Možnosti přenosu dat z mobilních telefonů s různými platformami do centrální databáze

Data transfer capabilities from mobile devices with different platforms to a central database

Petr Mikunda

Bakalářská práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr MIKUNDA**
Osobní číslo:	**A07068**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Téma práce:	**Možnosti přenosu dat z mobilních telefonů s různými platformami do centrální databáze**

Zásady pro vypracování:

1. **Zpracujte literární rešerši na dané téma.**
2. **Navrhněte strukturu aplikace pro přenos dat mezi mobilním zařízením a serverem.**
3. **Vytvořte aplikaci pro přenos dat na různých platformách (Windows Mobile, Android, Apple iOS).**
4. **V práci se zaměřte na možnosti zabezpečení přenášených informací.**
5. **Zhodnoťte jednotlivé platformy podle dosažených výsledků.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **MEIER, Reto. Professional Android? Application Development. , Indianapolis, Indiana : Wiley Publishing, Inc., 2009. 439 s. ISBN: 978-0-470-34471-2.**
2. **MEIER, Reto. Professional Android? 2 Application Development, Indianapolis, Indiana : Wiley Publishing, Inc., 2010. 576 s. ISBN: 978-0-470-56552-0.**
3. **BERTINO, Elisa, MARTINO, Lorenzo D., PACI, Federica, SQUICCIARINI, Anna C. Security for Web Services and Service-Oriented Architectures, 2010, Springer, 218 s. ISBN 978-3-540-87742-4.**
4. **ALI, Maher. iPhone SDK 3 Programming. Wiley Publishing, Inc., 2009, 672 s. ISBN 978-0-470-68398-9.**
5. **PETZOLD, Charles. Programming Windows Phone 7, Microsoft press, 2010, 1012 s. ISBN 978-0-7356-4335-2**

Vedoucí bakalářské práce: **Ing. Jiří Pálka**
Ústav elektroniky a měření

Datum zadání bakalářské práce: **25. února 2011**

Termín odevzdání bakalářské práce: **7. června 2011**

Ve Zlíně dne 25. února 2011

prof. Ing. Vladimír Vašek, CSc.
děkan

L.S.

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Cílem bakalářské práce bylo nalézt bezpečné způsoby přenosu dat z mobilního zařízení do centrální databáze a vytvořit testovací aplikace aplikovatelné na mobilní zařízení. Operační systémy pro mobilní zařízení jsou Android, iOS a Windows phone 7.

V teoretické části jsou vysvětleny pojmy jako SOAP, REST, JSON apod. Dále jsem zmínil všechny operační systémy a jejich programovací jazyky a vývojová prostředí.

Klíčová slova: WCF, SSL, TLS, HTTPS, Android, WP7, MAC, xCode, Java, C#, SOAP, REST, WSDL, Eclipse, Microsoft Visual studio, Webové služby, IIS, JSON, HTTPS

ABSTRACT

Main goal of this bachelor thesis was find secure possibilities of how to transfer data from mobile device to central database and create test applications applicable to mobile devices. Operating systems for mobile devices are Android, iOS and Windows Phone 7.

In theoretical part are explain terms such us SOAP, REST, JSON, etc. Furthermore, I mentioned all operating systems and their programming languages and development environments.

Keywords: WCF, SSL, TLS, HTTPS, Android, WP7, MAC, xCode, Java, C#, SOAP, REST, WSDL, Eclipse, Microsoft Visual Studio, Web-services, IIS, JSON, HTTPS

Děkuji Ing. Jiřímu Pálkovi za možnost realizace této bakalářské práce a také Janu Pálkovi za pomoc s realizací.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….……

podpis diplomanta

OBSAH

ÚVOD **9**

I TEORETICKÁ ČÁST **10**

1 POPIS PLATFORM **11**

1.1 ANDROID 11

1.1.1 Vývojové prostředí Eclipse 11

1.1.2 Android aplikace 13

1.1.2.1 Základy Android 13

1.2 WINDOWS PHONE 7 16

1.2.1 Vývojové prostředí Visual studio 17

1.3 OBJECT-C - IPHONE 17

1.3.1 Vývojové prostředí xCode 18

1.3.1.1 Struktura iOS aplikace 18

2 KOMUNIKACE **20**

2.1 WINDOWS COMMUNICATION FOUNDATION 20

2.1.1 Sjednocení technologií 21

2.1.2 Interoperabilita 21

2.1.3 Orientace na služby 22

2.2 ZABEZPEČENÍ KOMUNIKACE 22

2.2.1 HTTPS 23

2.2.1.1 SSL/TLS 23

2.3 ZPROSTŘEDKOVÁNÍ KOMUNIKACE 24

2.3.1 SOAP 24

2.3.1.1 ksoap2 25

2.3.2 REST 26

2.3.3 Komunikační formáty 26

2.3.3.1 XML 26

2.3.3.2 JSON 27

2.3.3.3 Atom/RSS 27

II PRAKTICKÁ ČÁST **29**

3 STRUKTURA APLIKACÍ **30**

3.1 WCF A WEBOVÉ SLUŽBY 31

3.1.1 Vytvoření webové služby 31

3.1.1.1 Konfigurace WCF 32

3.1.1.2 Interface/Contract 34

3.1.1.3 Implementace metod kontraktu 35

3.2 ANDROID APLIKACE 37

3.2.1 Aplikace pro stažení přes zabezpečený kanál 37

3.2.1.1 Počáteční problémy 37

3.2.1.2 Popis aplikace - Downloader 38

3.2.1.3 Kód aplikace - Downloader 41

3.2.1.4 Popis aplikace - SmsCentrum 46

3.2.1.5 Kód aplikace - SmsCentrum 49
3.3 WINDOWS PHONE 7 50
3.4 IPHONE APLIKACE 51
3.4.1 Downloader 51
3.4.2 Počáteční problémy 51
3.4.3 Popis aplikace - Downloader 52
3.4.4 Kód aplikace - Downloader 52
4 ZHODNOCENÍ ROZDÍLŮ 56
4.1 VÝVOJOVÁ PROSTŘEDÍ 56
4.2 PROGRAMOVACÍ JAZYK 56
4.3 VYTVÁŘENÍ APLIKACÍ 57
4.4 KOMUNIKACE SE SLUŽBOU 57
4.4.1 SOAP vs REST 58
ZÁVĚR 59
CONCLUSION 60
SEZNAM POUŽITÉ LITERATURY 62
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 63
SEZNAM OBRÁZKŮ 64
SEZNAM PŘÍLOH 65

ÚVOD

Bakalářská práce je zaměřena na bezpečnost komunikace mezi mobilním zařízením a centrální databází. Pro testování přenosu byly vybrány nejvíce používané platformy pro mobilní zařízení. Mezi nejpoužívanější platformy v dnešní době patří operační systém Android od společnosti Google, Windows phone 7 od společnosti Microsoft a také iOS od společnosti Apple.

Pro každou platformu byla vytvořena i drobná testovací aplikace, která komunikuje se službou, která k tomuto účelu byla také vytvořena. Možností zabezpečení komunikace existuje více, ale nejvíce se v dnešní době používá možností protokolu SSL. Stejně tak jsem jeho možností využil i já. Konkrétně protokolu HTTPS, který je podporován v dnešní době na všech platformách, operačních systémech a zařízeních určených pro běžnou lidskou činnost.

Mobilní zařízení komunikují se službou WCF. Jedná se o distribuovanou technologii od společnosti Microsoft. Tato technologie má oproti všem předchozím mnoho výhod jako například interoperabilitu, změna nastavení komunikace služby bez potřeby změny kódu nebo naprostého předělání aplikace jak tomu bývalo u starších distribuovaných technologií atd.

I. TEORETICKÁ ČÁST

1 POPIS PLATFORM

1.1 Android

Android je softwarová platforma založená na Linuxu pro mobilní zařízení (mobilní telefony, PDA, navigace,...)., která je původně vyvinuta společností Android Inc., kterou akvizicí převzala společnost Google v roce 2005. Vývoj aplikací se provádí za pomoci Android SDK, které umožňuje vývojářům psát aplikace v jazyce Java s využitím knihoven vyvinutých společností Google.

Přestože je Android postaven na jazyce Java, nevyužívá žádného ze standardů jako je Java SE nebo pro mobilní zařízení určený Java ME. Rovněž virtuální stroj pro běh programů napsaných v jazyce Java s názvem Dalvik má odlišnou architekturu od standardního Java virtuálního stroje. Dalvik kromě nové architektury stroje (který principielně nemůže spouštět standardní byte code Javy) staví na podmnožině knihoven projektu Apache Harmony, což je open source projekt, který je přepisem technologií Java jako open source. Důsledkem takového řešení je velká nezávislost na původních licencích, které si s sebou původní jazyk i prostředí Java nese.

1.1.1 Vývojové prostředí Eclipse

Jak již název napovídá, tak vývojovým prostředím pro Android aplikace a widgety je Eclipse. Pro vytváření aplikací je potřeba ještě doinstalovat JDK (Java Development Kit). Poté je potřeba stáhnou balíčky pro vytváření aplikací v Androidu (viz. Návod instalace ADT). Jakmile se balíčky stáhnout, tak je potřeba nastavit ve vývojovém prostředí cestu k těmto balíčkům a to se provede přes záložku *Window -> Preferences* v levém sloupci se vybere možnost *Android* a zde se nalezne cesta ke staženým balíčkům ADT. SDK je postupně aktualizováno a je v něm možné nalézt kromě samotných balíčků i emulátor pro testování vytvořených aplikací.

Každá aplikace má svou strukturu a stejně je tomu i u Androidu. Po vytvoření nového projektu lze vidět strukturu aplikace, která by měla být dodržena. Design je oddělen od kódu, stejně tak i proměnné lze ukládat do samostatného souboru. Design i ukládání proměnných je uděláno pomocí XML souborů, které jsou upraveny a mají svou vlastní strukturu.

Eclipse jako vývojové prostředí neslouží pouze k vytváření android aplikací, ale jem ožné do něj instalovat další pluginy, které moho sloužit k tvorbě dalších Java aplikací. Eclipse je v dnešní době velmi používaným nástrojem v mnohých firmách. Hlavní výhodou je multiplatformnost tohoto prostředí další výhodou je cena - Eclipse je totiž zdarma. Všechny tyto tři faktory - rozšiřitelnost, multiplatformnost a cena - jsou pro mnohé firmy velmi lákavé a proto je i jazyk Java velmi rozšířen a programuje v něm mnoho firem po celém světě. Eclipse se používá hlavně pro vytváření Java webových aplikací a samozřejmě i běžných aplikací. Stejně jako Visual Studio je to velmi silný nástroj pro tvorbu jakýchkoliv aplikací a díky své rozšiřitelnosti a množství dostupných pluginů je i velmi využívaný a oblíbený. V Eclipse je navíc možné upravit sirozmístění potřebných oken a záložek dle vlastního uvážení a proto pokud vyhovuje někomu rozmístění oken např. ve VS může si vše rozmístit stejně.

Já jsem k vytváření aplikací používal nejdřívě verzi 10.5 s označením Europa, ale později vyšla novější verze 3.6.1 s označením Helios. Nejdříve bylo možné využívat obě verze, ale později Google zrušil možnost využívání starší verze a nyní je možné stáhnout balíčky ADT i SDK pouze pro verzi Eclipse Helios.

Návod instalace ADT (Eclipse 3.6 Helios):

- Po spuštění Eclipse jděte do záložky *Help -> Install new software...*
- Kliknout na *Add* v pravém horním rohu.
- V dialogu *Add Repository* vložte název ADT pluginu do kolonky *Name* a následující URL do kolonky *Location*:

```
HTTPS://DL-SSL.GOOGLE.COM/ANDROID/ECLIPSE/
```

  klikněte na *OK*

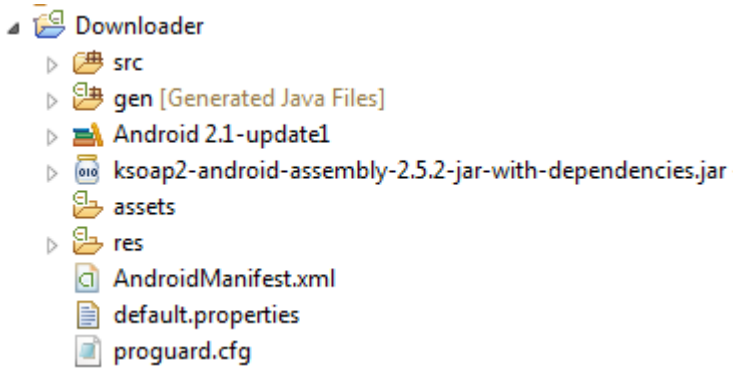
- V dialogi *Available Software* vyberte checkbox vedle *Developer Tools* klikněte na *Next*
- V dalším okně vidíme seznam dostupných nástrojů, které budou staženy. Klikněte na *Next*
- V dalším okně jsou smluvní ujednání, která by si přečtěte a potvrďte a poté klikněte na *Finish*

1.1.2 Android aplikace

Aplikace na Android, jak již bylo řečeno výše, se programuje v jazyce Java za pomocí balíčků od Google. Nad aplikací pro Android jsem strávil nejvíce času a to z toho důvodu, že jako na jediné platformě mám možnost ji otestovat na reálném zařízení. Právě z tohoto důvodu jsem vytvořil v aplikaci možnost získávat data jak pomocí SOAP tak pomocí REST. Oba způsoby komunikace jsou implementovány v jedné webové službě.

1.1.2.1 Základy Android

Android má svou vlastní strukturu ve které má oddělen design od kódu, který obstarává chod aplikací a stejně tak je i možné oddělit texty či konstanty od kódu či designu a vložit je do samostatného XML souboru, který pro tyto účely slouží. [Obr. 2.]



