

UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

Návrh a implementace systému pro správu finančních
prostředků pro mobilní zařízení

Lukáš Holoubek

Bakalářská práce
2015

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Lukáš Holoubek**
Osobní číslo: **I12132**
Studijní program: **B2646 Informační technologie**
Studijní obor: **Informační technologie**
Název tématu: **Návrh a implementace systému pro správu finančních prostředků pro mobilní zařízení**
Zadávající katedra: **Katedra informačních technologií**

Z á s a d y p r o v y p r a c o v á n í :

Teoretická část bude obsahovat rešeršní činnost technologií pro tvorbu aplikací pro mobilní telefony. Dále zde budou popsány technologie pro tvorbu grafických rozhraní aplikací. Praktická část bude zaměřena na návrh a implementaci systému pro správu finančních prostředků pro mobilní zařízení. Aplikace bude umožňovat následující operace:

- zaznamenávat příjmy/výdaje,
- nastavování fixních příjmů/výdajů,
- možnost spravování více peněženek,
- filtrování pomocí kategorií a času,
- zobrazení grafů.

Mobilní aplikace bude naprogramována pro operační systém Android/Windows Phone.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

MURPHY, Mark L. Android 2: průvodce programováním mobilních aplikací. 1. Vyd. Brno: Computer Press, 2011, 375 s. ISBN 978-80-251-3194-7.

LEE, Wei-Meng. Beginning Android application development. Indianapolis, IN: Wiley Pub., 2011, 448 s. ISBN 978-1-118-01711-1.

LACKO, L'uboslav. Vývoj aplikací pro Windows 8.1 a Windows Phone. 1. vyd. Brno: Computer Press, 2014, 328 s. ISBN 978-80-251-3822-9.

Vedoucí bakalářské práce:

Ing. Emil Řezanina

Katedra informačních technologií

Datum zadání bakalářské práce:

20. prosince 2014

Termín odevzdání bakalářské práce:

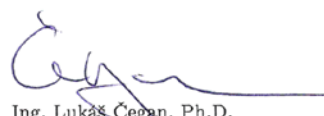
11. května 2015



prof. Ing. Simeon Karamazov, Dr.
děkan



L.S.



Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2015

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 11. 5. 2015

Lukáš Holoubek

Poděkování

Chtěl bych tímto způsobem poděkovat svému vedoucímu práce Ing. Emilu Řezaninovi za vedení a užitečné rady při zpracovávání bakalářské práce a také své rodině a blízkým, kteří mě v tomto směru všestranně podporovali.

Anotace

Cílem této práce je návrh a implementace aplikace pro správu finančních prostředků. Teoretická část se zabývá možnostmi vývoje aplikací třetích stran pro různé operační systémy, zejména iOS, Android a Windows Phone, a porovnáním podobných dostupných aplikací pro platformu Windows Phone. V praktické části práce je rozebrána samotná implementace aplikace pro Windows Phone 8.1. Jsou zde popsány použité návrhové vzory, životní cyklus aplikace a použití databázového systému SQLite, dále uživatelské rozhraní, implementace importu a exportu dat do souboru a tvorba grafů.

Klíčová slova

Mobilní aplikace, Windows Phone, XAML, MVVM, C#, SQLite

Title

Design and implementation of a financial resources management system for mobile devices

Annotation

Goal of this thesis is to design and to implement an application for financial resources management. Theoretical part deals with available options for third-party application development for different mobile operating systems, namely iOS, Android and Windows Phone and comparison of available similar applications for Windows Phone platform. The practical part analyses the implementation of the Windows Phone 8.1 application. Used design patterns are described here as well as an application lifecycle, use of SQLite database system, user interface, implementation of import and export of user data to file and charts creation.

Keywords

Mobile application, Windows Phone, XAML, MVVM, C#, SQLite

Obsah

Seznam zkratk	9
Seznam obrázků	10
Seznam tabulek	11
1 Úvod	12
2 Technologie pro tvorbu mobilních aplikací	13
2.1 Technologie pro tvorbu aplikací pro operační systém iOS	13
2.1.1 Úvod do technologie iOS	13
2.1.2 Struktura operačního systému iOS	14
2.1.3 Programovací jazyky pro tvorbu aplikací pro iOS	15
2.2 Technologie pro tvorbu aplikací pro operační systém Android	15
2.2.1 Úvod do technologie Android	15
2.2.2 Architektura operačního systému Android	16
2.2.3 Programovací jazyky pro tvorbu aplikací pro Android	18
2.3 Technologie pro tvorbu aplikací pro Windows Phone	18
2.3.1 Stručný přehled mobilních operačních systémů od firmy Microsoft	18
2.3.2 Struktura Windows Phone	19
2.3.3 Programovací jazyky pro tvorbu aplikací pro Windows Phone	20
2.4 Technologie pro tvorbu grafických rozhraní aplikací	20
2.4.1 Tvorba GUI pro iOS	20
2.4.2 Tvorba GUI pro Android	21
2.4.3 Tvorba GUI pro Windows Phone	21
2.5 Shrnutí vývoje a distribuce aplikací pro jednotlivé mobilní platformy	22
3 Aplikace pro správu finančních prostředků pro mobilní zařízení	24
3.1 Budget Buddy	24
3.2 Aplikace iMoney	24
3.3 Money Tracker Free	24
3.4 My Expenses	24
4 Návrh a implementace aplikace	26
4.1 Požadavky na aplikaci	26
4.2 Použité návrhové vzory	26
4.3 Architektura MVVM	26

4.4	Data binding	27
4.5	Životní cyklus aplikace.....	28
4.5.1	Stav Running	29
4.5.2	Stav Suspended.....	29
4.6	Databáze	29
4.6.1	Instalace knihovny SQLite	29
4.6.2	Databázový model	30
4.6.3	Práce s databází SQLite pomocí wrapperu SQLite PCL.....	31
4.7	Tvorba uživatelského rozhraní	32
4.7.1	Stránky uživatelského rozhraní	33
4.7.2	Úvodní stránka.....	33
4.7.3	Stránka přidání a úpravy příjmu a výdaje.....	34
4.7.4	Stránka přidání a úpravy štítku.....	35
4.7.5	Stránka Filtr položek	36
4.7.6	Stránka správy účtů	36
4.7.7	Stránka pro přidání a úpravu účtu	36
4.7.8	Stránka nastavení.....	37
4.7.9	Stránka pro zadání hesla	38
4.7.10	Stránka o aplikaci	38
4.7.11	Použití Blend for Visual Studio a vzorových dat	38
4.7.12	Tvorba grafů	40
5	Uživatelská část.....	41
5.1	Úvodní stránka.....	41
5.2	Stránka přidání a úpravy příjmu a výdaje.....	42
5.3	Stránka přidání a úpravy štítku.....	44
5.4	Stránka Filtr položek	44
5.5	Stránka správy účtů a přidání/úpravy účtu	45
5.6	Stránka nastavení.....	46
5.7	Stránka o aplikaci a pro zadání hesla.....	47
6	Závěr.....	48
	Použitá literatura.....	49
	Příloha A: Obsah přiloženého CD	53

Seznam zkratek

GPS	Global Positioning System
SDK	Software Development Kit
API	Application Programming Interface
VM	Virtual Machine
HTML	HyperText Markup Language
CSS	Cascade StyleSheet
XML	Extensible Markup Language
BSP	Board Support Package
OEM	Original Equipment Manufacturer
GUI	Graphical User Interface
MVC	Model View Controller
IDE	Integrated Development Environment
ADT	Android Development Tools
XAML	Extensible Application Markup Language
WYSIWYG	What You See Is What You Get
WPF	Windows Presentation Foundation
JSON	JavaScript Object Notation
MVVM	Model View ViewModel
MVC	Model View Controller
CLR	Common Language Runtime
PCL	Portable Class Library
LIFO	Last In First Out

Seznam obrázků

Obrázek 1: Model vrstev iOS	14
Obrázek 2: Architektura OS Android.....	17
Obrázek 3: Struktura OS Windows Phone	19
Obrázek 4: Podíl mobilních operačních systémů v ČR v letech 2012 až 2015.....	23
Obrázek 5: Životní cyklus aplikace	28
Obrázek 6: Relační Model databáze.....	30
Obrázek 7: Úvodní stránka – první tři sekce	41
Obrázek 8: Úvodní stránka – poslední dvě sekce.....	42
Obrázek 9: Stránka nového výdaje.....	43
Obrázek 10: Stránka úpravy výdaje.....	43
Obrázek 11: Stránka úpravy štítku	44
Obrázek 12: Stránka filtru položek.....	45
Obrázek 13: Stránka správy účtů a nového účtu	45
Obrázek 14: Stránka správy účtů a úpravy účtu.....	46
Obrázek 15: Stránka nastavení	47
Obrázek 16: Stránka o aplikaci a stránka požadování hesla.....	47

Seznam tabulek

Tabulka 1: Názvy verzí OS Android	16
---	----

1 Úvod

Cílem bakalářské práce je vytvořit aplikaci pro mobilní zařízení, která by uživatelům umožňovala zaznamenávat si své příjmy a výdaje, třídit je do kategorií a poskytovat souhrny ve formě grafů. V současné době je téma vývoje aplikací pro mobilní zařízení aktuální, a proto znalosti získané z této práce mohou mít další praktické využití.

V teoretické části bude provedena rešerše technologií pro vývoj aplikací pro různé mobilní operační systémy. Podrobněji zde budou rozebrány tři nejrozšířenější mobilní operační systémy iOS, Android a nejnovější Windows Phone. U každého operačního systému bude popsána jeho architektura, možnosti použití programovacích jazyků a přehled verzí daného OS.

Teoretická část práce je dále věnována dostupným nástrojům pro tvorbu grafických rozhraní aplikací. Po shrnutí možností vývoje a distribuce aplikací pro dané operační systémy a stručném porovnání samotných OS, je proveden průzkum dostupných aplikací, zabývajících se stejným tématem na vybrané platformě.

Praktická část se již zabývá samotným návrhem a implementací aplikace. Na začátku jsou zde zmíněny požadavky na aplikaci, následované použitými návrhovými vzory. Poté je zde popsán životní cyklus aplikace a vybraný systém pro ukládání dat společně s databázovým modelem. V podkapitole tvorby uživatelského rozhraní jsou popsány jednotlivé části vytvořeného uživatelského rozhraní s popisem implementace některých funkcí. Na konci kapitoly zabývající se vývojem aplikace je zmíněn způsob tvorby vzorových dat, využitých při návrhu grafického rozhraní, a způsob tvorby grafů.

V poslední kapitole se nachází popis uživatelské části aplikace se snímky grafického rozhraní.

2 Technologie pro tvorbu mobilních aplikací

Smartphone je označení pro mobilní telefon, který disponuje vlastním pokročilým operačním systémem. Smartphone neboli chytrý telefon má obvykle velký dotykový displej, fotoaparát a často i jednotku GPS. Éra smartphonů s velkou dotykovou obrazovkou, jak je známe dnes, se počíná datovat od vydání prvního iPhone společnosti Apple Inc. 29. června 2007. Současným smartphonům s dotykovými displeji předcházely tzv. hloupé telefony. Tyto telefony nepoužívaly „klasický“ operační systém, ale tzv. firmware, který v původním významu představuje programové vybavení na úrovni mezi software a hardware. Z toho důvodu byl firmware pro každý model telefonu trochu jiný. Chytré telefony s operačním systémem také používají firmware. Ten zde však hraje roli podpory či komunikace s hardwarem, je tedy pro různé modely telefonů odlišný. Jeho aktualizace často přináší nové funkce a možnosti operačnímu systému a hlavně aplikacím v něm běžícím. [1]

Prakticky všechny moderní mobilní telefony s jejich operačními systémy umožňují tvorbu aplikací třetích stran. Mezi nejznámější operační systémy patří iOS, Android a Windows Phone, následník Windows Mobile. Vývoj aplikací pro jednotlivé operační systémy je charakterizován zejména použitým programovacím jazykem a sadou nástrojů poskytnutou pro vývoj aplikací, zkráceně SDK. Zmíněné tři operační systémy mobilních telefonů budou dále v práci podrobněji zpracovány.

Za zmínku také stojí operační systém BlackBerry OS vyvinutý kanadskou společností Research In Motion Ltd. (RIM). Tato společnost byla přejmenována na BlackBerry s příchodem smartphonů BlackBerry řady 10. Ve své době byly telefony BlackBerry s tímto OS velmi oblíbené zvláště v oblasti Kanady a USA pro svoji podporu firemní komunikace. BlackBerry v té době měla na trhu mobilních telefonů dominantní pozici. V posledních letech je však, hlavně díky telefonu iPhone a operačnímu systému Android, podíl zařízení BlackBerry na trhu minimální. Z tohoto důvodu se nebudeme problematikou vývoje aplikací pro BlackBerry OS podrobněji zabývat. [2] [3]

2.1 Technologie pro tvorbu aplikací pro operační systém iOS

2.1.1 Úvod do technologie iOS

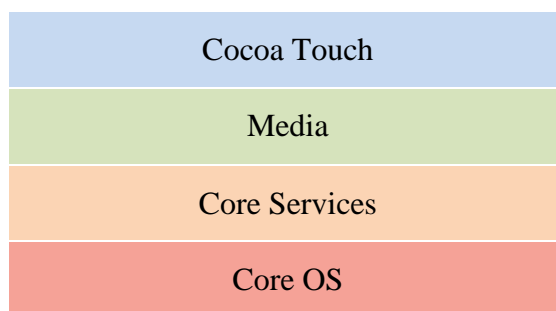
Operační systém iOS vyvinula společnost Apple Inc. pro svá mobilní zařízení (jako např. iPhone, iPod či iPad). Jde o odlehčenou verzi operačního systému OS X používaného ve stolních počítačích a noteboocích Apple. OS X a tedy i iOS jsou tzv. „UNIX-like“, což označuje podobnost jejich chování s operačním systémem UNIX.

Apple vydal první verzi svého operačního systému zároveň s uvedením svého prvního mobilního telefonu iPhone koncem června 2007. V té době ještě tento operační systém neměl žádné oficiální jméno. V březnu 2008 byla pro iPhone vydána sada vývojářských nástrojů, tzv. software development kit (SDK), a Apple pojmenoval svůj operační systém

jednoduše iPhone OS. Název „iOS“ byl používán až od verze 4.0. Poslední stabilní verze tohoto operačního systému je 8.1.3 z 27. ledna 2015.

2.1.2 Struktura operačního systému iOS

Jednou z hlavních funkcí operačních systémů je poskytnout programátorovi aplikační rozhraní, jakousi mezivrstvou pro komunikaci s hardwarem. Vytvořené aplikace tedy nikdy nekomunikují s hardwarem přímo, ale využívají programových struktur již implementovaných jako součást operačního systému. To umožňuje stejným programům fungovat na různých zařízeních s rozličným hardwarem. Podobně jako ostatní operační systémy je i architektura iOS uspořádána do vrstev, jak je ukázáno na Obrázku 1.



Obrázek 1: Model vrstev iOS [4]

Nižší vrstvy obsahují základní služby a jsou blíže spjaty s hardwarem, zatímco vyšší vrstvy poskytují větší míru abstrakce a stavějí na vrstvách nižších.

Vrstvu operačního systému iOS, která je nejbližší hardwaru, představuje **Core OS**. Tato vrstva obsahuje jádro operačního systému, které obstarává správu virtuální paměti, vláken, souborového systému a správu síťové komunikace a komunikace procesů. Z důvodu bezpečnosti je přístup do systému omezen na sadu rozhraní a frameworků. Rozhraní založená na jazyce C umožňují přístup k nízko-úrovňovým funkcím operačního systému, jako je konkurence POSIXových vláken, BSD socketů, alokace paměti a přístup k souborovému systému. Frameworky této vrstvy zase obsahují rozhraní pro zpracování signálů, zpracování obrázků, přístup k technologii Bluetooth, komunikaci s externími zařízeními atd.

Nad touto vrstvou se nachází vrstva **Core Services**. Ta obsahuje některé vysokoúrovňové funkce, například databázi SQLite, podporu XML nebo nákup uvnitř aplikací. Frameworky této vrstvy poskytují přístup k získání polohy zařízení a pohybovým senzorům nebo podporu datových typů kolekcí, řetězců, data a času apod.

Jak už název napovídá, vrstva **Media** obsahuje technologie pro uplatnění grafiky (zobrazování animací, textu a práci s obrázky), zvuku a videa v aplikacích.

Vysokoúrovňové funkce obstarává vrstva **Cocoa Touch**. Její frameworky pomáhají určit vzhled aplikace a poskytují podporu technologií, jako je vstup z dotykové obrazovky, upozornění a dalších vysokoúrovňových služeb.

