VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SPRÁVCE OSOBNÍCH INFORMACÍ NAD XMPP

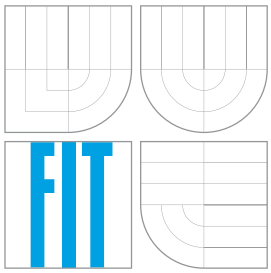
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ GUŇKA

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SPRÁVCE OSOBNÍCH INFORMACÍ NAD XMPP

PERSONAL INFORMATION MANAGEMENT OVER XMPP

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ GUŇKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MAREK SCHMIDT

BRNO 2009

Abstrakt

Tato práce se zabývá implementací správce osobních informací pomocí protokolu XMPP. Zaměřuje se hlavně na nejčastější potřebné funkce při plánování času. Těmito funkcemi se myslí: uchovávání poznámek, plánování událostí a práce s elektronickou poštou. To vše v co nejjednodušší formě, aby byla zachována možnost využití pomocí mobilních zařízení s přístupem k internetu.

Abstract

This work deals with implementation of personal information manager using the XMPP protocol. It is aimed mainly at most needed tasks in time planning. These task are: keeping of notes, events planning and work with electronic mail. This all in the easiest form so that the functionality could be used by mobile devices with internet access.

Klíčová slova

poznámky, kalendář, plánování, email, robot, bot, jabber, XMPP

Keywords

notes, calendar, scheduling, email, robot, bot, jabber, XMPP

Citace

Jiří Guňka: Správce osobních informací nad XMPP, bakalářská práce, Brno, FIT VUT v Brně, 2009

Správce osobních informací nad XMPP

Prohlášení

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

.....
Jiří Guňka
27. května 2009

Poděkování

Rád bych poděkoval panu Ing. Marku Schmidtovi za vedení mé bakalářské práce, odbornou pomoc a podnětné připomínky.

Obsah

Kapitola 1

Úvod

1.1 Motivace

Situace, kdy Vás napadne řešení některého z aktualních problémů, může nastat kdykoliv. Sedíte-li zrovna v zasedací místnosti těsně před poradou nebo jedete ráno do práce a mnoho dalších, kdy nejste u svého počítače, abyste mohli řešení vyzkoušet. Často se stane, že danou myšlenku zapomenete, pokud si ji ihned nezapíšete. Aby mohla být tato poznámka vždy s Vámi, když ji zrovna potřebujete, je zapotřebí mít dobrý nástroj pro její uchování.

V dnešní době, kdy ceny mobilních telefonů s možností připojení k internetu jsou srovnatelné s těmi bez ní, je jednou z nejlepších možností, aby nástroj fungoval online.

Neustále chceme být informováni o tom, co se děje. Protože většina běžných mobilních tarifů pro připojení k internetu je účtována za přenesená data, záleží na tom v jaké podobě informace přijmáme. Jestli spolu s grafikou bez nosné informace ve formě obrázků, kdy za každá zbytečná data platíme, nebo jako prostý text s konkrétní informací. Navíc nikdy nevíte u jakého zařízení se octnete a jaké budou jeho možnosti. Proto je dobré mít nástroj, který není platformě závislý a může být co nejsnadněji používán. Proto je dobré, mít vždy takový nástroj, jako je náš robot.

1.2 Cíle projektu

Za cíl jsem si vytýčil vytvořit XMPP robota s *jednoduchým* rozhraním, který bude spravovat nejnutnější osobní informace jako jsou poznámky, plánování události na konkrétní datum s upomínáním, dále příjem a odesílání elektronické pošty. Proto pro práci s robotem stačí připojení k internetu a zařízení (počítač, mobilní telefon), ve kterém je nainstalován *Jabber klient*. Pro větší pohodlí a snadnější ovládání s podporou *Ad-Hoc commands*.

1.3 Obsah technické zprávy

V druhé kapitole se seznámíme se *základními pojmy a použitými technologiemi* pro pochopení problematiky.

V třetí kapitole provedem *návrh*, jak by takový robot měl vypadat – jaké by měl mít funkce.

Ve čtvrté kapitole si ukážeme, jak robota *implementovat* a taky další možný směr vývoje.

V páté kapitole najdeme *závěr* a tedy zhodnocení této práce, dosažených výsledků, zopakování dalších možných postupů a vylepšení.

Dále v dodatcích nalezneme, jak robota ovládat a co je zapotřebí k jeho uspěšnému provozu. Obrázky z komunikace a obsah přiloženého CD.

Kapitola 2

Základní pojmy a použité technologie

V této kapitole se seznámíme se základními pojmy a technologiemi použitými pro vývoj robota.

2.1 Základní pojmy

2.1.1 Instant messaging

Instant messaging je forma internetové komunikace v realném čase, které se účastní alespoň dva uživatelé za pomoci sítě/protokolu k tomu určeného například *Jabber/XMPP*, *ICQ/OSCAR*, *MSN messanger/MSNP* a další. U většiny systémů je zapotřebí se před prvním použitím zaregistrovat. Podle síť/protokolu a klienta, kterého uživatelé používají, mohou využít různých rozšiřujících služeb, jako jsou například sdílení dokumentů, video přenos, odesílání SMS a mnoho dalších.

Rozdíl mezi elektronickou poštou a instant messagingem je ten, že komunikace je podobnější opravdovému rozhovoru. Rychlost odpovědi jedné strany druhé je proti běžnému rozhovoru zpožděna o akt napsáni odpovědi a zpoždění přenosu na trase mezi klienty.

Více viz [17][14]

2.1.2 Klient

Klient je program, který je nainstalován na uživatelově zařízení a pomocí kterého uživatel komunikuje. Klient zprostředkovává uživateli služby nabízené danou sítí. Pro většinu sítí existuje více klientů, ale né všechny společnosti dovolují používat jiného než oficialního klienta např. ICQ. Klient může umět pracovat s více druhy sítí najednou, což je velká výhoda, protože uživatel nemusí mít spuštěných více klientů a tím šetří systémové prostředky.

Klient	Operační systém	URL
Kopete	Kubuntu	http://kopete.kde.org/
Miranda	Windows	http://www.miranda-im.org/
Adium	Mac OS X	http://adium.im/

Tabulka 2.1: Tabulka více protokolových klientů

Klient nemusí být pouze nainstalovaný program ve vašem počítači. Klienta můžete mít ve svém mobilním zařízení, klient může být také napsán pro webové rozhraní. Za klienta může být považován i robot.

