Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Multiplatformní aplikace pro správu osobních financí

2025

Vojtěch Netrh

Vedoucí práce:
doc. RNDr. Jan Konečný, Ph.D.

Studijní program: Informatika,
Specializace: Programování a vývoj software

Bibliografické údaje

Autor:	Vojtěch Netrh
Název práce:	Multiplatformní aplikace pro správu osobních financí
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2025
Studijní program:	Informatika, Specializace: Programování a vývoj software
Vedoucí práce:	doc. RNDr. Jan Konečný, Ph.D.
Počet stran:	24
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Vojtěch Netrh
Title:	Cross-platform application for personal finance management
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2025
Study program:	Computer Science, Specialization: Programming and Software Development
Supervisor:	doc. RNDr. Jan Konečný, Ph.D.
Page count:	24
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Ukázkový text závěrečné práce na Katedře informatiky Přírodovědecké fakulty Univerzity Palackého v Olomouci, který je zároveň dokumentací stylu pro text práce v LaTeXu. Zdrojový text v LaTeXu je doporučeno použít jako šablonu pro text skutečné závěrečné práce studenta.

Synopsis

Sample text of thesis at the Department of Computer Science, Faculty of Science, Palacký University Olomouc and, at the same time, documentation of the LaTeX style for the text. The source text in LaTeX is recommended to be used as a template for real student's thesis text.

Klíčová slova: Flutter; Dart; multiplatformní; osobní finance

Keywords: Flutter; Dart; cross-platform; personal finance

Děkuji panu doc. RNDr. Janu Konečnému Ph. D. za cenné rady a podněty při tvorbě práce. Svým blízkým za podporu během celého bakalářského studia.

Odevzdáním tohoto textu jeho autor místopřísežně prohlašuje, že celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

1 Úvod **7**
- 1.1 Požadavky . . . 7

2 Přehled existujících řešení **8**
- 2.1 1Money . . . 8
- 2.2 Cashew . . . 8
- 2.3 Wallet . . . 9
- 2.4 Money Manager Ex . . . 9
- 2.5 Homebank . . . 12

3 Použité technologie **12**
- 3.1 Dart . . . 13
- 3.2 Flutter . . . 13
- 3.3 Material Design . . . 14
- 3.4 SQLite . . . 15
- 3.5 Balíčky . . . 15

4 Programátorská příručka **17**
- 4.1 Architektura aplikace . . . 17
- 4.2 Multiplatformní část projektu . . . 18
- 4.3 Uchovávání stavu . . . 18
- 4.4 Implementace responzivního designu . . . 19
- 4.5 Zajímavosti při tvorbě práce . . . 19

5 Uživatelská příručka **20**
- 5.1 První spuštění . . . 20
- 5.2 Úvodní obrazovka . . . 20
- 5.3 Zobrazení a úprava transakcí . . . 20
- 5.4 Využití reportů . . . 21
- 5.5 Nastavení aplikace . . . 21
  - 5.5.1 Přizpůsobení vzhledu . . . 21
  - 5.5.2 Přizpůsobení měn . . . 21
  - 5.5.3 Úpráva kategorií . . . 21
- 5.6 Export dat . . . 21

6 Možná rozšíření aplikace **21**

Závěr **22**

Conclusions **23**

Literatura **24**

Seznam obrázků

1 Aplikace 1Money 9
2 Aplikace Cashew 10
3 Aplikace Wallet 11
4 Aplikace Money Manager Ex 11
5 Aplikace Homebank 12
6 Skeleton loading a načtená obrazovka - převzato z [15] 16
7 Navigační prvky - převzato z [17] 19

1 Úvod

Peníze se vyskytují všude kolem nás a chceme-li nebo ne, hrají v našich životech podstatnou roli. Z hlediska osobního pohledu jednoho člověka je užitečné mít ve svých financích pořádek. Můžeme tak ušetřit peníze, případně je efektivněji využívat. V neposlední řadě by měl člověk vědět, kolik peněz by potřeboval v případě výpadku příjmů a jaká má být jeho „železná rezerva“.

V dnešní době již většina lidí používá internetové bankovnictví, která dost často poskytuje alespoň základní statistiky o našem nakládání s financemi. Na druhou stranu lidé mívají účty u více bank a reporty o stavu financí nejsou dostatečně přizpůsobitelné.

Kromě toho, že peníze se podaří člověku ušetřit je vhodné s nimi potom dále nakládat. Můžeme je ihned utratit (což není dlouhodobě vhodná varianta), spořit si nebo dále investovat. Spořit si je možné na řadu věcí – nové bydlení, automobil či vytvoření dostatečného objemu pěnez pro založení vlastního podnikání. Pro spoření existují dva základní způsoby jak peníze ukládat. Prvním z nich je mít je na běžném či spořícím účtu a jejich hodnota bude v čas pořád zhruba stejná. Druhým přístupem je aktivně investovat a snažit se pomocí nich vydělat další peníze.

1.1 Požadavky

Napsat něco jiného než minule, vhodný začátek podkapitoly.

- **Jednoduchost více než komplexní funkce** – snažit se implementovat dostatečný počet funkcí, avšak nezahltit uživatele příliš mnoha různými nastaveními, které mu přidají práci při volení parametrů. Pokud je pro uživatele pohodlné zapisovat transakce již v průběhu dne, aniž by měl pocit, že u aplikace tráví moc času, je to ideální scénář.

- **Mobilní telefony i počítače** – obsáhnout zařízení na různé škále velikosti přináší dostupnost aplikace pro více uživatelů. Někdo rád evidenci financí dělá na denní bázi (obvykle pomocí mobilního telefonu), někdo naopak až měsíc zpětně. Na zpracování více dat najednou je jistě počítač s větším displejem vhodnější volbou.

