Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Bakalářská práce

Instant messaging s podporou více témat

2015

David Trachtulec

Vedoucí práce: Mgr. Petr Krajča, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	David Trachtulec
Název práce:	Bakalářská práce (Instant messaging s podporou více témat)
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2015
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Petr Krajča, Ph.D.
Počet stran:	55
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	David Trachtulec
Title:	Bachelor thesis (Multi-topic Instant Messaging)
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2015
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Petr Krajča, Ph.D.
Page count:	55
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

V rámci této práce byla vyvinuta klientská aplikace pro operační systém Microsoft Windows pro textovou komunikaci dvou účastníků v reálném čase, jejíž grafické uživatelské rozhraní umožňuje rozdělení komunikace do různých témat. Aplikace komunikuje pomocí eXtensible Messaging and Presence protokolu (XMPP), který byl rozšířen o podporu komunikace ve více tématech. Komunikace je zpětně kompatibilní s klienty bez podpory témat a v omezené míře jim tématickou komunikaci umožňuje použitím hashtagů.

Synopsis

As a part of this thesis the instant messaging client for Microsoft Windows operating system, that allows splitting graphical user interface into different topics was developed. Application communicates using eXtensible Messaging and Presence Protocol (XMPP), that was extended for multi-topic support. Communication is backward-compatible with clients without topic support and provides limited topic support using hashtags.

Klíčová slova: instant messaging, chat, textová komunikace, hashtag, multi topic, XMPP, agsXMPP, MEF

Keywords: instant messaging, chat, realtime communication, hashtag, multi topic, XMPP, agsXMPP, MEF

Děkuji Mgr. Petru Krajčovi, Ph.D., za vedení práce na vlastní téma, za dobré návrhy k rozšíření funkcionality a za konzultace způsobů řešení. Také bych chtěl poděkovat všem blízkým za trpělivost.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam vět

Seznam zdrojových kódů

1 Úvod

Jednou ze základních lidských schopností je schopnost dorozumívat se a komunikovat s ostatními. V dnešní technologické době tato komunikace velmi často probíhá na dálku. S rozšiřováním dostupnosti síťové infrastruktury, zvyšováním přenosových rychlostí a snižování odezev se již komunikace na dálku stala prakticky neodmyslitelnou součástí našich životů.

Přestože z technického hlediska není problém přenášet v reálném čase zvuk i obraz, textová komunikace stále zůstává významně zastoupena. Nejvýznamnějším zástupcem této kategorie je e-mail. Oproti běžným dopisům má řadu výhod, jako úsporu papíru, jednodušší možnost kategorizace nebo rychlost doručení. Právě rychlost e-mailu je pro formální záležitosti plně dostačující a vyhovující. V méně formálním prostředí, kde je rozsah jednotlivých zpráv většinou malý, se však mohou e-maily stát zdlouhavými.

Zde přichází na řadu kategorie instant messengerů umožňujících v přehledné formě komunikovat v reálném čase. Existuje mnoho různých protokolů a klientských aplikací, které také většinou umožňují nastavení stavu, podle kterého lze na první pohled zjistit, zda protistrana je nebo není u počítače (či jiného zařízení), případně jen na chvíli odešla. Některé umožňují souběžnou komunikaci celých skupin uživatelů, přenosy souborů nebo dokonce hraní her.

Při psaném dialogu však může docházet k prolínání diskutovaných témat. Pokud například jeden účastník chvíli neodpovídá, druhého napadne otázka, která přímo nesouvisí s předchozím tématem, a odešle ji do stejného komunikačního okna. Pokud se takhle sejdou dvě podobné otázky za sebou, může být následná odpověď těžko přiřaditelná ke správně otázce. V horším případě může dojít i k naprosto odlišnému pochopení celého dialogu. Nepříjemným příkladem by mohla být následující hypotetická situace:

Ž: „Díval ses po těch klíčích od garáže v posledním šuplíku?“
Ž: „Už jsi byl pro děti ve školce?“
M: „Jo, ale nebyly tam.“

Na internetu se nepodařilo nalézt žádný nástroj, který by se snažil tuto problematiku řešit. Částečným řešením by mohlo být použití protokolu umožňujícího diskuzi ve více uživatelích a zakládání vlastních místností, do kterých by se poté připojili pouze příslušní dva uživatelé. Toto řešení by však nebylo příliš vhodné, protože by se do těchto místností mohli připojovat i cizí uživatelé. Tomu by se dalo zabránit nastavením hesla pro přístup do místnosti, přesto by však nešlo o zcela korektní použití dané funkcionality a mohlo by znepříjemňovat procházení místností ostatním uživatelům. Dalším problémem by bylo uživatelské rozhraní. Každou místnost by pravděpodobně reprezentovalo jedno okno, čímž by se při větším počtu místností stávala konverzace stále méně přehledná a výsledek by dost možná byl ještě horší, než prolínání různých témat v jednom komunikačním okně.

Právě absence takového nástroje se stala motivací pro vytvoření této práce na vlastní téma. Původní myšlenka počítala s vytvořením klientské aplikace a vlastního komunikačního protokolu. Po první konzultaci s vedoucím práce však bylo rozhodnuto, že bude vhodnější použít rozšiřitelný protokol XMPP.

2 eXtensible Messaging and Presence Protocol

Tato kapitola popisuje základní principy a fungování protokolu XMPP. Dále vysvětluje, jakým způsobem byl rozšířen a využit pro vytvoření komunikační základny pro přenos a identifikaci tématických zpráv. Ke studiu funkcionality protokolu XMPP byla použita kniha [1].

2.1 Popis protokolu

Jak již z názvu vyplývá, je protokol XMPP přímo koncipován tak, aby jej bylo možné rozšířit o novou funkcionalitu, která může být následně použita pro širokou škálu nových aplikací. Jedná se o otevřený protokol, nezatížený licenčními poplatky, a je tedy vhodný pro nekomerční použití. Licence umožňuje i jeho komerční použití, což může být díky nulovým počátečním investicím výhodou pro začínající projekty. Vznikl v roce 1999, byl vyvíjen open-source komunitou pod svým původním názvem Jabber a tvořil stejnojmennou komunikační síť pro instant messaging. Svou koncepcí a rozšiřitelností ale umožňoval nejen běžnou textovou komunikaci, ale i přenos příkazů a stavových zpráv mezi stroji nebo roboty. V roce 2004 byl popsán v RFC 6120 [2] a RFC 6121 [3] pod názvem eXtensible Messaging and Presence Protocol (XMPP).

Síťová architektura protokolu je typu klient-server. Serverů je celá řada a díky otevřenosti protokolu může vytvořit nový server prakticky kdokoliv. Tato síť má tedy decentralizovaný charakter a výpadek jednoho serveru nemá na funkčnost sítě, jako celku, vliv. Každý server má alespoň jednu svou doménu, např. jabber.org nebo jabbim.cz. Jedinečný identifikátor uživatele zvaný Bare JID[1] je potom tvořen volitelným jménem při registraci, zavináčem a právě doménou serveru — tedy například *jan.novak@jabbim.cz*. Přidáním dodatečného zdroje (*tzv. resource*), který odlišuje různá zařízení nebo klientské aplikace při souběžném připojení pod jedním Bare JID z různých míst, vznikne Full JID — například *jan.novak@jabbim.cz/Multi-topic-IM*. Ke vzájemné identifikaci serverů slouží DNS[2].

Veškerá komunikace protokolu XMPP je reprezentována XML[3] uzly. Použití XML poskytuje dobrou čitelnost komunikace a tím pádem i poměrně jednoduchou možnost rozšíření přidáním nových elementů mezi potomky uzlů komunikačních primitiv, které jsou popsány níže. Připojení klienta k serveru spočívá ve vytvoření TCP[4] spojení a vytvoření XML streamu na každé ze stran. Toto spojení zůstává otevřené po celou dobu připojení a komunikace mezi serverem a klientem probíhá zápisem do XML streamu protějšku.

[1] Akronym Jabber Identifier podle původního názvu protokolu

[2] Zkratka Domain Name System - hierarchický sytém pro překlad doménových jmen na IP adresy a naopak

[3] Zkratka eXtensible Markup Language - značkovací jazyk pro reprezentaci různých typů dat

[4] Zkratka Trasmission Control Protocol - spojovaný protokol transportní síťové vrstvy

2.2 Komunikační primitiva

Základní prvky tvořící většinu XML komunikace na protokolu XMPP jsou tři a liší se stavbou, použitím a tím, jestli protistrana očekává odpověď. V angličtině se tyto prvky nazývají „stanza“ a protože neexistuje, nebo není zaužívaný, český ekvivalent, budeme pro ně používat stejné, počeštěné slovo.

Tyto tři stanzy jsou následující:

1. Presence
2. Message
3. IQ

Další prvky, které zajišťují sjednání XML spojení, nastavení zabezpečení nebo další nízkoúrovňové funkce, nejsou pro účely této práce důležité a jejich definice se dají dohledat ve zmiňovaných RFC.

Základní definice protokolu je dále rozšiřována rozšířeními XEP[5] zaváděnými organizací XSF[6].

2.2.1 Presence

Stanza typu *presence* slouží k informování jednotlivých entit o stavu ostatních. Poskytuje informaci, zda je protistrana připojena k serveru a připravena komunikovat. Na odeslanou stanzu presence není od protistrany očekávána žádná odpověď. V případě více připojených zařízení pod jedním JID, volitelně poskytuje informaci o tom, která jsou preferovanější na základě číselné *priority* v rozsahu -127–128 obsažené v této stanze. Příklad presence stanzy je uveden ve zdrojovém kódu 1.

```
1 <presence from="jan.novak@jabbim.cz/Multi-topic-IM">
2   <show>chat</show>
3   <status>I'm using Multi-topic IM! :)</status>
4   <priority>10</priority>
5 </presence>
```

Zdrojový kód 1: Presence stanza - příklad

V atributu *from* je uveden JID odesílajícího uživatele.

Presence stanza může obsahovat atribut *type* využívaný pro určení stavu dostupnosti a pro zpracování autorizací (tzv. *subscription request*) pro příjem stavových *presence* stanz od daného kontaktu.

[5] Zkratka XMPP Extension Protocol - označuje standardizované rozšíření protokolu XMPP

[6] Zkratka XMPP Standards Foundation - nadace zodpovědná za standardizaci XEP rozšíření protokolu XMPP

- **available** — indikuje dostupnost uživatele
- **unavailable** — uživatel se odpojil a je nedostupný
- **invisible** — uživatel chce přijímat *presence* od ostatních, ale své jim posílat nechce (chce být „neviditelný“) — o tom instruuje server touto hodnotou
- **subscribe** — žádost o autorizaci
- **subscribed** — potvrzení autorizace
- **unsubscribe** — odvolání autorizace
- **unsubscribed** — potvrzení odvolané autorizace
- **probe** — vyžádání *presence*; tuto hodnotu by měl využívat pouze server
- **error** — indikuje chybu

Pokud není atribut *type* v presence stanze uveden, chápe se uživatel jako dostupný — available.
Příklad žádosti o autorizaci je uveden ve zdrojovém kódu 2.

```
1 <presence from="jan.novak@jabbim.cz/Multi-topic-IM"
2           to="petr.novak@jabbim.com"
3           type="subscribe"/>
```

Zdrojový kód 2: Presence stanza - příklad žádosti o autorizaci

Hlavně v instant messagingu[7] se používá atribut *show*, který určuje bližší stav uživatele. *Show* může nabývat následujících hodnot:

- **away** — Uživatel se na krátkou dobu vzdálil od počítače.
- **xa**[8] — Uživatel bude delší dobu jinde, ale zůstává připojen.
- **dnd**[9] — Uživatel si nepřeje být rušen. Tento stav může být využit například i při komunikaci strojů a indikovat, že je stroj v tuto chvíli zaneprázdněný a nepřijímá další příkazy.
- **chat** — Uživatel má zájem se zapojit do konverzace.

Dalším atributem je *status* — textová zpráva blíže popisující stav uživatele.

