Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Ekonomická fakulta

Katedra aplikované matematiky a informatiky

Bakalářská práce

Vývoj mobilní aplikace pro Windows Phone a její publikování ve Windows Phone Store

Vypracoval: Radek Němec

Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2017

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta ekonomická
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Radek NĚMEC**
Osobní číslo:	**E14376**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Název tématu:	**Vývoj mobilní aplikace pro Windows Phone a její publikování ve Windows Phone Store**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování:

Cílem práce je vytvořit mobilní aplikaci a provést její distribuci pomocí elektronického obchodu s aplikacemi.

Metodický postup:

1. Studium odborné literatury.
2. Publikace výsledků rešerše.
3. Návrh a popis vývoje a implementace výsledné aplikace, umístění do veřejnosti přístupného e-shopu.
4. Zhodnocení použitelnosti aplikace pro nasazení v reálném prostředí.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **40 - 50 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **Falafel software. (2013).** *Pro Windows Phone App Development.* **3. vydání. New York, USA: APress.**
2. **Vaughan, D. (2013).** *Windows Phone 8 Unleashed.* **Indianapolis, IN, USA: Sams.**
3. **Kotler, P. T., Armstrong, G. (2015).** *Principles of Marketing.* **16. vydání. Upper Saddle River, NJ, USA: Prentice Hall.**
4. **Troelesen, A., Japikse, P. (2015).** *C# 6.0 and the .NET 4.6 Framework.* **7. vydání. New York, USA: Apress.**
5. **Nathan, A. (2013).** *WPF 4.5 Unleashed.* **Indianapolis, IN, USA: Sams.**

Vedoucí bakalářské práce: **Mgr. Radim Remeš**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **15. ledna 2016**

Termín odevzdání bakalářské práce: **14. dubna 2017**

doc. Ing. Ladislav Rolínek, Ph.D.
děkan

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
L.S.

prof. RNDr. Pavel Tlustý, CSc.
vedoucí katedry

V Českých Budějovicích dne 31. března 2016

Prohlášení

Prohlašuji, že svoji bakalářskou práci na téma „Vývoj mobilní aplikace pro Windows Phone a její publikování ve Windows Phone Store“ jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47 zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to – v nezkrácené podobě/v úpravě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

..…… ……..

Datum Podpis

Poděkování

Děkuji vedoucímu bakalářské práce Mgr. Radimovi Remešovi za metodické vedení a cenné rady při vypracování bakalářské práce.

Obsah

1 ÚVOD ... 3

1.1 CÍL PRÁCE ... 3

2. PŘEHLED ŘEŠENÉ PROBLEMATIKY ... 4

2.1 VÝVOJ MOBILNÍCH SYSTÉMŮ OD MICROSOFTU A JEJICH BUDOUCNOST ... 4

2.1.1 Windows Mobile ... 4

2.1.2 Windows Phone 7 ... 5

2.1.3 Windows Phone 8 ... 5

2.1.4 Windows Phone 8.1 ... 6

2.1.5 Windows 10 ... 7

2.1.6 Podíl mobilních systémů na trhu ... 8

2.2 VÝVOJÁŘSKÉ NÁSTROJE A OSTATNÍ TECHNOLOGIE ... 9

2.2.1 Programovací jazyk ... 9

2.2.2 Vývojové prostředí ... 9

2.2.3 Windows Phone 8.1 Software Development Kit ... 10

2.2.4 Windows Presentation Foundation ... 11

2.2.5 Extensible application markup language ... 11

2.2.6 WinRT XAML Toolkit ... 12

2.3 FILOZOFIE DESIGNU ... 13

2.3.1 Šablony designu ... 14

2.4 ŽIVOTNÍ CYKLUS APLIKACE ... 17

2.4.1 Stav NotRunning ... 17

2.4.2 Stav Running ... 17

2.4.3 Stav Suspended ... 18

2.4.4 Přechodový stav Resuming ... 18

2.4.5 Ukončení aplikace ... 19

2.5 DISTRIBUCE ... 20

2.5.1 Vývojářský účet ... 20

2.5.2 Windows Store ... 20

2.5.3 Podmínky distribuce ... 21

3 METODIKA ... 23

4 ŘEŠENÍ A VÝSLEDKY 24
4.1. VÝBĚR APLIKACE PRO VÝVOJ 24
4.1.1 Analýza prostředí 24
4.1.2 Výběr aplikace pro vývoj 24
4.2 DESIGN APLIKACE 25
4.2.1 Návrh designu 25
4.2.2 Globální soubor 25
4.2.3 Aktuální kurzy 26
4.2.4 Převod měn 27
4.2.5 Graf 28
4.2.6 Ostatní stránky 28
4.3 APLIKAČNÍ LOGIKA 29
4.3.1 Získání dat ze stránek ČNB 29
4.3.2 Třída Kurzy 29
4.3.3 Třída List_nastaveni 32
4.3.4 Třída Graf 34
4.3.5 Background Agent 35
4.3.5 Metoda na zjištění internetového připojení 37
4.3.6 Progress Control 37
4.3.7 Převod měny 38
4.3.8 Aplikační lišta 39
4.3.9 Ukládání nastavení 41
4.3.10 Dotaz na ukončení aplikace 41
4.4 DISTRIBUCE APLIKACE 43
4.4.1 Sestavení balíčku 43
4.4.2 Odeslání balíčku 43
4.4.3 Přehled verzí aplikace 44
4.4.4 Zhodnocení úspěšnosti 45
5. ZÁVĚR 47
I SUMMARY A KEYWORDS 48
II SEZNAM POUŽITÝCH ZDROJŮ 49
III SEZNAM OBRÁZKŮ 52

IV SEZNAM ZDROJOVÝCH KÓDŮ 53
V SEZNAM GRAFŮ 54
VI SEZNAM TABULEK 55
VII SEZNAM PŘÍLOH 56
VIII PŘÍLOHY 57

1 Úvod

Lidský život je čím dál více spjatý s informačními technologiemi a internetem. Co bylo před pár lety pro některé lidi jako sci-fi, dnes již můžeme považovat za každodenní záležitost. Téměř každý člověk vlastní chytrý telefon (smartphone), který se dá v dnešní době sehnat opravdu za velmi slušnou cenu. S tímto rozvojem se samozřejmě postupně vyvíjely i jednotlivé mobilní systémy, z nichž některé jsou v dnešní době nejsilnějšími hráči na trhu mobilních systémů. Úspěch těchto systémů je dán jednak zčásti samotným systémem, ale větším a rozhodujícím faktorem je počet aplikací, které jsou v oficiálních obchodech. S tím samozřejmě souvisí i vývoj aplikací pro jednotlivé systémy. Je naprosto jasné, že většina vývojářů, ale i společností bude vyvíjet aplikace na těch platformách, které jsou na trhu nejsilnější, aby se jejich aplikace co nejvíce stahovaly. Vývojáři tak proto dávají v dnešní době přednost hlavně systémům Android a IOS. Pokud tyto systémy budou mít solidní aplikaci, která bude hodně stahovaná, mají vývojáři šanci vydělat nemalé peníze. Pak se na trhu nachází mobilní systém Windows Phone, který sice nemá takový podíl jako předchozí systémy, ale stále má velký potenciál a je jen otázkou času, až se Microsoft s tímto systémem dostane opět do popředí. Jelikož jsem velký fanoušek tohoto systému a jsem jeho aktivním uživatelem už od počátků, tak jsem si zvolil právě tento systém, pro který jsem vytvořil aplikaci.

1.1 Cíl práce

Cílem práce je popsat, jak probíhá vývoj aplikace pro systém Windows Phone 8.1 a její následné publikování v oficiálním obchodě Windows Store. Čtenáři budou postupně seznámeni s jednotlivými generacemi mobilních systémů od Microsoftu, poté budou vysvětleny obecná doporučení, postupy a technologie, které by měl vývojář znát k použití při samotném vývoji aplikace. Poté je popsáno, jak probíhá distribuce aplikace skrze web Windows Phone Dev Center a jaké podmínky musí aplikace splňovat, aby prošla certifikačním procesem. Nakonec jsou v práci zobrazeny a vysvětleny některé části kódu aplikace v objektově orientovaném jazyce C#. Závěrem je zhodnocena úspěšnost vytvořené aplikace.

2. Přehled řešené problematiky

2.1 Vývoj mobilních systémů od Microsoftu a jejich budoucnost

2.1.1 Windows Mobile

Windows Mobile byl představen v roce 2000. Tato verze se nazývala Pocket PC 2000. Z počátku zařízení s tímto systémem nemělo integrovaný telefonní modul, ale následně byla jeho podpora přidána, stejně jako rozdělení na přístroje s dotykovou a bezdotykovou obrazovkou. První verze s Windows Mobile se po grafické stránce nijak neodlišovaly od operačních systémů na počítačích v té době používaných – Windows 98, 2000 a XP (Havryluk, 2010).

V průběhu několika let následovaly další verze systému Windows Mobile (Pocket PC 2002 a WM ve verzích 2003, 2003 SE, 5.0, 6, 6.1 a 6.5), které se rozdělily do tří základních kategorií, a to na Smartphone, Pocket PC a Pocket PC Phone Edition (Obrázek 1). Tento systém je založený na Windows CE a obsahuje hybridní jádro (Havryluk, 2010). V roce 2007 to byl nejpopulárnější systém ve Spojených státech, ale kvůli konkurenci jiných mobilních systémů (IOS, Android) začal ztrácet podíl na trhu, a tak se Microsoft rozhodl ukončit vývoj a začal vyvíjet nový mobilní systém.

Obrázek č. 1: Domovská obrazovka Windows Mobile 6.5

Zdroj: Vlastní zpracování

2.1.2 Windows Phone 7

V roce 2010 Microsoft představil zcela nový mobilní systém, který byl nástupcem slavné verze Windows Mobile. Tento systém, nazývaný též dlaždicový, přinesl z pohledu uživatele zcela revoluční design. Systém je založený na novém uživatelském rozhraní Metro, kde se domovská obrazovka skládá z živých dlaždic, které se aktualizují a ukazují podrobné informace (Obrázek 2). Například dlaždice pro emailový účet ukazuje počet nepřečtených zpráv. Rychlost systému v porovnání s ostatními byla také na jiné úrovni. V konkurenci s ostatními byl Windows Phone jednoznačně nejrychlejší. Rychlost a design byly hlavními argumenty proto, aby se Microsoft stal opět konkurenceschopným vůči ostatním mobilním systémům na trhu (Windows Phone, 2015).

Windows Phone 7 je jediná verze Windows Phone, která má jádro založené na Windows Embedded Compact 7 verze Windows Embedded CE. V průběhu vývoje následovaly další verze této vývojové generace Nodo, Mango, Refresh, a WP 7.8 (Windows Phone, 2017).

Obrázek č. 2: Domovská obrazovka Windows Phone 7

Zdroj: Vlastní zpracování

2.1.3 Windows Phone 8

Je druhá generace mobilního systému Windows Phone od Microsoftu. Tento systém byl uveden na trh v roce 2012 a stejně jako předchozí verze používá rozhraní Metro.

Windows Phone 8 už nepoužívá architekturu Windows CE, ale je postaven na přizpůsobeném jádře ze systému Windows NT, které je využíváno ve Windows 8 a v jiných systémech. Mezi hlavní novinky patřila podpora MicroSD karet, Internet Explorer 10, Multitasking na pozadí a podpora pro vícejádrové procesory. (Pultzner & Pospíšil, 2012)

Tato generace systému se utvářela podle přání a požadavků uživatelů předchozí generace Windows Phone. Microsoft si tedy vzal zpětnou vazbu od uživatelů k srdci a snažil se systém vylepšit tak, aby co nejvíce konkuroval ostatním systémům. Během vývoje této generace byly vydány aktualizace s označením GDR1, GDR2, GDR3 a každá z těchto aktualizací přinášela důležitá vylepšení a nové funkce (Aktualizace Windows Phone, 2016).

2.1.4 Windows Phone 8.1

Je v pořadí třetí a zároveň poslední generace z řady Windows Phone. Tato generace byla uvedena na trh v roce 2014 a přinesla řadu nových funkcí. Z pohledu funkcí byly nejvýznamnějšími změnami personální asistentka Cortana a notifikační centrum. Microsoft ale také výrazně změnil design. Bylo možné přidat na úvodní obrazovku tři sloupce dlaždic namísto předchozích dvou. Nejzásadnější změnou v designu pak bylo nastavení vlastní tapety pod dlaždice (Obrázek 3). Tímto krokem tak Microsoft opět umožnil uživateli přizpůsobit systém jeho požadavkům. (Rubino, 2014).

U této generace byla vydána pouze jedna větší aktualizace s kódovým označením Denim. Pak se Microsoft rozhodl ukončit mobilní generaci Windows Phone, a proto byl tento systém nahrazen Windows 10 (Aktualizace Windows Phone, 2016).

Obrázek č. 3: Domovská obrazovka Windows Phone 8.1
Zdroj: Vlastní zpracování

2.1.5 Windows 10

Tento systém byl představen v roce 2015 a podle záměrů Microsoftu se zdá, že se opět jedná o přelomový počin. Vize Microsoftu spočívá v tom, že chce vytvořit jeden systém, který bude určený pro stolní počítače, notebooky, tablety, chytré telefony a další zařízení. V praxi to tedy znamená, že jeden systém bude schopný běžet na kterémkoli zařízení na stejném jádře, ale s nutnou modifikací. S další novinkou, kterou Windows 10 přišel, jsou univerzální aplikace. Vývojářům se tak otevřela možnost vytvořit aplikaci, která poběží na mobilech a na počítačích současně (Vrbacký, 2016). Poslední novinkou, která byla z pohledu vývoje také zcela novou záležitostí, byl program Windows Insider. Členové tohoto programu mohli získat vývojové větve systému, který se utvářel podle zpětné vazby a používání od samotných uživatelů. Členové mohli zvolit ze dvou možností dostávání aktualizací. První možnost byl Fast ring (rychlý kruh), který obsahuje verze systému, které prošly interním testováním, ale obsahují mnoho chyb a nemusí být stabilní. Druhou volbou byl Slow ring (pomalý kruh), který naopak obsahuje verze systému, které jsou stabilnější a vhodnější pro každodenní používání (Lukeš, 2016).

Z hlediska mobilů se Windows 10 až na pár změn v designu příliš nezměnil od předchozích verzí Windows Phone (Obrázek 4). Bylo přepracováno nastavení a Microsoft provedl výrazný zásah v logice systému. Neexistují už huby a centra určená pro konkrétní druh obsahu, ale systém je složen z jednotlivých systémových aplikací. Touto změnou tak Microsoft může vydávat dílčí aktualizace rychleji, protože nemusí aktualizovat celý systém, ale pouze systémové aplikace. (Vrbacký, 2016).

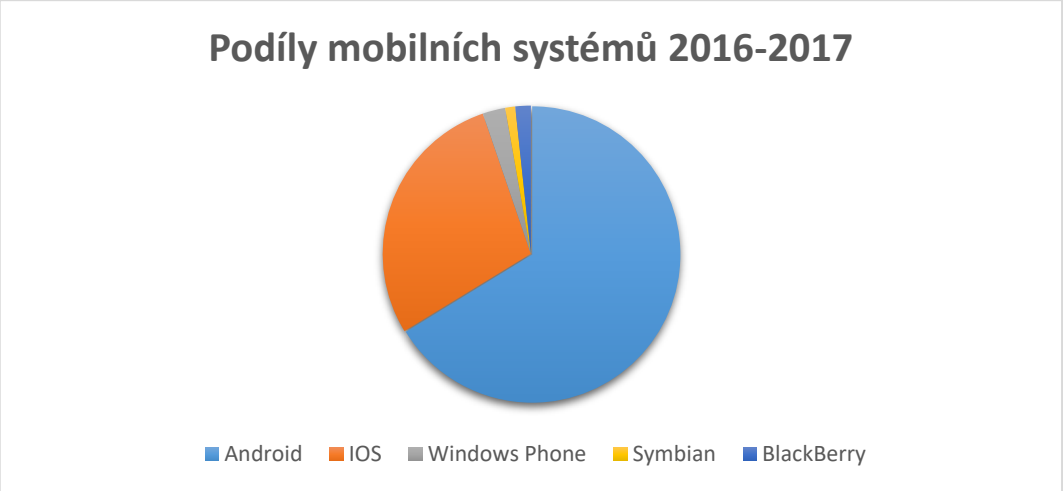
Obrázek č. 4: Domovská obrazovka Windows 10
Zdroj: Vlastní zpracování

2.1.6 Podíl mobilních systémů na trhu

V současné době zaujímá největší podíl na trhu systém Android. Důvodů, proč tomu tak je, je určitě mnoho. Nicméně rozhodující je velký počet aplikací na Google Play a široká škála nejrůznějších smartphonů všech cenových hladin. Na druhém místě je systém IOS. Jeho úspěch je založen také na velkém počtu aplikací v APP store a také na dobré pověsti firmy. Tyto dva systémy jsou tedy v dnešní době nejsilnější hráči na poli mobilních systémů a soupeří mezi sebou o to, kdo bude mít větší podíl na trhu. (Mikudík, 2016).

