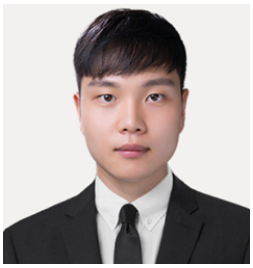

구동하 서버/백엔드 개발자

01068149669 | koo9811@naver.com

포트폴리오 블로그

개발을 좋아해 주 100시간 이상 몰입하는 [KAIST 비학위과정 SW사관학교] 커리큘럼 기반 과정을 이수했습니다. Pintos를 통해 OS 기능을 직접 구현하며 얻은 CS 기초지식을 통해 동시성 문제, 메모리 문제를 해결할 수 있으며 프로젝트 때 속도 문제를 해결하기 위해 Redis를 도입하는 등 주도적으로 문제를 분석해 솔루션을 찾아 개발합니다.

현재는 혼자 [Java 백엔드 프로젝트]를 진행하며 [MSA]를 익히고 [대용량 트래픽]을 대처하는 등 실제 현업에서의 문제해결 역량과 [알고리즘 공부]를 지속하며 자료구조와 수학적 사고능력을 꾸준히 키우고 있습니다.

프로젝트

주식 종목 토론장

2024.07 ~ 2024.08

Docker, MSA, 대규모 트래픽 처리 등 현업에서의 문제를 직접 겪어 보기 위해 도전한 개인 프로젝트

• 프로젝트 개요

Backend : Java / Spring Boot / Kafka / Docker / MSA / Redis / JWT / MySQL

Frontend : React / CSS / ReactDom / ReactRouter / MUI / Axios

인원 : 개인 프로젝트

구현 기능 : 주식 차트, 종목 별 게시판 등 기본적인 소셜 서비스, 뉴스피드 등

• 로그아웃을 Redis를 사용한 블랙리스트 방식으로 개선하여 토큰 관리 효율성 및 보안 강화

문제 상황 발생

- RefreshToken을 삭제하는 로직을 구현하였으나 클라이언트에 Token 정보가 남아 있다면 로그아웃을 했음에도 계속 로그인이 가능한 문제 발생
- 이는 곧 보안 문제와 직결되기 때문에, 빠르게 해결하고자 하였음

원인 추론

- AccessToken 기간을 하루로 잡아 놓았기 때문에, 로그아웃을 시켰어도 AccessToken 기간 이내임과 동시에 Token이 남아 있다면 로그인 과정을 거치지 않고도 계속 로그인이 가능
- 그러나 너무 짧은 AccessToken 기간은 사용자 입장에서 불편함을 느낄 우려가 있음

해결 방안

- Blacklist에 임시로 Token 정보를 담아 놓음으로서, 로그아웃 후 Tok
en이 있어도 해당 Token으로는 서비스를 이용하지 못하게 차단
- BlackList 때문에 DB에 테이블을 따로 설정하는 것이 비효율적으로 느껴졌고
, Redis를 도입해 임시로 캐시에 Token을 저장하는 방식을 통해 DB 공간을
절약

결과 확인
- 로그아웃 후 기존의 Token이 있더라도 다시 로그인 불가
- 유저 보안 및 사용자 경험 증대

- **데이터베이스 비용 및 쿼리 속도 50% 절감**

문제 상황 발생
- 중복되는 데이터베이스들로 인해 데이터베이스의 크기 증가 및 쿼리 속도 감소 현상
발생
- 각각의 서비스의 RDS 비용이 예상보다 높게 산정됨

원인 추론
- 완전한 MSA를 구현하기 위해 각 서비스마다 데이터베이스를 만들었는데, 이로 인해
중복되는 데이터도 다 따로 테이블을 만들어 저장해야 했음

