



2025.06.18 - 2025.07.20

Status 404 팀

멘토님 : 브레드

팀장 : 이정은

FE : 김태진, 강민경, 김남빈, 김성윤

BE : 김예진, 김지성, 이정은, 이하은



아이디어만 있다면? 펀딩은 **FUNDMATE**와 함께!

FUNDMATE는 누구나 쉽게 펀딩을 개설하고 참여할 수 있도록 돋는 AI 및 공공 데이터 기반 펀딩 플랫폼입니다



AI 콘텐츠 지원

- 한줄 소개 자동 생성
- 아이디어 확장 기능



공공데이터 통계

- 카테고리별 시각화
- 시장 인사이트 제공



확장성 높은 구조

- 백엔드 MSA 구조
- 프론트엔드 MFE 구조

FUNDMATE와 함께라면, 당신의 아이디어가 현실이 됩니다! 

펀딩의 새로운 경험을 지금 시작해보세요.

일정 개요

1

1주차 (6/18-6/21)

공통 - 프로젝트 주제 선정, 피그마 디자인, 역할 분담

2

2주차 (6/22-6/28)

FE - UI 제작

BE - ERD설계, 기능명세서 작성, 서버 환경 설정

3

3주차 (6/29-7/5)

FE - UI 제작 및 federation

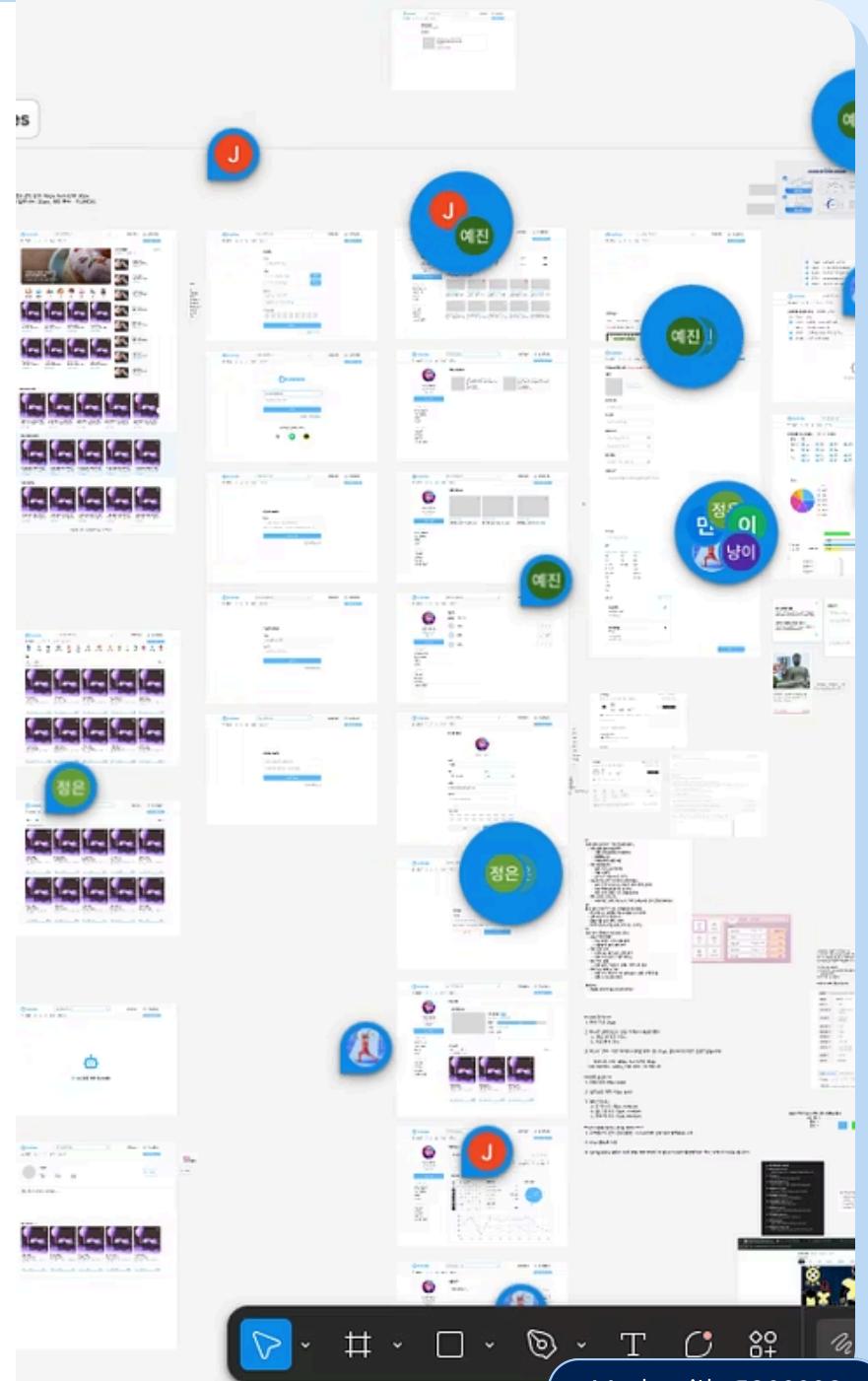
BE - 서버 환경 설정, API 기능 구현

4

4-5주차 (7/6-7/19)

