Dokumentation

Projekt B - Debt Collector 2.0

Projektleiter: Prof. Dr. Andreas Plaß
Studiengang: Media Systems WS19/20
Team:
André Voroschilin (Matrikel Nr.: 2298891)
Burhan Kiran (von der Uni HH)
Versionskontrolle: https://github.com/CaptainWeasle/Projekt-B-2.0

Die Idee

Die Grundidee war es eine Schulden App für mobile Geräte zu erstellen. Es soll auf allen großen mobilen Betriebssystemen (Android und iOS) laufen können damit wir, als auch Verwandte und Bekannte die App verwenden können. Die Hauptmotivation war es eine App für uns zu entwickeln, weil die Alternativen auf dem Markt entweder Geld kosten oder Werbung beinhalten.

Das Konzept

Von Anfang an war geplant, dass sich der Debt Collector 2.0 nur die grundlegendsten Features beinhalten soll. Man soll in einem einfachen UI in der Lage sein eine Schuld die man hat zu speichern. Dabei soll natürlich der Name des Schuldners und der Betrag gespeichert und in einer Liste angezeigt werden.

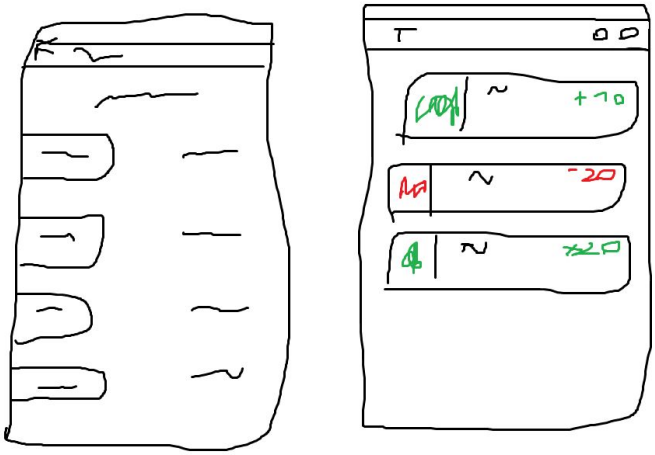


Abb. 1: Designkonzepte

Am Anfang des Projektes haben wir etwas gebrainstormed und mögliche Designs ausgedacht. Wir waren uns jedoch nicht wirklich einig und haben angefangen zu entwickeln, ohne uns auf ein festes Design zu einigen.

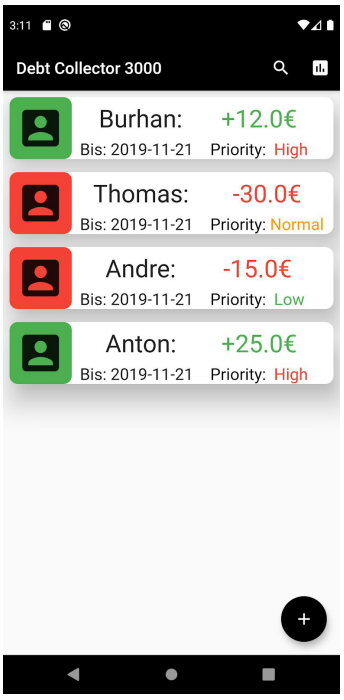


Abb. 2: altes Design

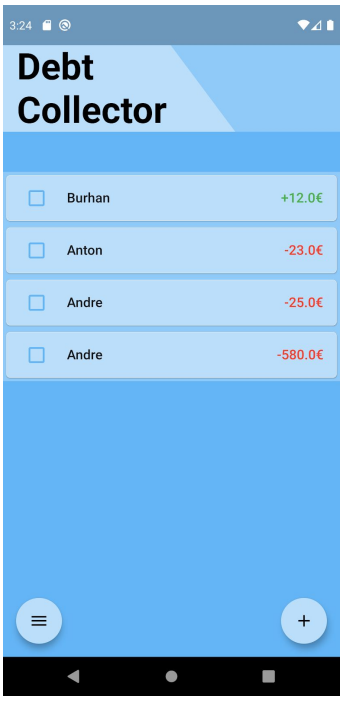


Abb. 3: aktuelles Design

Mit der Zeit haben wir mehrere Features hinzufügen können. Die aktuelle Version der App ermöglicht es sehr viel mehr Daten als nur Namen und Betrag zu speichern und bietet auch QoL Verbesserungen. Eine Liste der Features:

- Schuld speichert Name, Betrag, Datum an dem die Schuld erstellt wurde, Datum an dem die Schuld fällig ist, Priorität und wer wem schuldet.
- Schulden werden lokal gespeichert und verschwinden nach Schließen der Anwendung nicht.
- Schulden sind per RadioButton einfach als beglichen markierbar ohne sie zu löschen.
- Schulden sind per Swipe von rechts nach links einfach zu löschen.
- Detaillierte Ansicht bietet alle Infos zur Schuld, als auch eine optionale Beschreibung.
- Eine Zusammenfassung der Schulden, wie viel man bekommt, wie viel man zahlen muss.
- Zusammenfassung beinhaltet nur offene und nicht beglichene Schulden
- Knöpfe um beglichene Schulden oder alle Schulden zu löschen.
- Einfaches und intuitives Design.
- [BETA] Es gibt eine Suchfunktion, welche noch nicht fertig ist.

Die Plattform

Die Plattform unserer Wahl war Flutter, Google's UI Toolkit um plattformübergreifende Applikationen zu machen. Die verwendete Programmiersprache ist Dart. Wir haben während der Entwicklung auf Windows und MacOS mit jeweils Android Studio als auch Visual Studio Code auf Android und iOS Emulatoren an der App gearbeitet und sie getestet.

Der Code

Externe Bibliotheken und Imports

```
dependencies:
 sqflite: ^1.1.7+1  //Datenbank
 path_provider: ^1.4.0 //für lokale Datenabfrage
 datetime_picker_formfield: ^0.4.3 //UI um Datum auszuwählen
 flutter:
   sdk: flutter
```

Abb. 4: benutzte Dependencies, zu finden in der pubspec.yaml

Das Pattern

Um Unsere App anständig zu gliedern und einen einfacheren Workflow zu haben mussten wir uns für ein Pattern entscheiden. Wir haben das von Google für Flutter empfohlene Bloc Pattern entschieden. Es ist ein auf Streams basiertes Pattern welches die Logik von den Daten von der UI trennen soll.

In unserem Fall ist der Network Provider unsere Datenbank und zwischen Repository und Datenbank befindet sich unser DAO (Data Access Object) welches grundlegende Datenbank Methoden bereitstellt.