- **Zahraniční měny** – V dnešní době velká část lidí cestuje jak pracovně, tak i za dovolenou. Placení kartou v zahraničí bez nutnosti dopředu směnit peníze se stalo standardem. Jelikož každá banka používá jiný směnný kurz měny, měl by jít libovolně upravit, případně automaticky synchronizovat z internetu.

- **Export dat** – uživatel by měl mít možnost exportovat data v běžně používaných formátech jako je CSV nebo JSON. Export považuji za důležitý z důvodu přenesení dat do jiného prostředí či aplikace.

2 Přehled existujících řešení

K danému tématu již existuje řada aplikací nabízejících nástroje pro správu financí. Každé řešení přistupuje k problému jiným způsobem.

Nejblíže je z hlediska uživatele poskytnutí základních nástrojů přímo v bankovní aplikaci. Z mého pohledu je to nejméně vhodné řešení hned z několika důvodů:

- obvykle má člověk účet u více bank (aplikace jedné z nich neposkytuje přehled o celkových financích),
- jen část (i když v dnešní době většinová) transakcí je prováděna pomocí platební karty,
- uživatel nemusí chtít mít všechny pohyby na účtu zahrnuty do celkové analýzy.

Aplikace třetích stran v tomto segmentu, nabízí různé možnosti. Mobilní aplikace bývají obvykle jednodušší a s přívětivějším uživatelským rozhraním. Zatímco desktopové aplikace mají uživatelské rozhraní spíše starší, avšak poskytují nepřebernou škálu funkcí.

2.1 1Money

Ještě na konci roku 2024 byla aplikace 1Money dostupná na obě mobilní platformy, o pár měsíců později už pouze na iOS. Z mnou vyzkoušených aplikací největší favorit pro denní užití. Aplikace mě zaujala svojí obrazovkou *Categories*. Středem obrazovky je prstencový graf zobrazující poměr utracených peněz podle kategorií a v jeho středu se ještě nachází dva údaje - stav příjmů a výdajů. Zbytek obrazovky pokrývají kolečka označující kategorie, po jejichž stisknutí je uživateli ihned umožněno přidat transakci s danou kategorií a aktuálním časovým razítkem. Velmi intuitivní a rychlé řešení pro každodenní použití. Umožňuje vytvořit více různých účtů, včetně účtů pro spoření. Z hlediska analýzy a grafů zde najdeme pouze dvě velmi omezené možnosti - již zmíněný prstencový graf kategorií a sloupcový graf znázornění transakcí v čase. Tuto aplikaci jsem si já osobně oblíbil nejvíce.

2.2 Cashew

Multiplatformní aplikace založená na stejné technologii Flutter jako moje aplikace se jmenuje Cashew [1]. Implementuje komponenty z Material Design, ale rozvržení obrazovky si autoři uspořádali podle sebe. Uživateli umožňuje široké možnosti nastavení z hlediska vzhledu (barvy, typ písma, typ ikon či formáty data i čísel). Umožňuje zálohování dat na Google disk i import z csv souborů. Podporuje funkci více účtů. Obsahuje i placenou verzi, které je primárně podporu pro vývojáře.

