


Přenositelnost Windows Phone 8 aplikací na vývojovou platformu Xamarin

Bc. Tomáš Prokop

Diplomová práce
2017

 Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Tomáš Prokop**
Osobní číslo: **A15373**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **kombinovaná**

Téma práce: **Přenositelnost Windows Phone 8 aplikací na vývojovou platformu Xamarin**

Téma anglicky: **The Portability of Windows Phone 8 Applications to the Xamarin Development Platform**

Zásady pro vypracování:

1. **Nastudujte rozdíly prostředí mobilních operačních systémů iOS, Android a Windows včetně celkového způsobu zabezpečení aplikace a komunikace.**
2. **Prozkoumejte možnosti přenositelnosti aplikací programovaných pro platformu Windows Phone 8 do vývojového frameworku Xamarin pro publikaci na platformy iOS, Android a Windows.**
3. **Stručně popište prototypovou aplikaci Bitcoin wallet včetně popisu prostředí Bitcoin.**
4. **Provedte a popište implementaci prototypové aplikace Bitcoin wallet do vývojového frameworku Xamarin. Zaměřte se také na zabezpečení případně další specifické požadavky platformy Xamarin.**
5. **Aplikaci otestujte na reálném zařízení a zhodnoťte přenositelnost Windows Phone 8 aplikací do vývojového frameworku Xamarin.**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. Petzold, C.: **Creating Mobile Apps with Xamarin.Forms**. Microsoft Press., 2016, ISBN 978-1-5093-0297-0, 1187 s.
2. Tavlikos, D.: **iOS Development with Xamarin Cookbook**. Paclt Publishing, May 2014, ISBN 978-1-84969-892-4, 386 s.
3. FRANCO, Pedro. **Understanding Bitcoin: Cryptography, Engineering and Economics**. John Wiley, 2014, 288 s. ISBN 9781119019169.
4. HERMES, Dan. **Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals**. Apress, 2015, 432 s. ISBN 9781484202142.
5. **Owasp.org: OWASP [online]**. Listopad 2016. URL https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Ris
6. **Blockchain.info: Bitcoin Developer API's [online]**. 2016. URL <https://blockchain.info/api>

Vedoucí diplomové práce:

Ing. Radek Vala, Ph.D.

Ústav informatiky a umělé inteligence


Datum zadání diplomové práce:

3. února 2017


Termín odevzdání diplomové práce:

16. května 2017

Ve Zlíně dne 3. února 2017



doc. Mgr. Milan Adámek, Ph.D.
děkan



prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Diplomová práce se zabývá možnou přenositelností starších zdrojových kódů aplikace na multiplatformní framework Xamarin. Popisuje základní součásti Xamarinu pro vytvoření aplikačních stránek včetně navigace s běžnými vizuálními komponenty. Součástí práce je průzkum bezpečnosti v multiplatformním prostředí. Z části se práce zabývá popisem systému virtuální měny známé jako Bitcoin. Systém Bitcoin je zabezpečen pomocí kryptografických metod, které budou nastíněny. Práce objasňuje principy integrity transakcí, výměnu transakcí a emitaci nových peněz. Celé aplikační rozhraní, které slouží pro komunikaci s Bitcoin serverem a implementovanou aplikací, je v práci rozebráno. Samotný popis návrhu a implementace je v nemalé části popsán v rámci tohoto dokumentu a objasňuje možné problémy, na které může běžný vývojář narazit v Xamarinu.

Klíčová slova: Xamarin, MVVM, multiplatformní technologie, bezpečnost, Bitcoin, kryptografie

ABSTRACT

This master's thesis studies possible options for portability of an existing source code of an application into a multiplatform framework called Xamarin. It describes basic modules of Xamarin for creation of application pages including navigation using common visual components. In the scope of this work is also a survey of security in multiplatform environment and deep dive into Bitcoin as a cryptocurrency and digital payment system, which is the primary domain of the application. The security of Bitcoin is achieved by utilization of cryptographic methods. In the context of Bitcoin, the text clarifies principles of an integrity of transactions, an exchange of transactions and an imitation of the cryptocurrency. The whole application interface, which serves for a communication between a Bitcoin server and the implemented client application, is closely described in the text. The significant part of the text focuses on a design and an implementation of the application, including a summary of encountered issues which might be challenging for a developer using Xamarin.

Keywords: Xamarin, MVVM, multiplatform technology, security, Bitcoin, cryptography

V rámci diplomové práce bych chtěl poděkovat společnosti JetBrains a společnosti Telerik za poskytnutí školních licencí pro vývoj softwaru za pomoci jejich implementačních knihoven. Rád bych věnoval poděkování i pro kolegu Radovana Matouška, který mi pomáhal jako nestranná osoba v testování aplikace. V neposlední řadě děkuji Ing. Radku Valovi Ph. D. za skvělé vedení diplomové práce.

In the terms of third party support for this master's thesis, I would like to give thanks to the companies JetBrains and Telerik for providing student licences for their development products. I also appreciate help of my colleague, Radovan Matoušek, who participated on the testing of the application. At last I would like to express my gratitude to the supervisor of this work, Ing. Radek Vala, Ph.D., for having such a great patience with me, his guidance and valuable advices.

OBSAH

ÚVOD	8
I TEORETICKÁ ČÁST	9
1 BEZPEČNOST MOBILNÍCH ZAŘÍZENÍ.....	11
1.1 CHARAKTERISTIKY PLATFORMEM	12
1.2 BEZPEČNOSTNÍ PRVKY	15
1.2.1 Platforma iOS.....	17
1.2.2 Platforma Android.....	18
1.2.3 Platforma Windows	18
1.3 ODBLOKOVÁNÍ PLATFORMY	19
1.4 INSTALACE APLIKACÍ A LICENCOVÁNÍ.....	19
1.5 ŠIFROVACÍ ROZHRANÍ.....	20
1.6 SÍŤOVÉ KOMUNIKACE	21
1.7 NATIVNÍ SPUŠTĚNÍ KÓDU	22
1.8 MOBILNÍ PROHLÍZEČ.....	23
1.9 DESET NEJČASTĚJŠÍCH CHYB PŘI VÝVOJI.....	23
1.9.1 M1 - Nesprávné použití Platformem	23
1.9.2 M2 - Nebezpečně uložená data	24
1.9.3 M3 - Nezabezpečená komunikace	25
1.9.4 M4 - Nebezpečná autentifikace (ověřování)	25
1.9.5 M5 - Nedostatečná kryptografie.....	25
1.9.6 M6 - Nesprávná autorizace (oprávnění)	26
1.9.7 M7 - Kvalita kódování klienta	26
1.9.8 M8 - Manipulace s kódem.....	26
1.9.9 M9 - Reverzní inženýrství.....	26
1.9.10 M10 - Cizí funkčnost.....	27
2 ANALÝZA VÝVOJOVÉHO PROSTŘEDÍ XAMARIN	28
2.1 CO JE XAMARIN.FORMS	28
2.2 NAVIGACE (NAVIGATION).....	28
2.2.1 ContentPage	29
2.2.2 MasterDetailPage	29
2.2.3 NavigationPage	31
2.2.4 TabbedPage.....	31
2.2.5 CarouselPage	32
2.3 ROZVRŽENÍ (LAYOUTS).....	33

2.4	BĚŽNÉ PRVKY	33
2.4.1	Label	34
2.4.2	Tabulka	34
2.4.3	Styly	34
2.4.4	Fonty	35
2.4.5	Barvy	35
2.4.6	Tlačítka	36
2.4.7	Obrázky	36
2.5	DATABÁZE	36
2.6	SOUBORY	38
2.7	MAPY	38
2.8	TELERIK KNIHOVNA PRO GRAFY	39
2.9	XLABS KNIHOVNA	39
2.10	MODEL MVVM	40
2.10.1	Název návrhového vzoru a jeho klasifikace	40
2.10.2	Struktura	41
2.10.3	Účastníci	41
2.10.4	Důsledky	42
3	HASHOVACÍ FUNKCE	44
3.1	HASHOVACÍ FUNKCE V BITCOINECH	46
3.1.1	Funkce SHA (Secure Hash Algorithm)	47
3.1.2	Funkce RIPEMD (RACE Integrity Primitives Evaluation Message Digest)	47
4	ASYMETRICKÁ KRYPTOGRAFIE	50
4.1	ALGORITMUS RSA	50
4.2	DIFFIE-HELMAN (DH)	51
4.3	ELIPTICKÉ KŘIVKY	51
5	PROBLEMATIKA BITCOINŮ	53
5.1	PRINCIP A SMYSL BITCOINŮ	53
5.2	POPIS STRUKTURY	53
5.2.1	Block	53
5.2.2	Transakce	54
5.2.3	Horník	55
6	API PRO MOBILNÍ KOMUNIKACI	57
6.1	POPIS APLIKAČNÍHO ROZHRAŇÍ PRO BLOCKCHAIN.INFO	57

II	PROJEKTOVÁ ČÁST	59
7	POPIS PROTOTYPOVÉ APLIKACE	61
7.1	ROZDĚLENÍ PROJEKTU.....	61
7.2	MODEL MVVM.....	61
7.3	PROJEKTY.....	62
7.3.1	Projekt WebClient.....	62
7.3.2	Projekt RpcClient.....	62
7.3.3	Projekt DBase.....	62
7.3.4	Bitcoin MyWallet.....	62
8	NÁVRH PŘENOSU APLIKACE	64
8.1	PŘÍPRAVA MODELU MVVM.....	64
8.2	NÁVRH NOVÉ VERZE APLIKACE.....	64
9	IMPLEMENTACE V XAMARIN	67
9.1	PŘENOSITELNÉ KNIHOVNY.....	68
9.2	PLATFORMA WINDOWS.....	72
9.3	PLATFORMA ANDROID.....	72
9.4	INTEGRACE V REÁLNÉM ZAŘÍZENÍ A EMULÁTORY.....	72
10	POPIS APLIKACE A VERIFIKACE	73
11	TESTOVÁNÍ	77
11.1	ROZŠÍŘITELNOST APLIKACE.....	77
	ZÁVĚR	78
	SEZNAM POUŽITÉ LITERATURY	80
	SEZNAM OBRÁZKŮ	86
	SEZNAM PŘÍLOH	88

ÚVOD

V současné době je znát velký tlak na vývoj nových aplikací a tvůrci těchto aplikací nemají lehký úkol. Na trhu je velká konkurence a vyvíjená aplikace musí splňovat spoustu kritérií. Vývojář musí zajistit ve velké konkurenci kvalitu daného systému a jeho spolehlivost při různých uplatněních. Ne všichni si dokážou představit, co všechno vývoj nové aplikace musí zahrnovat. To je právě důvod, proč jsem se rozhodl sepsat tuto diplomovou práci.

Tato práce bude obsahovat v kapitole 1 seznámení se základními principy pro dodržení bezpečnosti při tvorbě nové aplikace na platformách iOS, Android a Windows. Každá platforma má své specifické odlišnosti, o nichž by měl čtenář být informován. Vývoj hardwaru a operačního softwaru se v tomto odvětví stále rychle vyvíjí, proto nelze jednoznačně tyto problémy definovat. Multiplatformně jsou tyto hrozby řešeny jinak, nemluvě o verzích operačního systému. Z obecného pohledu budou popsány hrozby, na které je třeba brát zřetel a neotáčet se k nim zády.

Práce naváže na přenositelnost z nativní platformy Windows na multiplatformní framework Xamarin v sekci 2.1 pro vývoj nových aplikací. Čtenář se dozví základní techniky, jak vytvářet aplikace multiplatformně za použití knihoven v Xamarinu, a bude kladen důraz na přenositelné projekty. Bude představen návrhový vzor pro tvorbu uživatelského rozhraní odděleného od aplikačního rozhraní, který zajistí jednodušší a přehlednější implementaci.

Aby bylo možné vysvětlit principy Bitcoinu, je třeba se seznámit okrajově s hashovacími funkcemi a asymetrickou kryptografií. Následně bude navázáno na problematiku virtuální měny Bitcoin. Čtenář se dozví o základních prvcích Bitcoinů, včetně vysvětlení celého systému obchodu s virtuální měnou Bitcoin. V neposlední řadě zde bude představeno aplikační rozhraní pro komunikační kanál se serverem Bitcoin.

Pro bližší pochopení je v Projektové části 7 rozebráno využití prototypové aplikace, která byla vyvinuta v rámci jiného projektu. Součástí celé aplikace je systém virtuální měny, kterou v nynější době používá celý svět. Pro systém virtuální měny byla prototypová aplikace vyvinuta na nativní zařízení Windows, je tedy ideálním výchozím bodem pro aplikaci v Xamarinu.

Následně bude představen nový návrh aplikací pro prostředí Xamarin. Budou prozkoumány možnosti přenositelnosti kódu se stávajícími aplikačními rozhraními, včetně uživatelského rozhraní. Při návrhu aplikace je kladen důraz na budoucí rozšíření, možnosti úprav a vývoj oddělených aplikačních a uživatelských rozhraní 2.10.

V implementační části 9 se čtenář postupně dozví, jak bylo docíleno požadavků z návrhu. Jakým způsobem byla vedena správa kódového celku, jaké bylo použité prostředí

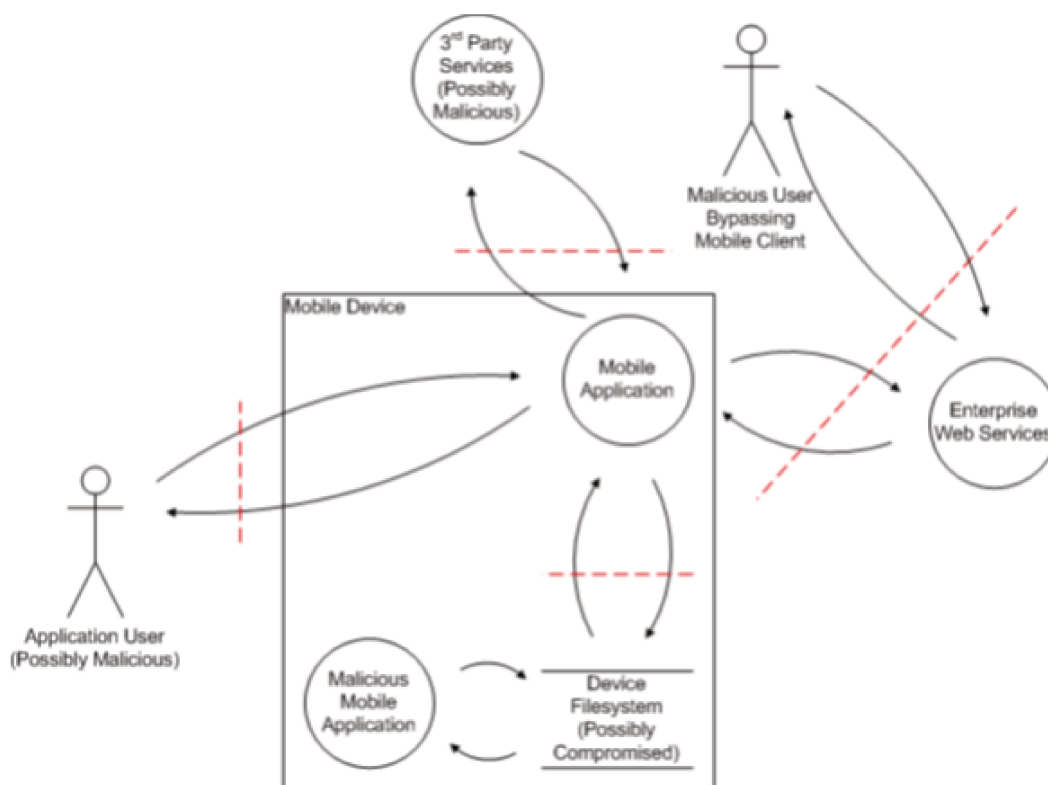
s nástroji a jak byla vedena technická dokumentace.

Souhrnně má práce za cíl zjistit, zda je možné přenést starší realizace aplikací na nový framework Xamarin, a případně zdůvodnit, co se musí upravit, aby aplikace byla ve funkčním stavu.

I. TEORETICKÁ ČÁST

1 BEZPEČNOST MOBILNÍCH ZAŘÍZENÍ

Významným úskalím vývojáře při tvorbě nové aplikace je seznámení se slabými místy každého zařízení. Vývoj bude vždy limitován hardwarem embeded zařízení. Mezi kritické prvky můžeme zahrnout nekvalitní čip, přijímač a další hardwarové prvky. Pokud bude směřovat vývoj na konkrétní zařízení nebo modelovou řadu či konkrétní platformu, vždy je potřeba se seznámit s datasetem. Na základě hardwarové platformy je třeba vybudovat případovou studii hrozeb. Navržená aplikace by měla z těchto studií vycházet a navrhnout správné ošetření. Aplikace bude bezpečná pouze tehdy, pokud využijeme všech možných informací pro zabezpečení. Musíme nastavit zabezpečení i vlastním způsobem bez známých standardů, které pro danou platformu útočníci dobře znají. Jako základní vstup můžeme brát obecný model hrozeb, který je představen na obrázku (Obr. 1.1).



Obr. 1.1 Obrázek popisující možné hrozby od uživatele zařízení, od ostatních aplikací v zařízení, služeb třetích stran a webových služeb. Zdroj [18].

Pokud označíme nedostatečně prověřený systém za bezpečný, podstupujeme potenciální riziko. Fungování systému pak může ohrozit pouze v něm běžící aplikace. Nemůžeme podceňovat útoky přes samotnou mobilní aplikaci, která se stává po nainstalování součástí systému. Útočník se pak nesnaží zaútočit na samotný operační systém, neboť pro něj existují jednodušší cesty, jak ovládnout nebo zneužít systém. Stačí vykrást nebo upravit stávající aplikace a následně podvrhnout v nové verzi s jiným vývojářem do

Google Play, App Storu od Applu či Windows Apps Storu. Potenciální nový uživatel okamžitě nepozná originální aplikaci od nebezpečné. Odvážnější útočníci vydávají se známím užitečné aplikace pro uživatele, které obsahují cílené metody k získání libovolných informací ze zařízení. Nejsou postavené na základě minulých analýz z různých aplikací na internetu.

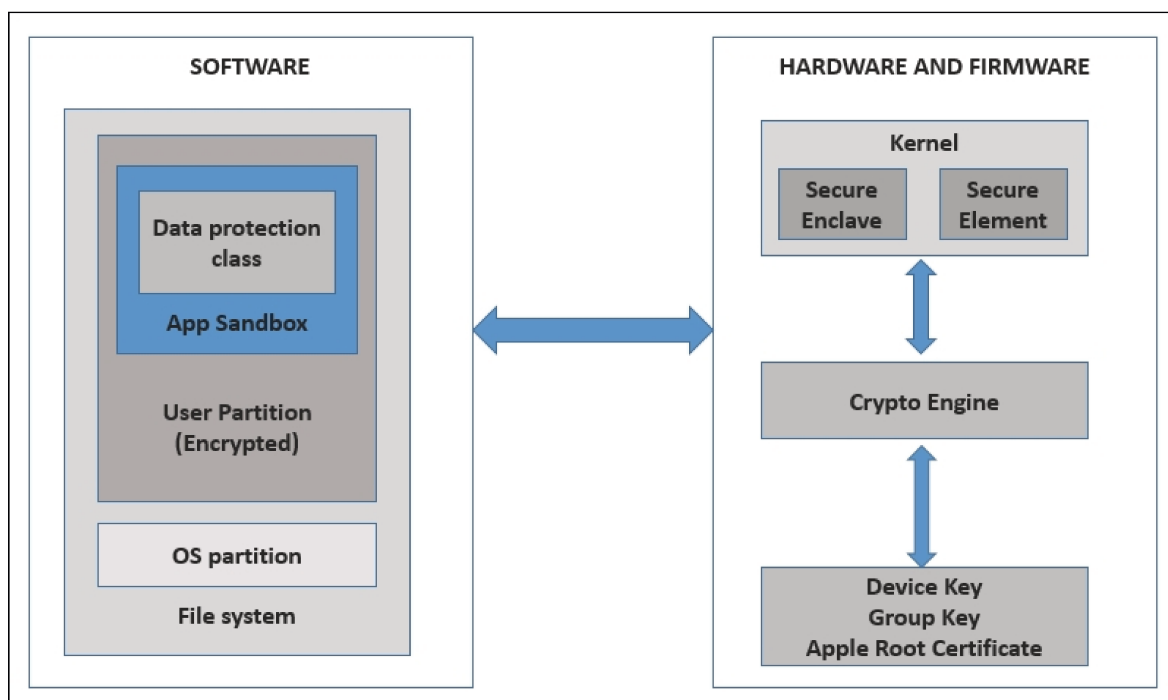
Konkrétní vstupní informace ohrožující bezpečnost by měly být kladně ověřeny a neměly by být používány pro prolomení zabezpečení. Vývoj aplikace vždy vyžaduje uložit uživatelská data v rámci spuštěné aplikace nebo jejich obnovení po uzavření aplikace. Každá platforma řeší způsob uložení dat jinak. Mezi další hrozby mobilních zařízení patří jejich odcizení, ztráta, rozbití nebo zneužití, při kterém se na chvíli ocitnou v nespřímných rukou. I když se zdá zařízení zničené, data zničená být nemusí a lze je znovu oživit. Přístupová oprávnění pro místní soubory a pro samotné databáze jsou také důležitá. Majitel zařízení může nevědomky nainstalovat aplikaci, která slabého místa v systému ihned využije. Síťová komunikace může být nelegálně odchycena a případně i upravena během přenosu, a proto je třeba si chránit citlivá data při jakékoliv komunikaci. Správně zabezpečená architektura aplikace a návrhové bezpečnostní zásady pro tvorbu nové aplikace mohou být velmi užitečné. Aplikace bude daleko bezpečnější, stabilnější a snadněji rozšiřitelná v budoucím vývoji. V této části bylo čerpáno z [21], [20], [22].

1.1 Charakteristiky platforem

Každá mobilní aplikační platforma má odlišné charakteristiky a správný vývojář by měl znát její specifikace. Pojďme si zodpovědět pár základních otázek k tomuto tématu. Jaký se používá jazyk při tvorbě první mobilní aplikace? Má daná platforma možnost emulace na virtuální prostředí? Jsou potřeba poplatky za vývoj v daném prostředí? Co je potřeba k vývoji na daném prostředí?

Pro iPhone a iPad se používá jazyk Objective-C kompilovaný v ARM strojovém kódu. Všichni vývojáři mají možnost vyzkoušet svou aplikaci nejdříve na emulátoru. To je ovšem možné pouze za splnění podmínky, že vlastní Apple počítač. Teprve na něm je možné provést překlad své aplikace. To poměrně prodražuje počáteční náklady na vývoj nové aplikace. Mezi další nutné náklady patří nákup balíčku vývojáře Apple Developer Program, přičemž poplatek nyní činí \$99 na rok podle [4], [6]. Bez těchto kroků nelze umístit produkt na App Store a mohl by se instalovat pouze na zařízení odemknutá pro vývoj nebo ta, u nichž proběhl jailbrake (anglický výraz pro prolomení zařízení). K vývoji je vhodné nainstalovat Xcode 8 studio. Následující obrázek informuje o architektuře (Obr. 1.2).

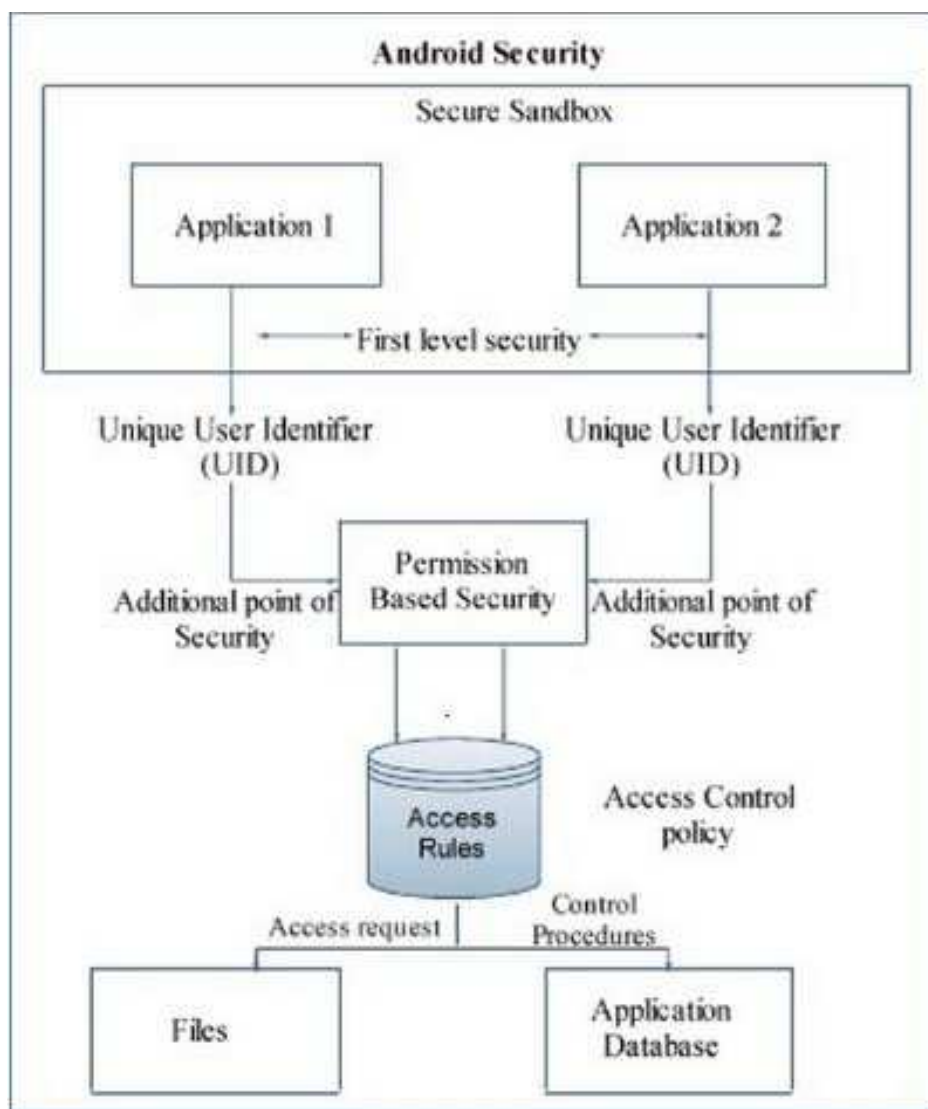
Android platforma naopak používá jazyk Java a zdrojové soubory jsou kompilo-



Obr. 1.2 Architektura zabezpečení iOS zdroj [23].

