

애자일 기반 건강관리 서비스 개발 프로젝트

2025 PORTFOLIO

CONTACT

yunhoyom@gmail.com

010 2303 1480



Information

염운호 / Yunho Yeom

1993.11.03

Tel. 010-2303-1480

Email. yunhoyom@gmail.com

서울특별시 금천구 가산동

GRADUATION

2012~2016 청운대학교 인터넷학과 중퇴

2016~2020 나사렛대학교 중등특수교육과 졸업

CERTIFICATE

2025.03. 정보처리기사(필기)

2020.02. 특수학교(중등) 정교사 2급(음악)

GITHUB

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

1. Spring Boot 기반 1차 스프린트

- 기간 : 2025. 05. 12. ~ 2025. 06. 01.(21일)
- 인원 : 5인
- 주제 : Spring Boot 기반 건강관리 서비스 개발

2. REST API 기반 2차 스프린트

- 기간 : 2025. 06. 16. ~ 2025. 06. 20.(5일)
- 인원 : 5인
- 주제 : REST API 서비스 리팩토링

3. MSA 기반 3차 스프린트

- 기간 : 2025. 06. 30. ~ 2025. 07. 15(16일)
- 인원 : 5인
- 주제 : MSA 리팩토링

서비스 및 역할 소개

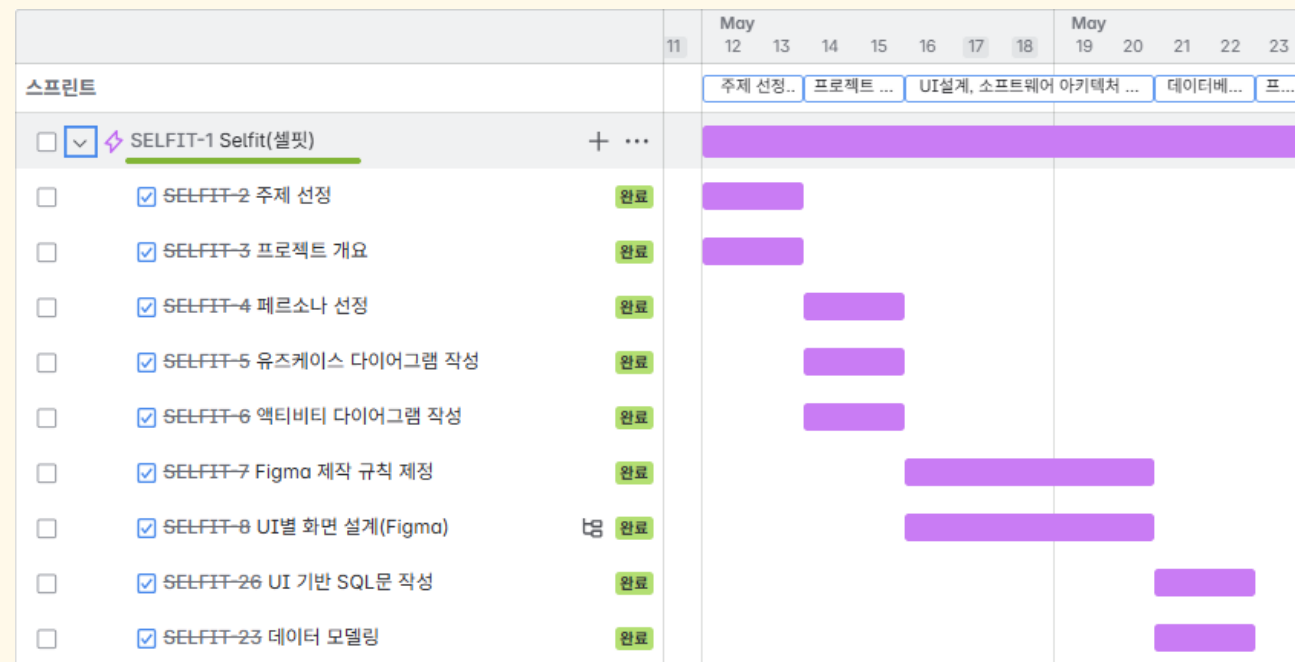
Selfit

스스로 운동을 계획하고 기록하며, 소통을 통해 건강한 습관을 만들어 가는 개인 건강관리 플랫폼

1차 스프린트	2차 스프린트	3차 스프린트
<ul style="list-style-type: none">- 프로젝트 리더- Jira 스크럼 관리- JPA 활용 운동 대시보드 데이터 처리 (Pair Programming)	<ul style="list-style-type: none">- REST API 기반 클라이언트 - 서버 분리- 버전 관리	<ul style="list-style-type: none">- 'exercise-service' 리팩토링- Rest Template 통한 서비스 간 데이터 조회

