박재성 (Backend Developer)

개발을 통해 **긍정적인 가치를 만들어낼 수 있는 백엔드 개발자**가 되고 싶습니다.
빠르게 학습하고 자주 도전함으로써 다양한 가치들을 만들어내고 싶습니다.

- Email : qkrwotjd1445@naver.com
- Phone : 010-5198-4332
- Github : https://github.com/jaeseongDev
- Education : 인하대 통계학과 졸업 예정

🤾 Project

댓글 모듈 서비스 (Darass)

팀 프로젝트
우아한 테크코스 3기
(21.06.22. ~ 21.10.29.)

✓ 서비스

- 어디든지 쉽고 간편하게 다는 댓글 모듈 서비스
- 메인 페이지 : https://darass.co.kr/

✓ 주 기술 스택

- Java, Spring Boot, JPA, MariaDB

✓ 기여한 부분

- DB Connection Leak에 대한 장애 해결
- 서버 장애가 났을 때, 10분 안에 장애를 진단할 수 있는 매뉴얼 제작
- Git Branch 전략 개선을 통해 main으로의 평균 배포 주기를 1일 이하로 단축
- SQL 튜닝을 통한 Throughput 4배 향상(110rps -> 410rps),
  Latency 4배 단축(860ms -> 240ms)
- 약 200개의 테스트 코드 작성 (단위 테스트, 통합 테스트, 인수 테스트)
- Mariabackup을 활용한 DB 백업 자동화 구현
- 부하 분산을 위한 DB Replication(Master-Slave) 구현
- 전체 인프라 아키텍처 구축 관여
- 유저 정보 수정, 프로젝트 생성 API 구현

대학가 상권 할인 앱 (DALY)

창업
DAU 약 100명
누적 다운로드 수 약 4K
(19.07.26. ~ 20.11.25.)

✓ **서비스**

- 대학가에 있는 상권의 할인 쿠폰을 판매하는 서비스
- 랜딩 페이지 : https://daly.kr/
- Android : https://play.google.com/store/apps/details?id=com.DALY
- iOS : https://apps.apple.com/kr/app/달리/id1487398668

✓ **주 기술 스택**

- Node.js, Express,js, Sequelize, MariaDB
- Vue.js, Vuex

✓ **기여한 부분**

- Node.js, Express.js를 활용한 약 40개의 REST API 설계 및 개발
  (상품 관련 기능, 결제 기능, 로그인 및 회원가입 기능)
- Row 수준의 Lock을 활용해 쿠폰이 2개가 들어가는 장애 해결
- 전체 인프라 아키텍처 구축

👩‍🏫 Education

우아한 테크코스 3기

(주)우아한형제들 주관 교육 코스
(https://woowacourse.github.io/)
(21.02.02. ~ 21.11.26.)

✓ **배운 점**

- 스프링 프레임워크 기반으로 웹 애플리케이션 개발
- Clean Code (읽기 좋은 코드)
- TDD, ATDD 기반 개발
- 테스트 코드 작성 방법 (단위 테스트, 통합 테스트, 인수 테스트)
- 객체 지향 프로그래밍(OOP) / 객체 지향 설계
- 레거시 코드 리팩토링 경험
- 대용량 서비스를 위한 시스템 아키텍처 설계
- 부하 및 성능 테스트