Při psaní zdrojového kódu aplikace se doporučuje používání frameworků vyšších vrstev. Frameworky vyšších vrstev tvoří objektově orientovanou abstrakci nižších vrstev, čímž usnadňují tvorbu a zkracují zápis zdrojového kódu. Nicméně je stále možné používat frameworky nižších vrstev, pokud frameworky vrstev vyšších nesplňují z nějakého důvodu naše požadavky. Bez těchto frameworků by byl vývoj aplikací poměrně obtížný a různí programátoři by vymýšleli ty samé věci znovu a znovu. Takto někdy poměrně složité věci mohou být v aplikaci zhotoveny pomocí pár řádků kódu. [4]

2.1.3 Programovací jazyky pro tvorbu aplikací pro iOS

Primární programovací jazyk používaný společností Apple v jejích operačních systémech je jazyk Objective-C. Jeho syntaxe je odvozena od nízko-úrovňového, spíše procedurálně orientovaného programovacího jazyka C, od něhož zdědil syntaxi, primitivní datové typy a příkazy řízení toku programu, a od objektově orientovaného jazyka Smalltalk. [5]

Dalším využívaným programovacím jazykem je jazyk Swift. Tento programovací jazyk je relativně nový, zvláště v porovnání s Objective-C nebo dokonce se starým jazykem C. Vývoj Swiftu započal v roce 2010 a teprve v červnu 2014 byla vydána první aplikace napsaná v tomto jazyce. Poměrně neobvyklá syntaxe volání metod v Objective-C, inspirovaná jazykem Smalltalk, byla nahrazena tečkovou notací a systémem jmenných prostorů známých z jazyků jako jsou C, C# a Java. [6]

Dalšími možnostmi pro vývoj aplikací pro iOS jsou jazyky C, C++, MacRuby, PyObjC, MonoTouch, atd. Těmi se však zde nebudeme zabývat.

Pro vývoj nativních aplikací pro iOS je prakticky nezbytné SDK, které obsahuje především:

- vývojové prostředí Xcode,
- iOS Simulator simulující prostředí iPhoneu či iPadu,
- knihovny Cocoa Touch a
- dokumentaci.

Aplikaci je nutné podepsat certifikátem, i když ji chceme pouze testovat na zařízení. Ten získáme, pokud jsme součástí programu iPhone Developer, za který se platí 99 dolarů ročně. Po zaplacení je zpřístupněn i portál iTunes Connect pro nahrání a správu naší aplikace v App Store a zobrazování různých statistik. Po odeslání aplikace je nutné několik dnů počkat, až ji zaměstnanci Apple zkontrolují, zda splňuje všechna daná pravidla. Poté se aplikace ocitá na samém App Store. Stejnou kontrolou musí aplikace projít i v případě, že jde pouze o její aktualizaci. To je obzvláště při výskytu nějaké závažné chyby poněkud nepraktické. [7]

2.2 Technologie pro tvorbu aplikací pro operační systém Android

2.2.1 Úvod do technologie Android

Operační systém Android původně vyvíjela společnost Android Inc., založená v roce 2003 v Kalifornii. V roce 2005 ji koupila společnost Google Inc. Jde o open source software,

tedy software s dostupným zdrojovým kódem. Výrobci mobilních telefonů si tak mohou operační systém upravit k obrazu svému a vývojáři nemusí procházet schvalovacím procesem, aby mohli své aplikace uvést do provozu, jako je to u jiných platform.

O necelý rok později byl ve Spojených státech uveden první telefon s OS Android ve verzi 1.6. Prvním telefonem s tímto OS se stal HTC Dream.

Nejnovější verzí je Android 5 Lollipop. Jeho zdrojový kód byl zveřejněn v listopadu 2014 na webu Android Open Source Project, což umožnilo výrobcům telefonů a vývojářům tvorbu vlastních úprav systému. Hlavní vylepšení v této verzi představuje změna běhu aplikací, změna vzhledu grafického rozhraní systému, vylepšení výdrže telefonu na baterii a systému upozornění. Upozornění (Notifications) nyní mohou být zobrazována ve formě karet na zamykací obrazovce nebo seskupována podle aplikace, která je vyvolala. Také lze specifikovat, která aplikace nebude posílat upozornění na zamykací obrazovku nebo vydávat upozornění vůbec. [8]

Jako zajímavost lze uvést, že od verze 1.5 dostávaly verze názvy po sladkostech, jak je uvedeno v Tabulce 1. Sloupec API Level udává revizi frameworku aplikačního rozhraní. V manifestu aplikace je pak udávána minimální, maximální a preferovaná hodnota tohoto čísla, pro které byla aplikace navržena. [9]

Tabulka 1: Názvy verzí OS Android [10]

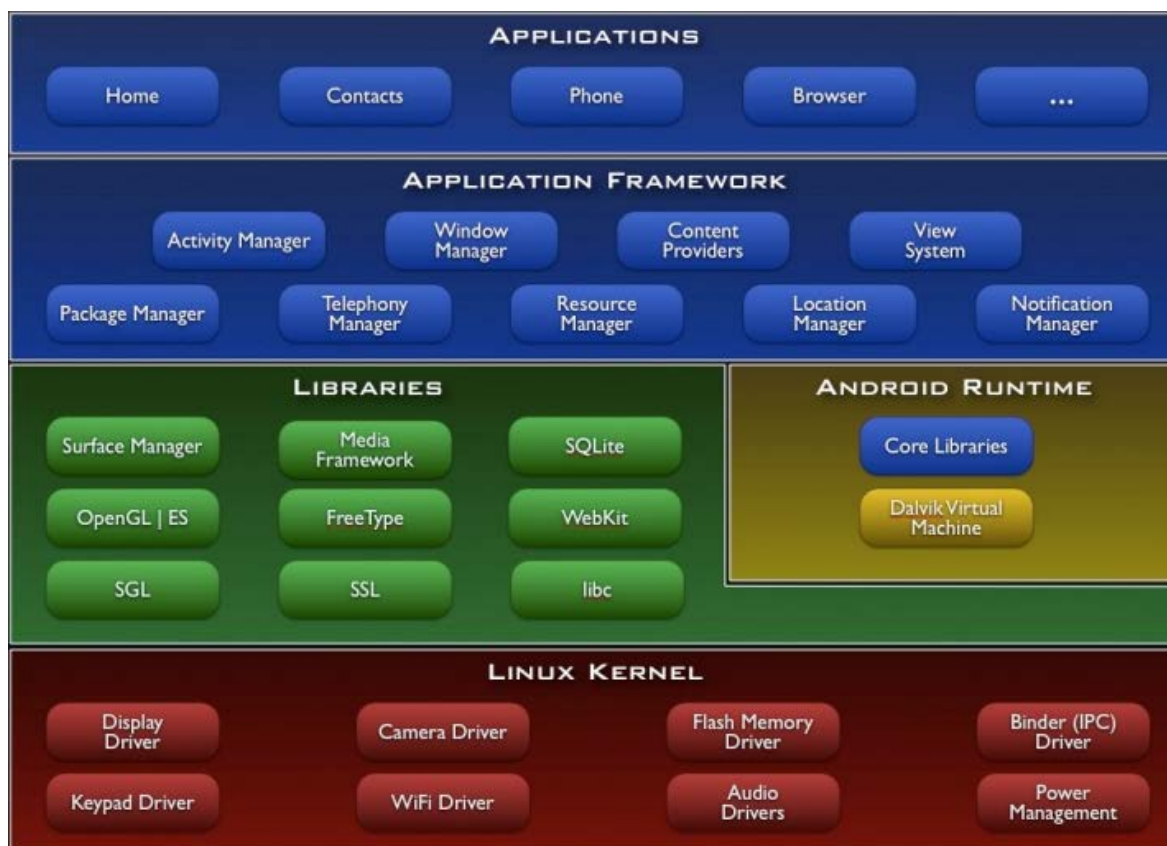
ČÍSLO VERZE	NÁZEV	API LEVEL	MĚSÍC & ROK VYDÁNÍ
1.0	Alpha	1	září 2008
1.1	Beta	2	únor 2009
1.5	Cupcake	3	duben 2009
1.6	Donut	4	září 2009
2.0 – 2.1	Éclair	5 – 7	říjen 2009
2.2 – 2.2.3	Froyo	8	květen 2010
2.3 – 2.3.7	Gingerbread	9 – 10	prosinec 2010
3.0 – 3.2.6	Honeycomb	11 – 13	únor 2011
4.0 – 4.0.4	Ice Cream Sandwich	14 – 15	říjen 2011
4.1 – 4.3.1	Jelly Bean	16 – 18	červen 2012
4.4 – 4.4.4	KitKat	19 – 20	září 2013
5.0 – 5.0.2	Lollipop	21	červen 2014

Od roku 2010 také Google začal vydávat svou řadu telefonů a tabletů Nexus, která představuje vlajkovou loď prezentující vždy nejnovější vlastnosti OS Android.

2.2.2 Architektura operačního systému Android

Architekturu Androidu tvoří čtyři hlavní vrstvy, viz Obrázek 2. Odlišnost barev na Obrázku 2 znázorňuje také odlišnost programovacích jazyků, ve kterých byly jednotlivé sekce napsány. Zelené části byly naprogramovány pomocí jazyka C/C++, zatímco modré

části jsou naprogramovány v Javě a běží ve virtuálním stroji Dalvik. V tomto bodě se Android liší od iOSu nebo Windows Phone, kde jsou aplikace většinou napsané v nativně kompilovaných jazycích jako je Objective-C, respektive v C/C++/C#.



Obrázek 2: Architektura OS Android [11]

Nejnižší vrstvu OS Android tvoří Linuxové jádro (**Linux kernel**) řady 2.6, upravené pro speciální potřeby úsporného hospodaření s napájením a pamětí.

Jak už název napovídá, vrstva **Libraries** obsahuje řadu knihoven, jako je například standardní knihovna jazyka C, SQLite databázi pro uchovávání aplikačních dat, knihovny pro přehrávání a záznam zvuku a videa a příslušné kodeky a zabezpečení síťové komunikace pomocí SSL.

Ačkoliv všechny aplikace a aplikační frameworky pro Android jsou napsány takřka výhradně v Javě, neběží ve standardním Java virtual machine, ale ve virtuálním stroji zvaném **Dalvik virtual machine**. Ten také používá vlastní formát bajtkódu, upraveného pro potřeby mobilního operačního systému, jenž je velikostně úspornější. [12] [13]

Ve verzi 5 „Lollipop“ je Dalvik VM oficiálně nahrazen Android Runtime (ART), který byl experimentálně použit již ve verzi 4.4 „Kit Kat“. Jde o multiplatformní prostředí pro běh programu (runtime) podporující architektury x86, ARM a MIPS ve 32 i 64 bitových variantách. Na rozdíl od Dalviku, který používá tzv. just-in-time kompilaci, ART zkompiluje aplikaci přímo do strojového kódu daného zařízení při její instalaci, tzv. ahead-of-time compilation, což značně zrychluje běh aplikace. [14]

Vrstva **Application Framework** poskytuje výše zmíněným aplikacím řadu služeb ve formě tříd v programovacím jazyce Java, které mohou využít právě vývojáři aplikací.

Vrstva **Applications** je místo, kde nalezneme všechny Android aplikace jako prohlížeč, hry, apod.

2.2.3 Programovací jazyky pro tvorbu aplikací pro Android

Programovací jazyk Java je jednoznačně nejpoužívanější programovací jazyk pro tvorbu aplikací pro OS Android. Ačkoli Android API používá velkou část Java 2 Standard Edition 5.0, jako jsou například balíčky `Java.io`, `Java.lang`, `Java.math`, `Java.net`, `Java.sound` apod., některé balíčky jsou v Android API vynechány buď proto, že jsou k dispozici novější a lepší balíčky, nebo tyto balíčky nemají v prostředí mobilních telefonů využití. Příkladem takových balíčků může být `Javax.print`, `Java.awt`, `Javax.swing`, `Javax.rmi` apod. [15]

Programovací jazyk C/C++ můžeme použít s pomocí Android Native Development Kit (NDK), což je sada nástrojů (kompilátory, knihovny a hlavičkové soubory), umožňující nám například využití již existujících knihoven napsaných v těchto jazycích. Použití C/C++ se však doporučuje pouze v případech, kdy potřebujeme zvýšit výkon nějaké důležité a na výkon a optimalizaci citlivé části aplikace, jako je například herní engine či simulace fyzikálních jevů.

Značkový jazyk XML je používán pro manifest aplikace, což je souhrnná informace o aplikaci, obsahující například minimální úroveň API, přístup k chráněným částem API či informaci o odkazovaných knihovnách.

Pro vývoj aplikací na OS Android je také dostupný balík nástrojů, který obsahuje mimo jiné debugger a knihovny, vzorové zdrojové kódy a tutoriály, vývojové prostředí Android Studio a emulátor se obrazem systému Android 5.0 s Google API. Lze jej nainstalovat na všechny současné platformy stolních počítačů.

2.3 Technologie pro tvorbu aplikací pro Windows Phone

2.3.1 Stručný přehled mobilních operačních systémů od firmy Microsoft

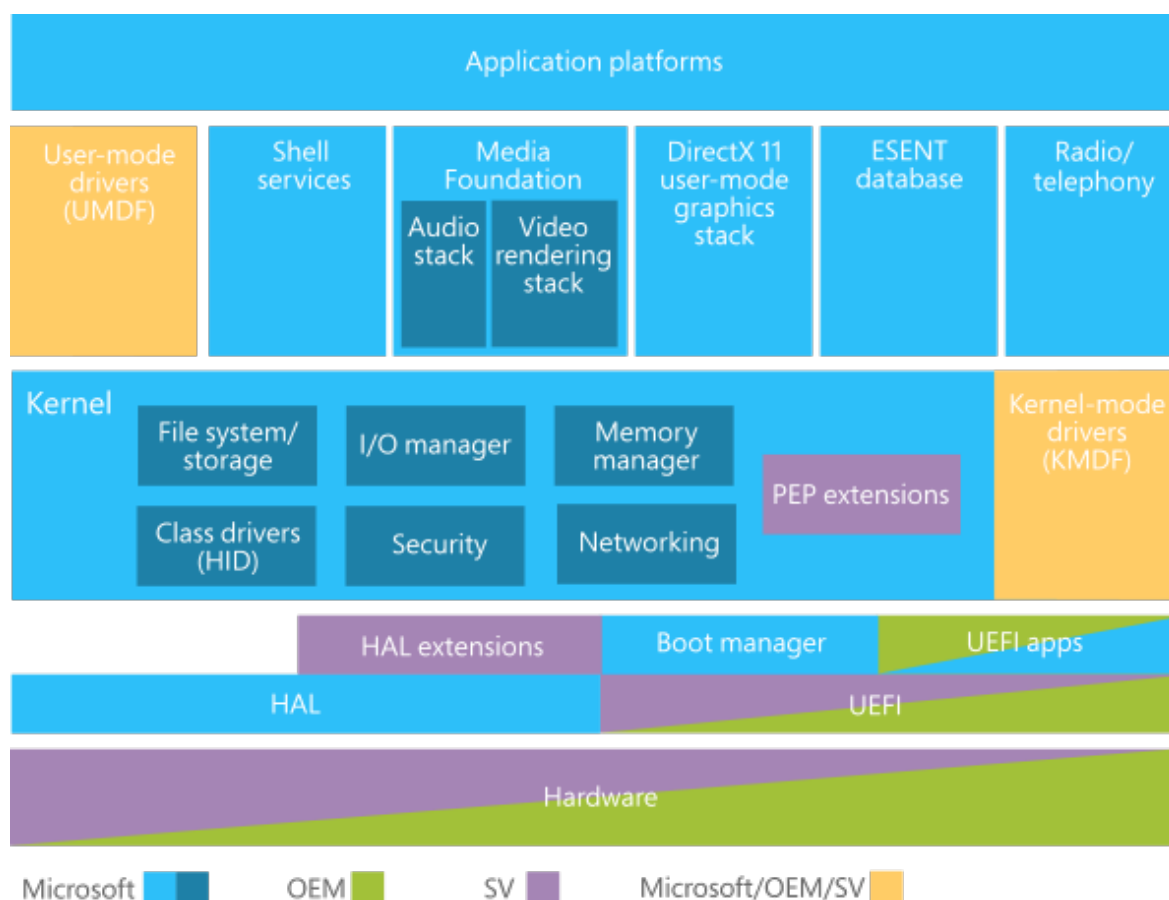
Práce Microsoftu na mobilních zařízeních začala už v roce 1992, kdy byl oficiálně zahájen vývoj Windows CE. V roce 2000 byl uvolněn Pocket PC 2000. Svým vzhledem operační systém připomínal Windows 98 a Windows 2000. Následoval Pocket PC 2002 a Windows Mobile 2003 v příslušných letech. V roce 2005 byla vydána verze Windows Mobile 5 a po dvou letech Windows Mobile 6. Poslední významná verze Windows Mobile 6.5, uvolněná pro výrobce telefonů 11. května 2009, po několika méně významných aktualizacích ukončila vývoj tohoto operačního systému. [16] [17]

Následníkem Windows Mobile se stal Windows Phone 7. Byl vyvinut poměrně rychle, což také vyústilo v nekompatibilitu s jeho předchůdcem. Na rozdíl od Androidu má Windows Phone poměrně originální grafické uživatelské rozhraní, které tvoří místo „íkonů na ploše“ tzv. dlaždice. Dalším prvkem designu nového operačního systému je strohost

zobrazení a vysoký kontrast. Tento vzhled byl před vydáním Windows 8 nazýván Metro. Nyní jsou však aplikace s tímto stylem oficiálně nazývány Windows Store apps. Nový OS je také méně upravitelný, čímž lze kontrolovat uživatelské prostředí, které nyní výrobci a uživatelé nemohou upravit. To může být výhodné, protože nadstavby a doplňky výrobců bývají často náročné na výkon a neplynulý systém má velký dopad na spokojenost uživatelů se systémem. První zařízení s Windows Phone 7 se začala prodávat v říjnu roku 2010 v Evropě a Austrálii. [18] [19]

2.3.2 Struktura Windows Phone

Obrázek 3 ukazuje strukturu operačního systému Windows Phone podle vrstev a barevně odlišuje tvůrce jednotlivých částí systému.