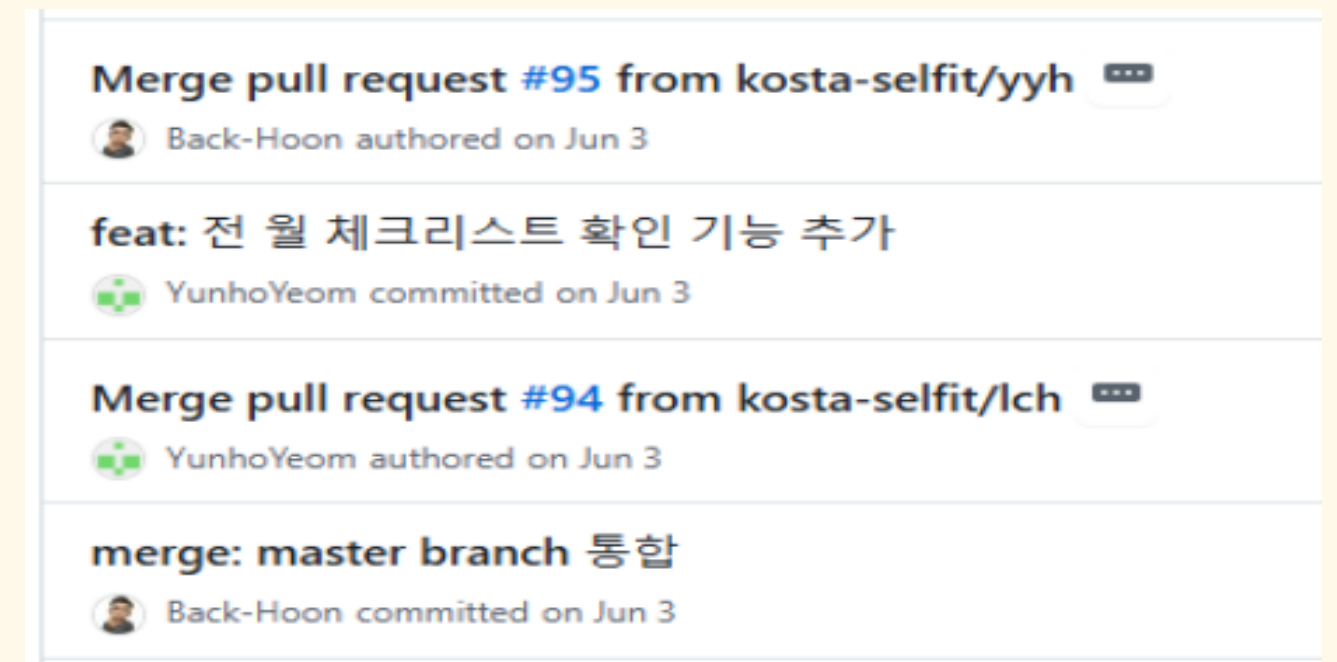
애자일 방법론 기반의 프로젝트 운영

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

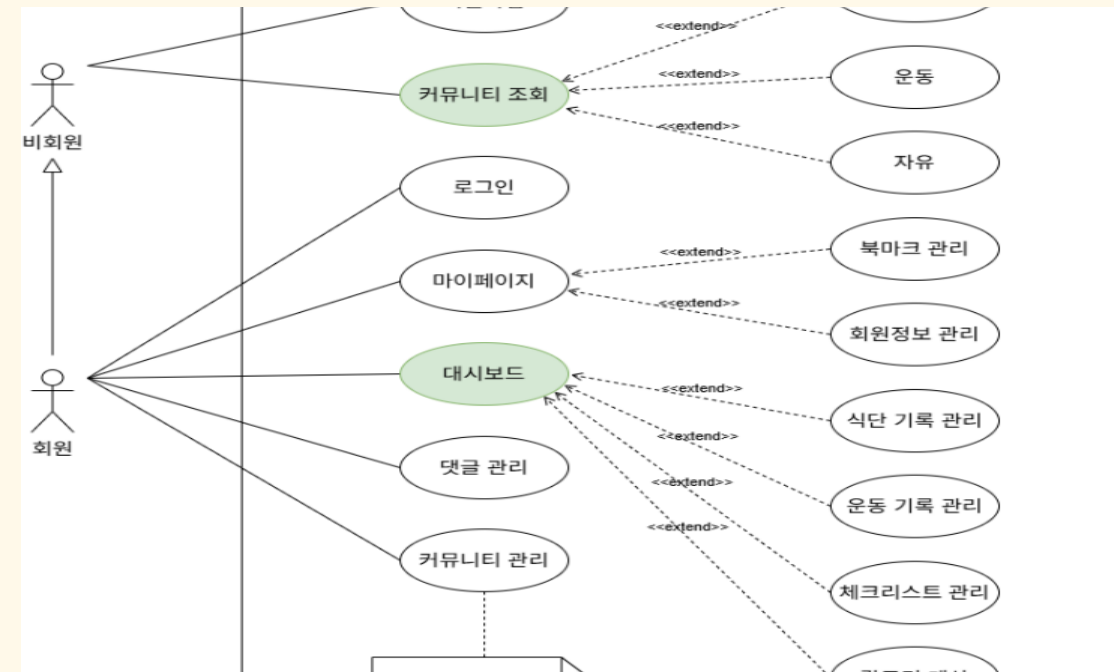
- 1) 업무 흐름
- 2) 소프트웨어 아키텍처
- 3) 회고

1) 업무 흐름

사이트	주요 기능	특징
Cronometer	섭취·운동 기록, 일일 요약, 물 섭취량 체크	개인 트레이킹 집
몬스터짐	카테고리별 글 작성·검색, 쇼핑물 연동	활발한 커뮤니티
