Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Vývoj aplikací pro OS Windows 8 a Windows 8 RT

Bakalářská práce

Autor: **Martin Brabec**

Vedoucí práce: Ing. Jiří Brožek Ph.D.

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačního inženýrství

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Brabec Martin

Informatika

Název práce

Vývoj aplikací pro OS Windows 8 a Windows RT

Anglický název

Development for Windows 8 and Windows RT

Cíle práce

Popsat a vysvětlit nejnovější technologie, nástroje, vzory a společnosti Microsoft, sloužící pro snadnější, rychlejší a efektivnější tvorbu aplikací pro dotykové rozhraní operačního systému Windows 8 (RT), ukázat si jejich využití na reálných aplikacích a zhodnotit, zda jsou navrženy vhodně, případně poté navrhnout možná alternativní řešení.

Metodika

Metodika bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Na základě syntézy teoretických poznatků budou shrnuty možnosti systému Windows 8/RT a dále budou vytvořeny příklady demonstrující různé vlastnosti systému.

Harmonogram zpracování

06/2013-12/2013 - studium a analýza informačních zdrojů
01/2013 - kontrola průběhu práce, zápočet
01/2013-03/2013 - tvorba příkladů, finalizace práce
03/2013 - odevzdání práce, zápočet

Rozsah textové části

35-40 stran

Klíčová slova

Windows 8, vývoj, software, Windows RT, C#, .NET

Doporučené zdroje informací

Freeman, Adam. (2012). Metro Revealed - Building Windows 8 Apps with HTML5 and JavaScript. New York 233 Spring Street, 6th Floor, NY 10013 : Springer Science+Business Media New York, 2012. 89 s. ISBN: 978-1-4302-4489-9

Brockschmidt, Kraig. Programming Windows 8 Apps with HTML, CSS and JavaScript. Redmond, Washington 98052-6399 : Microsoft Press, 2012. 833 s. ISBN: 978-0-7356-7261-1

Freeman, Adam. (2012). Metro Revealed - Building Windows 8 apps with XAML and C#. New York 233 Spring Street, 6th Floor, NY 10013 : Springer Science+Business Media New York, 2012. 112 s. ISBN:978-1-4302-4491-2

Sharp, John (2012). Microsoft VIsual C# 2010 : krok za krokem. Albatros Media a. s., Na Pankráci 30, Praha 4. 696 s. ISBN:978-80-251-3147-3

Burns, Kyle (2012). Beginning Windows 8 Application Development: XAML Edition. Apress, online. 328 s. ISBN:978-1-4302-4566-7

Vedoucí práce

Brožek Jiří, Ing., Ph.D.

Termín odevzdání

březen 2014

Ing. Martin Pelikán, Ph.D.
Vedoucí katedry

prof. Ing. Jan Hron, DrSc., dr. h. c.
Děkan fakulty

V Praze dne 12.9.2013

Čestné prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně s použitím literatury a pramenů uvedených v bibliografii.

V ..... dne .....

Podpis autora: .....

Poděkování

Rád bych poděkoval panu Ing. Jiřímu Brožkovi Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Vývoj pro Windows 8 a Windows 8 RT

Práce je zaměřena na základní popis nového uživatelského rozhraní pro operační systém Windows 8, představení vývojového prostředí a především na nové možnosti při vývoji aplikací pro Windows 8 a Windows 8 RT.

Klíčová slova: Windows 8, vývoj, software, Windows RT, C#, .NET

Development for Windows 8 and Windows 8 RT

This document is focused on basic description of new user interface for Windows 8 operating system, introduction to new development environment and mainly on new posibilities while developing Windows 8 and Windows 8 RT applications.

Keywords: Windows 8, development, software, Windows RT, C#, .NET

Obsah

1 Úvod

Operační systém od společnosti Microsoft je ve svých několika verzích již velmi dlouhou dobu nejrozšířenější operační systém na stolních počítačích a mezi klasickými notebooky. Trend vývoje elektroniky však ukazuje, že se tyto dva, dříve naprosto odlišné, typy zařízení postupně sjednocují a rozdíly smazávají. Stolní počítače a notebooky začali být nahrazovány jednoduchými tablety a mobilními telefony, a tak, chtě nechtě, Microsoft musel přijít s něčím novým, aby si udržel své zákazníky. S novou verzí svého operačního systému tak přinesl nejen mnohem větší přizpůsobení na dotykové ovládání prsty, ale také podporu nové architektury - ARM.

Ovšem představit v dnešní době nový, byť kvalitní, operační systém a jen doufat, že se rozšíří, by bylo velmi bláhové. S operačním systémem je nutné představit také tzv. ekosystém. To si lze představit jako jakési provázání operačního systému s jinými systémy, případně zařízeními, podobné grafické prostředí napříč všemi platformami pro daný ekosystém a v neposlední řadě sjednocením místa pro získávání nových aplikací a her.

V této práci bude popsána poslední zmíněná část, avšak z trošku jiného pohledu. Aby byl celý takový ekosystém úspěšný, nestačí pouze sjednotit místo pro získávání nových aplikací a her. Musí existovat také nějaký obsah, který toto místo vyplní. Jinak řečeno, musí existovat dostatečně velká základna vývojářů, která pro tento ekosystém bude vytvářet nové aplikace a hry. Taková základna ale nevznikne sama od sebe. Proto Microsoft spolu s novým uživatelským rozhraním představil také zcela nový způsob vývoje, který bude v této práci rozebrán a předveden.

2 Cíl a metodika

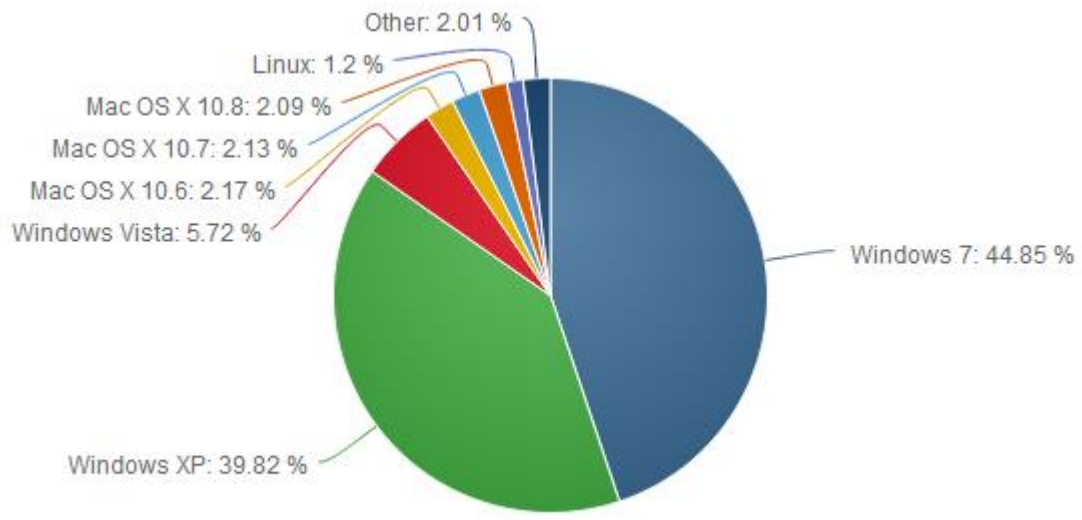
Cílem práce je popsat a vysvětlit nejnovější technologie, nástroje a vzory společnosti Microsoft, sloužící pro snadnější, rychlejší a efektivnější tvorbu aplikací pro dotykové rozhraní operačního systému Windows 8 (RT), ukázat si jejich využití na reálné aplikaci a zhodnotit, zda jsou navrženy vhodně, případně poté navrhnout možná alternativní řešení.

V této práci si nejdříve představíme nové uživatelské prostředí, jeho nové ovládací prvky a nové možnosti využití. Dále se zaměříme na vývojové prostředí, které společnost Microsoft nabízí pro vývoj aplikací a následně budou popsány nástroje pro implementaci nových ovládacích prvků. V praktické části práce bude poté na ukázkové aplikaci předvedena implementace několika ovládacích prvků a návrhových vzorů z předchozí části práce.

3 Nový operační systém Windows 8 (RT)

3.1 Windows 8

Windows 8, jakožto nová verze redmondského operačního systému, přináší nejen řadové a evoluční vylepšení. V oblasti prostředí a ovládání totiž přináší změny, které lze u systému Windows považovat za přímo revoluční.



Obrázek 1 - Rozdělení trhu operačních systémů na stolních počítačích - 4. čtvrtletí 2012 – ZDROJ: www.netmarketshare.com

Windows 8 byly představeny 26. Října 2012. Do té doby byl systém Windows připraven k používání na stolních PC, v kombinaci s myší a klávesnicí. Počet zařízení se systémem Windows, umožňujících ovládat systém dotykem bylo velmi málo. O dotykových zařízeních s operačním systémem iOS (Apple) nebo Android (Google) se toto říci nedalo. To byl pravděpodobně nejsilnější důvod Microsoftu pro nasazení nového uživatelského prostředí.

Systém Windows obdržel naprosto nové uživatelské prostředí, které je přizpůsobené pro ovládání dotykem (Microsoft, Get Started, 2013). Naštěstí se nejedná o náhradu klasického, obecně známého, prostředí s okny a ikonami. Lze se totiž mezi těmito dvěma rozhraními libovolně přepínat. Na novém prostředí je poznat, že při vývoji se začínalo od

nuly. Dokonce ani nebylo vývojáři příliš dbáno na již zaběhlé uživatelské zvyky. Výsledkem je sice nové, krásné prostředí, které je ale pro značnou část běžných uživatelů z velké části nepoužitelné. Ovládání dotykového prostředí je sice svým způsobem intuitivní, ale člověk nejdříve musí získat přehled o základních prvcích takového ovládání – gest, která budou konkrétněji popsána níže.

Microsoft si byl této neduhy vědom, a velmi se snažil a stále snaží, aby si všichni uživatelé zvykli na nové ovládání. Například tak, že po nainstalování, případně při prvním spuštění, se uživateli Windows 8 představí a na ukázkách mu předvedou několik základních ovládacích prvků (Microsoft, Get Started, 2013). Ty ostatní, méně používané, poté může uživatel najít v manuálu, nebo případně na webu http://windows.microsoft.com/cs-cz/windows-8/getting-around-tutorial (Pozn. - tento odkaz nyní odkazuje již na novější verzi Windows 8.1, které přinášejí několik dalších novinek, které jsou však nad rámec této práce).

3.1.1 Kompatibilita

Tento systém je zpětně kompatibilní s drtivou většinou aplikací, napsaných pro předchozí verze operačních systémů. A pokud nejsou, systém nabízí možnost spuštění aplikace v režimu kompatibility pro zvolený operační systém (Microsoft, Compatibility, 2013).

3.2 Windows 8 RT

Pro Windows 8RT platí téměř vše, co pro Windows 8. Hlavní rozdíl systému s přívlastkem „RT“ je jeho optimalizace pro procesory s architekturou ARM. Tato architektura bývá hojně využívána v mobilních zařízeních, jako například mobilní telefony a tablety. Procesory s architekturou ARM se oproti běžným architekturám x86 a x64 vyznačují především velmi nízkou spotřebou a schopností pracovat bez aktivního chlazení. Samozřejmě lze nalézt x86 procesory, které také nevyžadují aktivní chlazení. V mnoha případech však tyto procesory nedosahují kvalit adekvátního konkurenta s architekturou ARM. (Microsoft, Supporting Windows RT and ARM processors, 2013)

Jelikož jsou procesory ARM velmi oblíbené mezi výrobci mobilních zařízení a tabletů (DeGrasse, 2013), představení Windows 8 RT se na první pohled jeví jako krok správným směrem. Díky tomu by totiž Microsoft mohl dostat svůj „desktopový“ operační systém na rychle rostoucí trh s tablety. Avšak jak se postupem času ukazuje, Windows 8 RT jsou

mezi výrobci i zákazníky velmi neoblíbené, a to především kvůli některým odlišnostem oproti klasickým Windows 8 (Portnoy, 2013).

3.2.1 Kompatibilita

Všechny ostatní verze operačního systému Windows se vyznačovali velmi slušnou zpětnou kompatibilitou, avšak Windows 8 RT tuto tradici ruší. Microsoft neumožňuje na Windows 8 RT instalovat klasické, desktopové aplikace. Nelze tedy spustit téměř žádný soubor s příponou exe a nelze nainstalovat žádný software, který neprošel certifikačním procesem od Microsoftu (Microsoft, Windows App Certification Kit , 2013). Jedná se však o naprosto jiný proces, než ten, který je potřeba při vývoji aplikací pro dotykové rozhraní. Microsoft si velice pečlivě vybírá, které aplikace budou certifikovány pro Windows 8 RT. Mezi hrstku schválených aplikací tak patří například sada Office. Aplikace pro Windows RT totiž musí být sestaveny pro zcela jinou architekturu procesoru (Microsoft, Compatibility, 2013).

