VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

VÝVOJ APLIKACE PRO PLATFORMU WINDOWS PHONE

DEVELOPMENT OF APPLICATION FOR WINDOWS PHONE PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL INGR
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR DYDOWICZ, Ph.D.
SUPERVISOR

BRNO 2015

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2014/2015
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Ingr Michal

Manažerská informatika (6209R021)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává bakalářskou práci s názvem:

Vývoj aplikace pro platformu Windows Phone

v anglickém jazyce:

Development of Application for Windows Phone Platform

Pokyny pro vypracování:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrh řešení, přínos práce
Závěr
Seznam použité literatury

Seznam odborné literatury:

GARGENTA, Marko. Learning Android. 1st ed. Sebastopol, Calif.: O'Reilly, c2011, xvii, 245 p. ISBN 14-493-9050-1.
LEE, W.,M. Beginning Android application development. Indianapolis, IN: Wiley Pub., 2011. 428 s. Wrox beginning guides. ISBN 978-111-8087-800.
MARTIŠEK, D. Algoritmizace a programování v Delphi. 1. vyd. Brno: Littera, 2007. 230 s. ISBN 978-80-85763-37-9.
UJBÁNYAI, M. Programujeme pro Android. 1. vyd. Praha: Grada, 2012. 187 s. Průvodce (Grada). ISBN 978-80-247-3995-3.
VELTE, A., T. VELTE a R. ELSENPETER. Cloud Computing: praktický průvodce. 1. vyd. Brno: Computer Press, 2011. 344 s. ISBN 978-80-251-3333-0.

Vedoucí bakalářské práce: Ing. Petr Dydowicz, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2014/2015.

L.S.

doc. RNDr. Bedřich Půža, CSc. Ředitel ústavu	doc. Ing. et Ing. Stanislav Škapa, Ph.D. Děkan fakulty

V Brně, dne 28.2.2015

ABSTRAKT

Bakalářská práce se zabývá návrhem a vývojem aplikace ke službě eÚčty.cz pro platformu Windows Phone pomocí vývojového prostředí Microsoft Visual Studio 2013. Zaměřuje se také na následnou distribuci ve Windows Store a obsahuje návrhy ekonomického zhodnocení aplikace.

ABSTRACT

Bachelor‘s thesis is based on design and development of application for the service eÚčty.cz for the Windows Phone platform by using the development environment Microsoft Visual Studio 2013. It also focuses on subsequent distribution to the Windows Store and includes proposals economic evaluation of the application.

KLÍČOVÁ SLOVA

návrh, vývoj aplikace, Windows Phone 8.1, Visual Studio, ekonomické zhodnocení

KEYWORDS

design, development of application, Windows Phone 8.1, Visual Studio, economic evaluation

BIBILIOGRAFICKÁ CITACE

INGR, M. *Vývoj aplikace pro platformu Windows Phone.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2015. 60 s. Vedoucí bakalářské práce Ing. Petr Dydowicz, Ph.D..

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 3. června 2015 .....

Podpis

PODĚKOVÁNÍ

Chtěl bych poděkovat svému vedoucímu bakalářské práce Ing. Petru Dydowiczovi, Ph.D. za odborné vedení, rady a připomínky při vypracovávání této práce. Také děkuji Janu Lašovi, který je provozovatelem služby eÚčty.cz, za poskytnutí potřebných informací, veřejného klíče pro přístup ke službě a pomoc při vývoji aplikace.

OBSAH

ÚVOD

Jako téma bakalářské práce jsem si zvolil návrh a vývoj aplikace pro platformu Windows Phone, konkrétně aplikace pro službu eÚčty.cz. Tato služba nabízí přehlednou správu osobních či rodinných financí.

Pro tuto službu existuje aplikace pro mobilní telefony se systémem Android, ale uživatelé si stále častěji stěžují na chybějící aplikaci pro Windows Phone. Proto jsem se rozhodl pro Jana Laše, který je provozovatelem služby, aplikaci vyvinout.

Aplikace bude sloužit především pro rychlé zadávání transakcí z mobilního telefonu s operačním systémem Windows Phone 8.1, ale také pro náhledy na poslední transakce a zjištění stavů na účtech.

Protože služba eÚčty.cz je zcela zdarma, bude aplikace taktéž poskytována zdarma. Z tohoto důvodu v práci navrhnu ekonomické zhodnocení aplikace.

CÍLE PRÁCE A POSTUPY ŘEŠENÍ

Cílem bakalářské práce je návrh a vývoj aplikace pro mobilní platformu Windows Phone ke službě eÚčty.cz a její následná distribuce ve Windows Phone Store. Aplikace by měla mít praktický přínos pro uživatele služby, jednoduchý design a především intuitivní ovládání. Dílčím cílem práce je navrhnout možnosti ekonomického zhodnocení aplikace.

V úvodu práce budou představeny tři nejrozšířenější operační systémy mobilních telefonů a porovnány jejich pozice na trhu. Dále budou nastíněny možnosti vývoje aplikací pro každý z těchto operačních systémů, podrobněji se pak práce bude věnovat metodám vývoje aplikací pro Windows Phone 8.1.

Dále práce bude popisovat vývoj aplikace pro službu eÚčty.cz, ovšem před samotným vývojem bude služba eÚčty.cz podrobena zkoumání, následně budou vybrány funkce, které bude vhodné do mobilní aplikace implementovat, aby byla zachována jednoduchost aplikace, ale přitom aby obsahovala všechny základní funkce, které by mohli uživatelé vyžadovat. Poté bude analyzováno poskytované API rozhraní, aby bylo ověřeno, že veškeré vybrané funkce bude možno naprogramovat.

Podle zvolených funkcí pak bude navržen vzhled aplikace s ohledem na zachování podoby se webem služby eÚčty.cz. Zároveň bude preferována jednoduchost a čistota designu. Důraz bude také kladen na intuitivní ovládání, proto budou použity vhodné piktogramy či ikony, které umožní rychlou práci s aplikací.

Aplikace bude vyvíjena pomocí vývojového prostředí Microsoft Visual Studio 2013 s nainstalovaným doplňkem Windows Phone SDK 8.0. K programování v tomto prostředí se využívá jazyků VB.NET, C# či C++.

Po naprogramování aplikace bude následovat testování přímo ve smartphonu. Případné nedostatky a chyby budou odstraněny. Poté bude osloveno několik uživatelů, kterým bude aplikace představena a následně bude chování aplikace přizpůsobeno či změněno podle případných poznatků těchto uživatelů.

Po důkladném otestování bude aplikace nabídnuta pomocí distribučního kanálu Windows Phone Store široké veřejnosti a zároveň bude nabídnuto umístění odkazu pro stažení aplikace přímo ze stránky služby eÚčty.cz.

1 TEORETICKÁ VÝCHODISKA

První část bakalářské práce se bude okrajově zabývat historií mladé mobilní platformy Windows Phone a jejími rozdíly oproti přechozí verzi Windows Mobile. Dále se kapitola bude věnovat designu aplikací pro tento operační systém a doporučení pro pohodlné ovládání. V poslední části bude popsáno vývojové prostředí Microsoft Visual Studio včetně nutných doplňků pro vývoj a testování mobilních aplikací.

1.1 HISTORIE SYSTÉMU WINDOWS PHONE

Operační systém Windows Phone je poměrně mladý, avšak představení jeho předchůdce se datuje už do roku 2000. V dubnu toho roku Microsoft v New Yorku představil zcela nový operační systém Pocket PC 2000 určený pro mobilní zařízení, v té době především pro kapesní počítače [1]. Vizuálně se velmi podobal desktopovému systému Windows 98, ale byl postaven na hybridním jádru Windows CE. Součástí první verze systému byly také upravené verze známých aplikací Pocket Outlook, Pocket Word, Pocket Excel, Pocket Internet Explorer či Pocket Windows Media Player [2].

Ke konci následujícího roku byla představena nová verze systému Pocket PC 2002. Ta se od první verze příliš nelišila, vzhled byl trochu upraven po vzoru Windows XP, stejně tak byly převzaty i některé nové funkce. Mnohem významnější byl ovšem rok 2002, kdy byl vydán Pocket PC 2002 Phone Edition, který podporoval telefonní funkce a lze tvrdit, že tím se odstartoval vývoj chytrých mobilních telefonů [2].

Z tohoto důvodu došlo u další verze systému k přejmenování na Windows Mobile, nejprve na Windows Mobile 2003. Ten vyšel v několika verzích jak pro zařízení bez telefonního modulu tak i s ním, později v roce 2004 přibyla tzv. druhá edice, ve které je nově podporováno přepnutí do landscape režimu při otočení displeje na šířku [2].

V roce 2005 byl představen Windows Mobile 5, který je rozšířený o aplikaci PowerPoint Mobile a podporu technologií Direct3D. Mimo to přináší nový způsob práce s pamětí, který prodlužuje výdrž baterie, upravené uživatelské rozhraní a spoustu dalších drobných vylepšení [3].

Za poslední verzi systému Windows Mobile se dá považovat Windows Mobile 6, který byl po odmlce představen až v únoru 2007. Kromě značně přepracovaného vzhledu

přináší propojení se službami Windows Live, možnost automatických aktualizací, podporu VoIP apod. To ovšem nestačilo na stále sílící konkurenci a systém Windows Mobile začal ztrácet podíl na trhu [3].

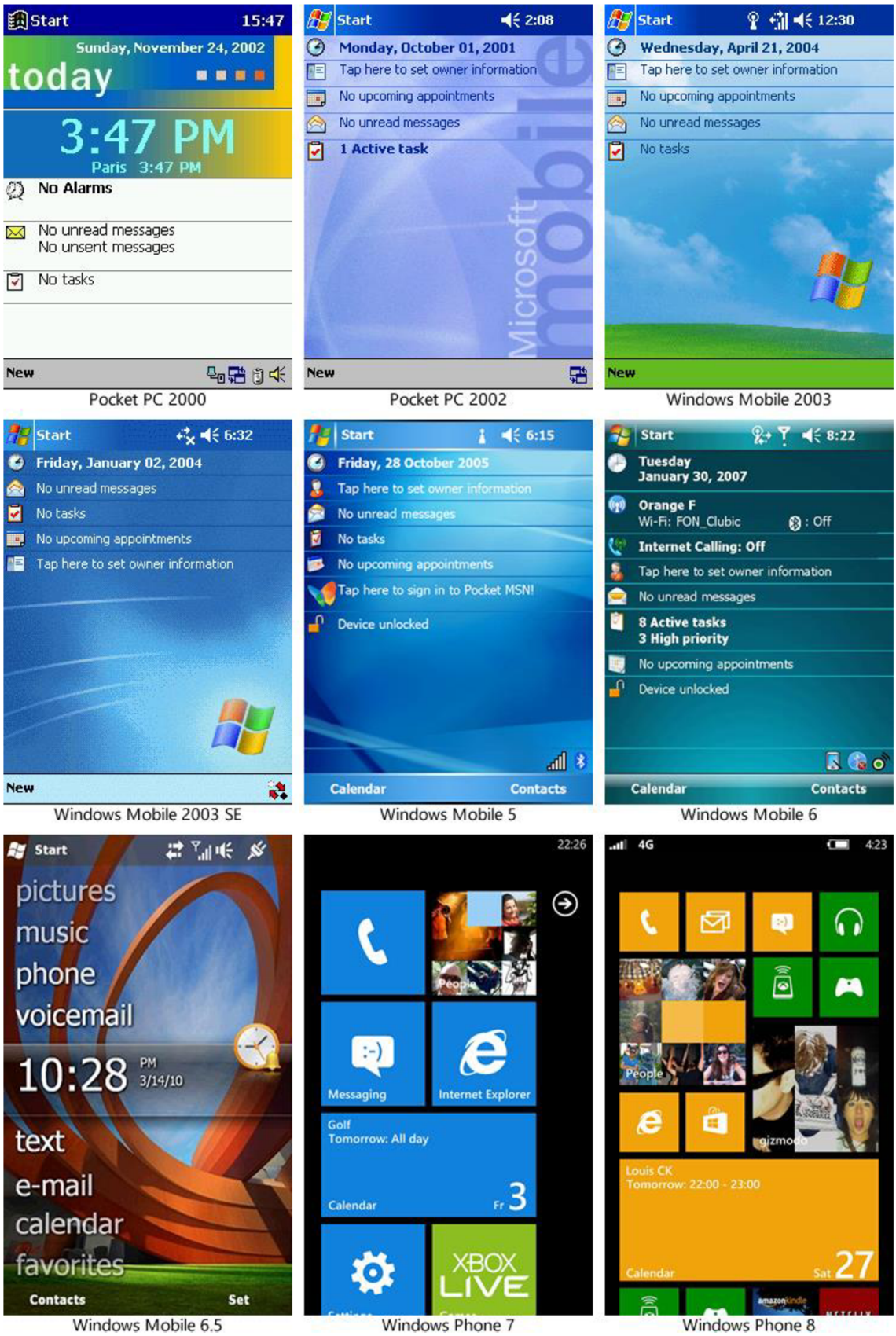
Systémy Pocket PC a Windows Mobile byly velmi otevřené, proto je bylo možné provozovat téměř na jakémkoli hardware. Spousta výrobců také systém modifikovala a přizpůsobovala pro svá zařízení, což značně komplikovalo práci vývojářům. Z tohoto důvodu, ale i proto, že tržní podíl Windows Mobile stále prudce klesal, učinil Microsoft nelehké rozhodnutí a začal vyvíjet zcela nový systém nesoucí označení Windows Phone, který měl být představen v roce 2009. Vývoj však trval příliš dlouho, proto byla vydána ještě aktualizace Windows Mobile 6.5, která přináší výraznou změnu designu, několik vylepšení a další drobné změny, aby zpomalila vznikající náskok konkurence [1] [3].

