# 백엔드 개발자
# 유예지입니다!

저는 **프로그램의 발전을 끊임 없이 고민하는 개발자**입니다.
프로젝트 완료 후에도 로그인 기능 추가, 로딩 속도 리팩토링, 서버 모니터링 시스템 도입 등 지속적인 개선을 통해 프로그램을 발전시켰습니다.

저는 **목표를 위해 새로운 기술을 도입하는 사람**입니다.
프로젝트에서 웹 컨트롤 기능이 필요했습니다. 이를 구현하기 위해 웹 컨트롤 라이브러리를 조사하던 중, 셀레니움이라는 도구를 알게 되었습니다. 이 기술은 저에게 익숙하지 않았지만, 목표를 달성하기 위해 빠르게 학습하고 작은 프로젝트를 만들어 여러 번 실험을 진행했습니다. 그 결과, 웹 자동화 프로그램을 성공적으로 제작할 수 있었고, 프로젝트의 목표를 달성하는 데 큰 기여를 할 수 있었습니다.

## PROJECTS.

### BURGERPUT *2023.05 ~*

**사용기술** AWS, GIT, MYSQL, JAVA, SELENIUM, SPRING, LINUX

#### 개요

서버에 제출하는 값의 오기입률을 0%로 만들기 위한 웹 봇 프로그램.

#### 주요업무

웹 페이지의 정보를 가져오는 크롤링 로직 설계 및 구현
웹 크롤링 라이브러리를 이용해 제 3자 웹페이지에서 온도를 입력하고 제출하는 로직 설계 및 구현
FRONT와 서로 다른 도메인에서 API를 구축하여 연동
백엔드 AWS인프라 구축
DB설계
GITHUB ACTION을 이용한 CI/CD 구축
쉘 스크립트를 사용해 서버 작동 결과를 확인할 수 있는 모니터링 시스템 구축
제안서, 설계서, 유저플로우, 로직 플로우, API명세서 작성

#### 상세내용

새로운 오픈소스 SELENIUM사용을 위해 TINY PROJECT를 만들어 실험을 거치며 개발 진행
사용자의 관점에서 개발하기 위해 쉬운 사용방법을 고려하며 개발 진행
주기적인 회의를 통해 업무를 계획하고 정해진 날짜에 배포할 수 있도록 함

#### 성과

사용자들에게 편리하다는 긍정적인 피드백을 받음
다른 매장에서 사용 요청받아 서버 배포 준비 중

#### REVIEW

서버를 직접운영해보면서, 서버의 사양을 고려해 애플리케이션을 개발하게 됨
처음 써보는 오픈소스를 다뤄보면서, 익숙하지 않은 기능을 어떤식으로 학습하고 개선해야하는지 알게 됨
팀원과 문제 해결을 위해 다양한 방법을 알아오고, 이를 공유하면서 다양한 아이디어의 중요성에 대해서 알게됨

# WORK EXPERIENCE

## 구루멘토링 *2019.11~ 2020.07*

## 한국 고전번역원

**사용기술** LINUX

### 개요
서버 증축과정에서 생겼던 포트 충돌 문제 해결

### 주요업무
APACHE포트포워딩 설정을 통해 포트 충돌 문제 해결

### 상세내용
총 세 개의 애플리케이션 중 두 개의 서버가 같은 포트를 사용해 발생한 포트 충돌문제를 해결하기 위해, 포트 번호를 변경하고 URI에 따라 포트포워딩 작업을 수행

### 성과
한 대의 서버 컴퓨터에서 세 개의 애플리케이션을 포트충돌없이 구동

### REVIEW
팀원과 밤늦게까지 함께 고민했던 경험 덕분에 팀원의 중요성을 깨닫게 됨

---

## 육군사관학교 FTP서버 이전

**사용기술** LINUX

### 개요
리눅스 서버 이전 작업중 한글 자료가 깨지는 현상 해결

### 주요업무
이전 작업중 윈도우 서버를 경유해 한글자료가 깨지는 현상 해결

### 상세내용
윈도우 서버로 경유해 EUC-KR을 CP-949로 윈도우 인코딩 형식으로 변환 후 UTF-8형식으로 인코딩 하는 과정을 거친 후 새로운 REDHAT서버로 이전시킴

### 성과
한글 파일이 깨짐없이 서버 이전작업 완료

## OTHER EXPERIENCES

### 항해99 코딩테스트 스터디 2기 *2024.06 ~ 2024.07*

TIL을 통해 하루 공부한 내용을 돌아보는 습관 형성
알고리즘 코딩테스트 문제를 풀어보며 자료구조, 알고리즘 학습
스터디원들과 TIL, 지식공유 [링크]

### 항해99 코딩테스트 스터디 3기 *2024.07 ~ 2024.08*

페어 프로그래밍을 통해 함께 알고리즘 풀이
4주차 우수 TIL에 선정됨[TIL링크]

### 3. 우수 TIL 작성자 : 유예지 (자바/챌린저)

- TIL 링크
- 선정 이유

  예지 님의 TIL은 문제 풀이 로직을 직접 손으로 풀어본 점이 인상적이었고, 코딩 테스트 문제 유형을 분석하여 체계적으로 접근한 점이 돋보여 우수 TIL로 선정되었습니다. 더불어, 이를 토대로 앞으로 보완하고 집중해야 할 과제들을 제시하여 중요한 포인트를 잘 짚어냈습니다.

### 솔데스크- 정보보안 침해대응 전문가 과정 *2018.03~2018.10*

리눅스/윈도우 관리자 과정
쉘 프로그래밍

## OTHER EXPERIENCES

### [졸업]인천재능대학교 컴퓨터 정보과 *2016.02~ 2018.02*

## LICENCES

- MOSMASTER 2016.12
- GTQ포토샵 1급 2016.06
- 네트워크관리사 2급 2019.07
- 리눅스 마스터 2급 2019.11
- 정보처리산업기사 2019.11

## CONTACT

Email. jwb27964@gmail.com
Phone. 010-4064-1306

## CHANNEL

Blog. https://velog.io/@bbubboru22/posts
GitHub. https://github.com/yellTa

# 자기소개서

## 사용자의 입장에서 문제를 해결하고, 도전을 두려워하지 않는 지원자

아르바이트를 하던 중, 웹 사이트에 매장의 온도를 기입하는 작업에서 오류가 자주 발생하는 문제가 발생했습니다. 그 이유는 매장에서 사용하지 않는 기기나 식품 관련 항목이 제외되지 않고 언제나 목록에 있었기 때문이었습니다. 이를 해결하기 위해, 데이터를 입력하기 전에 필요한 항목만 온도를 적어도 되는 필터링 웹 프로그램이 필요하다고 생각했습니다.

웹 자동화에 대해 알아보던 중, Selenium 라이브러리를 알게 되었습니다. 이를 바탕으로 프로그램을 개발할 수 있다는 확신이 생겼고, front 단을 담당할 수 있는 팀원이 필요하다는 생각에, 함께 아르바이트하던 친구에게 팀을 이뤄 프로젝트를 진행하자고 제안했습니다.

처음 사용하는 라이브러리와 기술들이 많았던 만큼, 프로젝트 초기 내내 '이게 과연 가능할까?'라는 의문을 품고 있었습니다. 주변 사람들 또한" 그런 프로그램이 있으면 좋은데, 정말 가능할까?"라는 반응을 보였습니다. 하지만 매주 회의 시간을 가져 체계적으로 프로젝트 일정을 관리했고, 작은 테스트 프로젝트를 만들어보며 Selenium의 기능을 익혀 프로젝트에 적용했습니다. 그 결과, 약 6개월 만에 프로그램을 성공적으로 개발해 배포할 수 있었습니다.

