가계부 포트폴리오_ㅇㅇㅇ

Contact

- **Phone :** 010-0000-0000
- **Email :** abc@naver.com

[포트폴리오 목차]

클릭하시면 바로 넘어갑니다.

[프로젝트 개요]

항목	내용
프로젝트 소개	가계부와 일기를 함께 작성하는 웹어플리케이션
개발 인원	총 5명 (프론트엔드 2명 + 백엔드 3명)
담당 역할	팀장, API 개발(수입, 보고서, 달력, 엑셀 출력), CI/CD 구성 및 RDS DB 세팅
개발 기간	총 42일 (2022-00-00 ~ 2022-00-00)
성과 및 결과	기한 내 목표한 기능 구현 완료, CI/CD 및 실제 서버 배포 완료.

⚡ 프로젝트 깃헙 바로가기

⚡ 배포된 웹 어플리케이션 바로가기

→ 테스트용 ID / PW : abc@kakao.com / password

[프로젝트 설명]

1) 사용 기술

Frontend `CRA` `React` `Material Design` `React Query` `Styled Components` `Zustand` `React Hook Form Design` `Chart.js` `React Router`

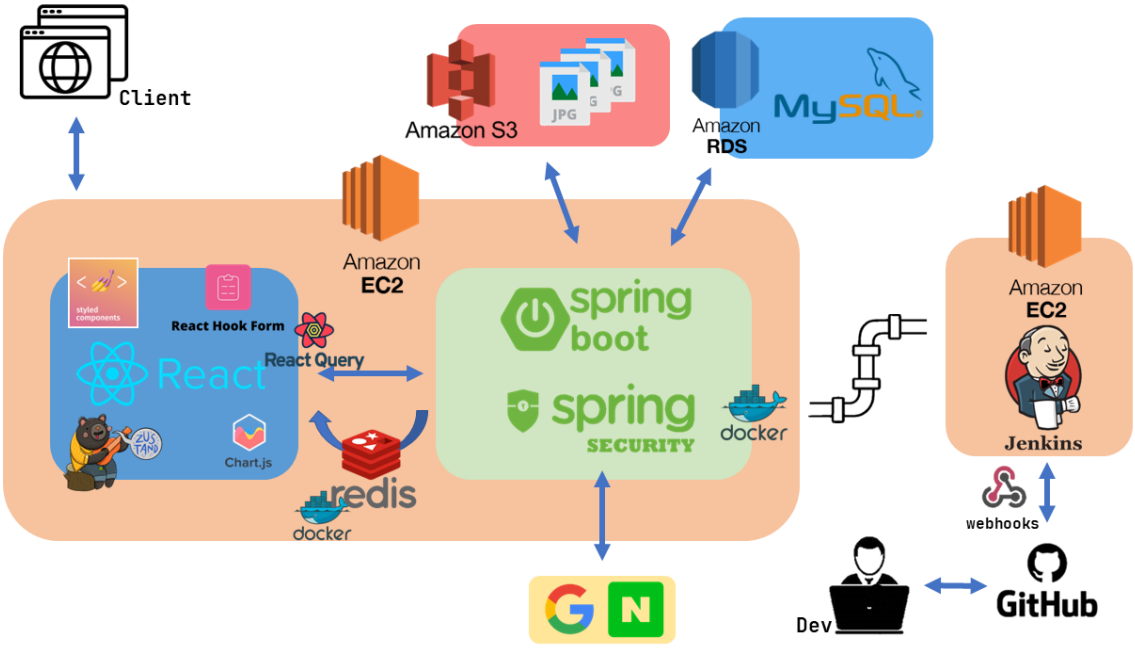
Backend `SpringBoot` `Gradle` `SpringSecurity` `SpringDataJPA` `QueryDsl` `Oauth 2.0`

Database `MySQL` `Redis` `H2` `AWS S3`

Deploy `AWS EC2` `AWS RDS` `Docker`

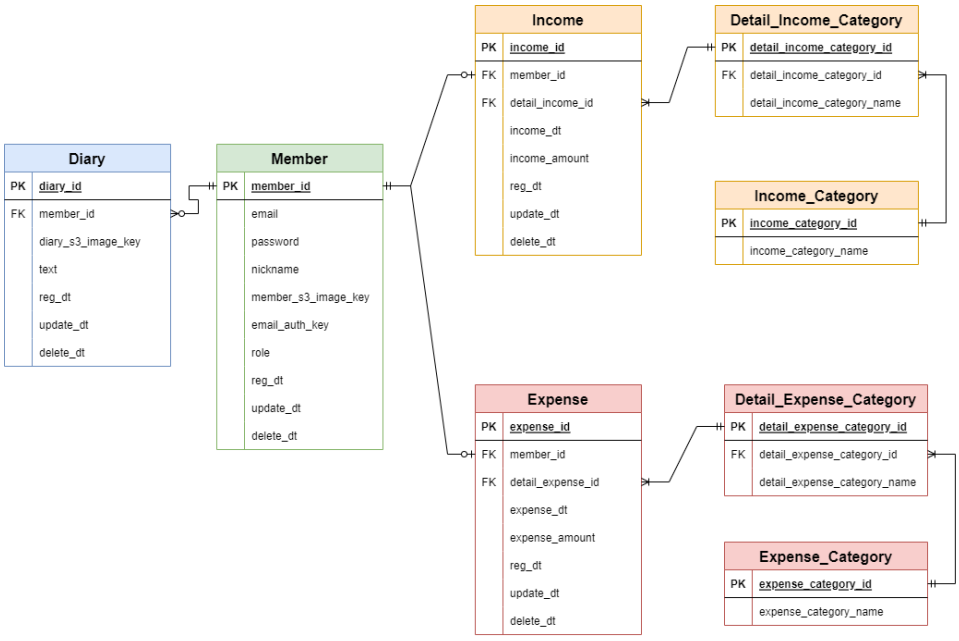
API `Notion` `Swagger`

2) 프로젝트 구조



3) ERD + 와이어 프레임

- ERD



- 와이어 프레임

https://www.figma.com/file/kacKFvkiQyAhxJnPNpjmJE/%EA%B0%80%EA%B3%84%EB%B6%80?node-id=2%3A4

4) 프로젝트 전체 구현 기능

1. **회원 가입 및 로그인, 비밀 번호 찾기**

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/fc5dd02a-17cd-443e-938c-47921cfcd129/%ED%9A%8C%EC%9B%90.mp4

▼ 회원 가입: 이메일을 인증하여 회원 가입한다.