Na třetím místě se nachází Windows Phone. Jeví se to jako velmi neočekáváné vzhledem k tomu, že Microsoft přišel s novým uživatelským rozhraním Metro, které se v počátcích Windows Phone zdálo velmi revoluční. Bohužel, nedostatek aplikací ve Windows Store a neochota výrobců vyrábět telefony s tímto systém, dostala systém na místo, na kterém je dnes. Je velmi těžké odhadovat, jakou cestou se Windows bude dál ubírat. Podle strategie Microsoftu se stále s Windows pro mobily počítá a díky univerzálním aplikacím a novému zařízení, které Microsoft chystá pod označením Surface Phone, by nakonec mohl získat zpět místo na trhu, které měl s Windows Mobile (Hrma, 2016; Pupík, 2016).

Ostatní systémy jako například Symbian a BlackBerry jsou postupně vytlačovány a dá se předpokládat, že jejich podíl bude i nadále klesat.



Graf 1: Podíl mobilních systémů v letech 2016 až 2017

Zdroj: Vlastní zpracování

2.2 Vývojářské nástroje a ostatní technologie

2.2.1 Programovací jazyk

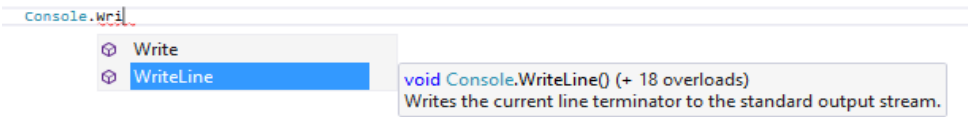
Při vývoji aplikací pro Windows Phone je možné využít dva objektově orientované programovací jazyky C# a Visual Basic .NET. Nejčastější volbou vývojářů je však C#, který jsem také využil při vývoji aplikace.

C# byl představen s integrovaným .NET Framework v roce 2002 a byl předpokladem nabídnout mnohem silnější, pružnější a jednodušší programovací model. Postupem času se tento jazyk stal velmi oblíbeným a Microsoft pokračoval v jeho dalším vývoji. S poměrně velkou změnou přišel Microsoft, když představil .Net Framework 4.5. Tato změna se projevila v C# ve zjednodušení asynchronního programování. Přibyla klíčová slova async a await, která zajistila zachování odezvy aplikace tím, že pokud aplikace stahuje data z internetu, tak nemusí uživatel čekat, než se stáhnou, ale může v klidu pokračovat v používání aplikace (Troelesen & Japikse, 2015).

2.2.2 Vývojové prostředí

Microsoft Visual Studio je integrované vývojové prostředí od Microsoftu. První verze tohoto populárního prostředí byla představena v roce 1995 a postupem času vzniklo mnoho verzí. Největší průlom přišel v roce 2002, kdy Visual Studio začalo využívat .NET Framework. Aktuální verze nese označení Visual Studio 2015 a podporuje vývoj pro různé platformy mobilních zařízení. Díky mobilnímu rozhraní Xamarin tak vývojář může najednou vytvořit jednu aplikaci pro systémy IOS, Android a Windows 10 (Snell & Powers, 2015).

Obliba Visual Studia spočívá v editoru kódu, který automaticky dokončuje a zvýrazňuje syntaxe pro proměnné, metody, ale i cykly za použití Intellisense. V praxi to tedy znamená, že programátor nemusí znát všechny názvy proměnných, protože mu to napoví Intellisense. Je to tedy jakýsi našeptávač, který umí procházet knihovny .NET Frameworku, ale také umí našeptávat i ze zdrojových kódů projektu (Obrázek 5). Díky této věci se tak velmi usnadňuje práce programátora, který se pak může plně soustředit na svoji práci (Žůrek, 2015).



Obrázek č. 5: Ukázka našeptávání včetně popisku metody Zdroj: Vlastní zpracování

2.2.3 Windows Phone 8.1 Software Development Kit

Je sada vývojových nástrojů, které umožňují vytváření aplikací. Základní instalace Visual Studia 2015 neobsahuje potřebné nástroje pro vývoj a je tedy nutné stáhnout je ze stránek Microsoftu. Microsoft tedy nabízí jednotlivé SDK bezplatně pro jednotlivé verze mobilních systémů (Falafel software, 2013).

Součástí vývojových nástrojů jsou i emulátory, na kterých probíhá testování (Obrázek 6). SDK obsahuje širokou paletu emulátorů, které se liší jednak rozlišením displeje, dále také velikostí operační paměti, aby co nejvěrohodněji simulovaly reálné zařízení. Testování nemusí ale probíhat pouze na emulátorech, a tak je zde možnost pomocí nástroje Windows Phone Developer Registration odemknout svůj telefon pro účely testování. Podmínkou je však vlastnit vývojářský účet u Microsoftu (Windows Software Development Kit (SDK), 2013).

Další velmi užitečnou věcí, která se nainstaluje společně s vývojářskými nástroji, je aplikace Windows Certification Kit. Tuto aplikaci je vhodné použít před odesláním aplikace do obchodu Windows Store. Otestuje totiž samotnou aplikaci testy, které jsou skoro totožné s testy, které se provádějí při certifikaci. Pokud by aplikace neprošla těmito testy, tak je zbytečné zkoušet odesílat aplikaci k certifikaci, protože by byla určitě zamítnuta. Proto se tato aplikace hojně využívá před odesláním aplikací. (Windows App Certification Kit, 2013).

Obrázek č. 6: Emulátor pro testování aplikací

Zdroj: Vlastní zpracování

2.2.4 Windows Presentation Foundation

Jedná se o přední technologii Microsoftu, určenou pro vytváření designu aplikací. Představuje rozhraní, které slouží pro návrh a zobrazování uživatelského prostředí. Je nástupcem starší technologie Windows Forms. A oproti předchozí verzi se snaží WPF poskytnout sadu komponent v základním vzhledu operačního systému, kdy má vývojář možnost si vzhled jednotlivých komponent změnit podle svého uvážení. Dokonce si vývojář může vytvářet i své vlastní komponenty. To předchozí verze neumožňovala a nabízela komponenty se vzhledem, který nebyl možný měnit. Tato technologie tedy umožňuje vytvářet graficky složitější aplikace a vývojářům dává větší možnosti při vytváření vzhledu aplikace. (Nathan, 2013; Jecha, 2012).

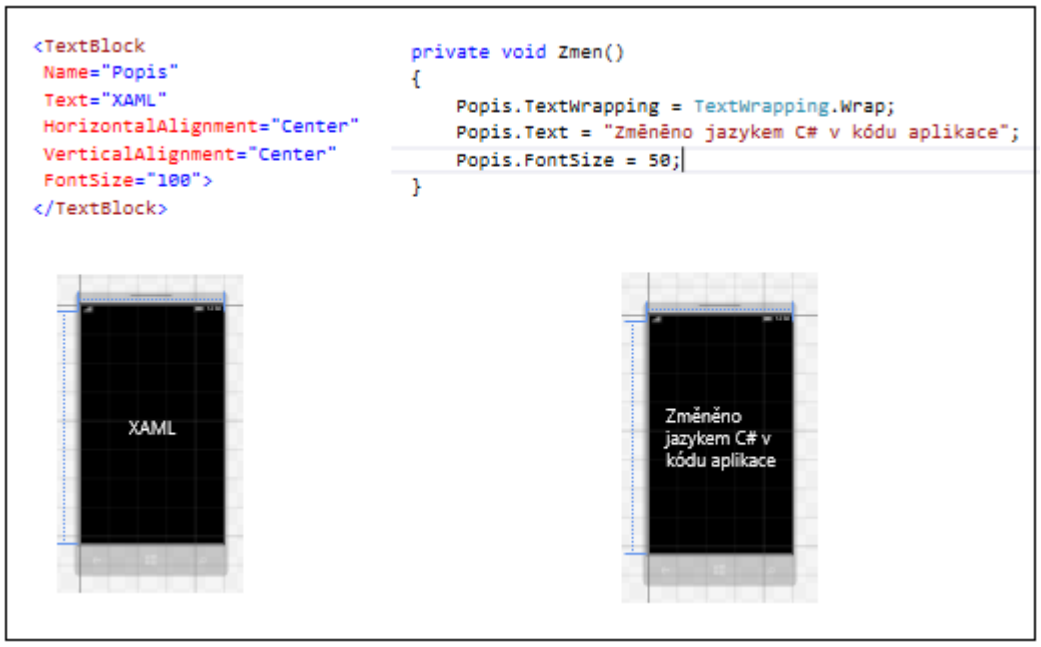
WPF nabízí celkově lepší objektový model, protože u elementů existuje celá řada nových událostí, vlastností a parametrů. A vývojář tak může upravovat, přidávat a reagovat na chování jednotlivých elementů. I co se týče využívání stylů WPF, opět zjednodušil práci vývojářům. Vývojář si může nadefinovat styl například pro nadpisy. Díky této věci může velmi jednoduše změnit vzhled všech nadpisů v rámci celé aplikace. Poslední a významnou skutečností je fakt, že WPF je založen na vektorové transformaci. To znamená, že vzhled výsledné aplikace je nezávislý na rozlišení zobrazovacího zařízení (Jecha, 2012).

2.2.5 Extensible application markup language

Je deklarativní jazyk založený na XML. Primárně slouží k navrhování uživatelského prostředí pro technologie WPF a Silverlight. XAML pracuje s objektovým modelem .NET. To znamená, že tento jazyk využívá komponenty z .NET Framework a také i z jiné knihovny, na kterou se ze XAML souboru odkážeme pomocí XML jmenného prostoru (Jecha, 2012).

Výhoda využití XAML spočívá v oddělení vzhledu od logické vrstvy aplikace. V praxi to znamená, že v jedné části se deklaruje vzhled aplikace, ve druhé se řeší, jak jednotlivé ovládací prvky mají fungovat. Dříve se uživatelské prostředí, ale i chování ovládacích prvků definovalo přímo na jednom místě. Nevýhoda pak byla v tom, že to nebylo moc přehledné a vykreslování uživatelského prostředí probíhalo až za běhu aplikace. Tento problém vyřešil XAML. Pokud se tímto jazykem definuje uživatelské prostředí, tak je již za běhu aplikace kód načtený a dojde tedy pouze k zobrazení (Přehled XAML, 2016; Nathan, 2014).

Došlo sice k oddělení definování vzhledu od logické vrstvy aplikace, ale stále má vývojář možnost přistoupit k jednotlivým ovládacím prvkům i z logické vrstvy. Důvod je poměrně prostý. Pokud budeme chtít například nastavit v aplikaci zviditelnění tlačítka v případě splnění nějaké podmínky, je nutné to provést přímo v kódu aplikace. Další příklad tohoto využití demonstruje (Obrázek 7).



Obrázek č. 7: Definování tagu v jazyce XAML a v kódu jazykem C#

Zdroj: Vlastní zpracování

2.2.6 WinRT XAML Toolkit

Představuje sadu ovládacích prvků, které rozšiřují a přidávají pomocné třídy pro aplikace Windows Runtime XAML. Tento projekt není přímo řízený Microsoftem, ale stojí za ním zaměstnanci Microsoftu. Jedná se o knihovny, které se dají do projektu prostřednictvím Nuget Manageru ve Visual Studiu přidat. Tyto knihovny pak přidají další ovládací prvky a celkově poskytnou designérovi větší výběr a možnosti vytvářet co nejlepší vzhled aplikace (WinRT XAML Toolkit, 2017).

Jako jedna z mála knihoven, které se dají instalovat bezplatně, poskytuje velmi kvalitní a pestrou nabídku grafů. Přináší podporu sloupcových, koláčových, bodových a mnoho dalších grafů. Graf stačí nadefinovat v XAML jazyce a poté stačí předat potřebná data, aby mohlo dojít k vykreslení. Velmi zjednodušuje práci programátora

a designéra, a proto se tato knihovna stává čím dál oblíbenější. Také se neustále vyvíjí a je dostupná pro všechny generace mobilních systémů od Microsoftu.

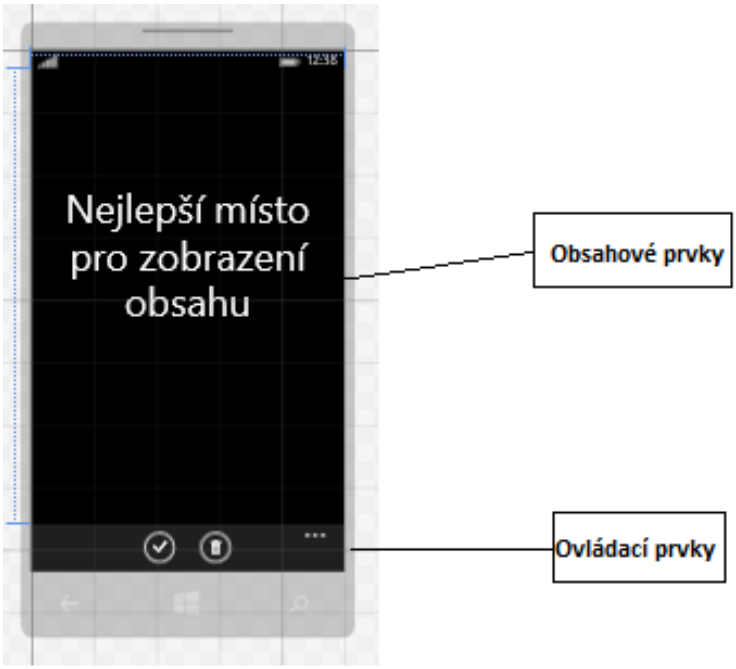
2.3 Filozofie Designu

Při vývoji aplikace by vývojář měl vycházet z designu, který je typický pro systém. Ať už budeme vyvíjet aplikace pro Android nebo Windows Phone, vždycky je důležité, aby bylo na první pohled jasné, že se jedná o aplikaci, která byla vytvořena pro určitý systém. Vývojář by se tak měl řídit určitými zásadami při navrhování designu, aby pro uživatele aplikace nebyla neznámým prostředím, ale aby se v ní uživatel mohl jednoduše zorientovat díky získaným návykům ze systému.

Windows Phone se liší od předchozí generace systémů hlavně v tom, že v aplikacích nejsou záhlaví okna a žádné tlačítko pro ukončení. Důvodů, proč k těmto opatřením došlo, je několik. Microsoft chtěl maximálně využít displej zařízení, protože v aplikacích Windows Mobile záhlaví okna a orámování aplikace zabíralo příliš velkou část vzhledu. Navíc Windows Phone je postaven na uživatelském prostředí Metro, a proto se samozřejmě musel změnit i design aplikací, aby korespondoval s novým systémem (Lacko, 2014).

U Windows Phone je možné měnit motiv celého systému. Uživatel si může vybrat tmavý nebo světlý motiv. Dále může měnit barvu dlaždic, která nepřináší jen změny v celém systému, ale i v aplikacích. Z pohledu vývojáře je nutné na tuto skutečnost pamatovat, protože se mnohdy stává, že při zvoleném světlém motivu není design aplikace na tuto skutečnost uzpůsobený a může se stát, že aplikace neprojde certifikačním procesem při publikování na Windows Store.

