애자일 기반 건강관리 서비스 개발 프로젝트

2025 PORTFOLIO

CONTACT

yunhoyom@gmail.com

010 2303 1480

Information

염윤호 / Yunho Yeom

1993.11.03

Tel. 010-2303-1480

Email. yunhoyom@gmail.com

서울특별시 금천구 가산동

GITHUB

https://github.com/yunhoop/kosta-project

GRADUATION

2012~2016 청운대학교 인터넷학과 중퇴

2016~2020 나사렛대학교 중등특수교육과 졸업

CERTIFICATE

2025.03. 정보처리기사(필기)

2020.02. 특수학교(중등) 정교사 2급(음악)

1. Spring Boot 기반 1차 스프린트

- 기간/인원 : 2025. 05. 12. ~ 2025. 06. 01. (21일 / 5인)
- 주제 : Spring Boot 기반 건강관리 서비스
- 나의 역할
  - 프로젝트 리더
  - Jira 스크럼 관리
  - MyBatis, JPA 활용한 데이터 처리

2. REST API 기반 2차 스프린트

- 기간/인원 : 2025. 06. 16. ~ 2025. 06. 20. (5일/5인)
- 주제 : REST API 기반 건강관리 서비스 개발
- 나의 역할
  - REST API 기반 서버 분리 리팩토링
  - 버전 관리

3. MSA 기반 3차 스프린트

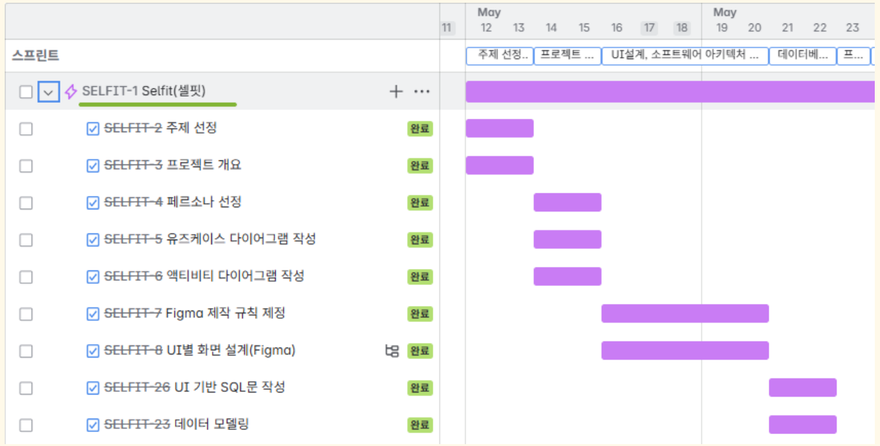
- 기간 : 2025. 06. 30. ~ 2025. 07. 15 (16일 / 5인)
- 주제 : MSA 리팩토링
- 나의 역할
  - Controller 기준 서비스 리팩토링
  - RestTemplate 활용한 서비스 간 통신

기술 스택

Backend	DB	Deployment
Java / Spring Boot MyBatis / JPA	Oracle Maria DB	AWS EC2 Docker
Frontend	**Tools / Test Code**	**Collaborations**
HTML, CSS, Javascript jQuery, axios	IntelliJ /Eclipse Junit5 Postman / Swagger	GitHub Jira / Confluence Figma eXERD

애자일 방법론 기반의 프로젝트 운영

SCRUM(관리방법론)



- Jira 기반 스크럼 환경 조성
- 일일 스크럼 회의를 통한 진행 점검
- KPT 회고를 통한 개선점 도출

XP(개발 방법론)

