Best Practice

더 좋은 코드를 고민하는 개발자 **Bepi**입니다

학력

- 숭실대학교 IT대학 글로벌미디어학부(2016.03 ~)

이력

- 스타트업 겨울방학 인턴십 DINO-STUDIO 백엔드 개발자 (2021.12 ~ 2022.02)
- 숭실대학교 창업지원단 소속 온더브릿지 백엔드 개발자(2021. 04 ~ 2021.08)

동아리 및 커뮤니티

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

백엔드 개발자 오진호

Resume: 링크
Email: ohjinho7@gmail.com
Github: https://github.com/ohjinhokor

DINO STUDIO
(스타트업 인턴십)

프로젝트 명 : 하이엔드(High-End)

○ 프로젝트 기간

2021. 12 ~ 2022. 02

○ 기술 스택

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

○ 프로젝트 개요

- 남녀의 만남을 주선하는 소개팅 어플입니다.
- 매일 9명의 이성을 추천합니다.
- 마음에 드는 이성에게 호감 표현을 할 수 있습니다.

○ 나의 구현 내용

⊙ 서버 API 구현 및 비즈니스 로직 작성

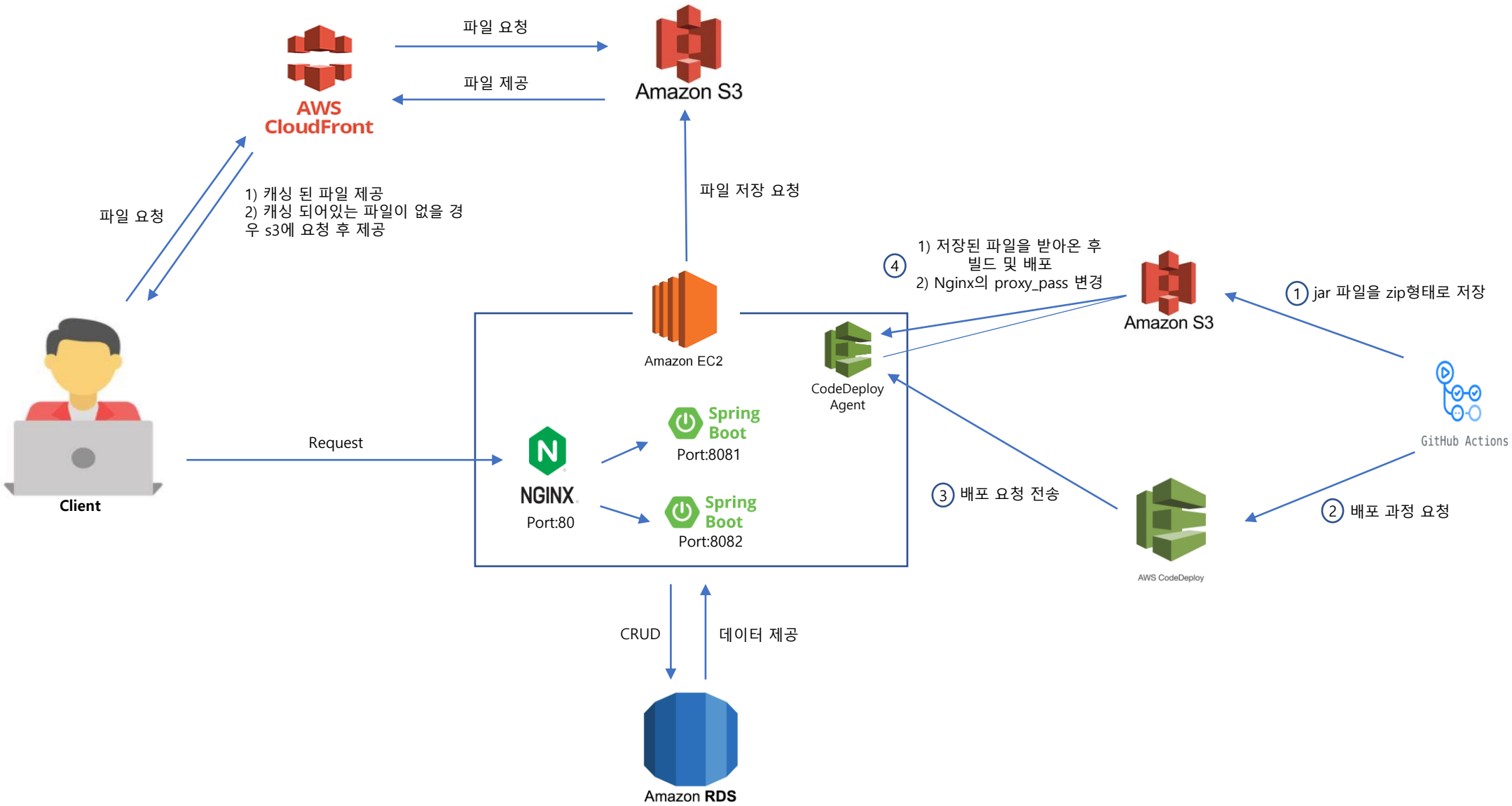
⊙ 인프라 구성 - EC2, RDS, S3, CloudFront

⊙ Spring WebClient를 이용한 외부 API와의 통신

⊙ 스웨거를 이용한 API 문서 자동화

⊙ CICD와 무중단 배포

○ 서버 구조도


파일 요청
파일 제공
AWS CloudFront
Amazon S3
파일 요청
1) 캐싱 된 파일 제공
2) 캐싱 되어있는 파일이 없을 경우 s3에 요청 후 제공
파일 저장 요청
4
1) 저장된 파일을 받아온 후 빌드 및 배포
2) Nginx의 proxy_pass 변경
Amazon S3
