PORTFOLIO

끊임없이 새로워지는 기술과 환경,그 안에서 빠르게 적응하고 성과를 내는 유연한 김세헌입니다

KIM, SEHEON

#FULL STACK

kshandy0221@gmail.com
010-3738-5871

김세헌

FULL STACK

생년월일 : 2001. 02. 21

주소 : 서울 강서구 양천로 62길 41

이메일 : kshandy0221@gmail.com

전화번호 : 010-3738-5871

GitHub: https://github.com/NewOld21

Velog: https://velog.io/@shk0221/posts

학력

2020. 03 ~ 2025. 02 성결대학교 컴퓨터공학과 졸업

2017. 03 ~ 2020. 02 마포고등학교 졸업

수상 및 자격

정보처리기사 필기 합격 (2024.07)

국가우수이공계 장학생 (2023.05)

스킬 – USE WELL

- Backend: Python, TypeScript, Node.js, Nest.js
- Frontend: React, HTML, CSS
- Database: PostgreSQL
- Tooling: GitHub, Slack, Notion

스킬 – HAVE USE

- Backend: JavaScript, C, Java, Spring
- Frontend: Tailwind CSS
- Database: MySQL, MongoDB
- Tooling: GitHub, Figma, Postman

프로젝트 – WaveFlow

- Backend: Python, TypeScript, Node.js, Nest.js
- Frontend: React, HTML, CSS
- Database: PostgreSQL
- Tooling: GitHub, Slack, Notion

프로젝트 – 마이맛

- Backend: Spring Boot, MyBatis, Java, Maven
- Frontend: HTML, CSS, JavaScript, Thymeleaf
- Database: Oracle Database

프로젝트 : WaveFlow

프로젝트 정보

- 주소 : https://waveflow.pro
- 기간 : 1개월 [2025.07]
- 팀원 : 5명
- 사용기술 : Nest.js, React, PostgreSQL, TypeScript, Tailwind CSS, Python 등
- GItHub : https://github.com/Team-Honey-Badgers

배경 및 필요성

- 음악 작업자들은 버전 관리를 찐막, 최최종 같은 파일명으로 대체하고 있음
- 협업 시 카카오톡, 이메일, 구글 드라이브 등 비효율적인 방식에 의존
- 개발자들의 Git처럼, 음악 협업에 특화된 형상 관리 도구의 부재를 해결하고자 함

개요

- 멀티트랙 기반 실시간 협업 오디오 편집 플랫폼
- Stage → Stem → Version 구조로 버전 관리 및 리뷰 워크플로우 지원
- 실시간 알림, 댓글, 오디오 비교 기능까지 통합한 종합 오디오 협업 환경 제공

주요기능

- 멀티트랙 오디오 스트리밍 및 재생 기능 제공
- 실시간 협업자 초대 및 권한 관리 시스템 구현
- 단계별 리뷰 워크플로우 및 버전 관리 지원
- 실시간 알림, 댓글, PR 기반 오디오 비교 및 승인 기능 제공

역할

Backend 개발

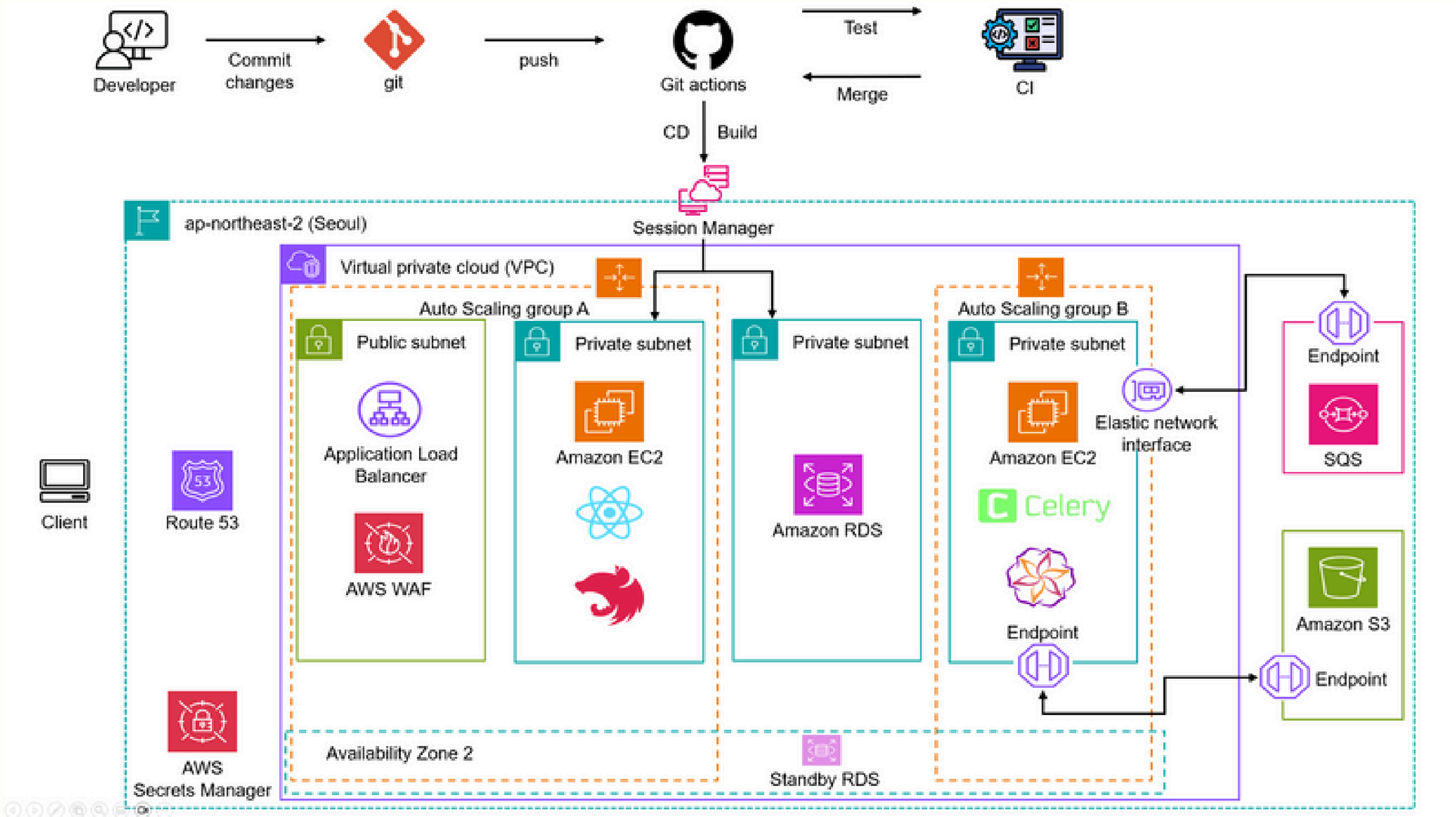
- NestJS 기반 RESTful API 설계 및 구현
- 데이터베이스 설계
- JWT 인증 및 Google OAuth 연동
- AWS S3 파일 업로드, 이메일 발송 시스템