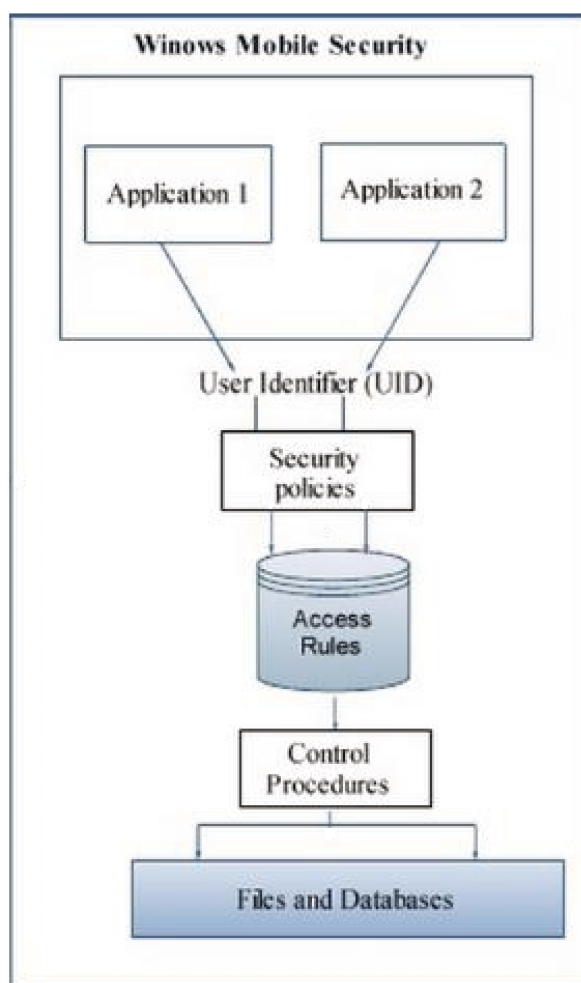
vány do Dalvik Executable (DEX). I zde vývojové prostředí, nazvané Android Studio, umožňuje vytvořenou aplikaci emulovat v místním simulátoru a následně otestovat skrz USB spojení přímo na vybraném zařízení. Lze to, pokud na zařízení bylo povoleno ladění. Oproti jiným platformám Android vyžaduje zápis neboli inicializaci aplikace do konfigurovatelného souboru `AndroidManifest.xml`, od verze 23 lze dynamicky požádat uživatele o oprávnění k přístupu ke zdrojům zařízení. Google Play je oficiálním distributorem všech aplikací pro všechny možné verze operačních systémů Android s certifikátem vydavatele. Poplatek za možnost vývoje je zde jednorázový ve výši \$25 [24] po následné registraci. K vývoji je potřeba nainstalovat jak Android Studio, tak i balíček SDK pro určený operační systém Android. Následující obrázek informuje o architektuře (Obr. 1.3).

Windows Phone platforma je nejbližší uživatelům Windows. Vývoj aplikace je nejčastější v C# nebo C++ s použitím Visual Studia a SDK pro Windows Phone s výborným nástrojem pro ladění. Vývoj v Universal Windows Platform 8 nebo 10 nyní umožňuje distribuovat aplikaci na všechna zařízení, která obsahují systém Windows, od telefonu, notebooku, tabletu až po televizi. Poplatek je zde nejnižší, a to jednorázové uhrazení částky \$19 podle [38]. Visual Studio obsahuje emulátor běžící v Hyper-V stroji. K emulátoru je potřeba minimálně Windows 7 Enterprise anebo vyšší verze. Po provedení registrace účtu je potřeba připojit zařízení a provést počáteční aktivaci, aby zařízení bylo odemknuté pro vývoj [40], [43]. Následující obrázek informuje o architektuře (Obr. 1.4).



Obr. 1.3 Architektura zabezpečení Android zdroj [7].

Na závěr si můžeme vše shrnout. Každá platforma má stejné možnosti pro vývoj. Odlišnosti jsou až v jazyce a v neposlední řadě ve verzích operačního systému. Android obsahuje obrovské množství odlišných operačních systémů. Na porovnání můžeme uvést platformu iOS a Windows, která má z 98% pouze dvě odlišné verze. Grafické zobrazení procentuálního zastoupení platforem na trhu v roce 2014 představuje (Obr. 1.6) a v roce 2016 (Obr. 1.7). Procentuální průzkum trhu s platformami v roce 2016 přehledně zobrazuje (Obr. 1.5).



Obr. 1.4 Architektura zabezpečení Windows Phone zdroj [7].

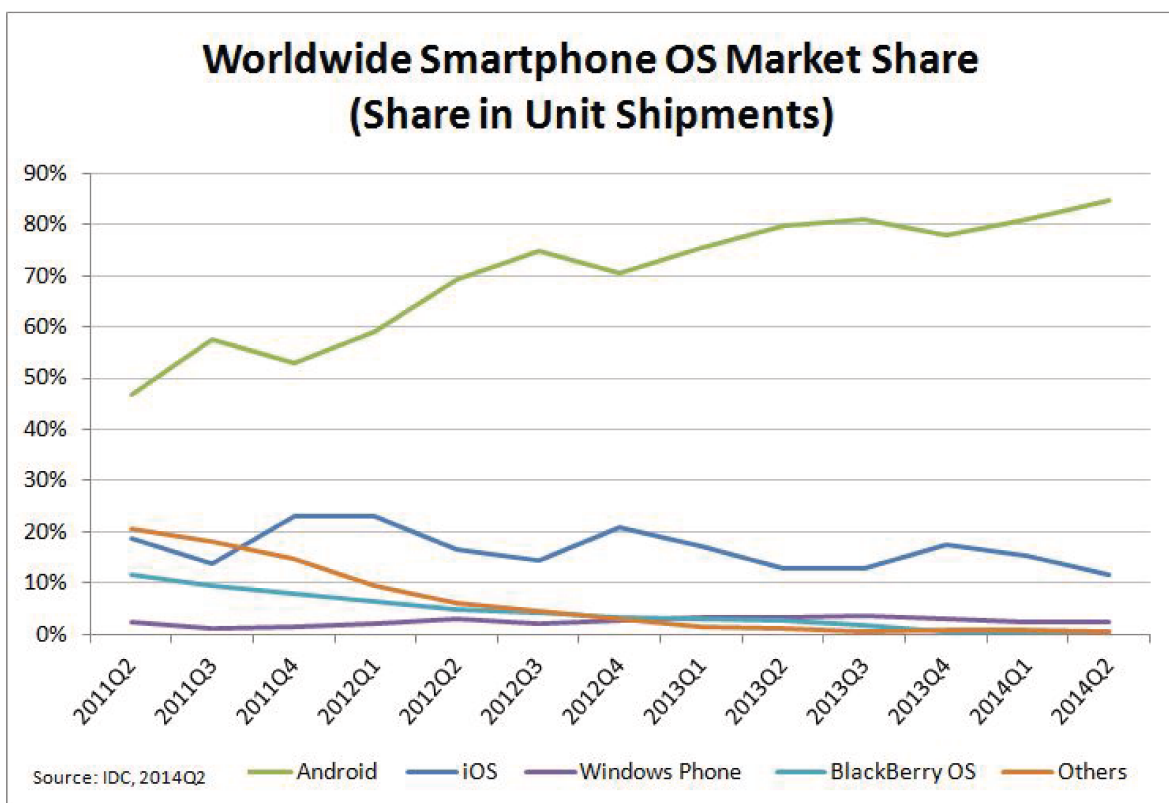
Period	Android	iOS	Windows Phone	Others
2015Q3	84.3%	13.4%	1.8%	0.5%
2015Q4	79.6%	18.6%	1.2%	0.5%
2016Q1	83.4%	15.4%	0.8%	0.4%
2016Q2	87.6%	11.7%	0.4%	0.3%

Source: IDC, Aug 2016

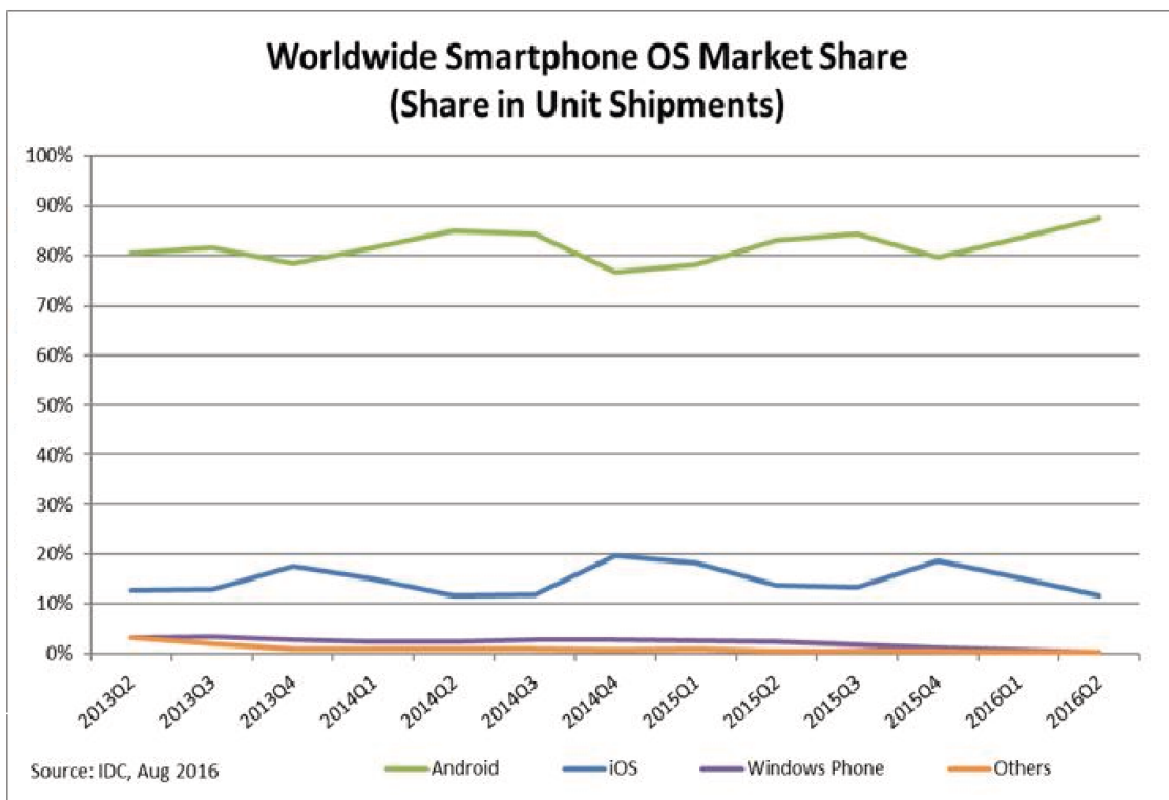
Obr. 1.5 Procentuální průzkum trhu s platformami v roce 2016 zdroj [27].

1.2 Bezpečnostní prvky

Stejně jako se snažíme zabezpečit si domy, auta a po softwarové stránce své počítače, nikdy nesmíme zapomenout i na nejmenší zařízení, která jsou stále při nás. Každá mobilní platforma má odlišné vlastnosti, kterým odpovídají odlišné bezpečnostní prvky.



Obr. 1.6 Zastoupení na trhu s platformami v roce 2014 zdroj [33].



Obr. 1.7 Zastoupení na trhu s platformami v roce 2016 zdroj [27].

Zařízení obsahují nepřehledné množství různých citlivých informací. Jako příklad můžeme uvést e-mailové zprávy, uživatelské kontakty, aktuální polohu zařízení a soukromé fotografie. Kromě toho mobilní zařízení mají přístup k citlivým zdrojům, navázání hovoru a posílání SMS zpráv. Samotný uživatel by si měl promyslet při povolování zdrojů pro aplikaci, zda tyto zdroje aplikace potřebuje. Například: *přehrávač hudby potřebuje přístup k reproduktoru a audio knihovnám, ale nikdy k e-mailové nebo kontaktové schránce.*

Uživatel rozhoduje o přidělení oprávnění k těmto zdrojům. Systém je zabezpečen proti podvodným akcím a proti přístupu k citlivým informacím, pokud uživatel neumožní výhradní právo přístupu. Vývojář by měl žádat o přístup pouze tehdy, pokud je to bezvýhradně nutné, aby se potenciálně zamezilo škodám.

Mobilní zařízení mají schopnost ukládat informace do souborů a databází. Během života zařízení může dojít k jeho ztrátě, odcizení nebo převedení na jiného uživatele bez vymazání informací. Vývojář aplikací by měl být velmi opatrný při ukládání citlivých informací přímo na zařízení. Omezení úplného nebo částečného ukládání informací na zařízení je krokem vpřed v ochraně citlivých dat a minimalizujete riziko zneužití dat.

1.2.1 Platforma iOS

Pro platformu iOS je vytvořeno několik způsobů pro zabezpečení a ochranu zařízení [5]. Jedním z nejběžnějších způsobů zabezpečení je kódový zámek [2], bez kterého se do zamknutého zařízení nelze dostat. Ten tak chrání zařízení a data před vniknutím třetí osoby. iPhone od verze 5s přináší senzor otisku, anglicky *Touch ID*. Systém iOS obsahuje funkci iCloud Keychain, která umí uchovávat hesla a údaje o platební kartě a mezi všemi zařízeními majitele je lze synchronizovat. Apple také vyvinul funkci Find My iPhone, která najde odcizené nebo ztracené zařízení. Aplikace je od první akti-vace zařízení nainstalována. Apple doporučuje zálohu dat do iCloud nebo přímo do MacBooku. Tak či onak se zálohují data jako záložky, kontakty, kalendáře, dokumenty, e-maily, zprávy, poznámky a fotky.

iPhone při instalaci aplikace požaduje od uživatele povolení přístupu ke specifickým zdrojům v rámci instalace [3]. Uživatel rozhodne kladně nebo záporně o své důvěře k aplikaci. Pokud se během používání zjistí, že aplikace je nedůvěryhodná, lze aplikaci nahlásit a oprávnění odebrat včetně odinstalování celé aplikace.

Dále iPhone používá zvláštní prostor označovaný jako karanténa, která zajišťuje oddělený přístup do pouze stanovených míst v zařízení. Soubory v ní umístěné mohou být označeny jako sdílené, které jsou pak dostupné z jiných částí systému. Označení souborů jako chráněné umožňuje přístup k souborům, pouze když je zařízení odemčené [?]. Seznam vlastností zajišťuje ukládání uživatelských preferencí neboli konfigurací

mezi aplikacemi [2].

1.2.2 Platforma Android

Mobilní platforma Android představuje nové řešení založené na komponentách pro vývoj nových aplikací. Aktivní komponenty tvoří základ uživatelského rozhraní, který vytváří nový framework pro vývoj. Uživatelské rozhraní funguje díky servisním komponentám aplikace, v jakémkoliv okamžiku stále komunikuje s uživatelem pomocí zpracování instrukcí na pozadí. Základní součástí je rozhraní poskytující správu sdílené paměti ve formě relační databáze, se kterou komponenty komunikují. Broadcast rozhraní poskytuje elegantní způsob asynchronní komunikace ze systému k aplikaci. Aplikační rámec vytváří jádro pro podporu aplikací, zajišťuje tak flexibilní stupeň spolupráce mezi aplikacemi.

Android vytváří pro každou aplikaci izolovaný prostor oddělený od ostatních aplikací. Samotné zdroje k aplikacím přiděluje nadřazený operační systém podle dostupnosti. Vždy prvně ověří, zda aplikace má povolený přístup ke zdroji definovaný v `AndroidManifest.xml` nebo od verze Android 6 dynamicky za běhu aplikace.

Pro ukládání informací existuje několik možností, například je možné je ukládat do interního úložiště s oprávněním obdobným jako na platformě Linux. Model navržený podle platformy Linux pak zajišťuje oddělení přístupu k souborům aplikace i oprávněním [19]. Další možností je využití přídavného paměťového modulu, který je opět možné chránit stejným modelem pro zabezpečení. Soubory mohou mít několik kontextů: `private`, `append`, `world_readable`, `world_writable`. V neposlední řadě Android vytváří SQLite databázi pro ukládání konfigurací a jiných informací [?].

1.2.3 Platforma Windows

Platforma Windows obsahuje několik bezpečnostních možností, jak správně a bezpečně vytvořit aplikaci. Aplikace mají několik možností pro autentizaci uživatelů od jednoduchých `single sign-on` módů až po vysoce bezpečnou dvoufaktorovou autentizaci. Dále platforma umožňuje použití zámku pro zabezpečení ověření uživatele, biometrický otisk, zabezpečení hesel, čipových karet, komunikaci mezi aplikacemi a v neposlední řadě využití kryptografie pro ochranu klíčů i dat [28], [42].

Jak již bylo zmíněno u platformy iOS, tak i platforma Windows požaduje nastavení oprávnění při instalaci aplikace. Dále platforma Windows podporuje známé typy souborů, které jsou v základním vybavení stolních počítačů s operačním systémem Windows. Umožňuje ukládat informace do prostoru izolovaného od ostatních aplikací v lokální složce zařízení včetně vytvoření lokální databáze SQL. Umožňuje speciální příznaky složky, např. složka společná, sdílená s kontextem (`Media`, `ShellCon-`

tent, Transfers). Informace lze ukládat nejen do interního datového úložiště, ale i na přídatnou paměťovou kartu opět nahrát i oprávnění.

1.3 Odblokování platformy

Všechny mobilní platformy jsou dodávány bez „root“ přístupu neboli bez administrativního přístupu do samotného systému. Uživatel pak na něj nahlíží jako na černou krabičku s nativním přístupem. Tato strategie od výrobců zabezpečuje jak samotný systém proti útoku, tak i proti nevědomému poškození samotným uživatelem při instalaci podvodné aplikace do systému. Ovšem pozor, nikdy se nezapomnělo na vývoj, každé zařízení umožňuje přepnout různým způsobem do vývojového prostředí. Jak toto prostředí zpřístupnit?

Na platformě Apple lze v nastavení zařízení přepnout do vývojového prostředí. Obdobným způsobem lze přepnout Android zařízení, akorát nabídka je skrytá a pro každý typ zařízení se zapíná jiným způsobem. Nejlepší metodou je dohledat informační zdroje na internetu. Windows Phone má zajímavější způsob možností odblokování. Pokud jste vývojář a skutečně chcete vyvíjet, musíte si pořídit vývojářský účet. V rámci registrace účtu vám společnost Microsoft umožní vaše zařízení odemknout. Po odemčení konkrétní platformy lze za pomoci USB kabelu nahrát vývojovou aplikaci přímo na zařízení.

1.4 Instalace aplikací a licencování

Každá nová mobilní aplikace by měla projít řádnou sérií testů ještě před zveřejněním, aby hned v počáteční fázi nebyla zneužita nebo vykradena kvůli chybě v programování. Aplikace by nikdy neměla požadovat přístup ke zdrojům zařízení bez souhlasu. Vývojář by si měl dobře rozmyslet, zda tyto zdroje opravdu potřebuje. Aplikace by měly být instalovány pouze z důvěryhodných zdrojů, jako jsou Google Play, Apple App Store a Windows Apps Store. Všechny tyto obchody zajišťují i zpětnou vazbu pro uživatele pomocí komentářů nabízených aplikací.

U **iPhone** lze instalovat aplikace pouze z oficiálních zdrojů, jako je App Store. Aplikace nejsou nějak bezpečnostně kontrolovány, zda splňují kritéria. Apple hlavně kontroluje regulérnost aplikačního rozhraní API a neporušování lidských práv. Apple zřídil černou listinu podvodných a zakázaných aplikací, kterou kontroluje zařízení při instalaci aplikace i během provozu. Licencování je zajištěno na straně App Store a zařízení dostává tyto informace během instalace.

Android umožňuje instalaci z obdobného internetového obchodu nazvaného Google Play. Navíc je možné instalovat aplikace i z APK balíčku přes USB. Obchod udržuje informace o vydavateli, datum zveřejnění, verzi aplikace i certifikátu aplikace. Za chodu zařízení je ověřována aktuálnost licenčních podmínek pro konkrétní nainstalované apli-

kace pomocí dotazů na Google Play.

Pro **Windows Phone** jsou stanoveny instrukce, jak mají být obdobně ověřeny aplikace ve Windows Apps Store. Certifikace aplikace obnáší splnění základních testů zabezpečení od Microsoft pro vstup do Storu [39]. K základním vstupním informacím, které je nutno do obchodu zadat patří snímky aplikace, popis funkčnosti a verzování obdobně jako u ostatních platforem.

1.5 Šifrovací rozhraní

Vzhledem k riziku zneužití je výhodné neukládat žádné citlivé informace do zařízení. V případech, kdy se vývoj aplikace neobejde bez uložení citlivých údajů, by měly být citlivé informace uloženy v šifrované podobě tak, aby se zabránilo vykradení. Zde vzniká největší problém, neboť šifrování dat pomocí zařízení je výpočetně náročný proces jak pro zašifrování, tak dešifrování. Otázkou tedy je, jak stanovit vhodné šifrování? Proces musí být takový, aby jeho uživatel dokázal rychle a snadno obnovit původní informace, ale zároveň byl dostatečně bezpečný proti náporu útočníků. Lze očekávat, že útočník má možnost si dané zařízení pořídit, nachystat si prostředí tak, aby byl schopný zařízení prolomit během několika minut v rámci komunikace nebo podvodné aplikace. Pak je vhodné se ptát: Jaké šifrovací knihovny jsou k dispozici pro nativní aplikační rozhraní? Jaké šifrovací knihovny třetí strany jsou k dispozici? Jaká jsou omezení daných knihoven? Jak se nejlépe chránit? Jak tyto ochrany zřídit na zařízeních?

iOS poskytuje přístup k různým funkcím ověření a správu klíčů. Aplikace mohou přistupovat různě k možnostem šifrování. Dále je možné využít přístup ke službě Keychain neboli řetězci klíčů, která umožňuje, aby aplikace mohla bezpečně ukládat lokální data, jako jsou hesla a šifrovací klíče. Aplikace mohou přistupovat ke svým řetězcům klíčů, ale jiné aplikace k nim nemají přístup. Položky uložené v řetězci klíčů mohou být rovněž uloženy tak, že mohou být získány pouze tehdy, když bylo zařízení odemčeno s PINem. Bez PINu bude zařízení daleko více ohroženo. Bohužel PIN lze snadno prolomit z desítek tisíc možných řešení pro čtyřmístnou variantu. Uživatel by se měl tedy hodně zamyslet, zda na tak malém zařízení chce skutečně ukládat tak citlivá data (např. bankovní přístupy). Tak malé zařízení ani v dnešní době nedokáže vytvořit v porovnání s výkonnějšími stolními počítači dostatečně silnou šifru. K šifrování databáze lze využít SQLCipher rozhraní s AES 256.

Android poskytuje přístup k šifrovacímu aplikačnímu rozhraní ve standardní databázi přes knihovny `javax.crypto`. Také lze použít knihovny Bouncy Castle Java, které vedou k dobrému řešení zabezpečení [28].

Windows Phone od verze 8 umožňuje zavedení šifrovacích metod v rámci celého zařízení opět s přístupem pouhého PINu, takže nedostatečného zabezpečení. Pro zvý-

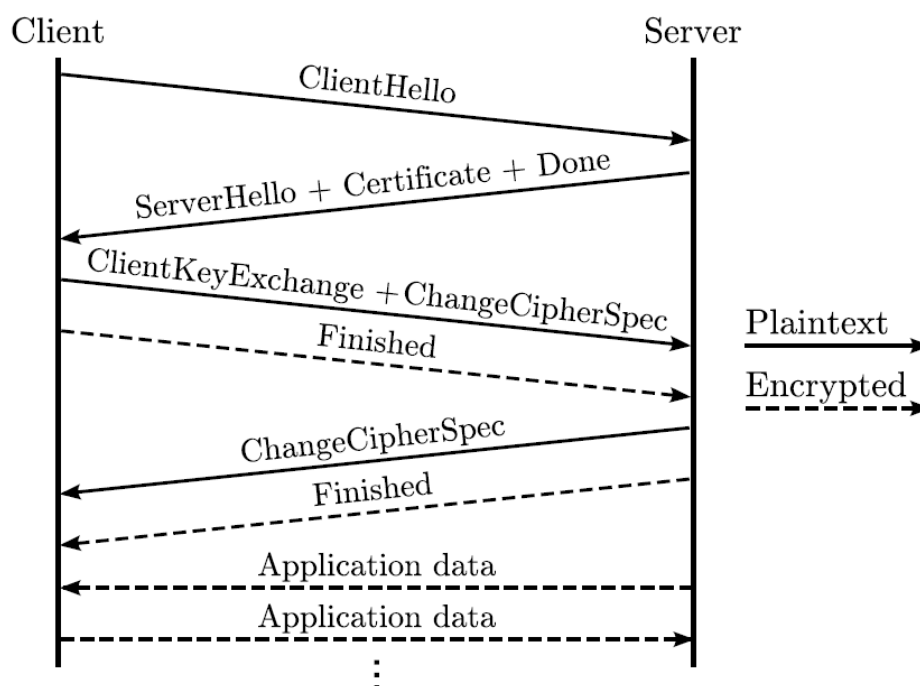
šení bezpečnosti lze využít kryptografické knihovny pro symetrické a asymetrické klíče, které jsou nyní dostupné na Universal Windows Platform.