1 jar 파일을 zip형태로 저장
Amazon EC2
CodeDeploy Agent
GitHub Actions
Request
Client
NGINX
Port:80
Spring Boot
Port:8081
Spring Boot
Port:8082
3 배포 요청 전송
2 배포 과정 요청
AWS CodeDeploy
CRUD
데이터 제공
Amazon RDS

ㅇ 아는 사람 피하기 기능

- 사용자가 '아는 사람 피하기' 기능을 사용할 경우 핸드폰에 저장되어 있는 지인은 추천 되지 않습니다.

⊙ 시나리오 예시 (순서)

1) 유저 A는 유저 B의 번호를 가지고 있고, 유저 B는 유저 A의 번호를 모릅니다. 이런 상황에서 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 테이블

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

2) AcquaintancePhoneNumber 테이블

- 해당 user보다 늦게 가입하는 지인을 피하기 위하여 user의 핸드폰에 등록된 번호 목록을 저장하는 테이블입니다.

- user 테이블과 N:1 관계입니다.(user의 pk를 외래키로 합니다)

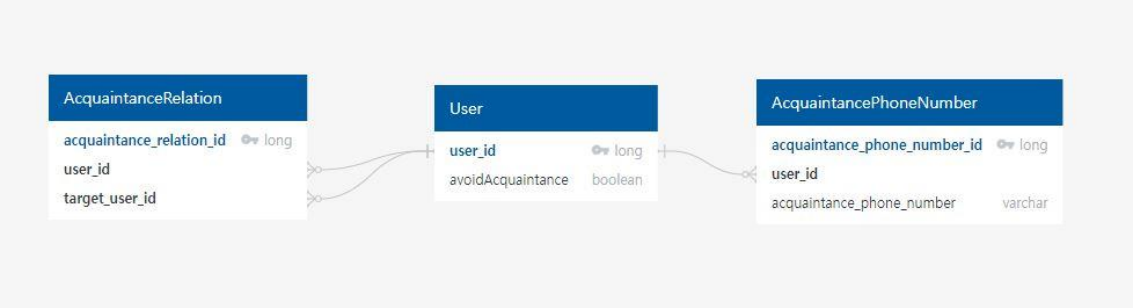
3) AcquaintanceRelation 테이블

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

- 두 user가 아는 사이임을 나타냅니다.

- user 대 user 의 N:N 관계를 풀어냈습니다.

⊙ 데이터 베이스 예시



○ 아는 사람 피하기 기능

아는 사람 피하기 기능은 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) 방금 전 등록한 번호 목록을 통해 아는 사이인 기존 유저들을 찾고 AcquaintanceRelation 테이블에 저장합니다.