프로젝트는 거기서 끝나지 않고, 사용자들의 피드백을 반영해 프로그램을 지속적으로 개선해 나갔습니다. 첫 번째 개선 사항은 로딩 시간을 줄이는 것이었습니다. 기존에는 웹 페이지를 사용하기 전 최초로 5분의 로딩 대기시간이 필요했습니다. 이를 해결하기 위해, 프로그램을 사용하지 않는 시간인 오전 8시 34분에 자동으로 웹 페이지를 미리 로드하도록 cron작업을 설정해 사용자가 바로 서비스를 이용할 수 있도록 개선했습니다.

두 번째로, 제출 로직을 리팩토링하여 총제출 시간을 4분에서 1분으로 단축했습니다. 특히 제출이 이뤄지는 페이지에서는 수행 시간을 120초에서 16초로 8배나 개선된 속도를 보여줬습니다.

현재 이 프로그램은 1개의 매장에서 실서비스 중이며, 다른 매장에서도 사용하고 싶다는 요청을 받아 추가 배포를 준비 중입니다. 이 프로젝트를 통해 저는 사용자의 요구를 반영한 서비스 구축과 팀을 설득해 도전을 현실로 만들어낼 수 있었습니다. 또한, 새로운 시도를 두려워하지 않는 도전정신 덕분에 프로젝트를 성공적으로 완료할 수 있었습니다.

## 팀원과 다양한 아이디어 공유를 통해 프로그램을 발전시키는 지원자

저는 프로젝트가 완전히 종료될 때까지 개선 사항에 대해 끊임없이 고민해야 한다고 생각합니다. 따라서 프로젝트 배포 이후에도 더 나은 프로그램을 만들기 위해 팀원들과 끊임없이 고민했습니다. 그 결과, 프로그램은 여러 기능을 추가하며 발전할 수 있었습니다.

첫 번째로, 매일 아침 로딩 결과를 확인할 수 있도록 메일 알림 시스템을 도입했습니다. 현재는 Shell Script를 사용하여 이 기능을 구현했으나, 앞으로는 Spring 백엔드 서버를 통해 메일을 전송하는 방법을 도입할 예정입니다.

두 번째로, 서버 로그를 확인한 결과, 의미 없는 GET 요청이 다수 발생하는 것을 발견했습니다. 이 요청들은 봇은 통한 악성 공격일 가능성이 있었습니다. 이를 방어하기 위해 JWT토큰을 사용한 인증 시스템을 도입해 GET크롤링 공격을 차단했습니다. 이후 'robots.txt' 파일을 수정해 크롤러가 서버를 탐색하지 못하도록 설정했습니다.

또한, 사용자 편의를 위해 최솟값과 최댓값 범위를 지정하고 랜덤값으로 온도를 선택해 보내주는 기능을 추가했습니다. 그리고 관리자가 사용자 불편 사항을 실시간으로 확인할 수 있도록 채팅 기능을 front 단이 구현하고 있습니다.

이처럼 저는 단순히 기능을 구현하는 데 그치지 않고, 지속적으로 더 나은 서비스를 제공하기 위해 팀원과 함께 고민하고 있습니다. 팀원과 다양한 의견을 공유하며 프로그램을 발전시켜 왔고, 그 과정에서 코드 리팩토링을 통한 코드 리뷰를 병행해 코드의 품질이 향상되었습니다. 이 경험을 통해 프로그램뿐만 아니라 저 자신도 함께 성장할 수 있었습니다. 또한, 프로젝트를 진행하면서 서버에 추가하게 되는 기술뿐만 아니라, 이를 뒷받침하는 java, CS, Spring, 시스템 설계에 대한 중요성을 깨닫게 되었습니다.

## 프로젝트 진행중 힘들었던 개발 경험

제출 및 로딩은 프로젝트의 핵심 기능으로, 이를 구현할 때 웹 크롤링 기술을 사용했습니다. 로직을 도식화하여 설계한 덕분에 빠른 시일 내에 기능을 완성할 수 있었습니다. 그러나 모든 기능을 완성하고 서버에 첫 배포를 시도하는 순간 문제가 발생했습니다. 로컬 환경에서는 정상적으로 작동하던 서비스가 AWS Ubuntu 환경에서는 구동되지 않는 것이었습니다.

문제 해결을 위해, 간단한 Selenium 프로젝트를 만들어 테스트해 보았습니다. 해당 프로젝트는 로그인 작업을 웹 크롤링으로 실행하는 프로그램이었습니다. 하지만 이마저도 서버에서 작동하지 않았습니다. 원인을 조사하던 중, 서버가 GUI모드가 아닌 CLI모드인 것을 알게 되었습니다. Selenium은 기본적으로 GUI렌더링이 필요했지만, headless 옵션을 사용하면 렌더링 없이 구동할 수 있다는 사실을 발견했습니다. headless 옵션을 추가한 후, 로그인 프로젝트는 성공적으로 실행되었습니다.

하지만, 메인 프로젝트는 여전히 headless 옵션을 추가해도 작동하지 않았습니다. 이때, 메모리 사용량이 문제일 수 있다는 생각이 들었습니다. 프로젝트의 웹 크롤링 방식은 페이지 이동이 많고, 웹 크롤링 로직까지 있어 메모리 소모가 많았습니다. 이를 해결하기 위해 SWAP 메모리 2G를 추가로 할당했습니다. 결국 메인 프로젝트도 성공적으로 구동시킬 수 있었습니다.

서비스가 정상적으로 구동된 후에도 서버가 자주 다운되는 문제가 발생했습니다. SWAP 메모리를 지정했음에도 메모리 부족 현상이 지속된다고 판단했습니다. 이를 해결하기 위해, 서버의 부담을 줄이고 비용을 최소화하는 방향으로 아키텍처를 개선했습니다. 병합되어 있던 front-end와 backend를 분리하여, 서버에는 렌더링 작업 없이 API만 주고받도록 변경했습니다. 이를 통해 서버 자원을 효율적으로 사용하면서 서비스 안전성을 확보할 수 있었습니다.

이 경험을 통해 문제 상황을 차례차례 개선하며 성장했음을 느꼈고, 서버의 사양을 고려한 애플리케이션의 개발에 대해서 깨닫게 되었습니다.

## 지원동기 및 포부

토스 증권의 출범은 단순히 증권사를 연결하는 브로커 역할이 아닌, 토스 자체만의 증권서비스를 만들자는 혁신적인 도전이었습니다. 특히, 기존의 증권 시스템이 특정 투자자로 이용자를 한정했지만, 토스는 사용자 친화적인 UI를 제공해 모든 투자자를 포용하는 서비스를 제공했습니다. 이런 사용자 중심의 접근방식이 인상 깊었으며 저 또한 개발을 수행할 때 중요하게 여기고 있는 부분이었습니다.

또한, 토스에서 여러 부서 간 협업을 통해서 프로그램을 빠르게 발전시키는 협업 문화가 인상 깊었습니다. 토스CX팀에서 더치페이 기능 개선을 여러 차례 제안 받았고, 이를 빠르게 개발팀에 전달해 고객의 요구를 반영한 사례는 토스가 사용자의 의견을 얼마나 중요하게 생각하는지 그리고 토스팀에서의 협업이 얼마나 중요한지 알 수 있었습니다.

토스 증권은 글로벌 서비스로 수많은 사용자의 데이터를 처리해야 합니다. 이 과정에서 Kafka를 사용해 실시간 데이터 스트림을 관리하고, k8s를 통해 애플리케이션의 부하에 따라 자동 스케일링할 것이라 예상됩니다. 비록 제가 아직 Kafka와 k8s에 대한 직접적인 경험은 없지만, 서비스를 운영하며 다양한 기술을 빠르게 습득하고 적용했던 경험이 있습니다. 이를 바탕으로, Kafka와 k8s도 빠르게 습득할 수 있다고 자신합니다.