1.6 Síťové komunikace

V dnešní době většina mobilních aplikací vyžaduje síťovou komunikaci, neboť všechny procesy nejsou zpracovávány přímo na zařízení. Každá mobilní platforma nabízí několik možností spojení s okolním světem, jako jsou například wi-fi, bluetooth a datový paušál. I zde vývojář musí myslet na citlivá data, aby nebyla během komunikace stažena nebo podvržena. Šifrování zvýší ochranu dat při komunikaci. Dokonce je možné naučit aplikaci, jaké sítě má používat k šifrované nebo nešifrované komunikaci při dané vnitřní činnosti. Citlivá data by nikdy neměla být v nešifrované komunikaci přítomna. Vývojář by měl vždy využít TLS/SSL socketu a síťového protokolu HTTPS [26].

Protokol TLS (anglicky *Transport Layer Security*, version 1.2) neboli Zabezpečení přenosové vrstvy a jejího předchůdce SSL (anglicky *Secure Sockets Layer*, version 3) popsany RFC 5246 poskytuje důvěrnost, integritu dat, neodmítnutí, ochranu přehrávání a ověřování prostřednictvím digitálních certifikátů, a to přímo na protokolu TCP. Bohužel verze 3 protokolu SSL již není pro použití z bezpečnostních důvodů doporučena a je zranitelná. Oba protokoly jsou často označovány přímo jako „SSL“, jsou to kryptografické protokoly, které zajišťují zabezpečení komunikace v počítačové síti. Tyto protokoly mají široké uplatnění v aplikacích, jako jsou prohlížeče webu, e-mailu, internetové faxování, rychlé zasílání zpráv, Voice over IP (VoIP) a virtuální protokoly soukromé sítě (VPN). Webové stránky používají protokol TLS k zajištění veškeré komunikace mezi svými servery a webovými prohlížeči. Cílem protokolu TLS je především poskytovat soukromí a integritu dat mezi dvěma komunikujícími počítačovými aplikacemi. Dokument RFC 4279 popisující protokol TLS předkládá použití těchto algoritmů: RSA, DH-RSA, ECDH-RSA (více v 4.1) a mnoha dalších převážně pro výměnu klíčů, ověření shody a autentizaci. Dále obsahuje zabezpečení šifry proti veřejně známým proveditelným útokům, například AES GCM, ARIA GCM a další. Důvěryhodnost je zajišťována převážně algoritmy HMAC-MD5, HMAC-SHA1 a HMAC-SHA256/384 (více v 3, jedná se o kombinaci staré a nové hashovací funkce).

Zde je jednoduchý nástin komunikace za pomoci protokolu TLS, kterou lze rozdělit na dvě fáze: Za prvé počáteční ustanovení spojení (anglicky *handshake*), kdy zpráva začíná ClientHello, částí obsahující údaje o identifikaci verze protokolu, použitý seznam šifrovacích sad a rozšíření. (Úplný seznam těchto identifikátorů je k dispozici na webové stránce IANA [26].) Všechny TLS zprávy vyměněné během počátečního handshake přenosu nejsou zašifrovány, dokud nebudou vytvořeny a potvrzeny sdílené klíče ukončené zprávy. Ve druhé fázi se ustanovují klíče pro komunikaci.



Obr. 1.8 Obrázek popisující počáteční ustanovení spojení [26].

Pro TLS/SSL komunikaci musí být užito vhodně silné šifrování, které bude rychlé, ale zároveň dostatečně bezpečné. Každá platforma nabízí možnost citlivou komunikaci zabezpečit pomocí kryptografických knihoven. iOS umožňuje využití BSD soketů pro komunikaci. Dále poskytuje pokročilé knihovny Bonjour, peer-to-peer a http/https. Android je vybaven standardními třídami `java.net`, balíčkem `org.apache.http`, specifickými třídami pro sokety, http/https a knihovny pro zabezpečenou komunikaci `android.net.SSLCertificateFactory`. Windows Phone využívá třídy `System.Net.Sockets` s možností TCP i UDP komunikace, třídu `WebClient` pro odesílání a přijímání dat v URI a také zde je možnost http/https komunikace. Knihovny se hlavně zaměřují na ověření identity, šifrování komunikace a dotazování na uživatelské jméno i heslo v kódování base-encoded.

1.7 Nativní spuštění kódu

Nativní kód otvírá další možnosti vzniku chyby v aplikaci, například přetečení zásobníku, formátování řetězců a správu paměti. Pokud aplikace vyžaduje přístup k nativnímu kódu, vývojář by měl využít utilit pro správu paměti včetně kontroly rozsahu polí využitých v aplikaci. Některé platformy dokonce nabízejí přístup k užitečnému technologickému řešení pro ošetření přetečení vyrovnávací paměti, správu nespuštěných zásobníků či správu adresního prostoru. Všechny tyto utility pomáhají zabezpečit aplikaci včetně samotného operačního systému.

U iOS platformy je Objective-C sestaven do ARM strojového kódu, kde hrozí zejména

přetečení zásobníků. Android naopak používá k programování Javu a následný kód je sestaven do bytekódu, který běží na virtuálním stroji Dalvik. Dále využívá možnosti knihoven „Native Development Kit“ pro spouštění rychlejšího nativního kódu v C/C++. Tento nativní kód pak neumožňuje využití automatické správy paměti z Dalviku. Windows Phone aplikace jsou typicky psány v C# a také zde lze využít nativní kód v C/C++ včetně všech rizik [41].

1.8 Mobilní prohlížeč

Mobilní platformy obsahují vestavěný prohlížeč s definovanými pravidly k přístupu na internet. Naprogramovaná aplikace může používat tuto aplikaci samostatně nebo ji zabudovat přímo. I pro útočníka je to přístupový bod, který lze využít k napadení. Vývojář by měl dobře znát možné verze prohlížečů a následně provést změny v aplikaci, aby zvýšil její bezpečnost. Útočníkům mohou zůstat otevřené různé cesty, kterými se pokouší zmanipulovat aplikace. Pokouší se za pomoci podvržených webových odkazů vykonat na cílové aplikaci stav nebo situaci, která by poškodila uživatele, a to na základě nežádoucích parametrů v URL adrese. Je tedy potřeba správně ověřit příchozí data a vyžádat si odpovídající potvrzení od uživatelů aplikace před provedením citlivých akcí.

iOS platforma využívá prohlížeč Safari s WebKit HTML rendering engine. U Android platformy to může být odlišné, protože ve světě je několik verzí OS, ale typicky má Android prohlížeč WebKit HTML rendering engine nebo Chrome s JavaScript engine. Windows Phone používá Internet Explorer pro mobilní platformu s JavaScriptem a Flashem.

1.9 Deset nejčastějších chyb při vývoji

Známý otevřený projekt OWASP (Open Web Application Security Project)[50], [49], který kontroluje nevydělečná organizace, v roce 2016 provedl nový průzkum bezpečnostních hrozeb. Samozřejmě pod organizací OWASP lze najít spoustu dalších užitečných doporučení pro odlišné projekty [59], [61], [44], [29]. Deset nejčastějších chyb, na které by si vývojáři měli dát pozor v roce 2016, viz (Obr. 1.9).

1.9.1 M1 - Nesprávné použití Platforem

Do této kategorie patří zneužití vlastností platformy nebo nepoužití bezpečnostních kontrol v rámci platformy. Například přístupové oprávnění v rámci dané platformy, zneužití TouchID, Keychain nebo jiné bezpečnostní kontroly, která je součástí mobilního operačního systému.



Obr. 1.9 Deset mobilních rizik podle OWASP v roce 2016 [47], [48].

1.9.2 M2 - Nebezpečně uložená data

Tato nová kategorie je kombinací M2 + M4 z desítky v roce 2014 [?]. To se týká nezabezpečeného ukládání dat a nechtěného úniku dat. Nikdy nelze předpokládat, že zařízení je v bezpečném prostředí. Zařízení může být ukradeno nebo může být různě manipulováno s informacemi uvnitř. Vývojář by měl dodržovat práci s citlivými daty a udržovat je v zabezpečené části zařízení, když s nimi potřebuje pracovat. Jinak mohou být zpracovány škodlivými aplikacemi, které mohou mít do veřejných míst přístup. Vývojář musí být obeznámen v první řadě s tím, jak se provádí cachování dat, logování, a s dalšími základními vlastnostmi systému.

Uživatel může některé hrozby snížit takto:

- Neukládat do zařízení příliš citlivá data, jako jsou přihlašovací údaje k bance.
- Případné nezbytné nutnosti ukládat do zabezpečeného úložiště v šifrované podobě.
- Hlídat si zařízení.

1.9.3 M3 - Nezabezpečená komunikace

Týká se špatného navázání spojení, nesprávné verze SSL, slabého vyjednávání, nešifrovaných sdělování citlivých aktiv atd. Při komunikaci by se mělo vždy vynutit TLS/SSL šifrování se silnými algoritmy. Obvyklou chybou při vývoji je nešifrované spojení k třetí stranám. Vyvarovat se povolení self-signed certifikátů nebo používání zastaralých protokolů.

- Vyžadovat vždy komunikaci se SSL, když jsou zpracovávány citlivé informace.
- Pokud to vyžaduje situace, používat silnější šifrovací algoritmy při komunikaci.
- Kontrolovat platnost certifikátů, zda nebyl z nějakého důvodu odvolán nebo zda nevypršel.

1.9.4 M4 - Nebezpečná autentifikace (ověřování)

Tato kategorie představuje ověřování koncového uživatele nebo špatné řízení relace. Většina chyb může vzniknout při vývoji, kdy špatně napsaná aplikace opětovně žádá ověření uživatele pro ustanovení relace nebo nesprávnou dobu se provádí ověření.

Příkladem mohou být:

- Není-li dostupná identifikace uživatele v určený čas. Vypršení časového rámce.
- Neschopnost udržet identitu uživatele.
- Slabosti sestavení sezení a jeho správy.
- Jakékoliv sezení by po odhlášení mělo zaniknout a nezůstat v paměti zařízení.
- Tokeny pro sezení by neměly být odhadnutelné.

1.9.5 M5 - Nedostatečná kryptografie

Jde o problémy vztahující se k nedostatečně šifrovaným citlivým informacím uživatele. Šifrování je nedostatečné z nějakého důvodu uvedeného níže, ale ne z důvodu jeho vytváření. Komunikace s TLS nebo SSL patří vždy do kategorie M3. V případech, kdy aplikace nedokáže používat šifrování vůbec, je porušena ihned kategorie M2. Tato kategorie představuje pokus o kryptografii, který ale nebyl proveden správně.

Špatné použití kryptografie:

- Aplikace využívá stejný proces k šifrování i dešifrování dat. Může to vést k zásadní chybě, pokud útočník využije tento proces k dešifrování dat.

- Mobilní aplikace využívá hodně slabé šifrovací algoritmy, které jsou snadno prolomitelné.

Vývojář by měl ctít tyto zásady:

- Dobrým pravidlem je využít stávající doporučené algoritmy a nevytvářet vlastní kvůli možným chybám.
- Vyvarovat se používání slabých algoritmů jako RC2, MD4, MD5, SHA1, u nichž už byla dokázána slabost. Blíže budou popsány v krátké kapitole o šifrování.
- Nemanipulovat špatným způsobem s klíčem pro dešifrování, aby nedošlo k jeho odchyčení jinou aplikací.

1.9.6 M6 - Nesprávná autorizace (oprávnění)

Kategorie se zabývá případnými nedostatky v oprávnění. Například rozhodnutí o autorizaci na straně klienta, vynucené prohlížení a další. Jde o případy, kdy aplikace nemá ověření uživatelů vůbec nebo ho nemá tam, kde by mělo být. Příklad: Poskytnutí anonymního přístupu k určitému zdroji či službě; následkem je odcizení informací nebo manipulace s aplikací i daty.

1.9.7 M7 - Kvalita kódování klienta

Jedná se o kódovou stránku na klientské části mobilní aplikace. Měly být ošetřeny části kódu jako přetečení zásobníku, formátování řetězců a různé jiné chyby na úrovni kódu včetně přepisu kódu.

1.9.8 M8 - Manipulace s kódem

Do této kategorie patří úpravy na binární úrovni, lokální modifikace zdrojů, metody hooking a swizzling a dynamická modifikace paměti. Útočník může buď přímo upravit kód, nebo měnit obsah paměti dynamicky. Dále se může pokusit změnit nebo nahradit systém aplikačního rozhraní; následkem je převzetí celého komunikačního kanálu. Zneužití dat pomocí modifikace či využití zdrojů aplikaci přiřazených. Útočník tak získává přímý přístup k aplikaci pro osobní nebo finanční obohacení.

1.9.9 M9 - Reverzní inženýrství

Tato kategorie zahrnuje analýzu finálního jádra aplikace v zdrojovém kódu knihoven, algoritmů a dalších balíčků. Vše může útočníkovi poskytnout více informací k napadení aplikace a nalezení zranitelných míst. Je potřeba se vyvarovat zveřejňování kódu aplikace a použít metody pro zneprůhlednění kódu ve finální verzi aplikace (anglicky

označeno *obfuscate*). Je zapotřebí útočníka co nejvíce zmást a zpomalit, pokud se pokouší zjistit informace o struktuře kódu.

1.9.10 M10 - Cizí funkčnost

Často se stává, že vývojář v aplikaci ponechá skrytou funkčnost (anaglicky *backdoor*) nebo jiné vnitřní bezpečnostní ovládací prvky při vývoji. Například vývojář může neúmyslně zakomentovat hesla nebo může dojít k zablokování dvoufaktorové autentizace v průběhu testování.

2 ANALÝZA VÝVOJOVÉHO PROSTŘEDÍ XAMARIN

Společnost Xamarin je softwarová společnost založená v květnu 2011 v San Francisku ve státě Kalifornie. Je založený na C# a vývojáři mohou používat i nástroje pro psaní nativních aplikací Android, iOS s možností sdílení kódu pro tyto platformy. V únoru 2016 společnost Microsoft oznámila, že podepsala definitivní smlouvu o koupi společnosti Xamarin. Podmínky transakce nebyly zveřejněny, novinový deník Wall Street Journal pouze oznámil cenu v rozmezí 400 až 500 milionů dolarů. Xamarin nyní používá tisíce zákazníků po celém světě včetně významných společností jako Alaska Airlines, Coca-Cola Bottling, Honeywell a JetBlue. Jedinečné řešení společnosti Xamarin je na trhu více jak čtyři roky.

2.1 Co je Xamarin.Forms

V kombinaci s programem Visual Studio nabízí společnost Xamarin bohatou mobilní vývojovou platformu, která umožňuje vývojářům vytvářet mobilní aplikace s použitím C# a přinášet plně nativní zkušenosti s mobilními aplikacemi do všech důležitých zařízení iOS, Android a Windows. Xamarin umožňuje vývojářům využívat produkty v prostředí .NET knihoven, používaných v mobilních aplikacích, a používat C# k zápisu do plné sady nativních rozhraní API a mobilních funkcí poskytovaných danou platformou. To umožňuje vývojářům snadno sdílet společný kód aplikace v aplikacích iOS, Android a Windows, a přitom stále přináší plnohodnotné zkušenosti pro každou platformu. Prostřednictvím služby Xamarin Test Cloud mohou všechny typy vývojářů mobilních aplikací testovat a vylepšovat kvalitu aplikací pomocí tisíců telefonů a zařízení poskytovaných v Cloudu [60].

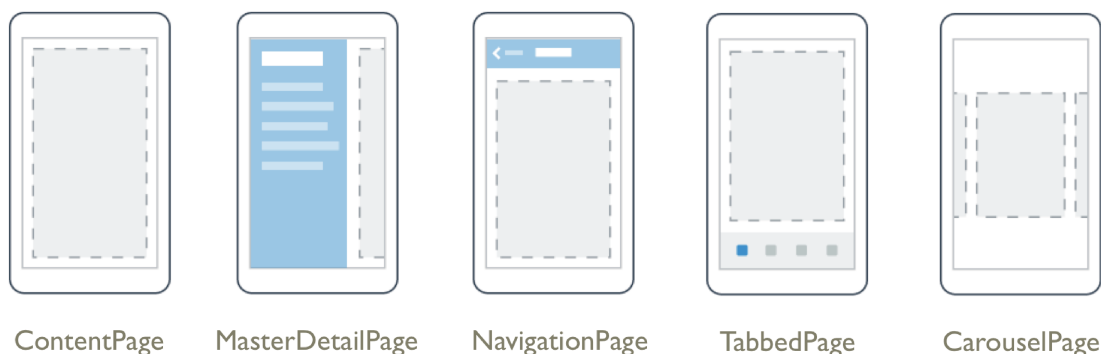
Xamarin.Forms je nejlepší pro:

- Aplikace pro zadávání dat
- Prototypy a důkazy o koncepci
- Aplikace, které vyžadují malou funkčnost specifickou pro platformu
- Aplikace, kde sdílení kódu je důležitější než uživatelské rozhraní

2.2 Navigace (Navigation)

Správa navigace kombinuje celou škálu odlišných stránek lze pomocí níže uvedeného popisu stran docílit obdobnou funkcionalitu jako na nativní úrovni platform iOS, Android a Windows. Uživatel je schopen se pomocí stránek navigovat, vytvářet základní náhledy a případně vytvořit rychle boční menu přes stránky dopředu a dozadu podle

potřeby. Třída také implementuje navigaci jako zásobník v podobě FIFO posledních objektů. Viz (Obr. 2.1)



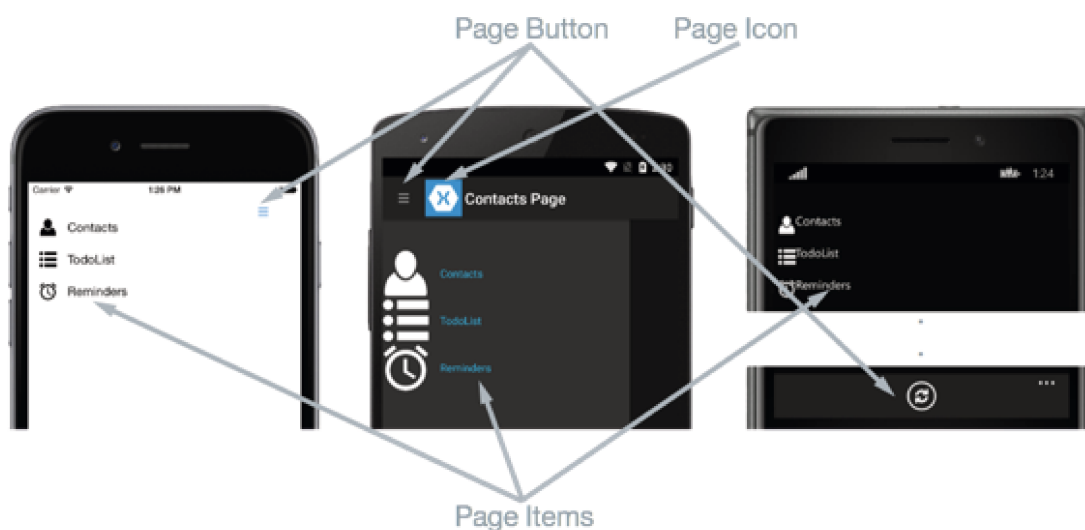
Obr. 2.1 Obrázek popisující možné kombinace stránek pro docílení základní funkcionality navigace. [51].

2.2.1 ContentPage

Jedná se o nejzákladnější třídu. Poskytuje běžné zobrazení stránky, kterou lze libovolným způsobem vyplnit.

2.2.2 MasterDetailPage

Speciální třída poskytující rychlé boční menu. Je určena pro správu hned dvou stránek, hlavní stránky pro zobrazení položek a stránky pro podrobnosti, která zobrazuje podrobnosti o položkách na hlavní stránce. Viz (Obr. 2.2)



Obr. 2.2 Obrázek popisující fungování MasterDetailPage [51].

Umístění seznamu položek je na každé platformě shodné a zvolením jedné z položek se dostanete na příslušnou stránku s podrobnostmi. Navíc hlavní stránka obsahuje také

navigační panel, který obsahuje tlačítko, které lze použít k navigaci na aktivní stránku s podrobnostmi. Umístění navigačního panelu závisí na platformě.

Navigační chování Průběh navigace mezi hlavními a podrobnými stránkami závisí na platformě: Na stránce iOS se stránka s *podrobnostmi posouvá* doprava, protože hlavní stránka snímků se posouvá zleva a levá část stránky s podrobnostmi je stále viditelná. V systému Android se *podrobnosti* a hlavní stránky *překrývají* navzájem. Platforma Windows umožňuje stránku s *podrobnostmi* pouze *zaměnit*, nelze je překrýt. V danou chvíli nebude viditelná hlavní stránka ani z části. Následující ukázka kódu dokumentuje vytvoření rychlého postranního menu (neboli menu s podrobnostmi) s jednou položkou pro přechod na vybranou stránku.

```
public class MainPageCS : MasterDetailPage
{
    MainPageCS masterPage;
    public MainPageCS ()
    {
        masterPage = new MainPageCS ();
        Master = masterPage;
        Detail = new NavigationPage (new ContactsPageCS ());
        ...
    }
    ...
}
```

```
public partial class MasterPage : ContentPage
{
    public ListView ListView { get { return listView; } }

    public MasterPage ()
    {
        InitializeComponent ();
        var masterPageItems = new List<MasterPageItem> ();
        masterPageItems.Add (new MasterPageItem {
            Title = "Contacts",
            IconSource = "contacts.png",
            TargetType = typeof(ContactsPage)
        });
        ...
    }
}
```

```

    ...
}

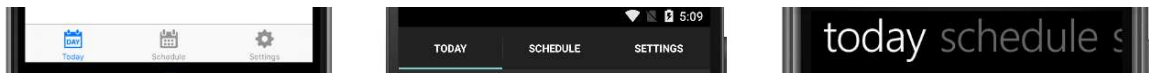
```

2.2.3 NavigationPage

Díky této třídě lze zakomponovat jednoduchou navigaci přes stránky dopředu a dozadu podle potřeby. Třída také implementuje navigaci jako zásobník v podobě FIFO posledních objektů, na které se lze zpětně odvolávat.

2.2.4 TabbedPage

Třída se kombinuje s třídou `ContentPage`, kde jednotlivé karty jsou vždy prázdná stránka. Navigace umožňuje pohybovat se doleva a doprava s viditelnými kartami. Každá karta pak může obsahovat různé detaily. Viz (Obr. 2.3)



Obr. 2.3 Obrázek popisující fungování `TabbedPage` [51].

Definice v Xaml:

```

<TabbedPage
  xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  xmlns:local="clr-namespace:TabbedPageWithNavigationPage;"
  assembly="TabbedPageWithNavigationPage"
  x:Class="TabbedPageWithNavigationPage.MainPage">
  <local:TodayPage />
  <NavigationPage Title="Schedule" Icon="schedule.png">
    <x:Arguments>
      <local:SchedulePage />
    </x:Arguments>
  </NavigationPage>
</TabbedPage>

```

Definice vytvoření v C#:

```

public class MainPageCS : TabbedPage
{
  public MainPageCS ()
  {
    var navigationPage = new NavigationPage

```

```
(
    new SchedulePageCS ()
);
navigationPage.Icon = "schedule.png";
navigationPage.Title = "Schedule";

Children.Add (new TodayPageCS ());
Children.Add (navigationPage);
}
}
```

2.2.5 CarouselPage

Na rozdíl od TabbedPage neobsahuje karty, ale pouze stránky, které uživateli umožní pohyb ze strany na stranu a procházet vytvořenou galerií. Modal Pages Modální stránka nabádá uživatele, aby dokončili samostatný úkol, od něhož nelze navigovat, dokud nebude úloha dokončena nebo zrušena. Displaying Pop-Ups Představuje vyskakovací nabídky, anglicky pop-ups, v Xamarinu jsou pro uživatelské rozhraní definovány dvě: upozornění a seznam akcí. Tyto prvky rozhraní mohou být použity k dotazování uživatelů jednoduchými otázkami a k vedení uživatelů prostřednictvím úkolů. Upozornění lze zapsat takto:

```
DisplayAlert ("Alert", "You have been alerted", "OK");
```

Upozornění s otázkou:

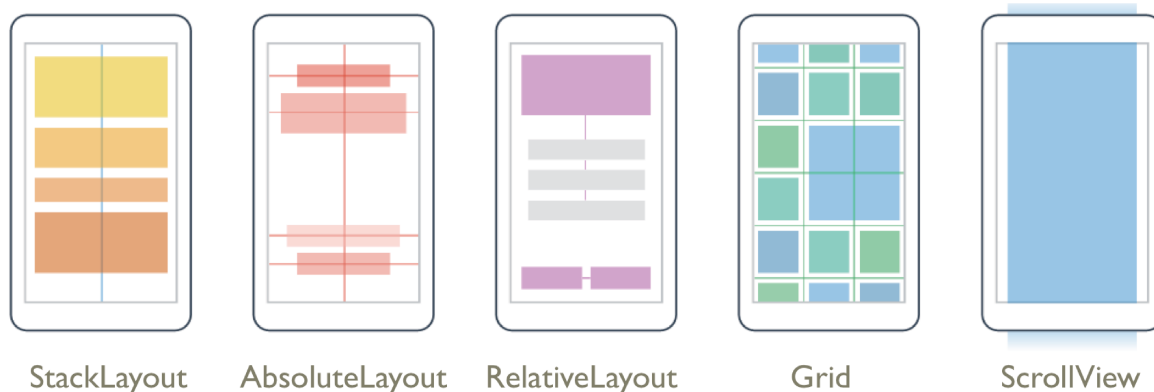
```
async void OnAlertYesNoClicked (object sender, EventArgs e)
{
    var answer = await DisplayAlert
        ("Question?", "Would you like to play a game", "Yes", "No");
}
```

Seznam akcí:

```
async void OnActionSheetSimpleClicked(object sender,
EventArgs e)
{
    var action = await DisplayActionSheet (
        "ActionSheet: Send to?", "Cancel", null, "Email");
}
```

2.3 Rozvržení (Layouts)

Xamarin má několik uspořádání a funkcí pro organizování obsahu na obrazovce. Rozložení, které uživatel zvolí ve své aplikaci, mu může pomoci, ale i ztížit práci s aplikací. Při vytváření atraktivní a použitelné aplikace je potřeba nějaký čas na zvážení, jak může každé rozvržení fungovat. Pomůže to pak psát čistší a škálovatelný uživatelský kód. Obrazovka může mít kombinaci různých rozvržení pro dosažení konkrétního návrhu. Rozvržení lze definovat jak v XAML části, tak i přímo v kódu C#. Viz (Obr. 2.4)



Obr. 2.4 Obrázek popisující možné kombinace rozvržení na stránce [51].

