VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WINDOWS 8 APLIKACE PRO PREZENTACI DAT Z WEBOVÉHO API

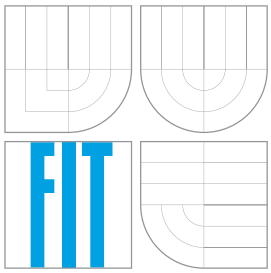
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ KALUS

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WINDOWS 8 APLIKACE PRO PREZENTACI DAT Z WEBOVÉHO API

PRESENTATION OF WEB API DATA IN WINDOWS 8 APPLICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JIŘÍ KALUS
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN MILIČKA
SUPERVISOR

BRNO 2015

Abstrakt

Cílem této bakalářské práce je návrh a implementace Windows 8 aplikace pro prezentaci dat z webového API. Přenesená data jsou pro uživatele citlivá a je zde kladen důraz na autorizaci a bezpečnost. Data jsou zobrazována v reálném čase. Práce obsahuje popis možností vývoje aplikace pro architekturu WinRT, komunikaci aplikace s webovým API, autorizaci a zabezpečení přenášených i offline dat. Byla vytvořena aplikace typu dashboard, která zobrazuje data z různých finančních kategorií. Pro zobrazení byly využity základní grafické komponenty, přičemž některé jsou obohaceny o animace.

Abstract

The aim of this bachelor's thesis is to design and implement a Windows 8 application that can provide data from web API. Transferred data are sensitive for the user with an emphasis for authorization and security. Data is displayed in real time. The thesis decribes possibilities for development of the application for WinRT architecture, communication of the application with web API, authorization and security of transffered and offline data. The application of dashboard type was created that display data from various financial categories. For the visualization basic graphic components were used, some of them includes animations.

Klíčová slova

Windows 8, WinRT, zabezpečení, zobrazování v reálném čase, webové API, animace

Keywords

Windows 8, WinRT, security, real time dislplayed data, web API, animations

Citace

Jiří Kalus: Windows 8 aplikace pro prezentaci dat z webového API, bakalářská práce, Brno, FIT VUT v Brně, 2015

Windows 8 aplikace pro prezentaci dat z webového API

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Miličky. Informace k práci jsem čerpal ze zdrojů uvedených v seznamu literatury a od zástupce firmy Cígler Software, pana Procházky.

.....
Jiří Kalus
14. května 2015

Poděkování

Tímto bych rád poděkoval mému vedoucímu bakalářské práce za odborné konzultace a zejména rychlé reakce na mé požadavky.

Obsah

Kapitola 1

Úvod

Náplní této práce je navrhnout a realizovat Windows 8 aplikaci pro prezentaci dat z webového API. Konkrétně produkt Dashboard Money S5 pro společnost Cígler Software, který očekává ostré nasazení. Aplikace využívá architekturu klient-server. Server obsahuje webové API, které poskytuje data aplikaci. Webové API má přístup k databázi konkrétního zákazníka. Jedná se o odlehčenou verzi účetnického programu Money S5 pro Windows, přednostně určenou pro přenosná zařízení. Inspirací byla především konkurenční aplikace *Inform Motion Dashboard*, která se nachází na *App store* od společnosti Apple. Textové zadání spolu s demonstračními obrázky mi bylo zveřejněno na privátních stránkách firmy Cígler Software.

Účelem produktu je vhodným způsobem uživateli zobrazit požadovaná data z různých finančních odvětví. Důraz byl kladen na jednoduchost a srozumitelnost grafických komponent, které byly vybrány na základě datových struktur. Vyjma autorizace je aplikace zcela pasivní a uživatel je informován jen o času poslední aktualizace. Aplikace byla optimalizována pro ztrátu internetového připojení a pokud je to možné, načte lokálně uložená data.

Základní varianta je pro zařízení, na kterých je nainstalován minimálně operační systém Windows 8. V rámci rozšíření byla prostudována i varianta pro chytré mobily. Nutnou podmínkou pro běh aplikace na mobilu je minimálně Windows Phone 8.1. Datový přenos je na přenosných zařízeních často limitován, a proto byla velikost přenášených dat minimalizována zvolením vhodného formátu serializace. S ohledem na výkon a především omezenou kapacitu baterií provádí aplikace aktualizaci jen na základě přijmutí notifikace ze serveru.

Účelem následujícího textu je poskytnout čtenáři informace o vývoji Windows 8 aplikací, možnostech reprezentace dat a prostředcích pro rychlou a bezpečnou komunikaci na síti. V úvodu práce lze nalézt základní informace o reprezentaci dat a technologiích, které byly použity pro vývoj aplikace. Architektura systému, notifikace aplikace včetně autorizace s možností načtení offline dat je obsažena v kapitole Architektura systému. V kapitole Komunikační protokol je čtenář seznámen s rozhraním webového API, aktualizací dat a struktuře přenesených dat. Implementačně netriviální témata jako push notifikace, zabezpečení lokálních dat či autorizace je popsána v kapitole Implementace. V konečné kapitole jsou popsána možná rozšíření aplikace.

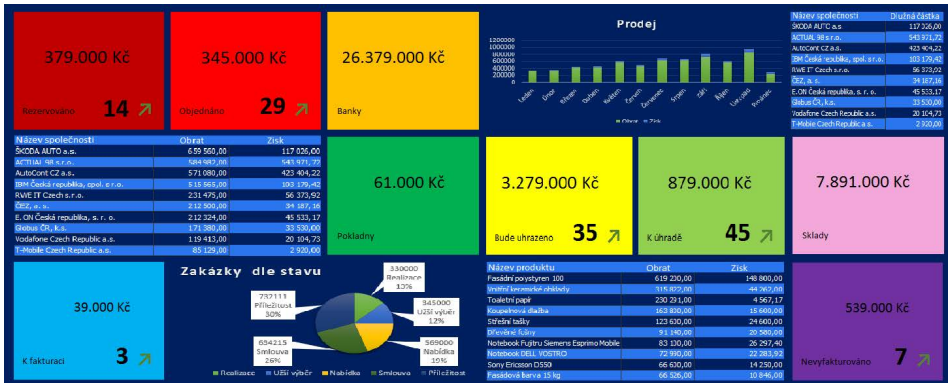
Kapitola 2

Způsoby reprezentace dat a vybrané technologie

2.1 Dashboard

Data jsou uživateli zobrazena v podobě dashboardu. Dashboard v informačních technologiích vyjadřuje generalizovanou kolekci dat. Typicky se jedná o jednoduché a přehledné uživatelské rozhraní, kde zobrazená data jsou odrazem aktuálního stavu systému. Velký důraz je kladen na srozumitelnost a zobrazení jen užitečných informací. Reprezentace dat silně závisí na druhu a složitosti zobrazované informace. V praxi se pro zobrazení využívá několik typů komponent, které jsou popsány níže v této kapitole. Účelem dashboardu je zobrazit uživateli data z komplexnějšího systému.

Aplikace Dashboard Money S5 zobrazuje data z různých finančních kategorií, jako je například seznam dlužníků, seznam neuhrazených faktur nebo stav bankovního účtu. Každá kategorie je zastoupena grafem, tabulkou či prostým zobrazením připomínajícím dlaždici. Po kliknutí na příslušnou kategorii je uživateli zobrazen její detail. Je-li dané zařízení připojené k internetové síti, pak jsou prezentovaná data vždy aktuální. Pokud aplikace projde autorizací a poté ztratí spojení, je možné načíst offline data, která byla ve stavu online uložena viz 5.4.



Obrázek 2.1: Původní návrh Dashboard Money S5 aplikace

Aplikace bude umístěna v digitální distribuční platformě *Windows Store*. Pro zobrazení dat je nutné, aby měl uživatel účet u společnosti Cígler Software. Široké veřejnosti je umožněno aplikaci stáhnout, ale bez přihlašovacích údajů jim zůstane zobrazena pouze

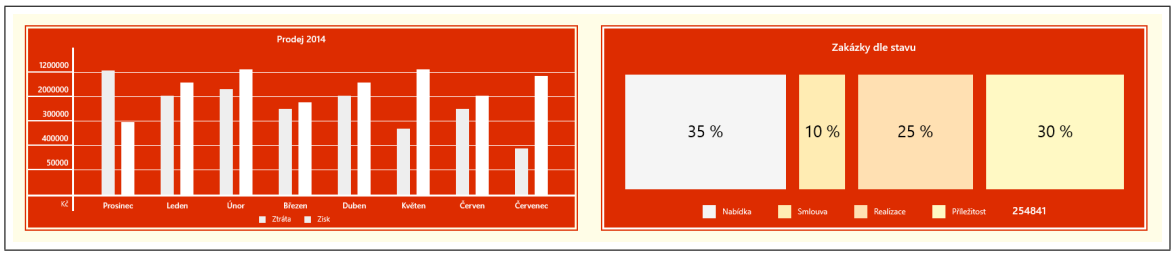
úvodní obrazovka vyzývající k zadání autorizačního klíče. Dle požadavků firmy je aplikace určena pro osoby z vyššího managementu, které si budou zobrazovat data své, případně cizí společnosti.

2.2 Komponenty pro reprezentaci dat

Dashboard aplikace se skládá z několika typů grafických komponent. Výhody a nevýhody komponent použitých v aplikaci spolu s detailnějším popisem jejich vzhledu a chování naleznete v následujících podkapitolách. Před samotným vývojem aplikace byly nastudovány možnosti zobrazení různých datových struktur [3]. Mezi nejpoužívanější patří sloupcový, koláčový graf a tabulky. Podněty ke vzhledu a chování komponent byly inspirovány produkty společnosti Telerik Kendo UI. Veškeré grafické prvky společně s jejich chováním byly od základu implementovány. Nebyly tedy použity žádné komponenty třetích stran.

Graf

Grafy typicky reprezentují strukturovaný datový typ kolekce. U tohoto typu zobrazení nejsou požadovány rozsáhlejší filtrace dat. V aplikaci byl použit sloupcový a stacked bar graf. Stacked bar graf nemá v češtině zavedený ekvivalent a proto je zde uveden anglický název. Příklady obou grafů jsou uvedeny níže.



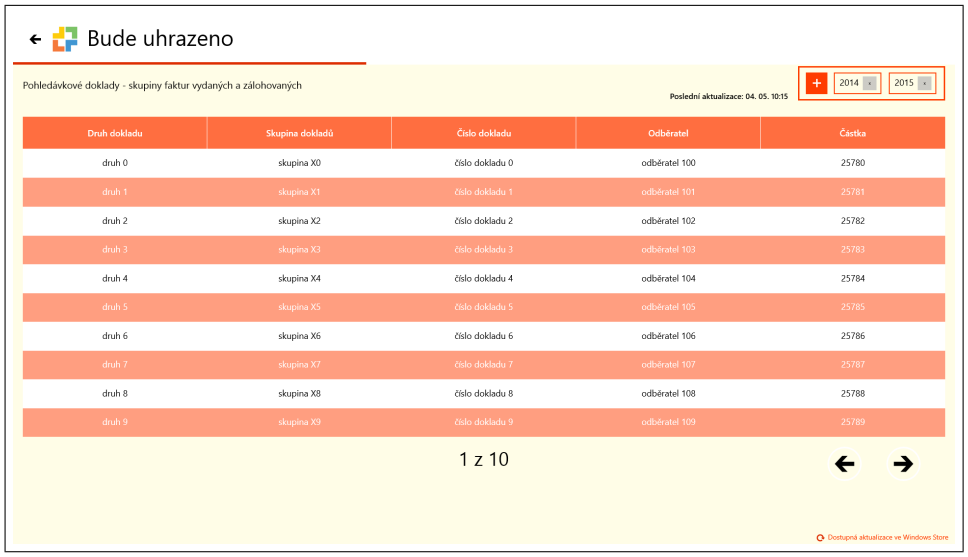
Obrázek 2.2: Použité grafy v aplikaci: Sloupcový (vlevo) a stacked bar (vpravo)

Na začátku vývoje aplikace byly grafy posílány do aplikace formou obrázku. Aplikace poslala informaci o svém rozlišení a poté jí byl zaslán odpovídající graf. Tato forma přenosu a zobrazení sice umožňuje centrální a především dynamickou správu, ale je zde absence animací a zvyšuje se datový přenos. Další možností je vytvořit graf pomocí HTML / CSS / JS a vystavit tento graf na internetu. Aplikace by pak použila vestavěný prohlížeč a graf zobrazila. Razantně se ale zvyšuje velikost přenesených dat. Finální řešení je zaslání hodnot přímo do aplikace, která je zpracuje a zobrazí.