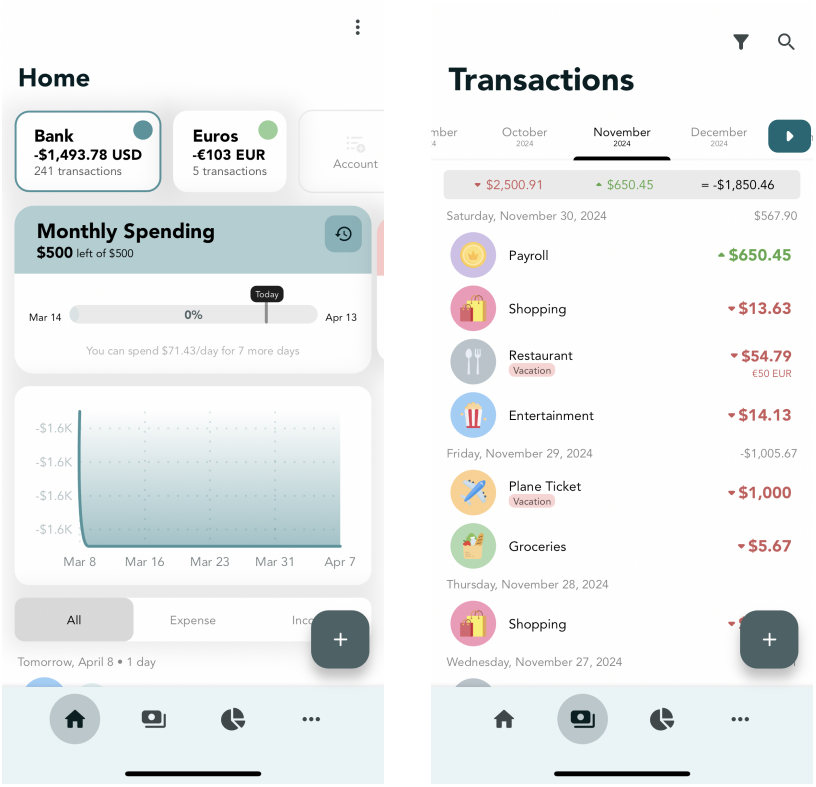
Obrázek 1: Aplikace 1Money

2.3 Wallet

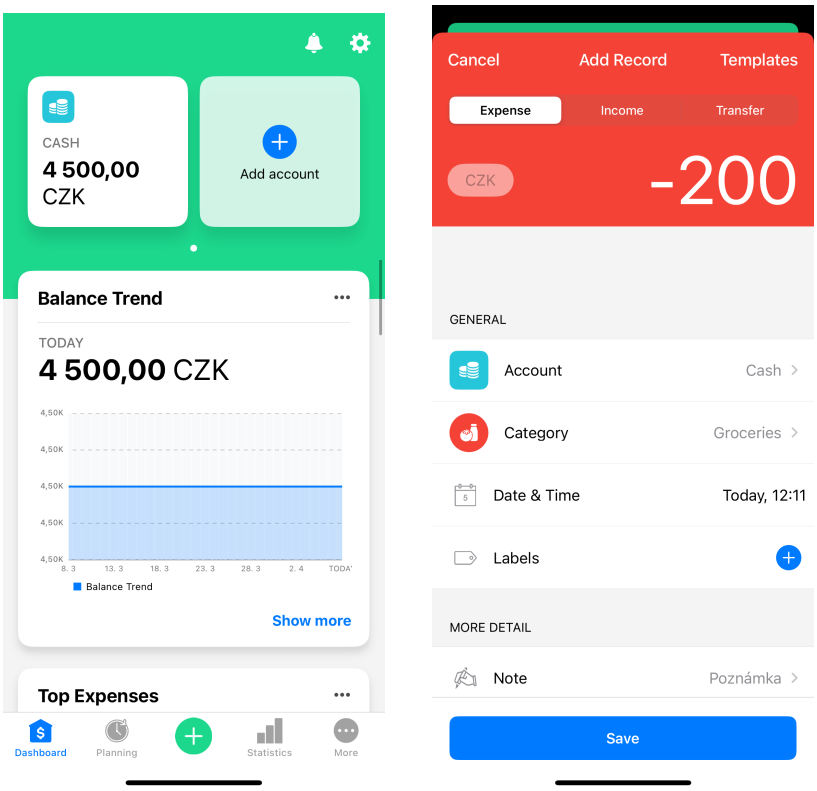
Mobilní aplikace Wallet [2] je dostupná na platformy Android i iOS. Za její stažení uživatel v první fázi nezaplatí. Nabízí všechny základní funkce, co by uživatel očekával – kategorie transakcí, podpora více měn, přidání poznámky. Z hlediska analýzy a grafového zobrazení je na výběr jen velmi málo možností v z mého pohledu nepřehledném uspořádání. Ze zajímavých funkcionalit stojí za zmínku plánované transakce nebo automatická bankovní synchronizace (tato funkce je už placená).

2.4 Money Manager Ex

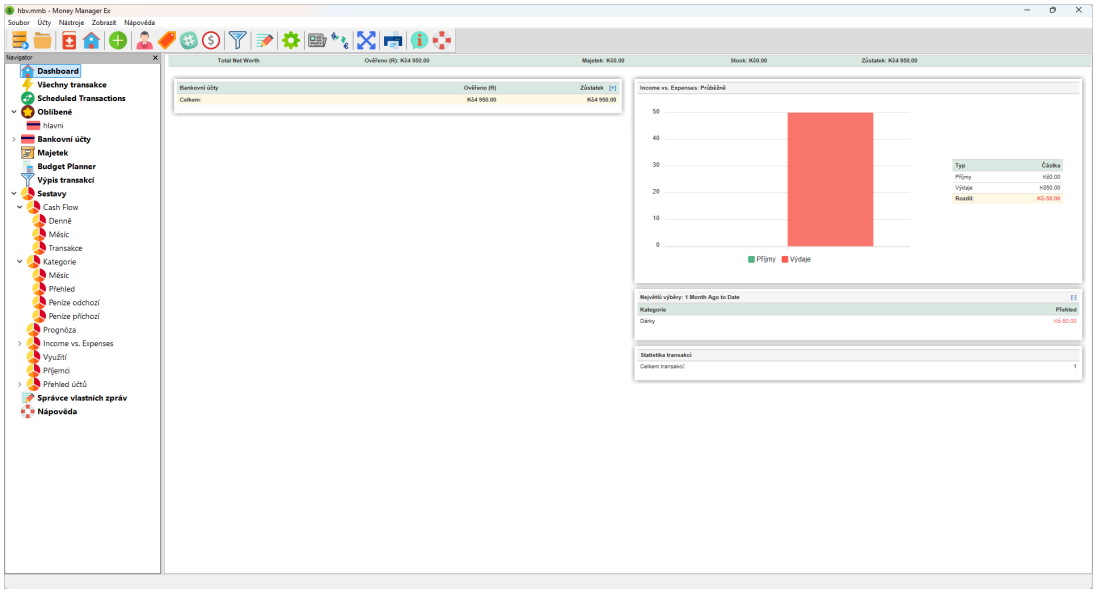
Aplikace dostupná jak na mobilní zařízení (nízký počet stažení [3]) tak především i na desktopové zařízení [4]. Uživatelské rozhraní je na první pohled relativně přívětivé, i když místy obsahuje širokou škálu tlačítek a možností k nastavení. Při prvním spuštění nevyžaduje žádné složité nastavení. Zde se mi velmi líbí velká nabídka grafů k analýze údajů a možnost exportu do PDF pro většinu obrazovek.



