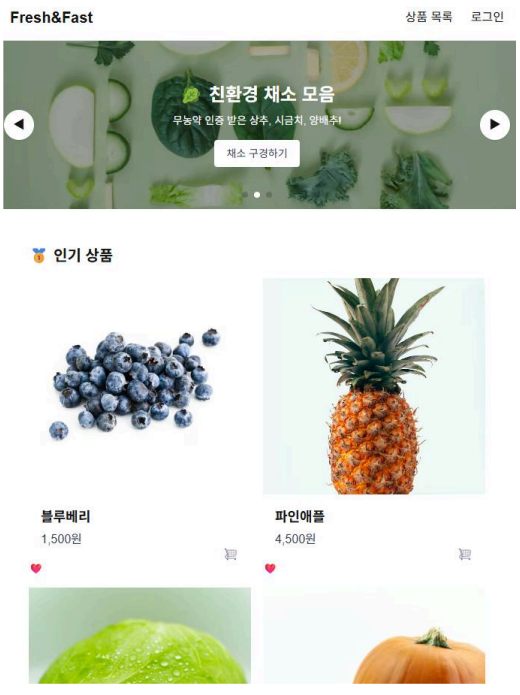
웹표준과 접근성을 고려한 UI 구현에 강점이 있는 프론트엔드 개발자 박민혜입니다.

박민혜 / minhyepark.dev@gmail.com



📌 프로젝트명 : 개인 프로젝트 Fresh&Fast (사이트 바로가기)

📅 진행기간 : 2025.07.21 ~ 2025.08.01

💼 역할 : 기획자, 디자이너, 프론트엔드 개발자, 백엔드 개발자

🛠️ 기술스택 : Next, TypeScript, Zustand, React-hook-form, Yup, Tailwind CSS, Lodash, Sharp, Supabase, Jest

🌐 프로젝트 소개

Next.js와 Supabase 기반의 온라인 쇼핑몰 프로젝트입니다.
상품 목록과 상세 조회, 장바구니, 주문 흐름, 마이페이지 기능을 구현했습니다.
실제 배포를 통해 성능 최적화, 보안(RLS), 상태관리 등 실무 요소를 다뤘습니다.
컴포넌트와 상태관리를 모듈화하여 유지보수성과 확장성을 높였습니다.
테스트코드 작성, 이미지 최적화, 반응형 구현 등 프론트엔드 실무 전반을 경험했습니다.

🚀 담당 역할 및 기여

1. 프론트엔드 전반 개발

- Next.js(App Router) 기반 페이지 설계 및 구현
- 상품 목록, 상세, 장바구니, 주문, 마이페이지 UI/UX 개발
- Zustand를 통한 전역 상태 관리 (로그인 유지, 장바구니 상태)
- Jest + RTL 기반 회원가입/로그인 기능 단위 테스트 작성
- Optimistic UI를 활용한 좋아요 기능 구현
- 상품 이미지 webp 변환
- throttle 기반 무한스크롤 최적화

2. 백엔드 연동

- Supabase를 이용한 데이터 CRUD, Auth, RLS 설정
- API 모듈화 및 공통 응답 데이터 camelCase 변환 유틸 구현

🎯 주요 성과

✅ 실서비스 배포: Vercel을 통한 CI/CD 환경 구축, 환경변수 관리 및 Preview 환경 설정
✅ UX 개선: 무한스크롤과 즉시 반영되는 좋아요 기능으로 인터랙션 속도 향상
✅ 보안 강화: Supabase Row Level Security(RLS) 적용으로 데이터 접근 제어
✅ 재사용성 확보: API 호출부, 타입 정의, 유틸 함수 공통화

⏳ 어려웠던 부분

Supabase 다중 테이블 조인 시 데이터 조회 문제발생

- 주문 상세 페이지에서 orders → order_items → products 테이블을 조인하여 데이터를 가져오는 기능을 구현하는 과정에서, Supabase의 PostgREST 기반 쿼리 문법에 익숙하지 않아 초기에는 다중 조인 시 데이터가 조회되지 않는 문제가 발생했습니다. 특히, 관계 설정(Foreign Key)은 정상적으로 되어 있음에도 중첩 select 구문이 제대로 동작하지 않아 개발이 지연되었습니다.