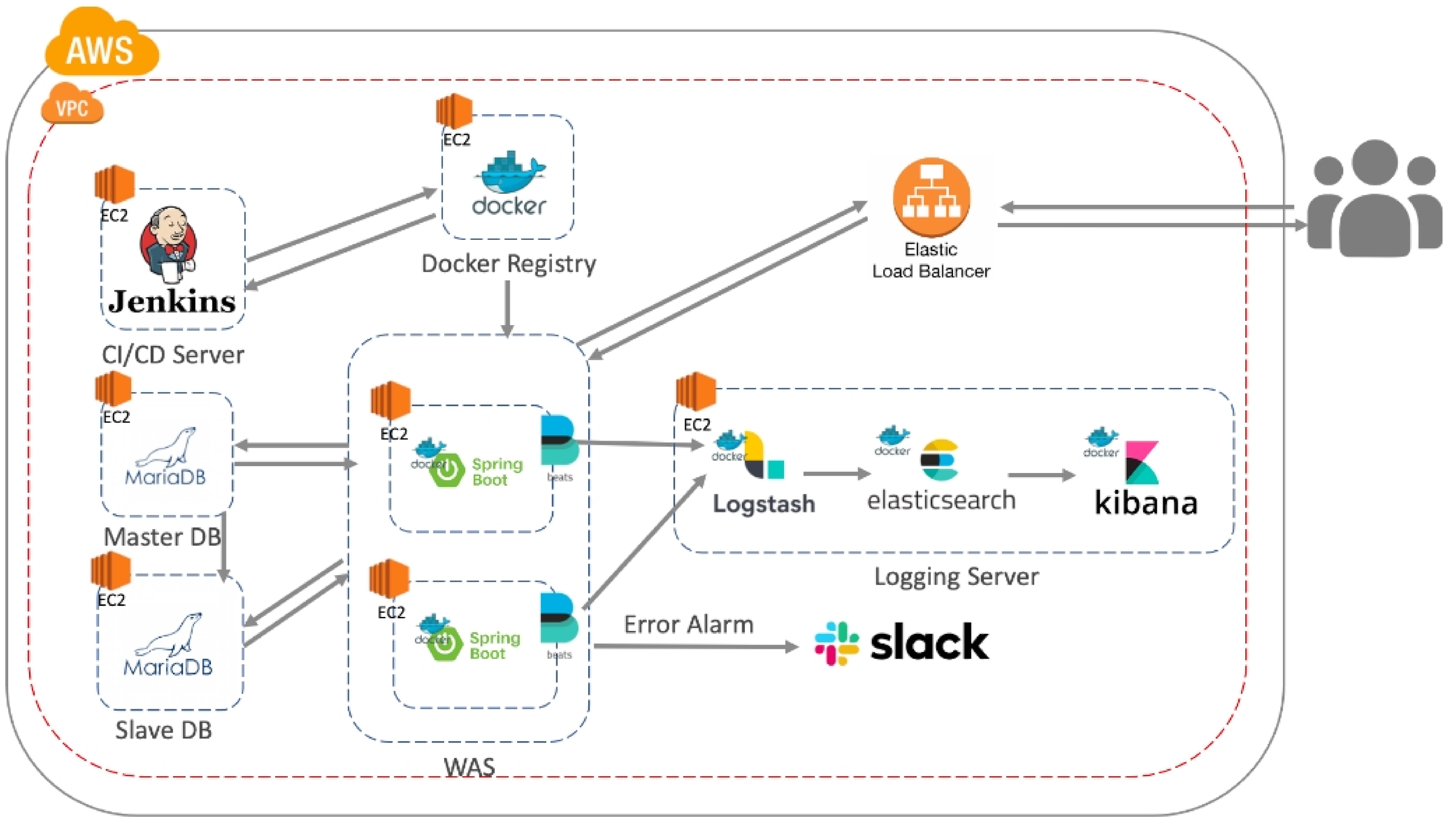
[Darass / 우아한 테크 코스 3기] 댓글 모듈 서비스

🐤 Introduce

스크립트 코드를 웹 페이지에 붙여넣는 것만으로 간편하게 댓글 기능을 추가할 수 있는 댓글 모듈 서비스입니다. 사용자에게 필요한 댓글 기능과 함께 운영에 필요한 댓글 통계 및 관리 기능도 제공합니다.

- 메인 페이지 : https://darass.co.kr/

🔨 Architecture



✓ Load Balancing을 위해 ALB 사용

✓ 부하 분산 및 SPOF를 피하기 위해 WAS 서버를 2개 배치

✓ 부하 분산을 위해 DB Replication(Master-Slave) 구성

✓ Private으로 Docker Image 저장소를 사용하기 위해 Docker Registry 사용

✓ Jenkins를 통한 CI/CD 구축

✓ ELKF(Elasticsearch, Logstash, Kabina, Filebeat)를 통해
여러 WAS 서버에 대해 로깅을 통합 저장 및 시각화

✓ 배포 및 개발 환경에서 Server Error(500)가 발생할 경우 Slack으로 알림이 오도록 설정

🛠️ Trouble Shooting

1. DB Connection Leak에 대한 장애 해결

✓ 문제

- 서비스를 운영환경에 배포를 하고 일정 시간이 지난 이후에 Connection Timeout 에러가 발생했다.