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)
<유즈케이스다이아그램>

페르소나

이름

성별

나이

직업

사는곳

김종국

남자

39

헬스트레이너

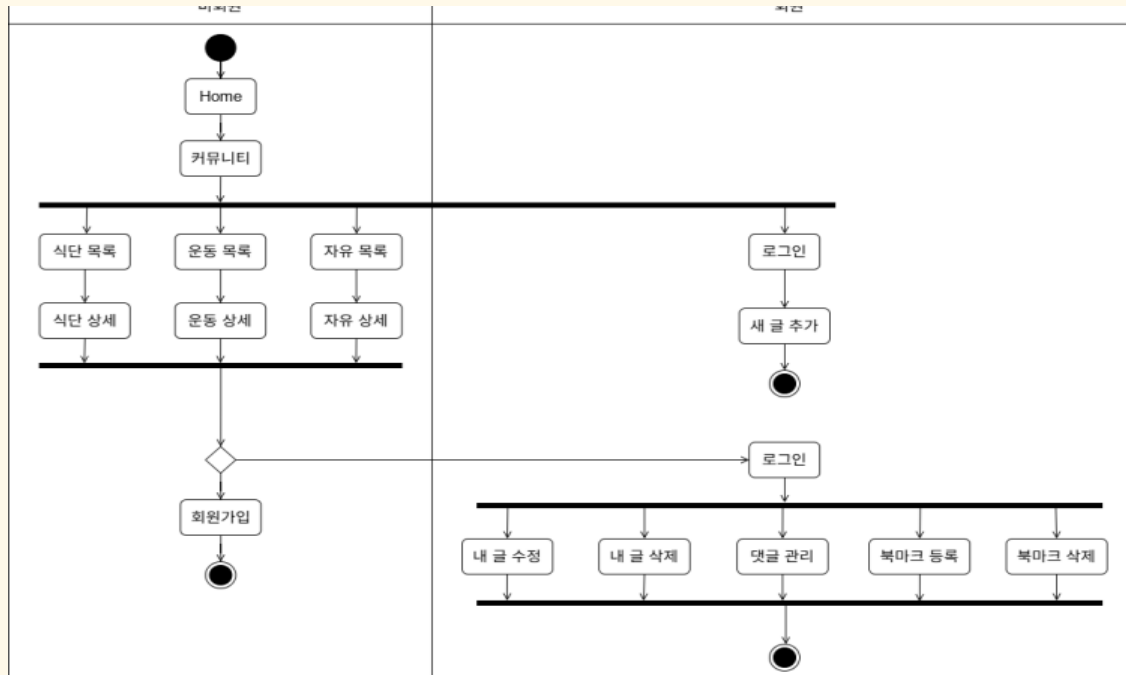
서울시 성북구

성향	다른 사람에게 조언하는 것을 좋아하고, 호기심이 많음.
니즈	커뮤니티를 통해 운동·식단 관리에 어려움을 겪는 사람들을 돕고 싶음.
문제점	질문을 받아도 개개인에 맞춘 피드백을 줄 정보가 부족해 맞춤형 조언 제공이 어려움.
주요기능	운동·식단 게시판, 질문 템플릿