Obrázek 1: Struktura aplikace v Android

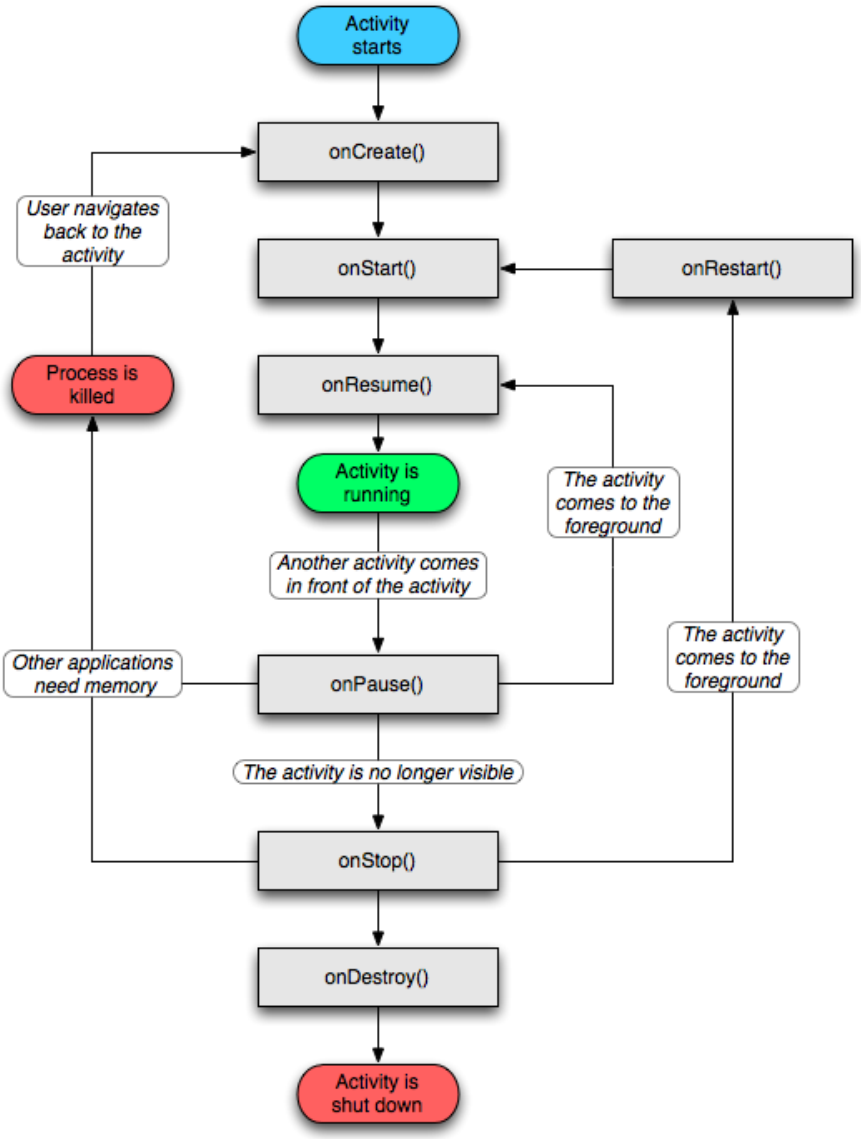
1.1.2.1.1 Popis struktury Android aplikace

- src - hlavní složka, která obsahuje balíčky ve kterých jsou uloženy třídy
- gen - složka obsahující soubor R.java, ve kterém se ukládají veškeré id prvků, uložené texty atd.
- Android 2.1-update1 - balíček android od Google, liší se podle verze pro kterou aplikaci vytváříme
- ksoap2-android-assembly-2.5.2-jar-with-dependencies.jar - jedná se o externí balíček, který využívám pro komunikaci skrze SOAP - android v sobě samém žádnou SOAP knihovnu neobsahuje

- assets - obsahuje další soubory potřebné k chodu aplikace, android negeneruje ID souborů uložených zde, proto se musí načítat manuálně
- res - složka obsahující zdroje tzn. ikony různých velikostí, texty, šablony pro tvorbu designu aktivit, menu, preferencí atd. odtud se generují odkazy do složky gen
- Android manifest - XML soubor, který slouží jako konfigurační soubor. Každá vytvořená aktivita, služba, reciever atd. se musí uložit také zde, stejně tak se zde nastavují i různá práva - jedná se o nejdůležitější soubor v androidu

1.1.2.1.2 Aktivity

Aktivita je třída sloužící k obsluze UI neboli uživatelského rozhraní. Díky aktivitám je uživatel schopný na Androidu ovládat grafické rozhraní. Každá obrazovka vyskytující se v aplikaci bude dědit z třídy Activity[2]. Aktivita má samozřejmě pevně danou strukturu. Ovšem jediná metoda, která je vyžadována při vytvoření aktivity je metoda OnCreate(Bundle).

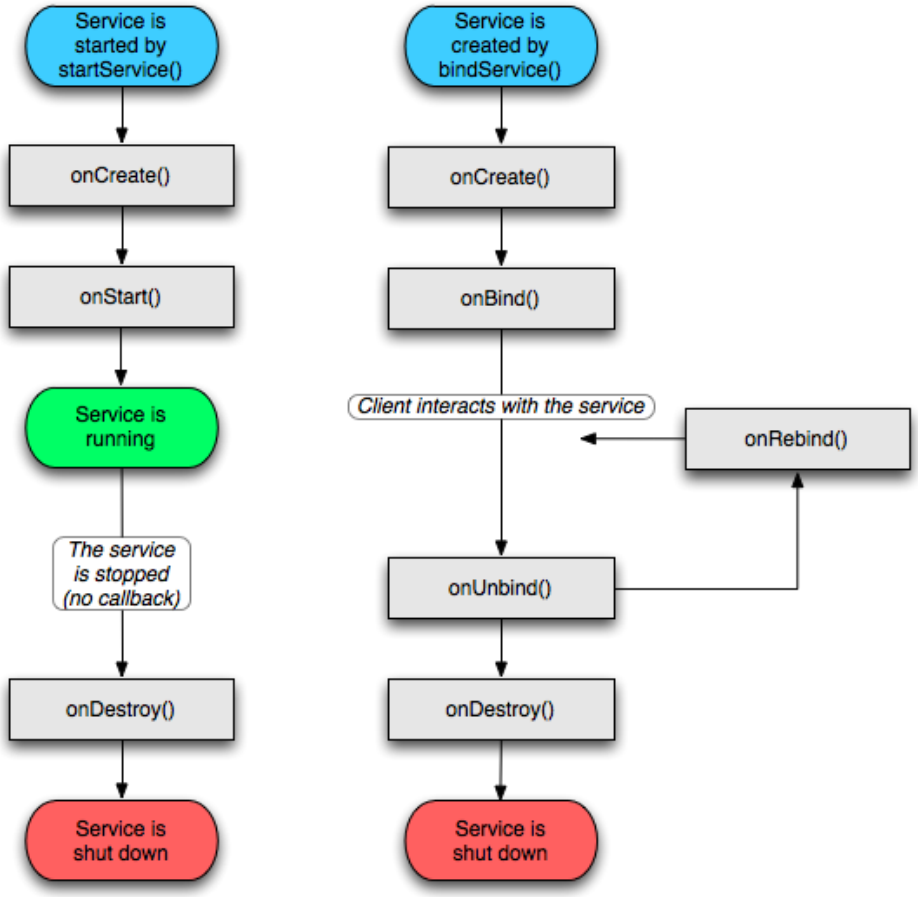


Obrázek 2: Životní cyklus aktivity

1.1.2.1.3 Služby

Služba je v Android brána jako nějaký proces, který se provádí v pozadí jako např. aktualizace dat nebo rozhraní, atd. Služba se volá podobným způsobem jako aktivita a má i podobnou stavbu. Rozdíl mezi aktivitou a službou je právě v tom, že aktivita obsluhuje

výhradně rozhraní a vykonává požadavky uživatele zatímco služba pracuje v pozadí a její funkce nezávisí na uživateli. Moderní mobilní zařízení jsou většinou multifunkční zařízení, ale bohužel kvůli velikosti svého displeje mohou obsluhovat pouze jednu interaktivní aplikaci.[1]



Obrázek 3: Životní cyklus služby

1.2 Windows Phone 7

Windows phone 7 je nejnovější platforma od firmy Microsoft. Celý operační systém je postaven na technologii silverlight, který má aktuálně označení verze 4. WP7 je kompletně předělaný systém, který nemá se starou verzí Windows mobile téměř nic společného.

Celý WP7 je zpracován mnohem lépe nejen po grafické stránce, ale i po stránce programové. Bohužel je WP7 uzavřený systém podobně jako operační systém od Applu a proto je problém s programováním vlastních widgetů nebo aplikací.

Aplikace a widgety na WP7 se programují pomocí jazyka C# v kombinaci se Silverlight. Silverlight je poměrně nová technologie vytvořená Microsoftem, která je ještě stále ve vývoji, protože stále nedokáže pracovat se všemi technologiemi, které Microsoft vydal. Teprve nedávno byla přidána možnost komunikace se službami, které jsou šifrované pomocí protokolu SSL - konkrétně se službami Windows Communication Foundation.

1.2.1 Vývojové prostředí Visual studio

Aplikace pro WP7 se vytvářejí v prostředí Visual studio. K jejich vytváření stačí pouze nainstalovat SDK pro WP7. Instalace SDK do VS je podstatně jednodušší, protože stačí pouze stáhnout instalační balíček a nainstalovat jej. Poté již není potřeba nic nastavovat stačí pouze spustit VS a spustit projekt pro vytváření aplikací a widgetů pro WP7.

SDK je možné stáhnout na oficiálních stránkách Microsoftu zdarma. Stejně jako v Android SDK i zde je k dispozici při vytváření aplikací emulátor ve kterém je možné testovat aplikace.

K vytváření WP7 aplikace jsem používal zatím nejnovější verzi Visual Studia a to konkrétně verzi Microsoft Visual Studio 2010, které nabízí mnoho užitečných nástrojů, které má v sobě implementovány a stejně tak spoustu nových technologií využívaných k tvorbě všech možných aplikací pro platformu Windows od webových aplikací přes služby webové i běžné až po Windows aplikace. Silnou stránkou Visual Studia je hlavně i množství nástrojů, které pomáhají programátorovi s testováním svých aplikací při vývoji. MS VS 2010 je opravdu silný nástroj a programování v něm je radost ovšem velkou nevýhodou je, že Microsoft stále zůstává hlavně na své platformě Windows a proto se nedají aplikace vytvořené ve VS považovat za multiplatformní. Aplikace vytvořené ve VS jedou pouze na platformě Windows stejně tak ke spuštění webové aplikace je potřeba mít server s podporou Windows server a nejlépe i IIS.

1.3 Object-C - iPhone

Všechny iPhone a iPad aplikace jsou programovány ve vývojovém prostředí xCode a jsou vytvářeny za pomoci jazyku Object-C. Object C je velice podobný obyčejnému jazyku C, ale zároveň má mnoho odlišností jako například alokování paměti nebo volání metod ze třídy.

1.3.1 Vývojové prostředí xCode

Tato aplikace slouží pro vytváření aplikací pro iPhone a iPad a lze ji nainstalovat pouze na operační systém Mac OS X. Vytváření aplikací na jiných platformách bylo do nedávna nemožné a stále je to i velice obtížné, ale některým společnostem jako např. Adobe se podařilo vytvořit v jejich aplikaci pro vývoj Flash kompilátor schopný generovat kód spustitelný právě na iPhone a iPad zařízeních.

Veškeré nástroje, které programátor využívá lze nalézt v menu, které je situováno u horního okraje obrazovky, další možností je za pomcí klávesových zkratech, ale ne všechny nástroje mají klávesovou zkratku. Celý design xCode vypadá jednoduše a ani toho na první pohled moc neukazuje, ale přesto v sobě skrývá více než se na první pohled může zdát. Změnou oproti např. visual studiu nebo Eclipse je, že se každý nástroj spouští v novém okně jako samostatná aplikace - je to způsobeno funkčností Mac OS X.

Já jsem využil verze xCode, která je dostupná pro Mac OS X Snow Leopard. Jak napovídá již samotný název Mac OS X jedná se o desátou verzi tohoto operačního systému od společnosti Apple. Společně s operačním systémem vydává Apple i vlastní tzv. MacBook, iMac, iPod a iPhone zařízení. MacBook je zařízení stejně jako v ČR známé notebooky. iMac je poněkud unikátní jedná se o desktopové zařízení nahrazující běžný počítač ovšem veškerý hardware je umístěn v jednom velkém "monitoru".

Zařízení od společnosti Apple jsou považovány za velmi výkonné a stabilní a to i díky jejich operačnímu systému Mac OS.

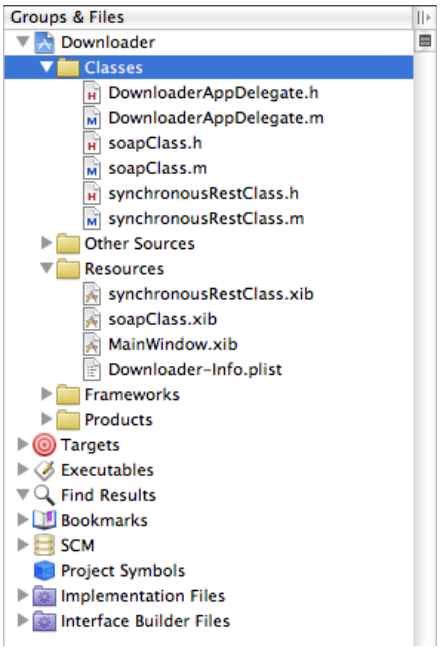
1.3.1.1 Struktura iOS aplikace

Stejně jako většina dnešních aplikací i Apple vytvořil strukturu tak, aby byla oddělena grafika od kódu a udržela se tak čistota celé aplikace. Samotný jazyk Object-C má oproti C určité odlišnosti, ale zároveň se naleznou i znaky, které mají velice podobné. Jedním takovým znakem je např. podobnost v struktuře kódu. Object-C má stejně jako např. jazyk C hlavičkový soubor .h ve kterém jsou uloženy definice proměnných a metod ve třídách a také část ve které se již funkce implementují a to soubor s příponou .m(v jazyce C má příponu .c). Hlavičkový soubor je vlastně interface.

Struktura iOS aplikace [Obr. 3] je rozdělena do složek kde jedna z nich se vygeneruje sama a tato složka je ta hlavní která nás zajímá, protože vše co jsem v xCode

udělal jsem vytvářel pouze zde. Tato složka má vždy název podle zvoleného názvu celého projektu tzn. v mém případě Downloader. Uvnitř nalezneme další složky z nichž jsou nejdůležitjší Classes a Resources.

- Classes - obsahuje hlavičkové a zdrojové soubory ve kterých je napsán kód
- Resources - obsahuje soubory s grafickým rozhraním a také soubor ve kterém se ukládají informace o projektu, které aplikace využívá k nastavení



Obrázek 4: Struktura iOS aplikace

2 KOMUNIKACE

Způsobů komunikace mobilního zařízení se serverem je více. Mobilní zařízení může komunikovat za pomoci mobilního internetu (EDGE, GPRS, HDSPA,..) a nebo také WI-FI, které se pomalu, ale jistě stává samozřejmostí u všech smartphonů.

Pro komunikaci s mobilním zařízením se dají použít i různé druhy služeb. Je možné vytvořit službu v Javě nebo pomocí nástrojů od Microsoftu například vytvořit webovou službu - Web service (.ASMX) nebo pomocí WCF. Já jsem zvolil vytvoření služby za pomoci WCF a to proto, že WCF přebírá výhody všech nástrojů pro tvorbu služeb, které Microsoft doposud vytvořil a odstraňuje většinu, ne-li všechny, jejich nevýhody. Další velkou výhodou je interoperabilita. Komunikaci lze zprostředkovat za pomoci SOAP nebo REST.

2.1 Windows Communication Foundation

WCF je součástí .NET frameworku od roku 2006, kdy vyšel jako jedna ze 4 novinek .NET frameworku 3.0:

- Windows Presentation Foundation (WPF)
- Windows Communication Foundation (WCF)
- Windows Workflow Foundation (WF)
- Windows Card Space

Mezi hlavní rysy WCF patří:

- Sjednocení předchozích technologií
- Interoperabilita napříč platformami
- Orientace na služby

2.1.1 Sjednocení technologií

V .NET 2.0 bylo možné vybrat si z různých distribuovaných technologií, problémem byla jejich nekompatibilita:

- ASP.NET Web Services (ASMX)
- Web services Enhancements (WSE)
- .NET Remoting
- Microsoft Messsage Queue (MSMQ)
- Enterprise services/COM+

Každá technologie měla své klady i zápory. Například k webové službě ASMX se lze připojit z různých platforem, ale nedosahuje takových výkonů jako .NET Remoting a Enterprise services.

Všechny tyto technologie byly sjednoceny právě ve WCF, díky tomu má WCF velké množství možností nastavení, komunikace a zabezpečení komunikace.