- 유효한 이메일 주소가 있으면 회원 가입이 가능하다.
- 이메일 전송은 비동기 처리한다. 따라서, 회원 가입 제출하면 완료 창이 바로 뜬다.
- 만약 회원 가입 버튼을 누르고 이메일이 전송됐는데 이메일이 삭제된 경우는?
- 다시 회원 가입을 요청하거나 로그인을 시도하면 이메일을 재발송하여 계정을 활성화할 수 있도록 한다.

▼ 로그인: 일반 로그인과 SNS 로그인

- 일반 로그인: In&Out 회원 가입을 통한 로그인
- SNS 로그인: In&Out에 회원 가입을 하지 않아도 OAuth 2.0을 통해 구글, 네이버 계정으로 간편하게 로그인할 수 있다.

▼ 비밀번호 찾기(초기화): 등록된 이메일과 연락처를 통해 비밀번호를 초기화할 수 있다.

- 이메일과 연락처를 확인해서 회원의 이메일로 비밀번호 초기화 링크가 전송된다.
- 이메일 전송 비동기 처리: 회원 가입과 마찬가지로 비동기 처리하여 알림창이 바로 뜨도록 한다.

2. **수입, 지출, 달력(다이어리)**

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/82e9df3e-2e3a-4009-8d4a-23ff262723bd/%EB%85%B9%ED%99%94_2022_11_17_23_49_14_762.mp4

▼ 수입 : 해당 날짜의 수입 사항을 적을 수 있다

- 작성 후 저장 버튼을 누르면 서버에 해당 기록 저장됨
- 선택 버튼을 누르고 해당 열 삭제 가능
- 한 번에 여러 열 삭제 가능
- 월 이동해서 그 달의 기록들을 확인 / 수정 / 삭제 가능

▼ 지출 : 해당 날짜의 지출 사항을 적을 수 있다

- 대부분의 사항 수입과 같음

▼ 달력: 해당 달의 수입/지출 기록 여부와 다이어리 등록 여부를 확인할 수 있다

- 금일 시점의 달로 렌더링 됨
- 상단에 그 달의 수입(in)과 지출(out)의 합계가 표시된다
- 금일 날짜에 색이 들어가서 확인 가능
- 수입/지출 가계부가 기록된 날짜에는 돋보기 아이콘이 표시되며 클릭 시 수입/지출 표가 출력 된다 (마우스 위치에 따라 수입/지출 표의 출력 위치가 바뀌며 X자를 누르거나 표 바깥 부분을 누르면 창이 사라진다)
- 각 날짜를 누르면 각 날짜 별 다이어리 창이 화면에 나타나고 그 날의 일기를 등록 / 수정 / 삭제 할 수 있다
- 일기가 등록된 날짜에는 다이어리 아이콘이 표시되며 사진의 유무에 따라 주황색 / 회색으로 구분 가능
- 상단의 화살표로 전 달과 다음 달로 이동 가능
- 화면 상에 나타나지만 해당 달에 속하지 않은 날짜로는 클릭으로 접근이 불가능하다

▼ 다이어리: 해당 날짜로 기록해둔 수입/지출 확인 가능하며 짧게 메모할 수 있다

- 왼쪽 표로 해당 날짜의 수입/지출 표를 볼 수 있다

- 왼쪽 표의 여백 공간은 수입/지출의 요소의 수에 따라 계산되어 출력 된다
- 오른쪽 공간에서 다이어리를 작성할 수 있다
- 이미지를 등록할 수 있으며 수정으로 들어가 이미지 상단에 X자를 누르면 등록하려고 했던 이미지가 삭제된다
- 오른쪽 공간 상단의 화살표 아이콘을 눌러 이전 날과 다음 날의 일기 페이지로 이동 가능
- 오른쪽 공간 상단의 날짜를 눌러 달력이 뜨면 원하는 날짜의 일기 페이지로 이동 가능
- 최상단의 X자를 누르거나 다이어리의 바깥 부분을 클릭하면 다이어리 창이 사라진다

3. **월간 / 연간보고서**

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/af4dfb9b-d800-4435-b533-1fcebc63cccb/%E1%84%87%E1%85%A9%E1%84%80%E1%85%A9%E1%84%89%E1%85%A5%E1%84%8B%E1%85%A7%E1%86%BC%E1%84%89%E1%85%A1%E1%86%BC1080.mp4

▼ 월간 보고서 : 현재 월에 해당하는 수입/지출 데이터를 카테고리 별로 분류한 차트를 볼 수 있다.

- 월 단위로 날짜를 이동할 수 있다.
- 차트를 막대형, 파이형으로 볼 수 있고 카테고리 별로 금액과 비율을 알 수 있다.
- 월간 보고서를 엑셀파일로 다운로드할 수 있다.

▼ 연간 보고서 : 현재 월 기준 12개월 전의 수입/지출 데이터를 매월 카테고리 별로 분류한 그래프, 표를 볼 수 있다.

- 월 단위로 날짜를 이동할 수 있다.
- 그래프는 꺾은선 그래프로 카테고리를 선택할 수 있다.
- 그래프는 월마다 카테고리에 대한 값의 합산, 합산에 대한 증감을 알 수 있다.
- 표는 월마다 수입/지출 모든 카테고리에 대한 값의 합산을 볼 수 있다.
- 연간 보고서를 엑셀파일로 다운로드할 수 있다.

4. **회원 수정, 비밀번호 변경, 회원 탈퇴**

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0299ca56-6dbb-44f9-90e8-a37878dd8ebe/%E1%84%92%E1%85%AC%E1%84%8B%E1%85%AF%E1%86%AB%E1%84%89%E1%85%A6%E1%84%90%E1%85%B5%E1%86%BC1080.mp4

▼ 회원 수정 : 이메일을 제외한 프로필 사진 및 정보를 변경할 수 있다.

- 일반 로그인 사용자는 회원가입 시 입력한 정보를 볼 수 있고 수정할 수 있다.
- OAuth 로그인 사용자는 이메일, 닉네임, 전화번호를 볼 수 있고 수정할 수 있다.
- 프로필 사진을 등록, 수정, 삭제할 수 있다.

▼ 비밀번호 변경 : 현재 비밀번호와 새 비밀번호를 입력하여 변경할 수 있다.

- 일반 로그인 사용자는 비밀번호 변경을 이용할 수 있다.
- OAuth 로그인 사용자는 이용할 수 없다.

▼ 회원 탈퇴 : 회원 탈퇴할 수 있다.

- 일반 로그인 사용자는 비밀번호를 이용하여 탈퇴할 수 있다
- OAuth 로그인 사용자는 바로 탈퇴할 수 있다.

- 탈퇴하게 되면 다시 로그인 할 수 없다.