FE - UI 제작, 버그 수정, API 연동, 배포

BE - API 기능 구현 마무리 및 리팩토링





핵심 기능



펀딩 시스템

펀딩 개설/후원, 마이페이지, 인증, 팔로우, 결제 시스템, 찜, 후기



AI 지원

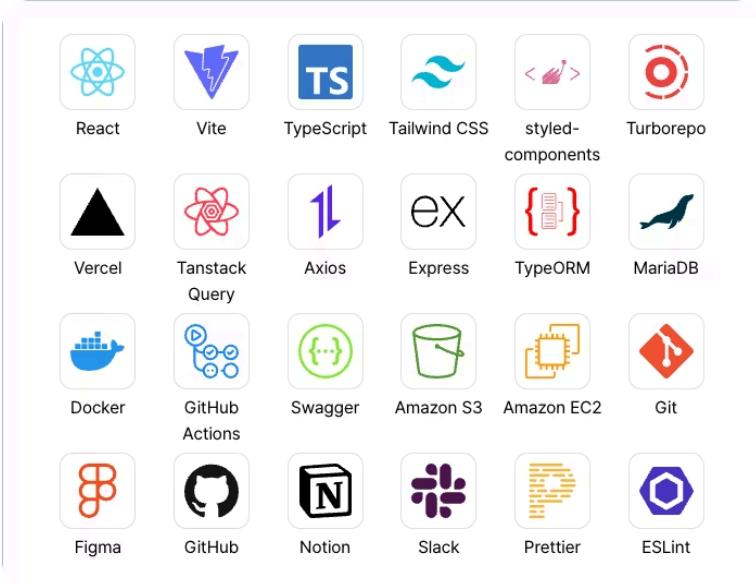
AI 아이디어 요약/확장 기능으로 펀딩 콘텐츠 제작 지원



데이터 시각화

공공데이터 시각화 기반 아이디어 탐색 및 시장 분석

기술 스택



프론트엔드

- React
- TypeScript
- Tailwind
- Vite
- Turborepo

백엔드

- Express
- TypeORM
- MariaDB
- AWS
- Docker

기타

- Grok AI
- 공공 API
- Nx, Github Actions
- Swagger, JWT, S3

강민경 - FE 개발

역할

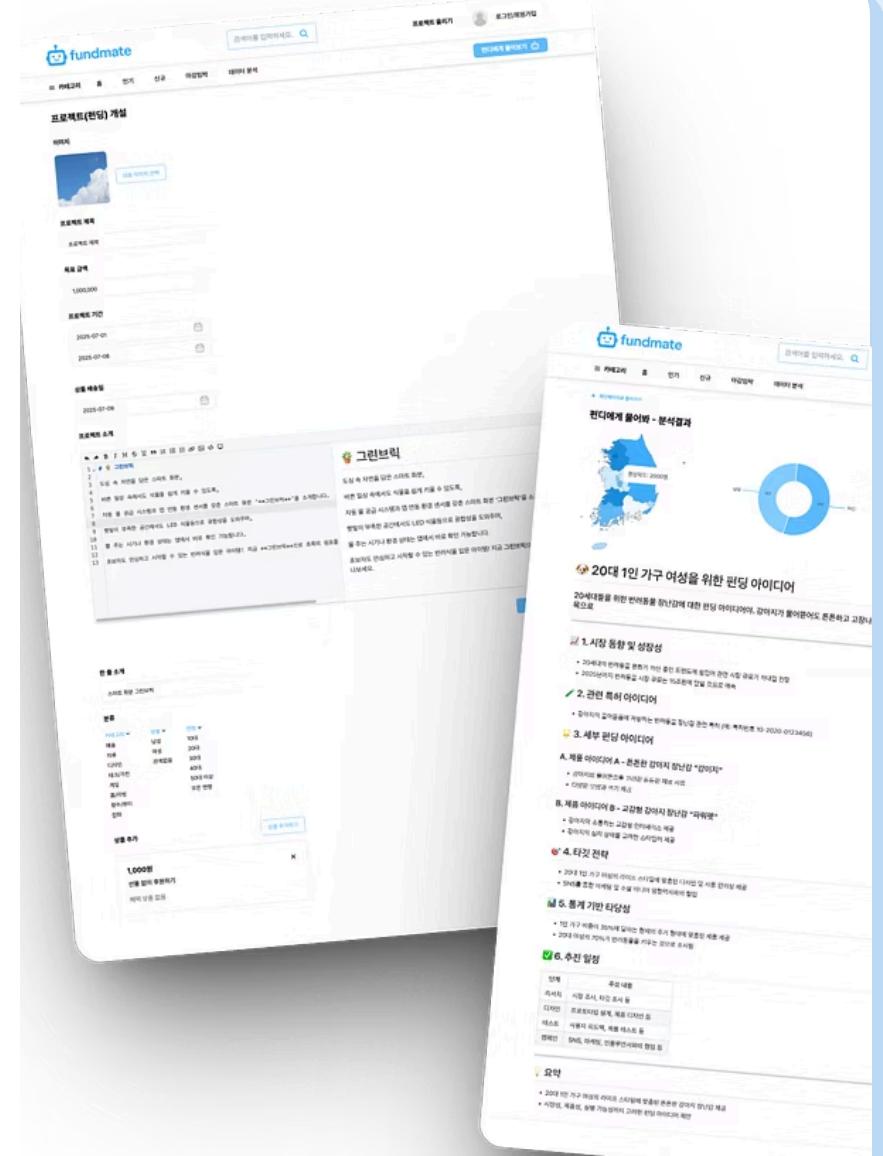
- 펀딩 개설 페이지 및 AI 분석 페이지 구현
 - 공통 컴포넌트 및 유тиль 기능 개발

