하루담은

건강한 하루 식단을 위한 맞춤형 쇼핑몰

어흥냥

목 차

영양 성분표는 어렵다!

〈그림 성인 하루 1,800 kcal 식단 예시〉

식품군별 권장횟수	식단	아침	점심	저녁	간식
		쌀밥, 콩나물국, 고등어구이, 시금치나물, 배추김치 	현미밥, 미역국, 불고기, 열무김치 	쌀밥, 두부김치찌개 도토리묵무침, 우엉조림 	바나나, 우유, 사과, 두유 
곡류	3회	쌀밥 210g(1)	현미밥 210g(1) 	쌀밥 147g(0.7) 도토리묵 200g(0.3) 	
고기 · 생선 · 달걀 · 콩류	3.5회	고등어 60g(1) 	쇠고기 60g(1) 	두부 40g(0.5)	두유 1컵(200ml)(1) 
채소류	7회	배추김치 20g(0.5) 콩나물 35g(0.5) 시금치나물 70g(0.5) 	양파 35g + 생표고 15g(1) 미역 15g(0.5) 열무김치 40g(1) 	배추김치 40g(1) 우엉 20g(0.5) 쑥갓 35g + 오이 35g(1) 	
과일류	2회				바나나 1개(1) 사과(중) 1/2개(1) 
우유 · 유제품류	1회				우유 1컵 (200ml)(1) 

다 중요한 건 알겠는데,
난 햄버거, 김밥 같은 음식을 먹잖아.
내가 먹은 음식들의 영양소를 일일이 체크하고 계산하기 어려워....
내가 원하는 음식을 고를 때마다 자동으로 영양 정보를 계산해주는 서비스가 있으면 얼마나 좋을까?

이커머스 + 헬스케어

1인 가구, 식사 불규칙·영양섭취 부실

연합뉴스 | 기사전송 2016-03-09 06:10 최종수정 2016-03-09 08:54

AI챗으로 요약

13 0

연구원이 보건복지부의 2013년 국민건강영양조사 원자료를 분석한 결과를 보면 곡류와 주류는 1인 가구 섭취량이 2인 이상 가구의 99.9%, 100.2%로 2인 이상 가구와 비슷했다.

그러나 2인 이상 가구와 비교해 1인 가구의 수산물 섭취량은 61.7%에 그쳤고, 과일(74%)·축산물(78.2%)·채소(89.4%) 등도 섭취량이 10~25%가량 적었다.

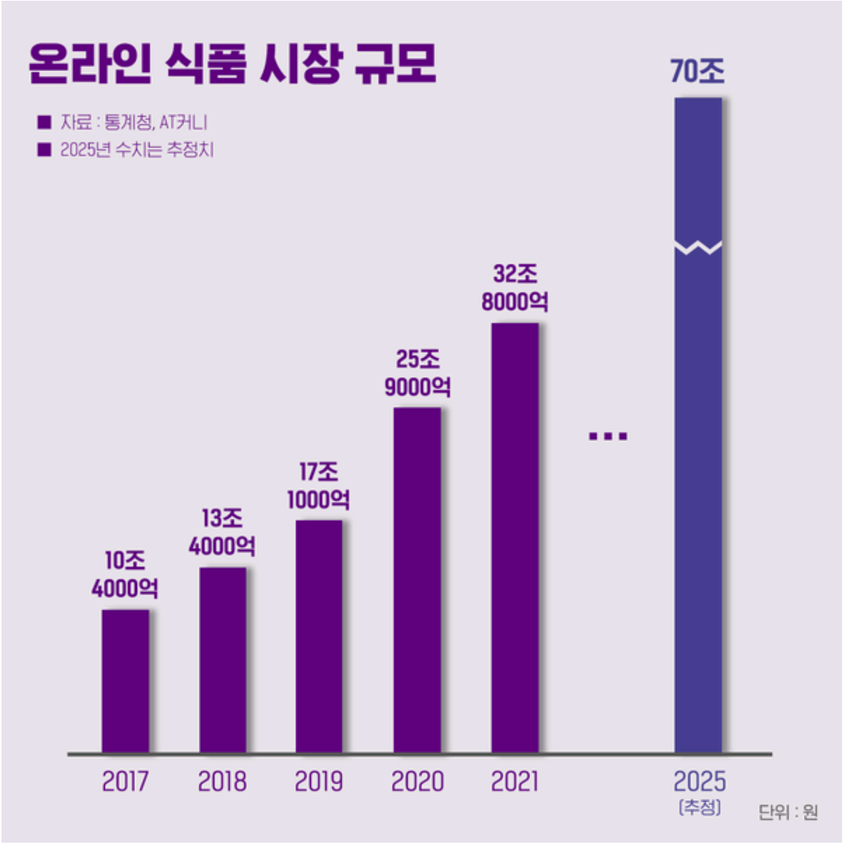
1인 가구의 권장섭취기준 대비 영양소 섭취 비율도 전반적으로 낮은 편이다. 칼슘(60.2%), 칼륨(77.4%), 비타민C(79.4%), 리보플라빈(85.3%), 비타민A(86.8%), 나이아신(93%) 등의 섭취량이 권장 섭취량에 못 미쳤다.