Obrázek 3: Struktura OS Windows Phone [20]

Nízko-úrovňová komunikace s hardwarem a ovladače pro bootování jsou poskytovány ve formě board support package (BSP). Jádro BSP je napsáno výrobcem procesoru (silicon vendor, SV), zatímco ovladače potřebné pro podporu hardwaru telefonu poskytuje výrobce koncového zařízení (OEM).

Jádro Windows Phone 7 je založeno na jádře Windows Embedded CE, zatímco jádro novější verze 8.1 je založeno na jádře Windows NT. To obsahuje minimum ze systému Windows, které bootuje, spravuje hardware a prostředky, obsahuje autentifikaci,

komunikuje po síti a obsahuje nízko-úrovňové funkce zabezpečení, doplněné některými specifickými spustitelnými soubory Windows Phone.

Vrstvu nad jádrem systému tvoří služby a programovací frameworky, které následně využívají samy aplikace. [20]

2.3.3 Programovací jazyky pro tvorbu aplikací pro Windows Phone

Jazyk C# je programovací jazyk, který se využívá pro většinu Windows Phone aplikací. Podobně jako Java je i C# překládán nejdříve do mezijazyka (IL). Mezijazyk IL však není interpretován jako bajtkód Javy, nýbrž je kompilován do strojového kódu v případě potřeby určité části programu. [21]

Programovací jazyk C++ jako nativní kód kompilovaný pro konkrétní procesor najde své využití převážně v aplikacích, nebo spíše částech aplikací, které jsou velmi náročné na výkon a optimalizaci, někdy v kombinaci s knihovnami DirectX, například akční 3D hry.

Aplikace se mohou kromě jazyka C# programovat pomocí jazyka Visual Basic. Tento jazyk se od C# liší v podstatě jen syntaxí, jelikož je překládán do stejného mezikódu jako C#. [22]

Emulátory Windows Phone jsou založeny na technologii virtualizace Hyper-V. Pokud tuto technologii procesor počítače nepodporuje, je možné testovat aplikace přímo na mobilním telefonu. V takovém případě je nutné telefon zaregistrovat k vývoji aplikací. To lze zdarma.

2.4 Technologie pro tvorbu grafických rozhraní aplikací

2.4.1 Tvorba GUI pro iOS

Vývojové prostředí Xcode obsahuje vestavěný nástroj Interface Builder. Tento nástroj umožňuje vytvořit kompletní uživatelské rozhraní bez nutnosti psát jakýkoliv kód. Toto oddělení uživatelského rozhraní a jeho implementace jsou možné díky MVC architektuře Cocoa Touch frameworku.

Celá iOS aplikace sestává z několika pohledů. Vztahy mezi nimi jsou definovány pomocí tzv. storyboards, jejichž tvorbu usnadňuje Storyboard Designer. Assistant zase umožňuje zobrazení výsledného GUI hned vedle zdrojového kódu, který definuje jeho chování. Interface Builder též podporuje systém rozložení zvaný Auto Layout umožňující výpočet velikostí prvků GUI podle určitých omezení, např. přizpůsobení textu. Struktura grafického rozhraní je uložena ve formě .nib souboru (zkratka pro neXt Interface Builder) nebo nověji s příponou .xib. Vývojářský nástroj Xcode vyžaduje ke svému spuštění počítač Mac s operačním systémem Mac OS X ve verzi Snow Leopard nebo vyšší. Pro vývoj aplikací s použitím iOS SDK a IDE Xcode je dále potřeba být zaregistrovaný jako Apple Developer. [23] [24]

Xamarin.iOS představuje alternativu k výše zmíněnému nástroji od firmy Apple. Program umožňuje také grafickou tvorbu uživatelských rozhraní. Vytvořená uživatelská rozhraní

ukládá v plně kompatibilním formátu, takže lze tytéž soubory editovat také v Xcode. Xamarin.iOS pro potřeby Designeru zpřístupňuje ve svém GUI všechny objekty uživatelského rozhraní poskytované firmou Apple. Program ke svému chodu vyžaduje počítač s Mac OS X Mountain Lion, tedy novější než zmíněný Xcode.

Možné je však vyvíjet s tímto nástrojem i v operačním systému Windows, na Visual Studiu. Bez kompilátoru od Apple však aplikaci nevytvoříme a bez certifikátů Apple aplikaci nevydáme, a tak musíme mít zároveň po síti připojený počítač s Mac OS X, který nám tyto věci zajistí a bez něhož se ani Xamarin iOS for Visual Studio nenainstaluje. Možné je také mít spuštěné Visual Studio s nástroji Xamarin iOS ve Windows uvnitř virtuálního stroje na Macu. [25] [26]

2.4.2 Tvorba GUI pro Android

Oficiální vývojový nástroj pro vývoj aplikací pro Android je Android Studio založené na vývojovém prostředí IntelliJ IDEA. První stabilní verze (verze 1.0) byla vydána teprve 8. prosince 2014. Nástroj Android Studio lze nainstalovat zdarma na všechny běžné desktopové operační systémy – Windows, Mac OS X 10.8.5 nebo vyšší a Linuxovou distribuci Ubuntu nebo Fedora s prostředím KDE, GNOME nebo Unity. Jako v každém správném nástroji pro tvorbu grafických uživatelských rozhraní je i zde obsažen WYSIWYG editor s real-time renderováním vzhledu aplikace. To znamená, že při psaní program okamžitě vykresluje zapsané změny na obrazovku. Editor také podporuje přetahování, tzv. drag and drop komponentů do uživatelského rozhraní. Android Studio je úzce spjato s open source buildovacím nástrojem Gradle. Součástí Android Studia jsou také emulátory telefonů řady Nexus a několika bezejmenných zařízení s různým rozlišením. Dále jsou zde k dispozici možnosti pro podepisování aplikací a nástroj Lint pro statickou analýzu kódu, který kontroluje zdrojový kód na výkonovou optimalizaci, kompatibilitu a další potenciální problémy. [27] [28]

Dostupným vývojovým prostředím (IDE) před vydáním Android Studia bylo Eclipse s pluginem Android Development Tools (ADT). Hlavní částí, kterou ADT obsahuje pro tvorbu grafických rozhraní, je layout editor, který se zpřístupní otevřením některého XML souboru popisujícího vzhled aplikace.

Další možnost tvorby uživatelského rozhraní představuje IDE Xamarin.Android, jehož designer, podobně jako ostatní nástroje, usnadňuje tvorbu grafických rozhraní aplikací jejich bezprostředním zobrazením v tzv. Design Surface.

2.4.3 Tvorba GUI pro Windows Phone

V případě Microsoftu máme výběr nástrojů poměrně jednoduchý. Visual Studio 2013 s Update 2 obsahuje vše, co je potřeba pro vývoj, ladění a testování aplikací pro Windows Phone 8.1, případně Windows Phone 8.0. Zmíněný vývojářský nástroj obsahuje také Visual Designer, který zobrazuje výsledný vzhled aplikace podle generovaného či napsaného kódu XAML. Tento značkovací jazyk vyvinuli ve společnosti Microsoft a je součástí kategorie funkcí Microsoft Windows Presentation Foundation (WPF), která je součástí Microsoft .NET frameworku od verze 3.5. Jádru WPF je založeno na vektorové

grafice a je optimalizované na rozdílná rozlišení obrazovek. Současně s kódem XAML popisujícím prvky uživatelského rozhraní se často používá kód v jazyce C# nebo Visual Basic, který obsluhuje události nebo manipuluje s prvky uživatelského rozhraní, podobně jako HTML a kód na pozadí ve formě Javascriptu. Soubor s tímto zdrojovým kódem má stejný název jako příslušný kód XAML, pouze přidává příponu `.cs`, resp. `.vb`, např. `App.xaml.cs`. [29] [30]

Nástrojem pro návrh grafického rozhraní Windows Phone aplikací je Microsoft Blend for Visual Studio. Tento nástroj byl do roku 2012 součástí balíku programů Microsoft Expression Studio. Dnes je volitelnou součástí instalace Visual Studia 2012 s Update 2 a novějších. Umožňuje otevírat projekty aplikací Windows Phone vytvořené v programu Visual Studio a naopak. Pro aplikace napsané v jazyce JavaScript obsahuje Blend sadu nástrojů pro práci s HTML5 a CSS3. Ty poskytují mimo jiné i tzv. interaktivní mód umožňující interakci s tvořeným rozhraním a zobrazení například prvků vytvořených při události vyvolané právě JavaScriptem. Pro aplikace napsané použitím Visual Basicu, C# nebo C++ jsou zde nástroje pro XAML pomáhající s aplikací stylů ovládacích prvků, tvorbou šablon a připojováním vzorových dat. [31] [32]

2.5 Shrnutí vývoje a distribuce aplikací pro jednotlivé mobilní platformy

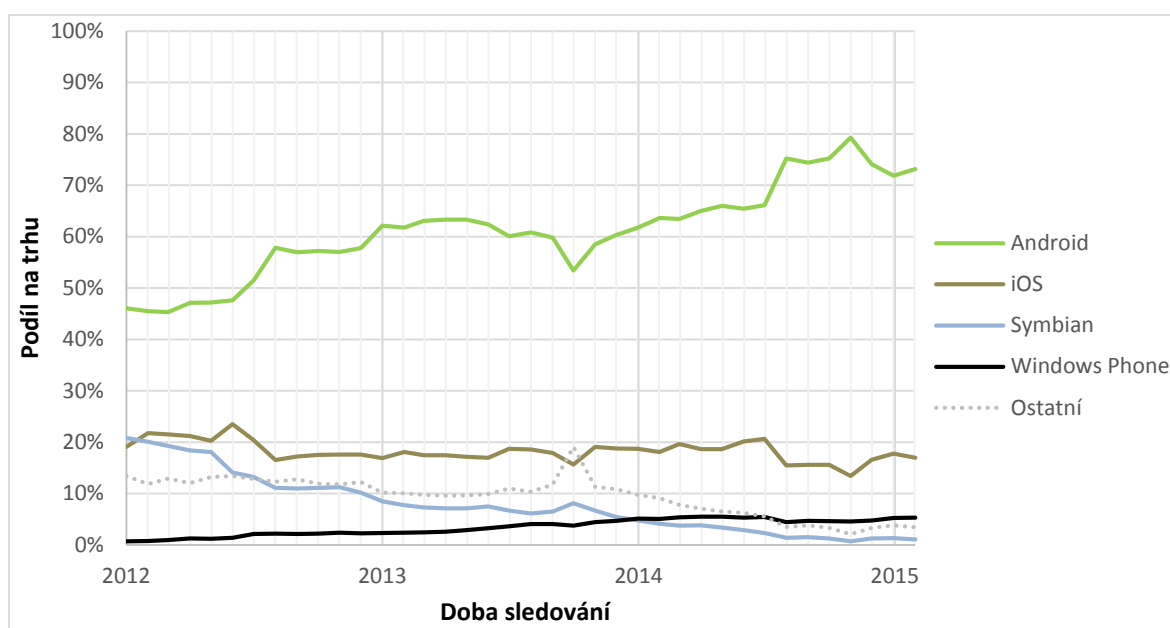
Největší nevýhodou vývoje aplikací pro iOS je jeho vázanost na systémy Apple. Pro vývoj aplikací (přesněji: jejich kompilaci) je nutný počítač Mac. Do dubna 2011 bylo k dispozici jediné vhodné vývojové prostředí, a to Xcode od společnosti Apple. Nepříjemností při tvorbě aplikací pro iOS je výše zmíněný schvalovací proces aplikace, která má být spuštěna na koncovém zařízení, včetně ročního poplatku za členství ve vývojářském programu. [33]

V tomto směru je Android pravým opakem. Vývoj aplikací může být prováděn na všech třech nejběžnějších desktopových systémech Windows, Mac OS a Linux. Pro publikování aplikace na Google Play Store je potřeba vývojářský účet, který stojí jednorázový poplatek 25 dolarů. Aplikace pro Android však může být distribuována i jinými způsoby, na rozdíl od iOS a Windows Phone. Nevýhodou Androidu oproti ostatním dvěma platformám je běh aplikací ve virtuálním stroji Dalvik, což zpomaluje jejich běh. To se však v nových verzích s příchodem Android Runtime mění. Různých zařízení, používajících ke svému běhu OS Android, je také na trhu daleko větší množství, což znesnadňuje vývoj, pokud má aplikace běžet na všech zařízeních správně. Vývoj aplikací také komplikuje poměrně časté vydávání nových verzí a s tím spojené změny API, které působí problémy při zpětné kompatibilitě aplikací určených pro nové verze.

Instalovat Microsoft Visual Studio, nutné pro vývoj aplikací pro Windows Phone, lze jen na OS Windows. Kromě distribuce aplikace na Windows Phone Store (ať už zařazené do vyhledávání, přístupné pouze pomocí určité adresy URL nebo v rámci beta testování) nebo s pomocí Visual Studia do telefonu připojeném přes USB lze využít i tzv. Company hub app, tedy aplikaci šířitelnou i mimo Windows Phone Store a podepsanou speciálním

certifikátem pro podniky. Taková aplikace pak primárně slouží jako rozcestník pro spouštění určitých podnikových aplikací. [34]

Co se rozšíření nejpoužívanějších mobilních operačních systémů v České republice týče, jasně největší zastoupení má OS Android, hlavně díky jeho otevřenosti a modifikovatelnosti, viz Obrázek 4. S tím samozřejmě souvisí i počet dostupných aplikací na Google Play Store, který se obsahuje společně s App Store od Apple přes milion aplikací. Oproti tomu Windows Phone Store obsahuje přibližně čtvrtinový počet aplikací, a ty navíc často nejsou příliš kvalitní. To však dává příležitost vývojářům, kteří si dají záležet na kvalitě. Proto byla také vybrána jako vývojová platforma pro praktickou část Windows Phone. Na Obrázku 4 je také zobrazen zastaralý operační systém Symbian, který se však v současnosti již téměř nepoužívá. [35] [36]



Obrázek 4: Podíl mobilních operačních systémů v ČR v letech 2012 až 2015 [37]

3 Aplikace pro správu finančních prostředků pro mobilní zařízení

V této části bude uveden stručný výčet některých aplikací pro správu finančních prostředků dostupných ke stažení na Windows Phone Store. Aplikace pro iOS a Android zde nejsou zahrnuty, protože obě zmíněné platformy jsou již značně rozšířené a množství aplikací pro tyto platformy je velké. Zmíněné aplikace jsou zdarma. Placené aplikace či jejich placené verze nebyly brány v potaz. U každé vybrané aplikace jsou stručně vypsány její výhody a nevýhody. Některé funkcionality těchto aplikací představovaly inspiraci pro praktickou část této práce.

3.1 Budget Buddy

První vyzkoušenou aplikací byla aplikace Budget Buddy¹. Značnou a v této oblasti nepříliš obvyklou vadou této aplikace jsou její pomalé reakce uživatelského rozhraní, někdy v délce skoro jedné sekundy.

Některé části aplikace byly od sebe vizuálně odděleny, takže se ztrácela jejich vzájemná propojenost, což místy ztěžovalo přehlednost. Nakonec se stalo, že aplikace nešla vůbec spustit. Po zobrazení načítacího obrázku se aplikace ukončila, a to i po aktualizaci.

3.2 Aplikace iMoney

Poměrně jednoduchá aplikace iMoney² prezentuje kategorie výdajů ve formě dlaždic s obrázky, do kterých se vkládají výdaje s poznámkami. Vytvořit lze i vlastní kategorie, ty však již neobsahují obrázky.

Peněžní částky lze však zadávat pouze v dolarech, a také data nejsou vypisována v českém formátu.

3.3 Money Tracker Free

Zdarma dostupná verze, aplikace Money Tracker Free³, představuje další alternativu pro správu financí. Její rozhraní není nejjednodušší, ale poskytuje některé užitečné funkce, například kalkulačku při zadávání peněžní částky nebo export do Excelu.

3.4 My Expenses

Aplikace My Expenses⁴ má poměrně jednoduchou koncepci uživatelského rozhraní. To představuje úvodní stránku typu Hub. Ta pomocí ikon ve skupinách odkazuje na ostatní

¹ Dostupné z: <http://www.windowsphone.com/cs-cz/store/app/budget-buddy/baa325fc-9b09-413e-9089-be1028380bbc>

² Dostupné z: <http://www.windowsphone.com/cs-cz/store/app/imoney/336b38f6-7f04-4bc6-88c1-b0d1c1bc5f8d>

³ Dostupné z: <http://www.windowsphone.com/cs-cz/store/app/money-tracker-free/3b70ddb7-7ba4-4a6b-8f27-f51ee89af50d>

stránky se zadáním či výpisem příslušných údajů. Jednotlivé stránky používají řadu různých barev písma, což nemusí ze stylistického hlediska každému vyhovovat, stejně jako přítomnost reklam. Výhodou této aplikace je možnost nastavení fixních (pravidelných) příjmů i výdajů.