2.1.2 Interoperabilita

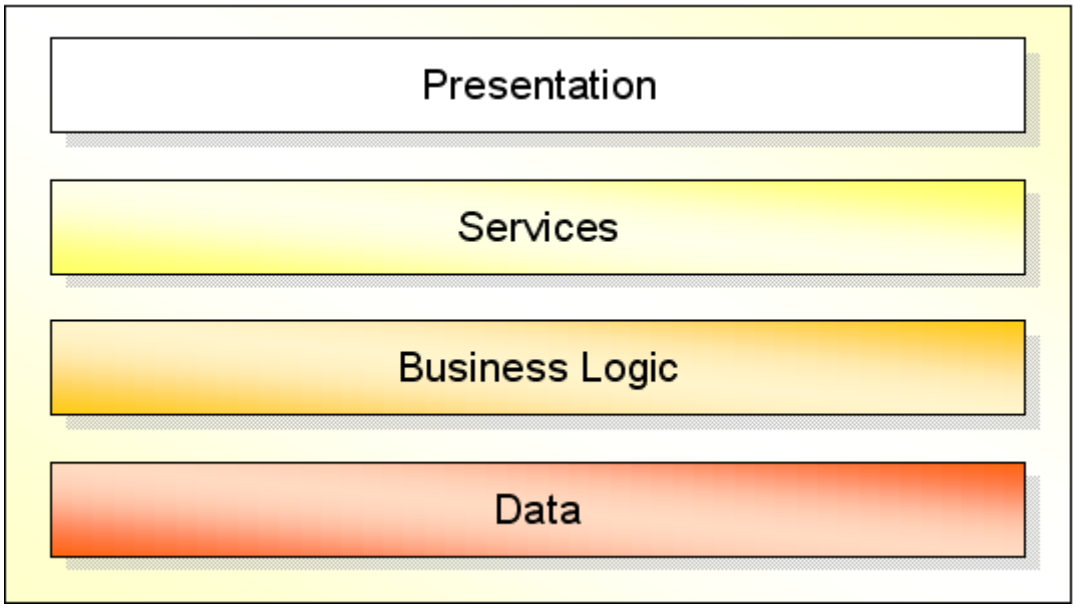
Interoperabilita je schopnost různých systémů spolupracovat. Microsoft si samozřejmě uvědomuje, že to je v dnešní době velmi žádaná a důležitá vlastnost, protože mnoho vývojářů vytváří aplikace na různých operačních systémech (Linux, Mac,...) a také v různých jazycích (Java, C++, Object C, Delphi,...), proto WCF touto schopností oplývá.

Z důvodu rozvoje interoperability velké softwarové společnosti vytvořili Web services interoperability organization, která vyvinula množství standardů pro vzájemnou komunikaci softwaru napříč platformami. Těmito standardy se řídí i právě WCF a díky tomu je pouze na vývojáři, který z daných standardů implementuje.

WCF služby využívají pro komunikaci SOAP, což je otevřený standard pro komunikaci mezi odlišnými operačními systémy a technologiemi. To znamená, že *WCF* může komunikovat například s aplikací na Linuxu vytvořenou v Javě. Hlavní podmínkou je, aby obě strany rozuměli otevřenému standardu pro komunikaci.

2.1.3 Orientace na služby

WCF je navrhnutý v souladu se service-oriented architekturou (SOA). SOA je architektura, která určuje jakým způsobem by měly být vyvíjeny distribuované systémy. Jedná se o způsob návrhu distribuovaného systému ve kterém několik různých služeb společně pracuje za pomoci posílání zpráv.



Obrázek 5: Příklad service-oriented architektury

2.2 Zabezpečení komunikace

Zabezpečit komunikaci lze různými způsoby. Nejčastěji se k tomu využívají různé šifrovací algoritmy, které jsou obsaženy v různých komunikačních protokolech. Jedním z takových protokolů je i protokol SSL/TLS, který je v dnešní době asi nejpoužívanější hlavně v oblasti webových stránek a to hlavně díky protokolu https, který zabezpečuje na webových stránkách šifrovaný přenos dat mezi klientem a serverem.

Zabezpečení komunikace je nutné z důvodu ochrany přenášených informací skrze Internet. Přenášené informace mohou být naše osobní údaje, citlivé dokumenty nebo data. Kromě samotné možnosti posílat data skrze šifrovaný kanál jsou i další možnosti ochrany našich dat:

- posílat data již zašifrovaná pomocí šifrovacího programu
- posílat data komprimovaná pomocí jedné z komprimačních metod (zip, rar, 7zip, ...)

- posílat data s elektronickým podpisem

Webové služby musejí zaručit silné zabezpečení, protože zabezpečení je dnes velmi důležitou součástí distribuovaných aplikací. Zaručit zabezpečení v dnešním otevřeném a dynamickém prostředí, které je charakterizované různorodostí jazyků a operačních systémů je velká výzva.[3]

2.2.1 HTTPS

Https je protokol http zkombinovaný s protokoly SSL a TLS z důvodu zabezpečení síťové komunikace a bezpečné identifikace webového serveru. Standardní port https je 443 a *http* je 80.

Protokol https používá asymetrické šifrování, které je založeno na předávání veřejných klíčů a jejich ověření pomocí privátního klíče. Veřejný klíč by měl být ověřený jiným komunikačním kanálem.

2.2.1.1 SSL/TLS

SSL 3.0 a TLS 1.0 jsou téměř identické vrstvy zabezpečující komunikaci mezi klientem a serverem. Pro šifrovanou komunikaci mezi serverem a klientem platí určitý postup:

- klient pošle zprávu *ClientHello* ve které serveru odešle informace o verzi TLS/SSL, náhodné číslo, seznam možných šifrovacích algoritmů a kompresních metod
- server odešle zpět klientovi *ServerHello* ve kterém odešle zvolenou verzi protokolu, zvolený podporovaný šifrovací algoritmus a komprimační metodu z přijatého seznamu
- server pošle svůj certifikát (pokud to zvolená šifra podporuje)
- server pošle zprávu *ServerHelloDone* zprávu, aby klientovi oznámil, že dokončil iniciační dohodu o použitých mechanismech
- klient odpoví zprávou *ClientKeyExchange*, který může obsahovat *PremasterSecret,* veřejný klíč nebo nic (v závislosti na zvolené šifře)

- klient a server poté z náhodně vybraných čísel a *PreMasterSecret* vytvoří za pomoci pseudonáhodné funkce *masterSecret*. Veškeré ostatní klíče jsou odvozeny od něj.
- klient nyní pošle zprávu *ChangeCipherSpec*, díky níž sděluje, že od něj teď bude probíhat veškerá komunikace šifrovaně
- na závěr odešle klient šifrovanou zprávu *Finished* obsahující hash a MAC předchozích iniciačních zpráv
- server se pokusí dešifrovat klientovu zprávu *Finished* a ověřit její hash a MAC. Pokud dešifrování selže komunikace by se měla ukončit
- nyní server pošle zprávu *ChangeCipherSpec* a následně zprávu *Finished* s hash a MAC, klient se jej pokusí dešifrovat a ověřit, poté je inicializace hotova a komunikace probíhá šifrovaně

2.3 Zprostředkování komunikace

Komunikace v mobilních zařízení probíhá především za pomoci protokolu http a to hlavně proto, že mobilní aplikace se připojují především k různým webovým službám (facebook, gmail, twitter, AccuWeather...) odkud stahují informace. Možností bezpečného přenosu dat z mobilních zařízení není mnoho a to z toho důvodu, že to není primárním cílem těchto zařízení.

Komunikace může probíhat skrze protokol SOAP nebo také přes rozhraní REST. V dnešní době je mnohem výhodnější využívat REST a to z toho důvodu, že pro komunikaci využívá více standardizovaných formátů a je časově i paměťově méně náročný.

2.3.1 SOAP

Jedná se o protokol pro výměnu zpráv založený na XML přes síť a to hlavně přes protokol http. SOAP je nástupcem XML-RPC, ale obálka, hlavička a tělo nejspíše vycházejí z WDDX. Výhodou protokolu je čitelný zápis pro klienta [Příklad 1,2] stejně jako množství programů, které dokážou XML přečíst, ovšem velkou nevýhodou je velikost přenášených dat.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1"
xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">http://tempuri
.org/IService1/TestMethod</Action>
  </s:Header>
  <s:Body>
    <TestMethod xmlns="http://tempuri.org/" />
  </s:Body>
</s:Envelope>
```

Příklad 1: SOAP Reuqest

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header />
  <s:Body>
    <TestMethodResponse xmlns="http://tempuri.org/">
      <TestMethodResult>1</TestMethodResult>
    </TestMethodResponse>
  </s:Body>
</s:Envelope>
```

Příklad 2: SOAP Response

2.3.1.1 ksoap2

Ksoap2 je tzv. knihovna třetí strany, která se může použít v androidu pro komunikaci pomocí SOAP. Samozřejmě si může každý kdo rozumí komunikaci pomocí SOAP vytvořit i vlastní knihovny, ale pro ty kteří SOAP nerozumí nebo potřebují využít komunikace skrze SOAP jen ojediněle je tento nástroj jako stvořený. Existuje mnoho verzí ksoap2 knihovny a to i pro Android. Lze ji nalézt na oficiálních stránkách vývojářské komunity Androidu.

Knihovna ksoap2 je uložena ve formátu JAR a jediné co stačí je vložit ji jako externí JAR soubor do projektu. Poté je možné využívat třídy a metody, které jsou její součástí. Využití knihovny ksoap2 pro komunikaci bylo potřeba z toho důvodu, že jsem se ještě s komunikací pomocí SOAP detailně nesetkal a také proto, že ADT žádnou SOAP knihovnu neobsahuje.

2.3.2 REST

Jedná se o rozhraní, které bylo vytvořeno společně s HTTP1.1. Toto rozhraní definuje jakým způsobem jsou zdroje definovány a adresovány. REST nemá využití pouze pro webové služby přes protokol HTTP, je možné vytvořit síť za pomoci REST i bez použití HTTP protokolu.

Pro komunikaci lze využít více standardizovaných formátů než pouze SOAP, který využívá pouze XML formát.

REST formáty:

- XML - univerzální formát, který funguje vždy a všude
- JSON - způsob zápisu dat nezávislý na počítačové platformě
- ATOM/RSS - sada protokolů pro publikaci a aktualizaci informačních zdrojů

2.3.3 Komunikační formáty

Jak bylo řečeno výše, tak se pro komunikaci používají standardizované formáty, kterým rozumí každá počítačová platforma (Linux, Windows,..). Každý formát má své výhody i nevýhody.

2.3.3.1 XML

Jedná se o značkovací jazyk, který se používá pro komunikaci mezi aplikacemi a také pro publikování dokumentů, u kterých popisuje strukturu podle věcného obsahu jednotlivých částí. XML se používá pro serializaci, ovšem není to jediný jazyk, který se k tomuto účelu používá (JSON,YAML,...).

Výhody XML formátu:

- mezinárodní podpora
- vysoký informační obsah
- snadná konverze do jiných typů dokumentů
- čitelný zápis pro uživatele

Nevýhody XML formátu:

- velikost přenášených dat

2.3.3.2 JSON

Stejně jako u XML i JSON je nezávislý na počítačové platformě a jedná se o standardizovaný formát, který se používá pro komunikaci - rozdíl je ve způsobu uchovávání dat. JSON může uchovávat data v podobě pole nebo může být agregován v objektech. Lze do něj vložit jakýkoliv datový typ či strukturu. Stejně jako XML se využívá pro serializaci dat. Rozdíl mezi XML a JSON je to, že JSON má celkově menší náročnost (paměťová i časová) při zpracování dat. Nízká paměťová náročnost je způsobena částečně i tím, že při přenosu si neuchovává informaci o datovém typu.

Výhody JSON:

- čitelný zápis pro uživatele
- malá časová i paměťová náročnost při zpracování dat

Nevýhody JSON:

- neumožňuje definovat znakovou sadu
- chybějící optimalizace pro přenos binárních dat

2.3.3.3 Atom/RSS

V případě Atomu a RSS se nejedná ani tak o formáty jako spíše o XML standardy, které se využívají k publikování a aktualizaci informačních zdrojů. Atom vznikl z důvodu nespokojenosti části internetové společnosti, kteří viděli nedostatky RSS.

Atom klade důraz na čistý a důsledný návrh a otevřenost. Do povědomí se dostal i díky společnosti Google, která jej začlenila do svých vlastních produktů. Atom pouze určuje pevně strukturu XML formátu, který slouží jako zdroj informačních kanálů.

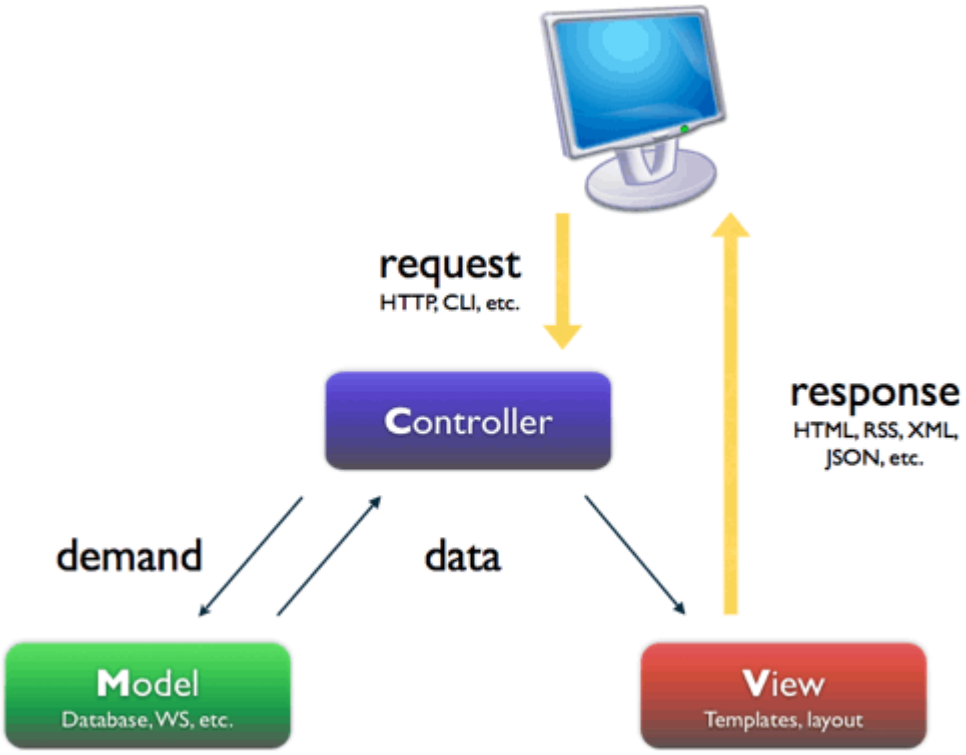
RSS je jeden z rodiny XML formátů, který slouží pro čtení novinek z informačních webů. Původně tento formát sloužil k předávání novinek mezi různými servery, kdy jeden webový server mohl zobrazovat články z jiného zdroje.

RSS poskytuje celý obsah článku nebo jen jeho část a také další metadata. Tato data jsou posílána v podobě XML na jiný server, který je schopen z něj získat data, která zobrazí uživateli.

II. PRAKTICKÁ ČÁST

3 STRUKTURA APLIKACÍ

Aplikace používají modelu podobného MVC [Obr. 1] pro komunikaci mezi mobilním zařízením a serverem kde je uložen soubor XML s daty. Na mobilním zařízení je aplikace, která se dá považovat za součást modelu View a to díky tomu, že android již má vytvořený systém tak, aby oddělil kód od designu. Aplikace se připojují na webovou službu vytvořenou ve WCF za pomoci protokolu HTTPS - Controller a služba teprve načítá soubor ve které je uložena datová struktura - Model.



