VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

BEZPEČNÁ KOMUNIKAČNÍ APLIKACE PRO WINDOWS PHONE

SAFE COMMUNICATION APP FOR WINDOWS PHONE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ VÍCHA**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Vícha Tomáš**
Obor: Informační technologie
Téma: **Bezpečná komunikační aplikace pro Windows Phone**
Safe Communication App for Windows Phone
Kategorie: Uživatelská rozhraní

Pokyny:

1. Prostudujte a popište existující řešení pro bezpečnou komunikaci na mobilních zařízeních.
2. Navrhněte funkčnost aplikace pro komunikaci na Windows Phone s point-to-point šifrováním.
3. Prototypujte prvky uživatelského rozhraní a funkčnosti řešené aplikace. Testujte prototypy na uživatelích.
4. Vytvořte řešenou aplikaci a testujte ji na uživatelích.
5. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2, značné rozpracování bodu 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Herout Adam, prof. Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2
L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cílem této práce je popis vývoje aplikace pro bezpečnou komunikaci mezi dvěma uživateli. Popis začíná analýzou již existujících řešení, pokračuje návrhem uživatelského rozhraní a stanovením cílových požadavků. Dále se zde popisuje výběr vhodných technologií a samotné etapy vývoje aplikace společně s jejím testováním. Výsledkem práce je uživatelsky přívětivá aplikace pro platformu Windows Phone, která provádí koncové šifrování jednotlivých zpráv.

Abstract

The aim of this thesis is to describe the application development for secure communication between two users. The description begins with the analysis of already existing solutions, continues with the design of the user interface and the determination of requirements. There is also described a selection of suitable technologies and stages of application development together with its testing. The result of this work is an user-friendly application for the Windows Phone platform that performs end-to-end encryption of individual messages.

Klíčová slova

Windows Phone 8.1, WP 8.1, C#, XAML, Silverlight 8.1, .NET, LINQ, XMPP, Ejabberd, JID, MVVM, uživatelské rozhraní, mobilní aplikace, bezpečná komunikační aplikace, šifrování, RSA, AES, TLS, koncové šifrování, SafeChat

Keywords

Windows Phone 8.1, WP 8.1, C#, XAML, Silverlight 8.1, .NET, LINQ, XMPP, Ejabberd, JID, MVVM, user interface, mobile application, secure communication application, encryption, RSA, AES, TLS, end-to-end encryption, SafeChat

Citace

VÍCHA, Tomáš. *Bezpečná komunikační aplikace pro Windows Phone*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Bezpečná komunikační aplikace pro Windows Phone

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Ing. Adama Herouta Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Vícha
17. května 2017

Poděkování

Rád bych poděkoval vedoucímu této práce panu prof. Ing. Adamu Heroutovi Ph.D. za odborné vedení a podnětné rady při vývoji aplikace. Dále bych tímto chtěl poděkovat i všem zúčastněným testovacím uživatelům za ochotu a cenné připomínky. Můj dík patří i mé rodině a přátelům, kteří mi byli při tvorbě práce velkou oporou.

Obsah

Kapitola 1

Úvod

V této závěrečné práci se zabývám vývojem mobilní aplikace pro operační systém Windows Phone. Kýžená aplikace by měla umožňovat komunikaci mezi dvěma uživateli v zabezpečené formě a tím uchránit obsah zpráv před neoprávněným přečtením třetí stranou. Protože se pro přenos zpráv využívá celosvětová síť internet, již z její povahy nelze zaručit nedotknutelnost tajemství obsahu jednotlivých zpráv. Existuje mnoho subjektů, které můžou chtít nahlédnout do obsahu rozhovoru mezi dvěma lidmi. Může se jednat o zvědavce, který je připojen na stejnou nezabezpečenou bezdrátovou síť, až po zločinecké struktury toužící po citlivých datech. Je proto pochopitelné, že lidé chtějí využívat nástroje, které jim umožnují držet jejich korespondence s druhými v tajnosti.

Vývoj aplikace začal seznámením se s historií mobilního operačního systému Windows Phone a jeho specifickým a pro mnohé fascinujícím pojetím uživatelského rozhraní. Dále bylo nutné se seznámit i prostředky, které umožňují samotnou tvorbu aplikací pro tuto mobilní platformu. Důležité je i seznámení se s protokolem, který je využit pro přenos samotných zpráv a také to, jak tyto zprávy zabezpečit. Nedílnou součástí samotného vývoje je také prostudování již existujících řešení a jejich zhodnocení. Návrh budoucí aplikace je také nutné představit budoucím uživatelům a na základě jejich připomínek tento návrh vyladit. Samostatnou kapitolou je pak vlastní implementace aplikace.

Kapitola 2

Použité technologie

V této kapitole, jež shrnuje získané teoretické znalosti, se věnuji technologiím, se kterými jsem se při studiu problematiky vývoje mobilních aplikací setkal. Jako první bych rád uvedl cílovou platformu, kterou jsem si vybral pro implementaci své mobilní aplikace. Touto platformou je operační systém Windows Phone, u kterého se zabývám jeho vývojem a pro tuto platformu specifickým grafickým rozhraním. Důvodů, proč jsem si zvolil právě platformu Windows Phone, je více, ale ten nejhlavnější je, že jsem vlastníkem chytrého telefonu právě s tímto operačním systémem a začal mě poněkud znepokojovat vývoj kolem dostupných aplikací. Tento vývoj souvisí s trendem klesajícího celosvětového podílu této platformy na trhu s mobilními operačními systémy. Dále se v této kapitole věnuji prostředí pro vývoj aplikací na platformu Windows Phone, protokolu XMPP a metodám pro zabezpečení přenášených zpráv.

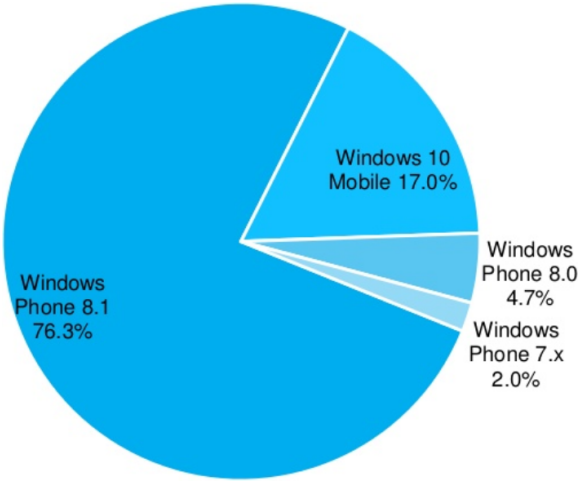
2.1 Mobilní platforma Windows Phone

Pro implementaci aplikace jsem si po předchozí rozvaze vybral verzi systému Windows Phone 8.1. Moje rozhodnutí bylo ovlivněno jak z hlediska zastoupení jednotlivých verzí mobilního operačního systému, tak i z hlediska dostupných knihoven, které mi pomohly s vývojem aplikace. V současné době (statistiky z února 2017) má podle společnosti AdDuplex největší zastoupení na trhu právě verze Windows Phone 8.1. Na druhém místě se umístil Windows 10 Mobile, který ovšem zahrnuje zpětnou kompatibilitu pro aplikace tvořené pro starší verze mobilních Windows. Zbytek zastoupení tvoří starší systémy, které dohromady obsazují necelých sedm procent trhu [7]. Těchto sedm procent jsem se rozhodl zanedbat. Podíly v zastoupení verzí systému je možné vidět na obrázku 2.1.

2.1.1 Uživatelské rozhraní

Uživatelské rozhraní operačních systémů Windows Phone je založeno na novém konceptu, jež se jmenoval Metro UI, ale kvůli pozdějším sporům o ochrannou známku s německou společností Metro AG byl název změněn na Modern UI (dále již pouze toto označení). Toto označení má symbolizovat nadčasovost rozhraní. Původní název měl také svoji symboliku, protože design rozhraní je inspirován stylem ukazatelů a tabulí používaných pro navigaci v oblasti veřejné dopravy, jako je například podzemní dráha, či letištní haly.

První použití filozofie nového rozhraní, jež je využití textu jako primárního prvku pro navigaci, lze nalézt již v programu Windows Media Center for Windows XP Media Center Edition vydaného v roce 2002. Jednalo se o multimediální přehrávač, který v sobě obsahoval



Obrázek 2.1: Zastoupení jednotlivých verzí mobilního operačního systému Windows podle agentry AdDuplex (únor 2017) [7].

možnost přehrávání a nahrávání televizního vysílání z příslušné rozšiřující karty počítače. Kromě lokálního multimediálního obsahu bylo možné přehrávat i obsah ze streamovacích služeb jako je například Netflix.

Dalším produktem, který již více rozvinul koncept dnešního Modern UI byl přenosný multimediální přehrávač Zune. Ten měl být odpovědí společnosti Microsoft na úspěšnou řadu přehrávačů iPod společnosti Apple. Vývojáři se zde snažili zaměřit hlavně na využití textu jako hlavního prvku celého uživatelského rozhraní, místo používání ikonek a dalších komponent, které by kazily celkovou čistotu prostředí. Myšlenky využité právě u produktů rodiny Zune byly následně použity při tvorbě uživatelského rozhraní pro systém Windows Phone 7.

Jak již bylo zmíněno, Modern UI klade velký důraz na využití typografie a snaží se více soustředit na samotný obsah. Pro výchozí písmo byla vybrána rodina Segoe, jež se poprvé objevila v operačním systému Windows Vista a pro využití v systémech Windows Phone byla drobně upravena jako Segoe WP. Typické je také užití velkých písmen, hlavně pro nadpisy a názvy, od této praktiky se ovšem postupně začalo ustupovat a to hlavně z důvodů zbytečného plýtvání místem. Důležitou součástí jsou také animace a přechody mezi jednotlivými sekcemi rozhraní, které mají vypadat přirozeně pro daný typ akce, kterou uživatel vykonává [13].

Středobodem rozhraní operačních systémů rodiny Windows Phone, ale i Windows 10 Mobile je úvodní obrazovka, která se skládá ze dvou částí. První část je samotná úvodní stránka, na kterou si uživatel přidává jednotlivé zástupce v podobě dlaždic. Ty můžou nabývat tří základních velkostí. Dvě tyto velikosti jsou čtvercové a jedna obdélníková. Dlaždice se umisťují do neviditelné mřížky a vyplňují tak plochu obrazovky. Dodatečná funkčnost, kterou mohou tito zástupci mít se nazývá živá dlaždice (anglicky Live Tile). Ta umožňuje, aby aplikace zobrazovala na těchto prvcích dodatečné informace, například počet událostí nebo aktuální povětrnostní podmínky v případě aplikace pro předpověď počasí. Tyto dodatečné informace mohou být jak v textové, tak i grafické podobě. Druhou části hlavní obrazovky je i seznam všech nainstalovaných aplikací, který je zobrazitelný tahem prstu po displeji zařízení zprava doleva.

2.1.2 Přehled verzí operačního systému

Operační systém Windows Phone si za svůj život prošel zajímavým a dynamickým vývojem. Od první verze, která nesla označení sedm, až po univerzální pojetí v podobě Windows 10. S postupujícím časem docházelo k rozšiřování funkcionality a podporování nových technologií. V následujícím textu jsou shrnuty podstatné milníky vývoje tohoto systému [1].

Windows Phone 7

První verze byla představena 15. února 2010 na Mobile World Congress v Barceloně a první telefony s tímto systémem se začaly prodávat 21. října téhož roku. Systém již nepočítal s využíváním stylusu (dotykové pero) jakožto dominantního prvku pro ovládání, ale spoléhal se výhradně na využití prstů uživatelů. Již od první verze podporoval vícedotykové ovládání. Důležitou vlastností tohoto systému jsou takzvaná centra (anglicky hubs), která slučovala aktivitu uživatelů do jedné aplikace. Za zmínku jistě stojí centrum lidé (anglicky People Hub), které slučovalo aktivity na přidružených sociálních sítích uživatele a ten k nim měl přístup z jednoho místa, což bylo pro mnohé velice vítanou funkcionalitou.

Aby společnost Microsoft zajistila dobrou uživatelskou zkušenost se svým operačním systémem, rozhodla se již od počátku stanovit poměrně přísné minimální požadavky na zařízení, na kterých měl být operační systém provozován. Jednalo se například o minimální velikost paměti RAM, která byla původně stanovena na 512 MB. Později ovšem společnost Microsoft vyslyšela proseb výrobců zařízení a s aktualizací, jejíž kódové označení bylo Tango, umožnila běh systému i na zařízeních s velikostí operační paměti pouhých 256 MB.

Windows Phone 8

Nová verze Windows Phone byla vydána 29. října 2012 a na rozdíl od předchozí verze, která podobně jako Windows Mobile stavěla na jádru Windows CE, stavěla již tato verze na základech Windows 8. Díky sblížení mobilního a plnohodnotného systému přibyla programátorům možnost jednodušší portace aplikací mezi jednotlivými systémy.

Nová mobilní Windows také začala podporovat více jádrové procesory, což s novou architekturou systému přineslo svižnější běh aplikací. Přibyla podpora pro externí uložiště ve formě microSD karet, kam bylo možné ukládat hudbu, videa a obrázky. Zvýšilo se také množství možných rozlišení displeje zařízení, oproti jednomu (800×480 px) u Windows Phone 7, rovnou na čtyři.

Evoluce se dostalo i uživatelskému rozhraní. Byla upravena úvodní obrazovka, na kterou bylo možné připínat novou čtvrtinovou velikost dlaždice, která ve své živé verzi zobrazuje počet událostí.

Z důvodů použití nové architektury již nebyla zajištěna kompatibilita se zařízeními poháněnými starším systémem. Ty tudíž již nebylo možné povýšit na aktuální verzi. Vznikla pro ně proto poslední aktualizace, a to Windows Phone 7.8. Tato aktualizace přinesla upravenou úvodní obrazovku a další prvky z novějšího systému.

Windows Phone 8.1

Tak jako vyšla velká aktualizace pro Windows 8 na počítačích, tak 14. dubna 2014 byla vydána i velká aktualizace mobilních Windows. Aktualizace už byla dostupná pro všechna zařízení s předchozí verzí systému a záleželo pouze na výrobci daného zařízení, jestli se rozhodne umožnit majitelům telefonů přechod na tuto verzi.

Mezi hlavní novinku patří přidání notifikačního centra podobného tomu, které obsahují konkurenční mobilní operační systémy v čele s Androidem. V předchozích verzích systému Windows Phone byl uživatel nucen buďto reagovat na notifikaci okamžitě, nebo se spolehnout na živou dlaždici dané aplikace, že jej o události informuje. Nyní si již uživatel mohl zobrazit přetažením prstu z horního okraje displeje notifikační centrum, kde viděl všechny neobsloužené události. Dále v tomto centru byla přidána sada rychlých přepínačů pro vypínání a zapínaní bezdrátových technologií a podobně.