페르소나

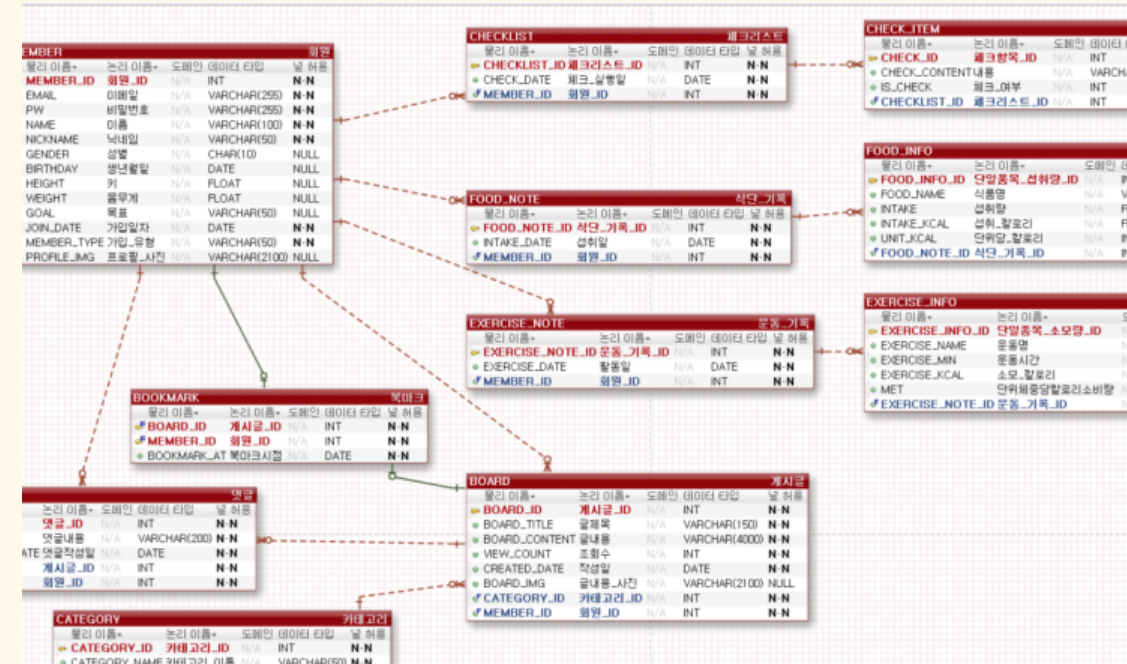
업무에 대한 3명의 페르소나 도출

1) 업무 흐름



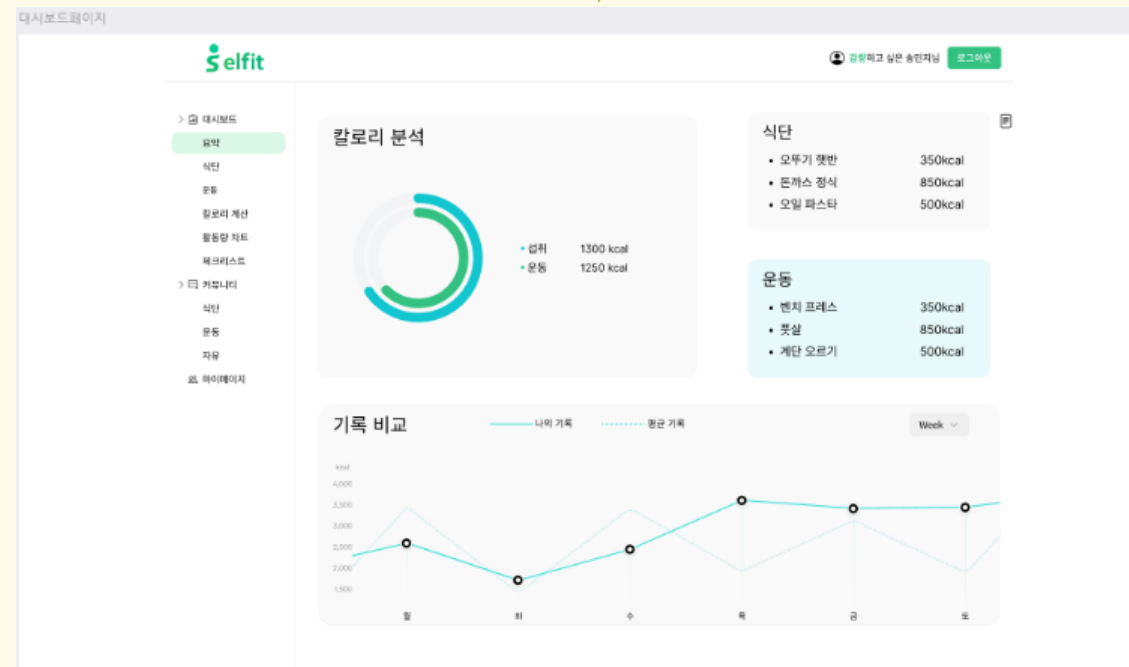
워크 플로

2가지 업무에 대한
기능별 흐름 파악
<액티비티 다이어그램>



데이터모델링

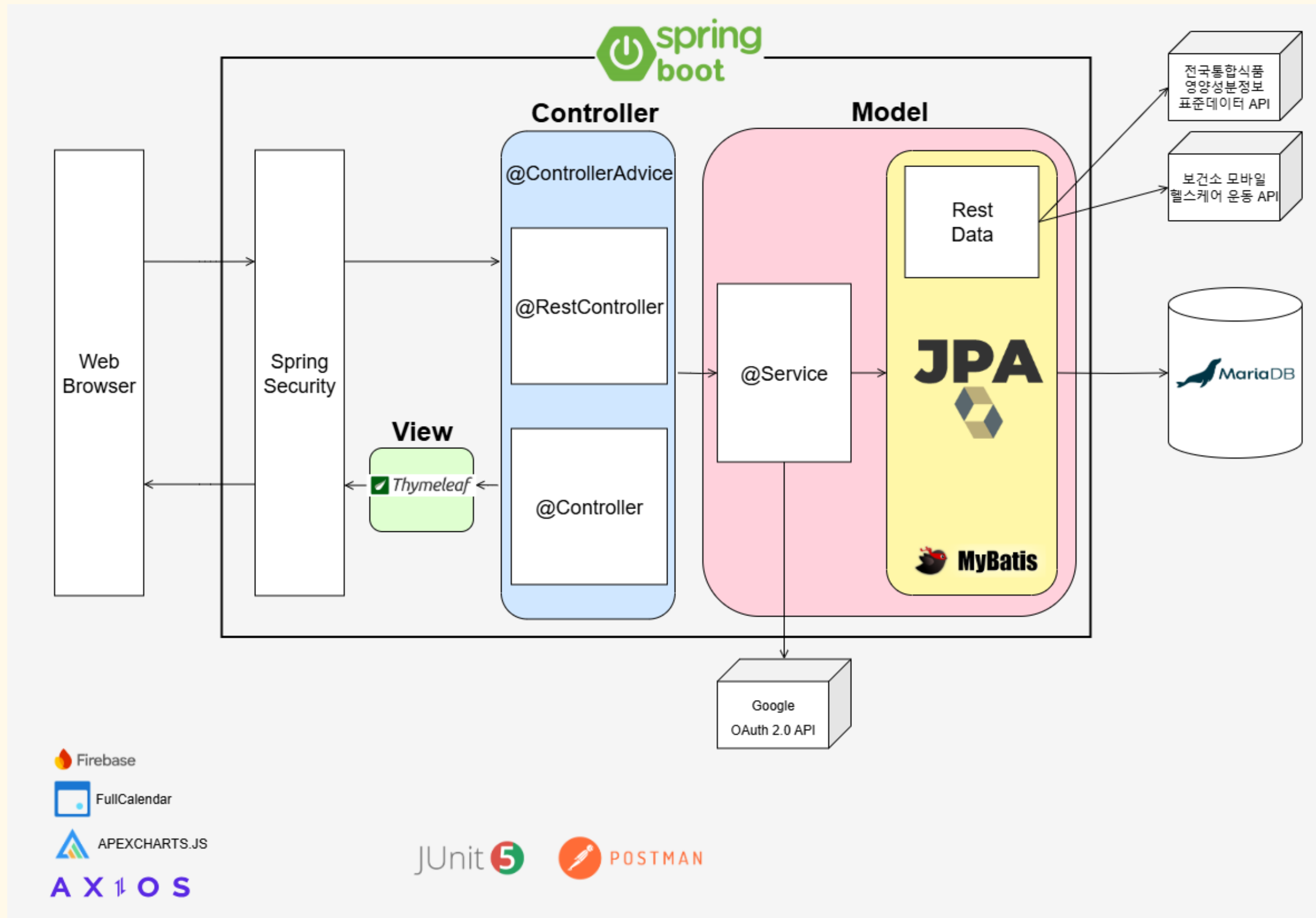
데이터모델링을 통한
DB 구조 설계
<eXERD>



UI 프로토타입

기능별 UI 프로토타입
제작
<Figma>

2) 소프트웨어 아키텍처



3) 회고

성과 및 발전사항

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. Spring Boot에서는 DI/IoC를 통해 필요한 시점에 한 번만 객체를 생성하고 재사용하니 메모리가 효율적으로 관리되고 있음을 느꼈다.

Spring Boot를 적용하며 DispatcherServlet을 내부적으로 관리하는 기능을 통해 안정적인 프로젝트를 진행할 수 있었다. 이전 프로젝트인 MVC Model2에서는 다른 조원들이 FrontController를 수정하면서 충돌이 생기는 경우가 있었다. DispatcherServlet을 감추는 Spring Boot에서는 충돌 없이 프로젝트가 진행될 수 있었다.