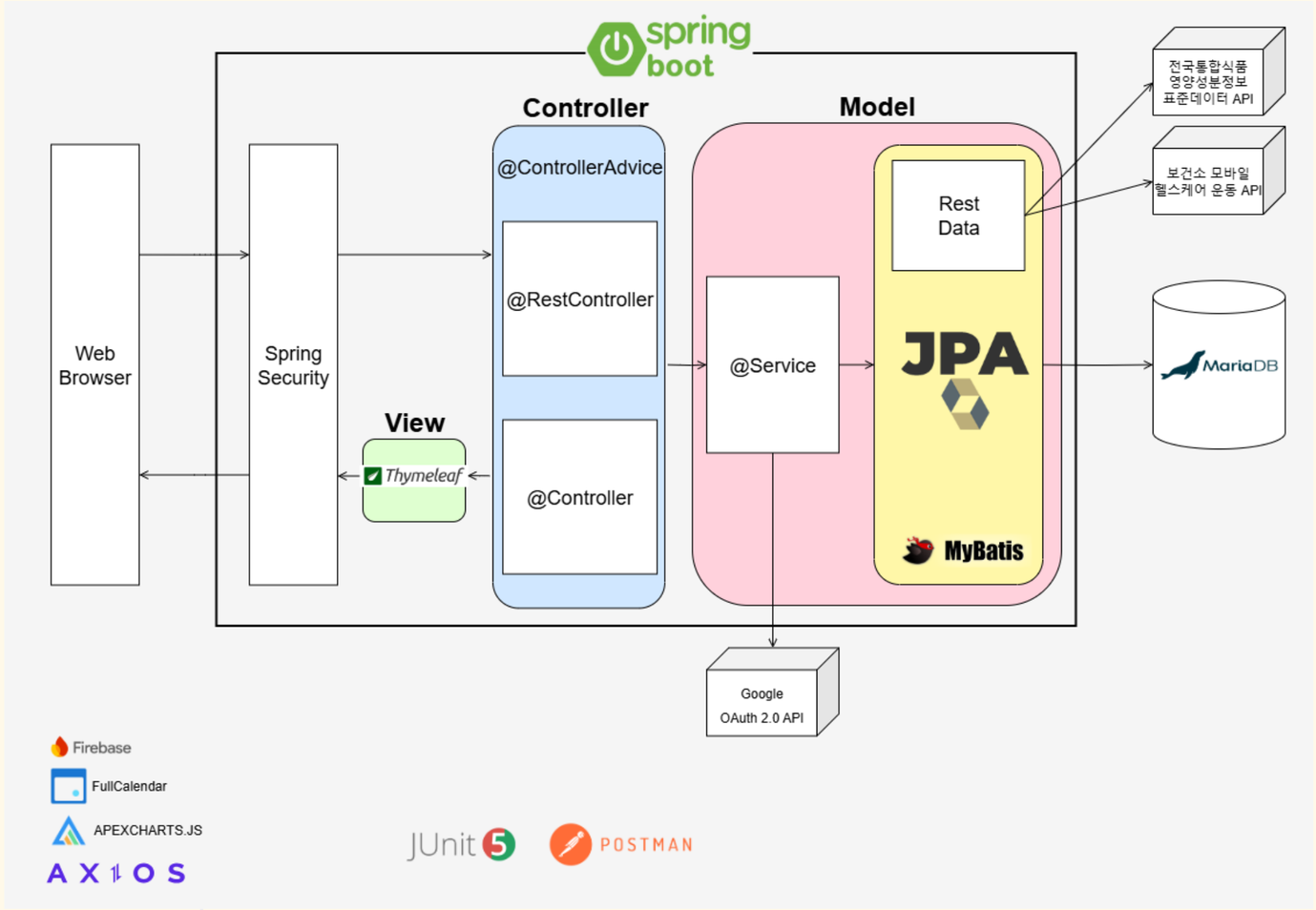
- Rest API, MSA로의 리팩토링
- JPA 기술에 대한 Pair Programming
- Pull Request를 통한 코드 리뷰

Sprint 01

SPRING BOOT 기반 건강관리 서비스

1) 업무 흐름

2) 회고 및 보완 과제



Selfit

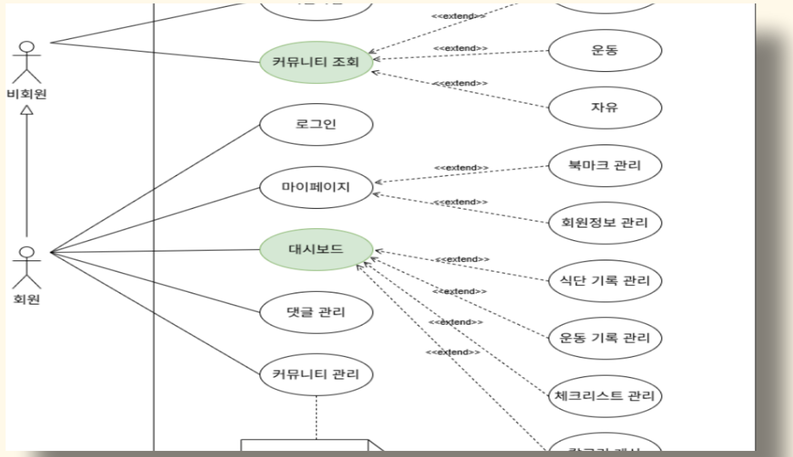
스스로 **운동을 계획하고 기록**하며,
소통을 통해 건강한 습관을 만들어 가는 **개인 건강관리 플랫폼**