Další novinkou byla možnost nastavení obrázkového pozadí dlaždic na úvodní obrazovce. Doposud byla jedinou možností plná výplň, která byla závislá na zvoleném systémovém motivu (barvě).

Pro vývojáře byla představena možnost tvorby tzv. univerzálních aplikací, které můžou sdílet poměrnou část aplikační logiky jak pro mobilní, tak i pro stolní systém Windows 8.1. Podmínkou ovšem je, aby vývojář vytvořil dvě uživatelská rozhraní.

Windows 10 Mobile

Zatím posledním vývojovým stupněm mobilních Windows je verze Windows 10 Mobile, ta si klade za cíl dovést myšlenku univerzálních aplikací k dokonalosti, tím, že se daná aplikace má vždy přizpůsobit koncovému zařízení, ať už se jedná o chytrý mobilní telefon, počítač, herní konzoli nebo brýle pro rozšířenou realitu.

Společnost Microsoft spolu s Windows 10 také podstatně změnila uživatelské rozhraní systému a aplikací. Adoptovala kontroverzní ovládací prvek tzv. hamburger menu, který do jisté míry přebírá funkcionalitu spodního aplikačního panelu typického pro starší verze Windows Phone.

2.1.3 Vývoj aplikací pro platformu Windows Phone

Aplikační modely a programovací jazyky

S příchodem Windows Phone 8.1 umožnila společnost Microsoft vyvíjet mobilní aplikace pomocí dvou různých modelů. Důvodem byla snaha o unifikaci prostředí a vyvíjení takzvaných univerzálních aplikací, které by sdílely podstatnou část aplikační logiky mezi verzemi pro mobilní telefony a počítače. Tyto dva modely se jmenují Silverlight a Windows Runtime.

Silverlight 8.1, je nástupcem původního modelu dostupného již od první verze Windows Phone 7 a také se jedná o posledního zástupce rozšiřujícího funkcionalitu, která čerpá z nového systému. Zajišťuje zpětnou kompatibilitu pro aplikace a knihovny vytvářené pro starší verze. Tvorba aplikací je možná v jazycích C# nebo Visual Basic s využitím .NET frameworku a XAML, tak i v C++ s využitím grafického API DirectX. Výsledný balíček aplikace má koncovku XAP.

Windows Runtime (zkráceně WinRT), jakožto druhý model, přináší API známé z vývoje aplikací pro Windows Store pro počítače na mobilní zařízení. Jedná se o model pro vývoj aplikací, který se snaží sjednotit API pro mobilní, i pro plnohodnotné Windows a tím programátorům usnadnit vývoj aplikací na více platformách současně. Mobilní verze Windows Runtime postrádá oproti té pro počítače některá API, která na telefonech nejsou využita, (například pro tisk). Verze pro počítače také postrádá některé funkce, například pro telefonování. Při vývoji lze používat stejné jazyky jako u Silverlight 8.1 a navíc lze využívat i psaní aplikací v jazycích HTML/Javascript. Ovšem oproti aplikacím psaným v Silverlight 8.1 postrádá WinRT možnost použití schránky ke kopírování textů, API pro VoIP a

také zde není možný běh aplikací s uzamčenou obrazovkou. Koncovka balíčků vzniklých vývojem aplikací pomocí tohoto modelu je APPX [2].

XAML

XAML (zkratka pro Extensible Application Markup Language) je deklarativní značkovací jazyk vycházející z jazyka XML. Slouží pro definici grafického rozhraní aplikace a tím ulehčuje její vývoj, protože se jednotlivé prvky rozhraní nemusí vytvářet programově. Každý prvek uživatelského rozhraní je definovaný pomocí elementu a jeho vlastnosti pomocí příslušných atributů.

Ke každému souboru s definicí uživatelského rozhraní pomocí jazyka XAML připadá i soubor s jeho aplikační logikou, kterému se v anglickém jazyce říká code-behind file. Aplikační logika může přímo přistupovat k jednotlivým elementům a měnit jejich vlastnosti, také se stará o zpracování událostí vzniklých interakcí s uživatelským rozhraním. Pro pohodlnější přístup se jednotlivé elementy mohou pojmenovat pomocí atributu x:Name [6].

Návrhový vzor Model-View-ViewModel

Hlavní myšlenkou návrhového vzoru Model-View-ViewModel je oddělení aplikační logiky od uživatelského rozhraní a od samotných dat [4]. Jedná se de facto o variaci na návrhový vzor Model-View-Controller. Pomocí oddělení dochází k zpřehlednění kódu a jeho údržba se stává méně náročnou. Použití tohoto návrhového vzoru je téměř nutností při práci v týmech, kdy se o tvorbu uživatelského rozhraní stará někdo jiný než o aplikační logiku.

Popis jednotlivých částí MVVM:

- **Model** – Stará se o uchovávání dat aplikace, nejčastěji se jedná o databázi. Model v žádném případě nesmí vědět nic o View
- **View** – Jedná se o samotné grafické rozhraní aplikace, nejčastěji je tvořeno deklarativními značkovacími jazyky jako je XAML
- **View-Model** – Hlavní součást návrhového vzoru MVVM, propojuje View a Model, stará se o zpracování dat, reaguje na události a obstarává celkovou logiku

Vývojové prostřední a ladící nástroje

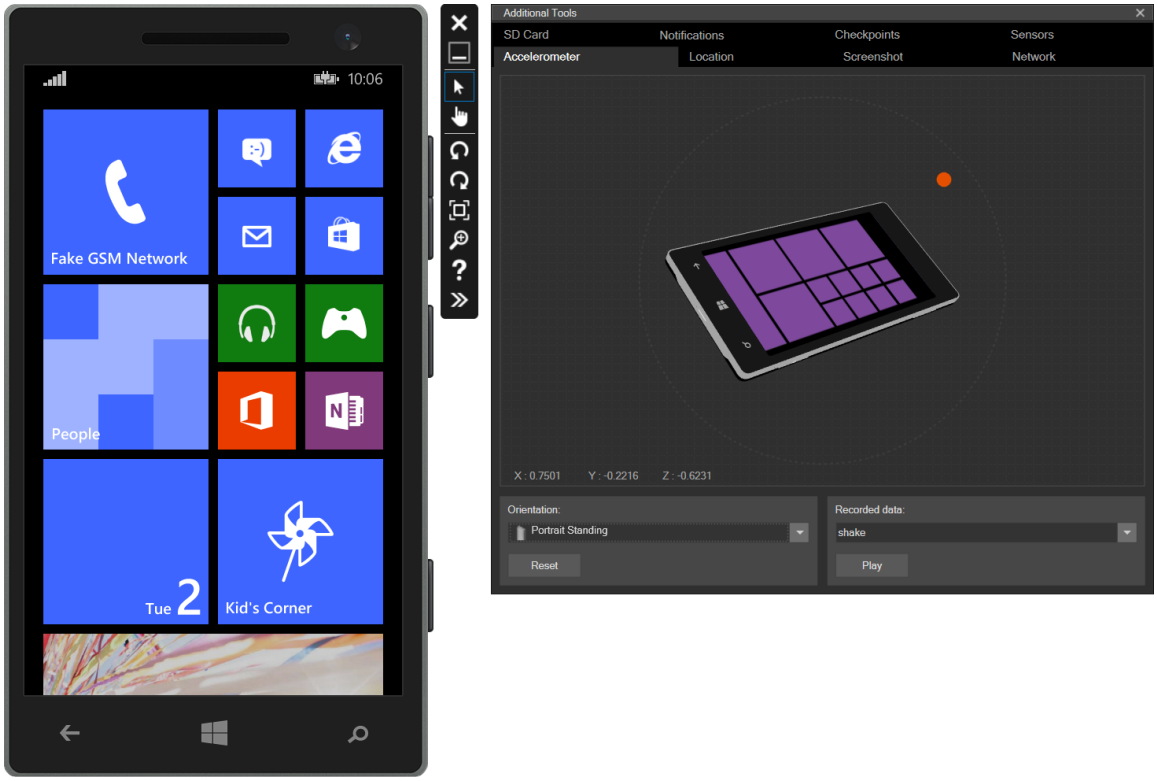
Vývoj aplikací pro platformu Windows Phone probíhá v prostředí Visual Studio. Při jeho instalaci je potřeba zatrhnout volbu pro nástroje Windows Phone 8.0 a 8.1, tyto nástroje lze stáhnou i samostatně jako SDK (Software Development Kit). Dále je potřeba se pro vývoj registrovat u společnosti Microsoft.

Pro testování aplikace lze použít emulátor operačního systému Windows Phone, který se chová jako skutečný telefon kromě možnosti telefonování. Pomocí nástrojů v prostředí emulátoru lze simulovat různé chování zařízení, například natočení v prostoru, GPS pozici, přítomnost a typ mobilní sítě a podobně. Výhodou je, že lze emulátor spustit s různou velkostí displeje a jeho rozlišením, případně i s jinou velikostí operační paměti RAM. Tohoto je dobré využít pro testování chování vytvářeného uživatelského rozhraní na různých typech zařízení.

Další možností pro testování je využití fyzického zařízení, které se musí připojit k počítači pomocí portu USB. Před samotným nahráním aplikace na fyzický telefon musí dojít k

jeho registraci jako vývojářského zařízení pomocí nástroje Windows Phone Developer Registration. Po registraci je také možné nahrávat i jiné aplikace, které nepocházejí z oficiálního obchodu.

Prostředí Visual Studia nabízí velice užitečný interaktivní ladící nástroj, díky kterému lze spuštěnou aplikaci běžící v emulátoru nebo na fyzickém zařízení pozastavit a prohlížet si obsah objektů v paměti. Obsah v paměti lze také modifikovat či přímo zadat příkaz jazyka, který se okamžitě provede. Toto zákonitě ovlivní následující běh samotné aplikace, a proto je nutné takovéto změny dělat obezřetně.



Obrázek 2.2: Ukázka prostředí emulátoru pro Windows Phone 8.1 s oknem pro nastavení parametru virtuálního zařízení.

Balíčkovací systém NuGet

Při vývoji aplikace je někdy potřeba využít knihovnu vytvořenou někým jiným. Tuto knihovnu je potřeba zahrnout do projektu a postarat se o potřebné odkazy na ní. NuGet je rozšíření pro Visual Studio, které se stará právě o stažení a začlenění knihovny třetí strany do projektu. Také se stará o to, aby daná knihovna byla v nejnovější dostupné verzi. Při odebrání knihovny zajistí zrušení všech změn v projektu, které se udály při jejím přidání.

Vyhledávání dostupných knihoven a výběr potřebné verze je možný pomocí galerie přístupné přes webové stránky projektu NuGet[1] nebo pomocí správce balíčků ve Visual Studiu. Kromě grafického prostředí je možné použít i konzoli.

[1] https://www.nuget.org/

2.2 XMPP

Pro potřeby zasílání zpráv mezi jednotlivými klienty existuje mnoho proprietárních protokolů, které svojí uzavřeností neumožnují rozšiřování jejich funkcionality třetí stranou a také neumožňují snadný vývoj aplikací bez znalosti jejich fungování. Naproti tomu existují i otevřené protokoly a jedním z nich je právě XMPP neboli Extensible Messaging and Presence Protocol [9]. Jedná se o otevřený protokol sloužící pro výměnu zpráv jak ve formě komunikace mezi jednotlivými uživateli (Instant Messaging), tak i pro potřeby výměny informací mezi aplikacemi.

Jak už vyplývá z názvu, protokol neslouží jenom pro výměnu zpráv mezi jednotlivými klienty, ale také pro řízení přítomnosti, kdy uživatel oznamuje svůj stav (připojen, neaktivní a podobně) a tento stav je poté možné získat ostatními klienty, kteří podle něj mohou reagovat.

XMPP je definován jako standard internetu a je popsán v dokumentech RFC 3920 pro obecnou specifikaci protokolu a také RFC 3921, který popisuje samotný instant messaging a zobrazení stavu. Je založen na značkovacím jazyce XML, a proto je také snadno rozšiřitelný, jak napovídá první písmeno zkratky protokolu. Rozšíření jsou realizována formou XEP (XMPP Extension Protocol), je jich velké množství a některé jsou také zakotveny v dokumentech RFC. V současné době se o vývoj protokolu stará skupina XMPP Standards Foundation.

Princip XMPP

XMPP je protokol aplikační vrstvy síťového modelu a na transportní vrstvě se spoléhá na protokol TCP. Oproti proprietárním řešením je síť využívající tento protokol decentralizovaná podobně jako email, se kterým sdílí ještě podobný tvar uživatelských identifikátorů. Identifikátor uživatele se jmenuje Jabber ID (zkráceně JID) a je tvořen samotným uživatelským jménem, znakem zavináč, doménou serveru, a nakonec nepovinným identifikátorem klienta, který se uvádí za znakem lomítka ihned po doméně. Možnost rozdělit si své uživatelské jméno pro více klientů je vhodné pro odlišení místa, kde se uživatel přihlašuje (například doma nebo v kanceláři). Uživatel může být v jeden čas přihlášen na vícero zařízeních a zprávy se posílají na určité zařízení podle jeho priority nebo při uvedení celého JID včetně identifikace zařízení odesilatelem.

Při vytváření nového XMPP účtu v síti internet má uživatel možnost se registrovat u již existujících serverů nebo si založit vlastní. Pokud si uživatel založí vlastní server, může a nemusí jej napojit na ostatní. Servery standardně naslouchají na portu 5269 pro komunikaci mezi sebou a na portu 5222 pro připojení klienta.

Při komunikaci se klient vždy připojuje na vlastní server, který jej ověřuje, na něj pak posílá zprávy. Pokud je zpráva adresovaná někomu, kdo má účet na jiném serveru, tak se servery spojí mezi sebou a vymění si zprávy adresované uživatelům toho druhého. Může nastat situace, že je server blokován (je na černé listině), v tom případě není zpráva předána, ale rovnou smazána. Když server obdrží zprávu pro uživatele, který je v současné době nepřipojen, je zpráva uložena až do doby, než se uživatel znovu přihlásí.

Komunikace je standardně nešifrovaná, ale pro zabezpečení lze využít vrstvu mezi samotným XMPP a transportní vrstvou pomocí protokolů SSL a TLS.

```
<message
  to='petr@domena1.cz'
  from='jiri@domena2.cz'
  type='chat'
  xml:lang='cz'>
        <body>Ukazkova zprava</body>
</message>
```

Zdrojový kód 2.1: Ukázka XMPP zprávy.