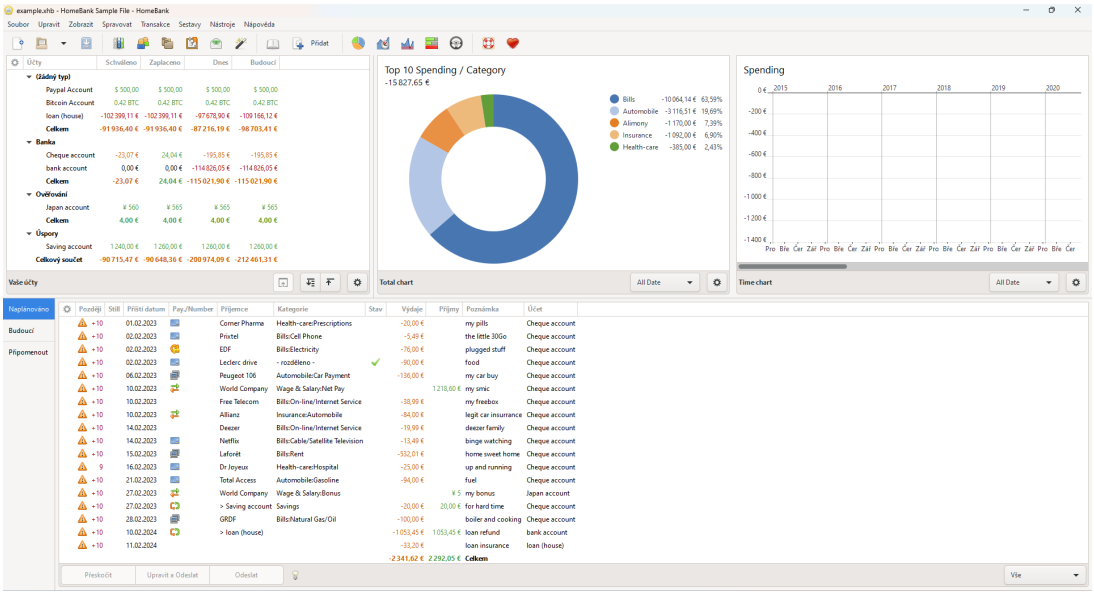
Obrázek 2: Aplikace Cashew



Obrázek 3: Aplikace Wallet



Obrázek 4: Aplikace Money Manager Ex



Obrázek 5: Aplikace Homebank

2.5 Homebank

Desktopová aplikace dostupná na Windows, Linux i macOS [5]. Z hlediska uživatelského komfortu je oproti Money Manager Ex jednodušší na používání. Při prvotním nastavení sice člověk musí vhodně nakonfigurovat svůj účet, v dalších fázích je už používání intuitivní. Centrem aplikace je obrazovka se stavy účtu, přehledem transakcí a jednoduchými grafy. Všechny ostatní funkce se pak otevírají jako nové okno. Uživatelské rozhraní působí moderním a intuitivním dojmem.

Oproti existujícím aplikacím by tato měla nabídnout přívětivější uživatelské rozhraní a ulehčit první použití. Rád bych přinesl i lepší vizualizaci v podobě grafů a zajímavých číselných údajů. V neposlední řadě by měla být spustitelná na libovolné platformně, tak aby uživatel nebyl limitován například při změně mobilního telefonu.

3 Použité technologie

Požadavkem byla multiplatformní vývoj existuje řada technologií [6], které však k tvorbě aplikace přistupují odlišným způsobem. Mezi nejpopulárnější technologie vhodné na tuto práci se řadí *React Native* a *Flutter*. Jelikož jsem za dobu, co se věnuji programování nepřilnul k webovým technologiím, na kterých primárně stojí React Native, zvolil jsem Flutter. Přispěl k tomu fakt, že se více blíží klasickým objektově orientovaným jazykům a jednoduše implementuje *Material Design*, který rád v uživatelském rozhraní vídám.

Aplikace je otestována a řádně funguje na obou mobilních platformách (Android iOS) a jako desktopová Windows aplikace. Ze stejného kódu jde sestavit

i aplikace na macOS a Linux. Tyto platformy jsem však netestoval a nemohu zaručit jejich bezproblémový chod.

3.1 Dart

Objektově orientovaný jazyk Dart [7] pochází od společnosti Google, jehož první verze byla zveřejněna 14. listopadu 2013. Momentálně je jeho aktuální verze 3. Jde o typově bezpečný jazyk, podporující třídy se syntaxí založenou na jazyku C. Může být kompilován do strojového kódu, jazyky JavaScript nebo WebAssembly. Taktéž je základem pro framework Flutter.

Je dodáván se širokou škálou základních knihoven, které umožňují obstarat běžný vývoj. Jako příklad mohu uvést knihovnu `dart:async`, která umožňuje asynchronní programování za využití tříd jako je `Future` nebo knihovna `dart:io` pro práci se souborovým systémem či HTTP protokolem.

Technologie jazyka Dart umožňují spustit kód dvěma způsoby. První z nich je určen pro mobilní a desktopové aplikace. Dart obsahuje virtuální stroj s JIT[1] kompilací, který se používá především ve fázi vývoje. Tento způsob umožňuje tzv. inkrementální rekompilaci jejíž výhodou je funkcionalita *hot reload* či nástroje *DevTools* pro aktuální metriky ladění. Dále i AOT[2] kompilátor pro tvorbu strojového kódu, který se využívá při sestavování apliakce pro použití v produkci. Druhý způsob se týká webové platformy, kde Dart umožňuje kód přeložit do JavaScript nebo WebAssembly. Opět i zde Dart využívá různé techniky pro ladění kódu a následné produkci. Oba tyto způsoby jsou stejné v tom že potřebují *běhové prostředí Dart*. Toto prostředí se stará o správu paměti pomocí garbage collector, vynucení kontroly typů a správu vláken.

```
1 class Point {
2   final double x;
3   final double y;
4
5   const Point(this.x, this.y);
6
7   bool get isInsideUnitCircle => x * x + y * y <= 1;
8 }
```

Zdrojový kód 1: Ukázka kódu v jazyce Dart

3.2 Flutter

Open source framework Flutter [8] slouží pro tvorbu uživatelských Využívá Dart a je taktéž vyvíjený společností Google. Poprvé zveřejněn byl v květnu 2017. Google sám ho využívá v aplikacích jako je Google Play nebo Goole Earth [9].