Frontend 개발

- React 기반 음악 협업 플랫폼 UI 구현(DashBoard,Track)
- WebSocket 실시간 협업 기능
- 사용자 인터페이스 및 상태 관리
- 접근 권한 가드

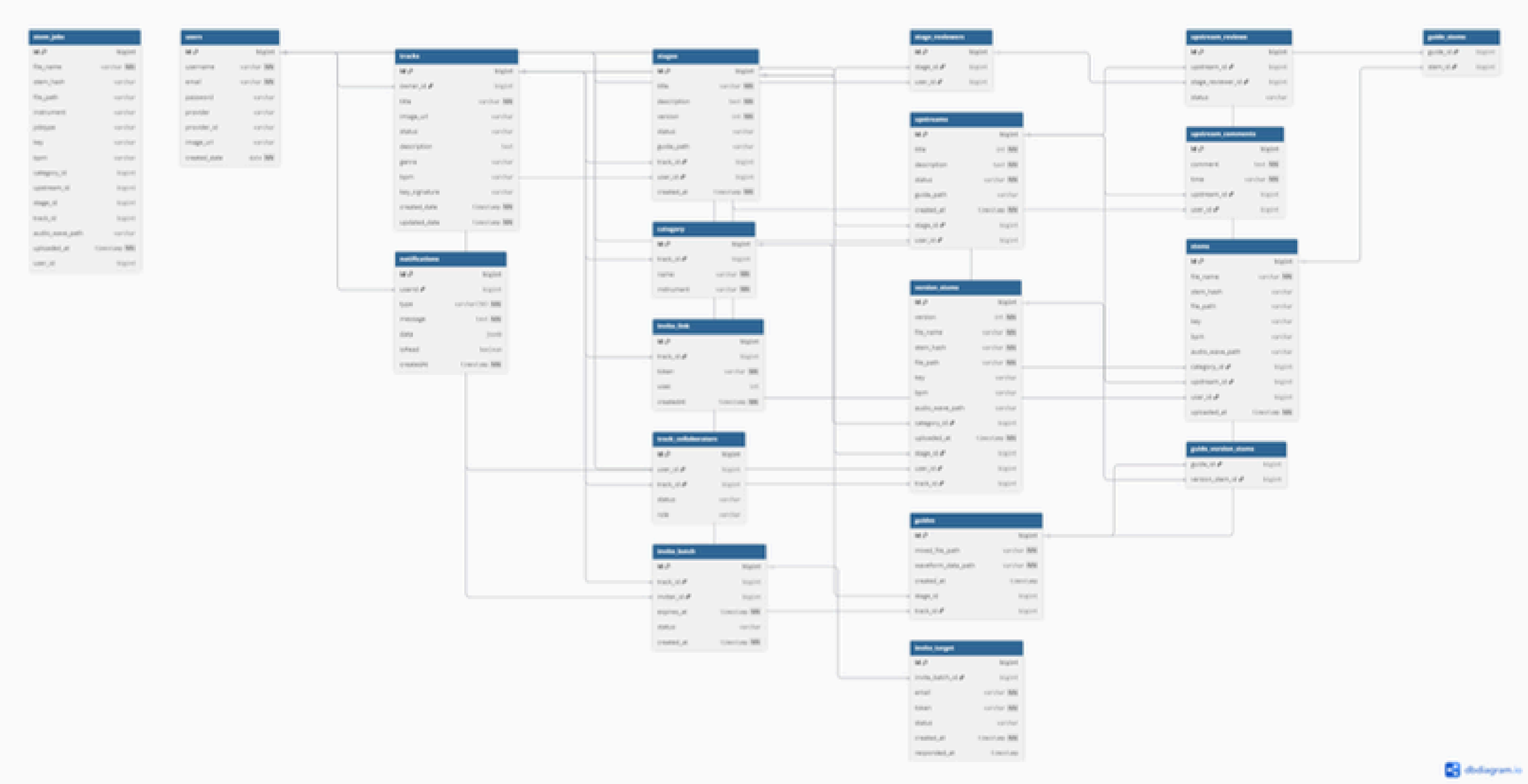
아키텍처 및 ERD

시스템 아키텍처



아키텍처 및 ERD

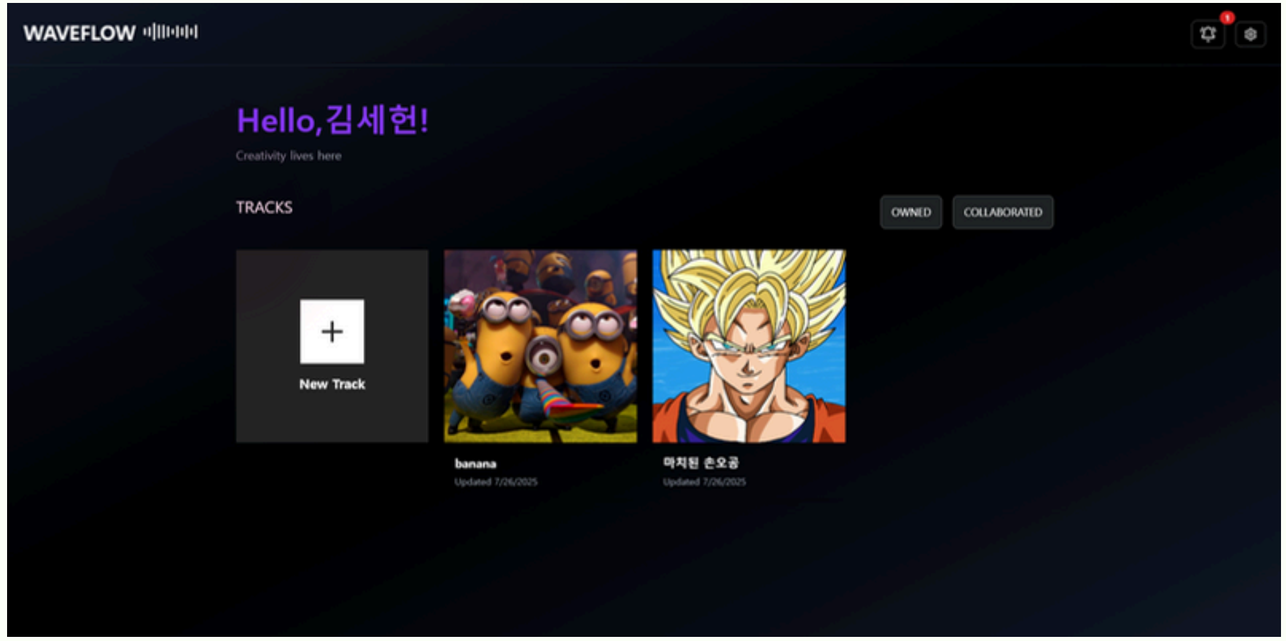
ERD



자세한 ERD 내용 : https://velog.io/@shk0221/WaveFlow-ERD

구현 기능 및 개발과정

대시보드 페이지



중점 구현 기능

JWT 기반 인증/인가 시스템
실시간 알림 시스템
페이지별 접근 권한 검증

대시보드 페이지

- 트랙 CRUD 시스템 (생성/조회/수정/삭제)