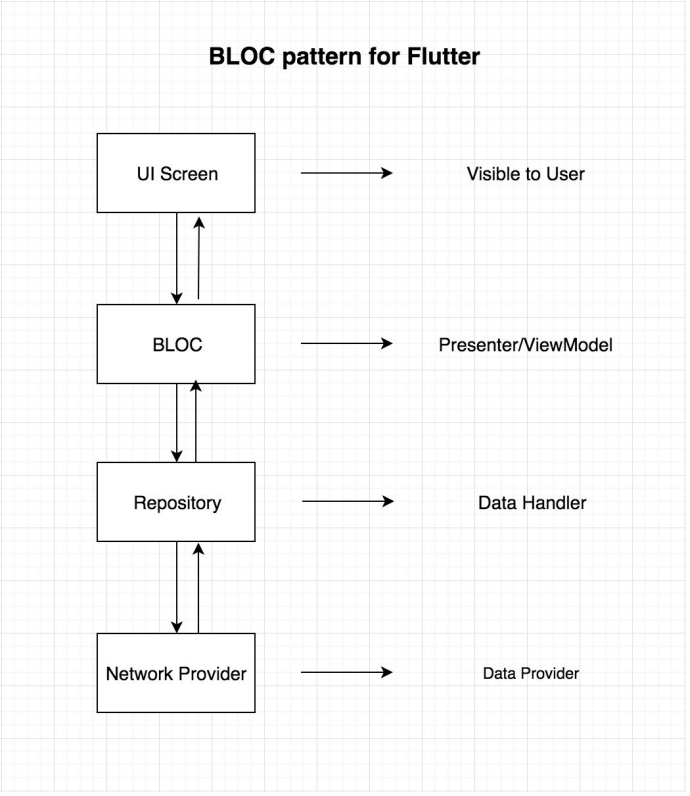


Abb. 5: Standard Bloc Pattern

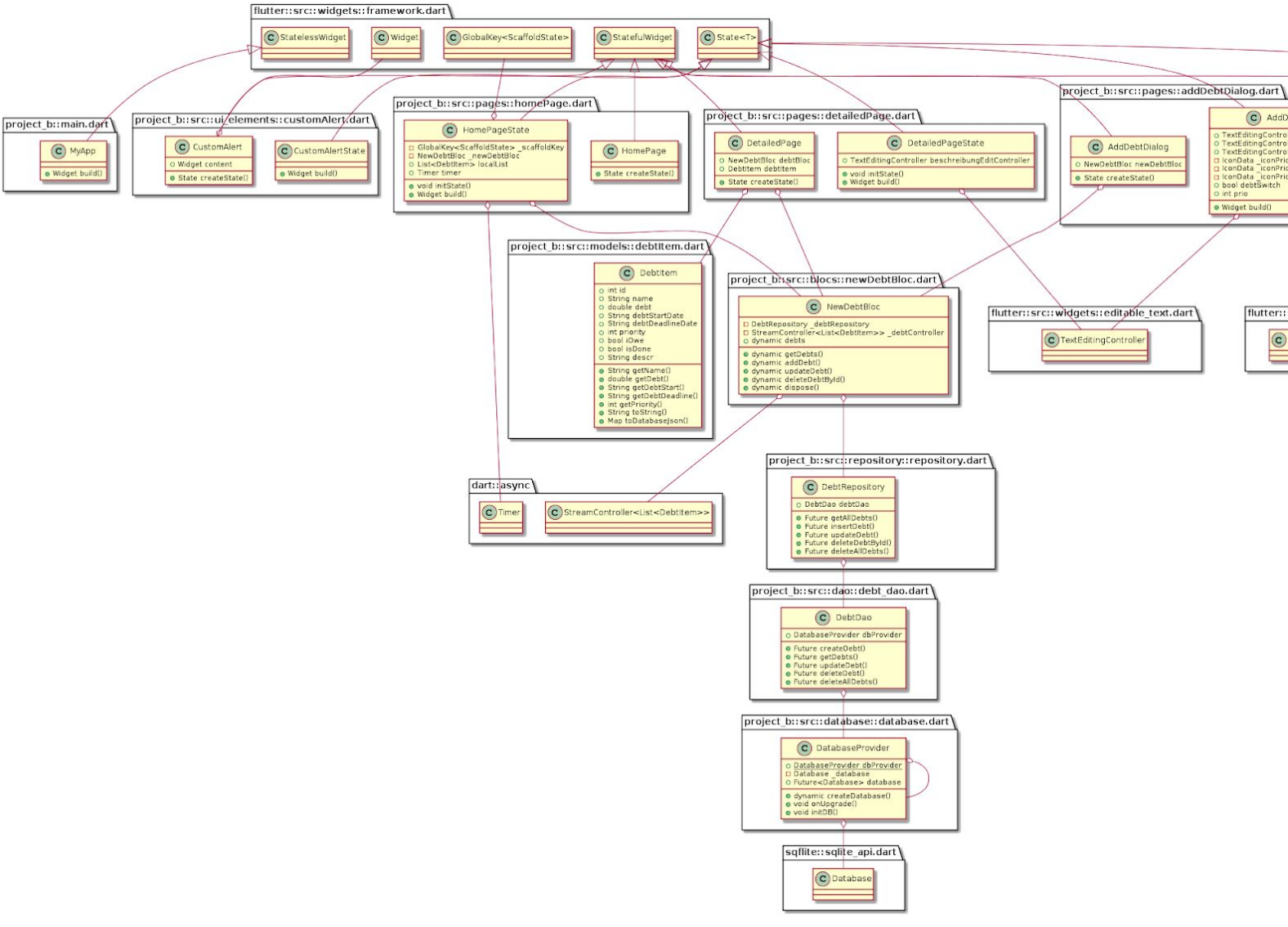
Die Datenbank

Um die Schulden nach dem Wiederaufrufen der Anwendung behalten zu können, müssen sie lokal gespeichert werden. Hierfür haben wir eine SQLite Datenbank verwendet. Dafür gab es einen entsprechenden Plugin für Flutter namens SQFlite, welcher auch oben in den Dependencies erwähnt wird.

Die Struktur

Die App besteht aus vier Grundkomponenten, das sind die *HomePage*, die *DetailedPage*, das *AddDebtDialog* und das *SummaryDialog*. Jedes dieser Klassen erbt von *StatefulWidget* und beinhaltet das UI.
Für die Logik sorgt der *NewDebtBloc*. Dieser besitzt eine Instanz von *Repository*, welche entsprechend die Methoden aus der *DebtDao* und des *DatabaseProviders* bekommt. Der *DatabaseProvider* sorgt um die Datenbank und der *DebtDao* erhält die wichtigen Operationen um mit der Datenbank arbeiten zu können, wie zum Beispiel die Methoden *createDebt()*, *updateDebt()* und *deleteDebt().*
Die *HomePage* hat und *DetailedPage* besitzt dann jeweils eine Instanz des *NewDebtBlocs* und kann dadurch auf die Datenbank zugreifen.

Abb. 6: UML Diagramm der kompletten App.



Fazit

Das Projekt ist unserer Meinung ein voller Erfolg geworden. Wir haben alles geschafft was wir in einer Schulden App erwarten und noch Ideen für mehr. Lediglich das Design ist noch umstritten.

