



백엔드 개발자 오진호

Best Practice

더 좋은 코드를 고민하는 개발자 **Bepi**입니다

이력

- 샵라이브 코리아(인턴) 백엔드 개발자(2022.03 ~ 2022.06)
- 스타트업 겨울방학 인턴십 DINO-STUDIO 백엔드 개발자 (2021.12 ~ 2022.02)
- 송실대학교 창업지원단 소속 온더브릿지 백엔드 개발자(2021. 04 ~ 2021.08)

동아리 및 커뮤니티

- 송실대학교 개발동아리 유어슈 백엔드 팀원(2021.10~)
- Google Developers Student Club 송실대 지부 Server/Cloud Member(2021.09~)

학력

- 송실대학교 IT대학 글로벌미디어학부(2016.03 ~)

DINO STUDIO

(스타트업 인턴십)

프로젝트 명 : 하이엔드(High-End)

○ 프로젝트 참여 기간

2021. 12 ~ 2022. 02

○ 프로젝트 개요

- 남녀의 만남을 주선하는 소개팅 어플입니다.
- 매일 9명의 이성을 추천합니다.
- 설정에 따라 지인에게 내 정보가 노출되지 않도록 할 수 있습니다.

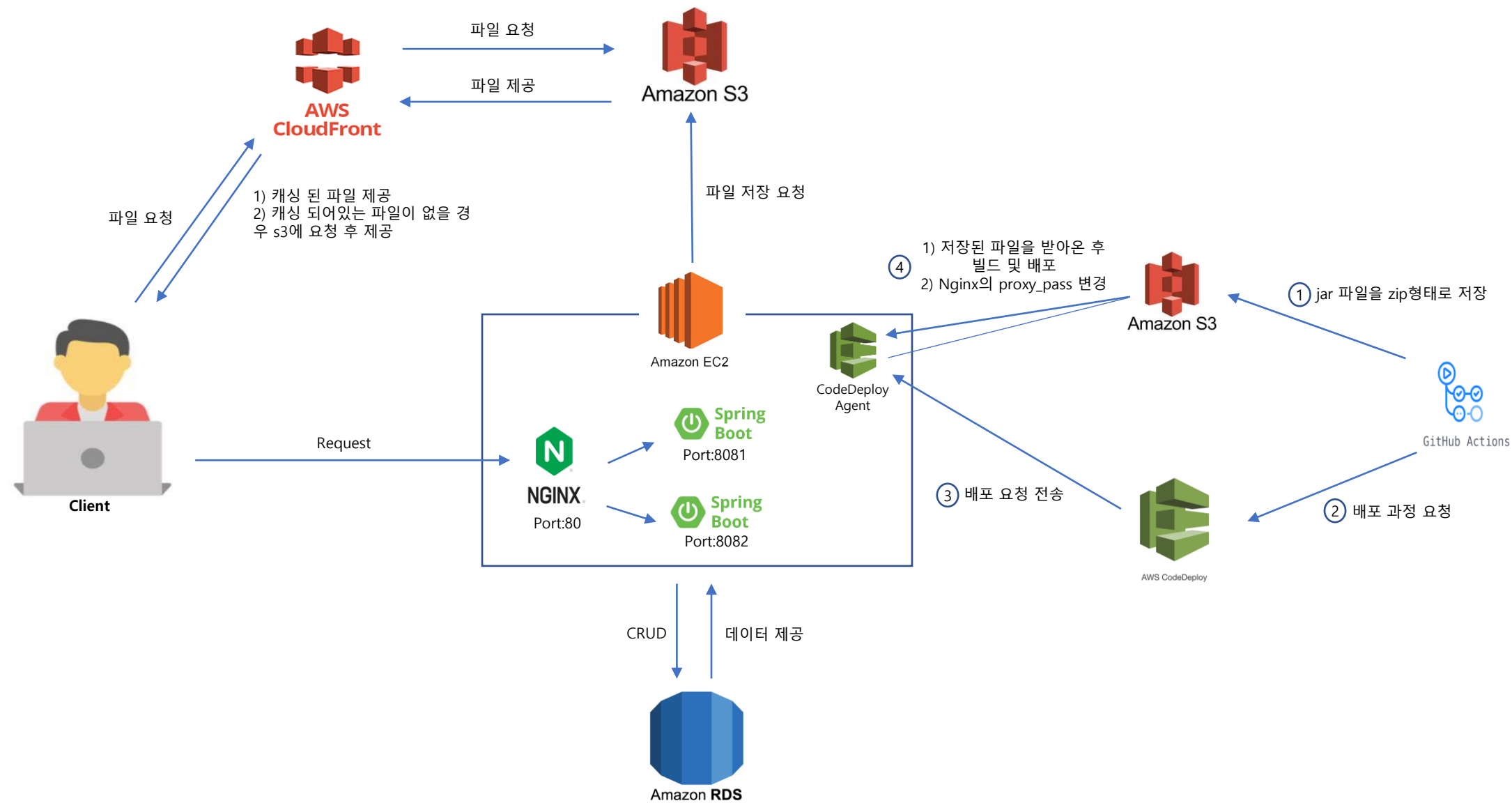
○ 기술 스택

- Java, SpringBoot, DataJpa, Querydsl, MySql
- EC2, RDS, S3, CloudFront
- CICD
- 무중단 배포

