Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

E-mailový klient pro prohlížeč Otter

2019

Jan Čulík

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

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Jan Čulík
Název práce:	E-mailový klient pro prohlížeč Otter
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2019
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Petr Krajča, Ph.D.
Počet stran:	38
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Jan Čulík
Title:	E-mail client for the Otter browser
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2019
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Petr Krajča, Ph.D.
Page count:	38
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Cílem práce je naprogramovat emailový klient jako modul pro prohlížeč Otter. Funkce tohoto klienta koncepčně vychází z emailového klienta M2 z Opery 12. Mezi funkce tohoto klienta patří podpora protokolů IMAP a SMTP včetně jejich šifrovaných variant, podpora více emailových účtů, HTML e-mailů a práce s přílohami.

Synopsis

The goal of the bachelor thesis is to create a module with e-mail client for Otter Browser. Functionality of this e-mail module follows up the e-mail client M2 from Opera 12. Basic features of this e-mail module are support for IMAP and SMTP protocols with their encrypted variants, support for multiple e-mail accounts, capability to preview HTML e-mails and ability to work with e-mail attachments.

Klíčová slova: E-mail; SMTP; IMAP; Otter Browser

Keywords: E-mail; SMTP; IMAP; Otter Browser

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a 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 zdrojových kódů

1 Vztah Opery a Otter Browseru, cíl bakalářské práce

Otter Browser je open source webový prohlížeč napsaný v programovacím jazyku C++ a ve frameworku Qt 5. Skupina vývojářů kolem projektu Otter se snaží vytvořit prohlížeč, který funkcionalitou a chováním navazuje na prohlížeč Opera ve verzi 12 z roku 2014. Tehdejší Opera měla velké možnosti úprav vzhledu a chování, prohlížeč disponoval funkcionalitou, kterou docenili spíše pokročilejší a náročnější uživatelé. Prohlížeč v sobě obsahoval i zabudovaného e-mailového klienta, bittorent klienta, IRC klienta, RSS čtečku, správce poznámek a další moduly.

Hlavním podnětem pro vznik projektu Otter byly zásadní změny v Opeře, kdy se společnost rozhodla, že veškerý dosavadní kód Opery kvůli složité udržitelnosti a kvůli vysokým nákladům na vývoj vlastního vykreslovacího jádra a javascriptového enginu Presto přestane udržovat a třináctá verze prohlížeče byla napsána od nuly kolem vykreslovacího jádra Webkit. Další vývoj nové Opery se od té doby ubírá odlišným směrem než v předchozích verzích, cílovou skupinou jsou masoví uživatelé. Původní moduly vývojáři zpět do Opery neimplementovali a ani to nemají v plánu. Nová Opera tak přišla i o klienty pro e-mail, IRC, RSS a o možnost výrazněji ovlivnit vzhled a chování prohlížeče uživatelem.

Michał Dutkiewicz proto v roce 2014 začal psát svůj vlastní prohlížeč, který pojmenoval Otter Browser. Jméno bylo zvoleno tak, aby začínalo písmenem O stejně jako Opera. Slovo Otter má taky stejný počet písmen jako slovo Opera. Otter má v angličtině podobnou výslovnost jako slovo *other*, takže v mluvené řeči dochází ke hříčce, kdy se projekt po překladu označuje jako *jiný* nebo *další* prohlížeč. Na vývoji prohlížeče se ve volném čase aktivně podílí 6 lidí, veškeré zdrojové kódy jsou dostupné pod open source licencí GNU GPL v3. Prohlížeč Otter funguje na operačních systémech Microsoft Windows, GNU/Linux, Mac OS X a na různých dalších operačních systémech vycházejících z Unixu.

Cílem projektu Otter není vytvořit přesnou kopii staré Opery, ale vybudovat prohlížeč s podobnou filosofií ovládání a konfigurace. Implementovaná funkcionalita v Otteru v první řadě odráží potřeby vývojářů, kteří za projektem stojí. Otter Browser dnes umí drtivou většinu věcí, které by běžný uživatel od webového prohlížeče čekal. Z větších věcí chybí snad jen podpora WebExtensions. V plánech do budoucna je podpora různých modulů s další funkcionalitou po vzoru staré Opery. V Otter Browseru je už hotový správce poznámek a základní RSS čtečka, ale například e-mailový klient a klienti pro protokoly IRC a Bittorent stále chybí.