[개인 내용]

1) 직접 구현한 기능

1. 수입 기능 (Income API Statement)

URL	메소드	상태	body	response	error	설명	creator	API TEST	front
api/income?startDt=""&endDt=""	GET	Done		status, List<DetailIncomeCategory>, List<Income>	1. member_validation 2. start_dt, end_dt 입력 x	수입 리스트 가져오기		Done	
api/income	POST	Done	List<Income>	status, message	1. member_validation 2. (메모 제외) 값이 없는 경우	수입 저장 및 수정		Done	
api/income	DELETE	Done	List<IncomeId>	status, message	1. member_validation 2. (메모 제외) 값이 없는 경우	수입 삭제		Done	

2. 보고서 조회 중 월간수입, 연간조회 (Report API Statement)

URL	메소드	상태	body	response	error	설명	creator	API TEST	front
api/report/expense/month?startDt=""&endDt=""	GET	Done		status, List<Report>	1. member_validation 2. start_dt, end_dt 입력 x	월간 보고서 가져오기		Done	
api/report/income/month?startDt=""&endDt=""	GET	Done		status, List<Report>	1. member_validation 2. start_dt, end_dt 입력 x	월간 수입 보고서 가져오기		Done	
api/report/year?startDt=""&endDt=""	GET	Done		status, List<Report>	1. member_validation 2. start_dt, end_dt 입력 x	연간 보고서 가져오기		Done	

3. 달력 조회 기능 (Calendar API Statement)

URL	메소드	상태	body	response	error	설명	creator	API TEST	front
api/calendar?startDt=""&endDt=""	GET	Done		status, List<CalendarMonthlyDto>	1. member_validation 2. start_dt, end_dt 입력 x	달력에 날마다 수입, 지출 리스트 가져오기		Done	

4. 엑셀 출력 기능 (Excel API Statement)

URL	메소드	상태	body	response	error	설명	creator	API TEST	front

Aa URL	메소드	상태	body	response	error	설명	creator	API TEST	front
api/excel/income?startDt=""&endDt=""	GET	Done		excel download 창 팝업	1. member_validation	startDt ~ endDt 까지의 수입내역을 Excel로 Export		Done	
api/excel/expense?startDt=""&endDt=""	GET	Done		excel download 창 팝업	1. member_validation	startDt ~ endDt 까지의 지출내역을 Excel로 Export		Done	
api/excel/year	POST	Done	startDt, List<YearlyExcelDto>	excel download 창 팝업	1. member_validation	startDt 부터 1년치의 연간보고서를 Excel로 Export		Done	

2) 프로젝트에서 발휘한 역량과 성장한 점

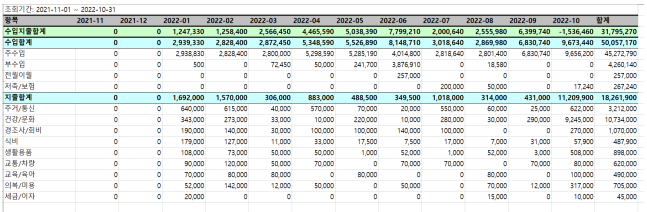
아래 내용은 모두 `제가 주도적으로 직접` 경험하고 해결한 내용입니다.

▼ 프로젝트에서의 높은 기여도: 35 / 100 (총 5명, 주관적인 점수)

- 프로젝트 전체 노션 페이지 기획 및 구성함
- 백엔드 서버 및 DataBase에 대한 AWS 배포 전반을 담당함
- 주 단위의 스프린트 계획 및 회고 주도함
- 칸반보드를 활용하여 프로젝트 전체 일정 관리하여 일정 내에 구현을 완성함
- 백엔드 API의 1/3을 담당함

▼ 엑셀 출력기능에서의 컴포넌트 설계

- 엑셀 출력기능에서의 컴포넌트 설계시 해당 기능을 1회성이 아닌 Component로 두어 확장성을 고려함
- 의존성 분리를 위해 Component는 엑셀 출력을 위한 데이터를 인자로 받으면 엑셀을 출력해주는 로직만을 담도록 구성
- Service 클래스에서는 첫 행의 컬럼정보만 세팅해주면 그에 맞는 엑셀과 데이터가 추출되도록 구현
- 이 과정에서 필요한 `apache.poi` 라이브러리를 새로 학습하여 사용하였으며, 구글링해서 알게 된 내용들을 가지고 스스로 컴포넌트 코드를 구성함



조회기간: 2021-11-01 ~ 2022-10-31

항목	2021-11	2021-12	2022-01	2022-02	2022-03	2022-04	2022-05	2022-06	2022-07	2022-08	2022-09	2022-10	합계
수입지출합계	0	0	1,247,330	1,258,400	2,566,450	4,465,590	5,038,390	7,799,210	2,000,640	2,555,980	6,399,740	-1,536,460	31,795,270
수입합계	0	0	2,939,330	2,828,400	2,872,450	5,348,590	5,526,890	8,148,710	3,018,640	2,869,980	6,830,740	9,673,440	50,057,170
주수입	0	0	2,938,830	2,828,400	2,800,000	5,298,590	5,285,190	4,014,800	2,818,640	2,801,400	6,830,740	9,656,200	45,272,790
부수입	0	0	500	0	72,450	50,000	241,700	3,876,910	0	18,580	0	0	4,260,140
전월이월	0	0	0	0	0	0	0	257,000	0	0	0	0	257,000
저축/보험	0	0	0	0	0	0	0	0	200,000	50,000	0	17,240	267,240
지출합계	0	0	1,692,000	1,570,000	306,000	883,000	488,500	349,500	1,018,000	314,000	431,000	11,209,900	18,261,900
주거/통신	0	0	640,000	615,000	40,000	570,000	70,000	20,000	550,000	60,000	25,000	622,000	3,212,000
건강/문화	0	0	343,000	273,000	33,000	10,000	220,000	10,000	280,000	30,000	290,000	9,245,000	10,734,000
관조사/경비	0	0	190,000	140,000	30,000	100,000	100,000	140,000	100,000	0	0	270,000	1,070,000
식비	0	0	179,000	127,000	11,000	33,000	17,500	7,500	17,000	7,000	31,000	57,900	487,900
생활용품	0	0	108,000	73,000	50,000	50,000	1,000	52,000	1,000	52,000	3,000	508,000	898,000
교통/차량	0	0	90,000	120,000	50,000	70,000	0	70,000	70,000	0	70,000	80,000	620,000
교육/육아	0	0	70,000	80,000	80,000	0	80,000	0	0	80,000	0	100,000	490,000
의복/미용	0	0	52,000	142,000	12,000	50,000	0	50,000	0	70,000	12,000	317,000	705,000
세금/이자	0	0	20,000	0	0	0	0	0	0	15,000	0	10,000	45,000