○ 나의 구현 내용

- ⊙ 서버 API 구현 및 비즈니스 로직 작성
- ⊙ 인프라 구축 - EC2, RDS, S3, CloudFront
- ⊙ Spring WebClient를 이용한 외부 API와의 통신
- ⊙ 스웨거를 이용한 API 문서 자동화
- ⊙ CICD와 무중단 배포

○ 서버 구조도



○ 아는 사람 피하기 기능

- 매핑 테이블을 이용하여 비즈니스 요구사항을 풀어냈습니다.

- '아는 사람 피하기' 기능을 사용할 경우 핸드폰에 저장되어 있는 지인은 추천 되지 않으며 나의 정보를 지인에게 노출시키지 않을 수 있습니다.

◎ 시나리오 예시 (순서)

- 1) 유저 A는 유저 B의 번호를 가지고 있고, 유저 B는 유저 A의 번호를 모릅니다. 이 때 유저 A가 아는 사람 피하기를 신청합니다.
- 2) 시간이 지나 유저 B가 아는 사람 피하기 기능을 신청합니다.
- 3) 이 과정 이후 유저 A가 아는 사람 피하기 기능을 취소하더라도 유저 A와 유저 B는 서로 추천되어서는 안됩니다. B는 여전히 아는 사람을 피하기 기능을 사용하고 있고, 비록 유저 B의 핸드폰에 유저 A의 번호가 없더라도 이전 과정을 통해서 두 사람이 아는 사이임을 알 수 있기 때문입니다.

◎ 데이터 베이스 설계

1) User 테이블에 기능 사용 여부 컬럼 추가

- '아는 사람 피하기 기능' 사용 여부를 알려주는 컬럼이 추가됩니다.

2) AcquaintancePhoneNumber 테이블

- user 핸드폰에 등록된 번호 목록을 저장하는 테이블입니다.

- user 테이블과 N:1 관계입니다.

3) AcquaintanceRelation 테이블

- 아는 사람 관계를 미리 저장하여 로직 수행 시간을 단축하기 위한 테이블입니다.

- 두 user가 아는 사이임을 나타냅니다.

- user 대 user 의 N:N 관계를 풀어냈습니다.

◎ 데이터 베이스 예시



○ 아는 사람 피하기 기능

지인 등록은 두가지 상황에서 동작합니다.

- 1) 이미 가입한 유저가 아는 사람 피하기 기능을 요청할 때
- 2) 새로운 유저가 가입할 때

1번 상황에서 지인의 번호를 모두 받은 후 이미 가입되어있는 지인들을 피하고,
2번 상황마다 미리 등록해놓은 번호를 이용하여 지인 여부를 분별한다면 모든 상황을 대처할 수 있습니다.

◎ 상황 1 – 아는 사람 피하기 기능 On

- 1) User 핸드폰에 저장된 모든 번호를 받아옵니다. 이후 Acquaintance PhoneNumber 테이블에 저장합니다.
- 2) 등록된 번호 목록을 통해 아는 사이인 기존 유저들을 찾고 AcquaintanceRelation 테이블에 저장합니다.

◎ 상황 2 - 새로운 유저 가입

- 1) 신규 가입자의 전화번호를 통해 기존 유저와 아는 사이임을 AcquaintanceRelation Table에 등록합니다.

◎ 상황 3 – 아는 사람 피하기 기능 Off

- 1) AcquaintancePhoneNumber 테이블에서 User와 연결되어있는 모든 Row를 삭제합니다.
- 2) AcquaintanceRelation 테이블은 User와 연결되어 있는 모든 Row를 지우면 안됩니다.
- UserA와 아는 사이라고 등록되어있는 **UserB가 아는 사람 피하기 기능을 사용하고 있다면** Row를 **삭제하지 않도록** 하였습니다

○ 아는 사람 피하기 기능

◎ DB 인덱싱을 통한 부하 감소

- AcquaintanceRelation 테이블에 index를 사용하여 서비스의 전체적인 로직 시간을 단축하였습니다.

◎ 인덱싱을 선택한 이유

1) 개발적 관점

- 이성 추천, 게시물 리스트 조회 등 대부분의 비즈니스 로직에서 지인 여부를 확인해야 했고 그에 따라 **Select문을 사용하는 경우가 update, delete, insert를 사용하는 경우보다 압도적으로 빈번했기** 때문입니다.