2.3 Šifrování

Potřeba zabezpečovat komunikaci provází lidstvo již od počátků věků. Ať už se jednalo o vojenské zprávy, které nesměly být vyzrazeny protivníkovi, nebo vzkazy, které si posílali milenci. Vždy bylo nutné, aby obsah přenášený mezi dvěma stranami byl čitelný jenom pro ně, a nikoliv pro kohokoliv mezi nimi.

Oboru zabývajícím se utajováním obsahu zpráv se říká kryptografie [3], slovo pochází z řečtiny, kde kryptós znamená skrytý a gráphein má význam psát. Tento obor si prošel svojí evolucí od jednoduchých šifer jako je Caesarova šifra, která funguje na principu posunutí písmen v abecedě o pevný počet míst dále, až po dnešní algoritmy používané v počítačích.

V moderní kryptografii se pro převod čitelné zprávy neboli prostého textu na zprávu nečitelnou neboli šifrovanou používají sofistikované algoritmy. Tyto algoritmy se v základu dělí na symetrické a asymetrické, a to podle použití klíčů pro šifrování, respektive pro dešifrování zprávy.

2.3.1 Symetrické šifry

Pro šifrovaní pomocí symetrických šifer se používá jediný klíč, který čitelnou zprávu převede do nečitelné podoby (zašifruje) a poté pomocí stejného klíče je zpráva zase dešifrovatelná do její původní čitelné verze. Z tohoto principu vyplývá nutnost držet použitý klíč v tajnosti, protože kdokoliv, kdo má příslušný klíč může zprávy jak šifrovat, tak dešifrovat.

Pro přenos klíčů mezi jednotlivými stranami komunikace se často používají asymetrické šifrovací algoritmy. Výhodou transformace zprávy do nečitelné podoby pomocí symetrických šifer oproti asymetrickým je jejich poměrně nízká výpočetní náročnost (v porovnání s asymetrickými algoritmy), která je v dnešní době také podpořená hardwarovou akcelerací nejpoužívanějších algoritmů v procesorech.

Symetrické šifry se dále dělí na proudové a blokové. Proudové provádějí transformaci původní zprávy nejčastěji po znacích, jež kombinují s pseudonáhodně generovaným klíčem (keystream) typicky za pomocí operace XOR. Keystream je generován ze základního klíče pomocí algoritmu dané šifry v průběhu procesu transformace vstupní zprávy. Naproti tomu blokové šifrovací algoritmy zpracovávají data po blocích pevné délky, poslední blok se dorovnává na požadovanou délku (padding). Samotné šifrování pak probíhá po těchto blocích dat pomocí klíče. Slabinou tohoto postupu je samotná aplikace stejné transformace (stejný klíč) na každý blok, proto se do šifrování vkládá ještě další vstup generovaný ze vstupního vektoru [8].

2.3.2 Asymetrické šifry

Také známé jako šifry s veřejným klíčem. Na rozdíl od symetrických šifer je u asymetrických používána dvojice různých klíčů. První z těchto klíčů je označován jako veřejný a slouží pro

zašifrování zprávy, je doručován protistraně, která má posílat zabezpečená data. Druhý klíč se nazývá privátní a slouží pro dešifrování zprávy, tento klíč je nutné držet na bezpečném místě, protože bez něj není možno za běžných podmínek rozluštit zprávu šifrovanou klíčem veřejným.

Algoritmy jsou postavené na tzv. jednocestných funkcích, kdy lze snadno vypočítat výstup, ale zpětný výpočet vstupu z výstupu již prakticky není možné provést. Bezpečnost šifrovacích algoritmů se proto stanovuje jako čas potřebný pro uhodnutí privátního klíče, kdy se tomuto útoku říká útok hrubou silou (brute force attack).

Asymetrické šifry lze kromě použití pro zamezení přečtení původního obsahu přenášené zprávy použít i pro elektronické podepisování zpráv, které se děje následujícím způsobem. Uživatel vypočítá kontrolní součet zprávy, ten pomocí svého privátního klíče zašifruje a společně se zprávou zašle příjemci, který zná veřejný klíč (ten může být dostupný všem). Příjemce znovu spočítá kontrolní součet zprávy a porovná jej s dešifrovaným kontrolním součtem zaslaným společně se zprávou. Pokud se shodují, je zjištěno, že autorem právy je právě i původce veřejného klíče [8].

RSA

Jedná se o jeden z nejpoužívanějších algoritmů pro asymetrické šifrování, krom samotného šifrování se používá i pro účely podepisování. Jeho původ je datová do roku 1977, kdy jej poprvé popsali pánové Ron Rivest, Adi Shamir, a Leonard Adleman, zkratka algoritmu je pak složeninou jejich příjmení. Princip je založen na předpokladu vysoké obtížnosti rozdělení velkého čísla na součin prvočísel. Samotná prvočísla se získávají generováním náhodných velkých čísel, která se následně testují (například pomocí Miller-Rabinova testu), jestli nejsou prvočísly [8].

Kapitola 3

Analýza existujících řešení pro bezpečnou komunikaci na mobilních zařízeních

V této kapitole bych se rád věnoval již existujícím aplikacím, které umožňují zasílání zpráv mezi jednotlivými uživateli a komunikaci zabezpečují šifrováním. Analýzu aplikací provádím jak z hlediska jejich uživatelského rozhraní, které se téměř vždy opírá o stejnou myšlenku, tak i z hlediska použitých technik pro zabezpečení komunikace. Aplikace jsem se snažil používat co nejdéle, tak abych pochopil potřeby, které uživatelé mají.

Za všemi mnou testovanými aplikacemi stojí týmy vývojářů a designerů uživatelských rozhraní, kteří jejich vývojem strávili stovky hodin. Během let, co jejich produkt používají desítky až stovky tisíc uživatelů, získali velké množství zpětných vazeb, na jejichž základě prováděli úpravy rozhraní a funkcionality. V poslední době, kdy se na povrch dostaly informace o možnostech a využívání sledovacích a monitorovacích prostředků vládních i nevládních organizací, se začalo více hledět na bezpečnost komunikace, která prochází přes internet. Proto i dále uváděné aplikace začaly přecházet na koncept, který víc než kdy dříve klade vyšší důraz na zabezpečení samotných zpráv uživatelů.

Naneštěstí v době psaní této práce byla odstavena jedna z aplikací, kterou jsem se zabýval a proto se o ní dále už nebudu zmiňovat. Tou aplikací je Facebook Messenger, pomocí které šlo zasílat zprávy až do 29. března 2017. Od tohoto data zůstává funkční jenom verze pro nejnovější operační systém Windows 10 Mobile. A to i přes fakt, že poslední verze tohoto systému na mobilních telefonech není zdaleka nejpoužívanější, viz obrázek 2.1.

3.1 Skype

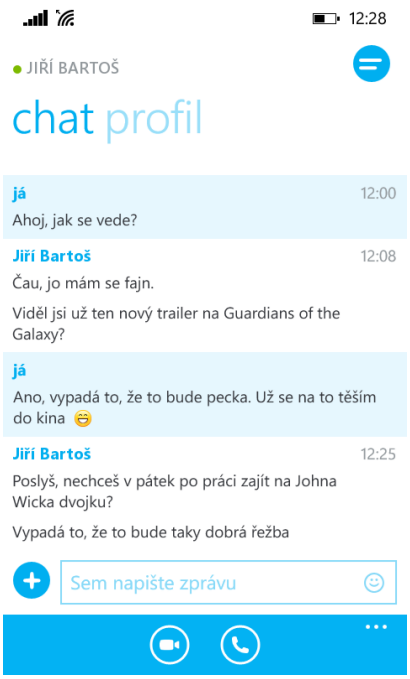
Jako prvního zástupce jsem si zvolil aplikaci Skype, která od května 2011 patří společnosti Microsoft. I přes tuto skutečnost se nejedná o vyladěnou aplikaci a její běh je na zvyklosti uživatelů systému Windows Phone pomalý. Zdlouhavé je samotné spouštění, ale také reakce prostředí na uživatelskou interakci jako je přechod mezi hlavní obrazovkou a zobrazením konverzace. Při testování jsem se občas setkal i s pády aplikace. Dalším negativem je i ukončení podpory k říjnu 2016 pro systémy Windows Phone. V průběhu tohoto roku by měla aplikace přestat fungovat úplně, protože postupně dochází ke změnám v provozu služby [11].

Při prvním spuštění je uživateli nabídnuto, aby pro přihlášení použil učet spárovaný se svým telefonem; to je možné právě díky tomu, že Skype patří společnosti Microsoft a ta jej integrovala do svých služeb. Tento krok pro přihlášení je víc než pohodlný, stačí pouze zmáčknout tlačítko „Přihlásit se“ a není nutné zadávat žádné údaje. Pokud chce ovšem uživatel využít jiný účet, tato možnost je mu také k dispozici.

Prostředí aplikace se sestává z hlavní obrazovky, která se dělí na tři karty. První karta zobrazuje nedávné konverzace seřazené podle času poslední komunikace. Další karta zobrazuje seznam všech kontaktů a poslední karta obsahuje dlaždice profilových obrázků oblíbených uživatelů. Na hlavní obrazovce jsou i dvě tlačítka, jedno pro zahájení hovoru a druhé pro zahájení chatu. Druhé jmenované tlačítko mi ovšem nedává velký smysl, protože vede na výběr uživatele, se kterým se má zahájit chat. Stejnou akci přitom vyvolá i kliknutí přímo na danou osobu na kartě kontaktů.

Poměrně zajímavě působí další obrazovka, která zobrazuje položky komunikace mezi dvěma lidmi. Oproti ostatním aplikacím se nesnaží využívat „bubliny“ pro zobrazení jednotlivých zpráv, ale pro jednotlivé položky využívá celou šířku obrazovky. Veškerý text je pak zarovnán doleva. K odlišení příchozích a odchozích položek konverzace poté slouží pouze podbarvení jednotlivých řádků, tak jak je vidět na obrázku 3.1. Na poměrně kladný dojem z této obrazovky působí špatně nic neříkající tlačítko v pravém horním rohu, které vede na úvodní obrazovku. Občas se mi stalo, že jsem toto tlačítko nechtěně stiskl a divil se, proč jsem se rázem ocitl na úvodní obrazovce aplikace.

Podle stránek technické podpory aplikace Skype využívá pro přenos zpráv mezi klientem a serverem protokol TLS. Pokud jsou ovšem zprávy přenášené pouze mezi klienty bez účasti serveru, je pro jejich zabezpečení použit algoritmus AES s 256 bitovým klíčem [10].



Obrázek 3.1: Obrazovka aplikace Skype, sloužící pro zobrazení zpráv mezi uživateli.

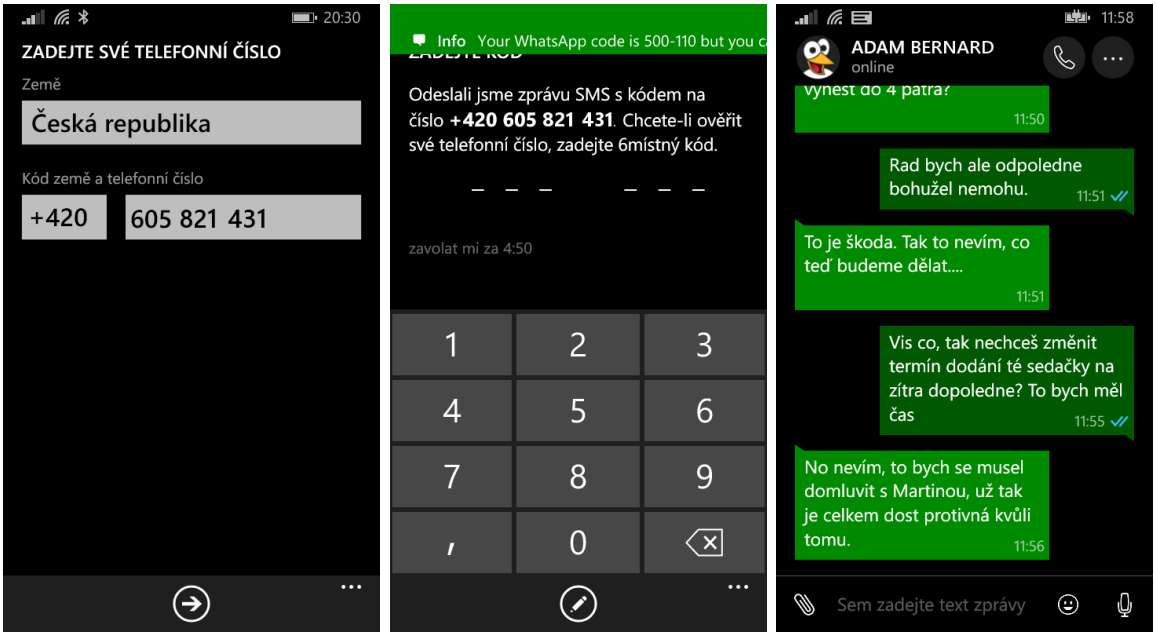
3.2 WhatsApp

Na první pohled velice sympatická aplikace, která svým jednoduchým a čistým rozhraním zapadá do prostředí systému. Je zde ovšem vidět spíše inspirace vzhledem novějších Windows 10. Jako identifikátor uživatele slouží jeho telefonní číslo. Na toto číslo a předvolbu státu se aplikace dotáže při prvním spuštění. Po jeho zadání a potvrzení dojde k zaslání ověřovacího kódu pomocí SMS zprávy. Tento kód uživatel zadá do aplikace a ta jej následně přihlásí. Zadání telefonního čísla a posléze ověřovacího kódu je poměrně elegantní a pohodlné řešení oproti nutnosti registrace a zadávání uživatelského jména a hesla.

Hlavní obrazovka se dělí na karty „Chaty“, „Hovory“ a „Stav“. První dvě karty jsou samovypovídající, jedná se o seznamy konverzací a uskutečněných hovorů. Stav obsahuje seznam příspěvků, a to jak vlastních, tak těch od kontaktů. Jedná se v podstatě o podobnou koncepci, jako je příspěvek na sociálních sítích, ovšem s jednou zásadní odlišností: doba životnosti tohoto příspěvku je omezena na 24 hodin. Po jednom dni od publikování se příspěvek smaže, a pokud jej nikdo nezkopíroval, už se k němu nikdo nedostane. Obsahem příspěvku může být obrázek nebo video s případným popiskem.

Při používání aplikace jsem zaznamenal její občasné pády vždy po uzamčení obrazovky, což na mě nepůsobilo dobře, protože ji bylo nutné opakovaně spouštět. Mezi další negativa bych zařadil poměrně úzké vstupní pole pro zadávání textu na obrazovce s konverzací, které je z obou stran obklopeno tlačítky pro přidání položek chatu jako jsou emotikony a přílohy.