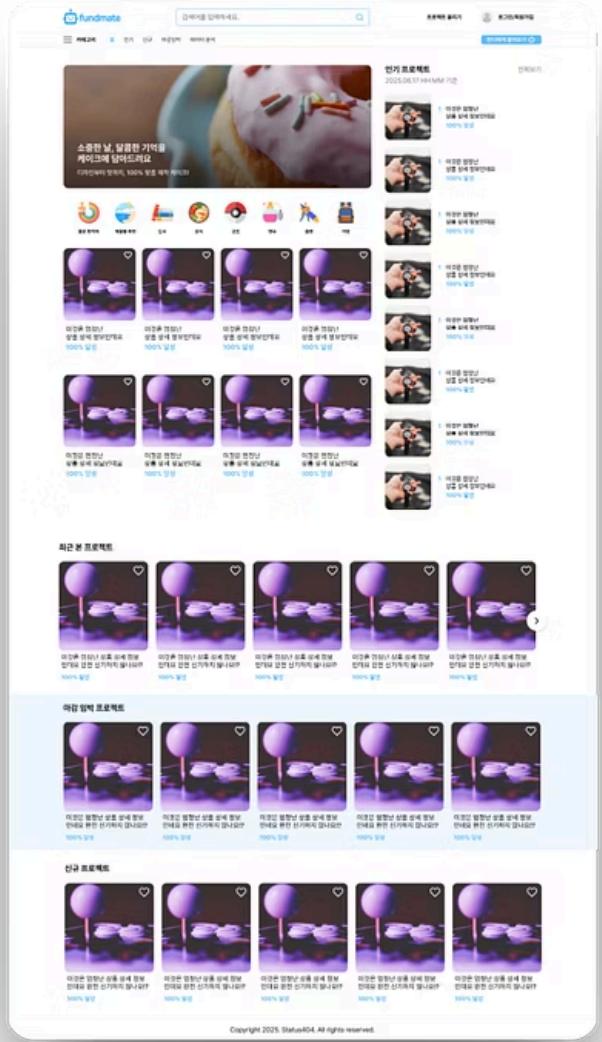
주요 구현 기능

- S3 이미지 업로드,
날짜 입력, 마크다운 에디터
 - AI 분석 결과 차트 및
지도 시각화
 - PDF 저장 기능, 로딩/모달
UI

기술 스택

- React, TypeScript, Tailwind, TanStack Query
 - Jest, nivo, react-simple-south-korea-map-chart, html2pdf.js





김태진 - FE 개발 & MFE 환경 구축

역할

- 메인 페이지, 공공데이터 시각화 및 인증 기능 개발
- MFE(Micro Frontend) 환경 설정 및 배포 구성

주요 구현 기능

- 공공데이터 시각화 (nivo, table), 쿼리 기반 렌더링
- 인증(로그인/비밀번호 변경), 반응형 레이아웃
- 공용 UI 컴포넌트 및 API 통신 관리
- MFE 구성 및 빌드 프로세스 설정