2) UX적 관점

- 사용자가 지인 번호를 등록 또는 삭제하는 경우 시간이 더 소요되는 것에 대해 불편함을 느끼는 것보다, 제공하는 서비스가 전반적으로 빠르게 동작하는 것에 대해 더 큰 만족감을 느낀다고 판단하였습니다.

○ 로그인

◎ JWT, Argument Resolver 사용

1. JWT를 사용하여 로그인 기능을 구현하였습니다.
2. Refresh Token을 사용하여 자동 로그인 기능을 구현하였습니다.
3. 반복되는 코드를 줄이기 위해 Spring에서 제공하는 Argument Resolver를 사용하였습니다.
기존 방법 : interceptor에서 JWT의 데이터를 추출하여 HttpServletRequest를 통해 컨트롤에 전달
리팩토링 : Argument Resolver를 사용하여 Controller에 반복되는 코드를 줄임

○ CICD, 무중단 배포

⊙ CICD - github action, code deploy, s3

⊙ Nginx를 이용한 무중단 배포

⊙ 순서

- s3에 jar파일 업로드
- code deploy를 이용하여 덮어쓰기 후 8081, 8082 중 사용하지 않는 포트로 배포
- Nginx 리버스 프록시 포트 변경

ShopLive Korea

서비스 명 : 샵라이브

○ 인턴십 기간

2022. 03 ~ 2022. 06

○ 서비스 설명

- 라이브 커머스 서비스
- 라이브 영상을 시청하며 실시간으로 물건을 구매할 수 있는 서비스입니다.
- B to B 서비스로서 고객사의 UI 화면에 플러그인 형식으로 서비스를 제공합니다.

○ 나의 구현 내용

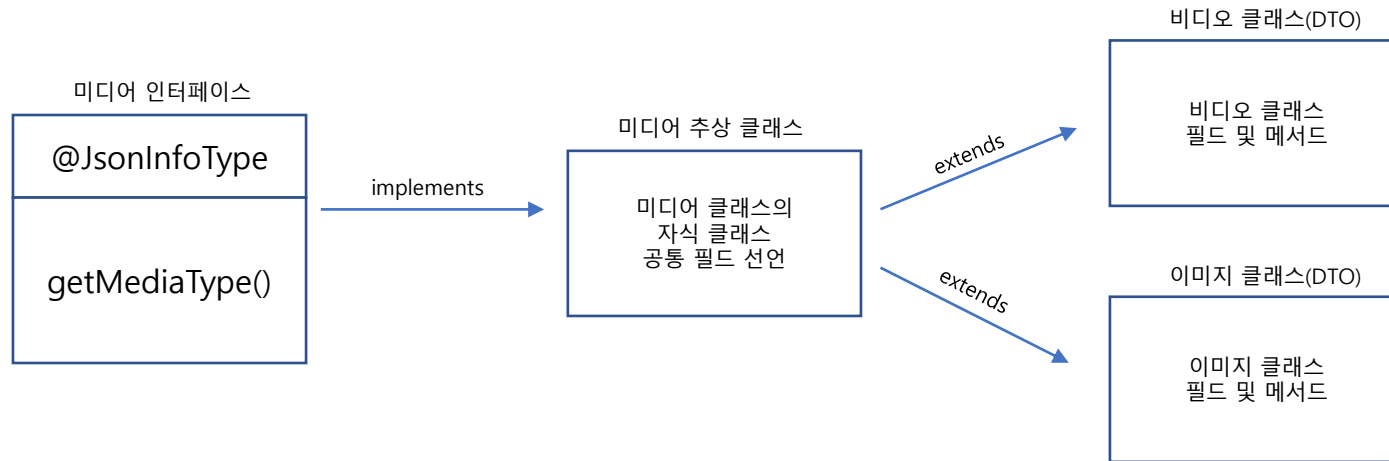
- ⊙ 서버 API 로직 작성
- ⊙ arcus(mem-cached)를 이용한 캐싱
- ⊙ 리팩토링을 통한 시간 단축 및 코드 간소화

○ 코드 중복 제거

⊙ 상황

- 비디오 클래스와 이미지 클래스는 미디어 클래스를 상속합니다.
- 비디오 클래스와 이미지 클래스는 다른 필드를 가지고 있으므로 요청에 대해 다른 DTO를 사용합니다. 따라서 비슷한 코드가 여러 번 작성됩니다.
- 미디어의 종류가 추가될 때마다 코드 중복이 다시 발생합니다.