Při vytváření uživatelského prostředí by se také nemělo zapomínat na ovládání. Windows Phone je systém, který je určený pro dotykové displeje, a proto je vhodné řídit se doporučeními, které říkají, kde mají být obsahové a ovládací prvky (Obrázek 8). Ovládací prvky by spíše měly být ve spodní části aplikace, aby na ně uživatel mohl dosáhnout palcem, který bývá nejčastěji používán. Obsahové prvky by se pak měly nacházet ve střední až horní části aplikace, aby při využívání ovládacích prvků nebyly obsahové prvky palcem, či jinou části ruky zakryty (Lacko, 2014).



Obrázek č. 8: Správné rozložení obsahových a ovládacích prvků

Zdroj: Vlastní zpracování

2.3.1 Šablony designu

Sada vývojářských nástrojů nabízí tři základní šablony (Pivot, Panorama a Hub). Každá z těchto šablon má své specifické vlastnosti a je vhodná pro určitý typ aplikací. Microsoft se snažil každou šablonu uzpůsobit hlavně z hlediska usnadnění práce vývojářům a také vytvoření jednotného vzhledu všech aplikací.

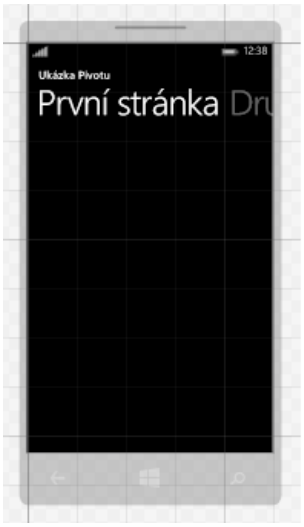
Pivot

Tato šablona se skládá z libovolného počtu stránek, které jsou vložené vedle sebe a mezi stránkami se prochází gestem – posunu doprava či doleva. Stránky se za sebou řadí v závislosti na pořadí deklarace a v hlavičce aplikace jsou vidět názvy jednotlivých stránek, aby uživatel věděl, jaké stránky se v aplikaci nachází (Lacko, 2014).

Z praktického hlediska je Pivot nejvyužívanější šablonou pro vývoj aplikací. Používá se především pro práci se seznamy, ale hojně je využíván také v aplikacích, kde každá stránka reprezentuje jednotlivé funkcionality dané aplikace.

Z hlediska deklarace se v jazyce XAML využívá element Pivot. Nejčastěji se v tomto elementu přidává atribut Title, který slouží jako nadpis aplikace. Pro přidání stránek se

využívá element s názvem PivotItem, který slouží k definování jednotlivých stránek. Do tohoto elementu se pak přidává atribut Header, který reprezentuje nadpis stránky (Vaughan, 2013; Falafel software, 2013). Názorná ukázka vytvoření Pivotu je pak znázorněna na (Obrázek 9).




```
<Pivot Title="Ukázka Pivotu">
<PivotItem Header="První stránka"></PivotItem>
<PivotItem Header="Druhá stránka"></PivotItem>
</Pivot>
```

Obrázek č. 9: Ukázka šablony Pivot v emulátoru a v deklaraci v jazyce XAML

Zdroj: Vlastní zpracování

Panorama

Možnost jak získat větší zobrazovací oblast nabízí tento typ šablony. Vzhled aplikace pak využívá virtuální plochu, která je na šířku podstatně větší, než je velikost samotného displeje. Displej se pak stává jakýmsi výřezem, který ukazuje aktuální oblast virtuální plochy. Aby uživatel věděl, že existují další obrazovky, je obraz rozdělen tak, aby v pravé části obrazovky byl vidět kousek začátku další obrazovky. Díky této šabloně se tak odstraňují omezené možnosti menších velikostí displejů (Lacko, 2014).

V jazyce XAML se pro deklaraci využívá element Panorama. Název aplikace je možné nastavit pomocí atributu Title. Do elementu Panorama se pak prostřednictvím PanoramaItem přidávají jednotlivé obrazovky. V elementu PanoramaItem se pomocí atributu Header nastavuje nadpis obrazovky. A Panorama.Background slouží k nastavení pozadí (Vaughan, 2013; Falafel software, 2013). Názorná ukázka vytvoření Panoramatu je pak znázorněna na (Obrázek 10).

Obrázek č. 10: Ukázka šablony Panorama v emulátoru a v deklaraci v jazyce XAML

Zdroj: Vlastní zpracování

Hub

Při navrhování se v jazyce XAML používá element Hub. Pro přidání obrazovek se využívá element s názvem HubSection. U obou těchto elementů je pak možné přidat atribut Header, který definuje nadpis aplikace a obrazovky (Vaughan, 2013; Falafel software, 2013). Názorná ukázka vytvoření Hubu je znázorněna na (Obrázek 11).

Obrázek č. 11: Ukázka šablony Hub v emulátoru a v deklaraci v jazyce XAML

Zdroj: Vlastní zpracován

Tato šablona je velmi podobná šabloně Panorama. Hub využívá také virtuální plochu, která je na šířku větší než velikost displeje. Microsoft takovou šablonu využíval v systémových aplikací a nejznámější aplikace tohoto typu byly Lidé a Hry.

2.4 Životní cyklus aplikace

Než začnu popisovat jednotlivé cykly neboli stavy aplikace, je ze začátku nutné říci, co se za pojmem životní cyklus aplikace skrývá. Životní cyklus definuje, jak se aplikace chová v různých stavech a vyjadřuje, jak se chová při přechodu z jednoho stavu do druhého (Ženíšek, 2013). Microsoft životní cyklus neustále vyvíjel, a tak se v každé generaci mobilních systémů výrazně lišil od předchozích verzí. Největší změna přišla s příchodem verze systému Windows Phone 8.1. Je tu jen jedna aplikace, která běží na popředí a všechny ostatní aplikace můžou být dočasně pozastaveny, ale jsou stále načteny v operační paměti telefonu, nebo jsou násilně ukončeny v případě, kdy systém potřebuje více operační paměti.

2.4.1 Stav NotRunning

Tímto stavem se označuje situace, kdy aplikace není spuštěná. Znamená to, že není zobrazeno uživatelské prostředí a aplikace není načtena v operační paměti telefonu. Tento stav je výchozí pro všechny aplikace.

2.4.2 Stav Running

Tento stav může nastat dvěma způsoby. První možností je situace, kdy uživatel na úvodní obrazovce, či v seznamu aplikací klikne na dlaždici nebo na název aplikace. Potom dojde k zobrazení úvodní obrazovky (anglicky splash screen) aplikace, kterou uživatel vidí během inicializace hlavních úloh sloužících ke spuštění aplikace. Tato doba by měla být co nejkratší, protože Microsoft dává doporučení, že doba inicializace by neměla přesáhnout 5 sekund. Když jsou všechny úlohy dokončené, tak úvodní obrazovka zmizí a zobrazí se uživatelské prostředí aplikace a tím pádem se aplikace dostane ze stavu NotRunning na Running. Druhým způsobem, jak se aplikace dostane do tohoto stavu je v případě, že se aplikace opět spustí ze stavu Suspended. V praxi to znamená, že aplikace již byla předtím spuštěná a teď dojde opět k jejímu spuštění. Stav Running je nejčastějším stavem aplikací, protože se příslušná aplikace využívá k účelům, k jakým byla vytvořena programátorem (Lacko, 2014).

2.4.3 Stav Suspended

Uživatel mnohdy využívá možnost dostat se na domovskou obrazovku systému ze spuštěné aplikace prostřednictvím stisku hardwarového tlačítka zpět nebo domů. Z pohledu systému pak dochází k tomu, že se aplikace dostane do úsporného stavu a následně se sama zastaví. Tento stav se nazývá Suspended a aplikace je při tomto stavu sice v paměti telefonu zavedena, ale nemůže provádět žádné operace.

Když aplikace přechází ze stavu Running do stavu Suspended, nastává takzvaný přechodový stav. To je situace, kdy se aplikace dostala na pozadí, ale Microsoft to v systému nastavil tak, že systém čeká ještě 10 sekund na to, kdyby se uživatel rozhodl do aplikace vrátit. Pokud uběhne 10 sekund a uživatel se do aplikace nevrátí, systém ji automaticky přesune do úsporného stavu (Lacko, 2014).

Velmi často se stává, že uživatel má spuštěných několik aplikací a přechází mezi nimi. Existuje zde pak jistá šance, že by uživatel mohl opět zvolit příslušnou aplikaci, a proto je tento interval z pohledu uživatele, ale i systému velmi výhodný, protože během běžícího intervalu se aplikace stále bere jako by byla spuštěná. Proto pak dojde k okamžitému zobrazení aplikace, do které se uživatel vrací.

Ještě předtím, než se aplikace dostane do úsporného stavu, má aplikace 5 sekund na to, aby uložila potřebné informace, protože je pak násilně ukončena. Existuje tak událost Suspending, která je spuštěna ještě předtím, než je aplikace ukončena. V této události je pak nutné uložit informace, se kterými aplikace pracuje v danou chvíli (Lacko, 2014).

Jakmile se aplikace nachází v úsporném stavu, může dojít ke dvěma situacím. Každý telefon disponuje určitou velikostí operační paměti, která určuje počet aplikací, které budou ve stavu Suspended. Pokud v systému dojde k nedostatku alokované paměti, tak systém ukončí určitý počet aplikací právě v tomto stavu, aby získal více volné paměti. Druhou možností je, že uživatel spustí aplikace z úsporného stavu a systém využije přechodový stav Resuming.

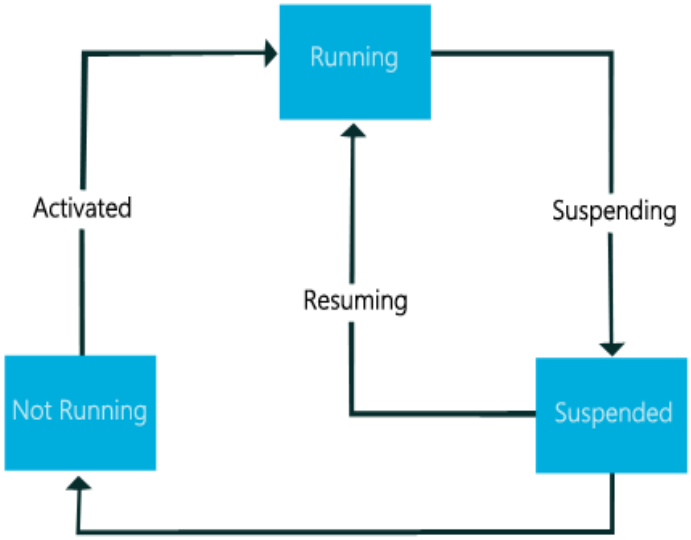
2.4.4 Přechodový stav Resuming

Tento stav se využívá při přechodu z úsporného stavu a aplikace se dostane do stavu Running. Během toho přechodu by se měly načíst informace, které se uložily v události Suspending, aby aplikace mohla pokračovat v práci tam, kde přestala předtím, než byla

pozastavena. Při opětovném spuštění se využívá událost Resuming, která zajistí načtení uložených informací (Lacko, 2014).

2.4.5 Ukončení aplikace

Jedná se o poslední cyklus, který je z větší části řízený systémem a uživatelé se tak nemusí starat o ukončování aplikací. Systém samozřejmě nabízí možnost i samotnému uživateli ukončovat příslušné aplikace gesty, které systém nabízí. Aplikace i v případě ukončení uživatelem nejdříve přejde do stavu Suspended, kdy si aplikace může uložit aktuální informace, poté je ukončena a přejde do stavu NotRunning (Lacko, 2014).



Obrázek č. 12: Grafické znázornění životního cyklu aplikací
Zdroj: (Manage app life cycle and state (XAML), 2012)

2.5 Distribuce

2.5.1 Vývojářský účet

Pokud chce vývojář své aplikace publikovat ve Windows Store, je nutné si vytvořit vývojářský účet. Pro jeho vytvoření stačí mít vytvořený účet u Microsoftu, který je možné používat najednou v tabletu, telefonu nebo počítači s Windows, Xbox Live, webu Outlook.com nebo OneDrive. K aktivaci vývojářského účtu pak stačí zaplatit jednorázový poplatek.

Microsoft se v roce 2014 snažil změnit strategii a změnil podmínky tak, aby přilákali nové vývojáře. Výše poplatku se snížila z původních 99 $ na docela příjemných 19 $ a hlavní změnou bylo zrušení ročního předplatného. To znamená, že při zřizování vývojářského účtu stačí poplatek zaplatit pouze jednou a účet je pak předplacen na doživotí. Microsoft tak zrušil často kritizovaný poplatek, který vývojář musel platit i tehdy, když žádnou aplikaci v tomto období nepublikoval. Zároveň když by tento poplatek nebyl zaplacen, tak by byly odstraněny všechny aplikace, které měl vývojář vystavené ve Windows Store (Mrázek, 2014).

Protože cílem Microsoftu bylo získat hodně nových vývojářů, tak chtěl podpořit ve vývoji i studenty, kteří byli členy programu DreamSpark. Tento program umožňuje studentům i vyučujícím bezplatné stažení nástrojů společnosti Microsoft nejen pro vývojáře a návrháře. Členové v rámci tohoto programu měli bezplatný vývojářský účet. Bohužel, v roce 2016 Microsoft změnil systém benefitů programu DreamSpark a bezplatný vývojářský účet byl nakonec odstraněn. Proto i studenti, kteří chtějí publikovat své aplikace, musí pro získání vývojářského účtu zaplatit jednorázový poplatek (DreamSpark, 2016).

2.5.2 Windows Store

Windows Store je obchod s digitálním obsahem od společnosti Microsoft. Je využíván jako distribuční kanál v mobilních a počítačových systémech, tabletech a na konzolích Xbox One. V České republice z něj můžou uživatelé stahovat pouze aplikace a hry, v zahraničí je navíc možné stahovat filmy, hudbu a seriály (Windows Store, 2017). Je to také jediný oficiální zdroj aplikací a her, které musí odpovídat podmínkám a požadavkům Microsoftu.

Donedávna existovaly dva distribuční kanály, z nichž jeden byl určený pro mobilní zařízení a druhý pro zařízení počítačové. S příchodem Windows 10 došlo ke sjednocení těchto obchodů a vznikl jednotný obchod, který je společný pro celou platformu s různými zařízeními. Součásti tohoto obchodu tak jsou nové univerzální aplikace napsané přímo pro Windows 10. Do Windows Storu byly současně převedeny také kompletně všechny Windows 8.x a Windows Phone 8 aplikace. Všechny aplikace jsou z hlediska funkcionality a ceny rozděleny do kategorií, aby zjednodušily uživateli jejich samotné hledání a výběr. Aby měl uživatel přehled o tom, jaké aplikace vlastní, existuje ve Windows Store záložka s názvem Moje knihovna. Ta zobrazuje seznam vlastněných aplikací, které uživatel získal buď zadarmo nebo nákupem (Pultzner, 2015).

Ve Windows Store je možné buď distribuovat volně dostupné aplikace, které si mohou lidé bezplatně stáhnout nebo placené aplikace. Placené aplikace může vývojář publikovat s časovým nebo funkčním omezením. Když vývojář vybere časové omezení, tak je aplikace plně funkční pouze po dobu, jakou zvolil. Tím se tak usnadňuje práce vývojáře, který časové omezení nemusí implementovat do aplikace, ale Windows Store to vyřeší za něj. Funkční omezení pak představuje omezené používání aplikace. Například ve hrách se nachází jenom omezený počet úrovní, v aplikacích jsou dostupné pouze základní funkce a ostatní jsou skryty apod. (Lacko, 2014).

2.5.3 Podmínky distribuce

Microsoft stanovil určité podmínky a doporučení, které je nutné splnit, aby aplikace mohla bez problému projít certifikačním procesem při publikování ve Windows Store. Tyto podmínky mají za cíl udržet určitou kvalitu aplikací a bezpečnost z pohledu uživatele. Takže každá aplikace musí přinášet určitý význam a funkcionalitu. Z hlediska bezpečnosti se pak testuje, zda aplikace vykonává činnosti, pro které byla vytvořena a jestli neprobíhají na pozadí aplikace ještě jiné procesy, které uživatel nemůže v žádném případě zjistit. Aplikace by například mohla shromažďovat informace o vlastníkovi a následně je zneužít. I proto kvůli této hrozbě Microsoft varuje a výrazně doporučuje uživatelům, aby instalovali aplikace pouze z Windows Store.

