Selfit

2차 스프린트

염윤호 고민표 박희수 이재원 이창훈

목차

01

REST API 서비스 리팩터링

1-1. 리팩터링 필요성
1-2. 변경사항
1-3. 소프트웨어 아키텍처
1-4. JWT 정책
1-5. 개발 및 운영 환경
1-6. 산출물

1-1. 리팩터링 필요성

- 2차 스프린트 아키텍처에 Thymeleaf 기술이 들어가 있었지만 실제 사용량이 많지 않음

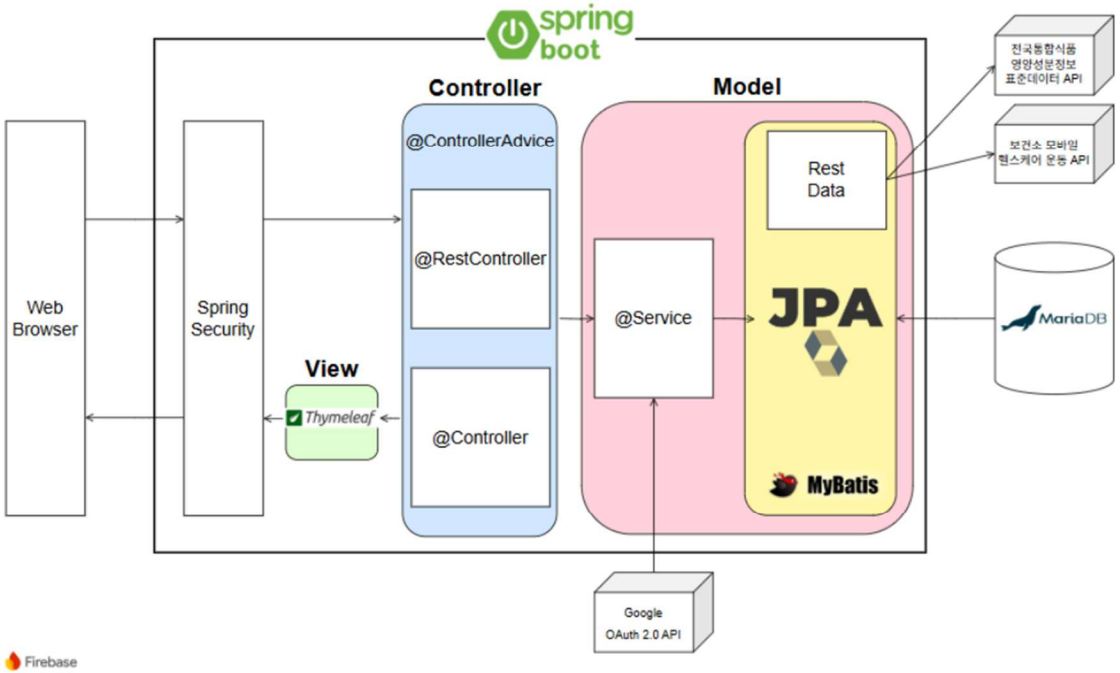
- REST API 제약 조건 중 클라이언트 – 서버 분리 원칙을 지켜야 함

