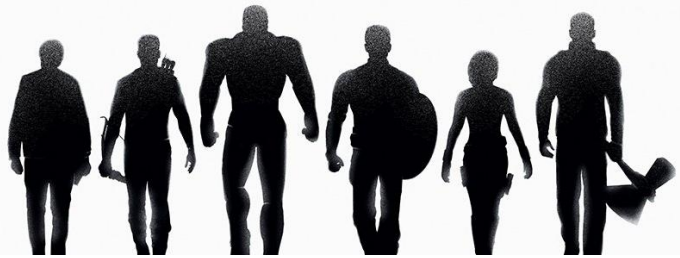
Best of Your Career

May.19.



Pray2U

Best of Your Career

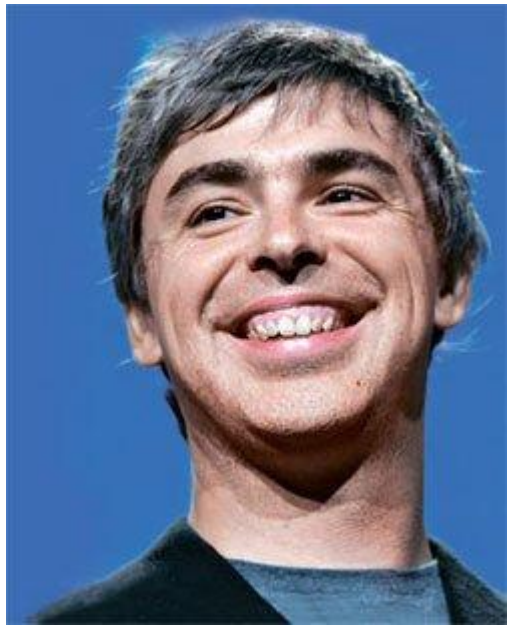
- 나는 어떤 개발자가 되고 싶나요?
- 침대위에서 일으킬 수 있는 가장 작은 기적, 밍기적
- 나.. 꽤 잘하고 있을지도..?
- 이대로 공부해서 취업까지 원 큐에 !
- 모두를 위한 커리어 준비

Choi JaeHun

나는 어떤 개발자가 되고 싶나요?

나는 어떤 개발자가 되고 싶나요?

Python의 창시자 **Guido Van Rossum**



Google의 공동창업자 **Larry Page**

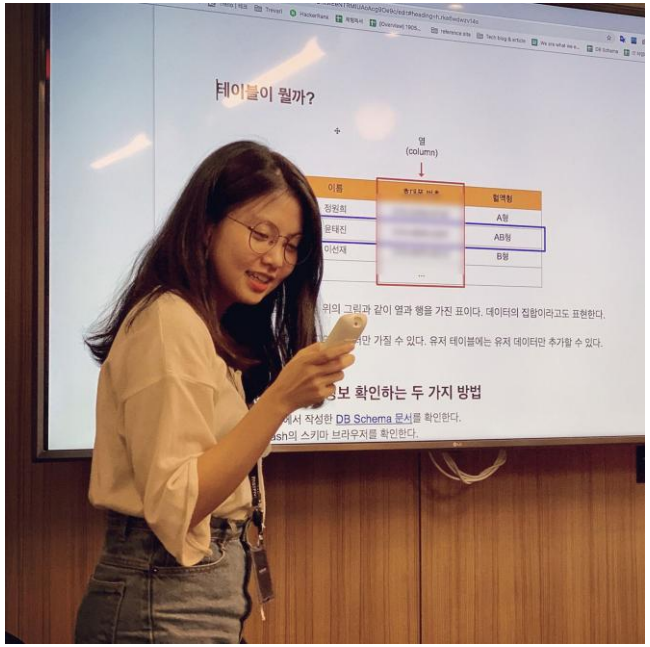
Spring의 아버지 **Rod Johnson**

나는 어떤 개발자가 되고 싶나요?

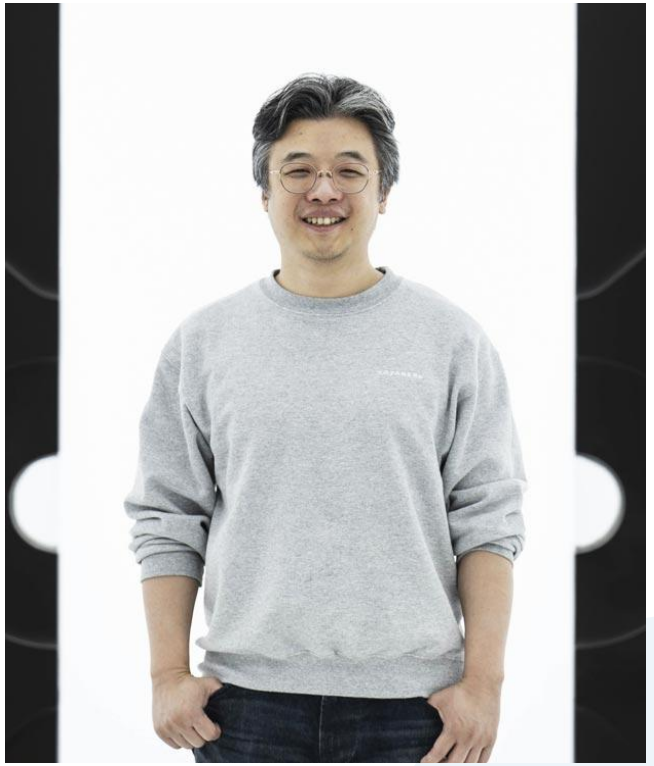
교수님 사랑합니다 ^^

나는 어떤 개발자가 되고 싶나요?

前 Facebook Engineer, **천인우**



8년차 셀럽 개발자, **정원희**

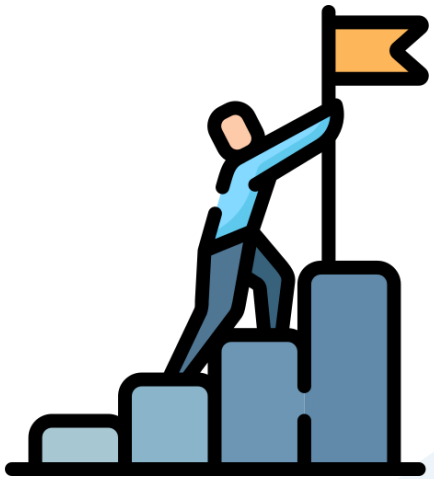


voyager X CEO, **남세동**

나는 어떤 개발자가 되고 싶나요?

목표

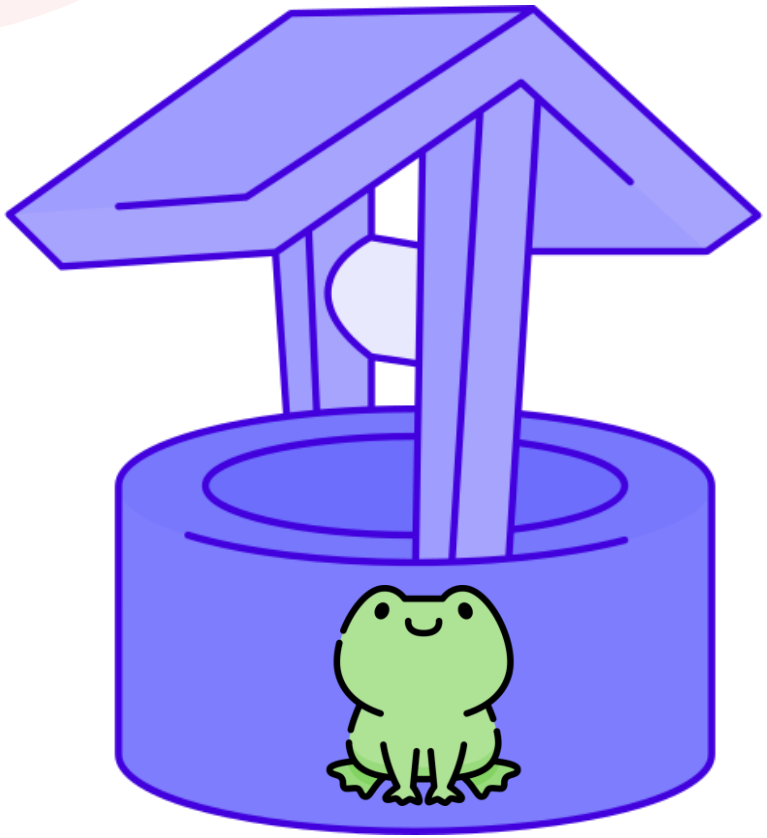
Role - Model



방법

침대위에서 일으킬 수 있는 가장 작은 기적, **밍기적**

나.. 꽤 잘하고 있을지도..?