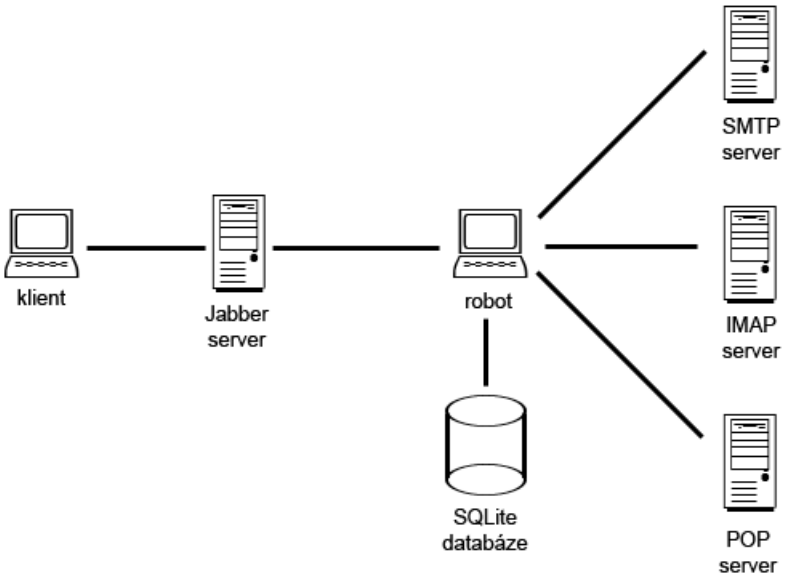
2.1.3 Robot

Robot nebo taky zkráceně Bot při XMPP je program, který se přihlásí k síti a ostatní uživatelé jej vidí jako jiného uživatele. Robot se může vyskytovat v konferencích nebo jako běžný uživatel. Může mít různé funkce, mezi které patří například předpověď počasí, vyhledání spojů MHD, zábavná konverzace s uživateli. V konferencích mohou působit jako sběrači rozmanitých statistických údajů, mohou přidělovat práva přednastaveným uživatelům, udržovat v konferencích pořádek atd.

Robota si může naprogramovat kdokoliv, kdo ovládá programovácí jazyk umožnující práci s protokolem XMPP. Jeho služby zpřístupnit ostatním a podpořit tak komunitu.

Mezi nejznámější roboty mezi českou Jabber komunitou patří *Taxator* a *Boticka*. Seznam dalších robotů naleznete zde http://www.jabber.cz/wiki/Bot

Na následujícím obrázku 2.1 můžeme vidět, že uživatelův klient nemusí obsahovat speciální funkce pro dané úkoly, ale stačí, aby měl uživatel v kontaktech robota, který umí vyhovět jeho požadavkům. Uživatel tak může měnit klienty, zařízení(počítače, mobilní telefony . . .), ze kterých se připojuje a stále může využívat různé rozšiřující služby. Vše obstará robot.



Obrázek 2.1: Komunikace klienta s robotem

2.2 XML

XML nebo-li *eXtensible Markup Language* je univerzální, otevřený, značkovací jazyk určený k uchování a výměně dat, vytvořený společností W3C [13]. Jazyk je strukturovaný a nenese

informaci o tom, jak bude dokument vypadat. O to se postarají například *Kaskádové styly*. Tento jazyk vychází z jazyka SGML.

V dnešní době, kdy neexistuje pouze jediný operační systém a jedna sada programů, se kterými by všichni pracovali, je potřeba využívat pro předávání informací co nejvíce přenositelný formát. Toto nám právě XML umožňuje.

Struktura dokumentu v XML je dána značkami, ale jména a pořadí značek není určeno. To už je na programu, aby věděl, co která značka znamená a podle toho dokument zpracoval. Dané je pouze to, že každý dokument musí obsahovat kořenový element. Používají se párové a nepárové značky. Značky se mohou zanořovat. Chceme-li, aby byl dokument validní, nesmí se tyto značky křížit.

Není-li uvedeno jinak, je XML dokument standardně kódován v kódování *Unicode*. Toto umožnuje přenositelnost mezi prostředími, která jinak využívají jinou znakovou sadu.

Jelikož je jazyk otevřený a jeho dokumentace je zdarma, může jej kdokoli nastudovat a následně využít pro svou aplikaci.

```
<?xml version="1.0" encoding="UTF-8"?>
<udalost>
   <datum>28.02.2009</datum>
   <cas>12.00</cas>
   <text>Velmi důležitá událost</text>
</udalost>
```

Ukázka, jak by mohlo vypadat uložení události pomocí XML

2.2.1 Využití XML

Už tedy známe některé vlastnosti tohoto jazyka a můžeme si ukázat nejznámější příklady jeho využití:

- *XHTL* – pokračovatel jazyka HTML
- *SVG* – jazyk pro popis vektorové grafiky
- *RSS* – čtení novinek na webu
- *Jabber* – výměna zpráv

Více o jazyku samotném a jeho využití naleznete zde [15]

2.3 XMPP

XMPP nebo-li *Extensible Messaging and Presence Protocol* je standardizovaný, otevřený protkol, založený na základě zasílání XML zpráv tzv. *stanz*.

2.3.1 Historie

Nyní si nastíníme v rychlosti historii protokolu XMPP.

Všechno začalo v roce 1998, kdy Jeremie Miller začal pracovat na projektu *Jabber*. V roce 2000 byla vydána první verze Jabber serveru *jabberd*.

Od té doby proběhlo několik pokusů o standardizaci, ale doopravdy k tomu došlo až v roce 2004. Byly vydány standardy XMPP: Core – RFC 3920 [7] základní prvek popisující elementární záležitosti a dále RFC 3921 [8], RFC 3922 [9] a RFC 3923 [10].

Podrobnou historii XMPP naleznete na `http://xmpp.org/about/history.shtml`

2.3.2 Vlastnosti

Protokol XMPP je *otevřený*, *decentralizovaný* protokol, což jsou asi jeho dvě nejzásadnější výhody proti jiným komunikačním protokolům.

Otevřený díky této vlastnosti může přispět k dalšímu vývoji kdokoli. Každý si může zvolit, jakého klienta bude používat. Zda některého z již vydaných nebo si naprogramuje vlastního. Díky tomuto faktu lze také vyvíjet ruzná rošíření a poskytovat tak další a další služby.

Decentralizovaný neexistuje jenom jeden server, který by shromažďoval veškeré informace, ale informace jsou rozděleny mezi servery poskytující tuto službu. Jabber server se dá provozovat *globálně* pro uživatele z celého světa například server *jabber.org*, pro firemní učely v rámci intranetu nebo jej můžete provozovat doma. Nabízené služby se pak mohou lišit podle toho, k jakému serveru se přihlašujete.

Další důležitou vlastností tohoto protokolu je *bezpečnost*. Pro komunikaci se serverem lze například použít šifrování SSL/TLS. Záleží tedy na každém, jestli bude provozovat komunikaci zabezpečenou nebo nikoliv.

Zprávy jsou přenášeny v mezinárodním kódování *UTF-8*.

Za ostatní vlastnosti můžeme vyjmenovat ještě transporty, konference a jiné. Seznam dalších vlastností můžeme najít na [16]

2.3.3 Komunikace

Přesto, že síť Jabber je decentralizována a nemá jeden centrální server, jako jiné sítě například ICQ, všichni se spolu mohou domluvit.

Probíhá za pomoci posílání XML zpráv.

JID

Aby bylo možné doručavat uživatelům zprávy, je potřeba je pomocí něčeho identifikovat. K jednoznačné identifikaci každého uživatele slouží tzv. *JID*.

