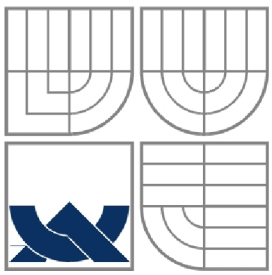
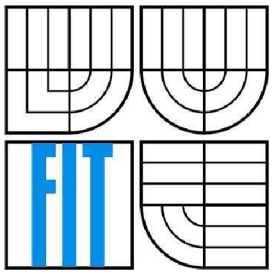


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ICQ KLIENT PRO PDA

ICQ CLIENT FOR PDA

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — Jiří Gregorovič
AUTHOR

VEDOUCÍ PRÁCE — Ing. Michal Pajgrt
SUPERVISOR

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Gregorovič Jiří**
Obor: Informační technologie
Téma: **ICQ klient pro PDA**
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se vývojovým prostředím pro platformu Windows Mobile.
2. Seznamte se s ICQ protokolem.
3. Implementujte ICQ protokol formou knihovny.
4. Vytvořte ukázkového ICQ klienta pro PDA, který bude demonstrovat funkčnost knihovny.
5. Diskutujte zvolené řešení a možnosti dalšího rozvoje projektu.

Literatura:

- CHAPPEL, David. Understanding .NET. [s.l.] : Pearson Education Inc., 2002. 327 s. ISBN 0201741628
- ISAKSSON, Henrik. ICQ protocol specification [online]. 2000. Dostupný z WWW: <http://iserverd.khstu.ru/docum_ext/icqv5.html>

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1-2 zadání

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: **Pajgrt Michal, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. **Pan**

Jméno a příjmení: **Jiří Gregorovič**
Id studenta: 79024
Bytem: Kollárova 793/28, 664 51 Šlapanice
Narozen: 12. 10. 1985, Brno
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: ICQ klient pro PDA
Vedoucí/školitel VŠKP: Pajgrt Michal, Ing.
Ústav: Ústav počítačových systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

V práci je představena cílová platforma Windows Mobile a její vývojové prostředky. Práce dokumentuje implementaci komunikačního protokolu ICQ formou knihovny, návrh a implementaci klienta pro vytvořenou knihovnu protokolu ICQ. Implementace je provedena ve vývojovém prostředí Visual Studio 2008, v jazyce Visual C#.Net.

Klíčová slova

ICQ, IM, OSCAR, Windows Mobile, PDA, .NET Compact Framework, Visual Studio

Abstract

This work deals with Windows Mobile platform and it‘s development tools. This work documents implementation of ICQ communication protocol in form of shared library, proposal and implementation of a client for created library. Implementation is made in development environment Visual Studio 2008 in a Visual C#.Net language.

Keywords

ICQ, IM, OSCAR, Windows Mobile, PDA, .NET Compact Framework, Visual Studio

Citace

Gregorovič Jiří: ICQ klient pro PDA. Brno, 2008, bakalářská práce, FIT VUT v Brně.

ICQ klient pro PDA

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Pajgrta
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

Poděkování

Děkuji mému vedoucímu Ing. Michalu Pajgrtovi za odborné vedení a poskytnutí cenných informací pro návrh a implementaci knihovny a programu.

Obsah

Úvod

Už od vzniku lidstva si lidé potřebovali vyměňovat informace. Z počátku bylo jedinou možností předání informace osobním kontaktem. Později vznikly zařízení pro výměnu informací. Mezi tyto zařízení například patří telegraf nebo telefon. Se vznikem výpočetní techniky a především internetu se pro výměnu informací otevřely nové komunikační kanály. K těmto kanálům patří například e-mail, nebo instant messaging (IM). Instant messaging je internetová služba pro posílání zpráv a souborů mezi uživateli. Je to jeden z nejrychlejších způsobů komunikace na velké vzdálenosti.

Tato práce se zabývá vývojem aplikace pro instant messaging. V práci je popsán návrh a implementace nejrozšířenějšího IM klienta v České republice, který se nazývá ICQ. Cílovou platformou vytvářeného klienta je PDA (Personal Digital Assistant) s operačním systémem Windows Mobile. PDA je kapesní počítač sloužící jako osobní zápisník a organizér.

V první kapitole je popsána platforma Windows Mobile a dostupné vývojové prostředky. Druhá kapitola popisuje komunikační protokol ICQ. Ve třetí kapitole je zdokumentována implementace protokolu ICQ formou knihovny a ve čtvrté implementace klienta pro vytvořenou knihovnu. V páté kapitole je diskutováno zvolené řešení a možnost dalšího rozvoje.

1 Platforma Windows Mobile

V této kapitole je popsána platforma Windows Mobile. Jsou zde vyjmenovány hardwarové platformy, které podporuje a stručný seznam existujících verzí. Kapitola obsahuje seznam dostupných vývojových prostředků a jejich porovnání.

1.1 Základní pohled

Windows Mobile je kompaktní operační systém se sadou základních aplikací pro mobilní zařízení založený na programátorském rozhraní WIN32 API. Základními aplikacemi jsou myšleny: e-mailový klient, webový klient, mobilní kancelářský balík, multimédiální přehrávač a mnohé další.

Windows Mobile je platforma vytvořená firmou Microsoft. Byla vytvořena tak, aby byla v mnoha ohledech podobná desktopové verzi Windows – snadno ovladatelná a estetická. Pro tuto platformu existují propracované vývojové nástroje a dokumentace usnadňující vývoj aplikací. Windows Mobile je jedním z nejrozšířenějších mobilních operačních systémů, důkazem je i fakt, že Microsoft dodává licence Windows Mobile čtyřem z pěti největších výrobců mobilních telefonů. Čerpáno z [3].

1.2 Hardware a verze

Windows Mobile je podporován na mnoha hardwarových platformách, jakými jsou PocketPC, SmartPhone, Portable Media Center a dokonce i některá zařízení v automobilech.

PocketPC (PPC) je zařízení na kterém operační systém Windows Mobile vznikl a které dělíme na dva typy. Na zařízení, které mají vlastnosti mobilního telefonu a na zařízení které je nemají. Může být snadno rozšířeno přídavnými moduly, jakými jsou GPS přijímače, čtečky čárových kódů a kamery. Pro PPC s vlastnostmi telefonu je dnes používána edice „Windows Mobile 6 Professional“ a pro PPC bez vlastností telefonu edice „Windows Mobile 6 Clasic“.

SmartPhone je další hardwarová platforma na které se začal používat operační systém Windows Mobile. Neexistuje průmyslový standard, který toto zařízení definuje. Podle firmy Microsoft se SmartPhone od PPC liší zaměřením na ovládání jednou rukou. Z toho vyplývá absence dotykové obrazovky a přítomnost jednoduché klávesnice. Pro SmartPhone se dnes používá edice „Windows Mobile 6 Standard“.

Portable Media Center je zařízení, které je zaměřeno na integraci s multimédiálním rozšířením operačních systémů Windows (Windows Media Center) a standartním multimediálním přehrávačem (Windows Media Player). Pro tato zařízení byla vytvořena edice „Windows Mobile for Portable Media Centers“.

Jako poslední hardwarovou platformou, kde se používá Windows Mobile, jsou vestavěná zařízení v automobilech. Tato zařízení obsahují jak USB port a bluetooth, tak i navigační (GPS) a komunikační (GSM) modul. Pro tyto účely se používá edice „Windows Mobile for Automotive“.

Operačních systémů Windows Mobile bylo vytvořeno mnoho verzí. První verze nazvaná PocketPC 2000 byla vytvořena pouze pro PPC. Podporovala rozlišení 320 x 240 bodů (QVGA) a CompactFlash nebo MultiMediaCard jako rozšíření interní paměti zařízení. Další verzi (PocketPC 2002) bylo již možné použít pro zařízení typu SmartPhone, díky přidání podpory hardwarové klávesnice, kterou bylo možné použít pro ovládání zařízení namísto dotykového displeje. Oproti předchozí verzi obsahovala více aplikací a vylepšené uživatelské rozhraní. Windows Mobile 2003 byla první verzí používající logo Windows Mobile a byla přidána podpora komunikačního standardu bluetooth. Ve verzi Windows Mobile 5 se oproti předchozím vydáním změnil typ použitých pamětí z RAM na Flash, což mělo sice za následek snížení odezvy celého zařízení, ale došlo ke snížení spotřeby energie až o 50%. V současné době je aktuální verze Windows Mobile 6.1 a byla oznámena verze Windows Mobile 7.0.

