



**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

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

Education

삼성 청년 SW 아카데미 2022.07.06 ~ 2023.06.30

Skills

Back-end

- Java Python
- Spring Boot Django
- MySQL Redis

Front-end

- JavaScript TypeScript
- React

Tools

- Github GitLab
- IntelliJ
- Jira

Projects (1/4)



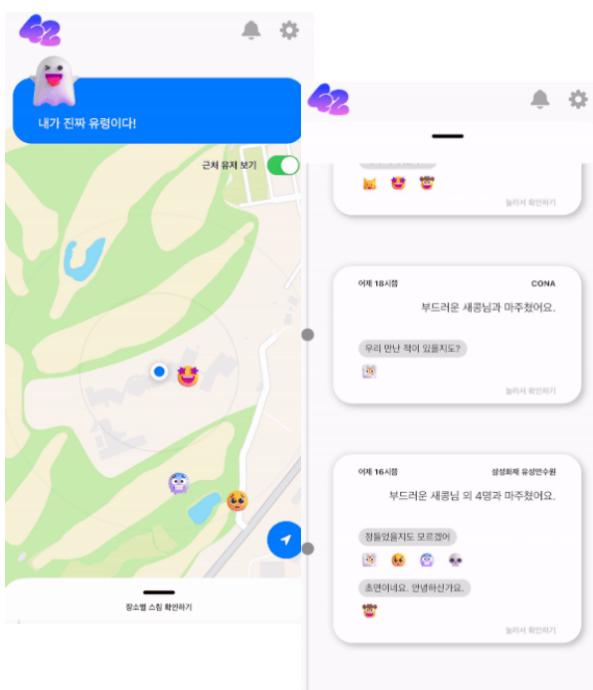
42 2023.04.10 ~ 2023.05.19 (6주)

위치 기반 근거리 SNS입니다.

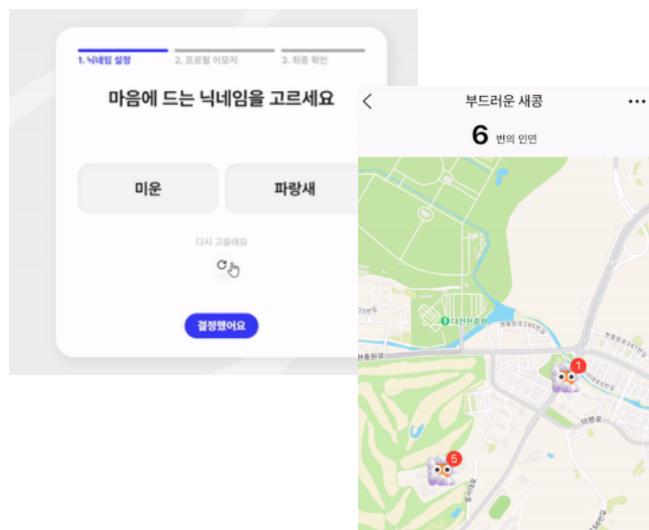
백엔드 업무를 맡아 Spring boot를 이용해 개발하였습니다.

<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 (1/4)

담당 역할

사용자 위치 기반 근처 사용자 찾기 알고리즘 개발

OAuth2.0기반 구글, 애플 회원 인증 개발

FCM을 이용한 알림 기능 개발

Google Map API를 이용한 근처 장소 조회 기능 개발

데이터베이스 설계

백엔드 개발환경

언어 JAVA 11

프레임워크 Spring Boot 2.7.10

DB MariaDB, Redis

IDE IntelliJ

Projects (1/4)

사용자 위치 기반 근처 사용자 찾기 알고리즘

백그라운드에서 모든 유저로부터 지속적으로 요청받는 API의 시간 복잡도를 크게 개선하였습니다.

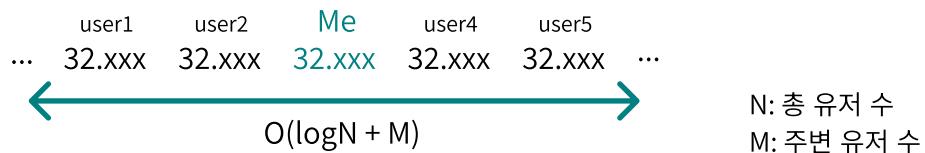
문제 상황

유저가 위치를 갱신할 때 주변 유저를 찾기 위해 모든 유저를 조회



해결 방법

Redis의 skip list 자료구조를 사용해 유저 위치를 정렬해 관리



Redis 동시성 문제 해결

RDB에 데이터를 추가하기 위해 사전 필터로 사용하고 있던 Redis에 동시성 문제가 발생하여 중복된 데이터가 동시에 생기는 문제가 발생했습니다.

이를 위해 분산锁을 걸어 최대한 성능을 보존하면서도 동시성 문제를 해결할 수 있었습니다.

