Опис проекту

Проект "Система управління транзакціями" - це Flutter додаток, який дозволяє користувачам додавати, редагувати та видаляти транзакції, а також аналізувати їх за допомогою графіків та статистики.

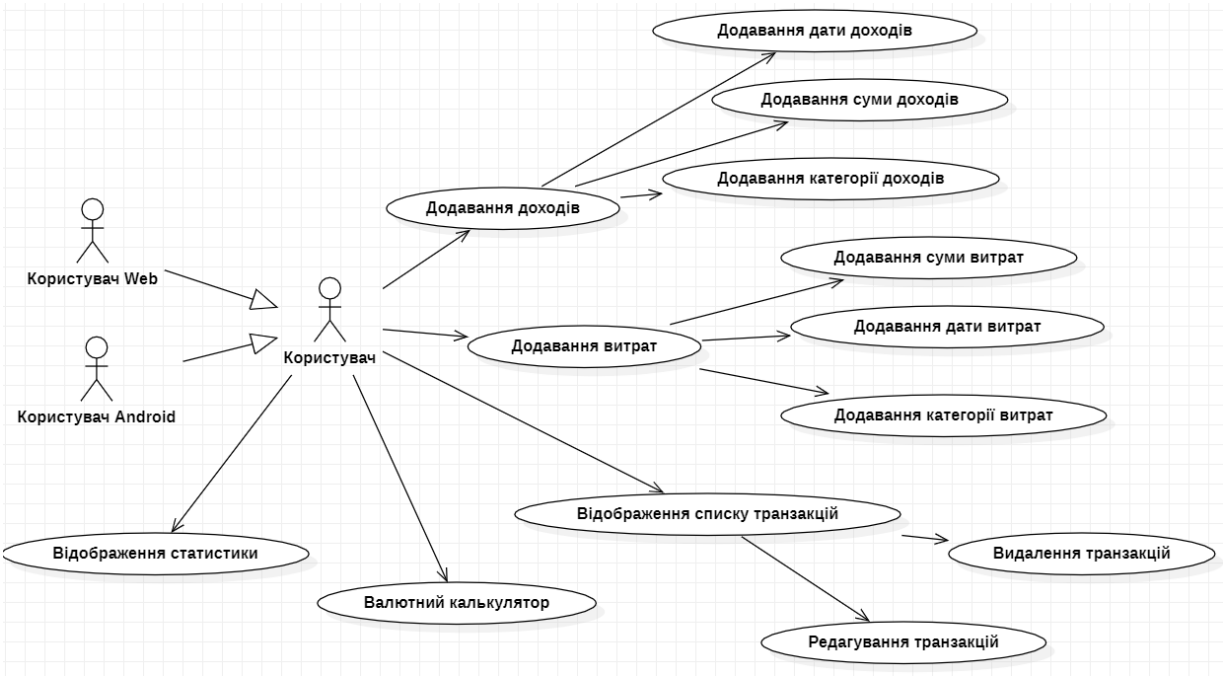
Мета проекту

Метою проекту є створення зручного та ефективного інструменту для керування фінансами, який допоможе користувачам контролювати свої витрати та надходження.

Огляд можливостей

Проект "Система управління транзакціями" надає такі можливості:

- Додавання, редагування та видалення транзакцій.
- Перегляд історії транзакцій з відображенням дати, категорії, суми та опису.
- Аналіз витрат та надходжень за допомогою графіків та статистики.



Встановлення

Для встановлення проекту виконайте наступні кроки:

1. Склонуйте репозиторій з кодом додатку.
2. Відкрийте термінал та перейдіть у папку з проектом.
3. Виконайте команду `flutter pub get`, щоб встановити всі необхідні пакети.
4. Запустіть додаток за допомогою команди `flutter run`.

Залежності та версії підтримуваних бібліотек

Додаток використовує Flutter SDK та ряд додаткових пакетів залежностей. Версії пакетів можуть бути знайдені у файлі `pubspec.yaml`.

