

특장차 제조/공정 관리 애플리케이션

제작 기간 : 2023.04.20 ~ 2023.06.02

조원 : 강유준, 김소윤, 김성중, 이소우, 최병권

목차

01

기획의도

02

사용 기술

03

Back-end

04

Front-end

01

기획의도

기획의도

그동안 코로나로 해외여행이 자유롭지 못했고
해외여행을 가지 못하는 대신 캠핑을 시작한
국민의 수가 많이 늘어났습니다
한국관광공사에 따르면 2022년 국내 캠핑 시장
규모는 약 6조3000억원으로 추정되어
동년 9월 말 기준 전국 야영장 또한 3205개로
역대 최대를 기록하는 등 캠핑이 트렌드가
되고 있습니다
이렇게 캠핑 장비와 캠핑카에 대한 관심이
계속 증가하고있어서 기획하게 되었습니다

업무를 진행 하다보면 직원들간 또는
직원과 고객들이 소통하고 마주할 일이
매우 많습니다
계약 관리 API는 기업과 고객용 API를 만들고
기업(현장, 관리)과 고객 App을 나누어
공정별로 업무 진행 상황을 계속 업로드 하여
서로 피드백을 주고 받기 편하도록 설계했으며
수시로 업로드 되기 때문에 고객이 계약 사항의
진행 정도를 바로 확인할 수 있습니다

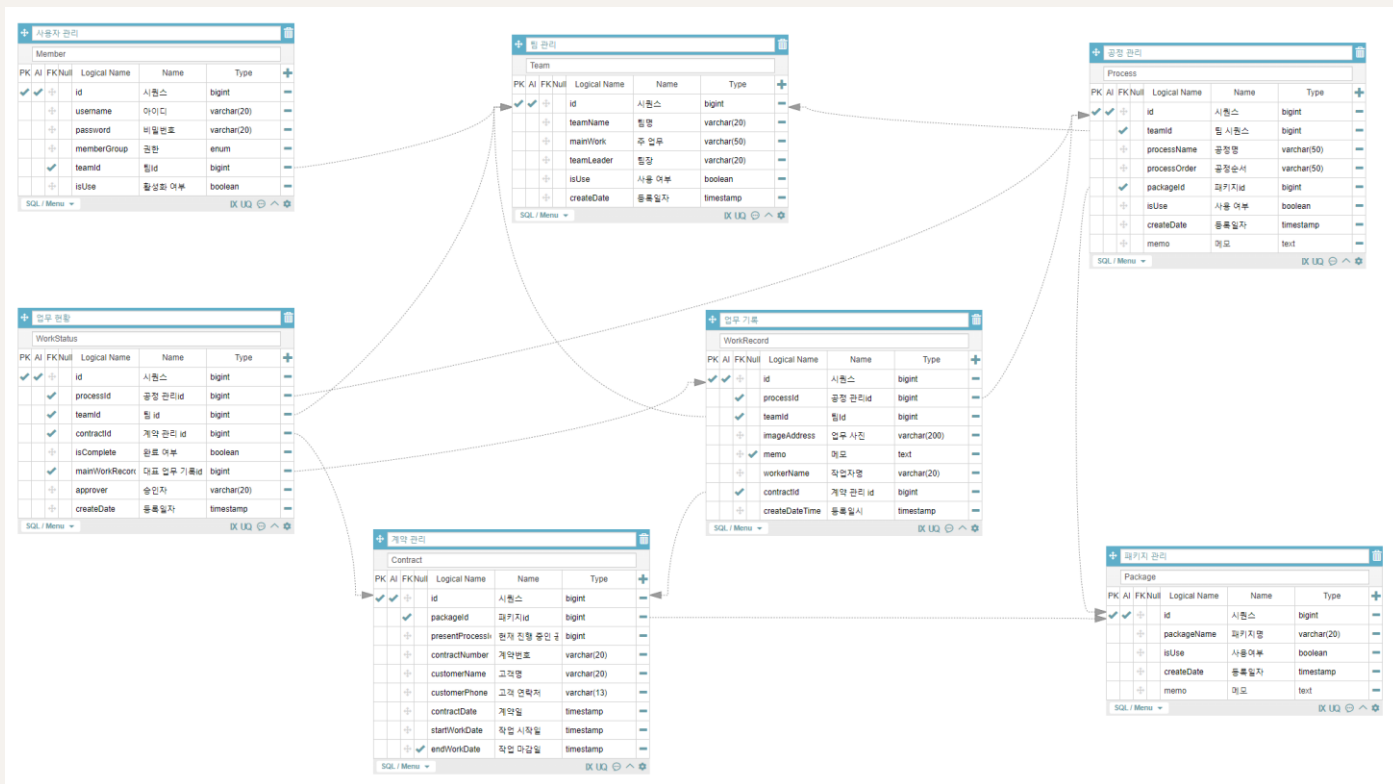
요구기능 정의서 - 1

No.	Module	요구사항 CATEGORY	요구사항명	상세 설명	필수 데이터
1	api-member	MEMBER	사용자 등록	자체 회원가입 기능	ID, 비밀번호, 권한, 팀 아이디
			사용자 목록 가져오기	전체 가져오기	
			사용자 활성화 여부 수정	활성화 여부 수정	
			로그인	로그인 필수. 로그인 안 했을 시 로그인창만 이용 가능	ID, 비밀번호
2		TEAM	로그아웃	로그아웃 기능	
			팀 등록	팀 등록하기	팀명, 주 업무, 팀장이름
			팀 목록 가져오기	전체 가져오기	
			팀 활성화 여부 수정	활성화 여부 수정	
3		PACKAGE	패키지 등록	패키지 등록하기	패키지명, 메모
			패키지 수정	패키지명 수정	패키지명
			패키지 활성화 여부 수정	활성화 여부 수정	활성화 여부
			사용 중인 패키지 목록 가져오기	사용여부 TRUE 인 것만 가져오기	
4	api-process	PROCESS	패키지 목록 전체 가져오기	등록된 패키지 전체 가져오기	
			공정 등록	공정 등록하기	공정명, 작업 팀 ID, 공정 순서, 패키지ID, 메모
			전체 공정 목록	공정 전체 목록 가져오기	
			사용 중인 전체 공정 가져오기	활성화 여부 TRUE인 것만 가져오기	
			패키지명 검색하여 가져오기		
			공정명 검색하여 가져오기		
			패키지 별 사용 중인 공정 가져오기	활성화 여부 TRUE이면서 패키지 번호(REQUEST) 원하는 목록 가져오기	패키지 ID
			공정 정보 수정	패키지 명 / 작업 팀 / 메모 수정	패키지ID, 팀ID, 메모
			공정 활성화 여부 수정	활성화 여부 수정	

요구기능 정의서 - 2

5		CONTRACT	계약서 등록	계약서 등록 -> 업무 현황 등록	패키지ID, 공정ID, 계약번호, 고객명, 연락처, 계약일자
			전체 계약서 목록 가져오기	전체 계약서 목록 가져오기	
			진행중인 계약 가져오기	마감일자가 NULL인 것만 가져오기	
			계약번호로 검색해서 가져오기	계약번호 (REQUEST) 일치하는 것 가져오기	계약번호
			완료된 계약 가져오기	마감일자가 NOT NULL인 것만 가져오기	
			연락처 수정	연락처 수정	계약서ID, 연락처
			날짜 범위 내 계약 가져오기	날짜 범위 내 계약 가져오기(계약 일자 기준 해당 범위 내)	찾을 범위(시작날짜 / 종료날짜) 날짜
6	api-work	WORK_STATUS	전체 업무 현황 보기 (팀장용)	전체 업무 보기(팀장용)	
			팀ID 별로 업무현황 보기	팀ID 별로 업무현황 확인	팀ID
			승인자 별로 가져오기	승인자 별로 업무현황 가져오기	승인자명
			날짜 별로 가져오기	날짜 별로 업무현황 가져오기	찾을 범위(시작날짜 / 종료날짜) 날짜
			업무현황 수정(확인)	완료 여부, 대표기록, 승인자 수정, 업무현황 다음 진행 업무 등록	업무 기록ID, 승인자명
			작업 목록ID에 계약번호 및 전화번호 전체로 업무현황	고객용 진행도 파악 / 계약번호로 업무현황 확인	계약번호, 전화번호
			업무 기록 등록	업무 기록 등록 (팀원용)	작업 목록ID, 팀ID, 작업자명, 계약 목록ID
7		WORK_RECORD	업무 전체 기록 확인	업무 전체 기록 확인 (팀장용)	
			팀별 업무 기록 확인	팀별 업무 기록 확인(팀원용)	팀ID
			계약 목록 별 기록 확인	계약목록ID로 기록 확인	계약 목록 ID
			날짜 범위 별로 기록 확인	날짜 범위 별로 기록 확인	찾을 범위(시작날짜 / 종료날짜) 날짜

ERD



역할분담

최병권 - 기업용(로그인, menu)

김소윤 - 고객용(로그인, 진행상황)

이소우 - 기업용(업무현황)