Zabezpečení komunikace v rámci aplikace WhatsApp je řešeno jako point-to-point už od 31. března 2016. Znamená to, že zprávy a hovory jsou mezi zařízeními přenášená šifrovaně bez možnosti je po cestě získat v čitelné podobě. K obsahu komunikace nemá přístup ani sama společnost WhatsApp, která provozuje tuto aplikaci. Při registraci se vytvoří několik dvojic veřejného a privátního klíče pomocí algoritmu Curve25519. Jedna z dvojic slouží pro ověřování identity uživatele a další pro ustavení komunikace mezi zařízeními dvou uživatelů. Veřejné části klíče jsou ukládány na servery WhatsAppu. Po ustavení komunikace se poté používá symetrické šifrování každé zprávy pomocí algoritmu AES s 256 bitový klíčem. Klíč je poté pro každou zprávu unikátní. Více podrobností o ustavení komunikace mezi uživateli a celkového procesu, který je pro zabezpečení použit lze nalézt v dokumentu věnovaném tomuto tématu na stránkách společnosti [16].



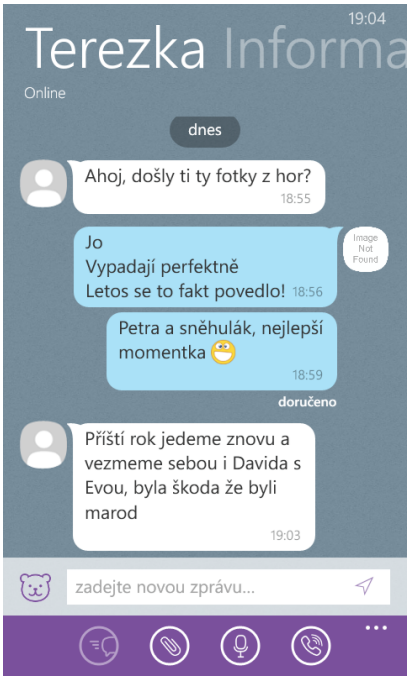
Obrázek 3.2: Ukázka registrace účtu a prostředí chatu aplikace WhatsApp.

3.3 Viber

Viber je dalším zástupcem bezplatných komunikačních aplikací a podobně jako WhatsApp i zde se používá telefonní číslo jako identifikátor uživatele a pro ověření registrace SMS zpráva s číselným kódem. Ihned po registraci jsou do aplikace načteny kontakty uživatele, které má v telefonním seznamu. Dojde k ověření, jestli kontakt existuje ve službě Viber, pokud existuje, zobrazí se v kartě kontaktů v sekci „Viber“ na hlavní stránce aplikace. Protože je Viber aplikace hodně zaměřená na internetovou telefonii, tak na kartě kontaktů je i sekce „Vše“, kde jsou zobrazeny veškeré uživatelovy kontakty, které mají telefonní číslo. Na tyto kontakty pak lze telefonovat i přes internet pomocí už zpoplatněné funkce Viber Out. Na hlavní obrazovce jsou poté ještě dvě karty, a to pro seznam všech konverzací a seznam uskutečněných hovorů.

Design aplikace plně nectí jednoduchost systému Windows Phone a ve výchozím režimu se zobrazuje na pozadí poměrně jasný obrázek, který společně s bílým písmem způsobuje horší čitelnost textu na hlavní obrazovce. Pozadí aplikace se dá naštěstí v nastavení změnit na tmavší a s bílým písmem kontrastnější obrázek. Další drobnost, která není v souladu s prostředím systému jsou pro Viber typické zakulacené profilové obrázky uživatelů. Zakulacení se také projevuje v konverzačních bublinách na stránce se zprávami. Až na tyto drobnosti, které ovšem nejsou na škodu a do celkového designu aplikace zapadají, se jedná o velice povedený produkt pro zasílání rychlých zpráv a internetovou telefonii.

Co se týká zabezpečení, tak je realizováno podobně jako u služby WhatsApp, s tím rozdílem, že pro šifrování zpráv se používá symetrický algoritmus Salsa20 a 128 bitový klíč. Naneštěstí u Viberu je point-to-point šifrování dostupné až od šesté verze komunikátoru, ten však již není pro telefony s operačním systémem Windows Phone aktualizován. Nejnovější verze se dočkala pouze zařízení s operačním systémem Windows 10 [14]. Více informací o použitých technikách zabezpečení se lze dočíst na stránkách služby Viber [15].



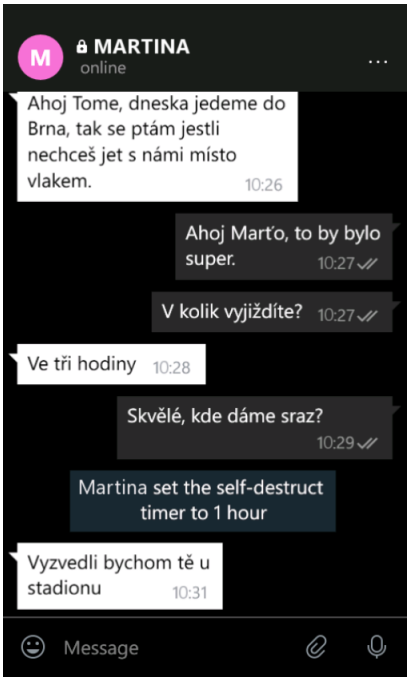
Obrázek 3.3: Ukázka rozhraní chatu v aplikaci Viber.

3.4 Telegram

Jako posledního zástupce bezpečných komunikačních řešení pro platformu Windows Phone jsem si zvolil aplikaci Telegram. Ta oproti aplikaci WhatsApp plně vychází z designového jazyka typického pro operační systém Windows 10 a proto do celkového pojetí staršího systému graficky nezapadá. Podobně jako předchozí zmíněné aplikace, vyjma aplikace Skype, i Telegram využívá pro identifikaci uživatele jeho telefonní číslo, a i registrace je téměř totožná.

Hlavní obrazovka aplikace se dělí na dvě karty a to „chats“ a „contacts“. Na kartě kontaktů se zobrazují všechny položky uživatelova telefonního adresáře. Pokud je kontakt ve službě Telegram již registrovaný, zobrazuje se nahoře. Pokud registrován není a uživatel na něj klikne, otevře se okno pro odeslání SMS zprávy s odkazem na stažení služby. Na kartě pro seznam konverzací (karta „chats“) se logicky zobrazují existující konverzace.

Zajímavostí je pojetí bezpečnosti zasílání zpráv v této službě, konverzace jsou totiž dvojího typu. První, běžný typ, nevyužívá point-to-point šifrování, ale spoléhá se na speciální protokol MTProto, který přenáší komunikaci šifrovaně mezi zařízením a serverem. Druhý typ se nazývá Secret Chat a ten už využívá jak protokol MTProto, tak i point-to-point šifrování pomocí algoritmu AES s 256 bitovým klíčem. U první varianty jsou zprávy uchovávány na serverech společnosti a přístup k nim je poté možný i po přihlášení na dalším zařízení. Naproti tomu Secret Chat uchovává zprávy pouze na zařízení a pro jeho ustavení je potřeba, aby byly oba uživatele přihlášeni. Další zajímavostí je možnost nastavení časovače pro „autodestrukci“ jednotlivých odeslaných zpráv, a to od jedné sekundy až po týden. Po uplynutí této doby zpráva zmizí na obou stranách. Funkce pro sebezničující zprávy je dostupná pouze pro Secret Chat. Více informací ohledně zabezpečení a protokolu MTProto se lze dočíst na stránkách aplikace [12].



Obrázek 3.4: Ukázka rozhraní chatu v aplikaci Telegram s nastavenou dobou pro sebezničení zprávy druhou stranou.

Kapitola 4

Specifikace požadavků a návrh řešení

Poté, co jsem se seznámil s již existujícími řešeními pro bezpečnou komunikaci na platformě Windows Phone, jsem došel do bodu, kdy bylo nutné na základě získaných poznatků sestavit požadavky na výsledné řešení mnou vyvíjené aplikace. Klíčové body, které musí výsledné řešení splňovat jsem diskutoval s budoucími uživateli, kteří se následně také podíleli na průběžném testování aplikace. Kromě konečných požadavků jsem s uživateli také diskutoval samotný vzhled budoucí aplikace.

4.1 Stanovení požadavků

Z výsledků diskuse s uživateli na požadavky budoucí aplikace vyplynuly body nejen k samotné funkčnosti, ale také ke vzhledu aplikace. Byly zde i myšlenky, které se kriticky dotýkaly již existujících řešení. Uživatelé často zmiňovali nespokojenost jak s rychlostí aplikací (například Skype), tak se zbytečnými funkcemi, které nevyužívají (například sdílení stavu v aplikaci WhatsApp).

Následující body shrnují hlavní požadavky na výsledné řešení:

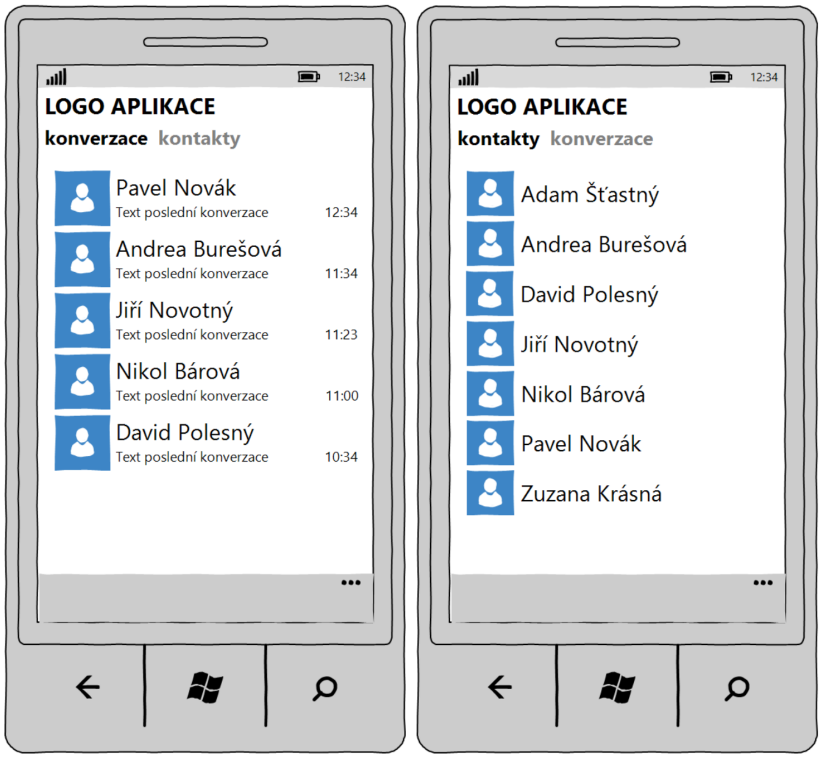
- Zabezpečení komunikace mezi uživateli šifrováním zpráv point-to-point
- Registrace uživatele pomocí telefonního čísla
- Automatické načtení a vyhledání kontaktů ve službě
- Jednoduché rozhraní a ovládání
- Design zapadající do systému Windows Phone
- Neplýtvání místem pro velké nadpisy
- Absence zbytečných funkcí nesouvisejících s posíláním zpráv

4.2 Návrh rozhraní aplikace

První návrhy rozhrání aplikace jsem vytvářel na papír pomocí tužky a gumy. Snažil jsem se vymyslet novou a neotřelou podobu prostředí, ale po diskuzi s budoucími uživateli jsem

od této myšlenky upustil. Hlavní důraz byl totiž kladen na celkovou jednoduchost, a proto jsem se nechal inspirovat již existujícími aplikacemi. Zvolil jsem tudíž jednoduchý koncept, kdy se aplikace sestává ze dvou hlavních obrazovek. První obrazovka má zobrazovat seznam existujících konverzací a také seznam uživatelů, druhá obrazovka pak samotnou konverzaci mezi uživateli v podobě po sobě jdoucím toku zpráv. Tento návrh se už u uživatelů setkal s pozitivním ohlasem, proto jsem mohl převést myšlenku z papíru do grafické podoby na počítači a ladit detaily. K tomu jsem použil program WireframeSketcher[1], ve kterém lze navrhnout umístění grafických prvků uživatelského rozhraní bez nutnosti jej programovat. Výhodou je, že program obsahuje prvky systému Windows Phone, které lze podle potřeby umisťovat a upravovat.

Jako první jsem navrhoval hlavní obrazovku, která obsahuje seznam konverzací a kontaktů. V horní části obrazovky je ponecháno místo pro logo aplikace a ve spodní části je umístěn aplikační panel, který by měl obsahovat méně používané položky (například odkaz na nastavení aplikace). Zbytek prostoru vyplňují seznamy kontaktů a konverzací. Pro ně jsem zvolil prvek, který seznamy oddělí do pojmenovaných karet, mezi kterými lze přepínat. Samotný prvek seznamu konverzací by poté kromě pojmenování měl zobrazovat i čas a úryvek poslední zprávy. Kliknutím na položku jednoho ze seznamů by pak mělo vést k přechodu na druhou obrazovku.



Obrázek 4.1: Výsledný prototyp hlavní obrazovky rozdělený na dva obrázky pro názornou ukázku obsahu jednotlivých karet.

Druhá obrazovka zobrazuje zprávy posílané mezi uživateli. V horní části stránky je prvek nesoucí název konverzace a dole se nachází aplikační panel s tlačítkem pro odeslání zprávy. Nad aplikačním panelem se nachází vstupní pole pro zadávání textu. Zbytek obra-

[1] http://wireframesketcher.com/

zovky vyplňuje tok zpráv, které se nacházejí v bublinách. Jednotlivé bubliny poté obsahují samotný text zprávy a čas, kdy byly odeslány. Pro odlišení jednotlivých dnů, ve kterých se konverzace udála, je použit textový prvek nesoucí příslušné označení data.

Obrázek 4.2: Výsledný prototyp obrazovky pro zobrazení konverzace mezi uživateli.

Kapitola 5

Implementace a testování

V této kapitole se zabývám vývojem cílového řešení, a to přes serverovou část až po samotnou implementaci aplikace. K řešení jsem přistoupil pomocí agilní metodiky iterativního vývoje. Pomocí této metodiky jsem průběžně vytvářel prototypy aplikace, které jsem testoval a diskutoval společně s uživateli. Při každé iteraci jsem zpracovával připomínky uživatelů z předchozího kroku a zároveň jsem implementoval další funkcionalitu.

5.1 Volba serverové části

Protože cíl této práce je bezpečná komunikační aplikace, která přenáší zprávy přes síť internet, bylo potřeba si na začátku vybrat potřebnou serverovou část. Jako jedna z možností se nabízela implementace vlastního programu, jež by zajišťoval jak registraci a autentizaci uživatelů, tak i samotné doručování zpráv mezi jednotlivými klienty. Tato možnost v sobě ovšem skrývala úskalí v podobě implementačních chyb, kterých bych se mohl dopustit. Protože serverová část je ze sítě internet přímo dostupná, každá možná chyba by nesla možné fatální následky pro bezpečnost mého řešení. Druhá varianta v podobě již existujícího řešení se proto zdála jako lepší cesta.

