ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA

PROVOZNĚ EKONOMICKÁ FAKULTA

Katedra informačního inženýrství

Diplomová práce

Vývoj mobilní aplikace pro Windows Phone 8

Bc. Ondřej David

Čestné prohlášení

Prohlašuji, že svou diplomovou práci „Vývoj mobilní aplikace pro Windows Phone 8“ jsem vypracoval samostatně pod vedením vedoucí diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30. 11. 2015 _____

Poděkování

Rád bych poděkoval Ing. Daně Vynikarové, PhD. za vedení mé diplomové práce, konzultace a podnětné rady na její zlepšení. Dále bych chtěl poděkovat své rodině a blízkým.

Souhrn

Hlavním cílem této diplomové práce je seznámit se s REST architekturou a vytvořit klientskou aplikaci pro Windows Phone 8 na webové službě, která obsahuje implementaci REST API.

Nabyté poznatky z REST architektury byly zapracovány přímo v rámci vývoje konkrétní aplikace. Výsledkem diplomové práce je fakturační mobilní aplikace vytvořená na platformě Windows Phone 8 a postavená na službě Fakturoid, která měla doposud mobilní aplikace pouze pro platformy Android a iOS.

Mobilní aplikace je určena pro živnostníky, proto by bylo vhodné ji do budoucna rozšířit o přehledy DPH a dalších užitečných funkcí a automatizaci vytváření daňových přiznání.

Klíčová slova: REST, API, klient, server, architektura, Windows Phone, XML, XAML, JSON, HTTP

Summary

Main goal of this master thesis is to apprise with REST architecture and to create client application for Windows Phone 8 based on web service which include REST API implementation.

Lessons learned in the REST architecture have been incorporated directly into the development of specific application. The result of this master thesis is an invoicing mobile application developed on Windows Phone 8 platform, and based on service called Fakturoid. This service had previously only Android and iOS applications.

The mobile application is developed primary for tradesmen. Nice to have feature could be extension for VAT overview and automatic creation of tax returns.

Keywords: REST, API, client, server, architecture, Windows Phone, XML, XAML, JSON, HTTP.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Každá doba má své trendy. Některé se mění v řádu desítek, jiné v řádu jednotek let. Není to tak dávno, co byly nejzajímavějšími obory IT tvorba webových stránek, portálů, a grafických návrhů. Autoři se předháněli o to, kdo vymyslí originální design, novou myšlenku, nebo určí směr ve svém oboru. Vznikaly různé diskuze, kdo chápe pojem design lépe a kdo je schopen přinést uživateli více zážitků z užívání svého díla. V té době byl trend jasně daný a nikdo nepředpokládal velké změny. Přesto přišlo překvapení v roce 2007, kdy byl společností Apple představen první skutečně použitelný dotykový chytrý telefon, který nabídl vývojářům možnost vytvářet vlastní aplikace pro zákazníky podle vlastního grafického návrhu. Díky tomu se světu otevřel úplně nový obor tvorba mobilních aplikací a výrazně se proměnil segment chytrých telefonů.

S rostoucím počtem uživatelů chytrých telefonů roste poptávka po mobilních aplikacích. Díky tomu je tvorba mobilních aplikací dynamickým, lákavým a rychle se rozvíjejícím odvětvím.

Čím dál více uživatelů využívá chytrý telefon jako pracovní nástroj, centrum zábavy a možnost sledování veškerého dění ve svém okolí. Uživatelé chtějí stále více vyřizovat většinu denních povinností na svém oblíbeném zařízení – emailovou konverzaci, objednání k lékaři, sdílení fotek, vytváření pracovních schůzek apod. Z toho důvodu se většina aplikací přesouvá z webového prostředí do mobilního. Proto se tato práce věnuje implementaci webové služby na mobilní platformu Windows Phone 8.

Diplomová práce je rozdělena na tři základní části. První část obsahuje vysvětlení pojmů a technologií, které jsou pro vytvoření mobilní aplikace stěžejní. Druhá část je zaměřena na prvky mobilní platformy Windows Phone 8 a pravidla pro tvorbu aplikací na této platformě. Poslední část se zabývá vytvořením mobilní aplikace. Celá tvorba je zde popsána od počátku celého procesu, který začíná stanovením cílů, na jejich základě je vytvořen návrh řešení. Po něm následuje návrh architektury reflektující komunikaci aplikace s dalšími systémy. Následuje seznam základních use case podle návrhu řešení, na základě kterého je sestaveno rozložení obrazovek. Dále je popsán kompletní návrh user interface mobilní aplikace a na závěr objasněno, jakým způsobem se hotová aplikace publikuje na Windows Store .

2 Cíl práce a metodika

2.1 Cíl práce

Cílem rešeršní části diplomové práce je seznámit s architekturou klient-server, na ní závislou architekturou REST a REST API. Dále prozkoumat a popsat základní principy vývoje aplikací pro operační systém Windows Phone 8.

Cílem praktické části diplomové práce je výběr služby, která využívá REST API a návrh a implementace aplikace této služby pro operační systém Windows Phone 8, její otestování a následné publikování.

2.2 Metodika

Metodika řešené problematiky diplomové práce je založena na studiu odborných informačních zdrojů a jejich praktické využití pro vývoj mobilní aplikace.

Teoretická část má za úkol seznámit s architekturou klient-server, detailně popsat základy a přístupy k architektuře REST, objasnit pojem RESTful, popsat způsob použití a uvést příklady práce s REST API. Dále se bude práce věnovat základům systému Windows Phone 8, jeho prvkům a procesu vývoje mobilní aplikace na této platformě.

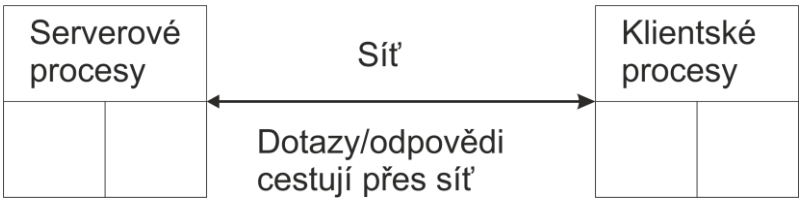
Praktická část má za úkol zvolit konkrétní webovou službu, která poskytuje REST API a na základě ní vytvořit mobilního klienta pro výše uvedenou platformu. Uvést příklady konkrétních REST API zvolené služby, jejich využití a konkrétní příklady vývoje mobilního klienta zvolené služby ve Windows Phone 8. Pokud to bude možné, předvést publikování aplikace.

3 Přehled pojmů a technologií

3.1 Architektura klient/server

Výraz klient/server se využívá k popsání počítačového modelu počítačových systémů. Tento model je založen na distribuci funkcí mezi dva nezávislé a autonomní procesy – server a klient. Klient je proces, který požaduje specifické služby po serverovém procesu. Server je proces, který poskytuje služby požadované klientem. Klient a server se mohou nacházet v jednom počítači, nebo každý samostatně, pak jsou oba propojeni sítí. (1)

Pokud jsou procesy klienta a severu na dvou a více nezávislých počítačích na stejné síti, server je schopen poskytovat služby pro více než jednoho klienta. Stejně tak i klient může žádat služby po více serverech na síti bez ohledu na fyzickém i logickém umístění serverů. Síť propojuje servery a klienty a poskytuje tak medium, přes které mohou klienti a servery komunikovat. Obrázek 1 ukazuje základní počítačový model klient/server. (1)



Obrázek 1: Základní klient/server počítačový model (1)

Z obrázku 1 je také patrné, že mohou být služby poskytovány několika počítači v síti naráz. Klíčovým pro tento model je místo, kde se zpracovávají dotazy. Např. u klient/server databáze je systémová funkcionalita rozdělena mezi server a více mezi sebou propojených klientů tak, aby mohli vykonávat úkoly rozdělené mezi sebe. (1)

3.1.1 Tlustý a tenký klient/server

Klient nebo server je tak pojmenovaný v závislosti na množství zpracování procesů na té či oné straně. Tenký klient zpracovává minimum procesů na své straně, naopak tlustý klient zpracovává na své straně značné množství procesů. Koncept tlustého klienta nebo tlustého serveru je otázkou jednoho kritéria – poměr rozmístění celé aplikace na klienta a server. (1)

3.1.1.1 Tlustý klient

Tato architektura dává více aplikační funkcionality na klientův stroj. Problém je pouze v údržbě těchto systémů, právě proto, že většina práce závisí na klientech. (1)

3.1.1.2 Tlustý server

Tato architektura přenáší většinu aplikační funkcionality na serverový stroj. Typicky tak server poskytuje více abstraktní a náročnější služby. Aktuální trend se ubírá spíše směrem k tlustým serverům. V tom případě klient většinou používá pouze prohlížecí mód na koncovém zařízení. Velká výhoda tohoto řešení je jednoduchost ve správě celého řešení, protože jediný software, který je třeba měnit a upravovat, běží na serveru. (1)

3.1.2 Stavový a bezstavový server

Bezstavový server je server, který zachází s každým dotazem nezávisle na jakémkoliv předchozím dotazu. Velkou výhodou bezstavového řešení je jednoduchost návrhu serveru, protože nepotřebuje dynamicky alokovat paměť na ukládání průběžně vyměněných informací během komunikace a nemusí řešit ztrátu spojení uprostřed komunikace. Je zde také jedna nevýhoda – někdy může být užitečné mít více informací o předchozích dotazech na serveru. Příkladem bezstavového serveru je WWW (World Wide Web) server. Pokud vynecháme cookies, přijímá dotazy na URI (Uniform Resource Identifier), které přesně specifikují potřebné dokumenty a není třeba žádného kontextu, paměti s předchozími dotazy oproti tradičním FTP (File Transfer Protocol) serverům, kteří provádí interaktivní relaci s uživatelem. Dotaz na soubor umístěný na serveru zahrnuje autentifikaci uživatele a nastavení aktuální složky, do které má uživatel přístup. (1)

3.1.2.1 Stavový server

Klientská data a informace o jejich stavu jsou udržovány serverem podle probíhající komunikace s klienty a server si také pamatuje, co klient požadoval a na základě toho udržuje informace o odpovědích na předchozí dotazy. Výhodou stavového serveru jsou efektivněji zpracovávané dotazy. Nevýhodou je případ, kdy na server chodí nespolehlivé a neúplné zprávy, pak se informace o různých stavech na serveru stávají neplatnými. Další nevýhodou je případ, kdy klient často selhává. Pak může být paměť serveru přehlcena nedokončenými požadavky klienta a různými stavy informací na serveru. Nejlepším příkladem stavového serveru je vzdálený souborový server. (1)

3.1.2.2 Bezstavový vs. stavový server

Existují analýzy porovnávající bezstavové a stavové servery. Stavový server uchovává klientská data (stav) od dotazu k dalšímu dotazu. Bezstavový server si neuchovává stav informací. Při používání bezstavového souborového serveru musí klient specifikovat celý název a adresu souboru v každém dotazu a přitom se pokaždé znovu autentifikovat. Používání stavového file serveru je snazší, na druhé straně bezstavový server je více robustní a ztráta spojení nezanechá stav v nefunkčním stavu a restart serveru nezpůsobí ztrátu stavu dat. Restartování klienta nemá na bezstavový server žádný vliv. (1)

3.1.3 Základní funkcionality

Hlavní operace vykonávané klientem jsou:

- Správa uživatelského rozhraní,
- kontrolovat a přijímat data vkládaná uživatelem,
- vykonávat aplikační logiku,
- vytvářet dotazy na serverovou databázi. (1)

Hlavní operace vykonávané serverem jsou:

- Přijmout a vykonat klientské dotazy do databáze,
- kontrolovat autorizaci,
- zajišťovat integritu dat,
- vykonávat dotazy, úpravy a na základě nich podávat klientovi patřičné odpovědi,
- udržovat systémový katalog,
- poskytovat souběžný přístup do databáze,
- poskytovat obnovovací nástroje. (1)

3.1.4 Topologie

Topologie odpovídá fyzickému umístění klientů a serverů v síti, přes kterou jsou spolu spojeni. Nejčastější topologie jsou následující:

a) Jeden klient, jeden server,
b) více klientů, jeden server,
c) více klientů, více serverů. (1)

3.2 REST

REST (Representational State Transfer) získal široké přijetí na webu jako jednodušší alternativa k SOAP (Simple Object Access Protocol) a WSDL (Web Services Description Language). Klíčovým bylo přijetí tohoto rozhraní hlavními hráči té doby na poli Web 2.0 služeb, jako jsou Yahoo, Google a Facebook. (2)

3.2.1 Základní informace

REST definuje sadu architektonických principů, s jejichž pomocí je možné navrhnout webové služby zaměřené na systémové zdroje, jejichž stav je adresován a přenášen přes HTTP (Hypertext Transfer Protocol) do širokého spektra klientů v různých programovacích jazycích. V posledních několika letech se REST stal dominantnějším prostředkem mezi webovými službami, zejména pro znatelně jednodušší použití. Tím začal vytlačovat SOAP a WDSL. (2)

Roy Fielding z University of Carolina představil myšlenku REST v dizertační práci „*Architectural Styles and the Design of Network-based Software Architectures*“, která analyzovala sadu softwarových architektonických principů, které využívají web jako platformu pro distribuované výpočty. V té době REST nepřitáhl tolik pozornosti jako dnes, kdy se vytváří různé frameworky speciálně jen pro práci s REST. (2)

Implementace webové služby REST vychází z následujících čtyř pravidel:

- Využití HTTP metod,
- bez stavů,
- výstavba URI podle adresářové struktury,
- formát koncových dat. (2)

3.2.2 Využití HTTP metod

Jednou z klíčových charakteristik RESTFul webových služeb je explicitní využití HTTP metod podle protokolu definovaného v RFC 2616. HTTP GET je definován jako metoda produkce dat, která je využívána klientskou aplikací k získání přístupu k webovému zdroji, získání dat z webového serveru, nebo k vykonání dotazu, který bude web serverem vyhodnocen a na základě toho vrátí sadu odpovídajících zdrojů. (2)

REST vyžaduje využívání HTTP metod explicitně a to v souladu s definicí protokolů. Na základě tohoto principu mapuje CRUD operace a HTTP metody jedna ku jedné podle následujících pravidel:

- Pro vytvoření zdrojů na serveru se využívá POST,
- pro získání zdroje se využívá GET,
- pro změnění stavu zdroje nebo úpravy se využívá PUT,
- k odebrání nebo smazání zdroje se využívá DELETE. (2)

Bohužel se často stává u různých web API (Application Programming Interface), že jsou HTTP metody použity nesprávně. Např. požadovaná URI adresa zavolaná pomocí HTTP GET obvykle fixně identifikuje právě jeden zdroj, nebo dotazovací řetězec v URI obsahuje fixní sadu parametrů, které definují vyhledávací kritéria. Je zde také mnoho případů, kdy se využívá HTTP GET ke spuštění transakčních akcí na serveru, jako například přidání záznamů do databáze. V tomto konkrétním případě HTTP GET není použita správně v rámci REST. Pokud web API používá HTTP GET na vyvolání procedury, vypadá to následovně (2):

```
GET /adduser?name=Robert HTTP/1.1
```

