

내 룸메이트와의 하모니를 위하여

Harmony

20618 주현명 20509 오유성

FIRST STEP

프로젝트 기획 의도, 자료조사 결과

룸메이트와 함께 동거중인
10~20대 다수가 말한 불편함

비친족 가구는 점점 증가하는 추세입니다.
하지만 늘어나는 비친족 가구에 비해
비친족 가구의 생활은 **개선되어지고 있지 않죠**

저희는 그래서 룸메이트와 함께 동거를 하고 있는
10~20대 여려명에게 설문조사를 진행하였는데요

설문조사 결과, **룸메이트 간의 소통이 원활하지 못하다는 점**이
가장 큰 문제점으로 꼽혔습니다.

“룸메이트간 **소통**이 불편해요”

공용으로 사용하는 물품이나
비용의 정산이 복잡해요.

각자의 일정을 **공유**하지 않아
서로에게 불편함을 주곤 해요.

룸메이트와의 **집안일 분배**가 어려워요.

공통된 **소통의 문제**

DEVELOPMENT PROCESS

기술 스택 선정 및 그 이유

채택된 기술 스택&이유

Frontend(주현명)

React Native React Query TypeScript

기술 선정 이유 – React Native

많은 사용자층과 확장성을 고려하여 크로스플랫폼 개발이 가능한 플랫폼을 생각하면서 웹 개발이 메인인 제가 다루기 편한 React Native를 주 기술스택으로 채택하게 되었습니다. React Native에서 CodePush를 통하여 사용자의 니즈를 빠르게 충족시켜줄 수 있는 장점이 있습니다.

Backend(오유성)

Nest.js TypeORM TypeScript

기술 선정 이유 – Nest.js

TypeScript를 언어로 사용하는 nest.js를 사용하여 프론트엔드 개발자와 소통과 스키마 설계가 더 수월하게 했습니다. 또한 SpringBoot 기반으로 만들어진 프레임워크인만큼 어노테이션, 데코레이션을 통한 스키마 설계, 컨트롤러 작성이 편리하였고 api 문서 또한 이러한 어노테이션과 데코레이터를 사용하여 간편하게 제작할 수 있었습니다.

DEVELOPMENT PROCESS

개발 진행 내용 정리 - Frontend(주현명)

개발 과정

- 
 - UI/UX User flow 설계
 - 와이어프레임 제작 및 디자인
 - Navigation 구조 설계 및 screen ui 구현
 - 백엔드 메소드 명세
 - 백엔드 연결 및 skeleton ui 개발
 - QA, 1차 배포

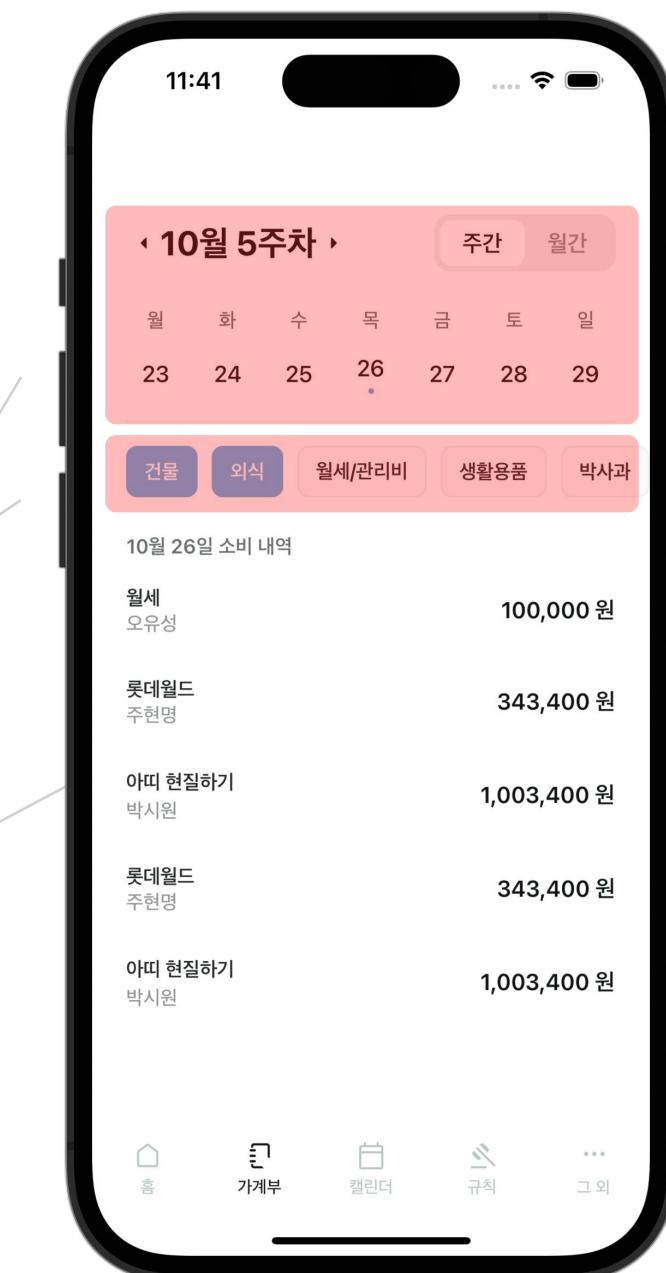
개발 현황 (UI)