Структура проекту

Структура проекту має такий вигляд:

`lib/`: Головна тека з вихідним кодом додатку.

`test/`: Тека з тестовими сценаріями.

```
project_root/
├── lib/
│   ├── main.dart                      # Вхідна точка програми
│   ├── screens/                       # Папка для екранів
│   │   ├── main_screen.dart           # Головний екран
│   │   ├── add_income_screen.dart     # Екран додавання доходів
│   │   ├── add_expence_screen.dart    # Екран додавання витрат
│   │   ├── edit_transaction_screen.dart  # Екран для керування транзакціями
│   │   ├── statistics_screen.dart     # Екран для перегляду статистики
│   │   ├── transaction_screen.dart    # Екран для керування транзакціями
```

```
│   │   └── ...                                  # Інші екрани
│   ├── widgets/                                 # Папка для віджетів
│   │   ├── bottom_navigation_bar.dart           # Віджет для списку завдань
│   │   └── ...                                  # Інші віджети
│   └── utils/                                   # Папка для допоміжних утиліт
│       ├── database_helper.dart        # Допоміжний клас для роботи з базою даних
│       ├── api_helper.dart             # Клас для взаємодії з API
│       ├── statistic_helper.dart       # Допоміжний клас для роботи зі статистикою
│       └── …
├── test/                                        # Папка для тестів
├── pubspec.yaml                                 # Файл конфігурації проекту та залежностей
└── ...
```

Класи проекту

AddExpenseScreen

Мета: Цей клас відповідає за додавання нових витрат.

Атрибути:

- amountController, dateController, descriptionController: контролери для введення суми, дати та опису витрати.
- dbHelper: об'єкт класу DatabaseHelper для взаємодії з базою даних.
- selectedCategory: обрана категорія витрати.

Методи:

- build(BuildContext): Widget: метод, який будує і повертає екран додавання витрат.

AddIncomeScreen

Мета: Цей клас відповідає за додавання нових доходів.

Атрибути:

- amountController, dateController, descriptionController: контролери для введення суми, дати та опису доходу.
- dbHelper: об'єкт класу DatabaseHelper для взаємодії з базою даних.
- selectedCategory: обрана категорія доходу.

Методи:

- build(BuildContext): Widget: метод, який будує і повертає екран додавання доходів.

EditTransactionScreen

Мета: Цей клас відповідає за редагування існуючих транзакцій (витрат або доходів).

Атрибути:

- databaseHelper: об'єкт класу DatabaseHelper для взаємодії з базою даних.
- _amountController, _dateController, _descriptionController: приватні контролери для введення суми, дати та опису транзакції.
- _selectedCategory, _selectedDate: обрана категорія та дата транзакції.

Методи:

- initState(), build(BuildContext): Widget, dispose(): методи для ініціалізації, побудови та видалення класу.

MainScreen

Мета: Цей клас відповідає за головний екран додатку, на якому відображається статистика та основні дії користувача.

Атрибути:

- databaseHelper: об'єкт класу DatabaseHelper для взаємодії з базою даних.
- apiHelper: об'єкт класу ApiHelper для взаємодії з API.

- currentBalance, totalIncome, totalExpenses, selectedCurrency, currencyRate: дані про баланс, загальний дохід, загальні витрати, обрану валюту та курс валют.

Методи:

- initState(), _fetchData(), build(BuildContext): Widget, _showCurrencySelection(BuildContext): методи для ініціалізації, отримання даних, побудови та відображення валютного вибору.

StatisticsScreen

Мета: Цей клас відповідає за відображення статистики про доходи та витрати.

Атрибути:

- databaseHelper: об'єкт класу DatabaseHelper для взаємодії з базою даних.
- statisticHelper: об'єкт класу StatisticHelper для обробки статистичних даних та побудови діаграм.
- currentChartType: тип діаграми (наприклад, кругова, лінійна).
- startDate, endDate: дати початку та завершення періоду статистики.