Tento návrh není vhodný, protože metoda obsahuje operaci měnící stav nad HTTP GET, navíc má tato metoda ještě vedlejší efekty. Pokud se metoda úspěšně provede, výsledkem volání je přidání nového uživatele jménem Robert do databáze. Hlavní problém je zde sémantika. Webové servery jsou navrženy tak, aby odpovídali na HTTP GET dotazy pomocí zdrojů nacházejících se na dotazované URI adrese, nikoliv přidávaly záznamy do databáze. Takové využití metody je v rozporu se stanoviskem HTTP 1.1 pro webové servery. (2)

Kromě výše zmíněné sémantiky je zde další problém s metodou GET – provedení mazání, úpravy, nebo přidání prvku do databáze láká spousty webových *cache* nástrojů a vyhledávacích *engine*, které mohou provést neúmyslné změny v databázi přístupem na uložený odkaz. Jednoduché řešení tohoto problému je přenést názvy parametrů a jejich hodnoty do XML (Extensible Markup Language) tagů, které je možné odeslat v těle HTTP POST dotazu následujícím způsobem (2):

```
POST /users HTTP/1.1
HOST: myserver
Content-Type: application/xml
```

```
<?xml Version="1.0"?>
<user>
      <name>Robert</name>
</user>
```

Tento příklad RESTful dotazu ukazuje správné užití HTTP POST s použitím dat uložených v těle dotazu. Na straně, která přijímá požadavek, může být vytvořen další zdroj přidáním části k adrese v těle požadavku, jak je uvedeného výše. To je možné realizovat přidáním prvku za „/users“. Tím vznikne vztah mezi novou entitou a jeho rodičem, specifikovaný v POST dotazu analogicky, jako podsložka ve složce. (2)

Klientská aplikace by měla obdržet nově vzniklou URI směřující pod „/users“, jak je ukázáno následujícím způsobem (2):

```
GET /users/Robert HTTP/1.1
HOST: myserver
Accept: application/xml
```

Používání GET tímto způsobem je explicitní a správné, protože GET slouží pouze k získání dat. (2)

Podobný refaktoring by se měl provést v případě, pokud se upravuje záznam pomocí metody GET, jak ukazuje následující případ (2):

```
GET /updateuser?name=Rober&newname=Bob HTTP/1.1
```

Tato operace změní atribut „name“ a může být použita u takto jednoduchých operací. Při komplexnějších operaci by tento vzor nefungoval. REST přístup je explicitně využívat HTTP metody a proto je lepší tuto operaci provést pomocí HTTP PUT následujícím způsobem (2):

```
PUT /users/Robert HTTP/1.1
HOST: myserver
Content-Type: application/xml
<?xml Version="1.0"?>
<user>
      <name>Bob</name>
</user>
```

Použití PUT místo předchozího GET je mnohem čistší, více konzistentní s principy REST a definicemi HTTP metod. PUT metodou uvedenou výše bude upraven systémový zdroj identifikovaný na základě URI, místo původní reprezentace metodou GET a změnou parametrů v těle požadavku. Výsledek tohoto elegantního řešení je, že bude přejmenován systémový zdroj z Roberta na Boba, tím pádem se změní URI z „/users/Robert“ na „/users/Bob“. Při dotazu na systémový prostředek pod původní URI bude vygenerována standardní chyba *404 Not Found*. (2)

Je vhodné dodržovat jeden z hlavních principů návrhu REST s využitím HTTP metod – používání podstatných jmen v URI místo sloves. Slovesa jsou v RESTful webových službách dána – GET, POST, PUT a DELTE. Stejně tak je zbytečné pojmenovávat operace v rámci URI – „/adduser“ nebo „/updateuser“, protože se jedná o entitu „/user“ a prováděnou operaci určuje jedna z HTTP metod, což je v souladu s REST návrhem. (2)

3.2.3 Bezstavost

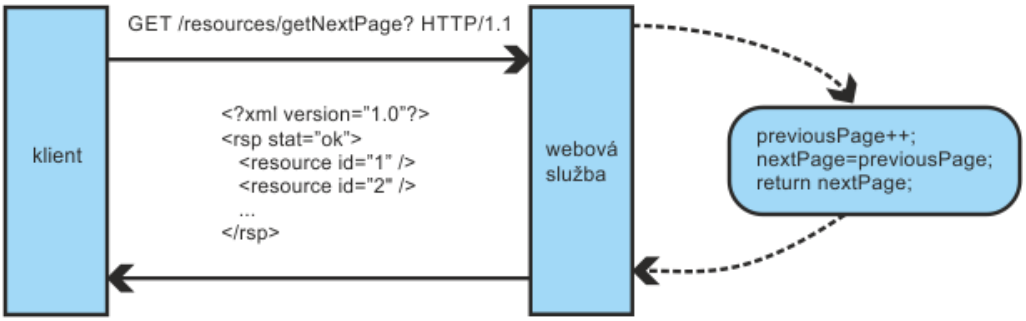
Webové služby postavené na REST musí být škálovatelné, aby byly schopny plnit nároky na zvyšující se výkon. Klastry serverů s *load-balancing* a *failover* funkcemi obsahující *proxy* a *gateway*, jsou typicky sestaveny do topologie, která umožňuje předávání dotazů z jednoho serveru na druhý, aby bylo možné snížit dobu odpovědi na webovou službu. (2)

Užití serverů, zprostředkujících komunikaci ke zlepšení škálovatelnosti, vyžaduje po klientech, komunikujících pomocí REST na webové službě, zasílat celé a nezávislé dotazy, aby bylo možné předávat celé zprávy přímo a nedržet je v různých stavech na několika místech. Kompletní a nezávislý dotaz nepotřebuje server pro získání jakékoliv části aplikačního kontextu nebo stavu. Webová služba postavená na REST, nebo klient této služby, má v HTTP hlavičce a tělu dotazu všechny parametry, kontext a data nutná pro vygenerování odpovědi na straně serveru. Bezstavost v tomto případě zlepšuje výkon webových služeb, zjednodušuje návrh a implementaci komponent na straně serveru, protože absence stavů na serveru odstraňuje potřebu synchronizovat relaci dat v externí aplikaci. (2)

3.2.3.1 Stavový návrh

Obrázek 2 ilustruje stavový návrh, ze kterého si klientská aplikace vyžádá další stránku, přičemž si služba musí uchovávat záznam, na které stránce byla klientská aplikace

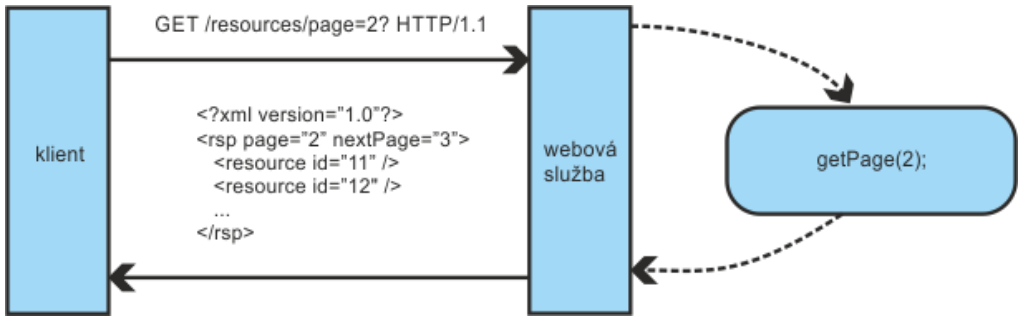
naposledy. V tomto návrhu si webová služba zvyšuje a ukládá hodnotu proměnné pro číslo předchozí stránky, aby byla schopna na dotaz vždy relevantně odpovědět. (2)



Obrázek 2: Návrh stavové webové služby (2)

3.2.3.2 Bezstavový návrh

Bezstavová webová služba na obrázku 3 vygeneruje pro klienta odpověď obsahující číslo následující stránky z celé množiny stránek. Klient si tuto informací uloží, aby se opět mohl dotázat webové služby na detail následující stránky (2).



Obrázek 3: Bezstavový návrh (2)

Konkrétní návrh RESTful webové služby je logicky rozdělen na dva přístupy – server a klient. (2)

Server

Server generuje odpověď obsahující odkazy na ostatní zdroje, aby se mezi nimi mohla aplikace pohybovat. Pokud je dotaz směřován na zdroj nějakého rodiče, typická RESTful odpověď obsahuje potomky rodiče, případně podřízené zdroje k nadřízenému zdroji, aby byla mezi nimi zachována vazba. (2)

Server odesílá odpovědi, které sami sebe označují, kdy jsou ukládány do *cache* a kdy nikoliv za účelem zvýšení výkonu, kterého je dosaženo pomocí redukce počtu dotazů na duplicitní zdroje. Server tuto funkčnost provádí na základě hlavičky HTTP odpovědi a jejích dvou prvků, kterými jsou *Cache-Control* a *Last-Modified*. (2)

Klient

Klient využívá *Cache-Control* informaci umístěnou v hlavičce odpovědi serveru, aby věděl, kdy má a nemá dočasně ukládat informaci o zdroji. Klient dále čte z hlavičky odpovědi informaci *Last-Modified*. Klient může poslat v hlavičce dotazu *If-Modified-Since*, do které uloží datum, ke kterému se ptá, zdali se dotazovaný zdroj změnil. Tomuto způsobu se říká podmíněný GET. Server odpoví kódem *304 Not Modified*, pokud se zdroj od dotazované doby nezměnil. Kódem *304 Not Modified* také odpoví, pokud klient může bez problému využít uložený zdroj jako nejaktuálnější a obcházet následné požadavky GET až do změny zdroje. (2)

Klient zasílá úplný dotaz, který může být obsloužen nezávisle na ostatních dotazech. To je možné, pokud klient plně využívá HTTP hlaviček specifikovaných rozhraním webové služby a pokud zasílá úplnou adresu zdrojů v těle dotazu. Klient posílá dotazy, které mají minimální povědomí o tom, jaké byly předchozí dotazy, zda existuje session na serveru, jestli server umí přiřadit kontext k dotazu, nebo o stavu celé aplikace. (2)

Výše zmíněná spolupráce mezi klientskou aplikací a webovou službou je nezbytná proto, aby služba byla skutečně RESTful. To zlepšuje výkon díky úspoře šířky pásma a minimalizace stavů aplikace na straně serveru. (2)

3.2.4 Výstavba URI podle adresářové struktury

Z hlediska klientské aplikace a adresování zdrojů URI určují, jak intuitivní budou RESTful webové služby, neboť třetí charakteristika RESTful webových služeb je URI (Uniform Resource Identifier). (2)

URI adresy RESTful webové služby by měly být intuitivní. Vymýšlení návrhu URI adresace by mělo směřovat k jasnému rozhraní, které nepotřebuje příliš vysvětlování a ani dokumentace. Na konci celého procesu návrhu URI by měla struktura být přímočará, předvídatelná a snadno pochopitelná. (2)

Jedním směrem, jak dosáhnout výše popsané použitelnosti, je definovat URI strukturu adresářově. Tento typ URI má hierarchii, kořen v jedné cestě a z větví vedou dílčí cesty. Dle této definice není URI pouze řetězcem odděleným znaky „/“, ale spíše strom s podřízenými a nadřízenými větvemi spojené v různých uzlech. (2)

Například pokud se jedná o službu, která shromažďuje témata a diskuze, může být struktura URI definována následovně (2):

```
http://www.myservice.org/discussion/topics/{topic}
```

V této struktuře existuje kořen „/discussion“, který má pod sebou uzel „/topics“. Pod nimi je sada konkrétních témat jako např. technologie, umění apod. Díky této struktuře je jednoduché vyskočit z tématu a skočit do jiného tak, že se za „/topics/“ napíše název jiného tématu. (2)

V některých případech má cesta ke zdroji ještě specifičtější adresářovou strukturu, jako v následujícím případě (2):

```
http://www.myservice.org/discussion/2008/12/10/{topic}
```

První fragment je čtyřmístné číslo značící rok, další fragment je dvoumístné číslo značící den a třetí fragment je dvoumístné číslo značící měsíc. (2)

Lidé a stroje mohou jednoduše sestavovat takovéto URI adresy, protože jsou postaveny na předepsaných pravidlech. Takto se tvoří vzor, podle kterého se budou URI adresy sestavovat (2):

```
http://www.myservice.org/discussion/{year}/{day}/{month}/{topic}
```

Zde jsou některé doporučené postupy, jak přemýšlet nad URI strukturou pro RESTful webovou službu:

- Skrývat přípony skriptovacích souborů na serveru (.jsp, .php, .asp),
- vše psát malými písmeny,
- nahrazovat mezery pomlčkami, či podtržítky,
- vyhýbat se dotazovacím řetězcům,
- místo zobrazování chyby *404 Not Found* vždy zobrazit výchozí stránku s konkrétnější odpovědí. (2)

3.2.5 Formát výstupných dat

Reprezentace zdroje typicky reflektuje jeho aktuální stav pomocí jeho atributů a času, kdy si o něj klient zažádal. Reprezentace zdroje, v tomto smyslu, jsou pouze jeho kopie uložené v různém čase. Je to podobné jako reprezentace záznamu v databázi, která obsahuje mapování mezi názvy sloupců a XML tagů, jejichž hodnoty obsahují hodnoty z řádků tabulky. Nebo pokud má systém datový model, je zdroj reprezentován kopií atributů v tomto datovém modelu. Všechny tyto věci by měla RESTful webová služba obsahovat. (2)

Jediné omezení návrhu RESTful webové služby se týká formátu data. To musí být vytvořeno tak, aby si aplikace a server mohly tyto údaje mezi sebou vyměňovat v dotazu/odpovědi. To je místo, kde se vyplatí ponechat návrh co nejjednodušší. (2)

Objekty v datovém modelu mají mezi sebou vazby, které musí být reflektovány při převodu do klientské aplikace. Pokud se vrátíme ke službě shromažďující témata a diskuze, jsou vazby například mezi kořenem tématu diskuze, jeho atributy a odkazy na odpověď do daného tématu. Následující kus kódu reprezentuje vlákno diskuze ve formátu XML (2):

```
<?xml version="1.0"?>
<discussion date="{date}" topic="{topic}">
  <comment>{comment}</comment>
  <replies>
    <reply from="joe@mail.com" href="/discussion/topics/{topic}/joe"/>
    <reply from="bob@mail.com" href="/discussion/topics/{topic}/bob"/>
  </replies>
</discussion>
```

Poslední věc, kterou klientská aplikace musí umět, je možnost požádat službu o odpověď ve formátu, v jakém potřebuje data obdržet. To se realizuje pomocí HTTP *Accept header*, kde vyplněná hodnota je typu *MIME-Type*. Soupis běžných typů je popsán v následující tabulce.