해결 방안
- 각 서비스에 있는 데이터베이스를 모두 합치고, SOA 방식으로 변경하기로 결정

결과 확인
- 기존의 서비스보다 RDS 비용 일부 감소
- 쿼리 속도 50% 절감

- **API-GATEWAY를 이용한 보안 강화 및 회복탄력성 구현**

문제 상황 발생
- 로그인을 하지 않아도 일부 서비스에서 엔드포인트로 접속이 가능한 상황 발생
- 한 서비스가 중단되면 연동되어 있는 다른 모든 서비스들을 실행할 수 없게 됨

원인 추론
- 처음 로그인을 할 때만 JWT를 검증하고, 그 이후 모든 서비스에는 따로 검증 기
능이 없기 때문에 다른 서비스에 API를 요청하면 인증 없이 엔드포인트에 도달
- 한 시스템의 장애가 다른 서비스에까지 영향을 미침

해결 방안
- API-GATEWAY에 JWT 필터를 만들어, 모든 서비스들에 요청이 들어오면 J
WT 검사를 먼저 선행하게 함
- BlackList 때문에 DB에 테이블을 따로 설정하는 것이 비효율적으로 느껴졌고
, Redis를 도입해 임시로 캐시에 Token을 저장하는 방식을 통해 DB 공간을
절약

- Resilience4j를 사용해 서킷 브레이커 및 재시도 매커니즘 도입을 통해 일시적인 네트워크 에러로 인한 중단을 막고 장애가 있는 서비스의 추가 호출을 막음

결과 확인

- 모든 서비스의 보안 향상
- 회복탄력성 구현을 통한 사용자 경험 증대

Github https://github.com/Acacian/Stock

ONCORE

2024.04 ~ 2024.05

교육 과정이 끝나고도 계속 온라인으로 함께 알고리즘 공부를 하기 위해 고안 및 배포했던 프로젝트

- **프로젝트 개요**

  기술 : Nest.js / React.js / Redis / MySQL / Socke
  t.io / Clova AI / CodeMirror / AWS EC2 , Ngnix
  / Puppeteer / Git
  인원 : 5명 (BE 2, FE 2, FS 1)
  구현 기능 : Rest API, 실시간 채팅, 백준 문제 크롤러, AI 보조

- **Redis 캐싱을 이용해 유저에게 백준 문제 데이터를 불러오는 속도 75% 개선**

  문제 상황 발생
  - 모든 유저가 여러 가지 종류의 알고리즘 문제를 크롤링하여 불러오는 데 시간이 오래 걸림. 특히, 바이너리 파일이 있는 특정 문제에 대한 데이터를 가져오는 데 시간이 많이 소요됨.

  원인 추론
  - 크롤링 방식으로 문제를 불러오면서 서버의 과부하 및 네트워크 지연이 생김
  - 동일한 문제를 반복해서 불러오는 경우가 많아 비효율적임.

  해결 방안
  - Redis를 도입하여 인메모리 캐싱을 사용해 원인을 개선하는 방향을 채택
  - 문제 주소를 캐싱해 다른 유저들이 더 쉽게 데이터를 꺼낼 수 있도록 구현:
  1. 문제 ID > 문제를 고유하게 식별하는 ID.
  2. 문제 주소 > 문제의 URL 주소.
  - 요청 시 캐시된 주소가 없을 경우 크롤링을 통해 데이터를 가져와 캐싱.
  - 문제가 업데이트될 때마다 Pub/Sub 시스템을 사용하여 Redis 캐시를 갱신.

  결과 확인
  - 데이터 접근 시간 단축으로 문제 불러오기 속도 75% 개선, 서버 부하 60% 감소.
  - 유저들이 백준 문제를 더 빠르게 불러올 수 있게 되어 사용자 경험 향상.

- **비동기 처리로 AI 응답 대기 시간 70% 개선**

문제 상황 발생
- AI 채팅과 일반 채팅 동시 사용 시 중요성이 높은 AI 응답이 지연되거나 일반 채팅이 먼저 게시되는 등 사용자들이 AI 응답을 받기까지 기다려야 하는 상황이 발생함.