Náhled grafu zobrazuje poslední přijaté hodnoty. Po kliknutí na náhled se zobrazí graf přes celou obrazovku, kdy je uživateli umožněno filtrovat data. Při zobrazení byly využity výše zmiňované animace hodnot. Přenos dat je také minimalizován, protože se přenáší hodnoty, nikoliv vzhled. Použití grafů může vést k problematickému zobrazení na zařízení s menší zobrazovací plochou.

Tabulky

Tabulka je vhodná pro datové objekty, u nichž potřebujeme podrobnější informace. Vizualizuje kolekci struktur. Profity tabulkového zobrazení je možná filtrace dat pro danou kolekci a její možnost přehledně zobrazit podrobnější informace k položce v tabulce.



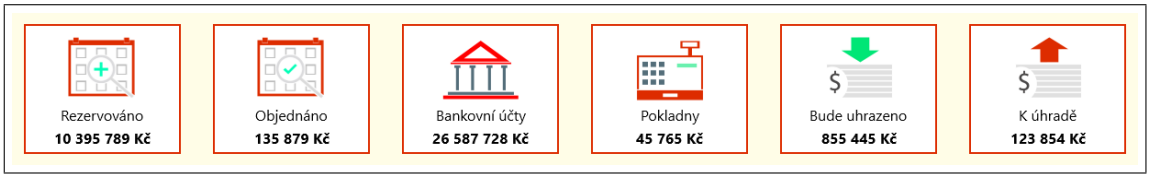
Obrázek 2.3: Tabulka použitá v aplikaci

V aplikaci je data možné filtrovat dvěma způsoby. První způsob je výsuvné menu, kde jsou jednotlivé položky pro filtraci. Druhý způsob filtrace je kliknutí na název sloupce. V závislosti na pořadí kliknutí jsou hodnoty daného sloupce setříděny (po směru abecedy, proti směru abecedy a původní stav). Oba typy filtrací se navzájem ovlivňují a lze tedy filtrovat data zároveň. Náhled tabulky obsahuje prvních pět položek z kategorie, kterou zastupují. Po kliknutí na náhled se zobrazí tabulka přes celou obrazovku, ve které je možné listovat a filtrovat. Stejně jako u grafů může použití tabulky vést k problematickému zobrazení na zařízení s menší zobrazovací plochou.

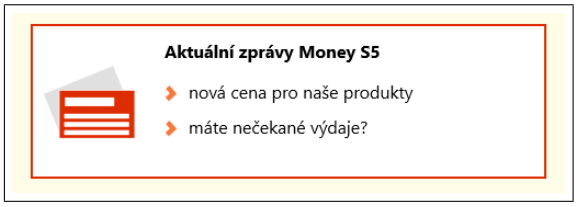
Dlaždice

Dlaždicí chápeme obdélníkově či čtvercově vymezené místo zobrazovací plochy, kde jsou uživateli předána data pomocí prostého textu nebo piktogramu/ikony (zisk, rozdíl, stav, počet, ...). Dlaždice lze obecně rozdělit na statické a živé. Statické zobrazují neměnná data jako například piktogram a jeho stručný popis. Živé dlaždice mění v čase svůj obsah, jako například náhledy fotek nebo číslo reprezentující počet nově příchozích emailů. Dlaždice se začaly široce používat s příchodem Windows 8 a rozhraním *Modern UI*.

Základní dlaždice v aplikaci Dashboard Money S5 mají čtvercový tvar a zobrazují agregované hodnoty pro finanční kategorii, kterou zastupují. Dlaždice obsahuje ikonu, jednoduchý název a agregované hodnoty z kategorie, kterou zastupuje. Po kliknutí na dlaždici jsou uživateli zobrazeny detailnější informace, nejčastěji formou tabulky. Jako další příklad lze uvést posuvný kontejner, který sdružuje bankovní účty a uvádí k nim logo příslušné banky.



Obrázek 2.4: Dlaždice použité v aplikaci



Obrázek 2.5: Dlaždice v aplikaci

V aplikaci je také dlaždice, která informuje uživatele o zprávách. Dlaždice disponuje dvěma odrážkami, u kterých se periodicky mění text. Kliknutím na dlaždici jsou uživateli zobrazeny interaktivní stránky ze serveru. Díky vestavěnému prohlížeči je tak uživatel stále v aplikaci. Výhoda tohoto zobrazení je hlavně v jednoduché a centrální správě.

2.3 Vývojové prostředí a jazyk

Aplikace

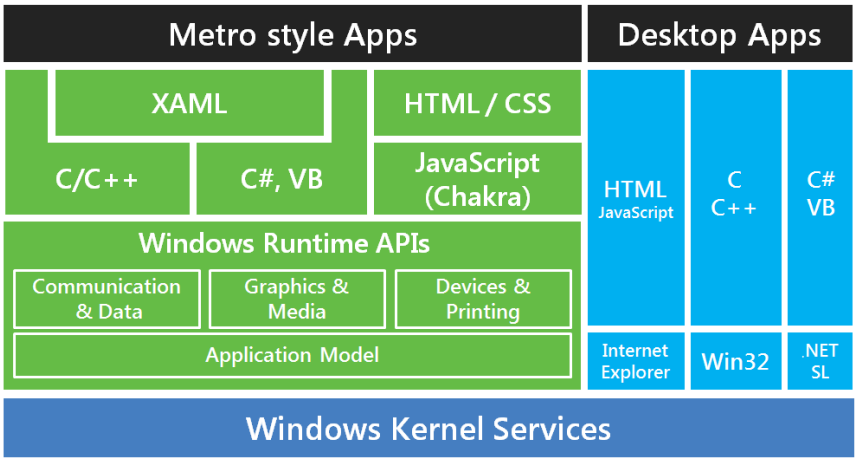
- **Visual Studio**
  Integrované vývojářské prostředí přímo od společnosti Microsoft. Pro vývoj *Universal Apps* je nutné mít Visual Studio 2013 Update 2 a vyšší.
- **Windows RunTime**
  Operační systém musí obsahovat architekturu *Windows RuntTime*, dále jen *WinRT*, která je podporována Windows 8 a vyšší.

WinRT

WinRT je homogenní architektura pro aplikace běžící na operačním systému Windows 8 a vyšších verzích [10]. *WinRT* je implementované v C++ a je to nejvyšší softwarová vrstva naší aplikace, která komunikuje s operačním systémem. Tento typ architektury je určen pro *Modern UI* aplikace, známé jako Metro. Aplikace vytvořené pod touto architekturou běží v *sandboxu*.

Běhové prostředí *sandbox* přináší výhodu hlavně v odděleném paměťovém prostoru. Ostatní aplikace tedy nemohou přistoupit k paměti, která pro ně není alokována. Je zde kladen vyšší důraz na bezpečnost. To s sebou přináší i určité nevýhody. Nemůžeme volat systémové funkce, které nejsou implicitně povolené, např. vypnutí nebo restartování zařízení.

Pro tvorbu *WinRT* [5] aplikací jsou uživateli umožněny dvě kategorie volby jazyka. První kategorie je za podpory .NET technologie, kdy je možné použít pro chování aplikace C#, VB.NET nebo C++ a pro vzhled aplikace je poté nutný XAML. Druhou možností je použít Javascript pro chování aplikace a vzhled je nutný implementovat v CSS/HTML. Příkladem první kategorie jsou ve *Windows store* Mapy, Kontakty nebo OneDrive, příkladem druhé kategorie je Skype nebo Počasí.



Obrázek 2.6: Architektura aplikací Windows 8 [2]

Presentation Foundation a XAML

Technologie *Windows Presentation Foundation*, dále jen WPF, je framework pro komplexní tvorbu obsáhlých formulářových aplikací, který je součástí .NET frameworku od verze 3.0. WPF je logický nástupce dříve používané technologie *Windows Forms*. Tato technologie je Microsoftem stále podporována a lze tedy vyvíjet aplikace i na nejnovějších operačních systémech (Windows 8, 8.1 nebo 10), avšak vhledem k rozsahu různých zobrazení aplikací není vhodné v dnešní době *Windows Forms* používat [12].

Důvodem vzniku WPF je unifikovat způsob návrhu mezi platformami. Pro *Windows Forms* aplikace je problém přizpůsobit se rozdílné fyzické velikosti koncového zařízení z důvodu slabé podpory DPI[1]. Nelze použít stejnou aplikaci na zařízeních s rozdílným rozlišením. WPF zavádí nezávislou jednotku délky DIP a všechny elementy vytváří vektorově. Díky těmto dvěma aspektům je výsledná aplikace nezávislá na rozlišení zobrazovacího zařízení. Původní grafické rozhraní (GDI) ve *Windows Forms* je nahrazeno a pro vykreslování formulářů se používá *DirectX*. V důsledku akcelerované grafiky má aplikace menší výpočetní režii na procesor a aplikace je svižnější.

Pro uživatelské rozhraní je ve WPF používán jazyk XAML, který je odvozený od značkovacího jazyka XML. Všechny prvky, které definujeme v jazyce XAML lze zapsat i v jazyce C# či Visual Basic. *WinRT* aplikace mají vyjma standardní funkcionality (desktopové aplikace) také podporu pro dotykové události.

Webové API a databáze

Pro tvorbu logiky webového API je možné použít několik programovacích jazyků. Mezi základní používané patří PHP, PERL, JAVA nebo ASP.NET. Výběr jazyka závisí na požadavcích zadavatele a zkušenosti vývojáře. Webové API pro aplikaci Money Dashboard S5 bylo implementováno v jazyce PHP 5. Databáze byla zvolena MySQL. Kombinace PHP a MySQL byla použita, protože existuje mnoho návodů a je zdarma. Vzhled a jeho chování bylo implementováno v jazyce HTML / CSS / JS.

[1] Device Independent Pixel

2.4 Síťová komunikace

Následující sekce popisují síťové technologie, které byly použity při implementaci aplikace. Přenos dat je šifrován přes HTTPS, samotná data jsou ve formátu JSON a autorizace probíhá pomocí protokolu OAuth.

2.4.1 HTTPS

HTTPS je nadstavba aplikačního protokolu HTTP. Jedná se o klient-server komunikaci, kdy se typicky klient dotazuje a webový server mu odpovídá. HTTP je nestavový protokol, což znamená, že webový server neví, kterému klientovi odpovídá. V důsledku toho server nezná kontext klienta, což je v některých případech klíčové. Stav klienta se dá zaručit použitím dočasných souborů na straně klienta (*cookies*) nebo sezením na straně serveru (*sessions*).

U HTTPS je prohlížeči typicky předán požadavek, aby použil kryptovací protokoly SSL/TLS k přenosu dat, které využívají asymetrické šifrování. Protokoly SSL/TLS zabraňují podvrhnutí zpráv či jejich odposlech. Nejčastěji je autentizován pouze server a klient zůstává neautentizován. Pro autentizaci obou komunikujících stran je potřeba nasazení infrastruktury veřejných klíčů pro klienty.

HTTP požadavek

Protokol HTTPS pracuje s HTTP požadavky a disponuje dvěma typy zpráv: požadavek a odpověď. Ve verzi HTTP/1.1 jsou základní metody:

- `GET` - získání obsahu objektu. Data jsou přenesena v části URL jako požadavek
- `POST` - přidání dat k obsahu objektu. Data jsou přenesena v těle zprávy
- `HEAD` - získání hlavičky objektu. Poskytuje metadata o požadovaném cíli
- `PUT` - nahrání obsahu souboru na server
- `DELETE` - smazání (části) objektu

Metody `HEAD` a `GET` jsou označovány jako bezpečné, protože by neměly změnit stav serveru. Slouží pouze k získávání informací. Metody `POST`, `PUT` a `DELETE` jsou určeny pro akce, které mohou změnit stav serveru. Metody `PUT` a `DELETE` jsou navíc označovány jako idempotentní, což znamená, že více shodných požadavků by mělo mít stejný účinek jako jeden požadavek. Zaslání totožného post požadavku vícekrát za sebou může způsobit nežádoucí účinky na serveru.