1-2. 변경사항

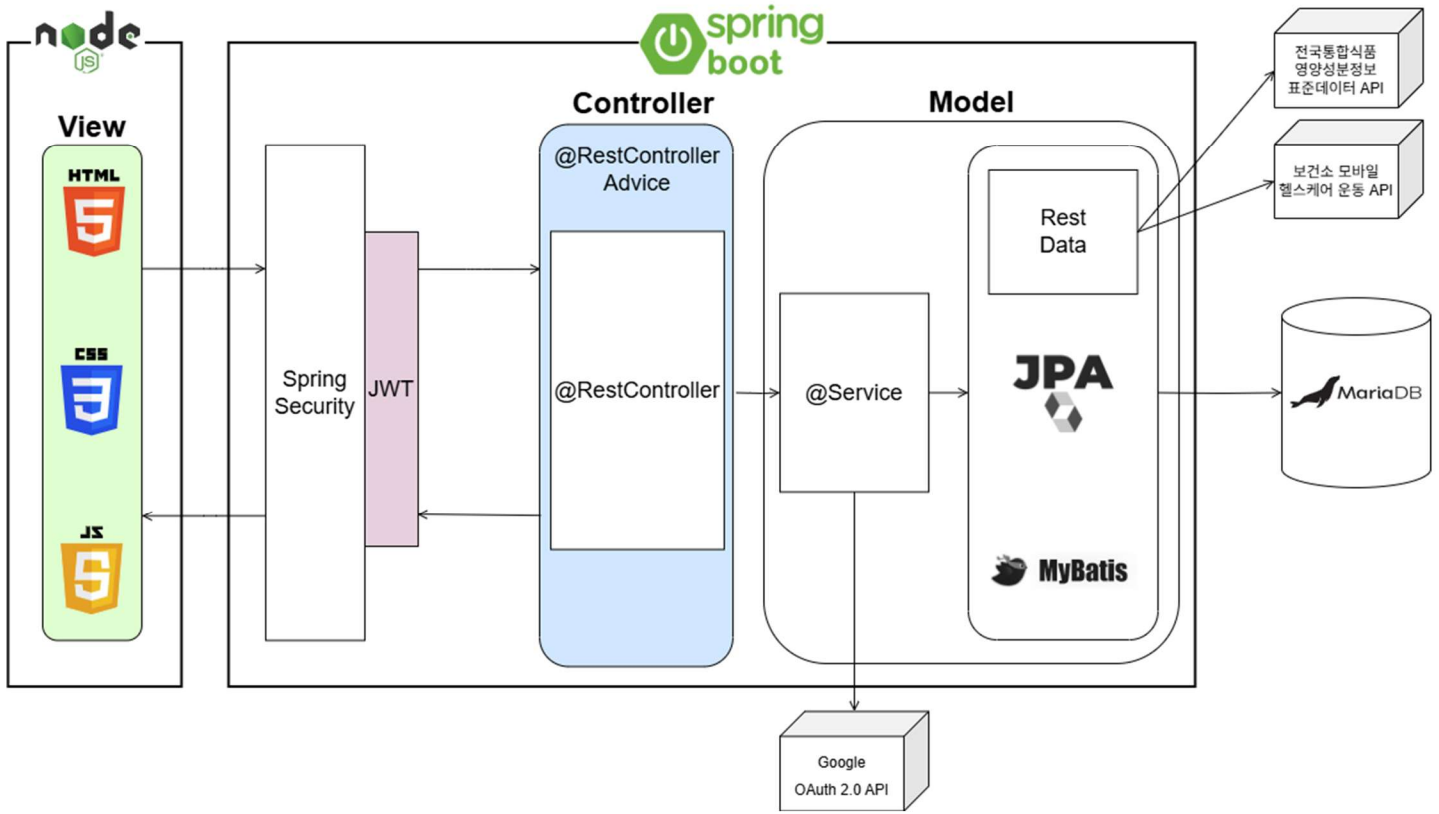
- Front - Back 서버 분리
- Thymeleaf와 @Controller 삭제
- 서버 분리로 인한 JWT 활용

- Front - Back 결합도를 낮추어 독립적 개발 및 배포 가능
- Thymeleaf 를 사용하지 않아 Html은 서버 동작이 필요 없어짐
- 다른 서버에서 사용자 인증 여부를 확인 가능