2의 예시코드

```
for (String phoneNumber : phoneNumbers) {
    userRepository.findByPhoneNumber(phoneNumber).ifPresent(
            targetUser -> {registerNewAcquaintance(user, targetUser);});
}
```

⊙ 상황 2 - 새로운 유저 가입

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

⊙ 상황 3 – 아는 사람 피하기 기능 Off

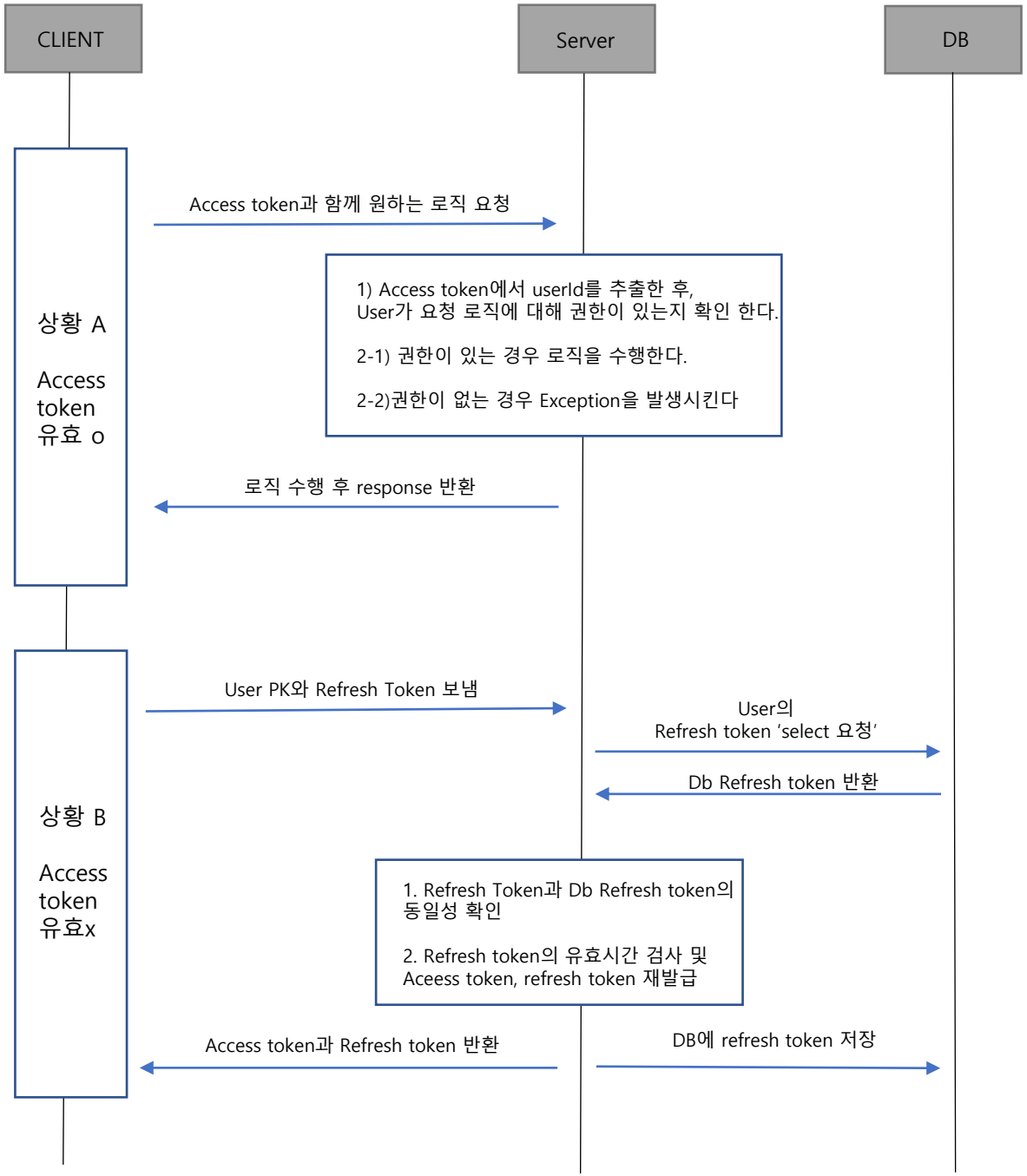
1) AcquaintancePhoneNumber 테이블에서 User와 연결되어있는 모든 Row를 삭제합니다.

2) AcquaintanceRelation 테이블은 User와 연결되어 있는 모든 Row를 지우면 안됩니다.