Mime-Type	Content-Type
JSON	application/json
XML	application/xml
XHTML	application/xhtml+xml

Tabulka 1: Běžné typy obsahu užívané v RESTful webových službách (2)

Díky tomu je možné využít službu různými klienty napsaných v různých jazycích běžících na různých platformách. Využití *MIME-Type* a HTTP *Accept header* je mechanismus, který se nazývá „vyjednávání o obsahu“. Ten nechává na klientovi, ať si vybere, jaký formát dat je pro něj nejvhodnější a minimalizuje tak nutná spojení mezi službou a aplikací. (2)

Vystavení systémových zdrojů pomocí RESTful API je flexibilní způsob, jak poskytnout různým aplikacím data standardním způsobem. To napomáhá k integraci požadavků, které jsou kritické pro vytváření systémů, kde mohou být data jednoduše kombinována, případně rozšiřována do větších celků. (2)

3.3 User Experience

User experience (UX) se zaměřuje na zážitek z užívání produktu (např. mobilní aplikace) a je nedílnou součástí návrhu výsledného produktu. Klíčová myšlenka je, aby se produkt jednoduše ovládal a splňoval potřeby uživatele. Interakce s produktem pro něj musí být intuitivní, užitečná, ale také příjemná. (3)

Dobrý UX návrh přinese radost zákazníkovi z užívání produktu a také bývá klíčovým rozhodovacím prvkem, proč zákazník u produktu zůstane. (3)

3.3.1 Kritéria UX

Správný UX design se hodnotí pomocí tří základních kritérií, které jsou platné napříč celým designovým odvětvím. Tyto míry jsou užitečnost, použitelnost a přitažlivost. (3)

3.3.1.1 Užitečný design

Produkt obsahuje funkce, vlastnosti a rozšíření tak, aby splnil potřeby uživatele. Zážitek z používání produktu musí být ve všech těchto směrech. (3)

3.3.1.2 Použitelný design

Produkt obsahuje funkcionality, které je možné jednoduše a intuitivně použít. Na opravdu nejzákladnější funkcionality produktu uživatel nepotřebuje velkou míru koncentrace. (3)

3.3.1.3 Přitažlivý design

Správný UX design evokuje líbivost a potěšení. Proto produkt neobsahuje jen užitečné funkcionality, ale navíc přidává velice chytrý a chytlavý vzhled, který silně působí na uživatele a jeho vizuální stránku. (3)

Zákazníci se často vrací k danému produktu právě proto, že má jednoduché ovládání, chytlavý vzhled a plní veškeré uživatelské potřeby. (3)

3.3.2 Základní prvky UX

Vytvoření UX design je částečně mezioborové cvičení, které integruje mnoho různých prvků. Tato část vyzdvihne ty hlavní, aby bylo snadnější pochopit, co dělá UX design UX designem. (3)

3.3.2.1 Informační architektura

Jedná se o jednu z nejzákladnějších prvků pro vytvoření UX designu. Zaměřuje se na rozebrání potencionálního produktu do následujících základních částí:

- Navigace (jak se uživatel v aplikaci pohybuje),
- organizace obsahu (kde jsou jaké informace a kam odkazují),
- důležitost vizuálních prvků (kde budou důležité a méně důležité prvky),
- interakce (prvky, se kterými bude uživatel v daných případech interagovat). (3)

Informační architekta často vzniká v souvislosti s vytvářením wireframes.

3.3.2.2 Forma a umístění obsahu

Forma a umístění obsahu určují, jak kdy a kde bude zobrazen libovolný prvek. Prvkem se myslí obrázky, texty, ovládací prvky apod.…). Použitím daného prvku by měl uživatel snadno docílit svého cíle. Každý prvek musí mít své místo a své opodstatnění. (3)

3.3.2.3 Interakce

Během tvorby designu jsou definována pravidla, jak bude uživatel interagovat s produktem. Co se stane, když uživatel zmáčkne konkrétní tlačítko, naviguje se na jinou stránku apod. Definování pravidel, jak se má uživatel „pohybovat“ v rámci produktu, zlepšuje uživatelský zážitek, protože se uživatel napříč produktem přesouvá po logických krocích. (3)

Dnešní chytré telefony umožňují různé typy interakce s dotykovou obrazovkou. Pokud tyto typy budou výrobci mobilních zařízení a mobilních aplikací respektovány, uživatelé je budou používat. Pokud výrobce mobilního zařízení nebo mobilní aplikace začne nutit uživatele měnit své zvyky a měnit typy akce, které interakce vyvolá, uživatel značku chytrého telefonu nebo mobilní aplikaci opustí. (3)

3.4 Wireframes

Wireframes („drátěné modely“) soustředí pozornost celého realizačního týmu na výsledný vzhled produktu. Na těchto modelech lze celkem snadno vidět, jestli je produkt nebo služba prezentována zákazníkovi správně, jak bude strukturován obsah jednotlivých stránek aplikace, která data budou pocházet z jakého zdroje, jak bude daná funkcionalita naprogramována a vizuálně jak bude konkrétní stránka vypadat a kde je třeba vzhled zlepšit.

Při tvorbě wireframes je důležité položit si sadu následujících otázek:

- Jaký má z modelu uživatel pocit?
- Odpovídá konkrétní vzhled potřebám uživatele?
- Jak model vypadá?
- Je model jasný, srozumitelný a dobře strukturovaný?
- Jak model funguje?
- Podporuje technická implementace vzhled a pocit z používání modelu? (4)

Pro uživatele je pocit z užívání důležitější než vzhled, což je ve výsledku důležitější, než do jaké míry aplikace funguje (uživatelé jsou rádi, když věci fungují, nezajímá je, do jaké míry). (4)

3.4.1 Zásady tvorby wireframes

3.4.1.1 Zachování proporcí

Při vytváření wireframes je důležité zachovat velikost modelu oproti skutečnosti v poměru 1:1. Usnadňuje to představu výsledného produktu. (4)

3.4.1.2 Priorita vizuálních prvků

Vizuální priorita jednotlivých prvků je důležitá v každém designu. Pokud by byly wireframes tvořeny pouze bílými boxy s černým ohraničením, nebylo by jasné, která část má jakou prioritu, neboli co je důležitý a méně důležitý prvek. Proto se používají různé odstíny bílé a šedé barvy pro zdůraznění vizuální priority prvků. (4)

3.4.1.3 Použití reálných dat a fotek

Používání reálných dat a fotek v drátěných modelech zachovává skutečnou představu o výsledném produktu. (4)

3.4.1.4 Vysvětlivky funkcionalit

Každý návrh obrazovky aplikace by měl obsahovat svůj seznam funkcionalit a jejich detailní popis. Slouží k lepšímu pochopení navrhovaného řešení. (4)

3.4.1.5 Zobrazení celé stránky

Pokud stránka aplikace bude rolovací, musí být zobrazeny veškeré její části pod sebou přesně tak, jak je uživatel uvidí. (4)

3.4.1.6 Proč se vyplatí dělat wireframes

V každém projektu je využita nějaká forma prototypů, ať už jako kresba na kusu papíru, návrh v Microsoft Visio nebo HTML prototyp. Jejich síla je v přenesení myšlenky do viditelné podoby a možnosti detailně řešit problém, který má budoucí produkt vyřešit.

Celý projektový tým odpovědný za výsledný produkt, by měl být zapojený do procesu návrhu od první skici na papír do finální podoby. V každé části tohoto procesu je třeba

důkladně projít a prodiskutovat stávající stav, navrhnout a zapracovat změny, které jsou následně prezentovány zákazníkovi. Výhoda použití wireframes je rychlost – velice snadno se prvky dají přeskládat bez větší časové a finanční náročnosti. (4)

3.5 Fakturoid

Fakturoid je webová aplikace na vystavování a správu faktur určená zejména pro živnostníky a malé firmy. Velký důraz kladou tvůrci na automatizaci – od vydávání a rozesílání faktur, přes automatické upomínky, až po párování plateb s bankou klienta. Fakturoid umí exporty do rozšířených účetních programů Money S3 a Pohoda a má mobilní aplikace pro iOS a Android. (5)

3.5.1 Služby

Fakturoid poskytuje následující služby:

- Fakturace,
- evidence nákladů,
- statistiky nákladů a příjmů,
- vyplnění daňového přiznání,
- export do účetních systémů,
- párování plateb,
- pravidelné faktury,
- upomínky za nezaplacené faktury. (6)

3.5.2 Fakturoid API

Tato kapitola obsahuje vybrané části API pro Fakturoid. (7)

3.5.2.1 Obecné informace

Požadavek

Požadavky je nutné posílat pomocí HTTPS. Všechny URL adresy začínají:

`https://app.fakturoid.cz/api/v2`. (7)

Pro identifikaci účtu slouží parametr slug (dříve subdoména), který je součástí adresy většiny požadavků. (7)

Identifikace vaší aplikace

Všechny požadavky musí obsahovat hlavičku User-Agent se jménem vaší aplikace a odkazem na ni nebo e-mailovou adresou. Na požadavek bez této hlavičky server vrátí odpověď *400 Bad Request*. (7)

Pouze JSON

API podporuje pouze formát JSON. Všude používejte kódování UTF8. Při použití jiného formátu dostanete ze serveru odpověď *415 Unsupported Media Type*. (7)

Ověření

Ověření se provádí přes HTTP BASIC AUTH, kde jako *username* slouží váš email a za heslo uvedete *api_token* (zjistíte v Nastavení vašeho účtu v záložce Já). Při zadání špatných přihlašovacích údajů dostanete ze serveru odpověď *401 Unauthorized*. (7)

HTTP cache

Pro maximální výkon API i vaší aplikace používejte HTTP cache. Všechny GET akce (kromě reportů) vrací hlavičku ETag a akce, které nejsou nad kolekcemi dat vrací i hlavičku *Last-Modified*. Při prvním dotazu si tyto hodnoty uložte a při dalších dotazech používejte hlavičky *If-None-Match* a *If-Modified-Since*. Pokud se hodnota zdroje od posledního dotazu nezměnila, dostanete zpět odpověď *304 Not Modified*, která ušetří čas vynaložený na zpracování dat na serveru a stahování dat, která už máte. (7)

3.5.2.2 Příklady API volání

Získání seznamu faktur

Seznam faktur je možné získat pomocí metody GET na následující adrese (7):

```
https://app.fakturoid.cz/api/v2/accounts/{slug}/invoices/regular.json
```

Vytvoření nové faktury

Vytvoření nové faktury pomocí metody POST na následující adrese (7):

```
https://app.fakturoid.cz/api/v2/accounts/{slug}/invoices.json
```

Hlavička dotazu obsahuje (7):

```
Content-Type: application/json
```

```
User-Agent: YourApp(yourname@example.com)
```

Tělo dotazu obsahuje následující JSON (7):

```
{
  "subject_id": "28",
  "number": "2012-0001",
  "currency": "CZK",
  "payment_method": "bank",
  "due": 10,
  "issued_on": "2012-03-30",
  "taxable_fulfillment_due": "2012-03-30",
  "bank_account_id": 7,
  "lines": [
    {
      "name": "Hard work",
      "quantity": "1.0",
      "unit_name": "h",
      "unit_price": "40000",
      "vat_rate": "21"
    }
  ]
}
```

4 Windows Phone 8

4.1 User Interface ve Windows Phone

User Interface (uživatelské rozhraní) obsahuje následující základní prvky:

- Rozložení prvků,
- ovládací prvky,
- šablony,
- text a písmo,
- aplikační lišta. (8)

4.1.1 Rozložení prvků

Rozložení prvků je proces přizpůsobení velikosti a umístění vizuálního objektu v aplikaci pro Windows Phone. (9)

Vizuální objekty musí být umístěny do tzv. kontejnerových prvků, ke kterým patří například Canvas, Stack Panel a Grid. Umístěním objektu do zmíněných prvků umožňuje jeho snadnou manipulaci a přizpůsobení obsahu. (9)

Systém rozložení prvků ve Windows Phone podporuje jak dynamické, tak absolutní zarovnání. Při absolutním zarovnání má prvek pevně danou pozic. Při dynamickém zarovnání je pozice prvku vztažená ke svému okolí, tudíž i k rozlišení obrazovky. (9)

4.1.1.1 Absolutní zarovnání prvků

Prvky se vkládají do sebe a drží si pozici pouze v rámci svého nadřazeného (rodičovského) prvku. Při takovém umístění prvku není bráno v potaz rozlišení obrazovky. Proto je třeba udělat pro každé rozlišení nový layout. (9)

Windows Phone má prvek, který se nazývá Canvas, a do něj je možné umisťovat prvky absolutně. (9)

4.1.1.2 Dynamické zarovnání prvků

Prvky se umisťují do svých nadřazených prvků a určuje se, jak se vůči nim bude daný prvek chovat. Díky tomu se vždy prvky přizpůsobí každému rozlišení obrazovky a sami se zarovnají. (9)

Tyto prvky je možné vložit do prvků Grid a Stack Panel a zachovat tím automatické rozmístění a zarovnání. (9)

4.1.1.3 Canvas

Canvas je oblast, do které se umisťují prvky na základě souřadnice x a y. (8)

4.1.1.4 Grid

Nejflexibilnější zarovnání, které podporuje vkládání prvků do více řádků a sloupců. Těm je možné nastavit automatickou velikost, nebo poměrnou velikost vůči celému obsahu. (8)

4.1.1.5 Stack Panel

Jednoduché zarovnání, kde jsou umístěny prvky do řady za sebe nebo pod sebou (ve výchozím nastavení). Nejčastěji se využívá k seřazení vizuálních prvků pod sebe. (8)

4.1.2 Ovládací prvky

4.1.2.1 Navigační prvky

Aplikace Windows Phone jsou postaveny na modelu stránek - uživatel prochází stránky s různým obsahem. Tento model je založen na rámci, který obsahuje ovládací prvky pro navigaci mezi stránkami. (8)

Obrázek 4: Model stránek

4.1.3 Obsahové ovládací prvky

Ovládací prvky, které obsahují vnořené ovládací prvky, jsou často označovány jako obsahové ovládací prvky. Ty jsou kontejnerem pro ostatní ovládací prvky a vizuální objekty. Jak název napovídá, jsou využívány k umístění obsahových prvků na obrazovku. Obsahové ovládací prvky také slouží jako kořen aplikace v rámci každé stránky. Všechny ostatní objekty v UI jsou zahrnuty v tomto kořeni. (8)

Při vytvoření nové Windows Phone aplikace je automaticky vytvořen Grid, který plní funkci kořenu. Takový kořen dále obsahuje název stránky a obsah celé stránky. (8)

Většina obsahových ovládacích prvků je odvozena od třídy Panel, např. Stack Panel, Canvas a Grid. (8)

