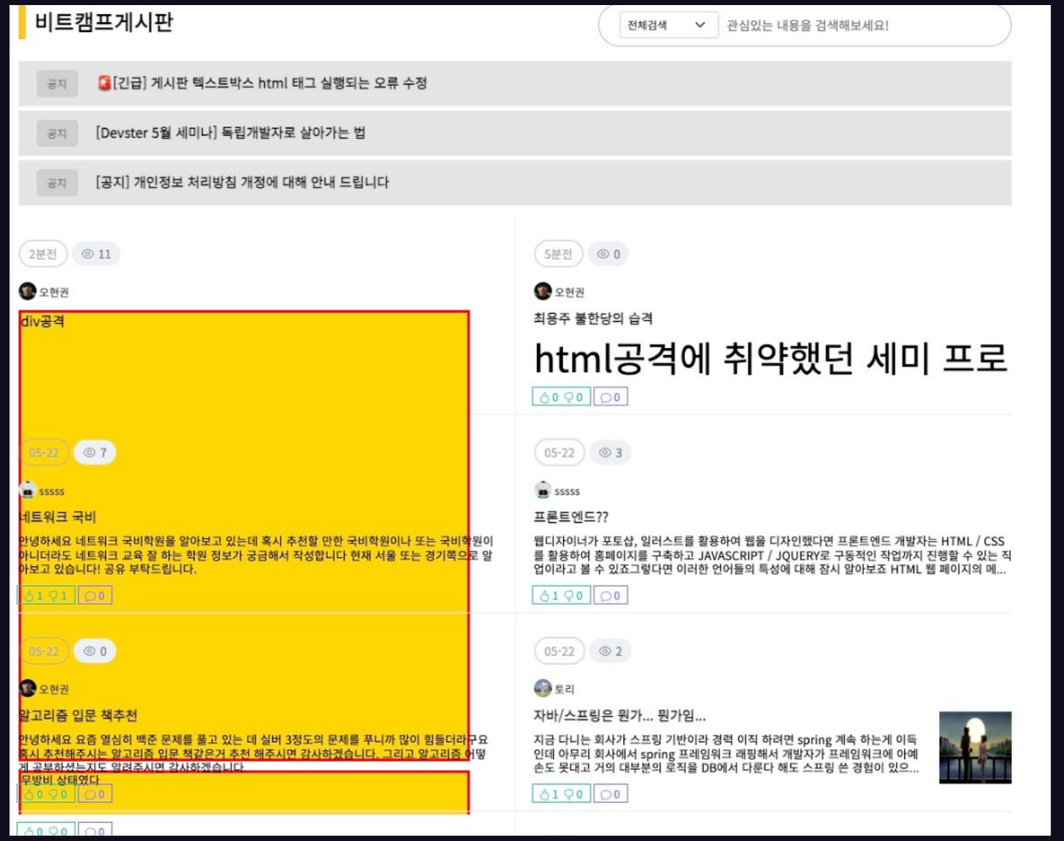
Devster

# 프로젝트 개요

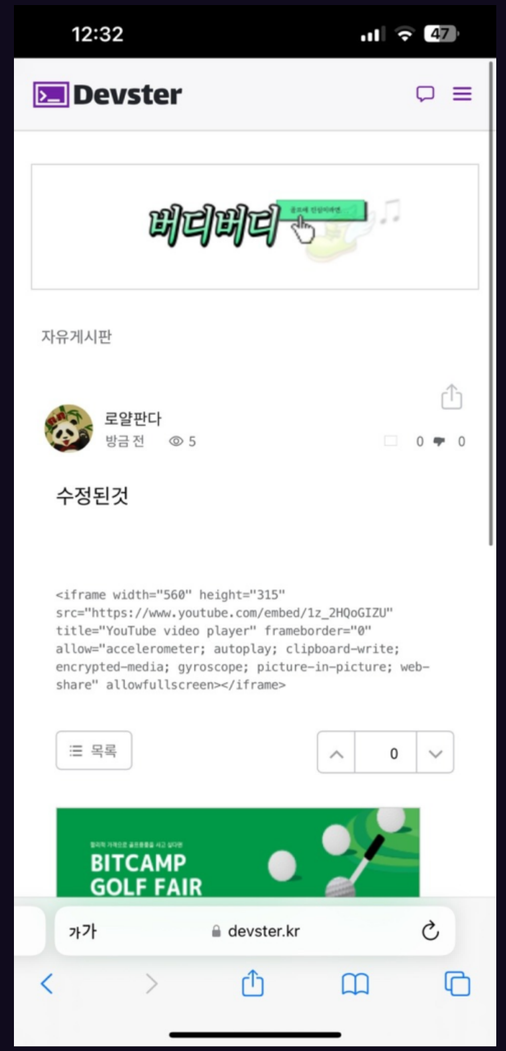
## Semi Project

● ● ● 리팩토링 #1

html 삽입 공격 등 취약했던 보안 강화




## Final Project



# 프로젝트 개요

## 리팩토링 #2

REST API 강화를 통해 코드 표현 구조 개선

### Semi Project

HireBoardController.java

```
권현오 *
@GetMapping("/hireboarddelete")
public String deleteHireBoard(int hb_idx){
    String hb_photo = hireService.getData(hb_idx).getHb_photo();
    storageService.deleteFile(bucketName, directoryPath: "hire    ",hb_photo);
    hireService.deleteHireBoard(hb_idx);
    return "redirect:list";
}
권현오
@GetMapping("/hireupdateform")
public String updateHireBoardform(@RequestParam(defaultValue = "1") int currentPage,
                                  @RequestParam(defaultValue = "0") int hb_idx, Model model)
{
    HireBoardDto dto=hireService.getData(hb_idx);
    model.addAttribute( attributeName: "dto", dto);
    model.addAttribute( attributeName: "currentPage", currentPage);
    return "/main/hire/hireupdateform";
}
권현오
@PostMapping("/hireupdate")
public String updateHireBoard(HireBoardDto dto,MultipartFile upload,int currentPage)
{
    String filename="";
    //사진선택을 한경우에는 기존 사진을 버켓에서 지우고 다시 업로드를 한다
    if(!upload.getOriginalFilename().equals("")) {
        //기존 파일명 알아내기
        filename=hireService.getData(dto.getHb_idx()).getHb_photo();
        //버켓에서 삭제
        storageService.deleteFile(bucketName,  directoryPath: "hire", filename);
        //다시 업로드후 업로드한 파일명 얻기
        filename=storageService.uploadFile(bucketName,  directoryPath: "hire", upload);
        dto.setHb_photo(filename);
    }
    dto.setHb_photo(filename);
    //수정
    hireService.updateHireBoard(dto);
    //수정후 내용보기로 이동한다
```

### Final Project

HireBoardController.java 8

src > main > java > data > controller > HireBoardController.java > Language Support for Java(TM) by Red Hat > { } data.controller

```
@GetMapping("/hboard/D0/{hb_idx}")
public Map<String,Object> getDetailPage(@PathVariable int hb_idx, int m_idx){
    return hireBoardService.getDetailPage(hb_idx,m_idx);
}



@DeleteMapping("/compmember/hboard/D1/{idx}")
public ResponseEntity<Void> deleteHireBoard(@PathVariable Integer idx){
    hireBoardService.deleteHireBoard(idx);
    return new ResponseEntity<>(HttpStatus.OK);
}


@PutMapping("/compmember/hboard/D1/{hb_idx}")
public ResponseEntity<HireBoardDto> updateHireBoard(@PathVariable int hb_idx, @RequestBody HireBoardDto dto) {
    hireBoardService.updateHireBoard(hb_idx,dto);
    return new ResponseEntity<HireBoardDto>(HttpStatus.OK);
}

@PostMapping("/compmember/hboard/D1/photo/{hb_idx}")
public ResponseEntity<String> updatePhoto(@PathVariable Integer hb_idx, @RequestBody List<MultipartFile> upload){
    return new ResponseEntity<String>(hireBoardService.updatePhoto(hb_idx,upload),HttpStatus.OK);
}

@DeleteMapping("compmember/hboard/D1/photo/{hb_idx}/{imageFileName}")
public ResponseEntity<Void> deletePhoto(@PathVariable Integer hb_idx, @PathVariable String imageFileName){
    hireBoardService.deletePhoto(hb_idx, imageFileName);
    return new ResponseEntity<>(HttpStatus.OK);
}


@GetMapping("/compmember/hboard/D1/form/{idx}")
public ResponseEntity<HireBoardDto> updateHireBoardForm(@PathVariable Integer idx){
    return new ResponseEntity<HireBoardDto>(hireBoardService.findByHbIdx(idx),HttpStatus.OK);
}
```