제가 만약 토스증권팀에 입사하게 된다면 k8s와 kafka를 익혀 대용량 메세지 스트림 처리 방법에 대한 전문성을 쌓겠습니다. 또한, 언제나 사용자 측면에서 기능을 생각하고 아이디어를 제안하여 더 나은 서비스를 제공하는 데 기여하겠습니다.

# 포트폴리오

## 웹 봇 사무자동화 프로그램 Burgerput *2023.05 ~*

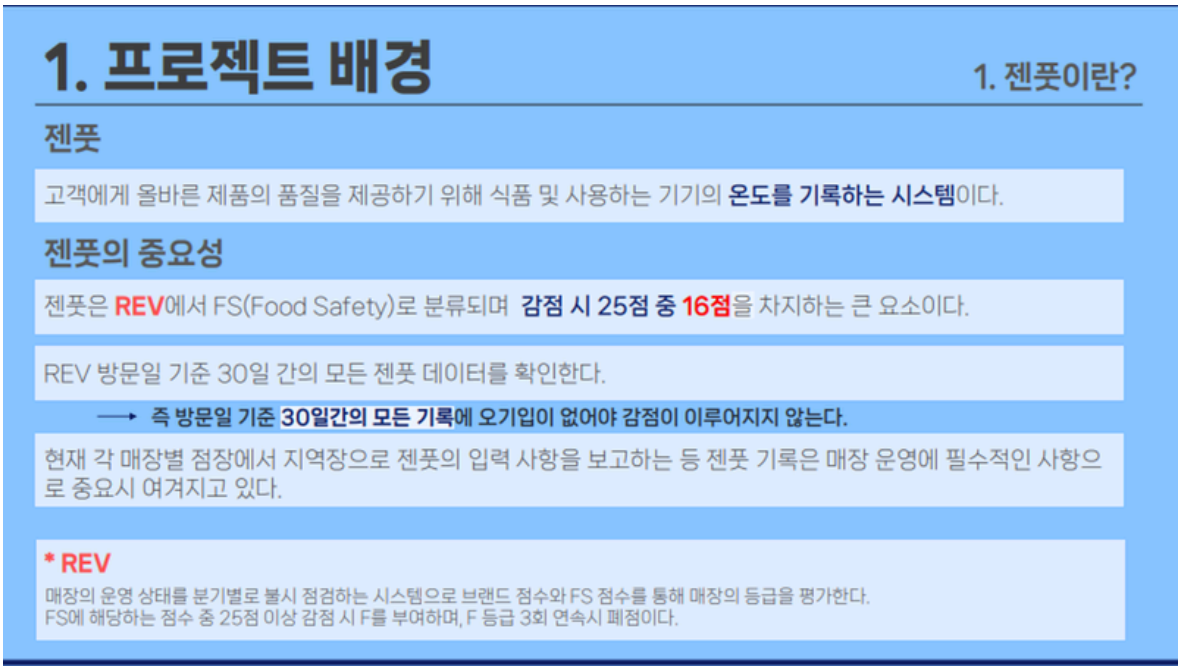
**github 링크** HTTPS://GITHUB.COM/BURGERPUT

**사용기술** AWS, GIT, MYSQL, JAVA, SELENIUM, SPRING, LINUX

### 개요

젠풋이라는 웹 페이지에서 불필요한 요소(사용하지 않는 기기, 제품)는 BURGERPUT을 통해 잘못된 입력 가능성을 0%로 만드는 웹 봇 프로그램이다.

### 시스템 배경





사람이 수기로 웹 페이지에 온도를 입력했기 때문에 오기입이 났었다 해당 문제가 일어나지 않도록 방지하고 근로자가 매장 운영에 좀 더 집중할 수 있도록 업무를 자동화하는 프로젝트를 고안하게 되었다.