1) 업무 흐름

사이트	주요 기능	특징
Cronometer	섭취·운동 기록, 일일 요약, 물 섭취량 체크	개인 트래킹 집
몬스터짐	카테고리별 글 작성·검색, 쇼핑몰 연동	활발한 커뮤니
다이어트신	칼로리 계산, 커뮤니티	기본 기능 중심
Examine	성분별 검색, 질환별 추천	정보 제공 집중
drugs.com	약 비교, 부작용, 성분표	전문 정보 제공
쿠스	원료 검색, 포럼	커뮤니티 연계

벤치마킹

주제가 유사한 여러 사이트들의 장단점 파악 후, 우리만의 정체성 파악



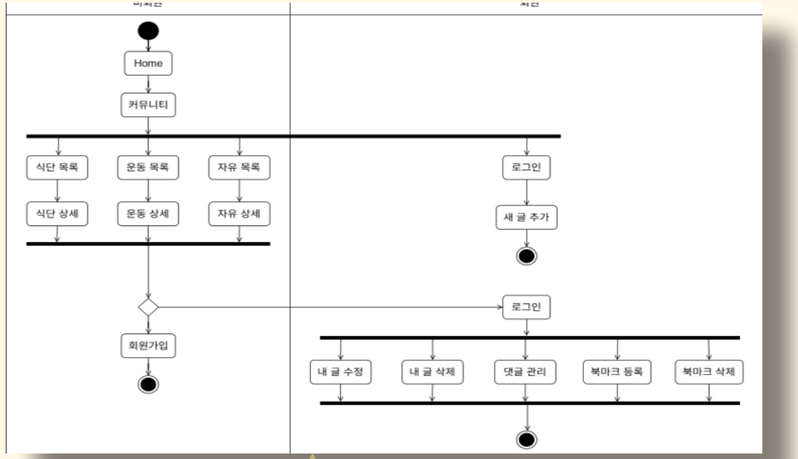
컨셉모델

요구에 기반한 기능 정의 (커뮤니티, 대시보드)
<유즈케이스다이어그램>

페르소나

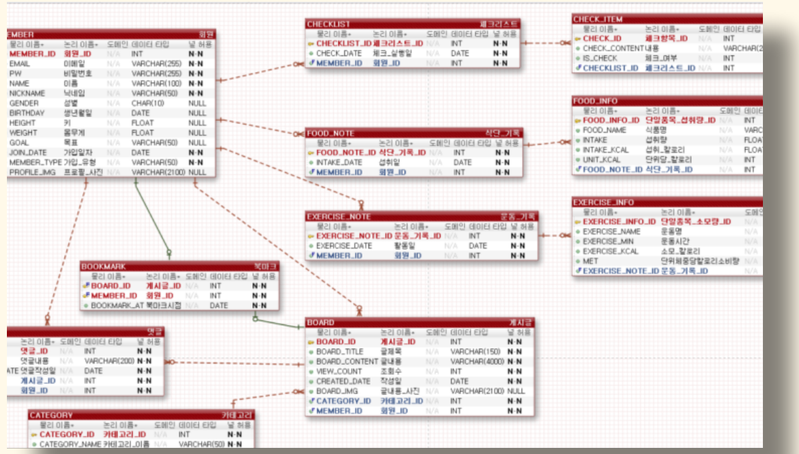
업무에 대한 3명의 페르소나 도출

1) 업무 흐름



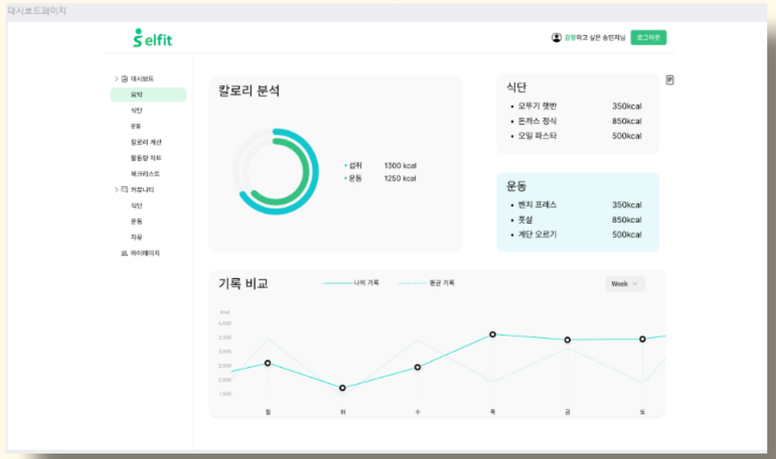
워크 플로