✓ 원인 파악 과정

1. Connection을 사용하고 반납하지 않는 로직이 존재하는 경우라고 추측
2. Connection의 반납 여부를 확인하기 위해 Connection 상태를 Logging하는 코드 추가
3. Close를 해주지 않아서 Connection을 반납하지 않을만한 로직이 있는 지 체크
4. Connection 사용 및 반납 시기에 대한 여러 가지 실험 및 학습
5. 각각의 API에 대해 Connection 반납 여부 확인

-> **while문의 무한 루프로 인해 Connection이 반납되지 않고 해당 요청을 계속 처리하고 있었음**

✓ 해결 과정

- 무한 루프를 돌지 않게 while문에 대한 로직을 개선

✓ 평가 및 회고

- Connection Timeout의 원인이 무한 루프라는 것을 밝혀내기까지 오랜 시간이 걸렸다.

다음 번에 이러한 문제가 발생했을 때에는 어떤 순서로 장애를 진단하면 좋을 지 고민해봤다.

1. Connection에 대한 close()를 처리해주지 않을만한 코드가 있는 지 확인
2. 무한 루프에 빠질만한 로직이 있는 지 체크
    2.1. 무한 루프가 의심될 경우, 가장 먼저 CPU Utilization과 Saturation을 파악
    2.2. CPU 과부하가 걸려있다면 Thread Dump를 통해 무한 루프 지점 파악

- CPU 과부하의 상태를 조기 진단하기 위해 **CloudWatch를 통해 경보 설정**을 해주었다.

✓ 해당 내용 링크

- https://url.kr/az4j3m

2. 서버 장애 시 10분 안에 장애를 진단할 수 있는 매뉴얼 제작

✓ **문제**

- 서버 장애가 났을 때 빠르게 장애에 대처할 수 없다면 오랜 시간 사용자에게 불편함을 줄 수 있다.
- 서버 장애를 진단할 수 있는 팀원이 연락이 안 될 경우, 훨씬 더 오랜 시간동안 장애를 방치할 수 밖에 없다.

✓ **해결 과정**

서버 장애시 장애를 진단할 수 있는 매뉴얼이 있다면, 보다 빠르게 대응할 수 있다고 생각했다.
또한 팀원 전체가 서버 장애를 진단할 수 있는 능력을 갖출 수 있다고 생각했다.
리눅스 유틸리티와 Thread Dump를 활용해서 대체로 자주 일어나는 장애를 진단하도록 구성했다.
간단하게 요약하자면 장애 진단의 순서는 다음과 같다.

1. Load Average 체크
2. 시스템 메시지 확인
3. CPU, Memory, Disk I/O에 대한 Utilization, Saturation 확인
4. 필요에 따라 Thread Dump를 활용해 정확한 원인 진단

✓ **평가 및 회고**

- CPU에 부하가 가해지는 상황, 무한루프 상황, 디스크 I/O가 걸리는 상황, DeadLock 상황 등을 가정해서, 매뉴얼을 바탕으로 서버 장애를 진단하는 시뮬레이션을 돌려보았다. 매뉴얼이 없을 때보다 매뉴얼이 존재하다보니 **훨씬 빠르게 장애를 찾아낼 수 있었다.**
- 추후에 매뉴얼을 바탕으로 진단이 안 되는 부분에 대한 장애를 만나게 되면, 다음 번에 해당 장애도 잡아낼 수 있게 **점진적으로 매뉴얼 내용을 보완**해가면 훨씬 좋겠다는 생각이 들었다.

✓ **해당 내용 링크**

- https://url.kr/el6nt5

3. SQL 튜닝을 통한 Throughput, Latency 개선

✓ 문제

- 서비스 운영 시 사용자들로부터 가장 많이 사용되는 API에 대해 요청 수가 늘어날 것을 대비하기 위해, 부하 테스트를 통해 비효율적인 SQL에 대해 튜닝을 하려고 한다.