Při konzultaci s vedoucím této práce jsem dostal tip na využití protokolu XMPP, který se používá právě v aplikacích pro přenos zpráv mezi uživateli a serverem. Protože se jedná o již dlouhou dobu používaný protokol, který je dokonce i standardizován, rozhodl jsem se prověřit možnosti, které by mi při tvorbě aplikace mohl poskytnout. Vlastnosti XMPP mě nakonec zaujaly natolik, že jsem se rozhodl jej využít ve své práci jako hlavní komunikační protokol. Důležitým pozitivem bylo také velké množství implementací serverových aplikací založených právě na XMPP.

5.1.1 Ejabberd

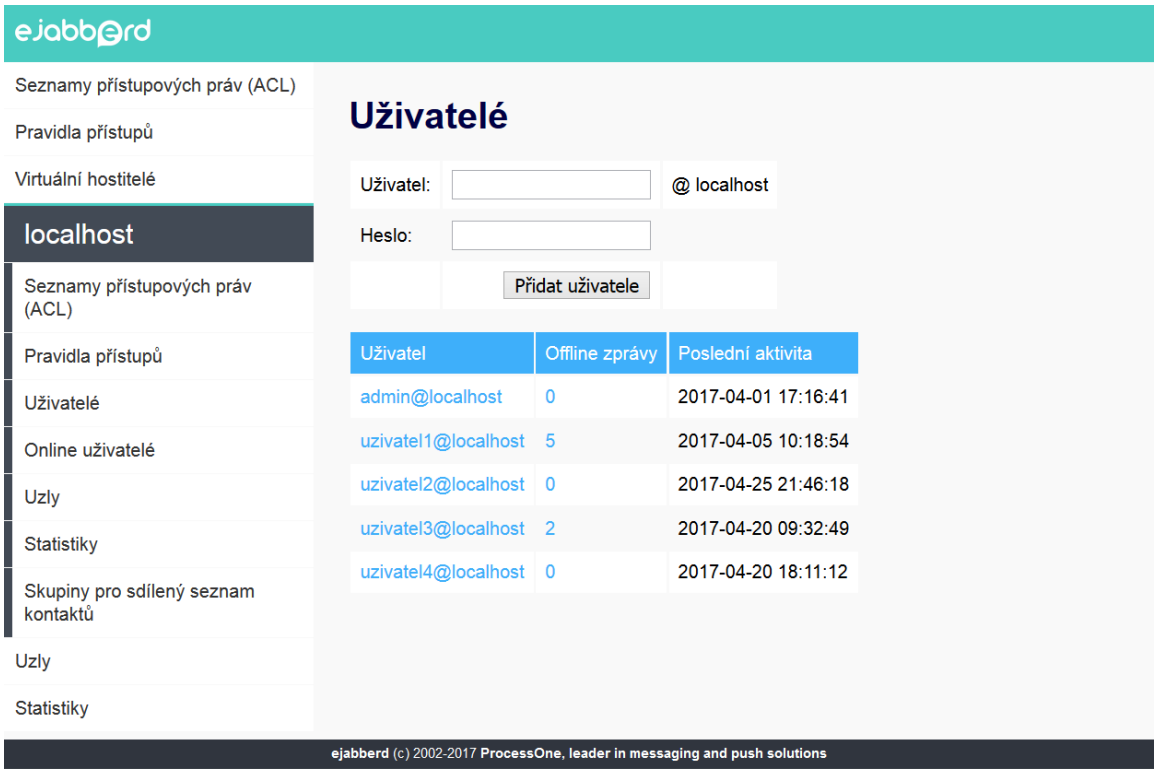
Z velkého množství dostupných XMPP serverů jsem se nakonec rozhodl využít právě Ejabberd. Mezi jeho hlavní pozitiva patří rozsáhlá možnost konfigurace a z toho plynoucí přizpůsobitelnost pro velký rozsah aplikací. V průběhu vývoje se mi také osvědčila užitečná dokumentace, která je v mnoha případech doplněna i srozumitelnými návody.

Samotný Ejabberd je open source projekt založen na jazyce Erlang. Je stavěný na myšlence vysoké modularity, kdy si administrátor může při konfiguraci zvolit běh jenom s moduly, které opravdu potřebuje. Pokud nastane situace, kdy je při provozu zjištěna potřeba zavedení dalšího modulu, není nutné server zastavovat a po změně konfigurace znovu

spouštět, stačí pouze využít konzoli serveru a provést potřebné akce pomocí příkazů za běžného provozu.

Konzole zpřístupňuje příkazy, respektive procedury jazyka Erlang, díky čemuž lze nejen měnit chování serveru, ale také například přidávat nové uživatele, odesílat zprávy nebo zjišťovat statistiky provozu. Prostředí konzole lze také zpřístupnit i pro použití v jiných programech či skriptech pomocí vzdáleného volání procedur. Toto chování zpřístupňuje zavedení modulu `ejabberd_xmlrpc`. Zde je ovšem nutné si dávat pozor, protože modul neumožňuje nastavení restrikcí přístupu ze sítě internet. Je proto nutné ručně zablokovat příslušný port pro vnější přístup a ponechat možnost komunikace pouze na lokální úrovni. Pokud bychom toto neprovedli, museli bychom se spolehnout pouze na nutnost autorizace přístupu k procedurám pomocí administrátorského XMPP účtu.

Kromě modulu pro vzdálené volání procedur existuje ještě možnost využití vestavěné webové administrační služby, která se zpřístupňuje pomocí modulu `ejabberd_http`. Jedná se o webové rozhraní, které umožňuje interaktivní formou ovládání Ejabberd serveru. Toto rozhraní, které lze vidět na obrázku 5.1, zobrazuje v přehledných kategoriích běžící uzly, registrované uživatele a statistiky. Je zde také možné provádět některé konfigurační úkony jako například správu přístupových práv. Do této služby se lze přihlašovat pomocí příslušného administrátorského XMPP účtu, který je nutno nejprve vytvořit v samotné správcovské konzoli a následně mu v konfiguraci přiřadit příslušná oprávnění. Konfigurace oprávnění pro webovou administrační službu a pro vzdálené volání procedur lze vidět v ukázce zdrojového kódu 5.1.



Obrázek 5.1: Ukázka webového administračního rozhraní serveru Ejabberd.

Pro ukládání informací o registrovaných uživatelích a pro dočasné uložiště nedoručených zpráv nabízí Ejabberd mnoho možností. Ve výchozím nastavení je pro tyto účely používaná

interní databáze Mnesia, kterou lze ovšem bez problému nahradit relační databází nebo LDAP serverem. Další velkou výhodou je, že jednotlivá uložiště lze kombinovat a tím využít například již existující databází uživatelů ve formě LDAP serveru. Pro účely mého řešení jsem došel k závěru, že nejlepší možností bude veškerá data ukládat do relační databáze MySQL, která se nachází na stejném serveru jako Ejabberd. Tato varianta mi totiž umožní jednoduchý přístup ke všem registrovaným uživatelům vyvíjené služby.

```
1 acl:
2   admin:
3     user:
4       - "adminuser": "localhost"
5 access:
6   configure:
7     admin: allow
8   xmlrpcaccess:
9     admin: allow
```

Zdrojový kód 5.1: Ukázka konfigurace přístupových práv pro administrátorský účet Ejabberd serveru.

Pro zabezpečení komunikace mezi serverem a klientem lze využít protokolu TLS. Pro jeho zprovoznění je potřeba splnění dvou podmínek. První z nich je ta, aby šifrovanou komunikaci pomocí protokolu TLS podporovala klientská část. Druhá podmínka je vlastnictví potřebného certifikátu, který obsahuje privátní a veřejný klíč. Certifikát lze vygenerovat pomocí nástroje OpenSSL a následně jej nechat podepsat důvěryhodnou certifikační autoritou. Druhou možností je, že jej podepíše sám jeho tvůrce, tomuto certifikátu se pak anglicky říká Self signed certificate [17].

5.2 Knihovna pro XMPP

Poté co jsem vybral serverové řešení, bylo nutné přejít k otázce obsluhy XMPP na klientské straně. Podobně jako u předchozího kroku jsem se zabýval myšlenkou vlastní implementace příslušené knihovny. Po zběžné prohlídce základního standardu XMPP jsem zjistil, že jediným schůdným řešením pro tuto práci bude použití již existující knihovny. Na stránkách organizace zastřešující samotný standard XMPP je dostupný seznam existujících knihoven, které jej implementují. Naneštěstí ani jedna z knihoven pro jazyk C#, nacházejících se na daných webových stránkách, nebyla pro použití v projektu vhodná. Většina z nich byla totiž podmíněna nákupem potřebné licence, u zbylých knihoven docházelo k problémům plynoucích z omezeného API dostupného na zařízeních s operačním systémem Windows Phone.

5.2.1 XMPP/Media Library

Naštěstí jsem v internetové diskuzi narazil na zmínku o balíku knihoven určených pro platformu Windows Phone 7, který mimo jiné obsahuje i implementaci XMPP včetně podpory protokolu TLS. Tento balík se jmenuje XMPP/Media Library for .NET and Windows Phone 7.5 a je publikovaný pod licencí MIT na stránkách hostující projekty s otevřeným kódem[1]. Jistou nevýhodou tohoto řešení je jeho nulová dokumentace, kterou do jisté míry supluje přítomnost zdrojových kódů.

[1] http://xmedianet.codeplex.com/

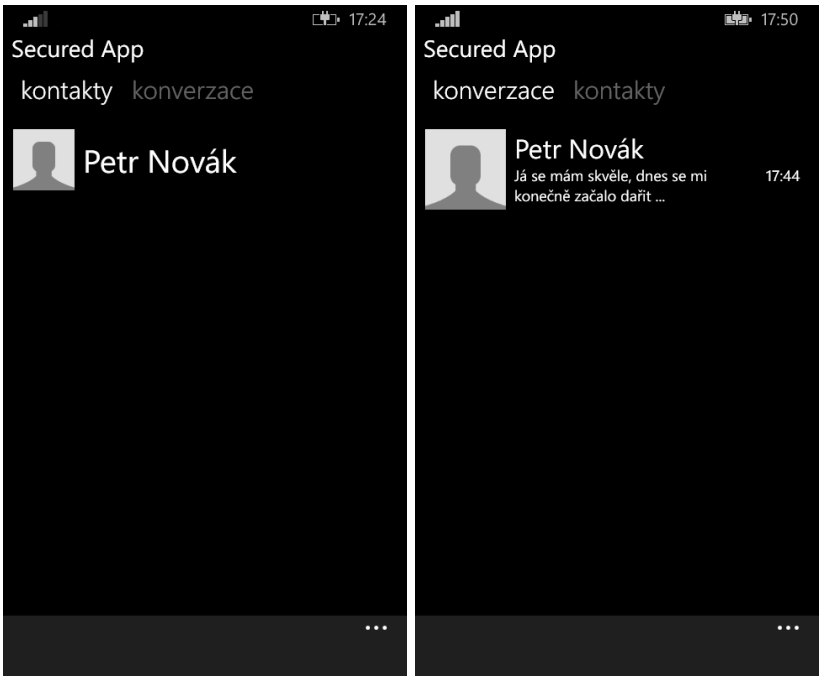
Pro zprovoznění klientské části XMPP s touto knihovnou je nutné vytvořit instanci třídy `XMPPClient` a nastavit jí příslušné vlastnosti v podobě adresy serveru a přihlašovacích údajů. Po následném zavolání metody `Connect`, která ustaví spojení se serverem je již možné odesílat zprávy metodou `SendChatMessage`. Tato metoda vyžaduje jako první parametr text samotné zprávy a jako druhý parametr instanci třídy `JID`. Tato instance reprezentuje identifikátor uživatele, kterému se zpráva posílá. Pro příjem nových zpráv je nutné reagovat na událost `OnNewConversationItem`. Tuto událost vyvolá krom přijetí zprávy i úspěšné odeslání zprávy metodou `SendChatMessage`.

5.3 Etapy vývoje

5.3.1 První etapa

Tato etapa se nesla ve znamení seznamování se s vývojovým prostředím Visual Studio 2015, jazykem C#, zvolenou knihovnou a tvorbou UI. Po pár pokusech jsem byl již schopen implementovat návrh podoby uživatelského rozhraní pomocí deklarativního značkovacího jazyka XAML. Pro tyto účely vznikly dva soubory, které označují jednotlivé stránky aplikace.

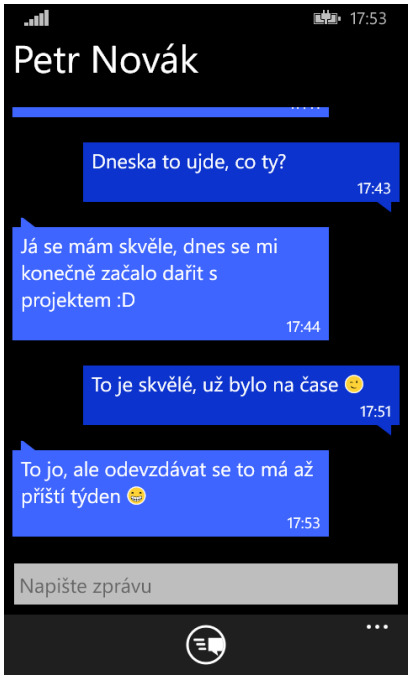
První z nich je soubor `MainPage.xaml`, který v sobě nese definici hlavní obrazovky. Podstatnou funkci zde hraje komponenta `Pivot`, která stránku rozděluje na dvě karty, respektive na dva prvky `PivotItem`. Toto rozdělení, jež je patrné na obrázku 5.2, je podstatné pro oddělení seznamu kontaktů a seznamu konverzací, mezi nimiž se dá lehce přepínat táhnutím prstu od jednoho okraje displeje směrem ke druhému. Po kliknutí na položku jednoho ze dvou seznamů dojde k přechodu na druhou stránku.



Obrázek 5.2: Hlavní stránka aplikace s využitím komponenty `Pivot`. Levý obrázek zobrazuje seznam kontaktů a pravý obrázek seznam konverzací.

