



안녕하세요.  
백엔드 개발자 윤성운입니다.

저는 팀 공동의 목표를 1순위로 생각합니다.  
팀원의 요구사항을 명확히 이해하며 효과적인 협업에 힘쓰겠습니다.

---

## Contact

Phone 010-5287-6691

Email [ysu6691@naver.com](mailto:ysu6691@naver.com)

Github <https://github.com/ysu6691>

---

## Skills

Back-end • Java Python

- Spring Boot Django
- MySQL Redis
- AWS EC2 AWS S3

Front-end • JavaScript TypeScript  
• React

Tools • Github GitLab  
• IntelliJ  
• Jira

# Projects (1/3)



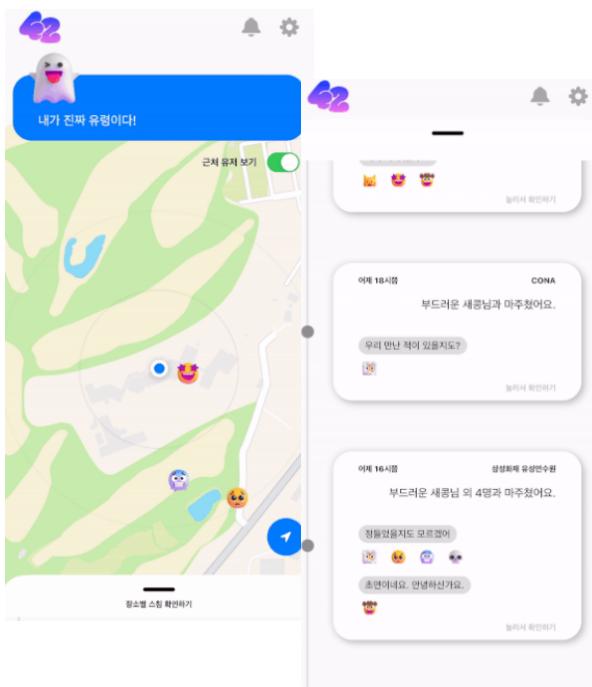
42 2023.04.10 ~ 2023.05.19 (6주)

근거리 위치 기반 익명 SNS입니다.

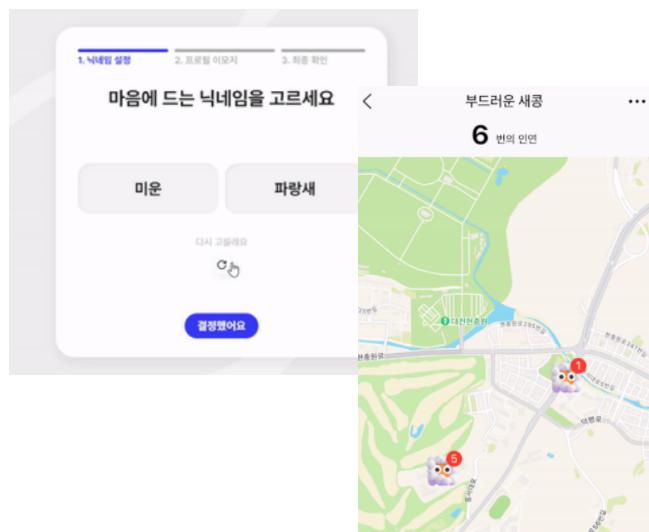
Back-end 업무를 맡아 Spring boot를 이용해 개발하였습니다.

<https://github.com/people42/people42>

## 주요 기능



- 내 주변에서 스쳤던 사람들의 생각을 조회할 수 있습니다.
- 근처 주변 유저 또한 실시간으로 확인할 수 있습니다.



- 한정된 유저 닉네임을 제공해 익명을 유지합니다.
- 여러 번 스친 다른 익명의 유저와의 기록을 조회할 수 있습니다.

# Projects (1/3)

## 담당 역할

사용자 위치 기반 근처 사용자 찾기 알고리즘 개발

OAuth2.0기반 구글, 애플 회원 인증 개발

FCM을 이용한 알림 기능 개발

REST API 개발

데이터베이스 설계

서비스 기획

## BE 개발환경

언어 JAVA 11

프레임워크 Spring Boot 2.7.10

DB MariaDB, Redis

IDE IntelliJ

# Projects (1/3)

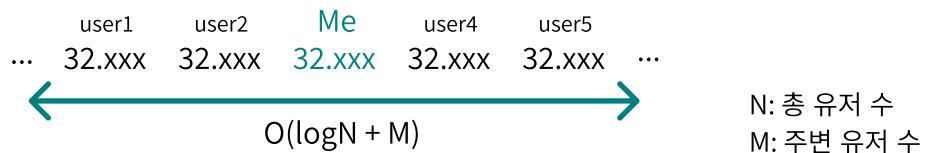
## 사용자 위치 기반 근처 사용자 찾기 알고리즘

백그라운드에서 모든 유저로부터 지속적으로 요청받는 API의 시간 복잡도를 크게 개선하였습니다.