⁴ Dostupné z: <http://www.windowsphone.com/cs-cz/store/app/my-expenses/e54ae512-7266-e011-81d2-78e7d1fa76f8>

4 Návrh a implementace aplikace

Tato část se zabývá návrhem a implementací systému pro správu finančních prostředků pro mobilní zařízení. Aplikace se jmenuje Peněženka a je tvořena pro operační systém Windows Phone 8.1 z důvodů zmíněných ve 2. a 3. kapitole. Celý vývoj aplikace se provádí v nástrojích Microsoft Visual Studio 2013 a Microsoft Blend for Visual Studio 2013.

Aplikace je psána v jazyce C# s využitím .NET frameworku pro Windows Phone aplikace. Pro popis grafického rozhraní aplikace je využíván značkovací jazyk XAML. K trvalému ukládání dat byla zvolena databáze SQLite místo souboru ve formátu JSON, který není pro větší objem dat vhodný, neboť při parsování dat uvnitř je nutné mít všechna data v paměti.

4.1 Požadavky na aplikaci

Pro plnění účelu, pro který je aplikace tvořena, by měla aplikace obsahovat následující funkcionality:

- zaznamenávat příjmy a výdaje,
- nastavovat fixní příjmy a výdaje,
- možnost spravovat více peněženek,
- filtrování záznamů pomocí kategorií a času a
- zobrazení grafů.

Dalším cílem vytvořené aplikace je její uživatelská přívětivost. Nový uživatel by se měl v uživatelském rozhraní rychle zorientovat, a zároveň by neměl postrádat žádné důležité funkce.

4.2 Použité návrhové vzory

Aplikace pro práci s daty a jejich reprezentaci uživateli bude využívat MVVM architekturu. Návrhové vzory, jako je tento, usnadňují práci a přehlednost u větších projektů, na kterých pracuje typicky více lidí.

4.3 Architektura MVVM

Návrhový vzor MVVM odděluje data od uživatelského rozhraní, což umožňuje práci na obou částech odděleně. Jde o variaci známého vzoru MVC (Model View Controller). Použití MVC architektury ve frameworkcích založených na XAML není příliš vhodné kvůli využití data bindingu, který umožňuje automatické zobrazení změny zdrojových dat v uživatelském rozhraní nebo změnou v uživatelském rozhraní měnit zdrojová data.

Model reprezentuje třídu nebo sadu tříd, které popisují strukturu uchovávaných dat. Pokud má změna nějakého atributu v modelu vyvolat příslušnou změnu v uživatelském rozhraní pomocí data bindingu, musí třída představující model implementovat rozhraní `INotifyPropertyChanged`, což zahrnuje obsluhu události `PropertyChanged`. Aby atribut

tedy informoval příslušné pohledy (Views) o své změně, je v jeho setteru vyvolána událost `PropertyChanged`.

Vrstva **ViewModel** představuje abstrakci mezi modelem a pohledem. Často obsahuje kolekce objektů založených na modelu a například metody získávající data do těchto kolekcí.

View zobrazuje a formátuje data získaná z ViewModelu. Bývá nejčastěji napsán v XAML, jako v našem případě. Jsou zde deklarativním způsobem zadány informace o data bindingu, což odděluje data od popisu uživatelského rozhraní.

4.4 Data binding

Efektivní propojení zdrojových dat a uživatelského rozhraní umožňuje data binding. Ten podle zvoleného módu reaguje na změny ve zdrojových datech anebo v uživatelském rozhraní. Při použití data bindingu se v cílovém elementu XAML použije rozšíření Markup extension `{Binding}`.

Propojení může mít různé směry. Ty jsou definovány pomocí vlastnosti `Mode` bindovacího objektu. Existují tři módy data bindingu:

- Mód **TwoWay** umožňuje jak změnu v uživatelském rozhraní při změně zdrojových dat, tak i změnou v uživatelském rozhraní změnit zdrojová data. Je tak vhodný typicky pro různé formulářové elementy.
- Mód **OneWay** je nastaven jako výchozí pro většinu prvků uživatelského rozhraní. Umožňuje pouze zobrazení dat, tedy pouze čtení.
- V případě módu **OneTime** je cílový prvek změněn pouze při inicializaci cílového prvku a následné změny nejsou propagovány, což poskytuje o něco lepší výkon v případě, kdy se zdrojová data nemění. [38]

Zdrojem dat pro data binding může být CLR (Common Language Runtime) objekt včetně elementu uživatelského rozhraní, který produkuje třída definovaná v C# nebo Visual Basicu, nebo objekt Windows Runtime, který obsahuje `BindableAttribute` nebo implementuje rozhraní `ICustomPropertyProvider`. Windows Runtime objekty jsou definovány v jazyce C++. [39]

Pro příklad zapsání data bindingu v XAML mějme třídu `MujZaznam`.

```
public class MujZaznam {  
    public string MujText { get; set; }  
}
```

Vlastnost `MujText` třídy `MujZaznam` lze připojit do elementu XAML třemi způsoby:

```
<!-- první -->  
<TextBlock Text="{Binding Path=MujText}" />  
  
<!-- druhý -->  
<TextBlock Text="{Binding MujText}" />
```

```

<!-- třetí -->
<TextBlock>
  <TextBlock.Text>
    <Binding Path="MujText" />
  </TextBlock.Text>
</TextBlock>

```

Tyto tři způsoby jsou identické funkcí i výkonem, liší se tedy pouze syntaxí. [40]

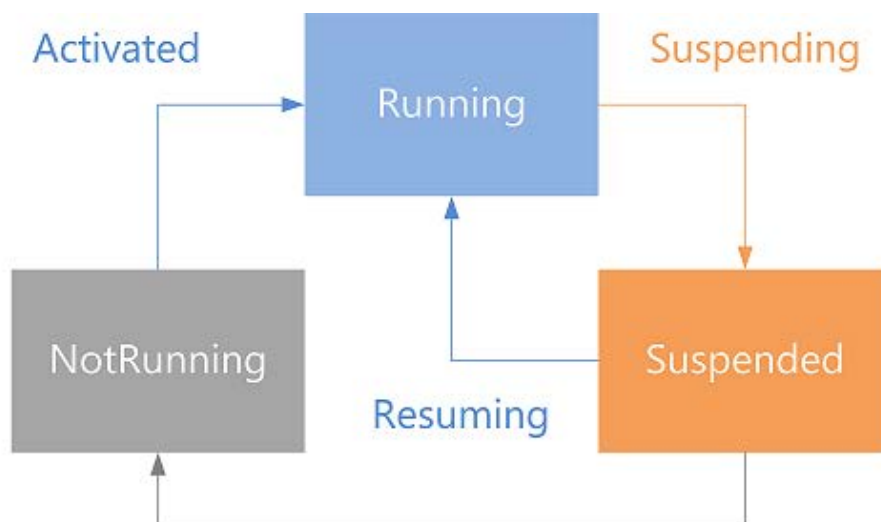
Zdroj dat pro data binding je ve výchozím stavu definován vlastností `DataContext` nadřazeného elementu. V případě jiného zdroje můžeme předchozí vlastnost obejít nastavením vlastnosti `Source` a zde definovat jiný zdroj dat.

4.5 Životní cyklus aplikace

Na rozdíl od programů či aplikací pro stolní počítače založené na Win32 API, které mohou v oknech běžet souběžně, v OS Windows Phone (obecně v aplikacích pro Windows Runtime) je tomu jinak. Při přepnutí na jinou aplikaci se ta předtím spuštěná pozastaví a přechází do stavu `Suspended`. V popředí tedy běží vždy jen jedna právě používaná aplikace, což šetří prostředky telefonu, a tedy i výdrž baterie. Aplikace na popředí má tak k dispozici více procesorového času, což je znatelné na plynulosti uživatelského rozhraní. Aplikace se může nacházet v jednom ze tří stavů:

- `Running` neboli běžící,
- `Suspended` neboli odložená nebo uspaná a
- `NotRunning` neboli neběžící.

Jednotlivé stavy a přechody mezi nimi ilustruje Obrázek 5. Za ním následuje popis jednotlivých stavů.



Obrázek 5: Životní cyklus aplikace [41]

Při testování aplikace pro Windows Phone 8.1 v IDE nelze ladit vzniklé stavy uspání, probuzení nebo uspání a následné ukončení, například při stisknutí tlačítka Zpět. Proto se v panelu nástrojů Visual Studia nachází přepínač, který tyto stavy simuluje.

4.5.1 Stav Running

Do tohoto stavu, kdy aplikace běží na popředí, se lze dostat dvěma způsoby.

První je klasické spuštění předtím neběžící aplikace, tedy Activation. Tento stav může nastat i v případě aplikace, která byla přesunuta do pozadí a z důvodu nedostatku paměti byla z paměti odstraněna. Při načítání aplikace se objeví úvodní obrazovka (angl. splash screen). Podle doporučení Microsoftu pro Modern UI aplikace by doba načítání aplikace neměla přesáhnout 5 vteřin. Jakmile jsou všechny potřebné hodnoty připravené, úvodní obrazovka zmizí a zobrazí se uživatelské rozhraní. Stav se změní ze stavu NotRunning na stav Running.

Druhé je obnovení aplikace přesunuté na pozadí neboli Resuming. Jelikož aplikace stále zůstává v paměti, je obnovení poměrně rychlé a bývá zachován stav stránek a formulářů.

4.5.2 Stav Suspended

Do tohoto stavu se lze dostat buď stiskem tlačítka Start nebo Hledat, přepnutím na jinou aplikaci podržením tlačítka Zpět nebo přechodem zařízení do úsporného režimu. Jedná se o stav, kdy aplikace je uchovávána v operační paměti, ale je zastaven všechen kód a časovače a nejsou vyvolávány žádné události.

Když uživatel přepne aplikaci na pozadí, aplikace ještě 10 vteřin čeká, jestli se uživatel nepřepne zpět na tuto aplikaci. Po tuto dobu je aplikace považována za běžící. Potom operační systém aplikaci uspává. Po uspání může být aplikace kdykoliv v případě potřeby odstraněna z paměti. O tomto odstranění není aplikace nijak informována. Proto při přechodu ze stavu Running do stavu Suspended má aplikace 5 vteřin na uložení všech potřebných dat. To může být pro ukládání velkého objemu dat příliš krátká doba. Dobře navržená aplikace by tak měla svá data ukládat průběžně, nejlépe při jejich změně a při obsluze události OnSuspending uložit jen stav uživatelského rozhraní. [42]

4.6 Databáze

Jak již bylo zmíněno v úvodu této kapitoly, pro ukládání dat byla zvolena databáze SQLite. Zdrojové kódy databáze SQLite jsou pod licencí public domain, a tudíž bez omezení použitelné jak soukromě, tak komerčně. Na rozdíl od většiny SQL databázových systémů nemá engine SQLite zvláštní serverový proces. Celá databáze SQLite se všemi tabulkami, pohledy a triggerem je uložena v jediném souboru. Nadto je tato knihovna poměrně kompaktní. Se všemi funkcemi dosahuje velikosti okolo 500 KiB. [43]

4.6.1 Instalace knihovny SQLite

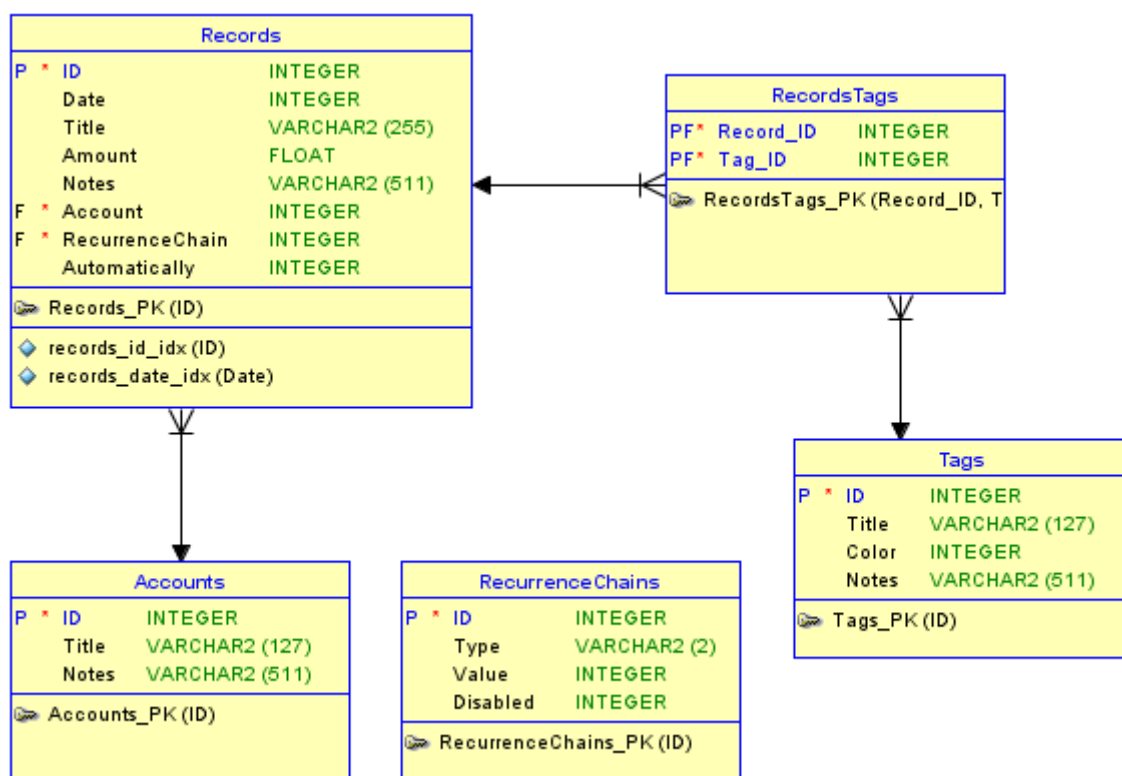
Aby bylo možné vyvíjet aplikaci pro Windows Phone 8.1, která bude používat SQLite, je nejdříve nutné do programu Visual Studio nainstalovat rozšíření SQLite for Windows Phone 8.1. Tímto způsobem se nainstalují příslušné .dll knihovny do globálního umístění

Visual Studio. Pro přiřazení rozšíření k projektu aplikace je ještě potřeba v okně Solution Explorer přidat rozšíření v záložce References.

Jelikož je samotný engine SQLite napsaný v jazyce C a aplikace samotná v jazyce C#, který je nejdříve překládán do mezijazyka, potřebujeme tzv. wrapper. Těch existuje celá řada, ale pro tuto aplikaci byl vybrán Portable Class Library for SQLite (zkráceně SQLite PCL) od společnosti Microsoft Open Technologies Inc., který podporuje komunikaci s databází pomocí SQL dotazů. Knihovna SQLite PCL je distribuována pod licencí Apache License 2.0, která nám trvale dovoluje používat, prodávat či tvořit odvozená díla od takto licencovaného objektu.⁵ Wrapper stáhneme a nainstalujeme v podobě NuGet balíčku. Ten zahrneme do naší aplikace podobně jako u samotného engine SQLite přidáním reference v záložce References. Po přidání této reference se mezi referencemi objeví ještě reference na samotné SQLite ve verzi stejné, jakou má wrapper. Tato reference je však nefunkční, proto ji odstraníme.

4.6.2 Databázový model

Samotná databáze v aplikaci sestává z pěti tabulek pro záznam příjmů a výdajů a s nimi souvisejících údajů. Tabulky a relace mezi nimi jsou zobrazeny na Obrázku 6.



Obrázek 6: Relační Model databáze [zdroj: autor]

Nejdůležitější tabulku představuje samozřejmě tabulka *Records*, která uchovává příjmy a výdaje. Každý záznam je jednoznačně identifikován polem *ID*. Datum záznamu je uloženo ve sloupci *Date* jako celé číslo ve formátu rokměsícden, např. 20150418. Čas se

⁵ Licence je dostupná na <https://sqlitepcl.codeplex.com/license>

neuchovává, neboť v praxi si člověk zaznamená, co ten den utratil, případně ve správném pořadí, a konkrétní čas zde nehraje velkou roli. Sloupce `Title` a `Notes` obsahují název a poznámky k položce. Pokud uživatel žádný název nezadá, ve výpise položek se objeví „Položka bez názvu“. V sloupci `Amount` se nachází desetinné číslo peněžní částky. Znaménko čísla určuje, jde-li o příjem či výdaj. Spočítání bilance se tak realizuje formou jednoduchého součtu těchto hodnot. Cizí klíče `Account` a `RecurrenceChain` se potom odkazují na příslušné tabulky, přiřazující záznamu účet, resp. řetězec opakování fixního záznamu. Poslední položka `Automatically` určuje, zda byl záznam vytvořen automaticky při přidávání nových fixních záznamů programem. Nad sloupci `ID` a `Date` je dále vybudován index, který urychluje vyhledávání položek podle těchto sloupců.

Tabulka `Tags` uchovává barevné štítky vytvořené uživatelem. Štítky v této aplikaci slouží jako forma kategorií. Uživatel bude moci ke každému příjmu a výdaji přiřadit jeden nebo více barevných štítků. Každý záznam v této tabulce má své unikátní `ID`. Sloupec `Title` obsahuje název štítku a sloupec `Notes` poznámky ke štítku. Pole `Color` obsahuje barvu uloženou jako datový typ `uint` ve formátu `ARGB`. Toto číslo je následně konvertováno na instanci tříd `Color` a `SolidColorBrush`, které využívá uživatelské rozhraní.