2.4.2 Architektura REST

Architektura rozhraní REST (*Representational State Transfer*) pro distribuované prostředí je způsob, jak jednoduše vytvořit, číst, editovat nebo smazat data pomocí HTTP volání. Styl komunikace musí dodržovat těchto 6 omezení, viz. [8]:

1. **Jednotné rozhraní** (*Uniform interface*) - rozhraní mezi klientem a serverem musí být definováno takových způsobem, aby mohly být tyto dvě části implementovány nezávisle

Základní principy pro jednotné rozhraní:

- Zdroje dat - jsou koncepčně oddělené od reprezentací, které jsou vráceny klientovi. Například server neodesílá svou databázi, ale pošle data serializovaná do jazyka XML nebo JSON
- Manipulace se zdroji dat - pokud má klient oprávnění, tak by měl mít dostatek informací k tomu, aby mohl změnit nebo vymazat data na serveru
- Zpracování zprávy - každá zpráva obsahuje i popis, jak ji zpracovat

2. **Bezstavovost** (*State-less*) - komunikace mezi klientem a serverem je bezstavová. V každém požadavku klienta je specifikováno, co přesně požaduje. Nevyužíváme tedy *sessions*. To nám zaručí, že stav zdroje dat je konstantní pro každého klienta, který si o něj požádá

3. **Mezipaměť** (*Cacheable*) - klienti mohou data ukládat do mezipaměti. Odpověď serveru ale musí obsahovat informaci, zda tato data mohou být ukládána nebo ne. Správně nastavené ukládání částečně eliminuje některé klient - server operace

4. **Klient - server** (*Client - server*) - rozhraní klienta a serveru na sobě nejsou závislá. Klienti nejsou propojeni přímo se zdrojem dat a servery také nejsou informovány o uživatelových stavech

5. **Systém vrstev** (*Layered system*) - klient nesděluje, zda komunikuje přímo s koncovým serverem. Architektura na straně serveru tak může lépe rozložit zátěž nebo zvyšovat úroveň zabezpečení

6. **Kód na požádání** (*Code on demand*) - server také může dočasně měnit logiku klienta například pomocí Java appletů nebo Javascriptu. Toto omezení je pouze volitelné. Pokud není v odpovědi serveru takto označené, nemůžeme komunikaci označit jako *RESTfull*[2].

Všechny zdroje mají vlastní identifikátor URI, kde REST definuje čtyři základní operace (CRUD) pro manipulaci se zdroji: *Create*, *Read*, *Update* a *Delete*. Tyto operace jsou namapovány na HTTP operace `POST`, `GET`, `PUT` a `DELETE`. Zdroje mohou mít různé reprezentace (JSON, XML, SVG, PDF). REST architektura umožňuje snadný přístup ke zdrojům, kterými mohou být data i stavy aplikace. REST je tedy orientován datově, nikoliv procedurálně (RPC[3]).

Možnosti autorizace

Většina služeb typicky vyžaduje autorizaci před přístupem ke zdroji dat. Využívat *sessions* není povolené. Provádět autorizaci při každém požadavku až na výjimečné případy není správná volba. Existují dva základní koncepty autorizace:

1. Jednoduché ověření přístupu (*HTTP Authentication*) - server vyzve přistupujícího klienta, aby v HTTP požadavku poslal také přístupové údaje (jméno a heslo). Údaje jsou poslány jako jeden řetězec, který je zakódován metodou `Base64` a odeslán s v rámci HTTP požadavku. Jedná se o základní ověření přístupu. Data se kódují, aby

[2]REST je architektonické paradigma. Restfull popisuje služby, které se tímto paradigmatem řídí
[3]RPC je vzdálené volání procedur

se nahradily znaky nepatřící do množiny povolených znaků[4] HTTP. Nejedná se tedy o zabezpečenou komunikaci.

2. Použít vestavěnou službu, která v zásadě autorizuje uživatele a vrátí přístupový *token*. Tento styl autorizace je popsán níže v této kapitole viz. 2.4.4

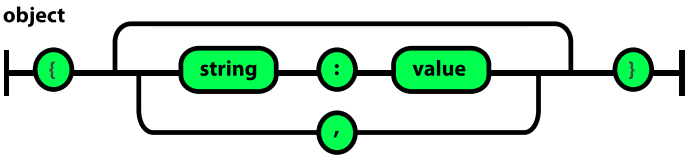
Obě možnosti autorizace mají své výhody a nevýhody. Výhoda prvního způsobu je snadná implementace a široká podpora napříč webovými prohlížeči. Výhoda druhého způsobu je omezení doby přístupu. Přístupový klíč jednoduše vyprší a uživatel je nucen provést autorizaci znovu.

2.4.3 JSON

JavaScript Object Notation je formát pro výměnu dat. Je založen na podmnožině programovacího jazyka JavaScript, Standard *ECMA-262 3rd Edition - December 1999* [1]. Jedná se o štíhlejší variantu XML formátu a typicky se používá pro výměnu dat mezi klientem a serverem ve webových aplikacích. Přestože je JSON odvozen z JavaScriptu, je jazykově nezávislý. Zápis v JSON je literálem v jazyce JavaScript a není proto potřeba speciální analyzátor. Všechny prohlížeče implicitně analyzují JSON. Data jsou serializována do těchto dvou forem:

Struktura v JSON (objekt)

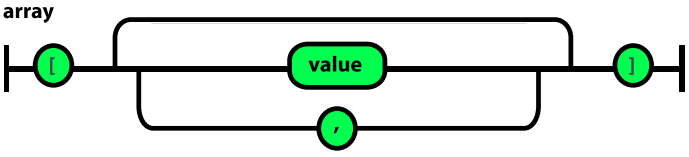
Objekt je definován jako neuspořádaná množina dvojicí jméno a hodnota. Každý objekt začíná levou složenou závorkou a končí pravou složenou závorkou. Každé jméno objektu je odděleno dvojtečkou, za níž je uvedena hodnota. Objekt je oddělen čárkami.



Obrázek 2.7: Struktura v JSON (objekt)

Kolekce v JSON (pole)

Pole je definováno jako uspořádaná kolekce hodnot. Pole začíná levou hranatou závorkou a končí pravou hranatou závorkou. Hodnoty jsou oddělené čárkou.



Obrázek 2.8: Kolekce v JSON (pole)

[4]definováno v druhé sekci (*Section 2: Characters*) RFC (`<https://tools.ietf.org/html/rfc3986>`)

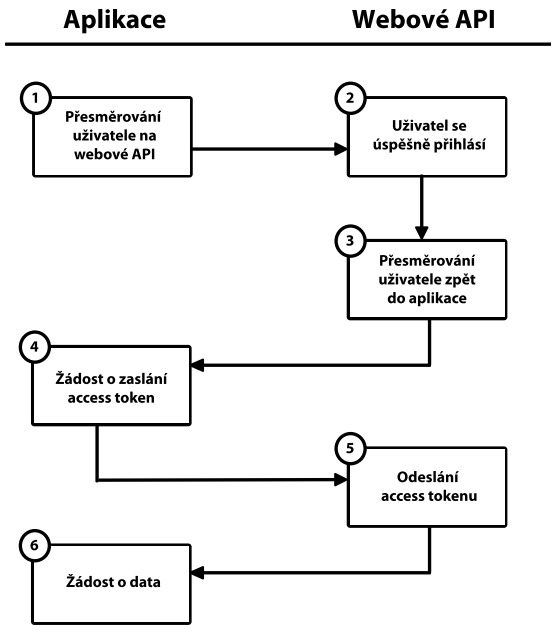
Hodnota v JSON může nabývat následující typy: *string*, *number*, *true*, *false*, *null*, *objekt* nebo *pole*.

2.4.4 OAuth

Moderní autorizační protokol, který se stal standardem pro zabezpečení RESTových webových služeb. OAuth je otevřený protokol navržený Blaine Cookem a Chrisem Messinou. Tento protokol vznikl v roce 2006 a umožňuje klientské aplikaci přístup k datům libovolné služby, aniž by této aplikaci byly zpřístupněny přihlašovací údaje. Dále umožňuje vymezit pravomoci jednotlivých klientských aplikací. Aktuální verze OAuth 2.0 definuje čtyři druhy rolí [11]:

- **uživatel** (*resource owner*) - vlastník chráněného zdroje. Typicky se jedná o koncového uživatele, který je schopný povolit nebo odepřít přístup ke chráněnému zdroji (datům)
- **služba** (*resource server*) - poskytovatel a hostitel chráněného zdroje. Tato služba obsluhuje požadavky (mající *access token*) ke chráněnému zdroji. Ve většině případů se jedná o serverovou službu vystavující API
- **klient** (*client*) - aplikace, která přistupuje ke chráněnému zdroji s patřičnými oprávněními
- **autorizační server** (*authorization server*) - server, který klientovi přiděluje *access token* v případě jeho úspěšné autentizace od uživatele. Typicky je autorizační server součástí služby

Princip autorizace aplikace vůči službě, dále jen webové API je následující:



Obrázek 2.9: Autorizace protokolu OAuth

1. Aplikace požádá webové API o jednorázový *request token*

2. Webové API potvrzuje validitu aplikace
3. Aplikace (uživatel) je přesměrována na přihlašovací bránu webového API a v požadavku předá *request token*
4. Po přihlášení na straně webového API je uživatel přesměrován zpět do aplikace. Poté aplikace zažádá o *access token*
5. Webové API vygeneruje *access token* a pošle ho zpět aplikaci
6. Aplikace požádá s *access tokenem* o data

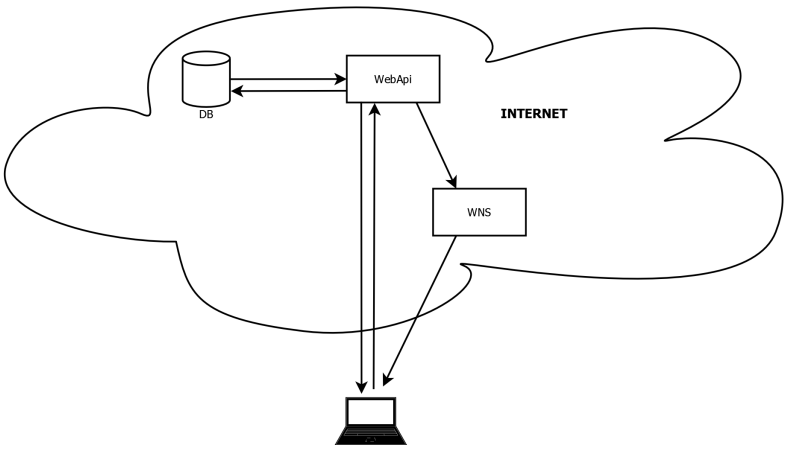
Specifikace OAuth detailně popisuje výměnu přístupových klíčů (tokenů). Není zde popsána komunikace mezi klientem (službou) a webovým API [7]. OAuth není spjatý s konkrétním typem API (SOAP, REST, WCF atd.). Je pouze omezen na HTTPS protokol.

Kapitola 3

Architektura systému

3.1 Schéma a úloha jednotlivých elementů

Celý systém se skládá z pěti částí: databáze, webového API, WNS a aplikace. Po spuštění aplikace je uživatel vyzván k zadání přístupového klíče. Uživatel tedy přistoupí v prohlížeči na jeden z endpointů webového API, kde si po přihlášení nechá vygenerovat klíč, který vloží do aplikace. Webové API naslouchá/odpovídá na HTTPS požadavky. Webové API je přímo propojené s databází, na které se nachází chráněné zdroje dat. K databázi tedy uživatel nemá přístup. Po úspěšné autorizaci uživatele webové API kontaktuje službu WNS, která přepošle notifikaci z webového API do aplikace. WNS je také využívána pro informování o aktualizaci dat. Webové API při změně dat kontaktuje WNS, které přepošle notifikaci do aplikace 3.2. Samotná data pro grafické komponenty jsou přenášena ve formátu JSON. Síťové připojení aplikace k webovému API je různé v závislosti na dostupné technologii (GRPS, 3G, LTE, Wifi, LAN, ...).