Dále jsou zde navíc dva další ovládací prvky – Panorama a Pivot. Tyto ovládací prvky jsou odlišné od ostatních, protože by měly obsahovat jenom malé množství položek. Tyto ovládací prvky umožňují uživateli přejetím po displeji zprava do leva přesun mezi položkami. (8)

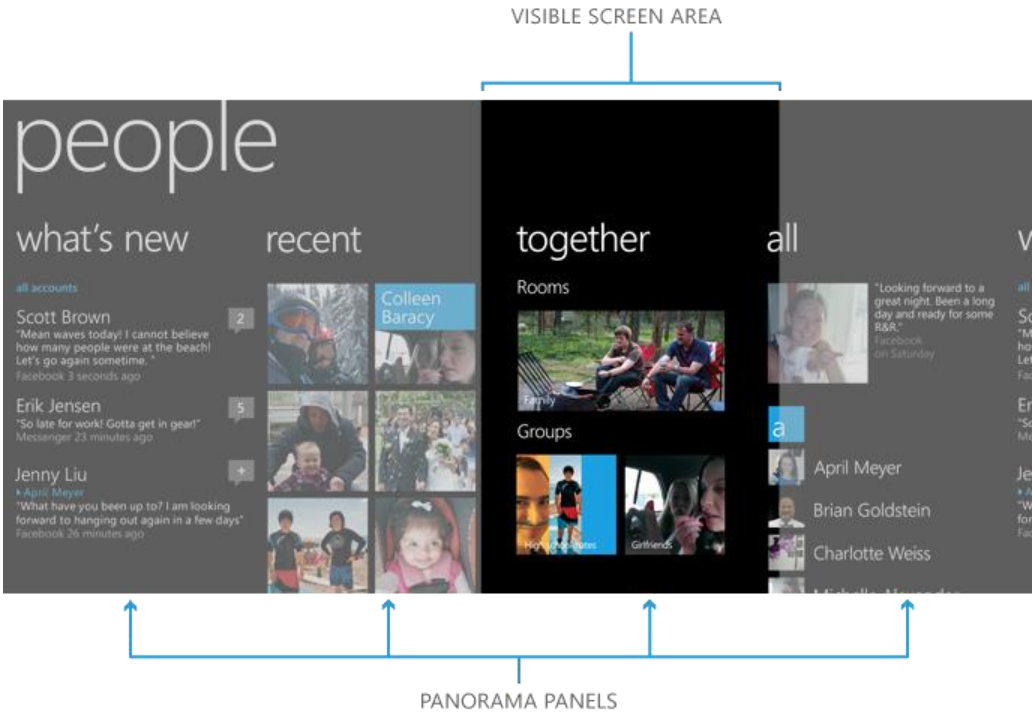
4.1.3.1 Panorama

Panorama je jeden ze základních rysů Windows Phone aplikací. Zatímco standardní aplikace na jiných platformách jsou pevně vztažené k obrazovce telefonu, panoramatická aplikace nabízí unikátní pohled na ovládací prvky, data a služby za pomocí dlouhého horizontálního plátna, které přesahuje hranice obrazovky telefonu. Tyto pohledy využívají dynamického pohybu tak, aby přechod mezi nimi, při listování zprava do leva, byl dostatečně přirozený. (10)

Jak již bylo zmíněno, základem Panorama je dlouhé horizontální plátno. To je rozděleno na PanoramaItems, které v sobě obsahují veškerý viditelný obsah a různé další ovládací prvky. (10)

Obsah v Panorma by měl být následující:

- Možnost procházet prvky v libovolném pořadí bez ohledu na to, zda zprava či zleva,
- strukturovaně rozmisťovat na jednotlivé PanoramaItems obsah, aby byl vždy na každé položce jednotný. (10)



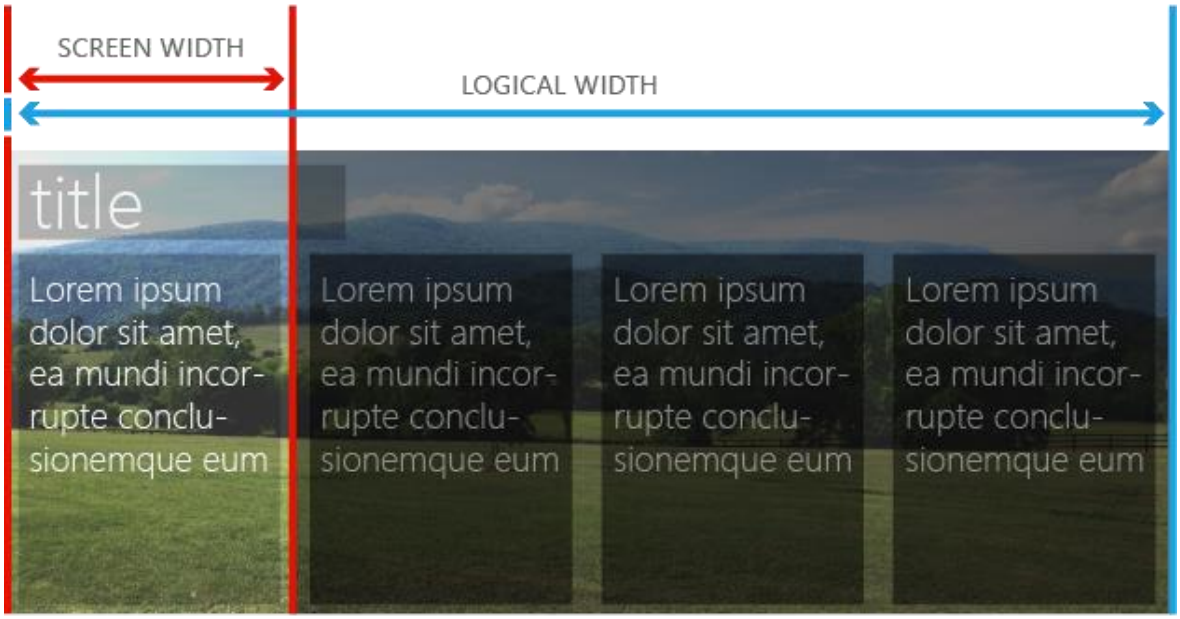
Obrázek 5: Aplikace Lidé ukazuje vzhled a chování Panorama (10)

4.1.3.2 Navigace

Panorama má zabudovanou veškerou podporu pro interakci s dotykem a navigací. (10)

Panorama nabízí následující efekty a gesta:

- Tažení z levé do pravé strany,
- přidržení a rychlý pohyb mezi jednotlivými prvky. (10)



Obrázek 6: Logika pohybu v Panorama 1 (10)

Obrázek 7: Logika pohybu v Panorama 2 (10)

4.1.3.3 Design guidelines

- Pokud používáte aplikační lištu, nastavte její zobrazení na minimalizovaný
- použijte maximálně pět podstránek,

- při načítání můžete zobrazit stav operace v systémové oblasti, nebo zobrazit na aktuálním kontejneru, první možnost neblokuje obsah kontejneru,
- nepoužívejte další typ stránky v kontejneru,
- nepoužívejte „jezdící“ prvky zprava do leva a naopak na kontejnerech, ruší to koncept ovládání stránky. (10)

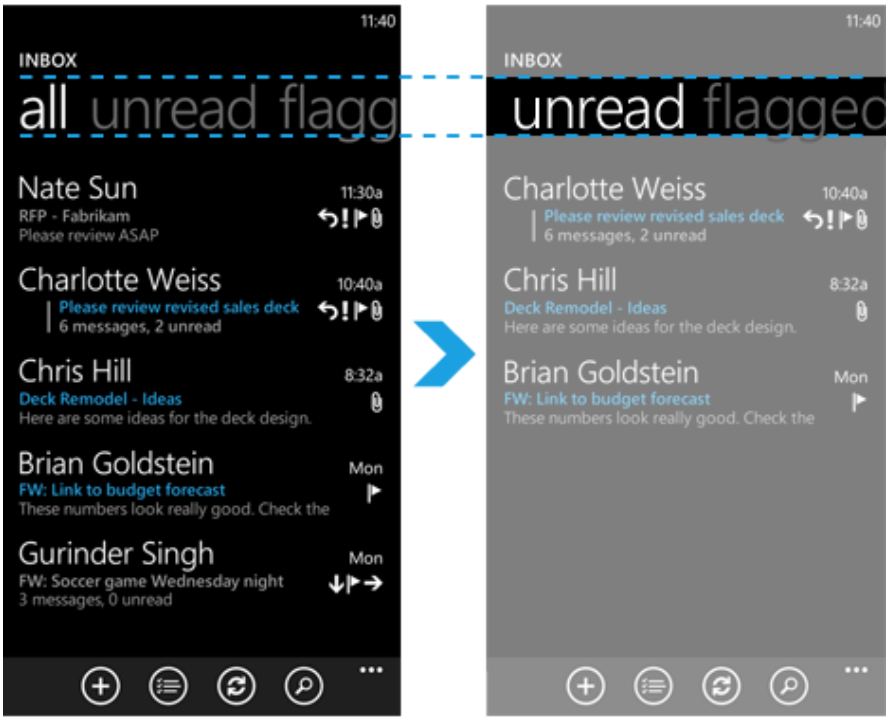
4.1.3.4 Best practices

- Ne vždy je nutné mít na stránce pozadí,
- při použití pozadí na stránku je lepší využít fotografie,
- prvky na stránce by měly splňovat jedno barevné téma,
- šetrně volit název stránky, aby zůstala nadále čitelná a nezasahovala do horní lišty telefonu (systémová oblast),
- držet počet prvků na stránce co nejmenší – více prvků začne obsah činit nepřehledným,
- pozadí stránky vybírejte tmavé, světlé, nebo s nízkým kontrastem,
- pokud chcete umístit na pozadí stránky obrázek, je lepší, aby byl jeden než několik obrázků vedle sebe,
- ideální rozměr pozadí je 480 x 800 - 1024 x 800 pixelů (šířka x výška). (10)

4.1.3.5 Pivot

Pivot poskytuje rychlou možnost přepínat mezi pohledy nebo stránkami. Používá se pro filtrování velkého množství dat, pohledů na data, nebo pohledů v rámci aplikace. (11)

Pivot obsahuje PivotItems. Ty v sobě mají celý obsah viditelné stránky, jako jsou texty a různé další prvky. (11)



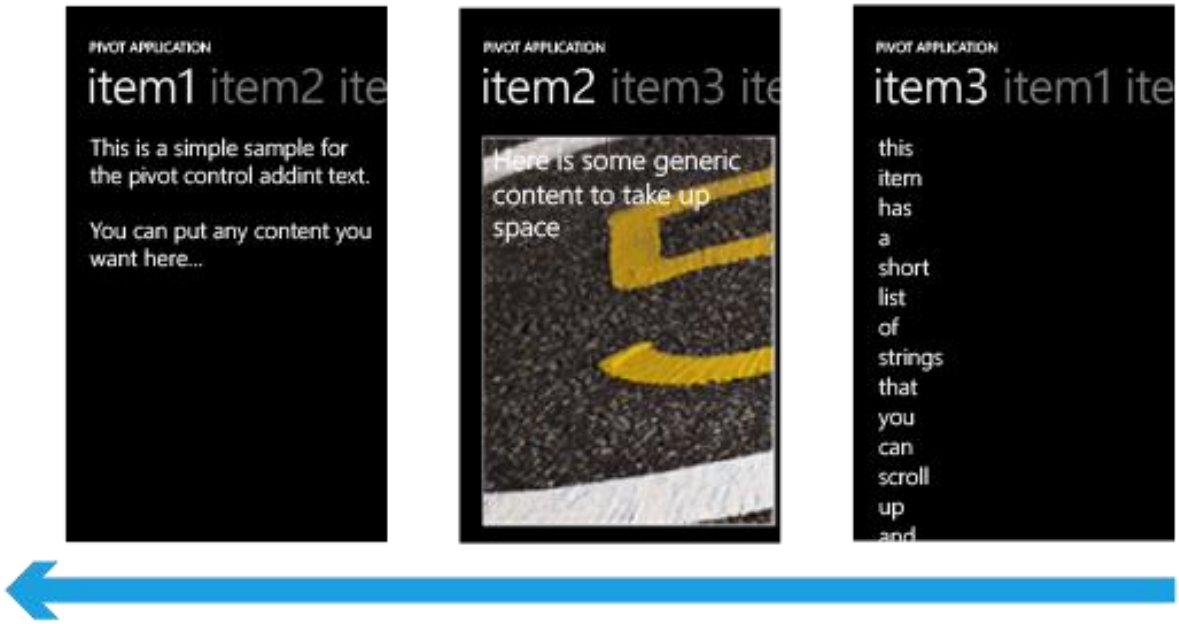
Obrázek 8: Přechod mezi PivotItems (11)

4.1.3.6 Navigace

Pivot má zabudovanou veškerou podporu pro interakci s dotykem a navigací. (11)

Pivot nabízí následující efekty a gesta:

- Tažení z levé do pravé strany,
- přidržení a rychlý pohyb mezi jednotlivými prvky,
- navigace mezi prvky pomocí dotyku nadpisu jednotlivého prvku. (11)



Obrázek 9: Navigace v aplikaci s Pivot (11)

Design guidelines

- Nevkládat Pivot do dalšího Pivotu,
- používat tam, kde je skutečně potřeba,
- používat v aplikacích pro data stejného typu (např. používat pro výčet detailů u každé položky seznamu),
- používat pro přirozený tok událostí (např. jednotlivé kroky registrace uživatele),
- pokud by obsah Pivotu měl být prázdný, je lepší o tom uživatele informovat formou krátkého textu, než celý kontejner odstranit,
- výška hlavičky má danou výšku a je neměnná,
- název aktuálního kontejneru v hlavičce by měl být sestaven z maximálně dvou slov a pokud možno tak, aby bylo zachováno přívětivé uživatelské prostředí,
- nepoužívat formulářové prvky s jezdícím efektem a ani mapy, je to matoucí pro uživatele,
- nepoužívat v kontejneru editor (např. obrázků). (11)

4.1.3.7 Dlaždice

Dlaždice umožňují prezentovat určitý obsah aplikace na domovské stránce Start, i když aplikace zrovna neběží. (12)

Obrázek 10: Příklad různých druhů dlaždic i s číslovkami (12)

Dlaždice mohou být živé (aktualizované pomocí notifikací), nebo mohou být statické. Svůj základní vzhled mají definovaný v manifestu aplikace. Statická dlaždice vždy zobrazí základní vzhled, kterým nejčastěji bývá logo aplikace. Živá dlaždice může aktualizovat tento vzhled a zobrazovat tak nový obsah a zároveň se může vždy vrátit do původního vzhledu. Každá dlaždice má vyhrazené místo, kde může dynamicky zobrazovat číslo vztažené k dané aplikaci. (12)

Dlaždice na domovské stránce Start může mít celkem tři velikosti – malá, střední a širokoúhlá. (12)

Důležité poznámky:

- Velikost dlaždice může vždy uživatel na domácí stránce Start změnit,
- každá aplikace by měla umožňovat vypnout nebo zapnout živé dlaždice. (12)