[7] Komunikace v reálném čase
[8] xa - zkratka eXtended Away
[9] dnd - zkratka Do Not Disturb

2.2.2 Message

Pro přenos zpráv je využívána stanza *message*. Tato stanza také nepožaduje potvrzení jejího přijetí protistranou. Pokud je však vyžadována vyšší spolehlivost s potvrzováním přijetí zpráv, je možné pro tento účel zavést rozšíření XEP-0184[10].

Ve zdrojovém kódu 3 je uveden příklad *message* stanzy. Atribut *from* obsahuje JID odesílatele zprávy, který do stanzy doplňuje server, aby nemohlo dojít k odeslání zprávy s falešným odesílatelem (tzv. *spoofing*). Atribut *to* určuje JID příjemce a *type* typ přenášené zprávy. V samotném těle stanzy se nachází text zprávy volitelně obsažený v elementu *body*.

```
1 <message from="jan.novak@jabbim.cz/Multi-topic-IM"
2          to="petr.novak@jabbim.com">
3          type="chat">
4   <body>Ahoj. Jak se daří?</body>
5 </message>
```

Zdrojový kód 3: Message stanza - příklad

2.2.3 IQ

Zkratka stanzy *IQ* znamená Info/Query. Jedná se tedy o prostředek, jak zjišťovat informace od protistrany a zasílat nastavující požadavky. Proto je u ní, na rozdíl od dvou předchozích stanz, vždy požadována odpověď, i kdyby jen oznamovala chybu zpracování.

Typ stanzy *IQ* určuje atribut *type*, nabývající následujících hodnot:

- **get** — požadavek o informaci
- **set** — požadavek o nastavení
- **result** — potvrzující informace o zpracování požadavku
- **error** — chyba při zpracování požadavku

Pro identifikaci *IQ* stanz se využívá atribut *id* generovaný klientskou aplikací.

[10]Specifikace rozšíření XEP-0184 - http://xmpp.org/extensions/xep-0184.html

2.3 Rozšíření protokolu o podporu více témat

Pro rozšíření protokolu o podporu přenosu a identifikaci tématických zpráv bylo potřeba navrhnout prvky obsahující identifikátor přesně určující konverzační téma, ke kterému se komunikace vztahuje. K tomu byl pro svou jedinečnost a dobrou podporu ze strany platformy .NET zvolen GUID[11]. V rozšíření je reprezentován XML elementem *topicGuid*. Pro rozlišitelnost témat uživatelem je potřeba textový název tématu obsažený v elementu *topicName*. Pro zjednodušení výběru témat potom hashtag[12] v elementu *hashtag*.

2.3.1 Rošíření message stanzy

Každý z elementů *topicGuid*, *topicName* a *hashtag* je vložen do elementu *topic*, který zastřešuje všechny potřebné informace tématické zprávy. Element *topic* obsahuje navíc atribut *xmlns* — jmenný prostor (tzv. *namespace*), který identifikuje použité rozšíření v rámci *message* stanzy. Jmenným prostorem navrženého rozšíření je řetězec "trachtulec:extension:multi-topic". Tvar rozšířené *message* stanzy je uveden ve zdrojovém kódu 4.

```
1 <message to="petr.novak@jabbim.com/Multi-topic-IM" type="chat">
2   <body>Našel jsem nový klient běžící na XMPP s podporou více témat
        a hashtagů!</body>
3   <topic xmlns="trachtulec:extension:multi-topic">
4     <topicGuid>cb0eb237-f862-4b0a-9f12-ace21dbe8047</topicGuid>
5     <topicName>Šikovné programy</topicName>
6     <hashtag>zajimaveaplikace</hashtag>
7   </topic>
8 </message>
```

Zdrojový kód 4: Message stanza rozšířená o podporu témat

2.3.2 Změna vlastností tématu

Klientská aplikace podporuje také změnu názvu zprávy i hashtagu. Pro informování protistrany o této změně je vhodná stanza *IQ*. Tato stanza musí být typu *set*, protože se jedná o nastavující požadavek. Její potomek *query* má opět, obdobně jako rozšíření v *message* stanze, nastaven jmenný prostor identifikující rozšíření a k tomu navíc typ požadavku. V tomto případě se jedná o jmenný prostor "trachtulec:extension:multi-topic:topic-info-iq". Data o změnách jsou obsažena v elementech *topicGuid*, *topicName* a *hashtag*. Tvar rozšířené *IQ* stanzy je uveden ve zdrojovém kódu 5.

[11] Akronym Global Unique Identifier - jedinečný 128-bitový identifikátor běžně zobrazovaný jako sekvence 32 číslic hexadecimální soustavy generovaný pseudonáhodným algoritmem

[12] Jednoduchý identifikátor složený ze znaku '#' následovaného textem bez mezer, například "#pocasi". Jde o běžně používaný způsob označování témat na sociálních sítích jako Facebook nebo Twitter.

```
<iq type="set" from="petr.novak@jabbim.com/Multi-topic-IM" to="jan.
    novak@jabbim.cz/Multi-topic" id="111218420">
  <query xmlns="trachtulec:extension:multi-topic:topic-info-iq">
     <topicGuid>cb0eb237-f862-4b0a-9f12-ace21dbe8047</topicGuid>
     <topicName>Šikovný SW</topicName>
     <hashtag>zajimaveapps</hashtag>
  </query>
</iq>
```

Zdrojový kód 5: Rozšířená IQ stanza pro úpravu tématu

2.3.3 Indikace podpory tématické komunikace

Aby měla protistrana informaci o tom, že uživatel používá klientskou aplikaci podporující komunikaci ve více tématech, byla rozšířena *presence* stanza o element *MultiTopicSupport* ve jmenném prostoru "trachtulec:extension:multi-topic". V těle tohoto elementu je obsažena hodnota '1' reprezentující logickou pravdu. Díky tomu má uživatel vždy aktuální informaci, zda protistrana podporuje tématickou komunikaci. Tvar rozšířené stanzy *presence* je uveden ve zdrojovém kódu 6.

```
<presence from="jan.novak@jabbim.cz/Multi-topic-IM">
  <show>chat</show>
  <status>I'm using Multi-topic IM! :)</status>
  <priority>10</priority>
  <MultiTopicSupport xmlns="trachtulec:extension:multi-topic">
    1
  </MultiTopicSupport>
</presence>
```

Zdrojový kód 6: Presence stanza s příznakem podpory tématické komunikace

2.3.4 Řešení konfliktu identifikátoru tématu a hashtagu

Při komunikaci ve více tématech může dojít k tomu, že mezi sebou nekorespondují hashtag a GUID identifikátor tématu odesílajícího a přijímajícího uživatele. K tomuto může například dojít, pokud jeden z uživatelů nainstaluje klientskou aplikaci na nový počítač poté, co ji již používal na starém. Ví tedy, že již pro komunikaci používal určitý hashtag. Pokud v nové instalaci založí nové téma s tímto hashtagem, vygeneruje se nový identifikátor, který ale bude jiný než má pro stejný hashtag v databázi uložena protistrana.

Pokud k této situaci dojde, je potřeba zajistit sladění kolidujících identifikátorů. Klientská aplikace uživatele, která konflikt detekovala, zjistí ze své databáze datum vytvoření tématu a s oběma GUID identifikátory jej odešle protistraně v *IQ* stanze typu *set*. Element *query* této stanzy má jmenný prostor

"trachtulec:extension:multi-topic:hashtag-guid-sync". Jak tato stanza vypadá můžeme vidět ve zdrojovém kódu 7.

```
<iq to="jan.novak@jabbim.cz" type="set" id="678641510">
  <query xmlns="trachtulec:extension:multi-topic:hashtag-guid-sync"
      force="false">
    <Hashtag>vylet</Hashtag>
    <ReceivedGuid>47445922-b5ad-4b25-9b96-a84dd27f70f9</ReceivedGuid>
    <Timestamp>2015-04-08T08:08:28</Timestamp>
    <RequestedGuid>
      48445922-b5ad-4b25-9b96-a84dd27f70f9
    </RequestedGuid>
  </query>
</iq>
```

Zdrojový kód 7: Požadavek na synchronizaci identifikátoru tématu

Při přijetí tohoto požadavku zkontroluje klientská aplikace ve své databázi datum vytvoření tématu, jehož identifikátor koliduje s protistranou. Pokud je datum tématu novější, než datum z požadavku, klient použije identifikátor a datum protistrany k nahrazení svých hodnot a tím je konflikt vyřešen. V opačném případě vytvoří obdobný požadavek, s vyměněnými identifikátory a svým (starším) datem tématu. Tomuto požadavku nastaví atribut *force* na *true* a odešle jej. Příjemce podle tohoto atributu pozná, že k porovnání dat již došlo a má své hodnoty nahradit přijatými. Příjemce požadavku na něj musí vždy odpovědět, což je dáno specifikací zpracování *IQ* stanz. V tomto případě jde o stanzu typu *result* s prázdným elementem *query*, který má jmenný prostor totožný s prostorem požadavku.

Další možný scénář vzniku konfliktu identifikátoru tématu a hashtagu:

1. Uživatel A a B komunikují v tématu T_1 s identifikátorem tématu G_1 a hashtagem #vylet
2. Uživatel B se odpojí od sítě
3. Uživatel A změní hashtag tématu T_1 na #exkurze
4. Uživatel A se odpojí od sítě
5. Oba uživatelé se připojí do sítě
6. Uživatel A zapomněl, že změnil hashtag tématu T_1 a vytvoří nové téma T_2 s nově vygenerovaným identifikátorem G_2 s hashtagem #vylet, ze kterého odešle uživateli B zprávu
7. Klient uživatele B zjistí konflikt — v jeho databázi již existuje hashtag #vylet, ale s odlišným GUID

V tomto případě by však ve výše popsané proceduře řešení konfliktu došlo k problému s tím, že předchozí identifikátor již v databázi uživatele A existuje. Řešením je několik následujících kroků:

1. Odešle se *IQ* stanza pro změnu hashtagu z #vylet na #exkurze uživateli B
2. Historie tématu T_2 uživatele A se v jeho databázi přesune do tématu T_1
3. Z databáze uživatele A se odstraní hashtag #vylet
4. Z databáze uživatele A se odstraní téma s identifikátorem G_2
5. Z databáze uživatele A se odstraní identifikátor G_2

Tímto postupem dojde k vyřešení konfliktu identifikátorů při stejném hashtagu a existenci obou identifikátorů v databázi.

2.3.5 Zpětná kompatibilita s podporou hashtagů

Message stanza byla rozšířena přidáním elementu *topic* mezi její potomky. Celková struktura stanzy i jejích dalších potomků zůstala zachována. Při přenosu stanzy se server o tento nový prvek nestará a na základě atributu *to* ji pošle určenému adresátovi. Pokud adresát používá klientskou aplikaci bez podpory témat, přidaný element také nevadí. Klient ze stanzy použije jen informace, kterým rozumí a ostatní elementy ignoruje. Text zprávy je obsažen mimo rozšiřující element *topic*, a proto jsou tématické zprávy poslané netématickému klientovi v pořádku přijaty, jako by se jednalo o běžnou zprávu. Zpětná kompatibilita je tedy automaticky zajištěna XML charakterem protokolu XMPP.

Na základě popsaného chování je jasné, že v této situaci opět vzniká na straně netématického klienta problém prolínajících se témat. Tématický klient má však informace o tom, který ze zdrojů kontaktu podporuje tématický chat a který ne. Na začátek těla zprávy odeslané netématickému klientovi, tedy přidá hashtag příslušící danému tématu. Tělo zprávy potom může mít například tvar "#vylet: Kdy budeme vyjíždět?". Podle toho uživatel pozná, ke kterému tématu se zpráva vztahuje. Když chce odpovědět, použije stejný postup — na začátek zprávy přidá hashtag cílového tématu. Odpověď tedy bude mít tvar "#vylet: Hned ráno, dokud nebude horko". Tato zpráva potom tématickému klientovi přijde do tématu s hashtagem #vylet. Pokud téma s tímto hashtagem neexistuje, je přijetím takovéto zprávy vytvořeno. Zprávy, které hashtag neobsahují, přijdou do obecného tématu „General“. Toto téma existuje v tématickém klientovi automaticky a může také sloužit pro konverzaci, pro kterou nemá význam zakládat samostatné téma.