[1] Just in time
[2] Ahead of time

Mezi nejznámější aplikace třetích stran používajících Flutter patří Alibaba nebo hra PUBG Mobile [9].

Používá své vlastní vykreslovací jádro [10], které pixely přímo vykresluje na obrazovku. Toto je podstatný rozdíl oproti řadě jiných frameworků, které se spoléhají na vykreslovací jádro dané platformy. Tento přístup umožňuje mít identicky vypadající uživatelské rozhraní napříč všemi podporovanými platformami.

Základní komponentou je *widget*. Ten se dále může skládat z dalších widgetů. Celé uživatelské rozhraní je poté poskládáno z těchto celků. Flutter sám o sobě poskytuje dva typy předdefinovaných widgetů – Material Design (viz kapitola 3.3) widgety a Cupertino widgety. Přestože oba mají svoji primární platformu, Flutter je umožňuje používat libovolně kdekoliv. Programátor si samozřejmě sám může definovat widgety vlastní.

Pro rozložení prvků na stránce (vytvoření *layout*) se taktéž používají widgety, přestože nejsou při zobrazení viditelné. Základními widgety pro tvorbu layoutu jsou `Row`, `Column` a `Container`. Pomocí `Container` můžeme ostatním widgetům přidávat odsazení atd. Tyto widgety slouží pro specifickou či nízkoúrovňovou tvorbu layoutu. Můžeme využít i specializované widgety jako `GridView` pro tvorbu mřížky nebo `ListView` pro rolovatelný seznam. Nejvíce specifické widgety jako `BottomNavigationBar` či `AppBar` zajistí dodržení pravidel stanovených Material Designem a umožní programátorovi velmi jednoduchou implementaci.

3.3 Material Design

V březnu 2025 je Material Design verze 3 [11] nejnovějším open source designový systém od společnosti Google. Systémem v tomto případě mám na mysli soubor pravidel pro uživatelské rozhraní, konkrétní komponenty i barevné provedení. Všechny tyto prvky jsou vytvořeny zkušenými designéry s respektem pro psychologický vliv uživatelského rozhraní na člověka.

Předchozí označení verze 3 napovídá, že v minulosti již proběhly aktualizace tohoto systému. Tento krok dává smysl vzhledem k vyvíjejícím se trendům v oblasti designu. Tyto aktualizace mohl běžných uživatel pozorovat v operačním systému Android nebo v aplikacích od Googlu, jelikož Google tento systém používá téměř všude.

Použití tohoto systému má ve Flutteru řadu výhod. Především většina Material widgetů je již implementována [12] a použití programátorem je velmi jednoduché. Programátor nemusí často řešit rozmístění jednotlivých prvků v konkrétní komponentě ani správnou adaptaci na velikost zařízení. Běžnou součástí aplikací jsou ikony. Ikony z Material Designu jsou ve Flutteru k přímému použití bez specifického nastavení. Programátor nemusí řešit žádné externí SVG soubory. Stejně to platí i pro použití fontů písem.

3.4 SQLite

Jedná se o velmi jednoduchou a rychlou relační databázi, která je uložena v jediném souboru [13]. Projekt SQLite byl zahájen v roce 2000. Obvykle jsou všechny potřebné závislosti již zabudovány v zařízeních, ať už jde o mobilní telefony nebo desktopové operační systémy. Je zdarma k užití pro libovolné účely. Přesto poskytuje plnohodnotnou SQL implementaci. Zajímavostí je, že onen soubor je multiplatformní. Nevadí mu přenos mezi 32bitovými a 64bitovými systémy nebo architekturami big-endian a little-endian.

3.5 Balíčky

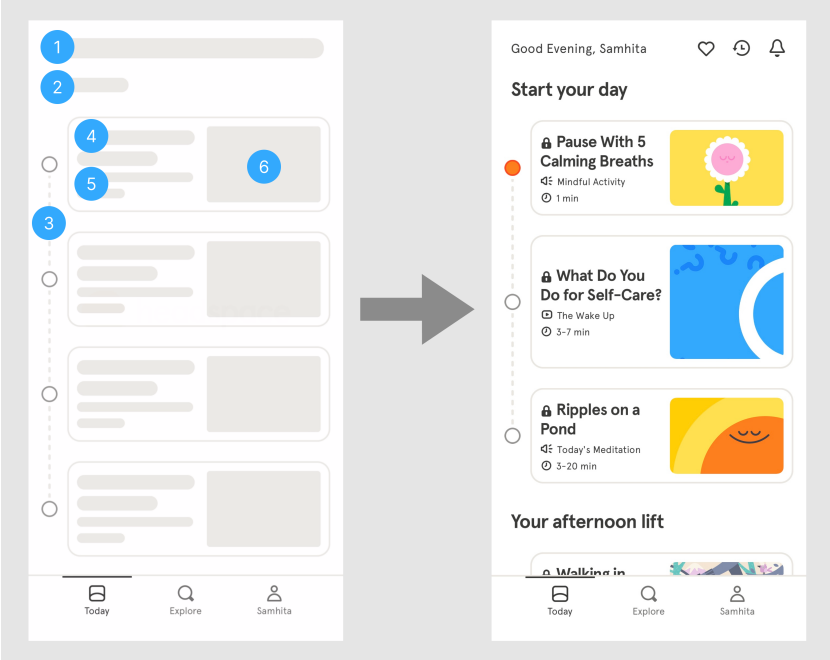
Přestože balíčky tvoří závislosti projektu na ostatních okolnostech a programátor se musí spoléhat na jiné programátory, že je budou udržovat aktuální, je nemožné se jim v dnešním době vyhnout. Dart a Flutter má výběr z široké veřejné knihovny balíčků, které jsou vyvíjeny ostatními programátory. Balíčky je nutné do projektu zahrnout. Celý jejich seznam je dostupný v souboru `pubspec.yaml`.