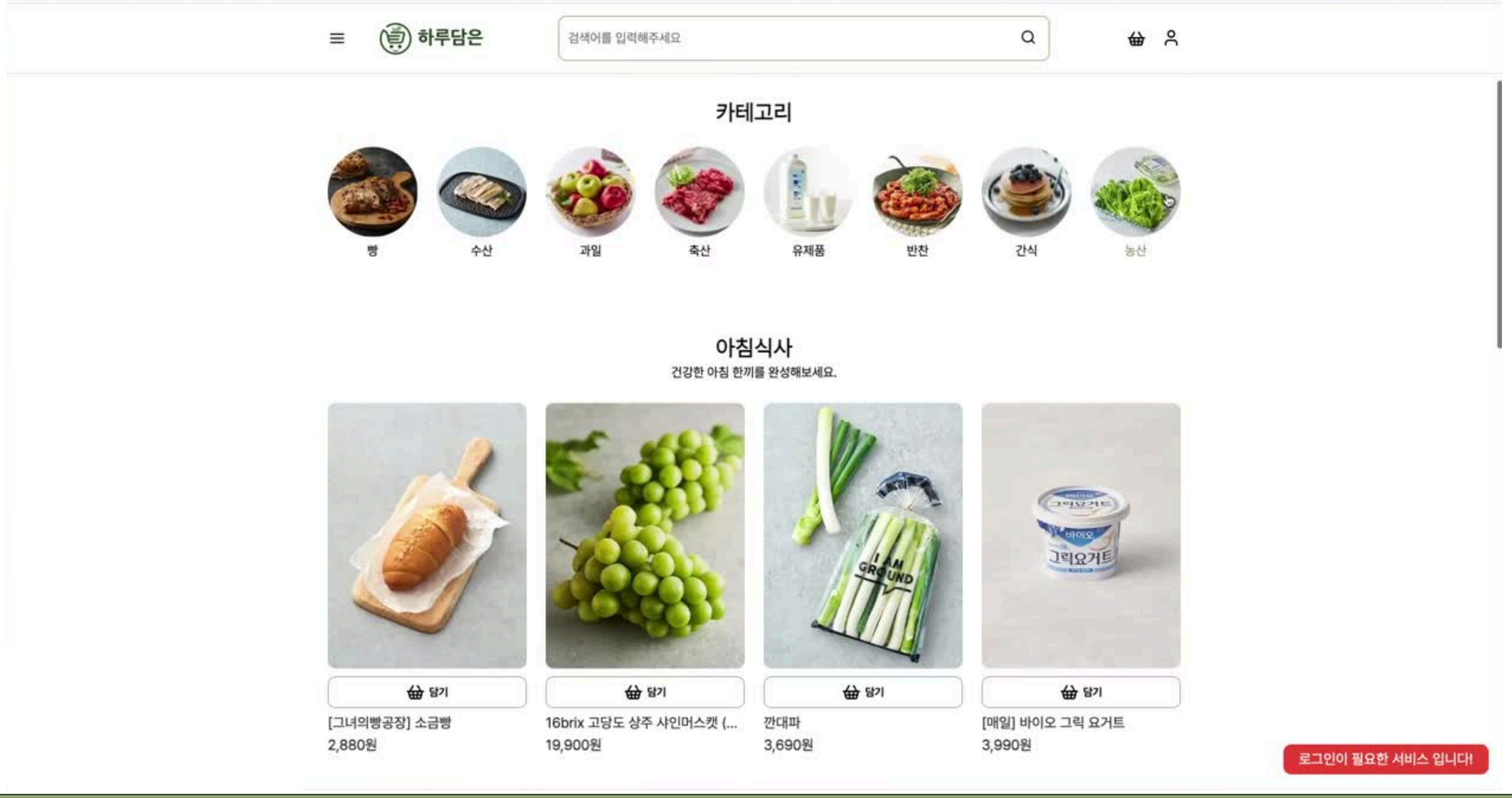
반면 2인 이상 가구에서는 칼슘(73.4%)과 칼륨(90%)을 제외한 대부분 영양소를 권장량을 초과해 섭취한다.

또 1인 가구는 탄수화물에 의한 에너지 섭취 비중이 70.1%로 2인 이상 가구(65.4%)보다 높았다. 반면 단백질(13.7%)과 지방(16.3%)은 2인 이상 가구(단백질 14.8%·19.7%)와 비교해 낮았다.