4.1.3.8 Principy návrhu dlaždice

Hlavním cílem je vytvořit co nejpůsobivější dlaždici. Pokud se jedná o živou dlaždici, je třeba vytvořit hodnotný obsah, který bude zobrazen uživateli na domácí stránce Start. Díky tomu bude uživatel v pokušení aplikaci neustále spouštět. (12)

Důležité je vyhnout se použití moc výrazných barev. Dlaždice by měla být jednoduchá, elegantní. (12)

Při navrhování dlaždic je třeba zamyslet se nad následujícími body:

- Dlaždice jsou „vchodové dveře“ do aplikace, poutavá dlaždice může zaujmout uživatele a zvýšit tak její používání,
- živá dlaždice může lépe prodat aplikaci než statická,
- uživatelé dávají přednost aplikacím s živými dlaždicemi před statickými, více je baví,
- pokud se uživatelům živá dlaždice líbí, je umístěna na důležitém místě na domácí stránce Start,
- pokud se uživatelům živá dlaždice nelíbí, je umístěna až na konci domácí stránky Start, nebo rovnou odepnuta,
- některé vlastnosti živých dlaždic, které je dělají zajímavějšími: jednoduchost, pravidelná aktualizace obsahu,
- poskytování aktuálních informací týkajících se aktuálního uživatele vždy přispěje k jeho spokojenosti. (12)

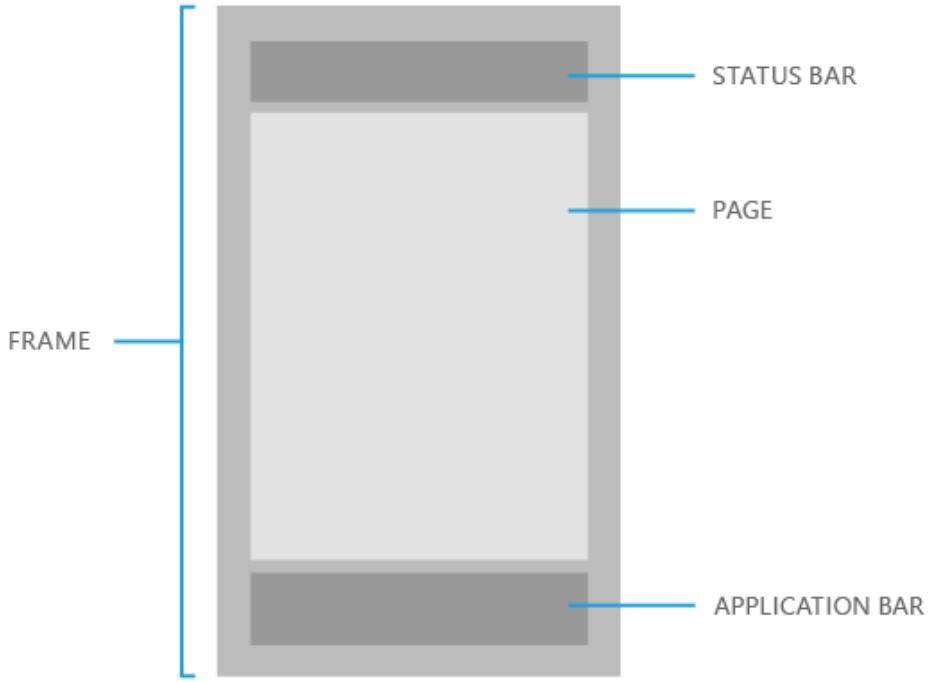
Obrázek 11:Živé dlaždice s aktuálním obsahem ve všech velikostech (12)

Obrázek 12: Statické dlaždice ve všech velikostech (12)

4.2 Navigace v aplikacích pro Windows Phone

Ve Windows Phone aplikacích se uživatel pohybuje mezi stránkami s různým obsahem. Dále se také může vracet pomocí hardwarového tlačítka Zpět. (13)

Dodržením těchto standardů je nespornou výhodou pro uživatele – jednou si zvykne ovládat aplikace ve Windows Phone a to bude neměnné. (13)



Obrázek 13: Navigace v aplikaci (13)

4.2.1 Rámec

Rámec je vizuálním základem aplikace. Má následující charakteristiky (8):

- Na základě nastavení stránky zvolí orientaci na šířku nebo výšku,
- stará se o správné vykreslení stránek,
- stará se o navigaci a přechod mezi stránkami,
- automaticky vypočítává místo pro stavovou a aplikační lištu.

4.2.2 Stránka

Vyplňuje rámeček s ohledem na stavovou lištu a obsahuje obsah stránky. (8)

4.2.3 Systémová lišta

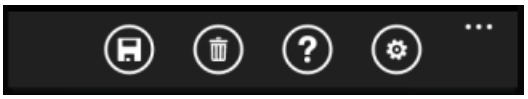
Systémová lišta zobrazuje systémové informace v jednoduchém provedení umístěné v horní části obrazovky (viz. Obrázek 13). Uživatel na něm může sledovat aktualizace, různé notifikace a důležitá oznámení. (8)

Systémová lišta má následující charakteristiky (8):

- Může být nastaven jako viditelný, nebo neviditelný,
- může mít nastavenu průhlednost,
- může mít nastavenu barvu pozadí a barvu písma,
- může obsahovat stav aktuální probíhající akce.

4.2.4 Aplikační lišta

Obsahuje hlavní akce v rámci dané stránky zobrazené pomocí ikon. Méně důležité akce jsou v jednoduchém menu pod ikonami. (8)



Obrázek 14: Aplikační lišta (8)

Aplikační lišta má následující charakteristiky (8):

- Může být nastaven jako viditelný, nebo neviditelný,
- může mít nastavenou průhlednost,
- může mít nastavenu barvu pozadí a barvu písma.

4.2.4.1 Best practices

- Rozměry výsledných ikon s kruhem jsou 48 x 48 pixelů,
- ikona obrázku použitá do kruhu by měla mát velikost 26 x 26 pixelů, aby nedošlo k jeho přesahu,
- kruh je vykreslen automaticky a neměl by být součást ikony obrázku,
- ikona obrázku by měla mít průhledné pozadí a bílé popředí – Windows Phone si je automaticky obarví na základě barevného schématu (světlé, nebo tmavé),

- nevytvářet tlačítko, které by mělo simulovat tlačítko zpět,
- pojmenovat ikony co nejkratším názvem je třeba,
- vybírat ikony tak, aby byl jasný jejich význam. (8)

4.2.5 Tlačítko Zpět

Hardwarové tlačítko zpět je využito k ukončení aplikace nebo k návratu na předchozí stránku v rámci aplikace. Dále je možné tímto tlačítkem schovat otevřenou klávesnici na obrazovce, dialogy nebo menu. (8)

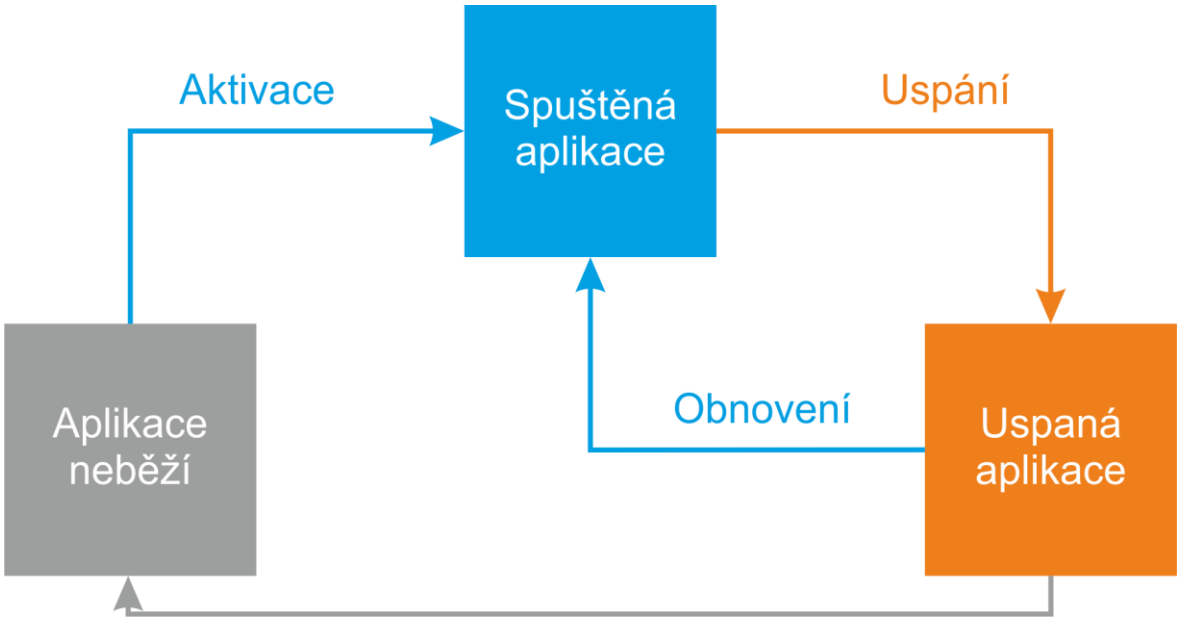
Při jeho delším přidržení je možné přepnout do jiné, již otevřené aplikace, nebo na přehled spuštěných aplikací na pozadí. (8)

4.3 Vývoj aplikací pro Windows Phone

4.3.1 Životní cyklus aplikace

Následující část popisuje životní cyklus aplikace od jejího spuštění po její ukončení. Správným ošetřením stavu uspání a probuzení aplikace je zaručena uživatelská přívětivost vyvíjené aplikace. (14)

Následující obrázek popisuje stavy životního cyklu aplikace: (14)



Obrázek 15: Stavy životního cyklu aplikace (14)

4.3.1.1 Spuštěná aplikace

Aplikace je spuštěna, pokud je aktivována uživatelem a je ve stavu „Aplikace neběží“. V tomto stavu může být aplikace, protože nebyla ještě spuštěna, byla spuštěna, ale skončila s chybou, byla řádně ukončena, nebo byla držena v paměti a nakonec ukončena. (14)

Pokud se aplikace spouští, operační systém zobrazí *splash screen* dané aplikace. Hlavními úkoly aplikace během spuštění jsou registrovat event handlery a nastavit základní UI. Tyto úkoly by měly trvat pouze pár sekund. Pokud aplikace potřebuje stáhnout informace z internetu nebo zpracovat velké množství dat, měly by být tyto operace prováděny mimo aktivaci aplikace. Aplikace může použít předpřipravený *splash screen*, nebo předpřipravený. (14)

Jakmile je hotova aktivace aplikace, změní se stav na „Spuštěná aplikace“ a *splash screen* zmizí. (14)

Aplikace může být dále aktivována např. pomocí hlasového příkazu, operací s peněženkou systému Windows Phone, nebo si ji vynutil systém na otevření nějakého souboru. (14)

4.3.1.2 Uspaná aplikace

Aplikace může být uspána, pokud se z ní uživatel přepne do jiné aplikace, nebo se telefon dostane do stavu slabé baterie. Většina aplikací se zastaví, pokud se z nich uživatel přepne jinam. Pokud uživatel takto přepne aplikaci na pozadí, operační systém čeká několik sekund, jestli se do ní uživatel hned nevrátí. Pokud se nevrátí, je aplikace uspaná. (14)

V aplikaci je možné připravit se na to, že aplikace bude uspána. Pokud je detekován příkaz k uspání aplikace od operačního systému, je možné v této fázi uložit nejdůležitější data do persistentního úložiště. Dále je možné, aby si aplikace uložila svůj stav a uvolnila některé své zdroje. Pokud aplikace nereaguje po dobu maximálně pěti sekund na žádost operačního systému o probuzení, operační systém se rozhodne aplikaci ukončit. (14)

Operační systém se snaží držet si co nejvíce uspaných aplikací v paměti. Díky tomu má uživatel zaručeno, že mezi nimi může plynule přecházet. Pokud operační systém nemá na provedení nějaké operace dostatek prostředků, jednoduše ukončí nějakou aplikaci, kterou si držel v paměti. Aplikace o tomto ukončení nedostane žádnou zprávu, proto je výhodné ukládat důležitá data těsně před uspáním aplikace. Tento stav je aplikaci oznámen, proto jej lze snadno zachytit a data tak ochránit.

4.3.1.3 Pokračování běhu aplikace

Uspaná aplikace se pokračuje v běhu, pokud se do ni uživatel vrátí, nebo se zařízení dostane ze stavu nízké baterie. (14)

Pokud se aplikace vrátí ze stavu uspání do stavu běžící, pokračuje ve své činnosti, kde byla uspána. Žádná data nejsou ztracena, pokud byla předem uložena do paměti. Přesto většina aplikací této možnosti nevyužívá. Aplikace může být uspána hodiny, až dny. Takže pokud aplikace používala nějaký speciální obsah, dá se naprogramovat, že při přechodu z uspání do běžícího stavu aplikace automaticky obnoví konkrétní obsah. (14)

4.3.1.4 Ukončení běhu aplikace

Obecně uživatelé nemusí zavírat aplikace, o to se postará operační systém. Zavřít aplikaci ve Windows Phone je možné pomocí přepínače úloh. Součástí UI aplikace nesmí být možnost ukončit aplikaci. Taková aplikace by neprošla certifikací ve Windows Store a nebylo by ji možné publikovat. (14)

Ve Windows Phone neexistuje indikace toho, že byla aplikace ukončena. Jakmile je uživatelem ukončena, je uspána, ukončena a převedena do stavu „Aplikace neběží“ během 10 sekund. (14)

4.3.1.5 Pád aplikace

Mobilní aplikace pro Windows Phone jsou navrženy tak, že pokud dojde k jejímu pádu, uživatel je automaticky přesměrován na domovskou obrazovku telefonu. (14)

Pokud aplikace zažije pád, přestane odpovídat, nebo vygeneruje výjimku, Windows Phone se zeptá, jestli chce uživatel tento problém nahlásit společnosti Microsoft, která pak předává různé statistiky a podrobnější informace o chybách vývojářům konkrétní aplikace. (14)

4.3.2 Programovací a značkovací jazyk

Mobilní aplikace pro Windows Phone se dají vytvářet pomocí tří kombinací jazyků. (15)

Prví možností je značkovací jazyk XAML (Extensible Application Markup Language) a programovací jazyk C#, nebo C++. Druhá možnost je použít značkovací jazky HTML 5 a JavaScript, nebo WinJS. Poslední třetí možností je DirectX a nativní C++, nebo MFC. (15)

Ze tří kombinací programovacích jazyků pro Windows Phone byla vybrána kombinace značkovacího jazyka XAML a programovacího jazyka C#.

Programovací jazyk C# byl vybrán pro svou rychlost, lepší čitelnost kódu a také pro různé silné nástroje, které obsahuje. Jeden z těchto zásadních nástrojů je dotazovací jazyk LINQ, který dokáže provést nad kolekcí nebo seznamem dat ve velice krátkém čase dotazy na bázi SQL jazyka.

