ABOUT ME

안녕하세요. Node.js 백엔드 개발자 이성현입니다.

저는 실시간 멀티플레이어 퀴즈 플랫폼을 1인 개발하였고, 서비스 운영 중 발견한 성능 문제들을 분석하고 개선하며 백엔드 개발의 즐거움을 알았습니다. 나아가 더 깊이 있는 기술적 도전을 경험하고 싶습니다.

CONTANCT

Email seonghyeon2528@gmail.com
Github https://github.com/tjdgus4546

EDUCATION

2017.02~2023.02	시각디자인과
백석대학교	3.06 / 4.5

SKILLS

Backend

- Node.js (Express)
- Javascript (ES6+)
- MongoDB
- AWS EC2, AWS S3

Frontend

- HTML5, CSS3, Javascript (ES6+)
- TailwindCSS

Tools

- Git, Github
- VSCode

AWARDS

2024-08~2024-09

2024 연구데이터 분석활용 경진대회

과학기술정보통신부

연구 데이터를 활용해 분석 활용 사례를 만드는 경진대회로 저희 팀은 논문을 분석하여 가독성 있게 알려주는 LLM을 제작하여 **우수상**을 수상했습니다.
저는 팀원이 개발한 모델을 웹 인터페이스에서 사용 가능하도록 실시간 채팅 기반 웹 서비스를 구현했습니다.

PROJECT

PLAYCODE.GG - 실시간 멀티플레이어 퀴즈 플랫폼

프로젝트 개요

- 개발기간: 2025.05 ~ 2025.11
- 1인 풀스택 개발 (기획/개발/배포/운영)
- 기술 스택: Node.js, Express.js, MongoDB, Redis, Socket.IO, Javascript, AWS EC2, AWS S3

프로젝트를 시작한 배경 및 목표

평소 친구들과 퀴즈 사이트에서 퀴즈를 풀며 경쟁하는 것을 즐겼지만, 기존 퀴즈 플랫폼들은 혼자서 문제를 푸는 형태가 대부분이었고 실시간으로 여러 명이 함께 경쟁하며 소통할 수 있는 서비스를 찾기 어려웠습니다.

이러한 문제를 해결하기 위해 최대 12명이 동시에 참여하고, 실시간 채팅으로 소통하며, 즉각적으로 순위를 확인할 수 있는 멀티플레이어 퀴즈 플랫폼을 개발했습니다.

특히 실시간 동기화, 동시성 제어, 서버 확장성 등 멀티플레이어 서비스 특유의 기술적 문제들을 직접 경험하고 해결하고 싶었습니다.

주요 기능

1. 실시간 멀티플레이어 퀴즈 세션

- 최대 12명 동시 참여
- Socket.IO 기반 실시간 동기화
- Redis 원자적 연산으로 퀴즈 경쟁성 구현
- 실시간 점수 및 순위 업데이트
- 게임 종료 시 퍼센타일 랭킹 표시

2. 다양한 문제 유형의 퀴즈 제작 지원

- 주관식, 객관식, 이미지, 영상, 오디오유형 지원
- AWS S3 기반 이미지 저장
- YouTube IFrame API (특정 구간 재생)
- 문제별 자동 저장 (작업 손실 방지)

3. 낮은 진입 장벽

- 게스트 모드 (비회원 즉시 플레이)
- OAuth 2.0 소셜 로그인 (Google, Naver)
- 초대 코드로 간편한 방 참여

4. 실시간 채팅 & 커뮤니티

- 게임 중 플레이어 간 실시간 소통
- 댓글 및 추천 시스템
- 신고 기능 및 관리자 대시보드

핵심 성과

1. 확장성 10배 개선

문제

서버에서 모든 세션의 타이머를 관리 ⇒ 세션 1000개 이상 생성시 메모리 50mb, cpu 5~10% 사용.
확장성 한계 도달.

해결

타이머 관리를 클라이언트로 이동.
서버는 제한시간과 시작 시간만 전송하고, 클라이언트가 직접 카운트다운 및 종료 처리.

결과

서버 메모리, CPU 사용률 99% 감소
동시 처리 가능 세션 1000개 → 10000개+ (10배)
제접속 시에도 정확한 게임 상태 복원

2. 로딩속도 120배 개선

문제

이미지를 base64로 인코딩하여 MongoDB에 저장.
퀴즈 목록 로딩 시 2분이상 시간 소요.

해결

AWS S3로 이미지 저장 방식 전환.
DB에는 S3 키만 저장하고, Presigned URL로 접근.

결과

로딩시간 2분 → 1초 이하로 120배 개선.
API 응답 크기 15mb → 100KB 99% 감소.
MongoDB 문서 크기 1MB → 1KB 99% 감소.

3. 동시성 제어

문제

여러 플레이어가 동시 정답 제출 시 Race Condition 이슈 발생. 첫 정답자 보너스 점수를 2명이상이 받는 버그 발생.

해결

Redis의 SET NX 원자적 연상으로 첫 정답자 판정.
키가 없을 때만 설정되는 방식으로 동시성 제어.

결과

Race Condition 100% 해결.
밀리초 단위 동시 제출에도 정확한 판정.
클러스터 모드에서도 정상 작동.

배운점

1. 부하 분산의 중요성

서버가 모든 것을 처리할 필요는 없다.
클라이언트가 할 수 있는 것은 클라이언트에게 맡기면 서버 부담을 크게 줄일 수 있다.

2. 확장성을 고려한 초기 설계

초기부터 확장성을 고려했다면 리팩토링 비용을 줄일 수 있었다. "일단 돌아가게"보다 "나중에도 견딜 수 있게" 설계하는 게 중요.

3. 트레이트오프 이해

클라이언트 측 타이머는 조작 가능성이 있지만, 서버 측 검증으로 보완 가능. 완벽한 해결책은 없고 상황에 맞는 최선의 선택이 있을 뿐.

4. 동시성 문제는 생각보다 복잡

단순히 if 문으로 체크하는 게 안전하다고 생각했지만, 실제로는 Race Condition이 발생. 멀티플레이어 서비스에서 동시성은 항상 고려해야 할 핵심 문제.

5. Edge Case 대비

재접속, 네트워크 지연, 시간 조작 등 다양한 상황을 고려해야 완성도 높은 서비스를 만들 수 있다.

OTHERS

사이트 주소 https://playcode.gg