- StackLayout slouží k zobrazení pohledů podél čáry, která je buď vodorovná, nebo svislá.
- AbsoluteLayout slouží k zobrazení pohledů, jejichž velikost a pozice jsou zadávány buď jako explicitní hodnoty, nebo hodnoty relativní k velikosti rozvržení.
- RelativeLayout se používá k zobrazení pohledů, jejichž velikost a pozice jsou určeny jako hodnoty relativní k hodnotám rozvržení nebo z jiného pohledu.
- Grid se používá pro zobrazování prvků v řádcích a sloupcích. Nikdy by se neměl používat na zobrazování dat v tabulce, k tomu slouží pohledy ListView nebo TableView.
- ScrollView je určeno k prohlížení dlouhého textu a postupného posouvání na konec obdobně jako ve webovém prohlížeči.

2.4 Běžné prvky

Prvky pro běžné použití, je jich hned několik, lze libovolně kombinovat a různě modifikovat. Pro seznámení budou popsány prvky, které lze najít na každé platformě.

2.4.1 Label

Xaml:

```
<Label TextColor="#77d065" FontSize="20"
  Text="This is label."/>
```

C#:

```
var label = new Label { Text="This is a label.",
  TextColor = Color.FromHex("#77d065"), FontSize = 20 };
```

2.4.2 Tabulka

```
public class App : Application
{
    public App()
    {
        MainPage = new ContentPage {
            Content = new TableView {
                Intent = TableIntent.Form,
                Root = new TableRoot ("Table Title") {
                    new TableSection ("Section 1 Title") {
                        new TextCell {
                            Text = "TextCell Text",
                            Detail = "TextCell Detail"
                        },
                        new EntryCell {
                            Label = "EntryCell:",
                            Placeholder = "default keyboard",
                            Keyboard = Keyboard.Default
                        }
                    },
                },
            },
        },
        ...
    }
    ...
}
```

2.4.3 Styly

Styly lze použít k úpravě vzhledu štítků, položek a editorů. Styly mohou být definovány jednou a používány mnoha pohledy, ale styl je možné použít pouze s pohledy

jednoho typu. Ke stylům může být přidán specifický klíč, pomocí něj lze styly snadněji přidělovat.

2.4.4 Fonty

Pro nastavení písma v kódu se použijí tři vlastnosti:

- **FontFamily** – název písma string
- **FontSize** – velikost písma jako double
- **FontAttributes** – řetězec specifikující informace o stylu, jako je Italic a Bold

FontSize lze nastavit pomocí přesné velikosti nebo použít třídu NamedSize s výčtovým seznamem. FontAttribute lze nastavit tyto styly: None (žádný), Bold (tučný), Italic (kurzíva). FormattedString neboli formátovací řetězec se skládá z jedné tříd Span, každá z nich může mít své vlastní atributy formátování. Třída Span má následující atributy:

- **Text** – hodnota, kterou je třeba zobrazit
- **FontFamily** – název písma
- **FontSize** – velikost písma
- **FontAttributes** – informace o stylu, tedy kurzíva a tučné písmo
- **ForegroundColor** – barva textu
- **BackgroundColor** – barva pozadí

2.4.5 Barvy

Příklad, jak lze obarvit text v popisku předem definovanými barvami až po vlastní variace barev.

```
var red = new Label {
    Text = "Red", BackgroundColor = Color.Red
};
var orange = new Label {
    Text = "Orange", BackgroundColor = Color.FromHex("FF6A00")
};
var green = new Label {
    Text = "Green", BackgroundColor = Color.FromRgb(38, 127, 0)
};
```

```
;

var transparent = new Label {
    Text = "Transparent", BackgroundColor = Color.Transparent
};
var @default = new Label {
    Text = "Default", BackgroundColor = Color.Default
};
var accent = new Label {
    Text = "Accent", BackgroundColor = Color.Accent
};
```

2.4.6 Tlačítka

```
Button button = new Button{
    Text = "Click Me!",
    Font = Font.SystemFontOfSize(NamedSize.Large),
    BorderWidth = 1,
    HorizontalOptions = LayoutOptions.Center,
    VerticalOptions = LayoutOptions.CenterAndExpand
};
```

2.4.7 Obrázky

Xaml:

```
<Image Source="waterfront.jpg" />
```

C#:

```
var beachImage = new Image { Aspect = Aspect.AspectFit };
beachImage.Source = ImageSource.FromFile("waterfront.jpg");
```

2.5 Databáze

Pokud potřebujeme ukládat lokálně data a vhodným prostředkem se již nejeví ukládání do souboru, přicházejí na scénu lokální databáze, které slouží ke složitější práci se strukturovanými daty. Dobrou zprávou je, že v mobilní aplikaci lze použít SQLite. SQLite je open source jednoduchý databázový server, který je umístěný pouze lokálně. Vytváří lokální databázi s možností běžných operací nad daty. Informace jsou uloženy uvnitř tabulek a datové operace lze provádět pomocí kódu C# a LINQ dotazů. SQLite je dokonalé pro multiplatformní aplikace díky své přenositelnosti. Je dokonalým

pomocníkem pro vytváření mobilních aplikací založených na datech s platformou Xamarin [57].

Implementace je velmi jednoduchá a hlavní jádro SQLite je již součástí iOS a Android, ale ne v systémech Windows. Zde je nutné doinstalovat binární balíček do projektu. Následně je potřeba do všech projektů nainstalovat balíčky pro přístup k datům. Existuje mnoho knihoven, které umožňují práci s databázemi SQLite. Pro práci s multiplatformním systémem Xamarin je potřeba speciální přenosná knihovna SQLite.NET, je to jednoduchá open source knihovna pro aplikace .NET, Mono a Xamarin [51].

Je třeba vytvořit rozhraní pro připojení databáze.

```
public interface IDatabaseConnection
{
    SQLite.SQLiteConnection DbConnection ();
}
```

Příklad pro připojení databáze v Android projektu a na ostatních platformách je hodně podobný.

```
using SQLite;
using LocalDataAccess.Droid;
using System.IO;
[assembly: Xamarin.Forms.Dependency(typeof(DBConn_Android))]
namespace LocalDataAccess.Droid
{
    public class DBConn_Android : IDatabaseConnection
    {
        public SQLiteConnection DbConnection()
        {
            var dbName = "CustomersDb.db3";
            var path = Path.Combine(System.Environment.
                GetFolderPath(System.Environment.
                    SpecialFolder.Personal), dbName);
            return new SQLiteConnection(path);
        }
    }
}
```

Po zprovoznění připojení je možné vytvořit tabulky podle našeho definovaného modelu, pokud již neexistují, a následně zpracovávat data pomocí dotazů.

2.6 Soubory

V Xamarinu má každá platforma vlastní souborový systém. Čtení a zápis do souboru je nejlepší vhodně provést na nativní úrovni pomocí rozhraní nad ním definovaným. Stačí přidat soubor do vestavěných zdrojů přímo v přenositelné části a následně s ním pracovat.

```
var assembly = typeof(LoadResourceText).GetTypeInfo().Assembly;
Stream stream = assembly.GetManifestResourceStream(
    "WorkingWithFiles.PCLTextResource.txt"
);
string text = "";
using (var reader = new System.IO.StreamReader(stream)) {
    text = reader.ReadToEnd();
}
```

Vzhledem k tomu, že Xamarin běží na více platformách, každá s vlastním systémem souborů, tak neexistuje jediný přístup pro načítání a ukládání souborů vytvořených uživatelem. Je nutné, obdobně jako u databází, vytvořit rozhraní pro metody na nativní úrovni. Následně pomocí třídy závislostí, anglicky Dependency, vytvořit metody. [assembly: Dependency(typeof(SaveAndLoad))]

2.7 Mapy

V Xamarinu lze použít rozhraní API [58] pro nativní mapy na každé platformě. Uživatelé snadněji používají mapy a mají z nich větší zážitek. Pro zprovoznění je potřeba provést konfiguraci map a následně je zde používat obdobně jako každý prvek v Xamarinu. Na každé platformě je potřeba zavolat inicializaci pomocí Xamarin.Forms.Forms.Init metod.

Definice pro iOS:

```
Xamarin.FormsMaps.Init();
```

Definice pro Android:

```
Xamarin.FormsMaps.Init(this, bundle);
```

Definice pro Windows:

```
Xamarin.FormsMaps.Init("INSERT_AUTHENTICATION_TOKEN_HERE");
```

Pro platformu iOS již nic dalšího není potřeba, ale pro platformy Android a Windows je potřeba zažádat o API klíč pro mapy. Android platforma používá Google Maps na rozdíl platformy Windows, která má zkomponované Bing Maps komponentu. Je tedy pro každou platformu odlišné zobrazení a lépe čitelné pro koncového uživatele.

2.8 Telerik knihovna pro grafy

Je bohatá, intuitivní a snadno ovladatelná knihovna pro vizualizaci dat, která využívá technologii Xamarin a umožňuje vytvářet nativní aplikace iOS, Android a Windows v C#. RadCartesianChart, jak název naznačuje, využívá karteziánský souřadný systém k vykreslení datových bodů v sérii grafů. Osy X a Y definují, jak jsou vypočteny souřadnice jednotlivých bodů v oblasti výměru. Základní typy jsou mřížky, série a koláčové grafy. Lze vykreslit tedy graf kategorizovaný, numerický, čárový, oblastní, spojené grafy a koláčové.

2.9 Xlabs knihovna

XLabs je projekt s otevřeným zdrojovým kódem knihoven, jehož cílem je poskytnout výkonnou a vícevrstvou sadu služeb a kontrol přizpůsobených práci s formáty Xamarin. Rozšiřuje základní prvky běžného Xamarinu o další ovládací prvky:

- AutoCompleteView (beta)
- BindablePicker (beta)
- Kontrola kalendáře (beta)
- Zaškrtačací políčko (beta)
- DynamicListView (beta)
- RozšířenéContentView (beta)
- ExtendedEntry (beta)
- ExtendedLabel (beta)
- ExtendedScrollView (beta)
- RozšířenéTabbedPage
- ExtendedTextCell (beta)
- ExtendedViewCell (beta)
- HybridWebView (beta)
- GradientContentView (iOS / Android beta)
- GridView (IOS beta)
- ImageButton (beta)

- RadioButton (beta)
- RepeaterView (beta)
- SegmentedControlView (IOS beta)
- Webový obrázek (beta)
- IconButton (IOS beta)
- CircleImage (IOS / Android alfa)
- HyperLinkLabel

2.10 Model MVVM

Při objektivě orientovaném programování často dochází k nutnosti řešit programátorské postupy, které můžeme označit jako best practise, tedy takové postupy, které jsou lety prověřeny a které zajistí jak lepší čitelnost kódu, tak i to, že po nějaké době nebudeme muset svůj kód přepisovat z důvodu chyb v návrhu. Jedná se o návrhové vzory (anglicky design patterns). Existuje velké množství návrhových vzorů a samozřejmě vznikají stále další.

2.10.1 Název návrhového vzoru a jeho klasifikace

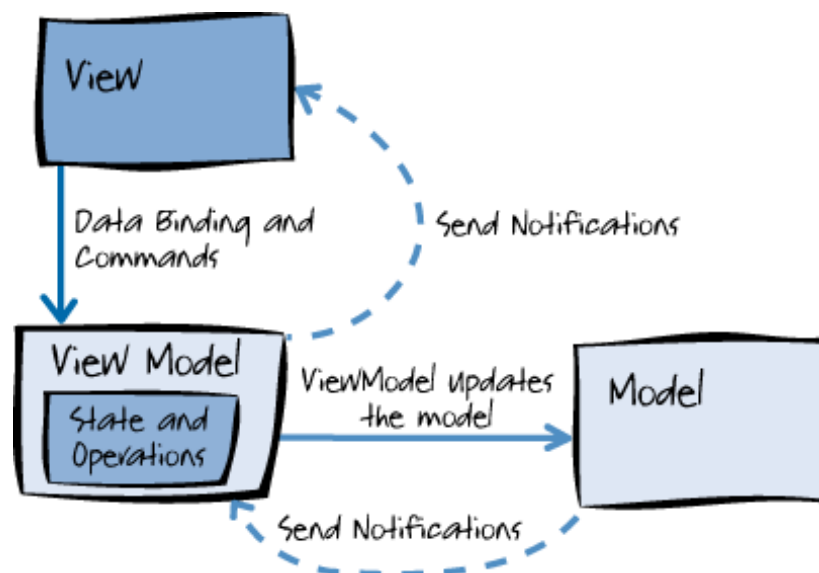
MVVM je zkratkou pro Model-View-ViewModel a jedná se o návrhový vzor, který umožňuje čisté oddělení vrstev mezi uživatelským rozhraním a modelem. Významnou výhodou MVVM je to, že využívá datové vazby k zobrazení informací a reagování na vstup uživatele. Model MVVM se přirozeně přizpůsobuje aplikačním platformám XAML, jako je Silverlight. Důvodem je to, že model MVVM využívá některé z konkrétních možností Silverlight, jako jsou vazby dat, příkazy a chování. MVVM je podobný mnoha jiným vzorům, které oddělují zodpovědnost za vzhled a uspořádání uživatelského rozhraní od odpovědnosti za prezentační logiku, jako například Model-View-Controller (MVC), který je používán pro webové aplikace.

Komponenty jsou vzájemně odděleny, což umožňuje:

- Snadnou náhradu jednotlivých komponent novou implementací.
- Interní implementace může být změněna bez ovlivnění ostatních.
- S komponenty jde pracovat odděleně tedy nezávisle.
- Lze vytvořit izolované testování příklady pro každou komponentu.

2.10.2 Struktura

V modelu MVVM jsou tři základní součásti: model, zobrazení a model zobrazení. Každý z nich má odlišnou a samostatnou roli. Následující obrázek (Obr. 2.5 znázorňuje vztahy mezi těmito třemi součástmi.



Obr. 2.5 Obrázek popisující návrhový vzor Model-View-ViewModel [36].

Kromě porozumění odpovědnosti těchto tří složek je také důležité pochopit, jak se vzájemně ovlivňují komponenty. Na nejvyšší úrovni: *zobrazení* „ví o“ *modelu zobrazení* a *model zobrazení* „ví o“ *modelu*. Na druhou stranu *model* si „neuvědomuje“ *model zobrazení* a *model zobrazení* si „neuvědomuje“ toto *zobrazení*.

Model zobrazení vytváří izolační můstek mezi zobrazením od tříd modelů a dovoluje vývoj modelu nezávisle na zobrazení.

2.10.3 Účastníci

Třídy a objekty, které se podílejí na tomto návrhovém vzoru, jsou:

Zobrazení (View) je zodpovědné za definování struktury, uspořádání a vzhled toho, co uživatel vidí na obrazovce. V ideálním případě je View definováno čistě pomocí XAML s omezeným kódem, který neobsahuje obchodní logiku.

V aplikaci Windows Phone je zobrazení v aplikaci typicky stránka. Navíc by mohlo být View rozděleno na subkomponentu nadřazeného View nebo DataTemplate pro objekt v položce ItemsControl.

View může mít svůj vlastní model zobrazení nebo může dědit model zobrazení rodiče. View získává data z modelu zobrazení pomocí vazeb nebo vyvolá metody v modelu

zobrazení. Během běhu se View změní, když ovládací prvky uživatelského rozhraní reagují na vlastnosti modelu zobrazení, které vyvolávají události oznámení o změně.

Existuje několik možností, jak provést kód v modelu zobrazení v reakci na interakce ve View, například klepnutí na tlačítko nebo výběr položky. Pokud je ovládací prvek příkazový zdroj (anglicky Command Source), vlastnost ovládacího prvku může být datově vázána na vlastnost ICommand v modelu zobrazení. Po vyvolání příkazu ovládacího prvku bude proveden kód v modelu zobrazení. Kromě příkazů může být chování připojeno k objektu ve View a může poslouchat buď příkaz, který má být vyvolán, nebo událost, která má být vyvolána. V reakci může chování vyvolat ICommand na modelu zobrazení nebo metodu na modelu zobrazení.

Model v MVVM je implementace modelu domény aplikace, který zahrnuje datový model společně s business logikou a validací. Příklady modelových objektů zahrnují repozitáře, business objekty, objekty přenosu dat (DTO) a další.

Model zobrazení (ViewModel) funguje jako prostředník mezi zobrazením a modelem a je zodpovědný za zpracování logiky zobrazení. ViewModel typicky interaguje s modelem vyvoláním metod ve třídách modelů. ViewModel potom poskytuje data z modelu ve formě, kterou lze v zobrazení snadno použít. ViewModel načte data z modelu a pak data zpřístupní zobrazení a může data přeformátovat nějakým způsobem, což usnadňuje zobrazení. ViewModel také poskytuje implementace příkazů, které uživatel v aplikaci inicializuje v zobrazení. Když uživatel například klikne na tlačítko v uživatelském rozhraní, může tato akce spustit příkaz ve ViewModel. ViewModel může také odpovídat za definici logických změn stavu, které ovlivňují některé aspekty v zobrazení, například indikací, že některé operace čekají. Aby se ViewModel podílel na obousměrném datovém spojení se zobrazením, musí jeho vlastnosti zvýšit událost PropertyChanged.

2.10.4 Důsledky

MVVM umožňuje skvělý workflow pro vývojáře i návrháře a poskytuje tyto výhody:

- Během procesu vývoje mohou vývojáři a projektanti pracovat nezávisleji a souběžně na jejich součástech. Návrháři se mohou soustředit na zobrazení a mohou snadno generovat vzorová data pro práci, zatímco vývojáři mohou pracovat na modelu zobrazení a modelových součástech.
- Vývojáři mohou vytvořit jednotkové testy modelu zobrazení a modelu bez zobrazení. Jednotkové testy modelu zobrazení mohou vykonávat přesně stejnou funkcionální, jakou používá zobrazení.
- Je snadné přepracovat uživatelské rozhraní aplikace bez dotyku kódu, protože

zobrazení je plně implementováno v XAML. Nová verze zobrazení by měla pracovat s existujícím modelem zobrazení.

- Existuje-li stávající implementace modelu, který zapouzdřuje existující business logiku, může to být obtížné nebo riskantní provést celkovou změnu. V tomto scénáři funguje model zobrazení jako adaptér pro třídy modelů a umožňuje vyhnout se provádění zásadních změnám v modelovaném kódu.

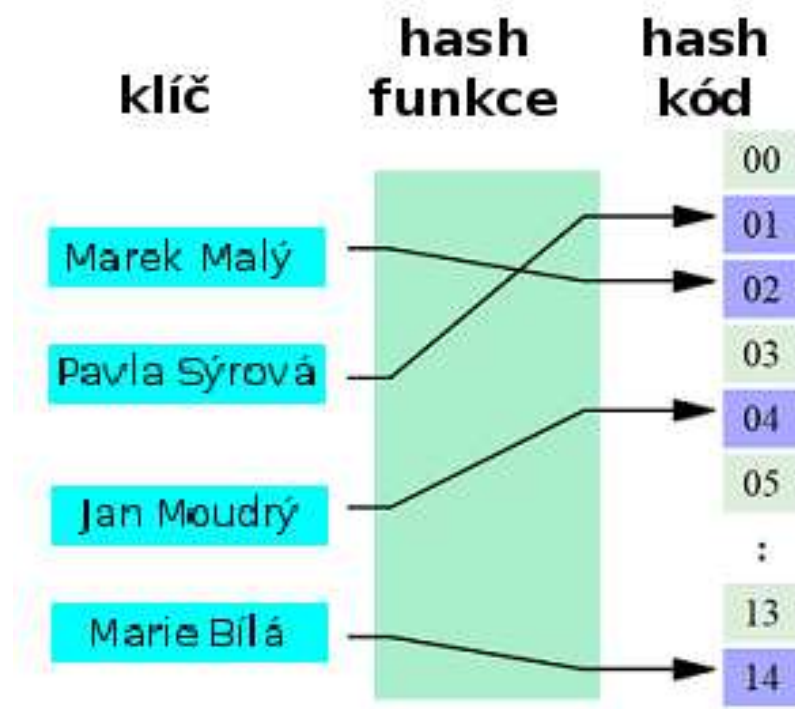
3 HASHOVACÍ FUNKCE

Hashovací funkce anglicky zvaná hash function je libovolná funkce, která může být použita pro mapování dat libovolné velikosti na data s pevnou velikostí. Z matematického pohledu se jedná o jednocestnou matematickou funkci, kterou lze použít na vstupní data o jakékoliv délce, a vytvářet takzvaný hash kód o dané bitové délce hashovací funkce.

Definice 3.1 (JF) Funkce $f()$ se nazývá *jednocestnou funkcí* (JF), jestliže splňuje následující vlastnost:

- Pro daný vstup x a danou funkci x je snadné vypočítat $f(x)$, ale pro danou $f(x)$ je nesnadné vypočítat x .

Vytváří relativně malé číslo oproti vstupním datům. Tento princip lze demonstrovat na obrázku 3.1. Tento jednoduchý příklad ukazuje, jak lze text převést algoritmem hashovací funkce na hash kód.



Obr. 3.1 Obrázek znázorňující průběh hashovací funkce [53].

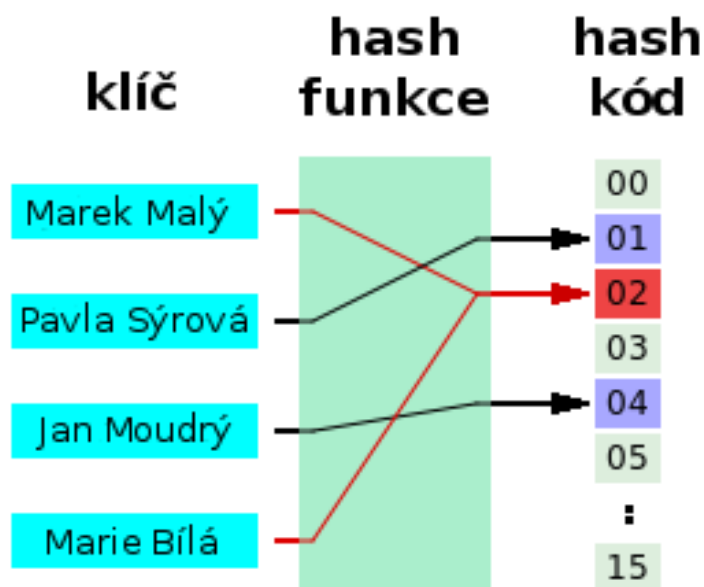
Vlastnosti Hlavní vlastností jakékoliv hashovací funkce je, že při drobné změně vstupních dat dochází k rapidní změně výstupního hash kódu. Vše se provádí na základě matematického výpočtu. Hashovací funkce není nic jiného než matematický vzorec. Ze zahashovaného hash kódu již nelze jednoduchým způsobem zrekonstruovat vstupní data, neboť matematický výpočet je jednocestný.

Význam Hash funkce urychlují vyhledávání dat v tabulce nebo se využívají k redukci dat v databázích při duplicitních datech. Jsou také užitečné v kryptografii, kdy kryptografická hashovací funkce umožňuje snadno ověřit, zda některá vstupní data mají shodnou hodnotu hash. Opakem bez znalosti vstupních dat nelze rekonstruovat jednoduchým způsobem se znalostí uložených hodnot hash opět vstupní data. Toto slouží k zajištění integrity přenášených dat a je základním prvkem pro HMAC, které poskytují autentizaci zpráv.

Definice 3.2 (JFK) Jednocestná funkce $f()$ se nazývá *jednocestnou funkcí s klíčem* (JFK), jestliže navíc splňuje následující vlastnost:

- Pro danou $f_K(x)$ lze snadno spočítat x tehdy, když je známa hodnota tajemství K .