Cílem bakalářské práce je do Otter Browseru chybějící modul pro práci s e-maily naprogramovat. Mezi základní věci, které by měl e-mailový klient podporovat je stahování zpráv z poštovního serveru protokolem IMAP, zobrazování textového a HTML obsahu přijatých zpráv, otevírání příloh a možnost zprávy mazat, přesouvat nebo kopírovat napříč složkami v e-mailovém účtu. Všechny destruktivní operace nad zprávami a adresáři by měly změnit stav i na poštovním

serveru. E-mailový klient by měl umět správně rozpoznat a upravit lokální kopii dat, pokud uživatel na svém e-mailovém účtu provede některou destruktivní operaci z jiného e-mailového klienta nebo počítače. E-mailový klient by měl podporovat odesílání zpráv protokolem SMTP. Klient by měl umět posílat zprávy s přílohami, odeslat zprávu většímu počtu příjemců a přidat další příjemce do kopie nebo skryté kopie. Stejně tak by měl klient umět přeposlat zprávu nebo odpovědět na zprávu, jejíž lokální kopii si předtím stáhl protokolem IMAP. Uživatelské rozhraní e-mailového modulu by mělo vycházet z e-mailového klienta M2 z Opery 12, uživatelské rozhraní by mělo být snadno použitelné i pro uživatele jiných tradičních e-mailových klientů, jako jsou například Microsoft Outlook nebo Mozilla Thunderbird.

Stejně jako zdrojové kódy Otter Browseru, i praktická část mé bakalářské práce je volně dostupná pod licencí GNU GPL v3. Zdrojové kódy mé verze Otter Browseru jsou na adrese https://github.com/honza-c/otter-browser.

2 Použité externí knihovny a frameworky

Otter browser je napsaný v programovacím jazyku C++ a ve frameworku Qt 5. Verze prohlížeče s e-mailovým modulem je naprogramovaná v Qt frameworku ve verzi 5.11. Qt framework poskytuje prostředky pro implementaci všech datových struktur, abstrakci nad SQLite databází i potřebné nástroje pro tvorbu grafického uživatelského rozhraní.

V e-mailovém modulu používám dvě knihovny třetí strany. První z knihoven *vmime*[1] je naprogramovaná v C++ a je dostupná pod GNU GPLv3 licencí. Knihovna poskytuje přístup k SMTP, POP3 a IMAP protokolům, potřebné parsery pro získávání informací ze stažených zpráv a generátory pro vytváření zpráv z dat z vlastních datových struktur. V knihovně *vmime* jsou chyby, které znemožňují její plnohodnotné používání s protokoly POP3 a SMTP, v e-mailovém modulu se proto používá jen k práci s IMAP servery.

Druhou externí závislostí e-mailového modulu je knihovna *simple-mail*[2] naprogramovaná v jazyku C++ a frameworku Qt 5, v modulu se používá k odesílání e-mailových zpráv protokolem SMTP. Knihovna *simple-mail* je dostupná pod licencí GNU LGPL 2.1+.

3 Přehled síťových protokolů

Tato kapitola obsahuje stručný přehled síťových protokolů SMTP[3], POP3[4] a IMAP[5] pro práci s elektronickou poštou. Všechny popisované protokoly běží na aplikační vrstvě protokolu IP a používají transportní vrstvu TCP. Veškerá síťová komunikace těmito protokoly včetně přihlašování probíhá nešifrovaně, síťové spojení ale lze zabezpečit použitím SSL nebo TLS.

3.1 SMTP

SMTP (Simple Mail Transfer Protocol) je protokol pro přímý přenos e-mailových zpráv mezi klientem odesílatele a serverem odesílatele, dále se používá pro přenos zpráv mezi serverem odesílatele a serverem příjemce zprávy. Protokol se už nestará o uložení zprávy do schránky příjemce. SMTP standardně komunikuje na TCP portu 25 u nešifrovaného spojení a na portu 465 nebo 587 u spojení šifrovaného. Protokol byl standardizován v roce 1982 (RFC 821[6]), dnes se používá jeho rozšíření Extended SMTP z roku 2008 (RFC 5321[7]).

3.2 POP3

POP3 (Post Office Protocol) je protokol pro stahování e-mailových zpráv ze serveru na klienta. POP3 standardně komunikuje přes port 110 u nešifrovaného spojení a na portu 995 u šifrovaného spojení. Protokol POP3 byl standardizován v roce 1988 v RFC 1939[8], předchůdci POP3 jsou protokoly POP1 a POP2, jejichž historie sahá až do roku 1984. U starších protokolů POP1 a POP2 e-mailoví klienti po připojení k serveru stáhli a uložili zprávy do počítače klienta. Po vyzvednutí zpráv klient zprávy ze serveru vždy smazal. U protokolu POP3 je mazání zpráv ze serveru po jejich vyzvednutí volitelné. Protokol POP3 je ve srovnání s protokolem IMAP výrazně jednodušší. Nepodporuje adresářovou strukturu, příznaky u zpráv (například informace o tom, zda je zpráva přečtena). POP3 taky neumožňuje stahovat zprávy po částech podle MIME[9] struktury, k e-mailovému účtu přes POP3 protokol může být současně připojen maximálně jeden klient. Z těchto důvodů je protokol POP3 pro dnešní potřeby nevyhovující, všechny nedostatky protokolu POP3 jsou navíc vyřešeny v protokolu IMAP.