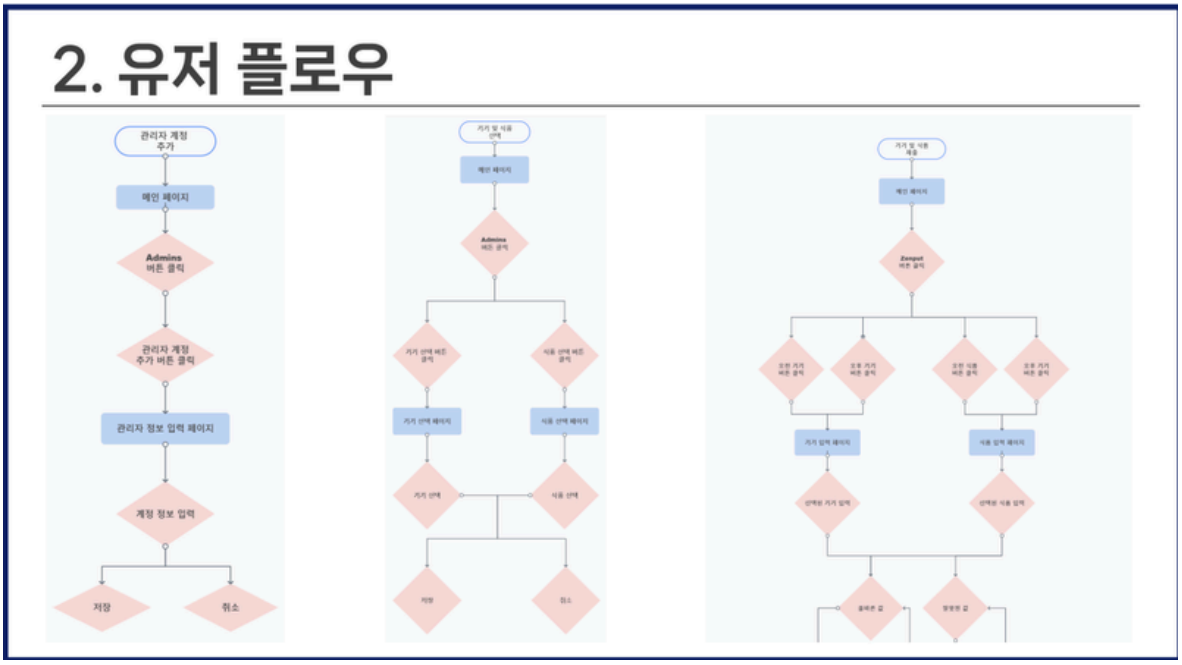
# 프로젝트 설계

## 1.요구사항 명세서

| ID | 화면1 | 화면2 | 화면3 | 요구사항 명 | 요구사항 내용 | 중요도 | 날짜 | 기타 |
|---|---|---|---|---|---|---|---|---|
| BZ-C-1 | 모든 페이지 | | | 네비게이션 | 화면 왼쪽에 네비게이션 바를 표시한다. | 하 | 16-Jan | 네비게이션에 링크 된 페이지로 이동(관리자, 센트 입력) |
| BZ-C-2 | 모든 페이지 | | | 화면 이동 | 네비게이션 바를 이용해 페이지를 이동한다. | 상 | 16-Jan | |
| BZ-C-3 | 모든 페이지 | | | 화면 팝업 | 네비게이션을 이용해 페이지 이동시 태블릿 형태의 화면을 팝업해 페이지를 표시한다 | 상 | 16-Jan | Main 버튼 클릭시 화면 비활성 |
| BZ-M-1 | 메인화면 | | | 페이지 이름 표시 | 페이지 이름을 타이핑 애니메이션을 이용해 표시한다. | 하 | 16-Jan | |
| BZ-M-2 | 메인화면 | | | 로고 이미지 표시 | 페이지 이름 및 로고 이미지를 표시한다. | 하 | 16-Jan | |
| BZ-M-3 | 메인화면 | | | 치트 모드 진입 | 로고 이미지를 3번 클릭하여 치트 모드를 활성화한다. | 중 | 16-Jan | 네비게이션 목록 가장 하단에 해당 페이지로 이동하는 버튼을 할 성화 시킨다. |
| BZ-A-1 | | 관리자 | | 관리자 화면 콘텐츠 | 기계 선택 페이지,식품 선택 페이지 버튼, 관리자 목록 수정, 관리자 계정 추가 버튼을 표시한다. | 하 | 16-Jan | |
| BZ-A-L-1 | | 관리자 | 관리자 목록 수정 | 관리자 목록 표시 | 현재 등록된 관리자 목록을 표시한다. | 상 | 16-Jan | |
| BZ-A-L-2 | | 관리자 | 관리자 목록 수정 | 관리자 목록 추가 | 관리자 목록에 관리자를 추가한다. | 상 | 16-Jan | 현 관리자(천상연, 신호준, 박찬숙, 유예지, 강다이, 최윤진) |
| BZ-A-L-3 | | 관리자 | 관리자 목록 수정 | 관리자 삭제 | 관리자를 목록에서 삭제한다. | 상 | 16-Jan | 버튼을 눌러 관리자를 삭제한다. |
| BZ-A-A-1 | | 관리자 | 관리자 계정 추가 | 입력 폼 표시 | 센풋 제출 시 사용되는 센풋 이메일, rfid 아이디, 비밀번호 입력 폼을 표시한다. | 상 | 16-Jan | 생체시물을 이용한 센풋 제출 시 해당 페이지 진입까지 이메일 및 아이디 비밀번호가 필요하다. |
| BZ-A-A-2 | | 관리자 | 관리자 계정 추가 | 입력 값 검증 | 이메일 양식, 아이디 양식, 비밀번호 양식에 대한 입력을 검증해 사용자에게 에러를 표시한다. | 상 | 16-Jan | |
| BZ-A-A-3 | | 관리자 | 관리자 계정 추가 | 비밀번호 표시 | 버튼을 토글해 비밀번호를 시각화, 비시각화한다. | 상 | 16-Jan | |
| BZ-A-A-4 | | 관리자 | 관리자 계정 추가 | 저장 | 입력된 값을 계정 정보에 저장한다. | 상 | 16-Jan | |
| BZ-A-A-5 | | 관리자 | 관리자 계정 추가 | 취소 | 취소 버튼을 클릭해 화면을 비활성화 및 초기화 후 초기 페이지로 이동 | 하 | 16-Jan | |
| BZ-A-A-6 | | 관리자 | 관리자 계정 추가 | 저장된 정보 표시 | 이전에 저장된 계정 정보를 페이지 진입시 입력 폼에 표시한다. | 상 | 16-Jan | |
| BZ-A-M-1 | | 관리자 | 선택 페이지 공통 | 선택 취소 | 취소 버튼을 클릭해 화면을 비활성화 및 초기화 후 초기 페이지로 이동 | 하 | 16-Jan | |

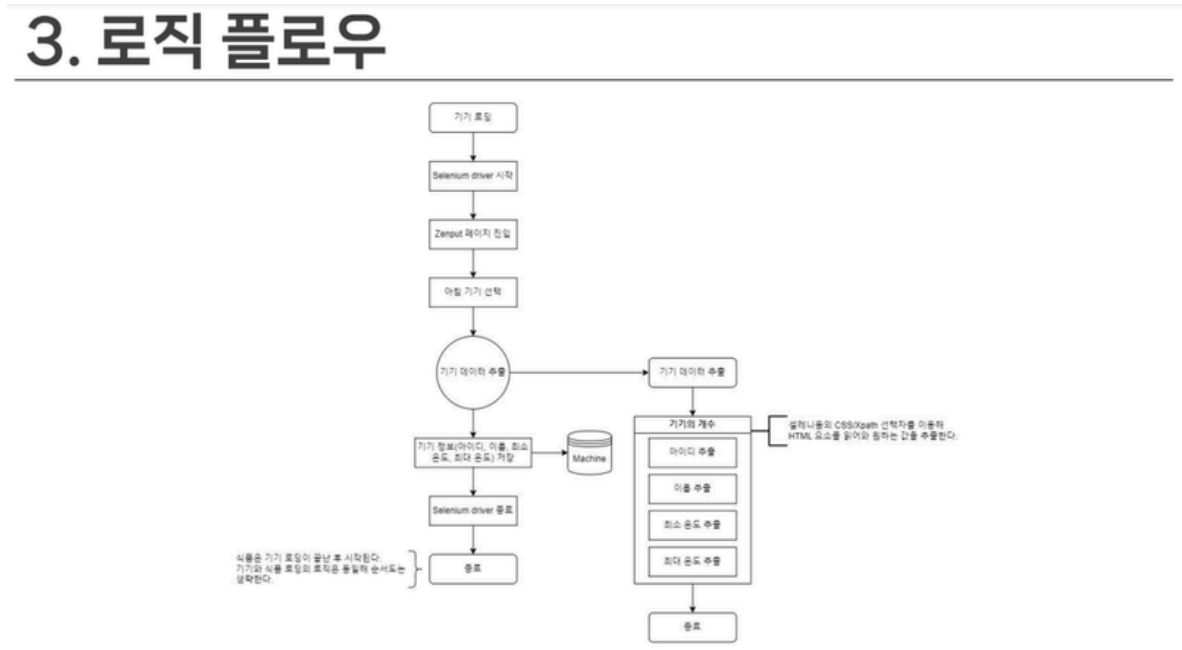
사용자에게 필요한 요구사항을 정리했다. 요구사항을 기준으로 개발을 수행할 항목을 정리했다.

## 2. 유저플로우




유저들이 주요 기능을 사용했을 때 플로우를 정의했다. 웹 페이지의 흐름을 도표화하여 어떤 기능이 필요한지 알 수 있었다.

## 3. 로직플로우



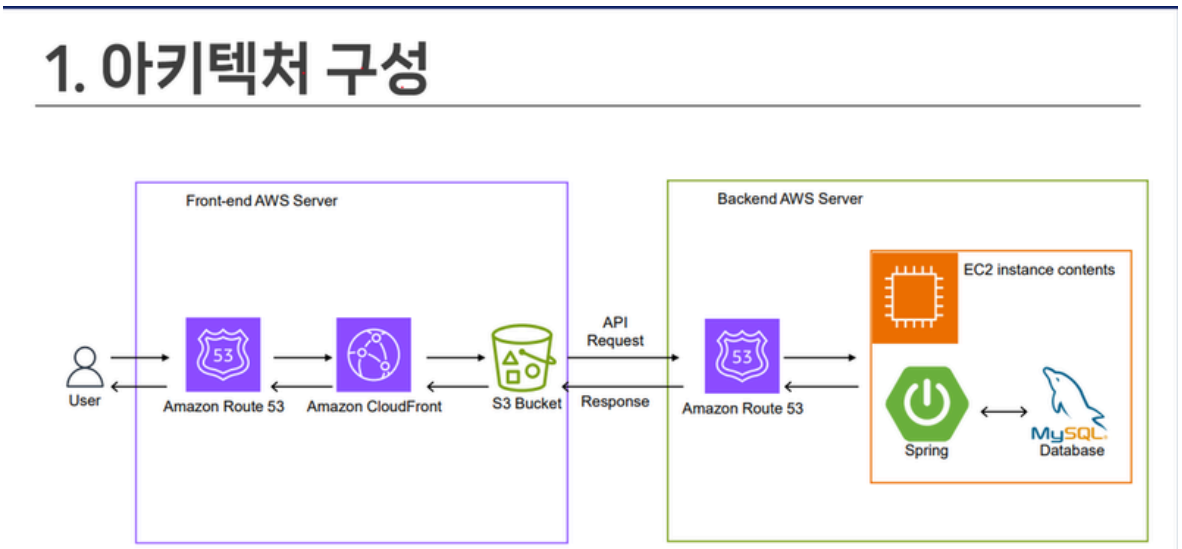

기기로딩의 로직 플로우이다. 해당 기능은 매일 아침에 실행되는 로딩기능의 일부이며 로직이 어떤 흐름인지 도표로 정의했다. 해당 플로우를 보면서 로직의 단계를 정의하고 각 세부적인 부분을 구체화해 구현할 수 있었다.