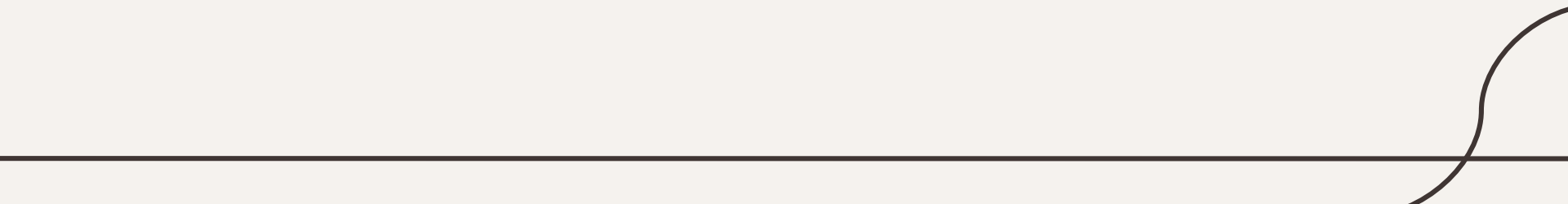
강유준 - 기업용(업무현황 상세보기)

김성중 - 기업용(업무 등록)



02

사용 기술



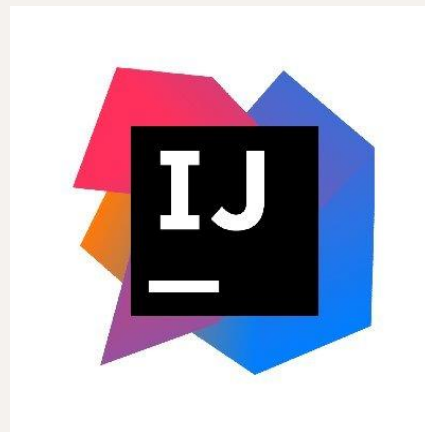
2. 사용 기술



JAVA 17



Flutter

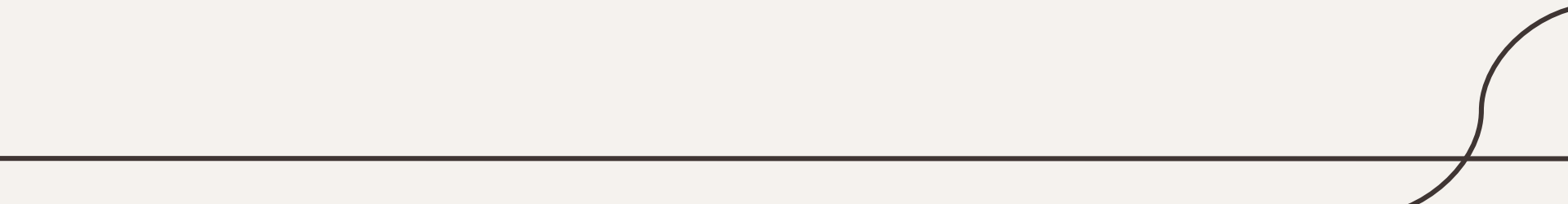


IntelliJ



03

Back-end



Entity

Member

(username) ID와 (password) PW 입력

MemberGroup,
로그인 할 때 멤버 권한 확인
isUse
등록된 회원 여부(활성화 여부) 확인
Team
Member Entity와 연계를 위해 추가

Builder 패턴을 사용해 번거로운 작업을 간소화

```
1 usage
public void putMember(MemberUpdateRequest request) { this.isUse = request.getIsUse(); }

5 usages
public static class MemberBuilder implements CommonModelBuilder<Member> {
    2 usages
    private final String username;
    2 usages
    private final String password;
    2 usages
    private final MemberGroup memberGroup;
    2 usages
    private Team team;
    2 usages
    private final Boolean isUse;

    3 usages
    public MemberBuilder(MemberRequest request) {
        this.username = request.getUsername();
        this.password = request.getPassword();
        this.memberGroup = request.getMemberGroup();
        this.isUse = true;
    }

    public MemberBuilder setTeam(Team team) {
        this.team = team;

        return this;
    }

    @Override
    public Member build() { return new Member( builder: this); }
}
```

Entity

Team

팀 정보 조회에 필요한 데이터들

teamName

팀명

mainWork

주업무

teamLeader

팀장

isUse

팀원여부 (활성화 여부)

createDate

생성날짜

마찬가지로 Builder 패턴을
사용해 번거로운 작업을 간소화

```
private Team(TeamBuilder builder) {
    this.teamName = builder.teamName;
    this.mainWork = builder.mainWork;
    this.teamLeader = builder.teamLeader;
    this.isUse = builder.isUse;
    this.createDate = builder.createDate;
}

1 usage
public void putTeam(TeamUpdateRequest request) { this.isUse = request.getIsUse(); }

2 usages
public static class TeamBuilder implements CommonModelBuilder<Team> {
    2 usages
    private final String teamName;
    2 usages
    private final String mainWork;
    2 usages
    private final String teamLeader;
    2 usages
    private final Boolean isUse;
    2 usages
    private final LocalDateTime createDate;

    1 usage
    public TeamBuilder(TeamRequest request) {
        this.teamName = request.getTeamName();
        this.mainWork = request.getMainWork();
        this.teamLeader = request.getTeamLeader();
        this.isUse = request.getIsUse();
        this.createDate = request.getCreateDate();
    }

    @Override
    public Team build() { return new Team( builder, this); }
}
```

Service

LoginService

passwordEncoder로 비밀번호 암호화 입력

jwtTokenProvider를 생성해 로그인 token키 발급

비밀번호 오류 시 회원정보가 없음을 swagger 화면에 표시

```
@Service
@RequiredArgsConstructor
public class LoginService {

    1 usage
    private final MemberRepository memberRepository;
    1 usage
    private final PasswordEncoder passwordEncoder;
    1 usage
    private final JwtTokenProvider jwtTokenProvider;

    2 usages
    public LoginResponse doLogin(MemberGroup memberGroup, LoginRequest loginRequest, String loginType) {
        Member member = memberRepository.findByUsername(loginRequest.getUsername()).orElseThrow(CMissingDataException::new);
        // 회원정보가 없습니다. 던지기

        if (!member.getIsUse()) throw new CMissingDataException(); // 회원정보가 없습니다. 던지기
        if (!member.getMemberGroup().equals(memberGroup)) throw new CMissingDataException(); // 회원정보가 없습니다. 던지기
        if (!passwordEncoder.matches(loginRequest.getPassword(), member.getPassword())) throw new CMissingDataException();
        // 회원정보가 없습니다. 던지기

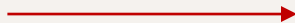
        String token = jwtTokenProvider.createToken(member.getUsername(), member.getMemberGroup().toString(), loginType);

        return new LoginResponse.LoginResponseBuilder(token, member.getUsername()).build();
    }
}
```

Service

MemberService

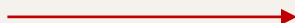
memberRepository와
teamRepository 데이터가 필수



getMembers로 로그인 가능
한 모든 회원 조회



putMember로 활성화여부만
수정 가능



```
@Service
@RequiredArgsConstructor
public class MemberService {

    4 usages
    private final MemberRepository memberRepository;
    1 usage
    private final TeamRepository teamRepository;

    no usages
    public void setMember(MemberRequest request, Long teamId) {
        Team team = teamRepository.findById(teamId).orElseThrow();
        Member addData = new Member.MemberBuilder(request).setTeam(team).build();

        memberRepository.save(addData);
    }

    1 usage
    public ListResult<MemberDetails> getMembers() {
        List<Member> originList = memberRepository.findAll();
        List<MemberDetails> result = new ArrayList<>();
        originList.forEach(item -> result.add(new MemberDetails.MemberDetailsBuilder(item).build()));
        return ListConvertService.settingResult(result);
    }

    1 usage
    public void putMember(long id, MemberUpdateRequest request) {
        Member originData = memberRepository.findById(id).orElseThrow(ClassCastException::new);
        originData.putMember(request);

        memberRepository.save(originData);
    }
}
```

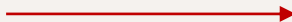
Service

TeamService

teamRepository 데이터 필수



마찬가지로 getTeams로
로그인 가능한 모든 팀원 조회



마찬가지로 putTeam으로
활성화여부만 수정 가능



```
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

2 usages
@Service
@RequiredArgsConstructor
public class TeamService {

4 usages
    private final TeamRepository teamRepository;

    public void setTeam(TeamRequest request) {
        Team addData = new Team.TeamBuilder(request).build();

        teamRepository.save(addData);
    }

1 usage
    public ListResult<TeamItem> getTeams() {
        List<Team> originList = teamRepository.findAll();
        List<TeamItem> result = new ArrayList<>();
        originList.forEach(item -> result.add(new TeamItem.TeamItemBuilder(item).build()));
        return ListConvertService.settingResult(result);
    }

1 usage
    public void putTeam(long id, TeamUpdateRequest request) {
        Team originData = teamRepository.findById(id).orElseThrow(ClassCastException::new);
        originData.putTeam(request);

        teamRepository.save(originData);
    }
}
```