JID nebo-li *Jabber ID* je podobný emailové adrese. Skladá se ze dvou povinných částí a to jména uživatele a adresy serveru, ke kterému se uživatel připojuje. Tyto dvě části od sebe odděluje znak *@*. JID našeho robota je *bron@jabbim.cz*. Z toho plyne, že uživatelské jméno je *bron* a připojuje se k serveru *jabbim.cz*.

Ještě je zde další parametr, který může JID obsahovat a tím je tzv. *zdroj*. Uvádí se za adresou serveru a je od ní oddělen znakem */*. Zdroj nám může posloužit k identifikaci připojení pro následnou odpověď, pokud má uživatel spušteňo více klientů. Nebo tak může jen dát najevo že je v práci např. *bron@jabbim.cz/prace*.

XML zprávy

Protokol rozlišuje tři druhy takovýchto zpráv a každá má svůj význam.

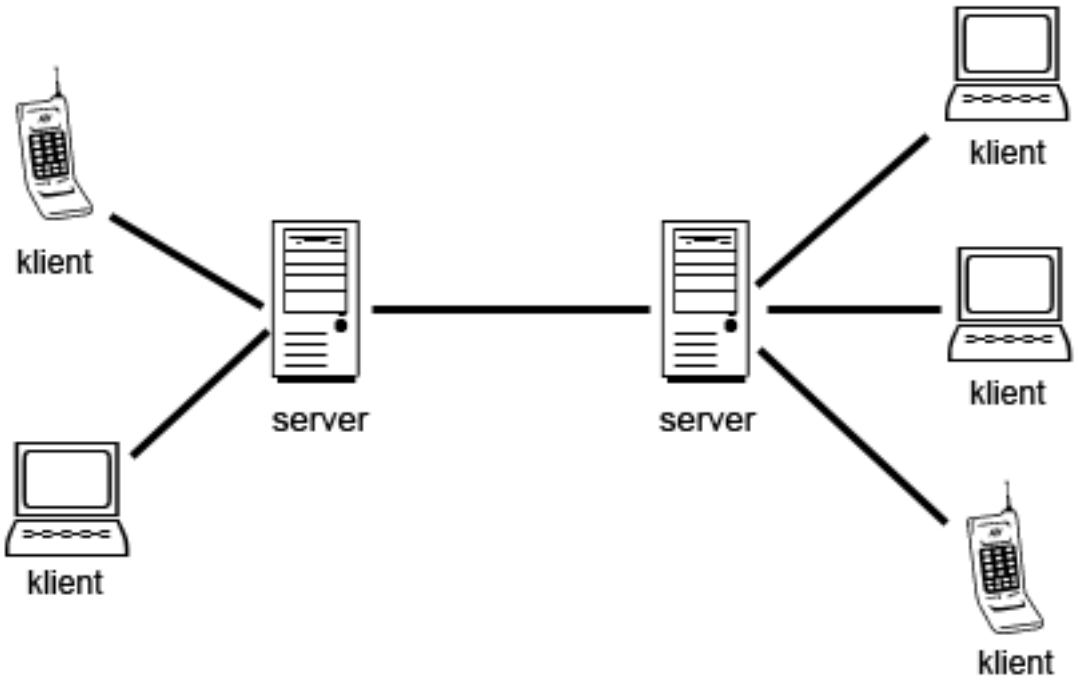
- *presence* – tento typ zprávy nese informaci o dostupnosti

- *message* – pomocí tohoto typu zpráv se přenáší běžná komunikace
- *iq* – iq je zkratka z anglického *dotaz/odpověď* tohoto je využíváno například při použití *Ad-Hoc Commands*

Nástin možné komunikace

Pokud chce uživatel poslat zprávu někomu, kdo je připojen ke stejnému serveru, komunikace proběhne zjednodušeně takto: *Klient − > Server (společný) − > Klient (uživatele, kterému je napsáno).*

Nejsou-li uživatelé připojeni ke stejnému serveru a budeme-li uvažovat případ, jako je na obrázku 2.2, bude to vypadat asi takto *Klient − > Server (ke kterému je připojen odesílatel) − > Server (ke kterému je připojen příjemce zprávy) − > Klient (uživatele, kterému je napsáno).*



Obrázek 2.2: Schéma možné sítě

2.3.4 Rozšíření

Rozšíření nebo taky *XEP*–XMPP Extension Protocols [22] (dříve JEP) rozšiřují vlastnosti samotného protokolu XMPP. Mnohá rozšíření jsou standardizována a některá jsou stále ve vývoji. Dále uvádím příklady XEP, které jsem při vypracovávání použil.

Data Forms XEP-0004

Tento dokument popisuje, jak vytvářet a jak pracovat s formuláři. Dále popisuje jejich strukturu a jednotlivé prvky. Formuláře se dají využít například pro ovládání jiného klienta, robota, posílání informací atd. Podrobný popis struktury a možností formulářů viz [5]. V naší aplikaci jsou formuláře spojeny s *Ad-Hoc Commands*.

Ad-Hoc Commands XEP-0050

Ad-Hoc Commands umožnují pro komunikaci využít *formuláře* 2.3.4, což usnaďňuje komunikaci. V aplikaci je toto využito například při práci s poznámkou, událostí ... Uživatel nemusí zadávat textový příkaz, ale může využít již připravený formulář. Tento formulář mu robot pošle jako typ zprávy (`iq`). Uživatel jej vyplní, pošle zpět robotovi a ten jej podle potřeby vyhodnotí. Podrobný popis i s ukázkami takových zpráv viz [6], nebo v dodatku *A Obrázky*

2.4 SQLite

> *SQLite je relační databázový systém, obsažený v relativně malé knihovně napsané v C. Je vyvíjen D. Richardem Hippem a šířen pod licencí public domain.* [18]

Každá databáze je uložena v samostatném souboru `.dbm`. SQLite implementuje téměř celý standard *SQL-92*. Další předností SQLite je její rychlost, ve většině případů je mnohonásobně rychlejší než PostgreSQL a MySQL.

Díky této zkutečnosti a malé velikosti databázového systému, umožňuje využít SQLite v různých aplikacích a na různých platformách. SQLite je využívána firmami, jako jsou *Apple*, ve svých počítačích (např. internetový prohlížeč Safari, emailový klient Apple Mail) nebo přenosných zařízeních (iPhone, iPod Touch). Dále tuto databázi využívá společnost *Google* ve svém operačním systému *Android* pro mobilní telefóny a mnoho jiných společností viz [21].
Domovské stránky SQLite [19]

2.5 IMAP

IMAP nebo-li *Internet Message Access Protocol* je protokol sloužící pro příjem elektronické pošty. V dnešní době se využívá čtvrtá verze s první revizí nebo-li *IMAP4rev1*.

Narozdíl od protokolu POP ponechává zprávy na serveru a umožnuje používat více klientů najednou. Dále na serveru podporuje práci se složkami.