Hlavní nevýhody tohoto omezení jsou především zpětná nekompatibilita s aplikacemi pro starší verze systému Windows, což velkou část potencionálních zákazníků odradí. Avšak při pohledu na Windows 8 RT jako na „nový systém pro tablety, s možností přepnutí do desktopového režimu“, rázem se z hlavní nevýhody může stát výhoda. Na Windows 8 RT sice nelze spustit běžné exe aplikace, ale ani běžné viry. Zablokování spuštění necertifikovaných spustitelných souborů je zakódováno hluboko v jádře systému, a tak lze říci, že se jedná o mnohem bezpečnější systém.

S bezpečnostní souvisí i další věc. Jak se do tohoto systému dostanou nové aplikace? Pomocí Windows Store, který bude popsán níže. Přes tento online obchod lze nakupovat a získávat pouze aplikace, které prošli certifikací, a nejedná se tedy o bezpečnostní hrozbu (Microsoft, Get Started, 2013).

3.2.2 Rozdíly oproti Windows 8

- Není zde plná zpětná kompatibilita s aplikacemi pro starší verze OS Windows (vlastní zdroj)
- Chybějící Windows Media Player (vlastní zdroj)
- Nelze vytvořit doménu (ale lze se do ní připojit) (vlastní zdroj)
- Vzdálená plocha funguje pouze jako klient (vlastní zdroj)
- Zařízení, jako flash disky musí být tzv. „Windows 8 RT ready“, protože Windows 8 RT nedovolí nainstalovat vlastní ovladače (vlastní zdroj)

Jak lze vidět, až na onu zpětnou kompatibilitu, nejsou rozdíly natolik dramatické. Avšak jak již bylo naznačeno dříve, Windows 8 RT se nerozšířili podle očekávání a velcí výrobci od nich dávají ruce pryč. Tomu odpovídá i podíl operačních systémů na trhu v prvním kvartálu roku 2013, podle kterého náleželo Windows 8 RT na trhu s tablety pouhé půl procento (Mainelli, 2013).

3.3 Nové uživatelské prostředí

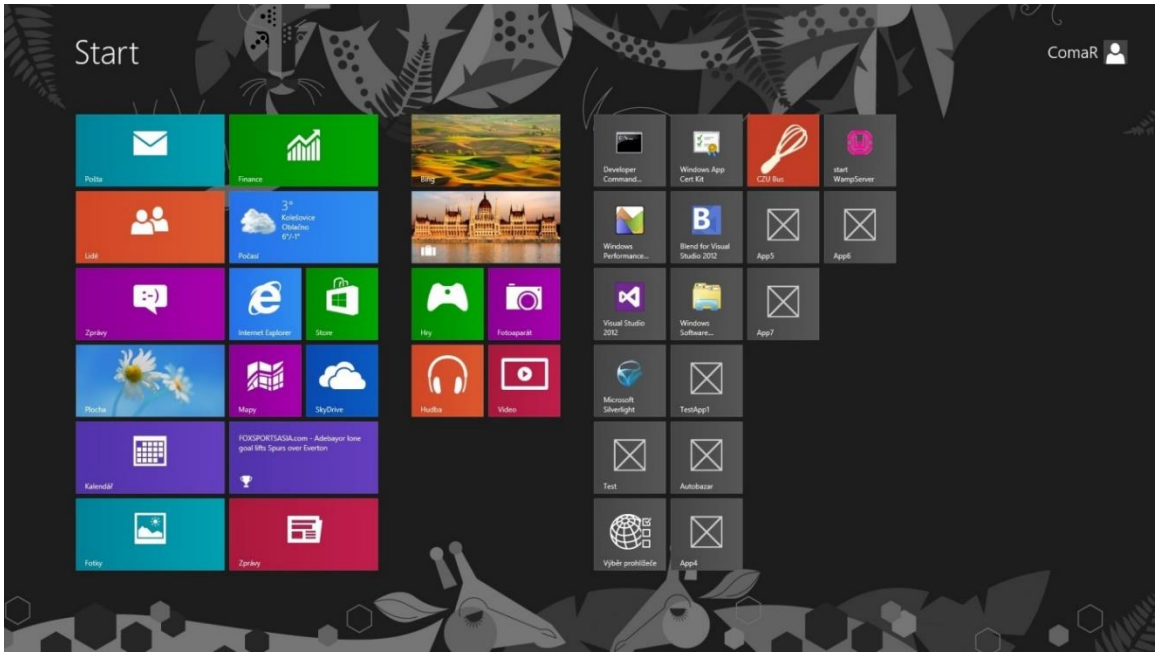
Windows 8 přišli s naprosto novým uživatelským rozhraním, uzpůsobeným především pro dotykové ovládání prsty. Pokud se však uživatel rozhodne používat pro ovládání klasickou myš a klávesnici, přijde o jistý komfort. Na prostředí je znát, že je primárně koncipováno pro dotyk, až poté pro ostatní ovládací zařízení.

Základem tohoto prostředí je minimalistický design, s hranatými prvky a bez různých 3D efektů. Všechny aplikace více či méně sdílejí různé ovládací prvky a koncepty. Jeden z nejzásadnějších designových konceptů je rozšiřování obsahu do strany, ne dolů. Díky tomu je konzumace obsahu přirozenější a více podobná listování v knize (Microsoft, Get Started, 2013). Absence stínování a odlesků má v systému také svůj význam. Uživatel není rozptylován obalem, ale může se plně soustředit na obsah.

Zároveň ale systém obsahuje efektní přechody a animace, které svým stylem podtrhují onen minimalistický design. Všechny animace v systému jsou tzv. „Fast and fluid“, což lze volně přeložit jako „Rychlé a plynulé“ (Microsoft, Windows User Experince Design Principles, 2013). Jelikož u Windows 8 bylo velmi zapracováno na výkonu a paralelním zpracování úloh, grafické úlohy mohly dostat vyšší prioritu zpracování. To v kombinaci s jednoduchým designem zajišťuje velmi příjemný pocit z používání (Microsoft, Windows User Experince Design Principles, 2013).

3.3.1 Startovací obrazovka

Tato podkapitola je zpracována podle článku „Designing the Start screen“ tehdejšího ředitele divize Windows, Stevena Sinfosky z roku 2011 (Sinfosky, 2011).



Obrázek 2 - Startovací obrazovka Windows 8 – ZDROJ: vlastní

Startovací obrazovka je oproti Windows 7 zcela přepracována. Skládá se z živých panelů, takzvaných „Live tiles“, které se mohou průběžně aktualizovat a je také možné změnit jejich velikost, umístění či barvu. Tyto živé panely bývají často označovány jako jakási okna aplikací. Každá aplikace pro dotykové prostředí Windows 8 má možnost vytvořit si takovýto panel. Ve zcela nejjednodušší formě pak tento panel slouží pouze jako obyčejný zástupce pro spuštění, podobně jako je tomu u klasických ikon. Značná část aplikací však využívá možnosti zobrazovat na svém panelu nějaká data. Například aplikace o počasí může zobrazovat aktuální počasí, hra může zobrazovat maximální dosažené skóre a fotogalerie může zobrazovat oblíbené obrázky.

Startovací obrazovka dovoluje seskupovat živé panely do skupin, měnit jejich velikost a uspořádání. Velmi zjednodušeně lze říci, že startovací obrazovka je jakási vylepšená plocha s vylepšenými ikonkami.

Naštěstí Windows 8 nebyly ochuzeny o klasickou plochu s klasickými ikonami. V rámci systému se plocha tváří jako aplikace, která samozřejmě může mít i svůj živý panel. Po „spuštění“ aplikace plocha se uživatel dostane do již známějšího prostředí, ne nepodobnému tomu z předchozích verzí systému. Popis tohoto prostředí a jeho novinek je nad rámec této práce, jelikož nesouvisí s novým uživatelským prostředím.

3.3.2 Dotyková Gesta

Dotyková gesta jsou nedílnou součástí nového systému. Díky nim se uživatel snadno pohybuje v celém systému, přepíná mezi aplikacemi a zobrazuje či schovává aplikační menu.

Následující část je zpracována podle části online dokumentace na Microsoft Dev Center – Gestures, 2013 (Microsoft, Gestures, 2013).

1. *Zmáčknout*
   Základní gesto, kterým lze otevřít, vybrat, nebo aktivovat cokoliv je potřeba. Velmi podobné kliknutí levým tlačítkem myši.
2. *Zmáčknout a držet*
   Tímto gestem lze například vyvolat nápovědu, nebo zobrazit skrytou nabídku. Velmi podobné kliknutí myši pravým tlačítkem.
3. *Roztáhnout nebo stáhnout dva prsty*
   Zvětší nebo zmenší například obrázky nebo map.
4. *Posunout*
   Posunutím prstu po obrazovce lze posunout obsah. Velmi podobné rolovacímu kolečku na myši.
5. *Posunout na jiné místo*
   Přesune libovolnou položku na jiné místo. Podobné funkci „chyť a pusť“.
6. *Posunout pro výběr*
   Přejetím prstem krátce přes nějakou položku ji lze například označit, nebo vyvolat pomocnou nabídku.
7. *Přejetí prstem z okraje obrazovky*
   Přejetí prstem z okraje obrazovky lze provést buď krátce, nebo dlouze přes celý displej.
   1. *Dlouhé přejetí z horního okraje*
      Ukončí právě běžící aplikaci.
   2. *Krátké přejetí z pravého okraje*
      Otevře nabídku „Postranní panel“, popsanou níže.
   3. *Krátké přejetí z levého okraje*
      Otevře poslední běžící aplikaci.
   4. *Krátké přejetí z levého okraje a hned zpět za okraj*
      Otevře list všech spuštěných aplikací.
   5. *Krátké přejetí z horního či dolního okraje*
      Otevře menu pro právě běžící aplikaci.

6. *Krátké přejetí z levého okraje a držení*
   Možnost rozdělit displej na části a mít tak zobrazeno více aplikací najednou.

Gesta myší a klávesové zkratky jsou nad rámec této práce.

3.3.3 Aplikace dotykového prostředí

V rámci aplikací pro dotykové rozhraní představil Microsoft velmi rozsáhlý popis a návod na designové vzory (Microsoft, Design patterns, 2013). Tyto vzory obsahují barvy, prvky, rozvržení prvků a animace. Vývojáři by se těchto vzorů měli držet a při vývoji postupovat tak, aby používal správné prvky pro správné věci.

Tyto designové vzory jsou velmi důležité, jelikož při jejich dodržování dostane výsledná aplikace vzhled a ovládání ne nepodobné ostatním aplikacím a celému systému. Uživatel má pak z celého systému pocit konzistence a při jeho ovládání by měl mít dobrý pocit. Aplikace samozřejmě může obsahovat vlastní grafické prvky, avšak základní rozvržení musí dodržovat designové vzory.

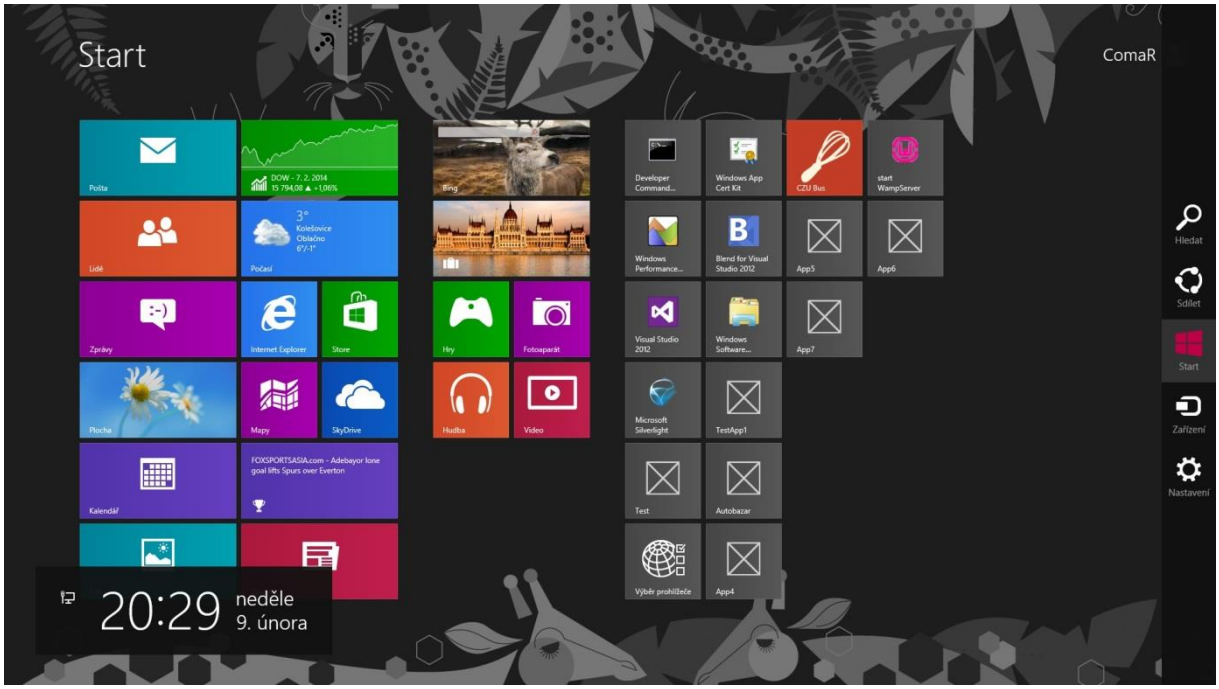
Následující část je zpracována podle části online dokumentace na Microsoft Dev Center – Navigation patterns, 2013 (Microsoft, Navigation patterns, 2013).