3.3 IMAP

IMAP (Internet Message Access Protocol) je protokol pro přístup k e-mailové schránce prostřednictvím e-mailového klienta. Nešifrovaný IMAP standardně komunikuje přes port 143, zabezpečená SSL varianta přes port 993. Protokol umožňuje trvalé připojení k e-mailové schránce a práci offline. Všechny zprávy a adresáře jsou uloženy na serveru a klient si stahuje jen ty informace, které opravdu potřebuje. IMAP narozdíl od POP3 umožňuje v rámci e-mailového účtu pracovat s adresářovou strukturou, zprávy mohou mít příznaky (flagy) a k serveru může být připojeno více klientů zároveň. IMAP taky umožňuje prohledávat zprávy na serveru podle různých kritérií bez nutnosti mít poštu staženou lokálně. IMAP od verze 4 podporuje MIME formát zpráv, kdy zprávy můžou mít stromovou strukturu a klient si může stáhnout jen ty části zprávy, které potřebuje (například pouze neformátovaný textový obsah, HTML textový obsah, přílohy nebo vložené objekty). První verze protokolu vznikla v roce 1986 (RFC 1064[10]) a význam zkratky IMAP různě označoval Internet Mail Access Protocol, Interactive Mail Access Protocol nebo Interim Mail Access Protocol. V současnosti se používá IMAP4rev1 z roku 1996 (RFC 3501[11]).

4 Návod na instalaci

Kapitola popisuje instalaci všech závislostí k překladu Otter Browseru s e-mailovým modulem a následně překlad samotného Otter Browseru v prostředí operačního systému GNU/Linux.

4.1 Instalace Qt frameworku a nástrojů pro překlad programů v C++

Pro úspěšný překlad Otter Browseru je potřeba mít vývojové knihovny Qt frameworku zkompilované proti stejné verzi knihovny OpenSSL, kterou má uživatel v systému. Univerzální balíčky pro Linux z webových stránek Qt frameworku nebudou fungovat. Vývojové knihovny Qt frameworku s podporou knihovny OpenSSL ale jsou součástí repozitářů drtivé většiny linuxových distribucí, takže není potřeba, aby si uživatel musel překládat celý framework sám. Instalaci Qt frameworku a vývojových knihoven pro C++ v linuxových distribucích z rodiny Red Hat z terminálu popisuje zdrojový kód 1, instalaci závislostí na systémech vycházejících z distribuce Debian z terminálu popisuje zdrojový kód 2.

```
1 sudo dnf install cmake? libgsasl libgsasl-devel gnutls-devel openssl
     -devel sendmail mesa-libGL-devel
2 sudo dnf install @development-tools
3 sudo dnf group install "C Development Tools and Libraries"
4 sudo dnf install qt5*-devel
5 sudo dnf install qt-creator
```

Zdrojový kód 1: Instalace Qt frameworku a C++ knihoven v linuxových distribucích z rodiny Red Hat

```
1 sudo apt-get install build-essential libgl1-mesa-dev libgsasl7-dev
     gsasl libgnutls28-dev libssl-dev sendmail
2 sudo apt-get install apt-get install qtcreator qt5*-dev
```

Zdrojový kód 2: Instalace Qt frameworku a C++ knihoven v systémech vycházejících z distribuce Debian

4.1.1 Překlad a instalace knihovny vmime

Překlad ze zdrojových kódů a instalaci knihovny *vmime* do systému z prostředí terminálu popisuje zdrojový kód 3.

```
wget https://github.com/kisli/vmime/archive/v0.9.2.tar.gz
tar -xf v0.9.2.tar.gz
mkdir vmime-0.9.2/build
cd vmime-0.9.2/build
cmake -DVMIME_BUILD_SAMPLES=OFF -DVMIME_SHARED_PTR_USE_BOOST=OFF -
    DVMIME_SHARED_PTR_USE_CXX=ON -DVMIME_SENDMAIL_PATH=/run/sendmail
     -DCMAKE_BUILD_TYPE=RELEASE ..
make
sudo make install
sudo ldconfig /usr/local/lib64
```