1.3 Vývojové nástroje

Pro vývoj nových aplikací pro mobilní zařízení je dostupných několik vysoce kvalitních nástrojů. Většina z nich je integrována do jednotného vývojového prostředí jménem Visual Studio.

Jsou tři možnosti jak vytvářet aplikace pro platformu Windows Mobile. První je možnost vytvořit program v nativním kódu pomocí programovacího jazyka Visual C++. Další možností je vytvoření aplikace v řízeném (managed) kódu. Tyto dva přístupy vytváří aplikaci jako tlustého (thick) klienta. Tedy výsledná aplikace obsahuje veškerou aplikační logiku. Rozdíly mezi oběma přístupy jsou probrány v následující podkapitole 1.3.1. Poslední možností je vytvoření aplikace v podobě tenkého (thin) klienta, kdy je aplikační logika převedena na třetí stranu, server, který zpracovává dotazy od klienta a vrací výsledky požadovaných operací.

Pro demonstrační aplikaci jsem zvolil způsob tlustého klienta, který se zdá pro danou úlohu nejvhodnější. Další informace viz [2].

1.3.1 Nativní nebo řízený kód

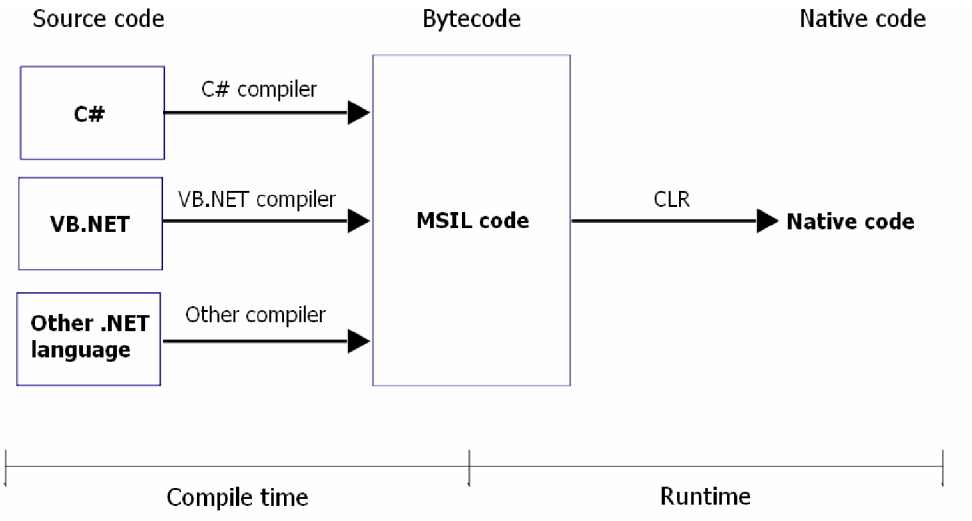
Nativní kód je vykonáván přímo v procesoru. Zdrojové soubory jsou překladačem přeloženy přímo do kódu procesoru v době kompilace. Naproti tomu řízený kód nelze spustit přímo v procesoru. Je to dáno tím, že zdrojové soubory nejsou přeloženy do kódu procesoru, nýbrž do mezikódu. Tento mezikód je do kódu procesoru přeložen až v době spuštění programu. Postup ilustruje obrázek 1. Tento přístup vyžaduje na počítači, kde je program spuštěn, přítomnost speciálního software. Jmenuje se Common Language Runtime (CLR) a společně s knihovnami základních tříd tvoří vývojovou

platformu nazvanou .Net Framework. CLR je virtuální prostředí, které řídí překlad mezikódu pomocí Just In Time (JIT) překladače. Obsahuje řadu služeb, jimiž jsou:

- Přidělování paměti
- Uvolňování paměti (Garbage Collection)
- Řízení vláken
- Zpracování vyjímek
- Bezpečnost

V porovnání nativního a řízeného kódu je nativní kód rychleji vykonáván, alokuje si méně systémových prostředků a výsledný program má menší velikost a je flexibilnější. Má však i nevýhody v podobě nutnosti uvolňovat alokované systémové prostředky a ošetřit mnohem více chybových stavů. To má za následek výskyt mnoha chyb ve výsledné aplikaci. Naproti tomu řízený kód je mnohem efektivnější a bezpečnější. Díky přítomnosti garbage collectoru se programátor nemusí starat o uvolňování systémové paměti.

Zadanou aplikaci jsem se rozhodl vytvořit v řízeném (managed) kódu.



Obrázek 1.1 Řízený (managed) kód – překlad a spuštění, převzato z [5]

1.3.2 .Net Compact Framework

.Net Compact Framework je odvozen od .Net Frameworku. Je však přizpůsoben pro vývoj aplikací pro prostředí operačního systému Windows Mobile. .Net Compact Framework používá některé knihovny tříd jako plný .Net Framework a také některé nové knihovny tříd určené speciálně pro mobilní zařízení.

Pro správný běh aplikací musí cílová platforma podporovat .Net Compact Framework Common Language Runtime (CF CLR), což Windows Mobile splňuje. CF CLR je virtuální prostředí CLR upravené pro mobilní zařízení. Také aplikace vytvořené pro .Net Compact Framework nelze spustit na počítači, na kterém je nainstalována plná verze .Net Frameworku. Není to proto, že by

výsledný mezikód nebyl mezi oběma platformami kompatibilní, ale z důvodu použití speciálního digitálního podpisu, který zabraňuje spuštění aplikace napsané pro .Net Framework na zařízení s nainstalovaným .Net Compact Framework.

V současné době je aktuální verze .Net Compact Framework 3.5 RTM. Vývoj aplikací pro .Net Compact Framework je například možný v integrovaném vývojovém prostředí Visual Studio.

1.3.3 Visual Studio 2008

Visual Studio je integrované vývojové prostředí (IDE) od firmy Microsoft. Může být použito k vývoji jak konzolových aplikací, Windows Forms aplikací tak i webových aplikací. Podporuje tvorbu programů jak v nativním tak řízeném (managed) kódu.

Toto integrované prostředí obsahuje mnoho nástrojů potřebných pro vývoj aplikací jakéhokoliv rozsahu. Jedním z těchto nástrojů je například vynikající debuger použitelný jak pro neřízený (native) kód tak pro řízený (managed) kód. Pokud je dostupný zdrojový kód, debuger ukazuje řádek právě zpracovávaného kódu a pokud není, tak ukazuje binární kód sledovaného procesu. Samozřejmostí je možnost krokování kódu či prohlížení a editace aktuálních hodnot proměnných. Debuger se také může připojit k již běžícímu procesu.

Dalším důležitým nástrojem je editor kódu se zvýrazňováním syntaxe a automatickým doplňováním kódu pomocí funkce IntelliSense. IntelliSense doplňuje kód nejen pro proměnné, metody, funkce, ale i pro klíčová slova a konstrukce jazyka (například smyčky, podmíněné skoky, apod.), čímž výrazně pomáhá programátorovi ve vývoji a urychluje psaní zdrojového kódu.

Dalším velice propracovaným nástrojem jsou vizuální designery. Vizuální designer je nástroj vytvářející zdrojový kód pomocí jeho grafické reprezentace. Například WinForms designer je použit pro tvorbu grafického uživatelského rozhraní aplikace (GUI), nebo Class designer pro vytváření, editaci tříd a jejich vztahů.

Protože podpora pro programovací jazyky je tvořena formou balíčků zvaných „Language service“ (Jazyková služba), nedá se striktně říci, jaké jazyky podporuje. Pokud pro daný jazyk existuje přídavný balíček tak jej Visual Studio podporuje. V základní verzi Visual Studia je integrováno několik jazykových služeb. Mezi ně patří C++ (Visual C++), C# (Visual C#), Visual Basic .Net, HTML/XHTML, JavaScript, CSS.

Visual Studio existuje v mnoha edicích od Visual Studia Express Edition (zdarma pro jakékoli použití), přes Visual Studio Standard (příležitostný programátor neprofesionál), Visual Studio Professional (profesionální programátor jednotlivec), až po Visual Studio Team Suite (pro různě velký programátorský tým).