Методи:

- initState(), setInitialDates(), build(BuildContext): Widget: методи для ініціалізації, встановлення початкових дат та побудови екрану.

TransactionScreen

Мета: Цей клас відповідає за відображення списку транзакцій (витрат та доходів).

Атрибути:

- databaseHelper: об'єкт класу DatabaseHelper для взаємодії з базою даних.

Методи:

- build(BuildContext): Widget: метод, який будує і повертає екран зі списком транзакцій.

ApiHelper

Мета: Цей клас відповідає за взаємодію з API для отримання курсу валют.

Методи:

- getCurrencyRate(String): Future<double>: метод для отримання курсу валют.

DatabaseHelper

Мета: Цей клас відповідає за взаємодію з базою даних SQLite для збереження та обробки транзакцій.

Атрибути:

- _database: об'єкт бази даних.
- _dbName: назва бази даних.
- _expenseTable, _incomeTable: назви таблиць для витрат та доходів.
- storage: об'єкт FlutterSecureStorage для збереження даних.
- startDate, endDate: дати початку та завершення періоду.

Методи:

- Методи для створення таблиць, додавання, оновлення та видалення записів, отримання статистичних даних та інших операцій з базою даних.

StatisticHelper

Мета: Цей клас відповідає за обробку та побудову статистичних даних та діаграм.

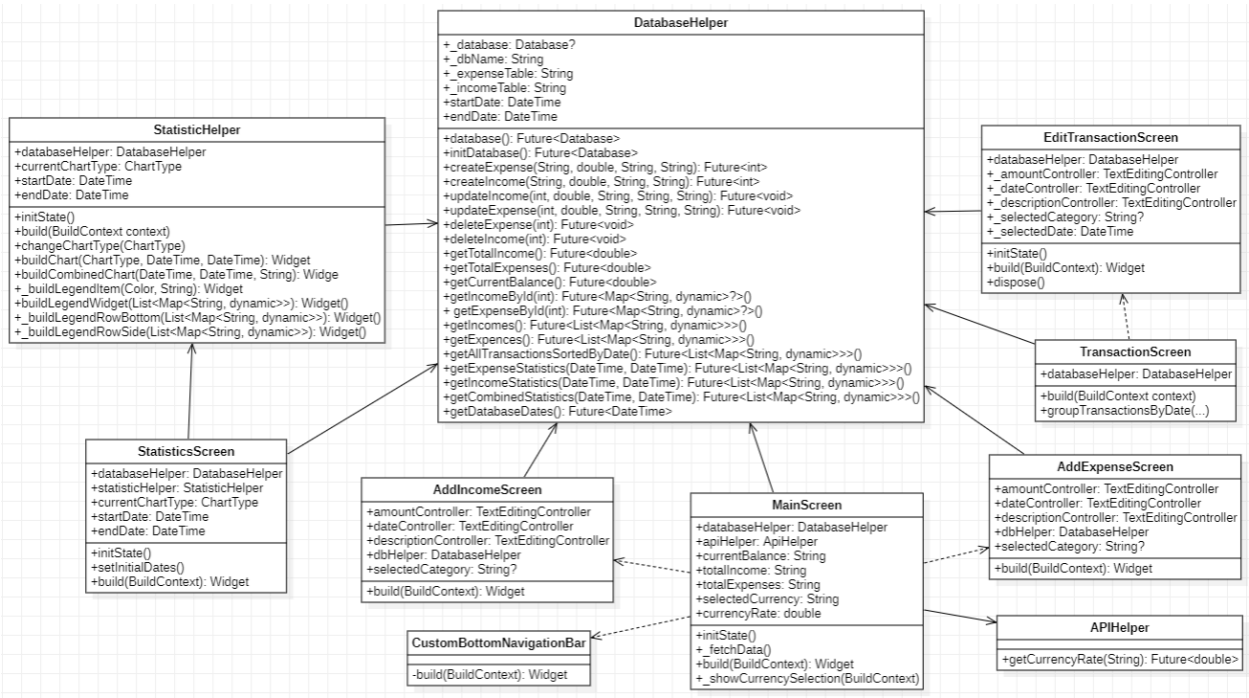
Атрибути:

- databaseHelper: об'єкт класу DatabaseHelper для взаємодії з базою даних.