- **Drift** je balíček poskytující rozhraní pro práci s SQLite. databází [14]. Její velkou výhodou oproti ostatním implementacím SQLite pro Dart je možnost multiplatformnosti. Funguje na všech platformách od Androidu přes Windows až po webové rozhraní. V oficiálním návodu je doporučena knihovna sqflite, ta však tuto výhodu neposkytuje. Dále poskytuje velmi jednoduché API pro vytváření dotazů nad databází bez použití jazyka SQL. Programátor vytváří strukturu databáze v jednom souboru, na jehož základě si Drift generuje interní soubory.

- **GoRouter** je deklarativní balíček pro organizaci navigace v rámci aplikace (tzv. *routování*). Funguje na principu URL adres. Umožňuje v adrese předávat parametry, zajistit přesměrování na základě práv uživatele či použití *vnitřních navigátorů* nejčastěji pro komponentu `BottomNavigationBar`, který zůstává neustále viditelný skrz více obrazovek. Pro přechody umožňuje nastavit vlastní libovolné animace. Výhodou je, že autorem jsou přímo oficiální vývojáři Flutter, tudíž by balíček měl zůstat udržovaný.

- **Skeletonizer** zjednodušeným způsobem poskytuje funkcionalitu označovanou jako *skeleton loading*. Používá se při načítání obsahu na stránce a zobrazuje hrubý náhled rozložení prvků na stránce. Vše je doplněno vhodnou animací, aby uživatel ihned poznal, že se obrazovka právě načítá. Zjednodušení použitím tohoto balíčku spočívá v tom, že programátorovi stačí již definovaný widget „obalit“ widgetem `Skeletonizer` z tohoto balíčku a o zbytek práce se balíček postará sám.

- **Another Flushbar** poskytuje vylepšenou komponentu `Snackbar` z Material Design. Účelem komponenty je krátce informovat uživatele o akci, která



Obrázek 6: Skeleton loading a načtená obrazovka - převzato z [15]

byla právě provedena. Obvykle se objevuje ve spodní části obrazovky. Vylepšení spočítá v možnosti většího přizpůsobením - přidání ikony, změna barvy či animace a libovolné umístění. Já tento balíček preferoval i z důvodu správného zobrazování při otevřeném dialogovém okně.

- **Syncfusion Flutter Charts** je rozsáhlá knihovna pro tvorbu grafů. Umožňuje vytvářet všechny druhy grafů - sloupcové, spojnicové, prstencové a další (autoři uvádí více než 30 druhů). Grafy lze dále přizpůsobovat - výběr animace, úprav jednotek na osách, výběr barev, možnost zobrazit legendu či podrobné informace daného bodu v grafu po přejetí myší. Přes širokou škálu úprav je základní použití velmi jednoduché. Velkým bonusem je rozsáhlá dokumentace s ukázkami všech typů grafů a jejich zdrojových kódů.

- **File Picker** dává programátorovi možnost využít nativní průzkumník souborů k výběru složky nebo konkrétních souborů pro jejich další zpracování v aplikaci. Je možnost soubory filtrovat či umožnit výběr více z nich najednou. Základní funkcionalitu poskytuje na libovolné platformně s velmi jednoduchou implementací. V mém případě jsem ho použil pro výběr složky, kam si uživatel chce uložit exportované soubory.

4 Programátorská příručka

Multiplatformní vývoj přináší jistá specifika (nutnost zvolit vhodné technologie, nutnost případné optimalizace zobrazení či zúžený výběr knihoven), ale z hlediska architektury či samotného procesu programování je většina věcí stejných. Zvolená technologie často sama navádí k použití vhodných prvků.

Mnou navržené a vytvořené aplikace jsem dal název Budget Buddy. Je určena pro mobilní telefony, tablety i desktopová zařízení. Testování probíhalo na operačních systémech Android verze xx a vyšší, iOS xx a vyšší a Windows xx a vyšší. Pro ostatní operační systémy není zaručena plná funkcionalita, i když z podstaty frameworku by se tak stát nemělo.

Pro vývoj aplikace jsem použil textový editor Visual Studio Code s řadou rozšíření. Část práce specifické pro platformu Android jsem realizoval v prostředí Android Studio. Především se jednalo o úlohy typu aktualizace Software Developement Kit či migrace na novější verzi Kotlin gradle, kde Androi Studio poskytuje jednodušší rozhraní a programátora výborně provede celým procesem.

4.1 Architektura aplikace

Pro multiplatformní vývoj neexistuje doporučená architektura už z podstaty věci širokého záběru různých zařízení. Je možné se držet dlouho známých architektur, které se používají v různých systémech (*Model-View-Controller* či *Mode-View-Viewmodel*) nebo se řídit podle zvyklostí daného frameworku (ve Flutteru se jedná například o *The Clen Architecture* nebo *Bloc*). Já se ve své aplikaci držel architektury Model-View-Viewmodel, jelikož s ní mám z minulosti zkušenosti a pro moji malou aplikaci je plně dostačující.