Die Arbeit am Projekt ist auch überwiegend problemlos vorangegangen. Weil es Erfahrungs Unterschiede zwischen den Gruppenmitgliedern gab, haben wir die Arbeit zwischen UI und Logik unterschieden. Das hat auch die meiste Zeit gut geklappt, aber im Endeffekt hat jeder etwas an Allem arbeiten können.

Burhan hat noch keine Erfahrung mit größeren Projekten und auch nicht mit Flutter oder der Programmiersprache Dart gehabt. Für ihn war das Projekt sehr interessant und er konnte sehr viel lernen. Nicht nur, dass er mit Flutter arbeiten konnte, auch erste Erfahrungen mit Git und externen Bibliotheken können aufgezeichnet werden.

André ist schon etwas erfahren in Flutter und hat auch schon mehrere größere Projekte gemacht. In vielen verschiedene älteren Projekten konnte er sich an neue Konzepte in Flutter ausprobieren, wie z.B. Pattern, Bluetooth Integration, Zugriff auf Sensoren und Netzwerkanbindung. Dieses Projekt ist in dieser Hinsicht nicht anders. Wir hatten die Chance mit eine lokale Datenbank anzubinden und mit dieser zu arbeiten. Dies sind sehr wertvolle Erfahrung, da man jetzt eine eigenständige App erstellen kann.

An der App wird definitiv weitergearbeitet und sie wird auf Android (und vielleicht auch bald iOS) Geräten von Freunden und Verwandten laufen.