Předchozí dvě tabulky `Records` a `Tags` propojuje tabulka `RecordsTags`, která obsahuje pouze cizí klíče odkazující na primární klíče těchto tabulek. Oba cizí klíče jsou zároveň primárními klíči.

Účty, ke kterým každý záznam přísluší, jsou uchovány v tabulce `Accounts`. Tato tabulka obsahuje sloupce `ID` účtu, název a poznámky k účtu. Při vytváření účtu v aplikaci je možné nastavit počáteční zůstatek. Ten se uloží formou prvního záznamu tabulky `Records` s cizím klíčem na tento účet.

Poslední tabulkou v naší databázi je tabulka `RecurrenceChain`, umožňující správu a generování fixních záznamů. Sestává z primárního klíče `ID`. Dále je formou textu určen typ opakování fixního záznamu. Toto pole nabývá hodnot „W“, „M“ a „Y“ pro týdenní, měsíční a roční opakování. Sloupec `Value` pak určí, který den v týdnu/měsíci/roce bude nový záznam generován.

4.6.3 Práce s databází SQLite pomocí wrapperu SQLite PCL

Práce s SQL PCL je poměrně snadná. Spojení s databází se uskuteční vytvořením nové instance třídy `SQLiteConnection` s názvem souboru jako parametrem konstruktoru. Toto spojení je uchováváno ve statické třídě `DB`, neboť potřebujeme vždy jen jedno spojení. V této třídě se také nachází metoda `PrepareDatabase`, která při startu aplikace vytvoří všechny potřebné tabulky, pokud neexistují. Také do prázdných tabulek `Accounts` a `RecurrenceChains` vloží nultý řádek kvůli cizím klíčům v tabulce `Records`. Tato třída také obsahuje metody pro automatické přidání fixních záznamů a pro vymazání všech záznamů v tabulkách.

Pro sestavení SQL dotazu se používá metoda `Prepare` volaná na instanci třídy `SQLiteConnction`. Metoda vrací hodnotu typu `SQLiteStatement`, se kterou se dále pracuje. Jako parametr metody se vkládá řetězec – SQL dotaz. Tento dotaz může obsahovat zástupné znaky či řetězce pro vkládání hodnot do SQL dotazu, což pak slouží také jako prevence proti vložení cizího SQL a narušení předpokládané funkce dotazu, tzv. SQL injection. Jako zástupný znak se používá otazník a vloží se na místo předpokládané hodnoty. Zástupných znaků může být více a při přiřazení jednotlivých hodnot se rozlišují podle pořadí, jak lze vidět na následujícím příkladu. Práce se zástupnými řetězci namísto znaků je obdobná. Otazník nahradí řetězec ve formátu „@retezec“. Ten se poté použije také jako parametr metody `Bind`. Provedení dotazu se uskuteční metodou `Step`.

```
var stmt = DB.Conn.Prepare(@"INSERT INTO RecordsTags (Record_ID, Tag_ID)
VALUES (?,?)");
stmt.Bind(1, recordId);
stmt.Bind(2, tag.ID);
stmt.Step();
```

Získávání dat z databáze je podobné. Metoda `Step` získá následující řádek z databáze. Metoda jako taková vrací informaci o stavu zpracovaného dotazu v podobě výčtového typu `SQLiteResult`. V případě, že metoda `Step` vrátí výsledek `SQLiteResult.DONE`, dotaz byl vykonán, ale z databáze nebyly vráceny žádné hodnoty. Pokud je vrácen `SQLiteResult.ROW`, obsahuje hodnota typu `SQLiteStatement` řádek hodnot, které se získávají metodami `GetInteger`, `GetFloat`, `GetText`, apod. s parametrem čísla sloupce v pořadí nebo názvu sloupce. Pokud metoda vrací hodnotu z pole o hodnotě `null`, vyvolá tato metoda výjimku `SQLiteException`.

4.7 Tvorba uživatelského rozhraní

Uživatelské rozhraní aplikace je, jak už bylo zmíněno, definováno pomocí značkovacího jazyka XAML. Mobilní telefony a tablety obsahují displeje různých velikostí a rozlišení, čímž se mění hustota pixelů. V případě, že jsou rozměry prvků uživatelského rozhraní definovány v pixelech, prvky budou na různých displejích různě velké. Takové chování není žádoucí. Proto je tvorba uživatelského rozhraní ve WPF optimalizována pro zařízení s různými rozměry a rozlišeními displeje. Velikosti prvků jsou v XAML zapisovány v jednotkách `independent pixel`, které pracují právě s hustotou pixelů. V případě elementu `Grid` lze rozměry řádků a sloupců zadat také pomocí násobků zbylého místa po vykreslení částí s pevným rozměrem nebo s rozměrem `Auto`, a to symbolem hvězdička. [44]

Responzivní design je zde také uplatněn, když prvky rozhraní přizpůsobují svou velikost k velikosti displeje. Typicky mají tyto prvky nastaven atribut `HorizontalAlignment` nebo `VerticalAlignment` na hodnotu `Stretch` se současným nastavením šířky resp. výšky na hodnotu `Auto`, což jsou výchozí hodnoty.

Při tvorbě uživatelského rozhraní lze také využít styly a šablony. Styly umožňují prvkům GUI, odvozeným od třídy `FrameworkElement` nebo `FrameworkContentElement`, sdílet informaci o jejich vzhledu, což snižuje duplicitu v kódu a usnadňuje tak jeho údržbu. Styly

mohou být definovány na úrovni stránky nebo aplikace v části `Resources`. Definice v externím souboru je také možná. Šablony se nejčastěji používají pro definici jednotlivých prvků v seznamech, tagem `DataTemplate`. Mohou být definovány buď přímo v elementu `GUI` nebo v části `Resources`, podobně jako styly. Dále existují šablony pro definici ovládacích prvků `GUI ControlTemplate` a šablony pro hierarchická data `HierarchicalDataTemplate` (ty však v této práci nebyly využity). Globálně dostupné styly a šablony se nachází v souboru `App.xaml`, v elementu `Application.Resources`.

4.7.1 Stránky uživatelského rozhraní

Zobrazované uživatelské rozhraní se skládá ze stránek, tzv. `Pages`. Každá stránka uživatelského rozhraní v projektu sestává ze souboru se samotným rozhraním zapsaným v XAML s příponou `.xaml`, například `HubPage.xaml`, a kódu na pozadí se stejným názvem včetně přípony, plus s příponou `.cs`, například `HubPage.xaml.cs`.

Stránka `GUI` je vždy uvedena tagem `Page`. V něm je deklarován název třídy atributem `x:Class`, ke které je rozhraní vázáno. Ta je vždy potomkem třídy `Page` a obsahuje příslušný kód na pozadí, například obsluhu událostí. Dále se zde nacházejí jmenné prostory používané na stránce, určené atributem `xmlns` s používaným zástupným názvem, a atribut `DataContext`, určující, odkud budou brána data pro data binding vnořených elementů.

Každá stránka může obsahovat tzv. `BottomBar`. Jedná se o vysunovací spodní proužek, který obvykle ve své horní části obsahuje kulatá ovládací tlačítka, tzv. `AppBarButton`. V naší aplikaci tato tlačítka slouží například pro uložení záznamu nebo k navigaci na nějakou stránku. Do těchto tlačítek lze vložit obrázek, znak nebo vektor. Pod těmito tlačítky se dále může nacházet seznam sekundárních textových tlačítek odkazujících na stránku s nastavením a podobně.

O zobrazení instancí stránek, navigaci mezi nimi a historii navigace mezi stránkami se stará třída `Frame`. Při startu aplikace ve třídě `App`, která je potomkem třídy `Application`, je instance třídy `Frame` umístěna jako statický atribut do třídy `Window`. Té se poté případně nastaví kolekce animovaných přechodů anebo registrace událostí. Každý `Frame` si eviduje svoji historii stránek ve formě zásobníku (LIFO), ze kterého se při stisknutí tlačítka `Zpět` implicitně vybírá poslední vložená stránka. Navigace mezi stránkami vypadá následovně:

```
Frame.Navigate(typeof (NewTabPage), parametr);
```

Jako druhý parametr při navigaci lze nové stránce poslat nějakou informaci, například upravovaný záznam.

4.7.2 Úvodní stránka

Nejdůležitějším prvkem této stránky je element `Hub`. Jde o speciální prvek, který sestává z horizontálně seřazených sekcí, které tvoří element `HubSection`. Každá sekce je užší než šířka displeje. Na pravém okraji tak vždy vidíme kousek sekce následující. Nad těmito sekcemi se nachází nadpis, který se pomalu posouvá, jak listujeme sekcemi do stran. Prvek `Hub` může mít specifikováno pozadí, které se posouváním mezi sekcemi také pomalu

posouvá, čímž je navozen efekt paralaxy. Tento element nepodporuje otočení displeje na šířku, proto není toto otočení řešeno ani v jiných částech aplikace.

První sekce v této aplikaci obsahuje tři tlačítka s obrázky, a to pro přidání příjmu, pro přidání výdaje a pro správu peněženek nebo účtů.

Druhá sekce vypisuje uložené příjmy a výdaje, které odpovídají zadanému filtrování. V tomto výpise jsou přítomny téměř všechny dostupné údaje, kromě polí `ID`. Pod seznamem příjmů a výdajů se nachází součty příjmů a výdajů vypsanych položek a celkový současný zůstatek. Všechny peněžní částky jsou formátovány pomocí funkce `string.Format` s parametrem pro měnu a parametrem `CultureInfo.CurrentCulture`, uchovávajícím nastavenou lokalizaci uživatelského rozhraní. Spodní proužek v této a následující sekci zobrazuje tlačítko pro navigaci na stránku filtru položek, kde se upravuje filtr položek pro tento výpis a pro grafy.

Za výpisem příjmů a výdajů se nachází sekce s grafy. Jsou zde výsečové grafy pro příjmy a výdaje, které odpovídají zadanému filtrování, seskupené podle štítků. Pod nimi je zobrazen lineární graf celkové bilance za zvolené časové období. Tento graf si automaticky nastavuje hodnoty os, což usnadňuje vývojáři práci. Pro všechny grafy byly jejich legendy přesunuty pod ně. Tak dojde k lepšímu využití dostupného prostoru.

Čtvrtá sekce vypisuje všechny očekávané fixní (opakované) příjmy a výdaje. Výpis má téměř stejnou formu jako položky ve druhé sekci. Pouze rámeček každé položky je vykreslen oranžovou barvou místo bledě modré.

Poslední, pátá část, pak náleží správě štítků. Jsou zde vypsány všechny dostupné štítky s jejich názvy, poznámkami a přiřazenou barvou. Spodní proužek v této sekci obsahuje tlačítko pro přidání nového štítku.

4.7.3 Stránka přidání a úpravy příjmu a výdaje

Tato a všechny další stránky, kromě té pro zadání hesla, obsahují dva nadpisy. První velkými písmeny označuje název aplikace a druhý již zobrazuje název konkrétní stránky. Tento styl aplikuje šablona Visual Studio pro přidání nové základní stránky, která zde byla využita.

Stránka může sloužit ke čtyřem účelům: přidání příjmu, přidání výdaje, úpravu příjmu nebo úpravu výdaje. Proto jsou zde k dispozici čtyři různé nadpisy. Při načítání stránky v metodě `NavigationHelper_LoadState` se podle navigačního parametru jeden z těchto nadpisů zviditelní.

Stránka se skládá převážně z formulářových prvků. Jako první se zde nachází element `ComboBox`, pomocí kterého lze vybrat z dostupných účtů, ke kterému bude záznam patřit. Pod ním se nachází prvek pro textový vstup `TextBox`, do kterého se zadává název položky. Pokud uživatel žádný název nezadá, uloží se do databáze prázdný řetězec, ale na výpise v úvodní stránce se pomocí `Converteru` pro data binding zobrazí „Položka bez názvu“.

Za názvem se nachází standardní `DatePicker` pro výběr data položky. Vedle něho je umístěno pole pro zadání peněžní částky příjmu nebo výdaje. Zadávání dat do pole je omezeno na čísla tím, že se uživateli při zadávání dat zobrazí klávesnice pouze s čísly a desetinnou čárkou. Vlevo od tohoto pole se v případě výdaje nachází znak mínus. Vpravo je zobrazen symbol měny podle třídy `CultureInfo` a vnořeného atributu `CurrencySymbol`. Pokud se jedná o úpravu příjmu nebo výdaje, zobrazí se pod částkou ještě zaškrtačací políčko, element `CheckBox`, pro změnu příjmu na výdaj a naopak. Kód na pozadí pak podle toho, zda se jedná o příjem či výdaj změní znaménko zadaného vždy kladného čísla.

Pod vstupy pro datum a peněžní částku se nachází textový vstup pro zadání poznámek. Ten na rozdíl od pole pro název přijímá také nové řádky.

Následuje nastavení opakování zadávaného záznamu. To se skládá z `CheckBoxu`, který indikuje, zda bude zadávaný záznam fixní nebo ne, a dvou až tří `ComboBoxů`. První z nich označuje, jak často se bude záznam opakovat. Je na výběr z hodnot týdně, měsíčně a ročně. Druhý, případně třetí označuje, který den v týdnu/měsíci/roce se výdaj vyskytne. Po uložení záznamu nebo spuštění aplikace se kontrolují všechny aktivní fixní příjmy a výdaje, které nebyly úpravou zrušeny, na jejich vložení do tabulky příjmů a výdajů.

Na konec se vybírá z dostupných štítků, které mají být záznamu přiřazeny. To je realizováno elementem `GridView` s parametrem `SelectionMode="Multiple"`. Jde o zaškrtačací mřížku barevných obdélníků. Problém se zde vyskytl při programovém zaškrtačování políček při úpravě záznamu, které se mělo stát přidáním objektů do kolekce `SelectedItems` prvku `GridView`. Přidání samotné bylo sice úspěšné, avšak bez vizuálního efektu. Proto byl místo toho přidán při úpravě příjmu/výdaje další `GridView` stejný jako je ve výpise příjmů a výdajů na úvodní stránce, pro zobrazení aktuálně přiřazených štítků. Štítky tedy musí být při úpravě přiřazeny vždy znovu.

Poslední část stránky je `BottomBar`, který obsahuje tlačítko pro uložení záznamu do databáze a tlačítko „storno“, které pouze naviguje na předchozí stránku.

4.7.4 Stránka přidání a úpravy štítku

Pro vytvoření nebo editaci štítku je možné zadat tři údaje – název, barvu a poznámky. Pro zadání názvu a poznámek jsou zde k dispozici textová pole, podobně jako u zadání příjmu a výdaje.

Mezi nimi se nachází tlačítko pro výběr barvy. Toto tlačítko obsahuje barevný čtvereček podle vybrané barvy, který tvoří element `Border`, a element `TextBlock`, vypisující název vybrané barvy. Při kliknutí na toto tlačítko se zobrazí vyskakovací okno, tzv. `Flyout`, s dostupnou paletou pojmenovaných barev. Pojmenované barvy jsou implementovány formou pole řetězců s názvy a polem čísel typu `uint`, které jsou na stránce spárovány do objektů typu `ColorItem` a vloženy do kolekce. Jde opět o element `GridView` s přiřazenou kolekcí barev. Při klepnutí na nějakou jeho položku se vybere příslušná barva a `Flyout` se zavře.

Spodní proužek opět obsahuje tlačítka „uložit“ a „storno“

4.7.5 Stránka Filtr položek

Na tuto stránku se lze dostat, jak již bylo zmíněno, tlačítkem „filtrovat“ z 2. a 3. sekce úvodní stránky.

Jako první lze položky filtrovat podle data. Tento filtr je použit vždy. Když totiž uživatel zapne aplikaci, implicitně se vytvoří filtr s daty prvního a posledního dne aktuálního měsíce. Lze zde tedy nastavit počáteční a koncové datum, které vybere položky, jež do tohoto intervalu spadají, a to včetně hraničních dat. Výskyt většího počátečního data než koncového není nijak validován. Výsledek této akce pouze nevrátí žádné položky.

Druhá část filtru specifikuje, které štítky musí příjem nebo výdaj obsahovat. Buď může uživatel zaškrtnout `CheckBox` „Všechny štítky“, nebo vybrat některé ze stejného seznamu jako předchozí stránce. Kvůli stejnému problému s programovým zaškrtnutím vybraných položek `GridView` se zde také nachází výčet štítků aktuálně platného filtru.

Poslední část filtru obsahuje zaškrťovací seznam všech účtů, který filtruje položky podle toho, k jakému účtu patří. Seznam je realizován elementem `ListView`, opět s atributem `SelectionMode="Multiple"`. Také zde se nachází `CheckBox` pro výběr všech účtů.

Spodní proužek stránky obsahuje tlačítko pro přijetí filtru a tlačítko „storno“ pro navigaci zpět. Přijetí filtru vytvoří ze zadaných hodnot objekt podle třídy `Filter`, který je navíc serializován do textového řetězce ve formátu JSON. Tento řetězec je poté poslán jako druhý navigační parametr při navigaci na úvodní stránku, kde je opět deserializován. To se děje proto, že při přechodu aplikace do stavu `Suspended` se veškerá navigace i s parametry ukládá do souboru. Proto by navigační parametry měly být jen základní datové typy, jako číslo nebo řetězec.