이대로 공부해서 취업까지 **원 큐에 !**

개인의 성장

취업의 성공/준비

이대로 공부해서 취업까지 **원 큐에 !**

☺ 개인의 성장

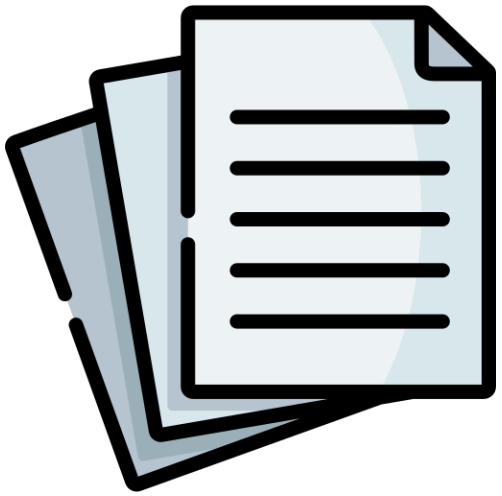
꾸준한 공부

커리어 탐색

가끔씩 휴식

이대로 공부해서 취업까지 원 큐에 !

😊 취업의 성공/준비

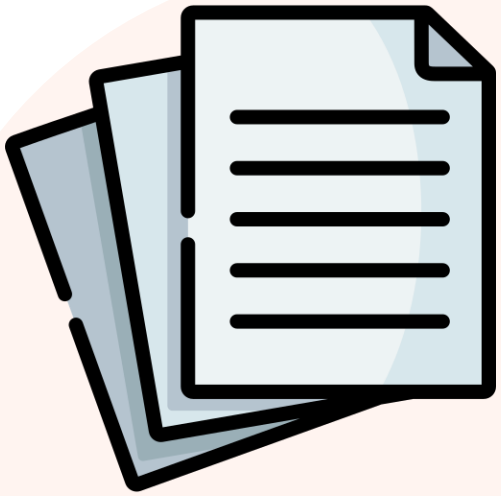


이력서

포트폴리오

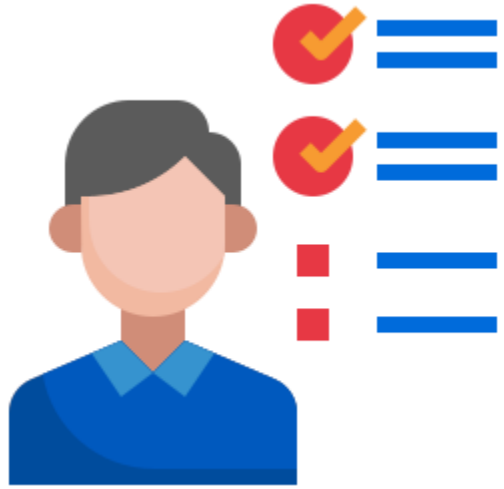
냉정한 본인평가

모두를 위한 커리어 준비 (feat. 대학생)



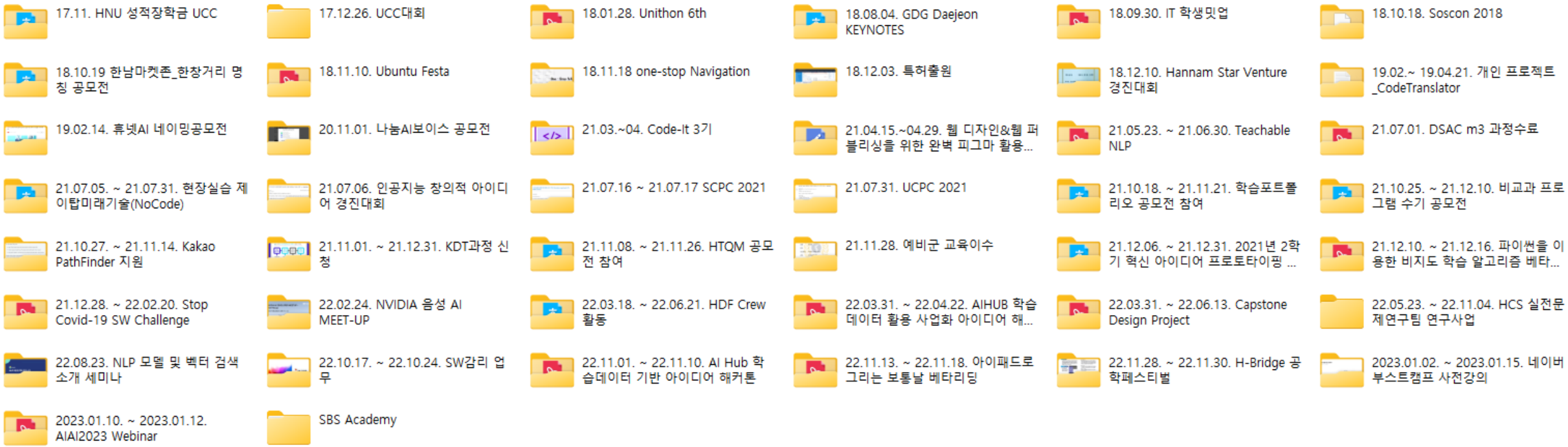
이력서

포트폴리오



냉정한 본인평가

모두를 위한 커리어 준비 (feat. 대학생) + 한남대



모두를 위한 커리어 준비 (feat. 대학생) + 한남대

1단계 : 프로그램 언어를 배우는 단계

- 절차적, 논리적으로 펼쳐내기(`컴퓨터적 사고` / `Computational Thinking`)
- `프로그램 언어`의 문법 등 개념 이해 및 응용
  ex) 변수, 조건문, 순환문, 함수 개념이해와 응용

다른 언어도 기본 구조는 같구나 / 시간 내서 배우면 응용할 수 있겠구나

2단계 : 데이터에 대해 배우는 단계(신입)

- `데이터의 이해`(데이터 기준-선언-분류-처리)
  ex) 학생 - 이름, 학번, 성적, 개인정보 등 선언 및 처리
- 프로그램 언어 명시 또는 묵시적으로 처리하는 `데이터 타입`의 다양성과 그 `데이터 처리`를 위한 구조적 표현방식을 개념적으로 이해
  ex) `변수` 타입, `자료구조`, `알고리즘`, `객체지향언어`

이제 개발자들과 최소한의 의사 소통능력이 갖출 수 있구나

3단계 : 프레임워크에 대한 이해(주니어)