- 멀티파트 파일 업로드를 통한 Stem 파일 관리
- 반응형 트랙 카드 UI 및 전체 페이지 레이아웃

개발과정

보안: JWT 토큰 기반 stateless 인증, 가드 패턴으로 라우트 보호
실시간성: WebSocket + JWT 검증으로 안전한 실시간 통신
권한 관리: 트랙별 협업자 권한 체계 구현

대시보드 페이지
Backend

- NestJS로 Track/Stem 엔티티 설계, AWS S3 Presigned URL 기반 대용량 파일 업로드 API 구현

Frontend

- React로 드래그앤드롭 파일 업로드, 실시간 업로드 진행률 표시, 트랙 관리 인터페이스 구현

통합

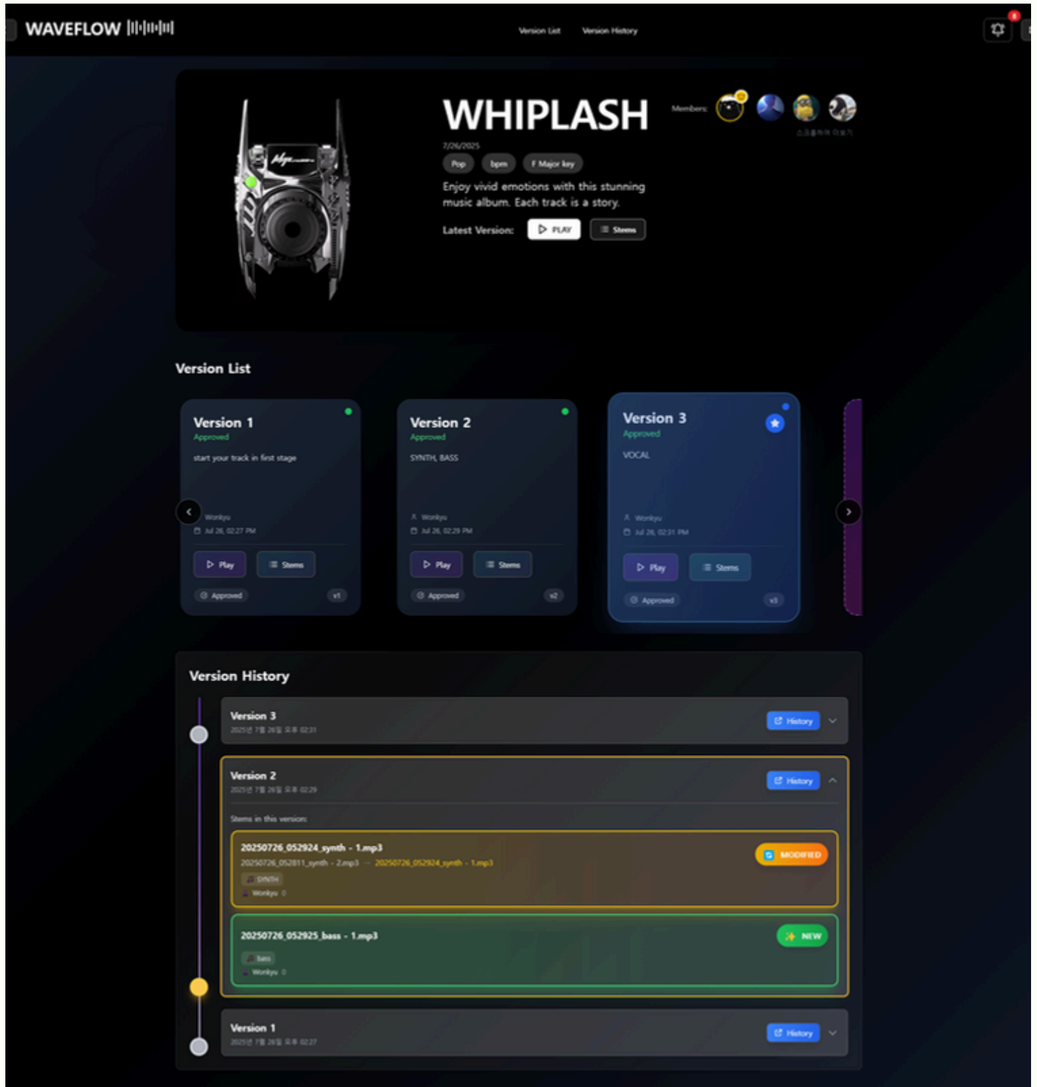
- RESTful API와 프론트엔드 상태 관리를 연동하여 실시간 트랙 목록 업데이트