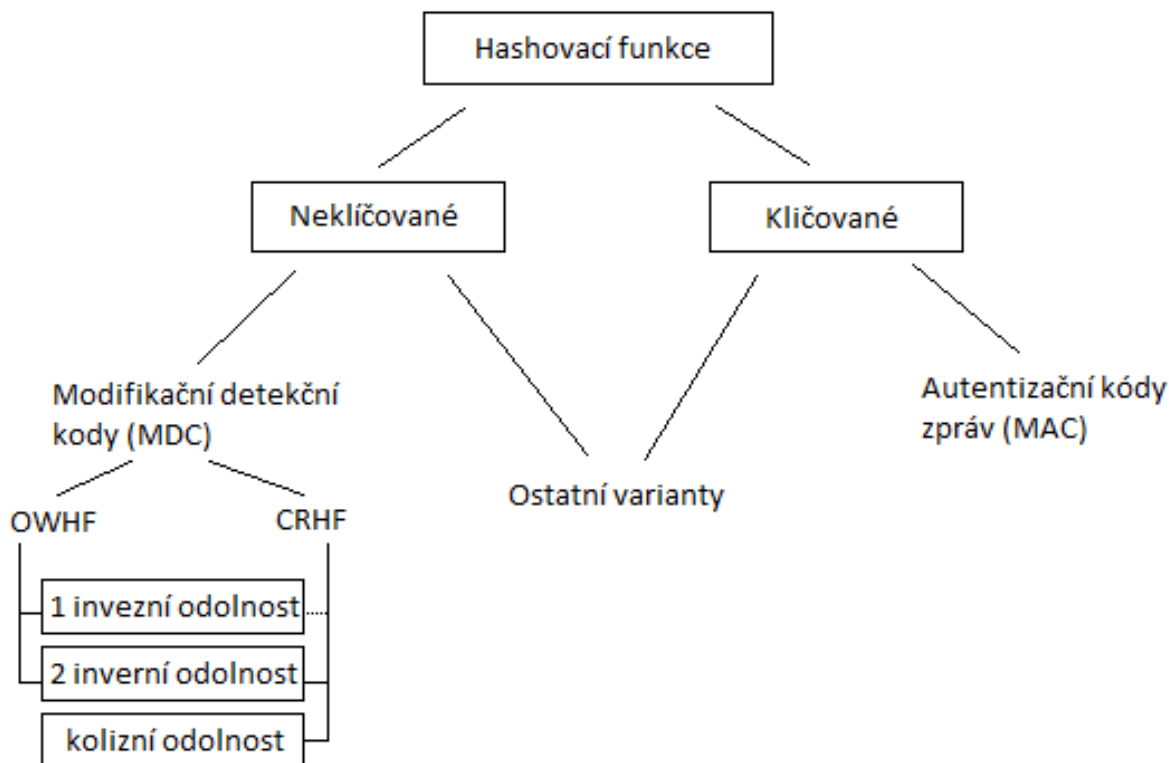
Takže při zahashování x_1 dostaneme $f(x)$ a je velmi nepravděpodobné, že při zahashování x_2 dostaneme opět $f(x)$. Pokud by tohle nastalo, říkáme, že nastala kolize. Ke kolizi dochází u hashovací funkce, když k různým x dostaneme stejné $f(x)$. Tento problém lze zřetelně vidět na obrázku 3.2.



Obr. 3.2 Obrázek znázorňující kolizi v hashovací funkci [53].

Hashovací funkce lze rozdělit do dvou skupin: neklíčované, kam patří definice JF, a klíčované definice JFK. Obě varianty se dají použít v kryptografii, ale vždy jiným směrem uplatnění. Neklíčované hashovací funkce patří do skupiny Manipulační detekční kódy, anglicky označované Manipulation Detection Codes (MDCs), se používají k zajištění integrity dat, zda data jsou platná a nepozměněná. Klíčované hashovací funkce

patřící do skupiny Autentizační kódy zpráv, anglicky označované jako Message Authentication Codes (MACs), slouží pro vytvoření a přidání elektronického podpisu ke zprávě, takže zajištění autenticity. Bližší informace najdete v mé předchozí práci [53].



Obr. 3.3 Obrázek popisující dělení hashovacích funkcí a závislostní mezi nimi [53].

Kvalitní neklíčovaná hashovací funkce má tři podmínky:

Podmínka 3.1 (preimage resistance) Odolnost proti nalezení vzoru, anglicky "preimage resistance":

- Pro dané y je výpočetně nezávládnutelné nalézt takové x , aby platilo $f(x) = y$.

Podmínka 3.2 (2nd preimage resistance) Odolnost proti nalezení jiného vzoru, anglicky "2nd preimage resistance":

- Pro dané x je výpočetně nezávládnutelné nalézt takové $x' \neq x$, aby platilo $f(x') = f(x)$.

Podmínka 3.3 (collision-resistant) Odolnost proti kolizím, anglicky "collision-resistant":

- Je výpočetně nezávládnutelné nalézt takové x a zároveň x' pro $x' \neq x$, aby platilo $f(x') = f(x)$.

3.1 Hashovací funkce v Bitcoinech

Pro Bitcoinů hrají různé hashovací funkce zásadní roli v zabezpečení transakcí, a tedy přenosu informací mezi jednotlivými účastníky v rámci systému. Používají se rodiny

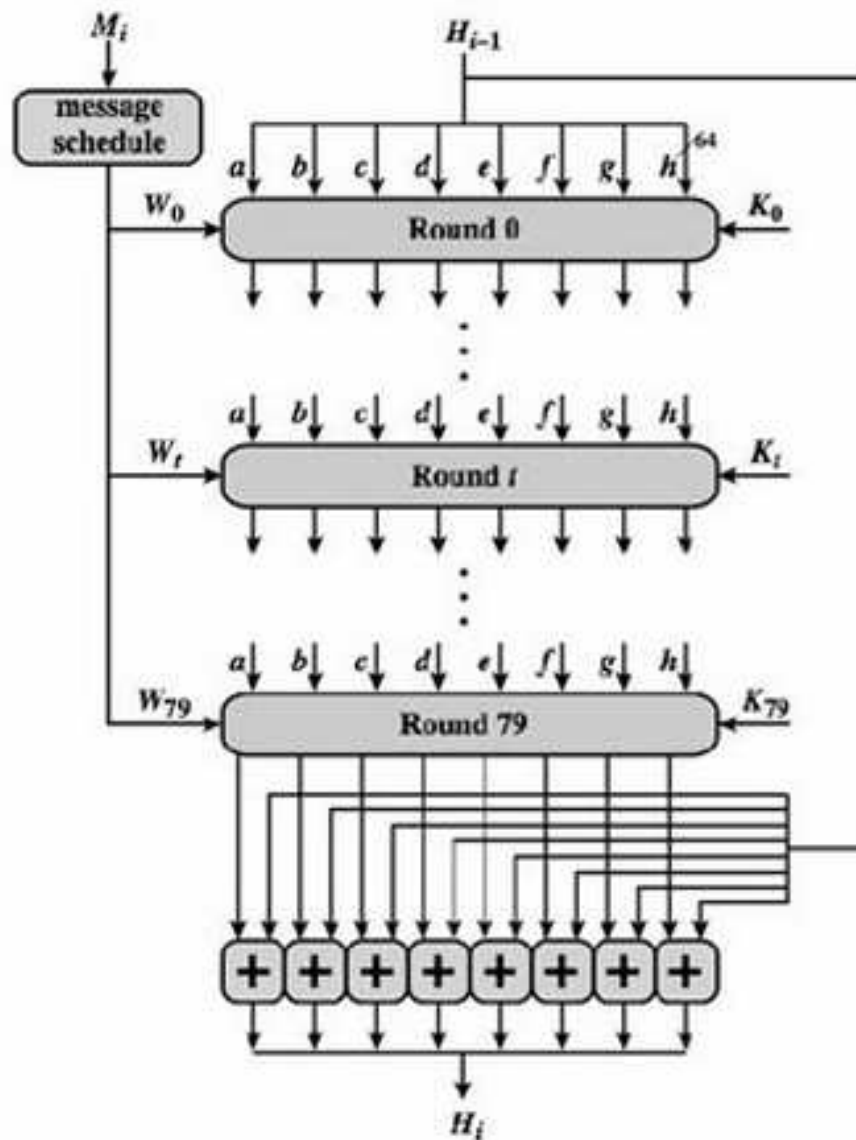
hashovacích funkcí SHA, MD a RIPEMD, ale nikdy v obyčejné variantě vždy se jedná o kombinaci algoritmů včetně kombinace s eliptickými křivkami. Zde zmíním pouze dvě variace; bližší informace najdete v mé předchozí práci [53].

3.1.1 Funkce SHA (Secure Hash Algorithm)

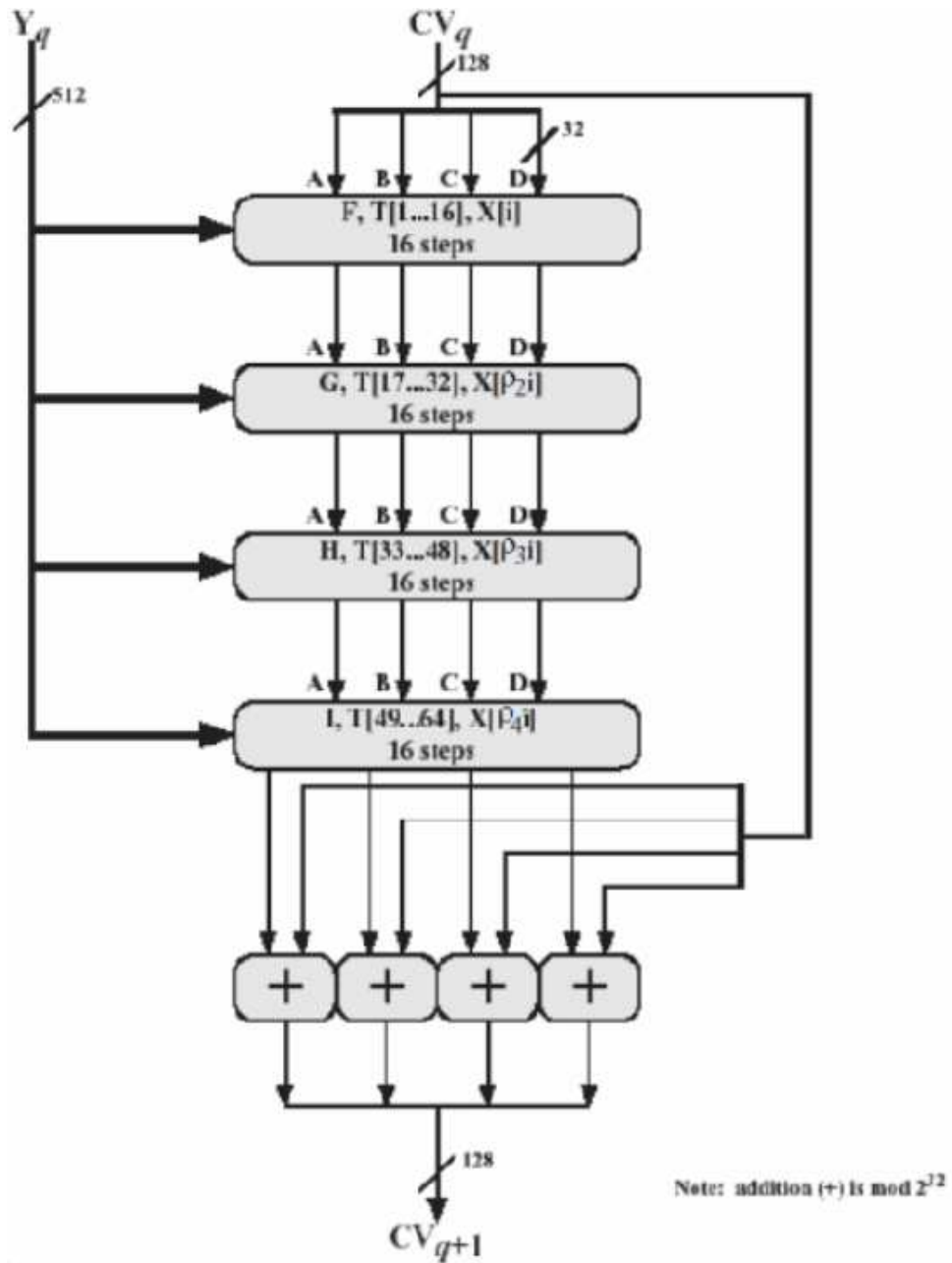
SHA-1 se vyvinula z původního standardu z roku 1993 vydaného společností NIST (Národním institutem pro standardy v USA), označovaného SHA-0. SHA-1 je odlišena od SHA-0 jednou bitovou rotací pomocí jednocestné funkce. Obtížnost se zakládá na nezměrném počtu možných řešení, které je potřeba ověřit k potvrzení výsledku. SHA je zkratka pro Secure Hash Algorithm. SHA je rodina čtyř hashovacích funkcí, které jsou algoritmicky strukturovány odlišně. V roce 2005 byl nalezen útok na SHA-1, který naznačuje, že algoritmus nemusí být dostatečně bezpečný pro stávající použití [56, 32]. Z těchto důvodů se hashovací funkce SHA-1 označila za nedostatečnou a vytvořila se nová verze SHA-2; tato hashovací funkce obsahuje hned 4 variace s odlišnou bitovou délkou. Jedná se o hashovací funkce SHA-224, SHA-256, SHA-384 a SHA-512. Bohužel v nynější době již ani tato varianta nestačí, a proto se v roce 2009 začalo pracovat na variantě SHA-3, původně označené jako Keccak, a roce 2012 se doporučoval přechod na tuto variantu. V roce 2015 se SHA-3 stala standardem a obsahuje tyto funkce: SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128 a SHAKE256, viz zdroj [53].

3.1.2 Funkce RIPEMD (RACE Integrity Primitives Evaluation Message Digest)

RIPEMD-160 je 160bitový hashovací algoritmus a patří mezi kryptografické hash funkce. Byl vyvinut ve spolupráci Hanse Dobbertina, Antoona Bosselaerse a Barta Preneela ve výzkumné skupině COSIC na Katolické univerzitě v belgické Lovani. Algoritmus byl poprvé publikován v roce 1996. Následující verze RIPEMD byla vylepšena na základě konstrukčních principů používaných v MD4 a její výkon je hodně podobný výkonu SHA-1. V základní verzi RIPEMD byla nalezena chyba, a proto byl rozšířen o 128-, 160-, 256- a 320bitové verze. RIPEMD je otevřen komunitě, je tedy jinou variantou než SHA-1 a SHA-2.



Obr. 3.4 Obrázek popisuje zpracování jednoho bitového bloku pro SHA-2 s provedením 80 kol [53].



Obr. 3.5 Obrázek popisuje zpracování jednoho 512bitového bloku na 64 kol, kdy používá čtyři základní funkce pro hashování, označované jako F, G, H, I [53].

4 ASYMETRICKÁ KRYPTOGRAFIE

4.1 Algoritmus RSA

Jedná se asymetrický algoritmus založený na faktorizaci čísel. Pojmenování získal podle autorů R. Rivesta, A. Shamira a L. Adlemana. Patří mezi první algoritmy s veřejným klíčem. Algoritmus se používá pro šifrování dat od subjektu A k subjektu B , kdy každý subjekt má veřejný a soukromý klíč.

Popis systému je následující, konstanta n je veřejný modulus, e je veřejný exponent (obvykle 3 nebo $2^{16} + 1$), d je soukromý exponent a p, q činitelé faktoru modulu, typicky prvočísla. Veřejný klíč tvoří dvojice (n, e) , obdobně soukromý klíč je dvojice (n, d) .

Postup pro generování klíčů:

Vstup Generátor náhodných čísel, který vrací pouze prvočísla.

Vystup Soukromý klíč d .

Procedura je následující:

1. Vygenerujeme prvočísla p, q , kde platí $n = p \cdot q$.
2. Vypočteme $\phi_n = (p - 1)(q - 1)$.
3. Zvolíme hodnotu $e < \phi_n$ takovou, že $\gcd(\phi_n, e) = 1$.
4. Vypočteme $d = e^{-1} \bmod \phi_n$.

Generátor náhodných čísel pro prvočísla musí být opravdu náhodný. Získaná prvočísla p, q by měla být dostatečně silná a neměla by se dát odhadnout. Šifrování se pak provádí ze zprávy m a vypočítává se šifra c

$$c = m^e \bmod n \quad (4.1)$$

a výpočet pro dešifrování

$$m = c^d \bmod n. \quad (4.2)$$

Aby byla zajištěna náležitá bezpečnost, algoritmus pracuje s čísly o bitové délce 1024, 2048 a vyššími. S navýšenou délkou bitového čísla vrůstá i doba šifrování a dešifrování. Pro operace s velkými čísly se využívají metody binárního umocňování zprava doleva nebo binárního umocňování zleva doprava, které lze implementovat i hardwarově.

4.2 Diffie-Helman (DH)

Algoritmus je určen pro výměnu klíčů přes nezabezpečený kanál díky asymetrické kryptografii a nikdy se nepoužívá samostatně, vždy se kombinuje s RSA nebo eliptickými křivkami. Slouží pro ustanovení symetrického klíče mezi komunikujícími stranami. Tento algoritmus je bohužel bezbranný vůči útoku Man in the middle, kdy útočník odposlouchává komunikaci mezi účastníky.

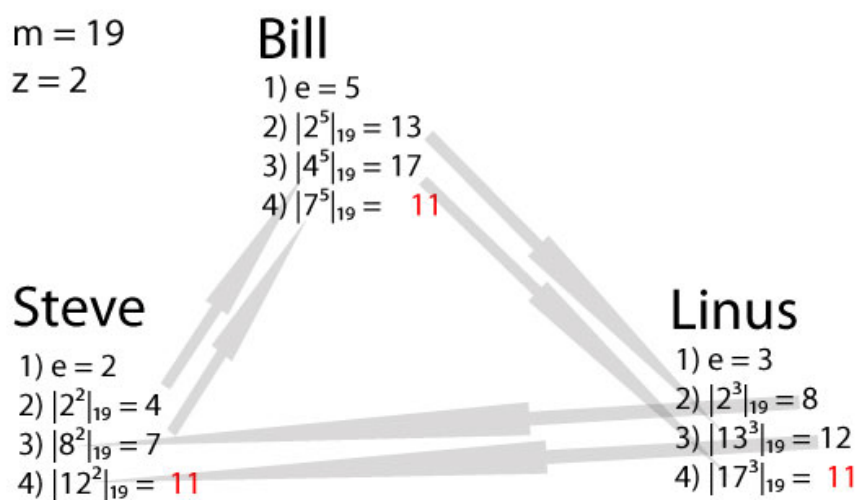
Algoritmus je založen na umocňování čísel,

$$(A^B)^C = (A^C)^B \quad (4.3)$$

respektive na modulární variantě tohoto vzorce

$$|(A^B)^C|_m = |(A^C)^B|_m. \quad (4.4)$$

Nástin komunikace:



Obr. 4.1 Obrázek popisuje základní komunikaci pro ustanovení klíčů založenou na Diffie-Hellman algoritmu [1].

4.3 Eliptické křivky

Eliptické křivky se využívají v kryptografii pro metody šifrování s veřejným klíčem. Kryptografie s veřejnými klíči je založena na neznalosti rychlého, matematicky popsaného algoritmu pro zvolenou eliptickou křivku. Pro zabezpečenou komunikaci dvou subjektů A a B s veřejným klíčem se používá dvojice klíčů, veřejný a soukromý klíč, včetně sady operací pro práci spjaté s klíči. Následně eliptické křivky představují vhodnou složitost výpočtu založenou na nesnadném nalezení diskrétního logaritmu, anglicky Discrete logarithm problem. Vše jen se základní znalostí počátečního bodu o dané křivce. Matematicky lze obecně popsat eliptickou křivku následující rovnicí:

$$y^2 = x^3 + a \cdot x + b. \quad (4.5)$$

a, b představují reálné konstanty, následkem jejich změny je odlišná eliptická křivka. Eliptické křivce odpovídají body x, y včetně definičního oboru směřujícího k ∞ . Veřejný klíč je bod Q na křivce a soukromý klíč je náhodné číslo k . Výpočet bodu lze tedy vyjádřit Q :

$$Q = k \cdot P = \left(\underbrace{P_1 + \dots + P_n}_{k \cdot P} \right), n \in N. \quad (4.6)$$

Základní operace s eliptickými křivkami jsou sčítání

$$P + Q = R, P \neq Q \quad (4.7)$$

a zdvojené násobení

$$2 \cdot P = P + P = R. \quad (4.8)$$

Obtížnost výpočtu je přímo úměrná velikosti eliptické křivky. To znamená, že ne každou eliptickou křivku lze využít v kryptografii, protože k některé eliptické křivce lze snadněji vypočítat výsledek. Z těchto důvodů má jasný předpoklad, že výpočetní složitost je odlišná, ne-li vyšší než ostatní algoritmy pro asymetrickou kryptografii. Složitost díky studii převyšuje algoritmus RSA [25]. Nejběžnější využití eliptických křivek se používá tam, kde je potřeba ušetřit výpočetní výkon, paměť a zvýšit propustnost. Asymetrických algoritmů s eliptickými křivkami máme hned několik. Každý z nich se používá k odlišným aplikacím.

Výčet základních eliptických křivek je následující:

- **ECDH** anglicky Elliptic curve Diffie-Hellman, kdy se tento algoritmus používá pro ustanovení klíčů pro následnou zabezpečenou symetrickou komunikaci [31],
- **ECIES** anglicky Elliptic curve Integrated Encryption Scheme [35],
- **ECMQV** anglicky Elliptic curve MQV [34],
- **ECDSA** anglicky Elliptic curve Digital Signature Algorithm [17, 30].

5 PROBLEMATIKA BITCOINŮ

5.1 Princip a smysl Bitcoinů

Jedná se o mezinárodní digitální měnový systém včetně šířitelného softwaru, anglicky označovaný OpenSource software, distribuovaný v rámci peer-to-peer sítě. Bitcoinů nejsou závislé na bankovním systému, korporacích či státech. Systém funguje jako jedna decentralizovaná distribuční síť, kterou nikdo centrálně neřídí. Pro provádění transakcí se využívá distribuovaná databáze napříč jednotlivými uzly peer-to-peer sítě [45]. Integrita sítě je zajišťována takzvanými horníky či těžaři, anglicky miners. Horník ověřuje jednotlivé transakce a za tyto služby získává virtuální peníze. K prvkům zabezpečení se používá kryptografie, která zajišťuje ověřování transakcí a důvěryhodnost celé sítě. Horníci potvrzují jednotlivé transakce, a zabraňují tak opětovnému použití utracených peněz v systému [8, 12]. Na fungování celého systému se podílejí dva druhy uživatelů. Jedním jsou horníci, kteří potvrzují zmíněné transakce, které můžeme označit jako bankovní převody. Druhou skupinu tvoří koncoví uživatelé, kteří zajišťují pohyby v systému pomocí směnárny nebo libovolně podle dohody provedou převod na jiný bitcoinový účet. Bližší informace naleznete v [53]. Systém virtuálních peněz řeší požadavky reálného světa moderních technologií.

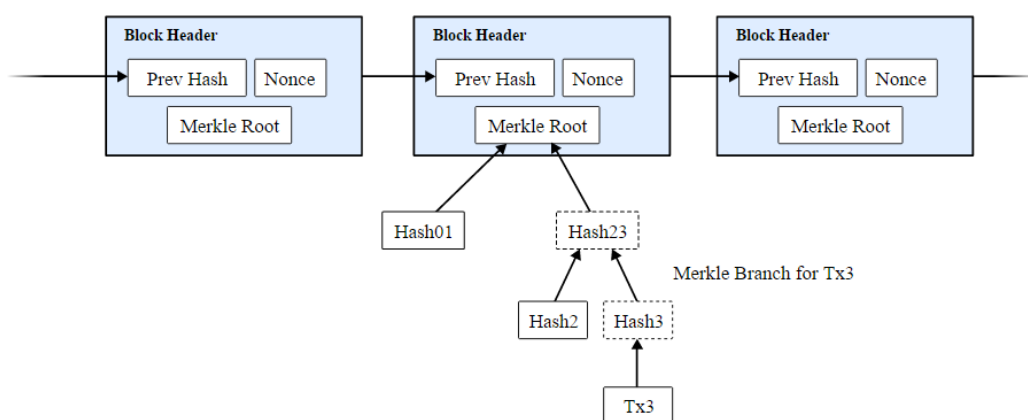
- Zajišťuje důvěryhodnost bez vyšších autorit, jako jsou banky či stát.
- Pro distribuci peněz v systému se používá silná kryptografie.
- Ve většině případů jsou transakce zpoplatněné, ale nemusí tomu tak být všude. Zpoplatněné transakce jsou dříve zpracovány.
- Platí zde nemožnost odvolatelnosti transakcí, která byla již zaúčtována. Případně se musí provést nový převod zpět.
- Integritu systému tvoří samotní horníci, kteří vlastní nadpoloviční většinu výpočetního výkonu v systému.
- Deflace měny hned od začátku založení je trvalá, nikdy nebude více měny než 21 mil. Bitcoinů (resp. 20 999 999,9769).
- Anonymita uživatelů v systému. Účet si může založit jakýkoliv uživatel bez nutnosti verifikace své osoby.