Service

MemberDataService (1)

관리자 ID, PW 생성

Member 아이디 형식과 비밀번호
일치 여부를 확인하여 아이디 생성

```
@Service
@RequiredArgsConstructor
public class MemberDataService {

    5 usages
    private final MemberRepository memberRepository;
    1 usage
    private final TeamRepository teamRepository;
    4 usages
    private final PasswordEncoder passwordEncoder;

    1 usage
    public void setFirstMember() {
        String username = "superadmin";
        String password = "abcd1234";
        boolean isSuperAdmin = isNewUsername(username);

        if (isSuperAdmin) {
            MemberRequest createRequest = new MemberRequest();
            createRequest.setUsername(username);
            createRequest.setPassword(password);
            createRequest.setPasswordRe(password);
            createRequest.setMemberGroup(MemberGroup.ROLE_ADMIN);

            setMember(createRequest);
        }
    }

    public void setMember(MemberRequest request) {
        if (!CommonCheck.checkUsername(request.getUsername()))
            throw new CNotValidUsernameTypeException(); // 유효한 아이디 형식이 아닙니다
        if (!request.getPassword().equals(request.getPasswordRe()))
            throw new CWrongPasswordException(); // 비밀번호가 일치하지 않습니다
        if (!isNewUsername(request.getUsername())) throw new CUsernameSignInFailedException(); // 아이디 생성에 실패하였습니다

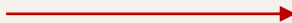
        request.setPassword(passwordEncoder.encode(request.getPassword()));

        Member member = new Member.MemberBuilder(request).build();
        memberRepository.save(member);
    }
}
```

Service

MemberDataService (2)

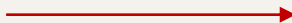
Team 정보에 필요한 Member의
ID와 PW 그리고 중복ID 여부 확인



중복된 ID 카운트 하기



비밀번호를 변경할 때 필요한 3가지
(현재 비번, 바꿀 비번, 바꿀 비번 확인)



```
public void setMember(MemberRequest request, long teamId) {
    if (!CommonCheck.checkUsername(request.getUsername()))
        throw new CLongUsernameException(); // 아이디는 20글자 이하로 생성이 가능합니다
    if (!request.getPassword().equals(request.getPasswordRe()))
        throw new CWrongPasswordException(); // 비밀번호가 일치하지 않습니다
    if (!isNewUsername(request.getUsername())) throw new CDuplicationUsernameException(); // 중복된 아이디가 존재합니다

    Team team = teamRepository.findById(teamId).orElseThrow(CMissingDataException::new);

    request.setPassword(passwordEncoder.encode(request.getPassword()));

    Member member = new Member.MemberBuilder(request).setTeam(team).build();
    memberRepository.save(member);
}

3 usages
private boolean isNewUsername(String username) {
    long dupCount = memberRepository.countByUsername(username);
    return dupCount <= 0;
}

1 usage
public void putPassword(long memberId, PasswordChangeRequest changeRequest, boolean isAdmin) {
    Member member = memberRepository.findById(memberId).orElseThrow(CMissingDataException::new);

    if (!isAdmin && !passwordEncoder.matches(changeRequest.getChangePassword(), member.getPassword()))
        throw new CWrongPasswordException(); // 비밀번호가 일치하지 않습니다

    String passwordResult = passwordEncoder.encode(changeRequest.getChangePassword());
    member.putPassword(passwordResult);
    memberRepository.save(member);
}
```

Service

CustomUserDetailsService

MemberRepository 데이터 필수



유저ID 순으로 정렬 후 정보를 확인하여 유저가 존재하지 않으면 오류 던지기



```
@Service
@RequiredArgsConstructor
public class CustomUserDetailsService implements UserDetailsService {

    1 usage
    private final MemberRepository memberRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        return memberRepository.findByUsername(username).orElseThrow(CMissingDataException::new);
    }
}
```

Process (entity)

누락 방지 및 유지보수를 위해 빌더 패턴을 사용했습니다.

memo의 경우 필수값이 아니기 때문에 필요할 경우에만 setMemo를 호출하여 값을 입력할 수 있도록 했습니다.

```
public static class Builder implements CommonModelBuilder<Process> {
    2 usages
    private final Team team;
    2 usages
    private final String processName;
    2 usages
    private final Integer processOrder;
    2 usages
    private final PackageInfo packageInfo;
    2 usages
    private final Boolean isUse;
    2 usages
    private final LocalDateTime createDate;
    2 usages
    private String memo;

    1 usage   강유준
    public Builder(Team team, PackageInfo packageInfo, ProcessCreateRequest request) {
        this.team = team;
        this.processName = request.getProcessName();
        this.processOrder = request.getProcessOrder();
        this.packageInfo = packageInfo;
        this.isUse = true;
        this.createDate = LocalDateTime.now();
    }

    1 usage   강유준
    public Builder setMemo(String memo) {
        this.memo = memo;

        return this;
    }
}
```

ProcessService

setProcess

새로운 공정을 등록

getAllProcess

모든 공정을 리스트 형태로 불러오기

getUseProcess

사용중인 공정을 리스트 형태로 불러오기

getUsePackageProcess

특정 패키지의 사용중인 공정을 불러오기

```
public void setProcess(ProcessCreateRequest request) {
    Team team = teamRepository.getById(request.getTeamId());
    PackageInfo packageInfo = packageInfoRepository.getById(request.getPackageInfoId());

    Process addData = new Process.Builder(team, packageInfo, request).setMemo(request.getMemo()).build();
    processRepository.save(addData);
}

1 usage  강유준
public ListResult<ProcessItem> getAllProcess() {
    List<Process> originList = processRepository.findAll();

    if (originList.isEmpty()) throw new CNotRegistrationProcessInfoException();
    List<ProcessItem> result = new LinkedList<>();

    originList.forEach(item -> result.add(new ProcessItem.Builder(item).build()));

    return ListConvertService.settingResult(result);
}

1 usage  강유준
public ListResult<ProcessItem> getUseProcess() {
    List<Process> originList = processRepository.findAllByIsUse(true);

    if (originList.isEmpty()) throw new CDontFindProcessInfoException();
    List<ProcessItem> result = new LinkedList<>();

    originList.forEach(item -> result.add(new ProcessItem.Builder(item).build()));

    return ListConvertService.settingResult(result);
}

1 usage  강유준
public ListResult<ProcessItem> getUsePackageProcess(long packageInfoId) {
    PackageInfo packageInfo = packageInfoRepository.getById(packageInfoId);

    List<Process> originList = processRepository.findAllByIsUseAndPackageInfo(isUse: true, packageInfo);
}
```

ProcessService

setPackageInfo
새로운 패키지를 등록

getPackageInfos
모든 패키지를 리스트 형태로 불러오기

getUsePackageInfos
사용중인 패키지를 리스트 형태로 불러오기

putPackageInfo
패키지 정보 수정하기

```
public void setPackageInfo(PackageInfoCreateRequest request) {
    PackageInfo addData = new PackageInfo.PackageInfoBuilder(request).setMemo(request.getMemo()).build();
    packageInfoRepository.save(addData);
}

1 usage  👤 강유준
public ListResult<PackageInfoItem> getPackageInfos() {
    List<PackageInfo> originList = packageInfoRepository.findAll();

    if (originList.isEmpty()) throw new CNotRegistrationPackageInfoException();
    List<PackageInfoItem> result = new LinkedList<>();

    originList.forEach(item-> result.add(new PackageInfoItem.PackageInfoItemBuilder(item).build()));

    return ListConvertService.settingResult(result);
}

1 usage  👤 강유준
public ListResult<PackageInfoItem> getUsePackageInfos() {
    List<PackageInfo> originList = packageInfoRepository.findByIsUse(true);

    if (originList.isEmpty()) throw new CDontFindNotUsingPackageInfoException();
    List<PackageInfoItem> result = new LinkedList<>();

    originList.forEach(item-> result.add(new PackageInfoItem.PackageInfoItemBuilder(item).build()));

    return ListConvertService.settingResult(result);
}

1 usage  👤 강유준
public void putPackageInfo(long id, PackageInfoCreateRequest request) {
    PackageInfo originData = packageInfoRepository.findById(id).orElseThrow(CNotRegistrationPackageInfoException::new);
    originData.putPackageInfo(request);

    packageInfoRepository.save(originData);
}
```

Part 3

Back-end

김유준

```
public ListResult<WorkStatusItem> getList() {  
    List<WorkStatus> originList = workStatusRepository.findAll();  
  
    List<WorkStatusItem> result = new LinkedList<>();  
  
    if (originList.isEmpty()) throw new CNotRegistrationWorkStatusException();  
    originList.forEach(item -> result.add(new WorkStatusItem.WorkStatusItemBuilder(item).build()));  
  
    return ListConvertService.settingResult(result);  
}
```