- `프레임워크`도 개발자가 만들었다는 점을 수용
  ex) `운영체제`, 표준라이브러리, 컴파일러, IDE, CI 등
- 프레임워크의 `내부를 디테일하게 검토`
  ex) API함수, 기능동작, 리턴값, 자료전달, 에러메세지 `매뉴얼에 있는 내용` 이해

품질이 높은 개발을 할 수 있는 단계 → 개발자로서 일을 할 수 있는 수준

4단계 : 효율적으로 알고리즘을 개선할 수 역량(시니어)

- 컴퓨터 자원(CPU, 메모리, 디스크)의 효율성을 높일 수 있는 `알고리즘 개선`
  ex) 되는 것→효율적인 것→누가봐도 효율적인 것
- `문제의 패턴이해` - 실제 프로젝트 경험을 통해 교과서에 나온 연습문제들이 왜 연습 문제인지 이해하고 해결하는 과정
  ex) 운영체제 - 동시성/병렬성/동기화 등

만렙 개발자 = 시니어 개발자

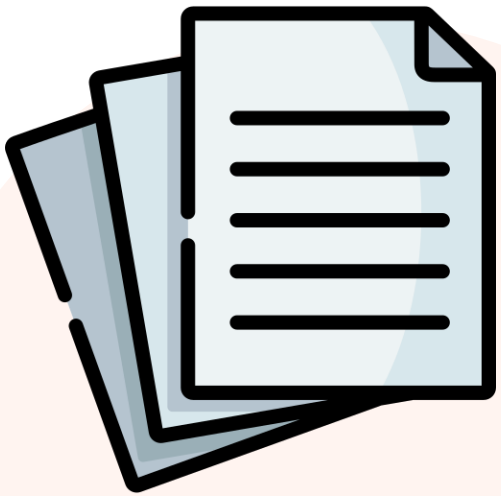
모두를 위한 커리어 준비 (feat. 대학생) + 한남대

기본기 + 사이드 프로젝트

모두를 위한 **커리어** 준비

- 서류전형 (= 이력서 검증)
- 코딩테스트 / 개발 과제 (온라인, 오프라인으로 코딩능력 검증)
- 기술면접 (실무자의 직접검증)
- 인성면접 / 컬쳐 핏 (회사의 문화에 어울리는지 검증)

모두를 위한 **커리어** 준비



이력서

- ✨ 본인을 전체적으로 드러내는 것, 양이 많을 필요 없다. (Max 2~3장)
- ✨ 이력서는 ‘내가 하고싶은 말’이 아닌 ‘남이 듣고 싶은 말’을 쓰는 것.
- ✨ 내용의 “압축 " 과 “제외 " 는 다르다.
- ✨ 간단한 자기소개와 본인 사진, 기술적 경험, 보유 기술, 자격증 등등
- ✨ Github 관리 / 본인 개발 블로그 / 진짜 이력서 문서작성

모두를 위한 **커리어** 준비

포트폴리오

- ✨ 자기암시 “나는 전문가다 "
- ✨ 과정 그대로를 기록할 것.
- ✨ 중요한 것은 어떠한 과정속에서 무엇을 배울 수 있었는지!
- ✨ 기술력만이 모든 문제의 해답이 아니다.
- ✨ 결국 아무리 프로젝트가 우수해도 표현을 못하면 의미없음!

이력서:

인적사항	성명	
	이메일	
	Blog	https://keydo.tistory.com/
	GitHub	https://github.com/NamKey
학력	2011.02 ~ 2019.02	의료전자시스템공학과 졸업
보유 기술	#Java #Kotlin #Spring #Spring Boot #JPA #MySQL #Redis #Pinpoint #Prometheus #Grafana	
프로젝트	프로젝트명	YOUSINSA
	소개	MUSINSA와 같은 패션 E-commerce 서버 개발
	기술	Java/Spring/JPA/MySQL/Redis/Pinpoint/Prometheus
	소속	개인 프로젝트 (참여인원 : 2명, Back-End : 2명)
	기간	2022.04 ~ 2022.08
	주요 성과	- Database Connection Pool 크기 최적화 - Write-Skew로 인한 재고 데이터 불일치 방지 - 재고 데이터 캐싱과 Event Pub/Sub 구조 도입 - Redis operation 최적화
	프로젝트명	DoPathology
	소개	AI를 활용한 자궁 경부암 진단 서비스 서버 개발
	기술	Kotlin/Spring/JPA/QueryDSL/MySQL/Redis/RabbitMQ/Prometheus
	소속	DoAI (참여인원 : 5명, Front-End : 2명, Back-End : 2명, AI : 1명)
	기간	
	주요 성과	- Paging이 있는 쿼리에서 N+1 문제 개선 - Application과 AI Engine 연동 - Legacy 코드 개선
	프로젝트명	DoPapsmear
	소개	개발 도상국을 위한 자궁 경부암 진단 플랫폼 개발
	기술	Kotlin/Spring/JPA/QueryDSL/MySQL/Android
	소속	DoAI (참여인원 : 2명, Android : 1명, Back-End : 2명)
	기간	
	주요 성과	- 확장성 있는 App 구조로 개선 후 테스트 배포 - Coroutine을 활용한 이미지 프로세싱, AI 추론 속도 개선 - 서버 설계 및 App 연동 초기 구현
경력	기간	2020.11 ~ 2022.06
	회사명	(주) 두에이아이(DoAI)
	주요 업무	- 백엔드 개발 (2021.03 ~ 2022.06) [Kotlin, JPA, MySQL, Redis, QueryDSL, Prometheus, Grafana] • Spring 프레임워크를 통한 REST API 개발 - 앱 개발 (2020.11 ~ 2021.02) [Kotlin, Flow, Coroutine, Jetpack Compose, Hilt] • Android Native App 개발
	기간	2018.11 ~ 2020.10
	회사명	(주) 브라이토닉스이미징
	주요 업무	- 의료영상 기기 데이터 분석 선행연구 [Matlab, Python, Hardware] - 의료 영상 기기 회로 설계, 성능 평가 • 전처리를 통한 성능 평가 • 성능 평가 알고리즘 개선
교육 활동	활동명	ICC 인재양성 프로그램
	기간	2018.07 ~ 2018.09
	설명	[redacted]방사선학과 연구실과 (주)바텍 주관으로 이미지의 노이즈를 웨이블렛 변환을 통하여 디노이즈 기술을 개발하는 참여프로그램
	활동명	[redacted]의료전자기시스템 연구실 학부연구생
	기간	2015.03 ~ 2018.06
	설명	기술이전, 시제품 개발의 과정을 지도 교수외 2명(박사 1명, 학부연구생 1명)과 수행
기타사항	수상내역	- ICC 인재양성 프로그램 과제 경진대회 우수 - [redacted]산학기술협력기술동아리 경진대회 장려

포트폴리오:

YOUSINSA

2022.04 ~ 2022.08 (참여인원 : 2명, Back-End : 2명)

github : https://github.com/NamKey/yousinsa
project tech-blog : https://keydo.tistory.com/category/Project/YOUSINSA

#Java #SpringBoot #Spring Data JPA #MySQL #Redis

#JUnit5 #Mockito

#Jenkins #Pinpoint #nGrinder #Prometheus #Grafana

프로젝트 요약

- MUSINSA와 같은 의류 종합 플랫폼의 트래픽을 처리할 수 있는 백엔드를 개발하는 프로젝트입니다.
- UI 부분은 카카오 Oven을 통하여 대체하고 App/Web에서 공통으로 사용하는 API 백엔드 서버 개발에 초점을 맞춰 진행했습니다.
- Unit Test, 코드의 재사용성과 유지보수성을 고려하여 프로젝트를 진행했습니다.
- 성능을 개선하는 과정에서 Stress Test Tool과 APM Tool을 사용하여 병목 지점을 찾아 여러 방법을 시도하여 해결책을 찾아 성능 튜닝을 진행했습니다.
- 초기 성능 개선 목표를 국내의 500명 이상의 동시 사용자를 처리하는 것으로 선정하고 테스트 후 튜닝 포인트들을 찾아 성능 테스트를 진행했습니다.