Devster



## Controller

FreeBoardController.java

```java
@GetMapping("/listajax")
@ResponseBody
public List<Map<String,Object>> list(int currentPage) {
    int totalCount = freeBoardService.getTotalCount();
    int perPage = 20; // 한 페이지당 보여줄 글 갯수
    int startNum; // 각 페이지에서 보여질 글의 시작번호
    int no; // 글 출력시 출력할 시작번호

    // 각 페이지의 시작번호 (1페이지: 0, 2페이지 : 3, 3페이지 6 ....)
    startNum = (currentPage - 1) * perPage;

    // 각 글마다 출력할 글 번호 (예 : 10개일 경우 1페이지 10, 2페이지 7...)
    // no = totalCount - (currentPage - 1) * perPage;
    no = totalCount - startNum;

    // 각 페이지에 필요한 게시글 db에서 가져오기
    List<FreeBoardDto> list = freeBoardService.getPagingList(startNum, perPage);

    List<Map<String, Object>> fulllList = new ArrayList<>();

    for (FreeBoardDto dto : list) {
        Map<String, Object> map = new HashMap<>();
        map.put("fb_idx", String.valueOf(dto.getFb_idx()));
        map.put("nickName", freeBoardService.selectNickNameOfMidx(dto.getFb_idx()));
        map.put("commentCnt", freeBoardService.commentCnt(dto.getFb_idx()));

        String m_photo = freeBoardService.selectPhotoOfMidx(dto.getFb_idx());

        if(m_photo.equals("no")) {
            m_photo = "/photo/profile.jpg";
        } else {
            m_photo = "http://kr.object.ncloudstorage.com/devster-bucket/member/"+freeBoardService.sele
        }
        map.put("m_photo",m_photo);
        map.put("fb_subject", dto.getFb_subject());
        map.put("fb_content", dto.getFb_content());
        map.put("fb_like", dto.getFb_like());
        map.put("fb_dislike", dto.getFb_dislike());
        map.put("fb_readcount", dto.getFb_readcount());
        map.put("fb_writeday", timeForToday(dto.getFb_writeday()));
```

## Service

FreeBoardService.java

src > main > java > data > service > FreeBoardService.java > Language Support for Java(TM) by Red Hat > { } data.service

```java
@Service
@Slf4j
public class FreeBoardService {
    private final FreeBoardRepository freeBoardRepository;
    private final MemberRepository memberRepository;
    private final FreeBoardLikeRepository freeBoardLikeRepository;
    private final NcpObjectStorageService storageService;
    private final FboardCommentRepository fboardCommentRepository;

    @Autowired
    public FreeBoardService(FreeBoardRepository freeBoardRepository, MemberRepository memberRepository, Fre
        this.freeBoardRepository = freeBoardRepository;
        this.memberRepository = memberRepository;
        this.freeBoardLikeRepository = freeBoardLikeRepository;
        this.storageService = storageService;
        this.fboardCommentRepository = fboardCommentRepository;
    }

    @Value("${aws.s3.bucketName}")
    private String bucketName;

    public FreeBoardDto insertFreeBoard(FreeBoardDto dto, HttpSession session) {
        try {
            if (session.getAttribute(name:"photo") != null) {
                dto.setFb_photo(session.getAttribute(name:"photo").toString());
            }
            FreeBoardEntity freeBoard = FreeBoardEntity.toFreeBoardEntity(dto);
            freeBoardRepository.save(freeBoard);
            session.removeAttribute(name:"photo");
            return dto;
        } catch (Exception e) {
            log.error(msg:"insert FreeBoard Error", e);
            throw e;
        }
    }

    public List<String> uploadPhoto(List<MultipartFile> upload, HttpSession session) {
        List<String> fullPhoto = new ArrayList<>();
        for (MultipartFile photo : upload) {
            fullPhoto.add(storageService.uploadFile(bucketName, directoryPath:"devster/fboard", photo));
        }
        if (session.getAttribute(name:"photo") != null) {
            storageService.deleteFile(bucketName, directoryPath:"devster/fboard", session.getAttribute(name
        }
        session.setAttribute(name:"photo", String.join(delimiter:",", fullPhoto));
        log.info(msg:"FreeBoard 사진 업로드 완료");
        return fullPhoto;
    }
```

### 리팩토링 #3

Controller 의 복잡했던 로직을
Service 단으로 이동시켜
유지보수에 용이하게 프로젝트 구조를 개선

Devster

# 02.
# 프로젝트
# 분석 및 설계

발표자 - 권현오

요구사항 정의서

아키텍쳐 구조도

화면 정의서

DB ERD

코딩 컨벤션

Gantt Chart

Devster

# 프로젝트 분석 및 설계

## 요구사항 정의서

프로젝트 분석과 설계 단계에서
구현해야하는 기능들을
대분류로 분류하고 ID를 부여해
프로젝트의 기능들을 정리해 보여주는
요구사항 정의서

| 요구사항 명세서 | | 작성자 | 장수현, 김애리 | 승인자 | 김동규 | | |
|---|---|---|---|---|---|---|---|
| | | 작성일 | 2023-06-29 | 버전 | v1.0 | | |
| 단계 | 분석 | 프로젝트명 | Devster | 마감일 | 2023-08-03 | | |

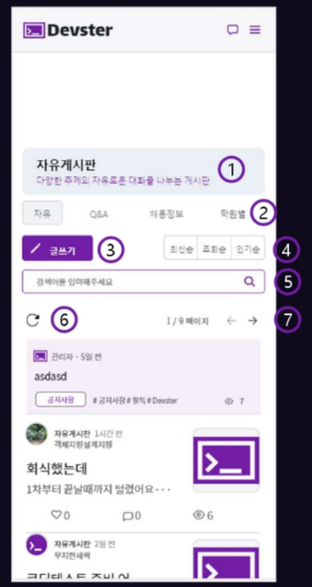
| 순번 | 요구사항 ID | 분류 | 기능명 | 상세 | 완료여부 | 비고 | 완료 날짜 |
|---|---|---|---|---|---|---|---|
| 1 | LOG01 | 로그인(일반회원) | 소셜 로그인 | 카카오, 네이버 | ☑ | | 2023.07.10 |
| 2 | | | 일반 로그인 | | ☑ | | 2023.07.10 |
| 3 | | | 아이디 찾기 | Java mail sender | ☑ | | 2023.07.13 |
| 4 | | | 비밀번호 찾기 | Java mail sender | ☑ | | 2023.07.17 |
| 5 | LOG02 | 로그인 (기업회원) | 이메일 찾기 | 네이버센스 | ☑ | | 2023.07.20 |
| 6 | | | 비밀번호 찾기 | 네이버 센스 | ☑ | | 2023.07.17 |
| 7 | SIGNUP01 | 회원가입 | 이메일인증 | 네이버센스 | ☑ | | 2023.07.20 |
| 8 | | | 일반 회원 유효성 검증 | 이메일 실 소유 확인<br>( 이메일 인증 ) | ☑ | | 2023.07.13 |
| 9 | | | 기업 회원 유효성 검증 | 휴대폰번호 실 소유 확인<br>( 휴대폰 문자인증 ) | ☑ | | 2023.07.10 |
| 10 | | | 기업회원 사업자 번호 인증 | 공공 데이터 포털 | ☑ | | 2023.07.17 |
| 11 | MYP01 | 마이페이지(일반회원) | 나의 정보 | 이름, 이메일,소속,<br>별명, 프로필 사진 | ☑ | | 2023.07.20 |
| 12 | | | 채용정보 북마크 | | ☑ | | 2023.07.20 |
| 13 | | | 이력서 작성 | 포트폴리오, 자격증, 이력서<br>인증을 위한 사진 업로드 | ☑ | | 2023.07.10 |
| 14 | | | 이력서 확인 | 본인 이력서 조회, 자기소개서 영문번역 | ☑ | | 2023.07.17 |
| 15 | | | 계정 설정 (수정) | 닉네임, 사진 수정 | ☑ | 이메일 수정 X | 2023.07.20 |
| 16 | | | 계정 탈퇴 | 이메일 본인 인증 성공시<br>회원 탈퇴 진행 | ☑ | | 2023.07.20 |
| 17 | | | 공지사항 | 공지사항 조회 | ☑ | | 2023.07.13 |
| 18 | MYP02 | 마이페이지(기업회원) | 회사 정보 | 조회 | ☑ | | 2023.07.17 |
| 19 | | | 계정 정보 수정 | 회사 정보 수정 및 담당자 정보 수정 | ☑ | | 2023.07.22 |
| 20 | | | 계정 정보 삭제 | 담당자 휴대폰 본인 인증 성공시<br>회원 탈퇴 진행 | ☑ | | 2023.07.20 |
| 21 | | | 구직자 리스트 출력 | 이력서 공개 설정된 회원<br>이력서 목록 조회 | ☑ | | 2023.07.10 |
| 22 | MAN | 관리자 전용 기능 | 학원 인증 | HRD 또는 고육 수강증 인증후<br>정회원 승급 | ☑ | | 2023.07.22 |
| 23 | | | 사업자 등록 인증 | 기업회원 사업자 등록증 인증후<br>정회원 승급 | ☑ | | 2023.07.13 |
| 24 | | | 공지사항 | 공지사항 게시글 작성 | ☑ | | 2023.07.20 |
| 25 | | | 게시글 리스트 출력 | | ☑ | | 2023.07.17 |
| 26 | | | 게시글 작성 | | ☑ | | 2023.07.22 |
| 27 | BOD<br>(01,03,04,05) | 게시판<br>일반(01)<br>학원별(03)<br>질문(04)<br>채용정보(05) | 게시글 수정 | | ☑ | | 2023.07.22 |
| 28 | | | 게시글 삭제 | | ☑ | | 2023.07.13 |
| 29 | | | 댓글 작성 | | ☑ | | 2023.07.17 |
| 30 | | | 대댓글 작성 | 자유게시판 | ☑ | | 2023.07.20 |
| 31 | | | 게시글 추천/비추천 | 실시간 동기화 | ☑ | | 2023.07.20 |
| 32 | | | 게시글 검색 | 페이지별/메인페이지 | ☑ | | 2023.07.22 |
| 33 | CRL | 크롤링 | 학원정보 | | ☑ | | 2023.07.10 |
| 34 | | | 회사정보 | | ☑ | | 2023.07.25 |
| 35 | REV | 리뷰 | 리뷰 작성 | | ☑ | | 2023.07.17 |
| 36 | | | 리뷰 수정 | | ☑ | | 2023.07.25 |
| 37 | | | 리뷰 삭제 | | ☑ | | 2023.07.25 |
| 38 | | | 회사 검색 | | ☑ | | 2023.07.13 |
| 39 | | | 회사정보 출력 (모달) | | ☑ | | 2023.07.17 |
| 40 | | | 별점 구현 | 별점 평균점수 산출 | ☑ | | 2023.07.25 |
| 41 | MES01 | 쪽지 | 쪽지 보내기 | | ☑ | | 2023.07.25 |
| 42 | | | 쪽지 리스트 출력 | | ☑ | | 2023.07.10 |
| 43 | | | 미확인 쪽지 알람 | | ☑ | | 2023.07.17 |
| 44 | RES | 이력서 | 이력서 작성 | | ☑ | | 2023.07.28 |
| 45 | | | 이력서 수정 | | ☑ | | 2023.07.28 |
| 46 | | | 이력서 삭제 | | ☑ | | 2023.07.28 |
| 47 | | | 이력서 영문 번역 | Papago | ☑ | | 2023.07.13 |
| 48 | MES03 | 소켓 | 학원별 단체 채팅방 | | ☑ | | 2023.07.10 |