기술 스택

React, Zustand, TanStack Query, Vite Federation, Turborepo

김남빈 - FE 개발

역할

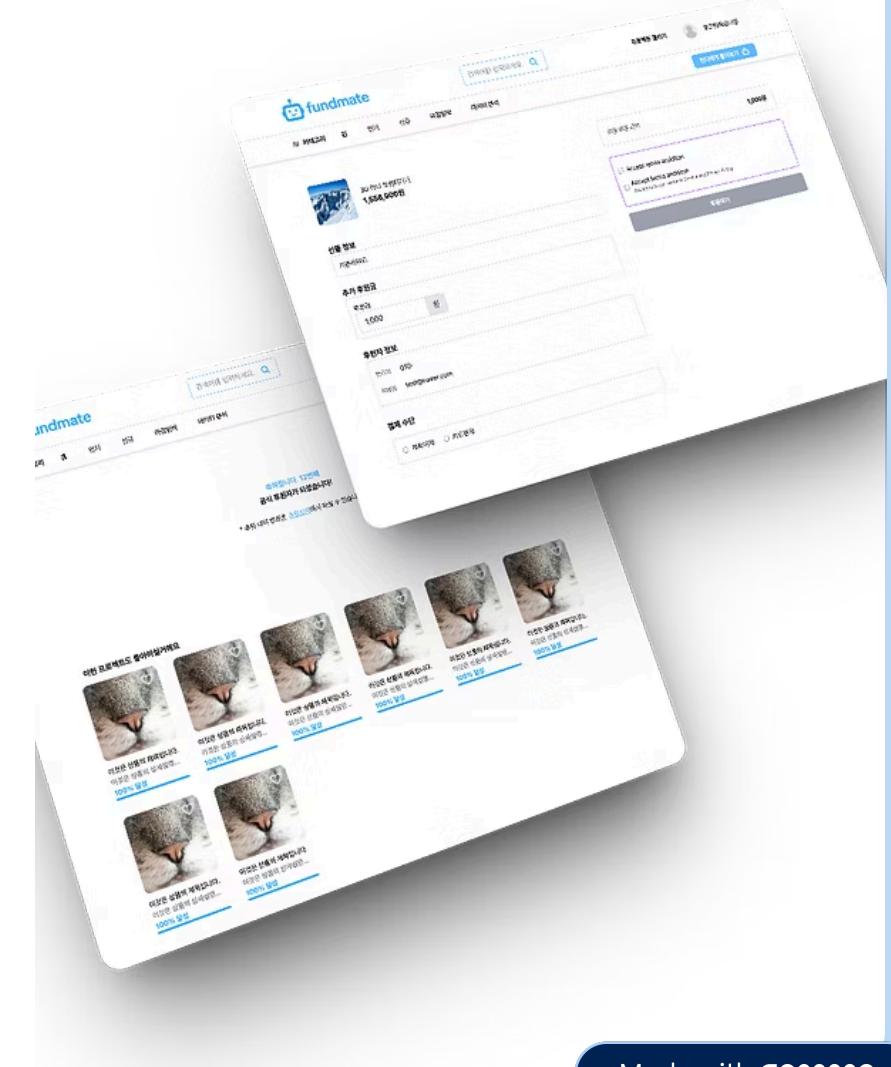
- 결제 서비스 UI 및 상태 관리 로직 구현

기술 스택

- React / TailwindCSS
- Zustand
- TanStackQuery
- Jest / MSW

주요 구현 기능

- 결제 수단 관리, 결제 테스트 환경 구성
- UX 개선을 위한 낙관적 업데이트 및 반응형 적용