문제 상황      유저가 위치를 갱신할 때 주변 유저를 찾기 위해 모든 유저를 조회



해결 방법      Skiplist 자료구조를 사용해 유저 위치를 정렬해 관리



## Redis 동시성문제 해결

RDB에 데이터를 추가하기 위한 사전 필터로 사용하고 있던 Redis에 [동시성 문제](#)가 발생하여 RDB에 중복된 데이터가 동시에 생기는 문제가 발생했습니다.

Redis에 [분산lock](#)을 걸어 최대한 성능을 보존하면서도 동시성 문제를 해결할 수 있었습니다.

| MariaDB [fourtytwo]> select * from brush; |                            |                            |              |              |           |           |           |  |
|-------------------------------------------|----------------------------|----------------------------|--------------|--------------|-----------|-----------|-----------|--|
| brush_idx                                 | created_at                 | updated_at                 | message1_idx | message2_idx | place_idx | user1_idx | user2_idx |  |
| 76                                        | 2023-04-20 13:10:10.000000 | NULL                       | 100          | 101          | 61        | 100       | 101       |  |
| 253                                       | 2023-05-16 23:49:00.252051 | 2023-05-16 23:49:00.252051 | 455          | 456          | 173       | 566       | 567       |  |
| 254                                       | 2023-05-16 23:49:00.252051 | 2023-05-16 23:49:00.252051 | 455          | 456          | 173       | 566       | 567       |  |
| 255                                       | 2023-05-16 23:49:00.251053 | 2023-05-16 23:49:00.251053 | 455          | 456          | 173       | 566       | 567       |  |
| 256                                       | 2023-05-16 23:49:00.251053 | 2023-05-16 23:49:00.251053 | 455          | 456          | 173       | 566       | 567       |  |
| 257                                       | 2023-05-16 23:49:00.253055 | 2023-05-16 23:49:00.253055 | 455          | 456          | 173       | 566       | 567       |  |
| 258                                       | 2023-05-16 23:49:00.253055 | 2023-05-16 23:49:00.253055 | 455          | 456          | 173       | 566       | 567       |  |
| 259                                       | 2023-05-16 23:49:00.252051 | 2023-05-16 23:49:00.252051 | 455          | 456          | 173       | 566       | 567       |  |
| 260                                       | 2023-05-16 23:49:00.260074 | 2023-05-16 23:49:00.260074 | 455          | 456          | 173       | 566       | 567       |  |
| 261                                       | 2023-05-16 23:49:00.269583 | 2023-05-16 23:49:00.269583 | 455          | 456          | 173       | 566       | 567       |  |

| MariaDB [fourtytwo]> select * from brush; |                            |                            |              |              |           |           |           |  |
|-------------------------------------------|----------------------------|----------------------------|--------------|--------------|-----------|-----------|-----------|--|
| brush_idx                                 | created_at                 | updated_at                 | message1_idx | message2_idx | place_idx | user1_idx | user2_idx |  |
| 76                                        | 2023-04-20 13:10:10.000000 | NULL                       | 100          | 101          | 61        | 100       | 101       |  |
| 262                                       | 2023-05-16 23:51:01.598366 | 2023-05-16 23:51:01.598366 | 457          | 458          | 173       | 568       | 569       |  |

# Projects (2/3)

Co-cook!

**Co-cook!** 2023.02.20 ~ 2023.04.07 (7주)

음성으로 제어하는 레시피 조작 어플리케이션입니다.

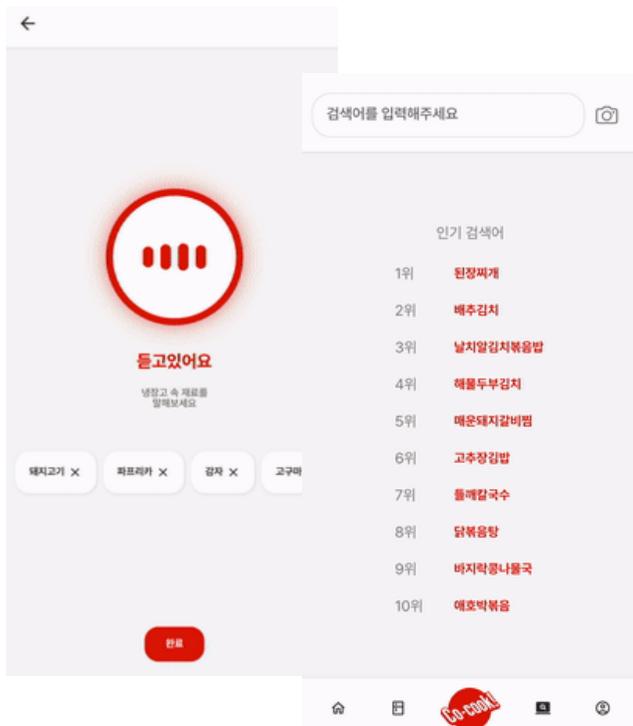
Back-end 업무를 맡아 Spring boot를 이용해 개발하였습니다.