Obrázek 6: Struktura MVC

3.1 WCF a webové služby

Pro komunikaci s mobilními zařízeními jsem zvolil webovou službu, která již dle názvu vyhovuje, neboť komunikuje přes protokol http i https. To znamená, že rozumí šifrovacím algoritmům obsažených v ve vrstvách SSL a TLS. K vytvoření webové služby jsem využil WCF, což je nová technologie společnosti Microsoft a přišla s novou verzí .NET Frameworku konkrétně s verzí 3.5. WCF není pouze webovou službou je to framework, který se nazývá "service-oriented" neboli zaměřený na služby.

3.1.1 Vytvoření webové služby

K vytváření webové služby za pomocí WCF je potřeba znát strukturu takové služby. Struktura WCF je vcelku jednoduchá neboť ji tvoří App.config popřípadě web.config - záleží právě na tom zda se tvoří webová služba nebo běžná služba. V .config souboru se nastavují veškeré parametry komunikace - jakým způsobem se budou data odesílat, koncová adresa, způsob zabezpečení, způsob autentizace uživatele atd. [Př. 1]

Další součástí je interface. V interface se deklarují názvy metod, návratové datové typy a datové typy parametrů a také je zde možné deklarovat vlastní datové typy. Interface WCF má ovšem speciální způsob zápisu pomocí kterému službě řekneme co která část má dělat tzn. musíme službě říct co je kontrakt služby(celý interface) co jsou pouze datové kontrakty (metody) a co jsou datové členy(vlastní datové typy). A také zde můžeme nastavit další možnosti komunikace jako např. nastavení pro komunikace za pomoci REST - nastavuje se zde URL cesta k metodě, způsob jakým bude prováděn dotaz a odpověď(JSON, XML), metoda odesílání dat (GET, POST, PUT, DELETE atd.). [Př. 2]

Posledním článkem je samotná deklarace služby, která využívá z interface a obsahuje již i těla funkcí, která říkají co má která funkce dělat. [Př. 3]

3.1.1.1 Konfigurace WCF

App.config nebo Web.config je vlastně XML soubor , který má jasně danou strukturu do které se vkládají informace o nastavení. Hlavní element konfiguračního souboru je element <configuration> </configuration> mezi tyto dva tagy se vkládají všechny ostatní informace potřebné k publikování služby. Moje konfigurace je velice jednoduchá a nenáročná, neobsahuje totiž žádnou autentizaci ať již uživatele nebo za pomoci podpisů atp.

Chování služby se nastavuje v části *Services*, kde se každému koncovému bodu služby nastavují pomocí atributů různé vlastnosti, které definujeme v dalších částech konfiguračního souboru. Moje služba má dva koncové body neboli adresy podle toho zda volám metodu, která využívá SOAP nebo REST využívám jeden nebo druhý koncový bod.

U každé služby lze nastavit i jejich individuální chování a to pomocí atributu *behaviorConfiguration*, která náleží přímo tagu *Services* a definuje se v části *Behaviors - ServiceBehaviors*.

Stejně jako u služeb i u koncových bodů - *Endpoint* - lze nastavit individuální chování a to např. pomocí atributů:

- contract - contractem je vždy Interface dané služby
- address - nastavuje koncový bod služby
- binding - vazba, která je použita pro připojení ke službě
- bindingConfiguration - nastavení vazby - definováno v *Bindings - typ vazby např. BasicHttpBinding* - zde se nastavuje převážně zabezpečení komunikace služby a to zda bude služba komunikovat s koncovým zařízením přes zabezpečený kanál, zda bude služba vyžadovat identifikaci uživatele atd.
- behaviorConfiguration - nastavení chování daného koncového bodu - definování v *Behaviors - EndpointBehaviors*

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <system.web>
    <compilation debug="true" />
  </system.web>
  <system.serviceModel>
    <services>
      <service name="AndroidSoapService.Service1"
behaviorConfiguration="WcfService.Service1Behavior">
        <endpoint address="soap" binding="basicHttpBinding"  contract="AndroidSoapService.IService1"
bindingConfiguration="TransportSecurity" />
        <endpoint address="rest" binding="webHttpBinding" contract="AndroidSoapService.IService1"
bindingConfiguration="RestConfig" behaviorConfiguration="RestBehavior" kind=""/>
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="WcfService.Service1Behavior">
          <serviceDebug includeExceptionDetailInFaults="true"/>
          <serviceMetadata httpGetEnabled="true"/>
        </behavior>
      </serviceBehaviors>
      <endpointBehaviors>
        <behavior name="RestBehavior">
          <webHttp/>
        </behavior>
      </endpointBehaviors>
    </behaviors>

    <bindings>
      <basicHttpBinding>
        <binding name="TransportSecurity" messageEncoding="Text">
          <security mode="Transport">
            <transport clientCredentialType="None"/>
          </security>
        </binding>
      </basicHttpBinding>
      <webHttpBinding>
        <binding name="RestConfig">
          <security mode="Transport">
            <transport clientCredentialType="None"></transport>
          </security>
        </binding>
      </webHttpBinding>
    </bindings>
  </system.serviceModel>
  <appSettings>
    <add key="path" value="C://AndroidXmlFilesRepository"/>
  </appSettings>
</configuration>
```

Příklad 3: Konfigurační soubor WCF

3.1.1.2 Interface/Contract

Interface slouží jako kontrakt pro službu, definuje strukturu služby - říká službě jaké metody s jakými parametry a návratovými hodnotami obsahuje a může využívat. Stejně tak se v interface můžou deklarovat i vlastní datové typy, které se budou využívat.

V interface můžeme ze SOAP služby vytvořit službu využívající i REST. Službě vlastně řekneme, že k volání této metody je možné využít i URL odkaz a určíme jaký ten odkaz bude. V tom je hlavní rozdíl mezi SOAP a REST, SOAP nabízí metody, které jsou zobrazeny ve WSDL souboru zatímco REST očekává konkrétní adresu URL a podle toho zavolá příslušnou metodu. Aby bylo možné používat REST je potřeba mít i správně nastavený koncový bod služby.

```
USING SYSTEM;
USING SYSTEM.COLLECTIONS.GENERIC;
USING SYSTEM.LINQ;
USING SYSTEM.RUNTIME.SERIALIZATION;
USING SYSTEM.SERVICEMODEL;
USING SYSTEM.SERVICEMODEL.WEB;
USING SYSTEM.TEXT;

NAMESPACE ANDROIDSOAPSERVICE
{
    [SERVICECONTRACT]
    PUBLIC INTERFACE ISERVICE1
    {
        [OPERATIONCONTRACT]
        [WEBGET(URITEMPLATE="TESTMETHOD",RESPONSEFORMAT=WEBMESSAGEFORMAT.JSON)]
        INT TESTMETHOD();

        [OPERATIONCONTRACT]
        [WEBGET(REQUESTFORMAT = WEBMESSAGEFORMAT.JSON, URITEMPLATE = "GETXML/{XMLFILENAME}",
METHOD = "POST")]
        STRING GETXML(STRING XMLFILENAME);

        [OPERATIONCONTRACT]
        [WEBGET(URITEMPLATE="SAVEDATA/{XMLSOURCE}/{XMLFILENAME}",METHOD="POST",
RESPONSEFORMAT=WEBMESSAGEFORMAT.JSON,BODYSTYLE=WEBMESSAGEBODYSTYLE.WRAPPED)]
        BOOL SAVEDATA(STRING XMLSOURCE,STRING XMLFILENAME);
    }
}
```

Příklad 4: Interface IService1.svc

3.1.1.3 Implementace metod kontraktu

Metody z interface implementuje v souboru s koncovkou svc neboli service. Zde se již určí co která funkce bude dělat a tvoří se stejným způsobem jako běžná třída v C#. Službu je možné si pojmenovat vlastním názvem stejně tak i interface, názvy které jsou použity v mé službě se vytvoří jako ukázkový příklad při vytvoření nového projektu.

```
USING SYSTEM;
USING SYSTEM.COLLECTIONS.GENERIC;
USING SYSTEM.LINQ;
USING SYSTEM.RUNTIME.SERIALIZATION;
USING SYSTEM.SERVICEMODEL;
USING SYSTEM.SERVICEMODEL.WEB;
USING SYSTEM.TEXT;
USING SYSTEM.CONFIGURATION;
USING SYSTEM.IO;
USING SYSTEM.XML;

NAMESPACE ANDROIDSOAPSERVICE
{
    PUBLIC CLASS SERVICE1 : ISERVICE1
    {

        PUBLIC INT TESTMETHOD()
        {
            RETURN 1;
        }

        PUBLIC STRING GETXML(STRING XMLFILENAME)
        {
            STRING PATH = STRING.FORMAT("{0}//{1}", CONFIGURATIONMANAGER.APPSETTINGS["PATH"],
XMLFILENAME);
            IF (FILE.EXISTS(PATH))
            {
                LIST<STRING> NAMES = NEW LIST<STRING>();
                XMLDOCUMENT DOC = NEW XMLDOCUMENT();
                TRY
                {
                    DOC.LOAD(PATH);
                }
                CATCH (XMLEXCEPTION)
                { THROW; }

                STRINGWRITER SW = NEW STRINGWRITER();
                XMLTEXTWRITER XW = NEW XMLTEXTWRITER(SW);
                DOC.WRITECONTENTTO(XW);
                STRING TEST = SW.TOSTRING();
                RETURN SW.TOSTRING();
            }
            ELSE RETURN "FILE NOT FOUND";
        }


        PUBLIC BOOL SAVEDATA(STRING XMLSOURCE, STRING XMLFILENAME)
        {
            STRING PATH = STRING.FORMAT("{0}{1}", CONFIGURATIONMANAGER.APPSETTINGS["PATH"],
XMLFILENAME);
            USING (STREAMWRITER OUTFILE = NEW STREAMWRITER(PATH))
```

```
            {
                TRY
                {
                    OUTFILE.WRITE(XMLSOURCE);
                }
                CATCH (IOEXCEPTION)
                {

                    RETURN FALSE;
                }
                CATCH (NOTSUPPORTEDEXCEPTION)
                {
                    RETURN FALSE;
                }
                CATCH (OBJECTDISPOSEDEXCEPTION)
                {
                    RETURN FALSE;
                }
                FINALLY
                {
                    OUTFILE.CLOSE();