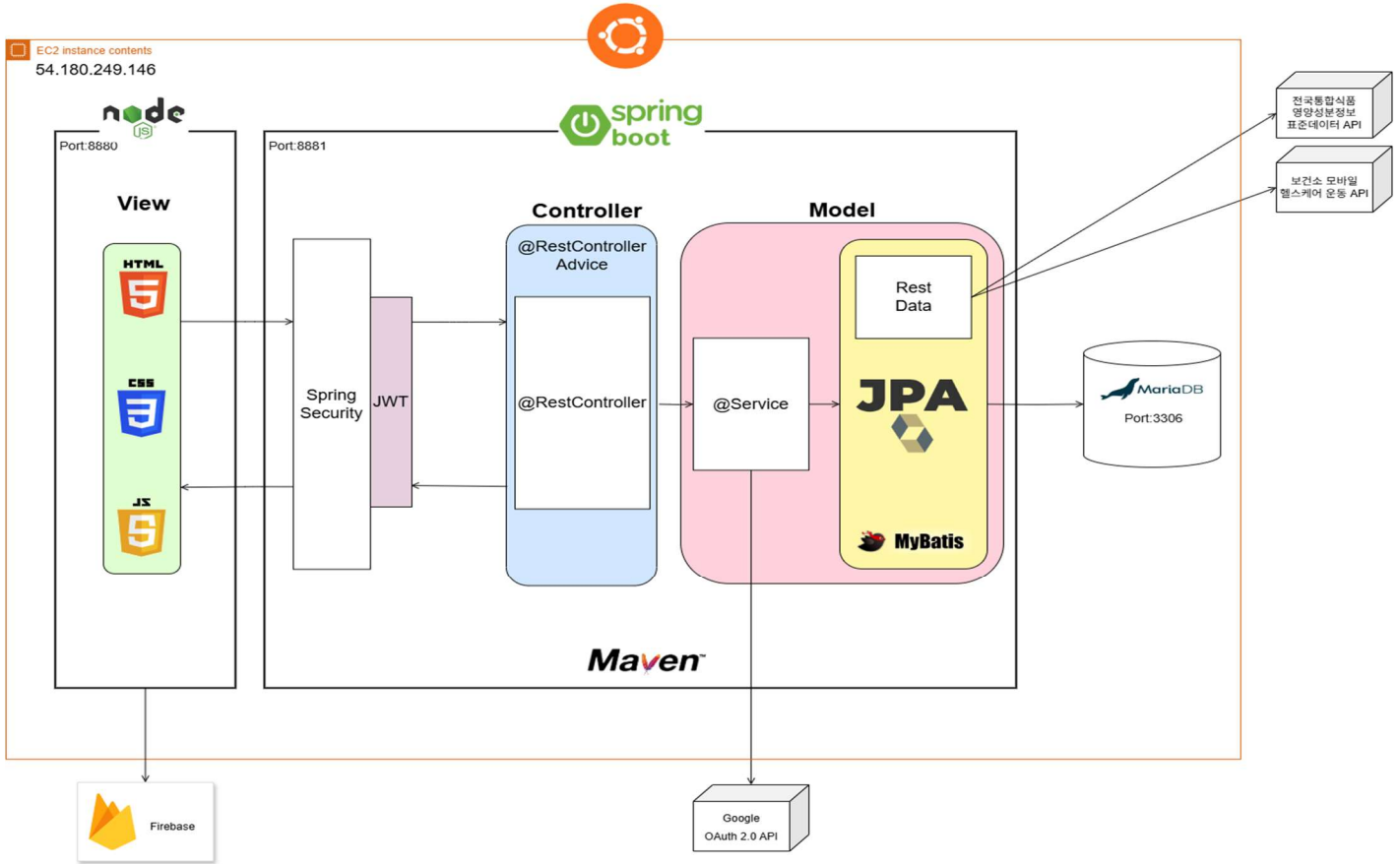
1-3. 소프트웨어 아키텍처(1차 스프린트)



1-3. 소프트웨어 아키텍처(리팩터링)



1-3. 소프트웨어 아키텍처(배포)



1-4. JWT 정책

SECRET	"selfit"
TIMEOUT	60분
TOKEN_PREFIX	"Bearer "
HEADER_STRING	"selfitKosta"
저장소	Local Storage