기술 스택 선정 이유

- AWS S3: 대용량 오디오 파일의 안정적 저장 및 CDN 활용
- Presigned URL: 서버 부하 없이 클라이언트에서 직접 업로드
- JWT: 확장 가능한 stateless 인증
- WebSocket: 실시간 협업 환경 구축

구현 기능 및 개발과정

트랙 페이지



중점 구현 기능

트랙 페이지

- 트랙 정보 시스템: 트랙 메타데이터 표시 및 실시간 업데이트
- 버전 관리 시스템: 전체 버전 리스트 조회 및 타임라인 형태 히스토리
- 원클릭 롤백: 이전 버전으로 즉시 복원 기능
- Stem 파일 관리: 버전별 개별 Stem 파일 모달 뷰어 및 재생
- 통합 오디오 플레이어: 멀티트랙 동시 재생 및 개별 제어
- Stage 생성 시스템: 리뷰 단계 생성 및 워크플로우 관리

개발과정

트랙 페이지

Backend

- 데이터베이스 설계: Track-Version-Stem 관계형 모델 구현, 버전 히스토리 추적을 위한 스키마 최적화
- API 엔드포인트: 트랙 상세 조회, 버전 관리, 롤백, Stem 파일 처리 등 15+ 엔드포인트 구현
- 버전 관리 로직: 트랙 상태 변경 시 자동 버전 생성, 롤백 시 데이터 무결성 보장
- Stage 시스템: 리뷰 워크플로우를 위한 Stage 엔티티 및 비즈니스 로직 구현

Frontend

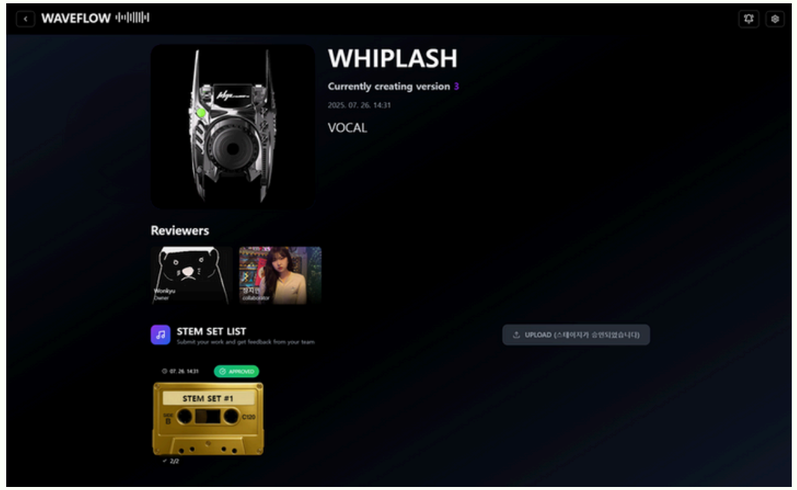
- 페이지 레이아웃: 트랙 정보, 버전 리스트, 히스토리를 직관적으로 배치한 반응형 UI 설계
- 버전 히스토리: 타임라인 형태의 시각적 버전 추적, 각 버전별 변경사항 표시
- 모달 시스템: Stem 파일 상세 보기 모달의 데이터 연동 및 API 통합 (디자인 제외)
- 인터랙션: 드래그앤드롭, 원클릭 롤백, 실시간 상태 업데이트

데이터 연동 및 상태 관리

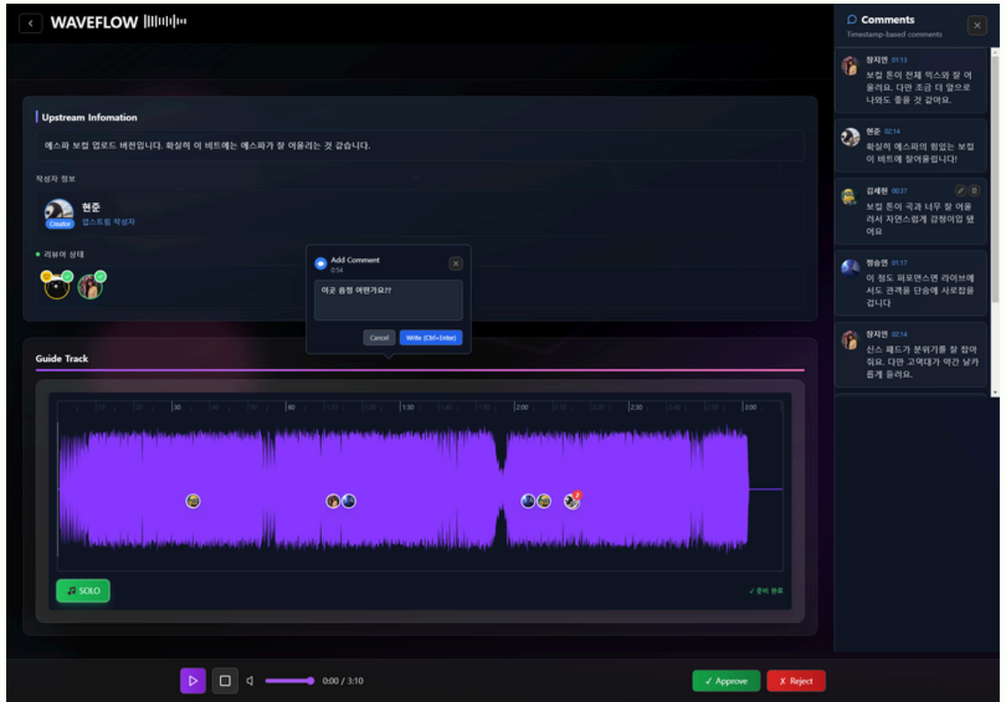
- API 통합: 모든 트랙 관련 API 엔드포인트와 프론트엔드 연결
- 실시간 업데이트: 버전 변경, 롤백 시 즉시 UI 반영
- 에러 핸들링: 파일 업로드 실패, 네트워크 오류 등 예외 상황 처리