✓ 해결 과정

1. SQL 튜닝 전후의 성능을 비교하기 위해, 부하테스트(K6)를 통해 Throughput, Latency를 측정
2. SQL 쿼리 분석 -> **N+1 문제 발견**
3. **Fetch Join**을 사용해서 해결하려 했으나, 해당 엔티티에 1:N 관계가 2개가 포함되어 있어서 MultiBagFetchException이 발생
4. **BatchSize**를 수정해서 N+1 문제를 해결
5. SQL 튜닝 후 Throughput, Latency를 측정해보니 4배 정도 성능이 향상되었음을 확인

(Throughput : 110rps -> 410rps / Latency : 860ms -> 240ms)

✓ 평가 및 회고

- SQL 튜닝만으로도 확연하게 성능을 개선할 수 있음을 느꼈다. Throughput, Latency를 개선하기 위해 단순히 WAS 서버의 수를 늘리기 전에 SQL 튜닝을 먼저 진행해보는 것이 좋겠다는 생각이 들었다.

✓ 해당 내용 링크

- https://url.kr/votgar

4. Git Branch 전략 변경 (Git-flow → Github-flow)

✓ 문제

Git-flow Branch 전략을 선택해서 개발을 진행하다보니, 다음과 같은 문제점이 발견됐다.

- Branch가 많아서 복잡하다.
- main에 배포할 때마다 PR 및 충돌 해결 과정이 번거롭고 복잡하다.
- 복잡한 배포 과정으로 인해 팀원들이 main에 병합해서 배포하는 과정을 꺼려한다.
- main으로 배포하는 주기가 길어지다보니 main에 배포하는 코드양이 많아지고, 이로 인해 발생하는 버그를 디버깅하기가 어려워진다.

✓ 해결 과정

1. Git-flow Branch 전략에서 **Github-flow Branch 전략으로 수정하자고 의견 제기**
2. Github-flow는 개발 환경에서 테스트를 해보지 못해서 버그가 발생할 가능성이 높다는 반대 의견과 충돌
3. **Github-flow Branch 전략으로 변경하기 위해 다음과 같은 근거로 설득**
   - 꼼꼼한 테스트 코드 작성과 CI를 통해 버그를 조기에 발견할 수 있도록 하기
   - PR을 통한 꼼꼼한 코드 리뷰
   - Github의 PR event가 발생할 경우, develop 환경에 배포되도록 해서 테스트할 수 있도록 환경 구성
   - Github-flow로 먼저 도전해보고, 문제 발생할 경우 다시 Branch 전략 논의

✓ 평가 및 회고

- Git Branch 전략 개선을 통해 main으로의 **평균 배포 주기를 1일 이하로 단축**시킬 수 있었다.
- 사용자에게 업데이트 된 기능을 훨씬 빠르게 제공할 수 있게 되었다.
- Git-flow Branch 전략을 사용했을 때에는 배포 환경에서 버그가 발생했을 때, Hotfix Branch를 따서 버그 수정하는 작업을 거치는 것도 번거로웠다. 하지만 Github-flow Branch 전략을 사용하니, 버그가 발생하더라도 재빠르게 대응할 수 있게 되었고, 이 덕분에 팀원들도 버그 수정 참여율이 올라가는 것을 확인할 수 있었다.