3 Postup řešení

Klientská aplikace umožňující vedení tématického chatu byla vyvinuta pod GNU GPL[13] licencí pro operační systém Microsoft Windows 7 a novější. Dostala jméno Multi-topic IM[14] (také MTIM) a v následujících podkapitolách je popsáno, jakými postupy probíhal její vývoj.

3.1 Koncepce aplikace

Aplikace byla navržena jako kompozitní, tedy složená z více prvků. Hlavním prvkem je jádro, které poskytuje okno seznamu kontaktů, spravuje nastavení, pracuje s databází, stará se o komunikaci s XMPP serverem a také zajišťuje propojení s dalšími prvky — pluginy. Druhý důležitý prvek této aplikace, kterým je komunikační okno umožňující přehledné rozdělení komunikace do témat, je tedy tvořen pluginem. Tento přístup do budoucna poskytuje jednoduchou možnost jeho nahrazení jinak řešeným uživatelským rozhraním. Aplikace může být rozšířena i různými dalšími pluginy vyvinutými podle určitých pravidel definovaných ve zdrojovém kódu projektu PluginInterface. Popis tohoto projektu se nachází na straně 48.

3.2 Vývojové prostředí

Pro vývoj aplikace bylo použito vývojové prostředí Microsoft Visual Studio 2013 Ultimate získané díky akademické licenci ze služby Microsoft DreamSpark. Visual Studio je funkčně bohaté prostředí pro vývoj aplikací nejen pro platformu .NET, která byla využita pro vývoj této aplikace. Poskytuje dobrou podporu ladění a to i ve formě zpětné analýzy běhu aplikace pomocí nástroje IntelliTrace.

3.3 Použité technologie

Při vývoji byly použity následující technologie:

- Microsoft .NET C# 4.0
- Windows Forms
- agsXMPP
- Managed Extensibility Framework (MEF)
- EventAggregator
- NAudio

[13] GNU General Public License - softwarová licence volně poskytující zdrojové kódy, při jejichž využití však požaduje dodržení stejné licence

[14] Zkratka Instant Messenger - klientská aplikace pro komunikaci v reálném čase

3.3.1 Microsoft .NET C# 4.0

K vývoji klientské aplikace byl využit programovací jazyk C# vyvíjený společností Microsoft. Jedná se o vysokoúrovňový objektově orientovaný programovací jazyk pracující na platformě .NET od stejné společnosti. Jeho vývoj probíhá již od roku 2002, kdy byla vydána jeho první verze. Zdrojový kod C# se kompiluje do jazyka CIL[15], který je při spuštění interpretován běhovým prostředím CLR[16].

3.3.2 Windows Forms

Pro tvorbu grafického uživatelského rozhraní byl využit soubor knihoven Windows Forms od společnosti Microsoft. Windows Forms poskytují programátorsky přívětivý způsob vytváření událostmi řízených uživatelských rozhraní a poskytuje řadu předpřipravených grafických komponent, díky kterým není třeba od základu vytvářet běžně používané prvky uživatelského rozhraní. Toto také zajišťuje podobnost uživatelských prostředí napříč různými aplikacemi, což zvyšuje uživatelský komfort při jejich ovládání. Vývojové prostředí Visual Studio umožňuje jednoduchý návrh těchto grafických uživatelských rozhraní pomocí WYSIWYG[17] designeru. Dnes již Windows Forms postupně nahrazuje modernější grafický systém WPF[18].

3.3.3 agsXMPP SDK

K realizaci komunikace pomocí protokolu XMPP a k jeho rozšíření o podporu témat byl využit SDK[19] agsXMPP. Jedná se o balíček vyvíjený v jazyce C# pod duální licencí. V komerčních aplikacích je zpoplatněn, v otevřených projektech je dostupný pod svobodnou GPL licencí. Licence vyvinuté aplikace je proto také GPL. Jeho autorem je společnost AG-Software. Pro tuto práci byl vybrán pro dobrou kompatibilitu s platformou .NET danou vývojem na stejné platformě. Do projektu jej lze jednoduše zařadit a jeho syntaxe je dobře pochopitelná. Na druhou stranu je jistou nevýhodou absence detailnější dokumentace, což často vyžadovalo procházení zdrojových kódů poskytnutých příkladů, s cílem zjistit přesný způsob použití funkcí a tříd.

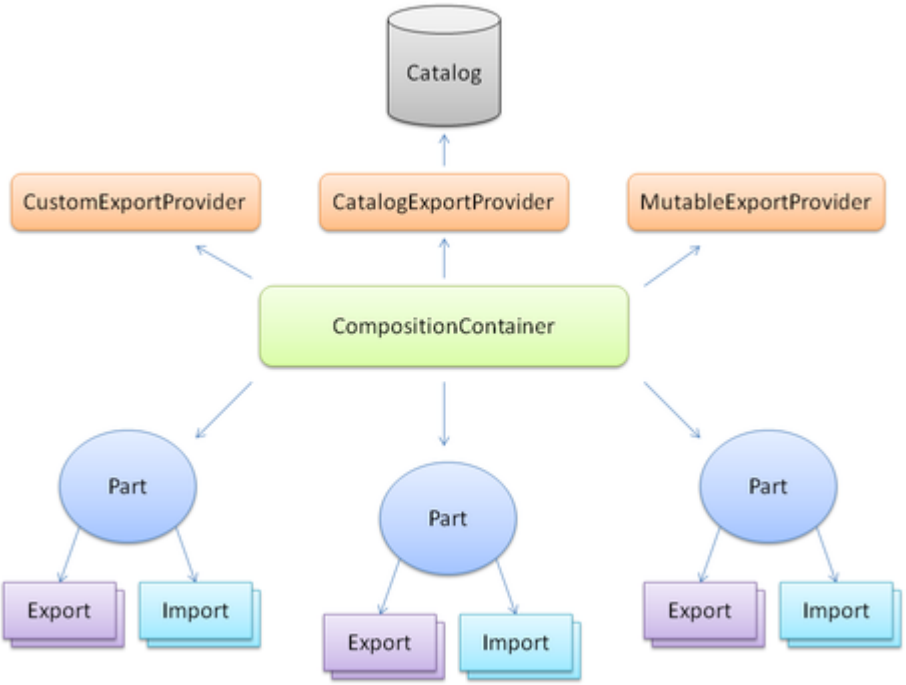
[15] Zkratka Common Intermediate Language - nízkoúrovňový programovací jazyk platformy .NET nezávislý na hardwarové platformě

[16] Zkratka Common Language Runtime - běhové prostředí (někdy nazývané virtuální stroj) tvořící spojovací prvek mezi CIL a nativním kódem

[17] Akronym What You See Is What Yout Get - způsob návrhu grafických prvků, který již při vývoji umožňuje jejich vytváření, zobrazení a úpravu jejich rozložení tak, jak budou vypadat po nasazení aplikace

[18] Zkratka Windows Presentation Foundation - grafický subsystém platformy .NET oddělující funkcionalitu a návrh rozložení uživatelského rozhraní

[19] Zkratka Software Development Kit - balíček nástrojů poskytujících určitou předpřipravenou funkcionalitu, která zjednodušuje práci s cílovou platformou nebo protokolem



Obrázek 1: Diagram vztahů prvků MEF [4]

3.3.4 Managed Extensibility Framework (MEF)

V průběhu vývoje frameworku .NET byla k dispozici řada postupů, jak do kódu zavést podporu pluginů. Většinou se ale jednalo o poměrně složitá řešení, která vyžadovala použití velkého množství kódu. Od čtvrté verze frameworku je jeho součástí knihovna MEF[20], která se z předchozích postupů vyvinula v programátorsky přívětivou, kódově nenáročnou metodu zavádění pluginů. Na rozdíl od předchozích řešení poskytují pluginy vyvinuté pro MEF dobrou znovupoužitelnost v jiných aplikacích, než pro které byly původně vyvinuty.

Princip zavádění pluginu spočívá v procesu *kompozice* (*composition*), která probíhá při běhu aplikace prohledáním určených adresářů na existenci knihoven vhodných pro jejich připojení. Ty jsou uloženy do *katalogu* (*catalog*). Typ knihovny/pluginu je zjištěn pouze na základě metadat, která určují *interface* (*rozhraní*), ze kterého je plugin odvozen. Tento typ určuje, zda je plugin pro *kompozici* vhodný. K vývoji pluginu je potřeba tento *interface* znát, a proto musí cílová aplikace poskytovat jeho specifikaci (například ve formě zdrojového kódu).

V kódu rodičovské aplikace se direktivou `[Import(typeof(Interface))]`, zvanou *import*, označí místa, ve kterých má dojít k připojení. K získání kolekce vhodných pluginů lze také použít direktivu `[ImportMany]`, kde žádaný typ

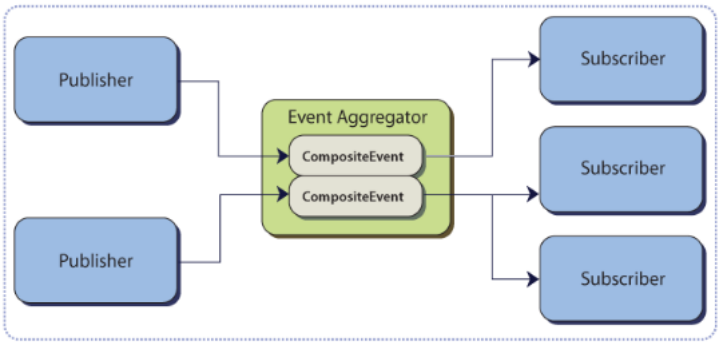
[20]Knihovna MEF - http://mef.codeplex.com/wikipage?title=Overview&referringTitle=Home

určí *interface* cílové kolekce — `IEnumerable<Interface> Plugins`. V kódu pluginu se místa propojení označí direktivou `[Export(typeof(Interface))]`, tzv. *export*. Tyto direktivy tedy určují, která část kódu pluginu se má připojit na které místo v rodičovské aplikaci. Aplikaci i pluginu obsahující *importy*/*exporty* se říká *část* (*part*). MEF umožňuje i opačný směr propojení, při kterém se část rodičovské aplikace připojí do pluginu. Vytvořená propojení jsou uchovávána v tzv. *kontejneru* (*container*). Vztahy prvků MEF můžeme vidět na obrázku 1.

3.3.5 EventAggregator

Pluginy musí s jádrem aplikace vzájemně komunikovat. Použití běžných událostí by však bylo komplikované, a proto bylo potřeba najít jiné řešení. Pro WPF je vyvíjen souhrn postupů a tříd jménem Prism[21], které podporují vývoj aplikací s kompozitní architekturou. Tato architektura je navržena tak, že je aplikace jako celek složena z dílčích částí, které lze jednoduše nahrazovat například za účelem přidání nové nebo jiné funkcionality bez ohledu na jiné funkční bloky. Jeden z autorů Prism se rozhodl upravit některé třídy tak, aby je bylo možné použít také ve Windows Forms [6]. Mezi zpřístupněné třídy patří EventAggregator a CompositeEvent.

EventAggregator[22] funguje na principu publish/subscribe, kde figuruje jako prostředník pro předávání kompozitních událostí základního typu CompositeEvent. Každá z těchto kompozitních událostí může mít několik odesílajících (*publisher*) i odebírajících (*subscriber*) zdrojů. Na obrázku 2 je tento koncept znázorněn. Pro tyto vlastnosti byl EventAggregator zvolen pro realizaci komunikace mezi jádrem aplikace a pluginy.



Obrázek 2: EventAggregator - Diagram kompozitních událostí [5]

[21] Souhrn postupů a tříd pro vývoj kompozitních aplikací - https://msdn.microsoft.com/en-us/library/ff648465.aspx

[22] Třída EventAggregator - https://msdn.microsoft.com/en-us/library/ff921122.aspx

3.3.6 NAudio

Tato open-source knihovna pro platformu .NET poskytuje řadu funkcí pro převod a přehrávání zvukových souborů. Je vyvíjena již od roku 2002 a její funkcionalita je dále rozšiřována. Pro přehrávání zvuku využívá nízkoúrovňových volání funkcí Windows. V tomto projektu je využita pouze pro přehrávání zvuku indikujícího příchod zprávy. Byla zvolena pro lepší podporu formátů zvukových souborů, oproti řešením poskytovaným platformou .NET. Jedním z těchto řešení je třída SoundPlayer, která však podporuje pouze formát WAV. Další řešením by mohla být COM komponenta aplikace Windows Media Player, která je však pro tento účel zbytečně náročná.