프로젝트 소개

MUSINSA, 29CM, ZIGZAG와 같은 패션 도메인의 E-commerce 서비스를 개발하는 프로젝트입니다. 웹 서비스 전반적인 개발을 진행하기보다 더 많은 트래픽을 처리 하기 위한 백엔드 개발을 중점적으로 프로젝트를 진행했습니다. 더 많은 트래픽을 처리하기 위한 방법을 바로 적용하는 것이 아닌 많은 트래픽이 발생했을 때 병목이 발생하는 지점들을 APM 툴을 사용하여 분석하고 이를 해결하기 위한 여러 방법 중 Trade-Off를 고려 후 프로젝트의 성격에 맞게 선택하여 Over-Engineering을 하지 않는 방법을 찾기 위해 고민하며 진행했습니다.

OOP 원칙들이 반영되어 있는 Spring을 사용하면서 확장 포인트들을 이해하고 활용하면서 확장성과 재사용성이 높은 코드를 작성하기 위해 노력했습니다. 확장성과 재사용성을 높이기 위한 코드를 작성하기 위해서 Unit Test를 작성하는 과정을 거쳐 기능 간의 의존도를 낮추는 방법을 적용했습니다.

Technical-Specification 작성을 통한 소통

프로젝트 초기에는 요청된 PR만 가지고 코드 리뷰를 진행했습니다. 코드를 작성한 입장에서는 당연히 도메인 리소스 간의 관계와 어떤 방식으로 비즈니스 로직이 작동할지 알 수 있습니다. 하지만 코드를 리뷰하는 입장에서는 작성한 코드가 어떻게 동작할지 PR을 전체적으로 확인하고 이해하는 과정과 서비스 도메인의 특성을 고려하는 과정이 모두 필요하다는 것을 알게 되었습니다. 또한 협업하는 개발자와 정해진 규약 없이 코드를 작성하면서 전체적인 코드의 통일성이 깨지는 문제가 발생했습니다.

이런 문제를 해결하기 위하여 맡은 기능에 대한 Technical-Specification을 작성했습니다. 문서에는 어떤 기능을 개발하고자 하는지, 왜 이 기능이 필요한지, 해당 기능을 개발하면서 예상되는 문제를 함께 명시하고 구조에 대한 부분을 클래스, 시퀀셜 다이어그램을 통해 표현하여 팀 전체가 기술 명세를 통하여 얘기하고 개선점들을 얘기할 수 있었습니다. PR 리뷰시에도 코드 작성자의 의도를 Technical-Specification을 통해 인지하고 리뷰하여 PR의 내용을 더 쉽게 이해할 수 있게 되어 생산성을 높일 수 있었습니다.

Database Connection Pool 크기 최적화

기획한 기능들을 완료하고 nGrinder를 테스트 툴로 사용하고 Pinpoint, Prometheus를 모니터링 툴로 사용하여 처리할 수 있는 트래픽에 대한 테스트를 진행했습니다. 모니터링 결과 응답 속도가 느린 요청의 경우 쿼리 수행 시간은 빠르지만 사용하고 있는 Database Connection Pool에서 데이터베이스와의 connection을 갖고 오는 시간이 대부분을 차지했습니다. 사용하고 있는 DBCP 라이브러리는 HikariCP로 다른 라이브러리보다 개선된 속도 성능을 갖고 있었고 기본으로 설정된 Pool의 크기가 충분하지 못해 connection을 갖고 오는 시간이 지연된다고 가설을 세우고 실험을 진행했습니다.

초기 테스트는 connection의 개수를 2개부터 20개까지 2개씩 증가시키며 2분간 측정 후 변동 폭이 커지는 connection 수를 선별하여 10분간의 테스트를 진행했습니다. 실험 결과 Pool의 크기가 클수록 더 많은 요청들을 처리할 수 있을 것이라 예상했지만 오히려 특정 수치를 넘으면 TPS 성능이 저하되는 것을 알 수 있었습니다. 이 실험을 통해 Connection Pool에서 connection들을 처리하는 것은 Thread이므로 많은 connection을 갖고 있다 하더라도 Context Switching이 자주 일어나게 되면서 성능이 저하된다는 것을 알 수 있었습니다.

최적화 실험을 통해 5 * vCPU(SSD 기준)라는 추정식을 세울 수 있었고 초기의 2vCPU 기준, 10개의 connection을 설정할 때 TPS 성능이 가장 높은 것을 확인할 수 있었습니다. 추정식을 검증하기 위해 vCPU를 Scale-Up하여 4vCPU로 한 경우도 추가적으로 테스트한 결과 20개의 connection을 설정할 때 TPS 성능이 가장 높을 것을 확인하여 추정식이 적합하다고 판단했습니다.

세션 불일치 문제

Scale-Up과 더불어 커넥션 점유 시간을 줄이는 방안을 적용했지만 동시 사용자 500명 정도의 트래픽을 받기에는 부족했습니다. 다시 Scale-Up을 수행하여 성능을 높일 수 있었지만 동시 사용자가 늘어날 경우마다 Scale-Up을 하는 것은 트래픽에 대해 유연하지 못한 구조가 된다고 판단했습니다. 또한 어플리케이션 서버가 단일 실패 지점이 되기 때문에

가용성을 위해서는 Scale-Out 방식을 도입할 필요가 있었습니다. Scale-Out 방식을 적용하기 위해서 세션 관리와 로드 밸런서를 적용하는 분산 환경으로 구조를 변경했습니다.

세션 관리가 필요한 이유는 분산 환경에서 Scale-Out 시 서버가 세션이라는 상태를 갖게 되면 확장하기 힘든 구조가 되므로 서버에 상태를 저장하지 않는 무상태 서버가 필요합니다. 구현하기 위한 방법으로 WAS를 활용하는 방법과 세션 스토리지를 활용하는 방법 중 어플리케이션 서버와 분리된 별도의 세션 스토리지를 선택했습니다. WAS를 사용하는 방식의 경우 Scale-Out시 추가적으로 리소스가 소모되는 데 반해 별도의 세션 스토리지를 분리하여 구축하면 어플리케이션 서버가 다운되는 상황이라도 유연하게 대처할 수 있을거라 생각해 Redis를 세션 스토리지로 선정했습니다. Redis와 함께 고려했던 세션 스토리지는 Memcached와 Hazelcast입니다. Memcached는 스냅샷 기능과 복제 기능이 없기 때문에 세션 스토리지에 바로 적용하기 적합하지 않다고 판단하였고 Hazelcast는 Core가 적은 경우 Redis보다 Throughput이 더 낮고 오픈 소스 커뮤니티의 크기가 작다는 점에서 초기 적용 시 Redis가 적합하다고 판단했습니다.

Scale-Out을 적용하면서 테스트 한 API들의 전반적인 TPS 향상은 있었으나 Database I/O가 많이 필요한 API의 경우 성능 향상 폭이 적었습니다. 이런 부분을 개선하기 위해 Cache-Layer를 도입하고 Eventually Consistency 모델을 통해 대규모 트래픽을 처리하기 위해 시도했습니다.