Obrázek 3.1: Schéma architektury

Webové API a databáze

V požadavcích firmy Cígler Software je pouze vývoj aplikace a následné napojení na jejich webové API. Po dohodě s vedoucím bakalářské práce jsem pro demonstrační účely imple-

mentovat vlastní webové API. Webové API naslouchá na jednotlivých *endpoints*[1] a podle požadavků odpovídá. Data jsou do aplikace generována přímo v kódu a nejsou perzistentně uložena. Autorizační údaje viz 3.2, stejně jako stav aplikace jsou uložené v databázi.

WNS

Služba *Push Notification Services* společnosti Microsoft, která zajistí doručení notifikace přímo do daného zařízení. Pro úspěšné doručení notifikace je nutné autorizovat vlastní webové API oproti WNS. Autorizace probíhá přes protokol OAuth 2.0 viz 2.4.4.

3.2 Notifikace

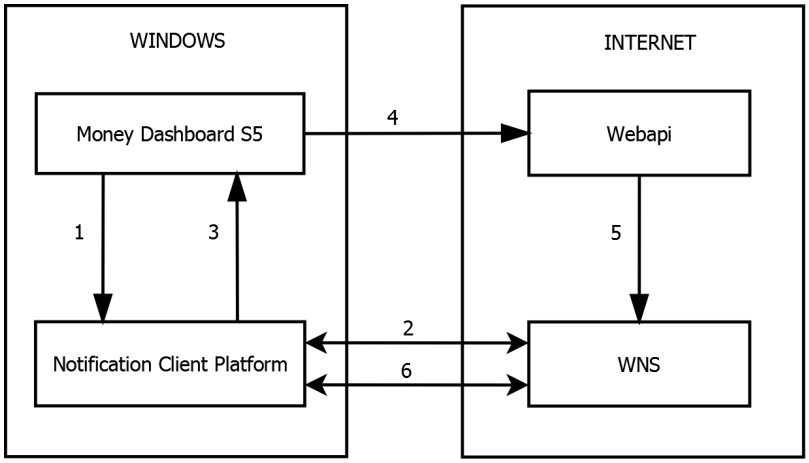
Jednou z možností předání informace uživateli/aplikaci je stav, kdy se aplikace aktivně dotazuje webového API, zda nedošlo ke změnám. Výhoda tohoto řešení je nezávislost na službách třetí strany. V praxi to však vede ke zbytečnému zatěžování sítě a vybíjení baterie na koncovém zařízení. Druhá možnost je zaslání *push* notifikace, kdy aplikace je v pasivním režimu a jen naslouchá. Po konzultacích s vedoucím bakalářské práce a zástupcem z firmy Cígler Software byla zvolena druhá možnost.

WNS neinformuje webové API o tom, zda byla notifikace doručena do zařízení. Pokud je aplikace offline, WNS uloží maximálně 5 posledních notifikací. Doručení nastane, jakmile aplikace přejde do stavu online. Microsoft negarantuje úspěšné odeslání notifikací. V případě, že se nepodaří doručit notifikaci do tří dnů, služba WNS notifikace smaže.

Postup autorizace pro WNS

Před zasláním notifikace je nutné autorizovat webové API oproti službě WNS. Autorizace probíhá dle protokolu OAuth 2.0 2.4.4. Webové API nejprve požádá WNS o *access token*. Pro získání *access tokenu* musí webové API předat identifikační údaje aplikace. Identifikační údaje jsou vygenerovány po zaregistrování aplikace do *Windows Store*. Tento krok provede zodpovědná osoba, nejčastěji vývojář aplikace. Identifikační údaje se již nezmění a skládají se ze dvou položek: *Package Security Identifier* a *Client Secret*. Získání těchto dvou položek je uvedeno v kapitole 5.1. Po získání *access tokenu* je společně s tokenem zaslán i *notification channel*, což je adresa, na které aplikace naslouchá. Pokud je *access token* stále platný, WNS zašle notifikaci přímo do příslušného zařízení. *Access token* má omezenou dobu platnosti a webové API je o tom informováno HTTP statusem 400. Po vypršení je potřeba webové API znovu autorizovat. Microsoft neuvádí přesnou dobu platnosti. Stejně tak má omezenou dobu platnosti i *notification channel* a webové API je o tom informováno HTTP statusem 410. Dle MSDN [9] je doba platnosti *notification channel* zhruba 30 dní. Schéma a postup v jednotlivých bodech je uveden níže:

[1] endpoint - vstupní brána, na které služba naslouchá



Obrázek 3.2: Workflow autorizace a zaslání notifikace

1. Aplikace Money Dashboard S5 pošle požadavek *Notification Client Platform*, dále jen NCP o *notification channel*, dále jen NC
2. NCP zažádá WNS o vytvoření NC. Tento NC je poslán zpět do volajícího zařízení formou *Uniform Resource Identifier*, dál jen URI
3. NCP vrátí URI aplikaci Money Dashboard S5
4. Money Dashboard S5 pošle URI na webové API, kde se uloží do databáze. Tento krok je celý pod kontrolou vývojáře aplikace
5. Nastane-li stav, kdy je nutné poslat notifikaci do aplikace, pak se webové API autorizuje oproti WNS a zažádá o zaslání notifikace
6. WNS obdrží žádost o notifikaci a přepošle ji do NCP, které doručí notifikaci přímo do aplikace

Kapitola 4

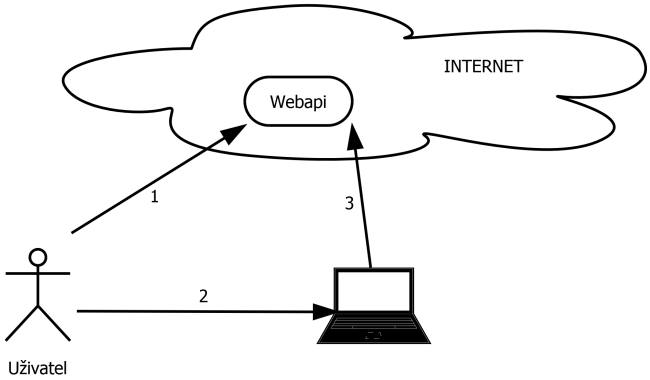
Komunikační protokol

4.1 Základní prvky komunikace

Komunikace mezi webovým API a klientskou aplikací je zprostředkována HTTPS dotazy. Velký důraz je kladen na minimální přenos dat mezi těmito komunikačními body. Aplikace využívá architekturu klient-server. Aplikace jako klient, webové API jako server. Aplikace je nejprve autorizována a poté jsou jí zaslána data. Data jsou serializována na straně webového API do JSONu a poté deserializována v aplikaci.

4.2 Autorizace

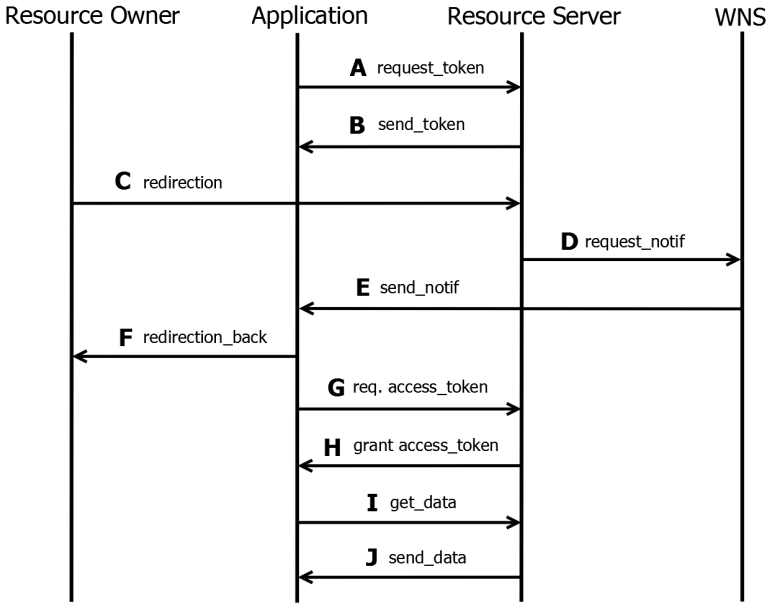
Po spuštění aplikace je uživatel požádán o zadání identifikačního klíče, dále jen *consumer key*. *Consumer key* generuje autoritativní webové API. Uživatel musí být nejprve zaregistrován u firmy Cígler Software, kde získá přihlašovací údaje pro autorizaci na straně webového API. Grafické rozhraní pro registraci nebylo implementováno, přihlašovací údaje jsou vkládány do databáze manuálně. Po přihlášení na portálu webového API je možné si vygenerovat klíč. *Consumer key* je uložen do databáze. Uživatel si tento klíč uloží a přejde do aplikace Dashoboard Money S5, kde klíč vloží. Po zadání klíče aplikace kontaktuje webové API a spolu s klíčem se u ní identifikuje. Díky *consumer key* je možné po prvním spuštění rozpoznat uživatele, který klíč zadal. Pokud uživatel nepoužije klíč do tří dnů, je klíč z databáze odstraněn. Při druhém a dalším spuštění aplikace již není nutné zadat *consumer key*. Aplikace si pamatuje *access token*, který byl vygenerovaný po přihlášení uživatele. Pošle *access token*, webové API v databázi vyhledá uživatele, kterému naposledy přidělilo tento klíč a odpoví aplikaci s novým *access tokenem*. Proces vygenerování *consumer key* spolu s krátkým popisem je zobrazen níže.



Obrázek 4.1: Workflow vygenerování consumer key

1. Uživatel se přihlásí na portálu webového API a vygeneruje si *consumer key*
2. Tento klíč vloží do aplikace
3. Aplikace kontaktuje webové API spolu s *consumer key* a přejde do dalšího stavu

Předchozí odstavec se zabýval prvním přihlášením uživatele. Nyní bude popsán proces autorizace, kdy uživatel vložil *consumer key* a stiskl potvrzení. Komunikace se skládá ze čtyř účastníků: *Resource owner*, *Application*, *Resource server* a WNS. *Resource owner* je uživatel, který se autorizuje u *Resource Server* a dává svolení přístupu *Application* k jeho datům. *Application*, též klient, je aplikace, která žádá o přístup k uživatelovým datům. *Resource server* (webové API) je autorita, která zprostředkovává aplikaci přístup k datům. WNS je služba, která pošle notifikaci aplikaci o tom, že se uživatel úspěšně autorizoval. Celý proces je schématicky nakreslen a poté popsán na následujícím obrázku.



Obrázek 4.2: Handshake komunikačního protokolu

Popis jednotlivých kroků autorizace:

- **A - request_token**
  Aplikace pošle žádost o token. Součástí tokenu je *consumer key* a adresa, na které aplikace naslouchá.
- **B - send token**
  Webové API pošle odpověď s tokenem, se kterým bude uživatel přesměrován na rozhraní webového API pro přihlášení.
- **C - redirection**
  Uživatel je společně s tokenem přesměrován na rozhraní webového API pro přihlašování. Zde zadá své přihlašovací údaje.
- **D - request_notif**
  Po úspěšném přihlášení požádá webové API službu WNS o zaslání *push* notifikace.
- **E - send_notif**
  WNS pošle *push* notifikaci do aplikace.
- **F - redirection_back**
  Aplikace obdrží notifikaci a přesměruje uživatele na obrazovku vyzývající k vytvoření pinu.
- **G - req. access_token**
  Aplikace požádá o *access token*. Tento token je vygenerován při přihlášení uživatele na rozhraní webového API. Zpřístupnění dat pomocí *access tokenu* je časové limitované.
- **H - grant access_token**
  Webové API pošle *access token*, se kterým bude aplikace přistupovat datům.
- **I get_data**
  Aplikace pošle požadavek metodou `GET`, kde specifikuje data, která jí mají být zaslána. Součástí požadavku je *access token*.
- **I send_data**
  Webové API ověří *access token* a metodou `POST` pošle zpět požadovaná data.