XAML je základní stavební kámen pro tvorbu veškerých UI v aplikacích ve Framework .NET, proto bylo na místě ho využít.

4.3.2.1 Co je to XAML

XAML je deklarativní značkovací jazyk. Používá se pro návrh UI v. NET Framework aplikacích. Zásadní rozdíl mezi ostatními značkovacími jazyky je, že je možné pomocí XAML vytvořit UI prvky a ty samé prvky je možné ovládat a ovlivňovat odděleně v logice aplikace. (16)

Soubory s XAML nejsou vlastně nic jiného než XML soubory, pouze mají příponu „.xaml“. Tyto soubory mohou být kódovány nějakým XML kódováním, jako nejčastější kódování se typicky používá UTF-8. (16)

Následující příklad demonstruje využití XAML v praxi: (16)

```
<StackPanel>
  <Button Content="Stiskni"/>
</StackPanel>
```

4.3.2.2 Objekty

Element objekt deklaruje instanci nějakého typu. Tento typ je vždy součástí deklarace. (16)

Deklarace objektu začíná vždy „<“. Dále následuje typ daného objektu, poté atributy a nakonec uzavírací „>“. Pokud objekt v sobě neobsahuje další obsah, je možné ukončit jeho deklaraci pomocí „/>“ (16)

V příkladu popsaném v předchozí kapitole jsou dva objekty. Jeden je typu StackPanel a druhý Button. Objekt typu Button je vložen v objektu typu StackPanel. Navíc je objekt typu Button prázdný – neobsahuje žádný další prvek, je ukončen „/>“. (16)

4.3.2.3 Vlastnosti

Vlastnosti objektu jsou vyjádřeny jeho atributy. Syntaxe atributů obsahuje název atributu, následuje symbol „=“ a nakonec v uvozovkách hodnota atributu. Syntaxe je velice jednoduchá a intuitivní, zejména pro vývojáře zvyklé pracovat se značkovacími jazyky. (16)

Následující příklad demonstruje použití atributů – tlačítko bude mít modré pozadí a červený text: (16)

```
<Button Background="Blue" Foreground="Red" Content="Toto je tlačítko"/>
```

4.3.2.4 Syntaxe vlastnosti elementu

V některých případech je potřeba blíže specifikovat vlastnosti a není možné použít syntaxi atributů popsaných v předchozí kapitole. (16)

Proto je nutné využít podrobnější syntaxi, která začíná „<názevTypu.názevVlastnosti>“ a končící „</názevTypu.názevVlastnosti>“. Následující příklad toto demonstruje: (16)

```
<Button>
  <Button.Background>
    <SolidColorBrush Color="Blue"/>
  </Button.Background>
  <Button.Foreground>
    <SolidColorBrush Color="Red"/>
  </Button.Foreground>
  <Button.Content>
    This is a button
  </Button.Content>
</Button>
```

4.3.2.5 Kolekce

Jazyk XAML obsahuje několik optimalizací, aby byl pro uživatele co nejčitelnější. Jedna z nich je, že uživatel nemusí vytvářet kolekce atributů, ale stačí vytvořit atributy a automaticky jsou zahrnuty do kolekce. (16)

Tato optimalizace je k vidění na následujícím příkladu: (16)

```
<LinearGradientBrush>
  <LinearGradientBrush.GradientStops>
    <GradientStop Offset="0.0" Color="Red" />
    <GradientStop Offset="1.0" Color="Blue" />
  </LinearGradientBrush.GradientStops>
</LinearGradientBrush>
```

4.3.3 Vývojové prostředí (IDE)

Vývojovým prostředím pro mobilní aplikace pro Windows Phone 8 a vyšší je Visual Studio společnosti Microsoft od verze 2013.

Pomocí sady Visual Studio může uživatel vytvářet mnoho různých typů aplikací: Windows Store aplikace, Windows Phone aplikace a Windows Universal aplikace, aplikace pro klasickou plochu, webové aplikace a webové služby. Uživatel může použít jazyky Visual Basic, Visual C#, Visual C++, Visual F# a JavaScript. (17)

4.3.4 Vývojářský účet

Pro publikaci aplikací do Windows Store je nutné mít vývojářský účet. Microsoft nabízí dva typy vývojářských účtů – individuální a firemní. Individuální stojí $19, firemní $99 USD. (18)

5 Vývoj mobilní aplikace

V této části bude zmíněna webová služba, na které bude vyvíjena Windows Phone aplikace. Dále zde bude popsán celý vývoj mobilní aplikace krok za krokem a všechny náležitosti s ním související až po její nahrání do Windows Store.

5.1 Webová služba Fakturoid

Mobilní aplikace pro Windows Phone 8 bude postavena nad službou Fakturoid. Zmíněná služba byla vybrána ze dvou hlavních důvodů – v době vzniku této práce nebyla dostupná klientská aplikace pro Windows Phone 8 a zároveň se jedná o populární službu, se kterou je lákavé spolupracovat.

Tvůrci Fakturoidu byli osloveni emailem s nabídkou bližší spolupráce s autorem této práce. Ti nabídku přijali s ochotou a podmínkou, že vytvořená mobilní aplikace nesmí být vydána pod jejich registrovanou značkou a barvami.

Aktuálně má Fakturoid oficiální mobilní klienty pro operační systémy Android a iOS.

5.2 Cíl mobilní aplikace

Hlavním cílem mobilní aplikace je zjednodušení práce s fakturováním a vytvořit možnost správy faktur v situacích, kdy nemá uživatel k dispozici počítač s připojením k internetu.

5.3 Návrh řešení

Mobilní aplikace bude poskytovat následující služby webové aplikace Fakturoid:

- Vytvoření a editace kontaktů,
- vytvoření, procházení a mazání faktur,
- vytvoření, procházení a mazání nákladů,
- stažení faktury ve formátu PDF,
- zobrazení stavů faktur,
- filtrování podle stavů faktur.

Mobilní aplikace bude komunikovat pomocí REST API webové aplikace Fakturoid. Veškerá data stažena z backendu budou uložena do paměti telefonu pro zobrazení i při režimu offline.

Data vytvořena aplikací mohou být okamžitě odeslána na backend, nebo uložena v mobilním zařízení pro pozdější synchronizaci dat.

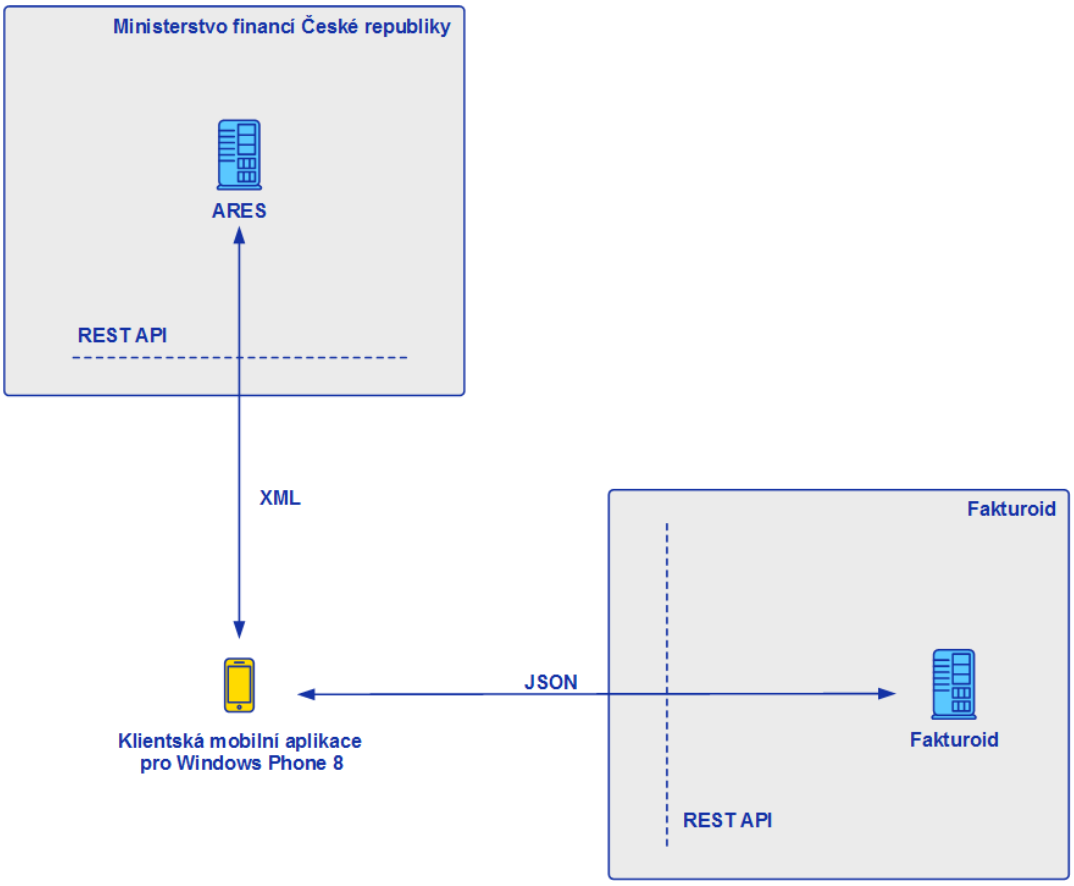
Pro pohodlnější vytváření kontaktů bude aplikace využívat ARES (Administrativní registr ekonomických subjektů). Na základě identifikačního čísla se vyplní základní kontaktní údaje firmy, kterou si uživatel chce přidat.

5.4 Architektura řešení

Mobilní aplikace bude zabezpečeně komunikovat s backendovým systémem webové aplikace Fakturoid pomocí RESTful API rozhraní poskytované českou službou Apiary.io a dále se službou ARES.

Komunikace mezi mobilní aplikací a bakcendem bude probíhat zabezpečeně a výměna dat bude realizována pomocí formátu JSON.

Mobilní aplikace bude dále komunikovat se službou ARES, která na základě zadaného identifikačního čísla vrátí informace o subjektu ve formátu XML.



Obrázek 16: Architektura mobilní aplikace

5.5 Seznam hlavních use case

Po vytvoření zadání mobilní aplikace a návrhu řešení je nutné sepsat hlavní use case napříč celou aplikací. Podle těchto use case se upřesňují funkcionality celé aplikace a na základě toho jsou vytvářeny jednotlivé obrazovky aplikace.

Tabulka 2 obsahuje seznam use case zařazených do kategorií (nadřazených use case).

Název use case	**Kategorie**
Přihlásit	N/A
Odhlásit	N/A
Synchronizovat data	N/A
Vytvořit fakturu	Spravovat faktury
Upravit fakturu	
Vymazat fakturu	
Zobrazit seznam faktur	
Zobrazit detail faktury	
Stáhnout fakturu v PDF	
Nastavit fakturu zaplaceno/nezaplaceno	
Vytvořit kontakt	Spravovat kontakty
Editovat kontakt	
Vymazat kontakt	
Zobrazit seznam kontaktů	
Zobrazit detail kontaktu	

Vytvořit náklad	Spravovat náklady
Upravit náklad	
Vymazat náklad	
Zobrazit seznam nákladů	
Zobrazit detail nákladu	
Zobrazit profil	Spravovat profil

Tabulka 2: Seznam hlavních use case

5.6 Rozložení obrazovek

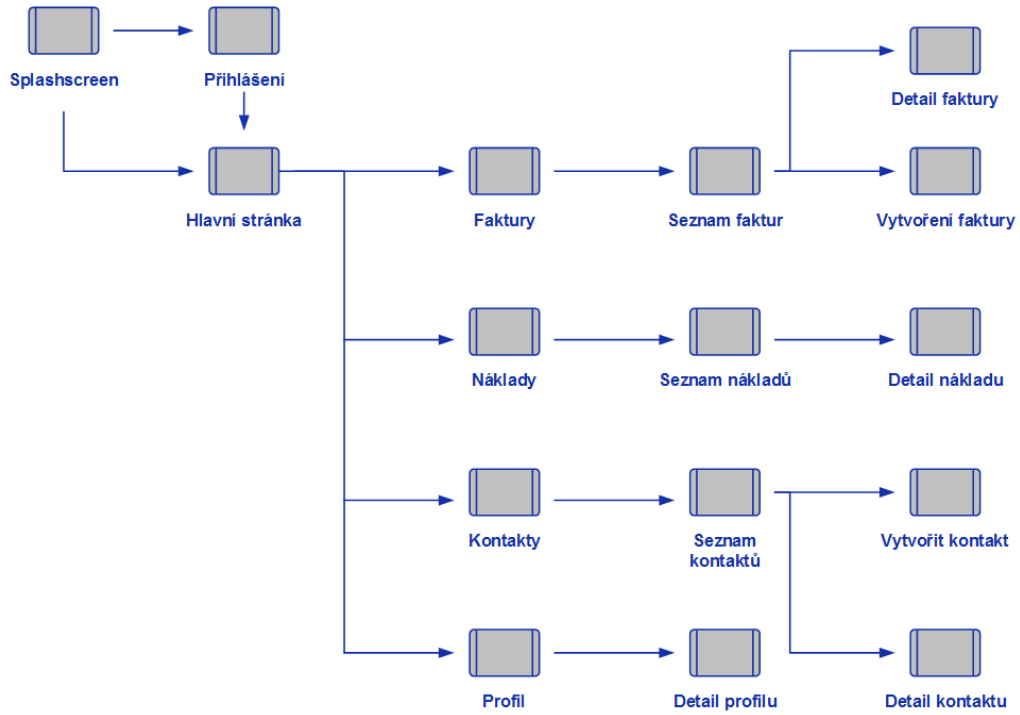
Rozložení obrazovek je jedním z důležitých prvků pro vytvoření kvalitní mobilní aplikace. Promyšlený a intuitivní průchod aplikací zvýší její použitelnost a tím i šance, že ji bude uživatel dlouhodobě používat, doporučí dalším uživatelům a její podíl na trhu poroste.

Autor aplikace, případně analytik, pokud na aplikaci pracuje početnější tým, si sestavuje obrazovky do logických celků.

V případě této aplikace jsou to:

- Splashscreen (úvodní obrazovka s logem před vstupem do aplikace),
- přihlášení (pokud se uživatel už jednou přihlásil, pokračuje na hlavní stránku),
- hlavní stránka,
- faktury,
- náklady,
- kontakty,
- profil.

Dílčí položky faktury, náklady, kontakty a profil obsahují vždy množinu prvků, kde je možné prohlédnout detail prvku nebo vytvořit prvek nový.



Obrázek 17: Rozložení obrazovek (zpracoval autor)

5.7 Návrh User Interface

Prvním krokem k návrhu správného UI mobilní aplikace je vytvoření wireframes. Na základě správné aplikaci zásad UX se wireframes upravují do finální podoby a spokojenosti autora.