연간 보고서를 엑셀로 출력한 화면 예시

▼ **ExcelService.java** 코드

```
[ExcelService.java]

// 컬럼의 이름과 셀의 넓이를 클래스에 담아서 컴포넌트에 넘길 수 있도록 구현함.
public static class ColConfig {
    public String colName;
    public int colWidth;
}

public ResponseEntity<InputStreamResource> downloadIncomeExcelFile(
    HttpServletRequest request, String email, LocalDate startDt, LocalDate endDt)
    throws IOException {

    Member member = findMemberByEmail(email);

    // ExcelComponent에 파라미터로 넘길 데이터도 메서드를 통해서 만들어옴.
    List<IncomeExcelDto> excelRowData = setIncomeExcelDataList(email, startDt, endDt);
```

```
    final ArrayList<ColConfig> headColumn = new ArrayList<>(Arrays.asList(
        new ColConfig("No", 1000),
        new ColConfig("일자", 3000),
        new ColConfig("카테고리", 3000),
        new ColConfig("세부카테고리", 4000),
        new ColConfig("내역", 8000),
        new ColConfig("금액", 3000),
        new ColConfig("메모", 5000))
    );

    try {
        return excelDownloadComponent.downloadIncomeExcelFile(request, email, startDt, endDt,
            excelRowData, headColumn);
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

▼ 쿼리 및 성능 개선

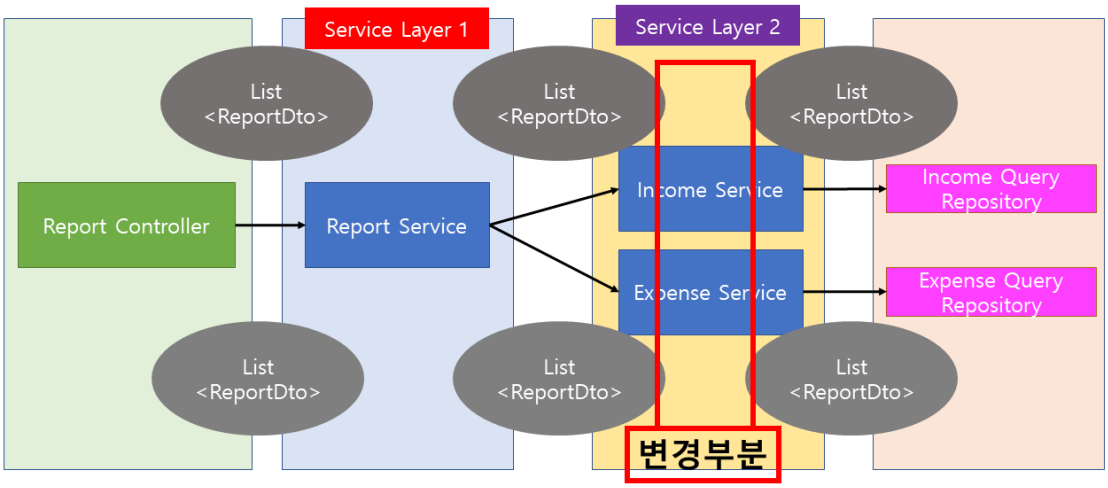
- **문제상황 및 원인**
    - 처음 기능 구현 시 쿼리를 생각하지 않고 로직을 구현한 것이 원인. 아무리 정보가 많이 필요한 연간보고서라도 쿼리가 50개가 나갔음.
        - 한 달의 내역을 가져오는 메서드를 연간에서 재활용 → 쿼리 X12
        - 한 달의 내역을 가져올 때 내부적으로 합계를 구하는 것을 쿼리로 했음 → 쿼리 X2
        - 위 작업이 수입, 지출로 나눠져 있었음 → 쿼리 X2
        - 멤버 Validation에서 쿼리 2개