3.4 Způsob testování

Aplikace byla testována na několika reálných i virtuálních systémech. Reálné testované systémy byly Windows 7 Professional 64-bit, Windows 8 Pro 64-bit a Windows 8.1 Pro 64-bit.Virtuální systémy potom Windows 7 Professional 64-bit a Windows 7 Enterprise 32-bit. Podmínkou bezproblémového běhu byly ve všech případech balíčky Microsoft .NET Framework 4.0 a Microsoft SQL Server Compact 4.0 SP1. Komunikace byla úspěšně testována na serveru jabbim.cz, jabber.org a lokálně spuštěném XMPP serveru ejabberd.

3.5 Instalátor

Instalátor byl vytvořen bezplatným nástrojem Inno Setup. Tento nástroj nabízí jednoduchého průvodce, který po vybrání potřebných souborů automaticky vytvoří instalátor aplikace. Po nainstalování aplikace je k dispozici i odinstalátor.

4 Vnitřní struktura aplikace

Struktura aplikace je tvořena následujícími projekty:

- **CustomForms** — pomocné dialogy
- **MT-Shared** — sdílené třídy
- **Multi-topic IM** — jádro klientské aplikace
- **PluginInterface** — rozhraní zprostředkující připojení pluginů
- **MessageWindowLib** — plugin komunikačního okna

4.1 Řešení GUI

Všechna okna aplikace tvoří komponenta Form knihovny Windows Forms. Z této knihovny pochází i ostatní použité grafické komponenty, některé však byly funkčně a/nebo graficky upraveny, aby vyhovovaly požadovanému použití.

4.1.1 Zobrazení kontaktů

Pro zobrazení kontaktů je použita upravená komponenta TreeView určená k práci se stromovou strukturou. Nová komponenta se jmenuje RosterControl a má předefinovanou metodu *DrawNode* zodpovědnou za vykreslování jednotlivých uzlů stromu. Jsou vykreslovány oddělující čáry, pozadí uzlů a ikony stavu. Pozadí uzlu se vykresluje odlišnou barvou, pokud je nad uzlem kurzor myši nebo je uzel vybrán. RosterControl obsahuje dva uzly rozdělující připojené a odpojené kontakty. Kontakty se potom do seznamu přidávají jako jejich potomci.

4.1.2 Výběr stavu a tlačítko hlavního menu

Oba tyto prvky jsou realizovány komponentou tlačítko s drobnou úpravou, díky které se při jeho označení nevykresluje vnitřní obdélník, který označení indikuje. Tento obdélník vypadal v grafickém prostředí rušivě.

4.1.3 Okno zprávy

Pro oddělení témat je použita komponenta TabControl obsahující TabPage pro každé téma. Každá TabPage obsahuje panel pro zobrazení komponent MessageContainer, složené ze dvou labelů, reprezentující zprávy. Pole pro psaní zprávy je založeno na komponentě TextBox, přidává však, po uvedení znaku '#', možnost doplňování existujících hashtagů klávesou TAB. Ostatní grafické komponenty jsou již standardní. ToolStrip pro nástrojovou lištu s tlačítkem historie. StatusStrip pro stavový řádek a Button pro odesílací tlačítko.

4.1.4 Historie

Pro zobrazení přehledu témat je využita upravená komponenta ListView, zobrazující několik sloupců, podle kterých lze seznam témat řadit. Seznam zpráv vybraného tématu je realizován stejně jako v okně zprávy.

4.2 Realizace pluginů a EventAggregator

Jak již bylo nastíněno v kapitole 3.1, o načtení a připojení pluginů se stará jádro aplikace. Tyto pluginy musí být implementovány určeným způsobem a jsou načítány z pracovního adresáře aplikace.

4.2.1 Specifikace pluginu

Pojem *interface* (*rozhraní*) v jazyce C# znamená prvek, udávající strukturu, kterou musí splňovat třídy od něj odvozené. To znamená, že definuje názvy a typy vlastností (*properties*), událostí, indexerů (umožňují přístup k prvkům třídy na základě indexu) a pro metody také parametrů, které následně musí definovat i třída, která je pomocí dědičnosti od tohoto *rozhraní* odvozená. Je zaužívaným postupem začínat název rozhraní písmenem 'I'.

Pro určení kompatibilních pluginů poskytuje aplikace tři rozhraní:

- IMultiTopicPlugin
- IMessageWindowPlugin
- IPluginSettingsTab

IMultiTopicPlugin

Toto rozhraní definuje obecné vlastnosti pluginů použitelných v této aplikaci. Jeho definice je uvedena ve zdrojovém kódu 8. Samotné toto rozhraní dědí od rozhraní `IPartImportsSatisfiedNotification`, které požaduje definici metody `OnImportsSatisfied()`. Tato metoda je zavolána po úspěšné *kompozici* a je tedy vhodným místem pro přihlášení odběru kompozitních událostí.

Třída pluginu odvozená od rozhraní *IMultiTopicPlugin* musí definovat vlastnost EventAggregator. Ten bude při *kompozici* poskytnut jádrem aplikace a bude zprostředkovávat komunikaci s ním. Následují metody `GetPluginIdentifier()` a `GetPluginName()`, které jsou nutné pro jednoznačnou identifikaci pluginu. Metoda `GetSettings()` slouží pro poskytnutí hodnot nastavení jádru a `SetSettings (String xmlSettings)` pro jejich nastavení jádrem. Poslední vyžadovanou metodou je `GetSettingsTabPage()`, která vrací objekt typu *IPluginSettingsTab* reprezentující stránku dialogu nastavení.

```
1 public interface IMultiTopicPlugin :
      IPartImportsSatisfiedNotification
2 {
3   IEventAggregator EventAggregator { get; }
4   String GetPluginIdentifier();
5   String GetPluginName();
6   String GetSettings();
7   IMultiTopicPlugin SetSettings(String xmlSettings);
8   IPluginSettingsTab GetSettingsTabPage();
9 }
```

Zdrojový kód 8: Definice rozhraní IMultiTopicPlugin

IMessageWindowPlugin

Rozhraní typu *IMessageWindowPlugin* slouží pro definici pluginu realizujícího okna zprávy. Je odvozeno od rozhraní *IMultiTopicPlugin* a má jednoduchou definici uvedenou v kódu 9. Nepřidává tedy žádné nové definice metod, ani vlastností, a slouží pro snadné odlišení od ostatních pluginů.

```
1 public interface IMessageWindowPlugin : IMultiTopicPlugin { }
```

Zdrojový kód 9: Definice rozhraní IMessageWindowPlugin

IPluginSettingsTab

Aby mohly pluginy poskytovat možnost nastavení, musí na základě rozhraní *IPluginSettingsTab* definovat třídu obsahující stránku *TabPage*. Ta je potom zobrazena v hlavním dialogu nastavení jádra aplikace. Pro získání této stránky musí být definována metoda `GetSettingsTabPage()`. Pro určení, zda se po zobrazení v dialogu nastavení mají hodnoty uchovat, slouží metody `StoreSettings()` a `RestoreSettings()`. K získání a zapsání hodnot nastavení této stránky jsou potom potřeba metody `GetSettings()` a `SetSettings(String settings)`. Definice tohoto rozhraní je uvedena v kódu 10.

4.2.2 Definice propojení a kompozice

Jádro aplikace i pluginy musí mít definovány body ve zdrojovém kódu, ve kterých dojde *kompozicí* k jejich propojení.

Tyto body jsou na straně jádra definovány ve třídě PluginManager. Ta obsahuje pole (*field*) *eventAggregator*, které je při vytvoření instance třídy PluginManager inicializováno novou instancí třídy *EventAggregator*. Tato instance třídy *EventAggregator* bude sloužit jako prostředník pro předávání kompozitních udá-

```
1 public interface IPluginSettingsTab
2 {
3   String GetSettings();
4   IPluginSettingsTab SetSettings(String settings);
5   void StoreSettings();
6   void RestoreSettings();
7   TabPage GetSettingsTabPage();
8 }
```

Zdrojový kód 10: Definice rozhraní IPluginSettingsTab

lostí mezi jádrem a pluginy. Musí být tedy pluginům k dispozici, což je zajištěno direktivou `[Export(typeof(IEventAggregator))]`.

PluginManager dále obsahuje kolekci typu *IEnumerable<IMultiTopicPlugin>*[23] jménem *Plugins* určenou pro načtení kompatibilních pluginů, což označuje direktiva `[ImportMany]`.

Na straně pluginu okna zprávy *MessageWindowsContainer* musí být definováno místo, kam bude připojen *EventAggregator* poskytovaný jádrem. K tomu poslouží direktiva `[Import(typeof(IEventAggregator))]` u pole (*field*) *eventAggregator* typu *IEventAggregator*.

Direktivou `[Export(typeof(IMultiTopicPlugin))]` je označeno, že má být tato třída připojena jako plugin typu *IMultiTopicPlugin*. Zdrojový kód 11 obsahuje část kódu této třídy s příslušnými direktivami.

```
1 [Export(typeof(IMultiTopicPlugin))]
2 class MessageWindowsContainer : IMultiTopicPlugin,
     IMessageWindowPlugin
3 {
4   // . . .
5   [Import(typeof(IEventAggregator))]
6   IEventAggregator eventAggregator;
7   // . . .
8 }
```

Zdrojový kód 11: Část třídy MessageWindowsContainer

Při spuštění aplikace vytvoří třída jádra ContactList.PluginConnection novou instanci třídy PluginManager, která bude existovat po celou dobu běhu aplikace, a iniciuje kompozici, kterou PluginManager zajišťuje. Při té dojde k načtení pluginů (*částí*) obsažených v pracovním adresáři aplikace a na základě definovaných *exportů* a *importů* se jejich odpovídající typy propojí. Kód kompozice je uveden ve zdrojovém kódu 14 na straně 53.

[23] Kolekce prvků typu *IMultiTopicPlugin* umožňující iteraci nad nimi

4.2.3 Komunikace kompozitními událostmi

Po *kompozici* již mají všechny *části* k dispozici stejnou instanci třídy *EventAggregator* a mohou tedy začít vzájemně komunikovat pomocí kompozitních událostí. Tyto události jsou definované ve třídě PluginInterface a jsou popsány na straně 48. Všechny zdroje přihlášené k odběru dané události ji obdrží po odeslání kterýmkoliv odesílajícím zdrojem.

Přihlášení k odběru události (*subscribe*)

Pro přihlášení k odběru každé z událostí musí mít *část* definovanou vlastnost *SubscriptionToken*, který příslušný odběr identifikuje. Následně je zavolán kód pro přihlášení k odběru, kterému se předá tento token, typ události a metoda, která bude událost obsluhovat. Tento kód se skládá ze tří kroků:

1. Získání typu žádané události od *EventAggregatoru*.
2. Zjištění, zda už není token použit pro odběr. Pokud ano, je odhlášen.
3. Přihlášení k odběru události.

Příklad přihlášení k odběru událostí typu *IncomingMessageEvent*, nesoucí novou příchozí zprávu, je uveden ve zdrojovém kódu 12 na straně 52. V tomto příkladě obsluhuje příchozí zprávu metoda *handleMessage*.

Odeslání události (*publish*)

Postup odeslání události je obdobný, jen není potřeba pracovat s tokenem. Kroky jsou následující:

1. Vytvoření nového kontejneru obsahujícího data odpovídající typu události.
2. Získání typu žádané události od *EventAggregatoru*.
3. Odeslání události.

Příklad publikace událost typu *IncomingMessageEvent* je uveden ve zdrojovém kódu 13 na straně 52.

5 Datová vrstva

Datová vrstva zajišťuje záznam historie komunikace a perzistentní uložení uživatelských nastavení. Uchovávání dat je realizováno dvěma způsoby:

- Aplikační nastavení platformy .NET
- Microsoft SQL Compact Edition

5.1 Aplikační nastavení platformy .NET

Pro ukládání uživatelských nastavení je využita funkcionalita Application Settings poskytovaná knihovnou Windows Forms. Platforma .NET a vývojové prostředí Visual Studio umožňují jednoduchou správu těchto nastavení a není zapotřebí rozsáhlého kódu pro jejich použití. Nastavení se ukládají ve formátu XML a jsou uživatelsky závislá. V operačním systému Windows tedy přímo zajišťuje oddělené nastavení jednotlivých uživatelů.