První verze nového systému byla představena 11. října 2010 a nesla označení Windows Phone 7. Bylo kompletně přepracováno uživatelské prostředí a objevují se zde poprvé dynamické dlaždice. Systém bohužel trpěl několika nedostatky, mezi něž patřila chybějící podpora paměťových karet, synchronizace obsahu telefonu pouze programem Microsoft Zune a nebo téměř žádné aplikace, protože více než milion aplikací pro Windows Mobile nejsou s novým operačním systémem Windows Phone kompatibilní [1] [3].

Pro české zákazníky měl Windows Phone 7 ještě jednu významnou nevýhodu, systém nepodporoval češtinu. To však napravila velká aktualizace v roce 2011 nesoucí název „Mango“, která přidává mimo české lokalizace několik stovek dalších funkcí a mění název systému na Windows Phone 7.5 [4].

Přes aktualizaci Windows Phone 7.8 se systém vypracoval k Windows Phone 8, který s sebou v roce 2012, kromě vylepšeného uživatelského prostředí, přináší další komplikaci pro vývojáře. Systém totiž přestává podporovat XNA Framework a aplikace, které jej využívají, musejí být přeprogramovány [3].

Poslední velkou aktualizací v červenci 2014 na Windows Phone 8.1 obdržel systém spoustu nových funkcí a velká část těch stávajících byla vylepšena podle zpětné vazby uživatelů. Výčet změn je dlouhý, zdůraznit lze hlasovou asistentku Cortana, centrum akcí, oddělené ovládání hlasitosti pro media a vyzvánění, sekvenční mód focení apod. [5].



Obrázek 1: Přehled vývoje designu OS pro mobilní platformu od Microsoft

Zdroj: Vlastní zpracování dle [6]

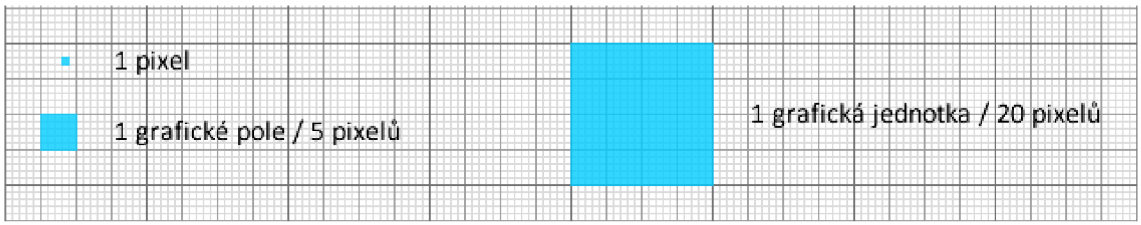
Za zmínku stojí i chystaná aktualizace systému na Windows Phone 10, která přinese opět spoustu nových funkcí a vylepšení. Navíc bude mít značnou výhodu týkající se aplikací, protože nově by měly jít aplikace vytvořené pro tablety a počítače spouštět na mobilním telefonu a naopak. Dojde totiž ke sjednocení těchto dvou vizuálně podobných systémů a Microsoft tak pod novým vedením docílí toho, systém Windows 10 bude jeden pro všechna zařízení. To by mělo velice usnadnit práci vývojářům, nutno však podotknout, že i v současnosti portování aplikace z Windows Phone 8.1 na Windows 8.1 a naopak není příliš složité [7].

1.2 DESIGN APLIKACÍ PRO WINDOWS PHONE

Systém Windows Phone je tvořen úplně od základu, stejné je to i s jeho designem, který se výrazně odlišuje od všech předchozích verzí ať už pro mobilní zařízení či klasické desktopy. Modern UI, tedy nový design uživatelského prostředí, je výsledkem spolupráce designerů, ergonomů, typografů a odborníků z mnoha dalších odvětví. Nový unifikovaný vzhled se ovšem netýká jen operačního systému, ale i samotných aplikací. Proto je vhodné se při návrhu aplikace řídit doporučeními, aby uživatel mohl využít svých návyků ze systému a věděl, kde má co hledat. Samozřejmě jsou to jen doporučení a není problém vytvořit zcela odlišnou aplikaci, ale je to pak jako sázka do loterie, zda uživatelé aplikaci přijmou, zorientují se v ní a oblíbí si ji, či nikoli [8].

1.2.1 ZÁKLADY DESIGNU APLIKACÍ PRO WINDOWS PHONE

Na rozdíl od předchozích verzí systému, v aplikacích pro Windows Phone není žádný rám, záhlaví okna, ani žádné tlačítko pro zavření. Aplikace tak může využít každý pixel displeje. Grafické objekty využívají vzájemné souhry tvarů, barev, pohybů, typografického fontu Segoe a ikon. Stmelujícím prvkem designu je mřížka, která určuje rozestup mezi jednotlivými objekty. Její použití se doporučuje pro zachování estetiky a zároveň účelnosti a k vytvoření konzistentního a strukturovaného rozvržení aplikace. Mřížka se skládá z takzvaných grafických jednotek (unit) o velikosti 20 × 20 pixelů, každá tato jednotka je dále rozdělena na 16 grafických polí (sub-unit) s rozměry 5 × 5 pixelů [8].

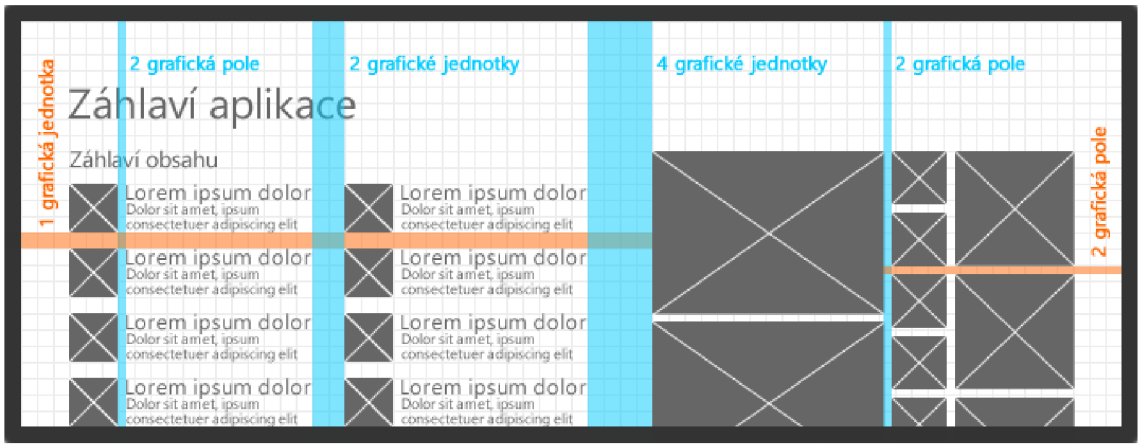


Obrázek 2: Porovnání grafické jednotky, grafického pole a pixelu

Zdroj. Vlastní zpracování dle [7]

V aplikacích pro Windows Phone se nedoporučuje organizovat prvky pomocí orámování či čar, ale jen oddělením pomocí prázdných míst, která opticky obsah ohraničují, výrazně zpřehledňují a tím zlepšují orientaci uživatele. Velikosti těchto mezer se liší, podle toho, zda jsou oddělovány prvky, které k sobě obsahově patří, či nikoli. Pro oddělení souvisejícího obsahu umístěného horizontálně se doporučuje mezera o 2 grafických polích, tedy 10 pixelech, naproti tomu pro oddělení sloupců seznamu by měla být použita mezera 2 grafických jednotek, což odpovídá 40 pixelů. Skupiny obsahů se pak oddělují volným místem o šířce 4 grafických jednotek [8].

Stejný princip se používá i při organizování položek umístěných vertikálně. Související grafické objekty rozmístí pomocí mezer o 2 grafických polích, pro oddělení položek seznamu se využije jedna grafická jednotka. Vše je pro přehlednost uvedeno na následujícím schématu rozvržení pro návrh aplikací pro desktopy a tablety, stejná doporučení však platí i pro mobilní telefony [8].



Obrázek 3: Oddělování obsahu pomocí prázdných míst

Zdroj: Vlastní zpracování dle [7]

1.2.2 FILOZOFIE OVLÁDÁNÍ

Při navrhování aplikace by se měla brát v potaz také filozofie ovládání. Windows Phone 8.1 je určen pro mobilní telefony ovládané dotykem, proto je při rozmísťování prvků vhodné řídit se doporučeními, kam umístit obsahové prvky a kam ovládací prvky. Je třeba si uvědomit, že uživatelé mobilní telefon většinou drží v jedné ruce a ovládají jej palcem, ovládací prvky je tedy vhodné umístit taky, aby byly v dosahu palců. Stejně tak platí, že obsahová část by měla být umístěna tak, aby ji palce nepřekrývaly [8].

Stejně důležité je i volba velikosti a rozestupy ovládacích prvků. Na příliš malé objekty blízko u sebe se půjde stěží trefit, přehnaně velká tlačítka by naopak zmenšovala prostor pro umístění obsahu. Ideálně by ovládací prvek při zobrazení na displeji neměl být užší než 9 mm a nižší než 7 mm, rozestupy by měly být minimálně 2 mm. Důležitá je také odezva na uživatelskou interakci například změnou barvy ovládacího prvku. Bez tohoto je možné, že aplikace ani neprojde schvalovacím procesem před publikováním do Windows Phone Store [7] [8].

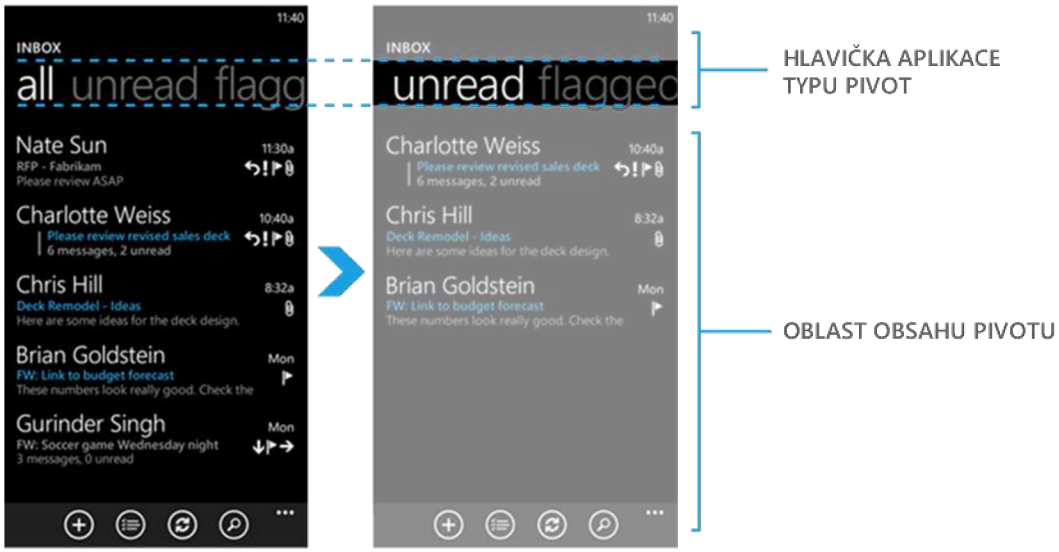
Obrázek 4: Doporučené umístění ovládacích prvků a obsahu

Zdroj: Vlastní zpracování dle [8]

1.2.3 PIVOT

Při návrhu designu aplikace se leckdy stane, že obrazovka mobilního telefonu je malá a nevejde se na ni vše potřebné. Řešením by mohlo být umístění objektů pod sebe, uživatel by pak svislým posunem procházel obsah aplikace. V případě, že ale spolu obsah přímo nesouvisí, není tato metoda příliš vhodná. Uživatel totiž nemusí vědět, že pod obsahem týkající se jedné věci se nachází obsah, který se zabývá věcí jinou.

Z tohoto důvodu je mnohem vhodnější využít aplikaci typu Pivot. Pivoty jsou libovolně vysoké stránky seřazené vedle sebe, lze je procházet jednoduchým gestem posunu doleva či doprava. Aby bylo poukázáno na existenci dalších stránek, zobrazují se v hlavičce aplikace šedě názvy dalších stránek. Tento typ aplikace se hodí především pro práci se seznamy a agregovanými údaji. Může být také použita na stránkové zobrazování, či filtrování položek.