영양섭취가 부족한 사람의 비율은 1인 가구가 11.7%, 2인 이상 가구가 6%였다. 1인 가구 중 20~30대(13.3%)나 60대 이상(12%)에서 영양섭취가 부족한 경우가 많았다.

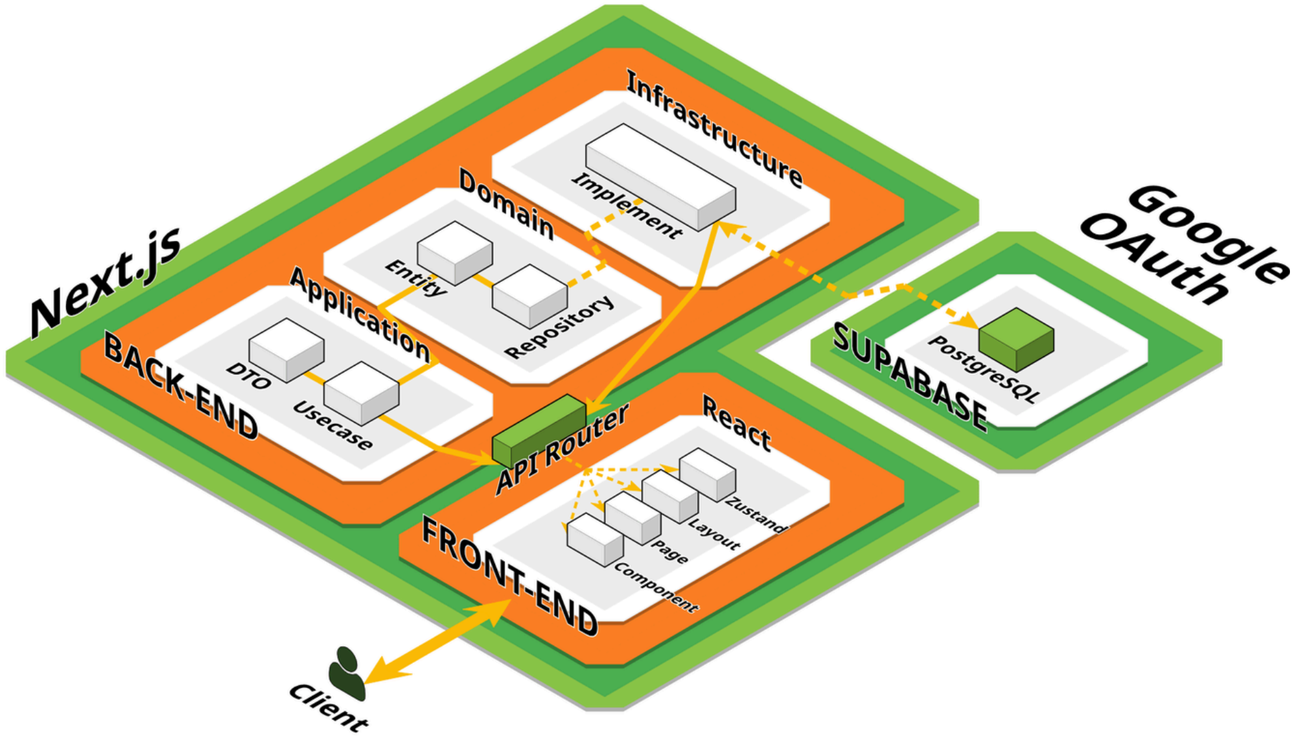
1인 가구의 월평균 식품비는 2014년 기준 28만7천원으로 2인 이상 가구 1인당 식품비(38만7천원)의 74.4% 수준이다.




하루담은
검색어를 입력해주세요
카테고리
빵
수산
과일
축산
유제품
반찬
간식
농산
아침식사
건강한 아침 한끼를 완성해보세요.
담기
[그녀의빵공장] 소금빵
2,880원
담기
16brix 고당도 상주 샤인머스캣 (...
19,900원
담기
깐대파
3,690원
담기
[매일] 바이오 그릭 요거트
3,990원
로그인이 필요한 서비스 입니다!

About 프로젝트

시스템 아키텍처



📌 주요 프레임워크 & 라이브러리
- Next.js 15.1.5 – React 기반 서버 사이드 렌더링(SSR) 프레임워크
- React 19.0.0 – UI 구성 라이브러리
- Zustand 5.0.3 – 전역 상태 관리 라이브러리

🎨 UI & 스타일링
- classnames 2.5.1 – 동적 CSS 클래스 핸들링
- lucide-react 0.474.0 – 아이콘 라이브러리
- CSS Modules – 모듈화된 CSS 스타일링