해결 전

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 23:49:00.251058	2023-05-16 23:49:00.251058	455	456	173	566	567
256	2023-05-16 23:49:00.251058	2023-05-16 23:49:00.251058	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

해결 후

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 (2/4)

Co-cook!

Co-cook! 2023.02.20 ~ 2023.04.07 (7주)

음성으로 제어하는 레시피 서비스입니다.

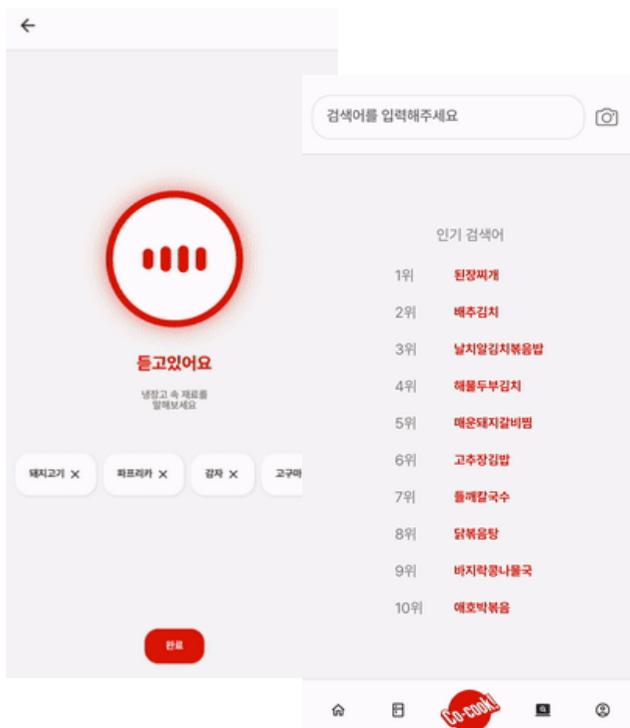
백엔드 업무를 맡아 **Spring boot**를 이용해 개발하였습니다.

<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 (2/4)

담당 역할

실시간성을 반영한 인기검색어 기능 개발

OAuth2.0기반 구글 회원 인증 개발

전역 예외 처리 기능 개발

초기 레시피 데이터 저장 기능 개발

데이터베이스 설계

백엔드 개발환경

언어 JAVA 11

프레임워크 Spring Boot 2.7.10

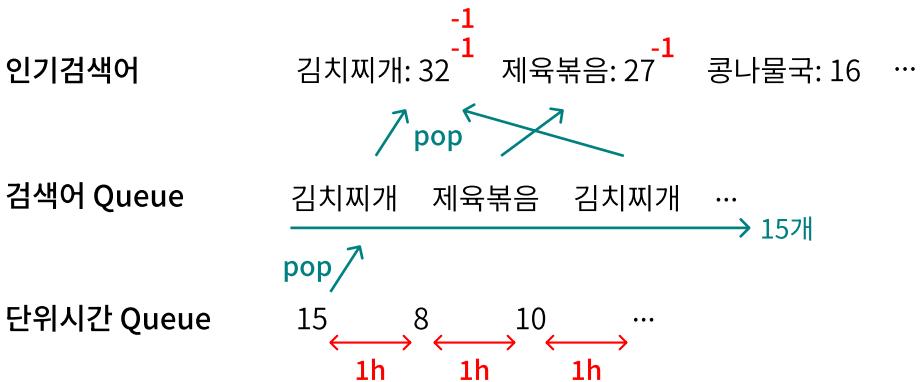
DB MySQL, Redis

IDE IntelliJ

Projects (2/4)

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 토큰을 한 경우
401	유효한 jwt 토큰이 아닌 경우
404	해당 레시피가 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 (3/4)

마이 리틀 쥬라기

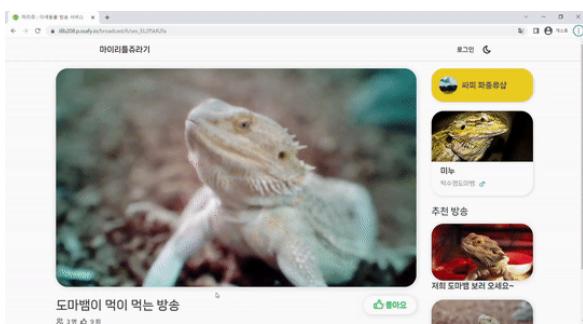
마이 리틀 쥬라기 2023.01.03 ~ 2023.02.17 (7주)

실시간 파충류 스트리밍 서비스입니다.

프론트엔드 업무를 맡아 React를 이용해 개발하였습니다.