                }

            }
            RETURN TRUE;
        }
    }
}
```

Příklad 5: Služba Service1.svc

Příklad 1, 2 i 3 tvoří WCF službu, která je součástí bakalářské práce a používá se ke komunikaci s aplikacemi na Androidu, WP7 a iPhone.

Tato služba je publikována na serveru v informační internetové službě IIS a je přístupná z Internetu tzn. tato webová služba je publikována. Služba dělá pouze to, že nabízí metody, které může uživatel využívat. V tomto případě jsou to tři metody :

- TestMethod() - tato metoda slouží pouze pro testování připojení ke službě - metoda pouze vrací číslo 1
- GetXml(String) - slouží ke stažení obsahu XML souboru, který je uložen ve složce definované v konfiguračním souboru a s názvem definovaným jako parametr této metody
- SaveData(String,String) - tato metoda slouží k opětovnému uložení XML souboru, kde první parametr určuje obsah souboru a druhý parametr určuje název souboru

3.2 Android aplikace

Aplikace pro Android je vytvořena tak, že využívá obě koncové adresy služby. To znamená, že komunikuje jak pomocí SOAP tak i pomocí REST. Oba kanály jsou zabezpečené tzn. komunikace je šifrovaná pomocí certifikátu, který je umístěn na serveru. moje služba je momentálně přístupná každému kdo na ni zná adresu, protože není v nastavení WCF spuštěna autentizace uživatele, který se pokouší ke službě přihlásit. Potřeba autentizace uživatele je ovšem důležitá, protože pokud by nebyla spuštěna, tak je samozřejmě naprosto zbytečné přenášet data po zabezpečeném kanále neboť se na službu může připojit kdokoliv a data si kdykoliv stáhnout.

Aplikace v Androidu mohou obsahovat spoustu různých tříd, XML souborů pro různá nastavení, obrázků, zvuků atd. přidaných při vytváření aplikace. Protože by ovšem taková aplikace byla náročná pro instalaci do mobilního zařízení, tak se již při debugování aplikace vytvoří speciální instalační balíček s příponou .apk, kterou lze nalézt ve složce projektu v podsložce s názvem bin.

3.2.1 Aplikace pro stažení přes zabezpečený kanál

Aplikace je pojmenovaná Downloader neboť je udělána tak, aby data pouze stahovala. Je to z toho důvodu, že hlavním cílem je pouze ukázat jak se data mohou přenášet skrze zabezpečený kanál a komunikace probíhá stejným způsobem ať již data stahujeme nebo odesíláme.

Druhá aplikace se jmenuje SmsCentrum a slouží ke stažení informací z databáze přes zabezpečený kanál. V případě, že se data podaří stáhnout se odešle SMS zpráva na telefonní číslo, které je součástí stažených dat. Společně s odeslanou SMS se zobrazí i upozornění přímo v mobilním zařízení a do databáze se odešle informace, že byla SMS odeslána. Během odesílání se zároveň aktualizuje i UI kde se zobrazuje kolik SMS bylo odesláno administrátorovi, uživateli a celkově a kdy byla odeslána poslední SMS.

3.2.1.1 Počáteční problémy

Když jsem začínal s vývojem aplikací pro Android, tak první kroky provázely problémy stejně jako asi každého začínajícího programátora. První aplikace byla Downloader a druhá

SmsCentrum. Které komunikuje přes zabezpečený kanál HTTPS se službami. U obou aplikací mě provázely problémy.

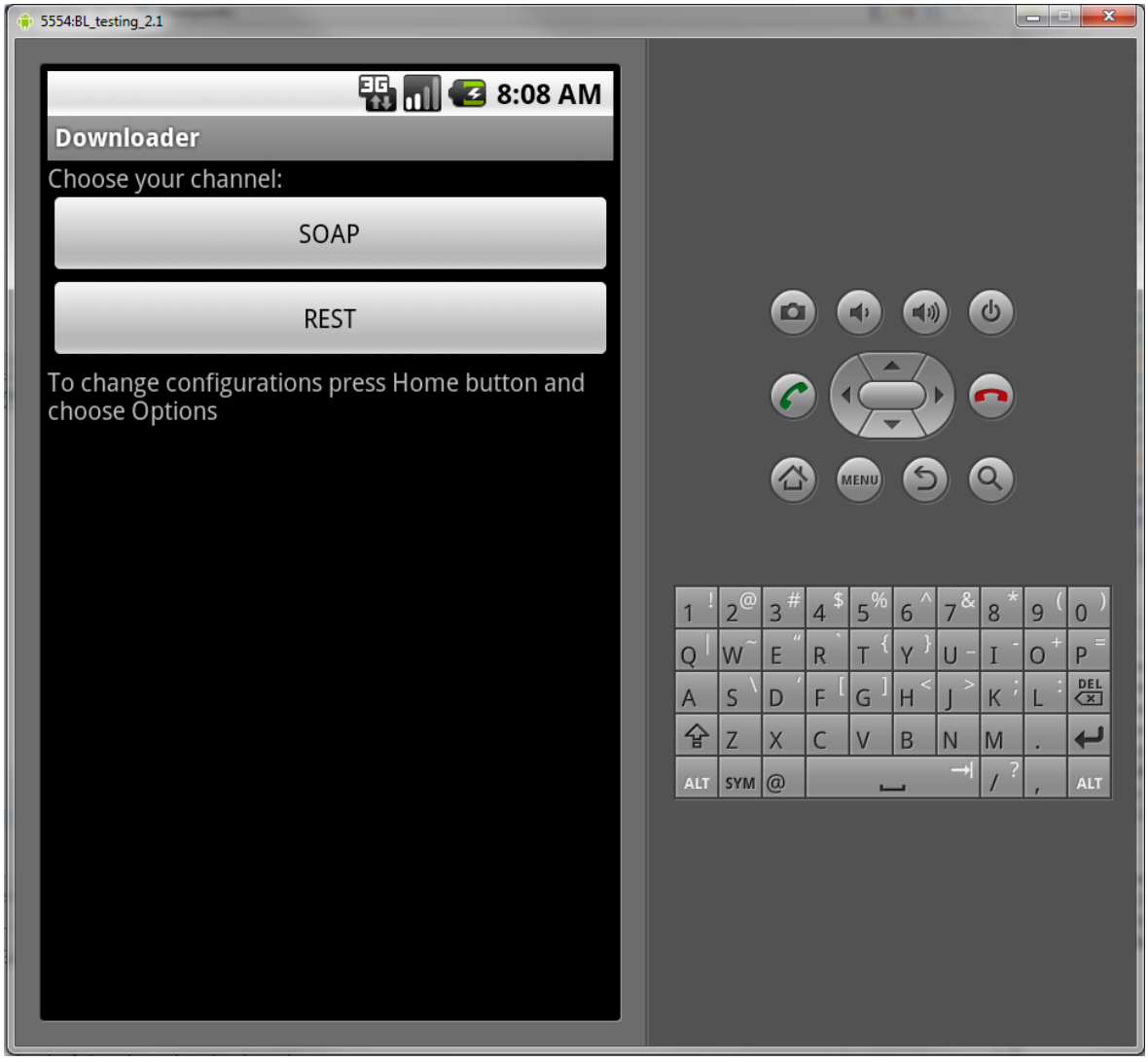
- Služba je umístěna na serveru na kterém je zakoupený certifikát a díky němu je komunikace se serverem zabezpečená. K tomu, aby vše fungovalo správně je ovšem zapotřebí mít vše správně nastaveno. Server, který využívám měl nastavenou doménu na jiný název než byl nastaven certifikát, což WCF službě při komunikaci skrze SOAP způsobovalo potíže, protože se neustále měnila koncová adresa. WSDL soubor se neustále snažil komunikovat na URL dle jména domény, ale skutečná adresa byla určována certifikátem. tento problém se nakonec vyřešil díky výměně serveru za novější a výkonnější a při jeho instalaci se již vše nastavilo správně. Zajímavé je, že když jsem vytvořil komunikaci skrze REST, tak služba komunikovala správně a vůbec neprotestovala, že je certifikát nastaven na jinou doménu.
- Dalším problémem byla komunikace Android služby s UI. Předávání informací ze služby do UI není tak jednoduché jako v např. v .NET. Každá aktivita nebo služba používá tzv. Intent k přenosu informací mezi aktivitami a službami. Problémem bylo, že Android služba je brána jako vlákno, které pracuje v pozadí celé aplikace a žádná její metoda neumožňuje přímo vstupovat zpět do aktivity. Na Internetu jsem nalezl spoustu návrhů jak přenášet data mezi službou a aktivitou, ale každý z nich měl nějakou vadu, díky kterému aplikace buď spadla nebo uvázla. nakonec se mi podařilo najít řešení za pomoci tříd a interface Handler, BroadcastReciever a Runnable.
- Dalším problémem, který se mi nepodařilo vyřešit je problém s vypínáním Android služby v případě, že mobilní zařízení není připojeno do sítě elektrického napětí. Vypadá to, že je to ochranný mechanizmus operačního systému, aby šetřil energii a proto všechny služby zastaví a po určité době jim dovolí se opět spustit.

3.2.1.2 Popis aplikace - Downloader

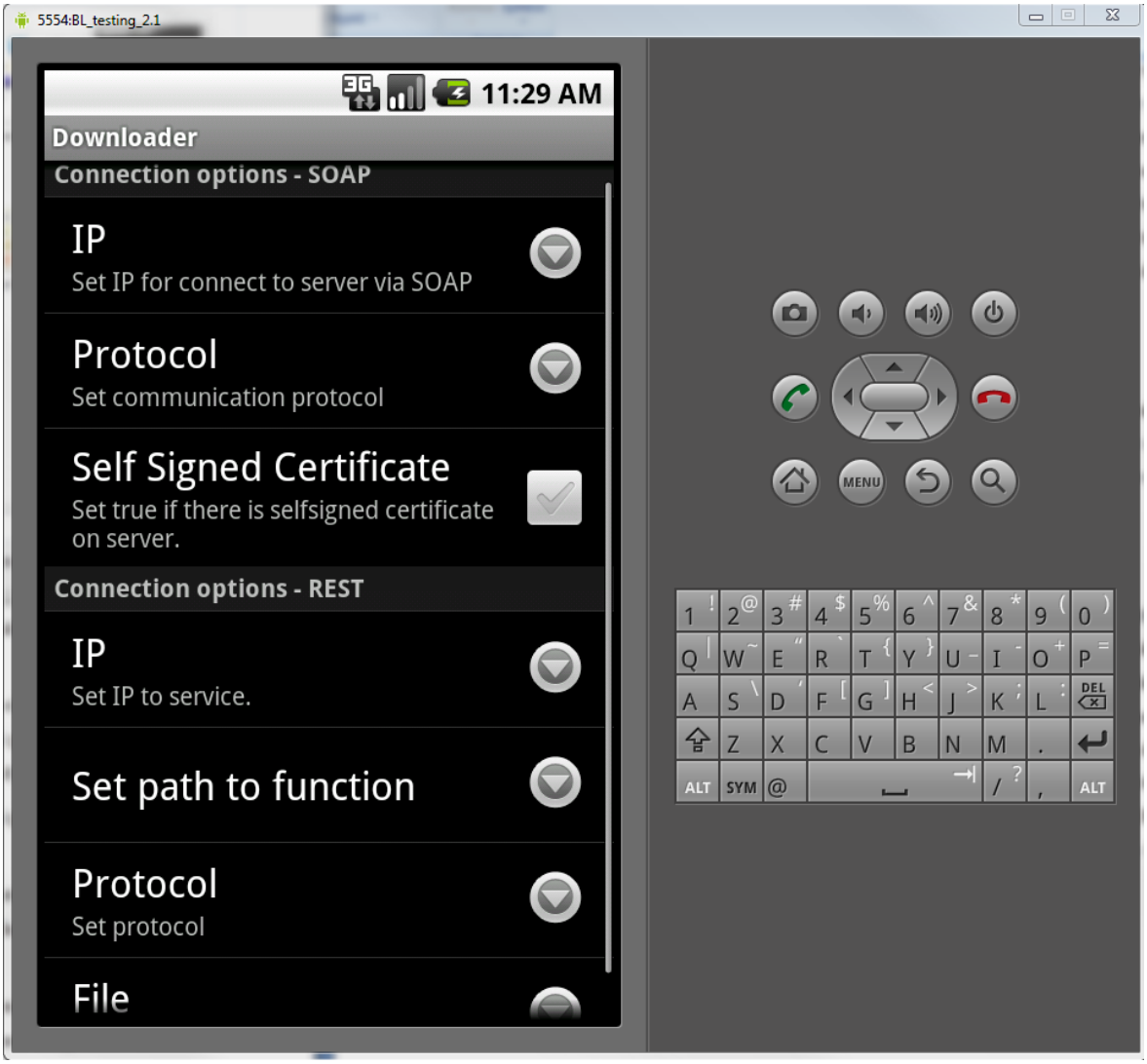
Hlavní menu [Obr. 5] se zobrazí ihned po spuštění aplikace. Skrývá se na něm ovšem více než je na první pohled viditelné. Na úvodní stránce se totiž kromě popisků a dvou tlačítek skrývá ještě jedno tzv. menu možností [Obr. 6], které se zobrazí po kliknutí na klávesu

Menu a také při úplně prvním spuštění aplikace. Zobrazí se menu ve kterém je potřeba nastavit adresy ke službám SOAP i REST, protokol přes který mají služby komunikovat (má služba má nastavenou komunikaci výhradně přes https protokol) . V nastavení pro SOAP je možné zvolit i to zda používá server self-signed certifikát nebo ne a v části nastavení REST je potřeba zapsat název metody, která se bude volat a název souboru který se bude stahovat.

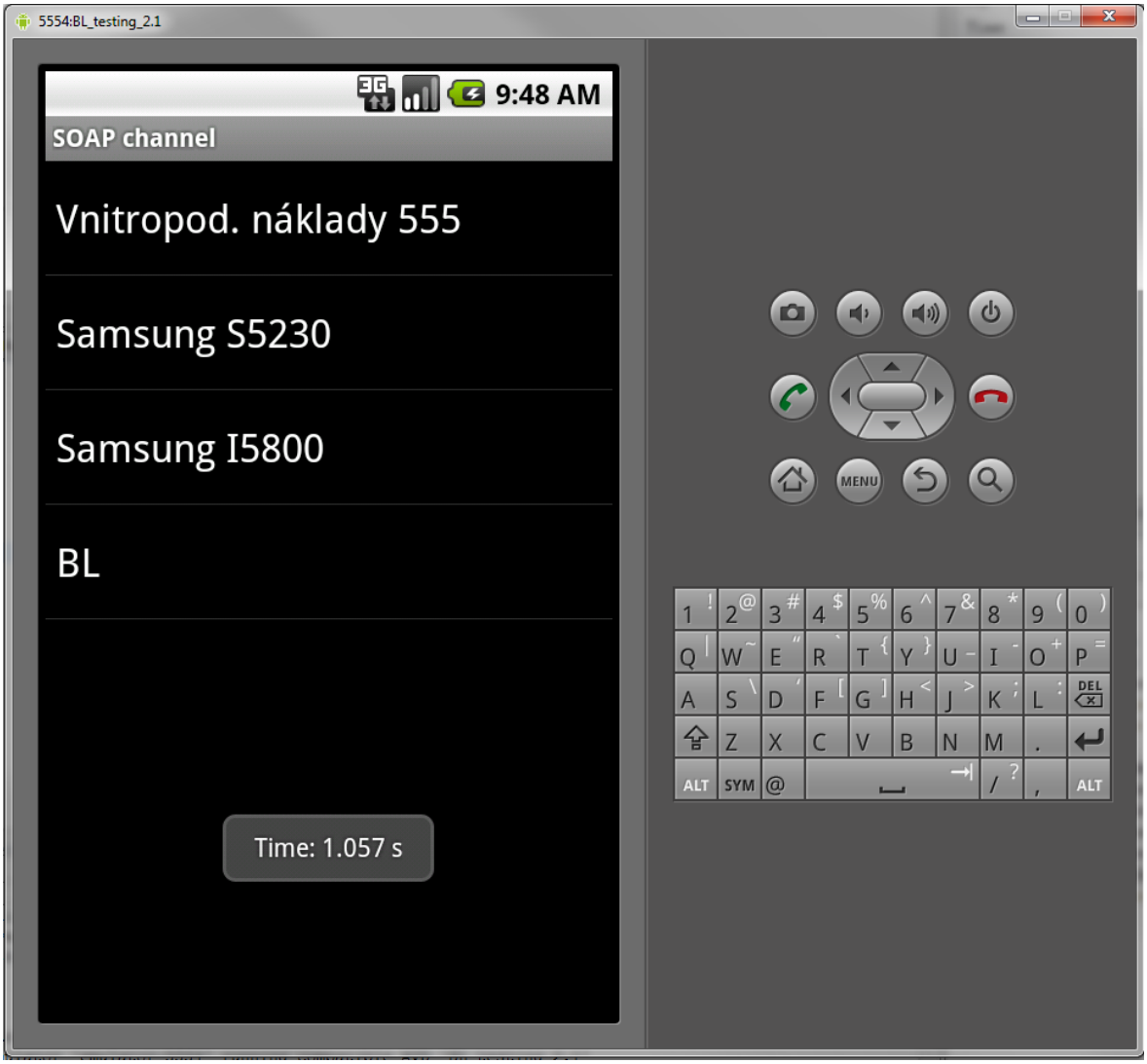
V samotném hlavním menu jsou dvě tlačítka s názvy SOAP a REST. Po nastavení potřebných konfiguračních údajů by se měl po stisknutí tlačítka SOAP nebo REST stáhnout XML soubor za pomoci služby a z něj poté parsovat data do seznamu (ListView) [Obr. 7]. Každý prvek seznamu je odkaz na další detaily [Obr. 8]. Při načítání seznamu se objeví i časový údaj v sekundách, který ukazuje jak dlouho trvalo samotné stažení souboru.



Obrázek 7: Hlavní menu Android



Obrázek 8: Menu možností



Obrázek 9: Seznam prvků

3.2.1.3 *Kód aplikace - Downloader*

Aplikace má několik aktivit, které uživateli umožňují ovládat aplikaci. Je zde základní aktivita, která se objeví ihned po spuštění aplikace. Android musí samozřejmě vědět, která aplikace je ta prvotní a taková informace se musí zapsat do souboru *manifest.xml*:

```
<ACTIVITY ANDROID:NAME=".DOWNLOADERACTIVITY"
          ANDROID:LABEL="@STRING/APP_NAME">
    <INTENT-FILTER>
        <ACTION ANDROID:NAME="ANDROID.INTENT.ACTION.MAIN" />
        <CATEGORY ANDROID:NAME="ANDROID.INTENT.CATEGORY.LAUNCHER"/>
    </INTENT-FILTER>
</ACTIVITY>
```

Příklad 6: Manifest aplikace Downloader

Aplikace z tohoto souboru zjistí díky atributům *action* a *category*, že se jedná o úvodní aktivitu a při každém spuštění ji jako první zobrazí.

V aplikaci se dají nastavit i některé důležité informace[Obr. 7] jako například IP adresa na službu, protokol přes který bude komunikovat atd. K tomu, abychom se do tohoto nastavení dostali musíme spustit nejdříve menu, přes které se do tohoto nastavení dostaneme. Což uděláme pomocí tlačítka *Home*, které vyvolá menu ve kterém mám jen jednu položku. O vyvolání menu se stará metoda onCreateOptionsMenu:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.options_menu, menu);
        return true;
}
```

Příklad 7: Metoda onCreateOptionsMenu

Tento kód pouze vyvolá zobrazení tlačítka popř. tlačítek, ale pro vyvolání metod nebo aktivit uložených pod každým tlačítkem se volá metoda onOptionsItemSelected:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()) {
        case R.id.option_menu:
                Intent i = new Intent(this, Options.class);
                startActivity(i);
        break;
        }
        return super.onOptionsItemSelected(item);
}
```

Příklad 8: Metoda onOptionsItemSelected

Zde se zavolá vytvoření nové aktivity ve které je uloženo menu pro nastavení komunikace.

Na úvodní stránce lze vidět i dvě tlačítka. Každé z nich stahuje data ze služby, která je publikována na serveru. Jak již názvy těchto tlačítek napovídají, tak jedno z nich stahuje pomocí SOAP a druhé využívá REST. Obě activity, které se skrývají za těmito tlačítky jsou tzv. ListActivity, které umožňují zobrazit stažená data jako seznam prvků.

Stažení pomocí SOAP probíhá pomocí knihovny třetí strany ksoap2. Tato knihovna byla vytvořena z toho důvodu, protože Android samotný neobsahuje žádnou knihovnu pro komunikace skrze SOAP implicitně. Pro stažení komunikace je potřeba vědět jak knihovnu použít. Pro vytvoření tzv. requestu neboli požadavku na server se používá *SoapObject*:

```
REQUEST = NEW SOAPOBJECT(NAMESPACE, METHOD);
```

Příklad 9: Vytvoření dotazu na server

- NAMESPACE - obsahuje jmenný prostor definovaný ve službě, já jsem nechal standardní jmenný prostor, který se zde využívá a tím je http://tempuri.org.
- METHOD - obsahuje název metody, která se bude na serveru volat

Pokud metoda kterou voláme obsahuje i parametry, tak je potřeba požadavku sdělit jaké parametry má požadovat. Tyto parametry vložíme pomocí metody *addProperty*:

```
REQUEST.ADDPROPERTY(INFO);
```

Příklad 10: Vložení parametru dotazu

K předávání dat a to jak požadavku tak odpovědi se používá SoapEnvelope. SoapEnvelope obsahuje především informace o verzi, text požadavku a po zavolání metody *call* i Response neboli odpověď dotazu:

```
SOAPENVELOPE = NEW SOAPSERIALIZATIONENVELOPE(SOAPVERSION);
SOAPENVELOPE.DOTNET = TRUE;
SOAPENVELOPE.SETOUTPUTSOAPOBJECT(REQUEST);
```

Příklad 11: Vytvoření SoapEnvelope a vložení dotazu

Požadavek i odpověď jsou v podobě SOAP dotazu a odpovědi, což je vlastně XML soubor a data se zněj musejí tzv. parsovat. Pro připojení ke službě a získání dat z dotazu jsem vytvořil vlastní metodu:

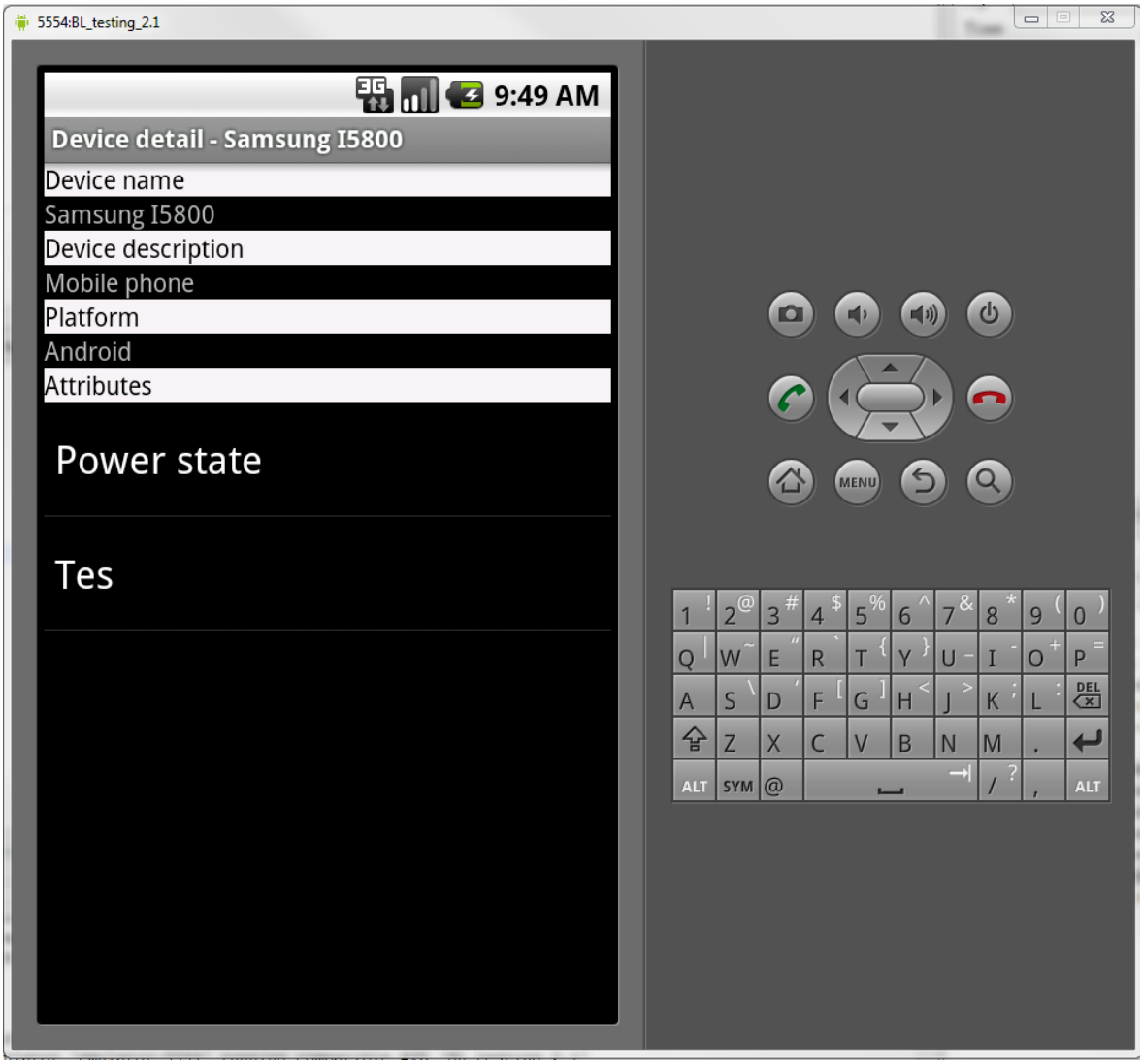
```
PUBLIC OBJECT HTTPCALL() {

        STRING URL = THIS.SCHEMA+THIS.URL;
        HTTPTRANSPORTSE TRANSPORT = NEW HTTPTRANSPORTSE(URL);
        OBJECT RESPONSE = NULL;
        TRY {
                IF(THIS.SELFSIGNED) {
                        TRY {
                                TRUSTCALLBACK();
                        } CATCH (KEYMANAGEMENTEXCEPTION E) {
                                // TODO AUTO-GENERATED CATCH BLOCK
                                E.PRINTSTACKTRACE();
                        } CATCH (NOSUCHALGORITHMEXCEPTION E) {
                                // TODO AUTO-GENERATED CATCH BLOCK
                                E.PRINTSTACKTRACE();
                        }
                }
                LONG STARTTIME = SYSTEM.CURRENTTIMEMILLIS();
                TRANSPORT.CALL(THIS.ACTION, SOAPENVELOPE);
                RESPONSE = SOAPENVELOPE.GETRESPONSE();
                LONG ENDTIME = SYSTEM.CURRENTTIMEMILLIS();
```

```
                DOWNLOADTIME = (ENDTIME-STARTTIME);
            } CATCH (IOEXCEPTION E) {
                E.PRINTSTACKTRACE();
            } CATCH (XMLPULLPARSEREXCEPTION E) {
                E.PRINTSTACKTRACE();
            }
            RETURN RESPONSE;
    }
```

Příklad 12: Připojení k WCF službě pomocí SOAP

Tato metoda zajišťuje kromě získávání dat ze služby i získání výsledného času stažení dat a kontrolu zda si uživatel zvolil, že jeho server obsahuje self-signed certifikát a pokud ano zavolá se callback, který zapříčiní to, že aplikace bude certifikátu důvěřovat a data se i přesto, že certifikát není podepsán důvěryhodnou certifikační autoritou, stáhnout.



Obrázek 10: Detail jednoho z prvků

Stažení pomocí REST již probíhá za pomoci třídy, která je součástí základní knihovny Android. jedná se o třídu HTTPUrlConnection pro komunikace skrze http nezabezpečeně a HTTPSUrlConnection pro šifrovanou komunikaci. Pomocí této třídy nastavíme parametry, metodu odesílání dat dokonce máme možnost nastavit time-out tzn. dobu po kterou se bude pokoušet odeslat dotaz službě. pro komunikaci skrze nezabezpečený kanál jsem vytvořil metodu:

```
PUBLIC STRING HTTPCONNECT() THROWS IOEXCEPTION {
              STRING PARAMETERS = NEW STRING();
              FOR (STRING PARAMETER : VALUES) {
                     PARAMETERS += "/"+URLENCODER.ENCODE(PARAMETER,"UTF-8");
              }
               STRING TARGET = ((SCHEMA.ENDSWITH("/") || IP.STARTSWITH("/"))  ?
SCHEMA : SCHEMA+"/")+ ((IP.ENDSWITH("/") || PATH.STARTSWITH("/")) ? IP : IP +
"/") + (PATH.ENDSWITH("/") ? PATH.SUBSTRING(0, PATH.LENGTH()-1) :
PATH)+PARAMETERS;
              URL URL = NEW URL(TARGET);
              HTTPURLCONNECTION HTTPCONNECTION = (HTTPURLCONNECTION)
URL.OPENCONNECTION();
              HTTPCONNECTION.SETCONNECTTIMEOUT(5000);
              HTTPCONNECTION.SETREQUESTMETHOD("GET");
              HTTPCONNECTION.SETINSTANCEFOLLOWREDIRECTS(FALSE);
              HTTPCONNECTION.SETUSECACHES(FALSE);
              HTTPCONNECTION.SETDOOUTPUT(TRUE);
              HTTPCONNECTION.SETDOINPUT(TRUE);
              LONG STARTTIME = SYSTEM.CURRENTTIMEMILLIS();
              HTTPCONNECTION.CONNECT();
              LONG ENDTIME = SYSTEM.CURRENTTIMEMILLIS();
              DOWNLOADTIME = (ENDTIME-STARTTIME);
              INT RESPONSECODE = HTTPCONNECTION.GETRESPONSECODE();
              INPUTSTREAM INPUTSTREAM;
              IF (RESPONSECODE == 200) {
                  INPUTSTREAM = HTTPCONNECTION.GETINPUTSTREAM();
              } ELSE {
                  INPUTSTREAM = HTTPCONNECTION.GETERRORSTREAM();
                  IF (INPUTSTREAM == NULL) {
                  RETURN(NULL);
                  }
              }
              INPUTSTREAMREADER READER = NEW INPUTSTREAMREADER(INPUTSTREAM, "UTF-8");
              STRINGWRITER WR = NEW STRINGWRITER();
              COPYREADERTOWRITER(READER, WR);
              STRING XMLFILE = WR.GETBUFFER().TOSTRING();
              XMLFILE = XMLFILE.REPLACE("\\", "");
              RETURN XMLFILE;
       }
```

Příklad 13: Připojení k WCF službě pomocí REST

V této metodě nastavím parametry do URL odkazu, opět zjistím jak dlouho trvalo připojení ke službě a odešlu zpět odpověď služby.

Metoda pro komunikaci skrze https je téměř stejná. Pouze se používá opět callback, který dovolí aplikaci komunikovat se serverem i v případě, že aplikace nedůvěřuje certifikační autoritě:

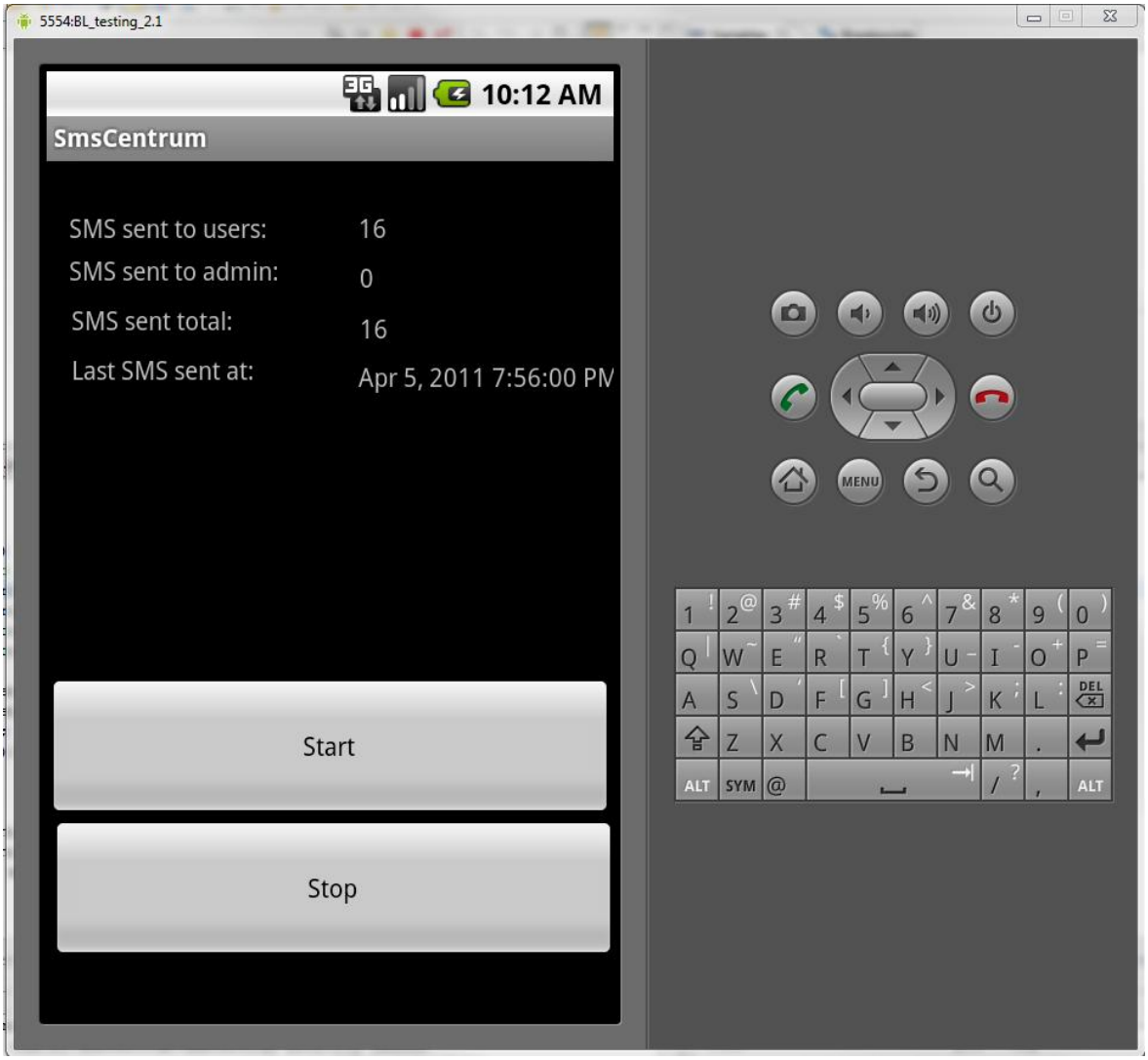
```
HTTPSCONNECTION.SETHOSTNAMEVERIFIER(NEW ALLOWALLHOSTNAMEVERIFIER());
```

Příklad 14: Verifikace self-signed certifikátu

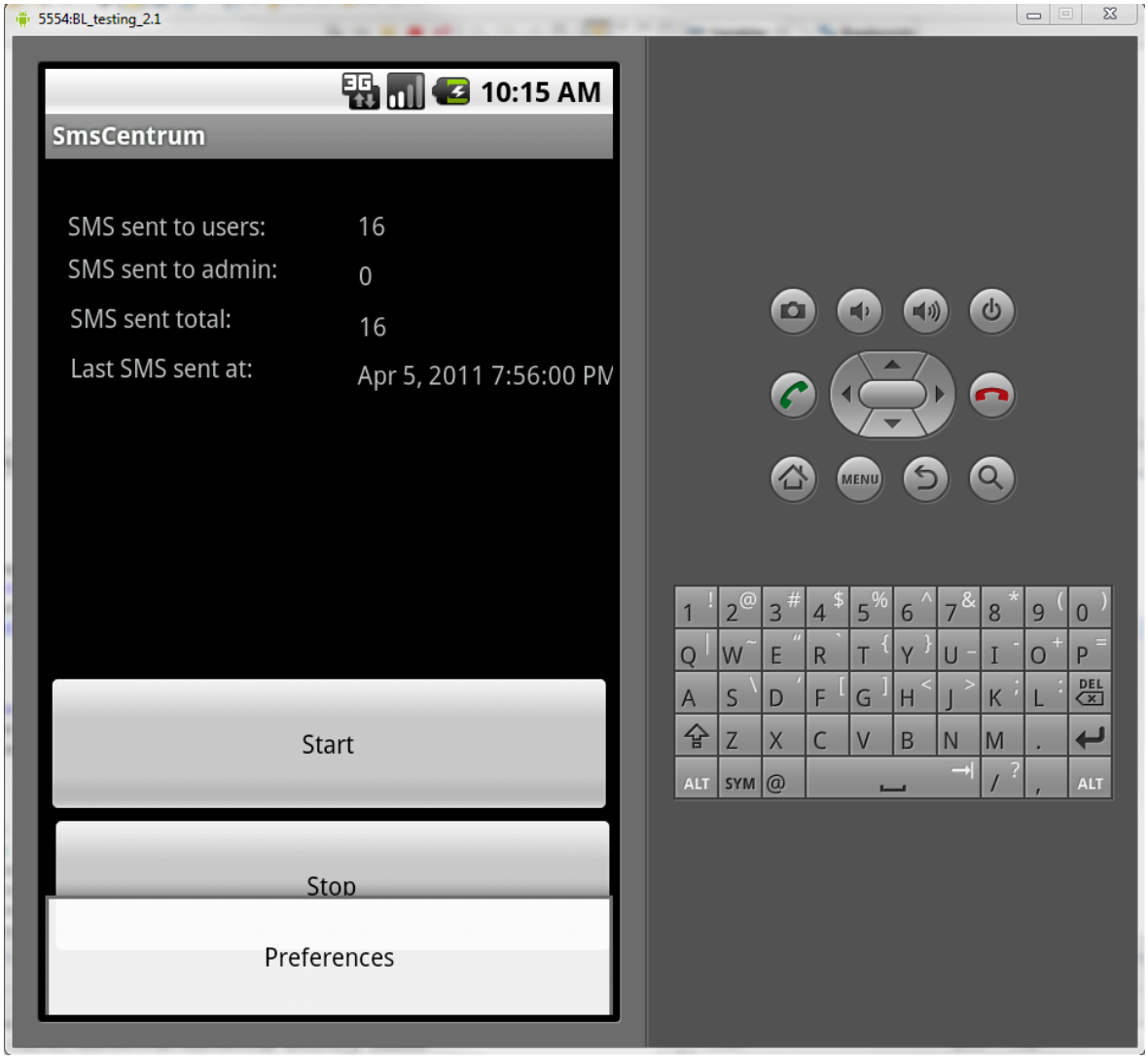
3.2.1.4 Popis aplikace - SmsCentrum

V Androidu jsem vytvořil dvě aplikace - první aplikace pouze stahuje a parsuje XML soubor a z něj vytváří seznam prvků v něm. Celá aplikace se skládá pouze z několika aktivit. Aplikace SmsCentrum využívá android službu, Handler, BroadcastReciever, Runnable a úvodní aktivitu. Ačkoliv celá aplikace vypadá jednodušeji byla o poznání složitější než Downloader a to z toho důvodu, že Android je mezi operačními systémy pro mobilní zařízení ještě dítě a snaží se velmi rychle dospět. S tím souvisí obrovské změny v každé nově výchozí verzi a tím i neaktuálnost návodů nalezených na internetu a to dokonce i na oficiálních stránkách vývojářů Androidu. A také proto, že aplikace SmsCentrum kromě komunikace se službou odesílá SMS zprávy, což je jeho prvotním určením a zároveň odesílá i tzv. notifikace nebo-li upozornění uživateli přímo v mobilním zařízení.

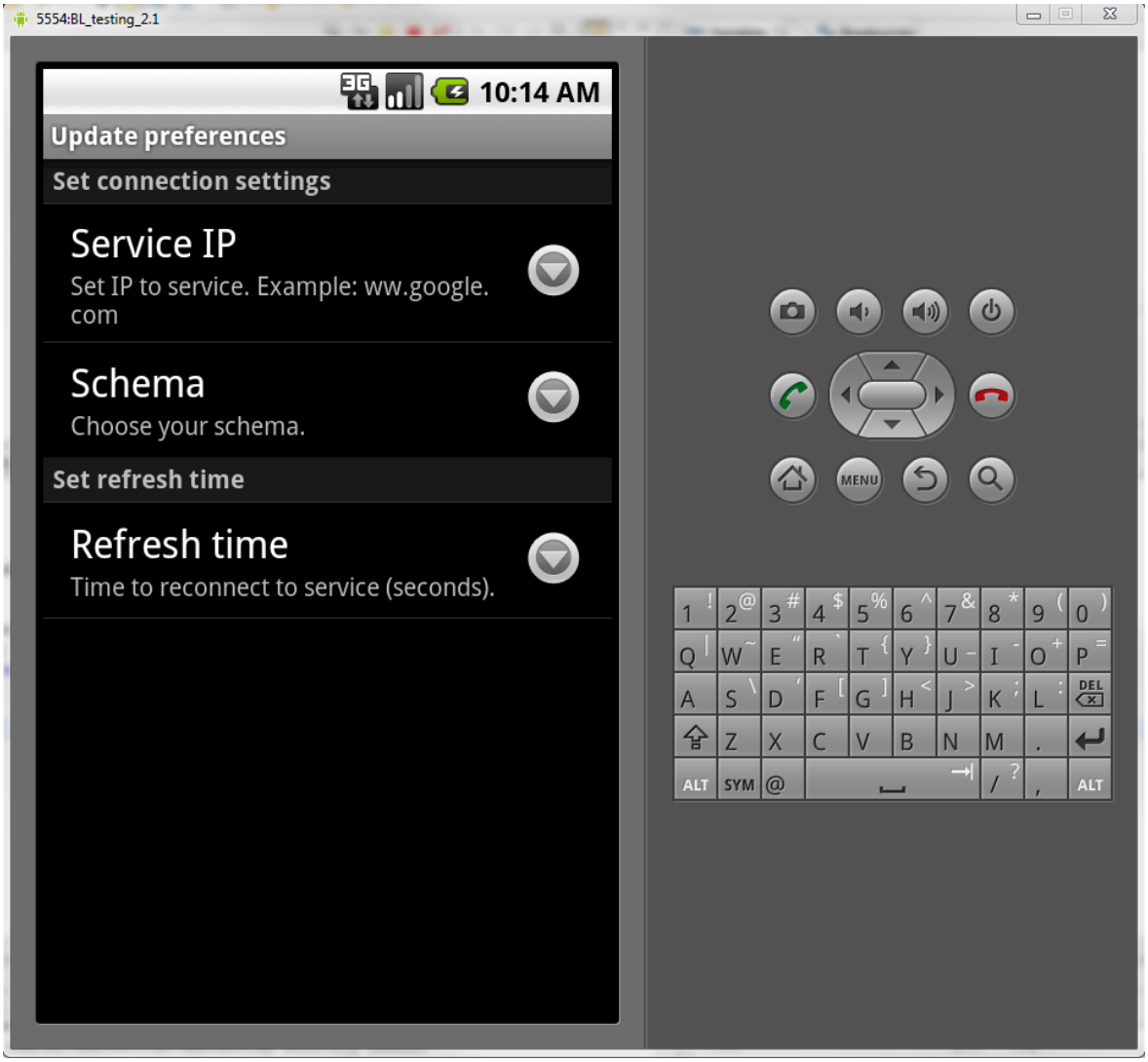
Na úvodní stránce při prvotním spuštění jdou vidět pouze texty bez hodnot a dvě tlačítka *Start* a *Stop*. Při zmáčknutí tlačítka *Menu* což je hardwarové tlačítko na mobilním telefonu se zobrazí další menu [Obr. 10] na které když se klikne, tak se otevře nová aktivita [Obr. 9] ve které může uživatel nastavit protokol přes který bude služba komunikovat, URL adresu na které je umístěna jeho služba a čas aktualizace.



Obrázek 11: Úvodní aktivita SmsCentra



Obrázek 12: Tlačítko Preferences - nastavení



Obrázek 13: Nastavení SmsCentra

3.2.1.5 Kód aplikace - SmsCentrum

Tato aplikace má stejně jako ta předchozí menu s nastavením a volá se stejným způsobem jako v předchozí aplikaci. Opět stejně jako předchozí aplikace má i dvě tlačítka ovšem funkčnost těchto dvou tlačítek se liší a to v tom, že se nespouštějí nové aktivity, ale služby.

Rozdíl mezi aktivitou a službou je v tom, že služba je něco co se vykonává na pozadí zatímco aktivita vykonává vše v popředí a navíc komunikuje s uživatelem:

```
        btnStart.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {

                        AlarmManager alarmManager =
(AlarmManager)getSystemService(ALARM_SERVICE);
                        Calendar calendar = Calendar.getInstance();
                calendar.setTimeInMillis(System.currentTimeMillis());
                        alarmManager.setRepeating(AlarmManager.RTC_WAKEUP,
calendar.getTimeInMillis(), app.getRefreshTime(), pendingIntent);
                        registerReceiver(broadcastReceiver, new
IntentFilter(MyService.BROADCAST_ACTION));
                        }
        });
```

Příklad 15: Nastavení události na tlačítko Start

K volání služby používám AlarmService, která napomáhá šetřit energii, protože mobilní telefon nevzbudí způsobem, který příliš zatěžuje baterii.

Tlačítko start nastartuje službu a ta poté vykonává další úkoly již v pozadí mezi tyto úkoly patří:

- zjistit zda je k dispozici připojení k internetu nebo službě WCF
- pokud připojení k internetu není odeslat upozornění pomocí notifikací androidu
- pokud není připojení ke službě odeslat notifikaci na androidu a odeslat SMS administrátorovi
- v případě přijmutí dat ze služby odešle informace na telefonní číslo, které získal od služby
- aktualizace grafického rozhraní

3.3 Windows Phone 7

Windows Phone 7 aplikaci se mi nepodařilo zprovoznit přes zabezpečený kanál pomocí WCF služby a také proto, že jsem se více věnoval aplikaci pro android, která mě zaujala mnohem více. Dalším důvodem byl i problém se samotným vývojovým jazykem - Silverlight. Měl jsem k dispozici server se zakoupeným certifikátem, který ovšem není podporován samotným Windows Phone - tuto informaci jsem zjistil na oficiálních stránkách MSDN. Aplikaci se mi podařilo vyzkoušet pouze přes nezabezpečený kanál, což je pro účely této bakalářské práce nepoužitelné.

Předpokládám, že v brzké době Microsoft přidá nové podporované certifikační autority. Ačkoliv seznam podporovaných certifikátů vypadá, vcelku dlouhý bohužel stále neobsahuje všechny obvykle podporované certifikační autority. Je to způsobeno hlavně tím, že zabezpečenou komunikaci mezi WP7 a WCF začal Microsoft řešit teprve koncem roku 2010 - začátkem roku 2011. Do té doby sice WP7 s WCF komunikovat dokázal, ale pouze po nezabezpečených kanálech.

Ve WIN aplikacích je řešení vcelku jednoduché za pomoci tříd, které dovolí vlastnoručně programátorovi nastavit, že důvěřuje všem certifikátům s kterými komunikuje. Tato třída bohužel ve WP7 zatím není a její vytvoření není procházka růžovou zahradou.

3.4 iPhone aplikace

iPhone aplikace stejně jako i aplikace pro android je rozdělena na dvě metody. Jedna z nich stahuje data za pomocí SOAP dotazu a druhý způsob využívá metody REST. Data se opět stahují přes zabezpečený kanál https ze služby stejně jako aplikace pro Android.

Tato je velmi jednoduchá, protože jsem se věnoval hlavně aplikaci pro Android a navíc jsem se v obou jazycích teprve učil programovat stejně jako vytvářet WCF službu. Aplikaci jsem rozdělil na dvě záložky, kde jedna slouží pro SOAP a druhá pro REST stahování dat.

3.4.1 Downloader

Aplikace jak bylo zmíněno výše stahuje informace ze služby pomocí SOAP a REST. Zde jsem nevytvářel ani žádné možnosti nastavení. Vše je nastavenou napevno v kódu.

3.4.2 Počáteční problémy

Zde bylo problémů podstatně méně a to hlavně i z toho důvodu, že aplikace není moc náročná jako je aplikace na android.

- Hlavním problémem byl nejspíše programovací jazyk a pochopení jeho syntaxe
- Druhým problémem bylo pochopit jakým způsobem se navážou vazby mezi metodou ve třídě a jednotlivými prvky v designu - události na tlačítko, atd.

- Dalším problémem bylo vybrat si zda použít synchronní nebo asynchronní komunikaci pomocí REST nakonec jsem se rozhodl k synchronní, protože byla podstatně jednodušší a pro mé účely stačila

3.4.3 Popis aplikace - Downloader

Aplikace dělá vlastně to samé co aplikace pro android. Stahuje data za pomocí REST i SOAP a zobrazuje je uživateli. Data jsou stejně jako v Androidu stahovány ze stejné služby - pouze u SOAP jsem zvolil jinou metodu, která vrací jiná data. Data jsou opět stahována skrze https protokol a proto jsou šifrována.

3.4.4 Kód aplikace - Downloader

V iPhone nevyužívám žádné knihovny třetích stran, protože jsem se rozhodl vytvořit vlastní SOAP dotaz, který jsem uložil do stringu a pouze jej upravuji podle mojí potřeby. Zbytek komunikace funguje podobně jako v Android kdy se odešle dotaz a poté se v metodě connectionDidFinishLoading získá odpověď a opět v ní můžeme data parsovat.

Nejdříve je potřeba sestavit dotaz, který zapříčiní to, že služba nám vrátí požadované informace zpět:

```
NSString *soapMessage = [NSString stringWithFormat:
@"<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n"
"<SOAP-ENV:Envelope\n"
"xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"\n"
"xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"\n"
"xmlns:SOAP-ENC=\"http://schemas.xmlsoap.org/soap/encoding/\"\n"
"SOAP-ENV:encodingStyle=\"http://schemas.xmlsoap.org/soap/encoding/\"\n"
"xmlns:SOAP-ENV=\"http://schemas.xmlsoap.org/soap/envelope/\">\n"
"<SOAP-ENV:Body>\n"
"<TestMethod xmlns=\"http://tempuri.org/\">\n"
];
 NSString *pripona = [NSString stringWithFormat:
@"</TestMethod>\n"
"</SOAP-ENV:Body>\n"
"</SOAP-ENV:Envelope>\n"];
       soapMessage = [soapMessage stringByAppendingString:pripona];

NSURL *url = [NSURL URLWithString:
@"https://findatab.fincentrum.com/AndroidWCF/Service1.svc/soap"];


NSMutableURLRequest *theRequest = [NSMutableURLRequest requestWithURL:url];
```

```
NSString *msgLength = [NSString stringWithFormat:@"%d", [soapMessage length]];


[theRequest addValue: @"text/xml; charset=utf-8"
forHTTPHeaderField:@"Content-Type"];
[theRequest addValue: @"http://tempuri.org/IService1/TestMethod"
forHTTPHeaderField:@"SOAPAction"];
[theRequest addValue: msgLength forHTTPHeaderField:@"Content-Length"];
[theRequest setHTTPMethod:@"POST"];
[theRequest setHTTPBody: [soapMessage dataUsingEncoding:NSUTF8StringEncoding]];
```

Příklad 16: Vytvoření SOAP dotazu

Poté se musí dotaz odeslat a přijmout odpověď:

```
 NSURLConnection *theConnection = [[NSURLConnection alloc]
initWithRequest:theRequest delegate:self];
        if( theConnection )
        {
                webData = [[NSMutableData data] retain];
        }
        else
        {
                NSLog(@"theConnection is NULL");
        }
```

Příklad 17: Připojení ke službě WCF

Pokud se dotaz odeslal, tak se naalokuje paměť pro uložení odpovědi od služby a pokud ne, tak se pouze vloží informace o neúspěchu do logu iPhone.

O přijetí odpovědi a dat se starají další metody:

```
didReceiveResponse:(NSURLResponse *)response
{
        [webData setLength: 0];
}
-(void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data
{
        [webData appendData:data];
}
-(void)connection:(NSURLConnection *)connection
didFailWithError:(NSError *)error
{
        NSLog(@"ERROR with theConenction");
        [connection release];
        [webData release];
}
```

Příklad 18: Získání odpovědi z WCF serveru

A nakonec metoda ve které již může uživatel nějakým způsobem získávat data z odpovědi služby tzn. parsovat data z XML:

```
- (void)connectionDidFinishLoading:(NSURLConnection *)connection {}
```

Příklad 19: Metoda pro zpracování odpovědi

Komunikace skrze REST je v iPhone podobná jako u Android využívá se zde tříd NSURL, NSURLRequest a NSURLResponse. Pomocí NSURL se uloží URL odkaz na třídu a budeme ji využívat k odeslání dotazu na server.