4.3 Metody webového API

První část podkapitoly obsahuje stručný popis metod, typ HTTPS dotazu, odpovědi a seznam všech parametrů.

Tabulka metod:

název metody	**HTTP dotaz**	**HTTP odpověď**	**popis funkcionality**
check_credentials	GET	POST	kontrola validity aplikace
auth_redirect	GET, POST	POST	autorizace uživatele
get_access_token	GET	POST	zpřístupnění *access token*
get_data	GET	POST	zpřístupnění dat

Seznam všech parametrů metod:

název parametru	**typ HTTP**	**popis funkcionality**
method	GET	název metody, která má být zavolána
app_id	GET	číslo, které identifikuje danou aplikaci
version	GET	verze aplikace
consumer_key	GET	řetězec identifikující uživatele
request_token	GET	přístupový klíč pro autorizaci uživatele
component	GET	název komponenty (graf, dlaždice, tabulka, ...)
id	GET	unikátní číslo určující danou komponentu
access_token	GET	přístupový klíč k datům
notif_channel	POST	notifikační adresa aplikace
uri	POST	adresa aplikace pro Windows Store
cs	POST	přístupový klíč aplikace

Detailnější popis metod:

1. check_credentials

První metoda nejprve zkontroluje identifikátor aplikace a verzi. Obě tyto položky jsou uložené v databázi - tabulka `Auth_apps`. Dále zkontroluje *consumer_key*, který je uložen v tabulce `Auth_UserCred`.

GET parametry: *method, app_id, consumer_id, version*

Vyjma chyb ze strany poskytovatele služby vrací metoda aplikaci jednu ze čtyř odpovědí:

HTTP status	**parametry odpovědi**	**popis funkcionality**
200	id=1&request_token=<*string*>	přístupový klíč uživatele
200	id=2&request_token=<*string*> &possible_upgrade	přístupový klíč uživatele, možná aktualizace
200	id=3&must_upgrade	nutnost aktualizace
200	id=4&app_not_supported	aplikace není podporována

Příklad dotazu:

```
https://www.example.com/dialog/auth.php/?method=check_credentials&app_id=1
    &consumer_key=56ar20&version=1
```

2. check_credentials

Druhá metoda slouží k autorizaci uživatele. Tato metoda obsahuje parametry v `GET` části i `POST`[1] části HTTP zprávy, viz 3.2.

GET parametry: *method, request_token*

[1]POST - channel, uri a client secret - tvoří příliš dlouhý řetězec, který přesahuje délku běžně posílaných dat pomocí metody GET

POST parametry:*notif_channel, uri, cs*

Webové API nevrací odpověď, ale požádá WNS o zaslání notifikace do aplikace. Notifikace je poslána, jen pokud se uživatel úspěšně autorizuje. Obdržená zpráva od WNS má tvar:

HTTP status	**parametry odpovědi**	**popis funkcionality**
200	type=1&OK	autorizace proběhla v pořádku

Příklad dotazu:

```
https://www.example.com/dialog/auth_redirect.php/?method=auth_redirect
    &request_token=49a73c27a30421c650dde7aca2ab005d
    body: { notif_channel = https://msapp.com/42
    uri = https://myapp.com/xyz42xyz
    cs = 123789852 }
```

3. get_access_token

Třetí metoda vrací aplikaci *access token*. Ten je vygenerován po přihlášení na straně webového API a má omezenou dobu platnosti.

GET parametry: *method, request_token*

Vyjma chyb ze strany poskytovatele služby vrací metoda aplikaci *access token*:

HTTP status	**parametry odpovědi**	**popis funkcionality**
200	id=1&access_token=<*string*>	přístupový klíč k datům

Příklad dotazu:

```
https://www.example.com/dialog/auth.php/?method=get_access_token&
    &request_token=49a73c27a30421c650dde7aca2ab005d
```

4. get_data

Čtvrtá metoda vrací aplikaci data ve formátu JSON. Konkrétní data aplikace rozpozná z hodnot parametrů uvedených v `GET` části dotazu.

GET parametry: *method, component, id, access_token*

Vyjma chyb ze strany poskytovatele služby vrací metoda aplikaci jednu ze dvou odpovědí:

Příklad dotazu:

```
https://www.example.cz/data/get\_data.php/?method=get_data&component=tile
    &id=2&access_token=8730d5cee5cfedd4a470e40b493eb0d2
```

HTTP status	parametry odpovědi	popis funkcionality
200	*JSON data*	data pro jednu komponentu
410	access_token_expired	vypršení přístupového klíče k datům

4.4 Aktualizace dat

Při změně dat zažádá webové API službu WNS o zaslání *push* notifikace. V těle notifikace je uveden identifikátor notifikace, typ komponenty a identifikační číslo komponenty. Příklad notifikace je uveden níže:

```
type=2&component=table&id=3
```

Po přijmutí *push* notifikace aplikace požádá webové API pomocí metody `get_data` o data. Jako parametry zvolí údaje, které přišly v těle notifikace (*comphonent* a *id*).

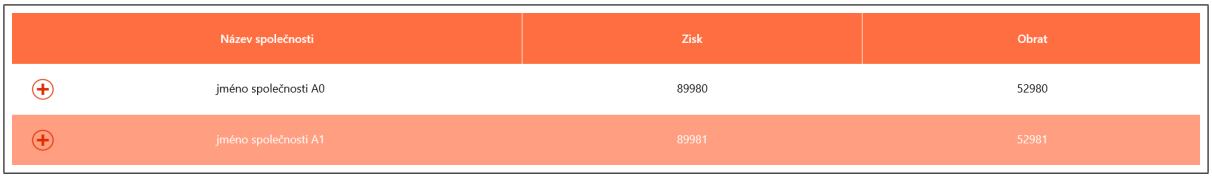
4.5 Struktura přenesených dat

Po zaslání dotazu aplikace na webové API jsou dle příslušného endpointu a metody přenesena data v serializačním jazyce JSON nebo v jazyce HTML. Každá grafická komponenta (graf, tabulka, dlaždice, ...) je na straně webového API zastoupena třídou, která obsahuje data. Data jsou uložena do jednoho pole, které je serializováno a přeposláno aplikaci. Položka pole obsahuje data například pro jeden řádek tabulky, jeden rok v grafu, atd.

Serializované pole pro tabulku *Produktů* o dvou řádcích by vypadala takto:

```
[{"nameCompany":"jméno společnosti A","profit":"89980","veer":"52980",
"year":2013},{"nameCompany":"jméno společnosti A1","profit":"89981",
"veer":"52981", "year":2013}]
```

A výsledná tabulka v aplikaci takto:



Obrázek 4.3: Ukázka tabulky produktů

Mimo tabulky *Produktů* a *Speciální dlaždice* jsou data přenesena vždy po kliknutí na grafickou komponentu v aplikaci. U tabulky *Produktů* je možné rozkliknout každý řádek tabulky a zobrazit detailnější informace. Detailnější informace jsou do aplikace posílány právě až po rozkliknutí. U *Speciální dlaždice* jsou data přenášena ve formátu HTML.

Kapitola 5

Implementace

5.1 Push notifikace

Notifikace můžeme z hlediska implementace rozdělit do dvou částí. První část je implementace na straně webového API v jazyce PHP a druhá část je na straně aplikace v jazyce C#. Fundamentálním zdrojem informací se stala demonstrační videa od sdružení TechEnd Austria 2014. Nejprve byla implementována notifikace v aplikaci. Správné fungování bylo ověřeno testovacím serverem[1]. Tento server není uveden v oficiálních dokumentacích od Microsoftu, avšak mnohé návody se na něj odkazují.

Webové API

Pro kontaktování WNS služby 3.2 potřebuje webové API identifikační údaje aplikace a adresu, na které naslouchá. Identifikační údaje jsou dostupné po zaregistrování aplikace do *Windows Store*. Jedná se o zaregistrování, nikoliv zveřejnění produktu. Pro zaregistrování je nutné disponovat účtem od Microsoftu a mít pro aplikaci název, který je jedinečný v rámci celého *Windows Store*.

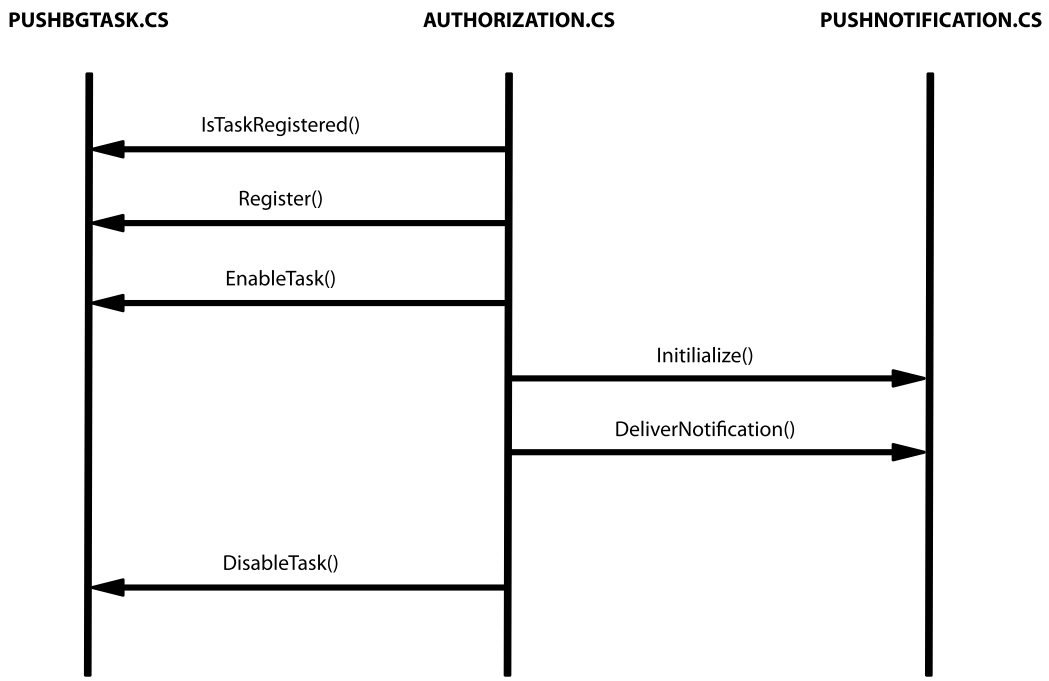
Správu notifikací má na starost třída `WnsNotif`. V konstruktoru této třídy dojde k autorizaci u služby WNS dle protokolu OAuth 2.4.4. Před odesláním žádosti o *request token* je nezbytné získat identifikační údaje o aplikaci z databáze, což je realizováno metodou `SetCredentialsFromDB`. Rozpoznání uživatele proběhne podle `ID` vytaženého z databáze, které je předáno v parametru konstruktoru. Po vytvoření instance má webové API v databázi uložen *access token*, kterým se bude pro zaslání notifikace autorizovat u WNS. K vytvoření instance dojde po přihlášení uživatele na portálu webového API. Přihlašovací formulář společně se stavem po přihlášení je v příloze C.1.

Pro rychlou analýzu případných problémů je kromě informace o úspěšném přihlášení uživatele uveden také výsledek autorizace webového API oproti WNS. Výsledek je HTTPS status, který webové API obdrží po zaslání žádosti o notifikaci. Metoda pro odeslání notifikace je nazvána `SendRawNotif` a v parametru jsou jí předána data, která mají být odeslána do aplikace. V hlavičce HTTPS požadavku je uveden typ obsahu, délka zprávy, typ notifikace a konečně *access token*. Inicializace, odeslání a uzavření požadavku na WNS je uskutečněno vestavěnými metodami PHP `curl_init`, `curl_setopt` a `curl_close`.