2가지 업무에 대한
기능별 흐름 파악
<액티비티다이어그램>



데이터모델링

데이터모델링을 통한
DB 구조 설계
<eXERD>



UI 프로토타입

기능별 UI 프로토타입
제작
<Figma>

2) 회고 및 보완과제

회고

DI/IoC를 통해 메모리의 효율성 문제를 해결할 수 있었다. 이전 MVC Model2 프로젝트에서는 new로 모든 객체를 직접 생성했다. 그 결과 Eclipse의 속도가 저하되는 모습을 보며 메모리가 과도하게 사용되고 있다는 것을 체감했다. Spring에서는 DI/IoC를 통해 필요한 시점에 한 번만 객체를 생성하고 재사용하니 메모리가 효율적으로 관리되고 있음을 느꼈다.

보완 과제

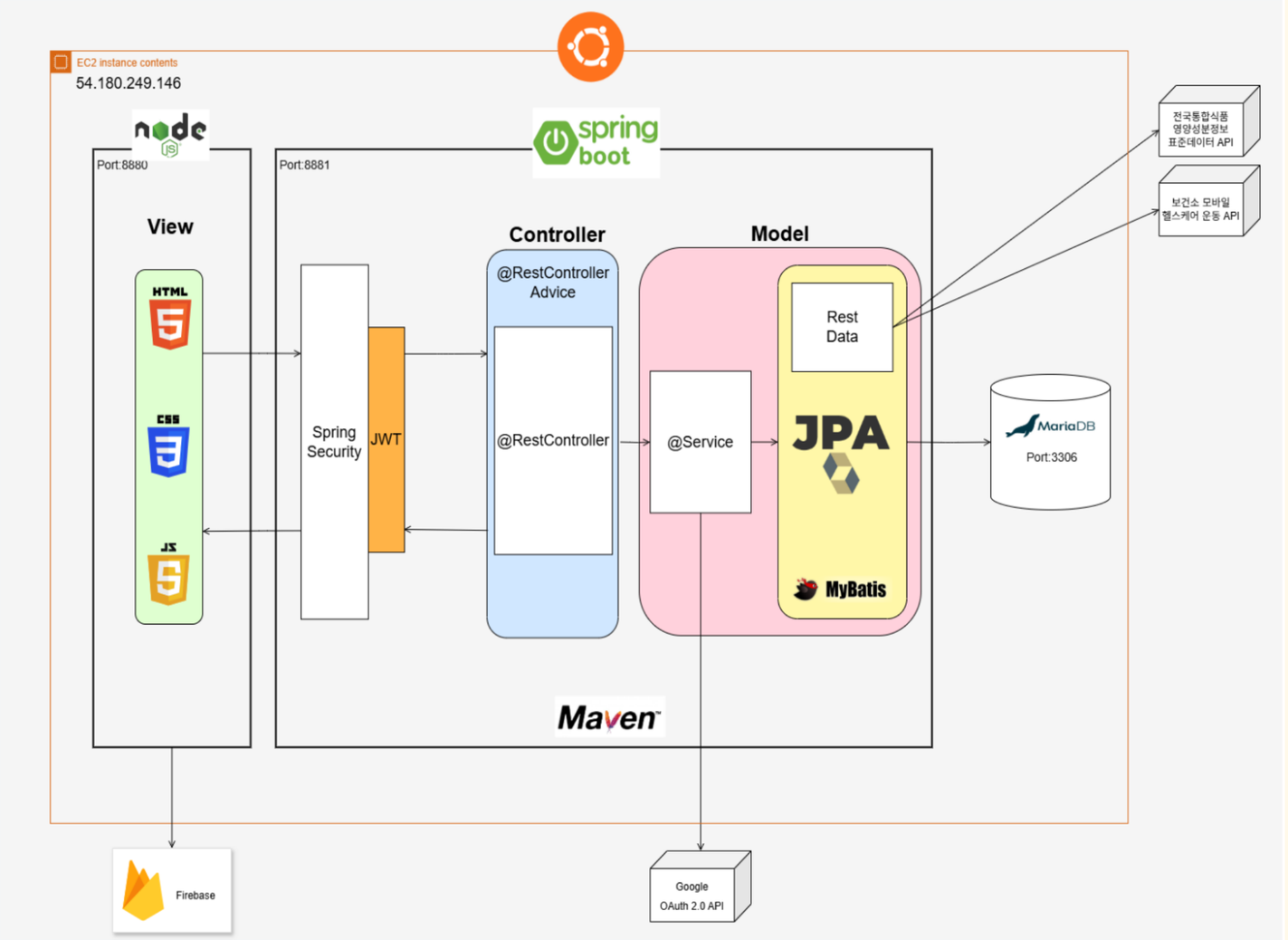
- 기능에 문제가 생기면 서버 전체 코드를 점검해야 함.
- Thymeleaf를 사용할 때 @Controller에서 반환할 HTML을 넣어주며 Back과 Front가 강하게 연관됨.

