김종민 프론트엔드 개발자

React, Next.js, TypeScript 기반의 4년차 프론트엔드 개발자로
다양한 도메인 개발을 진행해왔습니다.
레거시 개선과 문제의 원인을 파악하는데 큰 흥미를 느끼며
UX와 DX를 최우선으로 설계를 하려고 합니다.

Email. yung3152@gmail.com
Tel. 010-4023-6518
GitHub. github.com/Guksu
Blog. guksulog.com

Career

(주)클라이머스 2025.03 - 재직 중

취향 기반 이커머스 플랫폼(**찹스틱스**)에서 서비스, 셀러센터, 내부 어드민 개발을 담당하고 있습니다.

React, Next.js, Flutter, TypeScript, GraphQL, TanStack Query, Recoil

| 찹스틱스 서비스 개발

레거시 코드에 누적되어 있던 오류들을 분석하고 해결하여 서비스 안정성을 높였으며,
사용자 참여를 유도하는 커뮤니티 및 출석체크 기능을 설계하고 개발했습니다.

성능 최적화

번들 사이즈 최적화

- 현재 앱 규모에 비해 초기 로딩 속도가 느리고 번들 크기가 큰 문제가 있었습니다.
- lodash를 lodash-es로 교체, 대용량 이미지 경량화, tsx에서 불러오던 상수들을 ts 파일로 분리하여 트리쉐이킹이 가능하도록 개선했습니다.
- 결과적으로 공통 JS 번들 37% 감소, 각 페이지 평균 JS 번들 56.8% 감소를 달성했습니다.

TanStack Query 캐시 전략 재정립

- 모든 API에 동일하게 적용되어 있던 20초 기본 캐시 설정으로 인해 불필요한 네트워크 요청이 발생하고 있었습니다.
- 각 API 도메인 특성에 맞게 staleTime을 재설정하고, mutation 이후 쿼리 전체를 무효화하던 로직을 캐시 데이터만 수정하는 방식으로 변경했습니다.

상품 상세페이지 SSR 최적화

- 앱 내부에서 상세 페이지 전환이 느린 문제가 있었습니다.
- 원인을 분석한 결과, getServerSideProps에서 SEO에 불필요한 데이터(옵션, 리뷰, 배송 정보 등)까지 모두 가져오고 있었습니다.
- SEO에 필요한 최소한의 데이터만 SSR로 분리하고, 나머지는 CSR로 처리하도록 개선했습니다.
- 결과적으로 페이로드 74% 감소(3.4kB → 0.9kB), 응답 시간 68% 개선(~283ms → ~90ms)을 달성했습니다.

이미지 저장 방식 개선

- 앱에서 전달받는 Base64 이미지를 LocalStorage에 저장하면서 용량 제한 및 이미지 누락 문제가 있었습니다.
- IndexedDB를 활용하여 대용량 이미지도 안정적으로 저장할 수 있도록 개선했습니다.

비용 절감

Vercel 배포 비용 66% 절감

- 월 $300 수준이던 Vercel 배포 비용이 과도하게 지출되고 있었습니다.
- 사용하지 않는 Vercel Analytics를 제거하고, Next/Image의 minimumCacheTTL을 60초에서 1년으로 변경하여 이미지 최적화 비용을 줄였습니다.
- 결과적으로 월 비용을 $300에서 $100으로 약 66% 절감했습니다.

UX 개선

네이티브 앱 사용감 구현

- 웹뷰 기반 서비스지만 네이티브 앱과 유사한 사용자 경험을 제공하고자 했습니다.
- 카테고리가 있는 리스트에서 좌우 스와이프로 카테고리를 전환할 수 있는 공용 컴포넌트를 개발했습니다.
- 인스타그램 스타일의 핀치줌 이미지 확대 공용 컴포넌트를 개발했습니다.

보안 개선

민감 정보 보호 강화

- .env 파일에 노출되어 있던 API Key를 제거하고, SMS 인증번호 발송 등 민감한 로직을 백엔드로 이전했습니다.
- 개발 및 스테이징 환경에 미들웨어 기반 IP 화이트리스트를 적용하여 외부 접근을 차단했습니다.

DX 개선

개발 환경 및 버전 관리 체계화

- 기존에 모든 프로젝트가 0.0.0 버전으로 관리되어 배포 이력 추적이 어려웠습니다.
- 소스코드 버전 관리 체계를 도입하여 배포 버전을 명확하게 관리할 수 있도록 개선했습니다.
- 또한, Flutter 테스트 빌드에서 스플래시 화면 전 개발/스테이징/프로덕션 환경을 선택할 수 있도록 개선하여 QA 효율성을 높였습니다.

Flutter

Android API 35 대응 및 안정성 개선

- Android API 35 정책 변경에 따라 edge-to-edge 레이아웃을 적용했습니다.
- 백그라운드에서 포그라운드로 전환 시 발생하던 백화현상을 해결했습니다.
- Flutter v3.24에서 v3.32로 마이그레이션을 진행했습니다.