Podrobnou původní specifikaci protokolu IMAP4 nalezneme v RFC 1730 [11], aktuální verzi po první revizi nalezneme v RFC 3501 [12]

2.6 SMTP

SMTP nebo-li *Simple Mail Transfer Protocol* je protokol určený pro přenos elektronické pošty.

Specifikace původní normy je z roku 1982 v RFC 821 [4], novější specifikace se nachází v dokumentu RFC 2821 [3] z roku 2001.

2.7 Twisted

Twisted je knihovna napsána v jazyce python a vychází z knihovny *XMPPPY*. Podporuje řadu internetových protokolů. Ať je to práce s html, elektronickou poštou nebo instant messagingem. [1] Umožňuje uživateli programovat servery i klinety jednotlivých protokolů. Také poskytuje podporu pro pohodlnou práci s XML a řadu dalších možností, které by jinak programátor musel sám zdlouhavě programovat.

2.7.1 Words

Words je modul *Twisted* pro podporu instant messagingu. *Words* podporuje mnoho protokolů jako je například OSCAR, XMPP, IRC ...

Domish

Tato část modulu Words je určena pro práci s XML, což je pro nás velmi užitečné, protože protokol XMPP je vystavěn nad XML.

Protocols

Pro práci s XMPP se nám bude hodit část *protocols*, konkrétně *jabber*.

2.7.2 Internet

Tento modul využijeme pro navázání spojení s naším jabber serverem.
Domovská stránka Twisted viz [20]

Kapitola 3

Návrh

V této kapitole si popíšeme typický příklad použití takového robota. Z toho dále odvodíme, jaké funkce by měl takový robot poskytovat.

3.1 Obecný příklad použití

Předpokládejme, že uživatel má již našeho robota přidaného mezi ostatními kontakty. Nyní se rozhodne využít jeho služeb. Komfort použití záleží na tom, jakého klienta uživatel používá. Pokud jeho klient podporuje takzvané *Ad-Hoc Commands*, komfort využití robota je větší.

Nyní si nastíníme příklad, kdy *klient umožnuje práci s Ad-Hoc Commands*. Uživatel si vyžádá seznam nabízených funkcí, typ této zprávy bude *iq*. Robot mu odpoví opět zprávou typu *iq*, ve které jsou vypsány funkce, které poskytuje. Uživatel si vybere tu, kterou chce nyní použít a potvrdí svou volbu. Robot přijme tuto zprávu, rozpozná, že se jedná o zprávu typu *iq* a dále ji zpracuje tak, aby poznal, o jaký požadavek uživatele se jedná. Následně odpoví zprávou *iq*, která obsahuje podobu formuláře pro danou službu. Uživatel tento formulář vyplní a odešle jej zpět robotovi. Ten tuto zprávu analyzuje a podle výsledku s ní naloží.

Tentokrát si ukážeme, jaká bude situace, pokud uživatelův *klient nepodporuje Ad-Hoc Commands*, nebo uživatel chce využít textové rozhraní robota. Pokud si uživatel nepamatuje formát příkazu, který chce právě využít, stačí napsat příkaz pro vypsání nápovědy. Robot mu již odpoví seznamem příkazů spolu s jejich příklady použití, kterým rozumí. Uživatel tedy už ví, jaký příkaz a jak jej použít. Napíše robotovi klasickou zprávu (typ *message*), která bude mít formát požadovaný robotem, pro vykonání dané funkce. Robot tuto zprávu přijme, zpracuje ji, vykoná požadovanou funkci a uživateli nazpět pošle zprávu s výsledkem vykonané funkce. Pokud se robotovi funkci nepodaří provést, odpoví zprávou, proč se tomu tak zřejmě stalo.

Pomineme, jakým způsobem robot požadavek přijal a zaměříme se na konkrétní případ použití. Představme si, že uživatel je v práci a vzpomene si, že po práci musí zajít do obchodu. Nyní může využít hned všech služeb, které robot bude poskytovat. Protože je teprve poledne a do konce pracovní doby má daleko, rozhodne se, že si toto poznamená jako událost do kalendáře našeho robota a ta se mu připomene půl hodiny před odchodem z práce. Dále může pomocí robota odeslat manželce email, jestli má nějaké speciální přání a přijmout její odpověď. Jednotlivé položky seznamu si může uložit jako poznámky.

3.2 Seznam základních funkcí a požadavků

V této části navrhneme, jakou funkčnost budeme po robotovi požadovat podle předchozího *obecného případu použití*

- robot musí umět přijímat požadavky formou zprávy a také jako Ad-Hoc Commands, obojí proto, že né každý klient umí pracovat s Ad-Hoc Commands
- z předchozího bodu vyplývá, že budeme muset vytvořit funkci pro rozeznání o jaký požadavek se jedná
- poslat/přijmout email v době, kdy nemáme po ruce svého standardního emailového klienta, což je u mobilních telefonů zatím časté
- funkce pro komunikaci s mail servery
- uložení poznámky či události pro její pozdější připomenutí
- odstranění záznamu po ukončení doby platnosti dané poznámky – není žádoucí, aby zabírala místo a dělala seznam nepřehledným
- hlavní funkce pro příjem zpráv a rozhodování, co s nimi
- reakce na zprávu typu *message*, kdy uživatel poslal příkaz jako běžnou zprávu
- reakce na zprávu typu *iq*, pokud uživatel použil Ad-Hoc Commands, umět na ní reagovat a pracovat s ní
- nekonečná smyčka pro kontrolu aktuálních událostí, tato smyčka bude běžet v samostaném vlákně a kontrolovat aktualní události

Kapitola 4

Implementace

Když už máme představu, jak by měl program fungovat podle předešlé kapitoly *Návrh*, pojďme se podívat na konkrétní implementaci.

V kapitole se nacházejí tyto části

- Hlavní funkce tvořící robota
- Popis funkcí vykonávaných v rámci vláken
- Práce s databází
- Implementované funkce robota

4.1 Hlavní funkce robota

Po provedení nejnutnějších funkcí pro přihlášení k Jabber serveru se celý program rozdělí na dvě vlákna. To z toho důvodu, aby byl schopný v jeden okamžik přijímat požadavky klienta a zároveň kontrolovat co půl hodiny databázi, zda je aktuální některá z událostí v kalendáři. Kalendářem se myslí *tabulka Kalendar* v databázi.

4.2 Popis funkcí vykonávaných v rámci vláken

4.2.1 Vlákno pro kontrolu událostí v kalendáři

Toto vlákno obsahuje nekonečnou smyčku, ve které se vždy po půl hodině vytvoří SQL dotaz pro tabulku `kalendar`, zda existuje nějaká událost, která ještě nebyla splněna (neproběhlo upozornění) a její čas připomenutí je v rozmezí plus mínus aktuální půlhodinu. Pokud se taková událost v databázi vyskytuje, změní se její příznak z nesplněno na splněno. Tyto dvě hodnoty v databázi reprezentují jednoduché hodnoty *0* a *1*. Dále se vytvoří XML zpráva typu *message*, jejíž obsah tvoří detaily aktuální události. Tato zpráva je odeslána uživateli a událost je považována za splněnou.