- 커스텀 툴, 유тиль 함수 활용한 코드 리팩토링



김성윤 - FE 개발

역할

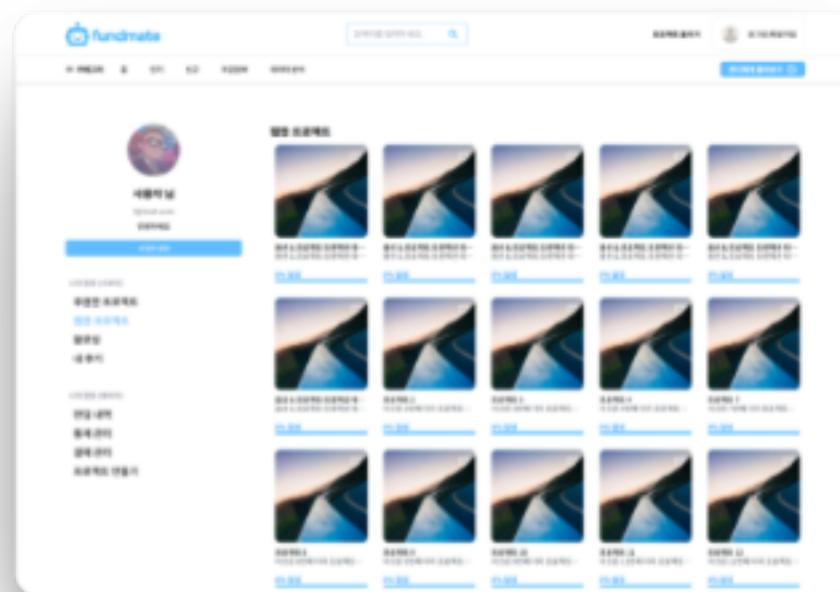
마이페이지 및 관리자페이지 UI 제작 및 API 연동

- 주요 구현 기능

- 서포터/관리자 마이페이지 구성
 - 통계(nivo), 결제관리, 정보 수정 등 UI 제작
 - 공용 사이드바 컴포넌트 제작

기술 스택

React , TypeScript, Tailwind, Zustand, Recharts, Micro Frontend, Module Federation