4.7.6 Stránka správy účtů

V této stránce se kromě nadpisů nachází pouze seznam dostupných účtů a spodní proužek. Seznam dostupných účtů nezobrazuje řádek tabulky `Accounts` s `id=0`, který se v tabulce nachází od jejího vytvoření a který nelze uživatelsky smazat. Při klepnutí na některý účet v seznamu je uživatel navigován na stránku úpravy účtu. Podržení některé položky zobrazí element `MenuFlyout` neboli vyskakovací menu s tlačítky pro úpravu nebo smazání účtu. Více informací o mazání účtu včetně snímků obrazovky se nachází v kapitole Uživatelská část.

Spodní proužek obsahuje tlačítko pro přidání nového účtu.

4.7.7 Stránka pro přidání a úpravu účtu

Navigace do této stránky se, jak již bylo zmíněno, děje ze stránky se seznamem účtů. Každý účet má název a poznámky, které se upravují podobně jako ve výše zmíněných stránkách. Tvorba nového účtu navíc umožňuje přidat počáteční zůstatek, který se po vytvoření účtu vloží jako nový záznam do tabulky `Records` s cizím klíčem na tento účet.

4.7.8 Stránka nastavení

Do nastavení se lze dostat pomocí sekundárního tlačítka „nastavení“ spodního proužku úvodní stránky. O možnostech v nastavení si povíme v následujících odstavcích.

První možností je nastavení vyžadování hesla po spuštění aplikace. Požadování hesla se zapne/vypne zaškrtačím tlačítkem „Požadovat heslo“. Zaškrtnutím tohoto políčka se pomocí data bindingu na tento element zpřístupní následující textová pole pro zadání nového hesla. Pokud je heslo již vyžadováno, je políčko při načtení již zaškrtnuto. Hesla však vyplněna nejsou, aby uživatel neviděl počet znaků. Pole pro hesla jsou dvě kvůli minimalizaci výskytu překlepu a následnému zablokování přístupu do peněženky. Při uložení nastavení se kontroluje, zda jsou zadaná hesla stejná.

Nastavení hesel se ukládá do lokálního úložiště aplikace, vytvořeného právě pro tyto účely. K tomu se používá objekt `ApplicationData.Current.LocalSettings.Values`, který s pomocí operátoru indexace ukládá nebo vrací nastavení ve formě páru klíč-hodnota. Jako obal pro zjednodušení použití nastavení v ostatních částech aplikace byla vytvořena statická obalová třída `AppSettings` s potřebnými metodami. Pod zadáním hesla se nachází část pro import a export uživatelských dat do souboru ve formátu JSON. Export probíhá do složky Dokumenty v telefonu do souboru `exportData.json`. Přístup k této složce je nutné deklarovat v manifestu aplikace, což je XML dokument deklarující například název aplikace, autora, verzi, různé ikonky a načítací obrazovky aplikace a zmíněné přístupy. Pro editaci manifestu poskytuje Visual Studio grafické rozhraní. Pokud však chceme deklarovat přístup právě do složky Dokumenty, je toto nutné udělat ručně a přidat do manifestu, do tagu `Capabilities`, řádek:

```
<Capability Name="documentsLibrary" />
```

Abychom mohli používat soubory s určitou příponou, je také nutné v manifestu deklarovat přiřazení takové přípony k naší aplikaci. Systém potom bude umožňovat tyto soubory otevřít v naší aplikaci také například z průzkumníka souborů.

Export a import dat z/do souboru a serializaci uživatelských dat zajišťuje statická třída `Export`. Ta pro serializaci dat používá třídu `DataContractJsonSerializer`, kterou poskytuje .NET framework. Tato třída bere jako parametr konstruktoru datový typ serializovaných dat. Ke třídám, které mají být serializovány, je nutné přidat atribut `[DataContract]` a k serializovatelným položkám těchto tříd atribut `[DataMember]`. K souboru pak přistoupíme pomocí objektu `KnownFolders.DocumentsLibrary`, představujícímu složku Dokumenty, na který zavoláme metodu `OpenStreamForWriteAsync`. Ta nám s parametrem názvu souboru a parametrem výčtového typu `CreationCollisionOption.ReplaceExisting`, který určí akci v případě kolize souborů, otevře soubor pro zápis. Na instanci serializátoru pak zavoláme metodu `WriteObject` s parametry otevřeného datového proudu a objektu k serializaci. Ta nám zapíše data ve formátu JSON do otevřeného souboru. Na konec metodou `Flush` na tento

proud dat vyprázdníme buffer do souboru a metodou `Dispose` uvolníme zabrané prostředky.

Import dat probíhá obdobně. Importovat data lze také otevřením souboru z průzkumníka naší aplikací. Pro tento případ je v třídě `App` přetížena metoda `OnFileActivated`, která jako parametr dostává atributy otevíraného souboru. Pokud není ve třídě `Window` zaregistrován žádný `Frame` nebo pokud jeho obsah je `null`, vytvoří metoda rámec podobně jako metoda `OnLaunched`, která je volaná při normálním spuštění aplikace. Potom se získaný parametr přepoše jako navigační parametr úvodní stránky nebo stránky pro zadání hesla, a ta jej pošle úvodní stránce. Tam se objeví výzva ve formě elementu `Flyout` k potvrzení importu a nahrazení stávajících dat v aplikaci, s celkovým počtem příjmů, výdajů, štítků a účtů v souboru a v aplikaci.

Poslední funkcí stránky nastavení je vymazání obsahů všech tabulek uživatelské databáze. Z tabulek se nemaže pouze nultý řádek tabulek `Accounts` a `RecurrenceChains` kvůli omezení cizího klíče v záznamech příjmů a výdajů, které jakoby k žádnému účtu nepatří nebo nejsou fixní. Při mazání obsahů tabulek se také resetují sekvence pro funkci `AUTOINCREMENT`. To se děje vymazáním řádku tabulky `SQLITE_SEQUENCE` se sloupcem `name` obsahujícím název naší tabulky.

Spodní proužek stránky opět obsahuje tlačítka „uložit“ a „storno“.

4.7.9 Stránka pro zadání hesla

Tato stránka se objevuje při spuštění aplikace, pokud předtím v nastavení bylo zaškrtnuto a uloženo vyžadování hesla. Stránka obsahuje pouze nadpisy, textový vstup pro zadání hesla a pod ním tlačítko pro potvrzení, jímž se zkontroluje správnost zadaného hesla, a které případně vypíše text o zadání špatného hesla. Pokud bylo heslo zadáno správně, je uživatel navigován na úvodní stránku. V případě, že byla aplikace spuštěna otevřením souboru z průzkumníka, je přichozí parametr o souboru zapamatován a přeposlán po potvrzení správného hesla úvodní stránce.

4.7.10 Stránka o aplikaci

Stránka o aplikaci obsahuje nadpisy, podobně jako předchozí stránky, informaci o verzi, kterou obsahuje objekt `Package.Current.Id.Version` a jméno autora aplikace. Následují tři odstavce o použitých knihovnách s odkazy na licence a informace o exportu a importu uživatelských dat. Stránka neobsahuje žádná tlačítka.

4.7.11 Použití Blend for Visual Studio a vzorových dat

Dobrým nástrojem pro tvorbu GUI nejen Windows Phone aplikací je Microsoft Blend for Visual Studio. Je to užitečný nástroj pro sestavování grafického rozhraní, kdy uživatel hned vidí, jak bude výsledek vypadat. Práce s ním je poměrně intuitivní a lze s ním celé rozhraní v podstatě naklikat, i když často je dobré si kód upravit ručně.

Ačkoliv aplikace určená pro telefon je kompilována pro platformu ARM, programová část Visual Studio a Blend for Visual Studio, která zobrazuje výsledné GUI, Design view,

nepodporuje cílové platformy x64 a ARM. Při otvírání projektu v programu Blend je tedy nutné mít projekt uložený pro platformu x86. Po otevření projektu lze už toto nastavení bez problémů změnit.

U většiny aplikací je důležité mít při designování GUI k dispozici vzorová data. Blend samotný umožňuje možnost tvorby vzorových dat. Vzorová data lze však zadat i několika způsoby ručně, z nichž jeden byl využit i v tomto projektu.

Pro potřeby použití vzorových dat na stránce přidáme tagu `Page` atributy `d:DataContext` pro data zobrazovaná při vývoji aplikace a `mc:Ignorable` udávající, které předpony názvů jmenných prostorů mají být ignorovány procesorem XAML. Následující úryvky zdrojového kódu, ukazující implementaci vzorových dat, byly vybrány ze souborů `HubPage.xaml` a `RecordsModel.cs`.

```
<Page
  x:Class="Penezenka_App.HubPage"
  xmlns:sampleData="using:Penezenka_App.SampleData"
  d:DataContext="{Binding Source={d:DesignInstance
Type=sampleData:RecordsModel, IsDesignTimeCreatable=True}}"
  mc:Ignorable="d">
```

```
namespace Penezenka_App.SampleData
{
    class RecordsModel
    {
        public ObservableCollection<Record> Records
        {
            get
            {
                return new ObservableCollection<Record>
                {
                    new Record
                    {
                        ID = 1,
                        Date = new DateTime(2015, 12, 1),
                        Title = "položka 1"
                        // část kódu vynechána
                    },
                    new Record
                    {
                        ID = 2,
                        Date = new DateTime(2015, 12, 20),
                        Title = "položka 2"
                        // část kódu vynechána
                    }
                }
            }
        }
    }
}
```

Takto vytvořená data pak dále vážeme na elementy rozhraní stejně jako ostatní data. Je proto vhodné mít vzorová data pod stejnými názvy jako data skutečná. Aby se nově zadaná vzorová data zobrazila v Design view, je potřeba projekt ještě zkompilovat.

4.7.12 Tvorba grafů

Pro realizaci grafů v aplikaci byla využita knihovna WinRT XAML Toolkit. Jde o sadu ovládacích prvků a rozšíření pro Windows Runtime aplikace používající jazyk XAML. Knihovna je distribuována zdarma pod licencí MIT, která nám umožňuje tento software bez omezení používat, upravovat či prodávat.⁶

Pro její instalaci opět použijeme správce balíčků NuGet a vyhledáme „WinRT XAML Toolkit“ a „WinRT XAML Toolkit Data Visualization Controls“. Oba balíčky pak přidáme do referencí podobně, jako knihovny SQLite. Tím se nám zpřístupní nové elementy, které také můžeme pro usnadnění práce zahrnout do Toolboxu Visual Studia. Výsečové a lineární grafy v kódu XAML představují elementy `PieSeries`, resp. `LineSeries` v elementu `Chart`.

Nevýhodou při implementaci těchto grafů bylo jejich nezobrazení v programové části Design view, a to ani při připojení vzorových dat.

⁶ Licence je dostupná na <https://winrtxamltoolkit.codeplex.com/license>

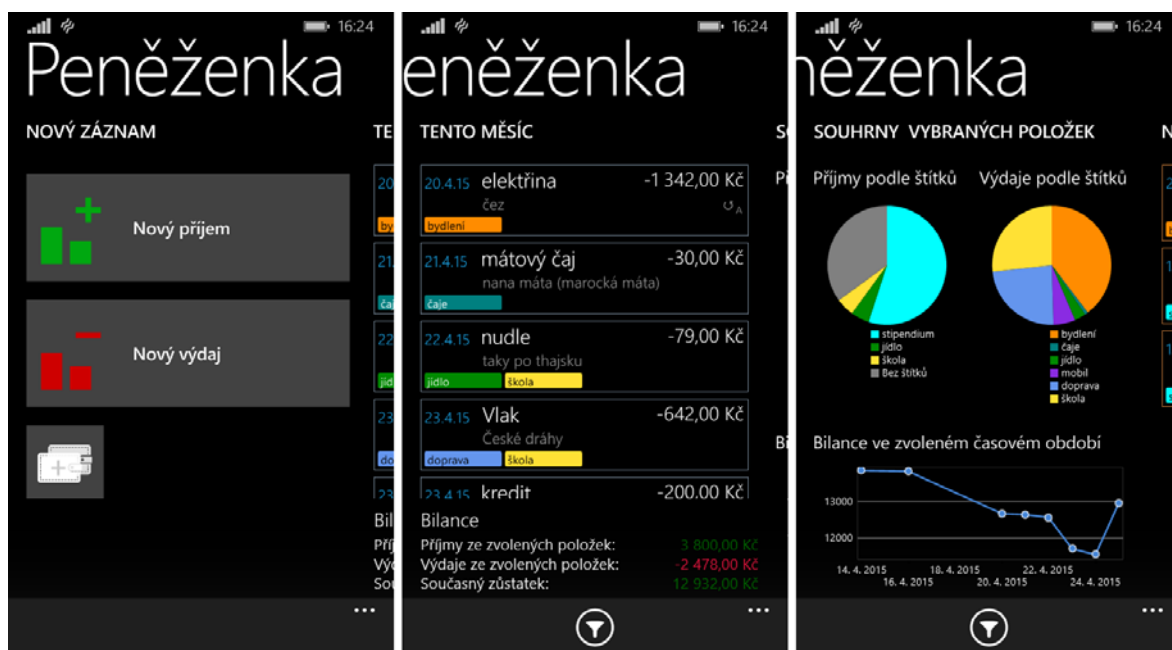
5 Uživatelská část

Uživatelská část slouží k seznámení s používáním aplikace Peněženka. Aplikace se skládá z celkem 9 různých stránek. Jako první stránka po instalaci a spuštění aplikace se nám zobrazí úvodní stránka.

5.1 Úvodní stránka

Úvodní stránka se skládá z pěti sekcí, které se nachází na Obrázku 7 a 8. Vpravo od každé sekce je vidět kousek sekce následující.

Spodní proužek je vysunovací. Vysuneme ho klepnutím nebo přetažením symbolu tří teček. To nám zpřístupní textová tlačítka „nastavení“ a „o aplikaci“. O stránkách, na které odkazují, si řekneme později. Jak můžete vidět na obrázcích, tento proužek v různých sekcích obsahuje také různá kulatá tlačítka, která se vztahují k aktuálně zobrazené sekci.



Obrázek 7: Úvodní stránka – první tři sekce [zdroj: autor]

V první sekci vidíme dvě velká tlačítka pro přidání příjmu a výdaje. Ty nás odkáží na stránku, kde je můžeme zadávat. Pod nimi se nachází menší tlačítko pro správu účtů.

V druhé sekci se vypisují příjmy a výdaje. Ve výchozím stavu se zde nachází příjmy a výdaje pro aktuální měsíc. Každý záznam je ohraničen bledě modrým rámečkem. Při podržení prstu na položce se ukáže menu s položkami „upravit“ a „odstranit“. Pokud klikneme na „odstranit“, objeví se ještě varování vyžadující potvrzení této akce. Položky se dají při poklepání rozšířit o další dva řádky. V prvním řádku můžete vidět datum, název položky a peněžní částku. Druhý řádek obsahuje pod názvem poznámku, a pokud je zadáný záznam opakovaný (fixní), vyskytuje se pod částkou symbol ∩. Pokud byl opakovaný záznam vytvořen automaticky a nebyl upraven, zobrazovaný symbol vypadá

takto: \mathcal{U}_A . Na dalších řádcích vidíme, na kterém účtu se záznam vyskytuje a jak je záznam opakován. Na konci položky jsou vždy vykresleny přiřazené štítky. Pod seznamem vidíme součet příjmů a součet výdajů z vypsanych položek a celkový současný zůstatek pro všechny účty a položky. Část se součty se dá kliknutím zmenšit na jeden řádek a zase zvětšit.

Ve třetí sekci vidíme výšečové grafy vybraných příjmů a výdajů seskupených podle štítků a pod nimi lineární graf vývoje zůstatku v průběhu vybraného období. I zde vidíme tlačítko pro filtr položek.



Obrázek 8: Úvodní stránka – poslední dvě sekce [zdroj: autor]

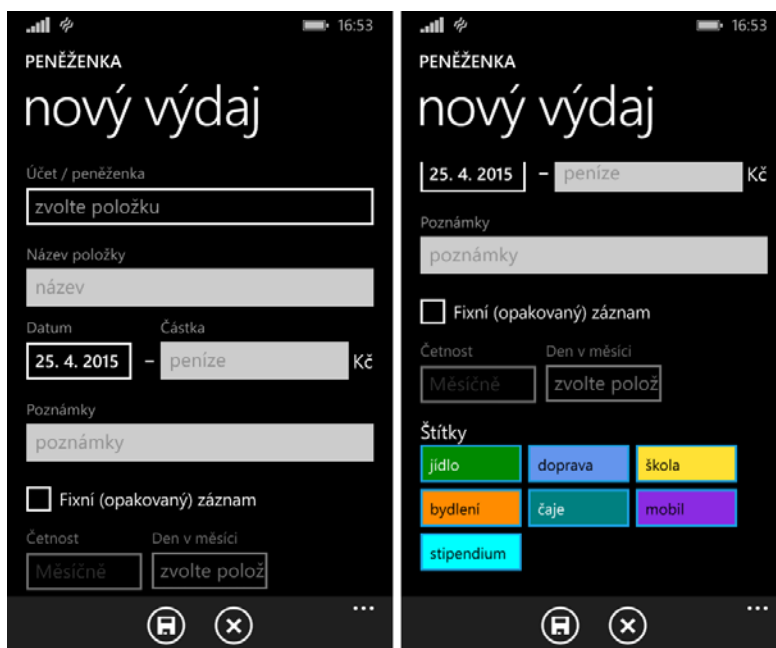
Následuje sekce s nadcházejícími příjmy a výdaji. Jejich zobrazení je v podstatě stejné jako u výpisu položek v sekci 2. Pokud bude aplikace spuštěna v zobrazené datum nebo později, bude tato položka automaticky přidána mezi ostatní příjmy a výdaje.