V druhém kroku se na základě hotových wireframes začne vytvářet finální design. V něm se ladí skladba barev pro jednotlivé obrazovky a různé kombinace tak, aby nebyla porušena pravidla UX a bylo dosaženo co nejvyššího a nejčistšího vizuálního efektu pro koncového uživatele.

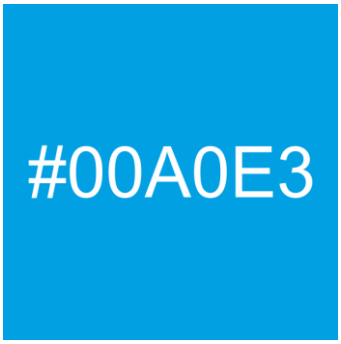
5.7.1 Výběr barev

K výběru barvy produktu vzniklo mnoho studií, založených primárně na psychologii, které potvrdili fakt, že barva produktu hraje výraznou roli při výběru produktu a jeho užívání. Každý typ barvy (např. modrá, zelená, oranžová atd.) má svůj psychologický význam pro uživatele.

Mobilní aplikace usnadňující proces fakturace, zaznamenávání nákladů, zobrazení přehledu zaplacených a nezaplacených faktur, musí působit na uživatele příjemně a důvěryhodně. Zadávání nákladů do aplikace, ať mobilní, webové, či desktopové, nevnímá uživatel

pozitivně – jeho peníze odešly. Pokud dostane do ruky jednoduchý nástroj, který mu zadání nákladu usnadní, bude ho vnímat jako pomocníka a zároveň bude působit vzhledem příjemně, negativní pocit z celého procesu se minimalizuje.

Na základě těchto faktů byla tematickou barvou mobilní aplikace vybrána barva modrá, která působí důvěryhodně a uklidňujícím dojmem. Konkrétně se jedná o následující odstín:



Obrázek 18: Tematická barva aplikace

Doplňkovými barvami v mobilní aplikaci budou tyto:

Obrázek 20: Doplňková barva pro pozadí

Obrázek 19: Doplňková barva pro text

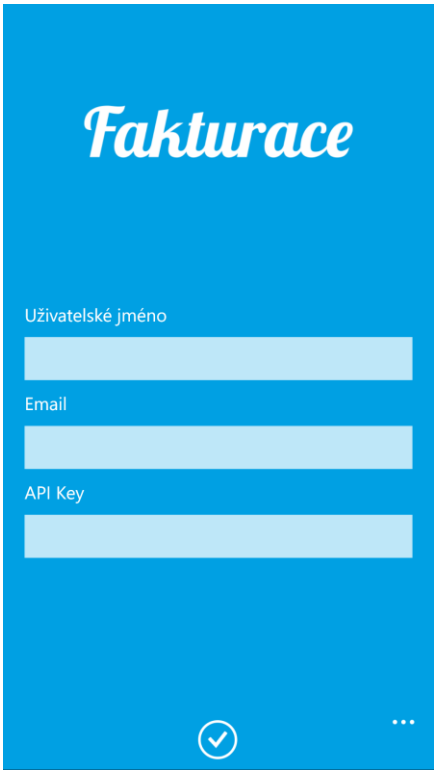
První doplňková barva je navržena na pozadí v aplikaci. Druhá doplňková barva je pro veškerý text, který bude v mobilní aplikaci. Mezi těmito dvěma barvami vznikne dostatečný kontrast, aby bylo písmo čitelné a nebolely z něj oči.

5.7.2 Vybrané obrazovky

Obsahem této sekce je porovnání navrhovaných wireframes a finálního designu vybraných obrazovek.

5.7.2.1 Přihlašovací obrazovka

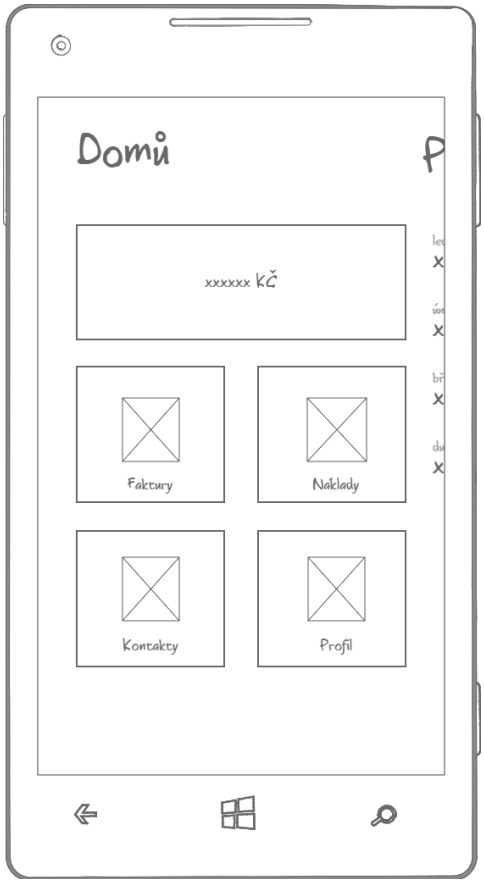
Obrázek 22: Wireframe přihlašovací obrazovky



Obrázek 21: Finální design přihlašovací obrazovky

Přihlašovací obrazovka má tři vstupní pole na zadání nejnutnějších údajů, bez kterých není možné se do aplikace přihlásit. Dále obsahuje potvrzovací tlačítko pro přihlášení a název aplikace.

5.7.2.2 Hlavní stránka s menu



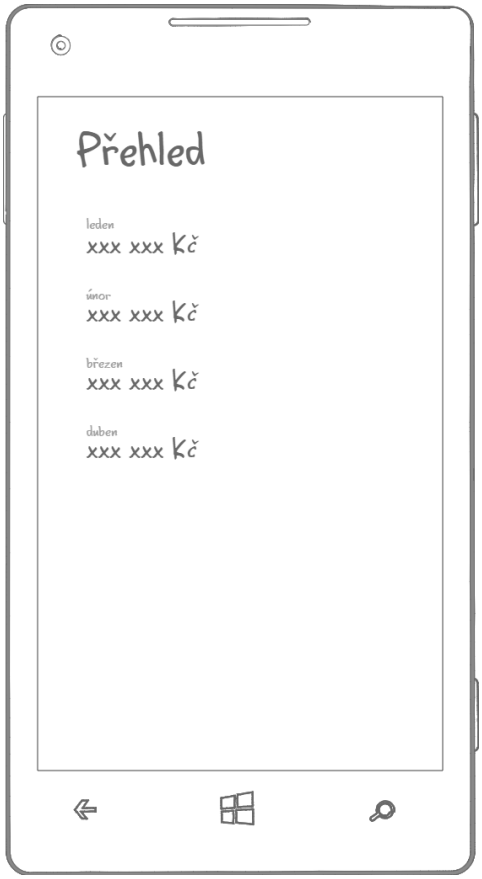
Obrázek 24: Wireframe hlavní stránky s menu

Obrázek 23: Finální design hlavní stránky s menu

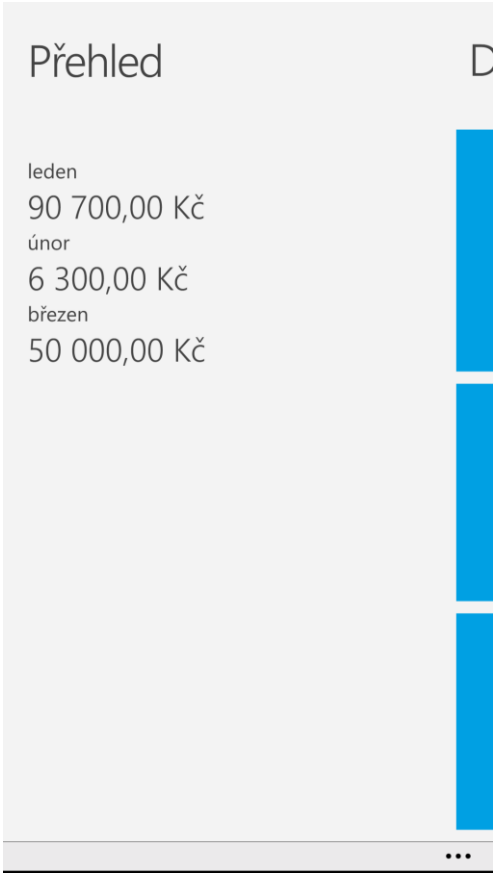
Hlavní stránka s menu ve stylu Panorama obsahuje součet vyfakturovaných částek a následující tile ikony:

- Faktury (ikona s šanony evokující zaevidované faktury),
- náklady (ikona s nákladním autem je zde spíše jako homonymum),
- kontakty (ikona s postavou a vedle něj text),
- profil (ikona samostatné postavy – uživatele aplikace).

5.7.2.3 Hlavní stránka s měsíčním přehledem



Obrázek 25: Wireframe měsíčního přehledu



Obrázek 26: Finální design měsíčního přehledu

Součástí hlavní stránky je stránka obsahující měsíční přehled vyfakturovaných položek za aktuální rok.

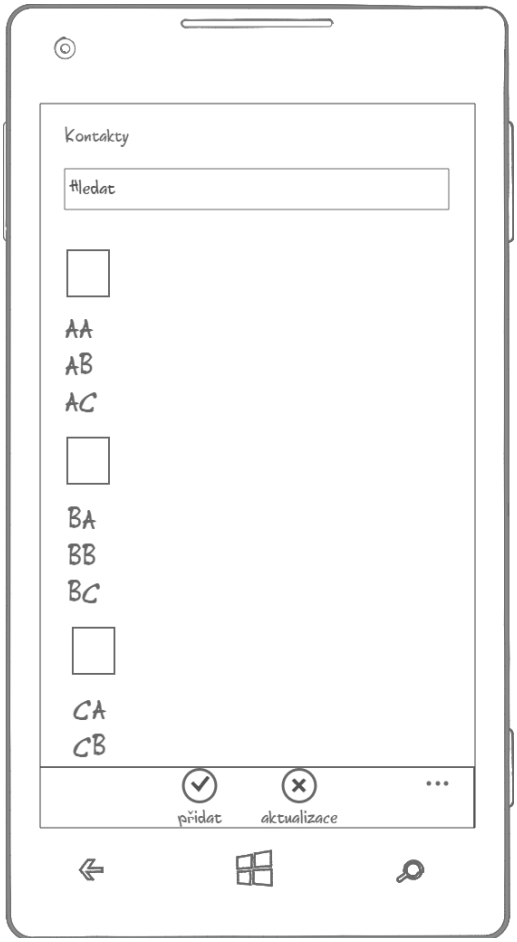
5.7.2.4 Přehled faktur

Obrázek 28: Wireframe přehled faktur

Obrázek 27: Finální design přehled faktur

Přehled faktur je rozdělen do tří základních filtrů ve stylu Pivot – všechny faktury vystavené a po splatnosti. Položka výpisu obsahuje číslo faktury, odběratele a celkovou fakturovanou částku.

5.7.2.5 Přehled kontaktů



Obrázek 29: Wireframe přehled kontaktů

Obrázek 30: Finální design přehled faktur

Přehled kontaktů obsahuje vyhledávácí pole pro vyhledávání mezi kontakty. Kontakty jsou abecedně seřazeny do skupin podle počátečního písmene kontaktu. Tyto skupiny slouží pro lepší orientaci ve dlouhém seznamu. Seznam je možné aktualizovat anebo vytvořit úplně nový kontakt pomocí tlačítek v aplikační liště.

5.8 Návrh dlaždice aplikace

Dlaždice (ikona) mobilní aplikace by měla být jedinečná, aby si ji uživatel nepletl s ostatními aplikacemi a zároveň by měla nějakým způsobem popisovat tematické zaměření mobilní aplikace.

V tomto případě se jedná o mobilní aplikaci zaměřenou na správu faktur a procesů s tím spojených. Vizuálním reprezentantem této aplikace, jak ji vnímá autor, je symbol peněz nebo list papíru.

Následující obrázky ukazují možnosti, jak by dlaždice mohla vypadat:

Obrázek 31: Dlaždice pro aplikaci s motivem měny

Obrázek 32: Dlaždice pro aplikaci s motivem faktury

Obrázek 33: Dlaždice pro aplikaci s motivem faktury 2

Obrázek 34: Dlaždice pro aplikaci s motivem faktury 3

Existuje dostatek aplikací, napříč všemi platformami, zabývající se financemi a jejich správou, které mají v ikoně aplikace symbol měny. Z toho důvodu první dlaždici (obrázek 31) vyřadíme z výběru. Může se totiž stát, že bude tato aplikace úspěšná a bude třeba ji vytvořit i na další platformy, kde však musí mít stejnou ikonu. To by v tomto případě nemuselo být jednoduché. Nyní zbývá druhý symbol pro dlaždici – list papíru. Aby tento list papíru nevypadal obyčejně a zaměnitelně, byl lehce zahnut.

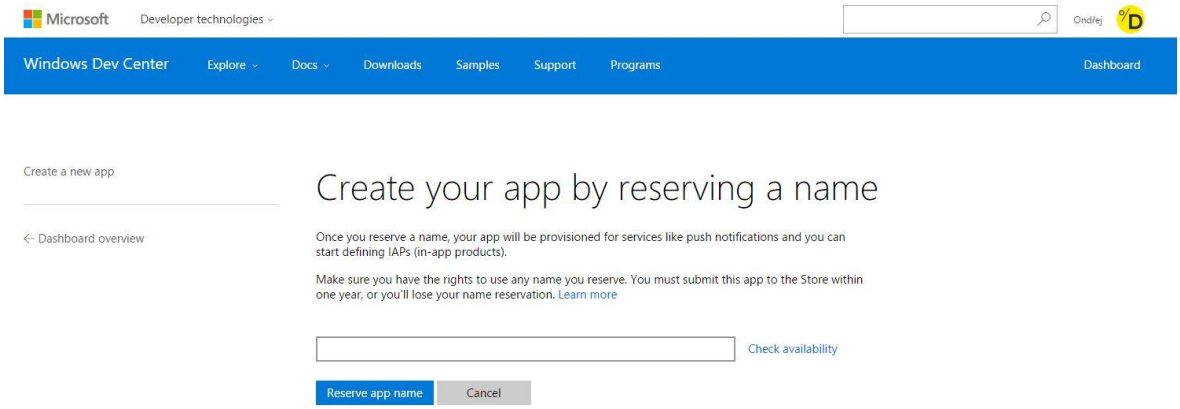
Symbol zahnutého papíru je ve třech variantách. První varianta (obrázek 32) ukazuje prázdný list papíru, který evokuje uživateli vytváření faktur v aplikaci. Druhá varianta (obrázek 33) ukazuje list papíru s vyplněnými údaji odběratele, což evokuje předpřipravenou fakturu pro odběratele. Třetí varianta (obrázek 34) ukazuje list papíru s vyplněnými údaji odběratele a fakturačních položek, což evokuje hotovou fakturu a možnost jejího prohlížení.