Sprint 02

REST API 서비스 리팩토링

1) 1차 스프린트 회고

2) 회고 및 보완 과제



1) 1차 스프린트 회고(KPT 회고)

Problem(문제점)	Try(시도)
코드의 일관성 부족	관련 업무 담당자에게 Pull Request 지정
수정 사항 및 공유사항에 대한 소통 부족	Confluence 회의록 및 공유 템플릿 활용
명확하지 않은 업무 분담	Jira 보드 내 하위 업무 분담 활성화

Keep(유지)

- 하루 2회 10분씩 일일 회의를 통한 진행상황 점검
- Jira를 통한 일정 및 업무 관리
- GitHub 커밋 컨벤션을 통한 협업 효율성 강화

2) 회고 및 보완 과제

회고

BackEnd - FrontEnd 서버를 나누어 역할을 명확히 구분하였다. 지난 스프린트에서는 Controller에 HTML 정보가 포함되며 BackEnd - FrontEnd가 강한 연관을 가지고 있었다. 이번 REST API 서비스 리팩토링을 통해 BackEnd에서는 JSON 값만 Response하고, FrontEnd는 데이터를 받아 화면을 구성한다. 그 결과 서버 간의 결합도가 느슨해지는 것을 볼 수 있었다.

보완 과제

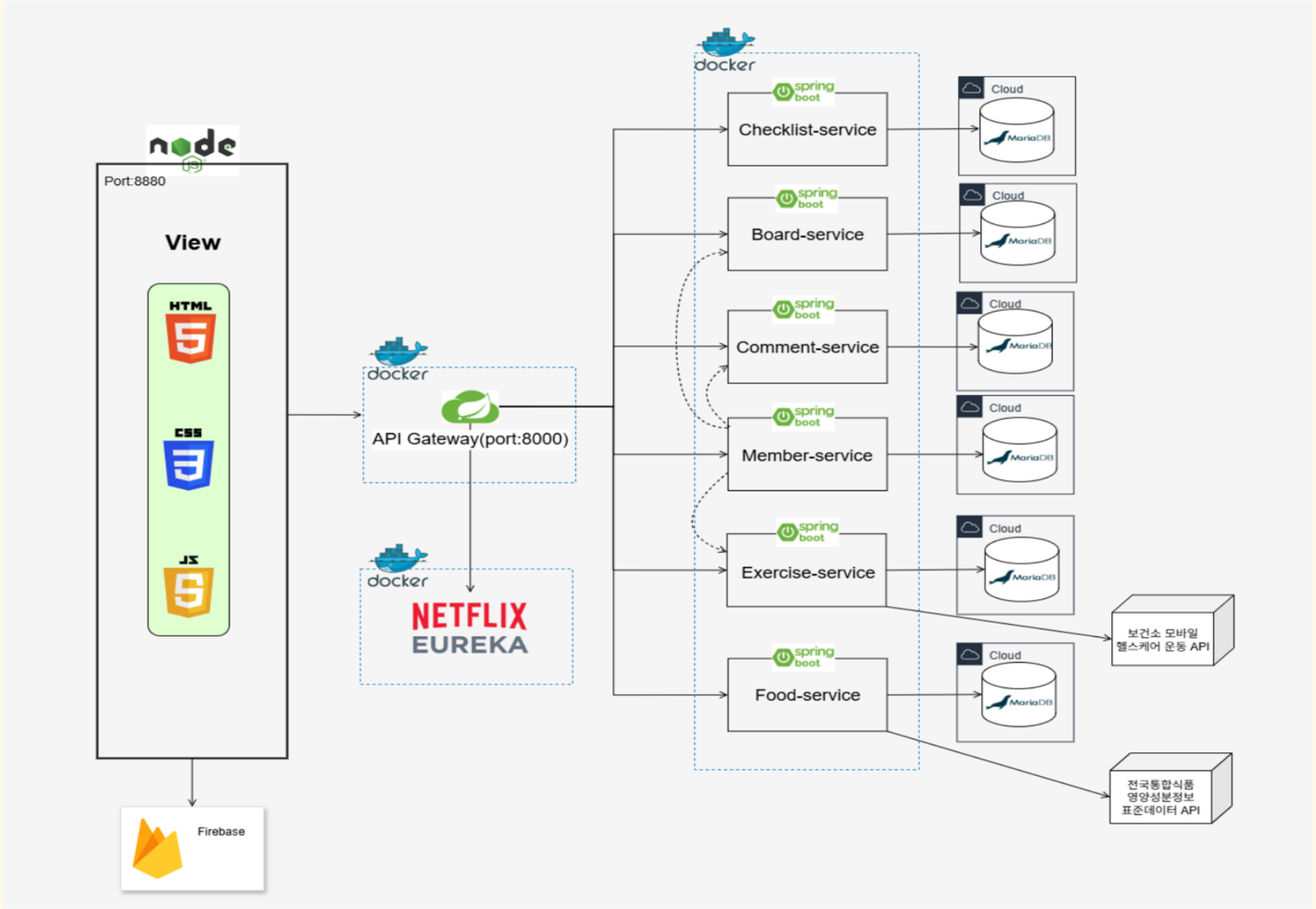
- BackEnd 서버는 여전히 하나라 협업 중 동시 개발이 어려움.
- 작은 수정에도 BackEnd 서버 전체를 다시 빌드해서 재배포를 해야 함.