보완 과제

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

Sprint 02

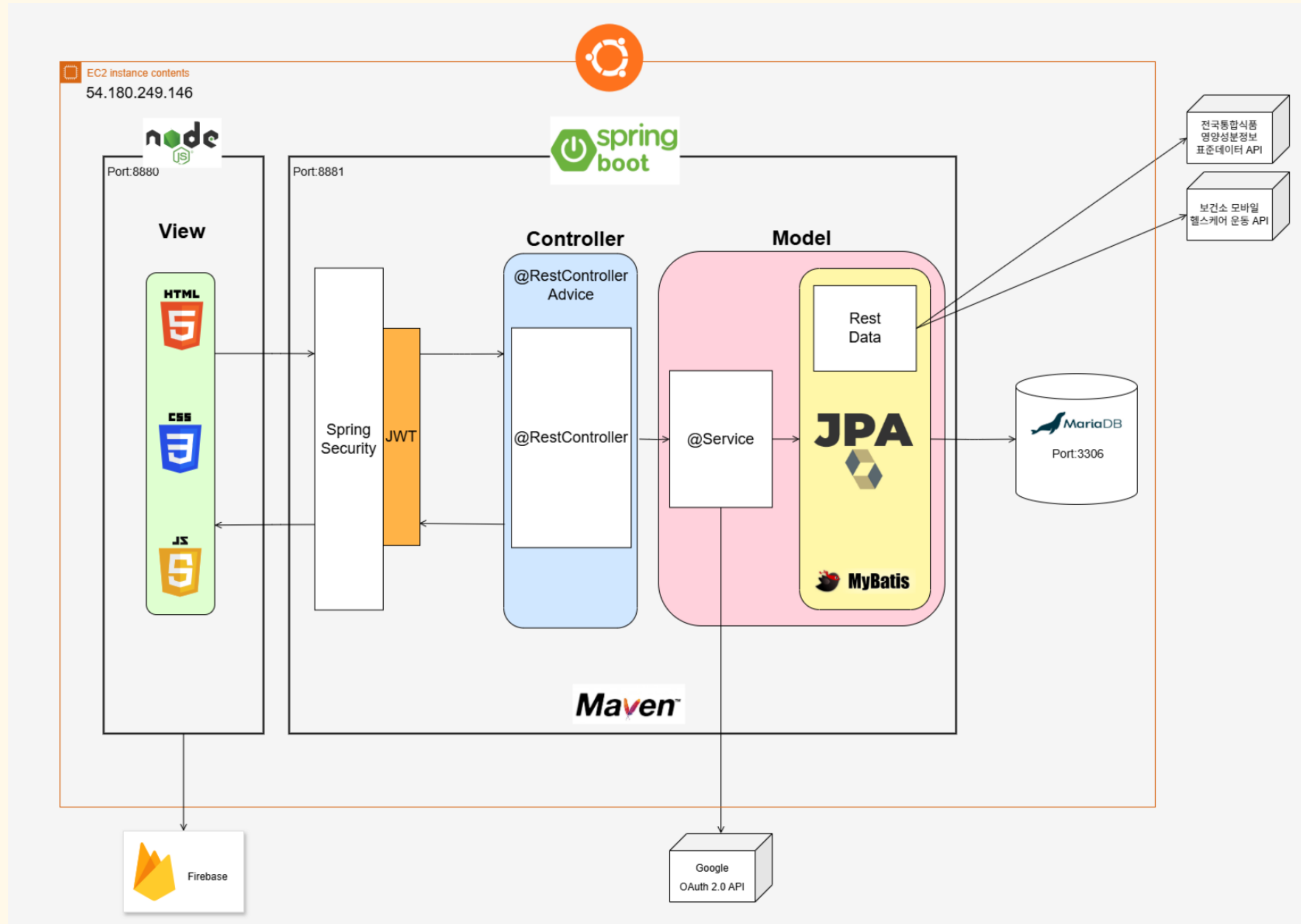
REST API 서비스 리팩토링

- 1) 1차 스프린트 회고
- 2) 소프트웨어 아키텍처
- 3) 회고

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)			
<ul style="list-style-type: none">하루 2회 10분씩 일일 회의를 통한 진행상황 점검Jira를 통한 일정 및 업무 관리GitHub 커밋 컨벤션을 통한 협업 효율성 강화			

2) 소프트웨어 아키텍처



3) 회고

성과 및 발전사항

REST API를 적용하여 디버깅을 수월하게 했다. 한 개의 서버로 운영되던 지난 스프린트에서는 하나의 코드에 문제가 생기면 서버의 모든 코드를 확인해야 하는 불편함이 있었다. REST API 기반 리팩토링 후에는 업무상세설계서의 Request, Response를 확인하여 Postman으로 응답 값만 검증해도 어느 서버에서의 문제인지 파악할 수 있게 되었다.