[1] http://pushtestserver.azurewebsites.net/wns/

Aplikace

První krok pro úspěšné přijetí *push* notifikace je informovat NCP, o samotné aplikaci. Nejprve je nutné vytvořit projekt typu `Windows Run Time Component`. Projekt je v řešení aplikace nazván `RawPushBGTask` a obsahuje třídu `PushBGTask`. Instance této třídy je vytvořena ve třídě `Authorization`, která je umístěna v projektu `Dashboard.Windows`. Postup aplikace pro příjem notifikací je znázorněn sekvenčním diagramem s podrobnějším popisem níže:



Obrázek 5.1: Postup pro příjem notifikací

Po přesměrování na přihlašovací formulář webového API je ve třídě `Authorization` zavolána metoda `IsTaskRegistered`, která zjistí, zda bylo vlákno pro příjem notifikací již vytvořeno. Vlákno působí mimo aplikaci a při nestandardním vypnutí, například pádu aplikace, se může stát, že se z vlákna stane sirotek. Při každém spuštění aplikace je tedy ověřeno, zda takovéto vlákno již neexistuje. Rozpoznání vlákna se uskuteční podle jména, které uvede vývojář. Neexistuje-li vlákno, pak je zavolána metoda `Register`. Zde je pojmenováno, registrováno a následně vytvořeno asynchronní vlákno, které je spravováno NCP. Poté je ve třídě `Authorization` zavolána metoda `EnableTask`, která spustí zaregistrované vlákno. Současně se zavolá metoda `Initialize`, která vytvoří *channel* (uri identifikátor aplikace). Konečně je zaregistrována událost `sharedPushComponent_deliverNotif` pro příjem notifikací. Je-li aplikace ukončena, pak je zavolána metoda `DisableTask`, která vlákno pro příjem notifikací zruší.

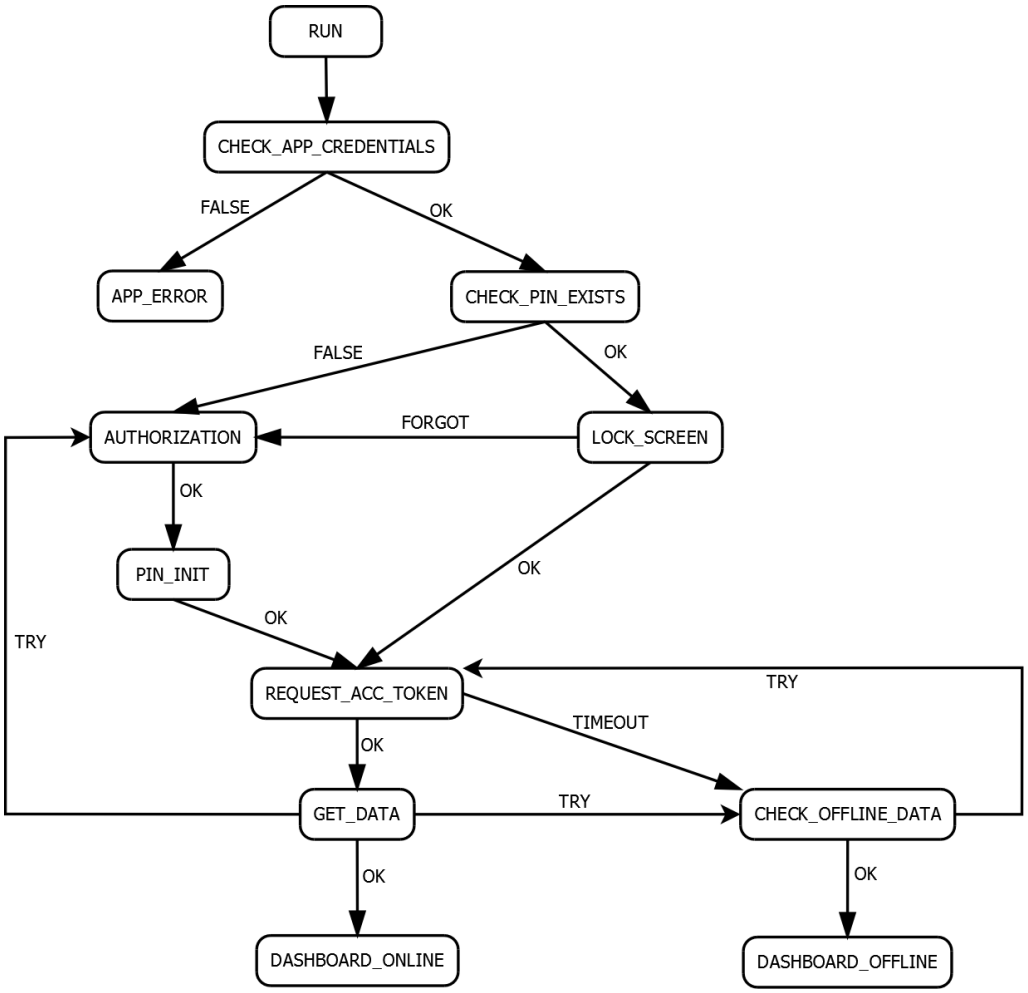
Během implementace se objevil problém, kdy notifikace dorazila do aplikace a u in-

strukcí sloužících k přesměrování uživatele na obrazovku vyzývající k zadání pinu, nastala výjimka. Při spuštění aplikace vznikne standardně vlákno pro vykreslování a interakci s uživatelem, dále jen UI vlákno. Jakmile programátor vytvoří další vlákno a pokusí se přistoupit k proměnným, které vlastní UI vlákno, tak program vygeneruje výše zmíněnou výjimku. Řešením je asynchronně přistoupit k jádru aplikace a požádat o zavolání UI vlákna [4].

V souhrnu *push* notifikace znamenají vytvoření a zaregistrování asynchronního vlákna, které notifikaci dopraví do aplikace. Aplikace využívá notifikace typu *raw*, které umožňují poslat jakýkoliv obsah.

5.2 Autorizace

Následující diagram popisuje průběh autorizace aplikace. Podrobný popis je uveden v textu pod diagramem.



Obrázek 5.2: Diagram průběhu autorizace

Po spuštění aplikace proběhne autorizace v několika krocích. První krok autorizace je na úrovni uživatele, kdy uživatel musí zadat unikátní identifikační řetězec vygenerovaný u autoritativního webového API. Aplikace ověří identifikační kód a přejde do stavu

`Check_app_credentials`, ve kterém jsou zkontrolovány údaje o aplikaci. Ověření validity aplikace je implementováno v metodě `Check_app_credentials`, třída `OAuth`. Dle odpovědi webového API přejde do dalšího stavu. Je-li vše v pořádku, zkontroluje se existence pinu. Metody pro práci s pinem jsou popsány níže v této kapitole. Existuje-li pin, pak je uživateli zobrazena obrazovka s žádostí o zadání pinu. Zde může zadat pin z klávesnice nebo použít virtuální číselník. Metody pro práci s pinem jsou implementovány ve třídě `LockScreen`. Pro zadávání pinu je použita vestavěná komponenta `PasswordBox`. Způsob zadávání pinu byl inspirován u operačního systému Windows. Jakmile uživatel zadá 4 číslice, je pin zkontrolován.
Pokud pin neexistuje, je vytvořena instance třídy `Authorization` a uživatel je přesměrován na přihlašovací portál webového API. Zde zadá své přihlašovací údaje. Aplikace je nyní v pasivním módu. Pro změnu stavu musí obdržet *push* notifikaci. Po obdržení notifikace je uživateli zobrazena obrazovka s žádostí o vytvoření pinu. Aplikace se nachází ve stavu `pin_init`.
Po zadání nebo vytvoření pinu aplikace požádá o *access token*. Tato metoda je opět implementována ve třídě `Oauth` a nese název `GetAccessToken`. Pokud vše proběhne v pořádku, aplikace přechází do stavu `get_data`, ve kterém se zavolá metoda `GetDataFromEndpoint`. Implementace této metody je uvedena v bázové třídě `DataModel`, ze které dědí každý grafický prvek zobrazený v dashboardu. Metodě je předán řetězec, ve kterém je uvedena adresa na příslušný *endpoint*, od něhož se očekávají data. Nepodaří-li se obdržet do 3 vteřin *access token*, aplikace přechází do stavu `check_offline_data`. V tomto bodě se vytvoří instance třídy `OfflineData`. Na základě návratové hodnoty metody `IsOfflineDataAvaiable` dojde/nedojde ke zobrazení offline dat.

5.3 Security Vault

Pin zadaný uživatelem je lokálně uložen tak, aby jej bylo možné získat i po vypnutí a opětovném zapnutí aplikace. Při uložení jsou hodnoty zašifrovány. Šifrování a přístup je v plné moci operačního systému, který využívá stejných služeb při přihlášení uživatele do Windows.

Pro uložení pinu byla vytvořena třída `SecurityVault`. Třída využívá rozhraní `Windows.Security.Credentials`. Rozhraní je podporováno u operačního systému Windows a Windows Phone. Slouží k uložení a správě hesel. Třída disponuje metodou `AddIntoVault`, která vytvoří instanci `PasswordVault`. Pro uložení hesla je nutné obecně specifikovat název aplikace, uživatele a heslo. V našem případě je za uživatele dosazen řetězec "pin"a heslo je uživatelův pin. Údaje jsou zašifrovány a vloženy do bezpečnostního trezoru. Hledat a mazat pin lze podle názvu aplikace a identifikačního řetězce "pin". K tomuto účelu slouží metody `GetValueFromVault` a `DeleteValueFromVault`.

5.4 Offline stav

Inicializace a detekce

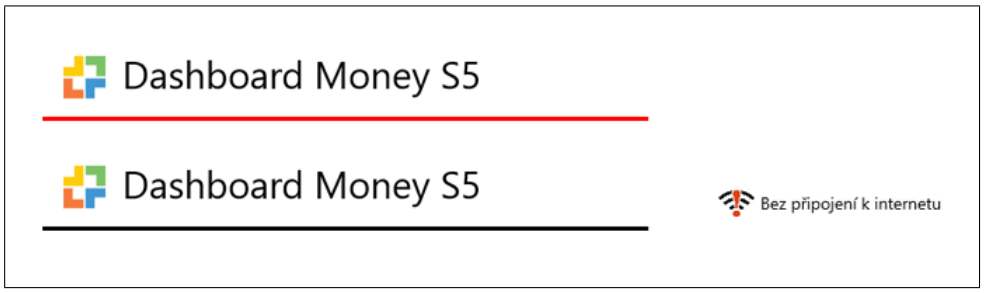
Při spuštění aplikace je vytvořena instance třídy `CheckConnection`, která obsahuje metody pro identifikaci připojení k internetu. V konstruktoru je inicializován a spuštěn časovač `DispatcherTimer`. Instance časovače vytvoří vlákno, jenž v určitých intervalech kontaktuje

hlavní vlákno, které v metodě `timer_Tick` vykoná požadované instrukce. Interval je v aplikaci nastaven na 5 vteřin. Každých 5 vteřin je tedy zkontrolováno, zda je aplikace připojena k internetu. Tomuto účelu slouží metoda `TryConnection`.

První verze detekce připojení bylo zaslání HTTP dotazu na jeden z endpointů webového API a pokud nepřišla odpověď do deseti vteřin, přešla aplikace do stavu offline. Tento způsob zbytečně zatěžuje zařízení, a také zvyšuje objem přenesených dat. Metoda byla optimalizována a v konečné verzi využívá vestavěných metod `NetworkInformation`, které kontrolují, zda je zařízení připojeno k nějakému profilu. Komunikace tedy probíhá výhradně uvnitř zařízení mezi aplikací a operačním systémem.

Upozornění uživatele

Přejde-li aplikace do offline stavu, je o tom uživatel informován několika změnami. První rozdíl je záměna červené lišty na šedou. Současně se v pravém horním rohu objeví pulzující ikona wifi s informativním textem. Efekt pulzování byl docílen animací `DoubleAnimation`, která se po vytvoření vloží do `StoryBoard`. `StoryBoard` je kontejner, do kterého je možné přidat více animací a tyto animace poté jednoduše spustit metodou `Begin`. Animovaná vlastnost objektu (ikony) je průhlednost. Příklad změny schématu a objevení ikony je uveden níže.