원인 추론
- AI 응답과 일반 채팅을 동시에 처리하기 때문에 AI 응답이 지연될 경우, AI 응답의 중요성이 반영되지 않아 전체 시스템의 성능이 저하됨.

해결 방안
- AI 응답을 async/await을 사용해 비동기적으로 처리하여, AI 응답이 완료될 때까지 일반 채팅을 차단함으로써 AI 응답 처리 속도를 높임.

결과 확인
- AI 응답 대기 시간 70% 개선.
- 우선도가 높은 AI 응답이 완료된 후 일반 채팅이 원활하게 재개되어 사용자 경험이 크게 개선됨.

- **수많은 에러들을 줄이기 위해 테스트 환경 변화를 통한 버그 빈도 100% 개선**

  문제 상황 발생
  - 컴파일 시 에러가 나지 않으면 주요 기능만 확인하고 바로 Merge, 종종 Redirect 에러 발생.

  원인 추론
  - 충분한 테스트 없이 Merge된 코드가 다른 컴포넌트와 충돌하여 에러 발생.

  해결 방안
  - Git을 사용하여 이전 안정된 버전으로 복원, UI/UX를 전면 재구성하여 Redirect 문제 해결.
  - 이후 단위 테스트 및 통합 테스트 작성, 코드 리뷰와 테스트 절차 후 Merge 및 배포.

  결과 확인
  - 테스트 커버리지 93% 이상 달성, 기능 통합 시 발생하던 버그 빈도 100% 감소.
  - Redirect 문제가 완전히 해결되고, 사용자 경험이 크게 향상됨.

- **Websocket, Pub/Sub을 활용해 실시간 채팅 성능 문제 70% 개선**

  문제 상황 발생
  - 모든 채팅 데이터를 MySQL에서 가져오는 방식으로 인해 응답 시간이 길어짐.

  원인 추론
  - 채팅 데이터의 양이 많아질수록 MySQL에서 데이터를 가져오는 데 시간이 오래 걸림.

해결 방안
- Socket.io를 사용하여 실시간 채팅 기능을 구현, 메시지를 Redis에 저장
해 캐싱하기로 결정.

결과 확인
- 실시간 응답 속도 70% 개선.

Github https://github.com/Acacian/ONCORE
Youtube https://www.youtube.com/watch?v=vHHZA0hJiAY

Pintos

2024.02 ~ 2024.04

OS의 기능들을 직접 C/Linux 환경에서 구현하는 프로젝트

- **프로젝트 개요**

  기술 : C / Linux(Ubuntu 20.04) / GDB / GNU Make /
  AWS EC2 / Git
  인원 : 3명
  구현 기능 : Thread, System Call, Virtual Memory

- **Thread Priority 역전 문제 해결**

  문제 상황 발생
  - 프로세스 내 쓰레드의 Priority대로 실행하게 했는데, 더 중요한 쓰레드가 L
  ock 때문에 실행되지 못하고 후순위의 쓰레드가 실행되면서 우선순위의 역전이 일어남.

  원인 추론
  - 높은 우선순위의 쓰레드가 낮은 우선순위의 쓰레드가 점유한 Lock을 기다리게 되면
  서 발생하는 것으로 추정.

  해결 방안
  - Priority Donation 기능을 구현하여, 높은 우선순위의 쓰레드가 Loc
  k을 기다릴 때, 낮은 우선순위의 쓰레드에게 자신의 우선순위를 양도하도록 함.
  - Lock을 해제할 때, 원래의 우선순위로 되돌림.

  결과 확인
  - 우선순위 역전 문제 해결.
  - 시스템의 전반적인 성능이 30% 개선됨.
  - 주요 작업의 지연 시간이 40% 감소됨.
  - 중요한 작업이 적시에 완료될 수 있도록 보장함.