5.2 Popis struktury

5.2.1 Block

Data v Bitcoin síti jsou trvale zaznamenána prostřednictvím souborů, které komunita nazvala bloky. Blok je záznam o některých nebo o všech nedávných transakcích Bitcoin

systemu, které ještě nebyly dosud zpracovány v žádném předchozím bloku. Nelze říci o všech, protože ostatní transakce buď jsou, nebo budou obsaženy v odlišném bloku. Pokud je blok vytvořen, systém vytvoří záznam a nelze jej odvolat nebo měnit [9, 11]. Všechny bloky, co byly nebo budou vytvořeny, tvoří řetězec bloků, anglicky Blockchain. Existuje i kaskádový online prohlížeč jednotlivých bloků, anglicky Block Explorer, [16, 9, 10]. Řetězec bloků je tvořen seznamem na sebe navazujících bloků s odkazy na další blok, a to ve formě URL.



Obr. 5.1 Obrázek popisuje strukturu bloku v Bitcoinu [46].

5.2.2 Transakce

Transakce vytváří požadavek na zpracování v systému a jedná se tedy o podepsaná data soukromým klíčem vlastníka, který žádá o zpracování. Jak bylo zmíněno, transakce vytvářejí seznamy označované jako blok. U transakcí také vzniká přímé propojení na další transakci v řadě.

U vstupní transakce je potřeba znát odkaz na předchozí transakci "hash_reference_transaction", index transakce a hodnotu "scriptSig". Hodnota "scriptSig" se skládá z podpisu a veřejného klíče.

```

Příklad vstupu transakce
Previous tx: "hash_reference_transaction"
Index: "0"
scriptSig: "signature"public_key"

```

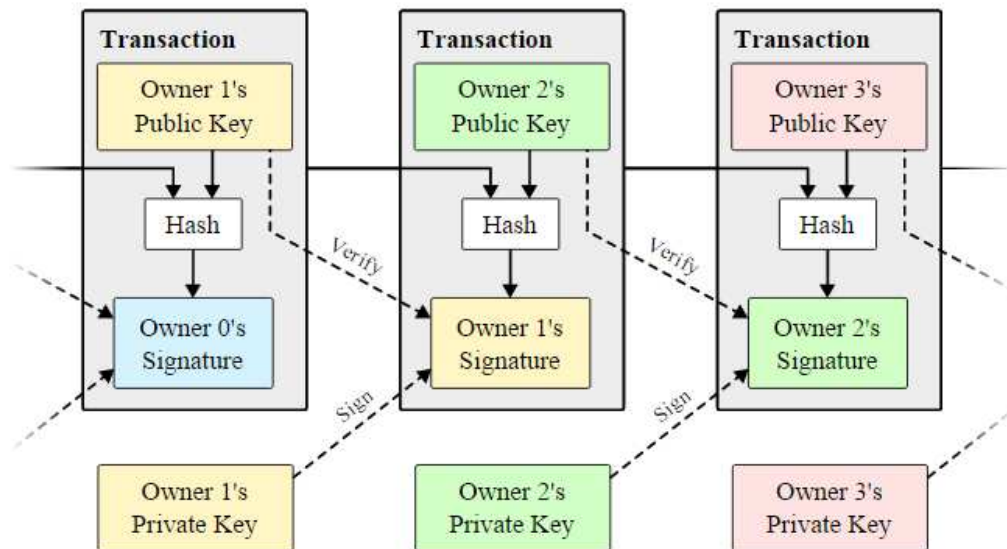
Na výstupu transakce je částka, která se má doručit příjemci, a zbytek zpět do účtu vystavovatele, a to vždy podle veřejného klíče transakce. Hodnota scriptPubKey označuje seznam pokynů pro práci s transakcí.

Příklad výstupu transakce s částkou 50 BTC, kde osm zbylých nul definuje drobné hodnoty měny známé jako satoshi jednotky:

Value: 5000000000

scriptPubKey: OP_DUP OP_HASH160

404371705fa9bd789a2fcd52d2c580b65d35549d OP_EQUALVERIFY OP_CHECKSIG



Obr. 5.2 Obrázek popisuje strukturu transakce v Bitcoině [46].

5.2.3 Horník

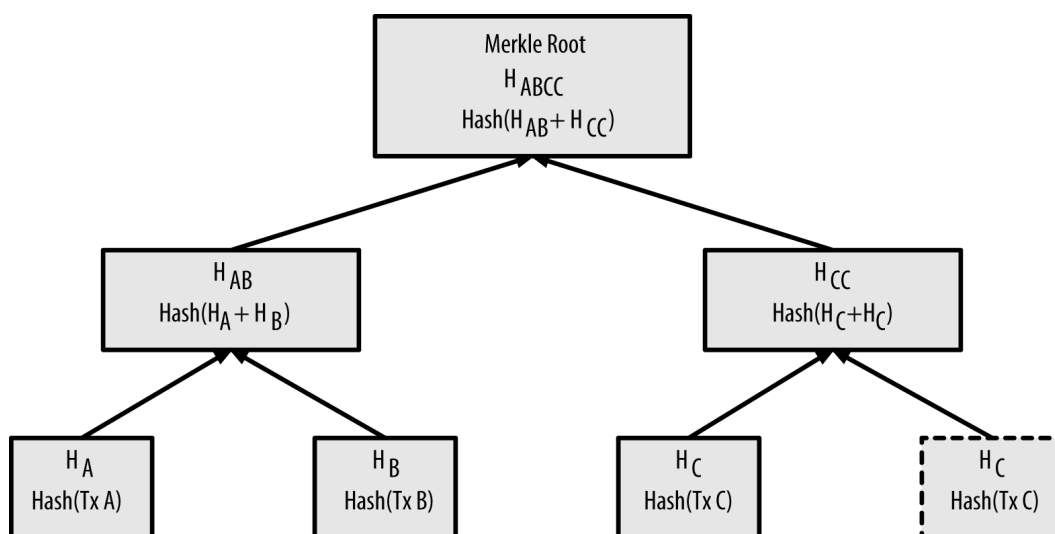
Takzvaná těžba Bitcoinů je prováděna pomocí různých uživatelů na celém světě. Uživatelé jsou označováni jako horníci, anglicky Miners. Horníci zajišťují ověřování neboli potvrzení transakcí v síti, a zaručí tak integritu celé sítě. Integrita neboli důvěryhodnost se skládá z celé výpočetní síly všech horníků. Nikdy by se nemělo stát, že člověk nebo komunita vlastní větší výpočetní sílu všech horníků na síti (tedy 51% a horníci 49%). Vznikl by tak negativní vliv na potvrzování transakcí a dalo by se s nimi manipulovat. Horníci za poskytování služby ověřování transakcí dostávají od Bitcoin systému odměny ve formě Bitcoin částek, nyní 25\$.

Horník, aby mohl obdržet odměnu, hledá v síti nepotvrzené transakce od uživatelů, provádí nad nimi dvojité hashování s SHA256 a snaží se sestavit jeden blok. Každý blok obsahuje Merkle kořen, který pod sebou má celý Merkle strom. Pro upřesnění se hashuje hlavička bloku neboli Blockheader.

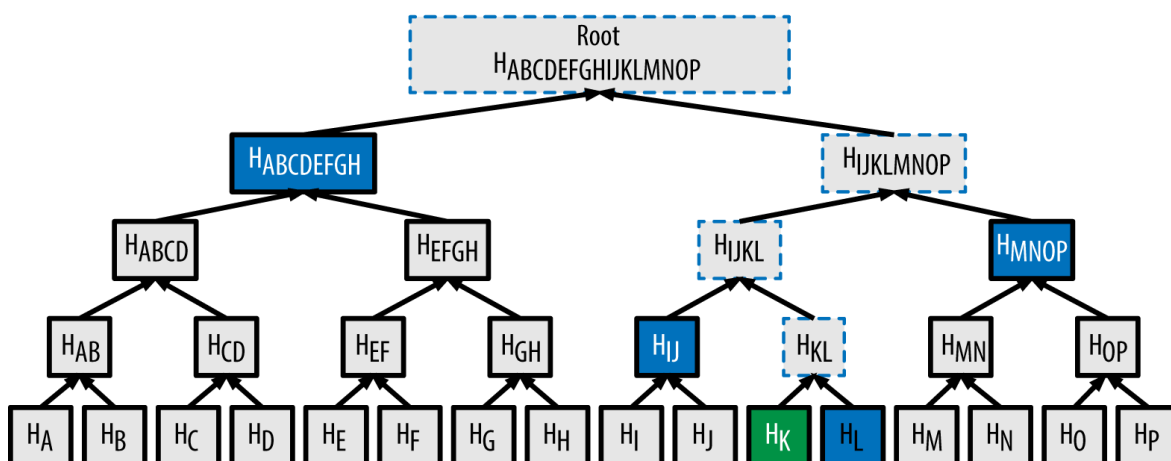
Příklad stromu Melker: $\text{hash} = \text{SHA256}(\text{SHA256}(\text{TransactionID}))$

Příklad kořene: $\text{hash} = \text{SHA256}(\text{SHA256}(\text{Blockheader}))$

Hash musí být menší nebo roven aktuálnímu cíli pro blok, aby byl přijat. Čím je nižší



Obr. 5.3 Obrázek popisuje strukturu Merkle stromu v Bitcoině [46].



Obr. 5.4 Obrázek popisuje strukturu kořene vytvářeného z Merkle stromu v Bitcoině [46].

cíl, tím obtížnější je vytvořit platný blok. Díky obtížnosti je zajištěno, že horníci s nižší výpočetní silou mají stejnou obtížnost nalezení řešení. V podstatě se jedná o loterii, kdy cílem je sestavit blok a získat odměnu několika Bitcoinů. Na síti se může stát, že pro jeden blok budou existovat dvě různá řešení. Které je tedy to správné řešení a mohou být správná obě? Síť je navržena tak, aby tento rozkol vyřešila v krátké době, jednu větev řešení zamítla a dál se v ní nepokračovalo. Výběr větve se provádí na základě délky všech řešení v dané větvi.

6 API PRO MOBILNÍ KOMUNIKACI

6.1 Popis aplikačního rozhraní pro Blockchain.info

Blockchain je přední světová softwarová platforma pro digitální aktiva. Nabízí největší výrobní blokovou platformu na světě a používá novou technologii k vybudování radikálně lepšího finančního systému. Jejich software podporuje více než 100 milionů transakcí a umožňuje uživatelům ve 140 zemích po celém světě obchodovat rychle a bez nákladných zprostředkovatelů. Nabízí také nástroje pro vývojáře a data transakcí v reálném čase, aby uživatelé mohli analyzovat rozvíjející se digitální ekonomiku. Blockchain prošel změnou aplikačního rozhraní z verze 1 na verzi 2. Původně otevřené aplikační rozhraní s komunikací přes HTTP se stalo bezpečnějším díky nové verzi a zabezpečení pomocí TLS s HTTPS.

Blockchain wallet API [15] a [13] je základní webový protokol pro získání informací o účtu, zaslání platby a pro případné archivování účtu. Přistupuje se přes adresu <https://blockchain.info/merchant/> a za ní bezprostředně následuje proměnná *\$guid/*, která představuje uživatelské ID vygenerované na serveru při zakládání účtu. K odkazu přidáme metodu, aby server věděl, co má vrátit. Metoda **payment?** očekává tyto parametry, kde každý je oddělený symbolem **&**:

- **password=\$main_password** – Je hlavní heslo do účtu.
- **second_password=\$second_password** – Představuje druhé heslo pro dvojitě šifrování.
- **to=\$addres** – Je adresa příjemce.
- **amount=\$amount** – Je posílanou hodnotou na 8 míst bez oddělovače celých čísel.
- **from=\$from** – Vyjadřuje adresu odesílatele (nepovinná).
- **fee=\$fee** – Tvoří poplatek za transakci. Server vyžaduje minimální hodnotu poplatku, a to 0.0001. Hodnotu lze pouze zvýšit (nepovinné).

Odpověď vypadá následovně, jestliže se platba zdaří:

```
{
  "message" : "zprava s odpovedi",
  "tx_hash" : "transakcni ID",
  "notice" : "pridana poznamka"
}
```

Metoda **balance?** očekává parametr:

- password=\$main_password – Hlavní heslo do účtu.

Odpověď zní následovně:

```
{ "balance ": 1000 }
```

Metoda **list?** očekává parametr:

- password=\$main_password – Hlavní heslo do účtu.

Odpověď, která vrací pole všech účtů v peněžence, vypadá následovně:

```
{
  "addresses ": [
    {
      "balance ": 1400938800,
      "address ": "1Q1AtvCyKhtveGm3187mgNRh5YcukUWjQC",
      "label ": "SMS Deposits",
      "total_received ": 5954572400
    },
    {
      "balance ": 79434360,
      "address ": "1A8JiWcwvpY7tAopUkSnGuEYHmzGYfZPiq",
      "label ": "My Wallet",
      "total_received ": 453300048335
    }
  ]
}
```

Metoda **new_address?** očekává několik parametrů, které jsou opět oddělené &:

- password=\$main_password – Hlavní heslo do účtu.
- second_password=\$second_password – Představuje druhé heslo pro dvojitě šifrování.
- label=\$label – Volitelný štítek.

Odpověď zní:

```
{
  "address " : "18fyqiZzndTxdVo7g9ouRogB4uFj86JJiy",
  "label ": "Main_account"
}
```

Metoda `api/v2/create` pro vytvoření nové peněženky očekává několik parametrů, které jsou opět oddělené `&`:

- `password=$main_password` – Hlavní heslo do účtu.
- `api_code=$api_code` – GUID pro vytváření nových účtů a musí se o něj požádat pro aplikaci.
- `priv=$priv` – Volitelný privátní klíč přidáný k peněžence.
- `label=$label` – Volitelný štítek.
- `label=$label` – Potřebné pole pro následnou validaci nové peněženky.

Odpověď zní:

```
{
  "guid": "4b8cd8e9-9480-44cc-b7f2-527e98ee3287",
  "address": "12AaMuRnzw6vW6s2KPRAGeX53meTf8JbZS",
  "label": "Main address"
}
```

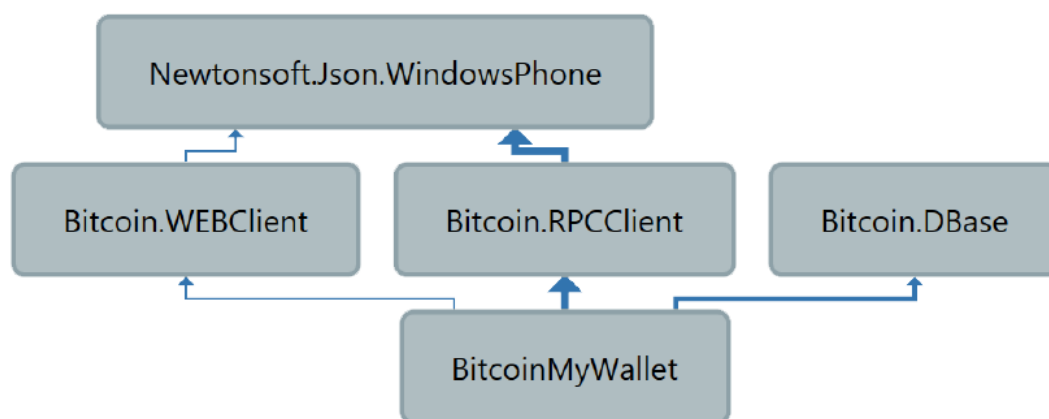
Zde je popsána pouze část vyzkoušených metod z tohoto rozhraní. Součástí rozhraní je několik metod. Pro podrobnější pochopení doporučuji čtenáři přímo navštívit webové stránky API rozhraní zde [13].

II. PROJEKTOVÁ ČÁST

7 POPIS PROTOTYPOVÉ APLIKACE

7.1 Rozdělení projektu

Celá aplikace je rozdělena do čtyř samostatných projektů, které zajišťují přehlednost celé aplikace. Obrázek 7.1 zobrazuje závislosti jednotlivých projektů mezi sebou. Do řešení bylo nutné začlenit i tehdy nejnovější verzi knihovny *Newtonsoft.Json* pro Windows Phone na projekt *RcpClient*. Bylo to z důvodu chybějících částí, které byly dříve dostupné pro testovací projekt *RrcClient* Windows konzole.



Obr. 7.1 Obrázek popisuje strukturu prototypové aplikace Bitcoin Wallet na Windows Phone [53].

Rozdělení projektů:

WebClient – projekt pro webové API rozhraní.

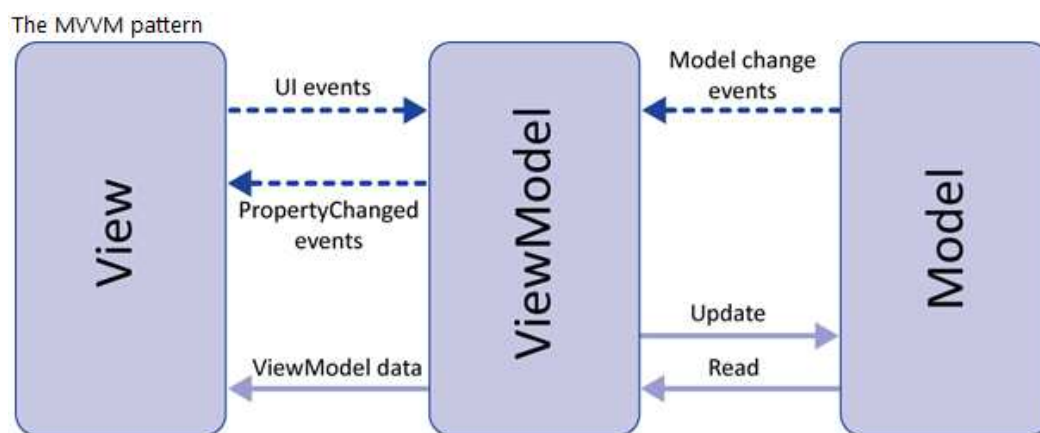
RpcClient – projekt založený na rozhraní Json-RPC funkcí.

DBase – databázový projekt.

Bitcoin My Wallet – uživatelské rozhraní.

7.2 Model MVVM

Aplikace byla vyvíjena v návrhovém vzoru Model-View-ViewModel (MVVM) [36], který představuje snadnější rozšiřitelnost a přehlednost. Projekty *WebClient*, *RpcClient* a *DBase* tvoří modelovou část modelu MVVM. Uživatelské rozhraní představuje část návrhového vzoru zobrazení a model-zobrazení, kde jsou oba celky rozděleny na samostatné adresáře. Pro tvorbu aplikace byly použity knihovny Telerik na grafy a zobrazení, knihovna ZXing pro QR kódér a dekodér, knihovna Windows.Phone.Toolkit pro uživatelské rozhraní, Catel.MVVM šablona pro návrhový vzor a Newtonsoft.Json knihovna pro parsování a sestavování JSONObjectu.



Obr. 7.2 Obrázek popisuje strukturu použitý návrhový vzor MVVM v prototypové aplikaci Bitcoin Wallet [53].

7.3 Projekty

7.3.1 Projekt WebClient

Projektová část řeší webový přístup přes "Blockchain Wallet API" rozhraní [13]. Využívá hlavně knihovnu Newtonsoft.Json a jsou zde implementovány funkce podle dokumentace na webu <http://blockchain.info>.

7.3.2 Projekt RpcClient

Projekt řeší přístup přes RPC funkce přes rozhraní "Json-RPC API" [14], které využívá knihovnu Newtonsoft.Json.

7.3.3 Projekt DBase

Tato část projektu zajišťuje databázi pro ukládání nových adres do adresáře a zaznamenání uživatelského loginu pro příští spuštění aplikace. Projekt je rozdělen do návrhového vzoru MVVM, kde View zobrazení je uplatněno v uživatelském rozhraní.

7.3.4 Bitcoin MyWallet

Celé uživatelské rozhraní má zajištěné závislosti na jednotlivé předcházející projekty, které zajišťují modelovou část návrhového vzoru. Projekt je rozdělen do několika následujících adresářů:

- Helpers – pomocné třídy pro získání dat.
- Models – modelové třídy, které nejsou vhodné na řešení v samostatném projektu.
- Resources – obrázkové zdroje.

- ViewModels – část projektu pro předání dat do View zobrazení.
- View – samotné zobrazení pro uživatele napsané ve XAMLu pro platformu Windows. Phone.

Aplikace měla zajištěnou tuto základní nabídku:

- history – třída pro získání historie o příchozích a odchozích transakcích.
- payment – třída zajišťující platební transakce.
- charts – pro získání kurzovních dat a grafů.
- book – pro adresář častých adres.

8 NÁVRH PŘENOSU APLIKACE

8.1 Příprava modelu MVVM

Po prozkoumání řešení na internetu pro novou aplikaci byl vybrán balíček Prism [52]. Je snadno implementovatelný a jeví se jako vhodný pro vytvoření prázdné aplikace ze šablony. Vygenerovaný projekt je multiplatformní aplikace, kdy pro každou platformu je vytvořena základní struktura. Aby takové řešení bylo možné pro balíček Prism, vývojáři vytvořili implementaci v podobě Portable Class Library (PCL), označenou jako přenositelná. Do tohoto projektu byly přidány balíčky NuGet pro Xamarin.Forms, Prism a Prism.Unity spolu se všemi jejich závislostmi.

V portable části pak vytváří MVVM vzor označovaný jako Prism MVVM [37]. V rámci balíčku lze použít i navigaci všech Views, které budou použity v řešení. Implementace je tak robustní, že umožňuje vývojáři implementovat hlavní funkcionalitu do portable části a ta se automaticky naváže na nativní úroveň pro jednotlivé platformy Android, iOS a Windows Phone. Funkcionalitu lze i dopisovat přímo do nativní úrovně každé platformy, pokud je to nezbytně nutné. Prism MVVM lze využít nejlépe v WPF aplikacích nebo i v Xamarin.Forms.

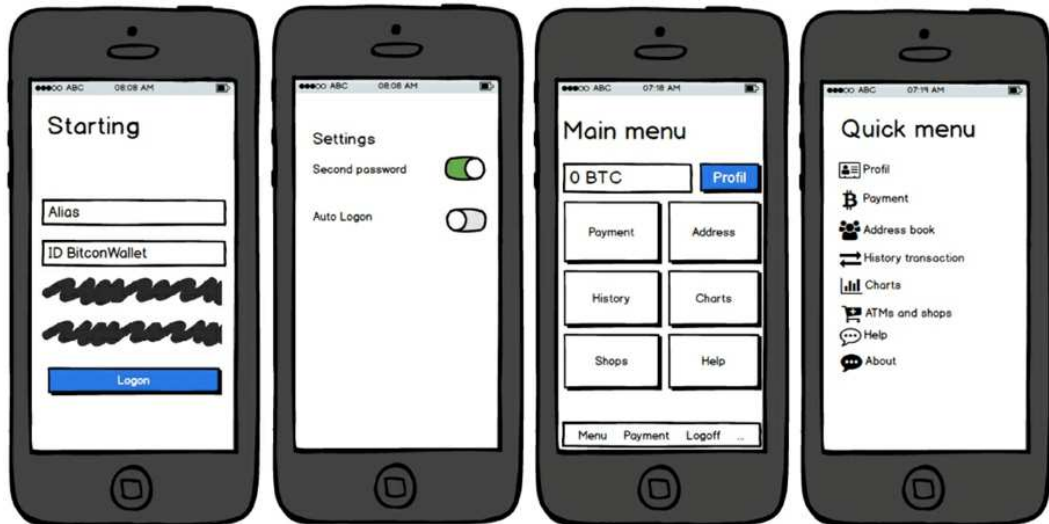
8.2 Návrh nové verze aplikace

Aplikace bude obsahovat dle rozvržení přihlašovací obrazovku s možným nastavením v budoucnu podle nutnosti rozšiřitelným, viz (Obr. 8.1). Obrázek také představuje základní zobrazovací stránku s menu, které bude informovat o množství Bitcoinů a také bude možné najít aktuální informace o profilu, platbách, kontaktech v adresáři, historii transakcí, tržních grafech, dostupnost obchodu i bankomatů v okolí, nápovědě a rychlého postranního menu pro přístup na jednotlivé položky menu.

Obrázek 8.2 zobrazuje rozvržení nabídky pro platební transakce. Následující dva obrázky ukazují rozvržení pro adresář kontaktů fungující na lokální databázi SQL.

Přístupové zobrazení pro historii transakcí lze vidět (Obr. 8.3) na prvních dvou obrázcích a následující obrázky představují zobrazení pro aktuální kurzovní lístek a grafy pro historii vývoje měn.

Poslední zobrazovací nabídkou v (Obr. 8.4) je nativní součást každé platformy, a to jsou mapy. Pro každou platformu bude potřeba nechat vygenerovat API klíč pro mapy. Tato nabídka bude představovat zobrazení všech dostupných bankomatů a obchodů na celém světě, včetně popisu pro jednotlivé body.



Obr. 8.1 Obrázek ukazuje návrh nové aplikace, která bude postavena na frameworku Xamarin s hlavním a zrychleným menu.



Obr. 8.2 Obrázek navrhuje další vrstvy realizace v prostředí Xamarin, konkrétně platební transakce a kontaktový adresář.



Obr. 8.3 Obrázek navrhuje další vrstvy realizace v prostředí Xamarin, konkrétně platební transakce a kontakový adresář.



Obr. 8.4 Obrázek navrhuje další vrstvy realizace v prostředí Xamarin, konkrétně platební transakce a kontakový adresář.