```
NSURL *url = [NSURL
URLWithString:@"https://findatab.fincentrum.com/AndroidWCF/Service1.svc/rest/GetXml/
new.xml"];
```

Příklad 20: Vložení URL na WCF službu

Poté se využije třídy NSURLRequest pomocí které zavoláme metodu, která nám vytvoří dotaz na URL na serveru.

```
NSURLRequest *request = [NSURLRequest requestWithURL:url];
```

Příklad 21: Vytvoření dotazu

Jakmile je dotaz vytvořen musíme jej serveru odeslat a přijmout informace, které nám server zašle.

```
plistData = [NSURLConnection
endSynchronousRequest:request
returningResponse:&response
error:&error];
```

Příklad 22: Přijmutí odpovědi WCF serveru

Data se uloží do proměnné plistData odpověď se uloží do proměnné response která je typu NSURLResponse a protože se zde může objevit i nějaká chyba je potřeba uložit id chybu do proměnné error, která je typu NSError.

Poté již jen parsujeme hodnotu odpovědi z proměnné plistData:

```
 NSPROPERTYLISTFORMAT FORMAT;
ID THEXML;
        NSSTRING *ERRORSTR;
        THEXML = [NSPROPERTYLISTSERIALIZATION PROPERTYLISTFROMDATA:PLISTDATA
    MUTABILITYOPTION:NSPROPERTYLISTIMMUTABLE
FORMAT:&FORMAT
ERRORDESCRIPTION:&ERRORSTR];
```

Příklad 23: Získání odpovědi

4 ZHODNOCENÍ ROZDÍLŮ

Rozdíl mezi platformami je vcelku markantní a to hlavně mezi androidem s WP7 oproti xCode. Nejtěžší bylo si vždy zvyknout na jiná vývojová prostředí, jiné programovací jazyky a také strukturu každé aplikace.

4.1 Vývojová prostředí

Když nepočítám s prostředím Visual studia, které používám již nějakou dobu, tak nejjednodušší bylo zvyknout si na MAC OS, protože má vše umístěno intuitivně - všechny aplikace v MAC OS jsou velice intuitivní jakmile si uživatel zvykne na poněkud netradiční způsob zobrazování jaké v MACu existuje. Neobvyklé je pouze pro uživatele Windows, uživatel MAC by zase oponoval, že netradiční zobrazování aplikací má Windows. Zastávám názor, že v jednoduchosti je síla a proto se mi asi nejvíce líbilo prostředí xCode pro Mac OS X, protože bylo velmi pěkně graficky zpracováno navíc kromě toho, že kód je oddělen od grafického rozhraní při programování, tak i nástroj pro vytváření kódu je oddělen od nástroje pro tvorbu designu. Dá se říci, že pro vytváření designu se využívá samostatného pluginu nebo nadstavby xCode.

4.2 Programovací jazyk

Zde byl asi nejpřívětivější Silverlight používaný k vytvoření aplikací pro WP7. Ačkoliv se mi nepodařilo vytvořit zabezpečené spojení mezi WP7 a WCF, kvůli nedokončenému Silverlightu, tak se mi v jazyce WP7, což je C# se Silverlightem zdálo nejjednodušší - opět to bylo nejspíše způsobeno tím, že v C# již nějakou dobu pracuji.

Nejhůře pro mě na tom byl Object-C jehož syntaxe je vcelku složitá a chvíli mi trvalo než jsem si na ni zvykl a pochopil jakým způsobem se co dělá. Práce s tímto jazykem a jeho naučení bylo asi nejsložitější a to i proto, že syntaxe jazyku C# a Java je velmi podobná - dokonce bych se nebál říci úplně stejná. Je to způsobeno i tím, že jazyk Object-C vychází z neobjektového jazyka C a jedná se vlastně o nižší programovací jazyk kde je nutná potřeba alokace paměti, což v novějších jazycích jako je C# a Java vyřešeno pomocí garbage collectoru, který uvolňuje paměť dle potřeby nebo dle využití paměti což znamená, že pokud paměť není využívána po nějakou dobu garbage collector data z paměti

odstraní a tím ji uvolní k dalšímu použití. Garbage collector v každém jazyce funguje podobně, ale uvolňuje se dle jiných podmínek.

4.3 Vytváření aplikací

Ve všech vývojových prostředích bylo možné si grafické rozhraní "naklikat", což bylo ve všech stejné, ale i přesto zde jsou rozdíly. Nejlépe na tom jsou nástroje pro tvorbu rozhraní v MS Visual studiu a xCode kde je vyřešeno jednoduše i pozicování jednotlivých prvků. Jednoduše se vezme např. tlačítko a umístí se přesně tam kde jej chceme. To v Androidu si musí uživatel dát pozor jaký layout vytváří. Pokud vytvoří linear layout, tak v něm není možné prvky pozicovat libovolně.

Mezi jazyky Java a C# není zase takový rozdíl v syntaxi a programovat v nich byla radost, ale Object-C byl ze začátku velmi nepřívětivý a to z toho důvodu, že celá jeho syntaxe je od těchto dvou jazyků odlišná, což bylo vysvětleno v předchozí kapitole.

Android je velmi mladým zástupcem na poli mobilních aplikací a velmi rychle se vyvíjí, proto během práce na bakalářské práci byl vývojový nástroj několikrát upraven a v mnoha ohledech i vylepšen. Vylepšení bylo zřetelné hlavně ve vytváření uživatelského rozhraní, které dostálo mnoha změn ačkoliv ne všechny tyto změny byly k lepšímu. Ostatní vývojová prostředí jsou na poli mobilních zařízení již delší dobu. WP7 je sice inovační novinkou od Microsoftu a využívá nových technologií narozdíl od staré verze Windows pro mobilní zařízení - Windows Mobile, ale i přesto zde jsou vidět zkušenosti z předešlých verzí a také dalších technologií uvedených na trh Microsoftem.

4.4 Komunikace se službou

Připojení ke službě bylo v každé službě uděláno trochu jinak. Ve VS pro komunikaci skrze SOAP stačilo vložit odkaz na tuto službu, protože WCF je na SOAP komunikaci postaveno, proto jakmile se vložil odkaz na službu a vytvořila se její nová instance bylo možné využívat všech metod této služby. Pro získávání dat pomocí REST je ve všech platformách potřeba nějakého HTTP klienta, protože REST získává data přes URL - tzn. když se zadá správná URL k metodě služby uživatel získá data. Tento způsob je také nejjednodušší právě ve VS od společnosti Microsoft.

V Androidu jsem použil ksoap2 knihovnu, protože v něm nelze vložit odkaz na službu a poté vytvořit její instanci a využívat metody. Zde se musela použít knihovna třetí strany z toho důvodu, protože zde nenalezneme nativní knihovnu pro práci se SOAP. Ksoap2 knihovna slouží k jednoduchému vytvoření SOAP dotazu na server, díky kterému službě řekneme jakou metodu má zavolat s jakými parametry a on nám podle tohoto SOAP dotazu vrátí SOAP odpověď. Dotaz pomocí REST je vytvořen podobně jako u VS tzn. za pomoci HTTP klienta.

V iOS jsem již nepoužil žádnou knihovnu pro vytvoření SOAP dotazu, ale samotný dotaz jsem vytvořil. Nevýhodou je to, že není jednoduché si celý dotaz upravit v případě změn. Navíc celý dotaz zabírá spoustu místa v pracovní ploše a v případě zadávání některých parametrů od uživatele je potřeba správně poskládat celý dotaz. Tohle všechno jsem nemusel řešit když jsem používal knihovnu v Androidu neboť vše bylo již obsaženo v knihovně, která po zadání správných údajů a parametrů vytvořila dotaz sama.

4.4.1 SOAP vs REST

SOAP i REST slouží ke komunikaci, ale stejně jako ve všem každý z nich má své pro a proti. A nejinak je tomu i mezi různými mobilními zařízeními. SOAP díky množství poskytovaných informací o datových typech, které přenáší je mnohem náročnější na přenos časově i paměťově.

REST může využívat XML stejně jako SOAP, ale i JSON, který se používá mnohem častěji a také například ATOM/RSS - obojí je také založeno na XML. JSON je méně náročnější na přenos, protože nepřenáší tolik informací jako SOAP např. nepřenáší datové typy. REST se v poslední době začíná čím dál tím více používat a získává si větší a větší oblibu, ale i přesto se stále nejvíce používá komunikace za pomocí SOAP a to i z toho důvodu, že rychlost Internetu, kapacita pamětí a rychlost procesorů v dnešních počítačích i mobilních zařízeních jsou již na takové úrovni, že SOAP není problém využívat.

Jak bylo řečeno výše SOAP využívá WCF služba, která vytvoří seznam metod a potřebných parametrů pomocí WSDL což je popisný jazyk využívaný k poskytnutí informací klientům využívajících webové služby. WSDL jazyk nevyužívá pouze WCF, ale lze jej využít i při vytváření Java služeb. WSDL je totiž popisný jazyk, který je vytvořen za pomoci XML a proto mu rozumí všechny platformy dostupné v dnešní době.

ZÁVĚR

Ve své bakalářské práci jsem se zaměřil hlavně na přenos informací pomocí protokolu HTTPS a to z toho důvodu, že je to dle mého názoru nejjednodušší a také nejpřístupnější možnost bezpečného přenosu dat z mobilních zařízení a ne jen pouze z nich. HTTPS se využívá nejčastěji právě na mobilních zařízeních, protože mnoho aplikací získává data z jiných serverů, na kterých bývají nejčastěji webové služby komunikující právě přes tento protokol.

Snažil jsem se vytvořit pro 3 nejpoužívanější platformy jednoduchou ukázkovou aplikaci pro komunikaci s webovou službou vytvořenou pomocí WCF. Tyto 3 platformy měli být Android, iOS a WP7, ale bohužel se mi kvůli nepodporovanému certifikátu nepodařilo připojit ke službě z WP7 a nepodařilo se mi v době vytváření aplikací pro mobilní zařízení naleznout řešení, aby se mi podařilo tento problém obejít. Důvod je ten, že Microsoft podle zdrojů, které jsem nalezl započal vývoj zabezpečené komunikace z WP7 teprve začátkem roku 2011.

Služba vytvořená ke komunikaci získává data z XML souboru, který je umístěn na serveru. Aplikace se ke službě pouze připojí a služba jim předá obsah XML souboru. Se kterým dále pracuje mobilní zařízení. Obsah se přesouvá po zabezpečeném protokolu HTTPS, což je protokol http, který využívá protokolů SSL/TLS pro zašifrování přenášených dat.

Vytvoření služeb pomocí REST i SOAP je hodně podobné, ale obojí s sebou nese určité problémy. Pokud potřebujete co nejmenší zátěž pro linku a paměť určitě je lepší využít REST, ale pokud potřebujeme znát více informací jako například datové typy nejlepší možnost je použít SOAP.

CONCLUSION

I focused to data transfer over HTTPS protocol, because I think it is the easist and most accessible way to data transfer from mobile devices and not only from them. HTTPS protocol is used on mobile devices because lot of applications get data from web services located on servers.

I tried to create applications on three common used paltforms to communicate with WCF web-service. This three platforms may be Android, WP7 and iOS, but because of problem with certificate on WP7 I could not find a solution for this problem to round it. Reason is that the Microsoft starts support ssl/tls communication on WP7 at the beginning of 2011.

WCF web service created to communicate get data from XML file located on server. Applications are connecting to the WCF service, this service forward XML to this application and the rest is happend on applications. Content is passed to mobile device throught https protocol which is http protocol enriched by SSL/TLS, this protocol provides encryption of transmitted data.

Creation of services using SOAP or REST is similar, but both have their problems. If user needs less bandwith and memory usage than th REST is the best way, but we need to know more informations such us data type the better choice is SOAP.

SEZNAM POUŽITÉ LITERATURY

[1] MEIER, Reto. Professional Android™ Application Development. , Indianapolis, Indiana : Wiley Publishing, Inc., 2009. 439 s. ISBN: 978-0-470-34471-2.

[2] MEIER, Reto. Professional Android™ 2 Application Development, Indianapolis, Indiana : Wiley Publishing, Inc., 2010. 576 s. ISBN: 978-0-470-56552-0.

[3] BERTINO, Elisa, MARTINO, Lorenzo D., PACI, Federica, SQUICCIARINI, Anna C. Security for Web Services and Service-Oriented Architectures, 2010, Springer, 218 s. ISBN 978-3-540-87742-4.

[4] ALI, Maher. iPhone SDK 3 Programming. Wiley Publishing, Inc., 2009, 672 s. ISBN 978-0-470-68398-9.

[5] PETZOLD, Charles. Programming Windows Phone 7, Microsoft press, 2010, 1012 s. ISBN 978-0-7356-4335-2.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ADT	Android Development Tools
HTTP	Hypertext Transport Protocol
HTTPS	Hypertext Transport Protocol Secure
IIS	Internet Information Services
iOS	Operační systém od Apple pro mobilní zařízení
JAR	Java Archive
JSON	JavaScript Object Notation
MSDN	Microsoft Developer Network
MVC	Model-View-Controller
REST	Representational State Transfer
RPC	Remote Procedure Call
RSS	Rich Site Summary
SDK	Software Development Kit
SSL	Secure Sockets Layer
SOAP	Simple Object Access protocol
TLS	Transport Layer Security
UI	User Interface
WCF	Windows Communication Foundation
WDDX	Web Distributed Data eXchange
WP7	Windows Phone 7
WSDL	Web Service Description Language
XML	Extensible Markup Language

SEZNAM OBRÁZKŮ

Obrázek 1: Struktura aplikace v Android 13
Obrázek 2: Životní cyklus aktivity 15
Obrázek 3: Životní cyklus služby 16
Obrázek 4: Struktura iOS aplikace 19
Obrázek 5: Příklad service-oriented architektury 22
Obrázek 6: Struktura MVC 30
Obrázek 7: Hlavní menu Android 39
Obrázek 8: Menu možností 40
Obrázek 9: Seznam prvků 41
Obrázek 10: Detail jednoho z prvků 44
Obrázek 11: Úvodní aktivita SmsCentra 47
Obrázek 12: Tlačítko Preferences - nastavení 48
Obrázek 13: Nastavení SmsCentra 49

SEZNAM PŘÍLOH

P I Přílohou práce jsou zdrojové kódy, dostupné na CD.