- **과도한 메모리 소모 문제 해결**

  문제 상황 발생

- RAM의 용량을 넘는 소프트웨어의 실행이 아예 불가능하고, 모든 소프트웨어의 내용이 RAM에 적재되기에 부하가 매우 큼.

원인 추론
- 프로그램이 실행될 때 모든 데이터를 한 번에 RAM에 적재하려고 하므로, 메모리 용량을 초과하고 시스템 부하가 발생하는 것으로 추정.

해결 방안
- Virtual Memory를 구현하여, 실제 물리 메모리(RAM)의 크기에 관계없이 더 큰 프로그램을 실행할 수 있도록 함.
- Lazy Loading 방식을 통해 프로그램의 전체 데이터를 한 번에 적재하지 않고, 실제로 필요할 때마다 데이터를 적재함.
프로그램 실행 시, 필요한 최소한의 데이터만 메모리에 적재.
나머지 데이터는 디스크에 저장하고, 실제 사용 시점에 메모리에 로드.

결과 확인
- 메모리 사용 효율성이 50% 향상됨.
- RAM의 용량을 초과하는 프로그램 실행 가능.
- 시스템의 과도한 부하가 60% 감소됨.
- 프로그램 실행 속도가 35% 개선됨.

Github https://github.com/Acacian/DHPintos
Blog https://blog.naver.com/koo9811/223370655840

경력 총 7년

공군교육사령부

2017.03 ~ 2024.03 (7년)

항공통제 / 중사

- **여러 항공작전의 핵심 역할 수행**

  평시에는 전투기의 원활한 임무 수행 및 북한의 특이사항 감시에 힘썼으며, 다양한 부서와

  긴밀하게 협력하여 문제를 해결했습니다. 이러한 경험을 통해 팀워크와 커뮤니케이션 능력
  을 키웠고 업무 성과를 인정받아 2023년 한미연합훈련에 참여해 미군을 포함한 여러 이
  해관계자들과 같이 임무하며 훈련을 성공적으로 마쳤습니다.

학력

평생교육진흥원
2022.02 ~ 2023.07

컴퓨터공학과

경일대학교
2020.02 ~ 2021.06

미래융합학과

기타 활동

크래프톤 정글 4기 수료
2024.01 ~ 2024.05

KAIST의 비학위과정 SW사관학교 정글 커리큘럼 기반으로 운영되는 과정으로
Pytho
n, 알고리즘부터 C언어, 자료구조, 네트워크, 운영체제 등 컴퓨터공학과에서
대부분 이
론적으로만 배우는 지식들을 5개월간, 주말 없이 매주 100시간 이상
몰입하여 직접 구
현하면서 체화하고 최종 프로젝트를 통해 실무에서 사용되는
프레임워크를 다루며 개발자로서
의 기본기를 다지는 과정입니다.

DevFest Cloud Busan 2023 참여
2023.12 ~ 2023.12

마이크로소프트 MVP 한상곤 개발자님이 참여하신 행사에서 회고의 필요성과
개발 팀에게
필요한 효율적인 회고 방식에 대한 내용을 알려주는 세션이었습니다.

2022 Startup Weekend Busa n 해커톤 참여
2022.07 ~ 2022.07

외국인들과 함께 5~6명이서 팀을 짜서 하나의 프로덕트를 만드는 해커톤이었습니다.
3일
간 진행되었으며, 여행객들에게 그곳에 사는 사람들이 실제로 맛집이라고 생각하는
곳들을
알려 주는 프로덕트를 제작했습니다. 대본 제작과 발표 및 Python으로
크롤링하여 데
이터를 모으는 작업을 맡아 진행했습니다.

ADSP(데이터분석 준전문가)
2020.12

빅데이터에 대한 관심으로 취득한 자격증입니다.

자동차운전면허증 2급

2016.12

군 생활을 하며 자차로 3년 반 정도 운전했습니다.