Aplikační lišta (App bar)

Aplikační lišta je hlavní příkazové rozhraní aplikace. Tento panel může být použit pro navigaci v aplikaci, příkazy a nástroje. Aplikační panel je schován a zobrazí se pouze po potažení prstu z dolní části obrazovky.

Aplikační panel může být umístěn nahoře, dole anebo na obou místech, přičemž oba panely mají jiný význam. Horní panel slouží převážně pro navigaci v aplikaci, změnu kategorie nebo otevření speciální stránky. Oproti tomu spodní panel, který slouží převážně pro příkazy, by měl obsahovat ikonky pro přidávání či odebírání obsahu, vybrání obsahu či zobrazení detailu položky.

Postranní panel (Charm bar)



Obrázek 3 - Windows 8 startovací obrazovka – ZDROJ: vlastní

Postranní panel je jedna z nejdůležitějších nabídek systému. Je dostupná kdykoliv a v jakékoliv aplikaci po přetažení z pravého okraje displeje. Současně s ní se v levé části zobrazí čas a datum. Postranní lišta obsahuje vždy 5 ikon, reprezentujících 5 funkcí.

Hledat – Otevře panel pro vyhledávání v celém systému.
Sdílet – Otevře panel se seznamem aplikací, se kterými je možné sdílet aktuální obsah.
Start – Vždy uživatele přesune na startovací obrazovku nebo otevře poslední aplikaci.
Zařízení – Otevře panel se seznamem zařízení, na které je možné odeslat právě zobrazovaný obsah.
Nastavení – Otevře panel s nastavením pro právě spuštěnou aplikaci, nebo nastavení systému.

Dialogy se zprávami

Toto je obdoba klasických dialogových oken v předchozích verzích systému. Vyžadují přímou interakci uživatele. Při zobrazení dialogové zprávy se pozadí zatmaví a zobrazí se dialogové okno. Tento prvek by měl vývojář použít pouze v případě, kdy vyžaduje zastavení konzumace obsahu uživatelem a jeho reakci na událost.

Překrytí (Flyout)

Překrytí jsou malá okna, zobrazující dočasné deaktivovatelné ovládací prvky, které nějak souvisí s obsahem. Tento prvek lze použít mimo jiné na potvrzení akce uživatelem, typicky například potvrzení smazání záznamu. Tento prvek je rozdílný od dialogové zprávy. Překrytí by měla aplikace zobrazit pouze po uživatelské interakci, oproti tomu dialogová zpráva se může zobrazit i bez přímé uživatelské interakce.

Upozornění (Toast)

Notifikace jsou malá okna, která se mohou zobrazit kdykoliv během běhu systému. Tyto upozornění jsou vhodná pro informování uživatele o nějaké skutečnosti v rámci aplikace. Pokud uživatel klikne na toto upozornění, spustí se příslušná aplikace s více informacemi.

Animace

Animace jsou pro Windows 8 velmi důležité. Dokazuje to i fakt, že všechny časově náročnější operace musí probíhat na pozadí, tak aby byl pro animace zajištěn dostatečný výpočetní výkon. Z kvalitní, správně navržené aplikace má uživatel pocit jakési „živosti“ a spolehlivosti. Animace navíc pomáhají pochopit obsah a jeho ovládání.

Drtivá většina animací je v aplikacích již předpřipravena.

4 Vývojové prostředí

4.1 Visual Studio 2013

Tato podkapitola je zpracována podle online dokumentace MSDN pro aplikaci Visual Studio (Microsoft, Visual Studio, 2014) a podle knihy Visual C# 2010 Krok za krokem (Sharp, 2012).

Microsoft pro vývoj na své platformě připravil vývojové studio, které je neustále vylepšováno již od první verze, představené v roce 1997. Visual Studio ve své nejnovější verzi 2013 nabízí různé programovací jazyky a spoustu doplňků usnadňující vývoj. Jakožto vývojové prostředí nabízí vše od editoru kódu, až po složité analytické algoritmy, pro analýzu náročnosti aplikace na systémové prostředky a na spotřebu energie. Vyvíjet pro dotykové rozhraní Windows 8 je možné již od verze 2012.

Visual Studiu je vždy nabízeno v několika verzích, přičemž jednotlivé stupně jsou odděleny podle velikosti vyvíjené aplikace, potažmo velikosti a rozsáhlosti vývojového týmu. V nejvyšších verzích je možné využít například Visual Studio Team Foundation Server, které je určeno především pro správu kódu a komplexní řízení projektu.

4.1.1 Edice Visual Studia 2013

Professional	Lze označit jako základní verzi s pokročilejšími nástroji
Test Professional	Speciální edice, určená především profesionálním testerům aplikací
Premium	Vhodné pro tvorbu komplexnějších aplikací
Ultimate	Tuto verzi využijí především velmi pokročilí vývojáři, softwaroví architekti a vedoucí projektů
Team Foundation Server	Nadstavba pro precizní správu a řízení kódu vývoje

Tabulka 1 - Verze vývojového prostředí Visual Studio 2013

4.1.2 Psaní kódu

Psaní kódu je ta nejpodstatnější část při vývoji aplikace. Visual Studio tuto práci značně usnadňuje nejen barevným rozlišením různých prvků kódu. Na přehlednosti kódu přidává také poloautomatické formátování, které mnohdy za programátora naformátuje daný kus kódu zcela automaticky.

Daleko zásadnějším vylepšením Visual Studia je však automatická kontrola syntaktických a části sémantických chyb. Kontrola syntaktických chyb je součástí mnoha vývojových prostředí, avšak kontrola chyb sémantických je mnohdy velmi složitá a povětšinou je nechávána na vývojáři. Visual Studio dokáže kód kontrolovat v reálném čase a vývojáře upozorní nejen při chybějící závorce, ale dokáže ho upozornit také na volání neexistující funkce, volání funkce se špatným počtem argumentů nebo například existenci nedosažitelného kódu, což je kód, který za žádných okolností nemůže být proveden.

4.1.3 Odstraňování chyb (Debugování)

Debugování, tedy detekce a odstraňování chyb, je další z velmi kvalitně provedených funkcí Visual Studia. Ve Visual Studiu lze pro tuto činnost zvolit několik postupů. Patrně nejpoužívanější z nich je kliknutí před chtěný řádek, čímž se přidá takzvaný „breakpoint“. „Breakpoint“ je bod, k němuž když se při běhu aplikace dostane, tak se pozastaví.

Tuto možnost nabízí také spousta konkurenčních vývojových prostředí, avšak Visual Studio přidává zajímavý bonus. Při takovémto pozastavené aplikace je možné v rámci jedné funkce přesunout bod pozastavení o libovolný počet řádků a nechat tak kupříkladu proběhnout část kódu znovu, nebo naopak část manuálně přeskočit. Po najetí a přidržení kurzoru myši nad libovolnou proměnnou se zobrazí nabídka s možností náhledu na objekt uložený v dané proměnné a případně lze některé hodnoty ručně upravit.

4.1.4 IntelliSense

Součástí Visual Studia a současně jednou z jeho velkých předností je pomůcka zvaná IntelliSense. Tato pomůcka pracuje ruku v ruce s vývojářem a pomáhá doplňovat kód za něj. Mimo doplňování rozepsaného kódu dokáže také napovídat další možnosti.

IntelliSense pracuje vždy v kontextu s daným kódem. Nabízí tedy vždy pouze relevantní možnosti, které náleží danému jmennému prostoru a možnostem kódu. Tím pomáhá při vývoji zbavit se zbytečných chyb a zvýšit produktivitu práce. Pokud se IntelliSense

nezobrazí při psaní kódu samo, lze tuto funkci vyvolat pomocí klávesové zkratky CTRL+Mezerník.

4.1.5 Microsoft Blend pro Visual Studio 2013

Tato podkapitola je zpracována na základě online dokumentace pro Microsoft Blend (Microsoft, Design Windows Store apps using Blend, 2014).

Ač je Visual Studio nejrobustnější, není jediným pomocníkem při vývoji aplikací pro Windows. Blend je vývojové studio pro návrh designu aplikace, podobně jako tomu bylo u předchozích verzí Visual Studia pro desktopové aplikace. Blend v sobě obsahuje několik nástrojů a funkcí pro snadné vytváření pokročilého designu aplikace. Svým způsobem se jedná o jakýsi WYSIWYG (WhatYouSeeIsWhatYouGet) editor. Zjednodušeně tedy lze říci, že to co uživatel vidí, to také dostane. Blend na základě uživatelských příkazů tvoří a skládá designové prvky.

Blend lze použít při vývoji na všech Microsoftem dostupných platformách pro vývoj. Konkrétněji tedy lze použít jak kombinaci CSS ve verzi 3 a HTML ve verzi 5, tak lze využít designovací jazyk, který se dříve používal pouze v Microsoft Silverlight aplikacích, a sice rozšířený jazyk xml – xaml. Současně Blend nabízí propojení designu s daty, jako je popsáno dále v této práci.

5 Vývoj

5.1 Ukázková aplikace

Aby mohly být některé pojmy vysvětleny co nejsrozumitelněji, přílohou této práce je aplikace pro dotykové rozhraní. Jedná se o jednoduchou aplikaci, na které bude demonstrováno několik zásadních prvků vývoje.

Aplikace má název CzuBus. Čte data z připraveného XML souboru a na tomto základě dokáže zobrazit 5 nejbližších odjezdů MHD autobusů ze zastávky Zemědělská univerzita do zastávky Dejvická a naopak.

Aplikace je napsána v jazyce JavaScript, avšak jádro je napsáno v jazyce C#. Design aplikace je popsán jazykem HTML5 a CSS3. Následující text bude obsahovat části zdrojového kódu aplikace jakožto praktickou demonstraci možností vývoje pro Windows 8.

5.1.1 Zdrojové soubory

Na zdrojových souborech ukázkové aplikace bude mimo jiné předvedena základní fyzická struktura aplikací pro dotykové rozhraní na platformě JavaScript a HTML. Seznam vychází z hierarchie souborů a složek jako na obrázku *Obrázek 5* - Struktura ukázkové aplikace CzuBus a postupuje shora dolů.

References

V této položce je uložen seznam všech odkazovaných knihoven v rámci ukázkové aplikace. Předpřipravená a zároveň nejdůležitější je zde Windows Library for JavaScript 1.0., která do aplikací vkládá velmi důležitý prvek WinJS, podrobněji popsaný v podkapitole 5.3.1.

Css

Složka, obsahující základní a předpřipravené kaskádové styly pro aplikaci. Její součástí je ve výchozím stavu soubor default.css, který sdružuje základní rozvržení a zobrazení.

Images

Obsahují základní obrázky k aplikaci. Ve výchozím stavu obsahuje obrázek pro načítání, obrázek pro obchod aplikací a dvě velikosti standardního loga, které mohou být použity pro zobrazení na startovací obrazovce v rámci živého panelu Live Tile.

Js

V této složce lze nalézt základní soubory s kódem aplikace. Nachází se zde především soubor default.js, kde lze nalézt hlavní vstupní bod při startu aplikace. Většinou se zde také nastavují akce při změně stavu aplikace, jako je přechod do režimu spánku, kontrola předchozího stavu aplikace a podobné.

Dále je zde ve výchozím nastavení také soubor navigator.js, který se stará o korektní práci při navigování mezi stránkami aplikace.

Tyto soubory, ač jsou ve výchozí složce, musí být v každé stránce odkázány, aby bylo možné je využít.

Pages

Jak již název napovídá, jedná se o seznam stránek. Konvence je taková, že každá stránka by měla mít vlastní složku, obsahující soubory *.html, *.css a *.js (JavaScript).

Buses.xml

Toto není defaultní soubor Windows 8 aplikace. Jedná se o soubor zdrojových dat pouze pro aplikaci CzuBus.