Pokud při dotazu není nalezena v databázi žádná aktuální událost, nic se neděje a uživatel není rušen. Program v tomto vlákně opět počká půl hodiny, než provede další dotaz.

4.2.2 Vlákno reagující na příchozí zprávy

V tomto vlákně se odehrává čekání na příchozí zprávy a následné reakce na ně. Jako první po příchodu nové zprávy, je zjištěno, jakého je typu, zda *message* nebo *iq*. Podle toho se různí další akce.

Příchozí zpráva typu *message*

Při příchodu zprávy typu *message* se dále zjišťuje, zda tělo zprávy obsahuje některý ze známých příkazů, kterému robot rozumí. Neobsahuje-li zpráva známý příkaz, robot odpoví stejně, jako na příkaz `!help`. Tedy odpoví seznamem příkazů, kterým rozumí spolu s jejich příklady použití. Rozumí-li příkazu v příchozí zprávě, provede tomuto příkazu přiřazenou funkci a o výsledku informuje uživatele.

Příchozí zpráva typu *iq*

Pokud robot obdrží zprávu typu *iq*, začne zjišťovat, co je obsahem atributu `type`.

Je-li obsahem `get`, ví, že uživatel žádá seznam poskytovaných služeb pomocí *Ad-Hoc Commands*. Vytvoří XML zprávu typu *iq* a jejím obsahem bude seznam nabízených funkcí pomocí Ad-Hoc Commands v podobě formuláře. Obsah atributu `type` bude nyní `result`.

Pokud obsahem atributu `type` není `get`, zaměří se robot na element `command` a jeho atribut `node`. Podle tohoto obsahu ví, jakou odpověď/formulář má uživateli poslat zpět.

To, že došlo k vykonání celého cyklu vybrané funkce, označí robot v elementu `command` a jeho atributu `status` obsahem `completed`.

Pro práci s XML je využívána knihovna *Twisted* 2.7.

4.3 Databáze

Robot využívá databázi k ukládání dat od uživatele. Mezi taková data patří informace o poznámce, události v kalendáři a informace o mail serverech. Databáze obsahuje tři tabulky.

- *poznámky* – tato tabulka je učena k uchování informací o poznámkach uživatele
- *kalendar* –v této tabulce se nachází informace o událostech
- *mailServer* – tato tabulka uchovává detaily o jednotlivých mail serverech

4.3.1 Uložení *poznámky*

Pokud uživatele napadne nějaká poznámka a chce si ji poznamenat, nejlépe někam, kde k ní bude mít pokaždé přístup, nabízí se právě naše řešení v podobě robota.

Po zadání jednoduchého příkazu 4.4.4 nebo vyplnění formuláře (Ad-Hoc Commands) pro uložení poznámky, robot tuto zprávu zpracuje a poznámku uloží.

Poznámky jsou ukládány do tabulky *poznamky* Tato tabulka má dva sloupce.

Položky tabulky *poznamky*

- *rowid* – jednoznačný identifikátor označující řádek v tabulce, jeho hodnota je automaticky nastavována samotnou databází
- *note* – tento sloupec obsahuje samotný text poznámky

4.3.2 Uložení kalendářní *události*

Nechce-li uživatel promeškat nějakou událost, zapíše si ji do kalendáře a nemusí na ni již myslet. Stačí, když vyplní její základní údaje a o zbytek se postará robot sám. Zadání může opět proběhnout buď pomocí obyčejné zprávy a nebo pomocí Ad-Hoc Commands.

Položky tabulky *kalendar*

Tato tabulka obsahuje, na rozdíl od předchozí, již více údajů, které slouží robotovi pro správné zacházení s událostí.

- *rowid* – jednoznačný identifikátor označující řádek v tabulce, jeho hodnota je automaticky nastavována samotnou databází
- *datum* – datum uplatnění události
- *cas* – čas, kdy se událost odehraje
- *ucas* – tato položka reprezentuje počet sekund od 1.1.1970 a je vypočtena robotem, uživatel ji nezadává
- *text* – zde je prostor pro poznámku k události
- *hotovo* – tento sloupec slouží robotovi k rozeznání, zda již byla událost uživateli oznámena nebo ne

4.3.3 Uložení detailu o mail serverech

Uložení těchto informací zrychluje a zvyšuje komfort využití služby *stažení nových emailů*. Uživatel, před prvním stažením nových emailů, pošle robotovi příkaz pro uložení nového emailového serveru do databáze spolu s detaily pro spojení s ním.

Pokud jsou informace o emailovém serveru uloženy, uživatel zadá robotovi jen svou emailovou adresu a heslo k ní (zde je prostor pro další vylepšení 4.5). Pokud bude příkaz poslán jako zpráva typu *message*, budou uplatněny regulární výrazy a za pomocí výsledků už robot sám z dat v databázi rozpozná, kterému serveru poslat dotaz a jaké budou parametry spojení.

Položky tabulky *mailServer*

V této tabulce jsou uchovávany detaily pro jednotlivé mail servery.

- *rowid* – jednoznačný identifikátor označující řádek v tabulce, jeho hodnota je automaticky nastavována samotnou databází
- *doména* – nachází se v emailové adrese za znakem `@` například *gmail.com*

- *server* – adresa serveru, kde beží služby IMAP nebo POP například *imap.gmail.com*
- *protkol* – jaký protkol bude použit pro komunikaci s email server IMAP nebo POP
- *port* – číslo portu, na kterém služba běží
- *SSL* – zda bude spojení šifrováno nebo nikoliv

4.4 Implementované funkce robota

Nyní si popíšeme implementované funkce, které robot umožnuje vykonat.

4.4.1 Práce s poznámkou

- *uložení* – jestliže Vás napadla poznámka k právě řešenému problému, díky této funkci si ji můžete poznamenat a kdykoliv se k ní vrátit
- *výpis* – pokud jste si poznamenali nějaký nápad, můžete si pomocí této funkce přečíst o čem byl
- *odstranění* – skončili jste práci na určitém úkolu a již nechcete, aby Vás pletly neaktuální poznámky, tak je smažete

4.4.2 Práce s událostí

S události máte stejné možnosti práce jako s poznámkou. Robot však musí obsahovat navíc funkci, která se stará o ohlašování aktualních událostí a práci s nimi.

Popis funkce hlídající aktuálnost událostí

Tato funkce kontroluje co půl hodiny události v databázi, zda je některá aktuální. Při implementaci této funkce vyvstaly otázky, jak zařídit, aby funkce byla univerzální. Nebyla závislá na dni v měsíci, na čase v rámci dne a letopočtu.

Problém nastal v případě, že aktuální čas je takový, že při přičtení nebo odečtení půlhodiny se mění datum. Musela by probíhat kontrola, zda je datum v rozmezí aktuálního měsíce případně jestli se nejedná o přestupný rok a jak toto všechno automatizovat.

Po vyzkoušení několika způsobů jsem se dostal k řešení, které je univerzální a implementačně prosté. Toto řešení převádí aktualní datum a čas na počet sekund od data 1.1.1970. To samé se děje s datem a časem události ukládané do databáze. Potom je již jednoduché přičíst/odečíst půl hodiny od tohoto údaje.