# 프로젝트 분석 및 설계

## 화면 정의서

프로젝트를 본격적으로 구현하기 전
필요한 기능, 정책 등
화면 내 서비스의 모든 요소에 대해
정의하여 서비스 화면의 이해를
돕기 위한 문서

| Page Title. | 일반 게시판 | Group Title. | 게시판 리스트 |
|---|---|---|---|



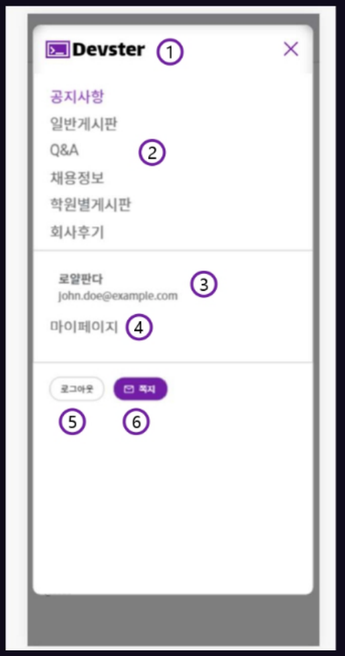

| Description. | Page. | 7 |
|---|---|---|

Summery.

- 상단바의 버튼으로 다른 게시판으로 이동.
- 새로고침 아이콘 터치시, 페이지 리로딩.
- 글쓰기 버튼 클릭시 로그인상태, Role 확인후 글쓰기 페이지로 이동

| | |
|---|---|
| 1 | 게시판 이름, 간단한 설명 |
| 2 | 게시판 선택 버튼 |
| 3 | 글 작성 버튼 |
| 4 | 게시글 리스트 정렬 버튼 |
| 5 | 게시글 검색 |
| 6 | 리스트 새로고침 |
| 7 | 페이지 이동 버튼 |

| Page Title. | 매뉴바 모달 | Group Title. | 모달 |
|---|---|---|---|




| Description. | Page. | 8 |
|---|---|---|

Summery

- 로그인한 회원에 따라서 다르게 출력되는 로직.
- 회원의 프로필사진 없을시, 사진 아예 미출력.
- 로그인, 로그아웃 상태에 따라 변화는 버튼

| | |
|---|---|
| 1 | 로고 겸 홈화면 이동 버튼 |
| 2 | 게시판 선택 버튼 |
| 3 | 현재 접속 회원 정보 |
| 4 | 마이페이지 이동 버튼 |
| 5 | 로그인, 로그아웃 버튼 ( 로그인 상태에 따라 변화 ) |
| 6 | 쪽지, 회원가입버튼 ( 로그인 상태에 따라 변화 ) |