<https://github.com/team-co-cook/co-cook>

## 주요 기능



- 음성만으로 조리 화면을 제어할 수 있습니다.
- 각 레시피에 맞는 이미지와 설명을 보면서 음성으로도 설명을 들을 수 있습니다.



- 음성으로 재료를 추가해 레시피를 검색할 수 있습니다.
- 현재 인기 검색어를 제공받을 수도 있습니다.

# Projects (2/3)

## 담당 역할

실시간성을 반영한 인기검색어 기능 개발

OAuth2.0기반 구글 회원 인증 개발

REST API 개발

데이터베이스 설계

서비스 기획

## BE 개발환경

언어 JAVA 11

프레임워크 Spring Boot 2.7.10

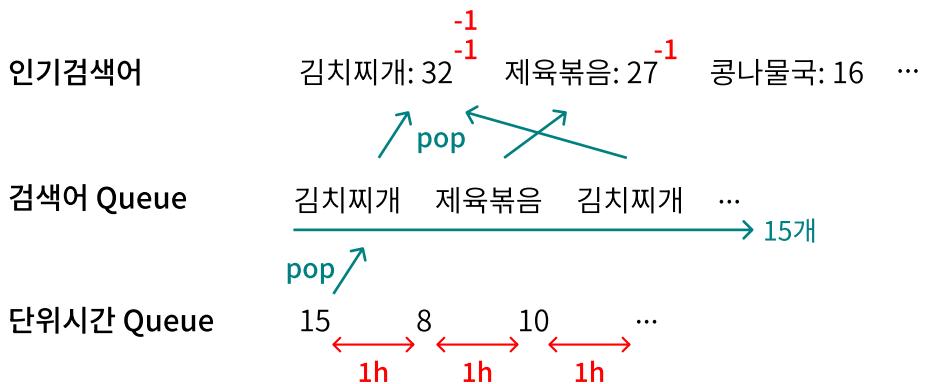
DB MySQL, Redis

IDE IntelliJ

# Projects (2/3)

## 실시간 인기검색어 구현

인기검색어 목록에 반영될 검색어 큐와 단위 시간동안의 검색 수를 저장하는 큐를 만들었습니다.  
검색어 큐에 들어있는 검색어들을 정렬하여 인기검색어의 실시간성을 유지하였습니다.



예외 처리

Front-end 개발을 하며 아쉬웠던 점을 해소하고자 발생하는 에러에 대한 상태코드를 상세하게 분기했고, 에러 메시지를 명확히 기술해 디버깅에 사용되는 리소스를 줄였습니다.

| Code | Message                                    |
|------|--------------------------------------------|
| 200  | OK                                         |
| 400  | 유효성 검사에 실패한 경우 or 이미 해당 <u>유저가</u> 찜을 한 경우 |
| 401  | 유효한 jwt 토큰이 아닌 경우                          |
| 404  | 해당 <u>레시피가</u> 존재하지 않는 경우                  |
| 405  | method가 잘못된 경우                             |

```
// 헤더가 비어있는 경우
no usages
@ExceptionHandler(MissingRequestHeaderException.class)
public ResponseEntity<ApiResponse<?>> handleMissingRequestHeaderException(MissingRequestHeaderException e) {
    ApiResponse<?> apiResponse = new ApiResponse<?>( message: "헤더에 '" + e.getHeaderName() + "'을 넣어주세요", status: 400, data: null);
    return new ResponseEntity<?>(apiResponse, HttpStatus.BAD_REQUEST);
}
```

# Projects (3/3)

## 마이 리틀 쥬라기

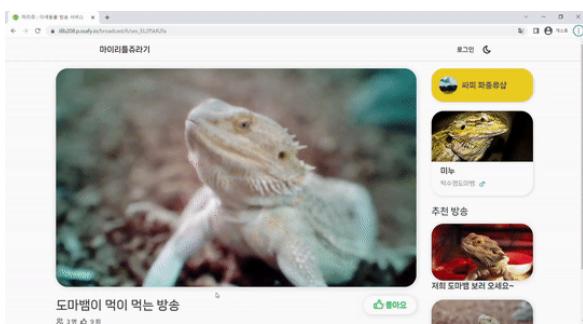
### 주요 기능

마이 리틀 쥬라기 2023.01.03 ~ 2023.02.17 (7주)

실시간 파충류 스트리밍 서비스입니다.

Front-end 업무를 맡아 React를 이용해 개발하였습니다.