Write-Skew로 인한 재고 데이터 불일치 해결

재고 관리는 서비스의 불필요한 지출을 줄이고 주문이 성공했다고 나왔지만 추후에 주문이 취소되는 사용자 경험을 없애는데 있어서 재고 데이터의 불일치를 해결해야 된다고 생각했습니다. 트래픽 요청이 한 순간에 집중되는 경우 트랜잭션들간의 경합으로 인해 쓰기 스큐가 발생할 가능성이 높아집니다.

현재 사용하고 있는 MySQL에서는 기본적으로 트랜잭션의 격리 수준을 반복된 읽기 허용과 MVCC를 사용함으로써 지속된 읽기를 가능하도록 해줍니다. 하지만 이 상황에서도 쓰기 스큐는 발생하기 때문에 수정하고자 하는 개체에 잠금을 수행하는 방식을 생각했습니다. 잠금을 수행하는 방안으로 Redis를 통한 분산 잠금과 데이터 베이스의 읽기 배타적 잠금을 고려했습니다.

Redis를 통한 분산 잠금의 경우 Redis 클라이언트 중 Redisson을 통해 구현할 수 있었습니다. 분산 잠금을 통해 재고 데이터를 검증할 시 Write-Skew를 예방하여 정합성을 보장할 수 있었지만 재고 데이터를 수정하는 부분에서 병목이 발생했습니다. 원인은 데이터 수정에서 잠금이 수행되기 때문에 트래픽이 많아짐에 따라 쿼리 요청들이 기다리는 시간이 길어지고 분산 잠금의 더 긴 대기 시간이 필요했습니다. 분산 잠금 대기 시간을 충분히 길게 설정하는 경우 재고 검증 시 읽기 쿼리의 대기 시간이 길어짐에 따라 단일 구매 API 테스트의 TPS 성능이 Write-Skew 개선 전에 비해 60% 저하되는 결과를 보였습니다.

데이터 베이스 읽기 배타적 잠금을 사용하면 다른 트랜잭션에서 수정이 완료되기 전까지 조회와 쓰기 모두 불가능하기 때문에 마찬가지로 재고 데이터의 불일치를 해결할 수 있었습니다. 또한 지속적으로 발생했던 DeadLock 문제도 해결되며 10% 정도의 성능 향상이 있었습니다. 정합성 보장과 성능 향상까지 있었으나 TPS 성능으로는 160명 정도의 동시 사용자 밖에 처리하지 못했기 때문에 성능을 더 높일 필요성이 있었습니다.

재고 캐싱과 Event Pub/Sub 구조

구매 주문에서 재고 데이터의 정합성과 성능 모두 만족해야 됐기 때문에 재고를 캐싱하여 검증하고 이를 Spring의 Event Publisher를 함께 도입하여 비동기적으로 동작하도록 구조를 변경했습니다. 재고를 캐싱함으로써 데이터 베이스에서 읽어오는 것이 아닌 캐싱된 재고를 바탕으로 수량에 대한 검증을 진행하여 데이터 베이스의 정합성을 위해 사용했던 배타적인 잠금을 사용하지 않아 성능 향상을 얻을 수 있었습니다.

캐싱을 통해 현재의 재고를 갖고 있지만 이를 데이터 베이스에 바로 기록하게 되면 데이터 베이스에 병목 지점이 생깁니다. 응답에 데이터 베이스의 병목을 없애기 위해서 Eventually Consistency 모델을 적용하기 위해 Spring Event Publisher를 적용했습니다. 이벤트 리스너에서는 구매 주문 접수 이벤트가 발행되는 트랜잭션이 끝나게 되면 해당 이벤트를 받아 구매 주문의 상태를 업데이트하고 재고를 차감하는 방식을 비동기로 진행합니다.

주문 상태의 업데이트와 재고 차감을 비동기적으로 수행함으로써 궁극적으로는 정합성이 맞게되며 주문을 접수하는 API의 TPS 성능 또한 만족할 수 있었습니다. 해당 구조를 도입한 결과 단일 품목을 구매하는 테스트의 경우 이전에 비해 373%의 성능 향상이 있었습니다.

Redis Operation 최적화

캐싱 시스템으로 사용하는 Redis의 경우 단일 스레드로 동작하기 때문에 하나의 연산은 모두 원자적입니다. 하지만 현재 구조에서 재고에 대한 검증과 재고 업데이트를 각각 Redis에서 수행할 시 하나의 연산이 아니기 때문에 원자적이지 않으므로 캐싱된 재고 데이터의 불일치가 발생합니다.

해당 연산을 원자적으로 만들기 위해 Redis에서 지원하는 트랜잭션을 적용하여 연산을 원자적으로 만들기 위해 시도했습니다. 하지만 Redis의 트랜잭션의 경우 Redis 클라이언트가 트랜잭션 내의 연산들을 모았다가 한 번에 보내는 방식으로 수행되기 때문에 트랜잭션 내의 읽기 연산을 지원하지 않았습니다. 읽기 연산을 지원하지 않는다면 재고 검증을 할 수 없으므로 주문이 접수되었지만 추후에 실패되는 안 좋은 사용자 경험을 줄 수 있기 때문에 재고 검증이 반드시 필요했습니다.

이런 부분을 해결하기 위하여 lua script를 도입하여 재고 검증과 수정을 하나의 연산으로 만드는 작업을 진행했습니다. lua script는 클라이언트가 아니라 서버에서 수행되고 읽기와 쓰기, 프로그램적인 문법까지 지원하기 때문에 구현하고자 하는 조건과 적합했습니다. 하나의 script는 하나의 연산임을 Redis가 보장하므로 구현한 기능의 원자적 특성까지 지원 가능했습니다.

주의했던 부분은 Redis의 특성입니다. Redis는 단일 스레드로 동작하므로 lua script로 작성된 연산이 길어지게 되면 전체적인 성능저하가 생길 가능성이 높다고 판단했습니다. 따라서 스크립트 내의 연산을 O(N)보다 작은 연산들로 구성해 작성하였습니다. lua script를 도입함으로써 단일 품목 구매 API 테스트에서 데이터의 정합성과 더불어 Redis 트랜잭션 대비 10%의 TPS 성능 향상까지 얻을 수 있었습니다.

이력서:

OOO

인적사항	성명	
	생년월일	
	휴대전화	
	E-mail	
	Blog	https://deveric.tistory.com
	Github	https://github.com/yyy9942
자격증	2019.04	정보처리기사
병역	구분 (필)	0000.00 - 0000.00 병장 만기전역
학력사항	2014.03 - 2020.02	
	한남대학교 공과대학	
	멀티미디어공학과 학사 졸업예정	
활동 내역	기간	2018.09-2018.12
	장소	한남대학교
	활동 내용	멀티미디어 학과 연구실 인턴
	-	NAS 서버 구축
	-	OS: ubuntu16.
	-	성과 세부내용: 클라우드 파일 서버 구축 및 운영을 진행 - 우분투 설치 및 환경 설정 - 사용자별 권한 관리 - 클라우드 파일 저장소 관리
	기간	2018.09-2018.12
	장소	한남대학교
	활동 내용	학과 코딩 튜터링
	-	C++, Java 기초 튜터링 성과 세부내용: 학과 내 후배들을 대상으로 C++, Java 기초부터 자료구조 구현까지 강의 진행. 언어의 특성 보다는 프로그래밍 기초 지식을 위주로 진행.