- UserA와 아는 사이라고 등록되어있는 **UserB가 아는 사람 피하기 기능을 사용하고 있다면** Row을 **삭제하지 않도록** 하였습니다

2의 예시코드

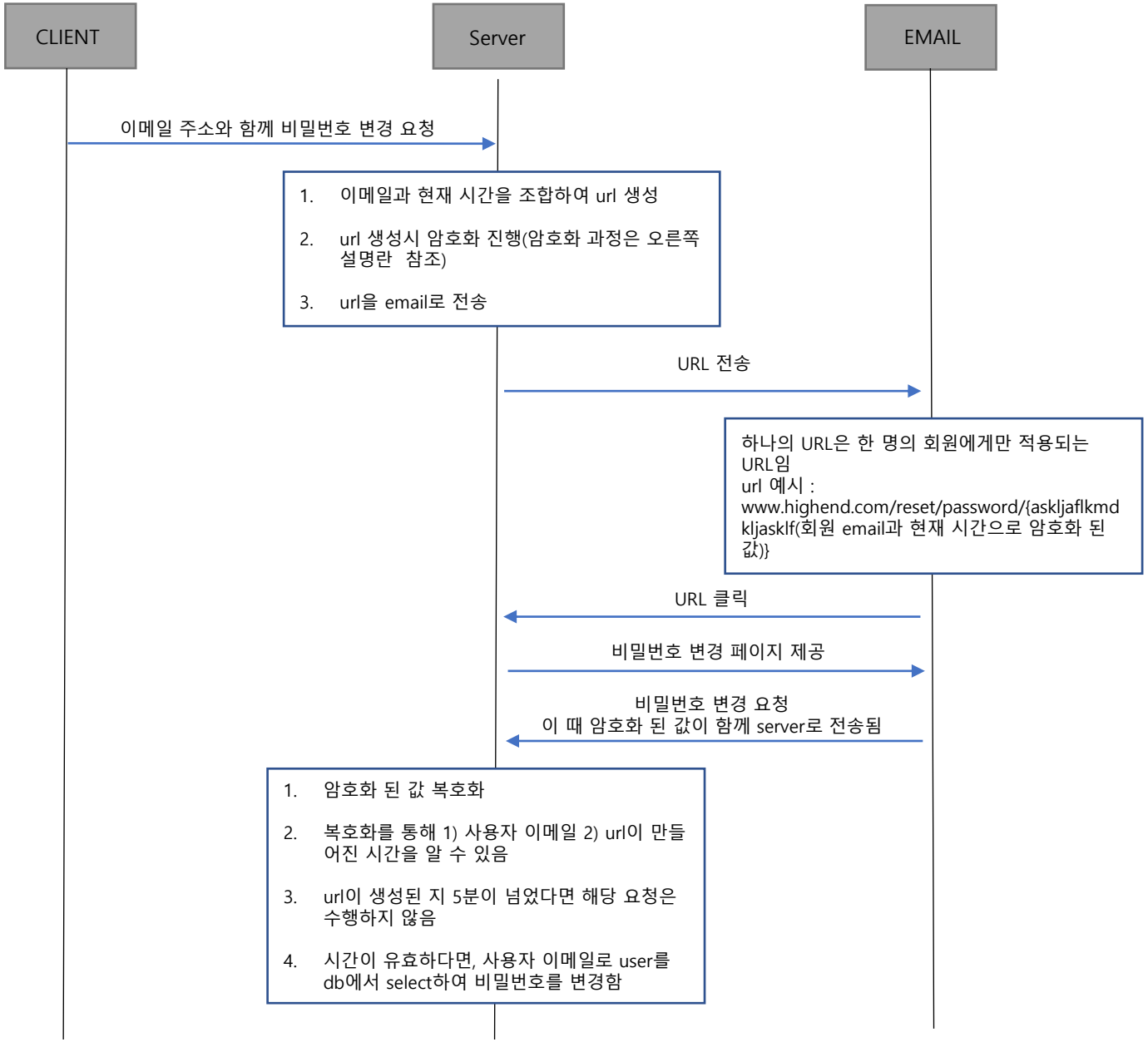
```
acquaintanceRelationRepository
        .findMyAcquaintanceList(user)
        .stream()
        .filter(acquaintance -> !acquaintance.getAvoidingAcquaintance())
        .forEach(acquaintance ->
acquaintanceRelationRepository.deleteAcquaintanceRelation(userRepository.findById(userId).orElseThrow(
()-> new NoSuchElementException("존재하지 않는 회원입니다")),acquaintance));
```



ㅇ Interceptor, JWT를 사용한 보안

- 개인 정보가 많이 사용되는 서비스였기 때문에 권한이 필요한 모든 요청마다 token에서 user의 pk를 추출하여 요청 로직에 대한 권한을 확인하였습니다.