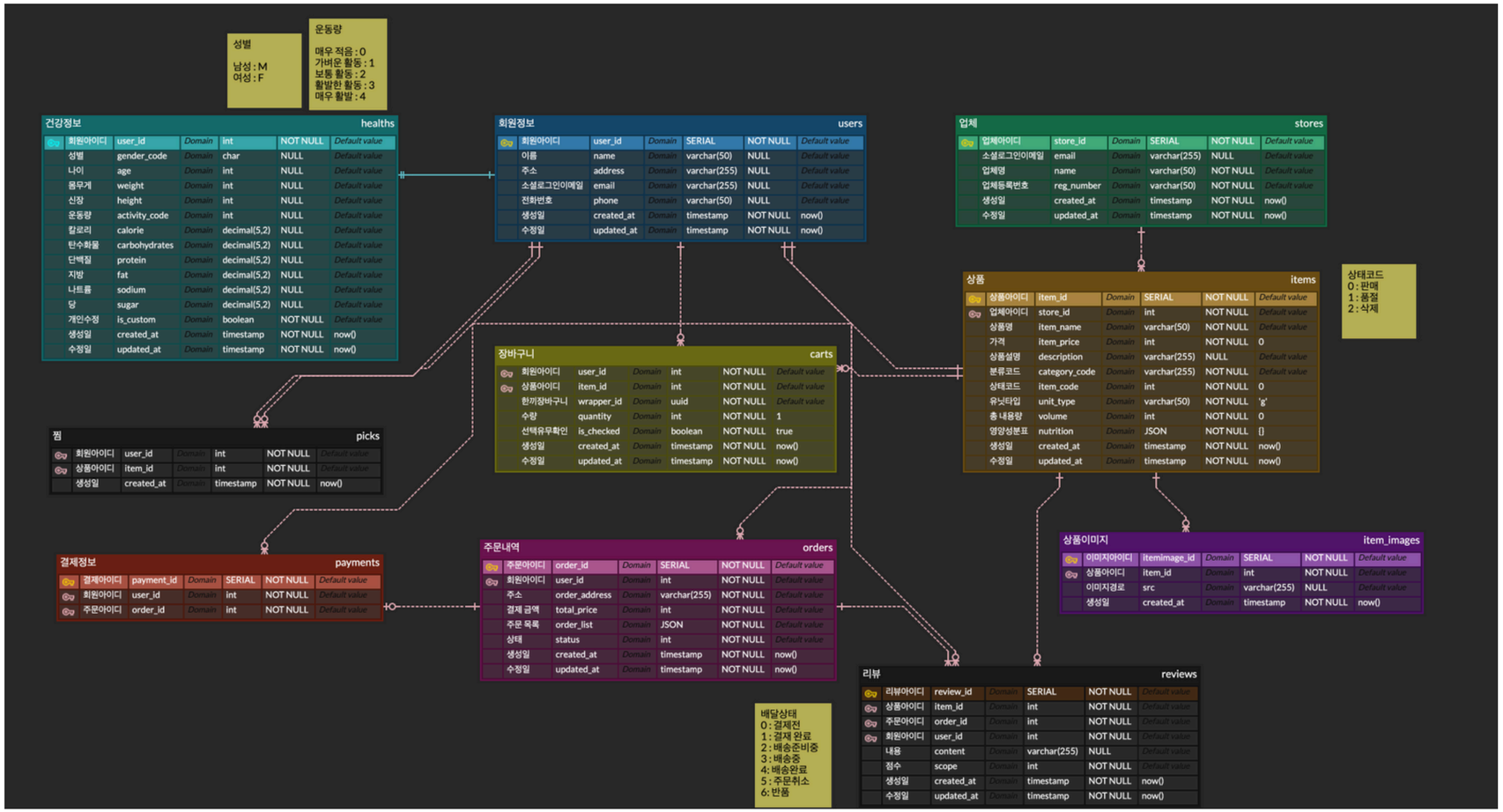
📊 데이터 시각화
- Chart.js 4.4.7 – 차트 시각화 라이브러리

🛠️ Drag & Drop
- @dnd-kit/core 6.3.1 – 드래그 앤 드롭 기능 구현

🔐 인증 & 보안 / 📦 데이터 관리
- Supabase (@supabase/supabase-js, @supabase/ssr)
  – 백엔드리스 인증 및 데이터베이스