4.4.3 Elektronická pošta a práce s ní

Né vždy, když si vzpomenete, že čekáte důležitý email nebo že chcete někomu něco připomenout, máte vedle sebe zařízení s plnohodnotným emailovým klientem. V případně mobilního telefonu nechcete čekat a platit za načtení webového rozhraní Vaší emailové schránky.

V tomto případě můžete využít emailových služeb robota. Robot zvládá příjem elektronické pošty pomocí dvou nejrozšířenějších protkolů IMAP4 a POP3. Dále lze za jeho pomoci email také odeslat (protokolem SMTP).

Příjmání elektronické pošty

Jak již bylo zmíněno, příjem elektronické pošty je zajištěn protkoly IMAP4 a POP3.

Aby byla práce s robotem a příjmáním elektronické pošty co nejpohodlnější, je potřeba do databáze uložit základní údaje o mail serverech. To může být provedeno při vytvoření databáze samotné skriptem `myCreateDb.py` viz *dodatek C* nebo až za běhu, příslušným příkazem viz 4.4.4.

Existuje-li již o mail serveru záznam v databázi, stačí robotovi poslat příkaz pro stažení pošty. Příkaz může být zadán pomocí formuláře (*Ad-Hoc Commands*) nebo zaslán jako zpráva v patřičném formátu. Po přijetí takového příkazu robot pomocí regulárních výrazů zjistí *doménu* z emailové adresy. Dále provede dotaz na databázi do tabulky `mailServer` na shodu domény s již uloženými záznamy. Jestliže najde shodu, použije data tohoto záznamu pro spojení s mail serverem. Uživatel tedy nemusí pokaždé zadávat adresu serveru, o jaký protkol a na jakém portu se jedná, případně jestli je spojení šifrováno či nikoliv.

Neexistuje-li záznam o mail serveru, musí jej uživatel vytvořit. Příkaz pro tento úkol viz 4.4.4.

Odesílání elektronické pošty

Odeslání elektronické pošty lze provést opět dvěma způsoby. Pomocí formuláře nebo formou zaslání zprávy. Robot oddělí jednotlivé části takové zprávy pomocí regulárního výrazu a dále využije podle potřeby. Formát takové zprávy viz 4.4.4.

Odesílání je zajištěno protokolem SMTP.

4.4.4 Příkazy

Pokud nepoužijete nebo nebudete moci použít k zadání příkazu *Ad-Hoc commands*, budete muset robota ovládat pomocí klasických zpráv. Pro tento případ se Vám bude hodit seznam příkazů (a příkládů pozužití), kterým robot rozumí.

Příkazy pro práci s *poznámkami*

- **!note** – Seznam uložených poznámek spolu s jejich ID
- **!nnote** – Uložení nové poznámky
  `!nnote Text nové poznámky`
- **!dnote** – Odstranění poznámky
  `!dnote ID` – ID je identifikátor, který byl poznámce přiřazen databází. Je zobrazován při výpisu uložených poznámek.

Příkazy pro práci s *událostmi v kalendáři*

- **!date** – Vypíše všechny uložené události spolu s jejich detaily
- **!ndate** – Uložení nové události
  `!nnote #DD.MM.YYYY #HH.MM #Text nové události`
- **!ddate** – Odstranění události
  `!ddate ID` – ID je identifikátor, který byl události přiřazen databází. Je zobrazován při výpisu uložených událostí.

Příkazy pro práci s *elektronickou poštou*

- **!listserver** – Vypíše všechny domény, pro které je uložena adresa mail serveru
- **!mail** – Odeslání nového emailu
  `!mail #prijemce #odesilatel #Předmět #Tělo emailové zprávy`
- **!smail** – Stažení nepřečtených emailů
  `!smail #emailovaAdresa #heslo`

Jiné

- **!q** – Ukončení běhu robota.

4.5 Další možné směry vývoje

V rámci této práce se mi podařilo implementovat robota s funkcemi, které bych popsal jako základní, né však neužitečné a triviální. Robot má tak dobrý základ pro další využití. Na těchto funkcích se dá dál stavět a rozvíjet schopnosti robota tak, aby se stal například plnohodnotným GTD nástrojem podle [2].

4.5.1 Jednotlivá možná vylepšení

- automatické čtení statusu uživatele a doplňování postupu na jednotlivých úkolech
- ukládání vybraných emailů do databáze pro jejich následné čtení bez potřeby spojení s emailových serverem
- propojení například s kalendářem společnosti Google
- automatická kontrola nově příchozích emailů po předem stanoveném čase
- práce s událostí či poznámkou (vyhledávání, řazení ...) podle různých atributů například kategorie, datum aj.
- vytvořit co neinteraktivnější rozhraní v rámci možností formulářů Ad-Hoc Commands
- příjem souborů
- sdílení událostí a poznámek s jinými uživateli
- účast v konferencích

Kapitola 5

Závěr

Již před začátkem této bakalářské práce mě zajímal protokol XMPP a také programovací jazyk Python. Bohužel jsem nenašel čas a záminku věnovat se této oblasti blíže. Až díky této práci se mi podařilo obě oblasti spojit v jednom problému, který mě bavilo řešit a postupovat stále kupředu.

Výsledkem mé bakalářské práce je funkční robot pro správu osobních informací. Poskytuje dobrý základ pro další možná vylepšení, jak po stránce funkční, tak ovládání. Dobře zvolený programovací jazyk *Python* přispěl k rychlému vývoji robota i přesto, že jsem s tímto jazykem neměl z dřívějška žádné zkušenosti. Jako výstup z této práce jako celku, si odnáším porozumění protokolu XMPP, programátorské zkušenosti s jazykem Python a věřím, že taky dobrý programový základ, na kterém budu moci dále stavět.

Protože mě zajímá problematika plánování osobního a pracovního času, doufám, že se mi podaří vylepšit stávajícího robota pro tuto oblast tak, aby byl použitelný nejen pro mě, ale i pro běžné uživatele. Mezi taková vylepšení patří například vytvoření co nejpřívětivějšího prostředí pro uživatele pomocí základních prvků formuláře, automatizované čtení stavu uživatele, jeho vyhodnocení, vyznačení míry úspěšnosti řešení daného problému a dále příjímání souboru s popisem problému, aby tak byl ušetřen čas, ztracený jeho přepisováním a další viz *Další možné směry vývoje* 4.5.

Literatura

[1] Abe Fettig, G. L.: *Twisted Network Programming Essentials*. O'Reilly, 2005, iSBN 0596100329.

[2] Geoffrey M. Bellman: *Getting things done when you are not in charge*. Berrett-Koehler Publishers, 2001, 156 s., iSBN 1576751724.