1. Controller로 들어가기 전 Interceptor에서 Token의 유효시간을 확인하는 작업을 진행합니다.
2. Token의 시간이 유효하다면, Interceptor에서 userId를 추출한 후 Controller로 userId 값을 넘겨줍니다.
3. 넘겨 받은 userId를 이용하여 User가 요청 로직에 대해 권한이 있는지 확인합니다. 예를 들어 특정 회원의 프로필을 조회하는 로직을 요청한다면 그 프로필이 해당 User에게 허락된 프로필인지 확인하는 절차를 거칩니다.
4. Refresh Token을 사용하여 보안을 높이고, 자동 로그인이 가능하도록 하였습니다.
5. 로그아웃을 하면 Refresh Token을 무효화 시킵니다.
6. Refresh Token의 저장위치는 DB로 하였습니다.



ㅇ 비밀번호 찾기 기능

- 자바의 Cipher 클래스를 이용하여 url 암호화

⊙ 암호화 방법

- AES/CBC/PKCS5Padding

⊙ 암호화 이용 과정

1. 현재 시간과 User의 Email을 암호화합니다

2. 암호화 된 값과 함께 url을 만든 후 User의 email로 전송합니다.

3. Client로부터 암호화 된 값과 함께 비밀번호 변경을 요청받습니다.

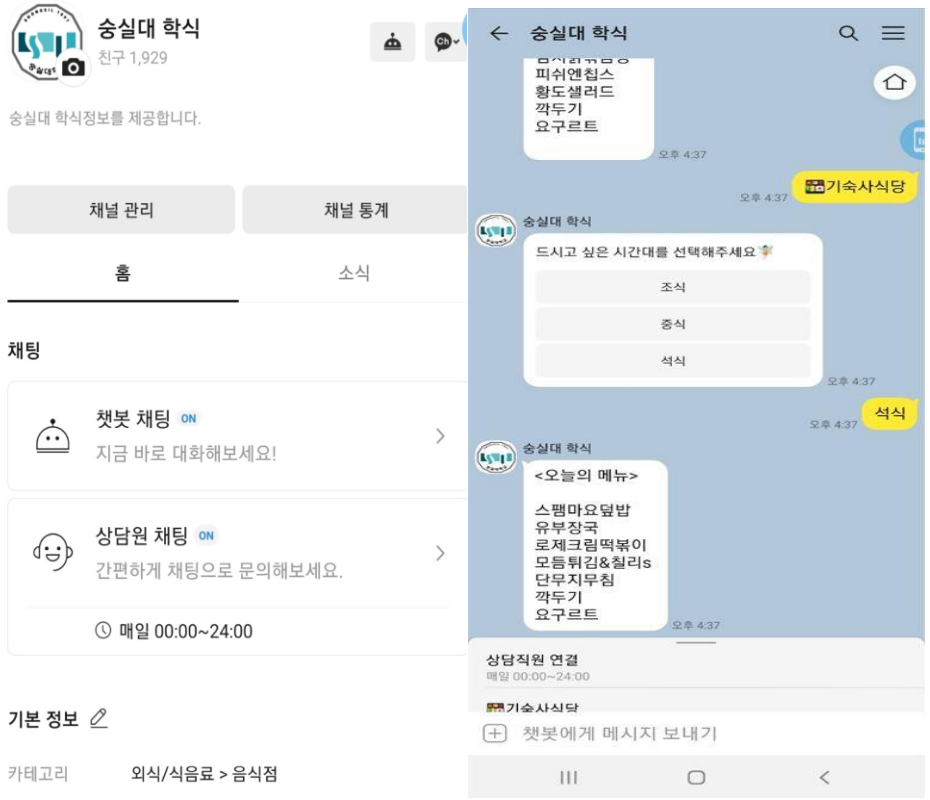
4. 암호화 된 값을 복호화 하여 url이 만들어진 시간과 Email을 얻어냅니다.

5. 유효시간을 확인 한 후 해당 User의 비밀번호를 변경합니다.