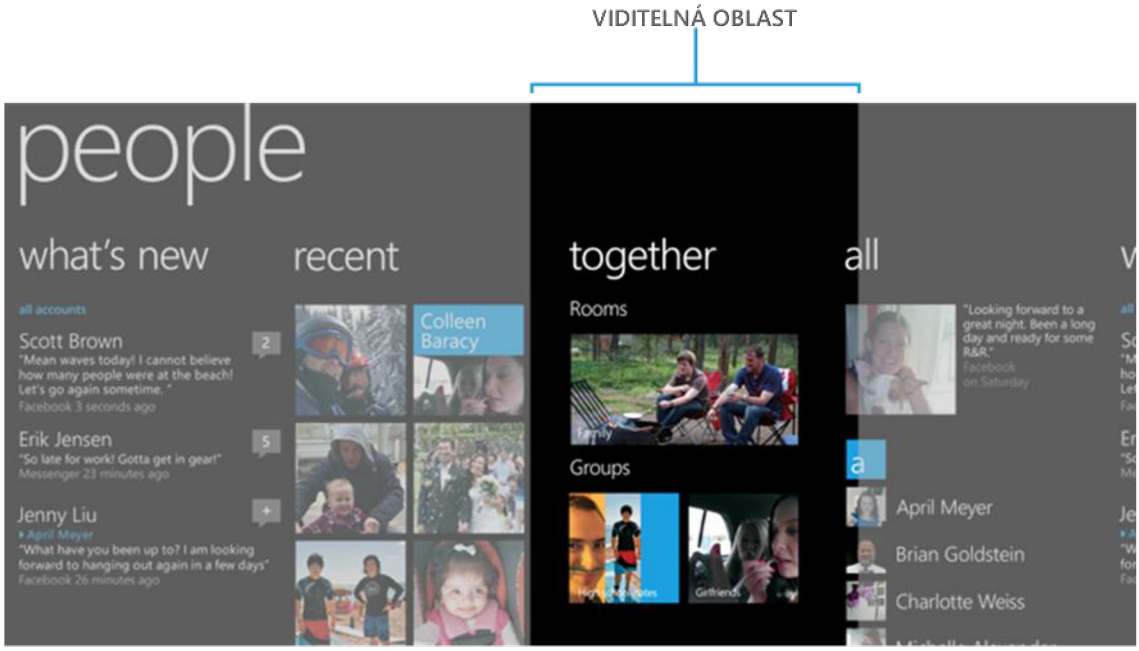
Obrázek 5: Aplikace typu Pivot

Zdroj: Převzato a upraveno dle [7]

1.2.4 PANORAMA

Další možností, jak získat větší zobrazovací oblast, je použít aplikaci typu Panorama. Jedná se o jakousi virtuální plochu, několikanásobně širší než zobrazovací zařízení. Displej telefonu pak plní funkci posuvného okna, čímž zobrazuje pouze výřez panoramatické plochy. Aby bylo uživateli zřejmé, že se lze v aplikaci posouvat i ve vodorovném směru, je vždy vpravo zobrazen kousek další obrazovky. Jednotlivé

obrazovky panoramatu lze posouvat i svisle, stejně jako u pivotu. Hlavní odlišností ovšem je možnost vytvořit obrazovky panoramatu širší, než je rozlišení displeje a tím lze aplikaci elegantně ozvláštnit, v tomto případě je však svislé rolování silně nedoporučováno.



Obrázek 6: Aplikace typu Panorama

Zdroj: Převzato a upraveno dle [7]

1.3 VÝVOJ APLIKACÍ PRO WINDOWS PHONE

Návrh designu aplikace je ovšem jen jedna část vývoje, druhou důležitou částí je aplikační logika. Tyto vrstvy každé vyvíjené aplikace by měly být oddělené za účelem vysoké kvality a přehlednosti zdrojového kódu, redukci jeho množství a zjednodušení pozdějších úprav. Stejné doporučení platí i pro vývoj aplikací pro systém Windows Phone. Microsoft proto vyvinul návrhový vzor MVVM (Model-View-ViewModel), který odděluje aplikační logiku od uživatelského rozhraní [9].

Obrázek 7: Schéma architektury Model-View-ViewModel

Zdroj: Zpracováno dle [7]

Z Obrázek 7 je patrné uspořádání architektury MVVM. Do vrstvy View, která zpracovává uživatelské vstupy a předává je další vrstvě, náleží veškeré prvky uživatelského rozhraní

aplikace. Vrstva ViewModel přenáší a tranformuje do vrstvy Model uživatelské vstupy a opačným směrem data aplikace. Tato data jsou uložena ve vrstvě Model v úložišti aplikace, zpravidla v databázi.

1.3.1 PROGRAMOVACÍ JAZYK

K naprogramování aplikační logiky nativních aplikací lze využít dvou, resp. tří objektově orientovaných jazyků a to C# resp. Visual Basic.NET, nebo C++. Tato bakalářská práce se bude zabývat vývojem prostřednictvím C#, a to především z důvodu jeho značné podobnosti s jazykem Visual Basic, široké uživatelské základně, jednoduchosti a také proto, že jej vydala v roce 2002 společnost Microsoft po vzoru jazyků C++ a Java.

C# je tedy moderní, objektivně orientovaný jazyk s integrovaným .NET Framework. Poslední verze 5.0 vydaná v roce 2012 nově podporuje i asynchronní programování, což je důležité pro zachování odezvy aplikace, i když právě vykonává nějakou časově náročnou operaci, typicky při načítání dat z internetu.

1.3.2 VÝVOJOVÉ PROSTŘEDÍ

K vývoji aplikací pro operační systém Windows je potřeba vývojového prostředí Visual Studio. To je pro studenty dostupné z Microsoft DreamSpark zcela zdarma. Lze však využít i odlehčenou verzi Visual Studio Express for Windows, která je dostupná zdarma prakticky pro každého [7] [8].

Součástí obou verzí je i návrhové prostředí Blend, které ocení jak designeři, tak i programátoři. Blend i Visual Studio jsou úzce provázány, změny provedené v jednom prostředí se rozpoznají a aplikují v druhém. Navrhovaný kód bude proto neustále aktuální. Navíc grafik, který navrhne design aplikace pomocí nástroje Blend, usnadní práci vývojářům. Ti se obvykle po předložení návrhu pustí do jeho implementace, ovšem musí jej přizpůsobit možnostem prezenční vrstvy a návrhového prostředí, proto výsledná podoba může být trochu odlišná. V případě, že ale designer navrhuje ve formátu, který je podporován i vývojovým prostředím, bude výsledný design naprosto totožný návrhu a vývojáři do něj jen naprogramují aplikační logiku [8].

Vývojové prostředí má v sobě zakomponován také kvalitní emulátor, díky němuž lze ladit aplikace bez nutnosti vlastnit zařízení, pro které je určena. V praxi to znamená, že díky

emulátoru lze na běžném počítači simulovat spuštění aplikace ve smartphonu či tabletu a testovat tak reakci aplikace na použití multidotyků, otočení zařízení, změnu aktuální polohy a podobně [8].

1.3.3 SADA NÁSTROJŮ PRO WINDOWS PHONE

Základní instalace Visual Studia 2013 neobsahuje potřebné nástroje pro vývoj aplikací pro Windows Phone 8. Proto je zapotřebí nainstalovat sadu Windows Phone SDK, která obsahuje i emulátor této platformy. Ten je vhodný pro testování aplikací, jak bylo popsáno výše [8].

Pokud však vývojář zařízení se systémem Windows Phone 8.1 vlastní, může testování aplikace provádět přímo na reálném zařízení. Nejprve je ale nutné telefon registrovat pro vývoj. To se provádí pomocí nástroje Windows Phone Developer Registration, který se nainstaluje společně se sadou Windows Phone SDK. Samotná registrace vyžaduje jen připojení požadovaného zařízení portem USB k počítači a vlastnění vývojářského účtu. Ten lze zdarma vytvořit na stránkách dev.windows.com [8] [10].

1.3.4 ŽIVOTNÍ CYKLUS APLIKACE

Aplikace pro Modern UI mají poněkud odlišný životní cyklus ve srovnání s přechozími verzemi systému Windows. Aplikace má 3 základní stavy. Výchozím stavem je Not Running, ten má aplikace v případě, že ještě nebyla spuštěna a není tedy načtena v paměti.

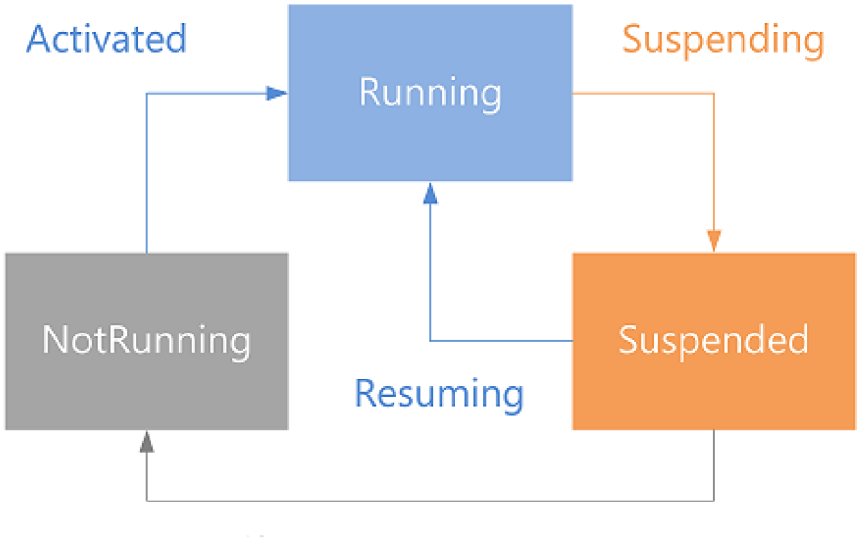
Po spuštění aplikace se zobrazí takzvaný Splash screen a následně se zobrazí uživatelské rozhraní aplikace přes celý displej mobilního telefonu. Tím se aplikace dostane do stavu Running.

V případě, že se uživatel přepne do jiné aplikace, či zobrazí úvodní obrazovku, aplikace přejde do úsporného stavu Suspended. Při přechodu do tohoto stavu má aplikace 5 sekund na to, aby uložila veškeré potřebné údaje, než dojde k jejímu násilnému pozastavení. Potřebnými údaji se rozumí vyplněná data uživatelem, aktuální pozice v uživatelském rozhraní apod., jedná se tedy o vše, co je potřeba uchovat v případě, že by aplikace z tohoto stavu už nebyla obnovena. Protože ukládání může při větším množství dat trvat delší dobu, je v některých případech vhodné ukládat data průběžně. Pokud uživatel bude

vyplňovat např. dlouhý dotazník, mohou být vložené údaje ukládány po každém přepnutí do jiného textového pole, případně po zapsání každého písmene.

Ve stavu Suspended se aplikace nachází, dokud ji uživatel znovu neaktivuje nebo dokud ji on sám či operační systém neukončí. Pokud dojde k reaktivaci, získá aplikace opět stav Running a zobrazí uživatelské prostředí ve stavu, v jakém se nacházelo před pozastavením.

Aplikace může být ukončena uživatelem, ale také operačním systémem z důvodu docházejícího volného místa v operační paměti. Pokud dojde k ukončení, získá aplikace nejdříve stav Suspended, aby mohla uložit potřebná data a poté stav Not Running. Tím dojde uvolnění aplikace z operační paměti.



Obrázek 8: Schéma životního cyklu aplikace pro Windows Phone

Zdroj: Převzato z [7]

2 ANALÝZA PROBLÉMU A SOUČASNÉ SITUACE

V dnešní době vlastní smartphone, tedy mobilní telefon s operačním systémem, dotykovou obrazovkou a výkonným hardware, více než 30 % obyvatel ČR [11]. Tato zařízení, oproti svým mobilním předchůdcům, neumožňují jen telefonování a zasílání krátkých textových zpráv, ale nabízí spoustu dalších funkcí, jako např. pořizování, editaci a prohlížení multimédií, hraní her nebo i správa textových či tabulkových dokumentů. Funkce každého smartphonu lze navíc rozšířit nainstalováním nejrůznějších aplikací, které vyvíjí firmy, ale i samotní uživatelé.

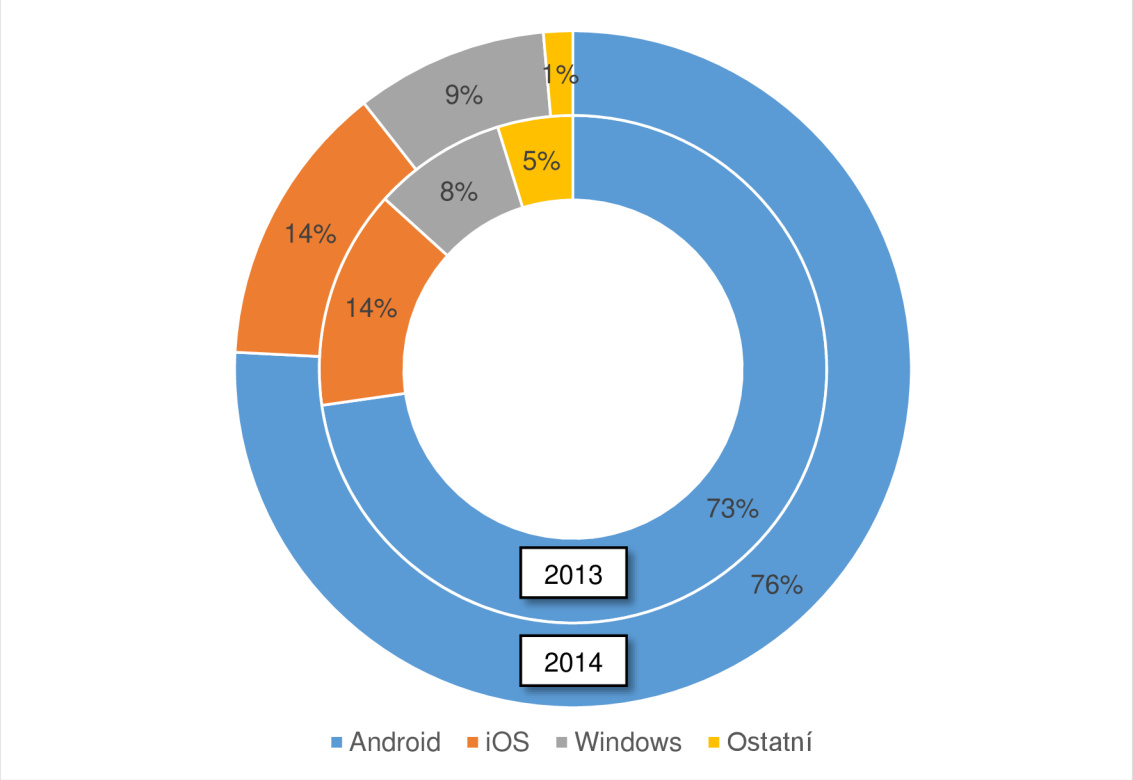
Společně s bezdrátovým internetovým připojením lze smartphone použít téměř ke všemu, k čemu je využíván počítač. Oproti počítači má však nespornou výhodu, která tkví v jeho vysoké mobilitě. Díky této vlastnosti je využívání pokročilejších funkcí, aplikací a především internetu a služeb, které nabízí, mnohem jednodušší a lze je využít bez ohledu na místo a čas. Není tedy problém kdekoli a kdykoli vyřídit pracovní e-maily, objednat lístky do kina, sdílet s přáteli právě pořízené fotografie, nebo dokonce ovládat svůj domov, od zapnutí vytápění či klimatizace, přes ovládání světel, po aktivaci a deaktivaci funkcí alarmu.