Devster

# 프로젝트 분석 및 설계

네이버 클라우드에 VPC를 생성하고

Web Subnet과 DB Subnet 구성

Internet Gateway, Router,

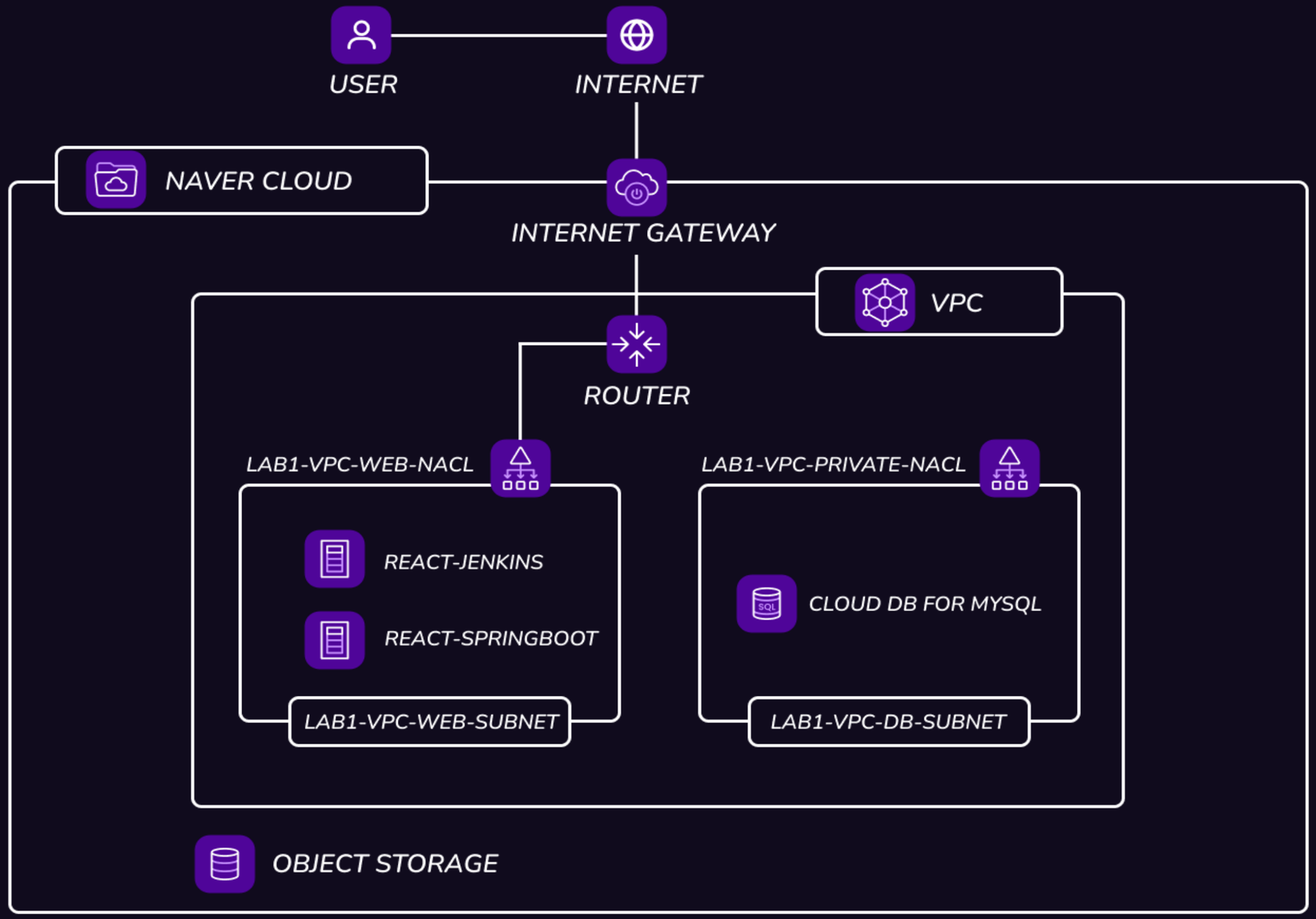
NACL 설정을 통한 네트워크 연결

오브젝트 스토리지 연결

가비아 호스팅 서비스를 통한

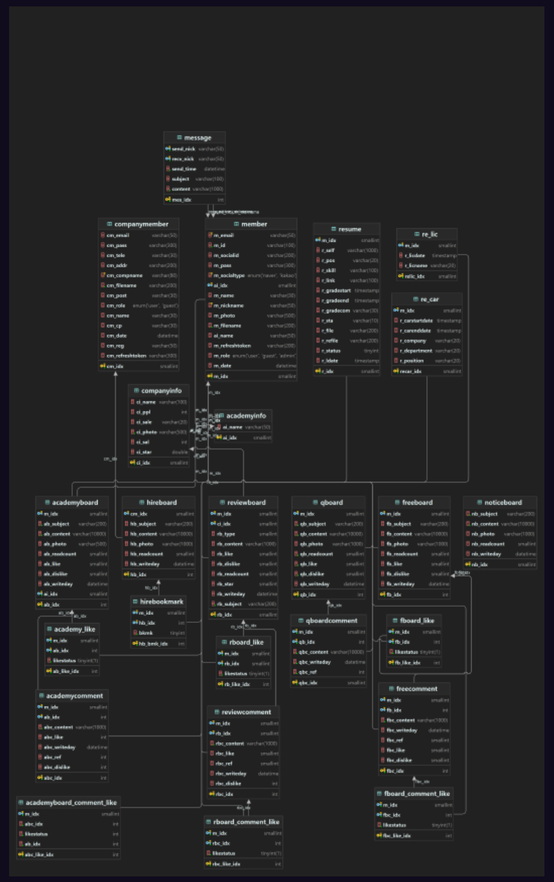
도메인 연결

아키텍쳐 구조도

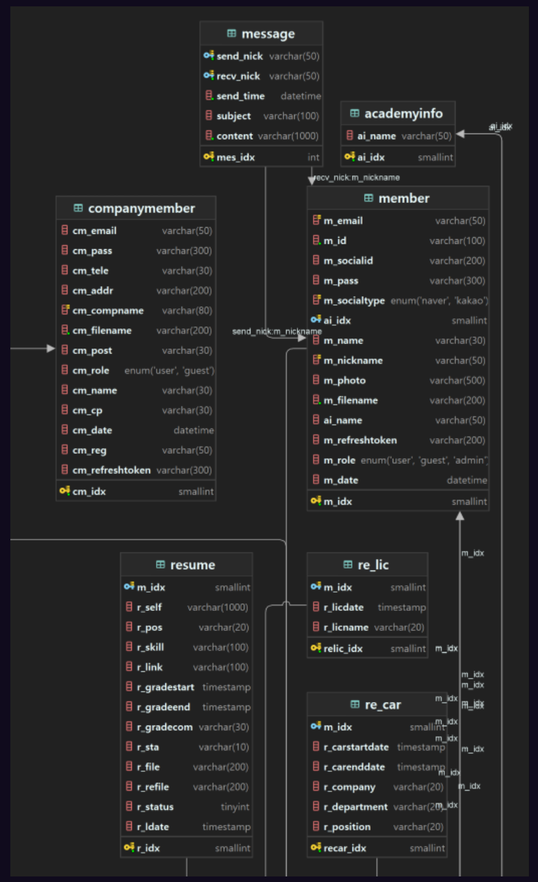


Devster

# 프로젝트 분석 및 설계

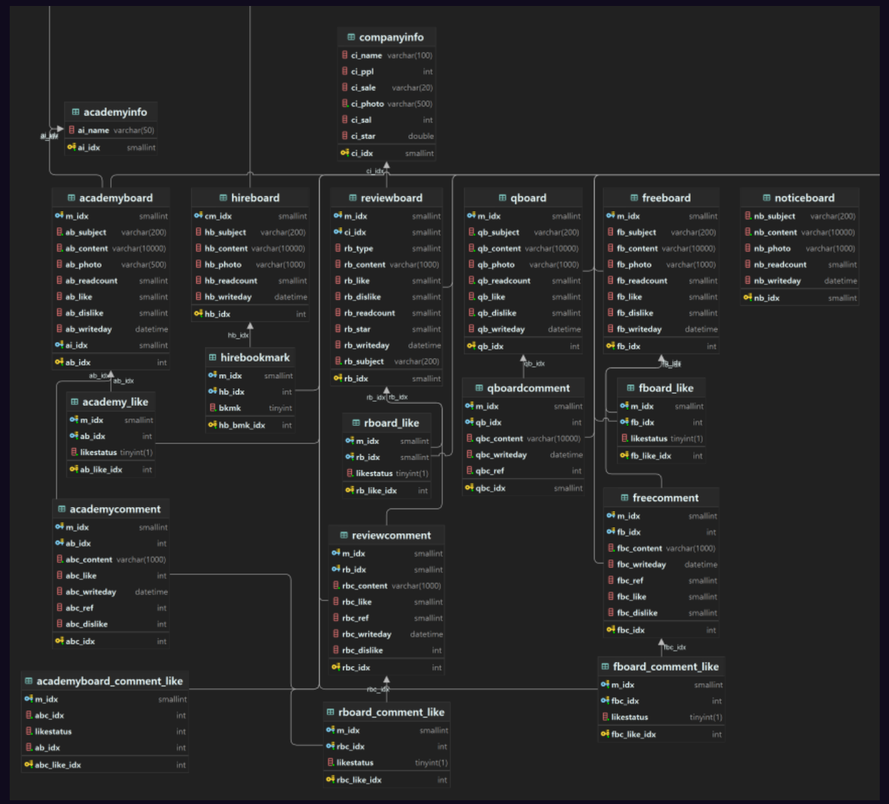


전체 Table




member 관련 Table 구조

## board 관련 Table 구조




DB ERD

Devster

# 프로젝트 분석 및 설계

## JAVA Convention

1.네이밍(Naming)
[identifier-char-scope]
1.1 식별자에는 영문/숫자/언더스코어만 허용
변수명, 클래스명, 메서드명 등에는 영어와 숫자만을 사용한다. 상수에는 단어 사이의 구분을 위하여 언더스코어(_)를 사용한다. 정규표현식 '[^A-Za-z0-9_]'에 부합해야 한다.

1.2 한국어 발음대로의 표기 금지
[avoid-korean-pronounce]
식별자의 이름을 한글 발음을 영어로 옮겨서 표기하지 않는다. 한국어 고유명사는 예외이다.
나쁜 예 : moohyungJasan (무형자산)
좋은 예 : intangibleAssets (무형자산

1.3 패키지 이름은 소문자로 구성)
[package-lowercase]
패키지 이름은 소문자를 사용하여 작성하되 단어별 구분을 위해 언더스코어(_)를 사용한다.
나쁜 예
package com.navercorp.apigateway
좋은 예
package com.navercorp.api_gateway

2.선언(Declarations)
2.1 소스파일당 1개의 탑레벨 클래스 당기
[1-top-level-class]
탑레벨 클래스(Top level class)는 소스 파일에 1개만 존재해야 한다. ( 탑레벨 클래스 선언의 컴파일타임 에러 체크에 대해서는 Java Language Specification 7.6 참조 )

나쁜 예
public class LogParser {
}

class LogType {

ResumeService.java 3

src > main > java > data > service > ResumeService.java > Language Support for Java(TM) by Red Hat > {} data.service

```java
@Service
@Slf4j
public class ResumeService {

    private final ResumeRepository resumeRepository;
    private final ResumeCareerRepository resumeCareerRepository;
    private final ResumeLicenseRepository resumeLicenseRepository;
    private final NcpObjectStorageService storageService;
    private final JwtService jwtService;


    @Value("${aws.s3.bucketName}")
    private String bucketName;

    @Value("${naver.translate.client_id}")
    private String client_id;

    @Value("${naver.translate.client_secret}")
    private String client_secret;

    public ResumeService(ResumeRepository resumeRepository, NcpObjectStorageService storageService, ResumeCareerR
        this.resumeRepository = resumeRepository;
        this.storageService = storageService;
        this.resumeCareerRepository = resumeCareerRepository;
        this.resumeLicenseRepository = resumeLicenseRepository;
        this.jwtService = jwtService;
    }

    public String insertResume(ResumeDto dto, List<ResumeCareerDto> resumeCareerDtoList, List<ResumeLicenseDto>
        ResumeEntity entity = new ResumeEntity();

        String file = (String) session.getAttribute( name: "file");
        if(file != null) {
            dto.setR_file(file);
        }

        String reFile = (String) session.getAttribute( name: "refile");
        if(reFile != null) {
            dto.setR_refile(reFile);
        }

        resumeRepository.save(entity.toResumeEntity(dto));
        log.info( msg: "이력서 기본정보 등록 완료.");

        for(ResumeCareerDto careerDto : resumeCareerDtoList) {
            ResumeCareerEntity resumeCareerEntity = new ResumeCareerEntity().toResumeCareerEntity(careerDto);
            resumeCareerRepository.save(resumeCareerEntity);
        }
        log.info( msg: "이력서 경력 등록완료.");
```

## React Convention

1.Naming Conventions
1.1 폴더명은 소문자로
좋은 예 : components, member, board …

1.2 파일명은 첫글자는 대문자로
좋은 예 : Header.js, Footer.js, SignUp.js …

1.3 속성명/변수명/함수명/inline스타일은 Camel Case를 사용해서 작성하기
좋은 예 : onClick, onSubmit, style={{backgroundColor..}} …

2.Bug Avoidance
2.1 null 또는 undefined일 수 있는 값은 optional chaining 연산자(?.)를 사용한다
좋은 예 : obj?.prop; , arr?.[index] , data?.map(…) …