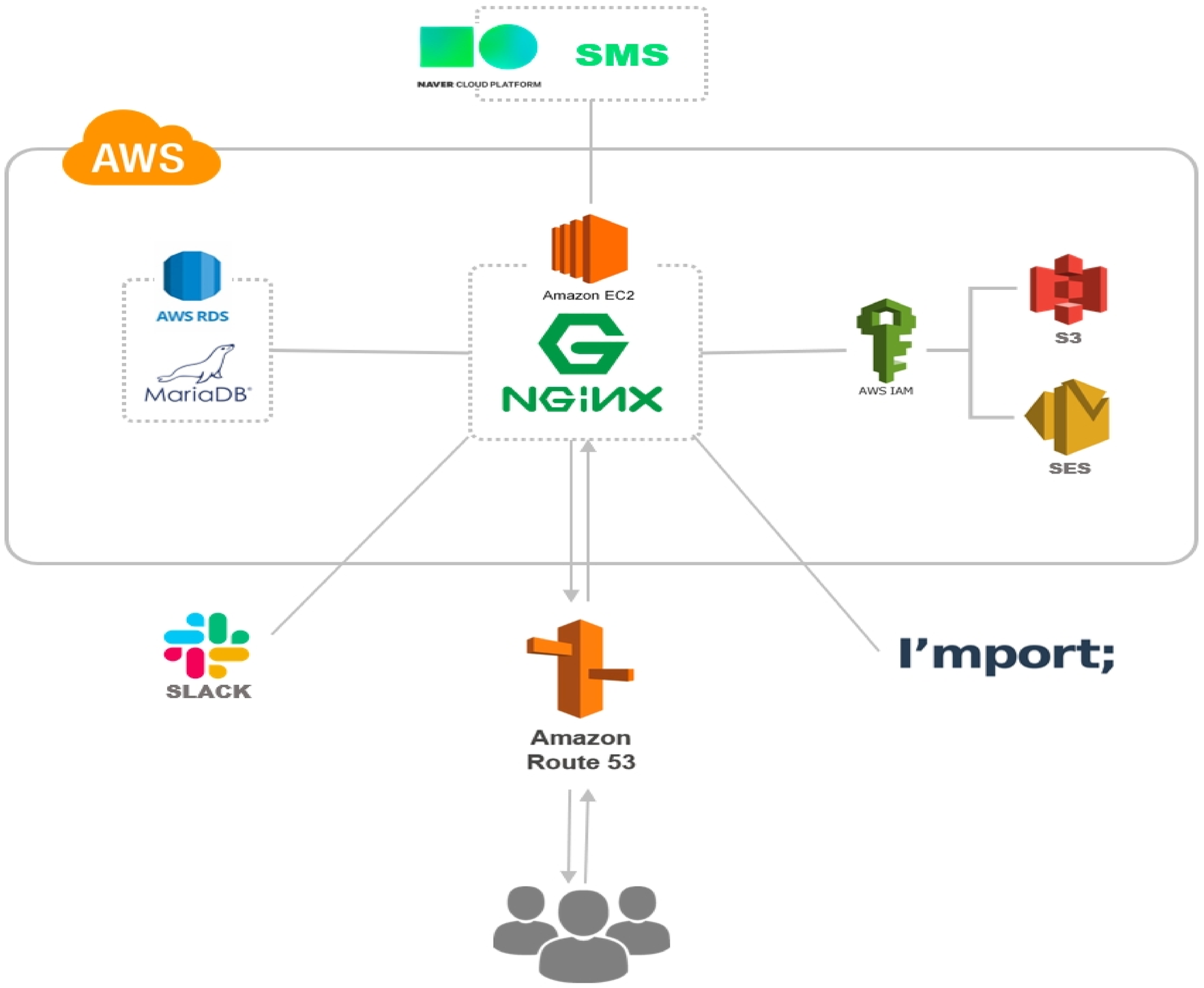
[DALY / 창업] - 대학가 상권 할인 앱

🐤 Introduce

이 서비스는 식당, 술집, 카페 등과 같이 **모든 일상 생활에서 할인을 제공**해주는 모바일 어플리케이션 서비스입니다. **할인된 쿠폰을 판매**함으로써 해당 가게에 가서 사용할 수 있는 구조로 구성되어 있습니다.

- 랜딩 페이지 : https://daly.kr/
- Android : https://play.google.com/store/apps/details?id=com.DALY
- iOS : https://apps.apple.com/kr/app/달리/id1487398668

🔨 Architecture



✓ Route53을 DNS로 활용

✓ NCP(SMS)를 문자 발송 기능 구현

✓ I'mport(외부 결제 모듈)를 통한 결제 기능 구현

✓ AWS(SES)를 활용해 이메일 발송 기능 구현

✓ Nginx를 Reverese Proxy로 활용

✓ 배포 및 개발 환경에서 Server Error(500)가 발생할 경우 Slack으로 알림이 오도록 설정

🛠️ Trouble Shooting

Row 수준의 Lock을 활용해 쿠폰이 2개가 들어가는 장애 해결

✓문제

- 결제를 한 번 했는데, 쿠폰 2개가 들어가는 현상이 발생.