구현 기능 및 개발과정

스테이지 페이지



스템세트 리뷰 페이지



중점 구현 기능

스테이지 페이지

- Stem 파일 업로드 시스템
- 트랙/스테이지 정보 조회 API
- Stem-set 데이터 관리

스템세트 리뷰 페이지

- 리뷰 상태 관리, 승인/거부 처리
- 댓글 시스템
- Upstream 데이터 조회
- 리뷰 버튼 연동 및 데이터 페칭

개발과정

스테이지 페이지

Backend

- 파일 업로드 : AWS S3 기반 Stem 파일 업로드 엔드포인트 구현
- 데이터 조회 : Stage 상세 정보 및 연관된 Track 데이터 조회 API 개발
- Stem-set 관리 : 리뷰용 Stem 파일 그룹 관리 로직 구현

스템세트 리뷰 페이지

- 리뷰 시스템: Review 엔티티 설계, 승인/거부 상태 변경 로직 구현
- 댓글 기능: 리뷰별 댓글 CRUD API 개발
- 데이터 연동: Upstream 데이터 조회 및 리뷰 상태 실시간 업데이트
- Frontend 연동: 리뷰 버튼 클릭 시 API 호출 및 상태 반영

트러블 슈팅

분석 서버 부하 분산을 위한 개별 파일 완료 처리

상황

- 클라이언트가 여러 파일을 병렬 업로드한 뒤, 전체 업로드 완료 후 일괄 서버 알림
- 서버는 모든 파일을 한 번에 SQS로 분석 서버에 전달
- 분석 서버에 분석 요청이 몰리며 순간적인 부하 집중 발생

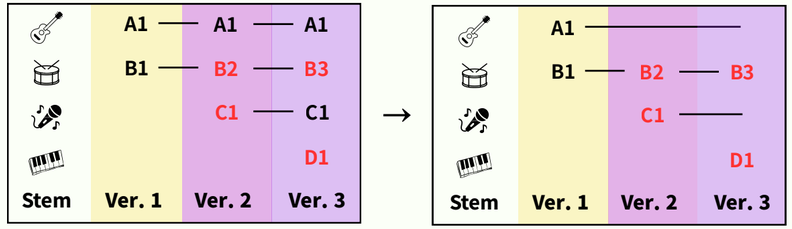
해결

- 개별 파일 완료 시점마다 서버에 알림 전송하도록 구조 변경
- 서버는 해당 파일을 즉시 SQS로 분석 서버에 전송
- 분석 요청이 시간차를 두고 분산
- 그 결과, 분석 서버 부하 완화처리와 속도 및 안정성 향상

해당 경험을 통해 알게된 점

- 병렬 구조에서도 이벤트 타이밍 조정만으로 부하를 효과적으로 분산할 수 있음을 경험
- 단순한 기능 구현보다, 흐름을 제어하는 설계가 시스템 성능에 더 큰 영향을 미침
- 실시간 처리와 안정성을 동시에 고려한 구조가 서비스 운영 품질에 직결된다는 점을 체감

증분 스냅샷 도입



상황

- 초기에는 Stage마다 전체 Stem 파일을 스냅샷 형태로 복제 저장하고 있었음
- 동일한 Stem 파일이 중복 보관되며 스토리지 낭비가 발생함
- 롤백 시 전체 스냅샷을 일괄 복원해야 했기에 복구 시간이 지연되는 문제가 있었음

해결

- Stage마다 수정되거나 추가된 Stem만 별도로 저장하는 증분 스냅샷 구조로 변경
- 변경 없는 파일은 이전 Stage의 레퍼런스를 참조하도록 설계
- 롤백 시 Cascade Delete를 활용해 이후 Stage만 삭제하고, 증분된 스냅샷만 복원
- 그 결과, 버전별 저장 용량 약 60% 감소, 복구 속도 약 2배 개선

해당 경험을 통해 알게된 점

- 기능 구현 외에도 스토리지 효율성과 복구 성능까지 고려한 구조 설계의 중요성을 체감
- 전체 복제보다 증분 저장이 더 나은 선택이 될 수 있다는 아키텍처적 인사이트를 얻게 됨
- 실제 사용 흐름(롤백, 병합 등)을 가정한 설계가 현실적인 문제 해결로 직결됨
- 기술적 개선이 사용자 경험과 운영 효율 모두에 긍정적 영향을 준다는 점을 직접 경험

프로젝트 성과 및 회고

프로젝트 성과

항목	내용
실시간 음악 협업 플랫폼	멀티트랙 오디오 스트리밍과 버전 관리 시스템 구현
RESTful API 및 DB 모델링	30개 이상의 API 구현과 Track-Version-Stem-Review 구조 설계
분석 서버 부하 분산	파일 단위로 SQS 분산 전송하여 서버 부하 분산 및 처리 지연 최소화
실시간 통신 시스템 구축	JWT 인증 기반 WebSocket으로 알림 및 협업 기능 실시간 처리

프로젝트 회고

아쉬운 점	개선 방향
파일 검증 로직 미흡	파일 형식/크기 제한, 바이러스 스캔, S3 버킷 CORS 및 접근 정책 강화
DB 조회 시 N+1 문제와 인덱스 부족으로 성능 저하 발생	인덱스 최적화, Eager Loading, Redis 기반 캐싱 적용
모니터링 및 이상 탐지 미흡	응답 시간/쿼리 로깅, 비정상 API 호출 감지, 보안 이벤트 알림 시스템 구축