2.2 전달된 매개변수가 유효한지 확인하기 위해 guard pattern을 사용한다
*Guard Pattern: Guard Pattern은 특정 조건을 만족하지 않는 경우에는 함수 또는 코드 블록의 실행을 중단하도록 설계된 패턴입니다. 리액트에서는, 컴포넌트에 전달된 props가 예상한 형태를 가지고 있는지 검사할 때 사용될 수 있습니다.
좋은 예 : javascript
Copy code
const UserProfile = ({ user }) => {
// user 객체가 존재하고, user.name 속성이 있는지 확인 (Guard Pattern)
if (!user || !user.name) {
return null; // 유효하지 않으면 렌더링 중단
}

return {user.name}'s Profile ;
};

2.3 비교연산자 사용시 '===, !==' 를 사용하여 비교한다
좋은 예 : eg) if(data.name === '4조') {…} ,
data.name===='4조'?'yeah':'다시입력';

DevChat.js

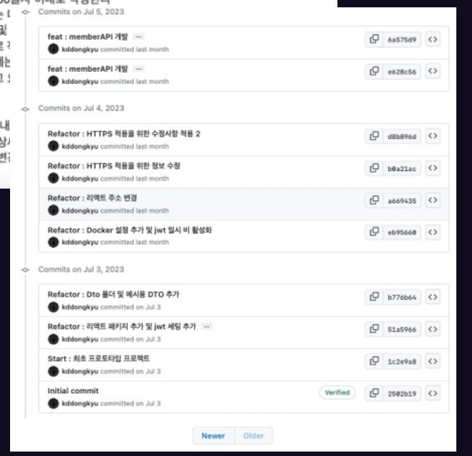
src > main > react.Js > my-app > src > pages > devchat > DevChat.js > …

```jsx
function DevChat(props) {
    const dispatch = useDispatch();
    const connected = useSelector(state => state.devChat.connected);
    const { enqueueSnackbar } = useSnackbar();
    const toastAlert = ToastAlert(enqueueSnackbar);

    useEffect(() => {
        const handleBeforeUnload = () => {
            dispatch(wsDisconnect());
        }
        window.addEventListener('beforeunload', handleBeforeUnload);
        return () => {
            window.removeEventListener('beforeunload', handleBeforeUnload);
        }
    }, [dispatch]);

    useEffect(() => {
        if (!connected) {
            const handleOnConnect = async () => {
                const de = checkToken();
                try {
                    const res = await axiosIns.get(`/api/member/D1/${de.idx}`);
                    if (res?.status === 200) {
                        dispatch(setAi_idx(res.data.ai_idx));
                        dispatch(setRoomName(res.data.ai_name));
                        dispatch(setUserName(res.data.m_nickname));
                        dispatch(setUserProfile(res.data.m_photo));
                        dispatch(wsConnect(res.data.ai_idx));
                    }
                } catch (error) {
                    jwtHandleError(error, toastAlert);
                }
            }
            handleOnConnect();
        }
    }, [connected]);

    return (
        <div>
            <div className='chat-list'>
                <Room />
            </div>
        </div>
    );
}

export default DevChat;
```

## Git Convention

1.Commit Message Structure
1.1 기본적인 커밋 메시지 구조 (각 파트는 빈줄로 구분한다.)
좋은 예 : Feat: 회원 가입 기능 구현

2.Commit Type
2.1 태그 : 제목 형식으로 기입
Feat -> 새로운 기능을 추가
Fix -> 버그 수정
Design -> CSS 등 사용자 UI 디자인 변경
!BREAKING CHANGE -> 커다란 API 변경의 경우
!HOTFIX -> 급하게 치명적인 버그를 고쳐야하는 경우
Style -> 코드 포맷 변경, 세미 콜론 누락, 코드 수정이 없는 경우
Refactor -> 프로덕션 코드 리팩토링
Comment -> 필요한 주석 추가 및 변경
Docs -> 문서 수정
Test -> 테스트 코드, 리팩토링 테스트 코드 추가, Production Code(실제로 사용하는 코드) 변경 없음
Chore -> 빌드 업무 수정, 패키지 매니저 수정, 패키지 관리자 구성 등 업데이트, Production Code 변경 없음
Rename -> 파일 혹은 폴더명을 수정하거나 옮기는 작업만인 경우
Remove -> 파일을 삭제하는 작업만 수행한 경우
3.Subject
3.1 제목은 50글자 이내로 작성한다
3.2 첫글자는 
3.3 마침표 및
3.4 영문으로 
3.5 과거시제는
3.6 간결하고 
4.Body
4.1 72자 이내
4.2 최대한 상
4.3 어떻게 변

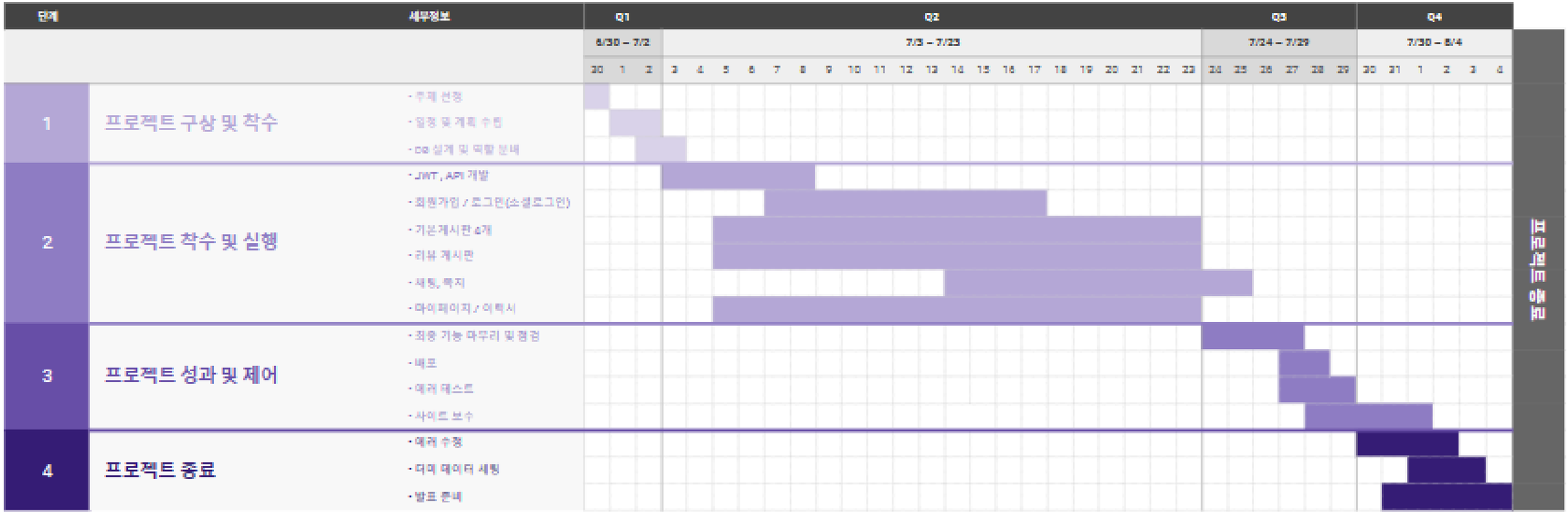



코딩 컨벤션

# Devster GanttChart

Gantt Chart

| 프로젝트 이름 | Devster | 팀명 | TFT (Task Force Team) |
|---|---|---|---|
| 팀장 | 김동규 | 기간 | 23.06.30 ~ 23.08.03 |



Devster

# 03.
# 시연 영상

발표자 - 김애리

# 시연 영상

Devster

# 04.
# 기술 소개 part 1

발표자 - 김규헌

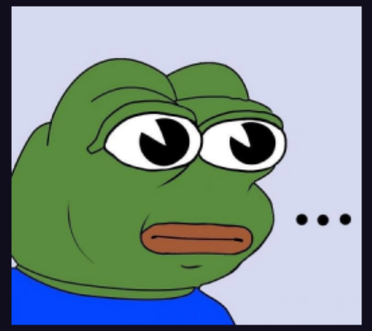

사용한 기술 스택

기술 설명 #1

- Redux 및 전반적인 프론트 기술
- 채팅 (웹소켓)

# Tech Stack

- FrontEnd: React18, HTML5, CSS3, JavaScript. Toast UI, redux, Stomp-JS
- BackEnd: JAVA 11, SpringBoot, Lombok, Gradle, JPA, MyBatis, Tomcat, Spring WebSocket, HTTPs, DNS, JWT, Spring Security, Oauth
- Naver Cloud & API: Naver Login, Kakao Login, Naver Sens, Naver Translate
- CI/ CD & Tools: WorkBench, Figma(locofy), IntelliJ, VS Code, Postman, Web Scraping(BeatifulSoup), Jenkins, Docker, Docker Hub, Gabia Domain