2 ICQ

ICQ je komunikační služba (tzv. instant messager), pomocí kterého je možné mezi uživateli posílat nejen textové zprávy, ale i soubory nebo hrát jednoduché hry a další funkce přidané pomocí zásuvných modulů (plugin). Byl vyvinut Izraelskou firmou Mirabilis v roce 1996. Mirabilis byla založena čtyřmi Izraelci Amnonem Amirem, Arikem Vardim, Sefim Vigiserem, Yairem Goldfingerem. Později 8. června 1998 ICQ koupila americká firma AOL (America Online) za 407 miliónů dolarů, která ho vlastní a provozuje dodnes.

Pro identifikaci uživatelů v ICQ je používáno identifikační číslo UIN (Universal Internet Number nebo Unified Identification Number). Toto číslo obdrží každý uživatel ICQ ihned po registraci svého účtu. Je to jediná identifikace uživatele, která se nedá u již registrovaného uživatele změnit. Všechny ostatní identifikační údaje, jako je jméno, přezdívka, e-mail a další, jsou modifikovatelné.

ICQ využívá pro svou komunikaci protokolu OSCAR (Open System for CommunicAtion in Realtime). Původně ICQ pro komunikaci nevyužívalo protokol OSCAR, ale používal svůj vlastní protokol. OSCAR byl navržen pro jiného IM klienta, proto byly do specifikace OSCAR přidány některé datové požadavky specifické pro ICQ. Dříve však bylo možné používat protokol TOC, pro který 19. srpna 2005 AOL ukončila podporu a ke konci roku 2007 jej vypnula. Jakmile byl TOC protokol vypnut přestali neoficiální klienti využívající protokolu TOC fungovat.

ICQ je služba využívaná převážně v České republice, Slovenské republice a Rusku. V celosvětovém měřítku je nejpoužívanější službou IM od firmy Microsoft Windows Messager. Čerpáno z [11].

2.1 Výhody / Nevýhody

Největší výhodou je kompatibilita ICQ s AIM, komunikačním programem od stejné firmy. Někteří uživatelé považují za výhodu i přítomnost tzv. Xtraz. Xtraz je webově založená služba, která umožňuje například jednoduché hry, video přenosy, a jiné. V zemích kde je ICQ nejrozšířenějším IM je také výhodou počet uživatelů této sítě.

Asi tou nejdůležitější nevýhodou je uzavřenost služby společně s velmi diskutovanou licencí. Při instalaci oficiálního klienta ICQ je vyžadováno přečtení a souhlas s podmínkami. Tyto podmínky obsahují klauzuli, která přikazuje uživateli pravidelně kontrolovat, zda nedošlo k jejich změně. Ze sporných bodů podmínek stojí za zmínku:

- ICQ nesmí používat osoba mladší 13ti let
- Z oficiálního klienta se nesmí odstraňovat reklamní bannery
- ICQ protokol může být kdykoli upraven a celá síť vypnuta

- Nesmí být použita jinak než pro soukromou potřebu
- Cokoli co je posláno přes ICQ je majetkem AOL a může být jakýmkoli způsobem použito
- Pro síť ICQ se nesmí používat jiný než oficiální klient

Další nevýhodou ICQ je absence šifrování přenášených dat. AOL je také kritizována pro prakticky nulovou zákaznickou podporu a pro přítomnost adware v podobě zobrazovaných reklam jak v okně s kontakty, tak v okně pro posílání a příjem zpráv.

Nevýhody také plynou z použitého komunikačního protokolu – OSCAR. Především pro nemožnost komunikovat s jinými uživateli využívajících služeb jiné komunikační sítě nebo centralizovanost služby.

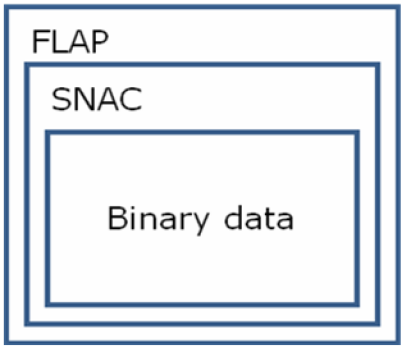
Z důvodu licenčních podmínek jsem se rozhodl implementovat klienta ve verzi protokolu, který používá veřejně dostupný server na starší verzi protokolu OSCAR v8. Tato verze není plně kompatibilní s momentálně používanou verzí v11.

2.2 OSCAR protokol

ICQ, stejně jako AIM, používá pro komunikaci se serverem komunikační protokol OSCAR (Open System for ComunicAtion in Realtime). Je to binární protokol komunikující po síti pomocí spojové služby TCP.

Protokol OSCAR je složen z několika dílčích protokolů. Prvním z nich je protokol FLAP, který se stará o komunikaci na nízké úrovni. Protokol FLAP je detailněji popsán v podkapitole 2.2.1. Dalším z použitých protokolů je protokol SNAC, použitý pro zapouzdření přenášených dat. Jeho struktura a vlastnosti jsou popsány v podkapitole 2.2.2.

Tato dokumentace nemá za účel detailní popis protokolu, ale jen základní pohled na jeho strukturu z pohledu studijní implementace v rámci bakalářské práce. Jeho detailní popis lze nalézt v literatuře [7] a [8].



Obrázek 2.1 Formát zprávy protokolu OSCAR

2.2.1 FLAP protokol

FLAP je nízkoúrovňový protokol, starající se o řízení komunikace na aplikační úrovni TCP modelu (model pro komunikaci na síti internet). Pro řízení komunikace je použito několika kanálů. Kanály jsou metodou pro další dělení komunikace napříč jednoho vytvořeného spojení (soketu). Poté co je vytvořeno nové spojení v kanálu číslo 1, jsou přenášena data v kanálu číslo 2, dokud server nepošle chybové hlášení v kanálu číslo 3, nebo nedojde ke korektnímu odpojení klienta prostřednictvím kanálu číslo 4. Přehled všech kanálů je v tabulce 2.1. Čerpáno z [7].

Číslo kanálu	Popis
0x01	Vytvoření nového spojení
0x02	SNAC data
0x03	FLAP chyba
0x04	Uzavření spojení
0x05	Udržování spojení

Tabulka 2.1 Seznam FLAP kanálů

2.2.2 SNAC protokol

SNAC protokol je použit pro přenášená data protokolu FLAP. Pro tento účel je v protokolu FLAP vymezen kanál číslo 2 a není přenášen v jakémkoli jiném kanálu.

Každý datový požadavek je rozpoznán podle rodiny a příkazu z rodiny. Rodina definuje skupinu příkazů mající jistou logickou spojitost. Datový požadavek potom obsahuje potřebná data charakterizující daný příkaz. V tabulce 2.2 je uvedeno několik základních rodin protokolu SNAC. Čerpáno z [7].

Číslo rodiny	Popis rodiny
1	Obecné funkce
2	Uživatelské informace
3	Řízení seznamu kontaktů
4	ICBM (Služba posílání zpráv)
19	SSI (Služba ukládání dat)
21	Služby použité v původním ICQ

Tabulka 2.2 Vybrané rodiny protokolu SNAC

2.3 TOC protokol

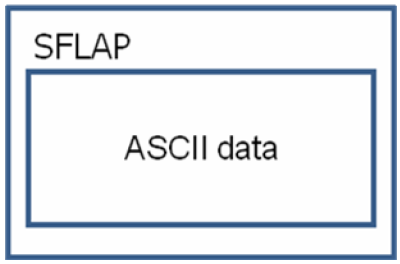
TOC (Talk to OSCAR) je protokol, který byl používán v klientech třetích stran a některými klienty vytvořenými AOL. Je to textový protokol komunikující po síti pomocí spojové služby TCP. TOC ve vztahu k protokolu OSCAR vystupuje jako tzv. wrapper. Jinak řečeno klient komunikuje s TOC serverem pomocí TOC protokolu a ten s ICQ serverem pomocí protokolu OSCAR. Předávaná data mezi protokoly překládá. TOC je podmnožinou OSCAR. Tedy všechny příkazy TOC jsou přítomny i v protokolu OSCAR, ale existuje několik OSCAR příkazů, které nejsou v protokolu TOC implementovány (např. ikona kontaktu, posílání souborů, a další). Tento protokol byl uvolněn jako otevřený pod GPL licencí, avšak ten samý rok byla licence změněna na uzavřenou. Nakonec byla podpora protokolu TOC zastavena v roce 2005.