Druhá stránka, na kterou v dalším textu budu odkazovat jako na stránku s chatem a jejíž definice se nachází v souboru `ChatPage.xaml`, se stará o zobrazení a odesílání konverzace

mezi dvěma uživateli, tak jak lze vidět na obrázku 5.3. Samotné zobrazení jednotlivých položek konverzace je zajištěno pomocí techniky Binding, která přebírá data z objektu typu `ObservableCollection<T>`, nacházejícího se v code-behind souboru této stránky. Data tohoto objektu jsou pak instance třídy `ChatItem`. Tato třída je potřebná pro rozlišení, která ze tří šablon se má použít pro vykreslení položky chatu. První z těchto šablon slouží pro zobrazení bubliny příchozí zprávy, druhá pak pro zobrazení odchozí zprávy. Poslední šablona slouží pro zobrazení oddělovače jednotlivých dnů, ve kterých se konverzace udála.



Obrázek 5.3: Obrazovka pro zobrazení konverzace s ukázkovými zprávami.

Již v této části byla využita knihovna pro obsluhu XMPP a zprávy bylo možné odesílat za pomocí tlačítka v aplikačním panelu na stránce s chatem. Odeslané a přijaté zprávy se zobrazovaly, ale ještě nedocházelo k jejich ukládání, neboť ještě nebyla vytvořena příslušná databáze. Pro testovací účely bylo nutné zadávat přihlašovací údaje k serveru přímo do zdrojového kódu před samotným překladem.

5.3.2 Druhá etapa

V této etapě jsem již začal vytvářet vnitřní strukturu aplikace společně s částmi pro ukládání dat ve formě databáze. Pro oddělení aplikační logiky od samotného datového modelu jsem využil návrhového vzoru MVVM popsaného v kapitole 2.1.3. Samotný model se nachází ve jmenném prostoru `SecuredComunicationAppSilverLight.Model` a je reprezentován datovým kontextem. Ten představuje jak strukturu samotné databáze, tak i most mezi daty v databázi a objekty v jazyce C#. Výsledná databáze je reprezentována třídou `ContactsAndConversationDataContext` a tvoří ji dvě tabulky.

První tabulka nese název `AllContacts`, představuje strukturu pro uložení jednotlivých kontaktů, které má uživatel k dispozici. Jejími položkami jsou:

- **Id** – pořadové číslo záznamu a také primární klíč této tabulky

- **Name** – zobrazované jméno kontaktu
- **JID** – identifikátor uživatele pro XMPP
- **Deleted** – příznak reprezentující, že daný uživatel již neexistuje na serveru
- **Tel** – telefonní číslo uživatele, na které je registrován
- **PublicKey** – veřejný klíč uživatele

Druhá tabulka jmenující se `AllConversations` slouží pro ukládání informací o založené konverzaci mezi dvěma uživateli. Jejími položkami jsou:

- **ConversId** – pořadové číslo konverzace a také primární klíč tabulky
- **LastConvers** – datum a čas poslední zprávy v konverzaci
- **NotSeenConvers** – počítadlo nepřečtených přijatých zpráv
- **ConversSnatch** – úryvek textu poslední zprávy (přijaté nebo odeslané), který se zobrazuje na hlavní stránce pod názvem konverzace
- **AesKey** – klíč pro symetrické šifrování zpráv algoritmem AES
- **AesIV** – inicializační vektor algoritmu AES
- **Contact** – reprezentuje cizí klíč na záznam z tabulky `AllContacts`

Pro použití databáze je nutné ji nejprve vytvořit, to se děje při prvním spuštění aplikace voláním metody `CreateDatabase`. Po vytvoření je možné již vkládat nová data a ta následně dotazováním pomocí technologie LINQ vybírat. LINQ, jenž je integrován do jazyka C#, poskytuje objektově-relační mapování pro práci s relační databází. Jeho zápis je ve formě lambda výrazů a podobá se jazyku SQL [5], tak jak je vidět na ukázce zdrojového kódu pro výběr všech konverzací z databáze a s jejich následným seřazením 5.2.

```
1 var conversations = (
2   from EstablishedConversation
3   in baseDB.AllConversations
4   select EstablishedConversation
5   ).OrderByDescending(x => x.LastConvers);
```

Zdrojový kód 5.2: Ukázka výběru dat z databáze pomocí technologie LINQ.

Další součástí návrhového vzoru MVVM je View-Model. Ten reprezentuje třída `ContactsAndConversationViewModel` nacházející se ve jmenném prostoru `SecuredComunicationAppSilverLight.ViewModel`. Instance této třídy vzniká již při spuštění aplikace v souboru `App.cs`, jež je vstupní branou při startu aplikace. Aby mohlo fungovat zobrazování kontaktů a konverzací na hlavní stránce aplikace, jež je zároveň komponentou View návrhového vzoru, je nutné při jejím vytváření přiřadit instanci právě View-Modelu do jejího datového kontextu. Tímto se zajistí předávání dat pomocí Bindingu z View-Modelu do View.

Pro přechod na stránku s chatem je nutné kliknutí na položku seznamu konverzace nebo kontaktů. Rozdíl mezi nimi je pouze v tom, že konverzace ještě nemusí existovat. V tom případě se na hlavní stránce zobrazuje pouze kontakt. Při následném kliknutí právě na něj,

dojde i k vytvoření nové položky konverzace. Při následném přechodu na stránku s chatem je předáván i parametr ve formě číselného identifikátoru dané konverzace. Tento parametr je po dokončení přechodu zpracován a na jeho základě je z objektu View-Modelu získána i instance dané konverzace, zároveň je také vynulováno počítadlo nepřečtených zpráv. Získaná instance slouží pro zobrazení jména uživatele, se kterým je konverzace otevřena a také pro účely zasílání zpráv. Zpracování získaného parametru je vidět v ukázce kódu 5.3.

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
  base.OnNavigatedTo(e);

  // osetreni neexistence potrebneho parametru
  if (!NavigationContext.QueryString.ContainsKey("convID"))
       NavigationService.Navigate(new Uri("/MainPage.xaml", UriKind.Relative));

  // zisk hodnoty parametru z uri adresy
  string convID = NavigationContext.QueryString["convID"];

  // ziskani instance konverzace s prislusnym ID
  currentConversationFromList = App.CACViewModel.GetConversationWithID(convID);

  // vynulovani pocitadla neprectenych zprav
  App.CACViewModel.ClearNotSeenConversation(currentConversationFromList);

       ...

}
```

Zdrojový kód 5.3: Zpracování přijatého parametru v souboru ChatPage.xaml.cs.

V code-behind souboru této stránky také dochází ke zpracování příchozích zpráv, pro tento účel slouží statická metoda `PrepareSaveConversation`. Jednotlivé zprávy se pak ukládají do separátní databáze, kde pro každou konverzaci existuje jeden soubor obsahující celé konverzační vlákno. Tento soubor obsahuje v názvu JID protistrany, a tak je jedinečně identifikován. Databáze pak obsahuje vždy pouze jednu tabulku, jejíž struktura je definovaná třídou `ConversationItem`. Položky této tabulky jsou následující:

- **ConversationItemId** – identifikátor jednotlivých zpráv sloužící zároveň jako primární klíč
- **ItemMessage** – samotné tělo zprávy (text)
- **ItemDate** – datum a čas odeslání zprávy
- **ItemTypeOfMessage** – výčtový ty určující typ zprávy (přijatá/odeslaná)

Na závěr této etapy proběhlo první uživatelské testování aplikace, do kterého se zapojili mojí nejbližší přátelé a rodina. Samotné testování probíhalo tak, že jsem musel nejdříve na serveru vytvořit potřebné uživatelské účty a následně jejich přihlašovací údaje zadat do zdrojového kódu aplikace. Dále bylo také nutné před překladem ručně vytvořit pro každého uživatele seznam kontaktů, jenž zahrnoval dvě až tři položky v databázi. Následně proběhlo nahrání aplikace na telefon. Celkem se testování účastnilo šest zařízení. Zde je dobré zmínit, že každé zařízení bylo nutné zaregistrovat jako vývojářské.

Z výsledků testování vyplynulo, že se samotným grafickým rozhraním byli uživatelé spokojeni a odpovídalo jejich představám, které získali při konzultacích samotného návrhu.

Připomínky měli spíše k funkcionalitě aplikace, což bylo v této fázi očekávatelné. Samotné připomínky, mezi nimiž nejsou ty, kterými jsem se hodlal zabývat bez ohledu na výsledek testování, jsou shrnuty v následujících bodech:

- Možnost odesílání zprávy pomocí stisku klávesy enter na klávesnici
- Zobrazení informace o příchozí zprávě, pokud se uživatel nachází v konverzaci s jiným uživatelem

5.3.3 Třetí etapa

V této etapě jsem se kromě připomínek uživatelů, plynoucích z prvního testování, věnoval i otázce registrace nových účtů na serveru. Jak vyplývá ze stanovených požadavků na cílové řešení této aplikace, registrace uživatelů má probíhat na základě jejich telefonních čísel. Naneštěstí API operačního systému nenabízí přístup k telefonnímu číslu, které využívá dané zařízení. Je proto nutné se při registraci na toto číslo dotázat uživatele. Nastává zde ovšem problém důvěry v neznámého uživatele, který může zadat telefonní číslo, které při nejlepším nemusí existovat, ale také může patřit někomu jinému. Bylo proto nutné vyřešit otázku verifikace vlastnictví daného čísla. Pro řešení tohoto problému jsem zvolil metodu ověření pomocí kódu zaslaného v SMS zprávě na telefonní číslo zadané při registraci. V následujícím textu bude popsána jak implementace v rámci aplikace, tak také potřebné řešení na straně serveru.

Na tomto místě je vhodné se zmínit o tvaru uživatelského jména, které je používané v rámci XMPP řešené aplikace. Toto jméno se skládá z předvolby země, samotného telefonního čísla uživatele, znaku podtržítka a šestimístného číselného kódu, který se inkrementuje při každé nové registraci na toto jedno telefonní číslo. Ukázka takového jména je vidět na obrázku 5.4.

Obrázek 5.4: Příklad podoby uživatelského jména v rámci aplikace.

Protože registrace prostřednictvím XMPP nenabízí potřebnou flexibilitu, rozhodl jsem se využít možnosti vzdáleného volání procedur, které nabízí server Ejabberd, a implementovat registraci a následně i další funkcionalitu pomocí jazyka PHP. Na straně serveru bylo proto potřeba nainstalovat webové rozhraní, a to následně nakonfigurovat tak, aby přenášená data byla šifrována a nemohlo dojít k jejich odposlechnutí a zneužití. Pro podporu registračního procesu bylo potřeba zajistit ukládání dat, k tomu byla využita databáze typu MySQL, která je také použita pro ukládání dat XMPP serveru. Do databáze byly přidány dvě nové tabulky, první z nich se nazývá `reg_code` a slouží pro ukládání informací souvisejících s ověřovacím kódem, tato tabulka obsahuje následující položky:

- **id** – položka primární klíče
- **tel** – telefonní číslo, na které je prováděná registrace
- **code** – vygenerovaný ověřovací kód
- **rem_attempts** – počet zbývajících pokusů pro zadání chybného ověřovacího kódu

- **date_created** – datum a čas vytvoření požadavku na registraci
- **used** – příznak použití ověřovacího kódu pro registaci

Další tabulka, jmenující se `used_username`, slouží pro úschovu dvojice telefonní číslo a sekvenční číslo. Tato dvojice je součástí každého uživatelského jména na serveru a v kontextu tabulky představuje již jednou použité jméno v rámci služby.

Registrace nového uživatele probíhá ve dvou krocích. V první kroku je na server odeslán dotaz s telefonním číslem. Skript na straně serveru vygeneruje náhodný šestimístný číselný ověřovací kód a zašle jej na SMS bránu společně s přijatým číslem, tím dojde k odeslání SMS zprávy s kódem vlastníkovi zadaného telefonního čísla. Samotný požadavek je zasílán na webové API SMS brány pomocí metody GET, jejíž data obsahují jak autorizační údaje pro účet na SMS bráně, tak i data pro samotnou zprávu. Dvojice ověřovací kód a telefonní číslo je po úspěšném odeslání zprávy uložena do tabulky `reg_code` a zpět do aplikace je vráceno `id` nového záznamu v této tabulce. V druhém kroku registrace se na server odesílá dotaz, jehož součástí je `id`, získané v předchozím kroku, a uživatelem zadaný ověřovací kód. Pokud se přijatý ověřovací kód na straně serveru shoduje s tím, který je uložený v databázi, dojde k registraci nového uživatele. Pokud se ovšem kódy neshodují, dojde k dekrementaci počítadla možných pokusů o zadání ověřovacího kódu a zpět do aplikace je vrácena příslušná chybová zpráva. Pokud je dosaženo maximálního možného počtu chybně zadaných ověřovacích kódů, dojde k zneplatnění předchozího kroku registrace a uživatel je o této skutečnosti informován. Po úspěšné registraci je ze serveru do aplikace vráceno nové vygenerované uživatelské jméno a heslo.

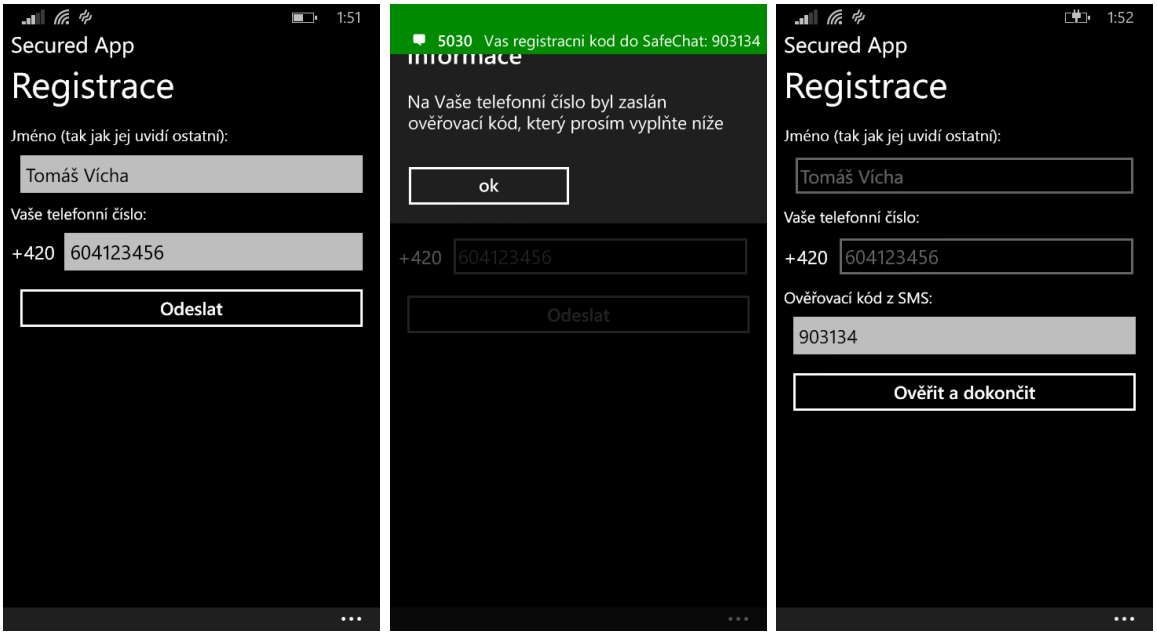
V rámci vyvíjené mobilní aplikace je registrační proces řešen na stránce `Register.xaml` a obsluha požadavků probíhá v code-behind této stránky. Pro účely fungování aplikace bylo v této etapě využito třídy `IsolatedStorageSettings`, která umožňuje permanentní uložení dvojice klíč-hodnota. Toto se hodí právě pro ukládání nastavení aplikace. Jedna z těchto položek nastavení je pak použita pro detekci proběhnutí registračního procesu. Ve výchozím stavu nabývá položka hodnoty false a tím dává najevo, že registrace ještě neproběhla. Na základě toho je při každém spouštění aplikace rozhodováno, jestli se má přejít rovnou na hlavní stránku nebo na stránku s registrací. Proces rozhodování o přechodu na příslušnou stránku je znázorněn ve formě zdrojového kódu 5.4.