⊙ 인터페이스와 추상클래스를 이용한 코드 중복 해결 및 확장성 증가



1. JsonInfoType 어노테이션을 이용하여 메시지 컨버터가 요청에 알맞은 구체 클래스로 데이터를 매핑합니다.
2. getMediaType 메서드를 이용하여 코드 레벨에서 인자로 들어온 클래스의 타입을 확인합니다. 이를 통해 어떤 구체 클래스의 객체로 역직렬화가 되었는지 알 수 있습니다. 이를 이용하여 다음 서비스 로직을 실행합니다.
3. 이를 통해 컨트롤러와 서비스에서 하나의 메서드로 모든 미디어 타입에 대한 요청에 응답할 수 있습니다.

○ 캐싱

◎ memcached를 이용한 캐싱

- 단시간에 많은 요청이 예상되는 로직에 캐싱 적용.
- 메모리의 데이터가 날아가더라도 서비스에 장애를 일으키지 않았기에 memcached를 적용

○ 코드 리팩토링

◎ DB 부하 감소를 위해 하나의 요청이 실행 될 때마다 생기는 쿼리의 개수를 줄임

기존 방법 : For문을 돌며 데이터를 row별로 받아오는 방식을 사용했고, 이에 따라 DB 커넥션이 많이 발생하여 서버의 부하가 큼

리팩토링 : Stream을 활용하여 필요로하는 Row의 PK를 리스트로 한 번에 모아 쿼리를 전송함으로써 DB 커넥션을 감소시킴

Side Project (소규모 프로젝트)

프로젝트 명 : 송실대학교 학식 알림이

약 2000명의 사용자를 보유하고 있는 학식 알림이 서비스입니다.

○ 프로젝트 기간

2021. 11 ~ 2021. 12

○ 기술 스택

- Java, SpringBoot, DataJpa, Jsoup라이브러리
- EC2, RDS
- CICD
- 무중단 배포

○ 프로젝트 개요

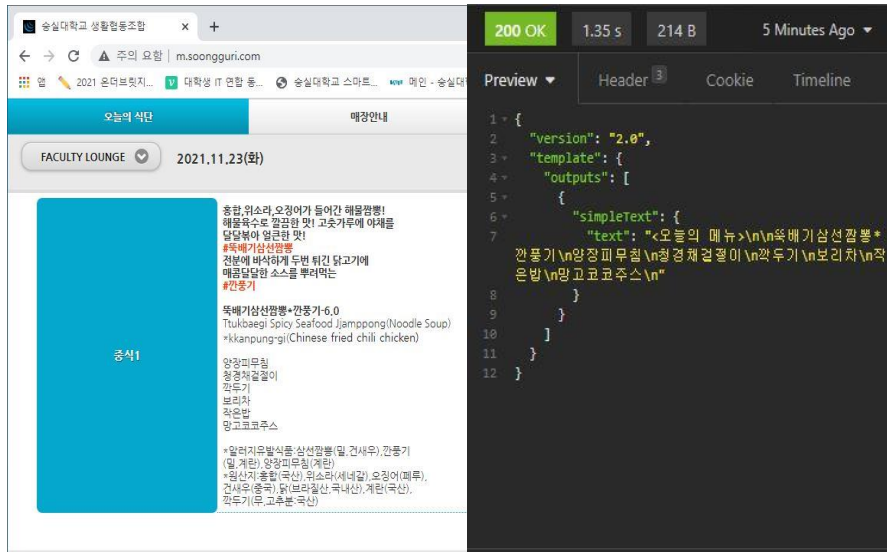
1. 웹 크롤링을 통해 학식메뉴를 보여주는 카카오톡 챗봇입니다.
2. 카카오 오픈 빌더 API 활용

○ 나의 구현 내용

- ⊙ Jsoup 라이브러리를 이용한 웹 크롤링
- ⊙ 자동화를 통한 성능 개선
- ⊙ 서비스 기획 ~ 배포 모든 과정 참여

○ 깃허브

URL : <https://github.com/ohjinhokor/SoongsilHaksik>



○ Jsoup Library를 통한 웹 크롤링

학식 데이터를 추출한 후, Json형식의 response를 제공하는 API입니다.

1. URL을 통해 해당 사이트에 접근합니다.
2. 추출하고자 하는 데이터를 html코드에 맞게 설정합니다.
3. 데이터를 추출한 후 카카오 챗봇 API 형식에 맞는 'JSON Response'를 반환합니다.



성능 개선을 위한 Refactoring

문제가 되었던 부분: 요청이 들어올 때마다 크롤링을 하기 때문에 속도가 느림

<해결 방안>

1. 위의 문제를 해결하기 위해 @Scheduled 어노테이션을 사용하여 자동화
2. 자정이 지나면 자동으로 그 날의 학식 데이터를 데이터 베이스에 저장.
3. 이후 학식 정보를 요청하는 Request가 들어오면 데이터 베이스에 저장해 놓은 데이터를 반환 함

<리팩토링 결과> 수행 시간이 1561ms -> 213ms로 줄어드는 성능 향상을 보였습니다

○ 순서도