Mezi ukládaná data patří přihlašovací údaje, nastavení notifikačního zvuku zprávy, klávesová zkratka pro vyvolání hlavního okna kontaktů, velikost a pozice hlavního okna, poslední stav při ukončení aplikace a některá další. Obsahuje také aplikační nastavení, která nejsou uživateli přímo dostupná. Jedná se o nastavení priorit stavů, které jsou využívány při komunikaci se serverem. Přednastavené hodnoty byly zvoleny při vývoji a mají vliv na chování aplikace a serveru při volbě vhodného příjemce, pokud je protistrana připojena z více zařízení nebo klientských aplikací — více různých *zdrojů*. Toto nastavení je však pokročilým uživatelům umožněno změnit ručně v konfiguračním souboru.

5.2 Microsoft SQL Compact Edition

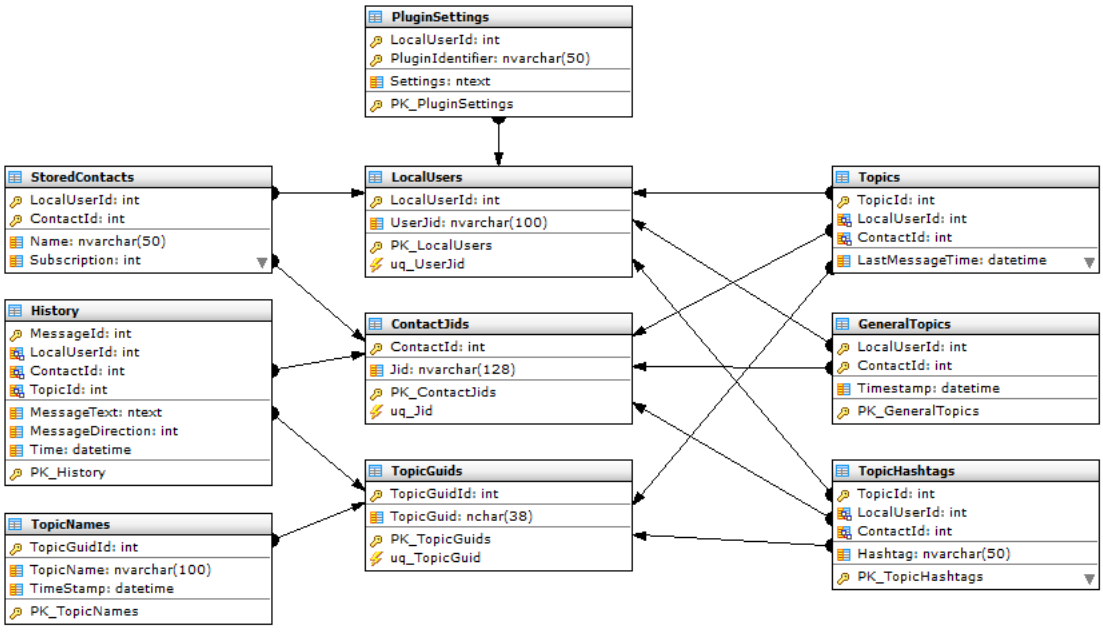
Data o tématické komunikaci jsou uchovávána v relačním databázovém systému SQL CE od společnosti Microsoft. Jedná se o systém využívající jako svou datovou základnu jediný soubor. Toto řešení je výhodné pro menší aplikace, ve kterých se nepracuje s velmi vysokými počty řádků, nedochází k mnoha dotazům za krátký časový úsek a odezva není kritickým kritériem. Klientská aplikace vyvinutá v rámci této práce je tedy pro využití tohoto databázového systému vhodná. Další výhodou je, že není potřeba instalovat celý samostatný databázový systém, který by měl vysoké paměťové nároky — až stovky megabytů. Paměťová náročnost SQL CE se běžně pohybuje maximálně v desítkách megabytů. Systém je dobře provázaný s platformou .NET a jeho použitelnost z programátorského hlediska je proto dobrá. Společnost Microsoft však od tohoto databázového systému postupně upouští například ve prospěch SQL Server LocalDB, který však již není embedded systémem. Soubor databáze je také uživatelsky závislý — každý uživatel Windows má vlastní databázi ve svém profilu.

Pro dotazování databáze a vkládání dat jsou místo standardních SQL[24] dotazů využívány objekty *SqlCeResultSet* a *SqlCeDataReader* poskytované SQL CE serverem pro přístup k datům databáze. Tyto funkce jsou optimalizované pro rychlejší procházení tabulek a vkládání do nich. Předpokládá se u nich, ale nevyžaduje, použití primárních klíčů nebo indexů.

5.2.1 Struktura databáze

Databáze obsahuje deset tabulek, které uchovávají informace o kontaktech, tématech, historii komunikace a také některá nastavení.

Seznam použitých tabulek: LocalUsers, ContactJids, TopicGuids, TopicNames, StoredContacts, Topics, GeneralTopics, TopicHashtags, History, PluginSettings. Diagram[25] tabulek je na obrázku 3.



Obrázek 3: Diagram tabulek databáze

[24] Zkratka Structured Query Language - široce využívaný dotazovací jazyk relačních databází

[25] Diagram byl vytvořen a exportován v nástroji pro správu SQL CE databází MS Compact Maestro - http://www.sqlmaestro.com/products/mssql/compact_maestro/

5.2.2 Popis tabulek

Následuje přehled použitých tabulek, popis jejich účelu a tabulky jejich schémat.

Tabulka LocalUsers

Uchovává seznam lokálních uživatelů, kteří se v aktuální instalaci přihlašovali, a přiřazuje jim číselný identifikátor.

Atribut	Typ	Primární klíč	Index	Integritní omezení
LocalUserId	int	✓		NOT NULL IDENTITY
UserJid	nvarchar(100)		✓	NOT NULL

Tabulka 1: Schéma tabulky LocalUsers

Tabulka ContactJids

Uchovává číselný identifikátor pro každý JID kontaktu.

Atribut	Typ	Primární klíč	Index	Integritní omezení
ContactId	int	✓		NOT NULL IDENTITY
Jid	nvarchar(128)		✓	NOT NULL

Tabulka 2: Schéma tabulky ContactJids

Tabulka TopicGuids

Uchovává číselný identifikátor pro každý GUID tématu.

Atribut	Typ	Primární klíč	Index	Integritní omezení
TopicGuidId	int	✓		NOT NULL IDENTITY
TopicGuid	nchar(38)		✓	NOT NULL

Tabulka 3: Schéma tabulky TopicGuids

Tabulka TopicNames

Uchovává jména jednotlivých témat a v atributu *TimeStamp* datum a čas založení tématu.

Atribut	Typ	Primární klíč	Integritní omezení
TopicGuidId	int	✓	NOT NULL, FOREGIN KEY TopicGuids(TopicGuidId)
TopicName	nvarchar(100)		NOT NULL
TimeStamp	datetime		NOT NULL

Tabulka 4: Schéma tabulky TopicNames

Tabulka StoredContacts

Uložené kontakty se jménem v atributu *Name*, stavem autorizace v *Subscription*, informací, zda byla kontaktu zaslána žádost o autorizaci v *RequestSent*, a indikací, zda má být kontakt odstraněn v *Remove*.

Atribut	Typ	Primární klíč	Integritní omezení
LocalUserId	int	✓	NOT NULL FOREIGN KEY LocalUsers(LocalUserId)
ContactId	int	✓	NOT NULL FOREIGN KEY ContactJids(ContactId)
Name	nvarchar(50)		NOT NULL
Subscription	int		NOT NULL
RequestSent	bool		NOT NULL
Remove	bool		NOT NULL

Tabulka 5: Schéma tabulky StoredContacts

Tabulka Topics

Určuje, kterému uživateli a kontaktu patří jednotlivá témata. Zároveň ukládá čas poslední zprávy v atributu *LastMessageTime*. Zavádí tři cizí klíče z tabulek TopicGuids, LocalUsers a ContactJids.

Atribut	**Typ**	**Primární klíč**	**Integritní omezení**
TopicId	int	✓	NOT NULL FOREIGN KEY TopicGuids(TopicGuidId)
LocalUserId	int		NOT NULL FOREIGN KEY LocalUsers(LocalUserId)
ContactId	int		NOT NULL FOREIGN KEY ContactJids(ContactId)
LastMessageTime	datetime		NOT NULL

Tabulka 6: Schéma tabulky Topics

Tabulka GeneralTopics

Protože hlavní téma „General“ každého kontaktu má prázdný GUID, není možné uchovávat informace o nich přímo v tabulce Topics, protože by docházelo ke konfliktu s primárním klíčem tabulky. K tomuto účelu tedy slouží tato oddělená tabulka.

Atribut	**Typ**	**Primární klíč**	**Integritní omezení**
LocalUserId	int	✓	NOT NULL FOREIGN KEY LocalUsers(LocalUserId)
ContactId	int	✓	NOT NULL FOREIGN KEY ContactJids(ContactId)
Timestamp	datetime		NOT NULL

Tabulka 7: Schéma tabulky GeneralTopics

Tabulka TopicHashtags

Uchovává hashtagy jednotlivých témat patřících příslušným uživatelům.

Atribut	Typ	Primární klíč	Integritní omezení
TopicId	int	✓	NOT NULL FOREIGN KEY TopicGuids(TopicGuidId)
LocalUserId	int		NOT NULL FOREIGN KEY LocalUsers(LocalUserId)
ContactId	int		NOT NULL FOREIGN KEY ContactJids(ContactId)
Hashtag	nvarchar(50)		NOT NULL

Tabulka 8: Schéma tabulky TopicHashtags

Tabulka History

Obsahuje historii zpráv. Každá ze zpráv má svůj jedinečný číselný identifikátor, příslušnost ke kontaktu a uživateli, směr zprávy a její datum a čas.

Atribut	Typ	Primární klíč	Integritní omezení
MessageId	int	✓	NOT NULL IDENTITY
LocalUserId	int		NOT NULL FOREIGN KEY LocalUsers(LocalUserId)
ContactId	int		NOT NULL FOREIGN KEY ContactJids(ContactId)
TopicId	int		NOT NULL FOREIGN KEY TopicGuids(TopicGuidId)
MessageText	ntext		-
MessageDirection	int		NOT NULL
Time	datetime		NOT NULL

Tabulka 9: Schéma tabulky History

Tabulka PluginSettings

Uchovává nastavení pluginů. Předpokládaný datový typ nastavení je XML, může se však jednat o jakoukoliv textovou reprezentaci.

Atribut	Typ	Primární klíč	Integritní omezení
LocalUserId	int	✓	NOT NULL FOREIGN KEY LocalUsers(LocalUserId)
PluginIdentifier	nvarchar(50)	✓	NOT NULL
Settings	ntext		-

Tabulka 10: Schéma tabulky PluginSettings

5.3 Bezpečnost

Bezpečnost databáze je ze souborového hlediska částečně zajišťována oprávněními souborového systému NTFS[26], která neumožňují uživatelům přístup do profilových složek ostatních uživatelů. Toto omezení však neplatí pro administrátorské účty.

Připojení k databázi je chráněno heslem, které je pevně určeno v kódu aplikace. Cílenému pokusu o proniknutí do databáze analýzou kódu aplikace nebo jejího běhu by toto zabezpečení neodolalo. Jednoduchému pokusu o připojení k databázi z jiné aplikace však zabrání. Vzhledem k úrovni citlivosti uchovávaných dat je tato úroveň zabezpečení dostačující.

[26] Zkratka New Technology File System - souborový systém používaný současnými verzemi operačního systému Windows

6 Uživatelská dokumentace

Obsluha aplikace je podobná jiným instant messengerům, takže by její používání nemělo být problémem ani pro méně zkušeného uživatele. Některá funkcionalita je však nová a je v následujících podkapitolách popsána.