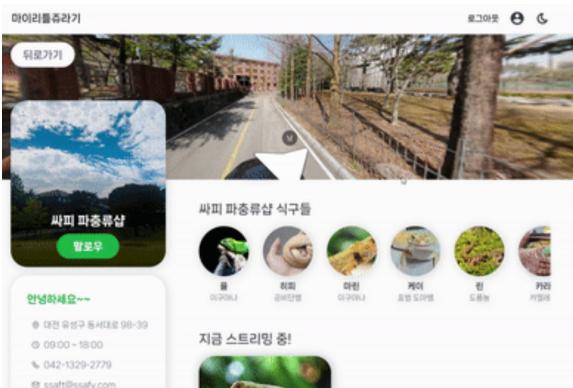
<https://github.com/My-Little-Jurassic/Marizoo>



- 선택한 파충류 스트리밍을 시청합니다.
- 해당 동물을 보유한 샵 정보를 확인할 수 있습니다.
- 다른 추천 방송도 제공받습니다.



- DB에 등록된 파충류의 상세 정보를 제공받습니다.
- 해당 방송이 방송중일 경우 해당 방송으로 이동할 수 있습니다.



- 특정 샵의 상세 정보와 보유한 동물을 확인할 수 있습니다.
- 해당 샵을 팔로우할 수 있습니다.

# Projects (3/3)

## 담당 역할

WebRTC를 이용한 실시간 스트리밍 기능 구현

동물 상세 페이지 구현

샵 상세 페이지 구현

UI 구현

UX 설계

서비스 기획

## FE 개발환경

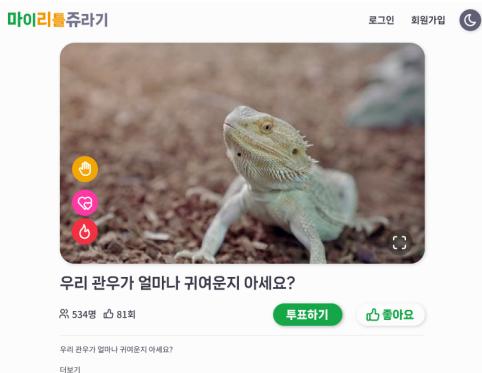
언어                    TypeScript

프레임워크            React

IDE                    VSCode

# Projects (3/3)

## WebRTC



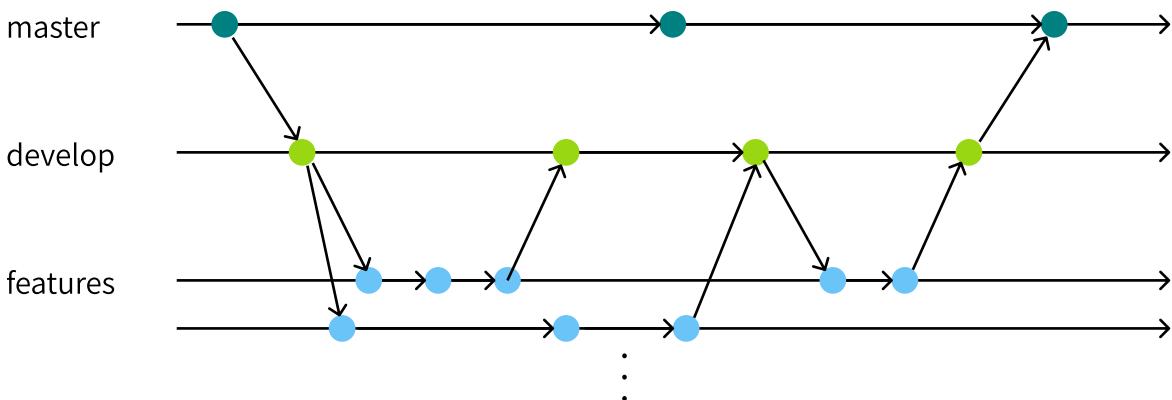
WebRTC를 이용한 실시간 스트리밍 기능을 맡았습니다.

한 방송에 여러 시청자가 들어와야 하는 특성상  
mesh 방식 대신 미디어 서버를 두고자 했습니다.

따라서 미디어 서버를 애플리케이션에 쉽게 적용할 수 있는  
OpenVidu 라이브러리를 사용해 구현하였습니다.

## Git Flow

Git Flow 전략을 사용하여 기능별로 branch를 만들어 관리하였습니다.  
Commit convention을 준수하여 git log의 가독성을 높였습니다.



## Front-end를 맡은 이유

Back-end 개발자로서 Front-end 개발자와의 협업 능력은 필수적이라고 생각했습니다.  
그래서 저는 하나의 프로젝트에서 Front-end 개발을 해봄으로써,  
Front-end 개발자 입장에서 겪을 수 있는 어려움을 경험해보고자 했습니다.