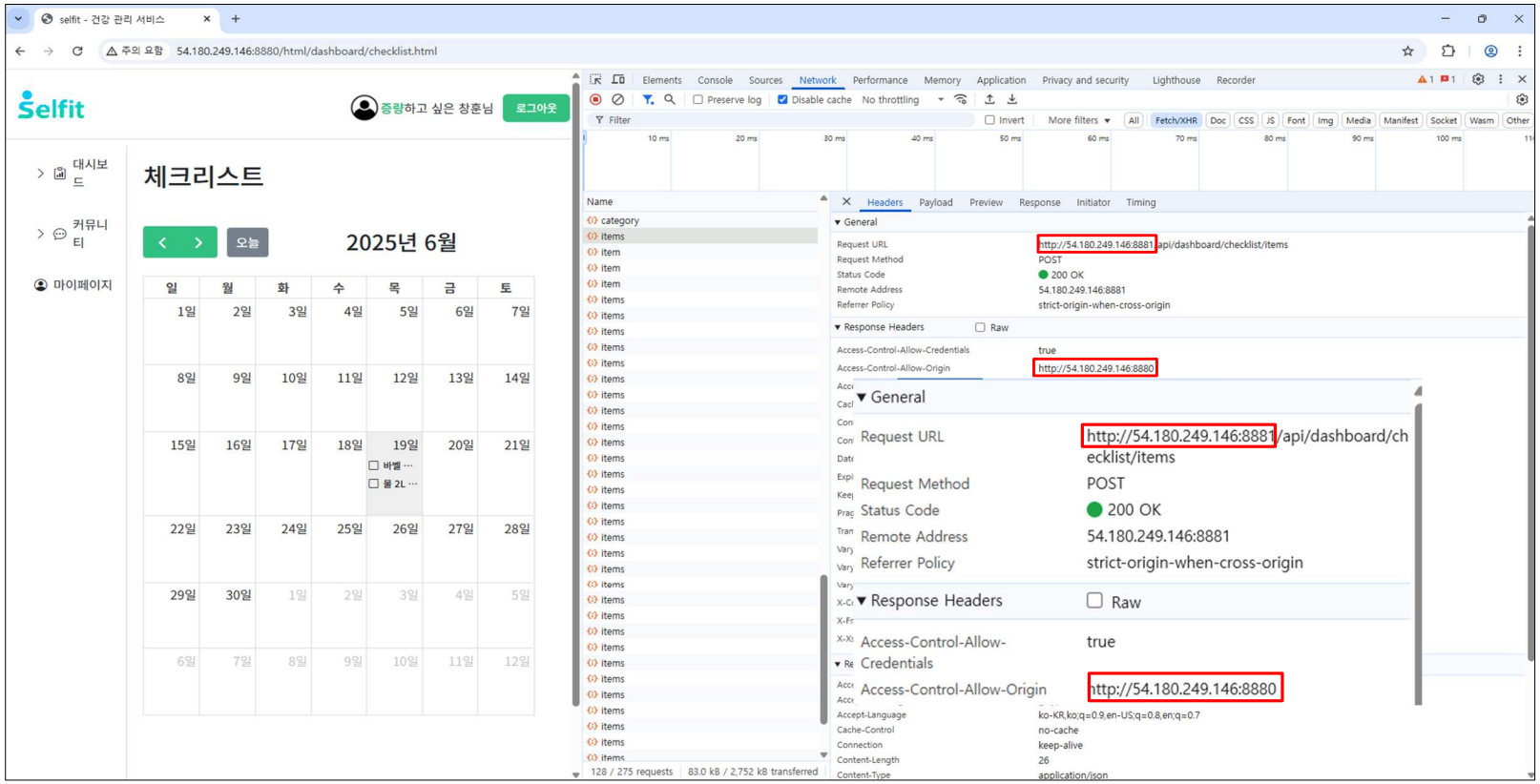
1-5. 개발 및 운영 환경

	개발환경	운영환경(Ubuntu 22.04)
SpringBoot	3.4.5	3.4.5
Maven	3.9.9	3.8.7
java	22	22
jwt	4.4.0	4.4.0
lombok	1.18.30	1.18.30
mybatis	3.0.3	3.0.3
gson	2.8.9	2.8.9
webmvc-ui	2.5.0	2.5.0
hibernate	8.0.1.Final	8.0.1.Final
maria DB	10.6.22	10.11.13
node.js	v22.16.0	v18.19.1
axios	1.6.8	1.6.8
firebase	11.8.1	11.8.1
bootstrap	5.3.3	5.3.3
bootstrap-icon	1.11.0	1.11.0
fullcalendar	6.1.11	6.1.11