```
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select incomecate2_.income_category_name as col_0_0_, sum(income0_.income_amount) as col_1_0_, cast(sum(income0_.income_amount)*? as doubl
Hibernate: select sum(income0_.income_amount) as col_0_0_ from income income0_ where income0_.member_id=? and (income0_.income_dt between ? and ?)
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select incomecate2_.income_category_name as col_0_0_, sum(income0_.income_amount) as col_1_0_, cast(sum(income0_.income_amount)*? as doubl
Hibernate: select sum(income0_.income_amount) as col_0_0_ from income income0_ where income0_.member_id=? and (income0_.income_dt between ? and ?)
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select incomecate2_.income_category_name as col_0_0_, sum(income0_.income_amount) as col_1_0_, cast(sum(income0_.income_amount)*? as doubl
Hibernate: select sum(income0_.income_amount) as col_0_0_ from income income0_ where income0_.member_id=? and (income0_.income_dt between ? and ?)
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select incomecate2_.income_category_name as col_0_0_, sum(income0_.income_amount) as col_1_0_, cast(sum(income0_.income_amount)*? as doubl
Hibernate: select sum(income0_.income_amount) as col_0_0_ from income income0_ where income0_.member_id=? and (income0_.income_dt between ? and ?)
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select incomecate2_.income_category_name as col_0_0_, sum(income0_.income_amount) as col_1_0_, cast(sum(income0_.income_amount)*? as doubl
Hibernate: select sum(income0_.income_amount) as col_0_0_ from income income0_ where income0_.member_id=? and (income0_.income_dt between ? and ?)
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select incomecate2_.income_category_name as col_0_0_, sum(income0_.income_amount) as col_1_0_, cast(sum(income0_.income_amount)*? as doubl
Hibernate: select sum(income0_.income_amount) as col_0_0_ from income income0_ where income0_.member_id=? and (income0_.income_dt between ? and ?)
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select incomecate2_.income_category_name as col_0_0_, sum(income0_.income_amount) as col_1_0_, cast(sum(income0_.income_amount)*? as doubl
Hibernate: select sum(income0_.income_amount) as col_0_0_ from income income0_ where income0_.member_id=? and (income0_.income_dt between ? and ?)
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select incomecate2_.income_category_name as col_0_0_, sum(income0_.income_amount) as col_1_0_, cast(sum(income0_.income_amount)*? as doubl
Hibernate: select sum(income0_.income_amount) as col_0_0_ from income income0_ where income0_.member_id=? and (income0_.income_dt between ? and ?)
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select incomecate2_.income_category_name as col_0_0_, sum(income0_.income_amount) as col_1_0_, cast(sum(income0_.income_amount)*? as doubl
Hibernate: select sum(income0_.income_amount) as col_0_0_ from income income0_ where income0_.member_id=? and (income0_.income_dt between ? and ?)
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select incomecate2_.income_category_name as col_0_0_, sum(income0_.income_amount) as col_1_0_, cast(sum(income0_.income_amount)*? as doubl
Hibernate: select sum(income0_.income_amount) as col_0_0_ from income income0_ where income0_.member_id=? and (income0_.income_dt between ? and ?)
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select incomecate2_.income_category_name as col_0_0_, sum(income0_.income_amount) as col_1_0_, cast(sum(income0_.income_amount)*? as doubl
Hibernate: select sum(income0_.income_amount) as col_0_0_ from income income0_ where income0_.member_id=? and (income0_.income_dt between ? and ?)
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select incomecate2_.income_category_name as col_0_0_, sum(income0_.income_amount) as col_1_0_, cast(sum(income0_.income_amount)*? as doubl
Hibernate: select sum(income0_.income_amount) as col_0_0_ from income income0_ where income0_.member_id=? and (income0_.income_dt between ? and ?)
Hibernate: select member0_.member_id as member_i1_7_, member0_.delete_dt as delete_d2_7_, member0_.reg_dt as reg_dt3_7_, member0_.update_dt as update
Hibernate: select expensecat2_.expense_category_name as col_0_0_, sum(expense0_.expense_card+expense0_.expense_cash) as col_1_0_, cast(sum(expense0_.
Hibernate: select expensecat2_.expense_category_name as col_0_0_, sum(expense0_.expense_card+expense0_.expense_cash) as col_1_0_, cast(sum(expense0_.
Hibernate: select expensecat2_.expense_category_name as col_0_0_, sum(expense0_.expense_card+expense0_.expense_cash) as col_1_0_, cast(sum(expense0_.
```

- **해결방법 및 배운점**

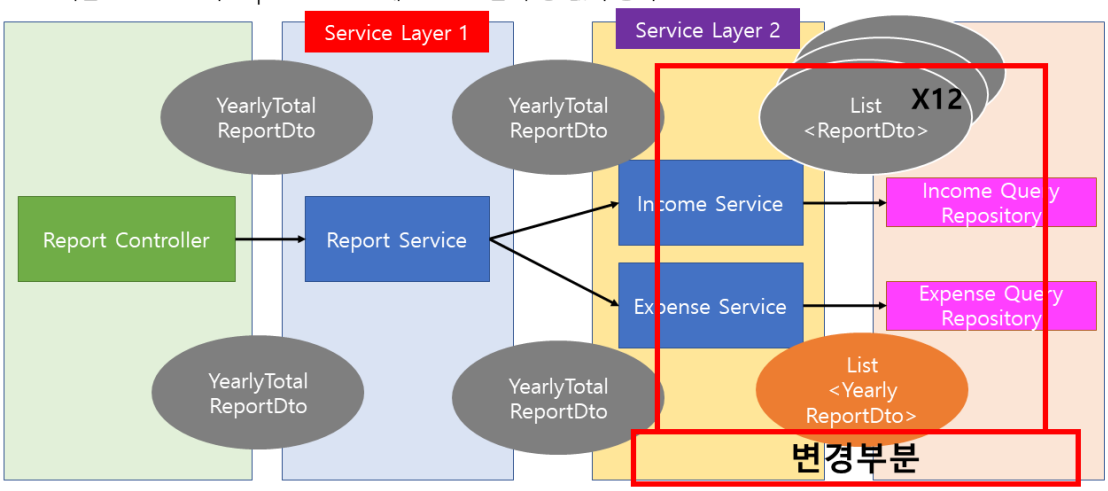
  [월간 보고서 조회 개선작업]

  - 서비스 로직 변경 (합계 구하는 쿼리 대신 List loop 돌면서 합계 계산)
  - Report Controller, Service Layer 1, Service Layer 2의 기존 테스트코드 그대로 통과



  [연간 보고서 조회 개선작업]

  - YearlyReportDto 구현 및 쿼리 메서드 구현
  - Service Layer 2 서비스 메서드 구현
  - 기존 Controller와 Report Service 테스트코드는 수정 없이 통과



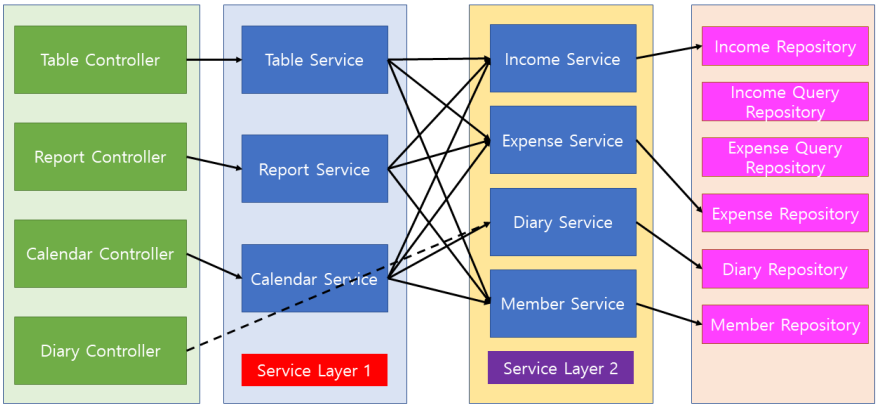
  - 의존성을 줄여서 코드를 작성하는 것의 효율성을 체감함.
  - 가장 뿌듯했던 것은, 쿼리 메서드 수정 부분을 제외하고는 테스트코드의 수정 거의 없이 재통과 가능했고, 바로 프로젝트에 적용할 수 있었던 부분
  - 나의 기존 코드와 로직에 갇혀있음을 인지하였고,
    과감하게 관점을 바꿀 용기를 내지 않으면 내 코드와 발전을 막을 수 있음을 깨달음.

- **성능 개선**

	Monthly			Yearly		
	기존	변경	개선비율	기존	변경	개선비율
쿼리개수	3	2	**33%**	50	4	**92%**
평균(ms)	124.6	66.1	**47%**	2875.5	149.1	**95%**
호출1	262	159		4841	232	
호출2	112	84		2888	117	
호출3	119	33		1217	137	
호출4	116	47		899	133	
호출5	116	48		841	148	
호출6	118	58		732	133	
호출7	91	55		6159	144	
호출8	98	59		8451	144	
호출9	106	66		2054	136	
호출10	108	52		673	167	

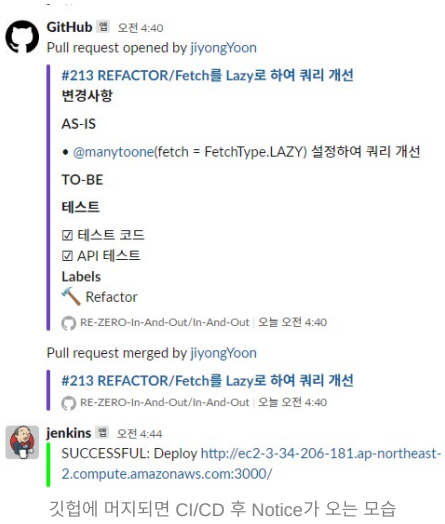
▼ 프로젝트 구조에 대한 고민과 구현

- 하나의 컨트롤러 안에 여러 서비스 메서드를 호출하는 경우가 발생하였는데, `그래도 괜찮을까?` 에 대한 의문으로 시작하여 프로젝트 구조에 대해 고민하게 됨 (저장버튼이 1개라서 `addIncome` 과 `updateIncome` 메서드가 함께 실행되어야 했음)
- 서비스 Layer를 나누어 관리하기로 결정하고 두 메서드를 묶는 서비스 클래스를 하나 더 둠
- 아래 그림처럼 구조를 가져감으로 Service에서의 순환참조나 순환구조를 애초에 방지하도록 설계함



▼ 서버 배포와 도커, CI / CD 경험

- CI/CD를 사용하기 위해 Jenkins를 프로젝트에 적용해야 했기 때문에 스스로 학습하고 프로젝트에 적용함
- Github에 PR이 Merge되면 `webhooks` 를 활용하여 Jenkins의 파이프라인 가동
- `Git Clone` - `TestCode Check` - `Jar Build` - `Docker Image Build` - `Docker Image Push & Pull` - `Docker Image Run` 순서로 CI / CD 진행



깃헙에 머지되면 CI/CD 후 Notice가 오는 모습

▼ 직접 작성한 Jenkins Pipeline 스크립트