김지성 - 백엔드 인프라 & Gateway/결제 서버 개발

역할

- 인프라 구축, MSA 구조 설계, API Gateway & Payment 서버 구현

주요 구현 기능

- 무중단 배포, CI/CD 설정, 공통 모듈/엔티티 정리
 - Presigned URL API, 로깅, Health Check
 - 결제 스케줄/히스토리/통계 API 설계 및 구현

기술 스택

- AWS (EC2, RDS, S3), Nx, Github Actions, Swagger
 - Pino 로거, JWT, @shared 구조 설계

```
[09:49] INFO: [SUCCESS] (304) GET: /users/myp  
[09:49] INFO: [REQUEST] (---) GET: /users/myp
```

```
[09:49] INFO: [SUCCESS] (304) GET: /users/myprofile[ ]=8&project_id[ ]=9&project_id[ ]=10&project_id[ ]=11  
[09:49] INFO: [REQUEST] (---) GET: /users/myprofile[ ]=8&project_id[ ]=9&project_id[ ]=10&project_id[ ]=11
```

[09:49] INFO: [REQUEST] (---) GET: /users/myp

[09:49] INFO: [REQUEST] (---) GET: /users/myp

```
[09:49] INFO: [SUCCESS] (304) GET: /users/myp  
[09:49] INFO: [REQUEST] (---) GET: /users/myp  
id[ ]=8&project_id[ ]=9&project_id[ ]=10&project  
query={"project_id": ["1", "2", "3", "4", "5", "6",  
"7", "8", "9", "10"]}
```

```
[09:49] INFO: [SUCCESS] (304) GET: /users/myp  
id[ ]=8&project_id[ ]=9&project_id[ ]=10&project  
[09:49] INFO: [REQUEST] (---) GET: /users/myp
```

```
[09:49] INFO: [SUCCESS] (304) GET: /users/myp  
[09:49] INFO: [REQUEST] (---) GET: /users/myp
```

```
[09:49] INFO: [SUCCESS] (304) GET: /users/myprofile  
[09:50] INFO: [REQUEST] (---) GET: /users/like
```

```
[09:50] INFO: [SUCCESS] (304) GET: /users/lik  
[09:50] INFO: [REQUEST] (---) GET: /users/lik
```

[09:50] INFO: [SUCCESS] (304) [GET](#) /users/111
Before: [Error](#) After: [Success](#) Made with [Genoonee](#)