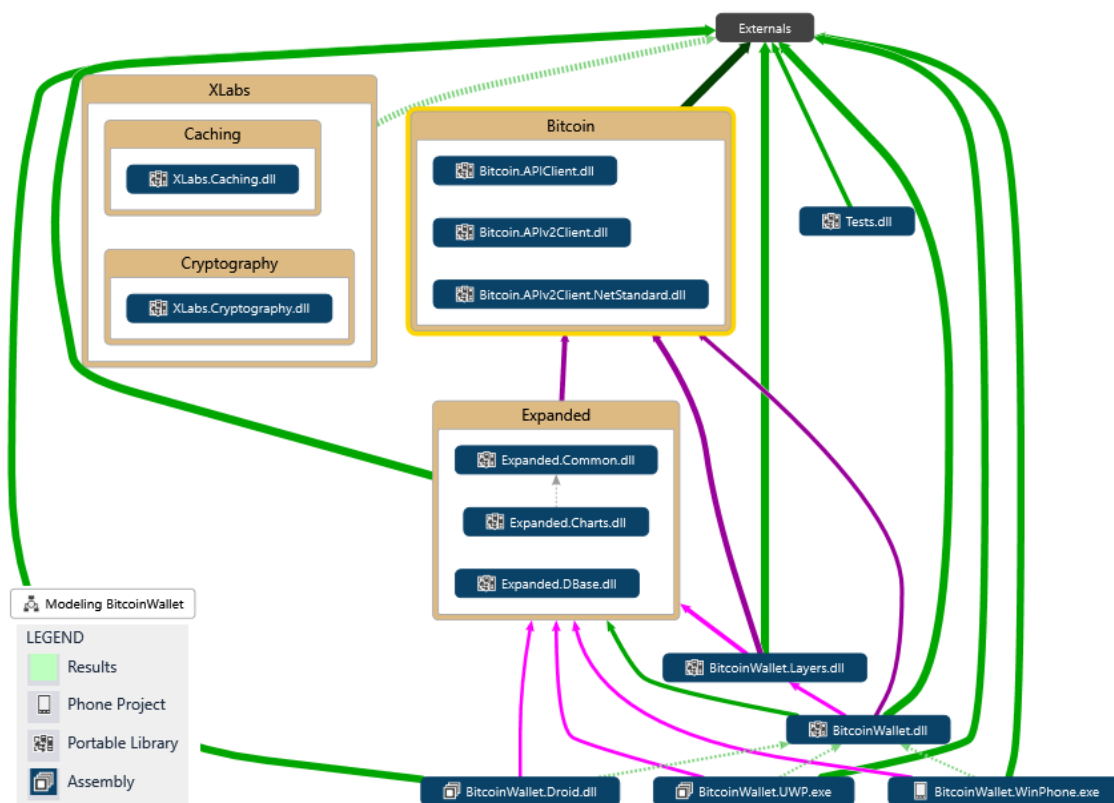
9 IMPLEMENTACE V XAMARIN

K vývoji aplikace bylo použito prostředí Visual Studio 2015 Enterprise. Dále, aby byl projekt atraktivnější, požádal jsem společnost Telerik o vývojový balíček mobilního prostředí Telerik UI for Mobile včetně zkušební verze Telerik UI for Xamarin. Společnost poskytla školní licenci pro vývoj založený na jejich knihovnách uživatelského rozhraní. Pro zvýšení efektivity práce a čistého kódu jsem požádal společnost JetBrains o školní balíček pro ReSharper 2015. Opět mi společnost vyšla vstříc.

Aplikace je založena na návrhovém vzoru MVVM od společnosti Prism, aby bylo aplikační rozhraní zcela oddělitelné od uživatelského rozhraní. Návrhový vzor usnadňuje případnou změnu celého uživatelského rozhraní bez nutnosti zasahovat do aplikačního rozhraní. Návrhový vzor se dodržuje ve všech přenositelných projektech. Řešení je úmyslně rozvrženo do 14 projektů viz (Obr. 9.1), aby všechny vytvořené knihovny byly použitelné pro jiné budoucí projekty nebo byly schopné se jednoduchým způsobem nahradit novější verzí. V oddílech níže jsou jednotlivé projekty popsány a jsou zobrazeny důležité třídy s metodami. Obrázek 9.2 zobrazuje závislosti jednotlivých projektů s názvy namespaces v C#. Do řešení bylo nutné začlenit i nejnovější verzi knihovny XLabs zcela otevřeného projektu, který rozšiřuje Xamarin.

Soupis projektových částí: Projekty XLabs.Caching, XLabs.Cryptography a Expanded.Common jsou z otevřeného řešení XLabs lehce upravené a jsou opravené chybné součásti, na které jsem narazil. Projekty Bitcoin.APIClient, Bitcoin.APIv2Client a Bitcoin.APIv2Client.NetStandard implementují aplikační rozhraní pro bitcoinový server Blockchain.com. Portable knihovna Bitcoin.APIv2Client implementuje aplikační rozhraní verze 2 pro server. Knihovna Bitcoin.APIv2Client.NetStandard je napsaná v NetStandard, verze 1.3, a implementuje stejné aplikační rozhraní verze 2 s tím rozdílem, že dokáže plně nahradit portable knihovnu Bitcoin.APIv2Client. Projekty BitcoinWallet, BitcoinWallet.Layers jsou portable knihovny napsané s prvky Xamarinu včetně vzoru MVVM, implementující hlavní část uživatelského rozhraní. Projekt Expanded.Charts implementuje uživatelské rozhraní grafů od společnosti Telerik. Projekt Expanded.DBBase implementuje databázi SQLite včetně metod pro dotazy na databázi. Projekt Tests je určen pro unit testy, výkonnostní testy a testy rozhraní. Poslední část projektů BitcoinWallet.Droid, BitcoinWallet.UWP a BitcoinWallet.WinPhone implementuje nativní úroveň na konkrétní platformy. Obsahují pouze závislostní třídy, obrázkové zdroje a konfigurační nastavení.

Celé řešení aplikace v Xamarinu je implementováno v těchto knihovnách zobrazených na obrázku včetně jejich závislostí. Knihovna Expanded.DBBase realizuje kompletně celou databázovou stránku realizovanou pomocí SQLite. Knihovna Bitcoin.APIv2Client zaštiťuje aplikační rozhraní pro Bitcoin server. Expanded.Charts obsahuje implemen-



Obr. 9.1 Obrázek zobrazující celkové rozvržení 14 projektů na soubě závislých částí se směrovaný komunikačním kanálem.

taci uživatelského rozhraní pouze pro grafy od společnosti Telerik. Knihovny BitcoinWallet a BitcoinWallet.Layers realizují hlavní součásti uživatelského rozhraní pomocí modelu MVVM, a to vše v přenositelné formě pro Xamarin.

Obrázky 9.4, 9.5 zachycující implementaci databáze SQLite s třídami pro operace nad daty. Celá databáze má tři implementační části. Hlavní část je implementována pomocí generických tříd pro obecné operace nad daty. Druhou část tvoří oddělené třídy metod pro konkrétní tabulky. Třetí částí jsou samotné modely, které představují modely neboli objekty pro tabulky. Načtení připojení na databázi se řeší veřejnou třídou a veřejnou statickou třídou. Na nativní úrovni pro jednotlivé platformy jsou implementovány závislosti na připojení k vytvořené lokální databázi.

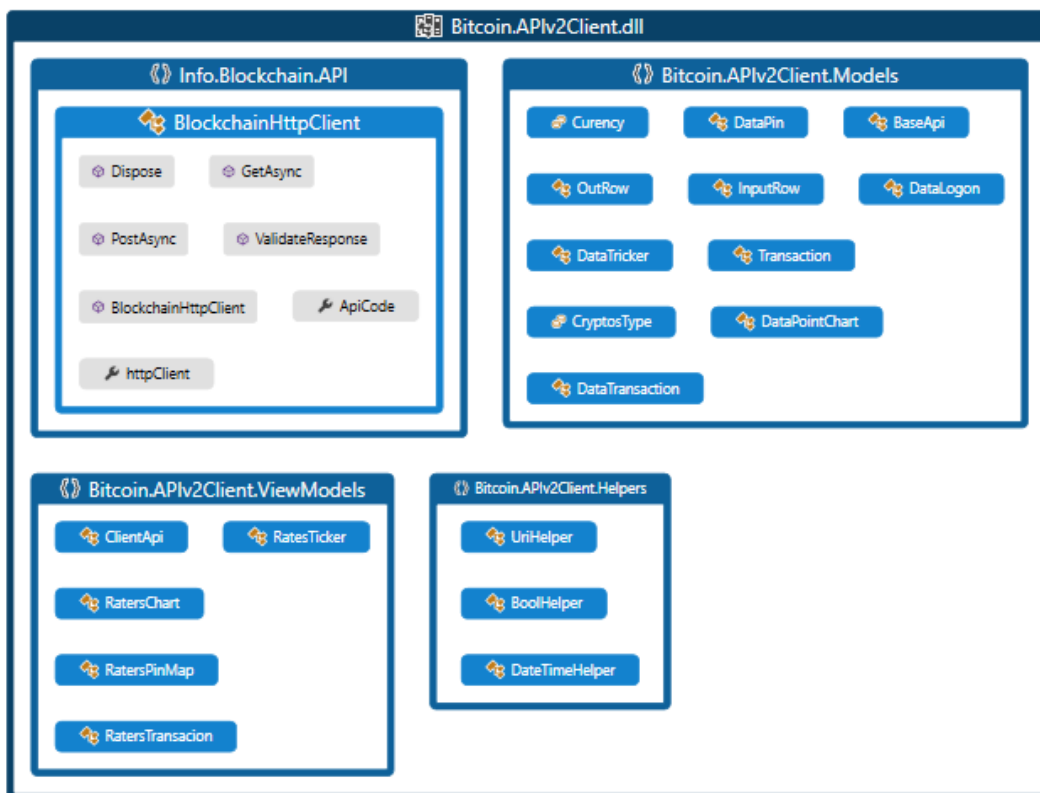
9.1 PŘENOSITELNÉ KNIHOVNY

Celá mobilní aplikace je primárně řešená pro Windows Phone 8.1 z důvodu původní implementace prototypového projektu. Prototypový projekt, jak bylo zmíněno dříve, byl tvořený přímo na nativní část Windows Phone 8.0. Z implementačního hlediska bylo potřeba přepsat uživatelské rozhraní pomocí komponent Xamarin. Aplikační rozhraní pro Bitcoin server BlockChain.com muselo být z 90 procent také znovu imple-

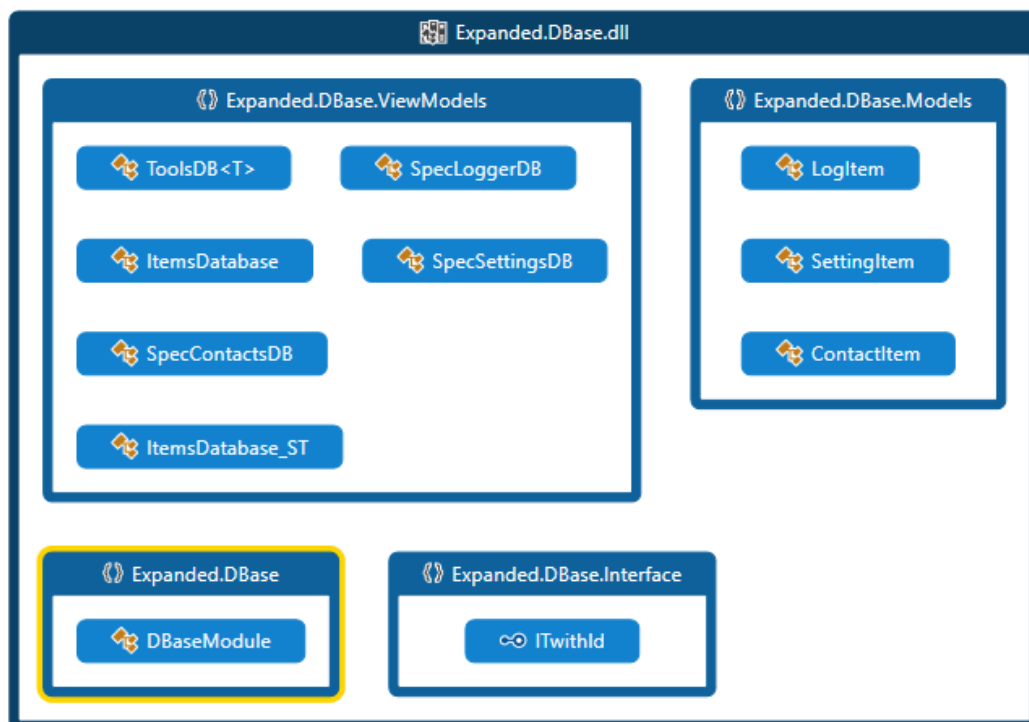


Obr. 9.2 Obrázek zachycuje šest hlavních projektových částí, které zajišťují hlavní funkční jádro aplikace v Xamarinu.

mentováno. Původní verze se během dvou let vyvinula na zabezpečenější variantu 2. Implementace byla z tohoto pohledu jednodušší, protože z části použila knihovny přímo z daného serveru jako přídatný zdroj. Databázová část byla implementována do přenositelného řešení SQLite. Knihovny pro zobrazení grafu musely být od společnosti Telerik znovu začleněny, ale již na projektovém řešení Xamarin. Aplikace je řešena zásadně pomocí přenositelných knihoven pro všechny platformy. Účelem tak velkého počtu projektů je docílit přenositelnosti knihoven na jiné budoucí projekty a zajistit přehlednost implementačních částí. Následující obrázek zachycuje implementaci hlavního jádra v knihovně BitcoinWallet a knihovna BitcoinWallet.Layers zajišťuje oddělené vrstvy uživatelského rozhraní s možností libovolného rozšíření. Knihovna z projektu Expanded.Charts je implementována odděleně z důvodu možné výměny placečných knihoven společností Telerik za libovolnou jinou knihovnu pro grafy, včetně celé logiky. Během implementace byl kladen důraz na to, aby 90 procent aplikace bylo napsáno v přenositelných knihovnách a aby se snadněji implementovaly nativní součásti



Obr. 9.3 Obrázek zobrazuje implementaci pomocných tříd, modelů a tříd pro komunikaci se serverem.

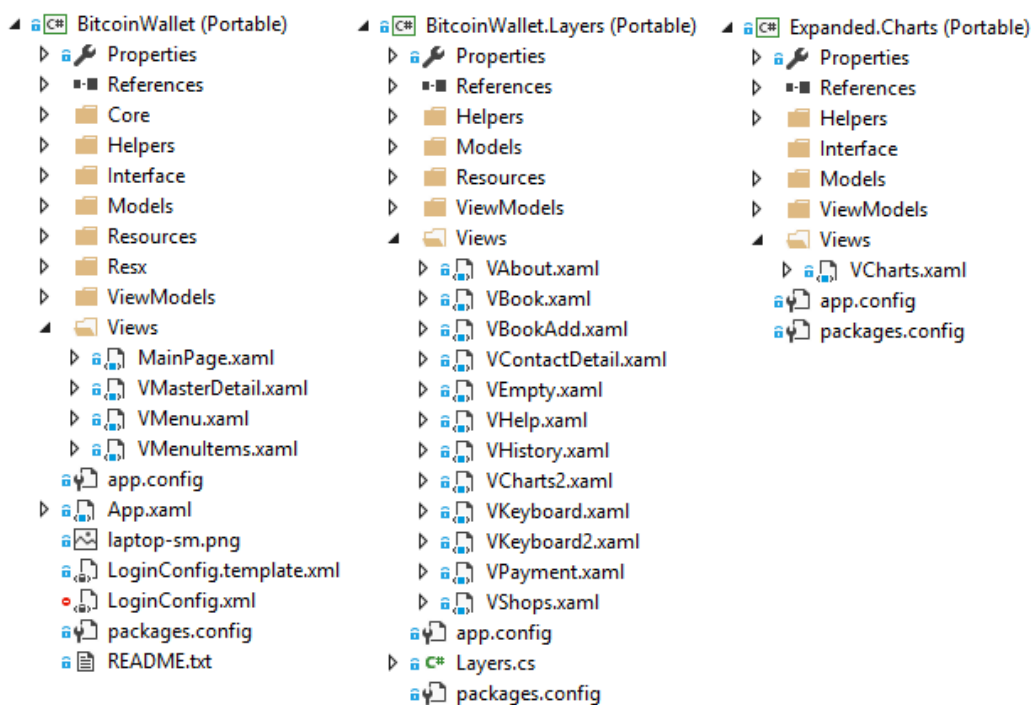


Obr. 9.4 Obrázek zobrazující implementační projekt pro databazi SQLite.



Obr. 9.5 Obrázek zobrazující modelovou implementací pro databazové tabulky SQLite.

pro každou platformu.



Obr. 9.6 Obrázek popisující kompletní řešení XAML tříd pro zobrazení neboli pohled.

9.2 PLATFORMA WINDOWS

Zajišťuje implementaci nativních součástí pro jazykovou lokalizaci, pomocnou třídu pro ukládání a otvírání souborů, třídu pro načtení lokální databáze SQLite a v neposlední řadě konfiguraci nativní součásti. V projektu bylo potřeba dokonfigurovat kompilaci Xamarin frameworku, inicializaci knihoven pro mapy a knihoven pro grafy Telerik. Nativní projekt BitcoinWallet.WinPhone zajišťuje implementaci pro zařízení Windows Phone 8.1. Zajímavější je ale projekt BitcoinWallet.UWP, který podporuje všechna zařízení s operačním systémem Windows 10, tedy mobilní zařízení, tablety, notebooky, stolní počítače, a dokonce i televize, pokud mají operační systém Windows.

9.3 PLATFORMA ANDROID

Oproti platformě Windows bylo potřeba provést odlišným způsobem konfiguraci povolených oprávnění v AndroidManifest.xml a pro Android verze 6 bylo potřeba provést implementaci dynamických oprávnění přímo do MainActivity.cs. Bylo také potřeba provést odlišnou inicializaci tříd pro mapy, včetně vložení API klíče pro Google Maps.

9.4 INTEGRACE V REÁLNÉM ZAŘÍZENÍ A EMULÁTORY

Pro každou platformu existuje možnost vyvíjet aplikaci za pomoci emulátoru pro obecný typ zařízení. I když z estetického hlediska každý emulátor vypadá jako konkrétní stroj, vždy funkční stránku hardwarového protějšku jen simuluje. Pro platformu Windows jsou za pomoci Hyper-V vytvořeny virtuální stroje, na které probíhá nahrávání vyvíjené aplikace. Pro platformu Android existuje hned několik otevřených verzí emulátorů. Některé implementují virtualizaci pomocí knihoven, které vytvářejí odlišné prostředí od Hyper-V. V tomhle pohledu je to problém, nelze pak používat Hyper-V s emulátory Windows. Druhou variantou jsou emulátory bez podpory rychlé virtualizace, ty jsou však nesmírně pomalé, na spuštění se čeká příliš dlouho. Dokonce i v rámci frameworku Xamarin je implementováno pre-zobrazení, to ale bohužel funguje jen se základními komponenty Xamarin. Pokud se použije jakýkoliv rozšiřující balíček pro Xamarin, pre-zobrazení končí. Vývoj Xamarinu stále pokračuje a časem vše může fungovat optimálně. Další možností je instalace přímo na zařízení, ale zde je potřeba opatrnosti. Visual Studio musí být puštěno s právy administrátora a musí být ve službách povoleno USB odesílání. Následně je možné aplikaci instalovat přímo na zařízení. Dále je vhodné v nastavení Visual Studia zapnout diagnostické nástroje během kompilování a nahrávání, jinak by v chybovém seznamu byly zobrazeny obecné chyby bez popisu.

10 POPIS APLIKACE A VERIFIKACE

Aplikace byla nahrána a ozkoušena na zařízení Windows Phone 8.1, Windows Phone 10, notebooku a stolním počítači se systémem Windows 10 a také ozkoušena na Android zařízení verze 6. Na následujících obrázcích 10.1, 10.2, 10.3 a 10.4 je ukázána:

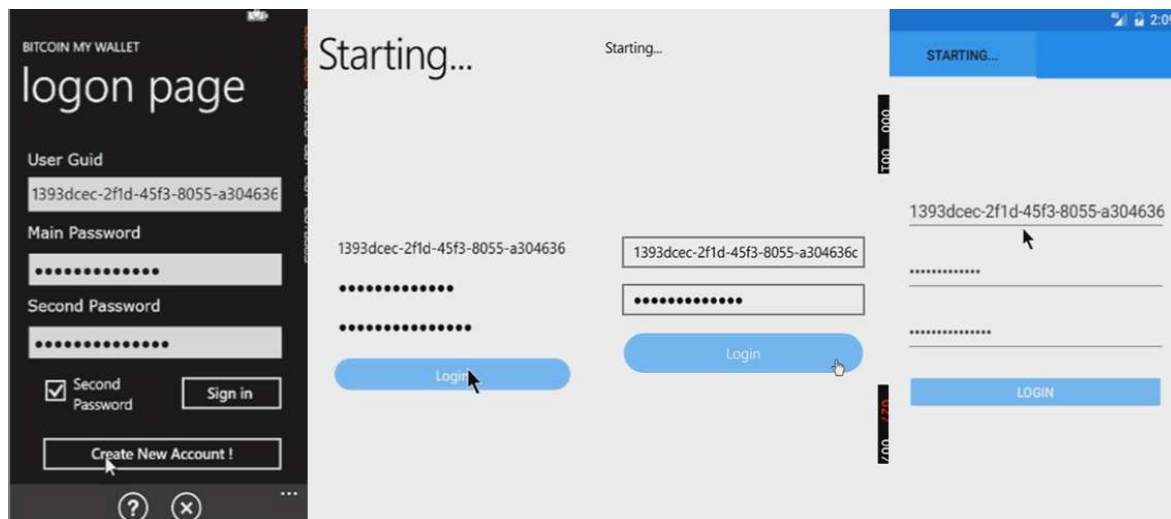
- Implementace prototypové aplikace na zařízení s Windows Phone 8.0
- Implementace aplikace v Xamarinu na zařízení s Windows Phone 8.1
- Implementace aplikace v Xamarinu na UWP zařízení
- Implementace aplikace v Xamarinu na Android zařízení

Pro každou platformu byli vytvořeno instruktážní video, které můžete zhlédnout zde [54].

Aplikace obsahuje tyto vlastnosti v rámci řešení:

- lokalizaci Eng/Cz (aspoň pro vstupní nabídku je nastaveno),
- databázi SQLite,
- logování do souboru, konzole, databáze,
- parsování z XML, JSON,
- stahování struktur z webu,
- menu je možné libovolně rozšířit,
- dodržena 90% implementace kódu (čteme "devadesátiprocentní") v přenositelné části,
- knihovny je možné přenést z Portable na Standard 1.3 nebo vyšší,
- je striktně dodržován návrhový vzor MVVM pro oddělení uživatelského rozhraní od aplikačního pro budoucí úpravy obou rozhraní, vývoj mohl být veden v oddělených týmech,
- jsou využity mapy s bankomaty a shopy pro Bitcoin, na Android platformě funguje i nápověda k jednotlivým PINům),
- pro mapy jsou použity i lokalizační balíčky pro určení polohy, aby bylo jasné, jak se dostat nejrychleji k bankomatu nebo obchodu,
- pro portál BlockChain.com bylo nezbytné dokonce odchytil komunikaci, neboť jejich API verze 2 je určená pro servis službu a není zcela veřejná,

- grafy jsou vytvořeny pomocí knihoven od společnosti Telerik díky školní licenci,
- celé řešení je vedené pomocí verzovacího systému GitHub a je veřejné [55]
- pro budoucí snadnější buildování je možnost využít software TeamCity od společnosti Microsoft, existuje i veřejný portál pro pravidelné buildování,
- k projektu jsou vytvořeny i webové stránky, bohužel z časových důvodů nejsou plně zaimplementované,
- řešení obsahuje 14 podprojektů, je to z důvodu opětovné použitelnosti jednotlivých knihoven pro jiné budoucí projekty, ale i z důvodu možnosti odděleného vývoje,
- aplikace umí uchovávat nastavení, kontakty a logy díky lokální databázi SQLite,
- pro logy existuje možnost provést export do souboru (xml, json, txt, html) a poslat e-mailem, ale bylo by potřeba doimplementovat 1–2 třídy do aplikačního rozhraní a jeden pohled do uživatelského rozhraní,
- z projektu je možnost vygenerovat technickou dokumentaci, neboť se jedná o komentovaný kód



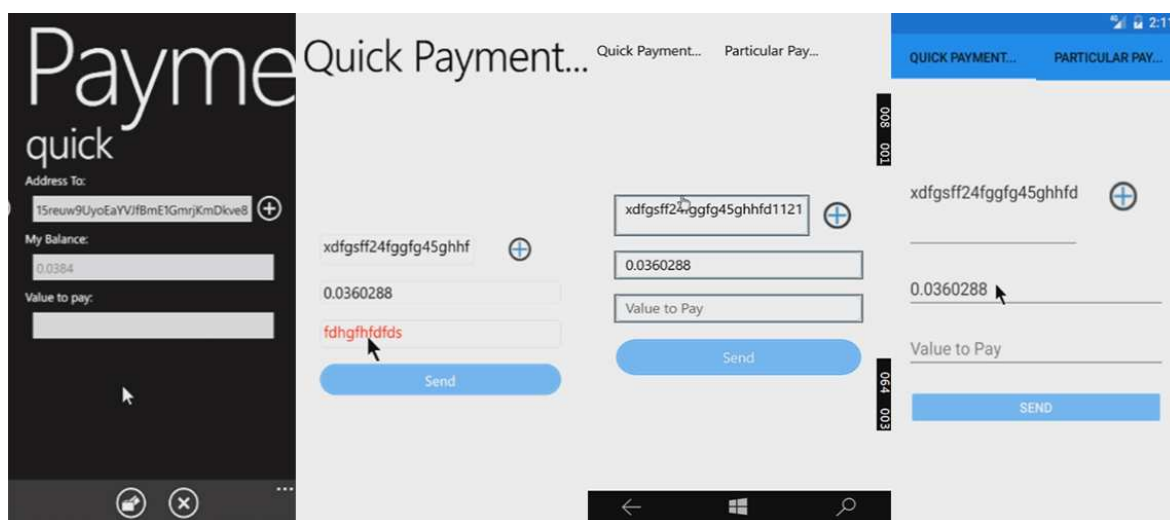
Obr. 10.1 Obrázek zobrazuje implementaci přihlašování v levé části je vidět prototypovou aplikaci a nesleduje Windows Phone, UWP a platforma Android.