### 4. 화면 설계서


구현할 화면의 설계서를 작성했다. 각 페이지의 기능, 필드의 역할, DESCRIPTION으로 부가적인 설명을 적어놓아 프로그램이 의도한 대로 명확하게 작동하도록 했다.

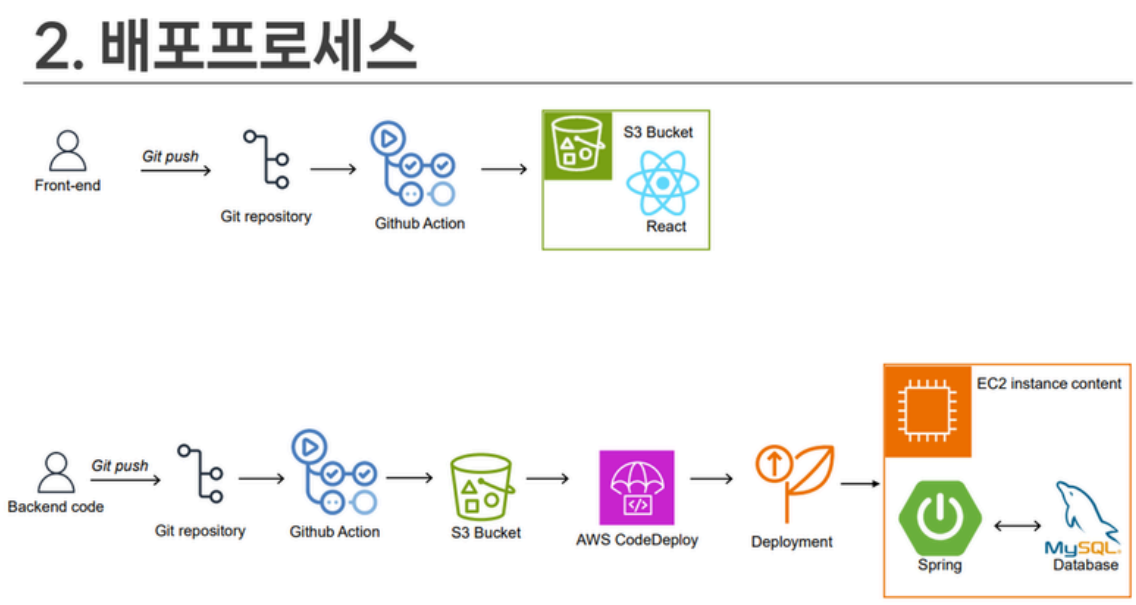
## 프로젝트 개발 계획

### 1.아키텍처 구성



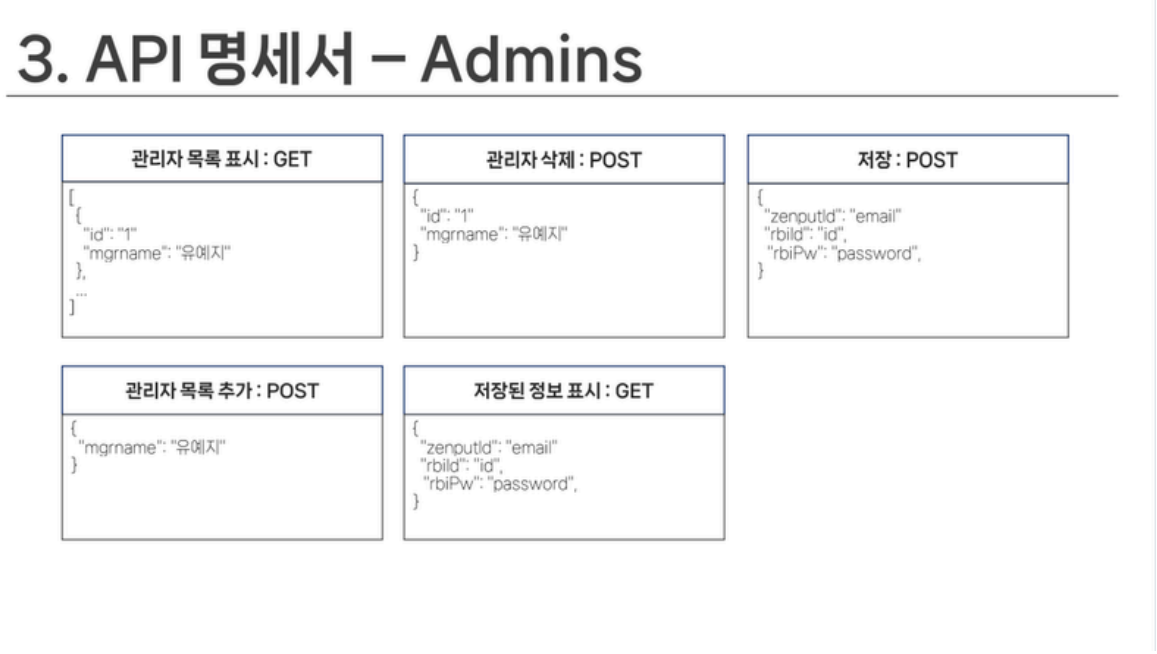

아마존 AWS를 이용하여 구성을 했다. 이때 FRONT단과 BACKEND단 모두 AWS를 따로 구성해서 사용했다. 해당 구성의 도표이다.
ROUTE53을 사용해 FRONT, BACKEND단 각각 도메인을 준비하고 통신하도록 설정했다.