Všechny tyto výhody jsou důvodem, proč se mobilní telefony s operačním systémem těší velké oblibě mezi uživateli, což dokazuje i prodej smartphonů v roce 2014, kdy se ve světě prodalo téměř 1,3 miliardy těchto zařízení. To je nárůst o více než 30 % oproti roku 2013 [12].

2.1 PODÍL MOBILNÍCH OPERAČNÍCH SYSTÉMŮ NA TRHU

V současnosti na trhu dominují tři operační systémy mobilních telefonů. Největší podíl na světovém trhu má s velkým odstupem operační systém Android s 81 % [12]. Jinak tomu není ani v Evropě, kde ve třetím čtvrtletí roku 2014 byl tento systém v necelých 76 % chytrých telefonů. Druhý největší podíl na evropském trhu patří společnosti Apple, tedy systému iOS s 13,2 %. Třetím největším hráčem na poli smartphonů je společnost Microsoft a její mladý operační systém Windows Phone. Ten ve třetím čtvrtletí dosáhl téměř na hranici 10 %, a jeho podíl se, alespoň v Evropě, zvětšuje.

Ostatní operační systémy jsou těmi dominantními postupně z trhu vytlačovány. Ve třetím kvartále roku 2014 nedosahovaly ostatní operační systémy v součtu ani 1,5% podílu na trhu a to v důsledku meziročního poklesu o necelé 4 % [13].



Graf 1: Podíl operačních systému na evropském trhu (EU5) v letech 2013 a 2014

Zpracováno dle [13]

Společnost Apple měla donedávna prvenství v počtu nabízených aplikací v App Store, ten aktuálně obsahuje 1,2 milionů položek. Konkurenční Android od Google však iOS předehnal a na konci roku 2014 bylo možné z Play Store nainstalovat více než 1,4 milionů aplikací [14]. Pozadu však zůstává Windows Phone se 340 tisíci aplikacemi ve Windows Phone Store, protože mnoho vývojářů tuto platformu stále opomíjí [15].

Podle internetového průzkumu společnosti Seznam.cz, a.s. je při výběru smartphonu dostupnost konkrétní aplikace podstatná pro 63 % uživatelů [16]. Pokud tedy vývojáři nebudou pro platformu Windows Phone aplikace vyvíjet, bude mobilní operační systém Microsoftu jen stěží dohánět konkurenci.

2.2 MOŽNOSTI VÝVOJE APLIKACE

Jak už bylo zmíněno, velkou výhodou chytrých telefonů je jejich rozšiřitelnost o množství nejrůznějších aplikací. Ať už se vývojář rozhodne pro vývoj aplikace pro kterýkoli z výše zmíněných operačních systémů, jsou k dispozici dva různé přístupy implementace. Jedním z nich, který je pro všechny tři nejrozšířenější platformy stejný, je využití podpory webových aplikací, tedy kombinace HTML5, CCS3 a jazyku JavaScript. Tuto možnost ocení především weboví vývojáři. Druhou volbou je pak vývoj nativních aplikací, které jsou pro každou platformu specifické [8].

2.2.1 WEBOVÉ APLIKACE

HTML je jednoduchý značkovací jazyk, který slouží primárně pro tvorbu webových prezentací. HTML5 je pak nejnovější verzí tohoto jazyka, jehož největší změnou je zavedení podpory přehrávání multimediálních souborů přímo v prohlížečích zařízení bez nutnosti instalace doplňků. V kombinaci s CSS3, které slouží pro stylování a skriptovacího jazyku JavaScript, lze vytvářet téměř plnohodnotné aplikace a několik vývojářů se nechalo slyšet, že v tomto triu vidí budoucnost mobilních aplikací [17].

Velkou výhodou webových aplikací je, že při minimálních úpravách kódu je lze distribuovat napříč operačními systémy a dají se tak považovat za multiplatformní. Drobné úpravy kódu jsou však potřebné, protože implementace standardů HTML5 závisí na výrobci daného systému a proto mohou vznikat rozdíly při vykreslování stejné aplikace na různých platformách. Rozdíly mohou vznikat i mezi jednotlivými verzemi stejného operačního systému. Optimalizace aplikace však nebývá příliš nákladná, většinou je zvládne jen jeden programátor [8].

Nevýhodou však bývá zpravidla nižší výkon, který je úzce spjat s rychlostí internetového připojení. I když HTML5 aplikace mohou využívat off-line režim, ve kterém se vše potřebné uloží do lokální paměti zařízení, je i přesto připojení k internetu jednou za čas vyžadováno, aby mohl být obsah aktualizován. Pokud však HTML5 aplikace vyžaduje neustálé připojení k internetu, mohou být někdy odezvy na uživatelskou činnost příliš zpomalené.

2.2.2 NATIVNÍ APLIKACE

Na rozdíl od webových aplikací je nativní aplikace programována přímo pro dané zařízení, resp. operační systém. Využívá možností, které zařízení nabízí a které jsou pro jeho systém specifické. Proto je nutné využít vývojové prostředí určené pro danou platformu. Stejně tak programovací jazyky jsou rozdílné. Pro Android se využívá jazyk Java, pro iOS Objective-C a pro Windows Phone je to jazyk XAML v kombinaci s programovacími jazyky C#, VB.NET nebo C++ [18].

Tabulka 1: Přehled programovacích jazyků a vývojových prostředí

Zpracováno dle [8] [9]

	Programovací jazyk	Vývojové prostředí
Android	Java	Eclipse, IntelliJ IDEA, NetBeans
iOS	Objective-C	Xcode, AppCode, Emarcadero
Windows Phone	XAML, C#, VB.NET, C++	Visual Studio

Z tohoto plynou problémy, které vznikají při vývoji nativních aplikací pro více platforem. Mezi hlavní patří vysoké náklady na vývoj, protože pro každou platformu je aplikace vyvíjena individuálně. Potřeba více programátorů je další nevýhodou, málokterý programátor se umí dobře orientovat ve všech vývojových prostředích a programovacích jazycích [18].

Výhodou nativních aplikací je vyšší bezpečnost a výkonnost díky využití API (rozhraní pro programování aplikace), které poskytují výrobci operačních systémů a tím umožňují využívat dané zařízení naplno [18]. Nativní aplikace navíc mohou mít přizpůsobenější uživatelské rozhraní, vykonávat velké množství funkcí a využívat hardware zařízení, jako je např. gyroskop, náklonové čidlo, fotoaparát, GPS apod. Takto lze tedy vytvářet mnohem sofistikovanější a specializované aplikace.

2.3 DISTRIBUCE APLIKACE

Různé platformy používají různé distribuční kanály pro rozšíření aplikací mezi uživatele. Všechny ovšem fungují na stejném principu. Jedná se o tzv. obchody s aplikacemi. Jsou to virtuální obchody, pomocí kterých lze vyhledat, stáhnout a nainstalovat aplikaci do chytrého telefonu [9].

Microsoft nabízí aplikace pro Windows Phone skrze Windows Phone Store. Je to jediný oficiální zdroj aplikací a her, z nichž každá prochází před zveřejněním schválením a ověřením, zda odpovídá podmínkám a požadavkům, které Microsoft stanovuje. Aplikaci může pomocí Windows Phone Store nabídnout každý vývojář, který se u Microsoftu zaregistruje jako vývojář [8].

2.3.1 PODMÍNKY PRO SCHVÁLENÍ APLIKACE

Microsoft stanovuje několik podmínek, které je nutné dodržet, aby aplikace prošla schvalovacím procesem a mohla být publikována na Windows Phone Store. V opačném případě se aplikace vrátí neschválená a je nutné ji opravit. Aktuální podmínky obsahující zásady ohledně funkčnosti, zabezpečenosti, použitelnosti apod. jsou k dispozici na webu Windows Dev Center a jsou rozděleny do následujících kategorií [8] [10]:

1. Aplikace pro Windows Store musí přinášet zákazníkům přidanou hodnotu.
2. Aplikace pro Windows Store může zobrazovat reklamy, ale musí poskytovat více 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.

Dále aplikace nesmí obsahovat nadměrně vulgární a nevhodný obsah, musí se řídit zákony a nesmí propagovat nasilí, nadměrné nebo nezodpovědné užívání alkoholu, tabákových výrobků, drog nebo zbraní [10]

2.4 POPIS SLUŽBY EÚČTY.CZ

V této práci bude vyvíjena aplikace pro službu eÚčty.cz. Tu využívá přes 5 tisíc lidí, především proto, aby získali přehled a kontrolu nad tokem svých osobních financí. Pomocí jednoduchých nástrojů lze zadávat příjmy a výdaje, řadit je do hierarchických kategorií a přiřazovat k účtům. Transakce lze zadávat ručně (převážně hotovostní výdaje a příjmy) nebo automaticky pomocí nastavení pravidelných plateb nebo importu bankovních výpisů. Ze zadaných dat se vytváří nejrůznější statistiky, grafy a přehledy. Lze pak jednoduše zjistit, jaké jsou např. měsíční výdaje na stravování, bydlení, dopravu, celkovou roční výši příjmů, jak se na výdajích podílí výdaje za sport nebo zábavu [19].

Dále lze ve službě kontrolovat hypoteční úvěry, uchovávat informace o zárukách kupovaného zboží, sledovat spotřebu energií v domácnosti a neposlední řadě evidovat počet ujetých kilometrů a spotřebu pohonných hmot automobilů, vč. možnosti nastavení upozornění na nutnost evidenčních kontrol či plateb povinného ručení.

Mezi ocenitelné funkce patří i možnost založit tzv. rodinný účet, pomocí kterého lze přidávat členy domácnosti, každému přiřadit oprávnění k jednotlivým funkcím a účtům, a tím umožnit sledovat tok finančních prostředků celé rodiny.

Předpokladem pro správná data, a tím i statistiky, je zadávání všech finančních transakcí, které uživatel provede. V případě pohybů na bankovních účtech nelze narazit na žádná úskalí, všechny příjmy a výdaje jsou dohledatelné v transakční historii bankovního účtu. Problém ovšem může nastat, platí-li uživatel hotově a neschová si, resp. nedostane doklad o zaplacení, např. zaplatí za nápoj v automatu, nechá spropitné v restauraci apod. Než se pak dostane k počítači s připojením k internetu, může zapomenout, že transakci provedl či jaká byla její přesná výše. Proto je vhodné nabídnout uživatelům aplikaci do chytrých telefonů, pomocí které mohou tyto transakce rychle a jednoduše uložit ihned po uskutečnění transakce.

V následující části bakalářské práce takovou aplikaci pro Windows Phone 8.1 navrhnu, a to jak po grafické, tak i funkční stránce. Po otestování její správné funkčnosti ji publikuji prostřednictvím distribučního kanálu Windows Phone Store.

3 VLASTNÍ NÁVRH ŘEŠENÍ

Hlavní úlohou aplikace bude tedy zadávání a editace transakcí, výpis účtů a jejich stavů ze služby eÚčty.cz. Komunikace se serverem bude probíhat přes poskytované API. Toto rozhraní používá protokoly SOAP a WSDL, které využívají syntaxi jazyka XML. Zde nastává však první problém, protože Windows Phone ve verzi 8.1 již přímo nepodporuje tyto protokoly, ale pouze architekturu REST. Hlavním rozdílem oproti známějším procedurálně orientovaným protokolům SOAP a WSDL je, že REST orientován datově, tedy umožňuje mazat, editovat, či přistupovat k samotným datům na serveru, nikoli k procedurám. Bude tedy nutné navrhnout metody, pomocí kterých bude komunikace se službou vedena. K tomu se však dostaneme až později, na začátku se totiž budeme zabývat návrhem uživatelského rozhraní a funkcí aplikace.