6.1 Postup instalace

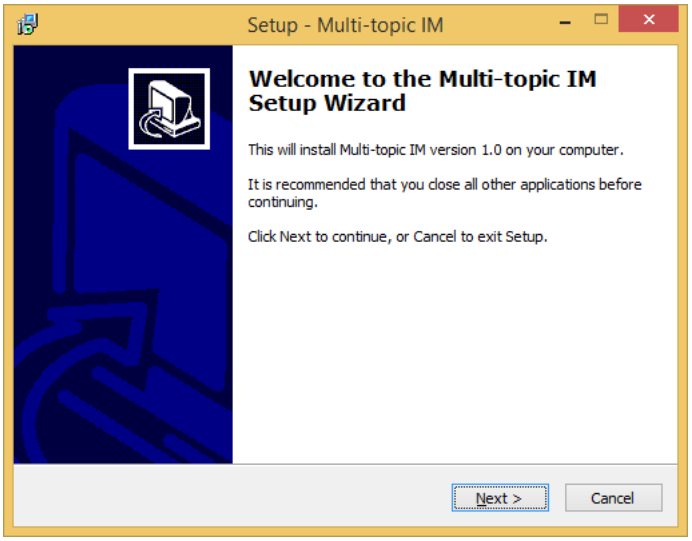
Před instalací samotné klientské aplikace je potřeba nainstalovat následující balíčky, které jsou pro její běh nezbytné:

- Microsoft .NET Framework 4.0 nebo vyšší
- Microsoft SQL Server Compact 4.0 SP1 nebo vyšší

Tyto balíčky jsou dostupné na přiloženém CD.
Instalátor je standardní a instalace by neměla být problémem ani pro začátečníka. Jeho úvodní obrazovka je na obrázku 4.

6.2 Spuštění

Aplikaci lze, v závislosti na volbách vybraných v instalátoru, spustit přes zástupce z plochy a podle verze Windows z nabídky Start nebo přes rozhraní Modern UI (dříve Metro UI). Další možností je spuštění přímo z instalačního adresáře „C:\Program Files (x86)\Multi-topic IM\Multi-topic IM.exe“.

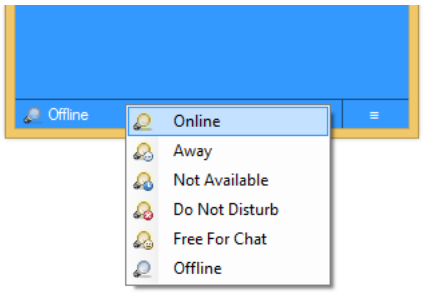


Obrázek 4: Úvodní obrazovka instalátoru

6.3 Přihlášení

Před prvním přihlášením je nejprve potřeba v nastavení zadat své přihlašovací údaje. Postup nastavení je uveden v kapitole 6.4.

Pro připojení do sítě slouží nabídka výběru stavu — obrázek 5.



Obrázek 5: Nabídka výběru stavu

6.4 Popis oken

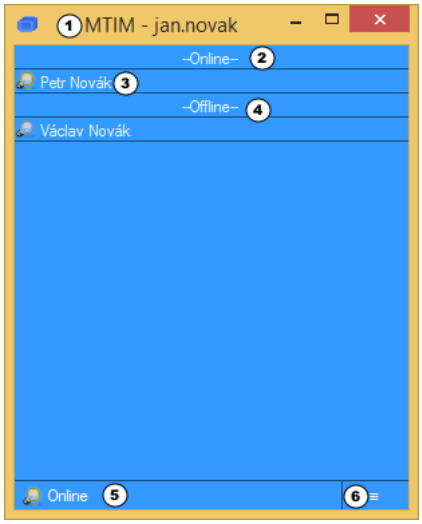
V aplikaci jsou dostupná následující okna:

- Seznam kontaktů
- Okno zpráv
- Historie
- Nastavení
- Záznam XML komunikace

6.4.1 Seznam kontaktů

Jak lze vidět na obrázku 6, v titulku okna (1) je zkratka názvu aplikace Multi-Topic Instant Messenger následovaná uživatelským jménem aktuálního uživatele. Samotný seznam kontaktů obsahuje kategorie připojených (2) a odpojených (4) kontaktů. Dvojklik na kontakt (3) nebo stisk klávesy Enter, pokud je kontakt označen, otevře okno zprávy.

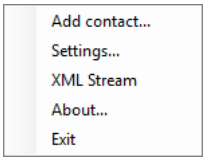
Pravým klikem na kontaktu je možno vyvolat kontextovou nabídku — obrázek 7. Její položka „Message“ otevře okno zprávy, následuje položka „History“, která zobrazí okno s historií komunikace a „Copy JID“ zkopíruje JID kontaktu do schránky. Položkou „Send subscription request“ lze odeslat požadavek o autorizaci příjmu *presence* kontaktu. „Rename“ dovoluje přejmenovat kontakt. Přejmenování kontaktu je kvůli ukládání jména na server umožněno jen pokud je uživatel připojen. Položka „Remove“ slouží k odstranění kontaktu.



Obrázek 6: Hlavní okno - seznam kontaktů

Tlačítko (6) vyvolá hlavní nabídku — obrázek 8. Její položky umožňují, v pořadí: přidání nového kontaktu, otevření dialogu nastavení, otevření okna XML komunikace, dialog o aplikaci a ukončení aplikace. Stejná nabídka se zobrazí i po kliku pravým tlačítkem na ikonu v oznamovací oblasti.

Obrázek 7: Kontextová nabídka kontaktu

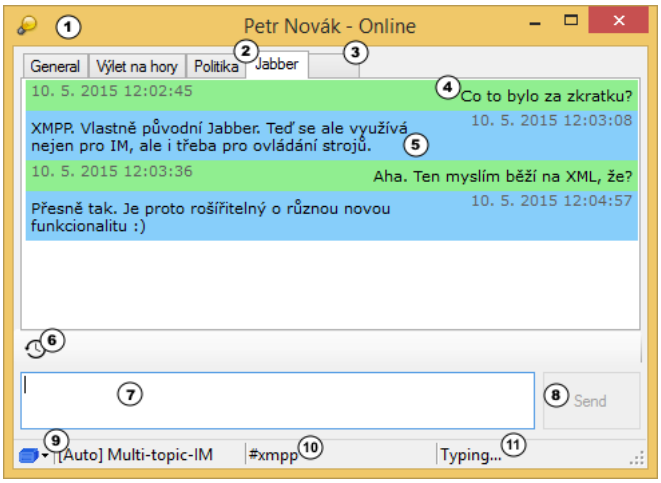


Obrázek 8: Hlavní nabídka

6.4.2 Okno zprávy

Okno zprávy z obrázku 9 slouží pro vedení tématické komunikace. Titulek okna (1) obsahuje jméno kontaktu, jeho status a ikonu odpovídající jeho stavu. Každá ze stránek (2) reprezentuje jedno téma a je nazvaná podle jména tématu. Klik

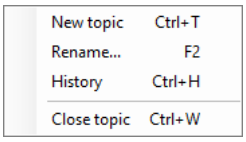
na „ouško“ stránky bez názvu úplně vpravo vytvoří nové téma s přednastaveným jménem „New Topic“ a vygenerovaným hashtagem. K vytvoření nového tématu také slouží klávesová zkratka Ctrl+T a položka „New topic“ z kontextové nabídky zobrazené klikem pravým tlačítkem na „ouško“ některé ze stránek. Kontextová nabídka je na obrázku 10.



Obrázek 9: Okno zprávy

Přejmenovat téma nebo změnit jeho hashtag lze z kontextové nabídky položkou „Rename“ nebo klávesou F2. Dialog přejmenování můžeme vidět na obrázku 11. Hashtag může být tvořen malými a velkými písmeny bez interpunkce, číslicemi a pokud se nebudou vyskytovat na jeho začátku nebo konci, pak i pomlčkami ‘-’ a podtržítky ‘_’. Hlavní téma „General“ nelze přejmenovat.

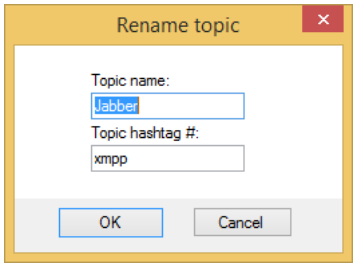
Kontextovou položkou „History“, klávesovou zkratkou Ctrl+H nebo tlačítkem (6) nad polem pro psaní zprávy se otevře okno historie, ve kterém se automaticky načte historie vybraného tématu.



Obrázek 10: Kontextová nabídka tématu

Pro zavření tématu slouží kontextová položka „Close“ a klávesová zkratka Ctrl+W. Téma „General“ není možné zavřít.

Pole pro psaní zprávy (7) poskytuje možnost doplňování dříve použitých hashtagů. Pro použití této funkce musí uživatel zadat znak ‘#’ a opakovaným stiskem klávesy TAB vybrat žádaný hashtag. Může také napsat například „#lo“. V tomto případě bude funkce doplňování nabízet jen hashtagy s prefixem „lo“. Potvrdit výběr lze mezerníkem nebo Enterem. V případě mezerníku se za zvolený hashtag



Obrázek 11: Dialog přejmenování tématu

doplní dvojtečka a mezera a předpokládá se, že uživatel bude pokračovat napsáním zprávy příslušící tomuto tématu. Pokud potom zprávu odešle Enterem nebo tlačítkem „Send“, zpráva se zařadí do tématu s vybraným hashtagem. Odeslání řetězce „#vylet: Doufám že nebude pršet.“ odešle zprávu do tématu s hashtagem #vylet a ponechá aktivní stávající téma. Pokud uživatel odešle jen „#vylet“, toto téma se odesláním pouze aktivuje. Chování je stejné i v případě, že byl tag vepsán ručně a dříve ještě nebyl použit. V tomto případě se vytvoří nové téma s tímto hashtagem.

Pokud je odesílána zpráva kontaktu, jehož aktivní zdroj (*resource*) nepodporuje tématický chat, z tématu bez nastaveného hashtagu, nabídne aplikace jeho nastavení.

Klávesová zkratka Ctrl+D automaticky odešle protistraně text uložený ve schránce.

Ve stavovém řádku se nachází indikátor (9) aktuálního zdroje a umožňuje jej ručně nastavit. V popisku (10) je zobrazen hashtag aktuálního tématu. Notifikace psaní zprávy protistranou je indikována v pravé části stavového řádku.

6.4.3 Historie

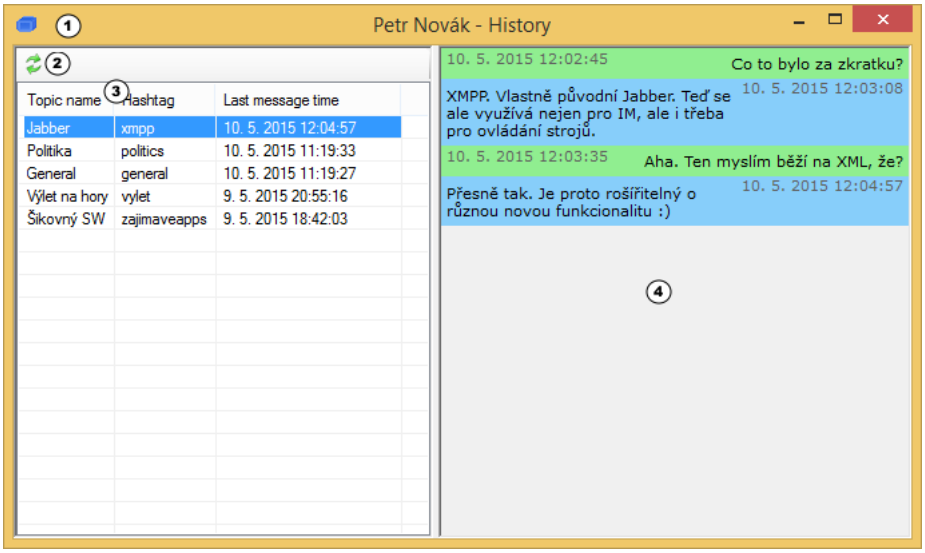
V titulním řádku okna historie (1) z obrázku 12 je nastaveno jméno kontaktu. V seznamu témat (3) lze vybrat téma, pro které se má v části pro zprávy (4) zobrazit historie komunikace. Seznam témat je možné seřadit vzestupně i sestupně klikem na záhlaví sloupce. Přednastavené řazení je sestupně podle času poslední zprávy. Klávesa F5 nebo tlačítko (2) aktualizuje seznam témat i zpráv.