보유 기술 및 프로젝트	Java Spring	배달 매칭 API(제작 중), 커뮤니티 게시판
	Java FX	부동산 통합 중개 플랫폼
	Android	일정관리 앱, 메시징 앱
	DBMS	MariaDB, Oracle, Redis
	R	후쿠시마 원전 사고 이후 방사능 수치변화 분석
	Cloud	AWS, Cafe24, Naver Cloud Platform

포트폴리오

DelFood 음식 배달 O2O 서비스

사용 기술 : Spring-boot, MyBatis, MariaDB, Redis, Docker, Jenkins, Naver cloud platform

기간 : 2019.09 - 2020.02

Github : https://github.com/f-lab-edu/food-delivery

Wiki : https://github.com/f-lab-edu/food-delivery/wiki

Blog : https://deveric.tistory.com/category/Project/DelFood

Jenkins : http://106.10.51.119:8080/

프로젝트 개요

'배달의 민족'을 모티브로 만든 음식 배달 O2O Rest API입니다. 위치 기반 배달 서비스를 제공하며 실시간 라이더 매칭을 제공합니다. 대한민국 공공 도로명 주소 DB를 활용하여 라이더에게 배달하는 건물 좌표와 출입구 위치를 제공합니다.

고객, 사장님, 배달원 3가지의 타입으로 서비스를 이용할 수 있습니다. 고객은 자신의 위치를 기반으로 주위 배달 음식점들을 조회하여 주문할 수 있습니다. 사장님은 자신의 매장만 등록하면 고객, 배달원과 자동으로 매칭됩니다. 배달원은 자신의 실시간 위치를 등록하여 매장과 매칭되고, 고객에게 배달을 진행합니다.

Naver Cloud Platform을 사용하여 배포하고 있습니다. Delfood의 서버는 CI/CD를 담당하는 Jenkins 서버, RDBMS를 담당하는 MariaDB 서버, 캐시와 세션데이터를 저장하는 Redis 서버로 구성되어 있습니다. Public IP는 메인 서버에만 열어두어서 외부에서 직접 DB에 접근할 수 없도록 설계하였습니다.

어플리케이션 성능 테스트를 위해 실제 트래픽을 발생시켜 모니터링하고 있습니다. NGrinder를 클라우드 서버에 설치하여 모니터링하고 있으며 그 결과를 바탕으로 성능 튜닝을 준비하고 있습니다.

프로젝트 진행 과정

프론트 개발에 들어가는 시간을 아껴 서버 공부에 투자하기 위하여 kakao oven을 이용해 프로토타입을 제작하고 기능 요구 사항을 추출하여 설계를 진행하였습니다

프로젝트의 전반적인 내용은 Github의 Readme, WIKI에 문서화를 진행하였습니다. Readme에는 프로젝트 설계를 요약본을 넣었습니다. wiki에는 Business Rule과 발생한 문제를 기술적으로 풀어낸 글을 정리하고 있습니다.

Github을 통해 코드를 리뷰하며 상호간 코드의 허술함을 줄여 코드 품질을 보완하고 있습니다. 서로의 코드에 대해 이해하기 어려운 부분 있으면 코멘트하여 그 부분에 대한 상세한 설명 주석을 달고 있습니다. 이를 통해 서로의 개발 진행상황을 파악할 수 있었고 공부한 기능을 공유할 수 있었습니다.

서로가 작성한 코드를 이해하기 위해 다른 사람이 작업한 코드에 테스트코드를 작성하고 있습니다. 이를 통해 서로 다른 내용의 작업을 했더라도 어떤 작업을 했고 어떤 기능을 작성하였는지 알 수 있었습니다.

이슈단위로 브랜치를 관리하고 있습니다. 브랜치 관리 전략은 Git Flow를 적용하고 있습니다.
프로젝트에 대한 더 자세한 정보는 Github의 ReadMe를 참고해주시길 바랍니다.

로그인 시 Redis를 활용한 세션 데이터 관리

해당 프로젝트는 확장이 용이한 구조를 지향하고 있습니다. 그렇기 때문에 회원 로그인 아이디 정보를 세션에 저장하였을 때 Scale-out 시 세션이 분리되어 일관적인 세션 데이터관리가 어렵다는 문제를 어떻게 해결하는 것이 좋을까 고민하였습니다. 세션 정보를 데이터베이스에 연동하여 관리한다면 여러 서버에서 하나의 세션 저장소를 공유할 수 있을 것이고, 어떤 DBMS를 이용해야 세션 관리가 효율적일지 장단점을 비교해보았습니다.

속도 보다는 운영의 효율성을 고려하여 Memcached보다는 Redis를 사용하기로 결정하였습니다. Memcahaed는 안정적인 응답 속도와 빠른 저장속도를 가진다는 장점이 있지만 상대적으로 서버 운영에 필요한 기능이 부족했습니다. 그에 비해 Redis는 Spring에서 공식적으로 지원하고 있기 때문에 참고할 수 있는 자료가 풍부하고 다양한 Eviction을 지원하여 메모리 관리 및 선택에 정교한 접근법 사용이 가능하다는 장점이 있고, DB 이미지를 Disk에 저장하여 서버 손상시에도 데이터를 복구할 수 있다는 장점이 있었습니다.

이를 통해 Redis서버를 구축하여 클라이언트와의 세션을 관리하기로 결정하였습니다. 멀티 서버 환경에서도 모든 서버가 Redis를 통해 세션 데이터를 저장하고 참조할 수 있게 설정하였습니다. 이를 통해 서버간 세션의 공유를 가능하게 만들었고 사용자의 세션에 저장한 데이터를 관리할 수 있었습니다.

검색 시 DB 서버 부하를 고려한 캐시 전략

회원 가입을 진행하거나 회원이 주소를 변경하고자 할 때 주소를 검색하는 로직이 DB에 큰 부하를 주었습니다. 1000만건 이상의 공공 주소 데이터를 DB에서 검색하기 위하여 인덱스도 설정해 보았지만 인덱스 만으로는 주소 검색 시 마다 DB에 가해지는 부하를 감당하기 어려웠습니다.

주소 DB의 사용을 추적해본 결과 동일 주소의 검색이 빈번하다는 것을 확인하였습니다. 회원이 가지고 있는 주소 코드를 기반으로 사장님에게 상세 주소를 전달해야 하고 회원에게도 배달 받을 주소를 전달해야 하기 때문입니다.

그렇기 때문에 WAS와 DB서버의 부하를 줄이기 위하여 캐싱을 하기로 하였습니다. 주소 검색은 동일 검색조건 하에 대부분 동일한 결과값의 반환이 이루어 지고 DB서버에 부하가 커서 캐싱을 적용하였을 때 성능과 속도 개선의 여지가 크기 때문입니다. 또한 주소데이터는 데이터의 변경이 거의 발생하지 않기 때문에 캐시 데이터의 만료시간을 길게 주어서 캐시의

재사용성을 높였습니다.

한번 캐싱된 데이터를 다른 서버에서도 조회할 수 있도록 Redis를 사용해 글로벌 캐싱을 적용하였습니다. 또 주소 데이터는 변경이 거의 일어나지 않기 때문에 캐시 데이터의 expire time을 길게 주었습니다. 이를 통해 평균 응답시간이 300ms에서 10ms미만으로 감소하였습니다.

Redis의 동시성 이슈를 고려한 로직 처리

고객은 매장에서 검색한 메뉴를 장바구니에 저장할 수 있습니다. 장바구니 기능은 expire 설정이 필요하고, 상대적으로 데이터의 중요도가 낮기 때문에 DB 서버의 부하를 줄이기 위하여 Redis를 사용하였습니다
장바구니에 메뉴를 저장하는 로직은 멀티스레드 환경에서 동시성 문제가 생길 가능성이 있었습니다. 장바구니는 최대 10개의 메뉴가 저장될 수 있고, 비어 있는 상태에서 장바구니를 생성하기 위해서는 expire time을 설정해 줘야 하는 로직이 있었습니다.