해결 방법

- Supabase SQL Editor를 활용해 순수 SQL 조인 쿼리를 먼저 검증한 뒤, 이를 PostgREST 문법으로 변환
- .select() 내부에서 관계형 키 이름과 별칭을 명확히 지정(product:products(name, price, image_url))
- API 호출부에서 응답 데이터를 camelCase로 변환하여 프론트엔드 타입 구조와 맞춤

결과

- 주문 상세 조회 기능에서 상품 정보가 정상적으로 포함되도록 조인 로직 안정화
- 이후 유사한 다중 조인 API 구현 시 개발 속도가 2배 이상 향상
- 코드 재사용이 가능해져 다른 페이지에서도 동일한 데이터 구조 활용 가능

📝 회고

🔷 프로젝트 회고

이 프로젝트에서는 프론트엔드 개발부터 Supabase 기반 API 연동, UI/UX 최적화, 상태 관리 개선까지 폭넓은 역할을 수행하며,
기술적 성장뿐만 아니라 사용자 경험과 운영 효율성을 함께 고려하는 개발의 중요성을 배울 수 있었다.
처음에는 단순히 기능 구현에 집중했지만, 진행하면서 코드의 유지보수성과 확장성을 위한 설계가 얼마나 중요한지 깨달았다.
특히, 공통 UI 컴포넌트를 구축하고 Zustand 기반 상태 관리 패턴을 적용하면서 개발 속도와 안정성이 크게 향상되었고,
이후 기능 추가나 유지보수가 훨씬 수월해졌다.
또한, 로그인 세션 유지, Optimistic UI, 무한 스크롤 등 작은 UX 디테일이 사용자 만족도를 크게 높일 수 있다는 점을 실감했다.

🔷 배운 점 & 개선할 점

🌱 배운 점

- 상태 관리와 컴포넌트 재사용성
  장바구니, 좋아요, 로그인 상태 등을 Zustand로 관리하면서, 상태 관리 패턴을 초기에 잘 설계하면
  전체 유지보수성과 확장성이 높아진다는 점을 체감함
- API 연동 및 협업 중요성
  Supabase를 통한 데이터 모델링과 RLS 설정 과정에서, API 설계와 백엔드 담당자와의 문서화·협업이 얼마나 중요한지 경험
- UX를 고려한 기능 설계
  무한 스크롤, Optimistic UI, 로그인 체크 등 사용자의 편의성과 서비스 완성도를 높이는 작은 디테일이
  사용자 경험에 큰 영향을 미친다는 점을 배움

🔍 앞으로의 개선점

- API 예외 케이스 사전 정의
  로그인 세션 만료, 비정상 데이터 응답, 네트워크 지연 등 다양한 예외 상황에 대한 핸들링 로직을 프로젝트 초기에 설계할 것
- 테스트 자동화 도입
  Jest + React Testing Library를 일부 기능에만 적용했으나, 전 페이지·주요 로직에 테스트 코드를 확장 적용하여
  배포 전 오류를 최소화할 것
- 디자인 시스템·상태 관리 패턴 명확화
  프로젝트 시작 시 컴포넌트 스타일 가이드와 상태 관리 구조를 문서로 정리해, 팀원 간 일관성을 높이고 리팩토링 비용을 줄일 것

🎉 마무리

이 프로젝트를 통해 단순한 기능 구현을 넘어, 사용자 경험·유지보수성·확장성까지 고려한 개발이
서비스의 완성도를 결정한다는 것을 깊이 이해할 수 있었다. 앞으로도 더 나은 코드 구조, 예외 상황을 고려한
안정적인 API 연동, 원활한 협업 방식을 고민하며 성장하는 프론트엔드 개발자로 나아가고자 한다.