Poslední část obsahuje seznam všech štítků s jejich názvy, barvami a poznámkami. Klepnutím na štítek se dostaneme k jeho úpravě. Podržení štítku dostaneme nabídku štítek upravit nebo smazat. Před smazáním štítku se nám ještě zobrazí výzva k potvrzení této akce.

5.2 Stránka přidání a úpravy příjmu a výdaje

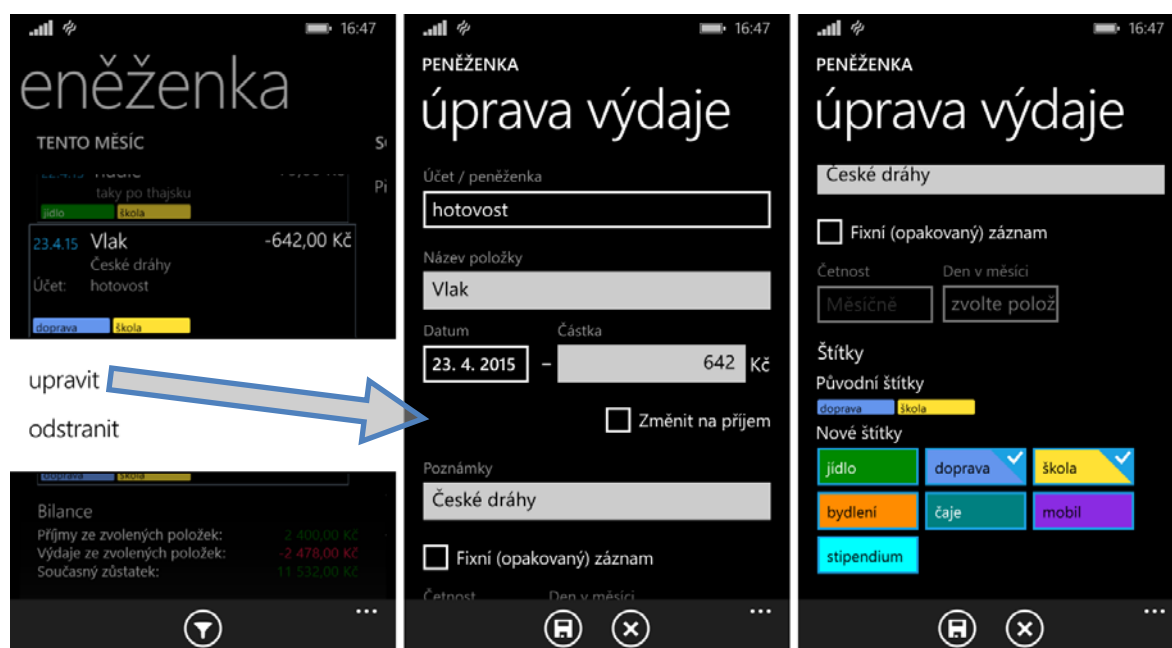
Tato stránka nám umožní vytvořit nový příjem nebo výdaj. Nadpisy nad každým políčkem nám už napoví, k čemu dané políčko slouží. Pokud již máme vytvořené nějaké štítky, jsou zde vypsány a klepnutím na ně je můžeme vybrat.

Na závěr můžeme spodními tlačítky zvolit, zda záznam uložit či se pouze vrátit na předchozí stránku. Snímky této stránky obsahuje Obrázek 9.



Obrázek 9: Stránka nového výdaje [zdroj: autor]

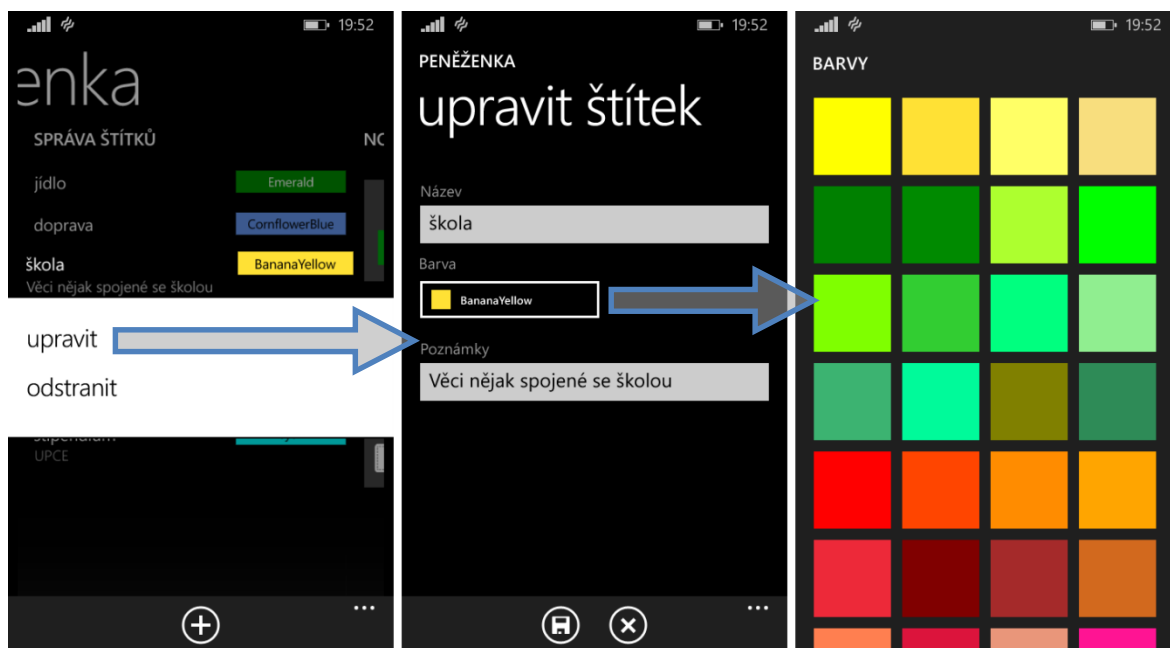
Na Obrázku 10 vidíme stránku pro úpravu výdaje. Na ni se dostaneme z 2. sekce úvodní stránky, podržením výdaje a zvolením možnosti „upravit“. Na rozdíl od nového výdaje se zde nachází pod částkou zaškrťovací políčko „Změnit na příjem“. Zaškrtnutím zmizí znaménko mínus a záznam se uloží jako příjem. Dalším novým prvkem je seznam aktuálně přiřazených štítků. Pokud tento seznam chceme zachovat stejný, musíme dané štítky zaškrtnout znovu.



Obrázek 10: Stránka úpravy výdaje [zdroj: autor]

5.3 Stránka přidání a úpravy štítku

Tato poměrně jednoduchá stránka je přístupná z 5. sekce úvodní stránky. Můžeme zde zadat název a poznámky ke štítku a barvu, kterou si vybereme z dostupné palety. Ta se objeví po stisknutí příslušného tlačítka, jak můžete vidět na Obrázku 11. K úpravě štítku se lze dostat buď klepnutím na štítek v seznamu, nebo podržením štítku a vybráním možnosti „upravit“.

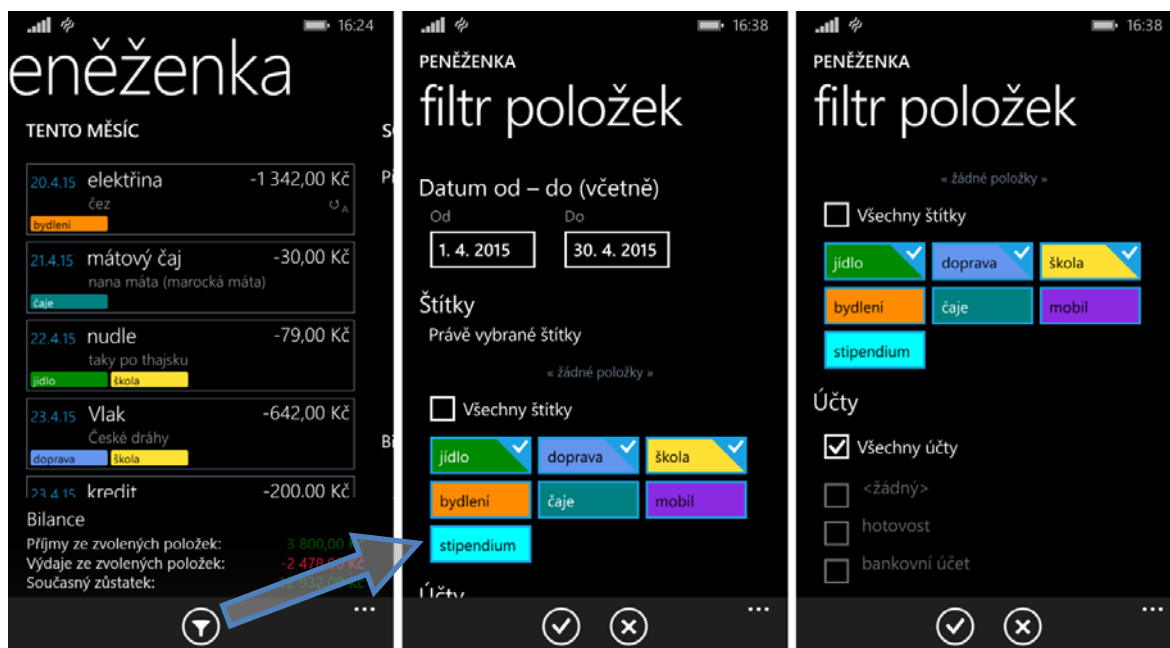


Obrázek 11: Stránka úpravy štítku [zdroj: autor]

5.4 Stránka Filtr položek

Na tuto stránku se dostaneme z 2. a 3. sekce úvodní stránky. Nastavuje se zde, které položky na úvodní stránce se mají zobrazit. Filtrovat položky můžeme podle data od – do, podle štítků, které obsahují, nebo podle účtů, na kterých se nacházejí, viz Obrázek 12. Stejně jako u stránky úpravy příjmu/výdaje je nutné všechny zvolené štítky zaškrtnout znovu.

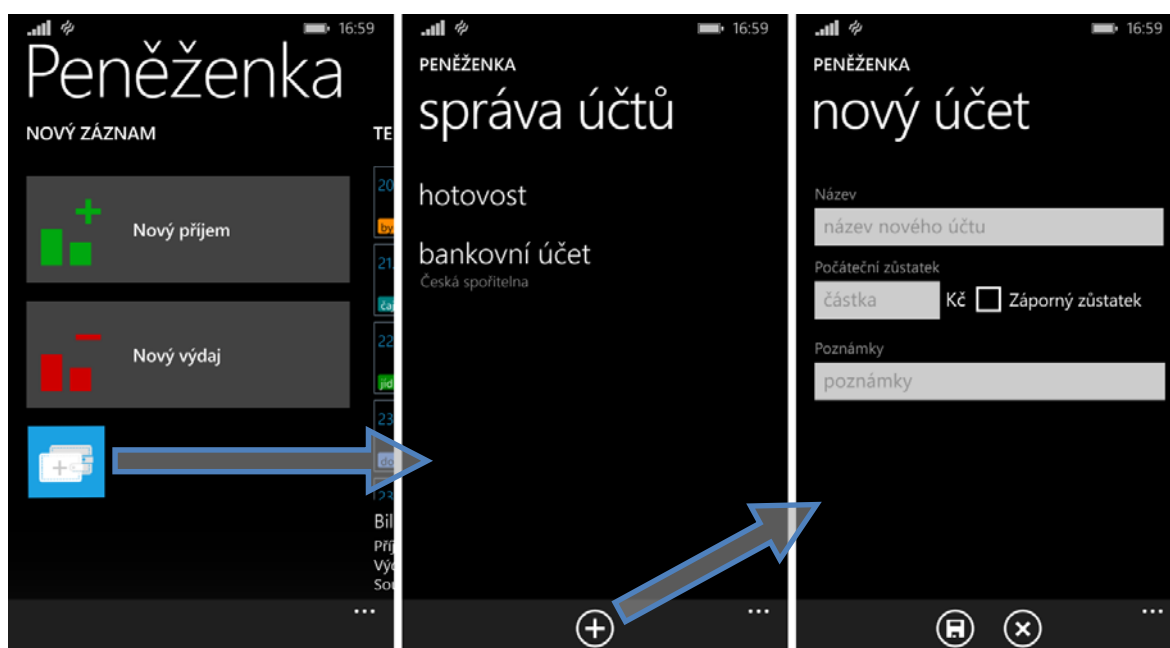
Filtr se aplikuje tlačítkem „přijmout“, které přesměruje uživatele na úvodní stránku s použitým filtrem. Klepnutím na tlačítko „storno“ se zadání filtru zruší a uživatel se pouze vrátí zpět na předchozí stránku.



Obrázek 12: Stránka filtru položek [zdroj: autor]

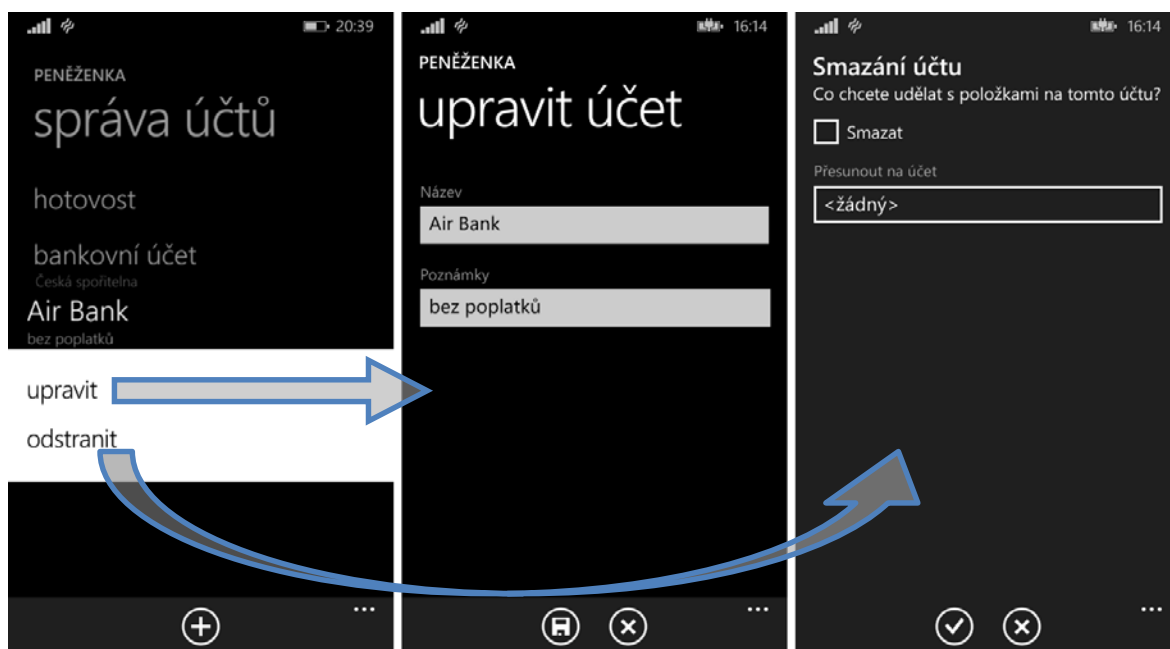
5.5 Stránka správy účtů a přidání/úpravy účtu

Stránka se seznamem dostupných účtů je přístupná z 1. sekce úvodní stránky pomocí tlačítka s obrázkem peněženek, viz Obrázek 13. Tlačítkem „přidat“ ve spodním proužku stránky se dostaneme k zadání nového účtu. Kromě názvu a poznámek zde můžeme zadat i počáteční zůstatek, a to jak kladný, tak záporný, což volíme pomocí příslušného zaškrtnutí políčka. Ten se ihned po vytvoření účtu přidá jako další položka k příjmům a výdajům. Lze jej pak stejně jako ostatní příjmy a výdaje kdykoliv upravit.



Obrázek 13: Stránka správy účtů a nového účtu [zdroj: autor]

Podržetím položky účtu dostaneme vyskakovací menu s tlačítky „upravit“ a „odstranit“. Klepnutím na tlačítko „odstranit“ nebo na účet ve výpise se dostaneme k jeho úpravě. Změny můžeme uložit nebo zahodit tlačítky „uložit“, resp. „storno“. Zvolením možnosti „odstranit“ se nám zobrazí vyskakovací okno s nabídkou akcí s položkami na tomto účtu. Tyto položky lze buď smazat, nebo přesunout na jiný účet, jak vidno na Obrázku 14.