예를 들어 Redis에 저장하는 장바구니 기능에서 물품을 추가할 때 다음과 같은 로직이 있습니다.

```
해당 회원 장바구니에 몇 개의 물품이 있는지 조회한다
-> 비어 있거나 최대 수량을 초과할 경우 적절한 로직을 호출한다
-> 장바구니에 물품을 추가한다.
```

이 때 장바구니를 조회한 후 유효성 검사를 하는 동안 다른 스레드가 장바구니에 접근하여 상태를 바꾸어 버린다면 동시성 이슈가 발생할 수 있을 것입니다. 그렇기 때문에 Redis를 호출하여 작업 시 해당 키를 watch하여 변경되는지 모니터링하여 로직을 진행하는 Redis Transaction을 적용하였습니다.

라이더의 매칭 도중 생길 수 있는 동시성 이슈 해결

매장에서 메뉴를 준비하여 배달원 매칭을 서버에 요청할 경우 매장 주위 일정 반경 이내에 있는 배달원들에게 푸시 알림 메시지를 전송합니다. 이 때 가장 먼저 매칭을 수락한 배달원에게 매장 배달 권한이 주어지는데 동시에 여러 배달원에게 요청이 올 경우에 동시성 이슈가 생길 수 있었습니다. 그 이유는 배달원의 현재 위치와 상태를 서버 로컬 메모리에 Map 형태로 관리하고 있었고, 라이더의 매칭 작업을 하는 도중 해당 정보가 변경된다면 여러 라이더가 동시에 매칭될 수 있었기 때문입니다.

이를 해결하기 위해 Map의 구현체로 동시성 이슈를 해결할 수 있는 SynchronizedMap과 ConcurrentHashMap을 적용하기로 하였습니다. 두 구현체를 비교하여 보니 SynchronizedMap은 전체적으로 lock을 거는 것에 비해 ConcurrentHashMap은 lock striping 기법을 적용하여 적용되는 lock을 분할하여 사용한다는 것을 알게 되었습니다. 성능적으로 ConcurrentHashMap이 월등하게 빨랐고 이를 적용하여 로직의 원자성을 지켜 동시성 이슈를 해결할 수 있었습니다.

로컬 Map에는 트랜잭션을 적용하기 어렵기 때문에 Redis를 사용하는 것도 좋은 방법이라고 생각하였습니다. 라이더 매칭과 관련된 기능을 추상화하여 구현체를 어떤 것을 사용하는지에 따라 로컬 메모리(Map), Redis중 하나를 선택하여 사용할 수 있도록 구현하였습니다. Redis를 사용할 경우 Redis 트랜잭션과 Key를 watch하는 방법을 통해 연산을 단일 연산으로 만들어 동시성 이슈를 해결하였습니다.

메뉴 주문 시 데이터 무결성을 위한 트랜잭션 적용

주문 로직은 여러 테이블에 insert와 update를 진행합니다. 저장과 수정을 진행하는 도중 오류 또는 예외가 발생한다면 데이터의 무결성이 깨질 것입니다. 그렇기 때문에 예외가 발생한다면 적용된 사항들을 저장하지 않고 롤백할 수 있도록 트랜잭션을 적용하였습니다.

주문이 실패하였다면 왜 실패했는지 기록하는 데이터가 필요했습니다. 해당 유저가 주문을 시도하였다는 정보를 따로 저장하기 위해 해당 로직을 별도의 트랜잭션으로 분리하였습니다. @Transactional의 propagation 속성을 NESTED로 설정함으로써 해당 로직은 상위 로직에 종속적이지 않는 별도의 트랜잭션으로 작동할 수 있도록 구현하였습니다.

AOP를 적용한 공통 관심사 분리

DelFood의 대부분의 서비스는 로그인을 해야 이용할 수 있습니다. 처음 프로젝트를 제작할 때 모든 서비스 마다 세션에서 로그인 정보가 있는지 확인하는 코드를 작성하였습니다. 하지만 이를 AOP를 적용하여 횡단관심사를 분리할 수 있었습니다. 이를 통해 로그인이 되었는지 확인하는 중복 코드들을 제거할 수 있었습니다.

공통 로직의 적용을 위해 어노테이션을 기반으로 AOP를 적용하였습니다. 이를 통해 하나의 클래스 안에서도 권한별로 호출 가능한 메소드를 더 세밀하게 조정할 수 있었습니다. 주소 검색시에도 비회원도 가능한 일반 검색, 회원이 가능한 내 주변 매장 주소 검색 등을 권한별로 나눌 수 있었습니다.

Reflection 적용한 사장님의 매장 권한 체크

사장님이 로그인하여 매장에 대한 조작을 진행할 때 해당 매장이 사장님의 소유인지 확인하는 로직을 먼저 작성해야 했습니다. Controller에서 매장에 대한 조작을 진행할 때 항상 매장 ID를 파라미터로 받아오는 것에서 아이디어를 얻어서 어노테이션을 적용한 후, 내부에 있는 파라미터에 매장 아이디를 저장하는 변수 명을 전달하면 자동으로 권한 체크를 진행하도록 하였습니다.

MethodSignature를 사용하여 Method와 Parameter를 추출하였고, 해당 파라미터의 이름을 조회하여 입력 받은 변수 명과 일치하는지 검사하였습니다. 일치할 경우 적절한 서비스를 호출하여 권한의 검사를 진행합니다. 일치하는 변수명이 없을 경우 예외처리를 진행하도록 구현하였습니다.

Mockito와 Junit으로 고립된 단위테스트 작성

소스코드를 작성하거나 변경 시 오류가 나지 않는지 기능을 하나하나 URL에 데이터를 보내 직접 테스트를 했어야 했습니다. 이런 불편을 개선하기 위해 서비스 로직을 대상으로 Junit과 Mockito를 활용한 단위 테스트를 작성하였습니다.

의존성이 있는 객체를 대상으로 Mock DAO 객체를 만들어 주입하였고, DB에서 예상되는 결과값들을 반환하도록 설정하였습니다. 그 결과 Service 레이어에 있는 로직을 중심으로 고립된 단위 테스트를 제작할 수 있었습니다. 또한 Mockito Framework를 사용하여 Spring container가 올라가지 않고, DB와의 커넥션을 하지 않기 때문에 빠른 테스트가 가능하도록 하였습니다.

Jenkins를 활용한 테스트 자동화와 CI/CD 적용

소스코드를 작성하거나 수정할 때 직접 테스트를 진행하고 서로 그 결과를 공유해야 하는 번거로움이 있었습니다. 이러한 번거로움을 해소하기 위하여 Naver Cloud Platform 서비스를 이용하여 CI 서버를 구축하였습니다. 이 서버에서 자동으로 테스트를 진행할 수 있도록 Github에서 PR이 발생할 때 Jenkins 서버로 hook을 날려 빌드와 테스트를 진행하도록 적용하였습니다. 그 결과 협업할 때 서로의 소스코드 신뢰성을 보장받을 수 있었습니다.

배포 전 어플리케이션 기능의 무결성을 검사하기 위해 꼼꼼한 단위테스트를 작성했습니다. CI서버에서 배포 전 해당 단위테스트를 실행하고, 이상이 발생할 때 개발자에게 메일로 알림을 주도록 설정했습니다.

서버 배포의 편의성을 위해 빌드, 테스트 완료 시 도커 이미지를 제작하여 메인 서버에 배포하도록 설정하였습니다. CI 서버에서 이미지를 구축하면 원격 스크립트를 통해 메인 어플리케이션 서버에서 이미지를 받아 실행할 수 있도록 스크립트를 작성했습니다. 그 결과 소스코드 변경이 발생하면 자동으로 서버에서 빌드, 테스트, 배포까지 진행하도록 만들 수 있었습니다.