## 2. CI/CD구성



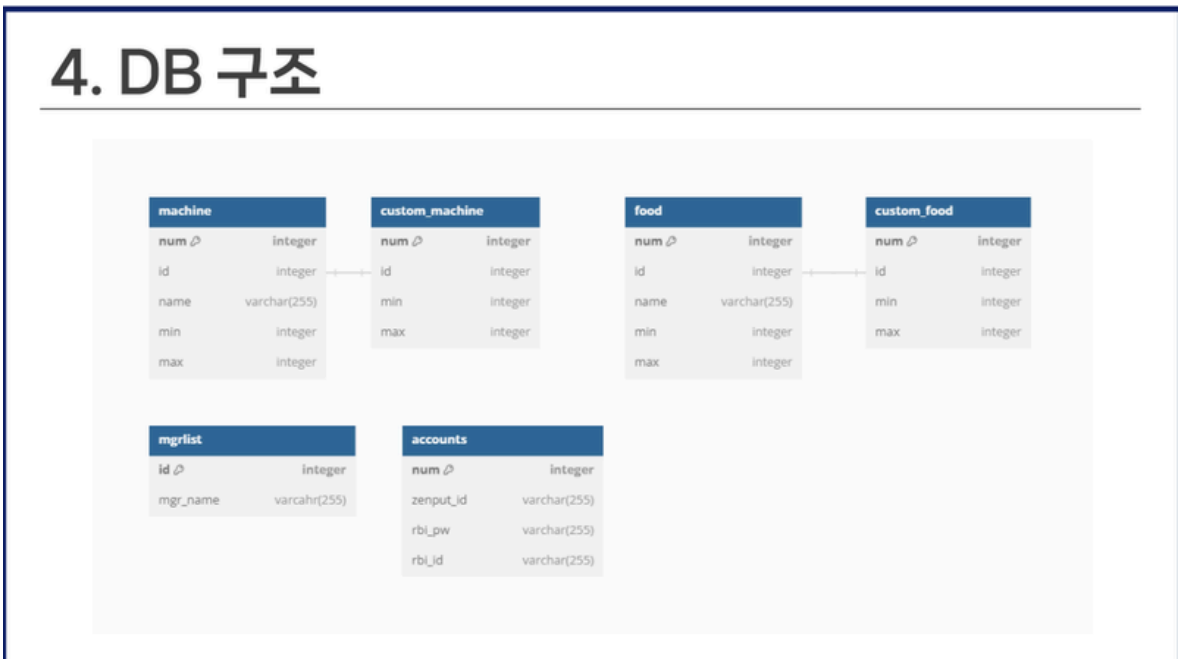

FRONT단 BACKEND단의 CI/CD 구성도이다.
프로그램의 잦은 업데이트로 CI/CD 구성의 필요성을 느껴 도입했다. 현재 MAIN브랜치에 MERGE하면 자동으로 배포가 되며 배포 날짜의 로그를 생성한다.

## 3. API명세서




FRONT단과 통신할 API의 규약을 지정했다. 이를 통해서 프로그램의 일관성을 지키고 API통신에생길 혼선을 방지할 수 있었다.

## 4.DB구조




DB의 구조를 도표로 그려 표현했다. 각 DB가 어떤 속성을 가지는지 한 눈에 확인할 수 있고 각 테이블간의 관계를 한눈에 확인하도록 했다.

## 5. 개발계획

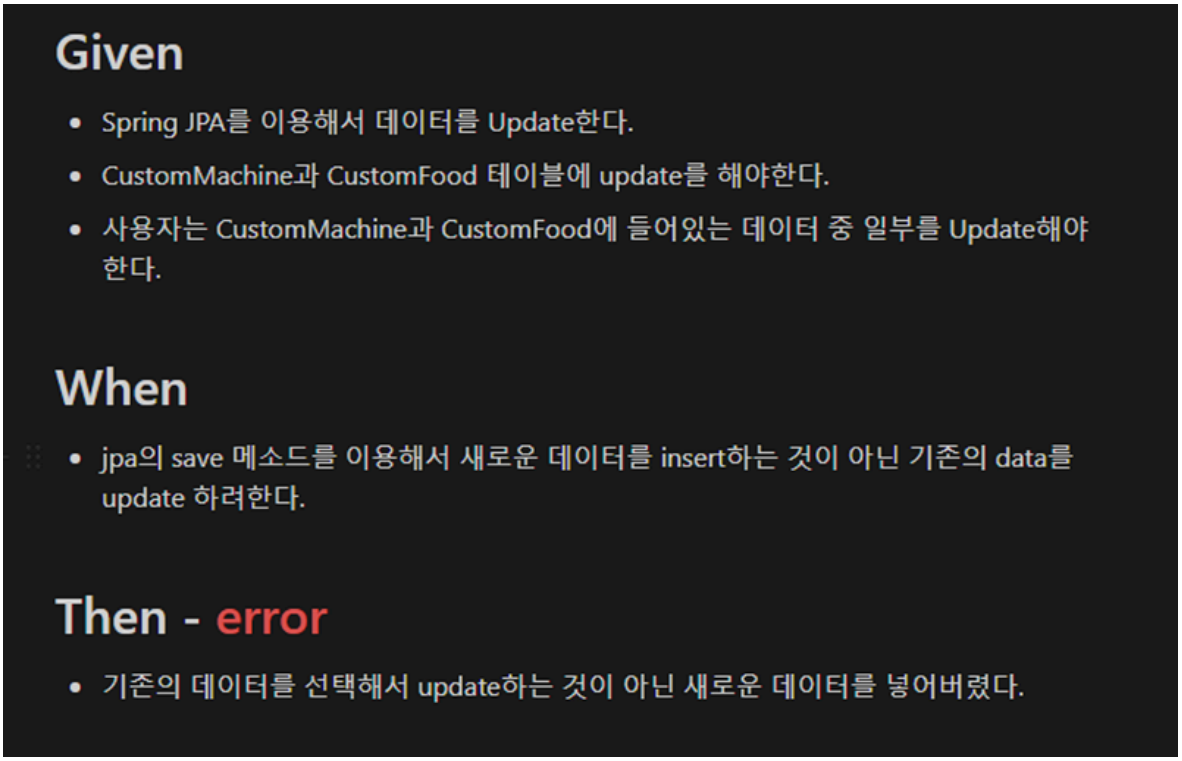



| Aa 이름 | 날짜 | 담당자 |
|---|---|---|
| 로딩로직 | @2023년 8월 1일 → 2023년 8월 20일 | YJ R |
| 항목표시(기기 및 식품 젠폿) | @2023년 8월 23일 → 2023년 8월 27일 | YJ R |
| 레이아웃 구성 및 라우팅 구현 | @2023년 8월 23일 → 2023년 8월 24일 | 지민 고 |
| 기기 및 식품 선택 페이지 구현 | @2023년 8월 25일 → 2023년 8월 27일 | 지민 고 |
| 기기 선택 및 식품 선택 | @2023년 8월 27일 → 2023년 8월 31일 | YJ R |
| 선택한 기기 및 식품 저장 | @2023년 8월 30일 → 2023년 8월 31일 | 지민 고 |
| 관리자 목록 수정 페이지 구현 | @2023년 9월 1일 → 2023년 9월 3일 | 지민 고 |
| 선택한 기기, 식품 목록 출력 | @2023년 9월 2일 → 2023년 9월 6일 | YJ R |
| 관리자 계정 추가 페이지 구현 | @2023년 9월 4일 → 2023년 9월 6일 | 지민 고 |
| 기기 및 식품 입력 페이지 구현 | @2023년 9월 6일 → 2023년 9월 10일 | 지민 고 |
| 매니저 리스트 기능 | @2023년 9월 6일 → 2023년 9월 10일 | YJ R |
| 관리자 표시, 추가, 삭제 | @2023년 9월 9일 → 2023년 9월 10일 | 지민 고 |
| 제출 | @2023년 9월 13일 → 2023년 9월 27일 | YJ R |
| 선택한 목록 표시 | @2023년 12월 1일 → 2023년 12월 2일 | YJ R |
| 치트 기능 | @2023년 12월 25일 → 2023년 12월 28일 | YJ R |
| 선택한 항목은 체크되도록 구현 | @2023년 12월 2일 → 2023년 12월 3일 | 지민 고 |
| 치트 - 사용 방법 및 주의사항 | @2023년 12월 28일 → 2023년 12월 29일 | 지민 고 |
| 치트 - 기기 및 식품 온도 범위 지정 페이지 | @2023년 12월 28일 → 2023년 12월 29일 | 지민 고 |
| 로딩 스피너 구현 | @2023년 12월 19일 → 2023년 12월 19일 | 지민 고 |
| 날짜 체크 알고리즘 구현 | @2023년 12월 22일 → 2023년 12월 22일 | 지민 고 |

구체적인 개발 계획을 설정했다. 계획을 기반으로 프로젝트를 진행해 원하는 배포날짜에 배포할 수 있었다.