Sprint 03

REST API 서비스 리팩토링

1) 최종 산출물

2) 회고 및 보완 과제



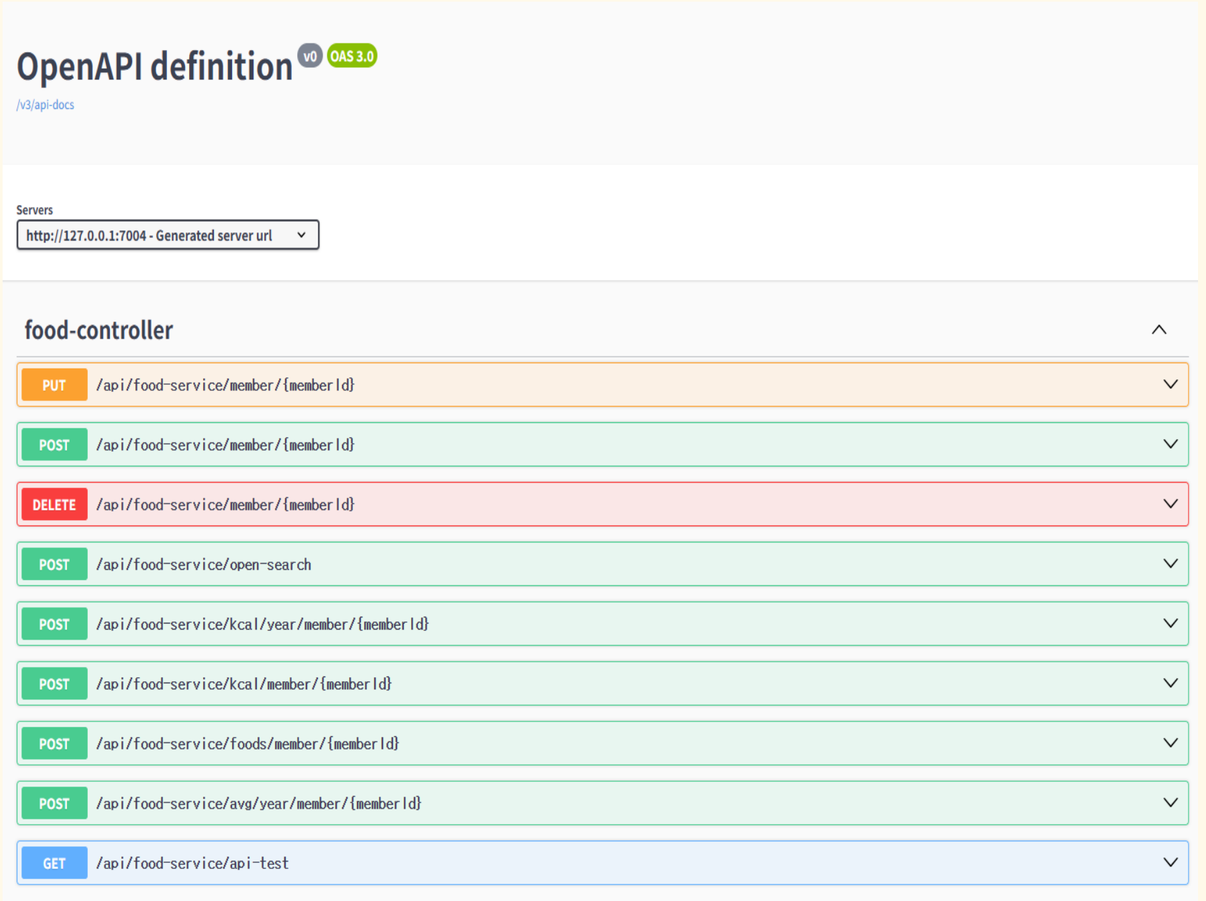
1) 최종 산출물

- 업무상세설계서

exercise-service

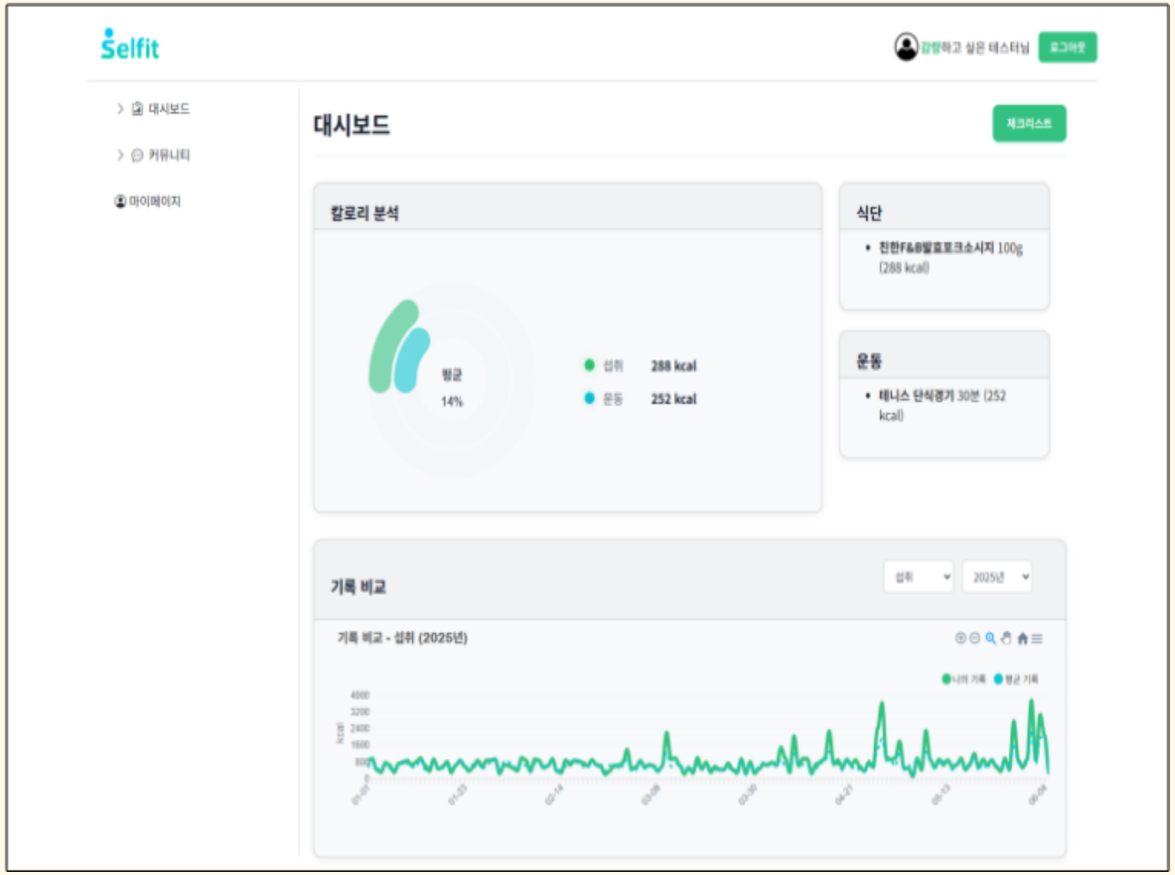
번호	업무	흐름	URL	종류	req	res
1	일별 운동칼로리 조회	칼로리 계산에서 해당 날짜를 클릭하면 소모칼로리 조회	/api/exercise-service/kcal/member/{memberId}	POST	@PathVariable String memberId @RequestBody ReqGetExerciseKcal reqGetExerciseKcal	ResponseEntity <ResGetExerciseKcal>
2	연도별 운동칼로리 조회	대시보드 운동에서 운동그레프에 연도별 소모칼로리 차트 출력	/api/exercise-service/year/member/{memberId}	POST	@PathVariable String memberId @RequestBody ReqGetYearExerciseKcal reqGetYearExerciseKcal	ResponseEntity <List<ResGetYearExerciseKcal>>
3	운동 항목 추가 (Member-Weight필요)	운동 상세에서 운동 항목 추가	/api/exercise-service/member/{memberId}	POST	@PathVariable String memberId @RequestBody ReqAddExercise reqAddExercise	ResponseEntity <ResMessage>