CzuBus_TemporaryKey.pfx

Soubor s certifikátem, umožňujícím snadné spuštění aplikace, která neprošla certifikačním procesem. Je vázán na vývojáře.

Default.html

Hlavní soubor aplikace. Osahuje odkazy na css a js soubory a zároveň může určovat základní rozložení stránky. K tomuto souboru je přidružen soubor default.js ze složky js.

Package.appxmanifest

Základní nastavení aplikace. Obsahuje nejen jméno a verzi aplikace, ale lze zde také nastavit, jaká hardwarová zařízení aplikace vyžaduje. To je důležité proto, aby již v obchodě s aplikacemi uživatel viděl, k jakým částem svého zařízení vyžaduje tato

aplikace přístup. Typicky se jedná například o fotoaparát nebo mikrofon. Uživatel musí použití těchto periferií explicitně potvrdit.

Jako další zde v tomto souboru nastavit aplikací podporované orientace obrazovky, logo aplikace a v neposlední řadě také podporované jazykové mutace.

5.2 Vývoj pro Windows 8

Vývoj je pro každý operační systém velmi důležitá část. V dnešní době se kvalitní operační systém neobejde bez kvalitních aplikací. I to je důvod, proč jsou stále rozšířenější obchody s aplikacemi. Tato skutečnost má dvě výhody.

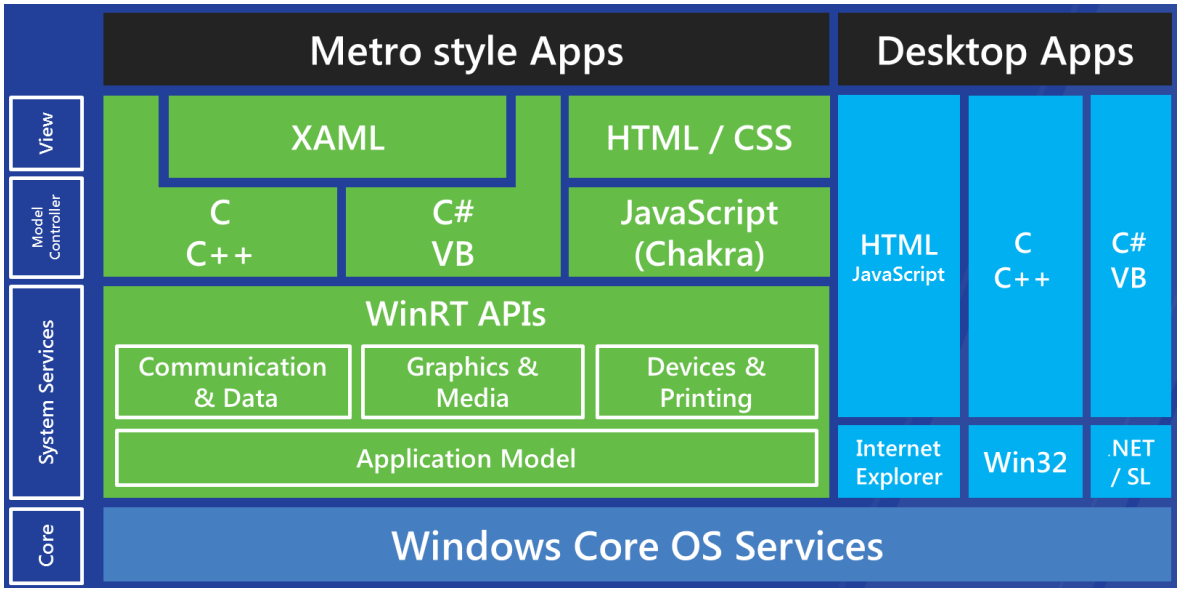
Jako první lze vyzdvihnout to, že je vývojář ušetřen od složitého propagování své aplikace pomocí různých kanálů. Může se tedy plně soustředit na vývoj. Avšak i přes to zde existuje několik věcí, které ovlivní hodnotu aplikace. Mezi nejzásadnější patří například ikona aplikace, tedy jakýsi úvodní obrázek aplikace v obchodu s aplikacemi. Pokud není tomuto malému obrázku věnována dostatečná péče, aplikace se hůře dostává do podvědomí více uživatelů.

Druhá výhoda se týká pohledu uživatele. Pokud má uživatel zájem o některou aplikaci, jednoduše navštíví obchod aplikací a vybere si přesně tu, která mu vyhovuje. Do Windows se poté aplikace nainstaluje jednoduchým kliknutím a případným zaplacením aplikace uživatelem. S tímto souvisí i jakási jedna další výhoda, a sice fakt, že veškeré aplikace pro dotykové prostředí Windows 8 musí projít certifikací. Certifikace je proces, kterým musí projít každá aplikace v obchodě s aplikacemi. Více o certifikaci bude řečeno dále v práci.

„Metro apps are an important addition to Microsoft Windows 8, providing the cornerstone for a single, consistent programming and interaction model across desktops, tablets, and smartphones.“ (Freeman, Metro Revelaed - Building Windows 8 Apps with HTML5 and Javascript, 2012)

Microsoft se snaží nalákat vývojáře jak nové, tak již zkušené vývojáře z jiných platforem. Je si totiž vědom, že bez vývojářů nejsou aplikace. I proto připravil velmi rozsáhlou dokumentaci Windows 8, pořádá výukové semináře ve větších městech po celé České Republice a celý proces vývoje se snažil výrazně zjednodušit. A nejen proces vývoje, ale také samotný nástroj k vývoji.

Co je oproti předchozím verzím systému Windows velmi odlišné, je výchozí rozhraní pro všechny aplikace pro dotykové rozhraní. Nazývá se **Windows Runtime** a je detailně popsáno v online dokumentaci na Microsoft Dev Center (Microsoft, Windows Runtime, 2014). Toto rozhraní, vytvořené v jazyce C++, podporuje aplikace napsané v jazyce C++, C#, Visual Basic, JavaScript a TypeScript. Všechny tyto jazyky se totiž vždy převádějí do nativního jazyka, tedy C++. Microsoft tento proces nazývá jako projekci. Windows Runtime lze spustit na procesorech s architekturou ARM, x86 a x64. Je navrženo tak, aby bylo možné pro vývoj aplikací zvolit již od začátku různé kombinace různých programovacích či skriptovacích jazyků.



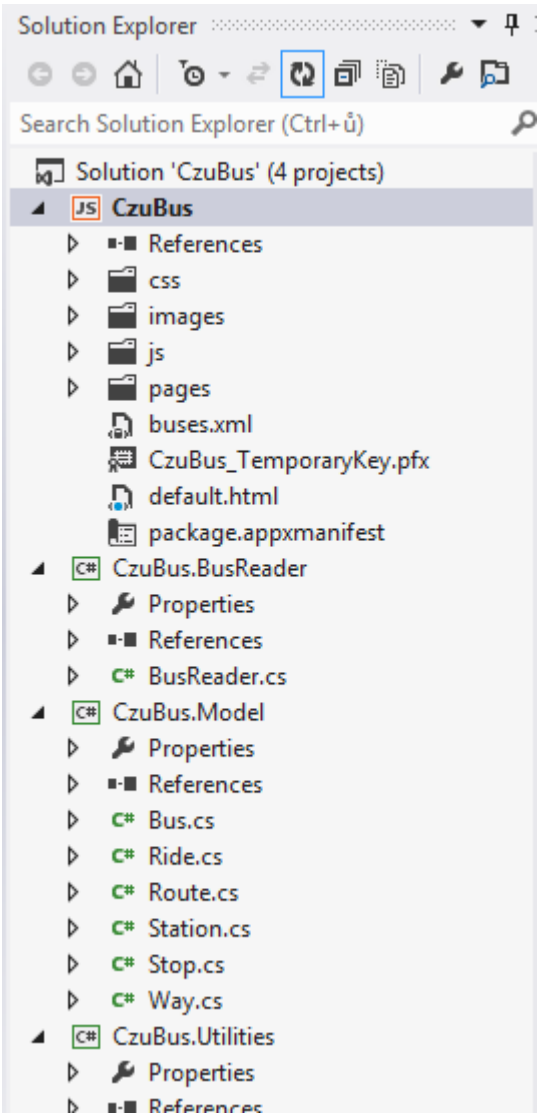
Obrázek 4 - Vývojové platformy - ZDROJ: http://programmers.stackexchange.com/questions/155521/what-is-the-difference-between-windows-8-winrt-and-windows-rt

5.2.1 Projekce jazyka

Projekcí jazyka, jak již bylo uvedeno, se rozumí konverze různých podporovaných jazyků do nativního jazyka Windows Runtime. Projekce jazyka se používá v celém systému napříč vývojovými platformami. Je také předvedena v ukázkové aplikaci, přiložené k tomuto projektu. Zde je toto využito ve smyslu použití jiné platformy pro jádro aplikace a jiné platformy pro zobrazení dat. Pro zpracování dat z XML souboru a také pro vytvoření datového modelu aplikace je použita platforma C#, pro ostatní je použit JavaScript.

Po otevření projektu lze vidět, že obsahuje několik menších projektů, jako například projekt CzuBus.BusReader. Ten má na starosti právě zpracování XML souboru a vytvoření

instancí příslušných objektů. Tento kus kódu je v aplikaci CzuBus volán z jazyka JavaScript, mimo jiné i ze souboru default.js.



Obrázek 5 - Struktura ukázkové aplikace CzuBus – ZDROJ: vlastní

Projekce z jazyka C#

Tato projekce byla využita i v ukázkovém projektu, kde byla použita nejen z efektivních důvodů, ale především z důvodu přehlednosti. Na obrázku Obrázek 5 - Struktura ukázkové aplikace CzuBuslze vidět, že první projekt s názvem CzuBus má u sebe červenou ikonku s iniciály „JS“, což značí, že se jedná o projekt, napsaný v jazyce JavaScript. Projekt CzuBus je samotná aplikace, obsahující veškerý grafický design a základní ovládací prvky.

Pod tímto projektem je ale několik projektů, majících u sebe malou ikonku s iniciály „C#“, což značí použití jazyka C#. Konkrétně se jedná o knihovny, nebo také soubory se zdrojovými kódy. Výstupem projektu typu knihovna je nespustitelný soubor, obsahující nějakou funkcionalitu. Tento soubor lze však nastavit jako podřízený soubor projektu CzuBus a využít tak veškerou jeho funkcionalitu. Například projekt CzuBus.Model obsahuje veškeré objekty, které jsou v aplikaci CzuBus použity.

Nejdůležitější ovšem je, že se nejedná o klasické soubory knihoven s příponou „.dll“. Jedná se o jiný typ knihoven, takzvaný **Windows Runtime Component** („.winmd“)**.** Tento typ knihovny je speciální právě v tom, že ho lze projektovat do nativního kódu Windows Runtime, což s klasickými knihovnami udělat nelze. Právě kvůli tomu tato knihovna postrádá různou funkcionalitu a má svá omezení.

Jako nejzásadnější z nich lze označit to, že veškeré třídy v projektu musí být „sealed“. Tedy z těchto tříd již nelze odvozovat (dědit) jiné třídy. Lze to označit jako daň za možnost multiplatformního vývoje.

```
public static IList<Route> GetBuses()
        {
            List<Route> routes = new List<Route>();

            // Připravit cesty
            routes.Add(new Route() { From = Station[0], To = Station[8] });
            routes.Add(new Route() { From = Station[8], To = Station[0] });

            foreach (Route route in routes)
            {
                foreach (Bus singleBus in Bus)
                {
                    singleBus.GetDepartures(route);
                }

                route.Departures = route.Departures.OrderBy(o =>
o.Departure).ToList();
                route.Departures = route.Departures.Take(5).ToList();
            }

            // Nyní máme všechny odjezdy z potřebné stanice, vrátíme prvních pět
            return routes.Take(5).ToList();
        }
```

Ukázka kódu 1 - Metoda pro získání odjezdů autobusů napsaná v jazyce C#

```
function updateDepartures(force)
    {
        routes = CzuBus.BusReader.BusManager.getBuses();

        route0 = routes[0];
        route1 = routes[1];

        dataList0 = new WinJS.Binding.List(getDeparturesViewData(route0));
        dataList1 = new WinJS.Binding.List(getDeparturesViewData(route1));

        if (force) {
            var list0 = document.getElementById("route0_ListView").winControl;
            list0.itemDataSource = dataList0.dataSource;

            var list1 = document.getElementById("route1_ListView").winControl;
            list1.itemDataSource = dataList1.dataSource;
        }

        canUpdate = true;
    }
```

Ukázka kódu 2 - Volání metody getBuses() z jazyka JavaScript

5.3 Vývojové platformy

V této kapitole bylo čerpáno především z online dokumentace MSDN pro vývojové platformy (Microsoft, App architecture, 2013).