Work Status Service(업무 현황 서비스)
Get 기능

getList :
등록된 업무 현황 전체 리스트를 가져오는 메서드

usage 김유준

```
public ListResult<WorkStatusItem> getListByApprover(String approver) {  
    List<WorkStatus> originList = workStatusRepository.findByApprover(approver);  
  
    List<WorkStatusItem> result = new LinkedList<>();  
  
    if (originList.isEmpty()) throw new CDontFindWorkStatusException();  
    originList.forEach(item -> result.add(new WorkStatusItem.WorkStatusItemBuilder(item).build()));  
  
    return ListConvertService.settingResult(result);  
}
```

getListByApprover :
등록된 업무 현황 중 승인자 기준으로 리스트를
가져오는 메서드

usage 김유준

```
public ListResult<WorkStatusItem> getListByTeam(long teamId) {  
    List<WorkStatus> originList = workStatusRepository.findByTeam_Id(teamId);  
  
    List<WorkStatusItem> result = new LinkedList<>();  
  
    if (originList.isEmpty()) throw new CDontFindWorkStatusException();  
    originList.forEach(item -> result.add(new WorkStatusItem.WorkStatusItemBuilder(item).build()));  
  
    return ListConvertService.settingResult(result);  
}
```

getListByTeam :
등록된 업무 현황 중 팀 기준으로 리스트를 가져
오는 메서드

1 usage 강유준

```
public void setContract(ContractCreateRequest request, Long packageInfoId) {  
    PackageInfo packageInfo = packageInfoRepository.findById(packageInfoId).orElseThrow(CMissingDataException::new);  
    List<Process> processList = processRepository.findByPackageInfoOrderByProcessOrderAsc(packageInfo);  
    Process process = processList.get(0);  
  
    Contract addData1 = new Contract.ContractBuilder(request, packageInfo, process).build();  
    contractRepository.save(addData1);  
  
    WorkStatus addData2 = new WorkStatus.WorkStatusBuilder(addData1).build();  
    workStatusRepository.save(addData2);  
}
```

공정 리스트를 가져오고 공정 순서(숫자)대로 정렬 한다.

그 후 리스트의 가장 첫 번째 공정 데이터를 process라는 이름으로 생성.

계약서 작성과 동시에 업무 현황에도 등록된다.

Contract Service(계약 관리 서비스)
Post 기능


```
l usage  👤 김유준
public ListResult<ContractItem> getContractList() {
    List<Contract> originList = contractRepository.findAll();

    List<ContractItem> result = new LinkedList<>();

    if (originList.isEmpty()) throw new CNotRegistrationContractException();
    originList.forEach(item -> result.add(new ContractItem.ContractItemBuilder(item).build()));

    return ListConvertService.settingResult(result);
}
```

```
l usage  👤 김유준
public ListResult<ContractItem> getBetweenDateList(LocalDate startDate, LocalDate endDate) {
    List<Contract> originList = contractRepository.findByContractDateBetween(startDate, endDate);

    List<ContractItem> result = new LinkedList<>();

    if (originList.isEmpty()) throw new CDontFindContractException();
    originList.forEach(item -> result.add(new ContractItem.ContractItemBuilder(item).build()));

    return ListConvertService.settingResult(result);
}
```

```
l usage  👤 김유준
public ContractResponse getContract(String contractNumber) {
    Contract originData = contractRepository.findByContractNumber(contractNumber);

    if (originData == null) throw new CDontFindContractException();
    return new ContractResponse.ContractResponseBuilder(originData).build();
}
```

Contract Service(계약 관리 서비스) Get 기능

GetContractList :
등록된 계약 전체 리스트를 가져오는 메서드

getBetweenDateList :
등록된 업무 현황 중 특정 날짜(시작 일자와
종료 날짜를 받는다.) 사이에 등록된 리스트
를 가져오는 메서드

getContract :
계약 번호로 계약 정보를 가져오는 메서드

Part 3

Back-end

```
usage  김유준
public ListResult<ContractItem> getProgressContractList() {
    List<Contract> originList = contractRepository.findByEndWorkDateIsNull();

    List<ContractItem> result = new LinkedList<>();

    if (originList.isEmpty()) throw new CDontFindProgressContractException();
    originList.forEach(item -> result.add(new ContractItem.ContractItemBuilder(item).build()));

    return ListConvertService.settingResult(result);
}
```

```
usage  김유준
public ListResult<ContractItem> getFinishContractList() {
    List<Contract> originList = contractRepository.findByEndWorkDateIsNotNull();

    List<ContractItem> result = new LinkedList<>();

    if (originList.isEmpty()) throw new CDontFindFinishContractException();
    originList.forEach(item -> result.add(new ContractItem.ContractItemBuilder(item).build()));

    return ListConvertService.settingResult(result);
}
```

Contract Service(계약 관리 서비스) Get 기능

getProgressContractList :
등록된 계약 중 현재 진행중인 리스트를 가져
오는 메서드

종료날짜가 null값이라면 아직 진행 중
Null값이 아니라면 종료된 계약

getFinishContractList :
등록된 계약 중 종료된 리스트를 가져오는
메서드

3. Back-end

WorkRecordService.java

setWorkRecord

업무 기록 등록하기

필요한 값을 요청 하였으나 받지 못했을 때
throw 처리를 한다

getWorkRecordList

전체 업무 기록을 리스트 형태로 불러오기

getListByTeam

팀별 업무기록을 리스트 형태로 불러오기

```
1 usage
public void setWorkRecord(WorkRecordCreateRequest createRequest, long processId, long teamId, long contractId) {
    Process process = processRepository.findById(processId).orElseThrow(CMissingDataException::new);
    Team team = teamRepository.findById(teamId).orElseThrow(CMissingDataException::new);
    Contract contract = contractRepository.findById(contractId).orElseThrow(CMissingDataException::new);
    WorkRecord addData = new WorkRecord.WorkRecordBuilder(createRequest, process, team, contract).build();

    workRecordRepository.save(addData);
}

1 usage
public ListResult<WorkRecordItem> getWorkRecordList() {
    List<WorkRecord> originList = workRecordRepository.findAll();

    List<WorkRecordItem> result = new LinkedList<>();
    originList.forEach(item -> result.add(new WorkRecordItem.WorkRecordItemBuilder(item).build()));

    return ListConvertService.settingResult(result);
}

1 usage
public ListResult<WorkRecordItem> getListByTeam(long teamId) {
    List<WorkRecord> originList = workRecordRepository.findByTeam_Id(teamId);

    List<WorkRecordItem> result = new LinkedList<>();

    if (originList.isEmpty()) throw new CMissingDataException();
    originList.forEach(item -> result.add(new WorkRecordItem.WorkRecordItemBuilder(item).build()));

    return ListConvertService.settingResult(result);
}
```

3. Back-end

WorkRecordService.java

getListByContractId

계약 별 업무기록을 리스트 형태로 불러오기

getListByDateBetween

날짜 범위의 업무기록을 리스트 형태로 불러오기

필요한 값을 요청 하였으나 받지 못했을 때
throw 처리를 한다

```
1 usage
public ListResult<WorkRecordItem> getListByContractId(long contractId ) {
    List<WorkRecord> originList = workRecordRepository.findByContract_Id(contractId);

    List<WorkRecordItem> result = new LinkedList<>();

    if (originList.isEmpty()) throw new CMissingDataException();
    originList.forEach(item -> result.add(new WorkRecordItem.WorkRecordItemBuilder(item).build()));

    return ListConvertService.settingResult(result);
}

1 usage
public ListResult<WorkRecordItem> getListByDateBetween(LocalDate startDate, LocalDate endDate) {
    LocalDateTime startDateTime = startDate.atTime( hour: 0, minute: 0, second: 0);
    LocalDateTime endDateTime = endDate.atTime( hour: 23, minute: 59, second: 59);

    List<WorkRecord> originList = workRecordRepository.findByCreateDateBetween(startDateTime, endDateTime);

    List<WorkRecordItem> result = new LinkedList<>();

    if (originList.isEmpty()) throw new CMissingDataException();
    originList.forEach(item -> result.add(new WorkRecordItem.WorkRecordItemBuilder(item).build()));

    return ListConvertService.settingResult(result);
}
}
```

3. Back-end

DropBoxService.java

uploadFile

작업 사진 업로드

파일을 업로드 할 때 이름에 업로드 날짜와
시간을 추가 하도록 설정 하여 같은 이름의
파일을 올리더라도 이름이 중복되지 않도록
설정했다

```
import ...

2 usages
@Service
public class DropBoxService {

1 usage
    public DropBoxItem uploadFile(DropBoxDir dropBoxDir, String middleDirName, MultipartFile multipartFile) throws IllegalStateException, IOException {
        String modifier = LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyyMMddHHmmss"));
        File file = DropBox.multipartToFile(multipartFile);
        String resultFileName = modifier + "_" + file.getName();
        String fullFileName = dropBoxDir.getBasePath() + middleDirName + "/" + resultFileName;

        DropBoxItem dropBoxItem = DropBox.uploadFile(file, fullFileName);
        dropBoxItem.setFileName(resultFileName);

        return dropBoxItem;
    }
}
```