TOC pro řízení přenosu používá podobný protokol jako OSCAR (SFLAP). SFLAP se od FLAP liší pouze v použitém počtu kanálů, které jsou pouze tři.

Seznam použitých kanálu je v tabulce 2.2.

Číslo kanálu	Popis
0x01	Vytvoření nového spojení
0x02	ASCII data
0x03	Není použit
0x04	Není použit
0x05	Udržování spojení

Tabulka 2.3 Seznam SFLAP kanálů



Obrázek 2.2 Formát zprávy protokolu TOC

3 Implementace protokolu ICQ

V následující kapitole je popsán návrh a implementace komunikačního protokolu ICQ.

3.1 Návrh

Úkolem je implementovat komunikační protokol ICQ formou knihovny. Rozhodl jsem se implementovat binární protokol OSCAR, protože je na internetu k dispozici mnohem více zdrojů k jeho pochopení a implementaci než k TOC protokolu, který už není dále používán a vyvíjen. Implementoval jsem jeho následující funkce:

- Připojení/Odpojení uživatele
- Správa kontaktů
- Příjem/Odeslání zprávy
- Upozoňování o stavu kontaktu
- Zjištění informací o kontaktu
- Změna informací přihlášeného uživatele

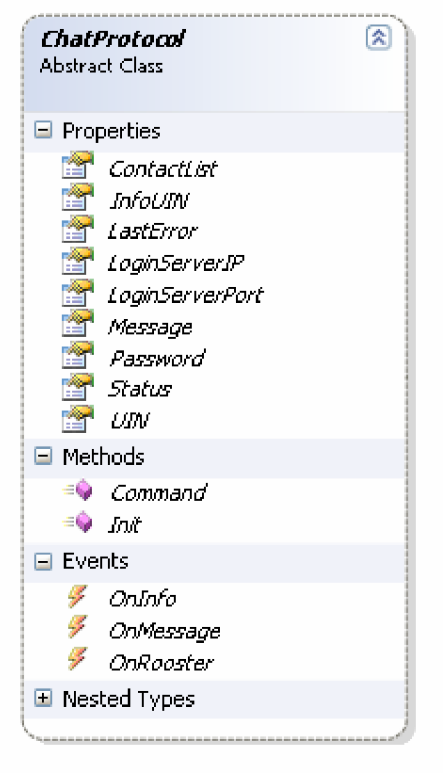
Návrh knihovny jsem rozdělil na dva základní problémy. Prvním je návrh rozhraní pro přístup k implementovaným funkcím knihovny a druhým je návrh třídy reprezentující vybraný komunikační protokol.

3.1.1 Návrh rozhraní knihovny

Rozhraní je navrženo univerzálně a je tedy možné na jeho základě implementovat jakýkoliv IM komunikační protokol.

Pro rozhraní jsem navrhl abstraktní třídu `ChatProtocol`, definující vlastnosti, metody a události, které musí implementovat třída reprezentující komunikační protokol. Navržená abstraktní třída je ilustrována na obrázku 3.1.

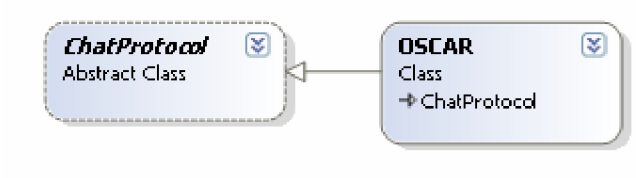
K inicializaci knihovny jsem vytvořil abstraktní metodu `Init()` a pro provedení požadované operace slouží abstraktní metoda `Command()`. Byly vytvořeny tři abstraktní události pro signalizaci příchozích dat. Událost `OnInfo` je určena pro předání informací o kontaktu ze serveru, druhá vytvořená událost `OnMessage` slouží pro předání příchozí zprávy a třetí událost `OnRooster` slouží pro předání informací o změně seznamu kontaktů.



Obrázek 3.1 Třída navržená pro rozhraní knihovny

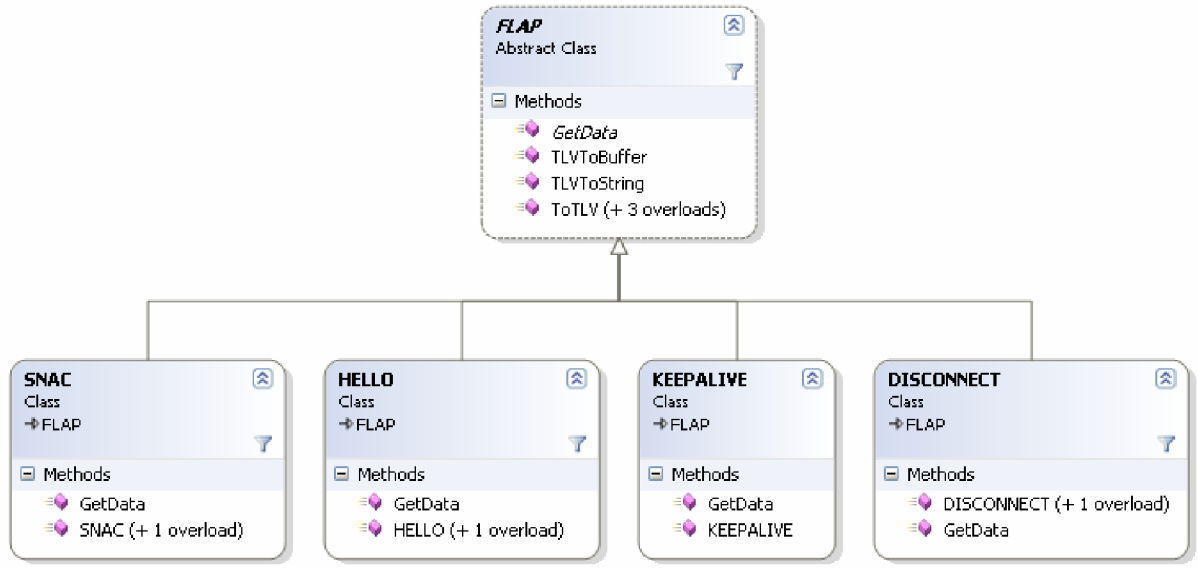
3.1.2 Návrh knihovny

Byla vytvořena třída `OSCAR` reprezentující použitý komunikační protokol, na kterou byla aplikována navržená abstraktní třída, použitá pro definici rozhraní knihovny. Ilustrováno na obrázku 3.2.



Obrázek 3.2 Diagram tříd knihovny komunikačního protokolu OSCAR

Pro řízení komunikace je použit protokol FLAP, který pro příjem a poslání dat používá několika kanálů. Aby bylo jednodušší vytvoření daného kanálu, je pro každý kanál vytvořena třída. Protože mají kanály společnou hlavičku definující velikost přenášených dat a číslo kanálu, byla vytvořena abstraktní třída `FLAP`, která slouží jako společný základ (předek) pro konstrukci kanálu. Dalším společným rysem kanálů je použití formátu dat (TLV), který definuje typ a délku dat. Proto byly vytvořeny metody pro sestavení a rozpoznání dat formátu TLV v abstraktní třídě `FLAP`. Tato třída také definuje abstraktní metodu `GetData()`, která má za úkol sestavení dat potřebných pro jejich poslání serveru. Pro analýzu příchozích dat je vyhrazen konstruktor každé třídy, reprezentující kanál, na kterém byla data přijata. Navržený diagram tříd demonstruje obrázek 3.3.



Obrázek 3.3 Diagram tříd navržených pro sestavení dat

3.2 Implementace

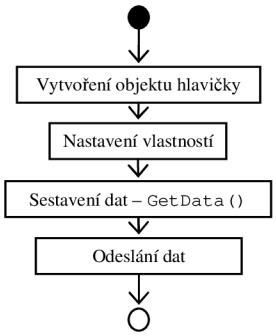
Prvním krokem bylo vybrat programovací jazyk, ve kterém bude knihovna implementována. Vývojová platforma .Net definuje základní sémantická pravidla, která jsou společná pro všechny jeho programovací jazyky. Tato sémantická pravidla se nazývají Common Language Specification (CLS). Protože jsou sémantická pravidla pro programovací jazyky .Net stejná, mažou se tak jejich výkonnostní rozdíly.