5.3.1 JavaScript a HTML

JavaScript a HTML5 již nejsou jen webové technologie. Tyto technologie lze použít pro vývoj aplikací pro dotykové rozhraní u Windows 8. Jak je již popsáno v kapitole Projekce jazyka, Windows 8 dovoluje projekci jazyka JavaScript do nativního kódu.

Samotný JavaScript však standardně nemá přístup k systémovým prostředkům a hardwaru přístroje. Microsoft proto připravil speciální knihovnu WinJS, díky které získá vývojář přístup ke všemu, k čemu má přístup nativní kód. Tato knihovna je při vytvoření aplikace již defaultně připravena. Podrobněji bude popsána níže.

Mimo WinJS je samozřejmě možné přidat do aplikace jakýkoliv již existující JavaScriptový framework (Freeman, Metro Revelaed - Building Windows 8 Apps with HTML5 and Javascript, 2012). Typicky se jedná například o jQuery nebo Prototype. Vývojář má v tomto ohledu naprostou volnost. JavaScriptový framework lze přidat naprosto stejně, jako se přidává do webových projektů – tedy vložením příslušného souboru s příponou js do projektu a nastavení jeho odkazu v hlavičce patřičné HTML stránky.

Jazyk HTML lze využít ve Windows 8 ve verzi 5, která podporuje pokročile vykreslování vektorové grafiky, vkládání videa a samozřejmě CSS ve verzi 3 pro popis stylů. HTML tak nahrazuje funkci jazyka XAML pro rozložení stránky. Microsoft tímto otevírá dveře pro vývojáře z řad webových dodavatelů. Ti mohou využít své znalosti z oblasti webových technologií a uplatnit je při vývoji aplikací pro Windows 8.

Kombinace JavaScriptu a HTML je tedy vhodná pro začátečníky, případně webové vývojáře, kteří chtějí vytvořit méně náročné až středně náročné aplikace a hry. Výhodou tohoto řešení je možnost přenositelnosti již existujících webových řešení do Windows 8 aplikace.

WinJS

Tato podkapitola byla zpracována na základě nastudování online ukázkové aplikace dostupné na http://msdn.microsoft.com/en-us/library/windows/apps/hh986964.aspx a na základě online dokumentace (Microsoft, App layout, 2013).

Microsoft přidal k Windows 8 JavaScript aplikacím také užitečnou knihovnu Windows Library for JavaScript – WinJS. Tato knihovna je něco jiného než Windows Runtime. Jedná se o čistě JavaScriptovou knihovnu, která je automaticky přidána do každého nového projektu. Tato knihovna je přístupná pouze při vývoji v jazyce JavaScript a sice pod jménem WinJS.

WinJS lze rozdělit na tyto základní části

- Implementace CommonJS Promises/A
- Pokročilou správu uživatelského rozhraní
- Nástroje pro správu DOM elementů
- Nástroje pro navigování v aplikaci

Vývojář není nucen tuto knihovnu použít a může použít libovolnou jinou knihovnu. V praxi ovšem WinJS nabízí spoustu užitečných funkcí, přičemž část z nich není nahraditelná jinými knihovnami.

Promises

Jak již bylo uvedeno výše, Windows 8 aplikace jsou velmi silně asynchronní. Veškeré časově náročnější operace jsou prováděny asynchronně. Nejzákladnější možností jak lze v JavaScriptu implementovat asynchronní zpracování, je volání takzvaných callback

metod. Callback metoda je anonymní funkce, která se předává volané metodě jako parametr. Volaná metoda musí přijímat odkaz na funkci jako jeden ze svých argumentů.

```
function zobrazHlasku(v){ /* zobrazuji hlasku */ };

function ulozData(soubor, zobrazHlasku) {
    var vysledek = ulozSouborNaDisk(soubor);
    zobrazHlasku(vysledek);
}

// Volání
ulozData(soubor, function (vysledek) {
    zobrazHlasku(vysledek);
})
```

Ukázka kódu 3 - Uložení dat a zavolání callback(zobrazHlasku) metody (vlastní zdroj - není součástí ukázkové aplikace)

V případě ukázky kódu *Ukázka kódu 3* - Uložení dat a zavolání callback(zobrazHlasku) metody (se metodě jako první argument předává soubor, který má být uložen a jako druhý parametr se předává ukazatel na libovolnou metodu, která bude zavolána po uložení souboru. Tento postup je však velice nepřehledný v případě volání několika vnořených callback metod. Mimo to je callback metoda předávána přímo a ihned při volání jako argument, což může být někdy také problém.

Efektivním a funkčním řešením jsou **Promises**. Promise je speciální objekt, který je vracen funkcí ihned. Nejdůležitější metodou objektu Promise je metoda then(), která jako první argument zpracovává odkaz na funkci, tedy callback metodu. Na první pohled se toto rozdělení může zdát zbytečné, avšak má své opodstatnění.

```
function zobrazHlasku(v) { /* zobrazuji hlasku */ };

function ulozData(soubor)
{
    var promise = new Promise();

    ulozSouborNaDiskAsynchrone(function (soubor) {
        // Toto bude zavoláno po uložení souboru
        promise.complete();
    });

    return promise;
}

// Volání
ulozData().then(zobrazHlasku);
```

Ukázka kódu 4 - Uložení dat pomocí Promise objektu (vlastní zdroj - není součástí ukázkové aplikace)

První podstatný důvod je přehlednost. Volání metody ulozData je přehledné a kód je sebe-popisující. Druhý důvod je možnost připojovat za sebe další a další metody then(). Metoda then() totiž vrací opět objekt typu Promise. Obecně lze říci, že použitím objektu Promise se rozdělí zavolání obslužné metody od zavolání callback metody, čímž se velice usnadní zpracování asynchronního kódu a ještě se zlepší přehlednost kódu.

Jak si lze povšimnout v ukázce kódu *Ukázka kódu 4* - Uložení dat pomocí Promise objektu (, na objektu typu Promise je volána metoda complete(). Po zavolání této metody se spustí callback metody předané jako argument funkce then().

Hlavní výhoda Promises je tedy fakt, že funkce vrací objekt typu Promise, na který je možné navázat libovolný počet callback metod, bez nutnosti předávat tyto metody jako argumenty přímo do volané funkce. Volaná funkce vůbec nemusí vědět které funkce má zavolat po dokončení. Jediné co musí udělat je zavolat funkci complete() a o ostatní se postará objekt Promise.

Při vývoji pro Windows 8 v jazyce JavaScript je znalost objektu Promise velice důležitá. Vývojář se s touto funkcí setkává téměř všude. Při jakémkoli pokusu provést složitější operaci se musí použít objekt Promise.

5.3.2 C#, Visual Basic, C++ a XAML

Tato kombinace je vhodná především pro středně náročné aplikace a hry, jelikož škálovatelnost jazyků C++, C# nebo Visual Basic je na jiné úrovni než je tomu u jazyka

JavaScript. V kombinaci s jazykem XAML je tato platforma většinou volena vývojáři, kteří přecházejí z platformy Silverlight, kde byl hojně využíván jazyk C# nebo Visual Basic a XAML.

Výkon aplikací napsaných v těchto jazycích by měl být srovnatelný s jazykem JavaScript. Všechny platformy se tak či tak konvertují do nativního jazyka rozhraní Windows Runtime. Jedná se tedy spíše o osobní preference vývojáře, jakou platformu zvolí.

Tato platforma je vhodná pro pokročilejší vývojáře, znalé některého ze zmíněných programovacích jazyků a mající zkušenosti s jazykem XAML. Hlavní přednost tohoto řešení je pokročilejší jazyka, který oproti JavaScriptu může být napevno typován a ve větších projektech se jeví jako přehlednější.

Stejně jako JavaScript, i jazyky jako C# podporují asynchronní zpracování procesů. Pro jejich implementaci jazyk C# nabízí různé nástroje již velmi dlouhou dobu. Novinkou je pouze možnost použití takzvaného async/await zápisu.

```
        private void VstupniBodProgramu()
        {
            Task<int> vysledek = UlozData(new object());
        }

        private async Task<int> UlozData(object data)
        {
            Task<int> ukladani = UlozDataNaDisk(data);

            // Zde je možno provést několik dalších operací, zatímco ukládání je
spuštěno na pozadí

            //Nyní již je vše hotovo a je potřeba vrátit výsledek ukládání
            return await ukladani;
        }

        private Task<int> UlozDataNaDisk(object data)
        {
            // Ukládám
            return Task.FromResult<int>(0);
               }
```

Ukázka kódu 5 - Asynchronní uložení souboru v jazyce C# (vlastní zdroj - není součástí ukázkové aplikace)

V tomto případě se před metodu přidá klíčové slovo async, které určí, že metoda se má spustit asynchronně. Tělo metody poté obsahuje klíčové slovo await, které říká, že na zpracování časové náročné, asynchronní metody se musí vyčkat. Metoda poté nevrací

přímo návratovou hodnotu, ale vrací objekt Task jako je tomu v ukázce *Ukázka kódu 5 - Asynchronní uložení souboru v jazyce C# (vlastní zdroj - není součástí ukázkové aplikace).*

Je důležité, aby byl výsledek metody UlozData uložen do proměnné. Pokud by tomu tak nebylo a v asynchronní metodě došlo k chybě, byla by tato chyba zahozena. Při uložení výsledku do proměnné má však běhové prostředí k této chybě přístup.

5.3.3 C++ a DirectX

Jazyk C++ je nativní jazyk Windows Runtime rozhraní. I přes to ale nabízí srovnatelnou rychlost jako například jazyk C#. To je dáno především omezenými možnostmi Windows Runtime. Avšak ve spojení s DirectX knihovnou se tato platforma jeví jako nejvhodnější řešení pro vývoj pokročilých her. Jazyk C++ dokáže s knihovnou DirectX komunikovat lépe než jazyk C#. C++ v kombinaci s DirectX je ovšem vhodné už pro velmi pokročilé vývojáře, případně tým vývojářů.

5.3.4 Ostatní

Mimo výše uvedené specifické platformy je také možné platformy kombinovat. Podobně jako je to také u přiložené ukázkové aplikace k této práci, viz 5.2.1.

5.4 Windows Store Aplikace

Zásadní skutečnost, která jistě stojí za povšimnutí je to, že veškeré aplikace pro dotykové rozhraní jsou spuštěny v takzvaném „sandboxu“. To znamená, že každá aplikace má své vlastní systémové zdroje a nemá přístup mimo svůj „sandbox“. Jinak řečeno, žádná aplikace neví o žádné jiné a zároveň v základu nemá přístup k některým kritickým systémovým prostředkům a hardwaru (Rozenblit, 2013). Ze svého „sandboxu“ může aplikace vystoupit pouze v případě, že to uživatel dovolí. Toto je typicky prováděno formou dotazů na uživatele, jako například „Může tato aplikace použít váš fotoaparát?“. Právě díky tomu se minimalizuje riziko nainstalování škodlivého softwaru.

Aplikace jako taková nemá všeobecně velmi známou příponu „exe“, ale končí příponou „appx“, což je kombinace formátu ZIP a jazyka XML. Celkové běhové prostředí je velmi silně přizpůsobeno pro běh aplikací asynchronně. Veškerá práce, která systému zabírá více času je ve Windows Runtime přepracována do formy, ve které ji lze spustit v jiném procesorovém vlákně. Výsledkem je, že všechny náročné operace jsou prováděny na

pozadí a uživatel má vždy přístupné a reagují komponenty aplikace. Díky tomu nemá uživatel u správně napsaných aplikací pocit „zamrznutí“ systému.