Často kontrolovanou věcí je také přizpůsobitelnost vzhledu aplikace, kdy uživatel může vybrat místo tmavého vzhledu vzhled světlý. Někdy se stane, že když má uživatel vybraný právě tento motiv, tak se vzhled aplikace stává nepřehledný a tím pádem

i použitelnost aplikace. Tato věc je tedy také velmi častým důvodem, proč aplikace neprojde certifikačním procesem.

Aplikace dále nesmí obhajovat diskriminaci, nenávist či násilí založené na etnických, národnostních, rasových, náboženských či jiných společenských skupinách, nebo na pohlaví, věku a sexuální orientaci jedince. Nesmí obsahovat obsah nebo funkčnost, která povzbuzuje, usnadňuje a idealizuje ilegální aktivitu. Aplikace nesmí působit urážlivě a pomlouvačně, nebo něčím hrozit. Je také zakázáno aplikací jakkoliv podporovat nebo idealizovat nadměrné a nezodpovědné užívání alkoholu, tabákových výrobků, drog a zbraní. Obsah aplikace také nesmí povzbuzovat k bezdůvodnému násilí, nebo k porušení základních lidských práv a svobod (Windows Store, 2017, Smlouva s vývojářem aplikací, 2016). Tyto podmínky jsou spíše obecné, které neplatí jenom pro Windows Store, ale i pro jiné platformy. Jsou tedy obecně závazné a všichni vývojáři je musí dodržovat. Společnosti, které spravují virtuální obchody by měly řádně hlídat, jestli jsou tyto podmínky dodržovány.

Aktuální podmínky Microsoftu se dají vyhledat na stránkách Windows Dev Center a jsou rozděleny do různých skupin z hlediska použitelnosti, bezpečnosti apod. Zde je 6 základních kategorií podmínek:

1. Aplikace pro Windows Store musí přinášet zákazníkům přidanou hodnotu.
2. Aplikace pro Windows Store může zobrazovat reklamy, ale musí poskytovat víc než jen zobrazení reklamy či zobrazení obsahu webové stránky.
3. Aplikace pro Windows Store se musí chovat předvídatelným způsobem.
4. Aplikace pro Windows Store jsou řízeny uživatelem.
5. Aplikace pro Windows Store musí být vhodné ke globálnímu nasazení.
6. Aplikace pro Windows Store musí být snadno rozpoznatelné a pochopitelné.

(Lacko, 2014)

3 Metodika

Pro vývoj aplikace bylo nutné získat potřebný software, který je nezbytný pro vývoj a který většinou nebývá zadarmo. Protože jsem ale studentem Jihočeské univerzity, tak mám možnost díky spolupráci Microsoftu a univerzity stahovat většinu produktů zadarmo. Tím jsem tak získal bezplatně operační systém a vývojové prostředí. Po instalaci tohoto softwaru jsem ještě musel stáhnout sadu vývojářských nástrojů pro vývoj aplikací pro platformu Windows Phone 8.1, které jsem po stažení nainstaloval. Tyto nástroje byly automaticky přidány do Visual Studia. Tímto krokem jsem tak měl vše potřebné pro samotný vývoj.

Protože je možné aplikace pro Windows Phone vyvíjet různými programovacími jazyky, tak jsem si musel vybrat jazyk, ve kterém budu aplikaci vyvíjet. Rozhodl jsem se pro C#, protože je to moderní objektově orientovaný jazyk, který jsem se navíc naučil na Jihočeské univerzitě. Do C# je zároveň integrovaný .Net Framework a jeho největší výhodou jsou knihovny, které usnadňují vývojářům práci např. s databázemi, formulářovými prvky apod.

Jelikož jsem neměl vůbec žádné zkušenosti s vývojem aplikací na této platformě, tak jsem před začátkem vývoje potřeboval zjistit, jak se má správně vytvářet vzhled aplikace a jak se určité věci programují. K tomu mi pomohlo nejen studium odborné literatury, ale také virtuální akademie od Microsoftu. Ta obsahuje několik desítek dílů, ve kterých programátoři názorně ukazují a vysvětlují, jak se jednotlivé věci programují a jak se mají správně využívat ovládací elementy. Díky shlédnutí všech dílů jsem tak získal potřebné zkušenosti a poté jsem se pustil do samotného vývoje aplikace.

Při vývoji je nutné využít několik ovládacích elementů (tagů):

- Texblock – slouží pro zobrazení malého množství informací
- Textbox – textové pole, umožňuje uživateli zadávat text
- ListView – zobrazuje seznam položek
- Grid - mřížka, která se skládá ze sloupců a řádků a slouží pro zarovnání elementů
- Combox – výběrové pole, zobrazuje data v rozevíracím seznamu
- DatePicker – umožňuje uživateli vybrat datumu

4 Řešení a výsledky

Hlavním úkolem je tedy vytvořit aplikaci, která bude umět stahovat data z internetu a bude s těmito daty dále pracovat. Aplikace má zjednodušit práci uživatelům, kteří pak nemusí navštěvovat stránky poskytovatele dat, protože jim potřebná data poskytne vytvořená aplikace.

4.1. Výběr aplikace pro vývoj

4.1.1 Analýza prostředí

Před začátkem vývoje aplikace jsem se musel rozhodnout, jakou aplikaci budu vyvíjet a jaký bude její hlavní význam. Protože jsem neměl žádnou konkrétní představu, tak jsem zjišťoval, jaké aplikace uživatelům Windows Phone chybí a jaké aplikace nesplňují přesně jejich požadavky. Ke zjištění mi pomohlo fórum webu Smartmania na adrese https://smartmania.cz/forum/viewforum.php?f=144, kde byla sekce náměty na aplikace a zde jsem našel potřebné informace. Díky těmto informacím jsem tak už získal základní přehled o tom, kterou aplikaci bych mohl začít vytvářet. Dále jsem sledoval web Wmmania na adrese https://wmmania.cz/clanky/, kde redaktoři přinášely informace o nejlepších aplikacích a jejich objektivní hodnocení. To mi pomohlo v tom, že jsem věděl, jaké aplikace je pro mě zbytečné vyvíjet, protože prosazení proti těmto velmi povedeným aplikacím by bylo velmi těžké. Těmito skutečnostmi jsem si tak vytvořil základní přehled o aplikacích, které se ve Windows Store vyskytují a u kterých je šance zabodovat kvůli menší kvalitě a konkurenci.

4.1.2 Výběr aplikace pro vývoj

Díky cenným informacím, které jsem získal z analýzy prostředí, jsem měl několik vytipovaných aplikací pro vývoj. Přemýšlel jsem o aplikaci zásilky, která by uživatelům umožnila sledovat jejich zásilky nejznámějších dopravců. Dále jsem měl představu o televizním programu, který by poskytoval seznam aktuálně běžících programů v televizi. A poslední aplikací byla měnová kalkulačka. Z analýzy prostředí jsem však věděl, že už na Windows Store existuje velmi povedená aplikace zásilky a že se její autor neustále snaží aplikaci vylepšovat. Proto jsem od této volby upustil a měl jsem na výběr mezi televizním programem a měnovou kalkulačkou. Shodou okolností jsem už předtím vyvíjel okenní aplikaci, která stahovala kurzy ze stránek České národní banky. Právě tato skutečnost a to, že jsem věděl, že získání dat je velmi jednoduché, byly pro mě pádným

důvodem k vybrání této aplikace. Navíc jsem ohledně televizních programů zjistil, že by nebylo vůbec jednoduché získat potřebná data, tak jsem se rozhodl pro měnovou kalkulačku. Důvod, proč jsem si vybral právě tuto aplikaci, nebyl daný jen jednoduchým získáním dat, ale z analýzy prostředí jsem zjistil, že na Windows Store sice nějaké aplikace toho typu jsou, ale nenabízejí takové funkce a design, které bych si představoval. Navíc většina těchto aplikací se prodávala na poměry ve Windows Store za docela vysokou cenu a já jsem doufal a věřil, že můžu vytvořit aplikaci, která by mohla být v mnoha ohledech lepší než konkurenční aplikace.

Stanovil jsem si tři základní funkce, které měnová kalkulačka bude nabízet. Aplikace bude zobrazovat aktuální kurzy, dále bude možné provádět převod mezi vybranými měnami a bude zobrazovat graf vybrané měny podle vybraného období.

4.2 Design aplikace

4.2.1 Návrh designu

Pro návrh designu aplikace jsem vybral jednu z přednastavených šablon s názvem Pivot. Tuto šablonu jsem si vybral proto, protože aplikace budou nabízet tři základní funkce a myslím si, že je vhodné, aby každá stránka reprezentovala jednu z těchto funkcí. Samotný design jsem navrhoval v jazyce XAML v takzvaném okně návrháře, ve kterém je prostředí Blend, kde je možné sledovat aktuální uživatelské rozhraní aplikace.

Před vytvářením designu jsem musel založit ve Visual Studiu nový projekt, do kterého jsem přidal šablonu Pivot. Do tohoto výchozího Pivotu jsem pak přidal elementem PivotItem tři stránky, které reprezentují funkce aplikace. První se jmenuje aktuální kurzy, druhá převod měn a třetí graf. Tyto Pivoty se nachází na stránce s názvem MainPage a z hlediska uživatele je to nejpoužívanější část aplikace. Do projektu jsem musel ještě přidat další stránky například pro nastavení, informace o aplikaci a zpětné vazby od uživatelů. Proto se v projektu setkáme se stránkami s názvem (Uprava_historie, Nastaveni, O_aplikaci, Chyby_namety, Historie, Historie2). Tyto stránky pak reprezentují další části aplikace, které slouží především k nastavení aplikace.

4.2.2 Globální soubor

V každém projektu se automaticky vytvoří soubor s názvem App.xaml. Tento soubor se označuje jako globální a obsahuje kód, který je přístupný v rámci celé aplikace. V tomto souboru se nejčastěji definují statické zdroje jako například styly jednotlivých

tagů a podobně. V tomto souboru jsem definoval název aplikace, protože bude používán i na jiných stránkách aplikace. Výhoda tohoto použití spočívá v tom, že kdybych se rozhodl změnit název aplikace, tak jednoduchou editací kódu v tomto souboru změním název v rámci celé aplikace. Název aplikace v podstatě představuje jakousi statickou proměnnou a definuje se následujícím způsobem:

```
<Application.Resources>
    <x:String x:Key="Nazev_aplikace">Měnová kalkulačka</x:String>
</Application.Resources>
```

Ukázka kódu 1 Vytvoření statického zdroje názvu aplikace

Když pak budeme chtít tento název využít, tak stačí v XAML souboru na místo, kde ho budeme potřebovat napsat následující kód:

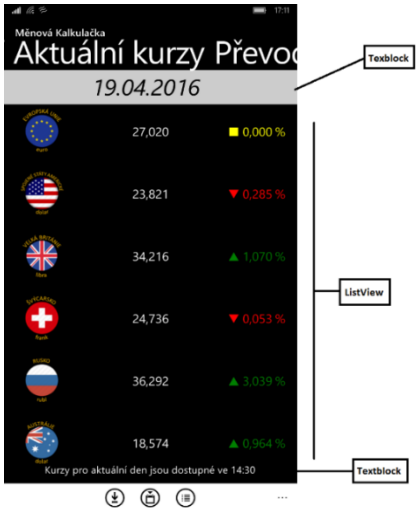
```
<Pivot Title="{StaticResource Nazev_aplikace}">
```

Ukázka kódu 2 Praktické využití zdroje při definování atributu Tile v Pivotu

Můžeme vidět, že klíčovým slovem StaticResource vlastně říkáme, ať je to hledáno ve statických zdrojích v souboru App.xaml. Nazev_aplikace pak slouží k identifikaci zdroje.

4.2.3 Aktuální kurzy

Tento Pivot je výchozí obrazovkou aplikace. Při navrhování jsem vycházel z toho, že nebude možné zobrazit všechny měny najednou, a tak jsem využil tag Listview. To umožní uživateli, aby mohl procházet seznam měn rolováním. Ve vrchní části stránky se nachází TextBox, který jsem upravil a nastavil tak, aby fungoval jako výběr data, pro který chce uživatel vidět kurzy. Ve spodní části jsem ještě definoval tag Texblock, který slouží k informování uživatele o tom, že kurzy jsou pro aktuální den dostupné ve 14:30. Názorné složení designu je vidět na (Obrázek 13).

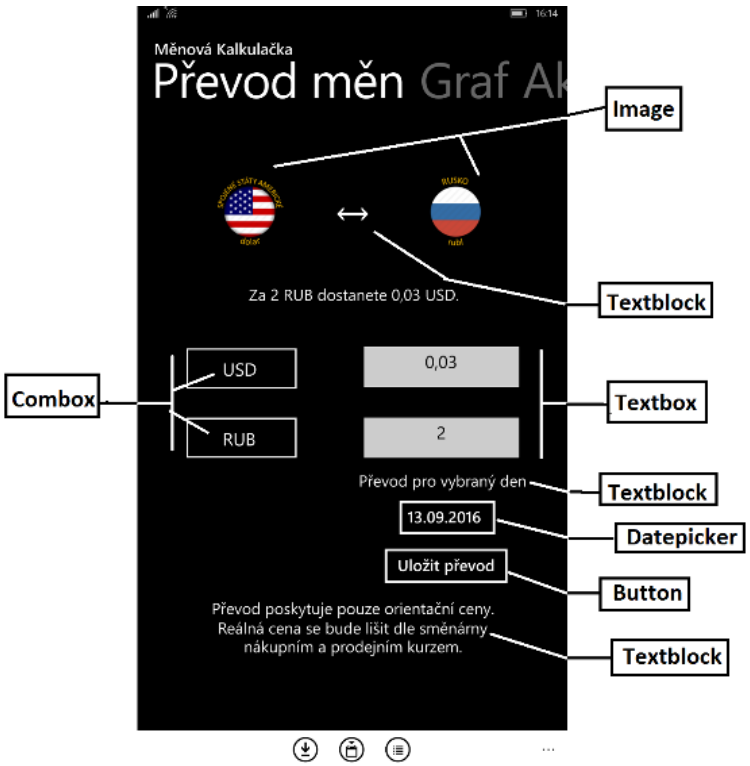


Obrázek č. 13: Obrazovka aktuální kurzy

Zdroj: Vlastní zpracování

4.2.4 Převod měn

Tato stránka slouží k převodu měn mezi vybranými měnami. Ve vrchní části se nacházejí dva tagy Image, které se mění podle výběru měny a mezi nimi je zobrazena ikonka šipky, která značí, že je možné provádět převod oboustranně. Pod touto části se nachází Texblock, který se objevuje, když dochází k převodům a informuje uživatele z jaké měny a v jakém množství na jakou měnu, byl převod uskutečněn. Ve střední části na levé straně jsem definoval dva Comboxy, které obsahují měny, které má aplikace k dispozici a slouží pro výběr požadované měny. V pravé části jsou pak dva tagy Textbox, do kterých uživatel zapisuje množství vybrané měny. Aby se zamezilo neočekávaným chybám, tak je zde nastaven atribut InputScope na číselnou klávesnici, aby uživatel nemohl množství měny zapsat jinak než číselně. Pod Textboxy se dále nachází Datepicker, který slouží pro výběr data, se kterým chce uživatel provádět převod. A nakonec můžeme vidět tlačítko s názvem uložit převod a v dolní části stránky ještě poslední tag Texblock, který informuje uživatele o tom, že převody jsou pouze orientační. Názorné složení designu je vidět na (Obrázek 14).