Java 스펙을 활용한 코드 품질 관리

Java에 어울리는 코드를 만들기 위하여 꾸준히 코드를 검수하고 있습니다. '이펙티브 자바', '자바 성능 튜닝 이야기' 등의 책을 참고하여 코드 품질 향상에 목표를 설정하였습니다. 이를 통해 프로젝트 코드를 아래와 같이 개선하였습니다.

프로젝트 초기에 null값 체크, 유효성 검사 등의 처리를 하기 위해 if-else문을 많이 만든 적이 있습니다. Controller에서 로직을 호출한 후 결과값을 받아 정상적인 결과값이 아니라면 직접 예외처리를 진행해야 했습니다. 이를 각 레이어의 책임을 분리하고, 이상이 생길 때 하위 로직으로 결과값을 반환하지 않고 예외를 던지도록 개선하였습니다. 이를 통해 무분별한 if-else문과 유효성 검사 코드를 제거할 수 있었습니다.

요청에 대한 반환 값을 Map을 통해 전달하는 것에서 명확한 객체를 통해 전달하는 것으로 변경하였습니다. Map은 어떤 데이터가 들어있는지, 데이터 타입이 무엇인지 명확하게 알 수 없다는 단점이 있기 때문입니다. 실제로 명확한 DTO, Response 객체를 사용하도록 변경하였을 때 개발의 편의성과 유지보수의 편리함이 있다는 것을 알 수 있었습니다.

스레드 안정성을 위해 필요한 경우가 아니라면 setter를 설정하지 않고 값을 복사하여 넣어주어 불변 객체를 만들어 전달하였습니다. 이를 통해 의도치 않은 값의 변화를 억제하였습니다.

static final 상수를 사용하여 불필요한 객체 생성을 지양하였습니다. 정적인 반환 값이 있다면 이를 상수로 지정해 준 뒤 반환하도록 설정하였습니다. 이를 통해 객체 생성에 필요한 리소스를 줄일 수 있었습니다.

Java 스펙에 맞는 코드를 제작하기 위해 계속해서 코드를 리팩토링하고 있습니다. 작동하는 코드에 만족하지 않고 더 안정적이고 좋은 코드를 만들기 위해 노력하고 있습니다. Github에 commit하는 코드의 약 절반정도는 기존 코드를 개선하는 코드입니다.

모두를

포트

0 0 0

(직무 : 데이터 엔지니어(AI))

생년월일	1995.00.00
이메일	Test00@test.com
전화번호	010-0000-0000
주소	서울시 00 구
연봉	협의가능

소개 / About Me

- 메인 언어 Python 로 데이터 엔지니어링, AI 모델 개발에 관심이 많은 개발자
- 활용할 수 있는 도구 소개
- 개발자로서 비전 / 닮고 싶은 개발자(롤모델) 소개(태도/가치관)

학력

- 2000.00 ~ 2000.00 서울 00 고등학교 졸업 / 최종학력

자격증

- 컴퓨터활용능력 1 급 / 대한상공회의소 / (2000. 00 취득)

교육 내용

- 2000.00 ~ 2000.00(00 개월) AI 과정 / 메타버스 아카데미 1 기 / 과학기술정보통신부
- 2000.00 ~ 2000.00(00 개월) 게임프로그래밍 / 서울 IT 아카데미 / 우수 수료생
- 2000.00 ~ 2000.00(00 개월) 웹프로그래밍 / 비트교육센터

수상 내역

- 메타버스 아카데미 (주관 : RAPA) 8 월말 프로젝트 1 등
- 미드나잇 캠프 (주관 : RAPA) 참가팀 25 팀 중 2 등
- 서울 IT 아카데미 우수 수료 수상

경력 사항 / Work Experience 총 0 년

재직 기간	회사명	부서 및 R&R	비고
2000.00~2000.00	00	정보연계부 – Linux 서버로 여러 공기관과 데이터 파일 송수신 관리 및 Trouble shooting. DB 적재시 이상 데이터 원인파악 및 처리. 내부 홈페이지 개선사항 수정.	사원

기술 스택 / Skill Set

기능 구현 등의 사용 경험이 있는 Skill Set

구분	Skill
개발환경	Linux, Visual Studio, Eclipse, Jupyter, Pycharm, Rider, Unity, D3D, MFC
언어	Java, Javascript, Python, C, C++, C#, SQL
프레임워크	Spring, Egovframework, Flask
웹 표준기술	HTML5, CSS, JQuery, Ajax, bootstrap
RDBMS	MySQL, MariaDB, Oracle
NoSQL	MongoDB
클라우드	AWS EC2, Heroku, GroomIDE
형상관리	Github

➔ Skill Set 작성 기준은 Skill Set 활용하여 프로젝트 경험이 있으면 왜 사용 했는 지 이유를 면접에서 설명 가능할 수준일 때 작성함

인테리어 랑랑 타운('인테리어 ai 추천 서비스, 미니 융합 프로젝트')

작업 기간	2000. 00. 00~2000. 00. 00 (0 주)
인력 구성	PM 0 명, Unity 0 명, AI 0 명, NET 0 명 (총 0 명)
프로젝트 목적	AI 와 Unity, NET 기술 간의 미니 융합 프로젝트로 통신 기술 습득과 융합 경험
역할/업무	<데이터 수집> 네이버 api, requests 으로 네이버 쇼핑 내의 가구 데이터를 수집하고 DB 에 저장함. 매일 새로운 상품들을 저장하기 위해 BackgroundScheduler 로 3 시간마다 데이터 크롤링 프로세스를 설정하였음. <AI 모델 서빙과 서버 배포> AI 추천 시스템과 가구 이미지 카테고리 분류 모델을 서빙하기 위해 Flask, gunicorn 사용. Postman 으로 Rest api 테스트 및 문서 작성. mongoDB 를 통해 Unity(client)의 사용자와 맵 데이터 저장 및 로드. Heroku 클라우드를 통한 배포를 하였고 git hub 연동으로 ci/cd 구축. Procfile 과 requirements 파일을 통한 패키지관리와 gunicorn 의 환경 설정 관리.
사용언어 및 개발환경	수집데이터 : 네이버 쇼핑 가구 데이터 개발환경 : Flask 언어 : Python 기타 기술 : gunicorn, Heroku, mongoDB, beautifulsoup
프로젝트 중 고민/문제해결	이미지 분류 모델을 서빙하자 메모리 초과 문제가 발생해 서버가 종료되는 문제가 발생함. Gunicorn 설정으로 인한 메모리 누수 현상임을 파악하고 Procfile 파일로 worker 프로세스 관리로 메모리 누수를 해결.
참고자료	포트폴리오 URL : 영상 URL : Github URL : 이력서의 목적은 위 포트폴리오 URL 주소를 클릭하게 만드는 것

모두를 위한 **커리어** 준비

현업 개발자의 목소리

프로그래밍 언어 이해 / 논리적, 절차적 사고를 할 수 있는 역량이 있음
→ 추가로 회사에서 사용하는 언어 경험이 있으면 우대

직무와 연관된 프로젝트 경험이 있음
→ 프로젝트 진행별 고민-문제-시도-해결을 확인할 수 있으면 우대

도메인 지식 및 개발자로 성장하기 위한 노력이 있음
→ 특히 비전공자일수록 검증절차가 까다로움
(깃허브, 블로그, 포트폴리오)

현업 개발자의 목소리

6개월 기간 동안 많은 것을 했다고 생각하지 않는다.
기본이라도 제대로 할 수 있었으면 좋겠다.
작은 부분이라도 주도적으로 직무을 수행할 수 있었으면 좋겠다.

Best of Your Career

May.19.

Pray2U