Zde je pro ukázku uvedena část kódu z přiložené aplikace. Jedná se o načítání souboru z disku, které je obecně náročnější na čas a ve Windows 8 tak musí být provedeno asynchronně, tedy na pozadí.

```
function InitializeBuses()
    {
        // Adresa xml souboru s autobusy
        var url = new Windows.Foundation.Uri("ms-appx:///buses.xml");

        // Zde otevřeme tento xml soubor asynchronně

Windows.Storage.StorageFile.getFileFromApplicationUriAsync(url).then(function
(file) {
            Windows.Storage.FileIO.readTextAsync(file).then(function (text) {

                // Předáme text xml souboru do metody pro zpracování
                CzuBus.BusReader.BusManager.fromXmlDocument(text);

            });
        });
    }
```

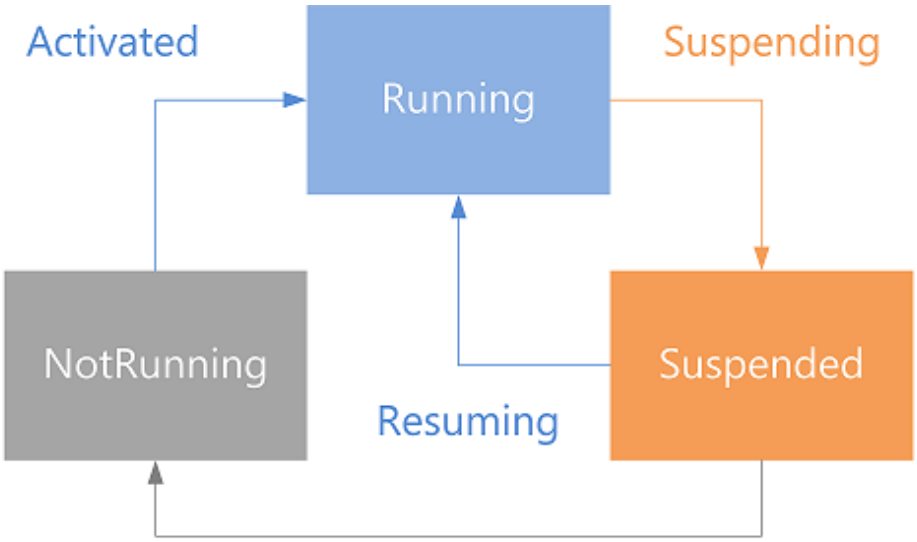
Ukázka kódu 6 - Načítání dat z XML

Toto je část z ukázkové aplikace, která se stará o načtení dat z přiloženého souboru formátu XML. Jak zde v této ukázce vidět, i zdánlivě jednoduchá práce jako načtení souboru musí být ve Windows 8 provedena asynchronně, tedy v jiném procesorovém vlákně než ostatní část aplikace. Toto je provedeno přesněji částí „Windows.Storage.FileIO.readTextAsync(file).then“. Právě tato metoda má na starosti ono přečtení souboru. Zavolání metody „then“ na konci má také svůj specifický význam. Jak již název napovídá, do této metody lze předat delegáta, tedy odkaz na metodu, která bude zavolána po načtení souboru. Jako parametr se funkci předává textový obsah onoho souboru. Více o metodě then a její implementaci lze najít v sekci 5.3.1.

5.4.1 Životní cyklus aplikace

Tato podkapitola je zpracována podle online vývojářské dokumentace k Windows 8 (Microsoft, Application lifecycle, 2013).

Životním cyklem aplikace lze nazvat vše co se s aplikací děje od momentu sestavení až po odstranění aplikace ze systému. Oproti předchozí verzi systému, Windows 8 mají zcela odlišný postup při přepnutí aplikace do pozadí. Pro správnou funkci aplikace je potřeba korektně pracovat s takzvanými stavy aplikace.



Obrázek 6 - Stavy aplikace – ZDROJ: http://msdn.microsoft.com/en-us/library/windows/apps/hh464925.aspx

Pokud v dřívějších verzích systému byla aplikace přesunuta na pozadí, stále mohla pracovat na svých výpočtech. Po přesunutí do popředí tak stačilo pouze znovu vykreslit uživatelské rozhraní. Ve Windows 8 je však tento proces složitější.

Aplikace spuštěné na pozadí totiž nejsou tak docela spuštěné, ale jsou „uspané“. Pokud je ve Windows 8 aplikace přesunuta na pozadí – například otevřením jiné aplikace, nebo otevřením úvodní obrazovky – má aplikace jen několik málo sekund na uložení aktuálního stavu, jako například název právě zobrazované stránky. Toto ukládání má svůj důvod, protože po uplynutí krátkého časového intervalu je aplikace přepnuta do režimu Suspended. V tomto režimu je držena, dokud není znovu otevřena, nebo zcela ukončena například z důvodu nedostatku systémových prostředků.

Některé aplikace vyžadují běh na pozadí. Typicky například přehrávače hudby nebo stahovače souborů. To je však nad rámec této práce.

Start aplikace (Launch)

Start aplikace je proces, při kterém přechází ze stavu NotRunning do stavu Running. Tento stav nastane, pokud aplikace ještě nebyla spuštěna, byla nečekaně ukončena anebo uspána, avšak z důvodů úspory paměti byla terminována (ukončena). Při takovémto startu aplikace se zobrazí při načítání její logo.

Během načítání aplikace musí nastavit základní uživatelské rozhraní a základní ovládací prvky. Ostatní, časově náročnější věci, jako například síťová komunikace, musí být provedeny až po nastartování aplikace.

Aktivace aplikace (Activation)

Aktivace aplikace může nastat z mnoha různých důvodů. Mimo jiné mezi ně patří také start aplikace, nebo obnovení aplikace z pozadí. Pro obsloužení této události je potřeba přihlášení k odběru události **Activated.** V ukázkové aplikaci je toto přímo v souboru default.js následovně.

```
    var app = WinJS.Application;
    var activation = Windows.ApplicationModel.Activation;
    var nav = WinJS.Navigation;

    app.addEventListener("activated", function (args) {
        if (args.detail.kind === activation.ActivationKind.launch) {
            if (args.detail.previousExecutionState !==
activation.ApplicationExecutionState.terminated) {

                InitializeBuses();
            } else {
                // TODO: This application has been reactivated from suspension.
                // Restore application state here.
            }

            if (app.sessionState.history) {
                nav.history = app.sessionState.history;
            }
            args.setPromise(WinJS.UI.processAll().then(function () {
                if (nav.location) {
                    nav.history.current.initialPlaceholder = true;
                    return nav.navigate(nav.location, nav.state);
                } else {
                    return nav.navigate(Application.navigator.home);
                }
            }));
        }
    });
```

Ukázka kódu 7 - Spuštění aplikace a událost Activated

Samotné přihlášení odběru události je zajištěno pomocí addEventListener("activated", function (args) { ... }), kde do těla funkce se vkládá kód, který se má spustit, pokud tato událost nastane.

Po aktivaci si aplikace musí zkontrolovat, z jakého stavu byla aktivována. To je ukázáno v ukázce kódu 3, konkrétně část args.detail.previousExecutionState. Tato proměnná uchovává údaje o předchozím stavu aplikace. Pokud je stav nastaven na **Terminated**, znamená to, že aplikace byla nejdříve uvedena do režimu **Suspended** a následně byla ukončena. V tomto případě je potřeba obnovit data, jako například poslední otevřená stránka, počet zaškrtnutých položek a podobně.

Zde jsou všechny možné předchozí stavy aplikace

Důvod	**Hodnota *previousExecutionState***	**Co je potřeba udělat**
Aplikace ukončena systémem, například z důvodu nedostatku paměti	Terminated	Načíst předchozí stav aplikace
Aplikace byla ukončena uživatelem	ClosedByUser	Načísti nová data
Nečekaně ukončená aplikace, nebo aplikace ještě nebyla spuštěna od posledního načtení operačního systému	NotRunning	Načíst nová data
Aplikace je „probuzena“ a nebyla terminována	Suspended	Nechat data tak jak jsou
Aplikace byla znovu aktivována během několika sekund a nestihla tak ani přejít do režimu Suspended	Running	Nechat data tak jak jsou

Tabulka 2 - Stavy ukončení Windows 8 aplikace

Uspání aplikace (Suspend)

Aplikace přechází do tohoto režimu v případě, že je odsunuta na pozadí. V kódu je pak možné přihlásit se k odběru události „Suspending“, která je zavolána těsně před uspáním aplikace. Právě tato událost se používá pro uložení dat, která později mohou být obnovena, aby uživatel nepoznal, že aplikace musela být ukončena. Uložení dat by nemělo trvat déle než jednu sekundu. Pokud aplikace neodpoví do pěti sekund, Windows ji rovnou terminuje.

Windows 8 se snaží udržet v paměti co nejvíce uspaných aplikací, aby mezi nimi mohl uživatel rychle a spolehlivě přepínat. Je-li takovýchto aplikací již mnoho, Windows některé aplikace terminuje, tedy zcela ukončí. Ve stavu suspended aplikace nepřijímá žádnou síťovou komunikaci.

Je důležité vědět, že aplikace při přechodu ze stavu Suspended do Terminated již nedostane žádný prostor pro uložení rozpracované práce nebo dat. Vývojář by se měl při uspávání aplikace zachovat jako by aplikace byla terminována.

Viditelnost aplikace (Visibility)

Při přepnutí aplikace 1 do aplikace 2 zůstane aplikace 1 stále ve stavu Running po dobu až pěti sekund. Pokud do této doby neodpoví, systém ji považuje za neaktivní a terminuje ji.

Pokud je při vývoji potřeba nějaké funkcionality při změně stavu viditelnosti aplikace, lze to pomocí přihlášení k odběru událostí **VisibilityChanged**.

Pokračování aplikace (Resume)

Do tohoto stavu se aplikace dostane, pokud je vrácena ze stavu Suspended. Její stav se změní na Running a může tak pokračovat v běhu tak kde přestala při přechodu do Suspended. Nedochází zde ke ztrátě žádných informací. Nicméně je možné, že aplikace byla ve stavu Suspended několik hodin, případně i dní. Proto je někdy potřeba aktualizovat zobrazovaná data aby odpovídala skutečnosti.

Pro aktualizaci obsahu po vrácení ze stavu Suspending lze použít přihlášení k odběru události **Resuming**.

Vypnutí aplikace (Close)

Obecně lze říci, že ve Windows 8 není potřeba aplikace vypínat. Aplikace na pozadí jsou ve stavu Suspended a tím pádem nezpomalují systém. I přes to ale uživatel může aplikaci vypnout pomocí klávesové zkratky Alt+F4 a nebo specifickým gestem, viz podkapitola Dotyková gesta.

Aplikace nesmí mít ve svém uživatelském rozhraní žádný ovládací prvek pro vypnutí aplikace. Pokud tomu tak je, aplikace neprojde certifikačním procesem.

Pokud byla aplikace ukončena uživatelem, přejde do stavu Suspended, následně Terminated a poté, většinou po deseti sekundách, přejde do stavu NotRunning.

Microsoft nedoporučuje vypínání aplikace programově, tedy v kódu. Pokud všem aplikace zaznamená zásadní chybu, může se z bezpečnostních důvodů ukončit. Stav aplikace je poté stejný jako kdyby byla aplikace neočekávaně ukončena.

Neočekávané ukončení aplikace (Crash)

Ani pád aplikace, či její neočekávané ukončení z důvodu chyby, nejsou ve Windows 8 opomenuty. V případě nečekané chyby musí být aplikace ihned ukončena a nesmí zobrazovat žádné chybové hlášky nebo varování. Aplikace dá rychlým, nečekaným ukončením uživateli najevo, že se něco porouchalo.

Pokud v aplikaci nastane chyba, Windows 8 nabídne uživateli možnost odeslání problému. Pokud uživatel problém odešle, Microsoft sesbírá veškerá data, při kterých chyba nastala, a připraví je pro vývojáře. Vývojář tak může vylepšit svou aplikaci tím, že sleduje svou aplikaci ve Windows Dev Center.

Při znovuspuštění aplikace je její předchozí stav nastaven na NotRunning.

Odstranění aplikace (Removal)

Při odinstalování aplikace jsou smazána její veškerá lokální data. Odstranění se ovšem netýká obsahu, který byl aplikací pořízen, jako dokumenty, fotky a podobné.

5.4.2 Rozvržení aplikace

Charakteristické pro aplikace pro dotykové rozhraní je velké odsazení od horního, levého a pravého okraje. Díky tomu je uživatel nepřímo naváděn ke konzumaci obsahu horizontálně. Tento postup by zvolen aby se používání Windows 8 přiblížilo prohlížení

knih, kde se obsah změní po otočení stránky. Proto je pro uživatele přirozenější „listování“ obsahu horizontálně, než vertikálně, jako je tomu například u webových stránek.

Rozložení stránky je zabudováno již v defaultních šablonách ve Visual Studio 2012 a vyšší. Je založeno na takzvaném Grid systému (Microsoft, Laying out an app page, 2014). Tedy systému sítě. Tato síť se skládá z jednotek 20x20 pixelů. Každá tato jednotka je rozdělena na pod-jednotky, skládající se z 5x5 pixelů. Každá čtvereční jednotka má tedy 16 čtverečních podjednotek. Tato síť je použita z důvodu přesného vykreslení obsahu na jednotlivé pixely. Obsah by poté neměl být na okrajích rozmazán ani nijak jinak porušen.

Samotná hlavička stránky aplikace by měla začínat 5 jednotek od shora a 6 jednotek zleva. Hlavní text je „SeogeUI“ velikosti 20, netučný. Obsah stránky začíná po 7 jednotkách shora a 6 jednotkách zleva. Spodní odsazení pro obsah je různé. Toto je použito také v ukázkové aplikaci. Hlavní text je zde „CzuBus“ a je odsazen přesně podle výše uvedeného Grid systému.