3.1 ROZVRŽENÍ UŽIVATELSKÉHO PROSTŘEDÍ

Ze všeho nejdříve je potřeba navrhnout rozvržení uživatelského rozhraní na základě funkcí, které aplikace bude vykonávat. Kompletní seznam poskytovaných funkcí je uveden v Příloha 1 vč. jejich krátkého popisu. V rámci této bakalářské práce budeme využívat funkce:

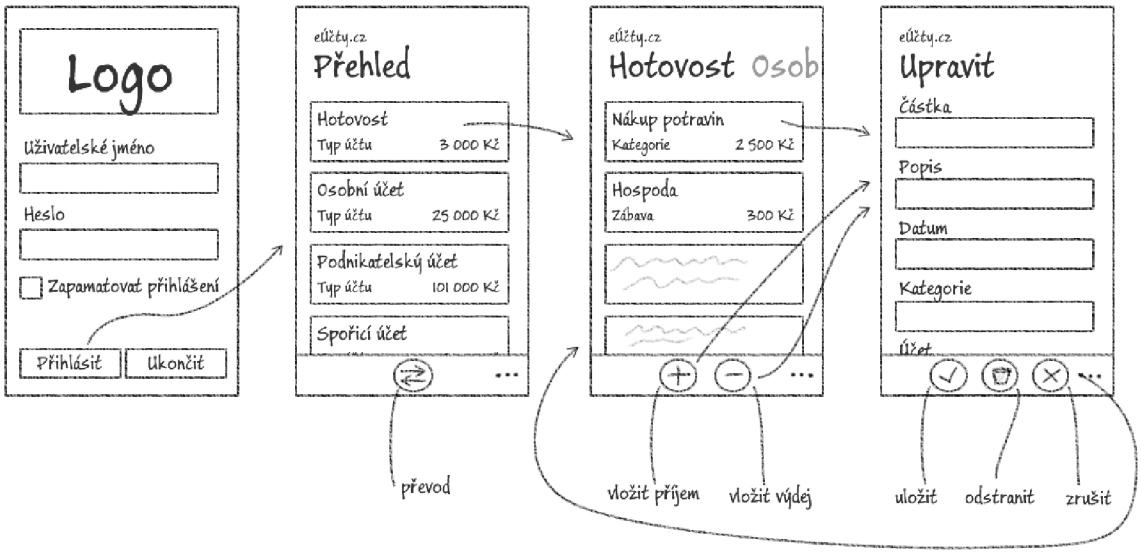
- TestLogin pro ověření uživatelského jména a hesla funkcí,
- AccountListWithAmount pro výpis seznamu účtů vč. jejich stavů,
- TransactionListLast50 pro výpis 50 posledních transakcí účtu,
- AddTransferToOtherAccount pro převod částky z účtu na účet,
- AddTransaction, DeleteTransaction a EditTransaction pro přidání, smazání a editaci transakcí.

Je tedy zřejmé, že první obrazovkou, kterou uživatel po spuštění aplikace uvidí, bude přihlašovací formulář. Po vzoru eÚčty.cz se po přihlášení zobrazí celkový přehled účtů s jejich stavy. Po otevření účtu se zobrazí výpis posledních 50 transakcí (API zobrazení více transakcí neumožňuje). Ovládací tlačítka pro přidání, mazání, ukládání transakcí, vložení převodu apod. budou umístěna ve spodní liště aplikace, která je v systému Windows Phone nazývaná Command Bar. Otevření editačního formuláře již uložené transakce se provede kliknutím na transakci, stejně tak se účet otevře kliknutím přímo na něj.

Pro jednodušší navigaci mezi účty využijeme rozložení Pivot, kdy jednotlivé účty budou naskládány vedle sebe, vertikálně pak bude umístěn seznam transakcí právě zobrazeného účtu.

Aby byla aplikace intuitivní, je potřeba její ovládání co nejvíce přiblížit k samotné službě eÚčty.cz na internetu. Zde se nezadávají transakce přímo, ale nejdříve se zvolí, zda se jedná o výdaj či příjem. Poté se teprve zobrazí formulář s poli pro vložení částky, data, kategorie, popisu transakce a s volbou účtu, ke kterému se má transakce přiřadit. Mobilní aplikace bude navržena stejně. Stránka s formulářem bude jednotná, ale podle typu úpravy předvyplní určitá data a změní popisy polí. Pokud se např. bude jednat o editaci existující transakce, bude formulář vyplněn aktuálně uloženými údaji, pokud půjde o nový výdaj, formulář bude prázdný, ale v záhlaví stránky se zobrazí nadpis „Nový výdaj“.

Základní rozvržení aplikace je shrnuto do následujícího obrázku, ze kterého je vše zřejmé.



Obrázek 9: Návrh rozložení uživatelského rozhraní

Zdroj: Vlastní návrh a zpracování

3.2 NÁVRH DESIGNU

Design aplikace by měl ctít doporučení Microsoftu uvedené na začátku této práce, ale měl by i korespondovat se službou eÚčty.cz. Pokud jde o barvy, základními barvami služby

jsou černá, bílá a žlutooranžová. Stejné barvy tedy budou použity i v aplikaci, avšak s ohledem na zvolený motiv telefonu.

Ve Windows Phone 8.1 jsou dva základní motivy a to světlý a tmavý, přitom druhý z nich je výchozí. Je také doporučený, protože z konstrukčních důvodů displeje je zobrazování světlých bodů energeticky náročnější a při použití světlého motivu tedy dochází k rychlejšímu vybíjení baterie. Aplikace bude nastavení tématu respektovat a bude mít tedy dvě varianty designu.

Samotný design aplikace bude navržen přímo v jazyce XAML v prostředí Visual Studio 2013. V takzvaném okně návrháře, ve kterém je zakomponováno rozšíření Blend, bude kontrolován aktuální stav uživatelského rozhraní aplikace.

Nejprve je tedy potřeba založit projekt, a protože aplikace bude využívat Pivoty, založíme projekt z přednastavené šablony Pivot Page, která tyto ovládací prvky již obsahuje. Tyto Pivoty, jak už bylo zmíněno, budou obsahovat výpisy transakcí k daným účtům, je tedy potřeba ještě doplnit zbývající stránky, tedy stránku s přihlašovacím formulářem (LoginPage), editačním formulářem (ItemPage) a přehledovou stránku s výpisem všech účtů (MainPage). Všechny vytvoříme opět z přednastavených šablon, konkrétně šablony Basic Page.

3.2.1 GLOBÁLNÍ SOUBOR APLIKACE

V projektu máme tedy 4 vytvořené soubory XAML a jeden globální App.xaml, jenž obsahuje kód, který bude k dispozici napříč aplikací. Do tohoto souboru budou doplňovány různé statické zdroje, jako jsou styly, texty či konvertory. Jako první můžeme vložit statický text s názvem aplikace. Název aplikace může být zobrazován na různých místech v různých stránkách. Kdyby došlo ke změně názvu služby, lze úpravou tohoto jediného statického textu upravit textový řetězec napříč celou aplikací. Jedná se tedy o jakousi konstantu a definujeme ji vložením následujících řádků:

```
<Application.Resources>
    <x:String x:Key="AppName">eÚčty.cz</x:String>
</Application.Resources>
```

V případě, že budeme chtít zobrazit tento text kdekoli v aplikaci, dosadíme do XAML kódu na dané místo následující řetězec:

```
{StaticResource AppName}
```

Klíčové sousloví `StatitResource` nám říká, že bude řetězec hledán ve statických zdrojích definovaných na dané stránce a také právě v souboru App.xaml. `AppName` je pak identifikátor zdroje.

3.2.2 PŘIHLAŠOVACÍ STRÁNKA

Při návrhu jednotlivých stránek se budeme řídit návrhem rozložení uživatelského prostředí. Většinou budeme využívat tzv. mřížku, v níž definujeme řádky a sloupce, do kterých budeme umisťovat obsah. Výjimkou bude jedině stránka PivotPage.

Návrh si podrobně ukážeme na stránce s formulářem pro přihlášení. Protože jsme stránku vytvořili ze šablony, je základní struktura již vytvořena a je potřeba doplnit obsahovou část. Ta bude obsahovat dvě textová pole pro zadání uživatelského jména a hesla, přepínač automatického přihlašování, textový blok a tlačítka pro přihlášení a zavření aplikace.

Mřížku si v tomto případě rozdělíme na dva stejně široké sloupce a dva řádky. Druhý řádek se bude přizpůsobovat obsahu, který v něm bude umístěn, první řádek pak bude mít takovou výšku, aby byla vyplněna zbývající část obrazovky. Toho se docílí tzv. hvězdičkovou syntaxí a parametrem `Auto`.

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="*"/>
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
</Grid>
```

V prvním řádku bude umístěno vše, mimo tlačítka. Ta budou umístěna až v řádku druhém, který se přizpůsobí jejich velikosti. Tímto se tlačítka zafixují u spodního okraje displeje. Sloupce budeme využívat jen pro tlačítka, kdy se každé z nich umístí do jednoho sloupce, aby obě byla stejně velká a přizpůsobovala se šířce displeje. Umístění se provede tak, že se prvkům přidá atribut `Grid.Row`, resp. `Grid.Column`, jejichž hodnota bude odpovídat indexu řádku, resp. sloupce. Pokud je potřeba některé prvky zobrazit přes více řádků, resp. sloupců, nastaví se hodnota atributu `Grid.RowSpan` resp. `Grid.ColumnSpan` na číslo odpovídající počtu těchto řádků, resp. sloupců.

Protože se ale v buňce mřížky umísťují všechny prvky nezávisle na sobě, zobrazily by se přes sebe. Proto je potřeba je umístit do tzv. Stack Panelu. Výsledný kód pro umístění všech prvků stránky vypadá následovně:

```
<Grid>
    <Grid.RowDefinitions>
        ...
    </Grid.ColumnDefinitions>
    <StackPanel Grid.ColumnSpan="2">
        <TextBox Header="Uživatelské jméno" />
        <PasswordBox Header="Heslo" />
        <ToggleSwitch Header="Automatické přihlašování"
                  OnContent="aktivní" OffContent="neaktivní" />
        <TextBlock Text="K přihlášení použijte stejné přihlašovací
údaje jako pro www.eucty.cz." />
    </StackPanel>
    <Button Content="Přihlásit se" Grid.Row="1"
        HorizontalAlignment="Stretch" />
</Grid>
```

Každý prvek je vhodné pojmenovat, obzvláště ty, které mohou být nějak svázány s aplikační logikou. To se provádí přidáním atributů `x:Name`. Funkčním prvkům, tedy například tlačítkům, je pak potřeba přiřadit název metody, která bude volána v případě interakce s tímto prvkem. Proto např. tlačítku pro přihlášení přidáme atribut události `Click`, jeho hodnotou bude název metody, která bude volána, tedy:

```
Click="PrihlasitSe_Click"
```

Protože existují mobilní telefony s různými rozlišeními displejů, je vhodné používat i předvolené styly, aby se například texty nezobrazovaly příliš malé či velké a podobně. Styl daného objektu určíme pomocí atributu Style, v jehož hodnotě se bude odkazovat na statický předdefinovaný zdroj. Celý atribut pro textový blok bude vypadat následovně:

```
Style="{StaticResource BaseTextBlockStyle}"
```

Dále ještě obalíme celou mřížku s obsahem tagem `<ScrollViewer>`, který umožní posouvání, tedy tzv. panning v momentě, kdy se celý obsah nevejde na displej. V našem návrhu se zdá, že je na stránce místa dost, ale zkusme zařízení otočit na šířku – při tomto zobrazení se na displej již nevleze tlačítko pro přihlášení a ukončení aplikace a scrollování je tedy nutné. Navíc je vhodné počítat i do budoucna s tím, že by mohlo být potřeba přidat na stránku nějaký další prvek.

Na závěr upravíme hlavičku přihlašovací stránky, kde místo jejího názvu a jména aplikace vložíme logo pomocí tagu `<Image>` a do App.xaml vložíme následující kód, kterým přebarvíme pozadí stisknutých tlačítek a přepínače, okraje aktivních polí a barvu označeného textu na přesně takovou žlutooranžovou, která je použitá v internetové verzi služby eÚčty.cz.