Side Project (소규모 프로젝트)

프로젝트 명 : 숭실대학교 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

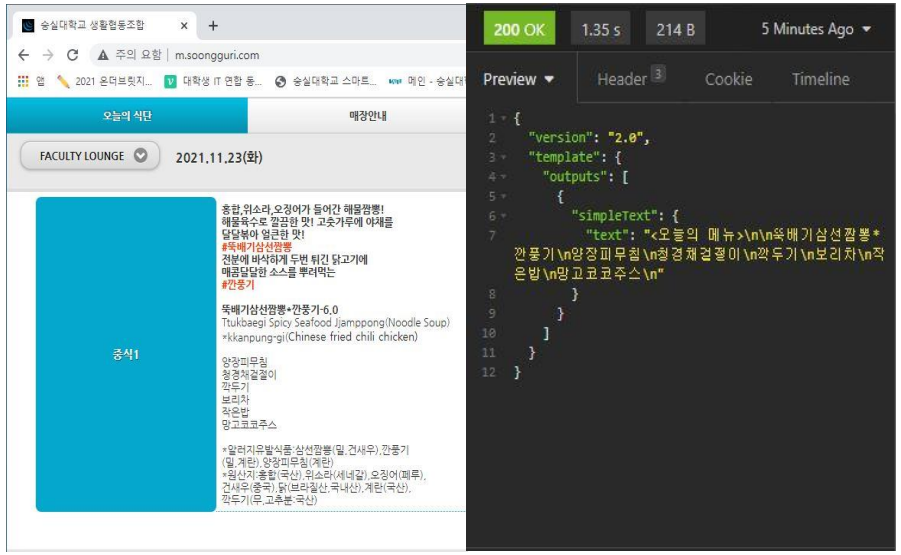
⊙ Jsoup 라이브러리를 이용한 웹 크롤링

⊙ 자동화를 통한 **성능 개선**

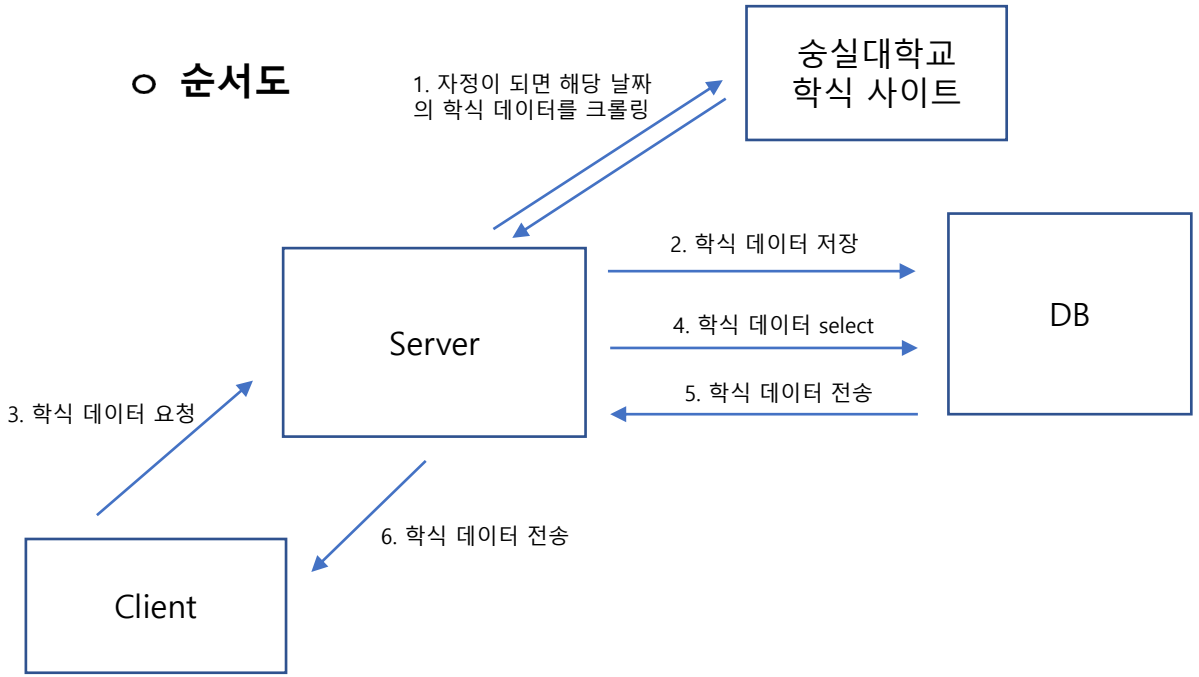
⊙ 서비스 기획 ~ 배포 모든 과정 참여

○ 깃허브

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



ㅇ 순서도



ㅇ 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