Pro aplikaci, zaměřenou na jednoduché vytváření faktur v mobilním zařízení a jejich jednoduchou správu, se hodí nejlépe druhá dlaždice (obrázek 32).

5.9 Publikování aplikace na Windows Store

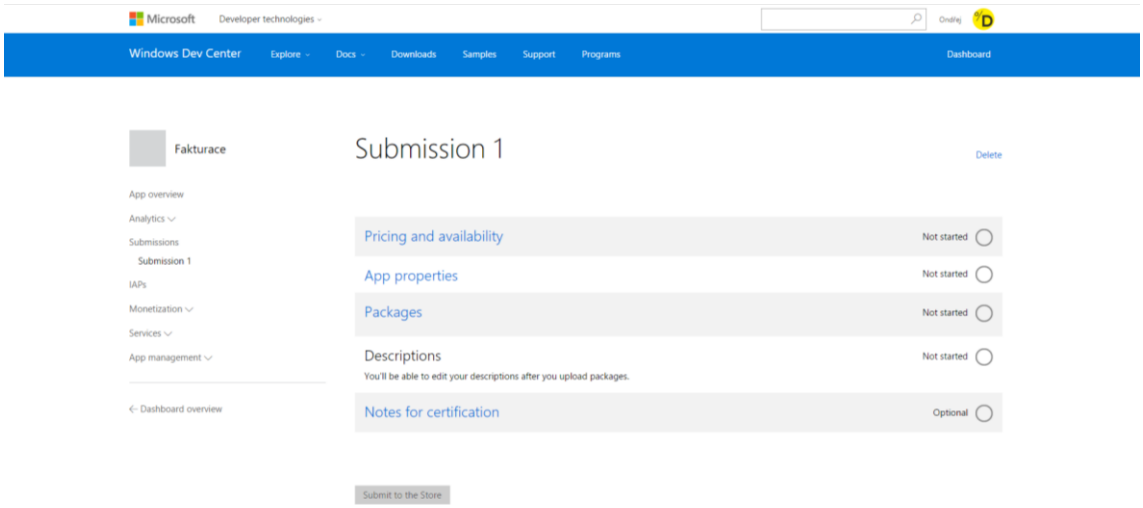
Publikování aplikace na Windows Store je podmíněno nutností mít vytvořen a schválen vývojářský účet u společnosti Microsoft.

Při publikování nové aplikace je nutné v prvním kroku zvolit její název, který bude ověřen vzhledem k ostatním existujícím aplikacím, aby nedošlo k duplicitě názvů.



Obrázek 35: Rezervace názvu aplikace

Poté je nutné vyplnit čtyři povinné kategorie z pěti, které jsou na obrázku 36.



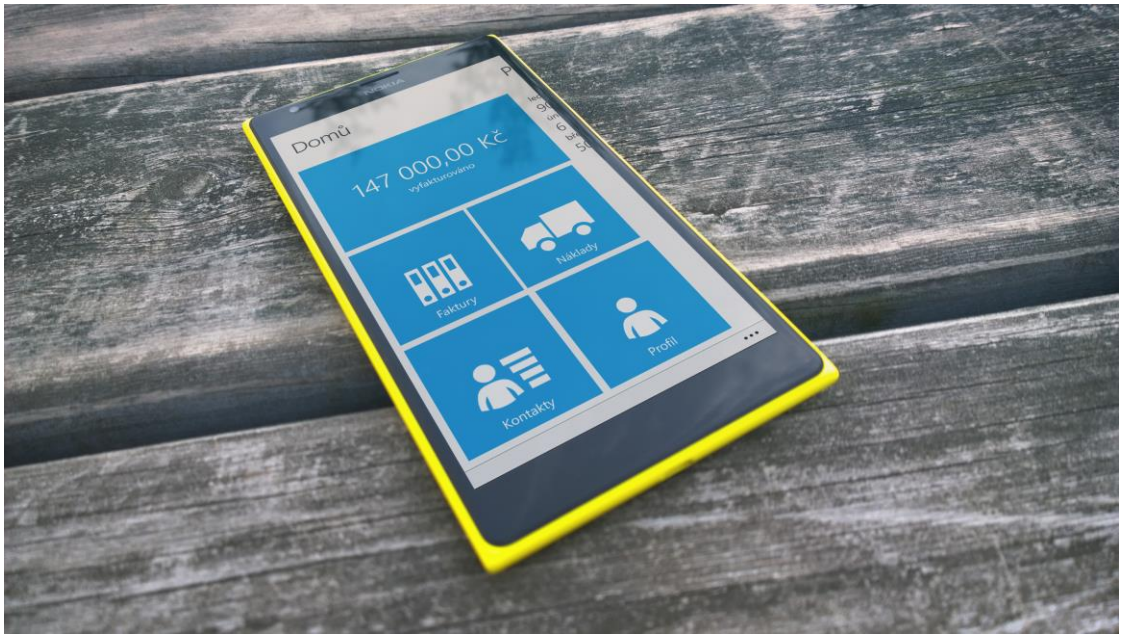
Obrázek 36: Vyplnění informací o aplikaci

Po jejich úspěšném vyplnění se dostává aplikace do schvalovacího procesu, který začíná kontrolou aplikace vůči základním požadavkům na aplikaci – dodržení definovaného vzhledu dané platformy a dodržení bezpečnosti v rámci aplikace. Po splnění tohoto kroku následuje certifikace aplikace. Po ní probíhá akceptace Windows Store a v posledním kroku je aplikace úspěšně publikována.

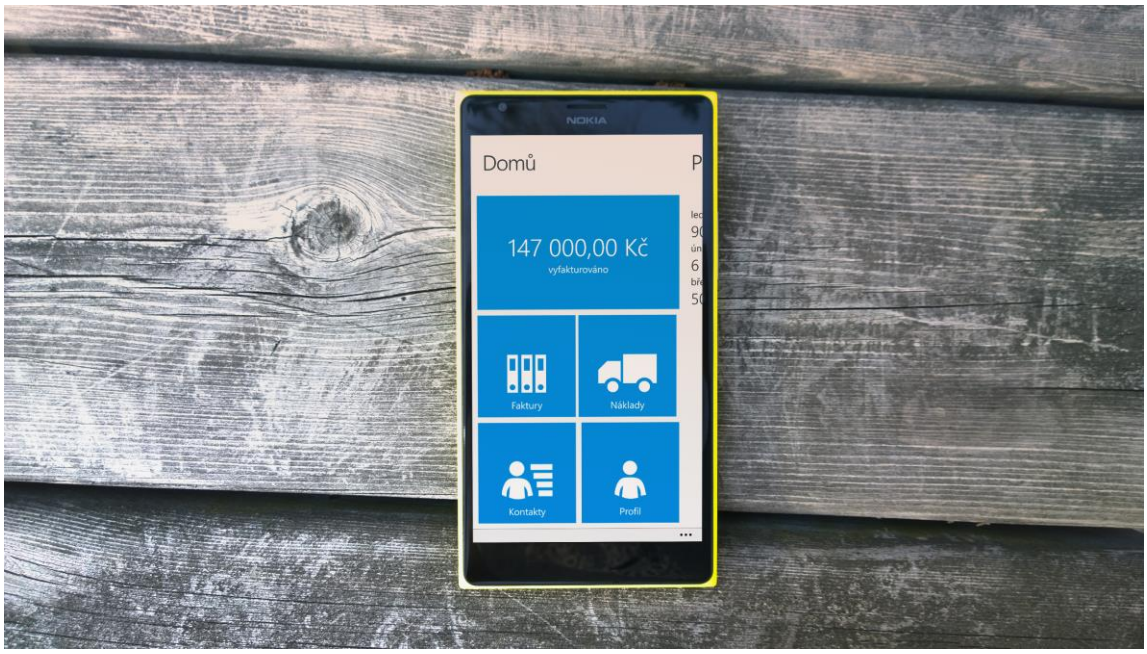
V případě publikace si uživatel může vybrat, zda má být aplikace publikována automaticky po schválení, na základě potvrzení uživatele, či pouze pro specifické uživatele. Poslední možnost je vhodná zejména pro testování aplikace v úzkém okruhu lidí.

5.10 Grafická propagace mobilní aplikace

Po publikování je nutné připravit článek na sociální sítě, ve kterém jsou informováni potencionální zájemci o nové aplikaci. Článek by měl obsahovat jak poutavý text vyzdvihující její funkcionality a nové možnosti, tak důvody pro stažení. Aby nebyl článek strohý, je třeba jej doplnit o zajímavé obrázky, na kterých je zachycena mobilní aplikace v přirozeném prostředí.



Obrázek 37: Propagace mobilní aplikace 1



Obrázek 38: Propagace mobilní aplikace 2

6 Závěr

Cílem diplomové práce bylo objasnit architektury klient/server a REST. Následně se seznámit s vývojem mobilních aplikací pro platformu Windows Phone 8. Na základě zjištěných poznatků vybrat webovou službu obsahující REST API a na ní postavit klientskou aplikaci pro Windows Phone 8.

Teoretická část diplomové práce byla věnována vysvětlení architektury klient/server a objasnění principů REST. Následně byly zmíněny specifické ovládací prvky a popis pravidel tvorby mobilní aplikace pro platformu Windows Phone 8. Praktická část byla zaměřena na vytvoření konkrétní mobilní aplikace postavené na webové službě Fakturoid s ukázkou celého procesu od základního návrhu až po její publikování.

Populární webové služby je nezbytné rozšiřovat o architekturu REST pro komunikaci na klientská zařízení. Určitá část vývojářů se při implementaci dopouští chyb pramenících z nepochopení základních principů. Přínosem teoretické části je objasnění těchto principů a pravidel s uvedením jasných příkladů, jak by měla být tato architektura implementována.

Přínos praktické části spočívá v detailním pohledu na vývoj mobilní aplikace. Jednotlivé kroky vývoje jsou popsány srozumitelně a logicky do sebe zapadají tak, aby je i laický čtenář pochopil. V případě, že by si chtěl udělat vlastní aplikaci, má v této diplomové práci jednoduchý návod, nad čím vším se musí zamyslet a jak má postupovat, aby měla jeho aplikace šanci na úspěch.

Výsledkem této práce je fakturační aplikace pro Windows Phone 8, která na trhu pro tuto platformu doposud chyběla. Uživatelé Windows Phone budou moci nyní využívat výhod webové služby Fakturoid implementované v mobilní aplikaci. Ta umožňuje jednoduchou správu veškerých faktur, odběratelů a dodavatelů na jednom místě kdekoliv a kdykoliv.

Vývoj aplikace probíhal na základě souhlasu se zástupci služby Fakturoid za podmínky, že nebude veřejně publikována, dokud zastupující strana Fakturoidu aplikaci neschválí k veřejné distribuci. Z toho důvodu nebyl splněn cíl publikovat aplikaci.

Vzniklá fakturační aplikace má několik možností na budoucí rozšíření. Některé z možností jsou např. přehled DPH, záznam pravidelných plateb, výpočet cash flow a zobrazení dat ve formě grafů, které jsou uživatelsky více přívětivé.

7 Zdroje

1. **Subhash Chandra Yadav, Sanjay Kumar Singh.** Introduction. *Introduction to Client Server Computing.* New Delhi : New Age International, 2009, stránky 1-7.

2. **Rodriguez, Alex.** RESTful Web services: The basics. [Online] 9. únor 2015. [Citace: 11. listopad 2015.] https://www.ibm.com/developerworks/webservices/library/ws-restful/ws-restful-pdf.pdf.

3. **Kevin Nichols, Donald Chesnut.** Defining UX and the Process. *UX For Dummies.* West Sussex : Wiley , 2014, stránky 8-11.

4. **Richard Caddick, Steve Cable.** Wireframes. *Communicating the User Experience : A Practical Guide for Creating Useful UX Documentation.* New York City : John Wiley & Sons, 2011, stránky 160, 167 - 169.

5. **Konarovský, Lukáš.** Fakturoid pro novináře. [Online] [Citace: 17. listopad 2015.] https://www.fakturoid.cz/press.

6. **Fakturoid.** Co umím. [Online] [Citace: 3. březen 2015.] https://www.fakturoid.cz/co-umim.

7. **Konarovský, Lukáš.** Fakturoid API v2. [Online] [Citace: 18. listopad 2015.] http://docs.fakturoid.apiary.io/#introduction.

8. **Microsoft.** User interface for Windows Phone 8. *Windows Dev Center.* [Online] [Citace: 25. leden 2015.] https://msdn.microsoft.com/en-us/library/windows/apps/ff967556(v=vs.105).aspx.

9. **Microsoft** Layout for Windows Phone 8. *Windows Dev Center.* [Online] [Citace: 25. leden 2015.] https://msdn.microsoft.com/en-us/library/windows/apps/jj207042(v=vs.105).aspx.

10. **Microsoft.** Panorama control for Windows Phone 8. *Windows Dev Center.* [Online] [Citace: 7. únor 2015.] https://msdn.microsoft.com/en-us/library/windows/apps/ff941104(v=vs.105).aspx.

11. **Microsoft** Pivot control for Windows Phone 8. *Windows Dev Centre.* [Online] [Citace: 7. únor 2015.]

12. **Microsoft.** Tile design guidelines for Windows Phone. *Windows Dev Center.* [Online] [Citace: 8. únor 2015.] https://msdn.microsoft.com/en-us/library/windows/apps/jj662929(v=vs.105).aspx.

13. **Microsoft**. In-app navigation for Windows Phone 8. *Windows Dev Center.* [Online] [Citace: 5. únor 2015.] https://msdn.microsoft.com/en-us/library/windows/apps/ff402536(v=vs.105).aspx.

14. **Microsoft**. Application lifecycle. [Online] [Citace: 16. březen 2015.] https://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh464925.aspx.

15. **Microsoft**. Developing apps. [Online] [Citace: 15. březen 2015.] https://msdn.microsoft.com/en-us/library/windows/apps/br229565.aspx.

16. **Microsoft**. XAML overview. [Online] [Citace: 15. březen 2015.] https://msdn.microsoft.com/en-us/library/ms752059(v=vs.110).aspx.

17. **Microsoft**. Začínáme se sadou Visual Studio. [Online] [Citace: 31. říjen 2015.] https://msdn.microsoft.com/cs-cz/library/ms165079(v=vs.120).aspx.

18. **Microsoft**. Account types, locations, and fees. [Online] [Citace: 18. listopad 2015.] https://msdn.microsoft.com/en-us/library/windows/apps/jj863494.aspx.

19. **Microsoft**. Controls for Windows Phone 8. [Online] [Citace: 1. února 2015.] https://msdn.microsoft.com/en-us/library/windows/apps/ff402561(v=vs.105).aspx.

20. **Network Working Group**. Uniform resource identifie (URI): Generic syntax. [Online] [Citace: 15. červen 2014.] http://www.rfc-editor.org/rfc/rfc3986.txt.

.