```
pipeline {
    environment {
        repository = "a/backend"
        DOCKERHUB_CREDENTIALS = credentials('dockerhub-jenkins')
        dockerImage = ''
    }

    agent any

    stages {

        stage('github clone') {
            steps {
                git branch: 'main',
                    credentialsId: 'github-jenkins',
                    url: 'https://github.com/RE-ZERO-In-And-Out/In-And-Out.git'
                echo 'git clone finish'
            }
        }

        stage('build jar'){
            steps{
                dir('server'){
                    echo 'build start'
                    sh'''
                    sudo cp /home/ubuntu/properties/application-dbsecret.properties /var/lib/jenkins/workspace/inandout-jenkin
                    sudo cp /home/ubuntu/properties/application-secret.properties /var/lib/jenkins/workspace/inandout-jenkins/
                    sudo chmod +x gradlew
                    ./gradlew clean build
                    '''
                }
            }
            post{
                success {
                    echo '***************.jar build success*****************'
                }
                failure {
                    error '***************.jar build failed & pipeline stops*****************'
                }
            }
        }

        stage('docker image build'){
            steps{
                dir('/var/lib/jenkins/workspace/Jenkins-pipeline/server'){
                    echo 'image build start'
                    sh'''
                    sudo chmod 666 /var/run/docker.sock
                    sudo cp /home/ubuntu/dockerfile/Dockerfile /var/lib/jenkins/workspace/Jenkins-pipeline/server/Dockerfile
                    sudo docker build -t inandout .
                    docker tag inandout:latest a/backend:inandout
                    '''
                }
            }
            post{
                success {
                    echo '***************docker image build success***************'
```

```
                }
                failure {
                    error '***************docker image build failed & pipeline stops*****************'
                }
            }
        }

        stage('Login_jenkins server'){
            steps{
                sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
            }
        }

        stage('Deploy our image') {
            steps {
                script {
                    sh 'docker push $repository:inandout'
                }
            }
        }

        stage('Cleaning up') {
          steps {
                  sh "docker rmi $repository:inandout"
            }
        }

        stage('Prepare'){
            steps{
                sshagent(['c3e3d23e-e1f8-4102-90c9-a79382e83bf8']) {
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@3.34.206.181 'sudo chmod 666 /var/run/docker.sock'"
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@3.34.206.181 'docker images'"
                }
            }
        }

        stage('Login_boot server'){
            steps{
                sshagent(['c3e3d23e-e1f8-4102-90c9-a79382e83bf8']) {
                    sh 'echo $DOCKERHUB_CREDENTIALS_PSW | ssh -o StrictHostKeyChecking=no ubuntu@3.34.206.181 docker login -u
                }
            }
        }

        stage('Docker Run') {
            steps {
                echo 'Pull Docker Image & Docker Image Run'
                sshagent(['c3e3d23e-e1f8-4102-90c9-a79382e83bf8']) {
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@3.34.206.181 'docker pull a/backend:inandout'"
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@3.34.206.181 'docker ps -q --filter name=inandout | grep -q . &
                    sh "ssh -o StrictHostKeyChecking=no ubuntu@3.34.206.181 'docker run -d --name inandout -p 8080:8080 a5/bac
                }
            }
            post{
                success {
                    echo '***************docker image run success***************'
                }
                failure {
                    error '***************docker image run failed & pipeline stops*****************'
                }
            }
        }
    }