```
<SolidColorBrush x:Key="ButtonPressedBackgroundThemeBrush"
Color="#FFF0AC25" />
<SolidColorBrush x:Key="ToggleSwitchCurtainBackgroundThemeBrush"
Color="#FFF0AC25" />
<SolidColorBrush x:Key="TextBoxPlaceholderTextThemeBrush"
Color="#FFF0AC25" />
<SolidColorBrush x:Key="TextSelectionHighlightColorThemeBrush"
Color="#FFF0AC25" />
```

Těmito jednoduchými úpravami získáme čistý a nenáročný design. Na obrázcích níže je zobrazen výsledný design a to jak pro tmavý, tak i pro světlý motiv. Třetí obrázek zobrazuje, jak je definována mřížka.

Obrázek 10: Design přihlašovací obrazovky

Zdroj: Vlastní zpracování

3.2.3 OSTATNÍ STRÁNKY

Ostatní stránky navrhneme obdobně, jako stránku pro přihlášení. Přidáme však ještě ovládací tlačítka do lišty na spodní straně displeje. Jejich ikony by měly jasně reprezentovat, jakou funkci tlačítka zastávají. Abychom je však nemuseli složitě vytvářet, lze použít mnoho připravených ikon. Výhodou je i to, že tyto ikony jsou součástí systému, konkrétně obrázkového fontu Seoge UI Symbol, a aplikace tak nebude muset obsahovat další obrázky, díky čemuž nebude narůstat její velikost.

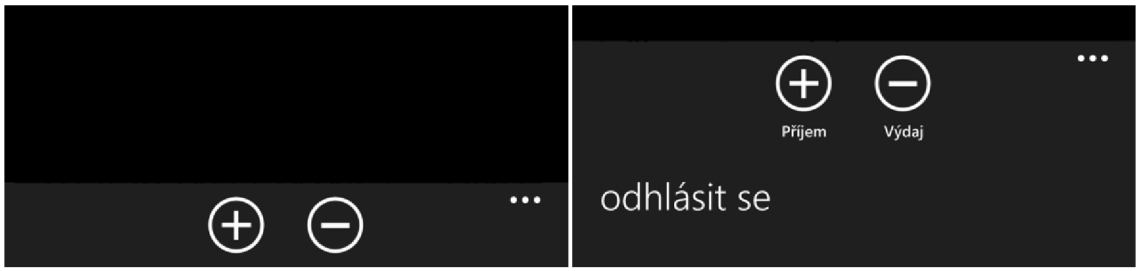
Ovládací tlačítka se vkládají do tagu `<CommandBar>`. Pokud bychom chtěli vložit i druhotná ovládací tlačítka, které se zobrazí po kliknutí na symbol tří teček, vložíme je do subtagu `<CommandBar.SecondaryCommands>`. Ta však už nemohou mít ikony.

Celý kód se pak vloží do lišty, která se zobrazuje na spodní straně displeje a je definována tagem `<Page.BottomAppBar>`. V následující části kódu je uveden příklad pro zobrazení dvou hlavních tlačítek pro vložení příjmu a výdaje a sekundárního tlačítka pro odhlášení uživatele.

```
<Page.BottomAppBar>
    <CommandBar>
```

```
        <AppBarButton Label="Příjem" Icon="Add" />
        <AppBarButton Label="Výdaj" Icon="Remove" />
        <CommandBar.SecondaryCommands>
            <AppBarButton Label="odhlásit se" />
        </CommandBar.SecondaryCommands>
    </CommandBar>
</Page.BottomAppBar>
```

Vizuální podoba uvedeného kódu je pak vidět na následujícím obrázku:



Obrázek 11: Command Bar zabalený a rozbalený

Zdroj: Vlastní zpracování

3.2.4 ZOBRAZENÍ SEZNAMŮ

V aplikaci se bude pracovat se seznamy účtu a transakcí. Data do nich budou přenášena pomocí Data Bingingu a proto se ještě podíváme na to, jak připravit stránky MainPage a PivotPage, kde tyto seznamy budeme využívat.

Základem je určení samotného prostoru, ve kterém se má seznam zobrazovat, pomocí tagu `<ListView>`. Jednotlivé položky se pak přidávají pomocí tagu `<ListViewItem>`, ale to v našem případě nevyužijeme. Protože data do seznamu budeme přenášet z aplikační vrstvy, vytvoříme šablonu položek a do seznamu přidáme pouze atributy `ItemTemplate`, kterým se budeme na tuto šablonu odkazovat, `ItemSource`, kterým definujeme zdroj dat a případně `IsItemClickEnabled` nastavíme na hodnotu `True`, aby bylo možné na jednotlivé položky seznamu kliknout. Výsledný seznam by pak pro stránku MainPage mohl vypadat takto:

```
<ListView x:Name="LvPrehled" ItemsSource="{Binding Ucty}"
          ItemTemplate="{StaticResource Ucet}"
          IsItemClickEnabled="True" />
```

Nyní je ale ještě potřeba vytvořit zmíněnou šablonu samotných položek. Protože se v atributu `ItemTemplate` odkazujeme na statický zdroj, je zřejmé, že šablona se bude nacházet právě mezi statickými zdroji pro danou stránku (což je tento případ), nebo pro celou aplikaci, je-li stejná šablona potřeba na více stránkách.

Tato šablona se uzavírá do tagu `<DataTemplate>` a musí obsahovat atribut `x:Name`, resp. `x:Key`, pomocí něhož se na šablonu odkazuje. Každá položka může mít svoji vlastní mřížku – v kombinaci s parametrem `HorizontalContentAlignment` a hodnotou `Stretch` každé položky je to vhodné zejména v případě, kdy chceme položku podbarvit v celé šíři obrazovky nebo kombinovat zarovnání prvků k levému i pravému okraji displeje. Pokud bychom mřížku nepoužili, každá položka by byla jen tak široká, jak široký je její obsah.

Šablona, která prezentuje jednu položku seznamu s účty na stránce MainPage, může vypadat následovně:

```
<DataTemplate x:Name="Ucet">
    <Grid Margin="0,0,0,20">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <TextBlock Grid.ColumnSpan="2" Text="{Binding Nazev}" />
        <TextBlock Grid.Row="1" Text="{Binding Typ}" />
        <StackPanel Grid.Row="1"
                    Grid.Column="1"
                    Orientation="Horizontal"
                    HorizontalAlignment="Right">
            <TextBlock Text="{Binding Castka, Padding="0,0,5,0" />
            <TextBlock Text="{Binding Mena}" Width="30" />
        </StackPanel>
    </Grid>
</DataTemplate>
```

Bindování dat probíhá tak, že pro každý objekt modelu definovaného v aplikační logice, je podle šablony vytvořena položka seznamu, do které jsou předány vlastnosti modelu. Ve výše uvedené části kódu to konkrétně jsou vlastnosti `Nazev`, `Typ`, `Castka` a `Mena` modelu `Ucty`.

Stejným způsobem vytvoříme i seznam transakcí na stránce PivotPage.

3.3 APLIKAČNÍ LOGIKA

Základem pro fungování aplikace je komunikace se serverem služby eÚčty.cz. Na úvodu třetí kapitoly byl zmíněn problém, který vznikl ukončením podpory protokolů SOAP a WSDL ve Windows Phone 8.1 ze strany Microsoftu. Právě tyto protokoly však používá služba eÚčty.cz v nabízeném API a proto je nutné vytvořit vlastní metody, kterými bude komunikace zpracovávána.

3.3.1 KOMUNIKACE SE SERVEREM

Abychom mohli k metodám, potřebným pro komunikaci se serverem, přistupovat odkudkoli v rámci celé aplikace, vytvoříme nový soubor Data.cs, do jehož hlavní třídy `Data` budeme vkládat všechny tyto metody.

Nejdříve se ale seznámíme se samotným rozhraním. To je realizováno technologií Web Service a podporuje protokoly SOAP, HTTP GET a HTTP POST. Každá metoda rozhraní má následující parametry:

- apiClientKey – veřejný klíč unikátní pro každou aplikaci využívající rozhraní,
- userLogin – uživatelské jméno,
- userPassword – uživatelské heslo.

Některé metody pak mají další parametry, jako je např. id mazané transakce.

V naší aplikaci budeme pro jednoduchost využívat protokol HTTP GET. Komunikace prostřednictvím tohoto protokolu probíhá tak, že aplikace vyšle GET požadavek na načtení internetové adresy, která směruje na konkrétní metodu rozhraní a pomocí parametrů v adrese jí předává požadované hodnoty. Jako odpověď pak dostává strukturovaný XML soubor. Např. adresa pro ověření uživatelského jména a hesla bude mít tvar:

```
http://eucty.cz/API/APIservice.asmx/TestLogin?APIClientKey=string&use
rLogin=string&userPassword=string
```

Struktura odpovědi pak bude vypadat následovně:

```
<?xml version="1.0" encoding="utf-8"?>
<apiResultBooleanMessage xmlns="http://www.eucty.cz/API">
  <ResultIsOk>boolean</ResultIsOk>
  <ResultMessage>string</ResultMessage>
  <ResultExceptionCode>long</ResultExceptionCode>
  <IsOk>boolean</IsOk>
  <Message>string</Message>
</apiResultBooleanMessage>
```

Vyslání požadavku je tedy jedna část komunikace, zpracování přijatého XML souboru část druhá a značně složitější. Navíc každá metoda vrací soubor s lišící se strukturou, je tedy potřeba naprogramovat metody, které přijaté soubory individuálně zpracují.

Jedno však metody budou mít společné a to je samotné čtení těchto souborů. Proto jako první vytvoříme univerzální statickou metodu, pomocí které vyšleme požadavek a odpověď načteme jako stream, tedy sekvenci bajtů, který vrátíme k dalšímu zpracování. Aby metoda mohla být opravdu univerzální, je potřeba požadavek http get sestavit v jejím těle. Proto bude mít metoda několik vstupních parametrů, mezi něž patří adresa metody, uživatelské jméno a heslo, případně další parametry. Protože adresa webové služby bude vždy stejná a veřejný klíč aplikace taktéž, vytvoříme pro tyto položky konstanty. Výsledná metoda vypadá následovně:

```
private const string ApiUri = "http://eucty.cz/API/APIservice.asmx/";
private const string ClientKey = "23ebaxxxxx";

public static Stream NacistXml(string operation, string userName,
string userPass, string otherParam = null)
{
    Uri dataUri = new Uri(ApiUri + operation + "?APIClientKey=" +
ClientKey + "&UserLogin=" + userName + "&userPassword=" + userPass +
otherParam);
    HttpClient client = new HttpClient();
```

```
    Stream stream = await client.GetStreamAsync(dataUri);
    return stream;
}
```

Metody, jejichž vykonání může trvat déle než zpravidla 50 ms, je doporučeno volat asynchronně. Načítání dat z internetu je typickým příkladem, kdy tato doba bývá překročena a je proto vhodné metodu změnit na asynchronní přidáním klíčového slova `async` a převedení návratového typu na `Task<Stream>`:

```
public static async Task<Stream> NacistXml(string operation, string
userName, string userPass, string otherParam = null) { … }
```

3.3.2 ZPRACOVÁNÍ XML SOUBORU

XML soubor, který vrací metoda `NacistXml`, je dále nutné zpracovat. To provedeme tzv. parsováním, kdy hodnotu každého atributu tohoto souboru uložíme do vlastností objektu odpovídající třídy. S těmito objekty pak můžeme dále pracovat. Podrobně si takové zpracování ukážeme na metodě, která bude načítat seznam účtů.

Z dokumentace k poskytovanému API lze vyčíst, že všechny metody vrací výsledek odvozený od třídy `apiResult`, proto v našem kódu definujeme nejdříve tuto třídu, kterou poté budou další třídy dědit.

```
public class apiResult
{
    public bool ResultIsOk { get; set; }
    public string ResultMessage { get; set; }
    public int ResultExceptionCode { get; set; }
}
```

Struktura XML souboru, který vrací seznam účtů, je následující:

```
<?xml version="1.0" encoding="utf-8"?>
<apiResultAccountList xmlns="http://www.eucty.cz/API">
  <ResultIsOk>boolean</ResultIsOk>
  <ResultMessage>string</ResultMessage>
  <ResultExceptionCode>long</ResultExceptionCode>
```

```
  <Data>
    <apiAccount>
      <Id>long</Id>
      <Name>string</Name>
      <Type>long</Type>
      <Currency>long</Currency>
      <IsActive>boolean</IsActive>
      <Locked>boolean</Locked>
      <Description>string</Description>
      <Ord>long</Ord>
      <UserAccountPermission>long</UserAccountPermission>
      <Amount>double</Amount>
      <IsHidden>boolean</IsHidden>
    </apiAccount>
    <apiAccount>
      …
    </apiAccount>
  </Data>
</apiResultAccountList>
```

Následně vytvoříme třídu, která strukturou přesně odpovídá přijímanému XML souboru. Třída dědí vlastnosti z `apiResult` a dále definuje podtřídu `apiAccount`, která představuje účet. V třídě `apiResultAccountList` je dále definována vlastnost `Data`, což je seznam objektů majících třídu `apiAccount`, tedy seznam účtů:

```
public class apiResultAccountList : apiResult
{
    public apiAccount[] Data { get; set; }
    public class apiAccount
    {
        public long Id { get; set; }
        public string Name { get; set; }
        public long Type { get; set; }
        public long Currency { get; set; }
        public bool IsActive { get; set; }
        public bool Locked { get; set; }
        public string Description { get; set; }
        public long Ord { get; set; }
```

```
        public long UserAccountPermission { get; set; }
        public double Amount { get; set; }
    }
}
```

Nyní vytvoříme ještě třídu, kterou použijeme při vytváření datového modelu. Ten nám pak poslouží pro bindování dat.

```
public class Ucet
{
    public Ucet(long id, string nazev, string popis, double castka)
    {
        this.Id = id;
        this.Nazev = nazev;
        this.Popis = popis;
        this.Castka = castka;
    }
    public long Id { get; set; }
    public string Nazev { get; set; }
    public string Popis { get; set; }
    public double Castka { get; set; }
}
```

Nakonec vytvoříme samotnou metodu, která bude data z XML souboru parsovat s vlastnostmi třídy a poté přidávat objekty s těmito vlastnostmi do kolekce a tím vytvoří datový model.

Protože metoda bude využívat asynchronní metodu `NacistXml`, musí být sama asynchronní.