3. Back-end

WorkRecord.java

업무 기록을 등록할 때 채워야 하는 값들

Process.java, team.java를
FK 설정 하여 필요한 값을 받아온다

```
import io.swagger.annotations.ApiModelProperty;
import lombok.AccessLevel;
import lombok.Getter;
import lombok.NoArgsConstructor;
import shop.connect.all.api.work.model.WorkRecordCreateRequest;
import shop.connect.all.common.interfaces.CommonModelBuilder;

import javax.persistence.*;
import java.time.LocalDateTime;

17 usages
@Entity
@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
public class WorkRecord {
    @ApiModelProperty(notes = "시퀀스")
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    1 usage
    @ApiModelProperty(notes = "공정 관리 시퀀스")
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(nullable = false, name = "processId")
    private Process process;

    1 usage
    @ApiModelProperty(notes = "팀 시퀀스")
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(nullable = false, name = "teamId")
    private Team team;
```

3. Back-end

WorkRecord.java

contract.java 를

FK 설정 하여 필요한 값을 받아온다

```
1 usage
  @ApiModelProperty(notes = "사진 이미지 주소")
  @Column(length = 200)
  private String imageAddress;

1 usage
  @ApiModelProperty(notes = "메모")
  @Column(columnDefinition = "TEXT")
  private String memo;

1 usage
  @ApiModelProperty(notes = "작업자명")
  @Column(nullable = false, length = 20)
  private String workerName;

1 usage
  @ApiModelProperty(notes = "계약서 관리 시퀀스")
  @ManyToOne(fetch = FetchType.LAZY)
  @JoinColumn(nullable = false, name = "contractId")
  private Contract contract;

1 usage
  @ApiModelProperty(notes = "등록일시")
  @Column(nullable = false)
  private LocalDateTime createDate;

1 usage
  private WorkRecord(WorkRecordBuilder builder) {
    this.process = builder.process;
    this.team = builder.team;
    this.imageAddress = builder.imageAddress;
```

3. Back-end

WorkRecord.java

작업 효율을 위해

Getter, Setter 를 사용하지 않고
Builder Pattern 사용

```
1 usage
private WorkRecord(WorkRecordBuilder builder) {
    this.process = builder.process;
    this.team = builder.team;
    this.imageAddress = builder.imageAddress;
    this.memo = builder.memo;
    this.workerName = builder.workerName;
    this.contract = builder.contract;
    this.createDate = builder.createDate;
}
```

```
2 usages
public static class WorkRecordBuilder implements CommonModelBuilder<WorkRecord> {
```

```
    2 usages
    private final Process process;
    2 usages
    private final Team team;
    2 usages
    private final String imageAddress;
    2 usages
    private final String memo;
    2 usages
    private final String workerName;
    2 usages
    private final Contract contract;
    2 usages
    private final LocalDateTime createDate;
```

```
1 usage
```

```
public WorkRecordBuilder(WorkRecordCreateRequest request, Process process, Team team, Contract contract) {
```


3. Back-end

WorkRecord.java

작업 효율을 위해

Getter, Setter 를 사용하지 않고
Builder Pattern 사용

```
private final String imageAddress;
2 usages
private final String memo;
2 usages
private final String workerName;
2 usages
private final Contract contract;
2 usages
private final LocalDateTime createDate;

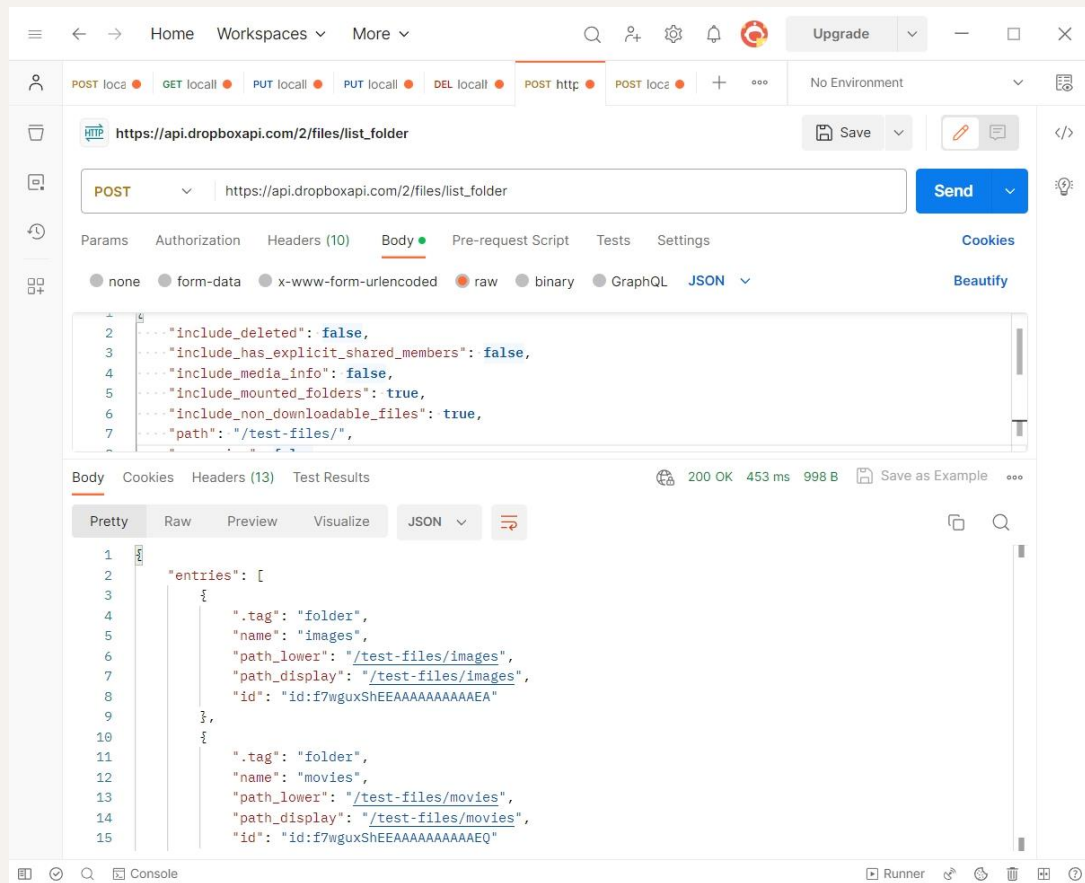
1 usage
public WorkRecordBuilder(WorkRecordCreateRequest request, Process process, Team team, Contract contract) {
    this.process = process;
    this.team = team;
    this.imageAddress = request.getImageAddress();
    this.memo = request.getMemo();
    this.workerName = request.getWorkerName();
    this.contract = contract;
    this.createDate = LocalDateTime.now();
}

@Override
public WorkRecord build() { return new WorkRecord( builder: this); }
}
```