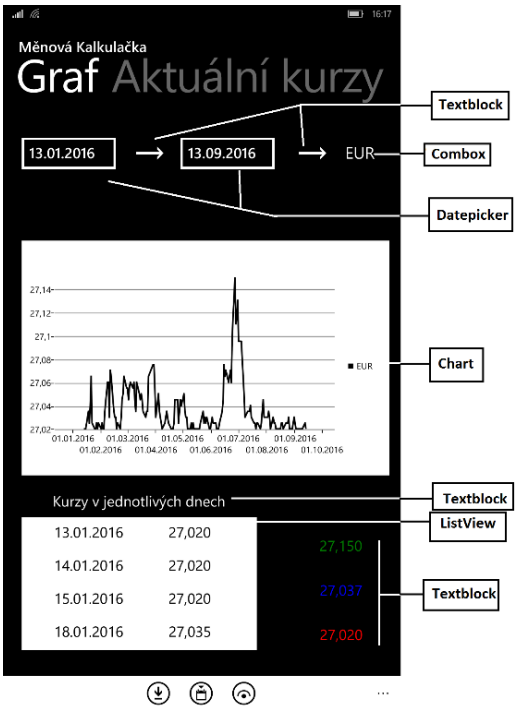


Obrázek č. 14: Obrazovka převodu měn

Zdroj: Vlastní zpracování

4.2.5 Graf

Poslední stránkou hlavního souboru MainPage je stránka Graf. Tato stránka obsahuje Datepickery, které slouží ke zvolení období, které má být zobrazeno v grafu. Pak se zde nachází jeden Combox, aby bylo možné zvolit měnu, pro kterou chceme vybrané období zobrazit. Ve střední části se pak zobrazuje samotný graf, který je přidaný z knihovny WinRT XAML Toolkit. Ve spodní části je tag Listview, který obsahuje všechny kurzy za zvolené období. A nakonec v pravé části dole jsou ještě tři Texblocky, které zobrazují maximální, minimální a průměrný kurz za vybrané období. Názorné složení designu je vidět na (Obrázek 15).



Obrázek č. 15: Obrazovka grafu

Zdroj: Vlastní zpracování

4.2.6 Ostatní stránky

Ostatní stránky byly navrženy obdobným způsobem a popis jednotlivých stránek by přesahoval rámec této práce. V další části práce tedy budou pouze popsány některé z důležitých funkcí, které plní právě tyto stránky a zejména stránka týkající se nastavení aplikace.

4.3 Aplikační logika

4.3.1 Získání dat ze stránek ČNB

Česká národní banka poskytuje devizové kurzy a odpovídají tomu, jak se s jednotlivými měnami obchodovalo na mezibankovním devizovém trhu. Tyto kurzy se používají hlavně pro účely účetnictví, k ohodnocování závazků a pohledávek a také v celním řízení apod. Slouží tedy pouze pro neobchodní účely.

Kurzy jsou poskytovány ve třech výstupních formátech a tato data je možné získat v podobě HTML, Excelu a v textu. Já jsem si vybral data v textu, protože pro následnou práci s daty to byl nejlepší formát. Důležitou skutečností je také to, že se kurzy vyhlašují každý den ve 14:30, vyjma státních svátků a nepracovních dnů, kdy se používá kurz z předešlého dne. Tato věc byla velmi zásadní a během implementace aplikační logiky jsem tuto skutečnost musel řádně ošetřit. ČNB dále nabízí buď aktuální burzovní lístek, kde jsou aktuální kurzy všech měn, nebo poskytuje databázi kurzů, kde si uživatel může vybrat měnu a časové období, za které chce vidět všechny kurzy (Obrázek 16). Protože jsem chtěl, aby aplikace pracovala i v případě nedostupnosti internetového připojení s určitým počtem dat, tak jsem využil tuto databázi. Při prvním spuštění aplikace dochází ke stažení kurzů k jednotlivým měnám od roku 2013 až do roku současného.

```
Měna: EUR|Množství: 1
Datum|Kurz
02.01.2017|27,020
03.01.2017|27,020
04.01.2017|27,020
05.01.2017|27,020
06.01.2017|27,020
09.01.2017|27,020
10.01.2017|27,020
11.01.2017|27,020
12.01.2017|27,020
13.01.2017|27,020
16.01.2017|27,020
17.01.2017|27,020
18.01.2017|27,020
```

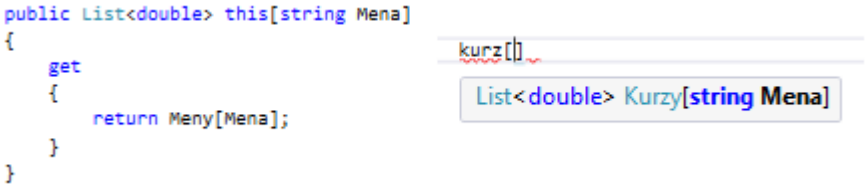
Obrázek č. 16: Ukázka dat z databáze ČNB

Zdroj: Vlastní zpracování

4.3.2 Třída Kurzy

Tato třída slouží primárně ke stažení kurzů a poskytuje další metody pro další manipulaci s kurzy. Obsahuje listy neboli kolekce, které slouží pro uchování kurzů k jednotlivým měnám a dále list datum, podle kterého zjistíme, k jakému dni se vážou

příslušné kurzy. Listy jsou privátní, aby k nim nikdo nemohl přistupovat zvenku a k listu datum je vytvořena vlastnost, která zpřístupňuje list pouze pro čtení. Pro snadnější práci s listy je pak vytvořena kolekce Dictionary, který obsahuje všechny listy měn. Aby se pak snadněji mohlo pracovat s vybraným listem měny, je ve třídě vytvořen indexer, který zpřístupňuje vybraný list měny právě z kolekce Dictionary. V indexeru je definováno pouze čtení a tím je zajištěno to, že nikdo zvenku nemůže data upravovat a jsou dodrženy základy objektově orientovaného programování.



Ukázka kódu 3 Definování indexeru a jeho následné využití

Zdroj: Vlastní zpracování

Třída obsahuje dva konstruktory, z nichž jeden je bezparametrický a slouží k inicializaci všech listů měn a data, druhý obsahuje parametr řetězec typu string. V tomto konstruktoru dochází také k inicializaci všech listů a dále se zde volá metoda pro naplnění listů z databáze, se kterou aplikace pracuje.

Pro zajištění načtení kurzů z databáze se zde nachází metoda Naplneni_listuzdatabaze, která má parametr řetězec typu string, který představuje už načtený soubor. Následně je tento parametr postupně separován, poté dochází k naplnění listů jednotlivých měn a list datum.

Metoda s názvem Stahni_kurzy_vcetnezapisu slouží ke stažení dat ze stránek ČNB. Je to asynchronní metoda, protože je velmi pravděpodobné, že vykonání metody bude probíhat delší dobu. Tím se zachová odezva aplikace a uživatel může v klidu pokračovat v jejím používání. Metodě jsem ještě přiřadil návratový typ Task, protože v případě volání je možnost před ní napsat klíčové slovo await, které říká, že v této části kódu se vyčká, až bude dokončena právě tato metoda. To umožní předejít problémům, které by vznikly, když by kurzy nebyly ještě staženy a docházelo by tak k nesprávnému nastavení jednotlivých tagů.

Stažení dat je zajištěno základní třídou HttpClient, kterou jsem přidal jmenným prostorem System.Net.Http. Tato třída umožňuje odesílání HTTP požadavků a příjem odpovědí. Aby HttpClient věděl, kde má získat odpověď, tak jsou v této metodě definována proměnná data typu string, která obsahují adresu, na které se nachází kurzy v textové podobě. Do této adresy se přidává parametr den a měna, které určují, pro jakou měnu a v jakém období mají být kurzy staženy. Postupně se pak získá obsah stránky a tento obsah je uložen do proměnné result, která je separována. Tím se získají kurzy a datum, které se pak přidají do listů.

```
string data = "http://www.cnb.cz/miranda2/m2/cs/financni_trhy/devizovy_trh/kurzy_devizoveho_trhu/vybrane.txt?mena=" + meny[i]

HttpClient client = new HttpClient();
HttpResponseMessage response = await client.GetAsync(data);
HttpContent content = response.Content;
string result = await content.ReadAsStringAsync();

char[] separator1 = new char[] { '.', '|', ':', ' ', '\n' };
string[] kurzy;

kurzy = result.Split(separator1, StringSplitOptions.RemoveEmptyEntries);
```

Ukázka kódu 4 Postup získání dat ze stránek ČNB

V rámci této metody ještě dochází při prvním spuštění aplikace k vytvoření textového souboru, který slouží jako databáze, která se při každém dalším spuštění doplňuje o stažené kurzy. Metoda je přetížená a můžeme jí předat parametr den typu datetime a řetězec typu string nebo jen samotný den. Pokud jí předáme oba dva, tak slouží ke stahování a doplňování kurzů do databáze. Pokud přidáme jen jeden, tak dochází k celkové změně databáze, protože uživatel chtěl změnit rok, od kterého chce mít uložené kurzy.

Metoda Kurz_podledata umožňuje zobrazení kurzů podle dat, které si uživatel zvolí. Tato metoda má parametr den typu datetime, se kterým se hledá shoda v listu datum. Jakmile se den rovná datu v listu, tak se získá index, který určuje, na jakém indexu se nachází potřebný kurz. A pak je zde ještě parametr typu string měna, který říká, jaký list měny se má vybrat. Protože zde může nastat možnost, že by vybraný den mohl být státním svátkem, tak jsem musel ještě vytvořit metodu, která vrací logickou hodnotu pravda nebo nepravda. Pokud je opravdu den státním svátkem, tak se den nastaví na předchozí den a opět dochází k hledání shody dne v listu datumy. To samé se děje i v případě, že se

parametr den rovná sobotě nebo neděli. I v tomto případě se den nastaví na předchozí pracovní den a opět dochází k hledání indexu, na kterém se nachází požadovaný kurz.

Poslední důležitou metodou je metoda s názvem Pridej_do_dictionary. Nejdříve se v metodě zjišťuje, jestli kolekce typu Dictionary už obsahuje nějaké listy měn. Pokud už nějaké má, tak dojde k jejich vymazání a potom se do něj opět přidají data. Do kolekce se přidávají listy tak, že se nejdříve napíše klíč, který bude sloužit k jejich identifikaci a pak se přidá samotný list měny.

```
public bool Zjisteni_svatek(DateTime den)
{
    List<DateTime> statni_svatky = new List<DateTime>() { new DateTime(den.Year, 1, 1),


    for (int i = 0; i < statni_svatky.Count; i++)
    {
        if (den == statni_svatky[i])
        {
            return true;
        }
    }
    return false;
}
```

Ukázka kódu 5 Metoda zjišťuje, zda je den státním svátkem

4.3.3 Třída List_nastaveni

Z důvodu zajištění bindování dat do List_men (ListView), který zobrazuje měny, musela být vytvořena třída, jejíž hlavním úkolem je vytvořit kolekci, která se následně předá jako zdroj dat do List_men. Třída ještě navíc obsahuje šest veřejných vlastností, které společně reprezentují jeden řádek seznamu měn na úvodní obrazovce. Vlastnosti mají nastavený zápis privátní a slouží pouze pro čtení.

Třída obsahuje dva konstruktory. První konstruktor je bezparametrický a slouží pro inicializaci listu datového typu List_nastaveni. A druhý konstruktor obsahuje 6 parametrů, které se přiřazují jednotlivým vlastnostem třídy a je využíván pro ukládání dat do listu.

Přidávání záznamů do listu má na starost metoda Nastav_list. Metoda má jeden parametr typu pole, double a tento parametr obsahuje vždy dva kurzy ke každé měně, aby se mohlo procentuálně zjistit, jak se aktuální kurz změnil oproti předchozímu. Vypočítá se tedy procentuální změna a podle této změny se pak ještě nastavuje příslušná barva

procent a symbol. Když se kurz zvýší, tak se přiřadí barva zelená a symbol reprezentující zvýšení. Pokud se sníží, tak je přiřazena barva červená a symbol reprezentující snížení. Pokud se kurz nezmění, tak jsou procenta žlutá a symbolem je obdélník. Protože se ke každé měně přiřazuje příslušná ikona, je nezbytné zde ještě definovat cestu ke složce, kde se ikony ke měnám nachází. To je velmi jednoduché, protože každá ikona je pojmenovaná podle měny a do adresy se nastaví pouze název složky a příslušná měna. Tím je zajištěno zobrazení správné ikony ke měně.

```
double procenta;
double[] result = new double[32];
int index = 0;

for (int i = 0; i < data.Length; i = i + 2)
{
    procenta = 100 - ((100 / data[i]) * data[i + 1]);
    result[index] = procenta;

    if (result[index] > 0)
    {
        barvy.Add(new SolidColorBrush(Colors.Green));
        znaky.Add("▲");
    }

    if (result[index] < 0)
    {
        barvy.Add(new SolidColorBrush(Colors.Red));
        znaky.Add("▼");
    }

    if (result[index] == 0)
    {
        barvy.Add(new SolidColorBrush(Colors.Yellow));
        znaky.Add("■");
    }
    index++;
```

Ukázka kódu 6 Zjištění, jak se změnil kurz oproti minulému a následné nastavení barvy a symbolu

Samotná metoda Nastav_list přidává měny v pořadí, které jen nastaveno defaultně. Uživatel má ale možnost si vybrat v jakém pořadí a jaké měny chce zobrazovat. Z tohoto důvodu jsem definoval metodu Setřiď_list, která se volá po vykonání kódu na konci v metodě Nastav_list. Na začátku kódu metody se zjistí, jestli uživatel změnil pořadí nebo měny, které chce zobrazovat a pokud došlo ke změně, tak je list setříděn. Pokud k žádné změně nedošlo, je metoda okamžitě ukončena klíčovým slůvkem return.

Aby bylo možné data bindovat, musíme ještě nastavit List_men, který slouží pro zobrazení kurzů na úvodní obrazovce. Proto jsem pomocí ListView.Itemtemplate přidal a nastavil tagy tak, aby korespondovaly s vlastnostmi třídy List_nastaveni a mohly být následně zobrazeny.

```
<ListView.ItemTemplate>
    <DataTemplate>
        <Grid>
            <Image Name="Ikona" HorizontalAlignment="Left"  Width="90" Source="{Binding Ikona}"></Image>
            <TextBlock x:Name="Kurz"  FontSize="22"  Text="{Binding Kurz}" TextAlignment="Center" VerticalAlignment="Center" Hori
            <TextBlock x:Name="Symbol" FontSize="22" MaxWidth="150" MinWidth="100" TextAlignment="Left" Text="{Binding Procenta}"
            <TextBlock x:Name="Symbol2" FontSize="22" TextAlignment="Right" Text="{Binding Znak}" Foreground="{Binding Barva}" Ve
        </Grid>
    </DataTemplate>
</ListView.ItemTemplate>
```

Ukázka kódu 7 Nastavení List_men na bindování dat

Bindování dat probíhá na stránce MainPage a zde se list typu List_nastaveni předá jako zdroj dat, který je potom zobrazen v tagu List_men jako seznam měn. Řádek seznamu měn je pak složen z vlastností ikona, kurz, symbol a procenta.

```
List_men.ItemsSource = List.Vrat_list;
```

Ukázka kódu 8 Předání listu měn do Listu_men

4.3.4 Třída Graf

Třída má za úkol zajistit vytvoření listu, který obsahuje data předaná jako zdroj pro LineChart. Obsahuje sedm vlastností a jeden list typu Graf. Je možné je pouze číst, protože zápis je nastaven modifikátorem přístupu na private. Třída má jeden konstruktor, který obsahuje čtyři parametry, které se přiřazují jednotlivým vlastnostem třídy. Tento konstruktor je pak využíván v metodě Nastav_listgrafu.