김예진 - 백엔드 AI/인터렉션 서버 개발

역할

- AI 서버 API 구현
- 인터렉션 서버 API 구현

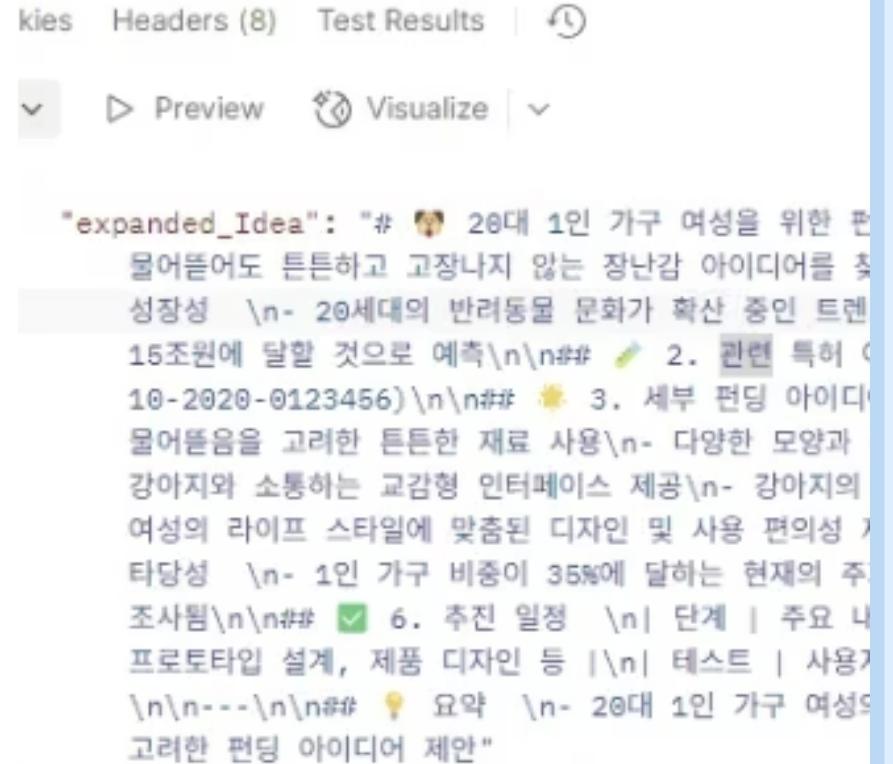
주요 구현 기능

- Grok 기반 AI 요약/확장 API 설계 및 파라미터 조정
- 사용자 타겟 기반 응답 설계
- 좋아요/후기 생성·조회·삭제 API
- JWT 인증 처리 및 TypeORM 활용

기술 스택

- Express, TypeORM, Grok AI, JWT, RESTful API

```
"input_text" : "20세대들을 위한 반려동물 장난감에 대한  
찾고싶어",  
"category" : "반려동물",  
"gender" : "여성",  
"age_group" : "20대",
```



이정은 - 백엔드 인증/유저 서버 개발

역할

- Auth 서버 API 설계/구현
- User 서버 API 설계/구현

주요 구현 기능

- 회원가입, 로그인, 이메일 인증, 소셜 로그인 API
- 마이페이지 및 프로필 정보, 회원 탈퇴, 팔로우/언팔로우 API
- 타 서버에서 데이터 수집 및 통합 응답 처리

기술 스택

- Express, TypeORM, JWT, OAuth, nodemailer
- 쿠키 기반 JWT 토큰 처리

```
✓ export const googleCallBack = async (req: Request, res: Response) => {
  const code = req.query.code;
  if (!code) {
    return res.status(HttpStatus.BAD_REQUEST).end();
  }

  const userRepo = AppDataSource.getRepository(User);
  const tokenRepo = AppDataSource.getRepository(Token);

  try {
    const tokenResponse = await axios.post('https://accounts.google.com/o/oauth2/token', {
      params: {
        code,
        client_id: process.env.GOOGLE_CLIENT_ID,
        client_secret: process.env.GOOGLE_CLIENT_SECRET,
        redirect_uri: process.env.GOOGLE_REDIRECT_URI,
        grant_type: 'authorization_code',
      },
      headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
    });

    const { access_token } = tokenResponse.data;

    const profileResponse = await axios.get(`https://www.googleapis.com/oauth2/v3/userinfo`, {
      headers: {
        Authorization: `Bearer ${access_token}`,
      },
    });

    const { id: snsId, email, name: nickname } = profileResponse.data;
    const provider = 'google';
    const user = await userRepo.findOne({ where: { snsId } });
    const token = await tokenRepo.create({
      provider,
      user,
      accessToken: access_token,
      refreshToken: tokenResponse.data.refresh_token,
    });
    await tokenRepo.save(token);
    await userRepo.save(user);
    res.cookie('refreshToken', token.refreshToken, { maxAge: 1000 * 60 * 60 * 24 * 30, httpOnly: true });
    res.end();
  } catch (err) {
    console.error(err);
    res.status(HttpStatus.INTERNAL_SERVER_ERROR).end();
  }
}
```