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

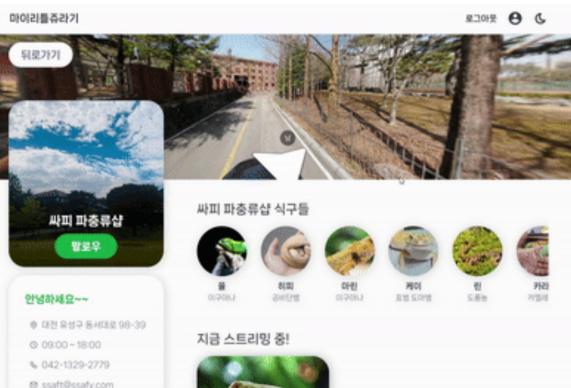
주요 기능



- 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 (3/4)

담당 역할

WebRTC를 이용한 실시간 스트리밍 기능 구현

동물 상세 페이지 구현

샵 상세 페이지 구현

UX 설계

프론트엔드 개발환경

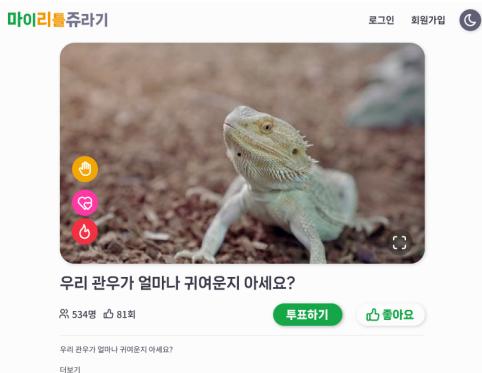
언어 TypeScript

프레임워크 React

IDE VSCode

Projects (3/4)

WebRTC



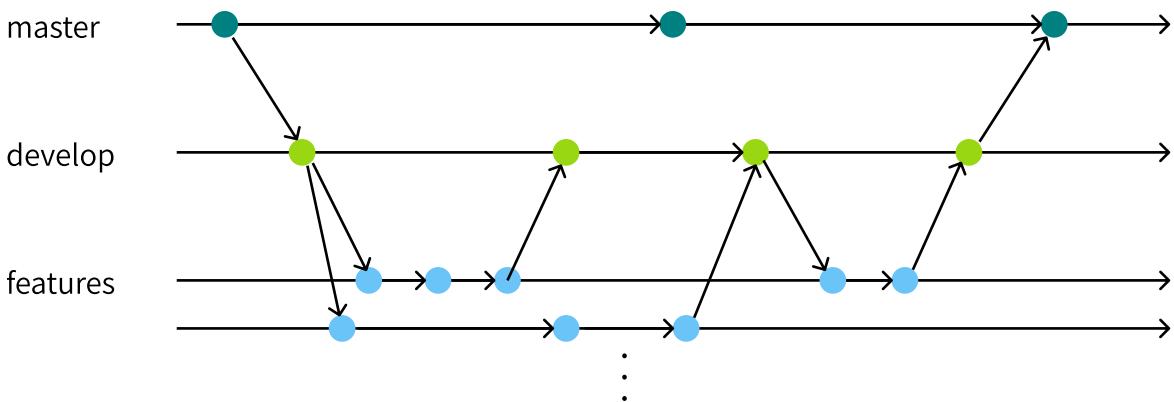
WebRTC를 이용한 실시간 스트리밍 기능을 맡았습니다.

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.

Projects (4/4)



Kibitz bugs

2023.08.15 ~ 지속 개발 중

1인 방송 플랫폼에서 사용 가능한 스트리머와 시청자간 오목 대결 서비스입니다.

백엔드 개발, 인프라 구성 업무를 맡았습니다.

<https://github.com/minu-j/kibitz-bugs>

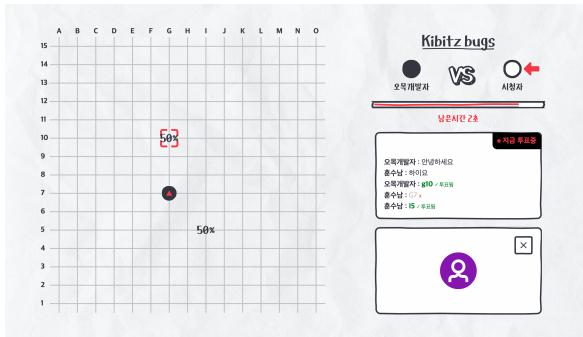
주요 기능



- Twitch 로그인을 이용해 로그인할 수 있습니다.



- 스트리머와 시청자의 돌을 선택합니다.
- 각각 시간을 다르게 부여할 수 있습니다.



- 스트리머는 마우스를 이용해 돌을 놓습니다.
- 시청자는 방송 플랫폼에 댓글을 작성함으로써 투표를 합니다.
- 가장 많은 투표를 얻은 자리가 시청자의 수가 됩니다.