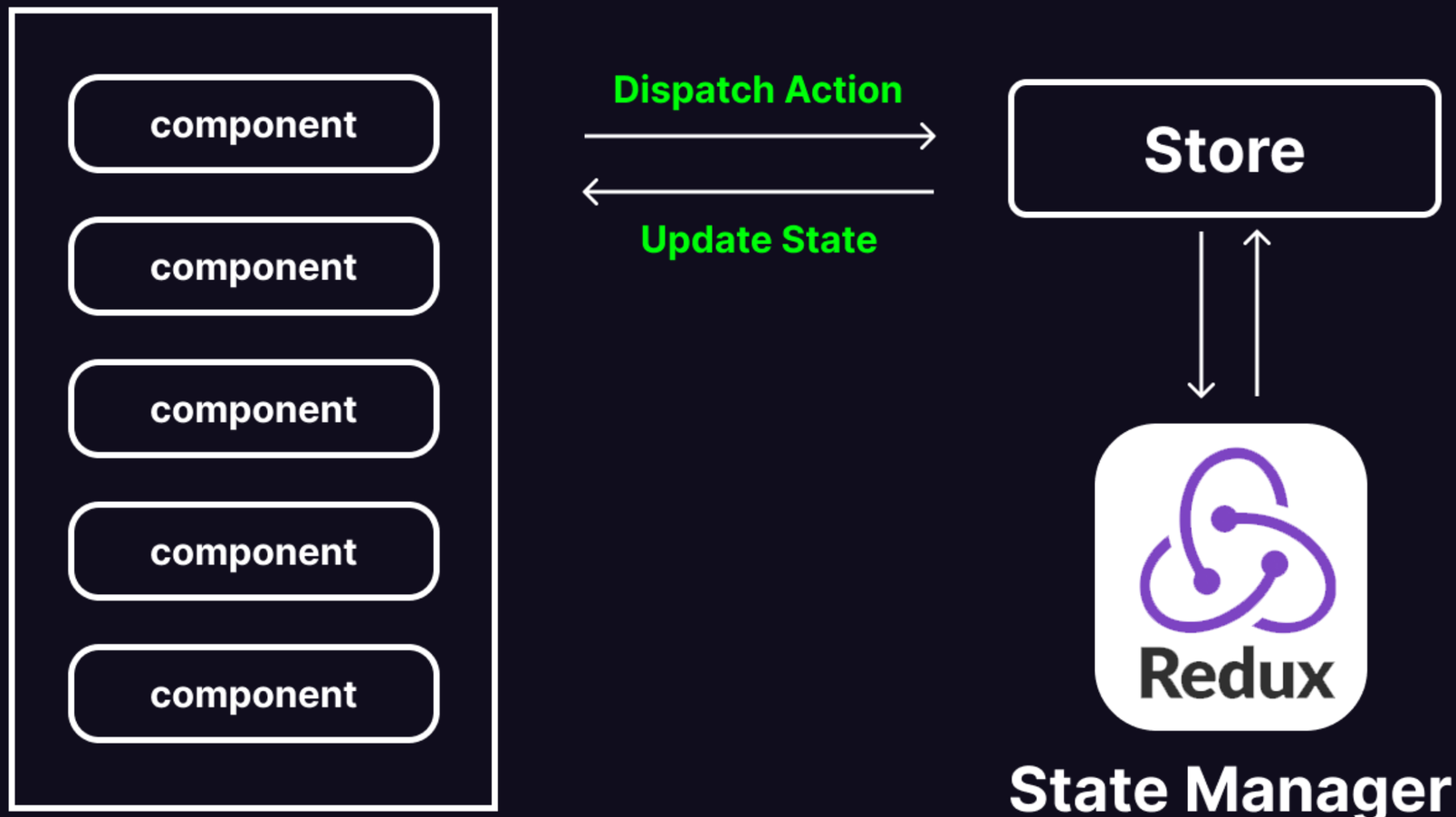


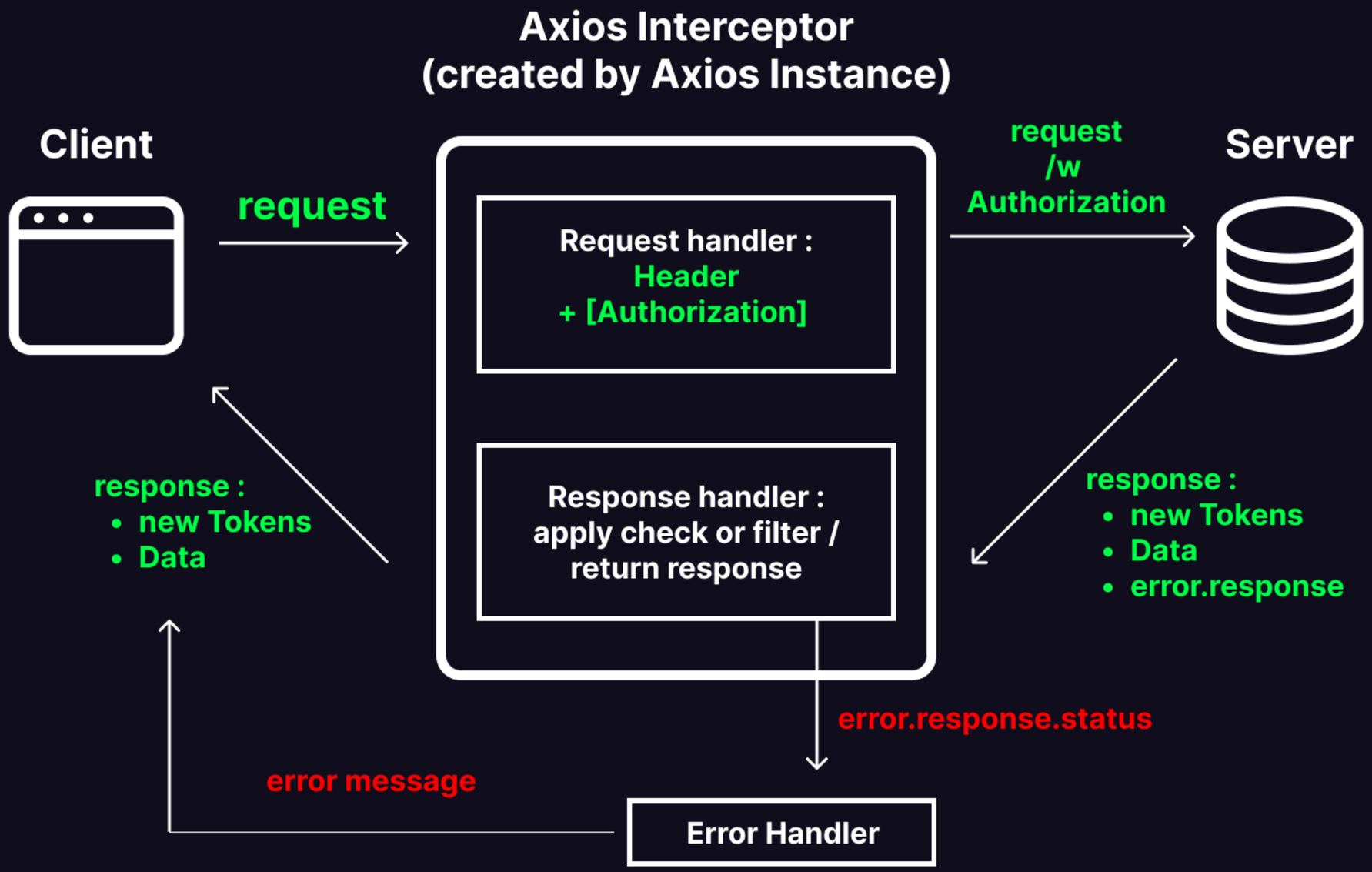


# Before Redux

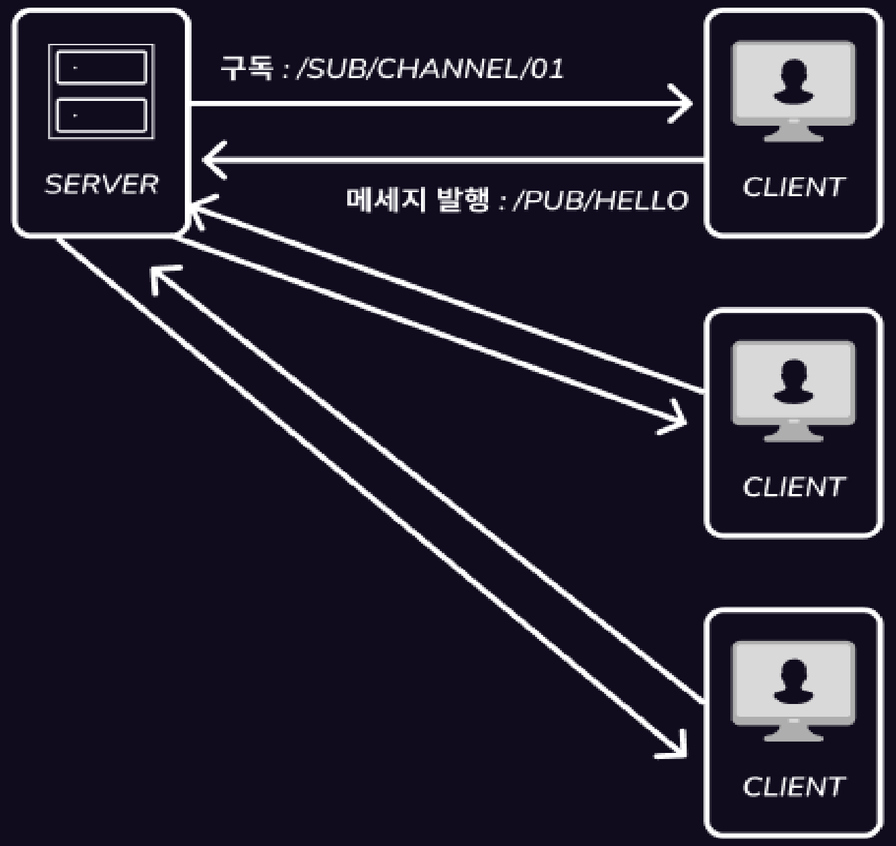
Devster

# After Redux




Axios Interceptor
(created by Axios Instance)
Client
request
Request handler :
Header
+ [Authorization]
request
/w
Authorization
Server
response :
• new Tokens
• Data
Response handler :
apply check or filter /
return response
response :
• new Tokens
• Data
• error.response
error.response.status
error message
Error Handler