    post {
        success {
            slackSend (
                channel: '#inandout-pr',
                color: '#00FF00',
                message: "SUCCESSFUL: Deploy http://ec2-3-34-206-181.ap-northeast-2.compute.amazonaws.com:3000/")
        }
        failure {
            slackSend (
                channel: '#inandout-pr',
                color: '#FF0000',
                message: "FAILED: Job '${env.JOB_NAME} [${env.BUILD_NUMBER}]' (${env.BUILD_URL})")
        }
    }
}
```

▼ Github 브랜치 컨벤션 및 PR 활성화, Mock Test와 API 테스트 진행.

- 프로젝트 현재 총 200개의 pr이 Closed 됨.

- pr 중 1/5 이상은 코드리뷰를 통해 코드 구조가 변경되거나 오류가 잡히고 효율이 개선되는 경험을 함.
- 코드에서 좋은 부분을 습득하여 내 코드에 적용하는 경우도 있었음.
- 브랜치 컨벤션과 라벨, pr-templete을 활용하였고, AS-IS / TO-BE / TEST 세 부분으로 구성하여 관리하였음.
- 이를 통해 테스트 코드가 깨지거나 부트가 빌드되지 않는 등의 비효율을 최소화함.

▼ 협업 툴을 활용하여 업무 효율 극대화

▼ 노션 회의록을 사용하여 진행사항을 체계적으로 정리하고 관리함

🚀 회의록

표 +

날짜 | 태그: Daily Scrum,Sprint Pl... | + 필터 추가

Aa 이름	태그	날짜
Weekly Review (1.week)	Weekly Review	2022년 10월 16일
Sprint Plan 1 (2.week)	Sprint Plan	2022년 10월 17일
Daily Scrum 1 (10.17)	Daily Scrum	2022년 10월 17일
Daily Scrum 2 (10.18)	Daily Scrum	2022년 10월 18일
Daily Scrum 3 (10.19)	Daily Scrum	2022년 10월 19일
Daily Scrum 4 (10. 20)	Daily Scrum	2022년 10월 20일
Daily Scrum 5 (10.21)	Daily Scrum	2022년 10월 21일
Weekly Review (2.week)	Weekly Review	2022년 10월 23일
Sprint Review 1 (2.week)	Sprint Review	2022년 10월 23일
Sprint Plan 2 (3.week)	Sprint Plan	2022년 10월 24일
Daily Scrum 6 (10.24)	Daily Scrum	2022년 10월 24일
Daily Scrum 7 (10.25)	Daily Scrum	2022년 10월 25일
Daily Scrum 8 (10.26)	Daily Scrum	2022년 10월 26일
Daily Scrum 9 (10.27)	Daily Scrum	2022년 10월 27일
Daily Scrum 10 - (10.28)	Daily Scrum	2022년 10월 28일
Weekly Review (3.week)	Weekly Review	2022년 10월 30일
Sprint Review 2 (3.week)	Sprint Review	2022년 10월 30일
Sprint Plan 3 (4.week)	Sprint Plan	2022년 10월 31일
Daily Scrum 11 - (10.31)	Daily Scrum	2022년 10월 31일
Daily Scrum 12 - (11.01)	Daily Scrum	2022년 11월 1일
Daily Scrum 13 - (11.02)	Daily Scrum	2022년 11월 2일
Daily Scrum 14 - (11.03)	Daily Scrum	2022년 11월 3일
Daily Scrum 15 - (11.04)	Daily Scrum	2022년 11월 4일
Weekly Review (4.week)	Weekly Review	2022년 11월 6일
Sprint Review 3 (4.week)	Sprint Review	2022년 11월 6일
Sprint Plan 4 (5.week)	Sprint Plan	2022년 11월 7일
Daily Scrum 16 - (11.07)	Daily Scrum	2022년 11월 7일
Daily Scrum 17 - (11.08)	Daily Scrum	2022년 11월 8일
Daily Scrum 18 - (11.09)	Daily Scrum	2022년 11월 9일
Daily Scrum 19 - (11.10)	Daily Scrum	2022년 11월 10일
Daily Scrum 20 - (11.11)	Daily Scrum	2022년 11월 11일
Weekly Review (5.week)	Weekly Review	2022년 11월 13일
Sprint Review 4 - (5.week)	Sprint Review	2022년 11월 13일
Sprint Plan 5 (6.week)	Sprint Plan	2022년 11월 14일

▼ 노션 페이지를 활용하여 API 수정사항 누락방지

API 테스트 수정요청사항

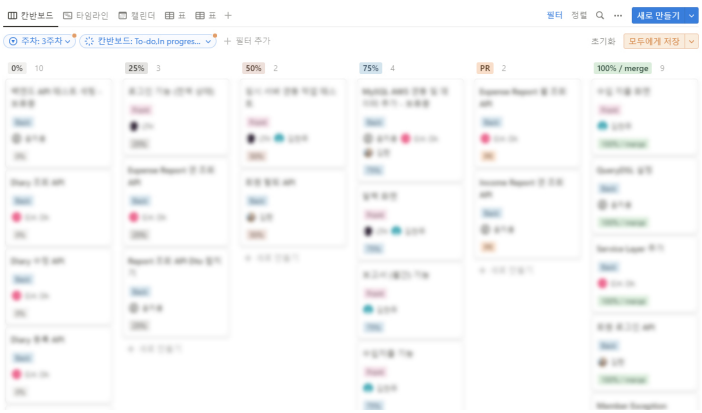
표 + 필터 정렬 새로 만들기

제목 없음

Aa 항목	요청자	요청일	진행상황	담당자	해결일
회원 정보 불러올 때 오류	윤지용	2022년 11월 10일	Done	김찬주	2022년 11월 14일
비밀번호 변경 CORS 오류 (수입 지출 등록에도 동일한 오류 발생)	윤지용	2022년 11월 10일	Done	윤지용	2022년 11월 11일
Calendar에서 지출금액 조회오류	윤지용	2022년 11월 11일	Done	김찬주	2022년 11월 14일
다이어리 get 요청 보냈을 때 지출 데이터 하나만 받아짐	LTH	2022년 11월 12일	Done	윤지용	2022년 11월 12일
수입/지출 테이블 수정 오류	윤지용	2022년 11월 12일	Done	김찬주	2022년 11월 12일