Metoda Nastav_listgrafu tedy slouží k naplnění listu a má čtyři parametry. První parametr představuje objekt typu Kurzy, druhý vybranou měnu a další dva parametry typu datetime reprezentují začátek a konec vybraného období, které chce uživatel zobrazit. Metoda není nijak složitá. Nejdříve se podle prvního datumu zjistí, na jakém indexu se toto datum nachází v listu datumu. Ten bude určovat začátek výběru dat. Totožná operace se provede s datumem a ten reprezentuje konec. Těmito kroky se tak přijde na to, která data z listu měny mám vybrat a následně naplnit do listu typu graf. Abychom věděli, s jakým listem měny se bude pracovat, tak do indexeru zapíšeme vybranou měnu, a tím se získá správní list. V této metodě se ještě zjišťují tři věci. Když už je známo, s jakými daty z listu měny se pracuje, tak se ještě zjišťuje maximální, minimální a průměrný kurz za dané období. To je velmi jednoduché, protože samotná kolekce list právě tyto metody nabízí, a tak stačí pro zjištění hodnot zavolat na listu příslušnou metodu.

Výsledný list je pak obdobným způsobem předán jako u List_men. List je předán jako zdroj dat pro LineChart a List_historie, který zobrazuje všechny kurzy, včetně datumů za zvolené období. U LineChart se na ose x zobrazují datumy a na ose y kurzy. Grafu stačí pouze data předat. Graf už sám vyřeší, jak dojde k jejich vykreslení. Nakonec se ještě nastaví Texblocky, které reprezentují maximální, minimální a průměrný kurz za vybrané období.

```
int index1 = kurz.Datumy.IndexOf(kurz.Zkonroluj_den(zacatek));
int index2 = kurz.Datumy.IndexOf(kurz.Zkonroluj_den(konec));
index2 = index2 - index1 + 1;

List<double> vybrany_list_meny = kurz[vybrana_mena];
List<DateTime> datumy = kurz.Datumy.GetRange(index1, index2);
vybrany_list_meny = vybrany_list_meny.GetRange(index1, index2);

for (int i = 0; i < datumy.Count; i++)
{
    list.Add(new Graf(vybrany_list_meny[i], datumy[i], datumy[i].ToString("dd.MM.yyyy"), vybrany_list_meny[i].ToString("N3")));
}

max = vybrany_list_meny.Max().ToString("N3");
min = vybrany_list_meny.Min().ToString("N3");
average = vybrany_list_meny.Average().ToString("N3");
```

Ukázka kódu 9 Naplnění listu typu Graf

4.3.5 Background Agent

Protože je velmi pravděpodobné, že uživatel nebude mít vždy připojení k internetu a bude chtít mít k dispozici aktuální kurzy, tak jsem v této aplikaci využil úlohu na pozadí. Tato úloha se stará o to, že doplňuje v čase, který si uživatel zvolí, kurzy do databáze aplikace. Uživatel se tedy nemusí o nic starat, protože vše probíhá na pozadí a uživatel si vlastně nemůže vůbec všimnout toho, že daná úloha právě probíhá.

Abych mohl přidat background agent, musel jsem do projektu aplikace přidat Windows Runtime Component, kterou jsem pojmenoval MyTask. V projektu se mi tedy vytvořil další projekt, který jsem ještě musel přidat do referencí celkového řešení projektu, aby agent fungoval. V projektu MyTask jsem pak vytvořil třídu s názvem FirstTask, která implementuje rozhraní IBackgroundTask. Protože implementuje právě toto rozhraní, vytvoří se ještě metoda s názvem Run, která bude vždycky volána při spuštění úlohy. Pak už stačilo vytvořit metodu, která zajistí stažení kurzů a uložení dat do databáze. V první části metody se tak zjistí poslední den, který databáze obsahuje a z tohoto dne se určí nadcházející den. Dále se zjišťuje, zda jsou na stránkách ČNB k dispozici nové kurzy. Pokud jsou, jsou následně uloženy. Protože je tato metoda asynchronní, tak se může stát, že background agent nemusí tuto metodu dokončit a mohl

by ji odložit. To by znamenalo, že by aktualizace neproběhla. Proto se musel nastavit takzvaný deferrral, který v podstatě neukončí úlohu na pozadí do té doby, než se dokončí všechny asynchronní metody. Tím je tedy zajištěno, že aktualizace kurzů nebude odložena a úloha bude ukončena až po jejím vykonání.

```
public sealed class FirstTask : IBackgroundTask
{
    public async void Run(IBackgroundTaskInstance taskInstance)
    {
        var deferral = taskInstance.GetDeferral();

        await Registrace_background();

        await Stahni_kurzy();

        deferral.Complete();
    }

    private async Task Registrace_background()...

    private async Task Stahni_kurzy()...
}
```

Ukázka kódu 10 Třída FirstTask včetně metody Run

S background agentem ještě souvisí nastavení, kdy se bude úloha spouštět. Microsoft nechtěl, aby tyto úlohy probíhaly v přesně stanovený čas, a tak se to musí určitým způsobem obejít. Při první registraci agentu tedy uživatel vybere čas, ve kterém mají být aktualizace prováděny. V tomto okamžiku se volá metoda, která dopočítává čas v sekundách, který je pak využíván při registraci agenta. Když se agent registruje, tak je to nastavené tak, že úloha bude spuštěná až následující den. To znamená, že metoda dopočítává čas, který představuje interval, po jehož uplynutí dojde ke spuštění úlohy následující den. Tím je vyřešeno správné proběhnutí agenta při prvním spuštění. Aby další spuštění probíhaly ve stejný čas, tak bylo nutné ve třídě MyTask ještě vytvořit metodu, která to zajistí. Tato metoda první úlohu odregistruje a zaregistruje novou úlohu se správným interval. Protože se úloha při prvním spuštění spustí ve stanovém čase, tak stačí při registraci nové úlohy zadat interval 24 hodin. Touto věcí je tak zajištěno správné spouštění agenta ve stanovém čase i při dalších spuštěních.

```
BackgroundTaskBuilder taskBuilder = new BackgroundTaskBuilder { Name = myTaskName, TaskEntryPoint = "MyTask.FirstTask" };
taskBuilder.SetTrigger(new TimeTrigger((uint)interval, false));
BackgroundTaskRegistration myFirstTask = taskBuilder.Register();
```

Ukázka kódu 11 Registrace agenta

4.3.5 Metoda na zjištění internetového připojení

Protože aplikace pro stahování kurzů potřebuje internetové připojení, je nutné před každou aktualizací kontrolovat, jestli je připojení dostupné. Proto jsem vytvořil metodu, která bude ověřovat, zda je telefon připojen k internetu. Tato metoda je součástí třídy Kurzy a má návratový typ typu bool. Pokud je internetové připojení k dispozici, tak vrací logickou hodnotu pravda a pokud není, tak vrací nepravda. Metoda je ještě typu static. To znamená, že tato metoda může být volána pouze bez instance objektu třídy Kurzy. Touto věcí tak můžu velmi pohodlně volat tuto metodu i v jiných částech aplikace, aniž bych musel vytvářet instanci objektu ke třídě Kurzy. Metoda je využívána při spuštění aplikace a podle jejího výsledku aplikace buď zjišťuje, jestli jsou dostupné nové kurzy nebo běží v off-line režimu, který využívá kurzy uložené v databázi.

```
public static bool Kontrola_Internetu()
{
    if (NetworkInterface.GetIsNetworkAvailable() == true)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Ukázka kódu 12 Metoda na kontrolu internetu

S touto metodou je ještě spojena metoda, která bude uživatele informovat při nedostupnosti o tom, že není dostupné připojení k internetu. Metoda využívá MessageDialog, který zobrazí uživateli hlášku, že není připojen k internetu. Dále je zde jednotným identifikátorem zdroje zajištěno přesunutí do nastavení systému, kde je možné zapnout připojení k internetu. Tím je uživateli ulehčeno to, že nemusí procházet nastavení a hledat bezdrátová připojení, protože je do nastavení přímo přesunut.

4.3.6 Progress Control

Aplikace při prvním spuštění zjišťuje, zda jsou dostupné nové kurzy a pokud jsou, tak dochází k jejich stahování. Tím pak může nastat situace, kdy uživatel vidí uživatelské rozhraní aplikace, ale ještě nebude načten seznam měn a bude tedy prázdný. Uživatel by pak mohl zpanikařit a nevěděl by, co se děje. Proto, aby uživatel věděl, že stále dochází ke stahování, tak je vhodné ho o tom informovat prostřednictvím Progress Control. Tento ovládací prvek je velmi užitečný, protože informuje uživatele o tom, že něco probíhá, či

v jakém stádiu se daná věc nachází. Uživatel pak ví, že se něco děje a že má čekat na její dokončení. V této aplikaci jsem využil StatusBar, který se zobrazuje ve vrchní části displeje pomocí běžících teček. Zvolil jsem ho proto, jelikož stahování bude probíhat v aplikaci několikrát, a proto je vhodné, aby byl uživatel tímto prvkem co nejméně obtěžován. Protože je StatusBar využíván při různých situacích, tak jsem pro jeho vytvoření a zpřehlednění kódu vytvořil metodu. Metoda obsahuje dva parametry, z nichž první je typu bool a druhý je typu string. První parametr bude určovat, zda se StatusBar zobrazí, nebo se skryje a druhý parametr bude informovat o průběhu operace (Obrázek 17). Tato metoda bude volána nejen před zahájením aktualizace kurzu, ale i při vykreslování grafu.

Obrázek č. 17: StatusBar zobrazující dokončení aktualizace

Zdroj: Vlastní zpracování

4.3.7 Převod měny

Jak už bylo popsáno v kapitole Návrh designu, tak převod mezi měnami probíhá ve dvou Textboxech a příslušné měny se vybírají prostřednictvím dvou Comboxů. Když se vybírají měny, tak se zde kontroluje, jestli uživatel nevybral stejné měny a pokud tomu tak je, tak je o tom informován. Další a velmi praktickou věcí je také to, že pokud se vybere jiná měna, tak dojde k automatickému převodu a uživatel už tedy nemusí dělat. U obou Comboxů je také vytvořená událost SelectionChanged. Tato událost se volá vždy, když uživatel vybere měnu a v této události je definováno nejen nastavení automatického převodu, ale také nastavení ikon v horní části, které reprezentují vybrané měny.

Samotný převod pak probíhá prostřednictvím dvou Textboxů, ve kterých je definována událost SelectionChanged. Tato událost se vyvolá vždy, když je do některého z Textboxů zapsána nějaká hodnota. V praxi to tedy znamená, že pokud uživatel zapíše částku do prvního Textboxu, tak ve druhém se automaticky zobrazí právě tato přepočítána částka ve vybrané měně. Protože je tato událost definována u každého Textboxu, tak je kód metod upravený tak, aby nedocházelo k zacyklení. Toho jsem dosáhl jednoduchou

podmínkou v každé metodě a příslušnou proměnnou typu bool. Když je proměnná nastavena na hodnotu pravda, tak to značí, že převod byl prováděn právě do tohoto Textboxu, a proto se už částka, která je v něm obsažená nepřepočítává v druhém Textboxu. To se zajistí příslušnou podmínkou na začátku obou metod, která kontroluje, jakou hodnotu proměnné mají a pokud splňují podmínku, tak se z metody odejde klíčovým slůvkem return.

Když dochází k převodu, tak se musí nejdříve zjistit s jakým kurzem se bude převod provádět. To se zjistí jednoduchou metodou, která vrací datum z tagu DatePicker. Pak už stačí zavolat metodu ze třídy Kurzy s názvem Kurz_podlemeny, která vrátí kurz k vybrané měně a pak už dochází k samotnému převodu.

4.3.8 Aplikační lišta

Protože aplikace obsahuje více stránek a metody, které budou uživateli často využívané, tak jsem přidal několik ovládacích tlačítek do spodní lišty obrazovky. Aby bylo na první pohled patrné, k jakému účelu se každé tlačítko využívá, tak se k nim přiřazují ikony, které jsou nabízeny z obrázkového fontu systému. Pro vložení tlačítek se musí vytvořit aplikační lišta tagem CommandBar a do tohoto tagu jsem postupně přidal 5 tlačítek pomocí tagu AppBarButton. Tyto tlačítka se tedy zobrazí ve spodní liště obrazovky. Protože je potřeba určitým způsobem přejít na jinou stránku aplikace, tak jsem ještě přidal druhotná tlačítka, která se zobrazí, když se klikne na tři tečky v aplikační liště. Tyto tlačítka nemají ikony a vkládají se do subtagu CommandBar.SecondaryCommands opět pomocí tagu AppBarButton. Nakonec aby se lišta zobrazila, přidá se do tagu Page.BottomAppBar.

```
<Page.BottomAppBar>
    <CommandBar Name="Spodni_nabidka" Background="White" Foreground="Black">

        <AppBarButton Name="Aktualizace" Background="Black" Label="Aktualizace"  Icon="Download" AllowDrop="True"
        <AppBarButton Name="Aktualni_kurzy" Background="Black" Foreground="Black" Label="Aktuální"  Icon="GoToToda
        <AppBarButton Name="Sdílení" Background="Black" Foreground="Black" Label="Sdílení" Icon="Mail" AllowDrop="
        <AppBarButton Name="Historie"  Label="Převody" Icon="List"   AllowDrop="True" Click="Historie_Click"/>
        <AppBarButton Name="Vykresli"  Label="Vykresli" Icon="View" AllowDrop="True" Click="Vykresli_Click"></AppB


        <CommandBar.SecondaryCommands>
            <AppBarButton x:Name="Prvni_tlacitko" Click="Prvni_tlacitko_Click" Label="Nastavení" />
            <AppBarButton x:Name="Druhe_tlacitko" Click="Druhe_tlacitko_Click" Label="Chyby a náměty"/>
            <AppBarButton x:Name="Treti_tlacitko" Click="Treti_tlacitko_Click" Label="O aplikaci"></AppBarButton>
        </CommandBar.SecondaryCommands>
    </CommandBar>
</Page.BottomAppBar>
```

Ukázka kódu 13 Definování aplikační lišty v jazyce XAML

První tlačítko s názvem Aktualizace se využívá v případě, že chceme zjistit, jestli jsou dostupné nové kurzy. Nejčastěji toto tlačítko využijí uživatelé v situaci, když např. spustí aplikaci před 14:30 a ještě nedojde ke stažení nových kurzů. Proto, aby pak nemuseli aplikaci vypnout a zapnout, slouží právě toto tlačítko, které po stisknutí zjistí, zda už jsou kurzy dostupné.

Druhé tlačítko nese název Aktuální a slouží k nastavení aktuálních kurzů a datumu. V praxi to znamená, že když uživatel chce zobrazit kurzy jiného dne a pak se chce vrátit na aktuální, tak stačí stisknout toto tlačítko, které zajistí nastavení aktuálních kurzů.

Pokud chce uživatel kurzy sdílet prostřednictvím emailu, tak může využít třetí tlačítko. Po stisknutí je uživatel přesměrován do systémové aplikace Pošta, ve kterém si vybere, z jakého emailu chce uživatel poslat kurzy. Poté se do předmětu vyplní datum, ke kterému se vážou kurzy a do obsahu se vypíšou kurzy ke všem měnám.

Čtvrté tlačítko slouží k přesměrování na stránku historie, kde jsou uložené převody, které si uživatel uložil, poslední tlačítko se využívá pro vykreslení grafu. Druhotná tlačítka pak obsahují tři tlačítka, kterou slouží pro přechod na stránky Nastavení, Chyby a náměty a O aplikaci.

Protože je patrné, že ne všechna tlačítka se dají využít na každé stránce, např. tlačítko Vykresli se využívá pouze na obrazovce Graf, tak bylo nezbytné některá tlačítka zobrazit a některá zase skrýt podle aktuální obrazovky. Proto jsem nejdříve v Pivotu využil událost SelectionChanged, která se zavolá při změně obrazovky a pomocí vlastnosti SelectIndex se zjistí, na jaké obrazovce se aplikace nachází. Pak už stačilo u každého tlačítka pomocí vlastnosti Visibility nastavit jejich zobrazení nebo skrytí.