구독 *(SUB)* 이면서 발행 *(PUB)* 역할을 동시에
-> 양방향 통신




# 채팅 (웹소켓)

Devster

# 04.
# 기술 소개 part 2

발표자 - 김동규


기술 설명 #2

- JPA와 MyBatis의 혼용
- HTTPS (ssl 인증서)
- 보안 (JWT)
- CI / CD

# JPA와 MyBatis의 혼용

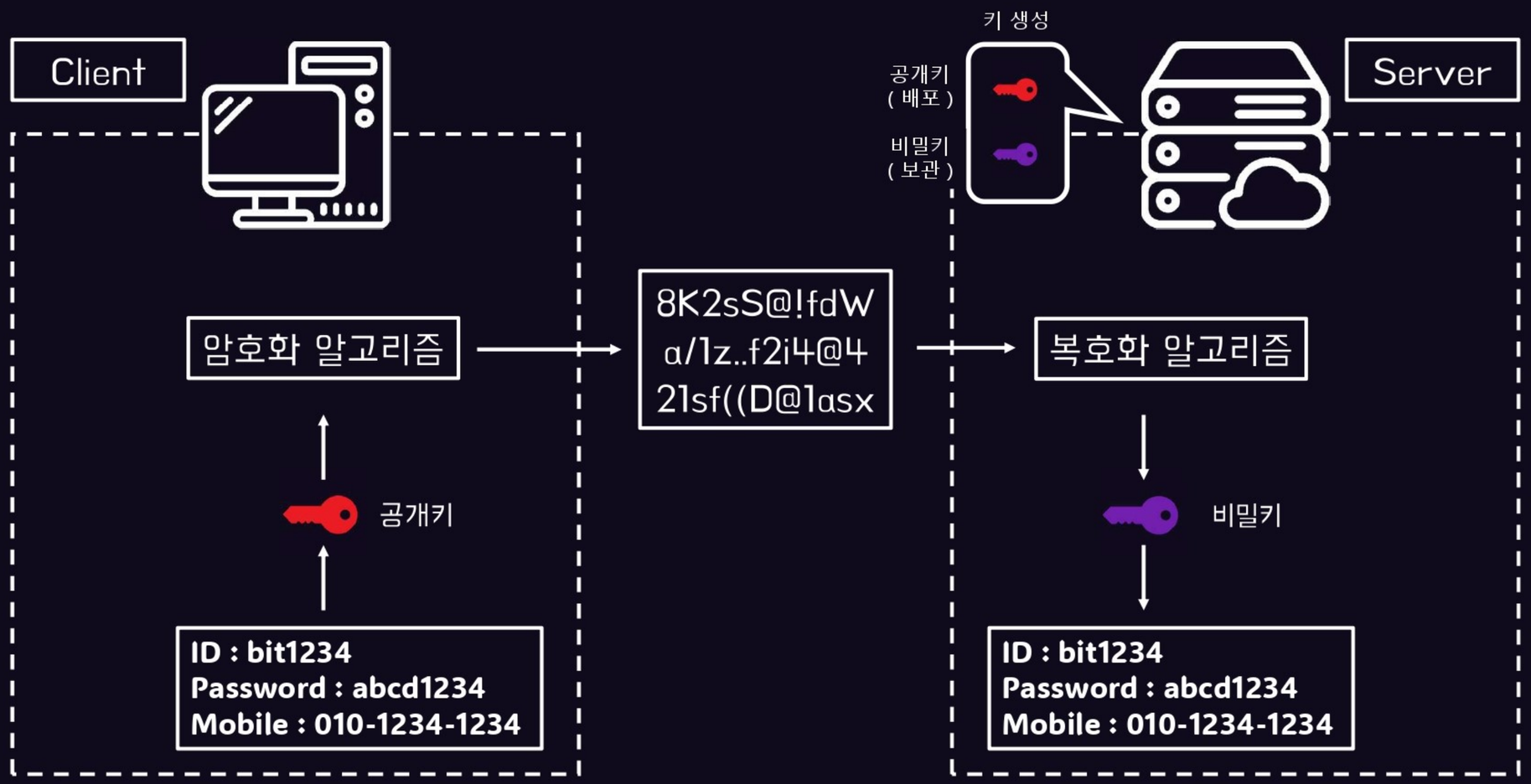
```
6개 사용 위치  kddongkyu
public interface QboardCommentRespository extends JpaRepository<QboardCommentEntity, Integer> {
    1개 사용 위치  kddongkyu
    List<QboardCommentEntity> findAllByQBidxAndQBcrefEqualsOrderByQBcwritedayDesc(int QBidx, int QBcref);
```

JPA

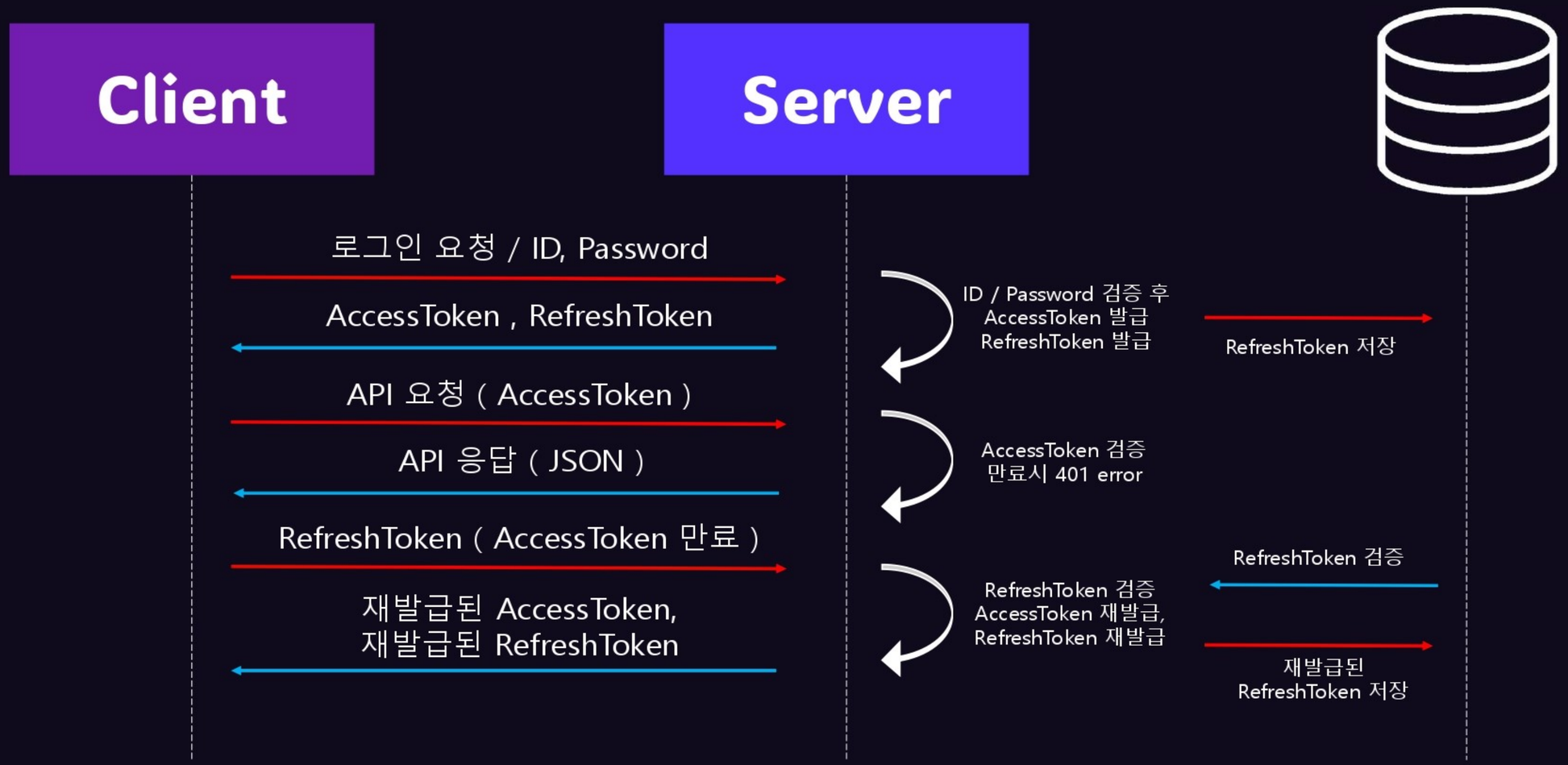
```
<select id="findAllComments" parameterType="map" resultType="QboardCommentEntity">
    SELECT *
    FROM QboardCommentEntity
    WHERE QBidx = #{QBidx} AND QBcref = #{QBcref}
    ORDER BY QBcwriteday DESC
</select>
```