Obrázek 7 - Základní rozložení stránky – ZDROJ: vlastní

Toto je pouze základní rozložení stránky. Microsoft ale doporučuje rozložení a velikosti pro ovládací prvky a mezery mezi nimi. Toto rozložení vždy vychází z Grid systému a tyto mezery jsou vždy násobky jednotek nebo pod-jednotek. Rozložení dalších prvků je nad rozsah této práce.

5.5 Ovládací prvky uživatelského rozhraní

Tato podkapitola je zpracována na základě online dokumentace (Microsoft, Controls, 2013) a na základě některých ukázkových aplikací (Microsoft, Sample app pack, 2014).

Microsoft připravil pro vývojáře velké množství již předpřipravených ovládacích prvků, které již splňují všechny návrhové vzory a designové předlohy. Součástí těchto prvků je také sada ovládacích ikon, specifických pro Windows 8. Tyto prvky často sdílejí název s prvky z předchozích verzí systému, avšak jejich podoba je ve valné většině případů značně odlišná. Pro účely této práce bylo vybráno pět náhodných ovládacích prvků. Rozbor a popis všech ovládacích prvků je nad rámec této práce.

Použití a implementace ovládacích prvků bude předvedena na prvku ListView, který je součástí přiložené aplikace. Implementace ostatních prvků je velmi podobná a v zásadě se příliš neliší.

5.5.1 ListView

ListView je kolekce položek v rámci aplikace. Je optimalizován pro ovládání dotykem a umožňuje uživateli snadno a intuitivně ovládat a konzumovat obsah. Používá se především v případech, kdy je potřeba zobrazit větší množství položek, jako například 10 oblíbených receptů, nebo kolekce obrázků ve fotogalerii. Jako ostatní prvky pro Windows 8 aplikace pro dotykové rozhraní, i tento je ve Visual Studiu 2013 předpřipraven.

Implementace do aplikace

Jako první věc, kterou je pro ListView potřeba získat je datový zdroj. Datový zdroj slouží nejen jako standartní zdroj dat, ale jako takzvaný „binded“ datový zdroj. „Binded“ datový zdroj lze přeložit jako „oboustranný datový zdroj“. Znamená to totiž, že po připojení daného datového zdroje k připravenému ListView se tyto dva prvky propojí navzájem. Kdykoliv se po tomto propojení změní hodnoty v datovém zdroji, tak jsou okamžitě promítnuty i do ListView. Je však velmi důležité, aby ListView odkazoval vždy na stejný datový zdroj v paměti.

Ukázková aplikace obsahuje na hlavní stránce dva objekty ListView, jedno pro zobrazení odjezdů ze školy a druhý pro zobrazení odjezdů do školy.

```
routes = CzuBus.BusReader.BusManager.getBuses();

route0 = routes[0];
route1 = routes[1];

dataList0 = new WinJS.Binding.List(getDeparturesViewData(route0));
dataList1 = new WinJS.Binding.List(getDeparturesViewData(route1));
```

Ukázka kódu 8 - Připravení datového zdroje pro ListView

Jak lze vidět na ukázce *Ukázka kódu 8* - Připravení datového zdroje pro ListView, jako první se naplní proměnná *routes* metodou getBuses(). Tato metoda vrátí pole dvou prvků, přičemž jeden obsahuje pět nejbližších odjezdů ze školy a druhý do školy. Následně se tyto dva prvky vloží do dvou různých proměnných a poté se vytvoří nový objekt WinJS.Binding.List, který bude později sloužit jako datový zdroj pro zobrazované ListView. Metoda getDeparturesViewData() pouze převede objekt route0 a route1 na jednodušší JavaScriptové objekty.

Následně je nutné připravit objekt ListView, který se bude zobrazovat. Z důvodu přehlednosti této práce je v následující ukázce kódu zobrazen pouze jeden ovládací prvek ListView. V ukázkové aplikaci jsou však dva.

```
                <div id="route0_ListView" style="height: 100%;"
                    data-win-control="WinJS.UI.ListView"
                    data-win-options="{ oniteminvoked : handler, itemDataSource :
DataSource.route0ItemList.dataSource, itemTemplate:
select('#route0_ListViewTemplate') }">
                </div>
```

Ukázka kódu 9 - Prvek ListView v HTML

V ukázce kódu *Ukázka kódu 9* - Prvek ListView v HTML je důležité si povšimnout dvou atributů, a sice **data-win-control**, která říká, o jaký ovládací prvek se jedná a **data-win-options**, kterým lze nastavit daný ovládací prvek. Toto nastavení je však možné provést také v kódu. Záleží na vývojáři, který přístup zvolí. V ukázkové aplikaci se nejdříve nastaví ovladač, který se má spustit při kliknutí na jednu z položek ListView. Tato akce se nazývá *oniteminvoked*. Dále je nastaven úvodní datový zdroj, což je položka *itemDataSource*. Tuto položku je také možné nastavit programaticky, což je využíváno i v ukázkové aplikaci. Pokud je totiž potřeba při běhu aplikace upravit datový zdroj, pro účely ukázky je přemazán celý datový zdroj, jako je tomu na ukázce kódu *Ukázka kódu 10* - Přiřazení datového zdroje k ListView.

```
var list0 = document.getElementById("route0_ListView").winControl;
list0.itemDataSource = dataList0.dataSource;

var list1 = document.getElementById("route1_ListView").winControl;
list1.itemDataSource = dataList1.dataSource;
```

Ukázka kódu 10 - Přiřazení datového zdroje k ListView

Jako poslední a velmi důležitá položka je zde *itemTemplate*. Tato položka slouží k nastavení jakási šablony jednotlivé položky ListView. Tato šablona je ukázána v ukázce kódu *Ukázka kódu 11*- Šablona jednotlivé položky ListView. Hodnota parametru data-win-control v tomto případě musí být nastavena na objekt šablony. Pak již pouze stačí kdekoliv v této šabloně přiřadit libovolnému prvku atribut **data-win-bind**, kde lze nastavit, jaká část jednotlivého objektu se má vypsat. Pokud by datový zdroj obsahoval 20 objektů typu auto, kde by objekt auto měl proměnné jméno a barva, poté by bylo možné v šabloně položky prvku ListView nastavit na jednom místě „data-win-bind='innerText: jmeno'“ a na druhém „data-win-bind='innerText: barva'“.

```
<div id="route0_ListViewTemplate" data-win-control="WinJS.Binding.Template">
        <div style="width: 270px; overflow: auto; display: block; background-
color: #2a2a2a;padding: 8px;">
            <div>

                <div>
                    <h1 data-win-bind="innerText: remaining"></h1>
                </div>
                <div >
                    <h4 data-win-bind="innerText: busName" style="float:
left;margin-right: 5px;"></h4>
                    <h3 data-win-bind="innerText: stationName" style="float:
left;margin-left: 5px; margin-right: 5px;"></h3>
                    <h3 data-win-bind="innerText: time" style="float:
left;margin-left: 5px; margin-right: 5px;"></h3>
                </div>
            </div>
        </div>
    </div>
```

Ukázka kódu 11- Šablona jednotlivé položky ListView

Nyní se po spuštění aplikace stane to, že se nejprve načtou odjezdy autobusů a uloží se do datových proměnných, poté se při vykreslování HTML napojí datový zdroj na objekt ListView a následně se podle šablony jednotlivého prvku ListView vykreslí všechny odjezdy autobusů.

Na stejném principu funguje značná část ostatních ovládacích prvků – tedy nastavení prvku, datového zdroje a šablony vykreslení.

5.5.2 Další ovládací prvky

Název ovládacího prvku	**Popis**
Back button	Navigaci v historii o položku zpět pomocí tlačítka.
Bottom app bar	Spodní lišta s nastavením, vhodná pro vložení příkazů aplikace.
Button	Standardní ovládací tlačítko.
Checkbox	Zaškrtávací pole, pomocí kterého může rozlišit dvě různé volby.
Context menu	Kontextové menu, které zobrazí krátký seznam příkazů a možností, které jsou dostupné v konkrétním kontextu.
Date picker and time picker	Dovolí uživateli vybrat datum nebo čas.
Drop-down list	Rolovací seznam hodnot, kde uživatel může zvolit pouze jednu hodnotu. Zobrazena je vždy aktuálně vybraná položka. Ostatní jsou zobrazeny pouze po aktivaci prvku Drop-down list.
Flip view	Zajistí možnost přepínat mezi prvky v libovolné kolekci. Typicky například listování mezi fotografiemi.
Flyout	Vyskakovací okno, které do zobrazí pouze v reakci na akci uživatele a zobrazí doplňující informace, případně i možnost volby.
Grid view and list view	Uspořádaný seznam položek.
Hub	Základní ovládací prvek, často označovací jako vstupní prvek aplikace. Pomocí tohoto prvku lze zobrazit základní kategorie aplikace. Jedná se o základní prvek hierarchické navigace.
Hyperlink	Odkaz (součástí textu) sloužící k přechodu na jinou stránku případně jiné místo v aplikace.
Label	Popisné jméno ovládacího prvku.
Media	Obstarává přehrávání audio a video obsahu.
Message dialog	Dialogové zprávy, obsahující zásadní sdělení uživateli s možností volby.
Progress bar and progress ring	Prvek, který svým pohybem dává uživateli jistotu, že aplikace stále pracuje. Může zobrazovat přibližnou dobu zpracování v procentech, nebo pouze ujišťovat, že operace stále probíhá.
Radio button	Podobné prvku Checkbox, avšak tento prvek se vždy vyskytuje v rámci jednoho kontextu s více Radio button prvky. Je možné zaškrtnout vždy maximálně jeden z nabízených možností.
Rating	Hodnocení. Tento prvek vykreslí pět hvězd, kde je možné vybrat pouze několik z nich a tím tak nastavit hodnocení.
Scroll bar	Lišta, informující o přibližné celkové délce obsahu a aktuální relativní pozici. Pomocí tohoto prvku je také možné se posouvat v rámci delšího obsahu.
Search box	Pole pro zadání výrazu, který má být vyhledán.
Select	Podobné prvku Drop-down list, avšak tento prvek je vždy rozbalen.
Semantic zoom	Speciální prvek, který visuálně zmenší obsah a nabídne tak uživateli rámcovou navigaci napříč aplikací.
Slider	Potenciometr (posuvník), sloužící k nastavení aktuální hodnoty z celkové možné.

Text input box	Pole pro zadání textu.
Text input spell check	Pole pro zadání textu s implementovanou automatickou korekcí textu.
Toggle switch	Základní prvek ro vypnutí/zapnutí některé funkce.
Tooltip	Je krátký popis, který se váže na některý ovládací prvek. Tento prvek pomáhá uživateli pochopit význam některého ovládacího prvku.
Top app bar	Horní lišta s nastavením, vhodná pro vložení navigace, nebo případně jiných ovládacích prvků pro pohyb v aplikaci.

Tabulka 3 - Seznam ovládacích prvků ve Windows 8 aplikacích

5.6 Sestavení aplikace

Sestavení aplikace je jeden z finálních kroků po vytvoření aplikace. Sestavením aplikace se rozumí vytvoření instalačního balíčku, který je následně možné odeslat do Windows Store obchodu, kde bude aplikace nabízena uživatelům (Microsoft, Publish, 2014). Velkou výhodou je, že se vývojář již nemusí zajímat o instalátor aplikace. Vše potřebné je v již zmíněném balíčku aplikace.

Samotný balíček aplikace je založen na takzvaných Open Packing Conventions standardech, které jednoznačně definují strukturu uložených dat pro formát ZIP. Celý tento balíček se skládá z následujících

- **App payload** (Základ aplikace). Soubory s kódem aplikace.
- **App manifest** (nastavení a informace o aplikaci). Tento soubor představuje nastavení a informace o aplikaci, jako například podporované orientace obrazovky, seznam periferií potřebných ke správnému fungování a informace o sestavení a aktualizacích.
- **App block map** (Soubor integrity aplikace). Obsahuje seznam všech souborů v App payload a jejich hash hodnotu. Díky tomu zajišťuje integritu a zjednodušuje aktualizace.
- **App signature** (podpis aplikace). Digitální podpis aplikace, zajišťující, že aplikace byla vytvořena ověřeným autorem a nebyla od momentu sestavení a podepsání upravena.

5.7 Obchod s aplikacemi (Windows Store)

Sestavený balíček aplikace je potřeba nahrát do Windows Store a nastavit několik věcí. Je nutné vyplnit název aplikace a její popis. Na tyto části je důležité klást velký důraz, jelikož to jsou první údaje, které o aplikace uživatelé uvidí. Stejně důležitá je i volba kvalitní grafiky a obrázků, společně s ukázkovými otisky obrazovek.

Dále je nutné nastavit pro jaké světové trhy je aplikace připravena a jaké podporuje jazyky. Microsoft dovoluje publikovat pouze do regionů, jejichž jazyk je aplikací oficiálně podporován. Za další je nutné zvolit správné hodnocení aplikace, takzvaný Age rating. Tedy od kolika let je aplikace přístupná. Pokud je nastaven špatně, aplikace neprojde certifikací.