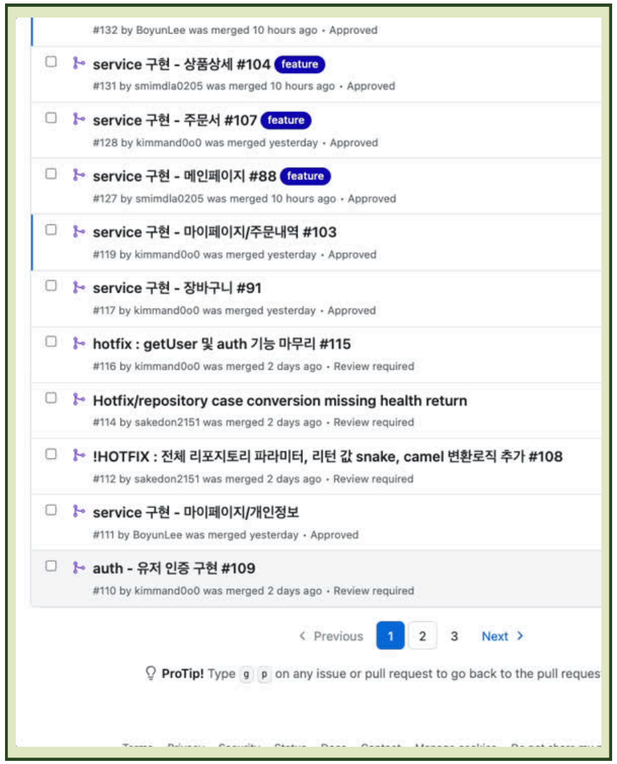
📝 코드 품질 & 개발 도구
- ESLint / Prettier – 코드 린팅 및 포맷팅
- TypeScript – 정적 타입 검사
- React Hook Form – 폼 상태 및 유효성 검사
- Nookies – Next.js에서 쿠키 관리