Obrázek 5.3: Online a offline stav hlavní lišty

Tato změna se projeví v záhlaví aplikace a je stejná pro všechny zobrazené stránky. Po kliknutí například na dlaždici je vpravo nad hlavním obsahem uvedeno časové razítko poslední aktualizace. Objeví se také komponenta `ProgressBar`, která má u uživatele vyvolat dojem, že se aplikace snaží o připojení.

Uložení do souborů

Každá grafická komponenta (tabulka, dlaždice, graf, ...) je uložen do samostatného souboru. Pokud aplikace obdrží data z webového API, tak jsou tato data zpracována a uložena v metodě `SaveDataToFile`. Metoda využívá rozhraní `Windows.Security.Cryptohgraphy`, které převede data do binárního tvaru a pomocí `Windows.Security.Storage` uloží do souboru. Vždy, když dojde k přijetí dat z webového API, je otevřen soubor, který sdružuje poslední aktualizace dané komponenty. V obsahu souboru je nalezena komponenta, která byla právě aktualizována a její čas aktualizace se přepíše. Následně dojde k uložení nového obsahu do souboru. Aplikace se snaží získat *access token*, aby mohla zobrazit aktuální data. O získání přístupového klíče se snaží samostatné vlákno (pokud je detekován režim online), které je vždy na 5 vteřin uspáno.

Kapitola 6

Rozšíření

6.1 Universal Apps

Současná aplikace Money Dashboard S5 je určena pro osobní počítače a tablety, které disponují operačním systémem Windows 8 a vyšším. Po analýze možností vývoje aplikace pro prostředí *WinRT* bylo zjištěno, že existuje technologie Universal apps. Tato technologie byla přidána do Visual studio v druhé aktualizaci, která proběhla v květnu 2014.

Implementace aplikace je rozdělena do třech projektů: `Windows, Windows Phone` a `Shared`. Zdrojový kód v projektu `Shared` je sdílen mezi dva zbývající projekty. V praxi se sdílí chování aplikace. Vzhled je upraven pro každý operační systém zvlášť. Nejedná se o razantní změny a ve většině případů stačí změnit velikost a uspořádání ovládacích prvků. Minoritní část případů zastupují prvky, které jsou rozdílné u obou skupin. Detailnější výpis skupin prvků a jejich specifika jsou uvedena níže:

- Běžné: Tyto prvky lze použít pro obě zařízení a jejich vzhled je identický (*Button, CheckBox, Slider*)
- Optimalizované: Tyto prvky lze použít pro obě zařízení, avšak jejich vzhled je modifikován pro každou platformu (*DatePicker, TimePicker*)
- Jedinečné: Tyto prvky jsou pro každé zařízení odlišné a to hlavně z důvodu různého ovládání prvku (GridView, Pivot, Hub)

Chceme-li přeložit určitou část zdrojového kódu pro každý operační systém zvlášť v projektu `Shared`, pak je nutné řídit překlad speciální direktivou:

```
#IF WP_8
#END IF
```

6.2 Živé dlaždice a interakce s uživatelem

Živé dlaždice

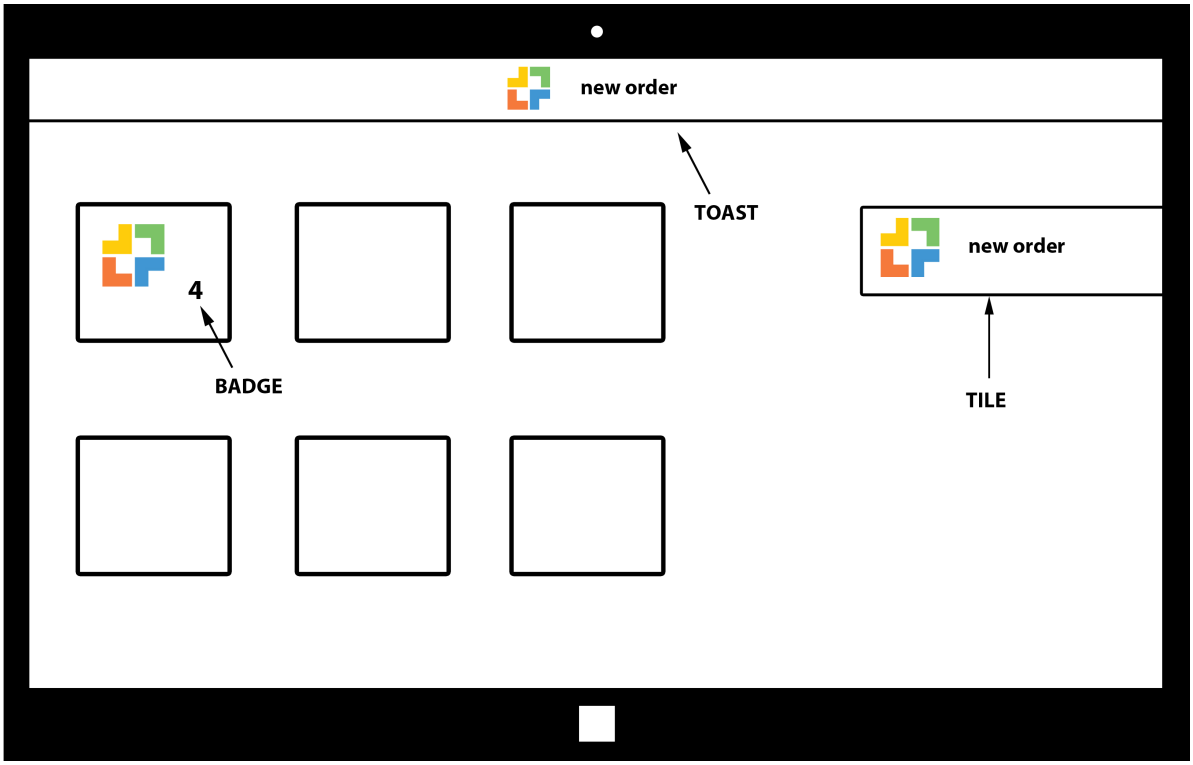
Podkapitola se zabývá možnostmi zobrazení živé dlaždice a následně jsou popsány všechna možná upozornění uživatele. „Dlaždice v novém prezentačním rozhraní Windows 8.1 umožňuje uživateli spustit aplikaci nebo se přepnout do již spuštěné aplikace. Na rozdíl od jiných platforem to nejsou jen statické ikony, ale dokážou v souladu s účelem příslušné aplikace, kterou na domovské obrazovce zastupují, dynamicky zobrazovat rozmanitý informační, případně ilustrační, obsah, a to i tehdy když aplikace není spuštěna“ [6]. Podmínka pro fungování živé dlaždice je její přítomnost na úvodní obrazovce prostředí *Modern UI*. Zobrazované obrázky mohou mít formát `JPEG` nebo `PNG` a nesmí přesáhnout 150 kB. Tvar dlaždice je striktně omezen na obdélníkový či čtvercový. Nastavení velikosti, tvaru a obsahu je možné v aplikačním manifestu `Package.appxmanifest`. Obsah dlaždice může být i obrázek, jehož velikost je různá v závislosti na velikosti obrazovky. Jejich měřítko je vyjádřeno procentuálně, konkrétně 80, 100, 140 a 180 procent základního rozměru. Všechny obrázky určené k tomuto účelu jsou uložené ve slože *Assets*. Z hlediska interakce uživatele je dlaždice zcela pasivní. To znamená, že obsah, který zobrazuje, uvidí uživatel jen v případě, pokud přejde na úvodní obrazovku.

Interakce s uživatelem

V *Modern UI* prostředí může být uživatel upozorněn na změnu stavu programu notifikací. Notifikace můžeme rozdělit do dvou kategorií. První kategorie je způsob doručení, druhou je způsob zobrazení. Způsob doručení je rozdělen na: *local*, *scheduled*, *periodic* a *push* 5.1. Způsob zobrazení je rozdělen na: *tile, badge, toast* a *raw*. Obě kategorie jsou úzce spjaty. Jejich vzájemný vztah popisuje tabulka níže:

Způsob doručení	**Způsob zobrazení**	**Popis**
local	tile, badge, toast	aplikace musí běžet. Vhodné například pro upozornění na nově přehrávanou písničku.
scheduled	tile, toast	aplikace také musí běžet, ale upozornění je zobrazeno v určitý čas. Vhodné pro aplikace pracující s časem (kalendáře, budíky).
periodic	tile, badge	aplikace musí běžet a dotazuje se serveru na změny. Vhodné například pro počasí nebo denní zprávy.
push	tile, badge, toast, raw	aplikace nemusí běžet. Uživatel je upozorněn ze serveru, například emailový klient.

Možnosti zobrazení notifikace[1]



Obrázek 6.1: Ukázka možných způsobů zobrazení notifikací

[1]RAW umožňuje zaslat jakýkoli řetězec a uživatel o ní není informován. Ostatní zobrazení vyžadují zápis v XML.

Kapitola 7

Závěr

Cílem této práce bylo navrhnout a implementovat Windows 8 aplikaci pro prezentaci dat z webového API. Uživatelé mohou sledovat data z různých, především finančních, kategorií v reálném čase. Velký důraz byl kladen na minimalizaci přenesených dat a bezpečnost. Finální řešení se skládá ze tří část: klientské aplikace, webového API a databáze. Aplikace zobrazuje citlivá data, přičemž jejich zdroj je právě databáze. Produkt bude umístěn v distribuční platformě *Windows Store*. Pracovat s ní budou osoby disponující účtem u firmy Cígler Software. Pro účely bakalářské práce bylo vytvořeno webové API a později bude nahrazeno proprietárním. Cílový produkt je určen pro zařízení s operačním systémem Windows 8 a vyšším. Dnes to mohou být tablety a přenosné/stolní počítače.

Autorizace je založena na protokolu OAuth 2.0, data jsou zabezpečena protokolem HTTPS. Za majoritní úspěch považuji implementaci notifikací do aplikace. Aplikace je tak výpočetně méně náročná, čímž šetří výkon a baterii zařízení. Webové API momentálně nemůže být označeno jako *RESTfull*, protože neuvádí zda klient data může ukládat do mezipaměti a také neuvádí, jak zpracovat serializovaná data.

V důsledku nahrazení webového API bude přizpůsoben komunikační protokol. Pro ověření správného zobrazení byl využit vestavěný simulátor *Visual Studia*. Činnost a grafická podoba aplikace byla konzultována se zástupcem zadavatele. Komunikační protokol byl navrhnut a implementován po konzultacích s vedoucím bakalářské práce.

Do budoucna se plánuje rozšíření aplikace pro Windows Phone. Produkt byl vyvíjen jako *Universal Apps*, což umožňuje sdílet většinovou část kódu mezi obě platformy (Windows 8 a Windows Phone).

Literatura

[1] ECMA-404 The JSON Data Interchange Standard. [online]. http://www.json.org/json-cz.html, 2015 [cit. 2015-01-17].

[2] Avram, A.: Design Details of the Windows Runtime [online]. http://www.infoq.com/news/2011/09/Design-Details-Windows-Runtime, 2011-09-21 [cit. 2015-18-01.

[3] Burget, R.: Struktury a kolekce, 2D reprezentace - vizualizace [online]. https://www.fit.vutbr.cz/study/courses/WAP/private/opory/IIS0502D.pdf, 2014-11-04 [cit. 2015-10-03].

[4] Freeman, A.: *Metro Revealed: Building Windows 8 apps with XAML and C#*. Apress, 2012, iSBN 978-1-4302-4491-2.

[5] István Novák, Z. A. D. F., György Balássy: *Begining Windows 8 Application Development*. John Wiley & Sons, Inc., 2012, iSBN 978-1-118-01268-0.

[6] Ľuboslav Lacko: *Vývoj aplikací pro Windows 8.1 a Windows Phone*. COMPUTER PRESS, 2014, iSBN 978-80-251-3822-9.

[7] Malý, M.: OAuth ? nový protokol pro autentizaci k vašemu API [online]. http://www.zdrojak.cz/clanky/oauth-novy-protokol-pro-autentizaci-k-vasemu-api, 2008-11-25 [cit. 2015-21-02].