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

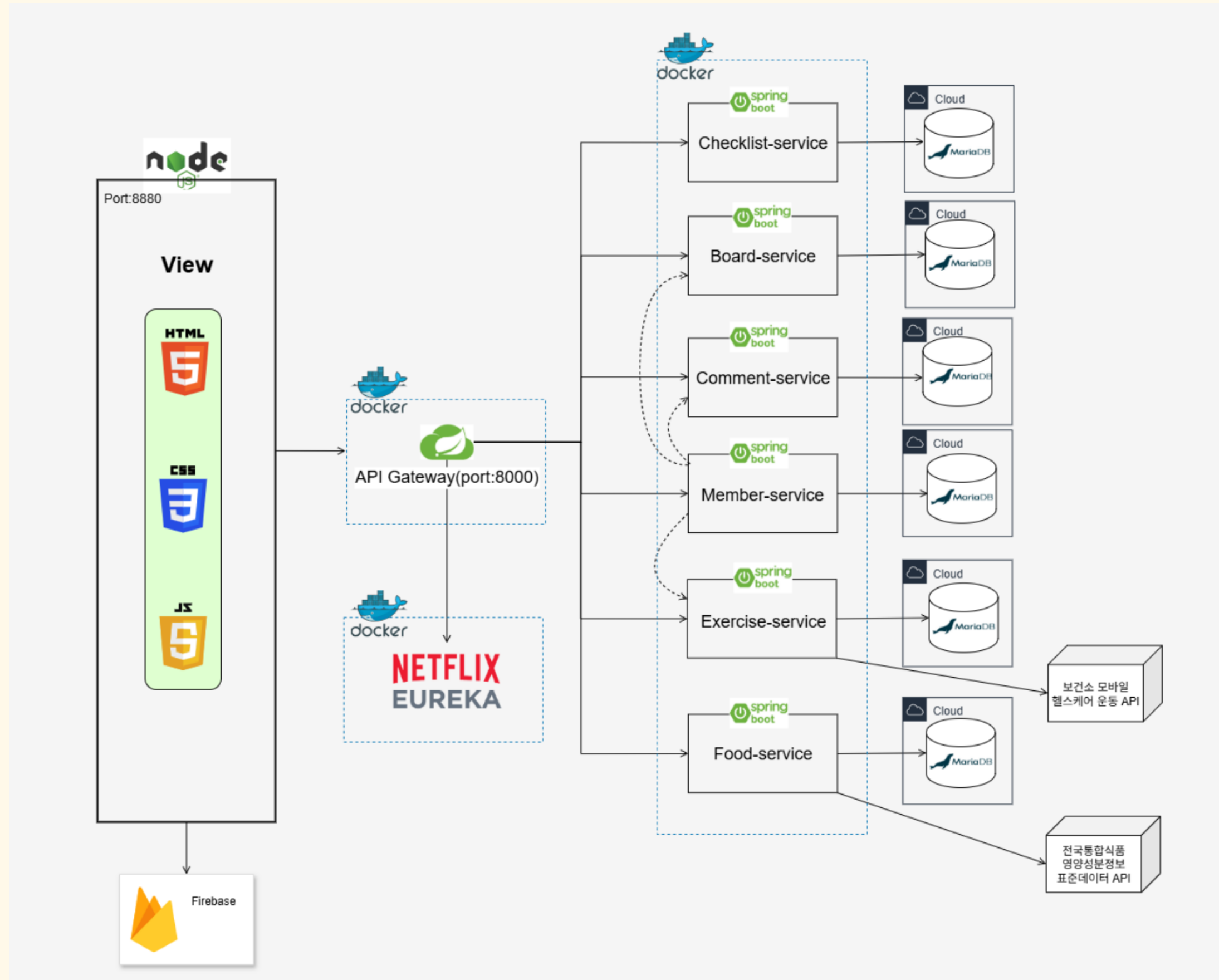
- BackEnd 서버는 여전히 하나라 협업 중 동시 개발이 어려움.
- 작은 수정에도 BackEnd 서버 전체를 다시 빌드해서 재배포를 해야 함.