✓원인

네트워크의 오류로 인해 결제 데이터가 제대로 DB에 저장이 안 되는 것을 방지하기 위해,
외부 결제 모듈 서버에서 Webhook을 통해 결제 완료에 대한 데이터를 한 번 더 발송한다.
특정 사용자가 결제를 할 때 기존에 결제 완료 데이터가 존재하는 지를 확인한 후에 쿠폰을 지급하는 형태이다.
이 때, **Repeatable Read 수준의 트랜잭션 격리 수준으로 인해 결제 완료 데이터가 Commit 되기 전에 데이터 존재 여부를 확인했을 때 결제가 완료되지 않았다고 판단해서 쿠폰이 2개가 들어가는 장애가 발생했다.**

✓해결 과정

1. DB Transaction으로 인해 문제가 발생했다고 판단되어, DB Transaction의 개념에 대해 학습함.
2. 결제 데이터에 대해 쓰기 작업이 진행되는 동안, **SELECT ~ FOR UPDATE를 활용해 해당 행(Row) 단위의 Lock을 걸어서 해결**. 이로 인해 트랜잭션에 대한 동시성 문제가 발생하지 않고 정상적으로 작동함.

✓평가 및 회고

- 서비스에 도입하는 기술의 작동 원리를 이해해야만, 치명적인 에러를 사전에 예방할 수 있고 트러블 슈팅도 훨씬 빠르게 할 수 있다는 것을 느꼈다.

⚙️ Skills

✓ Backend

- **Java, Spring Boot, JPA(Hibernate)**
- Node.js, Express.js, Sequelize
- **MariaDB**, MySQL

✓ Devops

- AWS (EC2, RDS, S3, CloudFront, Route53, ELB, SES), NCP(SMS)
- Docker, Docker Compose, Docker Registry, Nginx, Github Actions, Jenkins
- Git
- K6

✓ Frontend

- Vue.js, Vuex
- HTML, CSS, JavaScript

🏆 Experience & Awards

✓ 우아한 테크코스(https://woowacourse.github.io/) 3기 백엔드 과정 (2021.02. ~ 2021.11.)

✓ NEXTERS(http://teamnexters.com/) 17기 수료 (2020.07. ~ 2020.08.)

- 온라인 롤링페이퍼 웹 서비스 최우수상(1위)

✓ DALY(https://daly.kr/) 창업 (2019.07. ~ 2020.10.)

- 2019 IeTC 경진대회 장려상
- 인천 I-STARTUP LAB 우수기업 선정
- 누적 다운로드 횟수 4,000K (20년 11월 기준)

✓ SOPT(http://sopt.org/wp/) 24기 수료 (2019.03. ~ 2019.07.)

✓ 멋쟁이 사자처럼(https://www.likelion.net/) 6기 수료 (2018.03. ~ 2018.12.)

- 아이디어톤 대회 우승(1위)
- 신촌톤 대회 아이디어상 수상
- 수료 이후에 해당 아이디어를 창업으로 전환

👨‍💻 About Me

✓ 도전을 중요시합니다.

- 대학생 기간 동안 **3번의 창업을 시도**하였고, **세미나 및 멘토링을 자체적으로 기획**해서 도전해보았습니다.
- '초우량 기업의 조건(톰 피터스)'이라는 책에서 '해봐라! 안 되면 고쳐라! 다시 시도해봐라!'라는 말을 실천으로 옮기려고 노력합니다.

✓ 학습하거나 경험한 것을 다른 사람들에게 공유하는 것을 좋아합니다.

- 초창기 스타트업 대상 개발 컨설팅 진행 (2021.05. ~ 2021.08.)
- 개발 멘토링 진행 (2020.07. ~ 2020.11.)
- 공부법 및 진로 정기 멘토링 (2018.10. ~ 2018.12.)
- 약 50명 대상으로 교내에서 공부법 세미나 자체 기획 후 개최 (2018.10.)

✓ 협업 경험이 많습니다.

- 3번의 창업과 다양한 연합 동아리(멋쟁이 사자처럼, SOPT, NEXTERS)를 통해 **디자이너, 기획자, 개발자와의 협업 경험**이 많습니다.
- 단기간의 협업 뿐만 아니라, **6~12개월 이상 오랫동안 협업을 한 경험**이 많습니다.
- **팀워크를 중시**하며 **상대방의 감정을 배려하는 커뮤니케이션을 중요시**합니다.

✓ 이론과 경험의 조화를 중요시합니다.

- **프로그램의 동작 방식, 원리, 이유**에 대해 알고 사용하려고 합니다.
- 시간이 흘러도 잘 변하지 않는 기초인 **CS(Computer Science)에 대한 지식을 중요시** 합니다.
- 이론적 학습에 그치지 않고, **이론을 실전에 적용시키려고 노력**합니다.