3. Back-end

Postman

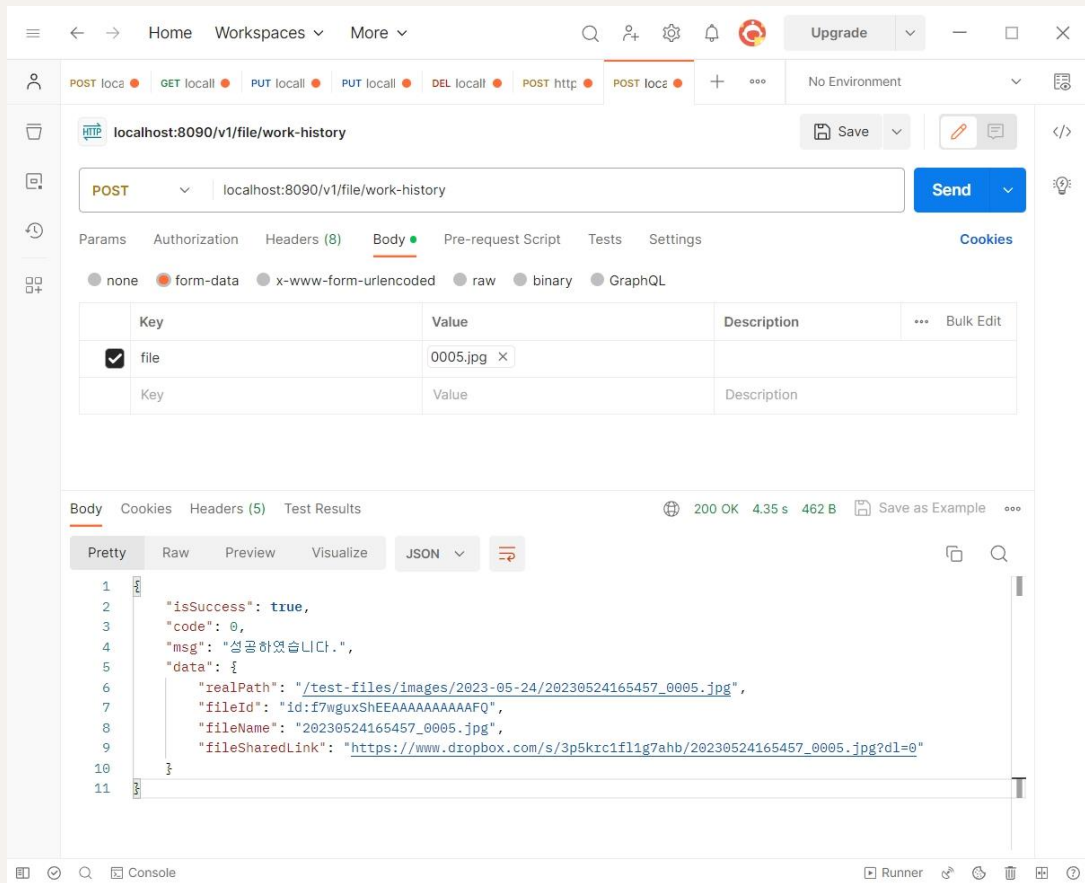
DropBox 에 생성된 폴더를 찾아가는 테스트



3. Back-end

Postman

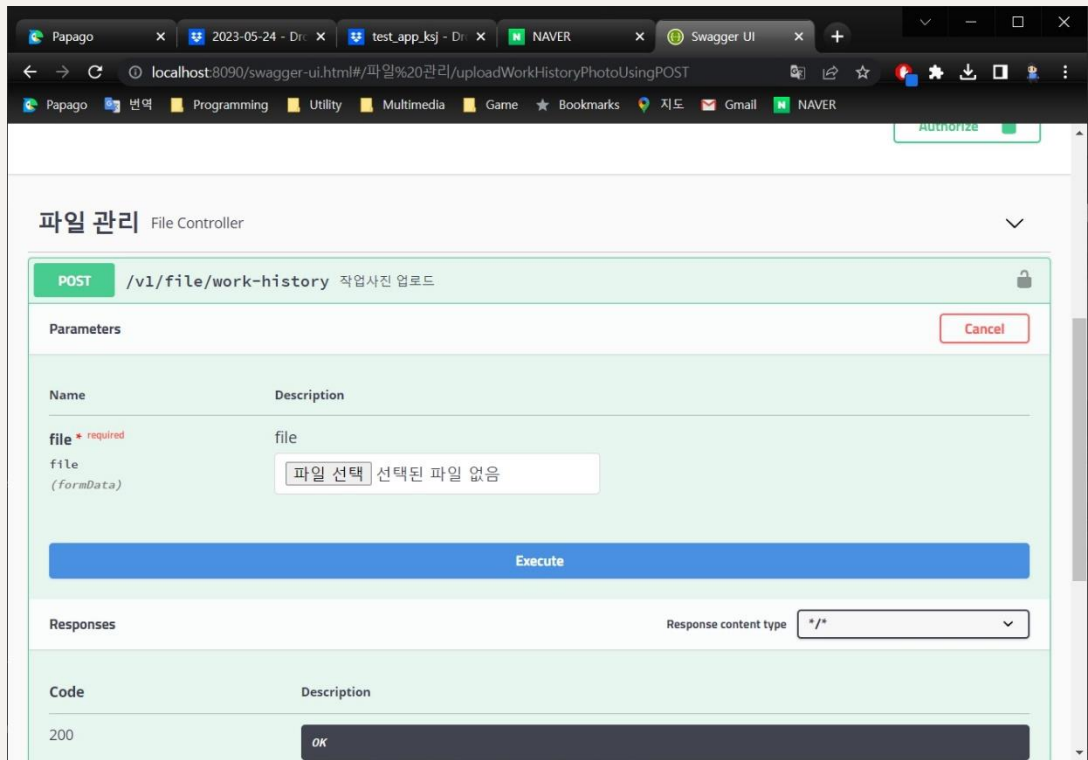
사진 업로드 테스트



3. Back-end

Swagger

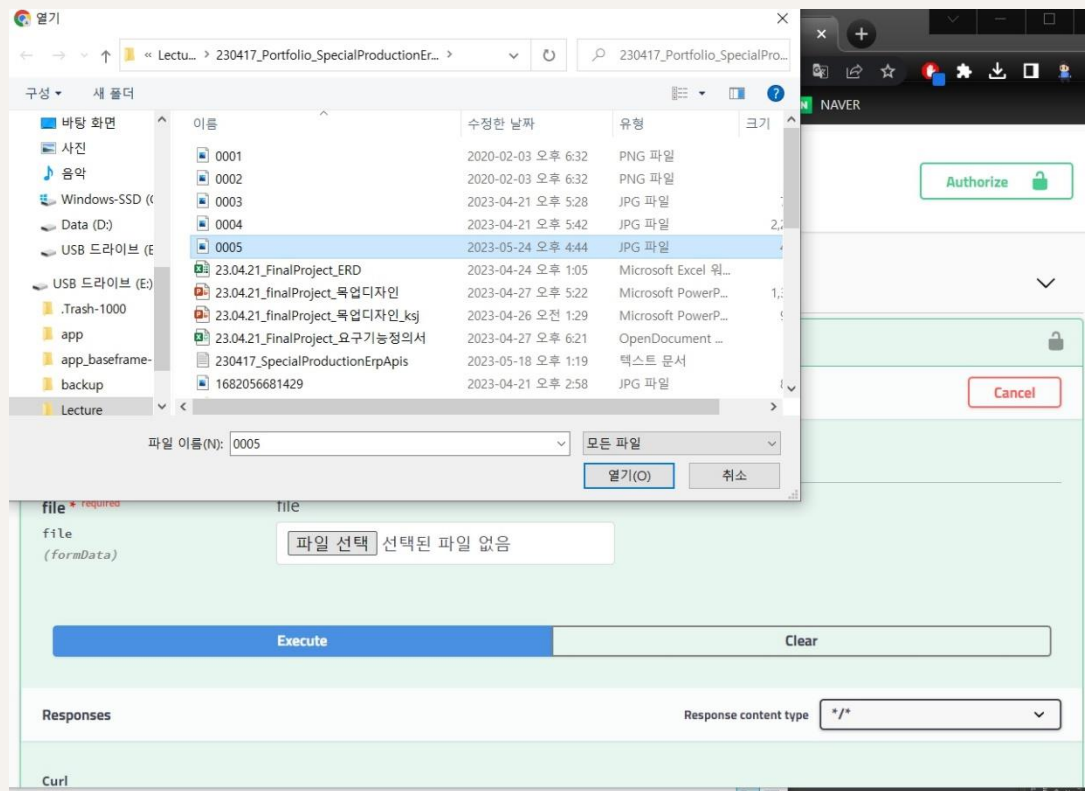
사진 업로드 테스트



3. Back-end

Swagger

업로드 할 사진 파일 선택



3. Back-end

DropBox

DropBox에 사진 업로드 성공



3. Back-end

DropBox

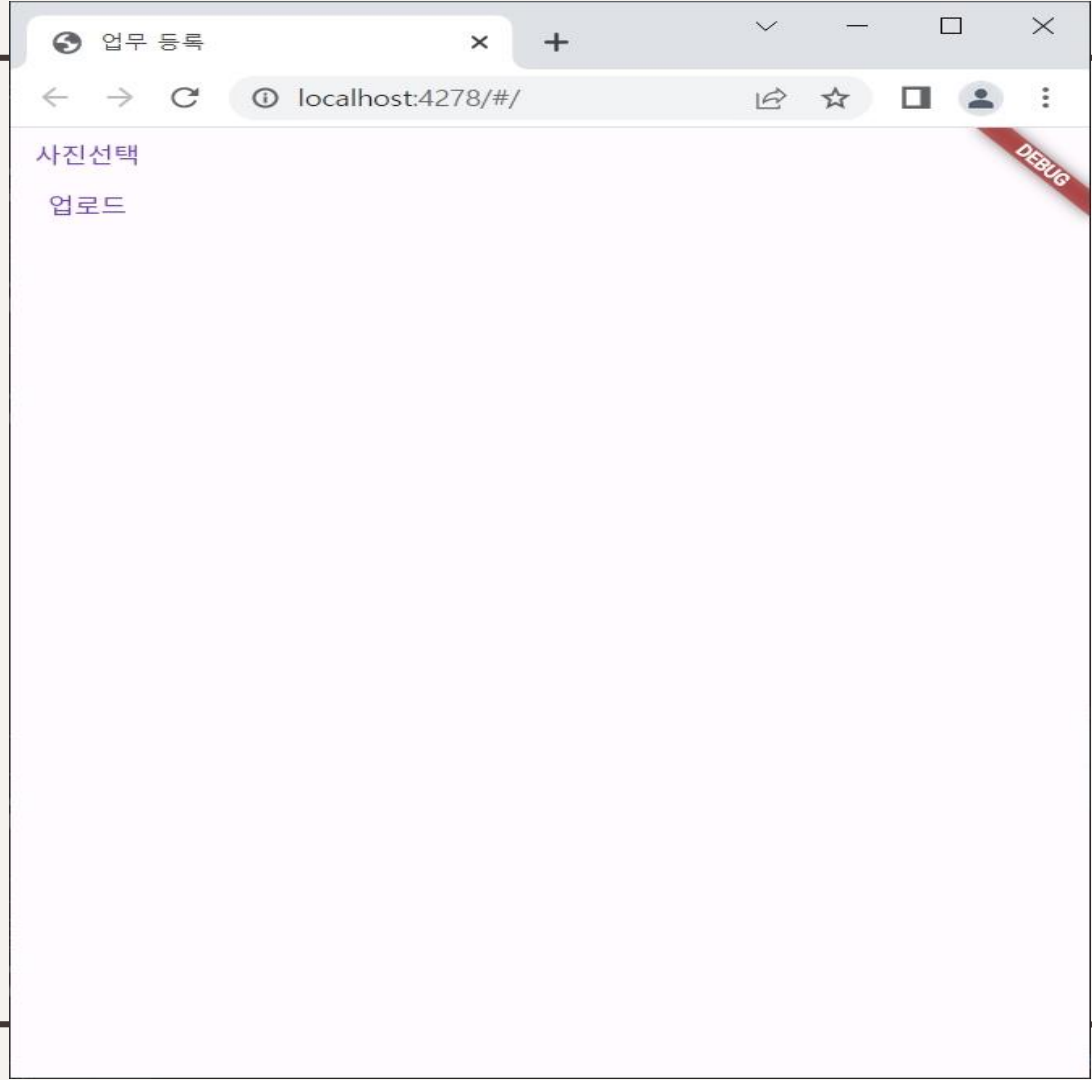
DropBox에 업로드 된 사진의 URL 주소 확인



3. Back-end

Test App

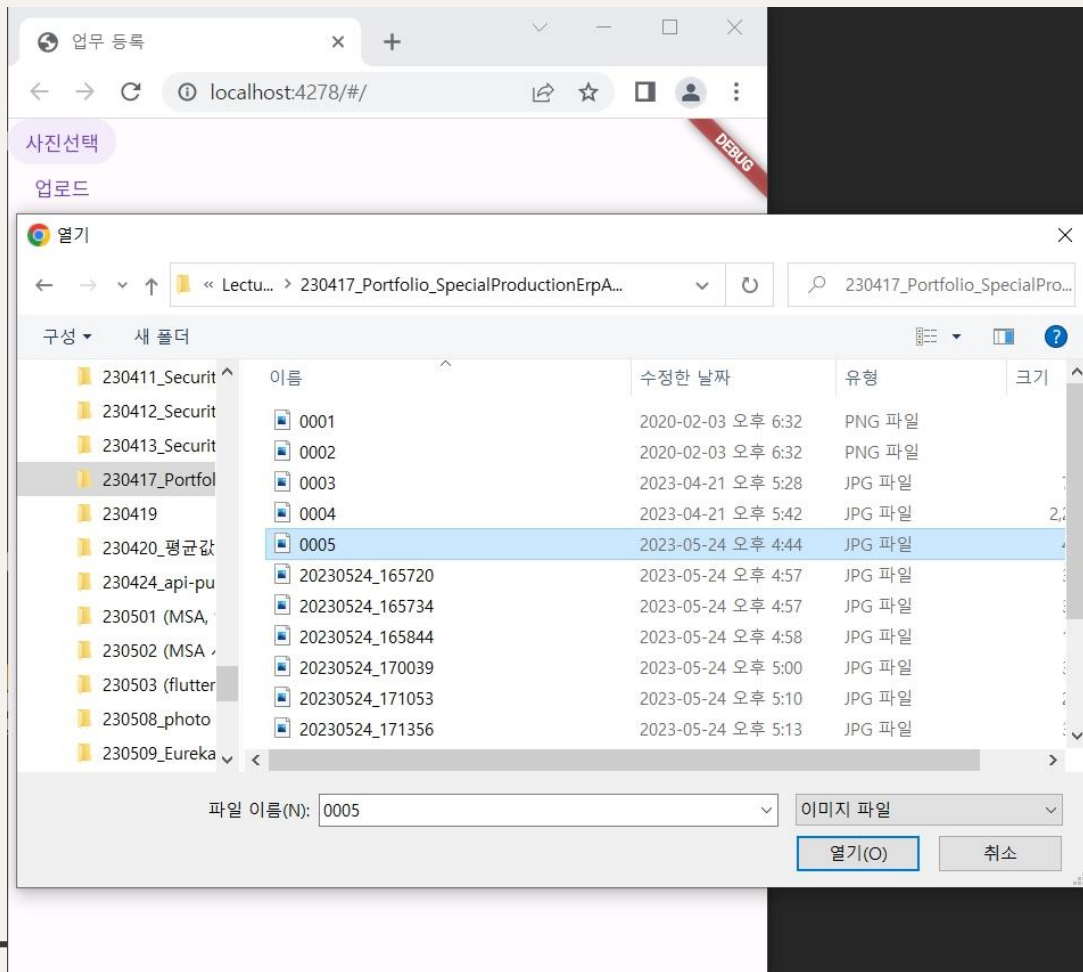
Test App 으로 사진 업로드 테스트



3. Back-end

Test App

업로드 할 사진 선택

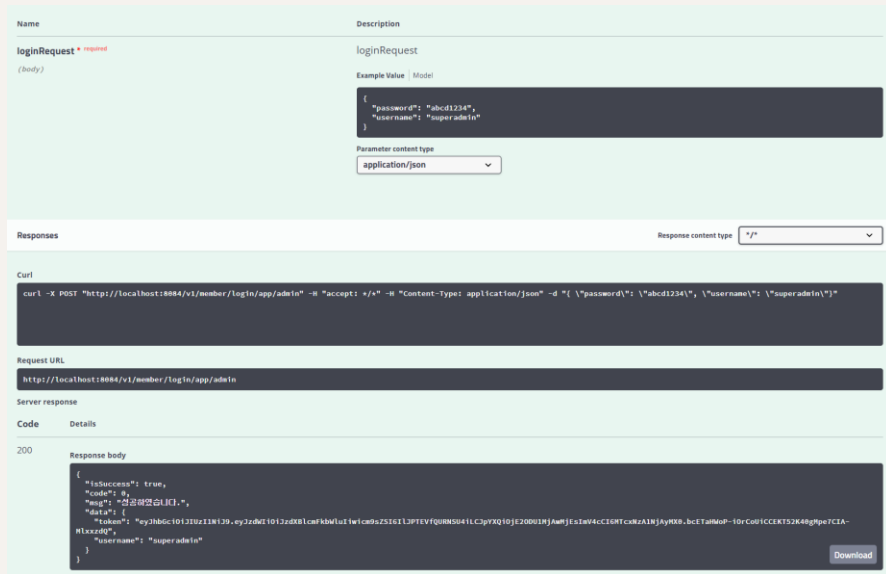




04

Front-end





현장용_업무메뉴

Flutter 로그인 후 화면

계약 번호 및 작업 정보 등록 화면

계약 현황 (완료 및 미완료 화면)

업무등록

업무현황

4. Front-end

고객용_Login

별도의 회원가입 없이 계약자 이름과 핸드폰 번호를 입력하면 일치하는 계약건을 불러옵니다.
textField에 validate를 넣어 누락된 정보가 없는지 확인하도록 했습니다.