Pro implementaci jsem zvolil jazyk C#, kde hlavním důvodem výběru tohoto jazyka byla podobnost s programovacím jazykem C.

Protože je knihovna implementována použitím řízeného kódu je nutné vybrat verzi .Net Compact Framework pro implementaci. Vybral jsem verzi 3.5, která je nejnovější a oproti předcházejícím verzím obsahuje více základních tříd. Protože je na trhu od listopadu roku 2007, nebude problém s jejím rozšířením mezi uživateli.

3.2.1 Sestavení a odeslání dat

Pro odeslání datového požadavku na server je zapotřebí nejprve vytvořit objekt reprezentující komunikační kanál. V dalším kroku je nutné nastavit vlastnosti tohoto objektu, které obsahují specifikaci požadavku. Poté je potřeba data sestavit zavoláním metody `GetData()` vytvořeného objektu. Ve finální fázi už stačí jen sestavená data odeslat. Postup zobrazen na obrázku 3.4.

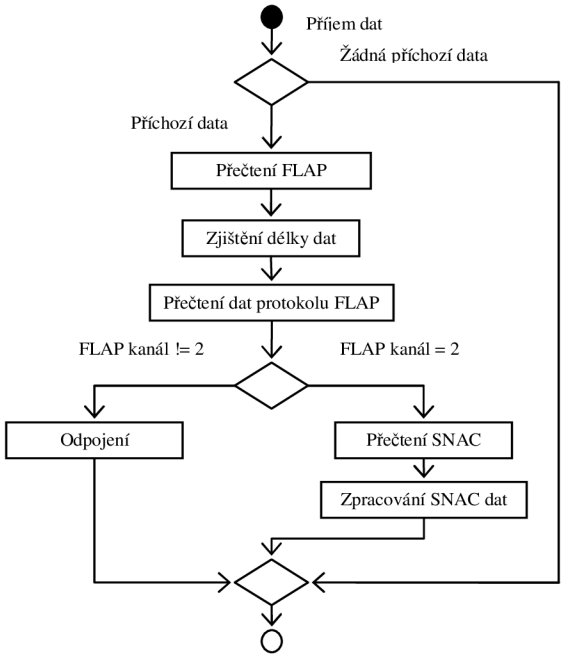


Obrázek 3.4 Sestavení a odeslání dat

3.2.2 Příjem a zpracování dat

Příjem a zpracování dat bylo implementováno v metodě `CheckServerMessage()` třídy `OSCAR`. Ta je volána periodicky za pomoci časovače, který je spuštěn po přihlášení uživatele na ICQ server a vypnut po jeho odhlášení.

Nejprve je zkontrolováno, zda byla přijata data. Pokud nebyla přijata, je metoda ukončena, v opačném případě je přečtena hlavička protokolu FLAP (6 bajtů). Z této hlavičky je zjištěna délka přijatých dat a kanál, na kterém byla data přijata. Tato data jsou přečtena a uložena do vyrovnávací paměti. Pokud byla data přijata na komunikačním kanálu číslo 2, jsou přečtena a zpracována data protokolu SNAC. Data přijatá v jiném kanále obsahují buď informace o odpojení, nebo chybové hlášení. Průběh přijetí dat je ilustrován na obrázku 3.5.

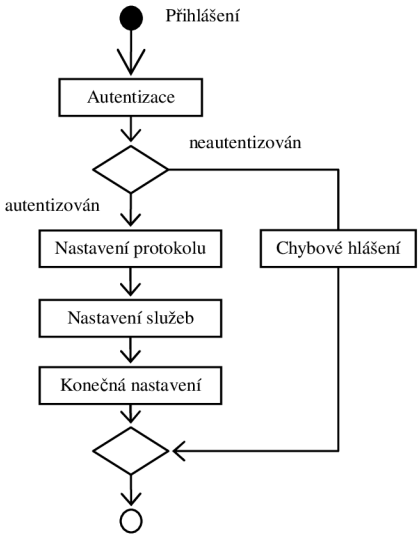


Obrázek 3.5. Příjem a zpracování dat

3.2.3 Přihlášení k serveru – příkaz `CMD_LOGIN`

Proces přihlášení k serveru je implementován v metodě `Login()` třídy `OSCAR`. Tato metoda je zavolána jako reakce na příkaz `CMD_LOGIN` předaný metodou `Command(EnumCommand)`.

Nejprve se klient přihlásí k autentizačnímu serveru. Pokud proběhne autentizace v pořádku je klientovi předána autentizační sekvence, která je poté použita k přihlášení na ICQ server. Postupně je po připojení k ICQ serveru vykonáno nastavení protokolu, nastavení služeb a konečná nastavení. Tento postup je zobrazen na obrázku 3.6.



Obrázek 3.6. Přihlášení k serveru

3.2.4 Provedení ostatních příkazů

Provedení ostatních příkazů je velice jednoduché. Po přijetí příkazu jako parametru metody `Command()` je vyvolána příslušná soukromá metoda. Většinou se jedná o poslání jednoho datového požadavku serveru. Jakmile je požadavek odeslán je provádění příkazu ukončeno i v případě očekávané odpovědi. Odpověď je následně přijata v soukromé metodě `CheckServerMessage()`, jejíž struktura byla vysvětlena v kapitole 3.2.2 a aktivována příslušná událost. Seznam implementovaných příkazů a prováděných soukromých metod je v tabulce 3.1.

Název příkazu	**Implementující metoda**
CMD_LOGIN	`private EnumError Login()`
CMD_DISCONNECT	`private EnumError Disconnect()`
CMD_ADDBUDDIE	`private EnumError AddBudie()`
CMD_DELBUDDIE	`private EnumError DellBudie()`
CMD_VIEWUSERINFO	`private bool GetUserInfo()`
CMD_SENDMESSAGE	`private bool SendMessage()`
CMD_UPDATESTATUS	`private bool UpdateStatus()`
CMD_CHANGEDETAILS	`private bool ChangeUserInfo()`

Tabulka 3.1 Seznam implementovaných příkazů

4 Implementace ICQ klienta

Kapitola obsahuje popis návrhu a implementace klienta pro ICQ, který má za úkol demonstrovat funkčnost knihovny s implementovaným komunikačním protokolem. Jsou zde popsány jednotlivé formuláře, jejich funkčnost a rozhraní.

4.1 Návrh

ICQ klient má sloužit pro demonstraci funkčnosti knihovny implementující komunikační protokol ICQ – OSCAR. Od toho se budou odvíjet i výsledné funkce aplikace. Těmito funkcemi jsou poslání/příjem zprávy, přidání kontaktu do seznamu, odebrání kontaktu ze seznamu, vyhledání kontaktu podle identifikačního čísla (UIN), zobrazení informací o kontaktu, zobrazení stavu kontaktu, změna stavu přihlášeného uživatele, změna detailů přihlášeného uživatele a zobrazení historie přijatých a odeslaných zpráv.

Aplikace bude obsahovat režim „na pozadí“. Ten umožní mít aplikaci spuštěnu a přijímat zprávy, i když není zrovna aktivní. Pro tento účel bude spuštění programu signalizováno v systémové liště. Tato signalizace bude obsahovat i aktuální stav uživatele aplikace (online, offline, away, n/a).

Klient bude mít implementovánu signalizaci stavu kontaktů a přijatých zpráv. Tato signalizace bude funkční pouze za předpokladu, že aplikace bude v režimu „na pozadí“. Při zobrazení signalizace bude možné přímo otevřít hlavní okno, nebo okno pro poslání zprávy kontaktu, který danou signalizaci vyvolal.

Aplikace bude mít i řadu nastavení rozdělených do dvou kategorií. První bude obsahovat nastavení uživatelského účtu (UIN, heslo). Do této kategorie také patří nastavení adresy, portu autentizačního serveru ICQ a možnost udržovat spojení speciálním paketem. Ve druhé kategorii bude nastavení pro automatické přihlášení po spuštění programu, zákaz signalizace změny stavu kontaktu. Toto nastavení bude ukládáno do konfiguračního souboru.

Vzhled klienta bude vycházet z klientů již existujících. Hlavní okno aplikace bude obsahovat seznam kontaktů, nastavení stavu přihlášeného uživatele, menu aplikace a menu pro práci s kontakty. Protože PDA je zařízení používané jedním uživatelem, bude klient optimalizován pro použití jedním uživatelem.