ERD

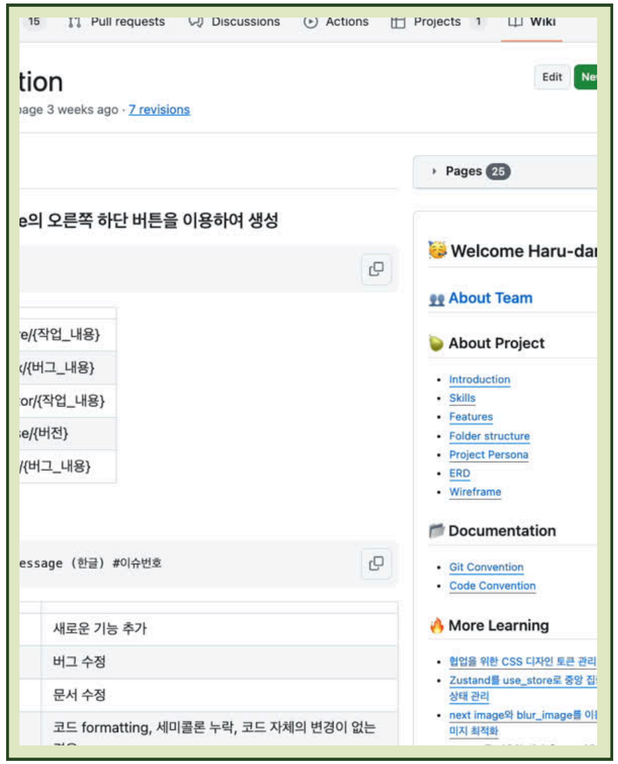

성별
남성 : M
여성 : F
운동량
매우 적음 : 0
가벼운 활동 : 1
보통 활동 : 2
활발한 활동 : 3
매우 활발 : 4
건강정보 healths
회원아이디 user_id Domain int NOT NULL Default value
성별 gender_code Domain char NULL Default value
나이 age Domain int NULL Default value
몸무게 weight Domain int NULL Default value
신장 height Domain int NULL Default value
운동량 activity_code Domain int NULL Default value
칼로리 calorie Domain decimal(5,2) NULL Default value
탄수화물 carbohydrates Domain decimal(5,2) NULL Default value
단백질 protein Domain decimal(5,2) NULL Default value
지방 fat Domain decimal(5,2) NULL Default value
나트륨 sodium Domain decimal(5,2) NULL Default value
당 sugar Domain decimal(5,2) NULL Default value
개인수정 is_custom Domain boolean NOT NULL Default value
생성일 created_at Domain timestamp NOT NULL now()
수정일 updated_at Domain timestamp NOT NULL now()
회원정보 users
회원아이디 user_id Domain SERIAL NOT NULL Default value
이름 name Domain varchar(50) NULL Default value
주소 address Domain varchar(255) NULL Default value
소셜로그인이메일 email Domain varchar(255) NULL Default value
전화번호 phone Domain varchar(50) NULL Default value
생성일 created_at Domain timestamp NOT NULL now()
수정일 updated_at Domain timestamp NOT NULL now()
업체 stores
업체아이디 store_id Domain SERIAL NOT NULL Default value
소셜로그인이메일 email Domain varchar(255) NULL Default value
업체명 name Domain varchar(50) NOT NULL Default value
업체등록번호 reg_number Domain varchar(50) NOT NULL Default value
생성일 created_at Domain timestamp NOT NULL now()
수정일 updated_at Domain timestamp NOT NULL now()
상품 items
상품아이디 item_id Domain SERIAL NOT NULL Default value
업체아이디 store_id Domain int NOT NULL Default value
상품명 item_name Domain varchar(50) NOT NULL Default value
가격 item_price Domain int NOT NULL 0
상품설명 description Domain varchar(255) NULL Default value
분류코드 category_code Domain varchar(255) NOT NULL Default value
상태코드 item_code Domain int NOT NULL 0
유닛타입 unit_type Domain varchar(50) NOT NULL 'g'
총 내용량 volume Domain int NOT NULL 0
영양성분표 nutrition Domain JSON NOT NULL {}
생성일 created_at Domain timestamp NOT NULL now()
수정일 updated_at Domain timestamp NOT NULL now()
상태코드
0 : 판매
1 : 품절
2 : 삭제
장바구니 carts
회원아이디 user_id Domain int NOT NULL Default value
상품아이디 item_id Domain int NOT NULL Default value
한끼장바구니 wrapper_id Domain uuid NOT NULL Default value
수량 quantity Domain int NOT NULL 1
선택유무확인 is_checked Domain boolean NOT NULL true
생성일 created_at Domain timestamp NOT NULL now()
수정일 updated_at Domain timestamp NOT NULL now()
찜 picks
회원아이디 user_id Domain int NOT NULL Default value
상품아이디 item_id Domain int NOT NULL Default value
생성일 created_at Domain timestamp NOT NULL now()
결제정보 payments
결제아이디 payment_id Domain SERIAL NOT NULL Default value
회원아이디 user_id Domain int NOT NULL Default value
주문아이디 order_id Domain int NOT NULL Default value
주문내역 orders
주문아이디 order_id Domain SERIAL NOT NULL Default value
회원아이디 user_id Domain int NOT NULL Default value
주소 order_address Domain varchar(255) NOT NULL Default value
결제 금액 total_price Domain int NOT NULL Default value
주문 목록 order_list Domain JSON NOT NULL Default value
상태 status Domain int NOT NULL Default value
생성일 created_at Domain timestamp NOT NULL now()
수정일 updated_at Domain timestamp NOT NULL now()
상품이미지 item_images
이미지아이디 itemimage_id Domain SERIAL NOT NULL Default value
상품아이디 item_id Domain int NOT NULL Default value
이미지경로 src Domain varchar(255) NULL Default value
생성일 created_at Domain timestamp NOT NULL now()
리뷰 reviews
리뷰아이디 review_id Domain SERIAL NOT NULL Default value
상품아이디 item_id Domain int NOT NULL Default value
주문아이디 order_id Domain int NOT NULL Default value
회원아이디 user_id Domain int NOT NULL Default value
내용 content Domain varchar(255) NULL Default value
점수 scope Domain int NOT NULL Default value
생성일 created_at Domain timestamp NOT NULL now()
수정일 updated_at Domain timestamp NOT NULL now()
배달상태
0 : 결제전
1 : 결제 완료
2 : 배송준비중
3 : 배송중
4 : 배송완료
5 : 주문취소
6 : 반품

어흥냥's의 협업

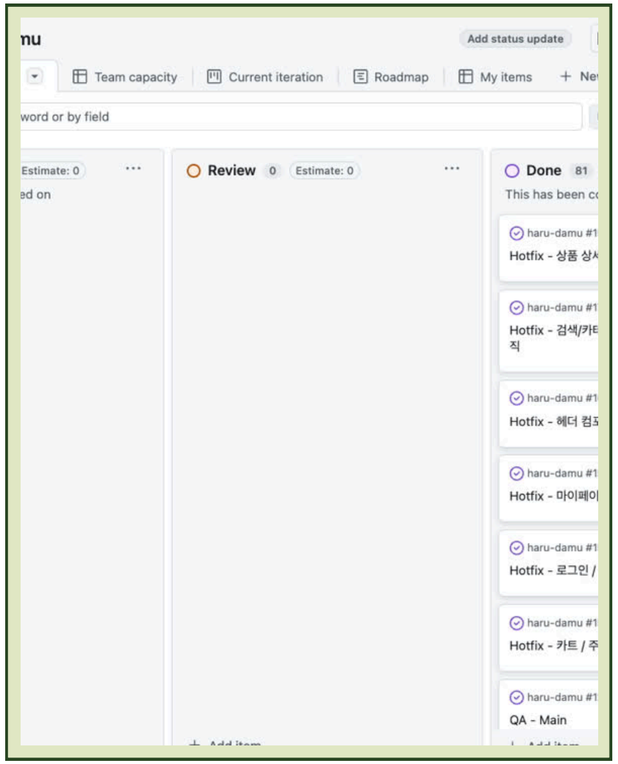
108개의 이슈, 66개의 PR을 통한 협업



Wiki를 이용한 자료 관리



칸반을 통한 현업무와 어려움 파악



저희의 협업이 궁금하다면 HARU-DAMU의 깃허브로 놀러오세요~!

협업을 위한 CSS 디자인 토큰 관리

저희는 협업이 처음이고, 피그마가 서툰 팀원이 많았습니다. 와이어프레임 단계에서 협업을 위해 색을 맞추는 것이 중요하다고 판단하고 css를 토큰화 하여 관리하기로 약속하였습니다.