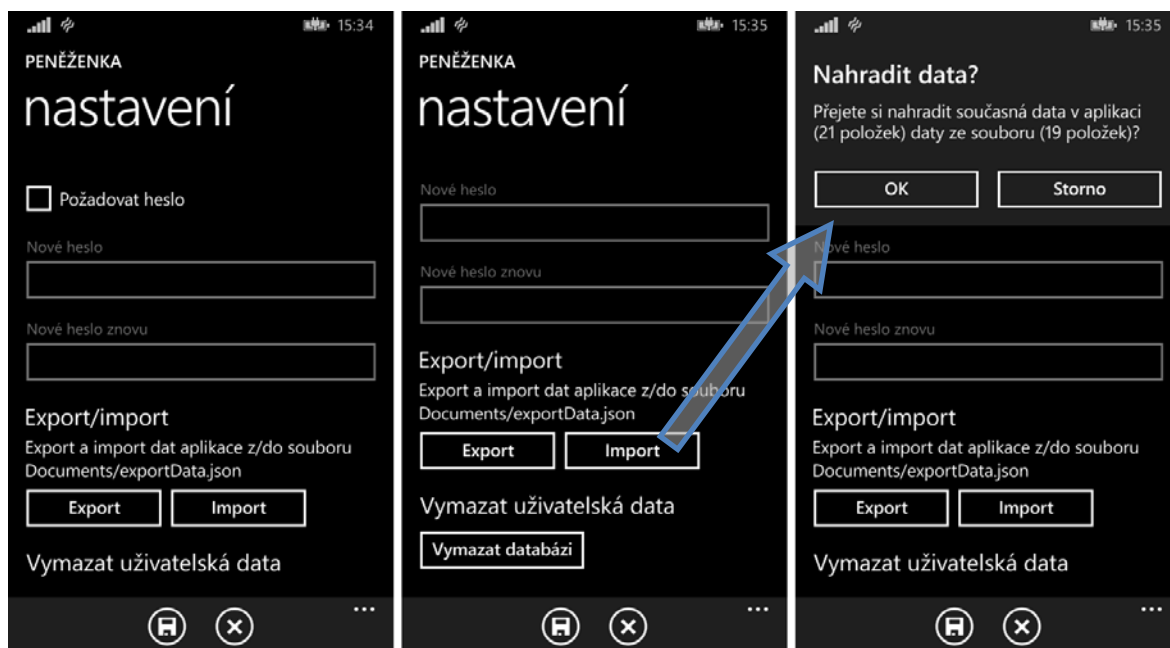
Obrázek 14: Stránka správy účtů a úpravy účtu [zdroj: autor]

5.6 Stránka nastavení

Tato část aplikace umožňuje uživateli nastavit peněženku ochranu heslem. Tato ochrana se zapíná a vypíná zaškrtnutím políčkem „Požadovat heslo“, jak je ukazuje první část Obrázku 15. Po zaškrtnutí políčka poklepáním lze zadat příslušné heslo, které je nutné 2× zopakovat. Po uložení nastavení a novém spuštění aplikace bude vyžadováno heslo, viz následující podkapitolu Stránka o aplikaci a stránka pro zadání hesla.

Dále se zde nachází možnost pro export a import dat. Stisknutím tlačítka „Export“ se všechny příjmy, výdaje, štítky a účty zapíše do souboru `Dokumenty/ExportData.json` na telefonu. Pokud soubor již existuje, bude přepsán. Import probíhá z téhož souboru. Před nahrazením uživatelských dat v aplikaci importovanými daty bude uživatel požádán o potvrzení. Import dat lze také provést otevřením souboru touto aplikací. V tom případě bude uživatel navigován na úvodní stránku (po případném zadání hesla) a vyzván k potvrzení importu dat.

Poslední funkcí této stránky je vymazání všech uživatelských dat. Před vymazáním bude opět uživatel vyzván k potvrzení.



Obrázek 15: Stránka nastavení [zdroj: autor]

5.7 Stránka o aplikaci a pro zadání hesla

Stránka o aplikaci zobrazuje informace o verzi, autorovi aplikace a o použitých externích knihovnách s odkazy na licence. Dále se zde nachází zmínka o exportu a importu dat.

Stránka pro zadání hesla, která byla zmíněna v předchozí kapitole, se zobrazí po spuštění aplikace, pokud bylo předtím nastaveno požadování hesla. Obě stránky se nachází na Obrázku 16.



Obrázek 16: Stránka o aplikaci a stránka požadování hesla [zdroj: autor]

6 Závěr

Všechny cíle bakalářské práce byly splněny v plném rozsahu. V teoretické části byly prozkoumány možnosti vývoje aplikací pro tři nejrozšířenější mobilní operační systémy iOS, Android a Windows Phone. Z důvodu značného množství dostupných aplikací pro operační systémy iOS a Android a poměrně přívětivých nástrojů vývoje aplikací pro Windows Phone byla pro vývoj zadané aplikace vybrána právě tato platforma.

Tyto možnosti následuje výběr a stručný popis vybraných aplikací pro Windows Phone, zabývajících se podobným tématem. U každé aplikace jsou zmíněny jejich přednosti a nevýhody.

Praktická část se zabývá implementací aplikace Peněženka. Na jejím začátku byly popsány základní informace o aplikaci a s tím spojenými požadavky na ni. Dále zde byl popsán návrhový vzor MVVM. S tímto návrhovým vzorem souvisí vázání zdrojových dat s jejich zobrazením, tzv. data binding, jehož možnosti a zápis jsou zde popsány. Následně byl popsán životní cyklus aplikací na operačním systému Windows Phone a jednotlivé stavy, ve kterých se aplikace může nacházet.

Pro trvalé uchovávání dat aplikace, které uživatel vytvoří, byla vybrána databáze SQLite. V rámci této databáze je zde popsán příslušný databázový model s tabulkami a její implementace v aplikaci. Dále je zde zmíněno využití knihovny, která umožňuje použití databáze SQLite v programovacím jazyce C#.

V této části je rovněž rozebrána tvorba uživatelského rozhraní, která se skládá z použitých funkcionalit WPF, struktury GUI a navigace v něm. V rámci struktury GUI jsou popsány jednotlivé stránky uživatelského rozhraní aplikace s popisem některých implementačně zajímavých funkcionalit, jako je například export serializovaných dat do souboru ve formátu JSON a jejich import či ochrana aplikace přístupovým heslem. Také je zde popsána implementace grafů s použitím knihoven WinRT XAML Toolkit a WinRT XAML Toolkit Data Visualization Controls. Praktickou část uzavírá kapitola věnující se popisu grafického rozhraní z pohledu uživatele.

Aplikace je plně funkční a byla otestována na telefonech Nokia Lumia 520, Nokia Lumia 625 a Nokia Lumia 1320. Rozdílnost telefonů vzhledem k vyvíjené aplikaci spočívala převážně v rozdílné velikosti a rozlišení displeje. Při testování byly v aplikaci provedeny jen menší změny, neboť rozdílné rozlišení dokáže simulovat již vývojové prostředí.

Projekt je možné dále rozvíjet například ve směru pokročilejších možností filtrování záznamů, implementace převodu měn a jazykových mutací aplikace či exportu dat do Excelu nebo na cloudové úložiště. Užitečná by byla také možnost přidání fotografie k příjmu či výdaji nebo tvorba rozpočtů.

Použitá literatura

1. OXFORD UNIVERSITY PRESS. smartphone. *Oxford Dictionaries* [online]. 2015 [cit. 2015-02-11]. Dostupné z: <http://www.oxforddictionaries.com/definition/english/smartphone>
2. ROCHA, E. RIM to rename itself BlackBerry, hoping for fresh start. *Reuters* [online]. 30. 1. 2013 [cit. 2015-02-03]. Dostupné z: <http://in.reuters.com/article/2013/01/30/rim-blackberry-name-idINDEE90T0D220130130>
3. Smartphone OS Market Share 2014, 2013, 2012, and 2011. *IDC* [online]. 2015 [cit. 2015-02-03]. Dostupné z: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>
4. APPLE. iOS Technology Overview. [online]. 17. 9. 2014 [cit. 2014-12-31]. Dostupné z: <https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iOSTechOverview.pdf>
5. APPLE. Programming with Objective-C. [online]. 17. 9. 2014 [cit. 2014-12-31]. Dostupné z: <https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/ProgrammingWithObjectiveC.pdf>
6. WALTER, R. a K. CLEGG. Swift - Learn Apple's New Programming Language Step By Step. *udemy* [online]. 2015 [cit. 2015-02-12]. Dostupné z: <https://www.udemy.com/swift-learn-apples-new-programming-language-by-examples/>
7. REICHL, P. Vývoj aplikací pro iPhone. *Zdroják* [online]. 20. 1. 2010 [cit. 2015-01-01]. Dostupné z: <http://www.zdrojak.cz/clanky/vyvoj-aplikaci-pro-iphone/>
8. MOLEN, B. We just played with Android's L Developer Preview. *Engadget* [online]. 22. 6. 2014 [cit. 2015-02-11]. Dostupné z: <http://www.engadget.com/2014/06/26/android-developer-preview-hands-on/>
9. uses-sdk. *Android Developers* [online]. [cit. 2015-02-03]. Dostupné z: <http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels>
10. Android versions comparison. *Comparison tables - SocialCompare* [online]. 22. 3. 2015 [cit. 2015-05-04]. Dostupné z: <http://socialcompare.com/en/comparison/android-versions-comparison>
11. TRAYNOR, B. Android Architecture. *Embedded Linux Wiki* [online]. 13. 6. 2011 [cit. 2015-02-04]. Dostupné z: http://elinux.org/Android_Architecture

12. BRÄHLER, S. Analysis of the Android. [online]. 6. 10. 2010 [cit. 2015-01-03]. Dostupné z: http://os.itec.kit.edu/downloads/sa_2010_braehler-stefan_android-architecture.pdf
13. Managing Projects from Eclipse with ADT. *Android Developers* [online]. [cit. 2015-02-04]. Dostupné z: <http://developer.android.com/tools/projects/projects-eclipse.html>
14. CARLSTROM, B. A. GHULOUM a I. ROGERS. The ART runtime. *Google I/O 2014* [online]. 2014 [cit. 2015-02-12]. Dostupné z: <https://www.google.com/events/io/io14videos/b750c8da-aebe-e311-b297-00155d5066d7>
15. BURNETTE, E. Java vs. Android APIs. In: *ZDNet* [online]. 12. 1. 2008 [cit. 2015-01-03]. Dostupné z: <http://www.zdnet.com/article/java-vs-android-apis/>
16. TILLEY, C. The History of Windows CE. In: *HPC:Factor* [online]. 26. 4. 2014 [cit. 2015-02-12]. Dostupné z: <http://www.hpcfator.com/support/windowsce/>
17. A brief history of Windows Mobile. *notebooks.com* [online]. 2015 [cit. 2015-02-12]. Dostupné z: <http://notebooks.com/2010/04/12/a-brief-history-of-windows-mobile/>
18. MINIMAN, B. Exclusive: Windows Mobile 6.5 vs. Windows Phone 7 (Video). *Pocketnow* [online]. 23. 9. 2010 [cit. 2015-01-03]. Dostupné z: <http://pocketnow.com/windows-phone/exclusive-windows-mobile-65-vs-windows-phone-7-video>
19. MACKIE, K. Microsoft Changes Windows 8 'Metro' Name to 'Windows Store App'. *Redmond magazine* [online]. 11. 1. 2012 [cit. 2015-02-12]. Dostupné z: <http://redmondmag.com/articles/2012/11/01/microsoft-changes-metro-name.aspx>
20. MICROSOFT. Windows Phone architecture overview. *Windows Phone | Dev Center* [online]. 15. 12. 2014 [cit. 2015-02-12]. Dostupné z: http://dev.windowsphone.com/en-us/OEM/docs/Getting_Started/Windows_Phone_architecture_overview
21. ŘEZANINA, E. Software pro podporu restauračních zařízení. Pardubice: Univerzita Pardubice, 2010, s. 21. Dostupné také z: <http://hdl.handle.net/10195/36814>
22. MICROSOFT. "Differences Between Visual Basic.NET and Visual C#.NET" white paper is available. *Microsoft support* [online]. 21. 4. 2012 [cit. 2015-02-12]. Dostupné z: <http://support.microsoft.com/kb/308470>
23. APPLE. Interface Builder Built-In. *Apple Developer* [online]. 2015 [cit. 2015-01-02]. Dostupné z: <https://developer.apple.com/xcode/interface-builder/>
24. APPLE. iOS Dev Center. *Apple Developer Support* [online]. 2015 [cit. 2015-02-11]. Dostupné z: <https://developer.apple.com/support/ios/ios-dev-center.php>

25. User Interface. *Xamarin* [online]. 2015 [cit. 2015-01-02]. Dostupné z: http://developer.xamarin.com/guides/ios/user_interface/
26. Installing Xamarin.iOS on Windows. *Xamarin* [online]. 2015 [cit. 2015-01-02]. Dostupné z: http://developer.xamarin.com/guides/ios/getting_started/installation/windows/
27. Download Android Studio and SDK Tools. *Android Developers* [online]. [cit. 2015-02-12]. Dostupné z: <http://developer.android.com/sdk/index.html#Requirements>
28. SEMECKÝ, V. Android Studio – nové vývojové prostředí. *Zdroják* [online]. 4. 11. 2013 [cit. 2015-02-04]. Dostupné z: <http://www.zdrojak.cz/clanky/android-studio-nove-vyvojove-prostredi/>
29. MICROSOFT. Development tools for Windows Phone apps. *Windows Dev Center* [online]. 2015 [cit. 2015-02-04]. Dostupné z: <https://msdn.microsoft.com/en-us/library/windows/apps/dn629632.aspx>
30. MICROSOFT. What is XAML? *Microsoft Developer Network* [online]. 2015 [cit. 2015-02-16]. Dostupné z: <https://msdn.microsoft.com/en-us/library/cc295302.aspx>
31. ELLIOT, I. Microsoft Kills Expression Suite. *I programmer* [online]. 21. 12. 2012 [cit. 2015-02-13]. Dostupné z: <http://www.i-programmer.info/news/90-tools/5236-microsoft-kills-expression-suite.html>
32. MICROSOFT. Create your first Windows Store app with Blend, part 2: the details page (HTML & JavaScript). *Microsoft Developer Network* [online]. 2015 [cit. 2015-02-04]. Dostupné z: <https://msdn.microsoft.com/en-us/library/dn435907.aspx>
33. MIKUSIAK, L. iOS vs. Android Development Comparison – Part 1. *PASSION4TEQ* [online]. 2013 [cit. 2015-02-19]. Dostupné z: <http://www.passion4teq.com/articles/ios-android-development-comparison-1/>
34. MICROSOFT. Company app distribution for Windows Phone. *Windows Dev Center* [online]. 19. 8. 2014 [cit. 2015-02-19]. Dostupné z: <https://msdn.microsoft.com/library/windows/apps/jj206943%28v=vs.105%29.aspx>
35. KOČÍ, M. Windows Phone Store už obsahuje přes 300 tisíc aplikací, tvrdí Microsoft. *Svět aplikací* [online]. 9. 8. 2014 [cit. 2015-02-19]. Dostupné z: <http://svetaplikaci.tyden.cz/windows-phone-store-uz-obsahuje-pres-300-tisic-aplikaci-tvrdi-microsoft/>
36. POLESNÝ, D. Microsoft přiznává, že kvalita aplikací ve Windows Store se musí zlepšit. *Živě.cz* [online]. 22. 8. 2014 [cit. 2015-02-19]. Dostupné z: <http://www.zive.cz/bleskovky/microsoft-priznava-ze-kvalita-aplikaci-ve-windows-store-se-musi-zlepsit/>

sc-4-a-175064/default.aspx

37. StatCounter Global Stats. *StatCounter* [online]. 2015 [cit. 2015-02-17]. Dostupné z: http://gs.statcounter.com/#mobile_os-CZ-monthly-201201-201502
38. MICROSOFT. Data Binding Overview. *Microsoft Developer Network overview* [online]. 2015 [cit. 2015-02-27]. Dostupné z: <https://msdn.microsoft.com/en-us/library/ms752347%28v=vs.110%29.aspx>
39. MICROSOFT. Data binding overview (XAML). *Windows Dev Center overview(xaml)* [online]. 2015 [cit. 2015-02-27]. Dostupné z: <https://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh758320.aspx>
40. NIXON, J. XAML Binding Basics 101. *MSDN Blogs* [online]. 12. 10. 2012 [cit. 2015-02-27]. Dostupné z: <http://blogs.msdn.com/b/jerry-nixon/archive/2012/10/12/xaml-binding-basics-101.aspx>
41. MICROSOFT. Application lifecycle (Windows Runtime apps). *Windows Dev Center* [online]. 2015, verze 2015.02.26.3 [cit. 2015-03-21]. Dostupné z: <https://msdn.microsoft.com/en-us/library/windows/apps/hh464925.aspx>
42. LACKO, Ľ. *Vývoj aplikací pro Windows 8.1 a Windows Phone..* Brno: COMPUTER PRESS, 2014. ISBN 978-80-251-3822-9.
43. HIPPEL, D. R. D. KENNEDY a J. MISTACHKIN. About SQLite. *SQLite* [online]. 2015 [cit. 2015-04-18]. Dostupné z: <http://sqlite.org/about.html>
44. MICROSOFT. WPF Graphics Rendering Overview. *Microsoft Developer Network* [online]. 2015 [cit. 2015-04-23]. Dostupné z: <https://msdn.microsoft.com/en-us/library/ms748373.aspx>

Příloha A: Obsah přiloženého CD

Na přiloženém CD se nachází:

- složka `Aplikace`, která obsahuje zdrojové kódy aplikace v projektu Visual Studio 2013 a
- složka `Dokumentace`, která obsahuje elektronickou verzi tohoto dokumentu ve formátu PDF.