[3] Information Sciences Institute University of Southern California: RFC 2821 - Simple Mail Transfer Protocol. 1982, [online], [cit. 23. 05. 2009].
URL http://tools.ietf.org/html/rfc2821

[4] J. Klensin: RFC 821 - Simple Mail Transfer Protocol. 2001, [online], [cit. 23. 05. 2009].
URL http://tools.ietf.org/html/rfc821

[5] Kolektiv autorů: XEP-0004: Data Forms. [online], [cit. 23. 05. 2009].
URL http://xmpp.org/extensions/xep-0004.html

[6] Kolektiv autorů: XEP-0050: Ad-Hoc Commands. [online], [cit. 23. 05. 2009].
URL http://xmpp.org/extensions/xep-0050.html

[7] Kolektiv autorů: RFC 3920 - Extensible Messaging and Presence Protocol (XMPP): Core. 2004, [online], [cit. 23. 05. 2009].
URL http://tools.ietf.org/html/rfc3920

[8] Kolektiv autorů: RFC 3921 - Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. 2004, [online], [cit. 23. 05. 2009].
URL http://tools.ietf.org/html/rfc3921

[9] Kolektiv autorů: RFC 3922 - Mapping the Extensible Messaging and Presence Protocol (XMPP) to Common Presence and Instant Messaging (CPIM). 2004, [online], [cit. 23. 05. 2009].
URL http://tools.ietf.org/html/rfc39212

[10] Kolektiv autorů: RFC 3923 - End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP). 2004, [online], [cit. 23. 05. 2009].
URL http://tools.ietf.org/html/rfc3923

[11] M. Crispin: RFC - 1730 INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4. 1994, [online], [cit. 23. 05. 2009].
URL http://tools.ietf.org/html/rfc1730

[12] M. Crispin: RFC 1730 - INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1. 2003, [online], [cit. 23. 05. 2009].
URL http://tools.ietf.org/html/rfc3501

[13] The World Wide Web Consortium: The World Wide Web Consortium. [online], [cit. 23. 05. 2009].
URL http://www.w3c.org/

[14] Wikipedia: Instant messaging — Wikipedia, The Free Encyclopedia. 2009, [online], [cit. 23. 05. 2009].
URL http://en.wikipedia.org/w/index.php?title=Instant_messaging&oldid=292002469

[15] Wikipedie: Extensible Markup Language — Wikipedie: Otevřená encyklopedie. 2009, [online], [cit. 23. 05. 2009].
URL http://cs.wikipedia.org/w/index.php?title=Extensible_Markup_Language&oldid=3982274

[16] Wikipedie: Extensible Messaging and Presence Protocol — Wikipedie: Otevřená encyklopedie. 2009, [online], [cit. 23. 05. 2009].
URL http://cs.wikipedia.org/w/index.php?title=Extensible_Messaging_and_Presence_Protocol&oldid=3472813

[17] Wikipedie: Instant messaging — Wikipedie: Otevřená encyklopedie. 2009, [online], [cit. 23. 05. 2009].
URL http://cs.wikipedia.org/w/index.php?title=Instant_messaging&oldid=3962659

[18] Wikipedie: SQLite — Wikipedie: Otevřená encyklopedie. 2009, [online], [cit. 23. 05. 2009].
URL http://cs.wikipedia.org/w/index.php?title=SQLite&oldid=3749587

[19] WWW stránky: SQLite Home Page. [online], [cit. 23. 05. 2009].
URL http://www.sqlite.org/

[20] WWW stránky: Twisted. [online], [cit. 22. 05. 2009].
URL http://twistedmatrix.com/

[21] WWW stránky: Well-Known Users of SQLite. [online], [cit. 23. 05. 2009].
URL http://www.sqlite.org/famous.html

[22] WWW stránky: XMPP Extensions. [online], [cit. 22. 05. 2009].
URL http://xmpp.org/extensions/

Dodatek A

Obrázky

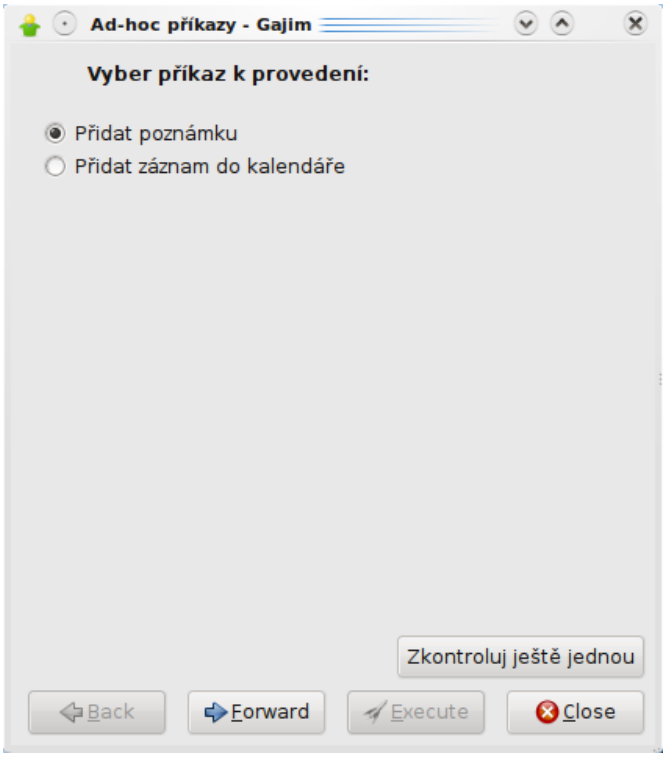
A.1 Zadávání příkazu pomocí *Ad-Hoc Commands*

Modře jsou vyznačeny zprávy robota a červeně odpovědi uživatele.

```
<iq to="bron@jabbim.cz/klon" type="get" id="86">
<query xmlns="http://jabber.org/protocol/disco#items" node="http://jabber.org/
protocol/commands" />
</iq>

<iq from='bron@jabbim.cz/klon' to='gerf@jabbim.cz/Gajim' xml:lang='cs' type='re
sult' id='86'>
<query xmlns='http://jabber.org/protocol/disco#items' node='http://jabber.org/pr
otocol/commands'>
<item node='note' name='Přidat poznámku'/>
<item node='date' name='Přidat záznam do kalendáře'/>
</query>
</iq>
```