My Batis

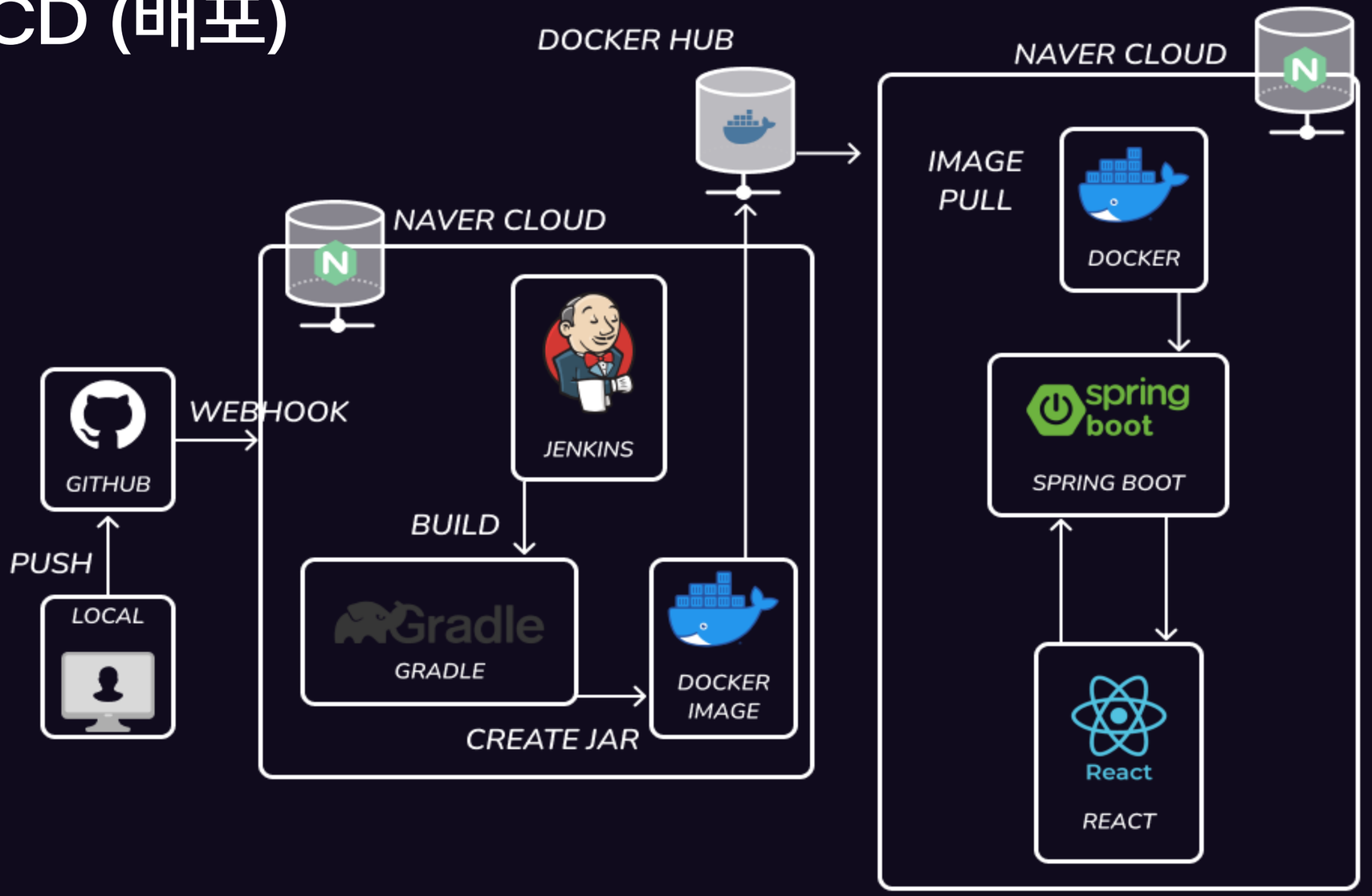
# HTTPS (ssl 인증서)



# 보안 ( JWT )


Client
Server
로그인 요청 / ID, Password
AccessToken , RefreshToken
API 요청 ( AccessToken )
API 응답 ( JSON )
RefreshToken ( AccessToken 만료 )
재발급된 AccessToken,
재발급된 RefreshToken
ID / Password 검증 후
AccessToken 발급
RefreshToken 발급
RefreshToken 저장
AccessToken 검증
만료시 401 error
RefreshToken 검증
RefreshToken 검증
AccessToken 재발급,
RefreshToken 재발급
재발급된
RefreshToken 저장

# CI / CD (배포)


DOCKER HUB
NAVER CLOUD
NAVER CLOUD
IMAGE PULL
DOCKER
JENKINS
WEBHOOK
GITHUB
spring boot
SPRING BOOT
BUILD
PUSH
LOCAL
Gradle
GRADLE
DOCKER IMAGE
CREATE JAR
React
REACT

Devster

# 05.
# 프로젝트 후기


후기 및 소감

마무리인사

# 개선점 & 차후 계획

## 개선 & 보완점

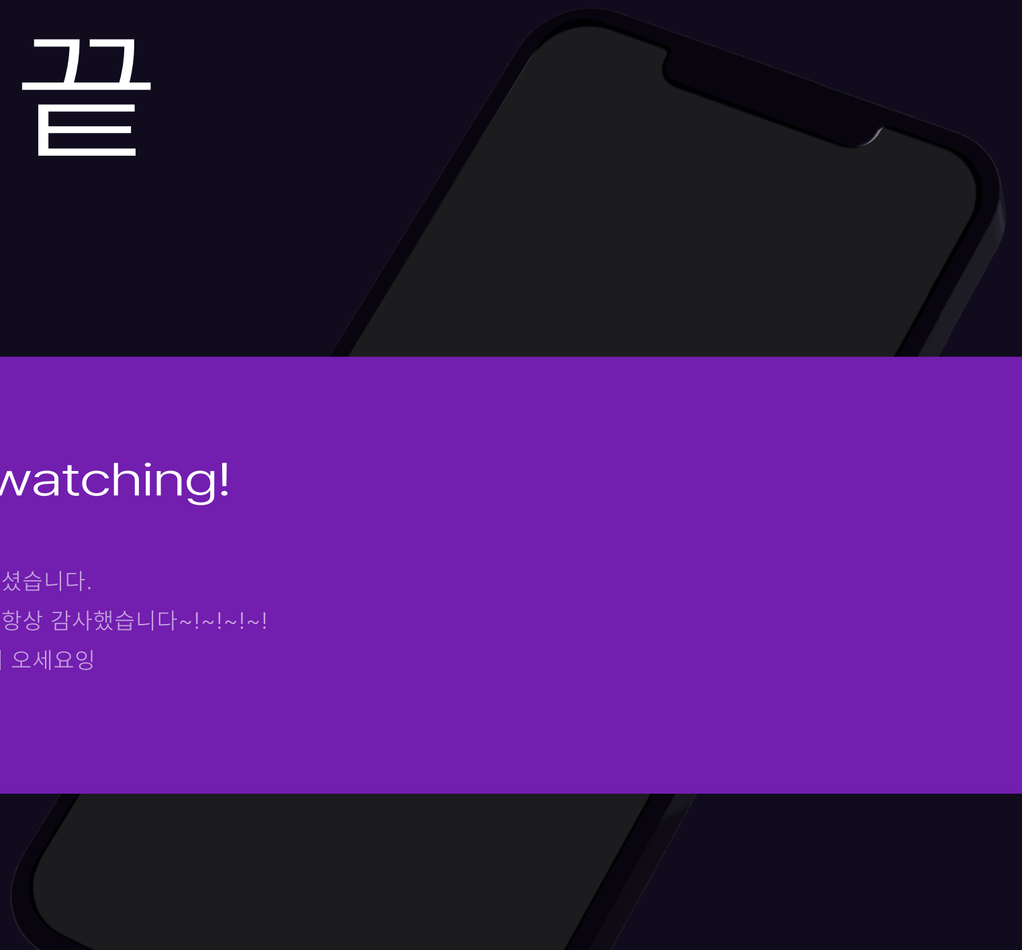
- 프로젝트 / 스터디 그룹 모집하는 기능
- 기업회원들만을 위한 기능 추가
- 실제 배포 및 서비스를 위한 서버 구축 및 운영

## 유지보수 계획

- 추가적인 보안 이슈가 확인 될 때마다 즉시 해결할 예정
- 지속적인 코드 최적화
- 시간날 때 추가 기능 구현
- 웹 전용 사이트 제작

Devster

# Devster
# 발표 끝



## Thank you for watching!

6게월간 너무 고생 많으셨습니다.
수료하고 나서도 자주 만나요 언니 오빠들 항상 감사했습니다~!~!~!~!
대부도 조심해서 잘 다녀 오세요잉