프로젝트 : 마이맛

프로젝트 정보

- 기간 : 3개월 [2022.09 ~ 2022.11]
- 팀원 : 1명
- 사용기술 : Spring Boot, MyBatis, Java, CSS, Thymeleaf, Oracle Database 등
- GItHub : https://github.com/NewOld21/Mymas

배경 및 필요성

- 개인이 방문한 맛집 정보의 체계적 관리 부족
- 맛집 정보의 개인화 및 맞춤형 서비스 부재
- 사용자 간 맛집 정보 공유의 효율성 개선

개요

- 개인이 방문한 맛집들을 체계적으로 관리
- 다른 사용자들과 공유할 수 있는 웹 서비스
- 기존의 맛집 정보 서비스들이 단순한 리뷰 나열에 그치는 한계를 극복
- 개인화된 맛집 지도를 통해 더 의미 있는 맛집 정보 관리 경험을 제공

주요기능

- 사용자별 맛집 등록 및 리뷰 관리
- 실시간 지도 시각화로 위치 기반 맛집 탐색
- 리뷰 등록 시 맛집 통계 자동 업데이트
- 다른 사용자 맛집 지도 조회 기능

역할

Backend 개발

- Spring Boot 계층형 아키텍처 설계
- Oracle DB 설계 및 MyBatis 쿼리 최적화
- HttpSession 기반 인증/권한 처리
- 맛집 통계 자동 업데이트 로직 구현

Frontend 개발

- Thymeleaf 템플릿 및 반응형 UI 구현
- jQuery 기반 클라이언트 로직 개발
- Kakao Map API 연동 및 마커 처리
- 페이징, 검색, 에러 처리 구현

구현 기능 및 개발과정

메인 페이지

중점 구현 기능

- 세션 기반 상태 관리: 로그인 상태에 따른 동적 네비게이션 메뉴 제공
- 서비스 허브: 주요 기능(검색, 마이맵, 아더맵, 마이페이지)으로의 원클릭 접근
- 반응형 네비게이션: 모바일/데스크톱 최적화된 메뉴 구조
- 중복 알림 방지: 브라우저 히스토리로 페이지 진입 방식 구분

개발과정

메인 페이지

Backend:

- 최소한의 서버 로직 : 메인 페이지는 정적 콘텐츠 중심으로 설계하여 서버 부하를 최소화하고 빠른 로딩 속도 확보
- 세션 상태 전달 : HttpSession에서 사용자 ID를 추출하여 템플릿 엔진에 전달하는 단순하고 안정적인 구조

Frontend

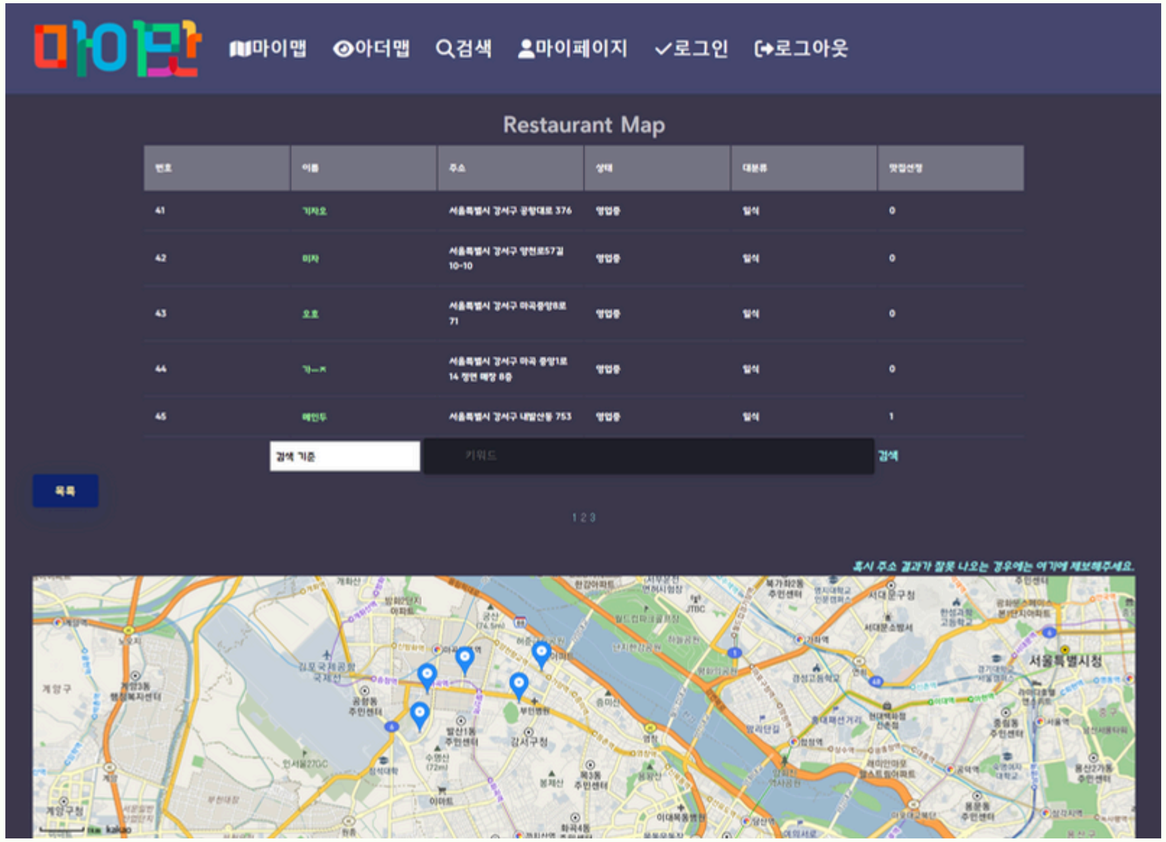
- CSS Grid 기반 레이아웃 : 3개 주요 기능을 균등 분할하여 사용자가 직관적으로 서비스를 탐색할 수 있도록 구성
- 조건부 렌더링 : 템플릿 엔진을 활용해 세션 상태에 따른 동적 URL 생성으로 개인화된 네비게이션 제공
- 상태 기반 피드백 : 브라우저 히스토리 객체를 활용하여 페이지 진입 방식을 구분하고 적절한 사용자 피드백 제공