Obrázek A.1: XML stream pro získání nabídky

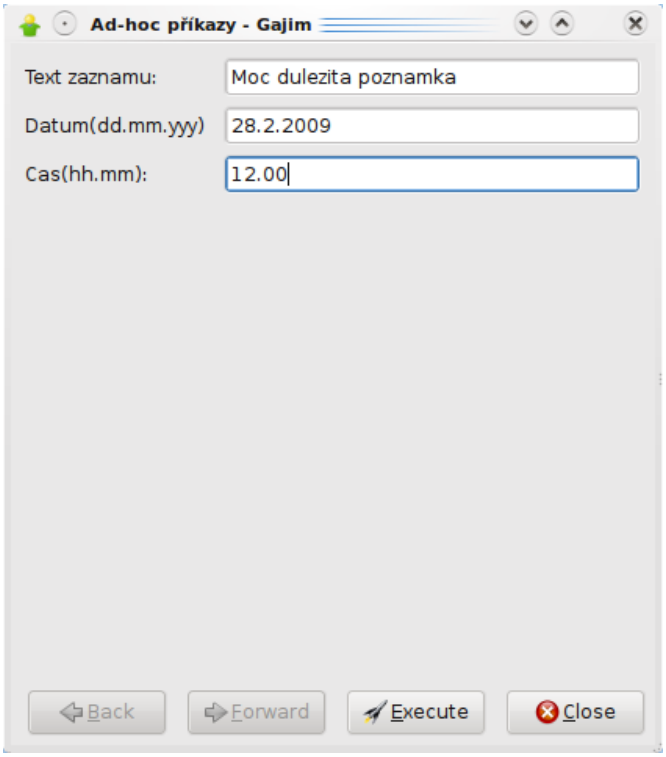


Obrázek A.2: Podoba nabídky vrácené robotem

```
<iq to="bron@jabbim.cz/klon" type="set" id="107">
<command node="date" action="execute" xmlns="http://jabber.org/protocol/co
mmands" />
</iq>

<iq from='bron@jabbim.cz/klon' to='gerf@jabbim.cz/Gajim' xml:lang='cs' type='re
sult' id='107'>
<command xmlns='http://jabber.org/protocol/commands' node='addDate' status
='executing'>
<actions execute='complete'>
<complete/>
</actions>
<x xmlns='jabber:x:data' type='form'>
<field var='dateText' type='text-single' label='Text zaznamu: '>
<value/>
</field>
<field var='dateDate' type='text-single' label='Datum(dd.mm.yyy)'/>
<field var='dateTime' type='text-single' label='Cas(hh.mm): '/>
</x>
</command>
</iq>
```

Obrázek A.3: XML stream zobrazení formuláře pro zadání údajů

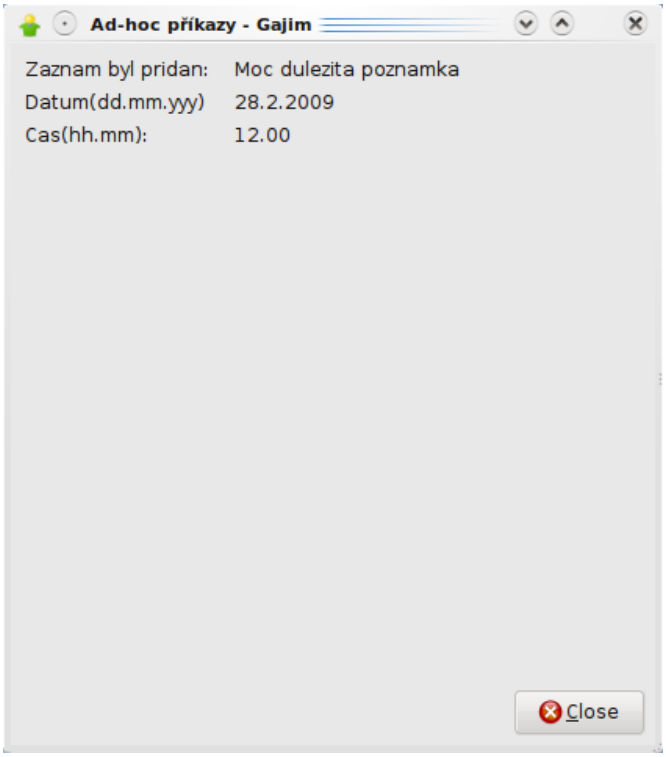


Obrázek A.4: Podoba formuláře poslaného robotem

```
<iq to="bron@jabbim.cz/klon" type="set" id="122">
<command node="addDate" action="execute" xmlns="http://jabber.org/protoco
l/commands">
<x xmlns="jabber:x:data" type="submit">
<field var="dateText" type="text-single" label="Text zaznamu: ">
<value>Moc dulezita udalost</value>
</field>
<field var="dateDate" type="text-single" label="Datum(dd.mm.yyy)">
<value>28.2.2009</value>
</field>
<field var="dateTime" type="text-single" label="Cas(hh.mm): ">
<value>12.00</value>
</field>
</x>
</command>
</iq>

<iq from='bron@jabbim.cz/klon' to='gerf@jabbim.cz/Gajim' xml:lang='cs' type='s
et' id='122'>
<command xmlns='http://jabber.org/protocol/commands' node='addDate' action
='execute' status='completed'>
<x xmlns='jabber:x:data' type='result'>
<field var='dateText' type='text-single' label='Zaznam byl pridan: '>
<value>Moc dulezita udalost</value>
</field>
<field var='dateDate' type='text-single' label='Datum(dd.mm.yyy)'>
<value>28.2.2009</value>
</field>
<field var='dateTime' type='text-single' label='Cas(hh.mm): '>
<value>12.00</value>
</field>
</x>
</command>
</iq>
```

Obrázek A.5: XML stream pro zrekapitulování uložených informací



Obrázek A.6: Podoba vrácené informace o uložených datech

Dodatek B

Uživatelská příručka

B.1 Instalace

Pro úspěšné spuštění robota na školním serveru *Merlin* stačí překopírovat celý obsah adresáře **bron** z přiloženého CD na Vámi určené místo na *Merlinovi*.

Před prvním spuštením skriptu `bron.py` je nutné spustit skript `myCreateDb.py`, který vytvoří databázi `MYDB` pro správnou funkci robota.

Pokud budete chtít robota spouštět mimo server Merlin, budete muset zajisti, aby na zařízení byla nainstalována databáze SQLite a knihovny nutné ke správnému běhu robota.

B.2 Spuštění

Jste-li v adresáři kde se nachází soubor `bron.py` stačí pro spuštění zadat:

```
python bron.py JID
```

jako JID zadejte JID, na který mají být doručovány reakce robota

B.3 Komunikace s robotem

Abyste mohli s robotem komunikovat, přidejte si jej nejdříve mezi své ostatní kontakty. Robotův *JID* je **bron@jabbim.cz**

Pokud nechete nebo nemůžete pro práci s robotem použít *Ad-Hoc Commands*, pošlete mu zprávu `!help`. On Vám odpoví seznamem příkazů, kterým rozumí a jejich příklady.

Pozn. Seznam příkazů, kterým robot rozumí najdete v kapitole Implementace.

Dodatek C

Obsah CD

- **bron** – adresář obsahuje vše potřebné pro chod robota
  - `bron.py` – hlavní funkčnost robota
  - `myCreateDb.py` – soubor pro vytvoření databáze
  - `mySqlite.py` – funkce pro práci s databází
  - `myMails.py` – funkce pro práci s elektronickou poštou
  - `myIq.py` – funkce obsluhující *iq* zprávy
  - `twisted` – adresář s knihovnou *Twisted*
  - `zope` – adresář s knihovnou *Zope*
- **zprava** – v tomto adresáři se nachází adresáře a soubory potřebné k přeložení technické zprávy a také již přeložený soubor `projekt.pdf`, který obsahuje text technické zprávy
- **readme.txt** – soubor obsahuje základní informace k instalaci, spuštění robota a popis obsahu CD