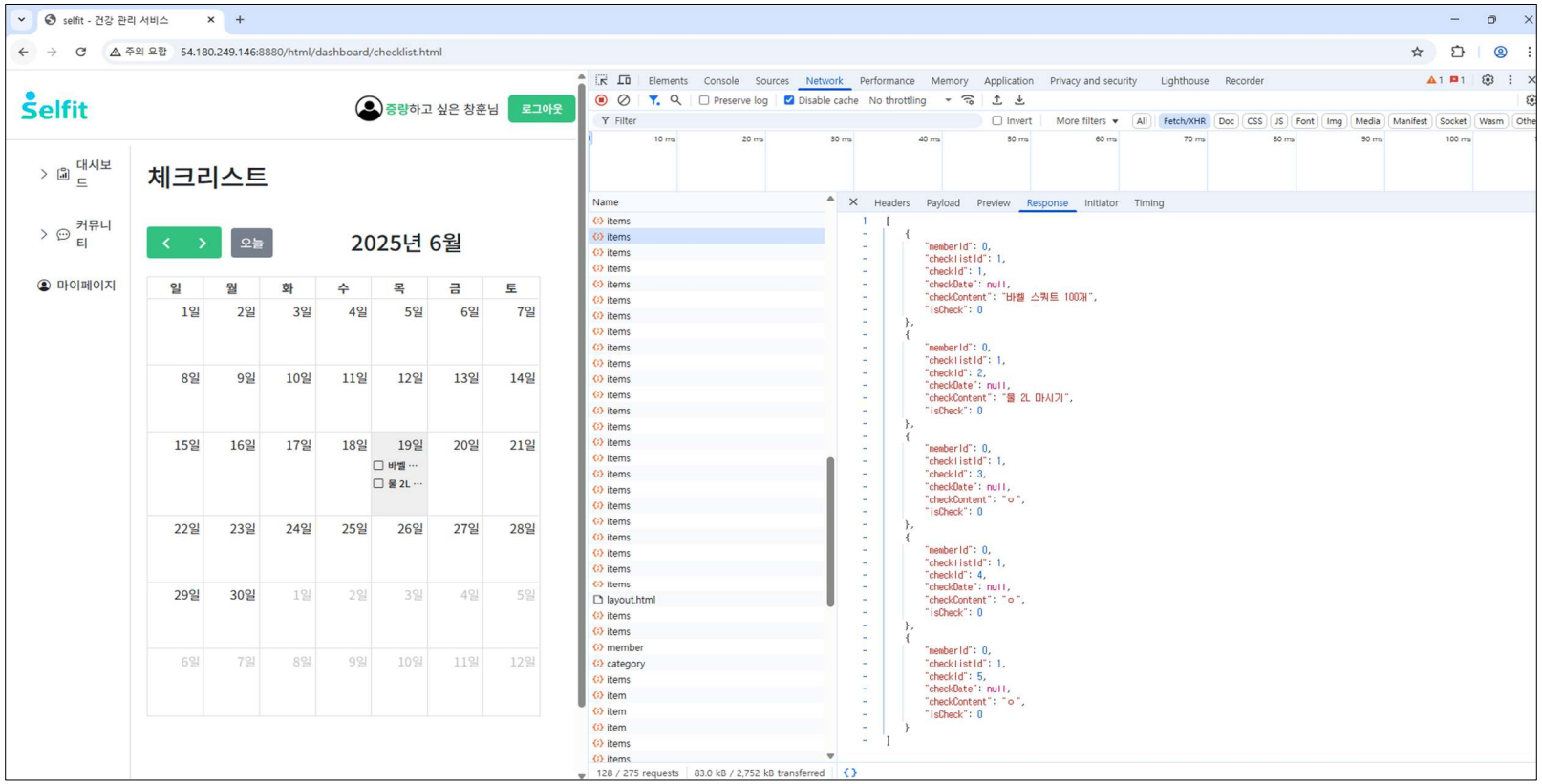
1-6. 산출물

http://54.180.249.146:8880/html/

https://github.com/kosta-selfit/selfit-server
https://github.com/kosta-selfit/selfit-ui-node