기술 스택 선정 이유

- Spring Boot: 빠른 개발과 내장 톰캣으로 배포 간소화
- MyBatis: 복잡한 검색 쿼리와 동적 SQL 처리에 유리
- Oracle: 대용량 데이터 처리, ROWNUM 페이징 성능 우수
- Thymeleaf: Spring Boot 완벽 통합, SEO 최적화
- jQuery: 단순한 DOM 조작, 가벼운 라이브러리

구현 기능 및 개발과정

맛집 검색 페이지



중점 구현 기능

맛집 검색 페이지

- 다중 조건 검색 시스템: 맛집명과 주소 기반의 유연한 검색으로 사용자의 다양한 검색 의도에 대응
- 고성능 페이징 처리: 대용량 맛집 데이터를 효율적으로 처리하여 빠른 검색 결과 제공
- 실시간 지도 연동: 검색 결과를 지도상에 마커로 시각화하여 위치 기반 정보 직관적 제공
- 반응형 검색 인터페이스: 모바일 환경에서 핵심 정보만 표시하는 적응형 UI로 가독성 확보

개발과정

맛집 검색 페이지

Backend:

- 동적 쿼리 설계: 검색 타입에 따른 조건부 쿼리 생성으로 유연한 검색 기능 구현
- 데이터베이스 성능 최적화: Oracle 인덱스 힌트와 ROWNUM 기반 페이징으로 대용량 데이터 고속 처리
- 검색 조건 통합 관리: 페이징 정보와 검색 조건을 하나의 객체로 관리하여 코드 일관성 확보

Frontend :

- 분할 레이아웃 설계: 검색 결과와 지도를 좌우 분할하여 정보 밀도와 시각적 효과의 최적 균형점 확보
- 비동기 지도 처리: 주소를 좌표로 변환하는 외부 API 연동으로 실시간 지도 마커 생성
- 동적 마커 관리: 검색 결과 개수에 따른 분기 처리로 지도 성능 최적화와 사용자 경험 향상

데이터 연동 및 상태 관리

- 외부 API 통합: Kakao 지도 서비스와의 비동기 연동으로 주소 기반 위치 시각화 구현
- 에러 처리 전략: 지도 API 호출 실패 시 기본 위치로 폴백하여 서비스 연속성 보장

구현 기능 및 개발과정

개인 맛집 페이지

중점 구현 기능

개인 맛집 페이지

- 사용자별 데이터 완전 격리: 세션 기반 인증으로 개인 맛집 데이터만 조회하여 프라이버시 보장
- 완전한 CRUD 시스템: 개인 맛집 리뷰의 전체 생명주기 관리로 사용자 맞춤형 맛집 관리 경험 제공
- 다층 권한 검증: 클라이언트와 서버 양단에서 로그인 상태를 검증하여 보안 강화

개발과정

개인 맛집 페이지

Backend:

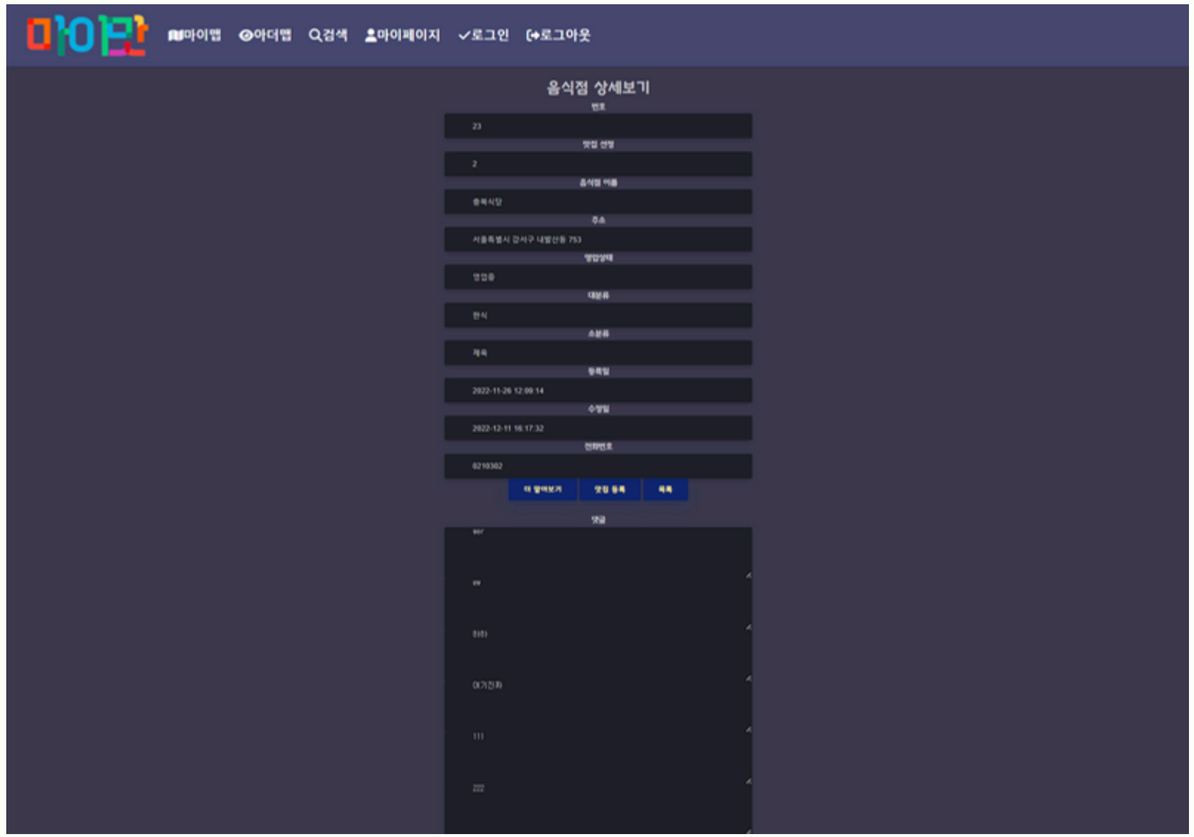
- 사용자 중심 객체 설계: 페이징 정보와 사용자 ID를 통합한 객체로 데이터 필터링과 페이징을 동시 처리
- 재사용 가능한 쿼리 구조: 사용자 ID 조건을 공통 프래그먼트로 분리하여 모든 개인 데이터 쿼리에 자동 적용
- 세션 기반 보안 설계: 컨트롤러에서 세션 검증 후 서비스 계층에 사용자 컨텍스트 전달하는 2단계 보안 구조