```
child: FormBuilderTextField(  
  name: 'contractNumber',  
  autovalidateMode: AutovalidateMode.onUserInteraction,  
  validator: FormBuilderValidators.compose([  
    FormBuilderValidators.required(errorText: formErrorRequired),  
    FormBuilderValidators.minLength(5, errorText: formErrorMinLength(5)),  
    FormBuilderValidators.maxLength(20, errorText: formErrorMaxLength(20)),  
  ]),  
  keyboardType: TextInputType.number,  
  decoration: const InputDecoration(  
    labelText: '계약번호',  
    border: OutlineInputBorder(),  
    focusedBorder: OutlineInputBorder(  
      borderSide: BorderSide(  
        color: Colors.black,  
      ),  
    ),  
  ),  
),
```

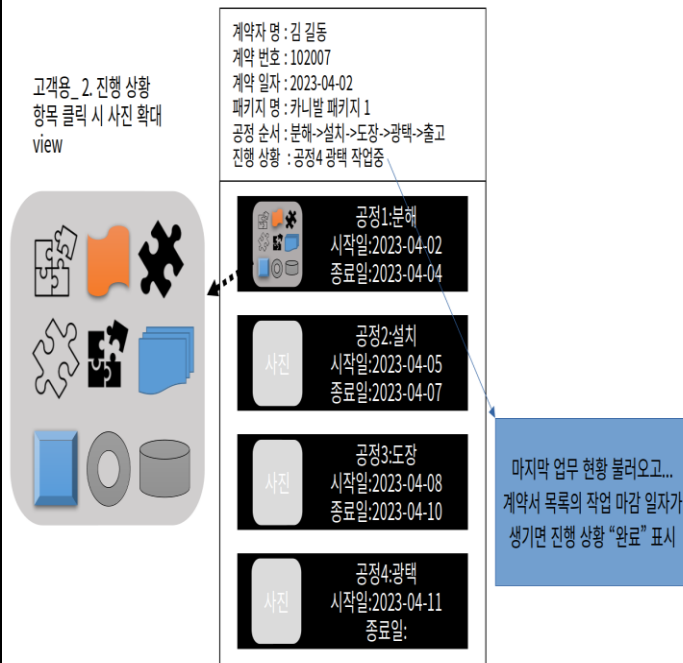
4. Front-end

고객용_View

PUSH 알림 받기 버튼을 누르면 진행상황이 갱신되었을 때 알림을 받을 수 있습니다.
공정 이미지를 클릭하면 확대됩니다.