Calendar U



가계부 화면 UI



The image shows a smartphone screen displaying a calendar and a list of tasks. The top status bar indicates the time as 11:43 and shows signal strength, battery level, and connectivity icons. A pink header box at the top left contains the text '11월 1주차' (Week 1 of November). Below this is a weekly grid from Monday to Sunday, with November 26th highlighted in blue. The main content area lists tasks for November 26th:

- 10월 26일에 해야 할 일 (Tasks to do on October 26th)
 - 프로젝트 회의 (Project Meeting)
 - 이나린
- 10월 26일에 예정된 일정 (Scheduled appointments on October 26th)
 - 프로젝트 회의 (Project Meeting)
 - 이나린

At the bottom are navigation icons: 홈 (Home), 가계부 (Family Budget), 캘린더 (Calendar), 규칙 (Rules), and 그 외 (Others).

캘린더 화면 UI

DEVELOPMENT PROCESS

개발 진행 내용 정리 - Backend(오유성)

개발 과정

- 스키마 모델 설계
- 모델 바탕으로 컨트롤러 작성
- 서비스 작성 및 API 문서 명세
- TDD(Test Driven Development) 방법론
- Authentication, Authorization(인증, 인가)
- Kakao OAuth 연동 후 KakaoGuard 적용
- 최종 배포

개발 현황 (Entity, Controller)

```
@Entity()
export class Room {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  roomCode: string; // 6자리 숫자 코드

  @Column()
  ownerUid: string; // 방주인의 uid
  events: any;
}

@Entity()
export class Member {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  uid: string; // 사용자 uid

  @Column()
  roomCode: string; // 가입된 방 코드
}

export enum TodoFrequency {
  Daily = 'Daily',
  Weekly = 'Weekly',
  SpecificDay = 'SpecificDay',
}

You, 1초 전 | 2 authors (OYS and others)
@Entity()
export class Todo {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  roomCode: string; // 방 코드

  @Column()
  title: string; // 투두 제목

  @Column({
    type: 'enum',
    enum: TodoFrequency,
    default: TodoFrequency.Daily,
  })
  frequency: TodoFrequency; // 반복 빈도

  @Column()
  dayOfWeek: number; // 요일 (0 = 일요일, 1 = 월요일, ..., 6 = 토요일)
}

@Entity()
export class Calendar {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  roomCode: string; // 방 코드

  @OneToMany(() => Event, (event) => event.calendar)
  events: Event[];
}

...
@Entity()
export class Event {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  title: string;

  @Column({
    type: 'enum',
    enum: EventType,
    default: EventType.Personal,
  })
  type: EventType;

  @Column()
  startDate: Date;

  @Column()
  endDate: Date;
}

@ManyToOne(() => Calendar, (calendar) => calendar.events)
calendar: Calendar;
}
```

```
@Controller('accountbooks')
export class AccountBookController {
  constructor(private readonly accountBookService: AccountBookService) {}

  @Post()
  async addAccountBook(@Body() accountBookData: AccountBook) {
    return this.accountBookService.addAccountBook(accountBookData);
  }

  @Put(':id')
  async updateAccountBook(@Param('id') id: number, @Body() accountBookData: Partial<AccountBook>) {
    return this.accountBookService.updateAccountBook(id, accountBookData);
  }

  @Delete(':id')
  async deleteAccountBook(@Param('id') id: number) {
    return this.accountBookService.deleteAccountBook(id);
  }

  @Get('room/:roomCode')
  async findAccountBooksByRoomCode(@Param('roomCode') roomCode: string) {
    return this.accountBookService.findAccountBooksByRoomCode(roomCode);
  }

  @Get(':id')
  async getAccountBookById(@Param('id') id: number) {
    return this.accountBookService.getAccountBookById(id);
  }
}
```

**Entity
(Schema)**

Controller

AFTER PLAN

차후 계획

백엔드 - TDD 도입 이후 API 문서 작성 필요

프론트엔드 - React Query 이용하여 비동기 통신 연결

Github Repository [public으로 전환](#) 후 앱스토어 출시 예정