[8] McVetta, J.: What is a RESTful API? [online]. http://advanced-python.readthedocs.org/en/latest/rest/what-is-rest.html, 2012 [cit. 2015-27-02].

[9] MSDN: Windows Push Notification Services (WNS) overview (Windows Runtime apps) [online]. https://msdn.microsoft.com/cs-cz/library/hh913756.aspx, [cit. 2015-20-03].

[10] Olson, J.: Reimagining App Development with the Windows Runtime [online]. https://msdn.microsoft.com/en-us/magazine/jj651567.aspx, 2015 [cit. 2014-27-12].

[11] Parecki, A.: OAuth 2 Simplified [online]. https://aaronparecki.com/articles/2012/07/29/1/oauth2-simplified, 2012-06-29 [cit. 2015-20-02].

[12] Samaras, C.: Differences Between WPF And Windows Forms [online]. http://www.myengineeringworld.net/2014/08/wpf-and-windows-forms-differences.html, 2014-08-27 [cit. 2015-19-01].

Příloha A

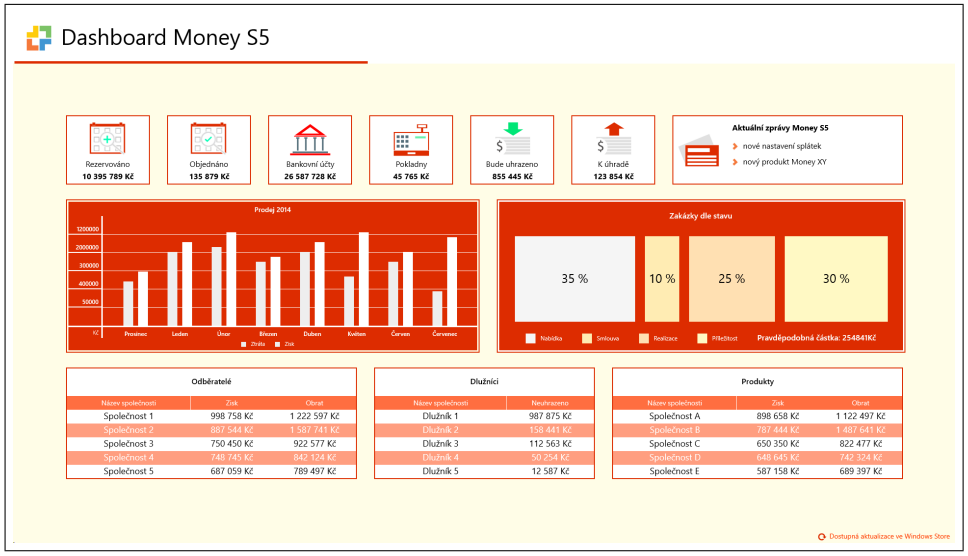
Obsah CD

Adresářová struktura CD

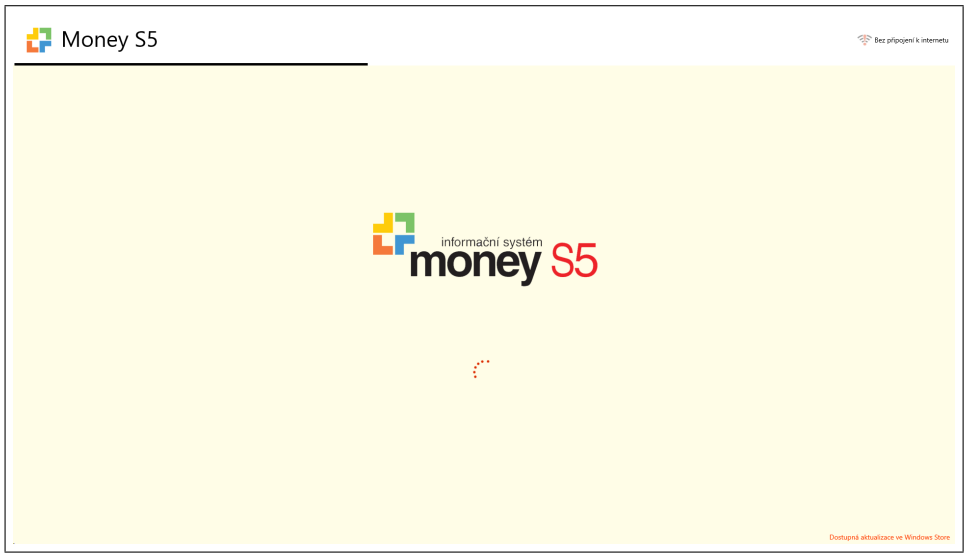
- `Source`
  - `Dashboard` - zdrojové kódy pro aplikaci
  - `WebApi` - zdrojové kódy pro webové API
  - `DB` - exportovaná MySQL databáze
- `OfflineData` - soubory s offline daty
- `Icons` - logo a ikony použité v aplikaci
- `Thesis` - zdrojové soubory LaTeXa `Makefile`
- `Manual` - PDF s návodem na instalaci a spuštěním aplikace

Příloha B

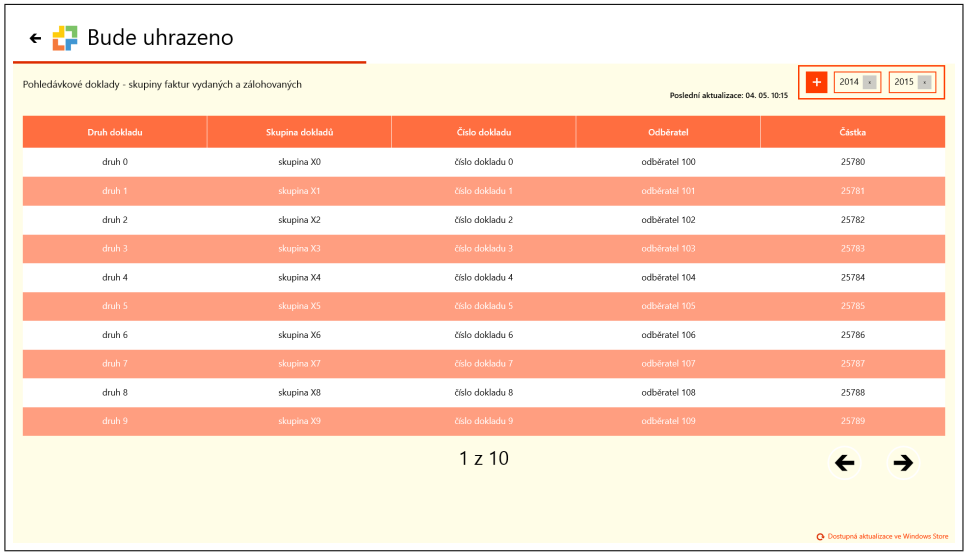
Screenshoty z aplikace



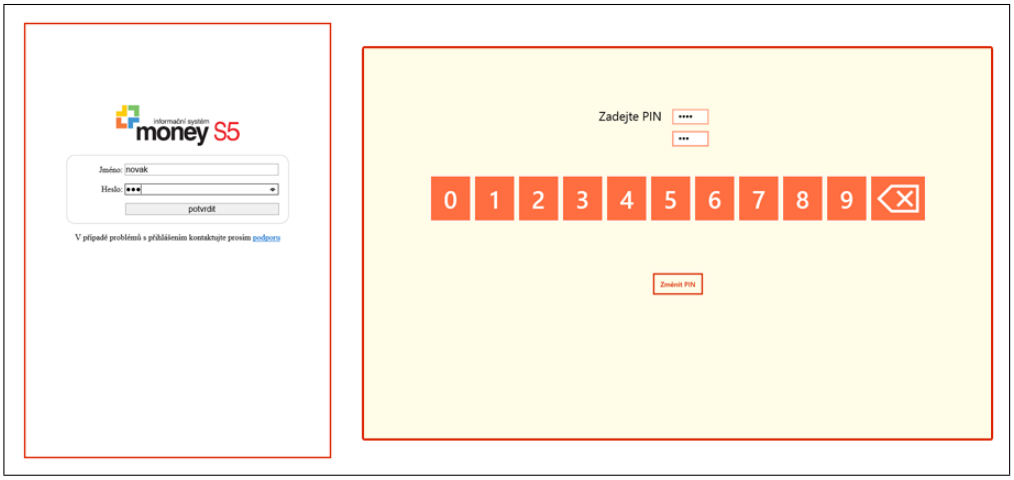
Obrázek B.1: Hlavní obrazovka aplikace



Obrázek B.2: Aplikace byla spuštěna bez připojení k internetu



Obrázek B.3: Tabulka neuhrazených faktur



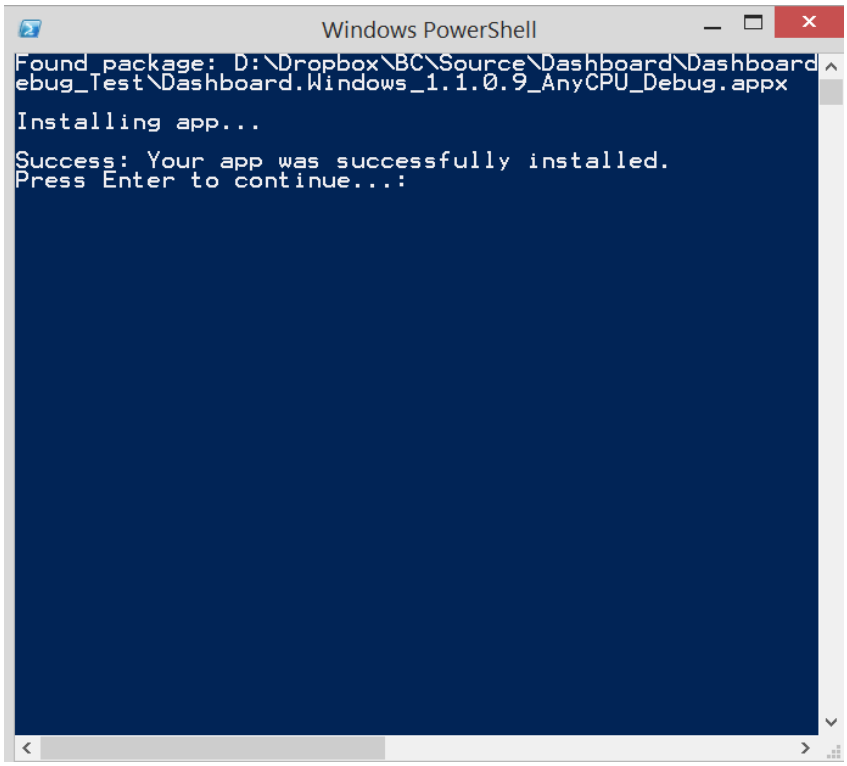
Obrázek B.4: Přihlašovací formulář a formulář pro zadání pinu

Příloha C

Manuál

Instalace aplikace

1. Aplikace podporuje pouze Windows 8 a vyšší.
2. V adresáři (viz A) `Instalace` je soubor `Add-AppDevPackage.ps1`. Tento soubor je nutné otevřít a spustit. Nejlépe kliknutím pravého tlačítka myši dát *Run with Powershell*.
3. Otevře se dialogové okno, ve kterém se spustí instalace.



Obrázek C.1: Přihlašovací formulář a formulář pro zadání pinu

4. Pro dokončení instalace stačí stisknout klávesu *Enter*. Pro spuštění je nutné přejít do prostředí *Modern UI* a aplikaci vyhledat.

Instalace webového API a databáze

1. V adresáři (viz A) `Source/WebApi/rest` naleznete zdrojové kódy pro webové API.
2. Obsah adresáře `rest` je nutné vystavit veřejně/lokálně (webové API, které bude naslouchat).
3. Adresu, na které je webové API uloženo musí být změněno v souboru `Endpoint.txt` v adresáři `Source/Aplikace/Dashboard/Dashboard.Shared`.
4. Dále je nezbytné, aby se provedl import MySQL databáze. Zdrojové kódy naleznete v adresáři `Source/DB`.
5. Konečně v souboru `connect.php` je nutné změnit adresu a přístupová práva databáze. Soubor lze nalézt v adresáři `Source/WebAPI/rest/dialog`.