4	운동 상세 조회	운동 상세에서 항목들 조회	/api/exercise-service/exercises/member/{memberId}	POST	@PathVariable String memberId @RequestBody ReqGetExercises reqGetExercises	ResponseEntity <List<ResGetExercises>>
5	운동 항목 수정(운동시간) Member-Weight 필요	운동 상세에서 수정 클릭 후 운동 시간 수정	/api/exercise-service/member/{memberId}	PUT	@PathVariable String memberId @RequestBody ReqSetExerciseMin reqSetExerciseMin	ResponseEntity <ResMessage>
6	운동 항목 삭제	운동 상세에서 삭제 클릭 후 운동 항목 삭제	/api/exercise-service/member/{memberId}	DELETE	@PathVariable String memberId @RequestBody ReqRemoveExercise reqRemoveExercise	ResponseEntity <ResMessage>
7	운동 검색	운동 기록에서 패널을 열고 운동 검색	/api/exercise-service/open-search	POST	@RequestBody ReqExerciseOpenSearch reqExerciseOpenSearch	Response<Mono <List<ResExerciseOpenSearch>>>
8	연도별 운동 칼로리 평균 조회	대시보드 요약에서 운동 필터 누르기	/api/exercise-service/avg/year/member/{memberId}	POST	@PathVariable String memberId, @RequestBody ReqGetYearExerciseAvgAll reqGetYearExerciseAvgAll	ResponseEntity <List<ResGetYearExerciseAvgAll>>

- Swagger



1) 최종 산출물

- UI 산출물



- Docker

	Name	Container ID
☐	selfit-ui	4486fc1862da
☐	selfit-back-deploy	-
☐	discovery-service	d24535cfb104
☐	comment-service	528cb55a768f
☐	food-service	f066b7e898de
☐	apigateway-service	3d0da0906c2f
☐	checklist-service	a45dfbca6dca
☐	board-service	0d542ee20ecd
☐	member-service	098de7f81cc5
☐	exercise-service	32fa5fdbcdf8

2) 회고 및 보완 과제

성과 및 발전사항

MSA의 서비스 독립성이 협업에서 장점임을 느꼈다. 기존 한 서버에서 여러 명이 GitHub에 Push할 때 충돌이 생기는 경우들이 있었다. MSA에서는 각각 서비스를 나누어 개발하니 서로 영향을 주지 않고 안정적으로 병합할 수 있었다. 이 경험을 통해 MSA의 분리된 구조가 협업에 효과적임을 체감했다.

보완 과제

- 각 서버 별로 .env 파일이 흩어져, 통합된 Config Server를 도입하여 중앙에서 환경 설정 관리가 필요함을 느낌.
- 인스턴스 확장에 대한 근거 마련을 위해 모니터링 서비스 도입이 필요함을 느낌.