Obr. 10.2 Obrázek zobrazuje implementaci hlavního menu v levé části je vidět prototypovou aplikaci a nesleduje Windows Phone, UWP a platforma Android.



Obr. 10.3 Obrázek zobrazuje implementaci rychlého postranního menu v levé části je vidět prototypovou aplikaci a nesleduje Windows Phone, UWP a platforma Android.



Obr. 10.4 Obrázek zobrazuje implementaci nabídky pro zadání transakce v levé části je videt prototypovou aplikace a nesleduje Windows Phone, UWP a platforma Adnroid.

11 TESTOVÁNÍ

Aplikace byla testována za pomoci kolegy Radovana Matouška. Na základě zpětné vazby bylo opravováno uživatelské rozhraní, opraveny všemožné pády aplikace při načítání, komunikace s databází a komunikace bez internetového připojení. V rámci testování byl kladen důraz na stabilitu aplikace, funkčnost, přehlednost nabídek včetně jednoduchosti. Aplikace se dostala z vývojové do beta verze, ale nemohu jasně říci, že je stoprocentně stabilní, vždy se najdou drobné chybičky. Beta verze je nyní připravena pro platformu Windows pro širší veřejnost a v následujícím měsíci bude přidána do Windows Apps Storu. Na Android platformu proběhlo jen pár základních testů a je potřeba provést další testy, nyní se nachází v alfa verzi.

11.1 Rozšiřitelnost aplikace

Aplikace je použitelná již ve stávající formě, ale existuje několik možných budoucích rozšíření, které by aplikaci zpříjemnily v používání.

- Možnost rozšířit aplikaci pro jiné servery Bitcoin, stačí pouze dopsat aplikační rozhraní.
- Přidat více funkcí, jako například skenování QR kódů a generování.
- Přidání informační lišty o synchronizaci.
- Statistiky z více směnárén.
- Dokonce je možné vytvořit k peněžence vlastní server pro Bitcoin účty.
- Aplikaci lze libovolně rozšířit o posuvné hlavní menu a o jakoukoliv funkcionalitu například na základě zpětných požadavků od široké veřejnosti.

ZÁVĚR

Hlavním cílem diplomové práce bylo prozkoumat možnou přenositelnost starších aplikací napsaných pro konkrétní platformu do nového frameworku Xamarin. Za tímto účelem byla zvolena prototypová aplikace BitcoinWallet vytvořená mnou v rámci jiného projektu a byla cílená na platformu Windows Phone 8.0.

Abyste docílilo daného výsledku, bylo třeba nastudovat nový framework Xamarin se všemi prvky potřebnými pro přepis a zvolit dobrý návrhový vzor pro konstrukci aplikace. Nakonec byl vybrán vzor MVVM od společnosti Prism. Byl zvolen z toho důvodu, že zaštiťuje všechny nyní používané mobilní platformy, a dokonce podporuje i stolní platformy Windows 8 a vyšší. Celé řešení je cíleno multiplatformně, za tímto účelem byly nastudovány odlišné specifikace platform. Poskytuje nový způsob implementace všech důležitých součástí pro mobilní zařízení tak, aby uživatel nepoznal rozdíl.

Prototypová aplikace byla také zvolena kvůli nutnosti implementovat aplikační rozhraní pro Bitcoin server. V rámci teoretické části byly současně vysvětleny hashovací funkce a asymetrická kryptografie plynule navazující na celou problematiku virtuální měny Bitcoin. Systém Bitcoin je v provozu pár let a již ho používají v nemalém počtu uživatelé z celého světa. Je nový, proto jsou dostupné pouze internetové zdroje na různých portálech nebo fórech. Nastudování technologií práci sice ztížilo, ale myslím, že tento bod zadání se mi povedlo pečlivě splnit. Informace o této problematice jsem čerpal z několika webů ověřených uživateli, které mi poskytly dobrý přehled o celém systému, a také ze své dva roky staré práce. Problematika je probrána ve starší práci podrobněji, a proto jsem se snažil popsat ji zde jen pro pochopení. Spíše jsem ale kladl důraz na změny a novinky hlavně v Bitcoin serverech.

Práce pokračuje návrhem a zhodnocením možného řešení přepisu prototypové aplikace do vývojového frameworku Xamarin. Následně navazuje na projektové řešení přímo do frameworku Xamarin. Přenesení aplikace nebyl snadný úkol. Přenositelnost aplikace je možná pouze pro aplikační rozhraní, pokud bylo dobře odděleno od uživatelského rozhraní. Dokonce mnou napsaná prototypová aplikace na takovou možnost myslela. Lze tedy provést přenesení? Bohužel jen asi 5 % kódové části. Je to z toho důvodu, že za pouhé dva roky se provedlo tolik změn v aplikačním rozhraní Blockchain.com, že bylo potřeba rozdíly implementovat znovu. Další problémem byla databáze včetně modelů pro tabulky. Prototypová aplikace řešila databázi platformně specifickým kódem. Tuto část sice lze přenést přímo do nativní úrovně dané platformy, ale bylo by pak třeba pro každou platformu znovu psát metody pro implementaci platformní databáze. Problém jsem vyřešil multiplatformním databázovým systémem SQLite, který má navíc ještě otevřený kód.

Díky dobře zvolenému návrhovému vzoru Model-View-ViewModel a díky důrazu

kladenému na rozšiřitelnost a přenositelnost mezi platformami bylo docíleno toho, že 90 % kódu je implementováno do přenositelných knihoven. Aplikace může být dále vyvíjena, a to ve dvou týmech řešících zvlášť uživatelské a aplikační rozhraní.

Implementace využívá všech výhod portálu <http://blockchain.info>. Portál umožňuje přístup do peněženky i pomocí webového rozhraní. Dále byly dostatečně vhodně popsány pro tento webový server protokoly pro komunikaci. Jako dostatečná se stále jeví dokumentace aplikačního rozhraní a také server Bitcoin pro správu peněženek byl dobře zvolený. Během dvou let se zvýšilo zabezpečení, ověřování přístupu, vylepšilo se aplikační rozhraní a začlenila se komunikace pomocí TLS protokolu.

Aplikace je řešena v rámci 14 subprojektů, z toho hlavní částí řešení jsou 3 projekty, které řeší uživatelské rozhraní a oddělené aplikační rozhraní na přenositelné úrovni. Platformní úroveň je tvořena projekty pro Windows Phone, UWP a Android. Zbylé projekty tvoří databázové a webové komunikační rozhraní, bez něhož by aplikace nefungovala, a bylo potřeba je od nuly realizovat.

Pro ověření funkčnosti a stability byl začleněn člověk, který všem součástem rozumí a bez kterého by aplikace měla pomalejší vývoj. Díky zpětné vazbě byly opraveny chyby v uživatelském i aplikačním rozhraní. V neposlední řadě byla aplikace integrována na skutečná zařízení, a ukázala tak funkčnost vybraného řešení za pomoci Xamarinu. Nyní je možno aplikaci předat světu jako beta verzi a na základě zpětné vazby ji dále rozšiřovat a vylepšovat.

SEZNAM POUŽITÉ LITERATURY

- [1] Algoritmy: Komunikace Deffie Hellman [online]. 2017, [cit. 2017-5-10].
URL <http://hdl.handle.net/11012/53481>
- [2] Apple: About Keychain Services [online]. Listopad 2016, [cit. 2016-11-20].
URL
<https://developer.apple.com/library/content/documentation/Security/Conceptual/keychainServConcepts/01introduction/introduction.html>
- [3] Apple: About Software Security [online]. Listopad 2016, [cit. 2016-11-20].
URL
https://developer.apple.com/library/content/documentation/Security/Conceptual/Security_Overview/Introduction/Introduction.html
- [4] Apple: Choosing a Membership [online]. Listopad 2016, [cit. 2016-11-20].
URL <https://developer.apple.com/support/compare-memberships/>
- [5] Apple: Introduction to Secure Coding Guide [online]. Listopad 2016, [cit. 2016-11-20].
URL <https://developer.apple.com/library/content/documentation/Security/Conceptual/SecureCodingGuide/Introduction.html>
- [6] Apple: Purchase and Activation [online]. Listopad 2016, [cit. 2016-11-20].
URL <https://developer.apple.com/support/purchase-activation/>
- [7] Atom: Architecture and Security of Android phones [online]. Leden 2011, [cit. 2017-05-1].
URL <http://totaltechsecurity.blogspot.cz/search/label/Android>
- [8] Bitcoin.it: Bitcoin [online]. April 2014, [cit. 2014-05-27].
URL <https://en.bitcoin.it/wiki/Bitcoin>
- [9] Bitcoin.it: Block Explorer [online]. April 2014, [cit. 2014-05-27].
URL https://en.bitcoin.it/wiki/Block_Explorer
- [10] Bitcoin.it: Block hashing algorithm [online]. April 2014, [cit. 2014-05-27].
URL https://en.bitcoin.it/wiki/Block_hashing_algorithm
- [11] Bitcoin.it: Blocks [online]. April 2014, [cit. 2014-05-27].
URL <https://en.bitcoin.it/wiki/Blocks>

- [12] Bitcoin.it: FAQ - Bitcoin [online]. January 2013, [cit. 2013-01-02].
URL https://en.bitcoin.it/wiki/FAQ#How_long_will_it_take_to_generate_all_the_coins.3F
- [13] blockchain.info: Blockchain wallet API [online]. 2014, [cit. 2014-05-27].
URL https://blockchain.info/api/blockchain_wallet_api
- [14] blockchain.info: JSON RPC API [online]. 2014, [cit. 2014-05-27].
URL https://blockchain.info/api/json_rpc_api
- [15] blockchain.info: Bitcoin Developer API's [online]. 2014, [cit. 2017-05-27].
URL <https://blockchain.info/api>
- [16] blockexplorer.com: Bitcoin Block Explorer [online]. May 2014, [cit. 2014-05-27].
URL <http://blockexplorer.com/>
- [17] Brown, D. R. L.: Generic Groups, Collision Resistance, and ECDSA. Cryptology ePrint Archive, Report 2002/026, 2002, <http://eprint.iacr.org/>.
- [18] Cornell, D.: Secure Mobile Application Development Reference [online]. 2010 - 2011, [cit. 2016-12-20].
URL <http://www.denimgroup.com/media/pdfs/MobileDevReference.pdf>
- [19] Cyanogenmod: Cyanogenmod [online]. Listopad 2016, [cit. 2016-11-20].
URL <https://www.cyanogenmod.org/>
- [20] DenimGroup: Developing Secure Mobile Applications [online]. Květen 2013, [cit. 2016-11-20].
URL <http://www.slideshare.net/denimgroup/developing-secure-mobile-applications-17732256>
- [21] DenimGroup: DenimGroup [online]. Listopad 2016, [cit. 2016-11-20].
URL <http://www.denimgroup.com/>
- [22] DenimGroup: Smart Phones Dumb Apps [online]. Říjen 2010, [cit. 2016-11-20].
URL http://www.slideshare.net/denimgroup/smart-phones-dumb?from=ss_embed
- [23] Domingo: Apple engineer presumes to be one of the safest organizations in the world [online]. Duben 2016, [cit. 2017-05-1].
URL <http://www.copyright.gov/1201/2008/comments/lohmann-fred.pdf>
- [24] Google: Google help [online]. Listopad 2016, [cit. 2016-11-20].
URL <https://support.google.com/googleplay/android-developer/answer/6112435?hl=c>

- [25] Hanáček, P.: Asymetrická kryptografie. Faculty of Information Technology, Technical University of Brno, Lecture 2014, 2007.
- [26] Husák, M.; Cermák, M.; Jirsík, T.; aj.: Network-Based HTTPS Client Identification Using SSL/TLS Fingerprinting. In *10th International Conference on Availability, Reliability and Security, ARES 2015, Toulouse, France, August 24-27, 2015*, IEEE Computer Society, 2015, ISBN 978-1-4673-6590-1, s. 389–396, doi:10.1109/ARES.2015.35.
URL <http://dx.doi.org/10.1109/ARES.2015.35>
- [27] IDC: Smartphone OS Market Share, 2016 Q3 [online]. Leden 2014, [cit. 2017-05-1].
URL <http://www.idc.com/promo/smartphone-market-share/os>
- [28] InfoSecurity, M.: Security Considerations in the Windows Phone 8 Application Environment [online]. Listopad 2016, [cit. 2016-11-20].
URL <https://www.mwrinfosecurity.com/our-thinking/security-considerations-in-the-windows-phone-8-application-environment/>
- [29] Jagtap, H.: All You Wanted To Know About OWASP Top 10 Mobile Security Project [online]. Listopad 2016, [cit. 2016-11-20].
URL <https://www.appvigil.co/blog/2015/06/all-you-wanted-to-know-about-owasp-top-10-mobile-security-project/>
- [30] Johnson, D.; Menezes, A.; Vanstone, S. A.: The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int. J. Inf. Sec.*, 2001: s. 36–63.
- [31] Johnson, E. B. B. D.; Smid, M. E.: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography. In *COMPUTER SECURITY*, 2007.
- [32] Kelsey, J.; Schneier, B.: Second Preimages on n -bit Hash Functions for Much Less than 2^n Work. Cryptology ePrint Archive, Report 2004/304, 2004, <http://eprint.iacr.org/>.
- [33] Kilián, K.: Android dominuje trhu mobilních telefonů [online]. Leden 2014, [cit. 2017-05-1].
URL <https://www.svetandroida.cz/android-trh-dominance-201408>
- [34] Malik, M. Y.: Efficient Implementation of Elliptic Curve Cryptography Using Low-power Digital Signal Processor. *CoRR*, 2011.

- [35] Martínez, V. G.; Álvarez, F. H.; Encinas, L. H.; aj.: A comparison of the standardized versions of ECIES. In *IAS*, IEEE, 2010, ISBN 978-1-4244-7407-3, s. 1–4.
- [36] Microsoft: The MVVM Pattern [online]. 2015, [cit. 2016-01-7].
URL <https://msdn.microsoft.com/en-us/library/hh848246.aspx>
- [37] Microsoft: MVVM QuickStart Using the Prism Library 5.0 for WPF [online]. 2016, [cit. 2017-05-2].
URL
[https://msdn.microsoft.com/en-us/library/gg430857\(v=pandp.40\).aspx](https://msdn.microsoft.com/en-us/library/gg430857(v=pandp.40).aspx)
- [38] Microsoft: Account types, locations, and fees [online]. Listopad 2016, [cit. 2016-11-20].
URL <https://msdn.microsoft.com/en-us/windows/uwp/publish/account-types-locations-and-fees>
- [39] Microsoft: App Developer Agreement [online]. Listopad 2016, [cit. 2016-11-20].
URL <https://blogs.msdn.microsoft.com/windowsstore/2012/03/12/licensing-apps/>
- [40] Microsoft: Getting paid [online]. Listopad 2016, [cit. 2016-11-20].
URL <https://msdn.microsoft.com/en-us/windows/uwp/publish/getting-paid-apps>
- [41] Microsoft: Native code for Windows Phone 8 [online]. Listopad 2016, [cit. 2016-11-20].
URL [https://msdn.microsoft.com/en-us/library/windows/apps/jj681687\(v=vs.105\).aspx](https://msdn.microsoft.com/en-us/library/windows/apps/jj681687(v=vs.105).aspx)
- [42] Microsoft: Security for Windows Phone 8 [online]. Listopad 2016, [cit. 2016-11-20].
URL [https://msdn.microsoft.com/en-us/library/windows/apps/ff402533\(v=vs.105\).aspx](https://msdn.microsoft.com/en-us/library/windows/apps/ff402533(v=vs.105).aspx)
- [43] Microsoft: Zaregistrujte se jako vývojář aplikací [online]. Listopad 2016, [cit. 2016-11-20].
URL <https://developer.microsoft.com/cs-cz/store/register>
- [44] Mokris, K.: Secure mobile development: Testing for the OWASP Mobile Top 10 [online]. Listopad 2016, [cit. 2016-11-20].
URL <https://www.nowsecure.com/blog/2016/10/13/secure-mobile-development-testing-owasp-mobile-top-10/>

- [45] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System [online]. September 2009, [cit. 2012-11-15].
URL <http://bitcoin.org/bitcoin.pdf>
- [46] Oreilly: The BlockChain [online]. 2017, [cit. 2017-5-9].
URL http://chimera.labs.oreilly.com/books/1234000001802/ch07.html#_structure_of_a_block
- [47] Owasp.org: OWASP Mobile Security Project [online]. Listopad 2016, [cit. 2016-11-20].
URL https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks
- [48] Owasp.org: OWASP Mobile Security Project [online]. Listopad 2016, [cit. 2016-11-20].
URL https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
- [49] Owasp.org: Top Ten Mobile Risks 2016 [online]. Listopad 2016, [cit. 2016-11-20].
URL https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10
- [50] Owasp.org: Top Ten Mobile Risks [online]. Listopad 2016, [cit. 2016-11-20].
URL https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks
- [51] Petzold, C.: *Creating Mobile Apps with Xamarin.Forms*. Microsoft Press., 2016, ISBN 978-1-5093-0297-0, 1187 s.
URL <https://developer.xamarin.com/guides/xamarin-forms/creating-mobile-apps-xamarin-forms/>
- [52] Prism: Prism Library [online]. 2016, [cit. 2016-02-27].
URL <https://github.com/PrismLibrary/Prism/blob/master/docs/Xamarin-Forms/1-Getting-Started.md>
- [53] Prokop, B. T.: BitCoin peněženka pro platformu Windows Phone [online]. 2017, [cit. 2017-5-10].
URL <http://hdl.handle.net/11012/53481>
- [54] Prokop, B. T.: Bitcoin Wallet Library movies gallery [online]. 2017, [cit. 2017-5-10].
URL https://www.youtube.com/playlist?list=PLJjTYBRz8jZz6Bvu94fnZ_4LbnzK4b81A

- [55] Prokop, B. T.: Bitcoin Wallet Library [online]. 2017, [cit. 2017-5-10].
URL <https://github.com/Terni/BCWLibrary>
- [56] Schneier, B.: Schneier on Security: Cryptanalysis of SHA-1 [online]. January 2013, [cit. 2013-01-04].
URL
http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html
- [57] Sole, A. D.: Xamarin - Working with Local Databases in Xamarin.Forms Using SQLite[online]. Únor 2016, [cit. 2017-1-20].
URL <https://msdn.microsoft.com/en-us/magazine/mt736454.aspx>
- [58] Tavlikos, D.: *iOS Development with Xamarin Cookbook*. Paclt Publishing, May 2014, ISBN 978-1-84969-892-4, 386 s.
- [59] Veracode: Veracode [online]. Listopad 2016, [cit. 2016-11-20].
URL <http://www.veracode.com/blog/2010/12/mobile-app-top-10-list>
- [60] Versluis, G.: *Xamarin Continuous Integration and Delivery - Team Services, Test Cloud, and HockeyApp*. Apress, 2017, ISBN 978-1-4842-2715-2,
doi:10.1007/978-1-4842-2716-9.
URL <https://doi.org/10.1007/978-1-4842-2716-9>
- [61] Wysopal, C.: Mobile App Top 10 List [online]. Listopad 2016, [cit. 2016-11-20].
URL <http://www.veracode.com/directory/mobileapp-top-10>

SEZNAM OBRÁZKŮ

Obr. 1.1	Obrázek popisující možné hrozby od uživatele zařízení, od ostatních aplikací v zařízení, služeb třetích stran a webových služeb. Zdroj [18].	11
Obr. 1.2	Architektura zabezpečení iOS zdroj [23].	13
Obr. 1.3	Architektura zabezpečení Android zdroj [7].	14
Obr. 1.4	Architektura zabezpečení Windows Phone zdroj [7].	15
Obr. 1.5	Procentuální průřezem trhu s platformami v roce 2016 zdroj [27]. . .	15
Obr. 1.6	Zastoupení na trhu s platformami v roce 2014 zdroj [33].	16
Obr. 1.7	Zastoupení na trhu s platformami v roce 2016 zdroj [27].	16
Obr. 1.8	Obrázek popisující počáteční ustanovení spojení [26].	22
Obr. 1.9	Deset mobilních rizik podle OWASP v roce 2016 [47], [48].	24
Obr. 2.1	Obrázek popisující možné kombinace stránek pro docílení základní funkcionality navigace. [51].	29
Obr. 2.2	Obrázek popisující fungování MasterDetailPage [51].	29
Obr. 2.3	Obrázek popisující fungování TabbedPage [51].	31
Obr. 2.4	Obrázek popisující možné kombinace rozvržení na stránce [51]. . . .	33
Obr. 2.5	Obrázek popisující návrhový vzor Model-View-ViewModel [36]. . . .	41
Obr. 3.1	Obrázek znázorňující průběh hashovací funkce [53].	44
Obr. 3.2	Obrázek znázorňující kolizi v hashovací funkci [53].	45
Obr. 3.3	Obrázek popisující dělení hashovacích funkcí a závislostní mezi nimi [53].	46
Obr. 3.4	Obrázek popisuje zpracování jednoho bitového bloku pro SHA-2 s provedením 80 kol [53].	48
Obr. 3.5	Obrázek popisuje zpracování jednoho 512bitového bloku na 64 kol, kdy používá čtyři základní funkce pro hashování, označované jako F, G, H, I [53].	49
Obr. 4.1	Obrázek popisuje základní komunikaci pro ustanovení klíčů založenou na Diffie-Hellman algoritmu [1].	51
Obr. 5.1	Obrázek popisuje strukturu bloku v Bitcoinech [46].	54
Obr. 5.2	Obrázek popisuje strukturu transakce v Bitcoinech [46].	55
Obr. 5.3	Obrázek popisuje strukturu Melker stromu v Bitcoinech [46].	56
Obr. 5.4	Obrázek popisuje strukturu kořene vytvářeného z Melker stromu v Bitcoinech [46].	56
Obr. 7.1	Obrázek popisuje strukturu prototypové aplikace Bitcoin Wallet na Windows Phone [53].	61
Obr. 7.2	Obrázek popisuje strukturu použitý návrhový vzor MVVM v prototypové aplikaci Bitcoin Wallet [53].	62

Obr. 8.1	Obrázek ukazuje návrh nové aplikace, která bude postavena na frameworku Xamarin s hlavním a zrychleným menu.	65
Obr. 8.2	Obrázek navrhuje další vrtstvy realizace v prostředí Xamarin, konkrétně platební transakce a kontaktní adresář.	65
Obr. 8.3	Obrázek navrhuje další vrtstvy realizace v prostředí Xamarin, konkrétně platební transakce a kontaktní adresář.	66
Obr. 8.4	Obrázek navrhuje další vrtstvy realizace v prostředí Xamarin, konkrétně platební transakce a kontaktní adresář.	66
Obr. 9.1	Obrázek zobrazující celkové rozvržení 14 projektů na soubě závislých částí se směřovaný komunikačním kanálem.	68
Obr. 9.2	Obrázek zachycuje šest hlavních projektových částí, které zajišťují hlavní funkční jádro aplikace v Xamarinu.	69
Obr. 9.3	Obrázek zobrazuje implementaci pomocných tříd, modelů a tříd pro komunikaci se serverem.	70
Obr. 9.4	Obrázek zobrazující implementační projekt pro databázi SQLite. . .	70
Obr. 9.5	Obrázek zobrazující modelovou implementací pro databázové tabulky SQLite.	71
Obr. 9.6	Obrázek popisující kompletní řešení XAML tříd pro zobrazení neboli pohled.	71
Obr. 10.1	Obrázek zobrazuje implementaci přihlašování v levé části je videt prototypovou aplikace a nesleduje Windows Phone, UWP a platforma Adnroid.	74
Obr. 10.2	Obrázek zobrazuje implementaci hlavního menu v levé části je videt prototypovou aplikace a nesleduje Windows Phone, UWP a platforma Adnroid.	75
Obr. 10.3	Obrázek zobrazuje implementaci rychlého postraního menu v levé části je videt prototypovou aplikace a nesleduje Windows Phone, UWP a platforma Adnroid.	75
Obr. 10.4	Obrázek zobrazuje implementaci nabídky pro zadání transakce v levé části je videt prototypovou aplikace a nesleduje Windows Phone, UWP a platforma Adnroid.	76

SEZNAM PŘÍLOH

P I. CD ROM

PŘÍLOHA P I. CD ROM

Obsahem CD jsou:

1. zdrojové kódy aplikace v jazyce C# bez balíčku i s balíčky.
2. instruktážní videa pro každou platformu včetně prototypové aplikace.
3. zdrojové kódy L^AT_EXu textové části práce.
4. technická dokumentace vygenerované v rámci programového kódu.
5. diplomová práce v elektronické podobě ve formátu *.pdf.