1-6. 산출물


selfit - 건강 관리 서비스
54.180.249.146:8880/html/dashboard/checklist.html
Selfit
증량하고 싶은 창훈님
로그아웃
대시보드
커뮤니티
마이페이지
체크리스트
오늘
2025년 6월
일 월 화 수 목 금 토
Elements Console Sources Network Performance Memory Application Privacy and security Lighthouse Recorder
Name
Headers Payload Preview Response Initiator Timing
128 / 275 requests 83.0 kB / 2,752 kB transferred

02

회고

2. 회고

염윤호 서버를 나누게 되면 독립적인 개발과 배포가 된다는 측면에서 인상깊었다. ui나 서버 하나만 수정해도 전체 서버를 재가동시키는 게 당연하다고 생각했다. 하지만 서버를 나누게 되면 서로의 결합도가 낮아지는 것을 보았다. 그 결과 효율적인 서버 운영이 가능했다. 앞으로도 효율적인 개발과 운영 측면에서도 고려해야겠다는 생각을 했다.

고민표 처음에는 프론트와 백엔드 구조를 분리하는 것이 단순히 폴더만 나누는 작업일 거라 생각했다.
하지만 실제로는 CORS 설정, 인증 처리 다양한 요소들을 고려해야 했고, 이에 따라 추가 학습과 설계 고민이 필요했다.
시행착오도 있었지만 결과적으로, 백엔드는 백엔드대로, 프론트는 프론트대로 명확히 분리되었고 코드 관리가 수월해졌다.

박희수 2일간의 리팩토링을 진행하면서, 기존 Controller를 최대한 RESTful하게 구성해 두었던 덕분에 JWT 인증 부분을 제외하고는 서버 코드를 거의 수정하지 않아도 되었다. 프론트엔드 부분만 집중적으로 수정하면 되었고, 이를 통해 처음부터 역할과 책임이 명확히 분리된 구조로 설계하는 것이 얼마나 중요한지 다시 한번 느낄 수 있었다.
이번 경험을 통해, 앞으로 프로젝트를 진행할 때 아키텍처 설계가 개발 생산성과 유지보수성에 얼마나 큰 영향을 주는지 깊이 체감할 수 있었다.

2. 회고

이재원 기존에 프론트, 백엔드를 하나의 서버로 운영하던것을 분리를 하니까 프론트엔드와 백엔드가 명확히 분리되어 코드 가독성 및 유지보수성이 향상된다고 느꼈고, React 같은 프레임워크 없이 순수 JS로 SPA를 구현하면서 직접 라우터 설계와 컴포넌트 렌더링 로직을 다뤄본 경험을 얻었고 프레임워크가 내부에서 어떤 방식으로 동작하는지 더 깊이 이해하게 되었습니다.

이창훈 이번 리팩토링을 통해 프론트-백엔드 분리 구조의 가치를 체감할 수 있었다. 단일 서버 구조에서는 느낄 수 없던 CORS 정책의 개념, 인증 헤더를 직접 전달해야 한다는 점이 인상깊었다. 특히 기존의 HttpSession 기반 인증 방식에서 JWT 토큰 기반 인증 구조로 전환하면서, 서버가 상태를 기억하지 않고도 인증을 처리할 수 있는 구조에 대해 깊이 이해하게 되었다. 사용자의 상태를 클라이언트가 직접 관리하는 방식은 처음엔 어색했지만, 이후 API 서버의 확장성과 독립성을 고려했을 때 훨씬 유리하다는 점을 느꼈다.