Projects (4/4)

담당 역할

오목 알고리즘 구현

로그인 알림 기능 구현

OAuth2.0기반 트위치 회원 인증 개발

인프라 구성

UX 설계

백엔드 개발환경

언어 JAVA 11

프레임워크 Spring Boot 2.7.10

DB MariaDB

IDE IntelliJ

인프라 구성

서버 AWS EC2

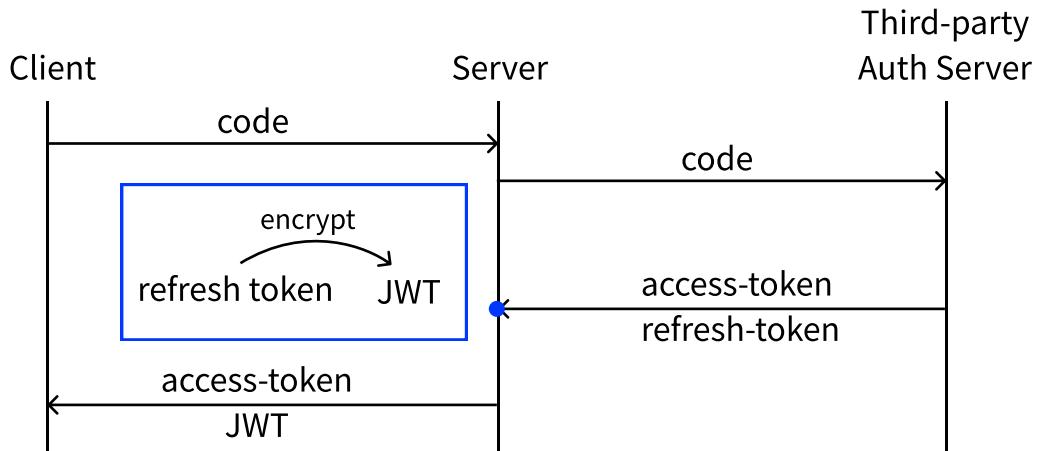
사양 vCPU 1 Core, 메모리 1GB

소스 Github

Projects (4/4)

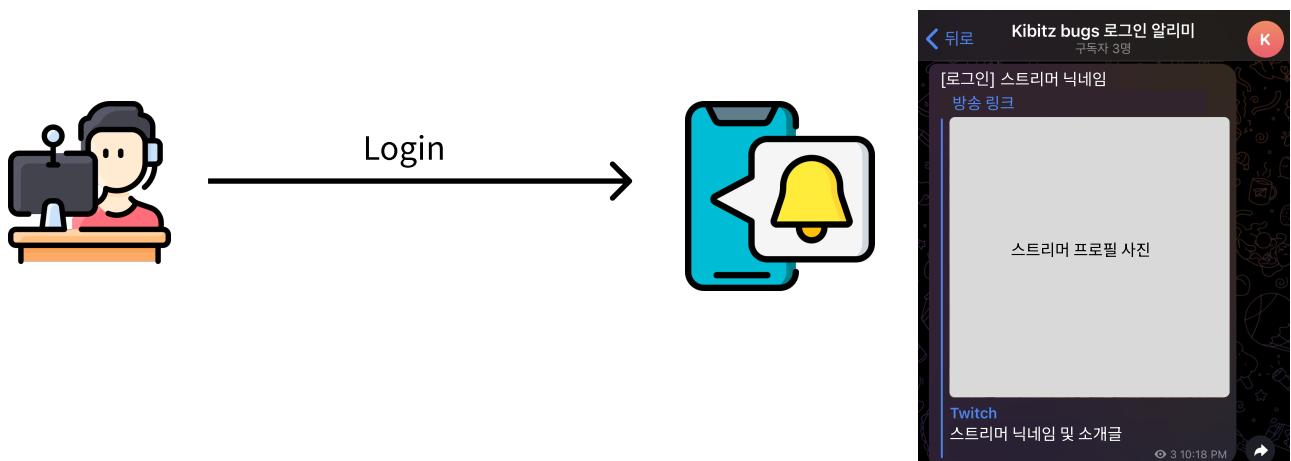
유저 인증

Authorization Code Flow 방식으로 유저를 인증해 보안을 강화했습니다.



로그인 알림 기능

스트리머 로그인 시 알림이 오도록 설정해, 방송을 시청하며 얻은 피드백으로 지속 개선하였습니다.



최대 19,000명의 동시 시청자 수 기록

영리를 목적으로 하는 서비스가 아니었기 때문에, 낮은 성능의 인프라 환경을 구성해야 했습니다. 성능 최적화를 위한 다양한 고민 후, Jmeter 및 소켓 부하 테스트를 거친 뒤 배포하였습니다. 현재 약 2,000명의 사용자 및 최대 19,000명의 동시 시청자 수를 기록하였습니다.