연간 보고서 카테고리 수정	윤지용	2022년 11월 12일	Done	김찬주	2022년 11월 12일
INCOME-DELETE-삭제안됨	윤지용	2022년 11월 14일	Done	김찬주	2022년 11월 14일
Expense delete 안됨	G.H. Oh	2022년 11월 14일	Done	김찬주	2022년 11월 14일
Excel-Monthly-Export	윤지용	2022년 11월 14일	Done	김찬주	2022년 11월 14일
Table-수입지출 금액 입력 시 0값, 날짜 연동	윤지용, G.H. Oh	2022년 11월 14일	Done	김찬주	2022년 11월 14일
Report - 보고서 파이형 그래프 수정	윤지용	2022년 11월 14일	Done	김찬주	2022년 11월 14일
EXCEL-GET-YEAR	윤지용	2022년 11월 14일	Done	김찬주	2022년 11월 14일
Member-setting-프로필 변경시 사진이 없어짐	윤지용	2022년 11월 15일	Done	김찬주	2022년 11월 17일
Table-수입/지출테이블에 0을 지우게 되면 null이 전달되는 상황	윤지용	2022년 11월 15일	Done	김찬주	2022년 11월 15일
Table-매달 1일이 날짜 입력이 안됨	윤지용	2022년 11월 15일	Done	김찬주	2022년 11월 15일
Calendar-화면이 안뜨는 기간존재 2	윤지용	2022년 11월 15일	Done	LTH	2022년 11월 15일
비밀번호 찾기 버튼			Done	김찬주	2022년 11월 16일

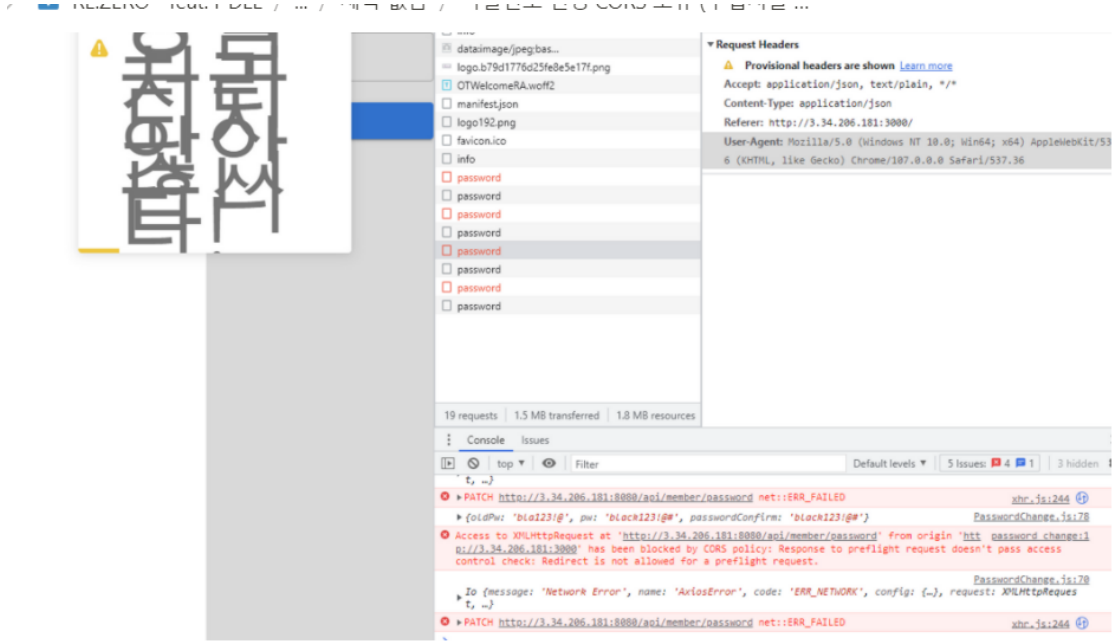
+ 새로 만들기

▼ 칸반 보드를 활용하여 업무일정 관리



▼ 협업 마인드

- 어떤 이슈가 생기거나, 도전해 보고 싶은 부분이 생겼을 때 **항상 'YES'인 마인드**로 임하였습니다.
- 나중에 글을 읽은 팀원들이 요청자에게 질문했을 때 'ㅇㅇ님이 해결해주셨습니다'라는 피드백을 많이 받았다고 합니다.
- API 수정 요청 시 문제를 정확하게 전달하기 위해 Network HTTP 요청을 확인하고, 해당 이슈로 추론되는 부분의 내용을 검색해 보고 공유하여 함께 해결함
- 이를 통해 프론트엔드 개발자의 관심사와 지식 및 업무 범위에 대한 내용을 알게 됨



예상원인(선택)

- 요 블로그 보면 spring security에서 해당 option request를 redirect 시켜서 그런것 같다고 하네요 (링크)
- 위 블로그 보다는 csrf 설정이 enabled 되어 있어서 안되서 그렇다는 것이 더 신빙성이 있는 것 같습니다 (링크)

API 수정 요청 내용

커버 추가 댓글 추가

API 테스트 수정요청사항

표

제목 없음

항목	요청자	요청일	진행상황	담당자	해결일
회원 정보 불러올 때 오류	윤지용	2022년 11월 10일	Done	김찬주	2022년 11월 14일
비밀번호 변경 CORS 오류 (수입 지출 등록에도 동일한 오류 발생)	윤지용	2022년 11월 10일	Done	윤지용	2022년 11월 11일
Calendar에서 지출금액 조회오류	윤지용	2022년 11월 11일	Done	김찬주	2022년 11월 14일
다이어리 get 요청 보냈을 때 지출 데이터 하나만 받아짐	LTH	2022년 11월 12일	Done	윤지용	2022년 11월 12일
수입/지출 테이블 수정 오류	윤지용	2022년 11월 12일	Done	김찬주	2022년 11월 12일
연간 보고서 카테고리 수정	윤지용	2022년 11월 12일	Done	김찬주	2022년 11월 12일
INCOME-DELETE-삭제안됨	윤지용	2022년 11월 14일	Done	김찬주	2022년 11월 14일
Expense delete 안됨	G.H. Oh	2022년 11월 14일	Done	김찬주	2022년 11월 14일
Excel-Monthly-Export	윤지용	2022년 11월 14일	Done	김찬주	2022년 11월 14일
Table-수입지출 금액 입력 시 0값, 날짜 연동	윤지용 G.H. Oh	2022년 11월 14일	Done	김찬주	2022년 11월 14일
Report - 보고서 파이형 그래프 수정	윤지용	2022년 11월 14일	Done	김찬주	2022년 11월 14일
EXCEL-GET-YEAR	윤지용	2022년 11월 14일	Done	김찬주	2022년 11월 14일
Member-setting-프로필 변경시 사진이 없어짐	윤지용	2022년 11월 15일	Done	김찬주	2022년 11월 17일
Table-수입/지출테이블에 0을 지우게 되면 null이 전달되는 상황	윤지용	2022년 11월 15일	Done	김찬주	2022년 11월 15일
Table-매달 1일이 날짜 입력이 안됨	윤지용	2022년 11월 15일	Done	김찬주	2022년 11월 15일
Calendar-화면이 안뜨는 기간존재 2	윤지용	2022년 11월 15일	Done	LTH	2022년 11월 15일
비밀번호 찾기 버튼			Done	김찬주	2022년 11월 16일

새로 만들기

개발 후 API 테스트 시 모든 테스트들을 성실하게 수행하여 가장 많은 수정요청사항을 게시함

▼ 프로젝트에서 개선하거나 추가로 하고 싶은 부분

- 관리자 페이지를 추가로 구현하고 싶습니다. 현재는 카테고리 추가를 하는 부분이 DB에서만 가능하기 때문에 DB접근 없이 온전히 서비스가 작동된다고 하기에는 부족하다고 생각합니다. 관리자 페이지에는 카테고리 추가, 삭제 부분을 메인으로 하고 유저관리(활성화, 비활성화, 아이디와 비밀번호 오류 등의 내용 처리가 가능하도록) 기능을 추가해보고 싶었습니다.
- 맨 처음 기획했던 방향인 [가계부 + SNS] 서비스를 추가해보고 싶습니다. 사실 프로젝트 초반에는 내가 작성한 가계부를 SNS 형식으로 공유하고 서로 피드백을 주고받는 서비스를 개발해보고 싶었지만, 저희의 역량과 기간, 그리고 완성도 등의 모든 것을 고려하여 범위를 조정하게 되었습니다. 결과적으로는 정말 잘한 선택이라고 생각합니다. 덕분에 더 긴시간 API 테스트를 하고 서버 배포도 더 구체적으로 할 수 있었고, 가계부 기능을 조금 더 완성도 있게 가져갈 수 있었습니다. 기간에 따른 목표치를 잘 설정하는 것도 꼭 필요한 역량이라고 생각합니다.

읽어주셔서 감사합니다 🙂