1. 디자인 일관성 유지
디자인 토큰은 색상, 폰트 크기, 간격 등의 속성을 중앙 집중화하여 관리할 수 있게 해줍니다. 이를 통해 모든 팀원이 동일한 디자인 기준을 따르게 되어, 하루담은 전반에 걸쳐 일관된 사용자 경험(UX)을 제공하고 있습니다.

2. 효율적인 협업과 커뮤니케이션
이 색상을 토큰화 하여 관리한 덕분에, 와이어 프레임에서 브랜드 primary-color를 중간에 변경하는 상황에서도 코드 변경을 최소화 하고, 다른 팀원의 작업에도 방해가 되지 않도록 하여 쉽게 바꿀 수 있었습니다.

하루-담은 사용 예:

```
--base-white: #ffffff;
--base-black: #000000;
--base-content-dark-text: #222222;
--base-content-sub-text: #999999;

--primary-color: #284521;
--secondary-color: #8E9866;
--accent-color: #FA8125;
--warning-color: #FBBC05;
--important-color: #EA4335;
--light-gray-color:#E5E4E7;

--pale-sage-color :#BCC89A;
--soft-pistachio-color :#DDE9B0;

--neutral-gray-color: #DDD;
--background-light-color: #F9F9F9
```

Zustand를 use_store로 중앙 집중식 상태 관리

1. zustand를 그냥 사용했을 때 문제점
zustand를 각각의 slice로 관리하다보면 다음의 단점이 있습니다.

- 각각의 store에서 상태를 가져와야하기 때문에 import와 const 선언문이 길어집니다.
- 각각의 store에 속해있는 상태값을 서로 참고하거나 가져올 수 없습니다.

2. use_store 훅을 통한 중앙 집중식 관리
백화점을 떠올리면 좋을 것 같습니다. 각각의 store를 use_store에 입점시켜 서로를 참조하고, 함께 가져올 수 있도록 합니다.
기본 구조는 zustand의 공식문서를 참고하여 작성하였습니다.
추가로, 휘발 될 수 있는 상태값을 localStorage에 저장하여 재사용성을 높였습니다.

```
// use_store.ts
export type State = TCartSlice;

export const useStore = create<State>()(
  subscribeWithSelector(
    persist(
      (...a) => ({
        ...createCartSlice(...a),
      }),
      {
        version: 144,
        name: "haru-damu",
        storage: createJSONStorage(() => storage),
        partialize: (keys) => {
          // eslint-disable-next-line @typescript-eslint/no-unused-vars
          const { ...persistData } = keys;

          return persistData;
        },
      },
    ),
  ),
);

export const selector = createSelectorFunctions(useStore);
```

방향(Direction)을 통한 dnd관리

여러 형태의 컴포넌트를 서로 드래그 앤 드랍하여 작업하기 위해서는 드래그가 가능한 것과 드랍이 가능한 컴포넌트를 분리하고, 방향을 정해주는 게 중요하다고 생각했습니다.

먼저 각각의 컴포넌트 타입을 정의합니다. 그 후 드래그가 가능한 방향을 정의하고 값을 만들어주는 유틸을 만들었습니다.

이렇게 지정 된 방향을 통해 이벤트를 넣어주기 때문에 서로 다른 컴포넌트 사이의 이벤트도 문제없이 처리할 수 있습니다.

이 방향을 정한것으로 dnd의 확장성을 넓힐 수 있었습니다.

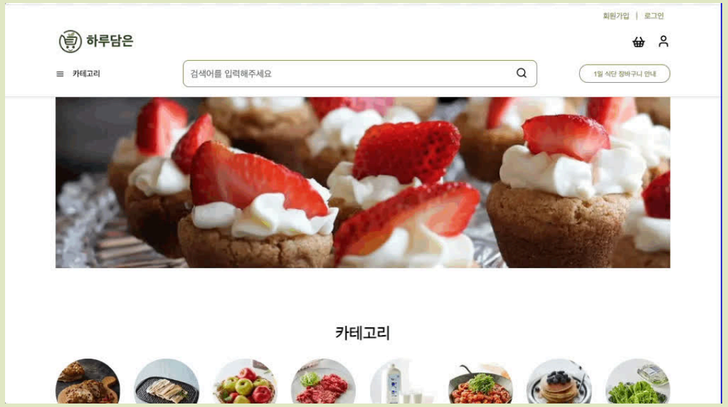
```
// type.ts
// 드래그 가능한 컴포넌트 타입 정의
export type DraggableComponent = (typeof DRAGGABLE_TYPES)
[number];
export type DroppableArea = (typeof DROPPABLE_ONLY_TYPES)
[number];
export type DraggableAndDroppable = (typeof DRAGGABLE_TYPES |
typeof DROPPABLE_ONLY_TYPES)[number];

export type FromDirection = DraggableComponent;
export type ToDirection = DraggableAndDroppable;

export type DragDirection = `${FromDirection} -> ${ToDirection}`;

// get-drag-direction.ts
import type { DragDirection, FromDirection, ToDirection } from
"@/types";

function getDragDirection(activeType: FromDirection, overType:
ToDirection): DragDirection {
  return `${activeType} -> ${overType}`;
}

export default getDragDirection;
```