Frontend :

- 테이블 기반 관리 UI: 개인 맛집을 구조화된 테이블로 표시하여 정보 스캔과 관리 작업의 효율성 극대화
- 사전 권한 검증: 페이지 로드 시 즉시 로그인 상태를 확인하여 미인증 사용자에게 친화적 안내 제공
- 컨텍스트 기반 링크: 각 맛집 항목에서 상세 정보와 원본 데이터로의 다중 탐색 경로 제공

구현 기능 및 개발과정

맛집 상세 및 리뷰 페이지



중점 구현 기능

맛집 상세 및 리뷰 페이지

- 종합적 맛집 정보 제공: 기본 정보부터 영업 상태, 카테고리까지 완전한 맛집 데이터 표시
- 통합 리뷰 생태계: 기존 리뷰 조회와 신규 리뷰 등록을 하나의 매끄러운 사용자 플로우로 구성
- 실시간 인기도 반영: 리뷰 등록과 삭제 시 맛집 통계가 즉시 업데이트되어 신뢰성 있는 정보 제공
- 외부 서비스 확장: 네이버 지도 연동으로 상세 위치 정보와 추가 리뷰 접근성 제공

개발과정

맛집 상세 및 리뷰 페이지
Backend:

- 관계형 데이터 모델링: 맛집과 리뷰 테이블 간 외래키 관계로 데이터 무결성과 참조 일관성 확보
- 복합 트랜잭션 설계: 리뷰 등록 시 리뷰 데이터 삽입과 맛집 통계 업데이트를 원자적으로 처리
- 다중 데이터 조회 최적화: 맛집 상세 정보와 관련 리뷰 목록을 단일 요청으로 조회하여 네트워크 효율성 향상

Frontend :

- 정보 중심 레이아웃: 맛집 상세 정보를 구조화된 폼으로 표시하여 정보 가독성과 접근성 최적화
- 컨텍스트 기반 액션: 맛집 등록, 외부 정보 확인 등 사용자 의도에 맞는 다양한 액션 버튼 제공
- 즉시 피드백 시스템: 리뷰 등록 성공 시 즉각적인 알림으로 사용자 행동에 대한 명확한 결과 제공

트러블 슈팅

대용량 데이터 페이징 성능 최적화

상황

- 맛집 검색 시 페이징 처리 속도가 3-5초까지 지연되는 성능 저하 발생
- 후반 페이지로 갈수록 응답 시간이 기하급수적으로 증가
- 사용자 이탈률 증가로 서비스 품질 저하

해결

- 기존 LIMIT/OFFSET 방식에서 Oracle의 ROWNUM 기반 페이징으로 변경
- 인덱스 힌트(/*+ index_desc(tbl_restaurant PK_rest) */)를 활용하여 쿼리 실행 계획 최적화
- 동적 쿼리에서 검색 조건을 WHERE 절 앞쪽으로 배치하여 인덱스 활용도 향상

해당 경험을 통해 알게된 점

- 데이터베이스별 최적화 기법의 중요성
- Oracle의 ROWNUM과 인덱스 힌트를 활용하면 대용량 데이터에서도 일정한 성능을 유지
- 쿼리 실행 계획을 이해하고 제어하는 것이 성능 최적화의 핵심임을 학습

사용자별 데이터 격리 보안 이슈

상황

- 개인 맛집 관리에서 URL 파라미터 조작으로 다른 사용자 데이터 접근 가능
- 세션 검증만으로는 완전한 데이터 격리 불가능
- 개발자 실수로 인한 보안 취약점 발생 위험 상존

해결

- MyBatis <sql id="cooc">재사용 프래그먼트로 모든 개인 데이터 쿼리에 사용자 ID 조건 자동 적용
- Controller에서 세션 검증 후 Service Layer에서 2차 검증하는 다층 보안 구조 구현
- 클라이언트 사이드에서도 JavaScript로 권한 검증 추가하여 사용자 경험과 보안 동시 확보
- 모든 개인 데이터 관련 쿼리에 사용자 필터 강제 적용으로 개발자 실수 방지

해당 경험을 통해 알게된 점

- 다층 보안 방어가 단일 지점 보안보다 훨씬 효과적
- ORM/SQL 매퍼 레벨에서의 자동화된 데이터 격리가 개발자 실수를 원천 차단
- 사용자 경험과 보안성의 균형점을 찾는 것이 실무에서 중요

프로젝트 성과 및 회고

프로젝트 성과

항목	내용
전체 개발 프로세스 경험	기획부터 배포까지 웹 애플리케이션의 전체 라이프사이클 경험으로 풀스택 개발 역량 확보
계층형 아키텍처 설계	Spring Boot MVC 패턴과 4계층 구조를 통한 유지보수 가능한 코드 설계 경험
실무 수준 성능 최적화	대용량 데이터 처리와 쿼리 최적화를 통한 실질적 성능 개선 달성

프로젝트 회고

아쉬운 점	개선 방향
기본적인 보안 수준	Spring Security 도입으로 JWT 토큰 기반 인증 및 비밀번호 암호화 구현
모놀리식 구조	RESTful API 설계로 프론트엔드와 백엔드 완전 분리, JSON 기반 통신 전환
서버사이드 렌더링 한계	React/Vue.js 도입으로 SPA 구현 및 사용자 인터랙션 향상

CONTACT

kshandy0221@gmail.com
010-3738-5871

THANK YOU