Okno zpráv bude implementováno poněkud netradičně. Všechny přijaté a odeslané zprávy se budou zobrazovat dohromady. Bude však možné si vybrat komu psanou zprávu poslat. Toto řešení jsem zvolil z důvodu malé plochy displeje, které obecně mobilní zařízení mají.

4.1.1 Návrh formulářů

Hlavním a nejdůležitějším formulářem, který jsem implementoval je `GICQForm`. Tento formulář představuje hlavní okno aplikace a obsahuje seznam kontaktů, nabídku aplikace a výběr stavu přihlášeného uživatele.

Vytvořil jsem formulář `SettingsForm` určený pro nastavení údajů pro přihlášení uživatele a pro nastavení ICQ klienta. Tento formulář je rozdělen na dvě části pomocí ovládacího prvku TabControl. V první části je nastavení komunikace a v druhé části je nastavení aplikace.

Pro zobrazení detailních informací o přihlášeném uživateli a o vybraném kontaktu ze seznamu kontaktů jsem vytvořil formulář `DetailsForm`. Pro zobrazení těchto detailních informací obsahuje formulář textové pole.

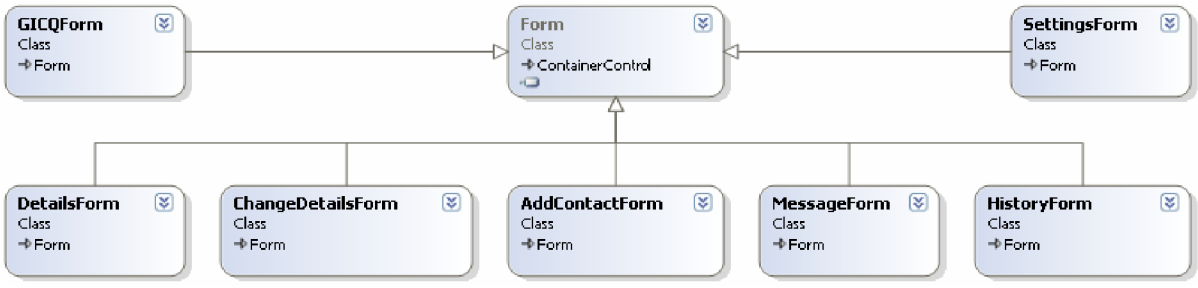
Pro změnu informací o aktuálně přihlášeném uživateli jsem implementoval formulář `ChangeDetailsForm`. Obsahuje ovládací prvky pro editaci nejvíce používaných informací o kontaktu (přezdívka, jméno, příjmení, e-mail).

Přidání kontaktu je možné ve formuláři `AddContactForm`. Tento formulář je možné použít pro vyhledání kontaktu podle identifikačního čísla ICQ (UIN). Přidání jsem rozdělil do dvou fází, kdy v první fázi je zapotřebí vyhledat kontakt a v druhé je možné nový kontakt přidat do seznamu kontaktů.

Pro poslání a příjem zpráv je implementován formulář `MessageForm`. Je rozdělen na tři části, kde v horní bude možné napsat novou zprávu, uprostřed zobrazovat historii poslaných/přijatých zpráv a v dolní části vybrat příjemce zprávy.

Vytvořil jsem formulář `HistoryForm` určený pro zobrazení historie přijatých a odeslaných zpráv. Historii je možné filtrovat podle odesílatele.

Vytvořené formuláře jsou ilustrovány na obrázku číslo 4.1.



Obrázek 4.1 Diagram tříd návrhu ICQ klienta

4.2 Implementace

Pro implementaci klienta jsem zvolil stejný programovací jazyk jako pro knihovnu (C#).

Implementaci klienta jsem rozdělil na tři části. V první části je popsána implementace konfiguračních souborů aplikace, v druhé části jsou popsány vytvořené třídy tvořící výslednou aplikaci. A v poslední části je popsán způsob implementace instalace klienta na cílové zařízení.

4.2.1 Implementace konfigurace klienta

Pro konfiguraci klienta je použito několika souborů. Prvním a nejdůležitějším konfiguračním souborem je **Settings.xml**. Obsahuje uložená nastavení aplikace a připojení k ICQ serveru. Podle přípony je zřejmé že informace jsou uloženy v XML formátu. Pro spuštění klienta je požadována přítomnost tohoto souboru.

Další použitý soubor je **History.xml**. Oproti souboru s nastavením není nutné, aby tento soubor existoval při spuštění aplikace. Automaticky se vytvoří, jakmile ho bude zapotřebí. Tento soubor slouží k uložení historie přijatých a odeslaných zpráv. Je uložen také ve formátu XML.

Při implementaci manipulace s těmito soubory se vyskytl jeden významný problém. Pracovní adresář, použitý pro přístup k souborům adresovaným relativně, se neshoduje s adresářem, ze kterého byla aplikace spuštěna. V dokumentaci k .Net Compact Framework jsem nenalezl metodu vracející adresář, ze kterého byla aplikace spuštěna. Tento adresář není předáván ani jako spouštěcí parametr aplikace. Proto musí být aplikace instalována v adresáři `Program Files\gicq_klient\` společně s těmito konfiguračními soubory.

4.2.2 Implementace klienta

Protože každý formulář je spuštěn ve vlastním vlákně, je v metodě, reagující na událost, nutné pro manipulaci s prvky okna použít metodu `Invoke(Delegate)`. Tuto metodu zdědí objekt formuláře od třídy `System.Windows.Form`. Protože parametrem této metody je delegát, je zapotřebí vytvořit příslušného delegáta.

4.2.2.1 Implementace Formuláře GICQForm

Pro přístup ke knihovně, implementující komunikační protokol, obsahuje formulář objekt `OSCAR`, který je typu abstraktní třídy `ChatProtokol`. Tento objekt je vytvořen konstruktorem třídy reprezentující komunikační protokol a inicializován. Při změně nastavení programu je zapotřebí tento objekt znovu vytvořit a inicializovat, aby jej bylo možné použít pro nově nastaveného uživatele nebo pro jiný ICQ server.

Seznam kontaktů je implementován vytvořením dvou objektů, uchovávajících informaci o jméně, identifikačním čísle a stavu kontaktu. Oba tyto objekty jsou implementovány jako slovník, kde

v obou objektech je klíčem identifikační číslo kontaktu. Hodnotou pro první objekt je jméno a hodnotou pro druhý objekt je stav. Pro zobrazení seznamu kontaktů je implementována metoda `RefreshBuddylist()`. Protože je seznam obnovován na událost změny stavu kontaktu je k této metodě vytvořen delegát.

Aktuální stav uživatele aplikace je zobrazen v prvku typu ComboBox a reprezentován objektem `StatusComboBox`. Pokud je uživatel ve stavu Offline a vybere nový stav, tak se klient pokusí připojit k ICQ serveru. Tato funkčnost je implementována v reakci na událost `SelectionChanged` objektu `StatusComboBox`.

Nabídka aplikace byla rozdělena na dvě části. V první části jsou přítomny položky pro ovládání aplikace a druhá část obsahuje položky pro ovládání seznamu kontaktů.

Protože formulář obsahuje funkce, které vyžadují nastavení, je vytvořena metoda `LoadSettings()`, která toto nastavení načte z konfiguračního souboru. Nastavení je načteno do objektu `XmlSettings`, který je možné použít k uložení nového nastavení.

V tomto formuláři jsem implementoval tyto reakce na události serveru:

- `OnMessage` – Přijetí zprávy od kontaktu
- `OnRooster` – Změna stavu kontaktu ze seznamu kontaktů

Grafická implementace formuláře je na obrázku 4.2.

Obrázek 4.2. Grafická implementace formuláře GICQForm

4.2.2.2 Implementace Formuláře SettingsForm

Formulář byl navržen pro nastavení funkcí aplikace a spojení s ICQ serverem.

Pro zobrazení aktuálního nastavení jsou vytvořeny příslušné vlastnosti, které je po vytvoření instance této třídy zapotřebí naplnit a zavolat inicializační metodu `Init()`. Uložení nového nastavení není součástí této třídy a musí být provedeno v rodičovském objektu.