scrollY를 이용한 헤더 축소로 사용자 경험 강화

쇼핑몰에서 가장 중요한 것은 아이템 카드 입니다. 가끔 사용하는 헤더의 아이콘들을 숨겨 유저의 사용성을 강화 하였습니다.

유저는 처음 접속하여 하루담은의 아이덴티티를 가진 헤더를 확인할 수 있습니다. 이때 스크롤이 발생해 100 이상의 움직임이 생긴다면, 헤더는 필요한 아이콘을 제외하고는 축소됩니다.

헤더는 모든 페이지에서 사용되기때문에 스크롤이 발생할 때 마다 이벤트를 발생합니다. 이를 막기 위해 y의 크기로 조건을 걸어 최소한의 이벤트를 발생하도록 개선하여 사용하고 있습니다.




```
const handleScroll = useCallback(() => {
  const y = window.scrollY;

  if (y > 200) return;

  setIsShrunk(y > 100);
}, []);

useEffect(() => {
  window.addEventListener("scroll", handleScroll);
  return () => window.removeEventListener("scroll", handleScroll);
}, [handleScroll]);
```

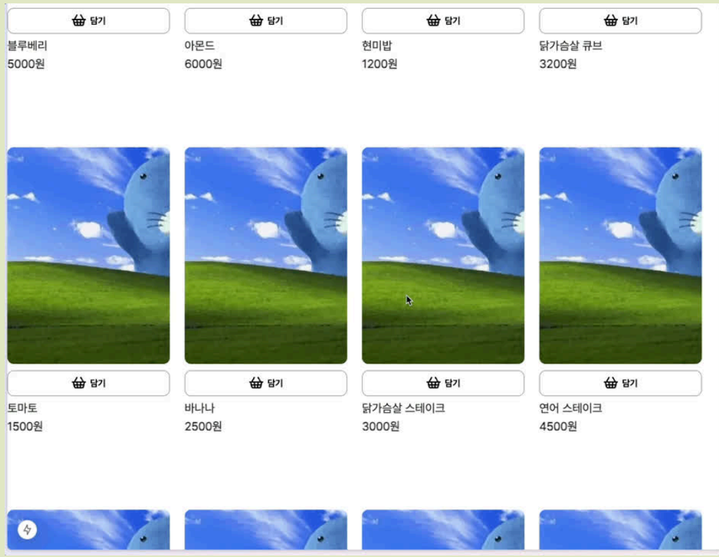
next image와 blur_image를 이용한 이미지 최적화

가장 유명한 쇼핑몰인 쿠팡에서도 화면을 불러오는 데 2초 이상 걸리기도 합니다. 쇼핑몰은 데이터도, 이미지도 많아 화면을 로딩하는 과정에서 유저의 이탈률이 많이 생긴다고 합니다. 이를 개선하기 위해 lazy-loading을 next/image에서 기본 제공을 하고 있다는 것을 알 수 있었습니다. 이를 이용해서 화면에 보이지 않는 이미지를 늦게 로딩하여 최적화를 할 수 있었습니다.

더해서 이미지 로딩을 더욱 개선하기 위하여 blur_image를 이용하여 사용자 경험을 향상시키는 것을 선택하였습니다.

get_blur_img.ts util 추가

- next.js/image에서 기본으로 제공하는 기능 중 placeholder를 이용하여 blur이미지를 제공해 유저 경험을 더 향상시킬 수 있음을 알게 되었습니다. 다만 외부 storage를 사용하기 때문에 blur이미지를 따로 만들어 넣어줘야하는 문제가 있었습니다.
- 공식문서는 이때 plaiceholder 라이브러리를 추천합니다.



앞으로의 하루담은

판매자 서비스 개발

상품을 등록, 수량, 삭제를 관리하고
주문 받은 내역을 관리하는 서비스 추가 개발

결제 서비스 개발

토스, 네이버 페이의 developer 페이 기능을 통해 결제 서비스 개발

코드 리팩토링

프로젝트 마감일에 쫓겨 정리하지 못한 코드 정리
데이터 변화가 적어 리렌더링이 없는 페이지 SSR로 개선
React-query등 데이터 캐싱 및 동기화를 통한 사이트 최적화

TDD 학습

테스트 주도 개발로 코드의 품질을 높이고 버그를 미리 방지하며 리팩토링을 용이할 수 있도록 학습

QnA

감사합니다