(주)엘케이벤쳐스 2024.04 - 2024.12

해외 전용 **이미지 다운로드 사이트** 개발과 **인생네컷 점주앱** 유지보수를 했습니다.

React, Next.js, TypeScript, next-i18next, TanStack Query, Recoil

이미지 다운로드 사이트
이미지 파일 다운로드 속도 개선
- 기획 요건상 외부 URL에서 브라우저로 직접 다운로드하지 않고, 서버를 경유해야 했습니다.
- Next.js API Route를 활용하여 파일 다운로드 로직을 구현했습니다.
- 초기에는 FileReader 방식으로 구현했으나 다운로드 속도가 느려, Stream 방식으로 변경하여 성능을 개선했습니다.

Canvas API 기반 이미지 합성 기능 구현
- Canvas API를 활용하여 사용자가 촬영한 사진과 템플릿을 합성하는 기능을 개발했습니다.
- 대용량 이미지와 Canvas 크기로 인해 모바일 Safari에서 메모리 제한 오류가 발생했습니다.
- Canvas와 이미지를 적절한 비율로 스케일링하고, 사용이 끝난 Canvas 객체를 즉시 해제하여 메모리 문제를 해결했습니다.

이미지 업로드 기능 개선
- HEIC 파일 업로드 시 서버에서 처리되지 않는 오류를 발견했습니다.
- HEIC 포맷의 낮은 브라우저 호환성을 고려하여 heic2any 라이브러리를 도입해 클라이언트에서 변환 후 업로드하도록 개선했습니다.
- Android 버전별로 HEIC 파일 처리 방식이 달라 발생하는 호환성 이슈는 버전별 분기 처리 로직을 추가해서 해결했습니다.

(주)버츄어라이브 2023.05 - 2024.04

관광객 대상 뷰티 예약 플랫폼(**비트립**) 과 **인생네컷 점주앱**을 개발했습니다.

React, Next.js, TypeScript, next-i18next, Vitest, TanStack Query, Recoil

Presentational-Container 패턴에서 Atomic 패턴으로 전환
- 기존 프로젝트는 로직과 뷰를 분리하기 위해 Presentational-Container 패턴을 사용하고 있었습니다.
- 그러나 실제 로직은 이미 커스텀 훅으로 분리되어 있어 Container의 역할이 모호했고, 폴더 구조 개선이 필요했습니다.
- 프로젝트 특성에 맞게 Atomic 패턴을 커스터마이징하여 도입하고, 컴포넌트 재사용성과 구조의 명확성을 높였습니다.

TypeScript 레거시 코드 정리 및 ESLint 룰 도입
- any 타입 남용과 재사용이 불가능한 일회성 타입이 많아 코드 품질이 낮은 상태였습니다.
- Sentry에 타입 관련 런타임 오류가 지속적으로 리포팅되어 전반적인 타입 정리가 필요했습니다.
- 공통 ESLint 룰이 없어 각 파일마다 코드 스타일이 달랐고, 이를 해결하기 위해 Airbnb 룰을 기반으로 ESLint 설정을 통일했습니다.

Vitest 기반 단위 테스트 환경 구축
- 인생네컷 점주 전용 앱 특성상 금액 계산 로직의 정확성이 중요했습니다.
- 개발 초기부터 Vitest를 도입하여 핵심 비즈니스 로직에 대한 단위 테스트를 작성하고, 안정성을 확보했습니다.

CI/CD 환경 구축
- GitLab Runner를 EC2에 등록하여 CI/CD 파이프라인을 구축했습니다.
- 커밋 및 배포 실패 시 Slack 알림을 연동하여 빠르게 대응할 수 있도록 했습니다.

(주)헬로비즈 2022.02 - 2023.02

외주 전문 개발업체에서 클라이언트와 직접 소통하여 **7개의 MVP제품**을 개발했습니다.

React, Next.js, TypeScript, next-i18next, TanStack Query, Zustand, Redux

Redux에서 TanStack Query & Zustand로 마이그레이션 추진
- 기존에는 Redux 하나로 서버 상태와 클라이언트 상태를 모두 관리하고 있었습니다.
- 보일러플레이트 코드가 많고 상태 흐름이 복잡해 유지보수가 어려웠습니다.
- 서버 상태는 TanStack Query로, 클라이언트 상태는 Zustand로 분리하는 마이그레이션을 추진했습니다.

React 컴포넌트를 범용 JS 모듈로 변환
- 카카오 지도를 사용하는 React 컴포넌트를 Android 앱에서 사용해야 하는 요구사항이 있었습니다.
- Android에서 index.html로 직접 접근할 수 있도록 React 의존성을 제거하고 순수 JavaScript 모듈로 변환했습니다.

DB 구조 분석 및 API 설계서 작성
- 백엔드 리소스가 제한된 상황에서 프로젝트 병목을 해소하기 위해 직접 DB 구조를 분석하고 API 설계를 주도했습니다.
- Node.js 기반 백엔드 코드를 파악하여 API 명세서를 작성하고, 백엔드 개발자와 협업하여 구현까지 연결했습니다.