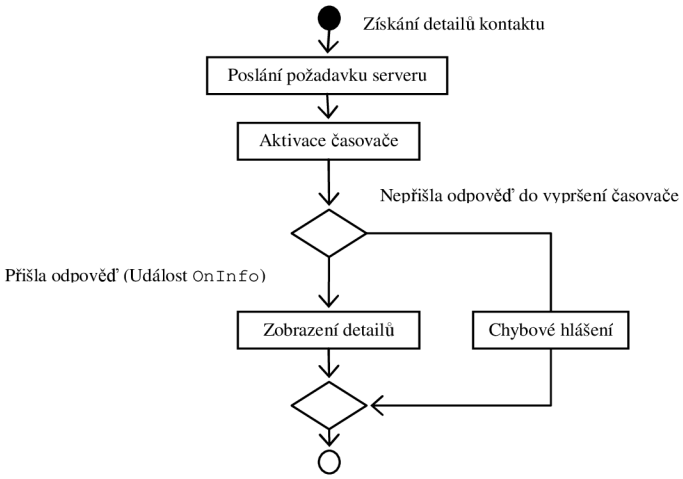
Grafická implementace formuláře je na obrázku 4.3.

Obrázek 4.3 Grafická implementace formuláře SettingsForm

4.2.2.3 Implementace Formuláře DetailsForm

Tento formulář byl navržen pro zobrazení detailních informací o přihlášeném uživateli, nebo kontaktu ze seznamu kontaktů.

Protože tento formulář komunikuje s ICQ serverem je zapotřebí objektu reprezentujícího komunikační protokol. Tento objekt je společně s identifikačním číslem požadovaného kontaktu (UIN) formuláři předán v konstruktoru `DetailsForm(ChatProtocol, string)`. V tomto konstruktoru je implementována veškerá logika pro získání detailních informací o předaném kontaktu, která je vyobrazena na obrázku 4.4.



Obrázek 4.4 Získání detailních informací o kontaktu

4.2.2.4 Implementace Formuláře ChangeDetailsForm

Účelem tohoto formuláře je změna detailních informací o přihlášeném uživateli.

Formulář obsahuje objekt pro komunikaci se serverem, který je předán v konstruktoru `ChangeDetailsForm(ChatProtocol, string)` společně s identifikačním číslem přihlášeného uživatele.

Protože formulář také zobrazuje aktuální informace o uživateli, jsou tyto informace nejprve ze serveru načteny (postup je ilustrován na obrázku 4.4 v kapitole 4.2.2.3).

Uložení nových informací o kontaktu musí být provedeno v rodičovském objektu.

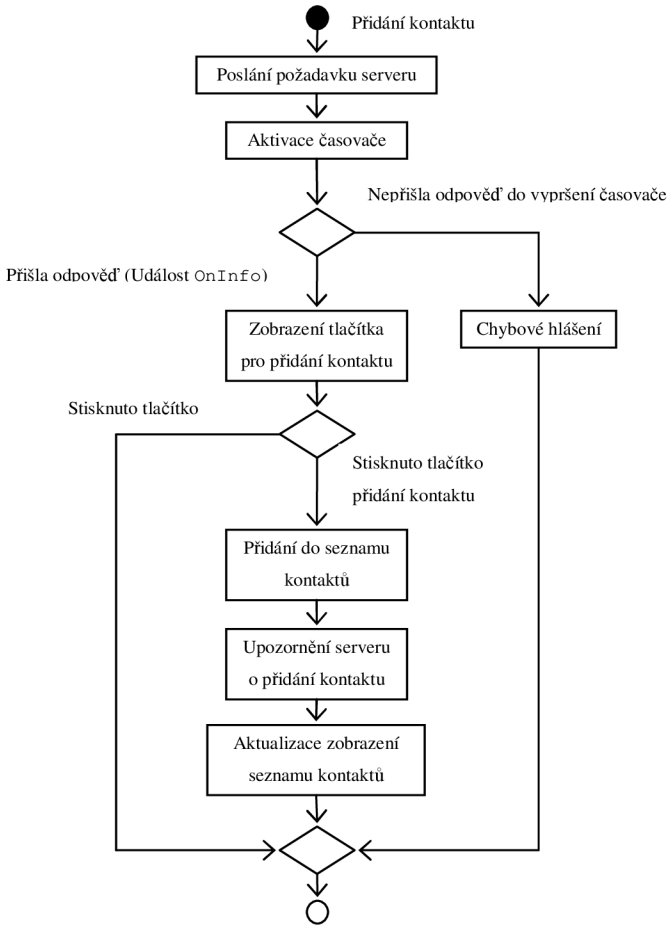
4.2.2.5 Implementace Formuláře AddContactForm

Formulář byl navržen pro vyhledání kontaktu podle identifikačního čísla (UIN) a pro přidání nového kontaktu do seznamu kontaktů.

Protože formulář vyžaduje komunikaci se serverem, musí obsahovat objekt reprezentující komunikační protokol, který je předán v konstruktoru `AddContactForm(ChatProtocol)`.

Pro zadání hledaného identifikačního čísla byl vytvořen ovládací prvek typu TextControl, který je reprezentován objektem `UinTextBox`.

Abychom mohli přidat nový kontakt, musíme jej nejprve vyhledat. Kontakt je vyhledán posláním příslušného požadavku serveru a čekáním na odezvu, pomocí časovače `m_UpdateTimer`. Pokud server odpoví posláním detailů o hledaném kontaktu, je zobrazeno tlačítko umožňující jeho přidání. Po stisknutí tohoto tlačítka je nový kontakt přidán do seznamu kontaktů a informován ICQ server. Tento postup je ilustrován na obrázku 4.6.



Obrázek 4.6 Přidání kontaktu do seznamu kontaktů

4.2.2.6 Implementace Formuláře MessageForm

Obsahuje objekt reprezentující komunikační protokol, protože serveru posílá vytvořenou zprávu. Tento objekt je nutné předat pomocí vlastnosti `Protocol`.

Protože je pro poslání zprávy zapotřebí znát příjemce obsahuje formulář seznam kontaktů. Tento seznam kontaktů je možné nastavit pomocí metody `RefreshReceivers(Dictionary<string, string>)`. Protože metoda může být volána z reakce na událost (změna seznamu kontaktů) je k této metodě vytvořen delegát.

Pro přidání přijaté zprávy je implementována metoda `AddMessage(string, string, byte[])`. Tuto metodu je také zapotřebí volat z reakce na událost (příjem zprávy od kontaktu), proto je pro tuto metodu vytvořen delegát.

Grafický návrh formuláře je na obrázku 4.7.

Obrázek 4.7 Grafická implementace formuláře MessageForm

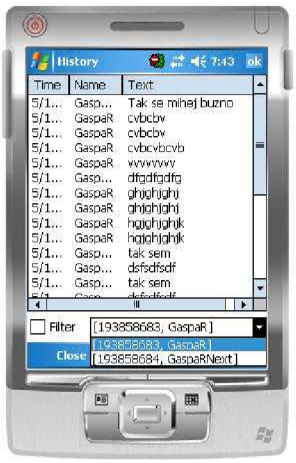
4.2.2.7 Implementace Formuláře HistoryForm

Formulář byl navržen pro zobrazení historie přijatých a poslaných zpráv.

Pro načtení historie ze souboru je implementována metoda `LoadHistory()`, která je volána v konstruktoru nebo při změně filtru zobrazených zpráv. Metoda si nejdříve kontroluje, jestli je přítomen soubor s historií. Pokud není, nic se neděje a seznam historie je prázdný.

Pro aktivování filtru je vytvořen objekt `FilterCheckBox`. Zprávy je možné filtrovat podle odesílatele zprávy, kde výběr je proveden pomocí objektu `ContactsComboBox`.

Grafický návrh formuláře je na obrázku 4.8.



Obrázek 4.8 Grafická implementace formuláře HistoryForm

4.2.3 Implementace instalace klienta

Pro instalaci výsledné aplikace jsem využil možnosti vytvořit soubor typu cabinet (*.cab). Ten po spuštění na cílovém mobilní zařízení zkopíruje vložené soubory na předem určené místo. To zajistí, že aplikace bude na jediném správném místě `Program Files\gicq_klient\`. Pro úspěšný provoz aplikace je už jen nutné mít správně nastavenu internetovou komunikaci.