Model-View-Viewmodel (zkráceně MVVM) je architektura, která odděluje v aplikaci uživatelské rozhraní, logiku a datovou část [16]. Ve Flutteru je velmi dobře podporována, jednak z hlediska komponent frameworku, tak i ofociální dokumentace na architekturu odkazuje jako vhodný příklad návrhu aplikace. Rozdělení aplikace je do následujících tří detailněji popsaných celků.

- **Model** je částí zapouzdřující data, se kterými aplikace pracuje. Udává strukturu poskytovaných dat i metody pro jejich získání objekty ve Viewmodel.
- **View** je zodpovědné za strukturu a zobrazení uživatelského rozhraní. Neměl by obsahovat žádnou logiku aplikace. Vstupy které přijme od uživatele předává ke zpracování na Viewmodel.
- **Viewmodel** je prostředníkem mezi View a Model. Do View poskytuje data a případně je upraví do vhodného formátu pro zobrazení. Reaguje na zprávy od View, které představují vstupy uživatele. Vlákno pro uživatelské rozhraní by neměl blokovat, zlepší se tím uživatelský zážitek.

V mé konkrétní implementaci je kód strukturován do složek podle MVVM architektury - `model`, `view` a `view_model`. Ve `view` je jemnější rozdělení na obrazovky `screens` a jednotlivé widgety `widgets`. Dále je v projektu i složka `data` pro databázi, složka `utils` pro jednoduché funkce používané skrz projektem (převod kódu ikony na Ikonu, zkrášlení číselného formátu a další) a složka `theme` spravuje vizuální téma aplikace.

4.2 Multiplatformní část projektu

Tady nevím jestli něco přesně bude, nebo cca co.

4.3 Uchovávání stavu

Uchování stavu jednotlivých obrazovek a jiných prvků je velmi důležité pro uživatelský komfort. Běžně se uživatel překlikne nebo přeskakuje mezi obrazovkami a nemá po každé změně vše nastavovat znovu nebo scrollovat na stejné místo v seznamu. Použití GoRouter za programátora řeší i tento problém. Není nutné nic nastavovat a stačí řídit se dokumentací GoRouter. Po zavedení do aplikace již uchování stavů funguje přesně tak, jak je pro uživatelský komfort vhodné. V první řade bylo nutné vytvořit aplikaci konstruktorem s `.router`. Do parametru `router` jsem vložil konfiguraci GoRouter (viz kód 2). V konfiguraci je uvedena cesta a obrazovka, která se při jejím zadání zobrazí.

```
1  // GoRouter konfigurace
2  final _router = GoRouter(
3    routes: [
4      GoRoute(
5        path: '/',
6        builder: (context, state) => HomeScreen(),
7      ),
8    ],
9  );
10
11 // konstruktor aplikace
12 class MyApp extends StatelessWidget {
13   @override
14   Widget build(BuildContext context) {
15     return MaterialApp.router(
16       routerConfig: _router,
17     );
18   }
19 }
```

Zdrojový kód 2: Ukázka použití GoRouter

Obrázek 7: Navigační prvky - převzato z [17]

4.4 Implementace responzivního designu

Responzivní design má zajistit, že aplikace se bude vhodně zobrazovat na libovolně velkém zařízení. Vyskytuje se především u webových technologií z důvodu běžného používání na různých typech zařízení. Stejná problemtika je u multiplatformních aplikací.

Podle velikosti obrazovky je nutné vhodně zvolit z určitých komponent, které slouží pro stejný účel, ale jsou jinak zobrazené. V této aplikace se jedná například o navigaci. Pro menší obrazovky jsem použil navigation bar, pro větší navigation rail, případně navigation drawer (viz obrázek 7). Dále je nutné vhodně zobrazit komponenty v hlavním těle obrazovky. Řazení komponent na menším zařízení jsem zvolil pod sebe do sloupce, na větším naopak vedle sebe v řádku (viz obrázek grafy v mé appce). Je dostupná i komponenta `GridView` vytvářející tabulku. Avšak není vhodná pro tvorbu layoutu. Material Design poskytuje i speciální kanonické rozložení (viz obrázek material), které je vhodné ve spojení se zobrazením seznamu. V této aplikaci by bylo vhodné pro obrazovku transakcí. Implementace pro Flutter zatím není dostupná, tudíž není možné ho použít. Dialogová okna jsem podle doporučení pro menší zařízení zvolil celoobrazovková, zatímco na větších jsou plovoucí (viz obrázek dialog okno moje).

Flutter poskytuje komponenty vhodné pro responzivní design, ale na programátorovi je ponecháno kdy jaké zvolí, včetně celé implementace. Zatím není dostupná žádná komponenta pro tvorbu responzivního rozložení, které by umožnila v jediném kódu pro uživatelské rozhraní rozhodnout do jakého widgetu kód „obalit“.

4.5 Zajímavosti při tvorbě práce

Představa a samotná tvorba práce místy přinesou nečekané komplikace, které je nutné řešit nebo redefinovat požadavek. I v tomto projektu jsem se s touto

výzvou setkal.

- **Problém s výběrem SQLite balíčku** nastal hned ze začátku programování aplikace. Při postupování podle dokumentace [18], která ukazuje práci s balíčkem sqflite, jsem při čtení dokumentace balíčku zjistil, že balíček podporuje pouze Android, iOS a macOS. Vzhledem k zamýšlenému provozu i na jiných platformách bylo nutné najít alternativu. Objevený balíček Drift naštěstí podporuje všechny platformy. Navíc přinesl i bonus v kvalitní API pro dotazování nad databází bez použití jazyka SQL.
- Nedodělané všechny komponenty Material designu.
- Smazání kategorie a co transakce v ní

5 Uživatelská příručka

Aplikaci Budget Buddy pro správu financí lze nainstalovat na operační systém Android a Windows se zárukou bezproblémové funkčnosti podle návodu. Na operační systémy macOS a Linux je nutné aplikaci pro tyto systémy sestavit a nainstalovat (tento postup je ponechán na zdatném uživateli). Návod se týka jen vybraných platforem z důvodu testování.