```
public ObservableCollection<Ucet> seznamUctu = new
ObservableCollection<Ucet>();
public async Task NacistUctyXml()
{
    // Načítání dat je přeskočeno, pokud již jednou proběhlo
    if (this.seznamUctu.Count != 0)
        return;
```

```
    Stream streamUcty = await NacistXml("AccountListWithAmount",
userLogin.UserName, userLogin.UserPass);
    // Inicializace XmlSerializéru
    XmlSerializer serializerUcty = new
XmlSerializer(typeof(apiResultAccountList), ApiNamespace);
    // Převedení XML do objektu třídy apiResultAccountList
    apiResultAccountList uctyXml = new apiResultAccountList();
    using (StreamReader streamReader = new StreamReader(streamUcty))
    {
        uctyXml =
(apiResultAccountList)serializerUcty.Deserialize(streamReader);
    }
    // Vytváření nových objektů typu Ucet a přidání do kolekce
    foreach (var ucetXml in uctyXml.Data)
    {
        this.SeznamUctu.Add(new Ucet(ucetXml.Id, ucetXml.Name,
ucetXml.Description, ucetXml.Amount));
    }
}
```

Protože takto vytvořená kolekce je veřejná a bylo by možné její data upravovat odkudkoli v rámci celé aplikace, což není zcela v souladu s uznávanými postupy v programování, změníme ji na privátní, stejně tak i metodu `NacistUctyXml`. Abychom k datům mohli nějak přistupovat, musíme vytvořit ještě veřejnou metodu, která bude volat metodu `NacistUctyXml` a její návratovou hodnotou bude kolekce objektů, tedy seznam účtů.

```
public static async Task<ObservableCollection<Ucet>> NacistUcty()
{
    await data.NacistUctyXml();
    return data.seznamUctu;
}
```

Podobně vytvoříme metody pro všechny metody webového rozhraní, tedy pro načítání transakcí, měn, kategorií, vkládání, editaci a mazání transakcí a ověřování uživatelského jména a hesla.

3.3.3 BINDOVÁNÍ DAT

První bindování dat se v aplikaci provádí po přesměrování na MainPage – zavolá se funkce, která vytvoří a naplní datový model a předá jej ViewModelu, který se postará o zobrazení těchto dat.

```
this.DefaultViewModel["Ucty"] = await Data.NacistUcty();
```

V hranatých závorkách metody `DefaultViewModel` je uveden identifikátor, pomocí kterého se určuje, která data se mají kam vložit. V tomto případě se jedná o `Ucty` a pokud se podíváme zpět do XAML souboru této stránky, ten stejný identifikátor v parametru, který definuje zdroj dat:

```
<ListView x:Name="LvPrehled"
          ItemsSource="{Binding Ucty}"
          ItemTemplate="{StaticResource Ucet}"
          IsItemClickEnabled="True" />
```

Obdobně se budou data bindovat i v ostatních částech aplikace.

3.3.4 OVĚŘENÍ INTERNETOVÉHO PŘIPOJENÍ

Pro fungování této aplikace je nutné internetové připojení. Proto by měla být součástí kódu metoda, která ověří, zda je připojení k internetu dostupné a pokud ne, zobrazí informativní zprávu. Vytvoříme tedy metodu, která zkontroluje internetové připojení.

```
private static bool StatusPripojeni()
{
    ConnectionProfile connectionProfile =
       NetworkInformation.GetInternetConnectionProfile();
    return (connectionProfile != null &&
       connectionProfile.GetNetworkConnectivityLevel() ==
       NetworkConnectivityLevel.InternetAccess);
}
```

Dále vytvoříme metodu, která bude zobrazovat hlášku o nedostupnosti internetového připojení se dvěma tlačítky – zkusit znovu a zrušit. Pokud uživatel zvolí „zkusit znovu“, aplikace znovu zkontroluje připojení a případně znovu zobrazí tuto hlášku. Pokud

uživatel zvolí zrušit, aplikace přeruší provádění požadované operace. Metoda bude tedy využívat cykly.

```
public async static Task<bool> KontrolaPripojeni()
{
    IUICommand command;
    do
    {
        if (StatusPripojeni() == false)
        {
            command = await Data.ZobrazitMsg("Aplikace využívá
internetové připojení, které ale není aktuálně dostupné." +
Environment.NewLine + "Zkontrolujte internetové připojení.", "Žádné
internetové připojení!", "zkusit znovu", "zrušit");
        }
        else
        {
            return true;
        }
    } while (command.Label.Equals("Zkusit znovu"));
    return StatusPripojeni();
}
```

Tato metoda bude volána vždy před operací, která posílá či načítá data z internetu, aby se zamezilo neočekávaným chybám, které by mohly nastat v případě, že by se internetové připojení ztratilo v průběhu používání aplikace. Metoda využívá pomocnou metodu pro zobrazování dialogových oken:

```
public static async Task<IUICommand> ZobrazitMsg(string zprava,
string nadpis = null, string tlacitkoOk = null, string tlacitkoCancel
= null)
{
    MessageDialog msg = new MessageDialog(zprava);
    if (nadpis != null) msg.Title = nadpis;
    if (tlacitkoOk != null)
    {
        msg.Commands.Add(new UICommand(tlacitkoOk));
        msg.DefaultCommandIndex = 0;
```

```
    }
    if (tlacitkoCancel != null)
    {
        msg.Commands.Add(new UICommand(tlacitkoCancel));
        msg.CancelCommandIndex = 1;
    }
    return await msg.ShowAsync();
}
```

3.3.5 AUTOMATICKÉ PŘIHLAŠOVÁNÍ

Přihlašování u této aplikace funguje pouze na principu ověření uživatelského jména a hesla odesláním na server, který odešle zpět zprávu o tom, zda údaje odpovídají. V aplikaci toto ověření obstarává metoda `Login`, která vrací hodnoty typu `bool`. Pokud je ověření úspěšné, provede se přesměrování na stránku `MainPage` a vymaže se tzv. `BackStack`, která uchovává informace o předchozích stránkách. To z toho důvodu, aby při stisknutí hardwarového (u levnějších modelů telefonu softwarového) tlačítka zpět nebyl zobrazován znovu formulář pro přihlášení, ale aplikace byla ukončena.

Přihlašovací údaje je ovšem potřeba odesílat i při volání každé z metod, proto je potřeba údaje uložit, aby je uživatel nemusel při každé komunikaci se serverem znovu zadávat. Data lze ukládat dočasně, nebo trvale. V tomto případě budeme data ukládat trvale, aby při příštím spuštění aplikace bylo předvyplněno uživatelské jméno.

Po vzoru aplikace Moje O2, která je dostupná na Windows Phone Store využívá i naše aplikace přepínač pro automatické přihlašování. V případě, že bude automatické přihlašování zapnuto, bude do přihlašovacího formuláře vyplněno i heslo a následně provedeno automatické přihlášení. K tomu je potřeba uchovávat v paměti údaj i o tomto přepínači.

Nejdříve vytvoříme třídu a následně vytvoříme nový objekt této třídy.

```
public class UserLogin
{
    public string UserName { get; set; }
    public string UserPass { get; set; }
    public bool AutoLogin { get; set; }
```

```
}
public static UserLogin userLogin = new UserLogin();
```

Dále vytvoříme metody pro ukládání uživatelského jména, hesla a stavu přepínače do paměti a také pro jejich načítání při opětovném spuštění aplikace.

```
public static void UlozitLogin()
{
    ApplicationDataContainer nastaveni =
           ApplicationData.Current.LocalSettings;
    nastaveni.Values["UserName"] = userLogin.UserName;
    nastaveni.Values["UserPass"] = userLogin.UserPass;
    nastaveni.Values["AutoLogin"] = userLogin.AutoLogin;
}

public static void NacistLogin()
{
    ApplicationDataContainer nastaveni =
           ApplicationData.Current.LocalSettings;
    if (nastaveni.Values.ContainsKey("UserName"))
    {
        userLogin.UserName = nastaveni.Values["UserName"].ToString();
    }
    if (nastaveni.Values.ContainsKey("UserPass"))
    {
        userLogin.UserPass = nastaveni.Values["UserPass"].ToString();
    }
    if (nastaveni.Values.ContainsKey("AutoLogin"))
    {
        userLogin.AutoLogin = (bool)nastaveni.Values["AutoLogin"];
    }
}
```

Při načtení vstupní stránky LoginPage pak ověříme, zda už objekt `userLogin` existuje a pokud ne, provede se pokus o načtení dat z paměti telefonu. Pokud jsou data nalezena, zachová se aplikace, jak bylo popsáno výše, tedy vyplní uživatelské jméno či provede automatické přihlášení.

3.3.6 PROGRESS CONTROLS

Aplikace by měla okamžitě reagovat na akce vyvolané v uživatelském rozhraní, ale z důvodu načítání dat z internetu to není zcela možné. Může nastat situace, kdy se např. po přihlášení provede okamžité přesměrování na MainPage, ovšem data ještě nebudou načtena a seznam účtů bude tedy prázdný. Aby uživatel věděl, že se data stále načítají, je vhodné jej o tomto informovat pomocí tzv. Progress Controls. Tento prvek může mít více vizuálních variant, funkci však zastává vždy stejnou – informuje uživatele o probíhající akci případně i o jejím pokroku.

Obrázek 12: Vizuální možnosti pokrokových indikátorů

Zdroj: Vlastní zpracování dle [20]

Pro tuto aplikaci zvolíme neurčitý Progress Bar, a protože načítání dat z internetu je prováděno v rámci aplikace několikrát, zobrazíme jej v horní části aplikace, aby příliš neobtěžoval. Ve Windows.UI.ViewManagement je definována třída StatusBar, kterou pro zobrazení a skrývání použijeme. Pro větší přehlednost lze použít také text, který se zobrazí u Progress Baru a může tak uživatele informovat o aktuálně prováděné operaci, tedy např. „Přihlašování“, „Ukládání“, „Načítání“ apod.

Pro zpřehlednění kódu vytvoříme následující metodu `ZobrazitPB`:

```
public static async Task ZobrazitPB(bool zobrazit = true, string text
= "Načítání")
{
    if (zobrazit)
    {
        StatusBar.GetForCurrentView().ProgressIndicator.Text = text;
        await
StatusBar.GetForCurrentView().ProgressIndicator.ShowAsync();
    }
    else
```

```
    {
        await
StatusBar.GetForCurrentView().ProgressIndicator.HideAsync();
    }
}
```

Tuto metodu pak budeme volat před zahájením komunikace se serverem a s hodnotou `false` prvního atributu po jejím ukončení. Tím se Progress Bar zobrazí po dobu načítání či ukládání dat, tedy např. po dobu načítání účtů a transakcí či ukládání nové transakce.

3.3.7 OVĚŘENÍ VALIDITY DAT FORMULÁŘE

Při vyplňování formuláře pro vložení nové či editaci již uložené transakce může uživatel zadat neplatné znaky, které server v zadaném poli neočekává. Jedná se především o pole, do kterého uživatel vyplňuje částku náležící transakci. I když je uživateli nabídnuta klávesnice obsahující pouze číslice a desetinnou čárku, uživatel může, ať už úmyslně či omylem, vložit znaky, které neodpovídají desetinnému číslu. Po odeslání formuláře by server vrátil chybovou odpověď, proto je vhodné vložená data před odesláním zkontrolovat a případně umožnit uživateli jejich opravu.

Kontrolu provedeme pomocí regulérního výrazu. Protože server očekává desetinné číslo v anglickém formátu s desetinnou tečkou, převedeme částku do správného tvaru, ve kterém ji metoda bude vracet. V případě, že zadaná částka bude mít špatný formát, metoda vrátí prázdnou hodnotu.