Finální a nejdůležitější krok je zvolit cenu aplikace. Ta může být zdarma, obsahujíc reklamy, nebo za poplatek. Případně obojí. Vývojářům je také často doporučována implementace dokupovatelného obsahu do aplikace.

5.8 Certifikace

Tato podkapitola byla z části zpracována podle online dokumentace (Microsoft, Certification, 2014).

Aby Microsoft předešel vkládání škodlivých aplikací do svého internetového obchodu Windows Store, přidal pro vývojáře povinnost nechat svou aplikaci certifikovat. Tato certifikace má zajistit, že se ve Windows Store nebudou objevovat škodlivé, nesmyslné, nefunkční nebo urážející aplikace.

Celý certifikační proces začíná nahráním aplikace do Windows Store. Během nahrávání se zkontroluje integrita aplikace, a zda nechybí zásadní soubory. Pokud aplikace tímto testem projde, zobrazí se vývojáři v seznamu instalačních balíčků, kde ji může odeslat k automatizované certifikaci.

Tento automatizovaný proces nejdříve zkontroluje aplikaci na různé druhy virů. Následuje testy stability aplikace, bezpečnostní, výkonnosti, přístupnosti a několik dalších testů. Aby Microsoft práci vývojářů zjednodušil, připravil instalovatelnou aplikace Windows App Certification Kit, která veškeré tyto testy provede na lokálním počítači vývojáře. Ten tak nemusí čekat na výsledek tohoto testu online.

Pokud aplikace projde i těmito testy, putuje dále k živým testerům, kteří aplikaci nainstalují, otestují a zhodnotí. Počet těchto testerů závisí na velikosti aplikace a na reputaci vývojáře. Pokud aplikace obsahuje například testovací účet, je důležité tuto skutečnost předat testerovy pomocí online rozhraní při odesílání aplikace k certifikaci.

Následuje konečná fáze, a sice shrnutí certifikace a výsledek, zda aplikace prošla či nikoliv. Pokud neprošla, vývojář uvidí, jakým konkrétním testem neprošla. Pokud neprošla kvůli špatné stabilitě, obdrží vývojář také další informace, aby mohl lépe identifikovat chybu.

V případě, že aplikace projde certifikačním procesem, může být vložena do obchodu pro aplikace ihned, nebo lze nastavit datum vystavení. Tímto lze vývoj aplikace označit za ukončený. Další část jako aktualizace aplikace a analýzy prodejnosti a stahovanosti jsou již nad rámec této práce.

5.9 Zhodnocení

Firma Microsoft přišla s vlastním, jedinečným uživatelským rozhraním, které jako celek funguje velmi dobře. Kvalitně je také provedena dokumentace vývoje aplikací, společně s nápovědou a ukázkovými aplikacemi. Jak již bylo řečeno v úvodu, nový systém by se na trhu neudržel dlouho, pokud by pro něj neexistovaly klíčové aplikace. A tak kromě kvalitní dokumentace Microsoft pořádá řadu přednášek a seminářů, pro vývoj připravil několik různých platforem a dovolil vývojářům tyto platformy i propojit.

Díky tomu vznikl velmi kvalitní ekosystém, který trpí především kvůli tomu, že přišel pozdě. Trh je totiž v současné době zahlcen zařízeními s konkurenčními systémy. Avšak lze předpokládat, že Microsoft bude ukrajovat opět větší a větší část trhu, jelikož používá moderní technologie jako HTML 5 a CSS 3 a nemalou část svých prostředků evidentně investuje do podpory vývoje.

Vývoj pro Windows 8, jak se dalo předpokládat, velmi těží z předchozích zkušeností a předchozích systémů. Vytvořil originální běhové prostředí a zároveň ho doplnil a velmi kvalitní příručkou pro zpracování nových aplikací pro toto prostředí. Připravená široká škála již předpřipravených šablon a prvků navíc dává vývojářům možnost vyvíjet aplikace ve velmi krátkém čase a zaměřit se přitom především na obsah.

Co se týče vývojového studia, zde nebyl nalezen téměř žádný nedostatek. Několik výtek připadá pouze k občas problematické práci při debugování samotné aplikace a občasnému vynechání IntelliSense. Vzhledem ke komplexnosti a použitelnosti lze toto považovat za opravdu malichernost.

U samotného vývoje aplikací je možná problémů více, avšak o žádném z nich nelze mluvit jako o zásadním. Patrně největším problém je nedostatek aplikací v obchodě. Konkrétně nedostatek kvalitních aplikací, jelikož Microsoft se v rámci snahy naplnit obchod co nejdříve nevěnoval certifikačnímu procesu tak, jak by měl. Výsledkem je, že tento obchod je plný spoustou nekvalitních aplikací. Toto ale neubírá na kvalitě celého, Microsoftem navrženého, vývojového procesu od nápadu až po prodej aplikace.

Východiskem by bylo, aby se Microsoft soustředil na proces certifikace ještě více a byl ve svých pravidlech ještě přísnější. Aplikací by sice nebylo takové množství, avšak obecně by byly mnohem kvalitnější. Podpora vývojářů je téměř perfektní, avšak trpí na stále malém rozšíření Windows 8. Pravděpodobně by bylo vhodné změnit obchodní strategii a zaměřit se více na prodej Windows 8.

6 Závěr

Cílem práce bylo představit nejnovější nástroje a možnosti pro vývoj aplikací pro dotykové rozhraní operačního systému Windows 8. V první části proto byla tato práce zaměřena na základní popis operačního systému Windows 8, a to především na nové uživatelské rozhraní. Po jeho nastudování a více než ročním používání, bylo toto rozhraní popsáno a vysvětleno v kapitole číslo 3.

Následující kapitoly již byly zaměřeny na podstatu této práce, a sice vývoj aplikací pro dotykové rozhraní. V kapitole číslo 4 byly popsány nástroje pro vývoj, tedy vývojový software a jeho možnosti. Konkrétně byl v této kapitole předveden software Visual Studio a software Blend for Visual Studio.

V kapitole číslo 5 byla práce zaměřena již na samotný vývoj. Byly zde představeny návrhové vzory, vývojové platformy a ovládací prvky aplikací pro dotykové rozhraní. V rámci praktické části této práce byla vyvinuta ukázková aplikace, na které byly některé vývojové prvky průběžně demonstrovány. Například v kapitole 5.5.1 byla podrobně rozebrána implementace ovládacího prvku ListView, čímž byla zároveň předvedena rychlost a efektivita vývoje. V závěrečné části této kapitoly bylo provedeno celkové zhodnocení vývoje na základě zkušeností a znalostí nabytých při zpracovávání předchozích kapitol.

7 Bibliografie

Brockschmidtn, K. (2012). *Programming Windows 8 Apps with HTML, CSS and JavaScript.* Redmond, Washington: Microsoft Press, 2012.

Burns, K. (2012). *Beginning Windows 8 Application Development: XAML Edition.* Apress, online.

DeGrasse, M. (2013). *Smartphone processor market may have peaked for now*. Načteno z RCRWireless: http://www.rcrwireless.com/article/20130719/chips/smartphone-processor-market-may-peaked-now/

Freeman, A. (2012). *Metro revealed - Building Windows 8 apps with XAML and C#.* New York: Springer Science+Business Media New York, 2012.

Freeman, A. (2012). *Metro Revelaed - Building Windows 8 Apps with HTML5 and Javascript.* New York 233 Spring Street, 6th Floor: Springer Science+Business Media New York, 2012.

Mainelli, T. (1. 5 2013). *IDC Worldwide Tablet Tracker*. Načteno z http://www.idc.com/getdoc.jsp?containerId=prUS24093213

Microsoft. (2012). *Quickstart: Adding a ListView (Windows Store apps using JavaScript and HTML)*. Načteno z Windows Dev Center: http://msdn.microsoft.com/en-us/library/windows/apps/hh465496.aspx

Microsoft. (2013). *App architecture*. Načteno z Microsoft Dev Center: http://msdn.microsoft.com/en-us/library/windows/apps/br211361

Microsoft. (2013). *App layout*. Načteno z Microsoft Dev Center: http://msdn.microsoft.com/library/windows/apps/br211361

Microsoft. (2013). *Application lifecycle*. Načteno z Windows Dev Center: http://msdn.microsoft.com/en-us/library/windows/apps/hh464925.aspx

Microsoft. (2013). *Compatibility*. Načteno z Microsoft Dev Center: http://windows.microsoft.com/en-us/windows-8/get-apps-devices-working

Microsoft. (2013). *Controls*. Načteno z Microsoft Dev Center: http://msdn.microsoft.com/library/windows/apps/dn439320.aspx

Microsoft. (2013). *Design patterns*. Načteno z Microsoft Dev Center: http://msdn.microsoft.com/en-us/windows/apps/hh779072.aspx

Microsoft. (2013). *Gestures*. Načteno z Microsoft Dev Center: http://msdn.microsoft.com/en-us/library/windows/apps/hh761498.aspx

Microsoft. (2013). *Get Started*. Načteno z Windows Dev Center: http://msdn.microsoft.com/en-US/windows/apps/br211386

Microsoft. (2013). *MSDN - Dev center*. Načteno z Get Started: http://msdn.microsoft.com/en-US/windows/apps/br211386

Microsoft. (2013). *Navigation patterns*. Načteno z Microsoft Dev Center: http://msdn.microsoft.com/library/windows/apps/hh761500.aspx

Microsoft. (2013). *Supporting Windows RT and ARM processors*. Načteno z Microsoft Dev Center: http://msdn.microsoft.com/en-us/library/windows/apps/jj863300.aspx

Microsoft. (2013). *Windows App Certification Kit* . Načteno z Microfost Dev Center: http://msdn.microsoft.com/en-us/windows/apps/bg127575

Microsoft. (2013). *Windows Dev Center - Windows Store apps*. Načteno z MSDN: http://msdn.microsoft.com/en-US/windows/apps/

Microsoft. (2013). *Windows User Experince Design Principles*. Načteno z Microsoft Dev Center: http://msdn.microsoft.com/en-us/library/windows/desktop/dd834141.aspx

Microsoft. (2014). *Certification*. Načteno z Microsoft Dev Center: http://msdn.microsoft.com/en-us/library/windows/apps/hh694083.aspx

Microsoft. (2014). *Design Windows Store apps using Blend*. Načteno z Microsoft Dev Center: http://msdn.microsoft.com/en-US/library/windows/apps/jj129478.aspx

Microsoft. (2014). *Laying out an app page*. Načteno z Microsoft Dev Center: http://msdn.microsoft.com/en-us/library/windows/apps/hh872191.aspx

Microsoft. (2014). *Publish*. Načteno z Windows Dev Center: http://msdn.microsoft.com/en-US/windows/apps/br230836

Microsoft. (2014). *Sample app pack*. Načteno z Microsoft Dev Center: http://code.msdn.microsoft.com/windowsapps/Windows-8-Modern-Style-App-Samples

Microsoft. (2014). *Visual Studio*. Načteno z MSDN: http://msdn.microsoft.com/en-us/vstudio/cc136611.aspx

Microsoft. (2014). *Windows Runtime*. Načteno z Microsoft Dev Center.

Microsoft. (nedatováno). *UX guidelines for Windows Store apps*. Načteno z Microsoft: http://msdn.microsoft.com/library/windows/apps/hh465424.aspx

Miyasaka, R. (6. 4 2012). *What is the difference between Windows 8, WinRT, and Windows RT?* Načteno z programmers.stackexchange.com: http://programmers.stackexchange.com/questions/155521/what-is-the-difference-between-windows-8-winrt-and-windows-rt

Portnoy, S. (1. 2 2013). *Microsoft Surface with Windows RT tablet sales disappoint in fourth quarter*. Načteno z ZDNet: http://www.zdnet.com/microsoft-surface-with-windows-rt-tablet-sales-disappoint-in-fourth-quarter-7000010688/

Rozenblit, J. (19. 3 2013). *Building Secure Windows Store Apps*. Načteno z MSDN Blog: http://blogs.msdn.com/b/cdndevs/archive/2013/03/19/building-secure-windows-store-apps.aspx

Sharp, J. (2012). *Microsoft Visual C# 2010 : krok za krokem*. Praha 4: Albatros Media a. s.

Sinfosky, S. (2011). *Designing the Start screen*. Načteno z Building Windows 8: http://blogs.msdn.com/b/b8/archive/2011/10/04/designing-the-start-screen.aspx?Redirected=true

7.1 Seznam obrázků

7.2 Seznam tabulek

7.3 Seznam ukázek kódu

7.4 Seznam příloh

Součástí této práce je CD-ROM, obsahující ukázkovou aplikaci ve formě projektu aplikace Visual Studio.