```
1 private void Application_Launching(object sender, LaunchingEventArgs e)
2 {
3   Uri nUri;
4   if (Settings.IsRegistered)
5         nUri = new Uri("/MainPage.xaml", UriKind.Relative);
6   else
7         nUri = new Uri("/Register.xaml", UriKind.Relative);
8   RootFrame.Navigate(nUri);
9 }
```

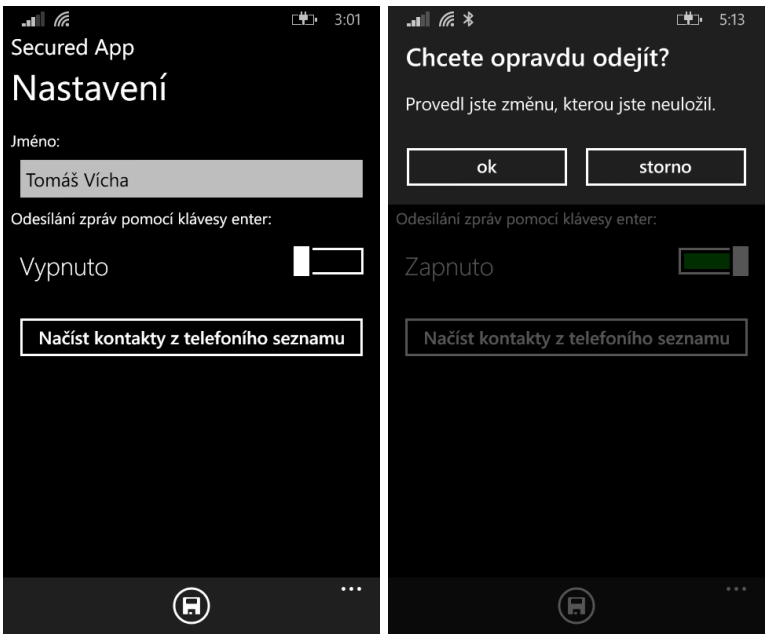
Zdrojový kód 5.4: Proces výběru první stránky po spuštění aplikace.

Proces registrace uživatele do služby je zachycen na obrázku 5.5. Kvůli omezením při využití SMS brány je možné provést registraci pouze na telefonní číslo s českou předvolbou, proto také není na stránce implementován prvek pro výběr země. Stránka samotná se poté skládá pouze ze vstupních prvků a tlačítek. Aplikační panel ve spodní části obrazovky obsahuje pouze odkaz na stránku O aplikaci. Po úspěšném dokončení registrace je uživatelské jméno a heslo uloženo do nastavení. Pro zajištění bezpečného uložení hesla je použita třída

`ProtectedData`, jež zprostředkovává Data Protection API operačního systému. Při ukládání je poté řetězec představující heslo zašifrován metodou `Protect`. Pro opětovné použití hesla je nutné jej převést zpět do čitelné podoby pomocí metody `Unprotect`.



Obrázek 5.5: Zachycení průběhu registrace v aplikaci od zadání jména a telefonního čísla uživatelem na prvním obrázku, přes obdržení SMS zprávy s ověřovacím kódem na druhém obrázku, až po jeho zadání na třetím obrázku.



Obrázek 5.6: Ukázka stránky s nastavením na prvním obrázku. Druhý obrázek znázorňuje situaci, kdy chce uživatel opustit stránku s nastavením po provedení změny, kterou neuložil.

V další části této etapy byla přidána stránka pro nastavení. Pro přístup na tuto stránku slouží odkaz umístěný v aplikačním panelu na hlavní stránce. Uživatel zde může změnit své jméno ve službě a také přibyl přepínač pro možnost odesílání zpráv stiskem klávesy enter, tak jak bylo od testovacích uživatelů vyžadováno. Samotný přepínač není součástí dostupných grafických prvků prostředí a bylo nutné pro jeho využití zahrnout do projektu i knihovnu Windows Phone Toolkit[2]. Pro aplikování změn je nutné, aby uživatel stiskl jediné tlačítko v aplikačním panelu. Provede-li uživatel změnu a bez uložení bude chtít stránku opustit, vyskočí příslušná hláška, informující jej o dané skutečnosti, tak jak lze vidět na obrázku 5.6. Uživatel má na této stránce také možnost ručně spustit skenování svého adresáře s kontakty pro zjištění, jestli některé z nich nevyužívají také stejnou službu. Toto prohledávání se také automaticky děje po úspěšné registraci. Když je při tomto procesu na serveru nalezen existující uživatel, je přidán do seznamu kontaktů na hlavní stránce aplikace.

Pro splnění další uživatelské připomínky z testování, a to konkrétně bodu o zobrazení notifikace o příchozí zprávě od jiného uživatele, než se kterým je zobrazená stránka s chatem, bylo potřeba do projektu začlenit další externí knihovnu. Systémové notifikace totiž nelze zobrazovat, když je daná aplikace v popředí. Pro tyto účely je nutné sáhnout po knihovně Toastinet[3]. Tato knihovna vytváří napodobeninu systémových notifikací, které lze vyvolat jednoduše přiřazením textového řetězce vlastnosti `Message` daného objektu třídy `Toastinet`. Aby bylo možné objekt využít, je potřeba nejprve v XAML souboru stránky nadefinovat příslušný element, například tak, jak je vidět v ukázce zdrojového kódu 5.5. Výsledná notifikace pak vypadá tak jako na obrázku 5.7. Po jejím stisknutí je vyvolána událost, díky které dojde k přechodu na stránku zobrazující konverzace s uživatelem, jehož příchozí zpráva notifikaci zobrazila. Notifikace jsem implementoval jak pro stránku s chatem, tak i pro hlavní stránku.

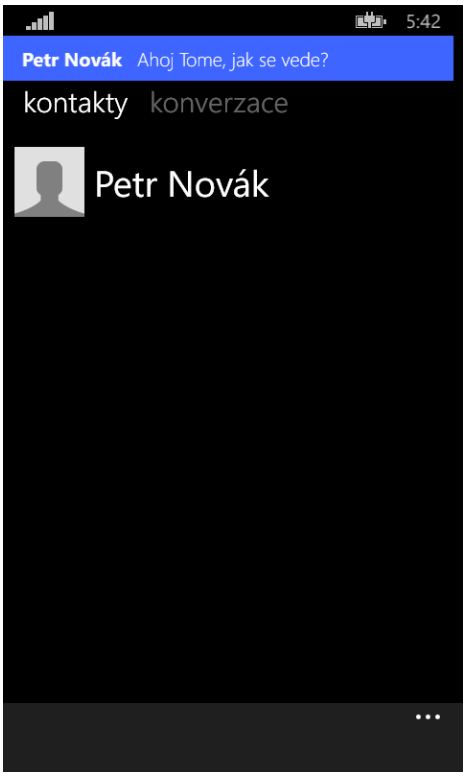
```
<toastinet:Toastinet x:Name="Toast"
                             Canvas.ZIndex="1"
                             VerticalAlignment="Top"
                             Foreground="{StaticResource␣PhoneForegroundBrush}"
                             Background="{StaticResource␣PhoneAccentBrush}"
                             AnimationType="LeftToRight"
                             Grid.Row="0"
                             Grid.RowSpan="2"
                             Grid.Column="0"
                             Grid.ColumnSpan="2"
                             Queued="True"
                             Tap="Toast_Tap"/>
```

Zdrojový kód 5.5: Ukázka možného nadefinování elementu toastinet, sloužícího pro zobrazení in-app notifikací.

Na konci této etapy proběhlo již druhé uživatelské testování. Oproti tomu předchozímu se mi podařilo rozšířit skupinu uživatel o jednoho člena a tím bylo dosaženo počtu sedmi zúčastněných přístrojů. Protože aplikace nebyla ještě vložena do obchodu Store, bylo nutné ji na každý telefon nahrát ručně. Tentokrát jsem již ale nemusel provádět nastavení ve zdrojovém kódu, ale každý uživatel se sám zaregistroval. Podle ohlasů byly splněny požadavky z předchozího testování, a i s uživatelským prostředím byli uživatelé spokojeni. V průběhu testování jsem byl dotázán na možnost přidání tlačítka pro vyhledání uživatele,

[2] https://phone.codeplex.com/
[3] http://toastinet.codeplex.com/

který není v kontaktech telefonu. Po krátké úvaze jsem se rozhodl, že dané tlačítko v další etapě implementuji.



Obrázek 5.7: Ukázka notifikace o nové zprávě na hlavní obrazovce.

5.3.4 Čtvrtá etapa

Čtvrtá a závěrečná etapa vývoje bezpečné komunikační aplikace musí bez pochyby dostát plnému významu slova bezpečná, a proto se následující část věnuje právě zabezpečení komunikace mezi uživateli. Kromě využití protokolu TLS, který používá XMPP knihovna pro komunikaci se serverem je nutné zajistit, že ani dočasně uložené zprávy na serveru nepůjde přečíst s pochopením jejich pravého významu. K tomuto jsem se rozhodl využít point-to-point šifrování samotného obsahu zpráv. Tímto lze docílit, že je obsah zpráv čitelný pouze zúčastněnými uživateli, kteří mají příslušný dešifrovací klíč.

K samotnému šifrování zpráv lze přistoupit dvěma způsoby. První z nich je využití symetrických šifrovacích algoritmů, které mají výhodu v poměrně vysoké rychlosti zpracování dat a k šifrování či dešifrování používají stejný klíč. Využití stejného klíče pro obě operace je ovšem i dosti problematické, protože každý, kdo jej vlastní může zprávy vytvářet, ale i číst. Zde je poté nutné nějakým způsobem zabezpečit předání klíče, aby nebyl odposlechnut a následně zneužit. Naproti tomu existuje řešení v podobě asymetrických algoritmů, kde jedna strana zpřístupňuje svůj veřejný klíč, kterým protistrana zašifruje zprávu. Tuto zprávu pak může dešifrovat pouze privátní klíč, který se nikde nepřenáší. Nevýhodou tohoto řešení je poměrně vysoká výpočetní náročnost a potřeba mít dva páry klíčů pro obousměrnou komunikaci. Spojením těchto dvou rozdílných způsobů lze následně získat rychlé šifrování a dešifrování zpráv pomocí symetrického algoritmu a bezpečný přenos využitého klíče pomocí asymetrického algoritmu.

Cestou využitím právě dvou algoritmů jsem se vydal i já ve své práci. Pro samotné šifrování obsahu zpráv jsem zvolil symetrický algoritmus AES s 256 bitovým klíčem, tento klíč je přenesen v zašifrované formě pomocí asymetrického algoritmu RSA, jehož klíč má délku 2048 bitů. Dvojice privátního a veřejného klíče je tvořena ve druhém kroku registraci uživatele na jeho zařízení. K tomuto posloužila třída `RSACryptoServiceProvider`, jež je stejně jako využitá třída pro algoritmus AES součástí API operačního systému Windows Phone. Aby bylo zajištěno bezpečné uložení hlavně privátní části klíče, je nutné využití pojmenovaného kontejneru třídy `CspParameters`. Ten umožní persistentní uložení po celou dobu, kdy je aplikace na zařízení nainstalována. Využití je pak patrné v ukázce kódu 5.6. Z vygenerované dvojice je na server odeslán veřejný klíč, který se ukládá do tabulky `pub_keys`, jejíž položky kromě samotného klíče tvoří ještě uživatelské jméno.

