1회차 프로젝트 발표 - 1들 (TEAM 4)

여행 계획 추천 플랫폼

TripHub

발표자 : 김동엽

목차

1. 프로젝트 소개

01

서비스 개발 배경 및 목적

- 여행 일정과 여행지 정보를 함께 공유할 수 있는 플랫폼 구축
- 방문 시간, 체류 시간, 예산, 메모 등 구체적인 여행 데이터 제공
- 지역별, 주제별 여행 계획/경험을 참고하여 더 나은 여행 루트 구성 지원
- 조건 기반 추천 챗봇을 통해 여행지 탐색 접근성 향상

02

주요 기능 소개

- 여행지 관리
  - CRUD
  - 이미지 추가 및 대표 이미지
  - 좋아요, 즐겨찾기, 조회수
  - 리뷰 CRUD
- 여행 일정 (게시글) 관리
  - CRUD
  - 좋아요, 즐겨찾기, 조회수
  - 리뷰 CR
- 검색 및 정렬
- 사용자 기능
- AI 챗봇

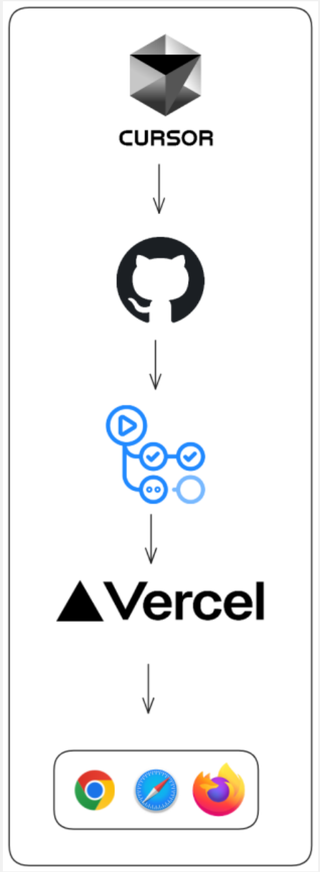
03

기술 스택 선정 이유

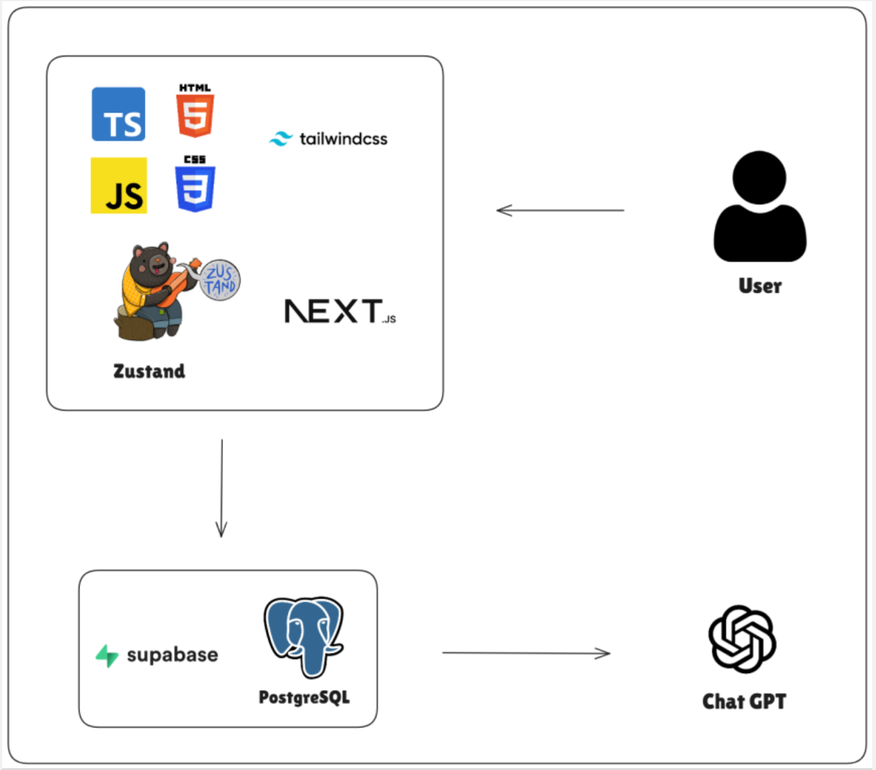
- Next.js: 빠른 렌더링(SSR), SEO 최적화, React 기반 코드 가능
- TypeScript: 안정적인 타입 지원
- Tailwind CSS: 효율적인 UI 스타일링
- Zustand: 가벼운 전역 상태 관리
- Prettier: 코드 스타일 통일
- Gitmoji: 커밋 메시지 가독성↑
- OpenAI GPT: 챗봇 구현용 API
- Supabase: 인증, DB, 스토리지 통합 제공
- Vercel: 빠른 배포 & CI/CD 연동
- Readdy.ai: AI 기반 UI 자동 생성