```
switch (Pivot.SelectedIndex)
{
    case 0:
        Sdílení.Visibility = Visibility.Visible;
        Vykresli.Visibility = Visibility.Collapsed;
        Historie.Visibility = Visibility.Collapsed;
        break;

    case 1:
        Historie.Visibility = Visibility.Visible;
        Vykresli.Visibility = Visibility.Collapsed;
        Sdílení.Visibility = Visibility.Collapsed;
        break;

    case 2:
        Vykresli.Visibility = Visibility.Visible;
        Sdílení.Visibility = Visibility.Collapsed;
        Historie.Visibility = Visibility.Collapsed;
        break;
}
```

Ukázka kódu 14 Zobrazení či skrytí tlačítek podle obrazovky

4.3.9 Ukládání nastavení

Aplikace nabízí určité možnosti, jak si uživatel může nastavit aplikaci podle svých přání a požadavků. Může např. měnit pořadí, v jakém chce zobrazovat měny, či nastavit výchozí obrazovku, která se zobrazí po spuštění aplikace. Aby aplikace byla schopná si tyto věci zapamatovat, tak bylo nutné vyřešit jejich ukládání a načítaní.

Microsoft na tyto věci pamatoval a vytvořil v systému lokální úložiště dat pro každou aplikaci. K přístupu do tohoto úložiště jsem musel přidat jmenný prostor Windows.Storage, kterým jsem zpřístupnil potřebnou třídu ApplicationDataContainer, pak jsem vytvořil objekt této třídy a pomocí vlastnosti LocalSettings jsem získal přístup k nastavení aplikace. Objekt má pak ještě důležitou vlastnost s názvem Values, která obsahuje všechna uložená nastavení aplikace. K jednotlivým nastavení se pak přistoupí pomocí klíče, který obsahuje uloženou hodnotu. Z hlediska přehlednosti kódu je to velmi výhodné, protože se ukládají odlišná nastavení, a proto je pak velmi jednoduché přistoupit ke konkrétnímu nastavení určité věci.

Uložení nastavení

```
ApplicationDataContainer lokalni_nastaveni1 = ApplicationData.Current.LocalSettings;
lokalni_nastaveni1.Values["Poradi_pivotu"] = Convert.ToString(i);
```

Načtení nastavení