APP 구현



목업 이미지

Part 4

Front-end

```
public void putFinishWork(WorkFinishRequest request, long workStatusId, long workRecordId) {
    WorkStatus workStatus = workStatusRepository.findById(workStatusId).orElseThrow(CNotRegistrationWorkStatusException::new);
    WorkRecord workRecord = workRecordRepository.findById(workRecordId).orElseThrow(CNotRegistrationWorkRecordException::new);
    workStatus.finishWork(request, workRecord);

    workStatusRepository.save(workStatus);

    Contract contract = workStatus.getContract();
    PackageInfo packageInfo = contract.getPackageInfo();

    List<Process> processList = processRepository.findByPackageInfoOrderByProcessOrderAsc(packageInfo);

    List<WorkStatus> workStatusList = workStatusRepository.findAllByContract(contract);
    if (processList.size() != workStatusList.size()) {
        Process process = processList.get(contract.getPresentProcess().getProcessOrder());

        contract.finishWork(process);

        contractRepository.save(contract);

        WorkStatus addData = new WorkStatus.WorkStatusBuilder(contract).build();
        workStatusRepository.save(addData);
    } else {
        contract.putEndWorkDate();
        contractRepository.save(contract);
    }
}
```


작업 종료

공정 순서를 리스트화

완료 버튼을 눌렀을 때 패키지의 필요한 공정 수만큼 업무가 존재할 경우 (공정이 전부 끝남)

```
usage 김유준
public void finishWork(WorkFinishRequest request, WorkRecord workRecord) {
    this.isComplete = true;
    this.workRecord = workRecord;
    this.approver = request.getApprover();
    this.finishDate = LocalDateTime.now();
}
```


등록일자	2023-05-01	DEMO
계약번호	a789456	
공정명	공정1	
팀명	a팀	



메모

등록일자 : 2023-05-01 / 작업자 : 김현성

선택



작업을 끝내겠습니까?

승인자 이름

취소

승인 완료

선택한

취소

완료 승인

완료 승인 서비스

작업이 끝났을 때 대표 승인자가 승인 완료 버튼을 누르면 말은 작업이 완료되고 공정 순서에 따라 다음 업무가 생성된다. (계약서 쪽에도 현재 공정 순서가 바뀐다.)

한 패키지에 들어 가는 공정의 수를 리스트로 만들고 그 리스트의 수가 업무의 수와 같아질 경우(공정이 전부 완료 되었을 때) if 문을 통해 더 이상 다음 업무를 생성하지 않게 만들었다.

APP 화면

Part 4

Front-end

```
1 usage  👤 경유준
public WorkStatusResponse getWorkStatus(long workStatusId) {
    WorkStatus workStatus = workStatusRepository.findById(workStatusId).orElseThrow(CDontFindWorkStatusException::new);

    return new WorkStatusResponse.WorkStatusResponseBuilder(workStatus).build();
}
```

업무를 가져오는 메서드(get 기능)

계약번호 : a789456	
	바닥 작업 완료
- 업무 정보 (패키지명 / 공정 공정 순서)	
A패키지	공정1 (1)
- 작업 날짜 (시작 일자 / 종료 일자)	
2023-05-01	~ 2023-05-01
- 승인자 정보	
김길동	a팀 / 김길동
뒤로가기	

APP 화면

```
{
  "isSuccess": true,
  "code": 0,
  "msg": "성공하였습니다.",
  "data": {
    "approver": "김길동",
    "createDate": "2023-05-01",
    "isCompleteText": "완료",
    "contractNumber": "a789456",
    "processName": "공정1",
    "teamName": "a팀",
    "mainWorkRecordImgName": "a12345678",
    "mainWorkRecordMemo": "바닥 작업 완료",
    "teamLeaderName": "김길동",
    "finishDate": "2023-05-01",
    "packageName": "A패키지",
    "processOrder": 1
  }
}
```


Download

Swagger 화면

Part 4

Front-end

등록일자 계약번호 공정명 팀명	2023-05-01 a789456 공정1 a팀	DEMO
---------------------------	------------------------------------	------

 	메모 등록일자 : 2023-05-01 / 작업자 : 김현성	선택
	메모2 등록일자 : 2023-05-01 / 작업자 : 김호수	선택

선택한 업무 기록

선택한 업무 기록

메모2

선택 시

```
{
  "isSuccess": true,
  "code": 0,
  "msg": "성공하였습니다.",
  "data": {
    "createDate": "2023-05-01",
    "contractNumber": "a789456",
    "processName": "공정1",
    "teamName": "a팀",
    "contractId": 1,
    "workRecordItems": [
      {
        "workRecordId": 2,
        "contractNum": "a789456",
        "contractDate": "2023-04-15",
        "packageName": "A패키지",
        "processName": "공정2",
        "teamName": "a팀",
        "imageAddress": "safesdf",
        "memo": "메모",
        "createDate": "2023-05-01",
        "workerName": "김현성"
      },
      {
        "workRecordId": 3,
        "contractNum": "a789456",
        "contractDate": "2023-04-15",
        "packageName": "A패키지",
        "processName": "공정2",
        "teamName": "a팀",
        "imageAddress": "asdfsdf",
        "memo": "메모2",
        "createDate": "2023-05-01",
        "workerName": "김호수"
      }
    ]
  }
}
```

Swagger 화면

```
usage  강유준
public WorkStatusMiniResponse getWorkStatusMini(long workStatusId) {
    WorkStatus workStatus = workStatusRepository.findById(workStatusId).orElseThrow(CDontFindWorkStatusException::new);

    List<WorkRecord> workRecords = workRecordRepository.findByContract_IdAndProcess_Id(
        workStatus.getContract().getId(), workStatus.getContract().getPresentProcess().getId());

    List<WorkRecordItem> workRecordItems = new LinkedList<>();
    workRecords.forEach(item -> workRecordItems.add(new WorkRecordItem.WorkRecordItemBuilder(item).build()));

    return new WorkStatusMiniResponse.WorkStatusMiniResponseBuilder(workStatus, workRecordItems).build();
}

private String createDate;
private String contractNumber;
private String processName;
private String teamName;
private Long contractId;
private List<WorkRecordItem> workRecordItems;
```

Model

업무 기록 리스트 가져오기

4. Front-end

기업용_2-1 현장 part 업무 등록

사진 아이콘 클릭 시 사진 등록 페이지로 이동

등록 버튼 누르면 등록하였습니다
팝업 창 열림

취소 버튼 누르면 업무 현황 페이지로 되돌아감

등록하였습니다

계약 번호 : 102007
작업 팀명 : A 팀
공정 이름 : 광택
작업자 명 : 김 땡땡
진행 상황 : 공정 3
도장 작업 완료

사진 등록



취소

등록

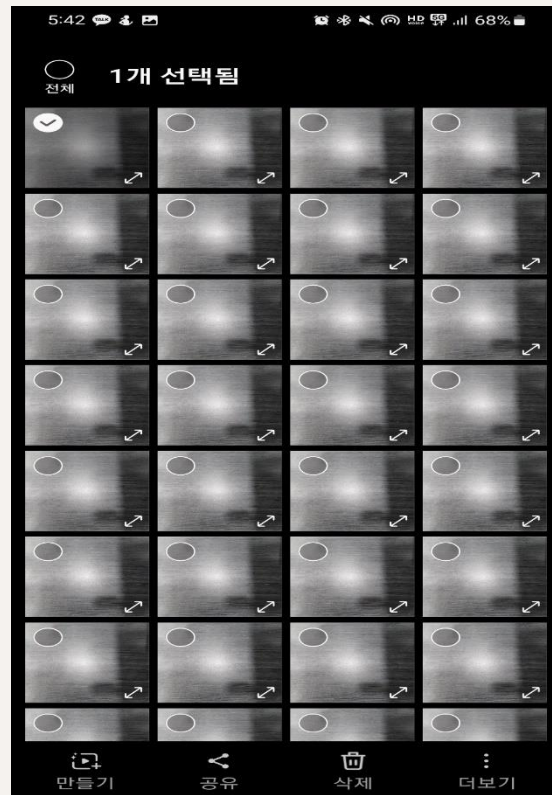
4. Front-end

기업용_2-1 현장 part 사진 등록

사진 아이콘 클릭 후 등록 버튼 누르면
사진이 등록 되었습니다 팝업 창 열림

취소 버튼 누르면 업무 등록 화면으로
되돌아감

사진이 등록
되었습니다



취소

등록