2. 시스템 아키텍처

개발 & 배포



서비스 구성



3. 데이터 모델 (ERD)

01

ERD 설계 개요 - 핵심 테이블 및 구조

- 여행지 정보와 사용자 활동 데이터 중심 추천 기능 설계
- 사용자를 중심으로 여행지와 게시글의 관계를 관리
- 주요 테이블
  - user - 가입 사용자 정보
  - places - 여행지 정보 (비용, 시간, 카테고리 등)
  - posts - 여행 일정 및 후기 게시글
  - post_places - 게시글에 포함된 여행지
  - reviews, likes, favorites, view_logs - 활동 기록

02

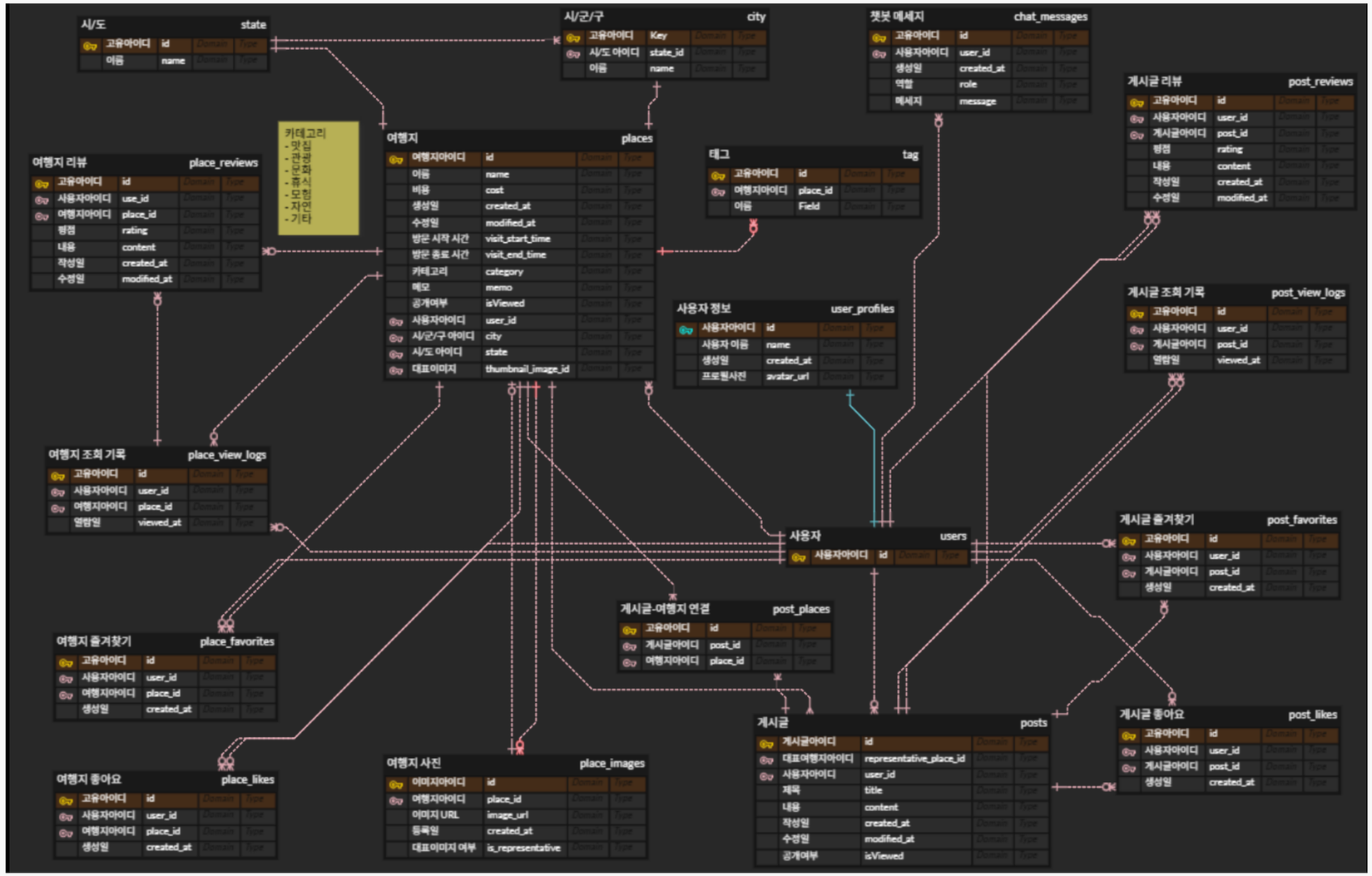
데이터 흐름 및 기능 연결 구조

1. 여행지 등록 및 활용
- 등록 : places + place_images
- 대표 이미지: thumbnail_image_id (FK)

2. 게시글 작성 및 연결
- 게시글: posts
- 포함된 여행지: post_places (1 : N 관계)
- 대표 여행지 썸네일 : representative_place_id

3. 사용자 상호작용
- 여행지 좋아요: place_likes
- 여행지 즐겨찾기: place_favorites
- 여행지 리뷰: place_reviews
- # 게시글 동일

4. 위치 정보 연결
- 시/도: state, 시/군/구: city