이하은 - 백엔드 펀딩/공공데이터 서버 개발

역할

- Funding 서버 API 설계/구현
- 공공 데이터 서버 API 설계/구현

주요 구현 기능

- 메인/사용자별 펀딩 목록 구현
- 펀딩 & 옵션 CRUD 구현
- 공공데이터 키워드/옵션별 조회 API

기술 스택

- Express, TypeORM, REST API 설계
- 사용자 맞춤형 데이터 응답 구성

```
s > funding-service > src > controller > FundingController.ts > [e] createFunding
1 import { Request, Response } from 'express';
2 import { AppDataSource } from '../data-source';
3 import { Project, OptionData } from '@shared/entities';
4 import { HttpStatusCode } from 'axios';
5 import { requestBodyValidation } from '../modules/request-validation';
6 import { addLikedStatusToQuery } from '../modules/query-builder';

7
8 // 프로젝트 생성
9 export const createFundingAndOption = async (req: Request, res: Response) => {
10   const { userId } = res.locals.user;
11
12   const values = [
13     req.body,
14     ...req.query,
15   ];
16
17   if (!requestBodyValidation(values)) {
18     return res.status(HttpStatusCode.BadRequest).json({
19       errors: requestBodyValidation(values),
20     });
21   }
22
23   const queryRunner = AppDataSource.createQueryRunner();
24   const optionRepo = queryRunner.manager.getRepository(OptionData);
25   const fundingRepo = queryRunner.manager.getRepository(Project);
26
27   await queryRunner.connect();
28   await queryRunner.startTransaction();
29
30   try {
31     const newFunding: Project = fundingRepo.create({
32       ...values[0],
33       optionId: values[1].optionId,
34       imageUrl: imageUrl,
35     });
36
37     const newOption: OptionData = optionRepo.create({
38       ...values[1],
39       fundingId: newFunding.id,
40     });
41
42     await queryRunner.commitTransaction();
43
44     return res.json({
45       funding: newFunding,
46       option: newOption,
47     });
48   } catch (error) {
49     await queryRunner.rollbackTransaction();
50     console.error(error);
51
52     return res.status(HttpStatusCode.InternalServerError).json({
53       message: 'Internal Server Error',
54     });
55   }
56 }
```

시연 영상

프론트엔드 트러블슈팅

1

MFE 환경 혹은 오류

vite.config.ts에서 라이브러리를 명시적으로 공유해 해결

2

headlessui 드롭다운 충돌

라이브러리 제거 후 드롭다운 리팩토링으로 해결

3

tailwind 스타일 깨짐

각 프로젝트의 tailwind.config.js에 동일 설정 적용

4

라우터 컴포넌트 공유 오류

라우터 미포함 페이지단위 컴포넌트로 재구성

5

MSW 404 오류

프로덕션 환경에서도 동작하도록 설정 변경

백엔드 트러블슈팅



Docker/Nx 경로 문제

project.json에 bundle: true 설정 추가, tsconfig.base 경로 수정



서버 수정 지연

Blue/Green 방식의 무중단 배포로 해결



EC2 디스크 용량 부족

docker system prune -a --volume 명령어로 리소스 정리



JWT 토큰 전달 문제

API Gateway에서 토큰 해석 후 사용자 정보 수동 전달

KPT 회고

✓ Keep - 잘한 점

- 프론트/백엔드 간 원활한 소통 및 문서화
- 명확한 PR 기준과 효율적인 코드 리뷰
- 도전적인 기술 시도와 문서 반영 속도
- 효율적인 Git 브랜치 관리



⚠ Problem - 아쉬웠던 점

- 완성과 경험 사이의 목표 혼란
- 담당자 명시 부족 및 역할 분배 모호함
- AI/공공데이터 기능의 활용도 부족
- 기능 우선순위 설정과 구현 계획 부족



💡 Try - 시도할 것

- 공용 컴포넌트 명확히 정의 후 디자인
- 아키텍처 사전 학습 후 레포 구조 기획
- 기술 공유 시간 마련 및 리팩토링 강화
- 인프라 학습 및 실제 적용

향후 계획



코드 리팩토링

중복 코드 제거 및 구조 개선



기능 개선 및 확장

사용자 피드백 기반 기능 확장



테스트코드 작성

안정성 확보를 위한 테스트 강화



반응형 디자인

프론트엔드 반응형 디자인 적용