# 프로젝트를 수행하면서 겪은 경험들...

## 1. Spring JPA update가 아닌 Insert가 되었던 경험




### 해당 로직의 문제점

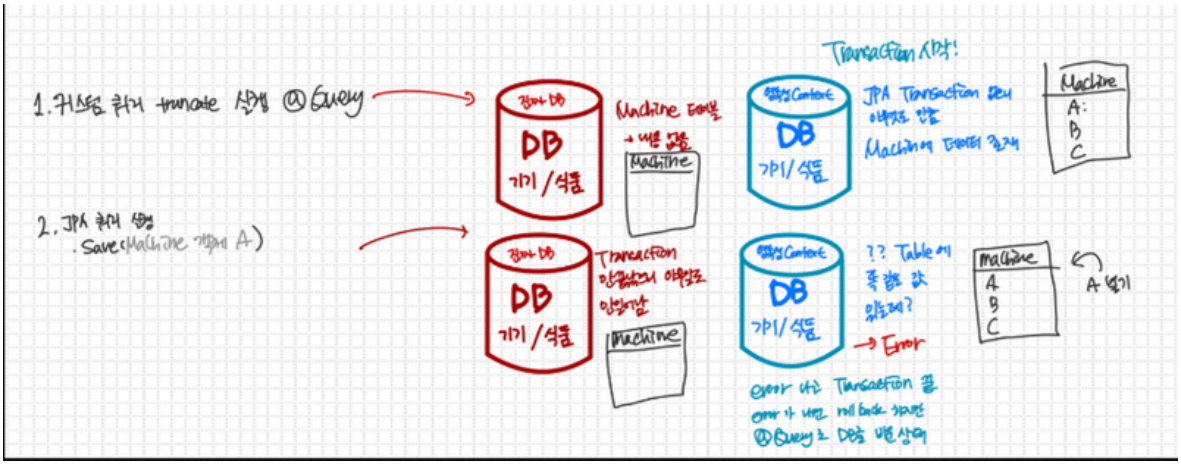
CustomMachine과 CustomFood에서 update를 수행하려면 테이블에 truncate를 수행하고 진행했었다.

```
@Transactional
public interface MachineRepository extends JpaRepository<Machine, Integer> {
    @Modifying(clearAutomatically = true)
    @Query(value = "truncate table Machine", nativeQuery = true)
    public void deleteAllMIne();

    @Query(value = "select * from Machine where id = :id ", nativeQuery = true)
    public Machine findMachineById(@Param("id") String id);
}
```

즉 위의 로직에서 DeleteAllMIne을 수행하고 JPA의 save메소드를 불러왔었다.
즉 로직의 순서가 JPQL → JPA이 두 가지를 함께 혼용해서 사용한 것이다.
이미지에는 @Modifying(clearAutomatically = true 가 있지만 에러가 났을 당시에는 없었다)

## 혼용으로 발생한 문제의 원인




해당 문제의 원인을 조사하기 위해 영속성 컨텍스트에 대해서 조사해보았다.
우선 TRANSCATION이 생성될 때 똑같은 PROXY DB인 영속성 컨텍스트를 생성한다. 즉 DB의 짝꿍을 생성하는 것이다. 하지만 JPQL같은 경우는 영속성 컨텍스트를 거치지 않고 DIRECT로 실제 DB에 적용이 되었었다.
그래서 한 메소드에 TRUNCATE(JPQL) 과 SAVE(JPA)를 사용하면 진짜 DB는 아무것도 없는 공란 상태가 되지만 영속성 컨텍스트에서는 데이터가 존재하기 때문에 ID가 같다는 에러가 발생하는 것이다.

## 해결안 1.

```
1 usage  new *
@Modifying(clearAutomatically = true)
@Query(value = "select from custom_food where id = id", nativeQuery = true)
public CustomFood findByidMy(int id);
```

JPQL을 작성해 DB에서 객체를 가져오는 형식을 사용하려 했다. 하지만 JPQL은 JPA와 달리 객체 형식을 반환할 수 없었고 오직 int형만 반환할 수 있었다. 따라서 위의 방법은 사용할 수 없었다.

## 해결안 2.

```
no usages  GoJiMin *
@Modifying(clearAutomatically = true)//insert update delete를 사용할 때 써줄 것
@Query(value="truncate table machine",  nativeQuery = true)
public void deleteAllMine(
);

2 usages  GoJiMin *
@Modifying(clearAutomatically = true)
@Query(value = "alter table machine auto_increment=1", nativeQuery = true)
public void initIncrement();
```

deleteAllMine 메소드에 @Modifying(ClearAutomatically =true)옵션을 주었다. 바로 영속성 컨텍스트를 비운다는 뜻이다. 그러면 영속성 컨텍스트는 truncate가 일어난 뒤의 DB의 본을 뜬다. 즉 처음에 의도한 대로 아무것도 없는 상태에서 JPA를 사용해 save를 할 수 있도록 하는 것이다.

### 여전히 남은 문제점

정확히 말하면 update가 일어나야하는 부분이지만 안타깝게도 처음에 index의 내용을 지키기위해 DB의 내용을 전부다 지운 다음에 다시 넣는 방향으로 생각했다. DB에 들어가는 내용의 양이 약 30개정도로 적었기 때문에 해당 방법이 괜찮다고 그때는 그렇게 생각했었다. 해당 문제를 발견했을 땐 이미 구현이 상당부분 진행되었고 배포가 우선이라고 판단했기 때문에 리팩토링을 진행하지 않았다.

### 해당 문제 해결을 위해 수행해야 하는 작업

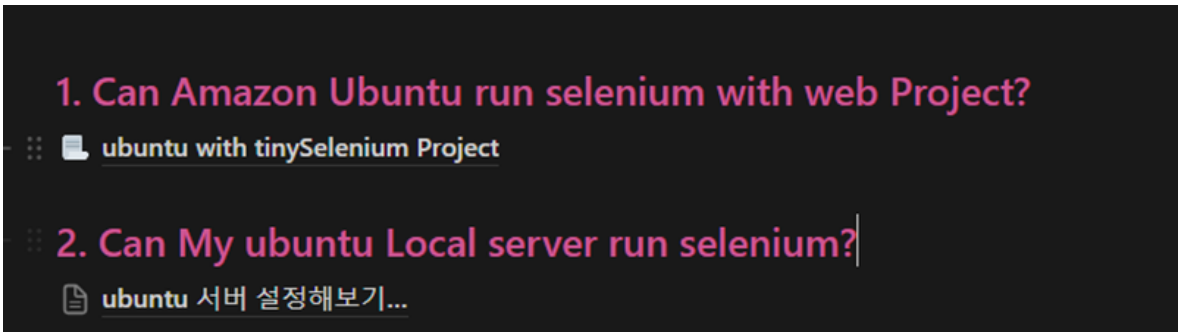
현재 DB에서 num을 pk로 사용하고 있다. 정확히 말하자면 num은 데이터의 순서를 의미한다. pk로 사용해도 괜찮지만 실제로 id라는 태그가 있고 해당 id는 데이터마다 가지는 고유한 값이기 때문에 id를 pk로 사용하는 것이 더 현명하다. 그 이유는 num값은 크롤링할 때 가져올 수 있는 값이 아닌 서버에서 부여하는 값이기 때문에 데이터와 연관성을 찾을 수 없다. 따라서 데이터를 update할 때, num이 아닌 id를 통해 찾아야 한다. 또한 값을 전부 삭제하고 다시 저장하는 것이 아닌 update를 수행해야한다. 그리고 가져온 객체를 num번호를 통해서 정렬 후 사용자에게 제공해야 한다.