5.1 První spuštění

Při spuštění se zobrazí rovnou úvodní obrazovka *dashboard*. Není nutné se přihlašovat, aplikace je nyní určena pouze pro jednoho uživatele a běží lokálně. Pro první spuštění jsou vložené základní kategorie transakcí a nejběžnější měny (obojí je možno upravit).

5.2 Úvodní obrazovka

Úvodní obrazovka, anglicky dashboard je centrem dění pro uživatele. Poskytuje základní přehled o finanční situaci. Konkrétně zobrazuje graf kategorií, stav účtu, dnešní útratu a posledních pět transakcí. Na zařízení s větší obrazovkou je rozšířena o legendu grafu a další zajímavé číselné údaje o stavu účtu. Úvodní obrazovka není uživatelsky přizpůsobitelná.

5.3 Zobrazení a úprava transakcí

Transakce a jejich úprava se provádí na druhé obrazovce s názvem *transactions*. Výpis je proveden v seznamu, kde je přímo vidět kategorie, měna, částka, typ, datum a případná poznámka. Po kliknutí na transakci se objeví dialogové okno a je možné všechny parametry transakce upravit, případně transakci odstranit. Transakce je možné filtrovat dvěma způsoby. Můžeme nechat zobrazit jen transakce ve specifickém období (den, týden, měsíc, rok či libovolný rozsah), k čemuž

slouží porstřední tlačítko v horní části. Šipky na stranách umožňují přesouvat se o zvolené období dozadu či dopředu. Dále můžeme vybírat ze všech parametrů, které jsou transakci přiřezené (kategorie, měna, typ atd.). Obrazovka s výběrem filtrů se otevře při stisknutí tlačítka *Filter* s ikonou trychtýře. Při použití více filtrů se transakce zobrazí ve výpisu pokud splňuje alespoň jednu podmínku. Aktivní filtry znázorňuje odznak u tlačítka filtrace.

5.4 Využití reportů

Grafy a důležité číselné údaje se nachází na obrazovce *reports*. Je zde na výběr z pěti přednastavených možností. Které údaje chce uživatel vidět se dá nastavit pod tlačítkem *Manage reports* ve spodní části.

5.5 Nastavení aplikace

5.5.1 Přizpůsobení vzhledu

5.5.2 Přizpůsobení měn

5.5.3 Úpráva kategorií

5.6 Export dat

6 Možná rozšíření aplikace

Osobně si myslím, že nikdy se nedá o softwaru prohlásit, že by nepotřeboval vývoj a není možné mu dodat vylepšení. Už jen vzhledem k vnějšímu vývoji technologií a proměně uživatelských požadavků je nutné aplikaci udržovat aktuální. Se změnou požadavků se pojí rozšíření o nové funkce a nebo naopak odstranění funkcí nepoužívaných.

1. Možnost založit více účtů
2. Umožnit nastavení vlastní měny
3. Import dat z jiných aplikací
4. Vylepšení uživatelského rozhraní

Závěr

Tady bude závěr.

Conclusions

Here will be conclusion.

Literatura

[1] *Cashew*. Dostupný z: ⟨https://cashewapp.web.app/⟩.

[2] *What is wallet - Wallet by BudgetBakers - Your New Personal Finance Manager*. Dostupný z: ⟨https://budgetbakers.com/what-is-wallet/⟩.

[3] *Android Money Manager Ex – Aplikace na Google Play*. Dostupný z: ⟨https://play.google.com/store/apps/details?id=com.money.manager.ex.android⟩.

[4] *MoneyManager Ex*. Dostupný z: ⟨https://moneymanagerex.org/⟩.

[5] *HomeBank | Free personal finance software, money management for everyone*. Dostupný z: ⟨https://www.gethomebank.org/en/index.php⟩.

[6] *The Six Most Popular Cross-Platform App Development Frameworks | Kotlin Multiplatform Development Documentation*. Dostupný z: ⟨https://www.jetbrains.com/help/kotlin-multiplatform-dev/cross-platform-frameworks.html⟩.

[7] *Dart overview | Dart*. Dostupný z: ⟨https://dart.dev/overview⟩.

[8] *UI | Flutter*. Dostupný z: ⟨https://docs.flutter.dev/ui⟩.

[9] *Showcase - Flutter apps in production*. Dostupný z: ⟨https://flutter.dev/showcase⟩.

[10] *Flutter architectural overview | Flutter*. Dostupný z: ⟨https://docs.flutter.dev/resources/architectural-overview⟩.

[11] *Material Design*. Dostupný z: ⟨https://m3.material.io/⟩.

[12] *Components – Material Design 3*. Dostupný z: ⟨https://m3.material.io/components⟩.

[13] *SQLite Home Page*. Dostupný z: ⟨https://www.sqlite.org/⟩.

[14] *Home - Drift*. Dostupný z: ⟨https://drift.simonbinder.eu/⟩.

[15] Tankala, Samitha. *Skeleton Screens 101*. Dostupný z: ⟨https://www.nngroup.com/articles/skeleton-screens/⟩.

[16] *Model-View-ViewModel - .NET | Microsoft Learn*. Dostupný z: ⟨https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm⟩.

[17] *Applying layout – Material Design 3*. Dostupný z: ⟨https://m3.material.io/foundations/layout/applying-layout/window-size-classes⟩.

[18] *Docs | Flutter*. Dostupný z: ⟨https://docs.flutter.dev/⟩.