```
public static string PrevodCastky(string castka)
{
    if (Regex.IsMatch(castka, "^-?[0-9]+,?[0-9]*$"))
    {
        double castkaCislo;
        double.TryParse(castka, out castkaCislo);
        return castkaCislo.ToString("F2", new CultureInfo("en-US"));
    } else {
        return null;
    }
}
```

V místech, kde bude potřeba částku ověřit (zpravidla před uložením transakce) zavoláme tuto metodu s parametrem, který bude obsahovat uživatelem zadanou částku. Pokud metoda vrátí hodnotu `null`, ověření považujeme za neúspěšné a necháme uživatele částku opravit. V případě, že návratová hodnota metody bude mít typ `string`, bude se jednat o korektní částku převedenou do anglického formátu, kterou použijeme k odeslání na server.

3.4 DISTRIBUCE APLIKACE

Po dokončení vývoje aplikace je potřeba aplikaci řádně otestovat a ideálně nabídnout k odzkoušení i lidem v okolí z různých věkových skupin pro zjištění případných nedostatků v intuitivnosti ovládání a získání první zpětné vazby. Testování lze provádět v emulátoru, ale mělo by být provedeno i ve fyzickém zařízení, tedy ve smartphonu.

Pokud se aplikace chová tak, jak se očekává, nepadá, je vhodné zkontrolovat podmínky, které je potřeba dodržet, aby aplikace prošla schvalovacím procesem. V aplikaci pro eÚčty.cz jsou některé podmínky porušeny. Jednou z nich je podmínka, která říká, že aplikace nesmí obsahovat ovládací prvky pro zavření aplikace. Před publikováním proto odstraníme tlačítko „Zavřít“ z přihlašovací stránky. Druhou je podmínka, která stanovuje, že pokud dochází k přenosu dat po síti, je třeba uživatele informovat o zásadách ochrany osobních údajů. Proto do aplikace přidáme jednoduchou stránku s popisem aplikace vč. informace o tom, že aplikace žádné osobní údaje neshromažďuje a nezpracovává. Odkaz na tuto stránku přidáme do druhotných ovládacích prvků ve spodní liště aplikace.

V případě, že jsou všechny podmínky splněny, lze přistoupit k samotnému publikování aplikace. K tomu je však potřeba být registrovaným vývojářem u společnosti Microsoft a mít aktivní vývojářský účet. Studenti VUT na fakultě podnikatelské jsou zařazeni do programu Microsoft DreamSpark a mohou se jako vývojáři registrovat zdarma. Ostatní vývojáři či firmy musí platit roční aktivační poplatek v řádech desítek EUR.

3.4.1 SESTAVENÍ BALÍČKU APLIKACE

Pokud jsme registrovanými vývojáři, můžeme přistoupit k samotnému sestavení balíčku aplikace, tedy jediného souboru, který později odešleme ke schválení.

Sestavování se provádí přímo ve vývojovém prostředí, tedy Visual Studiu, v nabídce Projekt – Store. Ze všeho nejdříve je potřeba rezervovat unikátní jméno aplikace, tedy název, který bude výstižný, ale přitom nebude ještě ve Windows Phone Store použitý. Dále je potřeba určit umístění aplikace, její verzi, podporované procesory apod.

Po úspěšném sestavení balíčku je nabídnuto lokální ověření aplikací Windows App Certification Kit. Jedná se o soubor testů, kterým bude aplikace podrobena a které je doporučeno provést před odesláním ke schválení. Pokud některý z testů skončí neúspěchem, je zbytečné aplikaci do Windows Phone Store odesílat, protože stejnými testy prochází i při schvalování.

3.4.2 ODESLÁNÍ BALÍČKU APLIKACE

Pokud byly všechny testy úspěšně dokončeny, přistoupíme k samotnému odeslání aplikace do schvalovacího procesu. To se provádí pomocí webu Windows Phone Dev Center, na který nás Visual Studio přesměruje.

Po přihlášení k vývojářskému účtu vyplníme další informace o aplikaci, jako je zařazení do kategorie, cena, omezení distribuce a zda má dojít k automatickému publikování aplikace po schválení aplikace. Dále nahrajeme balíček aplikace, logo, které se má zobrazovat ve Windows Phone Store, alespoň jeden screenshot a vyplníme popis aplikace vč. popisu jejích funkcí a klíčových slov. Volitelně můžeme vyplnit informace o poskytovateli reklamy a nastavit ceny pro konkrétní země.

Poté aplikaci odešleme ke schválení. Schvalovací proces trvá obvykle několik hodin až dní, některé části jsou automatické, některé ale provádí člověk. Jednotlivé stavy schvalovacího procesu, ve kterém se aplikace nachází, lze sledovat ve vývojářském účtu vč. vysvětlení jejich významu. Těchto stavů je celkem 11, chtěný stav je označován slovem Published (v překladu zveřejněno) a říká nám, že aplikace prošla úspěšně schvalovacím procesem a byla zahrnuta do Windows Phone Store. Do 24 hodin se pak aplikace v obchodě začne zobrazovat.

3.4.3 REPORTY APLIKACE

Součástí Windows Phone Dev Center jsou i statistické informace o počtu stažení, nákupů či pádů aplikace. Právě neočekáváné ukončení aplikace je dobré sledovat a v případě

jejich výskytů analyzovat důvody těchto chyb a opravit je vydáním aktualizace. Ta se pak sestavuje a nahrává ke schválení obdobným způsobem.

3.5 EKONOMICKÉ ZHODNOCENÍ

Základní verze aplikace bude poskytována zdarma, proto nelze mluvit o přímém ekonomickém zhodnocení. Vývoj této aplikace mi však přinesl mnoho zkušeností a znalostí, které použiji při vývoji dalších aplikací. Zhodnocení pak bude přicházet s každou další vyvinutou aplikací ve formě úspory času, který bude potřebný pro vývoj.

Aplikaci zařadím také do svých referencí, což mi pomůže získat zákazníky, kteří budou potřebovat vyvinout aplikaci pro platformu Windows Phone. Jedná se tak vlastně o reklamu a zhodnocení aplikace lze srovnat s ušetřenými prostředky, které by bylo nutné vynaložit pro placenou reklamu. Navíc, pokud uživatelé aplikaci kladně ohodnotí, získám důvěryhodnost a případný zákazník si tak dokáže udělat představu o kvalitě aplikace.

Pro přímé ekonomické zhodnocení je možné doprogramovat další funkce, které budou součástí pouze placené verze. Pokud uživatel bude tyto funkce chtít využít, bude muset zaplatit jednorázový poplatek. Druhou možností je příjem ze zobrazování reklamy přímo v aplikaci, ovšem tuto variantu pro tuto aplikaci zavrhuji, protože by narušila čistý a jednoduchý design a mohla by od používání samotné aplikace odrazovat.

ZÁVĚR

Chytré telefony s mobilním internetem se stávají neodmyslitelnou součástí každodenního života mnohých z nás. Aplikace pro tyto telefony pak mohou uživatelům některé činnosti ulehčit či zkrátit dobu nutnou, pro jejich vykonání. Smartphone tak lze přirovnat k mobilní kanceláři, jímž vyřídíme e-maily, telefonáty, objednávky či mnoho dalších, i specializovaných úkonů, a to bez ohledu na místo, kde se právě nacházíme.

Ačkoliv spousta vývojářských společností produkuje aplikace pro všechny tři nejrozšířenější operační systémy mobilních telefonů, najdou se stále tací, kteří nejmladší systém Windows Phone opomíjejí a zaměřují se pouze na Android a iOS. A to i přes to, že v procentuálním zastoupení na Evropském trhu systém Microsoftu upevňuje svoji pozici a v některých zemích je dokonce prodávanější než konkurenční systém od společnosti Apple.

Bakalářská práce se v teoretické části zabývá stručnou historií platformy Windows Phone, analýzou trhu s mobilními telefony a jejich operačními systémy, shrnuje doporučení pro návrh designu a představuje možnosti vývoje aplikací pro Windows Phone 8.1.

Praktická část práce popisuje návrh designu uživatelského rozhraní v jazyce XAML. Zohledněna jsou doporučení vydávané společností Microsoft i stávající vzhled internetové služby eÚčty.cz, pro níž byla aplikace vyvíjena. Důraz je kladen na intuitivní ovládání a jednoduchý, čistý design.

Navržena je i aplikační logika v jazyce C#. Podrobně práce rozebírá důležité metody pro odesílání požadavků na server, zpracování odpovědi serveru a bindování dat do uživatelského rozhraní. Dále jsou popsány metody pro bezproblémový běh aplikace, jako je kontrola internetového připojení a ověření validity uživatelem vložených dat. Nelze opomenout ani metodu, která umožňuje automatické přihlašování uživatele a tím urychlí zadávání transakcí.

V poslední kapitole jsou diskutovány možnosti ekonomického zhodnocení včetně nefinančního přínosu aplikace. Tímto byly splněny veškeré cíle bakalářské práce.

Výstupem bakalářské práce je fungující aplikace publikovaná ve Windows Phone Store, kterou si za týden od zveřejnění stáhlo 6 uživatelů.

SEZNAM POUŽITÝCH ZDROJŮ

[1] MICROSOFT. *News Center* [online]. © 2015 [cit. 2015-02-16]. Dostupné také z: http://news.microsoft.com/

[2] HRMA, Jiří. Platforma Pocket PC/Windows Mobile dnes slaví 10 let od svého vzniku. *SMARTmania.cz* [online]. SMARTmania, 2010 [cit. 2015-03-7]. Dostupné také z: http://smartmania.cz/clanky/platforma-pocket-pc-windows-mobile-dnes-slavi-10-let-od-sveho-vzniku-244

[3] VALDAUF, Zdeněk. *Vývoj aplikací pro Windows Phone „Mango"*. České Budějovice, 2013. Dostupné také z: www.petrpexa.cz/diplomky/valdauf.pdf. Bakalářská práce.

[4] Historie aktualizací Windows Phone 7. *Windows Phone* [online]. Microsoft, © 2015 [cit. 2015-03-15]. Dostupné také z: http://www.windowsphone.com/cs-cz/how-to/wp7/basics/update-history

[5] Historie aktualizací Windows Phone 8. *Windows Phone* [online]. Microsoft, © 2015 [cit. 2015-02-27]. Dostupné také z: http://www.windowsphone.com/cs-CZ/how-to/wp8/basics/windows-phone-8-update-history

[6] *PDAdb: The World's Largest Smartphone, Tablet, PDA, PNA, Netbook & Mobile Device Database* [online]. PDAdb, © 20 [cit. 2015-03-03]. Dostupné také z: http://pdadb.net/

[7] *MSDN-the microsoft developer network* [online]. Microsoft, © 2014 [cit. 2014-10-20]. Dostupné také z: http://msdn.microsoft.com/

[8] LACKO, Ľuboslav a Martin HERODEK. *Vývoj aplikací pro Windows 8.1 a Windows Phone*. 1. vyd. Brno: Computer Press, 2014, 328 s. ISBN 9788025138229.

[9] HOLUŠA, Václav. *Vývoj aplikací pro Windows Phone 8 a portace na Windows 8*. Brno, 2014. Dostupné také z: http://is.muni.cz/th/395668/fi_m/Diplomova_prace.pdf. Diplomová práce.

[10] *Windows Dev Center* [online]. Microsoft, © 2015 [cit. 2015-03-8]. Dostupné také z: https://dev.windows.com/

[11] KOCMAN, Rostislav. Jak v ČR používáme chytré mobily a tablety?. *Internet pro všechny* [online]. 2014 [cit. 2014-11-24]. Dostupné také z: http://www.internetprovsechny.cz/jak-v-cr-pouzivame-chytre-mobily-a-tablety/

[12] MALETÍNSKÝ, Václav. V roce 2014 se prodalo téměr 1,3 miliardy smartphonů. *Itbiz* [online]. Argonit, 2015 [cit. 2015-02-03]. Dostupné také z: http://www.itbiz.cz/zpravicky/v-roce-2014-se-prodalo-temer-1-3-miliardy-smartphonu

[13] Apple sales hold steady despite iPhone 6 pre-launch demand. *Kantar Worldpanel ComTech* [online]. London: TNS UK Limited, 2014 [cit. 2014-11-24]. Dostupné také z: http://www.kantarworldpanel.com/global/News/Apple-sales-hold-steady-in-Britain-despite-iPhone6-pre-launch

[14] STRAKA, Martin. Obchod Google Play poprvé předhonil AppStore v počtu aplikací. *Tabletnet.cz* [online]. 2015 [cit. 2015-02-05]. Dostupné také z: http://tabletnet.cz/android/4115-google-play-poprve-predhonil-appstore-v-poctu-aplikaci

[15] CAMERON, Sean. Windows Phone and Windows 8 Stores now have over 560,000 apps combined. *WinBeta* [online]. 2015 [cit. 2015-02-05]. Dostupné také z: http://www.winbeta.org/news/windows-phone-and-windows-8-stores-now-have-over-560000-apps-combined

[16] Jak si vybíráme chytrý telefon. *Seznam.cz: Výzkumník* [online]. Seznam.cz, 2014 [cit. 2015-01-23]. Dostupné také z: https://vyzkumnik.seznam.cz/zpravodaj/jak-si-vybirame-chytry-telefon

[17] CASTRO, Elizabeth a Bruce HYSLOP. *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek*. Brno: Computer Press, 2012, 439 s. ISBN 978-80-251-3733-8.

[18] BĚLOVSKÝ, Vojtěch. *Vývoj mobilních aplikací pro platformu Apple iOS*. Brno, 2013. Dostupné také z: http://is.muni.cz/th/374032/fi_b/Bakalarka.pdf. Bakalářská práce.

[19] LAŠ, Jan. *Eúčty.cz* [online]. 2014 [cit. 2014-11-20]. Dostupné také z: http://www.eucty.cz

[20] DreamSpark Premium. *MSDN – Microsoft Developer Network* [online]. Microsoft, © 2014 [cit. 2014-11-24]. Dostupné také z: http://msdn.microsoft.com/cs-cz/ff898347.aspx

SEZNAM OBRÁZKŮ, GRAFŮ A TABULEK

SEZNAM OBRÁZKŮ

SEZNAM GRAFŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

Příloha 1: API Service

Metoda	Popis metody
AccountList	Seznam účtů
AccountList2	Seznam účtů - rozšířené vlastnosti účtů
AccountListWithAmount	Seznam účtů se stavem na účtu
AccountListWithAmount2	Seznam účtů se stavem na účtu - rozšířené vlastnosti účtů
AddAccount	Přidání účtu
AddCarFuel	Přidání tankování
AddCategory	Přidání kategorie
AddTranferToOtherAccount	Přidání převodu na jiný účet
AddTransaction	Přidání transakce
CarFuelList	Seznam tankování pro auto (posledních 50)
CarList	Seznam aut
CategoryTree	Strom kategorií
CurrencyList	Seznam podporovaných měn
DefineTransactionList	Seznam šablon transakcí
DefineTransactionList2	Seznam šablon transakcí - rozšířené vlastnosti
DeleteAccount	Smazání účtu
DeleteCarFuel	Smazání tankování
DeleteCategory	Smazání kategorie (včetně všech podkategorií, transakce převede do 'Bez kategorie')
DeleteTransaction	Smazání transakce
EditAccount	Úprava účtu
EditCarFuel	Úprava tankovani
EditCategory	Úprava kategorie
EditTransaction	Úprava transakce
TestLogin	Ověření přihlašovacího jména a hesla
TransactionListLast50	Vrátí 50 posledních transakcí na daném účtu
TransactionListLast50_2	Vrátí 50 posledních transakcí na daném účtu
UserList	Seznam uživatelů