👉 [ERD Cloud] : https://www.erdcloud.com/d/JPNZrTTS5buYdTdSr

4. 프로젝트 시연 – 실제 동작 예시

주요 기능 시연

- 여행지 / 게시글 등록 및 수정 흐름
- 좋아요 / 즐겨찾기 / 리뷰 / 임시저장 기능
- 검색 , 필터, 정렬 조합
- 반응형 UI/UX 구현 사례

사용자 여정 시나리오

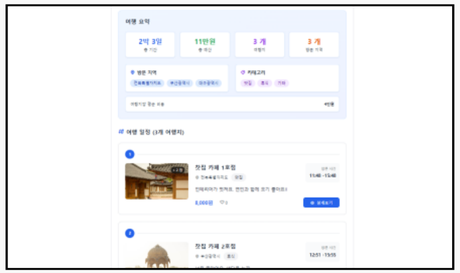
- 회원가입부터 여행 일정 생성까지
- 여행지 탐색 및 북마크 기능
- 일정 공유 기능
- 챗봇을 이용한 여행지 추천

실시간 데이터 처리

- 여행지 / 게시글 좋아요, 즐겨찾기 조회수 실시간 반영
- Supabase의 실시간 데이터 연동 기능 사용
- 상태 변화에 따른 화면 UI 업데이트

4. 프로젝트 시연 – 경쟁 서비스와의 차별점

01



일정 중심 구조

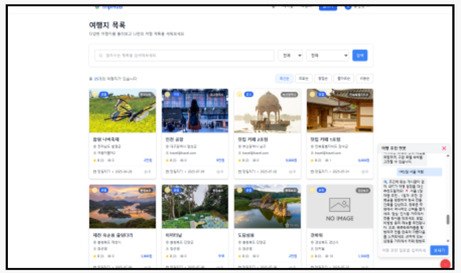
- 게시글을 여행 일정 단위로 구성
- 여행지 여러 개를 하나의 일정에 연결
- 여행 순서까지 저장 가능한 구조
- 일정을 시각적으로 구성하고 공유 가능

02

여행지 상세 데이터

- 방문 시간, 예산, 메모 등 세부 정보 입력 가능
- 여러 장의 사진과 대표 이미지 설정 가능
- 장소 외에 여행 맥락이 함께 저장됨
- 여행지 비교 및 계획에 실용적인 정보 제공