Sprint 03

MSA 리팩토링

- 1) 소프트웨어 아키텍처
- 2) 최종 산출물
- 3) 회고

1) 소프트웨어 아키텍처



2) 최종 산출물

- 업무상세설계서

exercise-service												
번호	업무	흐름	URL				ExerciseService			ExerciseRepository		
				종류	req	res	method	input	output	input	output	
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>	getExerciseKcal	exerciseDto	ExerciseDto	exerciseDate, memberId	List<ExerciseEntity>	
2	연도별 운동칼로리 조회	대시보드 운동에서 운동그래프에 연도별 소모칼로리 차트 출력	/api/exercise-service/year/member/{memberId}	POST	@PathVariable String memberId @RequestBody ReqGetYearExerciseKcal reqGetYearExerciseKcal	ResponseEntity <List<ResGetYearExerciseKcal>>	getYearExerciseKcal	exerciseDto	List<ExerciseDto>	memberId, start, end	List<ExerciseEntity>	
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>	addExercise	exerciseDto	void	exerciseEntity memberId, exerciseDate	ExerciseEntity ExerciseTotalEntity	
4	운동 상세 조회	운동 상세에서 항목들 조회	/api/exercise-service/exercises/member/{memberId}	POST	@PathVariable String memberId @RequestBody ReqGetExercises reqGetExercises	ResponseEntity <List<ResGetExercises>>	getExercises	exerciseDto	List<ExerciseDto>	exerciseDate, memberId	List<ExerciseEntity>	
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>	setExerciseMin	exerciseDto	void	exercisId memberId, exerciseDate	ExerciseEntity ExerciseTotalEntity	
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>	removeExercise	exerciseDto	void	memberId, exerciseDate memberId, exercisId	ExerciseTotalEntity boolean	
7	운동 검색	운동 기록에서 패널을 열고 운동 검색	/api/exercise-service/open-search	POST	@RequestBody ReqExerciseOpenSearch reqExerciseOpenSearch	Response<Mono <List<ResExerciseOpenSearch>>>	getExerciseOpenSearch	exerciseDto	Mono<List <ExerciseDto>>	pageNo,numOfRows	Mono<List<ExerciseApi>>	
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>>	getYearExerciseAvgAll	exerciseDto	List<ExerciseDto>	-	-	

2) 최종 산출물

- Discovery Server(EUREKA)

Instances currently registered with Eureka			
Application	AMIs	Availability Zones	Status
APIGATEWAY-SERVICE	n/a (1)	(1)	UP (1)
BOARD-SERVICE	n/a (1)	(1)	UP (1)
CHECKLIST-SERVICE	n/a (1)	(1)	UP (1)
COMMENT-SERVICE	n/a (1)	(1)	UP (1)
EXERCISE-SERVICE	n/a (1)	(1)	UP (1)
FOOD-SERVICE	n/a (1)	(1)	UP (1)
MEMBER-SERVICE	n/a (1)	(1)	UP (1)

- Swagger

OpenAPI definition v0 OAS 3.0	
/v3/api-docs	
Servers	
<div>http://127.0.0.1:7004 - Generated server url</div>	
food-controller ^	
PUT	/api/food-service/member/{memberId} v
POST	/api/food-service/member/{memberId} v
DELETE	/api/food-service/member/{memberId} v
POST	/api/food-service/open-search v
POST	/api/food-service/kcal/year/member/{memberId} v
POST	/api/food-service/kcal/member/{memberId} v
POST	/api/food-service/foods/member/{memberId} v
POST	/api/food-service/avg/year/member/{memberId} v
GET	/api/food-service/api-test v

2) 최종 산출물

- Docker

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	<div><div></div>selfit-ui</div>	4486fc1862da	selfit-ui:1.0	8880:3000 ↗	0%	1 day ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	<div><div>▼</div><div></div>selfit-back-deploy</div>	-	-	-	3.64%	3 minutes ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	<div><div></div>discovery-service</div>	d24535cfb104	yapyapboo/discovery-service:1.0	8761:8761 ↗	1.64%	3 minutes ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	<div><div></div>comment-service</div>	528cb55a768f	yapyapboo/comment-service:1.0		0.27%	3 minutes ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	<div><div></div>food-service</div>	f066b7e898de	yapyapboo/food-service:1.0		0.23%	3 minutes ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	<div><div></div>apigateway-service</div>	3d0da0906c2f	yapyapboo/apigateway-service:1.0	8000:8000 ↗	0.38%	3 minutes ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	<div><div></div>checklist-service</div>	a45dfbca6dca	yapyapboo/checklist-service:1.0		0.23%	3 minutes ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	<div><div></div>board-service</div>	0d542ee20ecd	yapyapboo/board-service:1.0		0.35%	3 minutes ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	<div><div></div>member-service</div>	098de7f81cc5	yapyapboo/member-service:1.0		0.27%	3 minutes ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	<div><div></div>exercise-service</div>	32fa5fdbcdf8	yapyapboo/exercise-service:1.0		0.27%	3 minutes ago	<div><div></div><div></div><div></div></div>

3) 회고

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.

작은 서비스 단위로 인해 수정한 기능만 빠르게 배포할 수 있다는 점이 인상깊었다. 기존 모놀리식에서는 작은 수정에도 모든 백엔드 서버를 재배포해야 했다. MSA에서는 수정한 기능에 대한 서버만 재배포하면 된다. 비록 금전 문제로 배포하지 못했지만, 작은 서비스 단위로 나뉜 구조 덕분에 빠르게 효율적인 배포가 가능하다는 부분에서 공감이 되었다.

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.