```
1  private string createNewPrivatePublicKeyPair(string id)
2  {
3    // instance kontejneru pro ulozeni dvojice asymetrickych klicu, jeho pojmenovani je
         zavisle na id uzivatele
4    CspParameters cspProvider = new CspParameters();
5    cspProvider.KeyContainerName = "key_pair_container_" + id;
6    // pri vytvoerni instance dojde i k naplneni kontejneru nove vygenerovanou dvojici
         klicu o delce 2048 bitu
7    RSACryptoServiceProvider rsaCryptoProvider = new RSACryptoServiceProvider(2048,
         cspProvider);
8    // na server se bude odesilat pouze verejna cast klice (to zajisti parametr false), ta
          se jeste prekoduje do BASE64
9    byte[] publicKeyBytes = Encoding.UTF8.GetBytes(rsaCryptoProvider.ToXmlString(false));
10   return Convert.ToBase64String(publicKeyBytes);
11 }
```

Zdrojový kód 5.6: Funkce sloužící pro vygenerování dvojice privátní a veřejný klíč a následnému vrácení veřejné části klíče zakódované do podoby BASE64.

Veřejný klíč je pak získáván společně s dalšími daty ze serveru při přidávání nových kontaktů do uživatelova seznamu. Když chce uživatel začít konverzaci, která ještě neexistuje, dojde k volání metody `CreateNewAesKeyAndSend`. Tato metoda se stará o vygenerování nového klíče a inicializačního vektoru pro symetrický algoritmus AES. Tyto dvě hodnoty se uloží v databázi pod příslušnou konverzací a následně se také zašifrují veřejným klíčem protistrany. Poté dojde k jejich odeslání ve formě zprávy. Od této chvíle jsou již veškeré odchozí textové zprávy šifrované. Jakmile protistrana obdrží zprávu se zašifrovaným klíčem a inicializačním vektorem algoritmu AES, dojde k volání metody `processKey`. Ta pomocí privátního klíče obsah zprávy dešifruje a uloží příslušné hodnoty do databáze k nově vzniklé konverzaci. Tímto dojde k ustavení bezpečné komunikace i druhou stranou.

Aby měl uživatel jistotu, že si jeho konverzace nikdo nepřečte ani v situaci, kdy půjčí svůj telefon třetí osobě, rozhodl jsem se implementovat zabezpečení aplikace pomocí PIN kódu. Tento číselný řetězec si uživatel může vytvořit v nastavení. Po jeho uložení bude vždy při spouštění aplikace vyzván k jeho zadání, tak jak je vidět na obrázku 5.8. Stránka pro zadání PIN kódu se sestává z matice tlačítek a textového pole. Při návrhu byl kladen důraz na podobnost s obrazovkou sloužící stejnému účelu při odemykání uzamčeného telefonu. Při zadávání číselného kódu, jsou uživateli zobrazovány znaky hvězdička v přesném počtu, jako je počet již zadaných číslic. Pokud se uživatel splete, má možnost mazání již zadaných znaků. Pokud zadá stejný počet číslic, jako je hodnota PINu uloženého v nastavení, dojde k porovnání obou řetězců. Pokud se shodují, je uživatel propuštěn dál do aplikace. Pokud se řetězce neshodují, je uživatel o tomto stavu informován a má ještě dva pokusy. Pokud

ani jeden z nich není úspěšný, je aplikace ukončena. Délka PIN kódu není nijak omezována a uživatel má nad ní plnou svobodu.

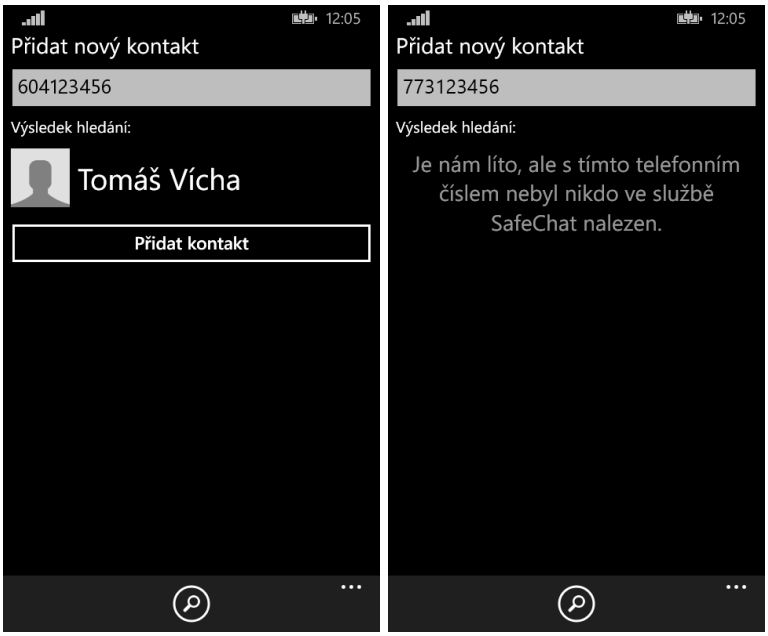
Obrázek 5.8: Obrazovka pro zadání PIN kódu, která se zobrazí ihned po spuštění aplikace, pokud si uživatel zadal ochranu PIN kódem.

Kromě stránky pro zadávání kódu PIN přibyla i stránka pro ruční vyhledávání kontaktu pomocí telefonního čísla, tak jak bylo žádáno testovacími uživateli. Odkaz na ní vede z hlavní stránky, kde na aplikačním panelu přibylo tlačítko se symbolem plus a textem „přidat kontakt“. Samotná stránka obsahuje pouze vstupní textové pole na zadání telefonního čísla a tlačítko v aplikačním panelu na spuštění vyhledávání. Pokud je uživatel se zadaným telefonním číslem na serveru nalezen, je zobrazeno jeho jméno spolu s tlačítkem pro přidání do seznamu kontaktů. Pokud ovšem na serveru nikdo se stejným telefonním číslem nalezen není, je o tom uživatel informován. Tyto dvě situace lze vidět na obrázku 5.9.

V předposlední části této etapy bylo potřeba vymyslet název a logo pro nově vzniklou aplikaci. S tímto mi pomohli mí přátelé, coby testovací uživatele aplikace. Po pár nápadech vznikl název SafeChat, který dokonale vystihuje podstatu celé aplikace. Jako logo byl vybrán podobný obrázek, jaký je v mobilním operačním systému Windows Phone používán pro dlaždici aplikace zprávy. Do něj jsem místo emotikonu v podobě znaků umístil obrázek visacího zámku, který má symbolizovat zabezpečení každé jednotlivé přenášené zprávy.

Na závěr proběhlo ještě poslední testování na uživatelích, ve kterém jsem si ověřil, jestli je koncept celé aplikace srozumitelný a jednoduchý na používání. Toho jsem dosáhl rozšířením okruhu uživatelů o tři doposud nezúčastněné jedince, kteří měli možnost vyzkoušet již hotovou aplikaci. Z jejich závěrů vyplynulo, že se jedná o odlehčenou verzi již existujících řešení, což potvrdilo splnění jednoho z bodů požadavků stanovených na začátku. Celkově se uživatelům rozhraní aplikace líbí a jsou spokojeni s jeho jednoduchostí. Mezi body, které ovlivňují jejich celkový dojem na výsledné řešení a také na rozhodnutí používat tuto aplikaci v budoucnu patří následující:

- Současná nemožnost běhu aplikace na pozadí a s tím související nutnost neustálého spouštění pro kontrolu nových zpráv
- Potřeba šifrovaného přenosu malých souborů jako jsou dokumenty a fotografie



Obrázek 5.9: Obrazovka pro vyhledání kontaktu ve službě SafeChat. Na obrázku vlevo bylo vyhledávání úspěšné a je nabídnuta možnost přidat kontakt. Pravý obrázek reprezentuje neúspěšné hledání.

5.4 Zhodnocení aplikace

Aplikace SafeChat splnila všechny požadavky, které byly stanovené společně s budoucími uživateli při jejím návrhu. Uživatelské rozhraní bylo již od prvního návrhu, přes všechny etapy vývoje kladně hodnoceno za svoji střídmost a jednoduchost. Mezi další pozitiva lze označit znatelně rychlejší spouštění a obnovování na všech testovacích zařízeních oproti konkurenčním komunikačním nástrojům na platformě Windows Phone. Toto je do jisté míry způsobeno absencí přidané funkcionality, která je u ostatních aplikací poměrně bohatá. Z pohledu bezpečnosti přenášených zpráv je výsledné řešení na dobré úrovni díky samotnému point-to-point šifrování a využití protokolu TLS pro komunikaci se serverem. Aplikace také umožnuje nastavení PIN kódu, který zabraňuje neoprávněnému spuštění a tím také zvyšuje celkové zabezpečení. Pro budoucnost aplikace, ale také celé služby bude nutné dále pracovat na vylepšeních a přidávat funckionalitu, kterou si uživatelé žádají.

Kapitola 6

Závěr

Cílem této závěrečné práce bylo vytvořit uživatelsky přívětivou a jednoduchou aplikaci, která bude poskytovat zabezpečenou komunikaci pro její uživatele. Avšak v průběhu řešení začala spíše vznikat celá komplexní služba pro uživatele mobilních telefonů s operačním systémem Windows Phone. Tato služba jim umožňuje mezi sebou komunikovat prostřednictvím zpráv, jejichž obsah je čitelný pouze pro zúčastněné strany dané konverzace. Přínos této práce vidím v nabídnutí alternativního komunikačního nástroje pro uživatele, kteří chtějí komunikovat se svými přáteli, rodinou či kolegy způsobem, který zaručí, že si jejich zprávy nepřečte nikdo jiný.

Pro budoucí vývoj aplikace, ale i celé služby, mám již v tuto chvíli nápady a postřehy, které bych rád převedl do reálné podoby. Jako první bych rád zajistil běh mobilní aplikace na pozadí a tím umožnil informování uživatele o nově příchozích zprávách, i když zrovna aplikaci nepoužívá. Další krok pro zatraktivnění služby a také jedna z věcí, kterou testovací uživatelé v závěru zmínili, je možnost bezpečného přenosu malých souborů jako jsou fotografie nebo dokumenty. Nakonec bych ještě zmínil záměr o rozšíření aplikace, která je v současné době určená pouze pro uživatele operačního systému Windows Phone, i na další mobilní platformy jako je Android a iOS.

Literatura

[1] Allison, M.: *A History of Windows Phone: The life and death of Microsoft's mobile platform*. Září 2015, [Online; navštíveno 19.04.2017].
URL https://mspoweruser.com/a-history-of-windows-phone-the-road-to-threshold/

[2] Bryntesson, P.: *Windows Phone 8.1 for Developers–Converging the App Models*. Březen 2014, [Online; navštíveno 17.04.2017].
URL https://blogs.msdn.microsoft.com/thunbrynt/2014/03/31/windows-phone-8-1-for-developersconverging-the-app-models/

[3] Burda, K.: *Úvod do kryptografie*. Brno: Akademické nakladatelství CERM, vydání první. vydání, 2015, ISBN 9788072049257.

[4] Dajbych, V.: *MVVM: Model-View-ViewModel*. Duben 2009, [Online; navštíveno 20.04.2017].
URL http://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel

[5] Hilyard, J.; Teilhet, S.: *C# 6.0 cookbook*. Sebastopol, CA: O'Reilly, fourth edition. vydání, 2015, ISBN 9781491921463.

[6] Luboslav Lacko: *Vývoj aplikací pro Windows 8.1 a Windows Phone*. Brno: Computer Press, první vydání, 2014, ISBN 9788025138229.

[7] Mendelevich, A.: *AdDuplex Windows Device Statistics Report–February, 2017*. Únor 2017, [Online; navštíveno 12.04.2017].
URL http://blog.adduplex.com/2017/02/adduplex-windows-device-statistics-report-february-2017.html

[8] Piper, F. C.; Murphy, S.: *Kryptografie*. Praha: Dokořán, první vydání, 2006, ISBN 8073630745.

[9] Saint-André, P.; Smith, K. T.; Troncon, R.: *XMPP*. Sebastopol, CA: O'Reilly, první vydání, c2009, ISBN 9780596521264.

[10] Skype: *Does Skype use encryption?* [Online; navštíveno 02.05.2017].
URL https://support.skype.com/en/faq/FA31/does-skype-use-encryption?platform=windows-phone

[11] Skype: *What do I need to start using Skype?* [Online; navštíveno 02.05.2017].
URL https://support.skype.com/en/faq/FA10328/what-do-i-need-to-start-using-skype?q=system+requirements

[12] Telegram: *Telegram FAQ*. [Online; navštíveno 02.05.2017].
URL https://telegram.org/faq

[13] Vasile, C.: *Modern UI Style Design by Microsoft*. Listopad 2012, [Online; navštíveno 25.04.2017].
URL https://designmodo.com/modern-ui/

[14] Viber: *How is my Viber account secured and encrypted?* [Online; navštíveno 02.05.2017].
URL https://support.viber.com/customer/portal/articles/2017401-viber-security-faq

[15] Viber: *Viber Encryption Overview*. [Online; navštíveno 02.05.2017].
URL https://www.viber.com/en/security-overview

[16] WhatsApp: *WhatsApp Encryption Overview*. Listopad 2016, [Online; navštíveno 02.05.2017].
URL https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf

[17] Żółkiewski, Z.: *Securing ejabberd with TLS encryption*. Březen 2016, [Online; navštíveno 17.04.2017].
URL https://blog.process-one.net/securing-ejabberd-with-tls-encryption/

Přílohy

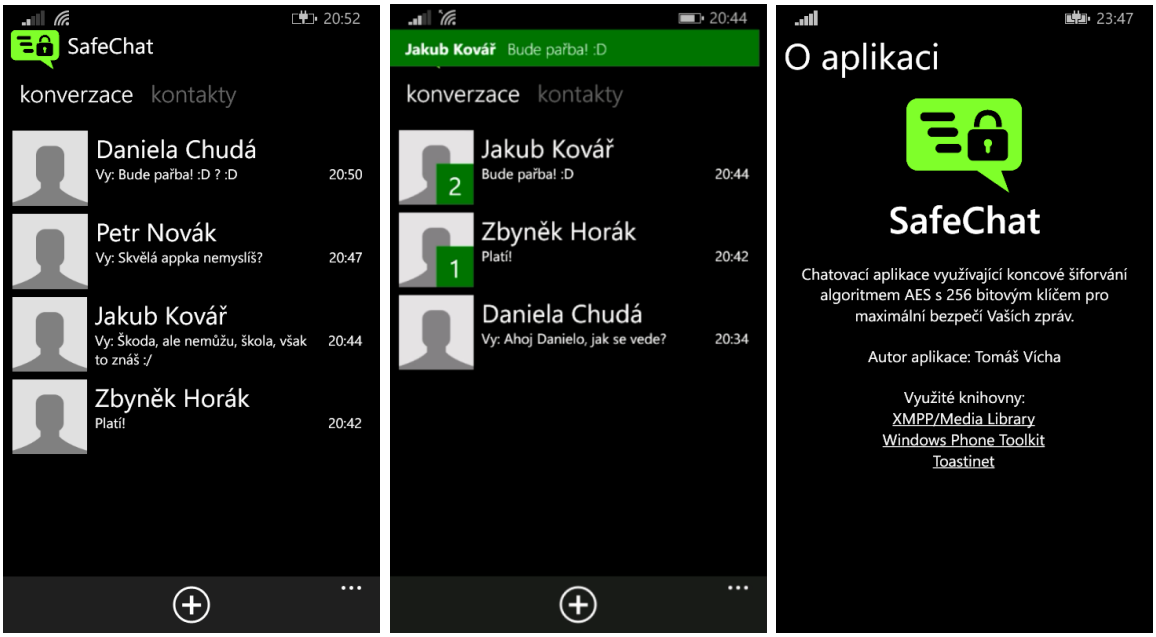
Příloha A

Obsah přiloženého paměťového média

- `/video` – Demonstrační video
- `/poster` – PDF dokument s plakátem
- `/doc` – Zdrojové kódy dokumentu této bakalářské práce
- `/src/project` – Projekt vývojového prostředí Visual Studio 2015
- `/src/project_xap` – XAP balíček výsledné aplikace pro platformu Windows Phone
- `/src/server_php` – Obsahuje zdrojové kódy v jazyce PHP
- `/src/server_sql` – Obsahuje skript pro vytvoření tabulek v databázi MySQL
- `/bp_xvicha04.pdf` – Elektronická verze dokumentu bakalářské práce ve formátu PDF
- `/README.txt` – Obsahuje podrobnější informace o obsahu paměťového média

Příloha B

Snímky obrazovky finální verze aplikace



Obrázek B.1: Levý a prostřední obrázek pocházejí ze zobrazení konverzací na hlavní stránce aplikace. Na prostředním obrázku je vidět notifikace o nově příchozí zprávě a také počty nepřečtených zpráv u jednotlivých konverzací. Pravý obrázek reprezentuje stránku „O aplikaci“