03



데이터 기반 탐색

- 지역, 카테고리, 평점 등 필터 기능 제공
- 예산, 체류 시간 기준 정렬 지원
- 조건 기반 GPT 챗봇 추천 연동
- 사용자 맞춤 여행지 탐색 가능

04

사용자 활동 기록

- 사용자 프로필 정보 별도 저장
- 여행지/게시글별 좋아요, 즐겨찾기, 조회수 기록
- 리뷰와 별점도 독립적으로 관리
- 향후 개인화 추천에 활용 가능

5. 개발 과정 및 트러블 슈팅 – 개발 규칙 및 컨벤션

01

Git & 협업 규칙

- GitHub Flow 전략 사용 (이슈 생성 → 브랜치 → PR → 코드리뷰 → 머지)
- 기능 단위 브랜치 명명 (feature/, fix/, refactor/ 등)
- 팀원 간 PR 리뷰 필수, Merge 전 승인 절차 거침

02

커밋 컨벤션

- Gitmoji 사용으로 변경 내역 직관적으로 구분
- 커밋 메시지 예시: Feat: 여행지 등록 API 연결
- PR 템플릿을 통해 변경 요약·이슈 번호 자동 연결

03

코드 스타일 통일

- Prettier, ESLint 설정으로 팀 내 코드 포맷 자동화
- TypeScript로 정적 타입 지정 → 코드 안정성 향상
- 중복 로직 제거를 위한 컴포넌트/함수 분리 노력

04

기술 도입 기준

- Zustand: 학습 곡선이 낮고, 팀 내 익숙한 멤버 존재
- Tailwind CSS: 빠른 UI 구성 및 유지보수 편의성
- Supabase: 인증 + DB + API 통합으로 빠른 개발 가능
- date-fns: 날짜 계산이 직관적이고, 동적으로 다루기 쉬움
- nanoid: uuid보다 가볍고, 작은 프로젝트에 적합한 ID 생성기

5. 개발 과정 및 트러블 슈팅 – 주요 이슈 및 해결 사례

01

브랜치 네이밍 규칙 변경

- 문제: ID-1, ID-2 형식의 브랜치 명이 직관적이지 않고 추적이 어려움
- 해결: feature/기능명 형태로 변경해서 협업 시 이해도 향상

02

Supabase auth.user 외래키 문제

- 문제: Supabase의 auth.user와 연결된 FK 관계에서 직접 쿼리 시 user 정보를 불러올 수 없었음
- 해결: user_profiles 테이블을 별도로 구성해 계정 정보 연결

03

useCallback 의존성 누락 이슈

- 문제: API 호출 함수에서 user를 사용했지만 useCallback의 의존성 배열에 포함하지 않아 파라미터 누락 발생
- 해결: useCallback의 의존성 배열에 [user] 명시해 정상 처리

04

조회수 중복 증가 문제

- 문제: 게시글을 한 번 조회했는데 조회수가 2~3번씩 증가
- 원인: React StrictMode로 인해 useEffect가 두 번 실행됨
- 해결: 배포 환경에서는 문제 없이 정상 동작 확인

6. 결론 및 마무리

01

개발 과정에서의 주요 도전과제

- 초기에 기획 방향을 명확히 정리
- 용어 통일로 팀 내 소통을 원활하게 정리
- 실사용을 고려한 구조 설계
(일정 중심, 관계형 DB 구조 등)
- Supabase 인증
- 조건 기반 챗봇 구현과 GPT 연동 처리

02

향후 개선 계획

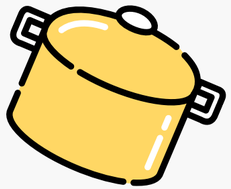
- AI 기능 고도화
- 게시글 등록시 저장한 여행지 불러오기
- 국외 여행지 추가
- 태그 기능 추가
- 여행지에 인원 관련 로직 추가

03

배운 점 및 느낀 점

- 실서비스 구조 설계, 협업 워크플로우의 중요성 체감
- 단순 구현보다 유지보수와 확장성을 고려한 개발이 중요함을 느낌
- GitHub Actions, Supabase, Zustand 등 새로운 기술 스택에 대한 실전 경험
- 문제 발생 시 빠르게 공유하고 해결하는 팀 협업의 효과

Q & A



!! 감사합니다 !!