Požadavky vytvořené aplikace jsou:

- Operační systém Windows Mobile 5.0 a novější
- Nainstalován .Net Framework 3.5
- Připojení k síťi
- 100 kB volného místa

5 Závěr

Zadáním práce bylo vytvořit knihovnu implementující komunikační protokol ICQ a demonstrovat její funkčnost vytvořením ICQ klienta pro PDA. Aplikace a knihovna byly vytvořeny v integrovaném vývojovém prostředí (IDE) Visual Studio pomocí programovacího jazyka Visual C#.

Knihovna implementuje nejpoužívanější funkce ICQ, kterými jsou správa kontaktů, odeslání/příjem zpráv, vyhledání uživatelů a nastavení stavu přihlášeného uživatele. ICQ klient demonstruje jejich použití a navíc implementuje historii přijatých/odeslaných zpráv a signalizaci událostí (zpráva přijata, kontakt přihlášen/odhlášen) při minimalizaci aplikace. Testování klienta proběhlo pouze ve virtuálním prostředí, protože jsem neměl k dispozici žádné PDA. Pro emulaci PDA jsem použil nástroj Device Emulator, který je integrován do vývojového prostředí Visual Studio. Virtuální prostředí má nevýhodu v tom, že neimplementuje všechny vlastnosti reálného systému, ale pro odzkoušení vytvořené aplikace plně postačuje.

Další rozvoj knihovny by se měl zaměřit na implementaci zbývajících funkcí komunikačního protokolu ICQ (OSCAR). Rozvoj ICQ klienta by se měl soustředit na rozšíření práce s historií a seznamu kontaktů. Také by bylo vhodné vytvořit systém pro vkládání zásuvných modulů (tzv. plugin).

Při implementaci jsem využil znalostí získaných z několika absolvovaných předmětů. Z předmětu Programování .Net a C#, jsem využil znalosti o platformě .Net a programovacím jazyku C#. Znalosti z předmětu Tvorba uživatelských rozhraní mě pomohly při návrhu vzhledu a ovládání ICQ klienta. Pro implementaci komunikačního protokolu ICQ jsem využil poznatky získané z předmětu Počítačové komunikace a sítě.

Literatura

[1] CHAPPEL, David. *Understanding.NET*. [s.l.] : Pearson Education Inc., 2002. 327 s. ISBN 0201741628

[2] LUBOSLAV, Lacko. *Programujeme mobilní aplikace : ve Visual Studiu .NET*. Computer Press., 2004. 480 s. ISBN 8025101762

[3] *Windows Mobile. Wikipedia* [online]. Poslední modifikace: 8. Května 2008 [cit. 2008-5-9]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Windows_Mobile>

[4] *Windows Mobile. Microsoft* [online]. Poslední modifikace: 25. Března 2008 [cit. 2008-5-9]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/bb847935.aspx>

[5] *Common Language Runtime. Wikipedia* [online]. Poslední modifikace: 25. Dubna 2008 [cit. 2008-5-9]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Common_language_runtime>

[6] ISAKSSON, Henrik. *ICQ protocol specification* [online]. Poslední modifikace: 28. Prosince 2000 [cit. 2008-5-9]. Dostupný z WWW: <http://iserverd.khstu.ru/docum_ext/icqv5.html>

[7] ISAKSSON, Henrik. *OSCAR (ICQ v7/v8/v9) protocol documentation* [online]. Poslední modifikace: 7. února 2005 [cit. 2008-5-9]. Dostupný z WWW: <http://iserverd.khstu.ru/oscar/>

[8] FRITZLER, A. *AIM/Oscar Protocol Specification* [online]. Poslední modifikace: 8. Dubna 2000 [cit. 2008-5-9]. Dostupný z WWW: <http://www.oilcan.org/oscar/>

[9] RÜDIGER, Kuhlmann. *ICQ/OSCAR protocol (version 7,8,9,10)* [online]. [cit. 2008-5-9]. Dostupný z WWW: <http://www.climm.org/ICQ-OSCAR-Protocol-v7-v8-v9/index.html>

[10] *OSCAR protocol. Wikipedia* [online]. Poslední modifikace: 5. Května 2008 [cit. 2008-5-9]. Dostupný z WWW: <http://en.wikipedia.org/wiki/OSCAR_protocol>

[11] *ICQ. Wikipedia* [online]. Poslední modifikace: 31. Března 2008 [cit. 2008-5-9]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/ICQ>

Seznam příloh

Příloha 1

Návod ICQ klienta

Hlavní okno

Hlavní okno klienta se skládá ze dvou částí. První částí je seznam kontaktů, který obsahuje informace o jejich stavu (Online, Offline, Away, Not Available, Do Not Disturb). Druhá část (dole) obsahuje nastavení stavu uživatele (stejné stavy jak pro kontakty). Přihlášení k ICQ serveru je možné nastavením stavu jiného než Offline, který naopak slouží pro odpojení.

Pro přepnutí aplikace do režimu „na pozadí“ je zapotřebí uzavřít všechna okna klienta křížkem v pravém horním rohu.

Obrázek P1.1. Hlavní okno ICQ klienta

Hlavní nabídka a nabídka kontaktu

Hlavní nabídka obsahuje následující možnosti:

- **View My Details** – Zobrazí detailní informace o přihlášeném uživateli
- **Change My Details** – Změní detailní informace o přihlášeném uživateli
- **History** – Zobrazí historii přijatých a odeslaných zpráv
- **Settings** – Nastavení ICQ spojení a aplikace
- **Exit** – Ukončení aplikace

Nabídka kontaktu obsahuje možnosti (přístupná jen při přihlášení uživatele):

- **Add Contact** – Přidá nový kontakt
- **Remove Contact** – Odebere vybraný kontakt ze seznamu kontaktů
- **View Contact Details** – Zobrazí detailní informace o vybraném kontaktu ze seznamu
- **Send Message** – Otevře okno pro poslání zprávy

Obrázek P1.2 Nabídky aplikace

Nastavení aplikace

Nastavení aplikace je rozděleno na dvě záložky. Na první záložce lze nastavit ICQ účet, na který se bude uživatel přihlašovat. Dalším nastavením na první záložce je ICQ autorizační server (adresa, port, udržování spojení speciálním paketem). Druhá záložka obsahuje nastavení chování aplikace. Význam položek je následující:

- **Logon on startup** – Automatické přihlášení po spuštění aplikace
- **User Online notification** – Upozorňování na přihlášení kontaktu v režimu „na pozadí“
- **User Offline notification** – Upozorňování na odhlášení kontaktu v režimu „na pozadí“

Pro uložení změněných hodnot stiskněte tlačítko OK a pro zrušení změn tlačítko Cancel.

Obrázek P1.3 Nastavení aplikace

Změna detailních informací kontaktu

Po zobrazení dialogu jsou načteny aktuální detailní informace o přihlášeném uživateli. Pro uložení provedených změn stiskněte tlačítko Change a pro zrušení tlačítko Cancel.

Obrázek P1.4 Dialog změny detailních informací uživatele

Přidání kontaktu

Pro přidání kontaktu, nebo jen jeho vyhledání stačí do pole „ICQ UIN to add“ zadat jeho identifikační číslo (UIN). Poté stiskněte tlačítko View uin info. Pokud kontakt existuje, budou zobrazeny jeho detailní informace. Pokud kontakt neexistuje, budete o tom informováni. Jakmile je kontakt nalezen je ho možné přidat do seznamu kontaktů stisknutím tlačítka Add to CL. Pro uzavření dialogu bez přidání kontaktu stiskněte Cancel.

Obrázek P1.5 Přidání kontaktu

Historie zpráv

Položka historie zpráv obsahuje:

- **Time** – Čas přijmutí/odeslání zprávy
- **Name** – Jméno kontaktu
- **Text** – Posílanou zprávu

Pro filtrování zpráv podle kontaktu je nutné zaškrtnou pole „Filter by contact“ a kontakt vybrat ze seznamu kontaktů.

Obrázek P1.6 Historie zpráv

Odeslání/Příjem zprávy

V horní části okna zpráv je možné napsat zprávu pro odeslání. Pro odeslání zprávy je nejprve zapotřebí dole vybrat příjemce a stisknout tlačítko Send.

Všechny příchozí a odchozí zprávy jsou zobrazeny uprostřed okna. Okno je možné zavřít tlačítkem Close.

Obrázek P1.7 Okno zpráv