- currentChartType: тип діаграми.

Методи:

- Методи для зміни типу діаграми, побудови діаграм та інших операцій з обробки статистичних даних.



AddExpenseScreen і AddIncomeScreen класи використовують клас DatabaseHelper для роботи з базою даних. Тому вони мають асоціацію з DatabaseHelper.

EditTransactionScreen та TransactionScreen також використовує DatabaseHelper для отримання списку транзакцій, оновлення даних про транзакцію, тому він також має асоціацію з DatabaseHelper.

MainScreen використовує DatabaseHelper і ApiHelper для отримання та відображення даних, тому він має асоціацію з обома цими класами.

StatisticsScreen також використовує DatabaseHelper для отримання даних для побудови статистики. Також він використовує StatisticHelper для обробки даних та побудови графіків, тому має асоціації з обома цими класами.

TransactionScreen залежить від EditTransactionScreen, оскільки воно відображається після того, як користувач вибрав редагування транзакції на TransactionScreen.

MainScreen залежить від AddIncomeScreen та AddExpenseScreen, оскільки воно може відображати їхні функції під час взаємодії з користувачем.

Архітектура

Проект має комбіновану архітектуру: клієнт-серверна з локальним сховищем, де клієнтська частина написана на мові програмування Flutter, а серверна частина написана на мові програмування Dart.

Ця архітектура поєднує в собі переваги клієнтської та серверної архітектур, забезпечуючи ефективну взаємодію між клієнтом (користувачем) та сервером, а також зберігання та керування даними локально на пристрої користувача.

Клієнтська частина (Клієнт): це частина додатка, що відповідає за відображення інтерфейсу користувача та інтеракцію з ним. Вона взаємодіє з сервером для отримання даних, таких як курси валют, а також з локальною базою даних для зберігання та керування даними, такими як історія транзакцій.

Серверна частина (Сервер): програма може використовувати віддалене API для отримання актуальних курсів валют та оновлення їх у вашій локальній базі даних. Серверна частина забезпечує доступ до цих даних через віддалені запити.

Локальне сховище (База даних): локальна база даних та допоміжні утиліти (наприклад, `database_helper.dart`) використовуються для зберігання та керування локальними даними, такими як історія транзакцій, безпосередньо на пристрої користувача. Це дозволяє прискорити доступ до даних та підвищити продуктивність за рахунок роботи з даними локально.

Ця архітектура забезпечує ефективну взаємодію з сервером для отримання актуальних даних, а також швидкий доступ до локальних даних, забезпечуючи зручність і продуктивність для користувача.

3. Зв'язки між компонентами:

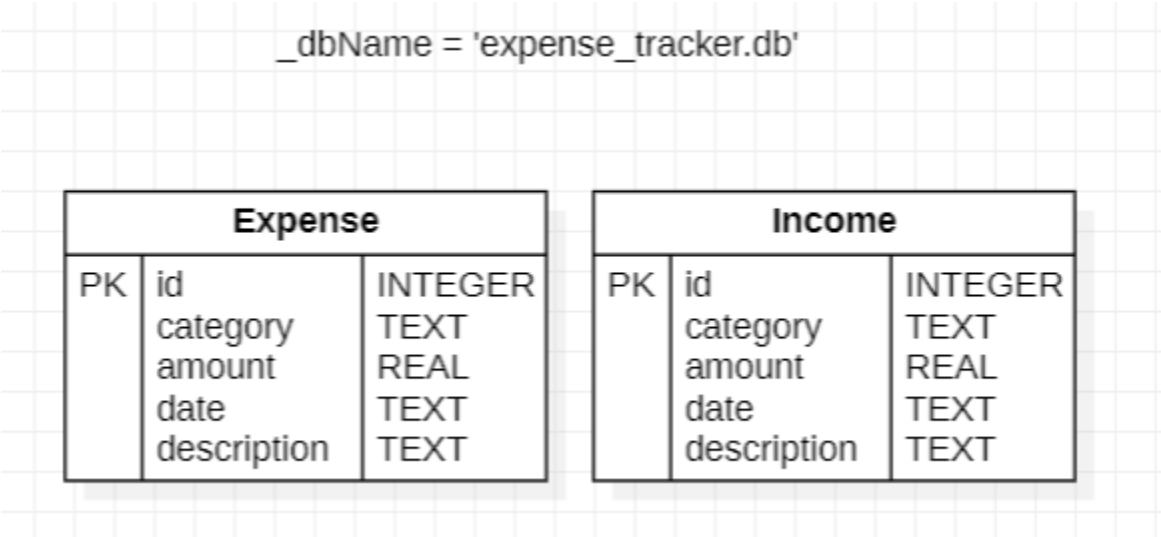
- Екрани використовують сервіси для отримання та обробки даних, що відображаються на екранах.

- Сервіси (наприклад, DatabaseHelper, StatisticHelper) використовуються для роботи з даними та виконання бізнес-логіки.

- APIHelper використовується для взаємодії з сервером, відправляючи HTTP-запити та обробляючи відповіді.

База даних

База даних складається з двох таблиць: "Expense" і "Income".



Таблиця "Expense" містить наступні поля:

- id: унікальний ідентифікатор витрати (ціле число),
- category: категорія витрати (текстове значення),
- amount: сума витрати (дійсне число),

- date: дата витрати (текстове значення),
- description: опис витрати (текстове значення).

Таблиця "Income" містить аналогічні поля:

- id: унікальний ідентифікатор доходу (ціле число),
- category: категорія доходу (текстове значення),
- amount: сума доходу (дійсне число),
- date: дата доходу (текстове значення),
- description: опис доходу (текстове значення).

Тестування

Тестування складається з 4 частин: модульне тестування функцій бази даних, обчислення статистики, UI-тестування екранів додатку та тестування взаємодії з зовнішнім API:.

1. Модульне тестування функцій бази даних:

- Створюється об'єкт DatabaseHelper.

- Налаштовуються тестові умови, зокрема, ініціалізація бази даних та створення записів про витрати та доходи.

- Перевіряються різні операції з базою даних, такі як додавання, оновлення, видалення записів, отримання загальних сум та статистики.

- Використовуються методи test та expect для проведення тестування різних функцій бази даних.

2. UI-тестування екранів додатку:

- Створюються об'єкти екранів додатку, такі як MainScreen, StatisticsScreen, AddExpenseScreen, AddIncomeScreen.

- Перевіряється, чи відображаються необхідні елементи на кожному екрані.

- Використовуються методи testWidgets та expect для проведення UI-тестування.

3. Тестування обчислення статистики

-Створюється об'єкт DatabaseHelper.

-Визначаються початкова та кінцева дата для обчислення статистики.

-Здійснюється виклик функцій getExpenseStatistics, getIncomeStatistics та getCombinedStatistics з передачею в них визначених дат.

-Перевіряється, чи отримані результати не є null та чи містять вони очікувані значення.

-Використовуються методи test та expect для проведення тестування обчислення статистики.

4. Тестування взаємодії з зовнішнім API:

-Створюється об'єкт ApiHelper.

-Викликається метод getCurrencyRate з передачею символьного коду валюти.

-Перевіряється, чи метод повертає очікуване значення курсу валюти або викидає виняток у випадку помилки сервера чи неправильної відповіді.

-Використовуються методи test та expect для проведення тестування взаємодії з зовнішнім API.

Використання

Для використання проекту додайте дані про транзакції, редагуйте їх за необхідності та аналізуйте витрати та надходження за допомогою статистики та графіків.