6.4.4 Nastavení

V okně nastavení se nachází několik stránek voleb rozdělených do kategorií Connection a General, následovaných stránkami nastavení pluginů.

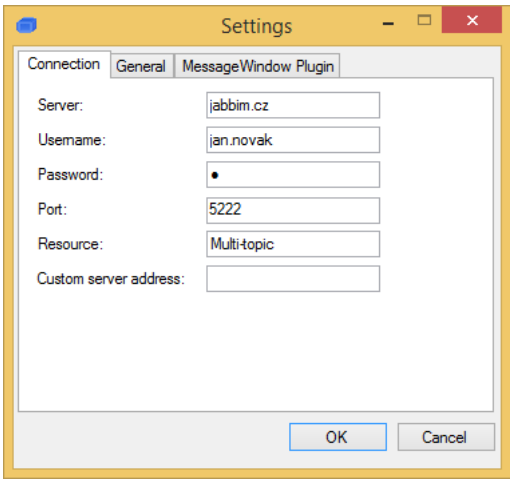
Stránka Connection

Na této stránce se nastavují přihlašovací údaje pro připojení k XMPP serveru. Položka „Server“ pro doménu serveru, „Username“ pro uživatelské jméno,



Obrázek 12: Okno historie

„Password“ pro heslo, „Port“ pro číslo portu, „Resource“ pro zdroj a „Custom server address“ pro adresu serveru lišící se od jeho domény. Výřez této stránky je na obrázku 13.

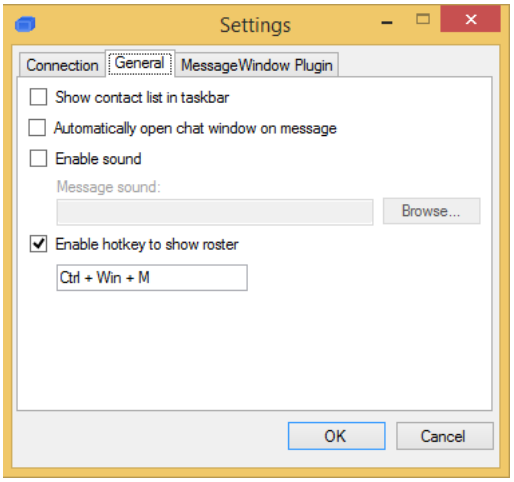


Obrázek 13: Okno nastavení - stránka Connection

Stránka General

Stránka General nastavuje chování aplikace. Tato stránka je na obrázku 14. První položka slouží k určení, zda se má hlavní okno aplikace zobrazovat na hlavním panelu. Pokud je aktivovaná druhá položka, příchozí zpráva automaticky otevře okno zprávy. Dalšími dvěma volbami lze aktivovat a nastavit notifikační zvuk

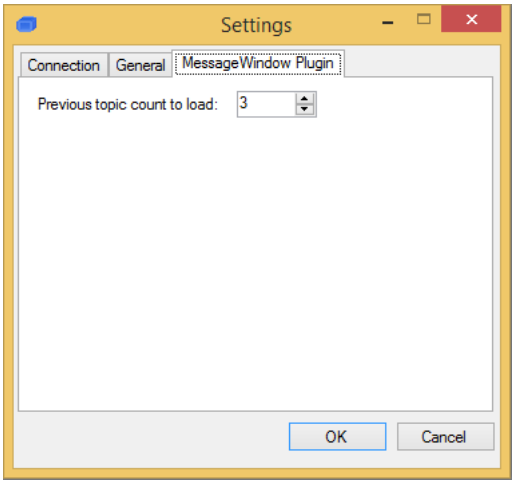
zprávy a globální klávesovou zkratku pro zobrazení/skrytí hlavního okna kontaktů.



Obrázek 14: Okno nastavení - stránka General

Stránka nastavení pluginu

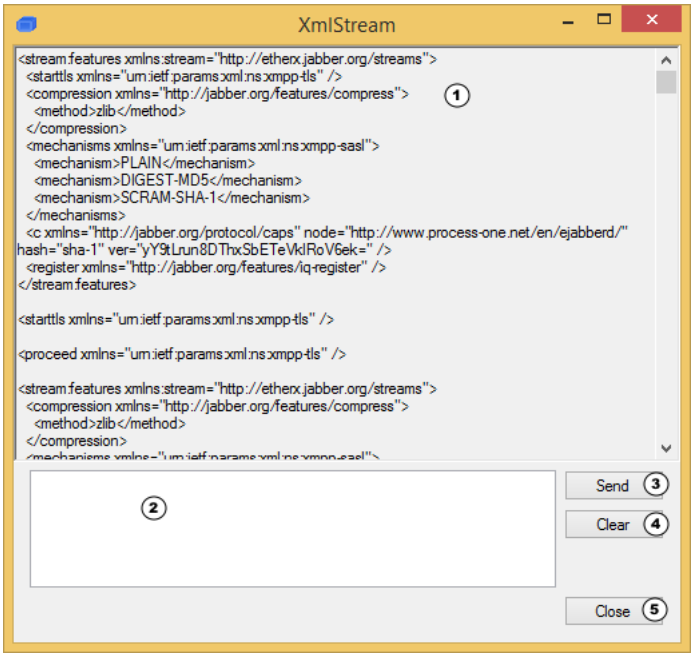
Tato stránka je poskytovaná použitým pluginem okna zprávy. Při použití dalších pluginů, které budou poskytovat nastavení, přibudou v dialogu nastavení další stránky. V případě základního pluginu okna obsahuje stránka pouze jedno nastavení, které určuje kolik posledních témat se má při otevření okna zprávy načíst. Výřez této stránky je na obrázku číslo 15.



Obrázek 15: Okno nastavení - stránka pluginu okna zprávy

6.4.5 Záznam XML komunikace

Okno záznamu XML komunikace je na obrázku 16. V textovém poli (1) se zaznamenává veškerá příchozí i odchozí XML komunikace se serverem. Do pole (2) je možné zadat vlastní XML stanzu a tlačítkem (3) ji serveru odeslat. Tlačítko (4) záznam vymaže a tlačítko (5) okno záznamu XML komunikace zavře.



Obrázek 16: Okno XML komunikace

6.5 Odinstalace

Odinstalovat aplikaci je možné několika způsoby:

1. Z nabídky Start ze složky Multi-topic IM zvolením položky „Uninstall Multi-topic IM“.
2. Z Programů v Ovládacích panelech.
3. Z instalačního adresáře (standardně „C:\Program Files (x86)\Multi-topic IM“) spuštěním „unins000.exe“.

Závěr

Myšlenka vedoucí k tématu této práce vznikla kvůli absenci textového komunikačního nástroje, který by řešil problém prolínání probíraných témat. Za tímto účelem byl rozšířen komunikační protokol XMPP a vyvinut klient s původně neplánovanou podporou pluginů. Komunikace je v grafickém uživatelském rozhraní logicky oddělena na různá probíraná témata a tím výrazně redukuje problém slévání témat. Aspekt, který aplikace nedokáže ovlivnit, je důslednost uživatelů v držení se probíraného tématu. Pokud dojde k výraznějšímu odchýlení, je lepší založit téma nové, jinak se výhoda tématicky oddělené komunikace vytrácí. Pokud se však uživatel s touto funkcionalitu sžije, dojde k zefektivnění jeho textové komunikace. Není potřeba pro jistotu čekat na dořešení jednoho tématu před započetím nového, protože uživatel může v době, kdy protistrana píše zprávu, psát do tématu jiného.

Při vývoji byly využity poměrně nové zajímavé postupy, jako MEF nebo EventAggregator, které umožňují větší tvárnost aplikací a poskytují jednoduchou možnost rozšiřování jejich funkcionality, a to i třetími stranami. Použitý rozšiřitelný protokol XMPP ukázal svou sílu a univerzálnost, která není limitovaná pouze na instant messaging řešení.

Cílem práce nebylo vytvoření klientské aplikace, která podporuje všechna XEP rozšíření, ale vytvoření nové funkcionality, která by se v budoucnu sama mohla stát standardizovaným rozšířením protokolu XMPP. Vývoj aplikace bude dále probíhat přidáváním funkcí jádru aplikace i pluginu okna zprávy. Je možný i vývoj nového pluginu, který by řešil oddělení témat jiným způsobem. Zajímavou výhodou by také jistě byl přenos tématické funkcionality na velmi oblíbené mobilní platformy Android a iOS.

Conclusions

Thought leading to the topic of this bachelor thesis origins in absence of instant messaging tool, that would solve topic merging problem. As a solution for this absence an extension of XMPP protocol and client application with initially unplanned support of add-ons were created. Graphical user interface of client application is divided into topics while significantly reducing topic merging problem. Although to fully harness this feature users must learn to direct their messages into according topics. In case the topic diverges from it's direction, a new topic should be created to prevent recurrence of topic merging. An effectiveness of communication is significantly increased by learning this practice. User doesn't have to wait for finalization of current topic before starting a new one, as he can start composing a message in another topic, while waiting for his partner's response.

During development were used two innovative techniques - MEF and EventAggregator. These techniques make application more variable by providing simple way to extend its features even by third parties. Communication protocol XMPP proved to be powerful and universal tool that might not be limited only for instant messaging purposes. The goal of this bachelor thesis wasn't to create full-featured client application with support of all XEP extensions, but to create innovative functionality, that might even become standardized in the future.

Application will be further developed by adding new functions to both core and message window add-on. There is also possibility of creating a new message window add-on with different topic separation approach. Interesting would also be porting multi-topic features for widely used mobile platforms Android and iOS.

A Popis projektů a jejich tříd

Níže je popsán účel jednotlivých projektů se stručným popisem jejich tříd.

Projekt CustomForms

Tento projekt obsahuje vytvořené dialogy používané v následujících projektech. Seznam dialogů:

- **inputForm** — obecný vstupní dialog s možností nastavení popisu vstupního pole
- **inputFormDouble** — obecný vstupní dialog podobný předchozím, ale se dvěma vstupními poli
- **inputFormJid** — vstupní dialog s jedním vstupním polem, které přijímá pouze hodnoty ve tvaru JID

Projekt MT-Shared

V tomto projektu se nachází třídy využívané v následujících projektech. Jsou zařazeny do zvláštního projektu, aby nebyly definovány několikrát a také aby nevznikaly kruhové závislosti projektů. Obsahuje následující třídy:

- **ButtonWithoutCues** — upravené tlačítko, které při svém označení nezobrazuje vnitřní rámeček
- **ContactInfo** — třída obsahující informace o uživateli v seznamu kontaktů. Některé z nich:

  Status — stav kontaktu

  Nickname — jméno nebo přezdívka kontaktu

  Subscription — stav autorizace

  MultiTopicSupported — podpora tématické komunikace

  CurrentResource — aktuálně vybraný zdroj

  Resources — seznam všech zdrojů
- **CustomNumericUpDown** — upravený ovládací prvek pro zadávání čísla, který má skryté šipky na straně, a je využíván pro zadávání čísla portu v nastavení
- **Enum** — výčty ContactStatus, ChatStates a DBHistoryDirection
- **HistoryItem** — položka historie používaná při načítání z databáze
- **HistoryItems** — kolekce položek historie
- **HistoryTopicItem** — informace o tématu uloženém v historii

- **HistoryTopicItems** — kolekce informací o tématech
- **ResourceInfo** — informace o zdroji uživatele používaná ve třídě ContactInfo
- **SharedConstants** — sdílené konstanty používané ostatními projekty
- **WindowCoverChecker** — poskytuje metody volající systémové funkce pro zjištění překrytí okna

Projekt Multi-topic IM

Tento projekt je jádrem celého řešení. Zobrazuje seznam kontaktů a umožňuje jeho správu, stará se o spojení s XMPP serverem a komunikuje s databází. Umožňuje měnit nastavení chování aplikace i pluginu okna zprávy. Také umožňuje zobrazení komunikace se serverem v XML podobě. Obsahuje následující formuláře a třídy:

- Form **ContactList** — hlavní okno programu, obsahující seznam připojených a odpojených kontaktů, umožňuje nastavení stavu a obsahuje hlavní menu
- Form **HistoryForm** — okno pro zobrazení seznamu dříve komunikovaných témat a jejich obsahu
- Form **settingsForm** — okno nastavení aplikace
- Form **XmlStreamForm** — okno zobrazující XML komunikaci se servem
- Form **AboutBox** — dialog o aplikaci
- **BufferedTreeView** — upravená třída s podporou double-bufferingu
- **Common** — rozšiřující funkce
- **Constants** — konstanty
- **ContactList.Comm** — komunikace se serverem pomocí agsXMPP
- **ContactList.DB** — zajišťuje dotazování SQL CE databáze
- **ContactList.PluginConnection** — po připojení pluginu inicializuje používané události
- **Encryption** — třída pro zašifrování/odšifrování hesla při zápisu/čtení z nastavení
- **GlobalHotkey** — spravuje globální klávesové zkratky
- **HashGuidSyncElement** — element rozšíření pro sladění různých GUID identifikátorů při stejném hashtagu

- **HashGuidSyncIq** — *IQ* stanza ke sladění identifikátorů
- **HistoryTopicView** — upravená grafická komponenta ListView zobrazující seznam témat v historii
- **PluginManager** — stará se o načtení pluginu a jeho kompozici s jádrem
- **PresenceCustom** — upravená *presence* stanza
- **RosterControl** — uživatelská grafická komponenta zobrazující seznam připojených a odpojených kontaktů
- **RosterViewClasses** — pomocné třídy využívané komponentou RosterControl
- **SettingsData** — třída reprezentující data nastavení
- **SettingsManager** — třída spravující ukládání a načítání nastavení
- **SqlCeExceptionEventArgs** — vyjímka databázového systému
- **TopicElement** — element rozšiřující stanzu *message* o podporu témat
- **TopicInfoQuery** — element pro přenos požadavku o změnu vlastností v *IQ* stanze
- **TopicSupportElement** — element indikující podporu tématické komunikace
- **Versioning** — podpora automatické inkrementace verze

Projekt PluginInterface

Tento projekt definuje požadavky pro vytvoření pluginu a poskytuje třídy událostí použitelných pro komunikaci jádra aplikace a pluginu okna zprávy. Dále obsahuje třídy argumentů těchto událostí. Projekt obsahuje následující:

- Interface **IMessageWindowPlugin** - definovaný *interface*, ze kterého musí být odvozena třída pluginu pro okno zprávy
- Interface **IMultiTopicPlugin** - definovaný *interface*, ze kterého musí být odvozeny pluginy
- Interface **IPluginSettingsTab** - definovaný interface, kterému musí odpovídat stránka nastavení, která bude importována do nastavení
- Enum **WindowStateChangeTypes** — výčet typů změny stavu okna

- Třídy datových kontejnerů pro události:

  InstantMessage — komunikační zpráva

  HashtagEventData — informace o hashtagu

  TopicInfoEventData — data o tématu pro úpravu jeho vlastností

  WindowStateChangedEventData — informace o změně stavu okna

  ContactInfoEventData — informace o změně dat kontaktu

  ChatStateEventData — informace o změně stavu psaní zprávy

  HistoryRequestEventData — data požadavku o načtení historie

  HistoryResponseEventData — vrácená data se zprávami z historie

  TopicGuidChangeEventData — informace pro změnu GUID tématu

  OpenTopicEventData — informace požadavku o otevření tématu

  OpenHistoryEventData — informace požadavku o otevření historie

  LoadHistoryTopicsEventData — data požadavku o načtení seznamu témat

  LastTopicsEventData — data požadavku o načtení posledních komunikovaných témat

  TopicInfoRequestEventData — data požadavku na změnu vlastností tématu

- Třídy událostí typu CompositeEvent publikované jádrem aplikace:

  IncomingMessageEvent — událost příchodu zprávy

  HashtagDataEvent — událost přenosu informací o hashtagu

  IsConnectedEvent — událost indikující změnu stavu připojení lokálního uživatele

  RequestWindowOpenStateEvent — událost požadavku na zjištění stavu okna

  TopicInfoSetEvent — událost nastavení informací o tématu

  PublishContactInfoEvent — událost informující o změně stavu kontaktu

  OpenMessageWindowEvent — událost požadavku o otevření okna zprávy

  SetChatStateEvent — událost nastavení stavu psaní zprávy

  ResponseHistoryDataEvent — událost načtených zpráv historie

  ChangeTopicGuidEvent — událost požadavku o změnu identifikátoru tématu

OpenTopicEvent — událost příkazu k otevření tématu

SetMyShowTypeEvent — událost nastavení statusu

SetLastTopicsEvent — událost dat posledních komunikovaných témat

SetTopicInfoEvent — událost nastavení vlastností tématu

- Třídy událostí typu CompositeEvent odebírané jádrem aplikace:

  SendMessageEvent — událost pro odeslání zprávy

  RequestAvailableHashtags — událost požadavku na zjištění existujících hashtagů

  ResponseWindowOpenStateEvent — událost odpovědi o stavu okna

  TopicChangeInfoEvent — událost změny vlastností tématu

  WindowStateChangedEvent — událost informující o změně stavu okna

  SendChatStateEvent — událost pro odeslání stavu psaní zprávy

  RequestHistoryDataEvent — událost požadavku o načtení historie

  OpenHistoryEvent — událost požadavku o otevření historie

  RequestLastTopicsEvent — událost požadavku o seznam posledních témat

  RequestTopicInfoEvent — událost požadavku na zjištění informací o tématu

Projekt MessageWindowLib

Tento projekt je pluginem okna zprávy. Plugin je definovaný podle rozhraní *IMessageWindowPlugin* z projektu PluginInterface, aby jej bylo možné korektně propojit s jádrem klientské aplikace.

- **Common** — třída společných pomocných funkcí a konstant
- **MessageContainer** — uživatelská grafická komponenta tématické, stavové nebo informační zprávy
- **MessageLabel** — uživatelská grafická komponenta obsahující text zprávy
- **MessageLog** — uživatelská grafická komponenta zobrazující zprávy třídy MessageContainer
- **MessageWindow** — formulář okna zprávy jednoho kontaktu
- **MessageWindowsContainer** — řídící třída pluginu, spravující jednotlivá okna MessageWindow, obsluhuje komunikaci s jádrem aplikace

- **SettingsTabPage** — stránka nastavení pluginu, která se importuje do nastavení aplikace
- **SuggestionTextBox** — komponenta textového pole, umožňující napovídaní hastagů
- **TopicTabPage** — stránka tématu
- **WindowSettings** — data nastavení okna

B Zdrojové kódy

```
1  private void SubscribeMessageEvent()
2  {
3    // krok 1
4    IncomingMessageEvent messagePostedEvent =
5      eventAggregator.GetEvent<IncomingMessageEvent>();
6
7    // krok 2
8    if (incomingMessageSubscriptionToken != null)
9      messagePostedEvent.Unsubscribe(incomingMessageSubscriptionToken);
10
11   // krok 3
12   incomingMessageSubscriptionToken = messagePostedEvent.Subscribe(
13     handleMessage,
14     ThreadOption.SubscriberAffinityThread,
15     false);
16 }
```

Zdrojový kód 12: Kód pro odběr události IncomingMessageEvent

```
1  private void PublishMessageEvent(String fromJid, String messageText,
       Guid topicGuid, String topicName, String hashtag, Boolean
       autoOpenWindow, DateTime messageTime)
2  {
3    // krok 1
4    InstantMessage newMessage = new InstantMessage(fromJid,
5        messageText,
6        topicGuid,
7        topicName,
8        hashtag,
9        autoOpenWindow,
10       messageTime);
11   // krok 2 a 3
12   eventAggregator.GetEvent<IncomingMessageEvent>()
13       .Publish(newMessage);
14 }
```

Zdrojový kód 13: Kód pro odeslání události IncomingMessageEvent

```
1  public class PluginManager {
2     // Kontejner uchovávající data o proběhlé kompozici
3     private CompositionContainer container;
4     // Export třídy EventAggregator
5     [Export(typeof(IEventAggregator))]
6     public IEventAggregator eventAggregator = new EventAggregator();
7     // Import pluginů implementujících interface IMultiTopicPlugin
8     [ImportMany]
9     public IEnumerable<IMultiTopicPlugin> Plugins;
10    // Pole (field) pro plugin okna zprávy
11    public IMessageWindowPlugin MessageWindowPlugin;
12    public void InitPlugins() {
13      // Nový katalog částí (part)
14      AggregateCatalog catalog = new AggregateCatalog();
15      // Načtení dostupných částí
16      catalog.Catalogs.Add(new AssemblyCatalog(typeof(Program).
            Assembly));
17      catalog.Catalogs.Add(new DirectoryCatalog(AppDomain.
            CurrentDomain.BaseDirectory));
18      // Vytvoření nového kontejneru na základě načtených částí
19      container = new CompositionContainer(catalog);
20      // Spuštění kompozice
21      try { this.container.ComposeParts(this); }
22      catch (CompositionException compositionException) {
23        Console.WriteLine(compositionException.ToString()); }
24      // Výběr pluginu okna zprávy
25      if (Plugins != null) {
26        IMessageWindowPlugin resultWinPlg = null;
27        foreach (IMultiTopicPlugin plg in Plugins) {
28          if (plg is IMessageWindowPlugin)
29          { resultWinPlg = (IMessageWindowPlugin)plg; }}
30        MessageWindowPlugin = resultWinPlg;
31    }}}
```

Zdrojový kód 14: Třída PluginManager realizující MEF kompozici

C Obsah přiloženého CD/DVD

Na samotném konci textu práce je uveden stručný popis obsahu přiloženého CD/DVD, tj. jeho závazné adresářové struktury, důležitých souborů apod.

bin/
: Instalátor MTIM-Setup programu. Adresář obsahuje i všechny runtime knihovny a další soubory potřebné pro bezproblémový běh instalátoru.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

src/
: Kompletní zdrojové texty programu Multi-topic Instant Messenger se všemi potřebnými (příp. převzatými) zdrojovými texty, knihovnami a dalšími soubory potřebnými pro bezproblémové vytvoření spustitelných verzí programu.

readme.txt
: Instrukce pro instalaci a spuštění programu Multi-topic Instant Messenger, včetně všech požadavků pro jeho bezproblémový provoz pro účel testování při tvorbě posudků práce a pro účel obhajoby práce.

Navíc CD/DVD obsahuje:

data/
: Ukázková a testovací data použitá v práci a pro potřeby testování práce při tvorbě posudků a obhajoby práce.

install/
: Instalátory aplikací, runtime knihoven a jiných souborů potřebných pro provoz programu Multi-topic Instant Messenger, které nejsou standardní součástí operačního systému určeného pro běh programu. vztahující se k práci.

U veškerých cizích převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro všechny použité (a citované) materiály, u kterých toto není splněno a nejsou tak obsaženy na CD/DVD, je uveden jejich zdroj (např. webová adresa) v bibliografii nebo textu práce nebo v souboru `readme.txt`.

Reference

[1] P. Saint-Andre, K. Smith, R. Troncon. *XMPP: The Definitive Guide* : Building Real-Time Applications with Jabber Technologies, O'Reilly Media, 2009.

[2] P. Saint-Andre. *RFC 6120* : Extensible Messaging and Presence Protocol (XMPP): Core. http://xmpp.org/rfcs/rfc6120.html [cit. 12.5.2015]

[3] P. Saint-Andre. *RFC 6121* : Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. http://xmpp.org/rfcs/rfc6121.html [cit. 12.5.2015]

[4] Microsoft. *Managed Extensibility Framework* : http://mef.codeplex.com/wikipage?title=Overview&referringTitle=Home 22.12.2009, [cit. 12.5.2015]

[5] Microsoft. *Event Aggregator* : https://msdn.microsoft.com/en-us/library/ff921122.aspx. [cit. 12.5.2015]

[6] Brian Noyes. *Composite Extensions for Windows Forms* : http://briannoyes.net/2008/10/13/composite-extensions-for-windows-forms/ 13.10.2008, [cit. 12.5.2015]