```
ApplicationDataContainer lokalni_nastaveni = ApplicationData.Current.LocalSettings;
object hodnota = lokalni_nastaveni.Values["Poradi_pivotu"];
```

Ukázka kódu 15 Nastavení výchozí obrazovky

4.3.10 Dotaz na ukončení aplikace

Protože Microsoft striktně nařídil, aby aplikace neobsahovaly žádné tlačítko na ukončení aplikace, tak nastal problém, jak tuto situaci vyřešit. Řada uživatelů by aplikaci ráda ukončila sama a přímo v aplikaci, než aby aplikaci musel sám systém ukončit. Proto jsem v aplikaci využil třídu HardwareButtons, která mi zpřístupnila metodu, která se volá při stisku tlačítka zpět. Následně jsem definoval dvě možnosti ukončení aplikace. Pokud se bude uživatel nacházet na výchozí obrazovce aplikace a stiskne tlačítko zpět, tak dojde k ukončení aplikace. Avšak postupem času a díky

získaným zkušenostem z používání aplikace jsem zjistil, že to není až tak šťastné řešení, protože řada uživatelů může tlačítko zpět stisknout omylem, aniž by si uvědomili, že tím dojde k ukončení aplikace. Proto jsem vytvořil druhou možnost ukončení aplikace, kterou si uživatel může zvolit v nastavení. Je to dotaz na ukončení aplikace a princip této věci je velmi jednoduchý. Pokud uživatel bude na výchozí obrazovce a stiskne tlačítko zpět, tak se aplikace okamžitě neukončí, ale zobrazí se na 2 sekundy hláška, která informuje uživatele o tom, že pokud chce aplikaci ukončit, tak musí stisknout tlačítko zpět ještě jednou a to do doby, po kterou bude hláška zobrazena. Pokud ho stiskne, tak se aplikace okamžitě ukončí a pokud ne, tak se hláška skryje a uživatel může pokračovat dál v používání aplikace.

Aby to celé mohlo fungovat, tak jsem musel vytvořit asynchronní metodu, která vytvoří na 2 sekundy hlášku a přidá jí do elementu Grid (Obrázek 18). Pak když uživatel opět stiskne tlačítko zpět, tak se zkontroluje, jestli se změnil počet potomků v elementu Grid a pokud dojde ke změně, tak je aplikace následně ukončena.

Obrázek č. 18: Dotaz na ukončení aplikace

Zdroj: Vlastní zpracování

4.4 Distribuce aplikace

4.4.1 Sestavení balíčku

Když byla aplikace hotová, tak se mohlo přistoupit k odeslání aplikace do Windows Store. Abych mohl publikovat aplikaci, tak jsem si musel zřídit vývojářský účet u Microsoftu. To bylo v podstatě jednoduché, protože už předtím jsem vlastnil účet u Microsoftu, jelikož jsem aktivním uživatelem produktů od této společnosti. Zaplatil jsem registrační poplatek, a tak se tento účet povýšil na vývojářský.

Poté se už mohlo přejít k samotnému vytvoření souboru, který se odešle na certifikaci. Vytvoření tohoto souboru probíhá ve vývojovém prostředí Visual Studio, kde se vybere v nabídce položka s názvem Store a dále vytvořit balíček. Před vytvořením je pak ještě nutné doplnit určité informace, které jsou nezbytné pro vytvoření souboru. Je potřeba rezervovat unikátní jméno aplikace, aby žádná aplikace, která se na Windows Store nachází, tento název neměla a dále ještě nastavit verzi aplikace, podporované procesory apod.

Nakonec když byl soubor úspěšně vytvořen, bylo automaticky nabídnuto aplikací Windows App Certification Kit provedení testů, které jsou obdobné s těmi, které probíhají při certifikaci na Windows Store. Proto jsem soubor nechal těmito testy zkontrolovat, abych věděl, jestli tam jsou nějaké chyby, které by určitě při samotné certifikaci byly zjištěny a bránily by v publikování. Protože jsem chtěl, aby aplikace fungovala korektně a neměla žádné chyby, tak jsem aplikaci ještě testoval v různých situacích přímo ve svém telefonu a dále mi jí pomohlo otestovat i pár přátel z mého okolí, aby už i první verze aplikace fungovala bez problémů.

4.4.2 Odeslání balíčku

Když jsem měl připravený a zkontrolovaný balíček aplikací Windows App Certification Kit, tak jsem mohl přistoupit už k finálnímu odeslání aplikace k certifikaci. Přihlásil jsem se web Windows Phone Dev Center pomocí vývojářského účtu a postupoval jsem podle pokynů. Potřebné informace, které se musí vyplnit k aplikaci, jsou rozděleny do čtyř kategorií. Vyplňují se kategorie cena, popis apod. Poté se ještě musí přidat ikona a screeny z aplikace, aby si uživatelé mohly udělat představu o tom, jak aplikace vypadá. Volitelně se může vybrat, pro jaké trhy má být aplikace dostupná a je možné nastavit, jestli má být aplikace ihned publikovaná nebo až později. Když jsou

všechny potřebné informace přidány, tak stačí přidat balíček aplikace a pak zahájit certifikaci.

Samotný certifikační proces pak probíhá v několika krocích. Některé kroky jsou dělány automaticky a některé dělá sám člověk. Tento proces může trvat i tři dny a průběh certifikace je možné sledovat na webu. Nakonec když proces přejde do stavu Published, tak to znamená, že aplikace úspěšně prošla certifikační procesem a bude do 24 hodin zveřejněna ve Windows Store, pokud jsme tedy nenastavili jiný datum publikování.

Na webu Windows Phone Dev Center Microsoft nabízí i důležité statistické informace o zablokování či zhroucení aplikace a také o využívání aplikace v jednotlivých dnech. Průměrně tak z těchto informací vyplývá, že aplikaci využívá přibližně každý den 500 lidí. Zhroucení aplikace je také dobré sledovat a zjišťovat, jaké jsou důvody pádů a reagovat na ně vydáváním aktualizací.

4.4.3 Přehled verzí aplikace

Aplikace byla uveřejněna ve Windows Store na začátku roku 2016 a od té doby vznikly 4 verze, které reagovaly na zpětnou vazbu uživatelů a přinášely celou řadu nových funkcí (Tabulka 1). První verze přinesla pouze základní funkce a chtěl jsem touto verzí zjistit, jestli o aplikaci bude zájem a případně které funkce a vylepšení se budou muset ještě do aplikace dodělat, aby byla lepší než konkurenční aplikace. Zpětnou vazbou jsem tedy získal cenné rady, jak by se aplikace měla zlepšit, a tak byla později vydána další verze, která přinesla z pohledu uživatele obrovské množství změn. Nejzásadnější změnou byla možnost změnit pořadí, v jakém se mají měny zobrazovat a dále to, že už nebyl nastaven převod pouze s českou korunou, ale mohly se provádět převody i mezi jinými měnami.

V dalších verzích už nepřibylo takové množství změn jako v té předchozí. Nicméně i tak ještě přinesly některé zajímavé funkce a také spoustu oprav. Poslední vydaná verze z mého pohledu už možná byla poslední verzí, protože si myslím, a také i díky zpětné vazbě vím, že aplikace nabízí vše, co uživatelé potřebují.

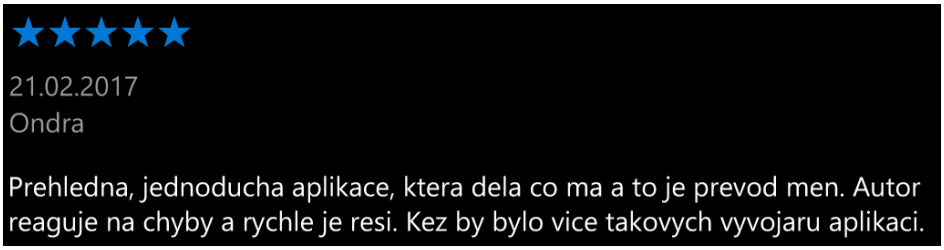
Verze	Popis změn
2.0.0.0	Změna pořadí a mazání měn, které se mají zobrazit
	Převod mezi různými měnami
	Přidána obrazovka o aplikaci
2.0.0.5	Přidány měny forint a šekel
	Oprava stahování kurzů na pozadí
2.0.0.6	Změna číslování verzí
2.0.1.0	Přidána obrazovka obecné
	Dotaz na ukončení aplikace
	Volba výchozí obrazovky

Tabulka č. 1: Přehled verzí aplikace

Zdroj: Vlastní zpracování

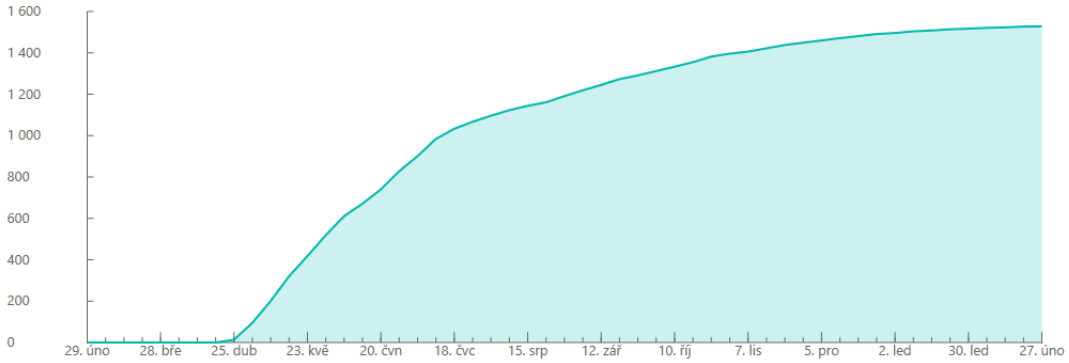
4.4.4 Zhodnocení úspěšnosti

Jak již bylo řečeno, tak aplikace byla uveřejněna v roce 2016 a za tuto dobu pokořila hranici 1500 stažení (Graf 2). Pro mě to bylo upřímně velkým překvapením, protože jsem vůbec neočekával, že by moje první aplikace mohla dosáhnout takového počtu stažení. I co se týče zpětné vazby, tak ohlasy byly vesměs pozitivní, protože aplikace získala vždy 5 hvězdiček kromě jednoho případu, kdy jsem obdržel pouze jednu hvězdu za problém, který byl okamžitě v další verzi aplikace opraven (Obrázek 18). I tak toto hodnocení pro mě bylo největší odměnou, protože si myslím, že jsem dokázal vytvořit aplikaci, která má nějaký význam a troufám si tvrdit, že v mnoha ohledech poskytuje více funkcí než konkurenční aplikace, které jsou navíc placené, zatímco moje aplikace je zcela zdarma.



Obrázek č. 19: Hodnocení aplikace jedním z uživatelů

Zdroj: Vlastní zpracování



Graf 2: Vývoj v počtu stahování aplikace v letech 2016 až 2017

Zdroj: Windows Phone Dev Center

5. Závěr

V teoretické části jsem se nejprve snažil vystihnout základní informace o jednotlivých generacích mobilních systémů od Microsoftu a snažil jsem se objektivně zhodnotit jejich současný podíl na trhu stejně jako jejich budoucnost. Poté už jsem kladl důraz hlavně na vývojářské nástroje a technologie, které se při vývoji aplikací používají. Poté byla popsána obecná doporučení a postupy při vytváření designu a tři základní šablony, které se nejčastěji využívají. Nakonec jsem se ještě věnoval životnímu cyklu aplikací, jak probíhá distribuce a jaké podmínky musí aplikace splňovat, aby bez problému prošla certifikačním procesem.

Praktická část se už věnovala čistě vývoji samotné aplikace a jejímu následnému uveřejnění ve Windows Store. V této části byly uvedeny důvody, proč jsem si zvolil zrovna tuto aplikaci a byl popsán návrh designu uživatelské rozhraní, který je vytvořen na základě doporučení Microsoftu. Poté jsem již popsal některé důležité třídy a metody v rámci aplikační logiky aplikace, které jsou nezbytné pro její bezproblémové fungování. Nakonec jsou popsány jednotlivé kroky při distribuci aplikace a zhodnocení její úspěšnosti.

Protože jsem aplikaci vyvíjel už minulý rok, tak během tohoto roku vzniklo několik verzí, které hlavně reagovaly na zpětnou vazbu uživatelů. Abych získal i větší pozornost médií a veřejnosti, že se na Windows Storu taková aplikace nachází, tak jsem vytvořil na webu Smartmania téma s jejím názvem, kde mohli lidé psát jejich připomínky a diskutovat o samotné aplikaci. Touto činností jsem tak získal hodně názorů od uživatelů a rapidně se poté zvýšil počet stahování samotné aplikace.

Samotný vývoj první verze aplikace trval zhruba 2 měsíce a kompletní kód aplikace se skládá zhruba z 5000 řádků. I přesto aplikace zabírá přibližně 5 MB úložného prostoru, což ale není moc a je možné ji nainstalovat do interní paměti telefonu nebo na SD kartu. Aplikace je sice vytvořená pro platformu Windows Phone 8.1, ale protože Microsoft uvolnil na mobily nový systém Windows 10, tak je možné ji nainstalovat i na tento systém.

I Summary a keywords

This thesis describes the mobile operating system Windows Phone 8.1 (launched in the middle of 2014) and an application development. A partial goal is to introduce the system, including all parameters, features and technologies, which are necessary for application development. The main goal of my work is to create an application that downloads exchange rates of selected currencies from the website of the Czech National Bank. The app shows current exchange rates. It is possible to convert between different currencies. Also, there is a graph of the selected currency showing exchange rate history in selected period. My app is fully customizable for every user, so they can adjust it to suit their needs. In my thesis there is a summarization of all the information necessary for introducing Windows Phone 8.1 to the beginners in application development.

Key words: windows Phone 8.1, development, application, currencies, exchange rates

II Seznam použitých zdrojů

Aktualizace Windows Phone. (2016). *Microsoft*. Dostupné z:

https://support.microsoft.com/cs-cz/help/12662/windows-phone-update-your-windows-phone

DreamSpark. (2016). *MSDN - the microsoft developer network.* Dostupné z:

https://msdn.microsoft.com/cs-cz/ee460845

Falafel software. (2013). *Pro Windows Phone App Development*. 3. vydání. New York, USA: APress. ISBN 978-1-4302-4782-1.

Hrma, J. (2016). Microsoft: S Windows Mobile máme velké plány. Pracujeme na nové generaci produktů. *Smartmania*. Dostupné z: http://smartmania.cz/microsoft-windows-10-mobile-plany-zarizeni/

Havryluk, M. (2010). Mobilní OS pro experty: Vzestup a strmý pád Windows Mobile. *Mobil.idnes*. Dostupné z: http://mobil.idnes.cz/mobilni-os-pro-experty-vzestup-a-strmy-pad-windows-mobile-pnm-/telefony.aspx?c=A100811_180849_chytre-telefony_ham

Jecha, T. (2012). Jazyk XAML. *Dotnetportal*. Dostupné z: http://www.dotnetportal.cz/clanek/198/Jazyk-XAML

Jecha, T. (2012). Úvod do Windows Presentation Foundation (WPF). *Dotnetportal*. Dostupné z: http://www.dotnetportal.cz/clanek/196/Uvod-do-Windows-Presentation-Foundation-WPF-

Lukeš, J. (2016). Program Windows Insider. *Wmmania*. Dostupné z: https://wmmania.cz/clanek/program-windows-insider-preview-zapojte-se-do-testovani-vyvojovych-verzi-systemu-windows-10-mobile-s/

LACKO, Ľ. (2014). *Vývoj aplikací pro Windows 8.1 a Windows Phone*. Brno: Computer Press. ISBN 978-80-251-3822-9.

Mikudík, R. (2016). Android s IOS drtí mobilní svět. Ostatní jsou v klinické smrti. *Mobil.idnes*. Dostupné z: http://mobil.idnes.cz/android-s-ios-drti-mobilni-svet-dmu-/mob_tech.aspx?c=A160225_194408_mob_tech_ram

Mrázek, J. (2014). Microsoft ruší roční poplatek za vývojářský účet. *Dotekomanie*. Dostupné z: http://dotekomanie.cz/2014/09/microsoft-rusi-rocni-poplatek-za-vyvojarsky-ucet/

Manage app life cycle and state (XAML). (2012). *MSDN - the microsoft developer network.* Dostupné z: https://msdn.microsoft.com/en-us/library/windows/apps/dn765019.aspx

Nathan, A. (2013). *WPF 4.5 Unleashed*. Indianapolis, IN, USA: Sams. ISBN 978-0-672-33697-3.

Nathan, A. (2015). *XAML Unleashed*. Indianapolis, IN, USA: Sams. ISBN 978-0-672-33722-2.

Přehled XAML. (2016). *MSDN - the microsoft developer network.* Dostupné z: https://msdn.microsoft.com/cs-cz/library/ms752059(v=vs.110).aspx

Pupík, A. (2016). Budoucnost Windows 10 Mobile a kde Microsoft dělá chyby? *Infoek*. Dostupné z: http://infoek.cz/budoucnost-windows-10-mobile-a-kde-microsoft-dela-chyby/

Pultzner, M. (2015). Jeden Store vládne všem. Jak to bude ve Windows 10? *Mobilenet*. Dostupné z: https://mobilenet.cz/clanky/jeden-store-vladne-vsem-jak-to-bude-ve-windows-10-19984

Pultzner, M & Pospíšil J. (2012). Vyzkoušeli jsme nové Windows Phone 8: co je nového? *Mobilenet*. Dostupné z: https://mobilenet.cz/clanky/vyzkouseli-jsme-nove-windows-phone-8-co-je-noveho-video-10459

Rubino, D. (2014). Windows Phone 8.1 Review. *Windows Central*. Dostupné z: http://www.windowscentral.com/windows-phone-81-review

Smlouva s vývojářem aplikací. (2016). *MSDN - the microsoft developer network* Dostupné z: https://msdn.microsoft.com/cs-cz/windows/apps/hh694058.aspx

Snell, M. & Powers, L. (2015). *Microsoft Visual Studio 2015 Unleashed*. Indianapolis, IN, USA: Sams. ISBN 978-0-672-33736-9.

Troelesen, A., Japikse, P. (2015). *C# 6.0 and the .NET 4.6 Framework*. 7. vydání. New York, USA: Apress. ISBN 978-1-4842-1333-9.

Vaughan, D. (2013). *Windows Phone 8 Unleashed*. Indianapolis, IN, USA: Sams. ISBN 978-0-672-33689-8.

Vrbacký, J. (2016). Windows 10 Mobile: Jeden systém vládne všem. *Mobilmania*. Dostupné z: http://www.mobilmania.cz/windows-10-mobile-jeden-system-vladne-vsem-recenze/a-1333451/default.aspx

WinRT XAML Toolkit. (2017). *GitHub*. Dostupné z: https://github.com/xyzzer/WinRTXamlToolkit

Windows Software Development Kit (SDK). (2013). *MSDN - the microsoft developer network* Dostupné z: https://developer.microsoft.com/cs-cz/windows/downloads/windows-8-1-sdk

Windows App Certification Kit. (2013). *MSDN - the microsoft developer network.* Dostupné z: https://developer.microsoft.com/en-us/windows/develop/app-certification-kit

Windows Phone. (2015). *Windows Central*. Dostupné z: http://www.windowscentral.com/windows-phone

Windows Phone. (2017). *Wikipedia*. Dostupné z: https://cs.wikipedia.org/wiki/Windows_Phone

Windows Store. (2017). *Wikipedia*. Dostupné z: https://en.wikipedia.org/wiki/Windows_Store

Ženíšek, J. (2013). *Mobilní platforma Windows Phone 8*. Dostupné z: https://www.vse.cz/vskp/eid/36139

Žůrek, M. (2015). Visual Studio – Pomocníci při psaní kódu. *Itnetwork*. Dostupné z: http://www.itnetwork.cz/csharp/visual-studio/tutorial-visual-studio-pomocnici-pri-psani-kodu

III Seznam obrázků

Obrázek č. 1: Domovská obrazovka Windows Mobile 6.5 4
Obrázek č. 2: Domovská obrazovka Windows Phone 7 5
Obrázek č. 3: Domovská obrazovka Windows Phone 8.1 6
Obrázek č. 4: Domovská obrazovka Windows 10 7
Obrázek č. 5: Ukázka našeptávání včetně popisku metody 9
Obrázek č. 6: Emulátor pro testování aplikací 10
Obrázek č. 7: Definování tagu v jazyce XAML a v kódu jazykem C# 12
Obrázek č. 8: Správné rozložení obsahových a ovládacích prvků 14
Obrázek č. 9: Ukázka šablony Pivot v emulátoru a v deklaraci v jazyce XAML 15
Obrázek č. 10: Ukázka šablony Panorama v emulátoru a v deklaraci v jazyce XAML. 16
Obrázek č. 11: Ukázka šablony Hub v emulátoru a v deklaraci v jazyce XAML 16
Obrázek č. 12: Grafické znázornění životního cyklu aplikací 19
Obrázek č. 13: Obrazovka aktuální kurzy 26
Obrázek č. 14: Obrazovka převodu měn 27
Obrázek č. 15: Obrazovka grafu 28
Obrázek č. 16: Ukázka dat z databáze ČNB 29
Obrázek č. 17: StatusBar zobrazující dokončení aktualizace 38
Obrázek č. 18: Dotaz na ukončení aplikace 42
Obrázek č. 19: Hodnocení aplikace jedním z uživatelů 46

IV Seznam zdrojových kódů

Ukázka kódu 1 Vytvoření statického zdroje názvu aplikace 26
Ukázka kódu 2 Praktické využití zdroje při definování atributu Tile v Pivotu 26
Ukázka kódu 3 Definování indexeru a jeho následné využití 30
Ukázka kódu 4 Postup získání dat ze stránek ČNB 31
Ukázka kódu 5 Metoda zjišťuje, zda je den státním svátkem 32
Ukázka kódu 6 Zjištění, jak se změnil kurz oproti minulému a následné nastavení barvy a symbolu 33
Ukázka kódu 7 Nastavení List_men na bindování dat 34
Ukázka kódu 8 Předání listu měn do Listu_men 34
Ukázka kódu 9 Naplnění listu typu Graf 35
Ukázka kódu 10 Třída FirstTask včetně metody Run 36
Ukázka kódu 11 Registrace agenta 36
Ukázka kódu 12 Metoda na kontrolu internetu 37
Ukázka kódu 13 Definování aplikační lišty v jazyce XAML 39
Ukázka kódu 14 Zobrazení či skrytí tlačítek podle obrazovky 40
Ukázka kódu 15 Nastavení výchozí obrazovky 41

V Seznam grafů

Graf 1: Podíl mobilních systémů v letech 2016 až 2017 8
Graf 2: Vývoj v počtu stahování aplikace v letech 2016 až 2017 46

VI Seznam tabulek

Tabulka č. 1: Přehled verzí aplikace 45

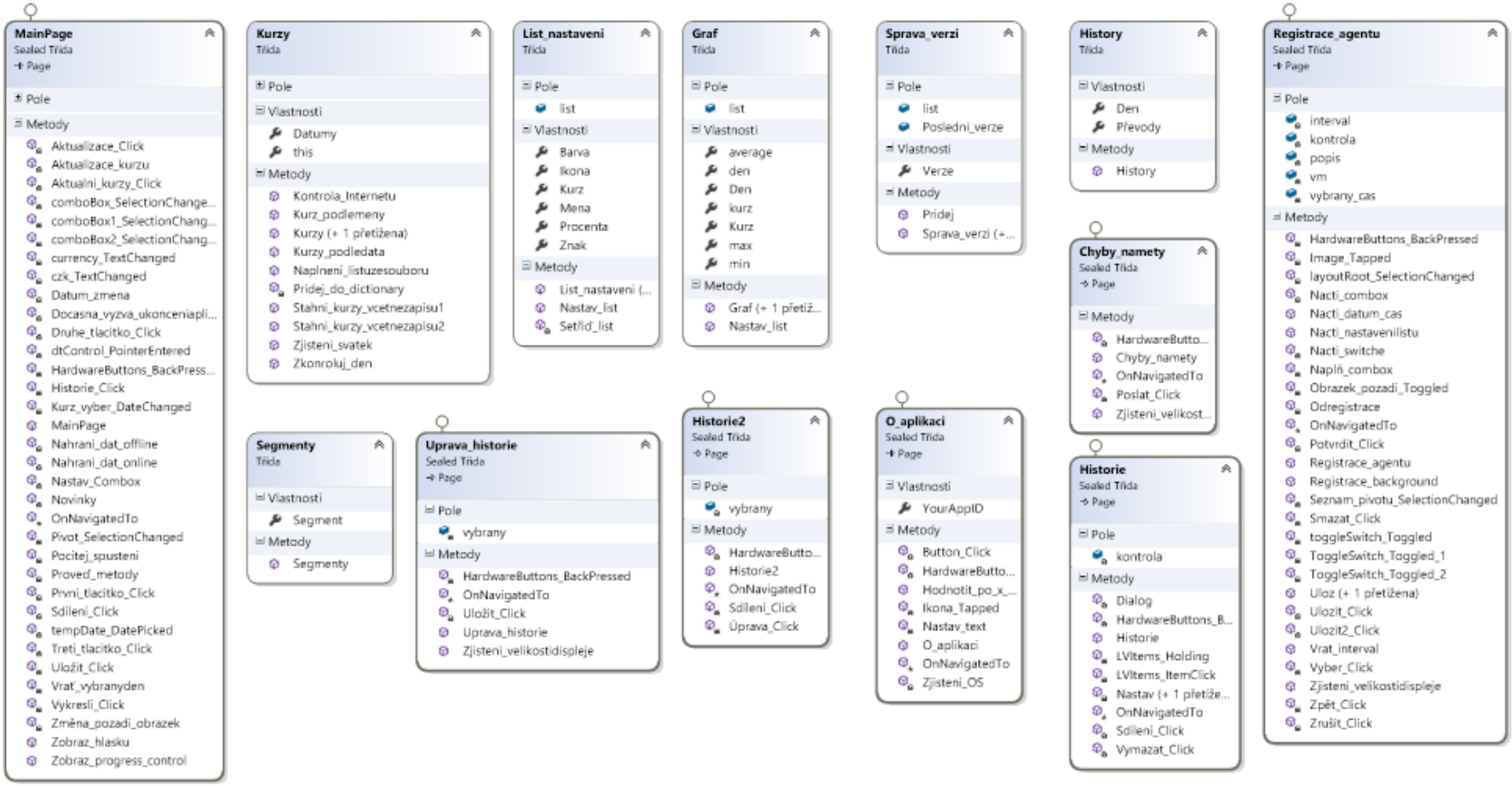
VII Seznam příloh

Příloha A Diagram tříd

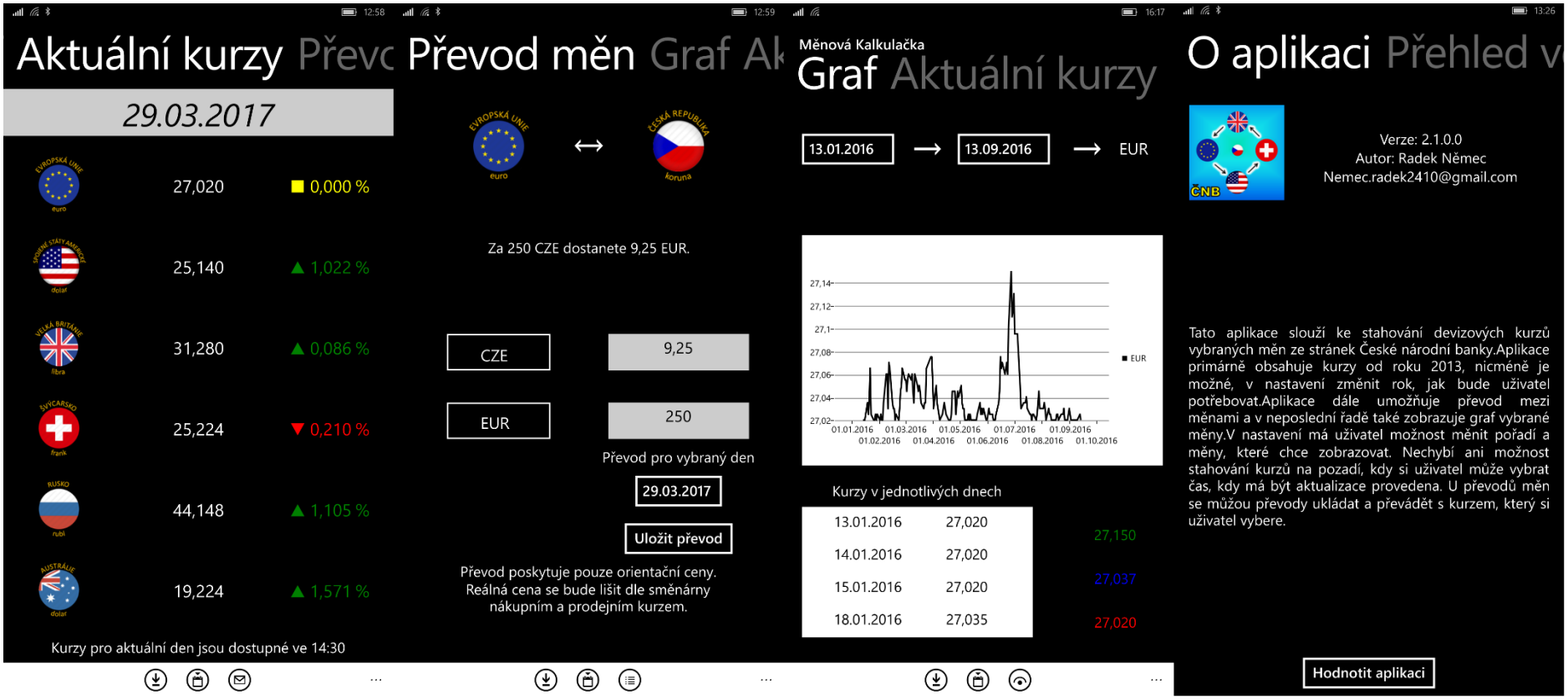
Příloha B Screenshoty aplikace

Příloha C Zdrojový kód aplikace je přiložen na datovém nosiči

VIII Přílohy



Příloha A Diagram tříd



Příloha B Screenshoty aplikace