## 2. AWS Freetier서버 사양문제

```
chrome is no longer running so chromedriver is assuming that chrome has crashed.
```

첫 배포를 수행할 때 마주쳤던 셀레니움 에러이다. 로컬 테스트에서는 정상적으로 작동했지만 실제 AWS에서 배포를 수행했을 때 Seleinum이 사용하는 Chrome Driver가 제대로 동작하지 않았다.

### 해당 문제의 해결 방안




두 가지의 가설을 정하고 실험을 해보았다. 첫 번째로 Ubuntu가 Selenium웹 프로젝트를 실제로 수행할 수 있는지 여부를 확인해보았다.

**tinySelenium Project 구동 결과**
TinySeleinum 프로젝트는 네이버 로그인을 Seleinum 봇으로 수행하는 것이다. 구동 결과는 두 가지로 나뉘었었는데

**1. headless옵션이 없는경우**
실패
**2. headless옵션이 있는 경우**
성공!

실제로 아마존 Ubuntu에서 tinySelenium 프로젝트가 돌아가는 것을 확인했고 그 전제 조건이 Selenium Web Driver의 headless 옵션이라는 것도 확인할 수 있었다.

### 결과

수행당시 서버는 CLI환경에서 구동되고 있었고, ubuntu에서 셀레니움을 돌리면 GUI모드로 실행됐었다. 셀레니움이 GUI모드로 실행되면, 이미지, 리소스등을 로딩해야하기 때문에 자원도 많이 소모되고 시간도 많이 걸린다. 즉 애초에 AWS Freetier는 GUI모드로 구동한다고 해도,내가 만든 프로그램을 구동할 수 없는 사양이었다.

### 해결안

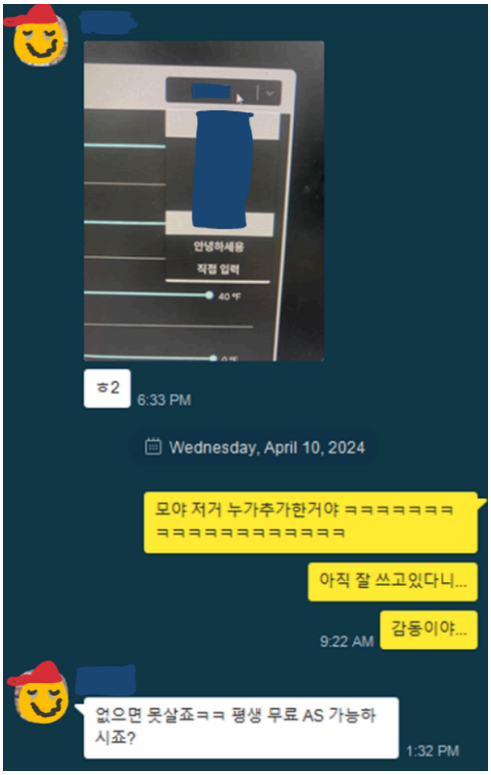
1번에서 headless옵션을 통해서버가 돌아가는 것을 확인했다. 하지만 한 가지 문제가 발생했었는데 그래도 여전히 느렸다는 것이다. 그때 당시 로딩의 속도는 5분이 넘었었고 jar파일 안에 front, backend 모든 기능이 다 들어가있어서 프로젝트가 AWS에겐 살짝 무겁기도 했다. 실제로 서버가 다운되는 경우가 잦았다.

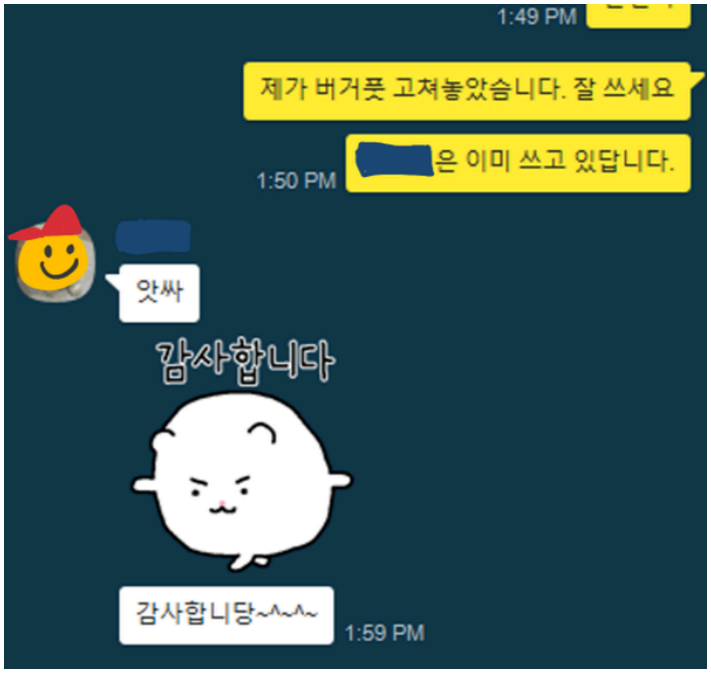
이를 해결하기 위해 SWAP메모리 2G를 추가적으로 설정해 메모리 부족문제를 해결했고, 그 후에도 발생했던 메모리 부족 문제를 해결하기 위해 front와 backend를 분리하여 서버의 부담을 줄일 수 있었다.

## 프로젝트를 수행하며 겪은 한 줄 경험들...

- JWT토큰을 사용한 안전한 로그인 시스템 구축 관련 블로그 링크
- 로딩 속도 리팩토링, 속도를 **120초에서 16초로 8배 향상** 관련 블로그 링크
- **HTTP 응답- 요청 쌍을 이해할 수 있었던** 제출로직 버그 수정 관련 블로그 링크

## 유저의 프로젝트 사용 후기




```
6월   3 20:46 zenputMachine2024-06-0320:46:59.248179831.png
6월   4 17:19 zenputMachine2024-06-0417:19:43.745214877.png
7월  10 16:46 zenputMachine2024-07-1016:46:40.124850461.png
7월  11 16:36 zenputMachine2024-07-1116:36:09.368418434.png
7월  15 16:33 zenputMachine2024-07-1516:33:05.963445976.png
7월  22 09:31 zenputMachine2024-07-2209:31:20.630520382.png
7월  23 16:40 zenputMachine2024-07-2316:40:28.455996325.png
7월  28 16:47 zenputMachine2024-07-2816:47:56.432882906.png
7월  29 16:36 zenputMachine2024-07-2916:36:45.615314023.png
7월  30 17:26 zenputMachine2024-07-3017:26:26.516437114.png
8월   2 16:40 zenputMachine2024-08-0216:40:45.543896561.png
8월   5 17:19 zenputMachine2024-08-0517:19:56.222709074.png
8월   6 18:43 zenputMachine2024-08-0618:43:07.113408538.png
8월   7 16:53 zenputMachine2024-08-0716:53:35.371851797.png
8월   8 09:37 zenputMachine2024-08-0809:37:13.387491351.png
8월   9 08:55 zenputMachine2024-08-0908:55:14.267456151.png
8월  12 16:51 zenputMachine2024-08-1216:51:58.403557718.png
8월  13 17:22 zenputMachine2024-08-1317:22:38.539062735.png
8월  17 09:31 zenputMachine2024-08-1709:31:27.513998512.png
8월  17 16:32 zenputMachine2024-08-1716:32:56.958393537.png
8월  18 17:57 zenputMachine2024-08-1817:57:11.096152905.png
8월  19 09:13 zenputMachine2024-08-1909:13:30.854982391.png
8월  24 08:38 zenputMachine2024-08-2408:38:44.327018982.png
```

제출하면 자동으로 저장되는 이미지 로그, 매달 사용자가 잘 사용하고 있음을 보여준다.