Zdrojový kód 3: Překlad a instalace knihovny vmime

4.2 Překlad knihovny simple-mail

Překlad knihovny *simple-mail* popisuje zdrojový kód 4. Po překladu v adresáři *build/src* vzniknout tři soubory s příponou *so*. Ty je nutné zkopírovat do kořenového adresáře s repozitářem Otter Browseru.

```
 git clone https://github.com/cutelyst/simple-mail
 cd simple-mail
 mkdir build
 cd build
 cmake .. -DCMAKE_BUILD_TYPE=Release -DDISABLE_MAINTAINER_CFLAGS=
    off
 cmake --build . --config Release
```

Zdrojový kód 4: Instalace knihovny simple-mail

4.3 Překlad a spuštění Otter Browseru

Získání zdrojových kódů mé verze Otter Browseru s e-mailovým modulem v prostředí terminálu popisuje zdrojový kód 5. Po zkopírování artefaktů knihovny *simple-mail* z předchozího kroku do projektu stačí otevřít soubor *otter-browser/CMakeLists.txt* ve vývojovém prostředí QtCreator a zde program spustit z hlavního menu *Build* kliknutím na položku *Run*.

```
 git clone https://github.com/honza-c/otter-browser
```

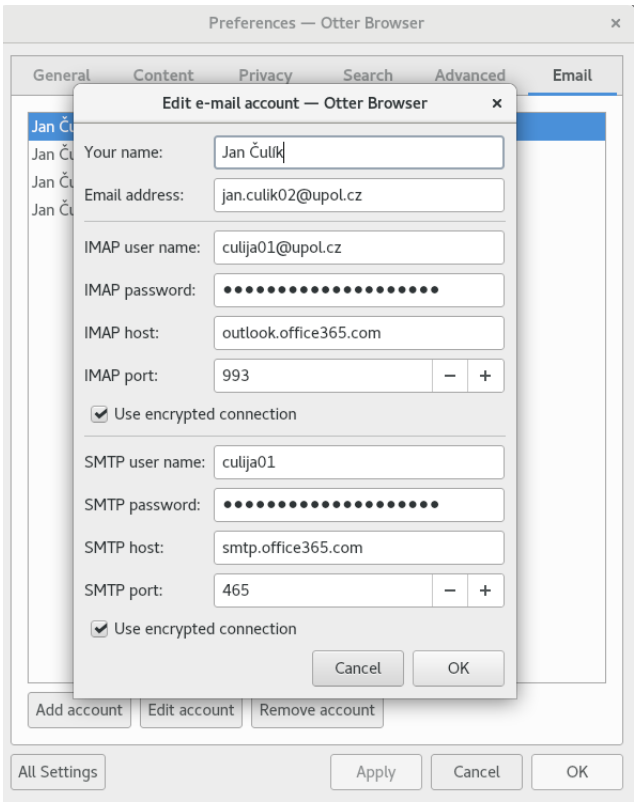
Zdrojový kód 5: Získání zdrojových kódů Otter Browseru

5 Popis vlastností e-mailového klienta

Tato kapitola popisuje funkcionalitu a ovládání e-mailového klienta.

5.1 Dialog s nastavením e-mailových účtů

E-mailové účty se nastavují ve standardním dialogovém okně s nastavením prohlížeče, které je dostupné z hlavního menu *Tools* pod položkou *Preferences*. Po otevření záložky *Email* se v okně vykreslí seznam stávajících e-mailových účtů a tlačítka pro přidání nového účtu a úpravu a odebrání stávajícího účtu. V případě, že v seznamu není vybraný žádný účet, jsou tlačítka pro úpravu a odebrání účtu neaktivní.



Obrázek 1: Nastavení e-mailového klienta

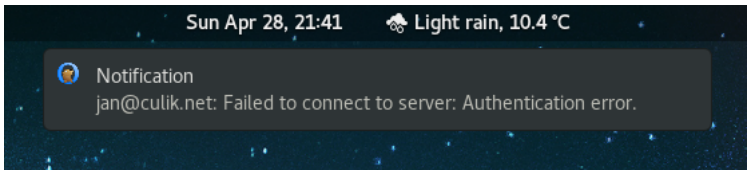
Kliknutí na tlačítko *Add account* otevře nové okno, ve kterém jsou pole pro vyplnění všech potřebných údajů o e-mailovém účtu. Kliknutí na tlačítko *OK* okno zavře a přidá do e-mailového modulu nový e-mailový účet s údaji, které v okně vyplnil uživatel. Kliknutí na tlačítko *Cancel* okno zavře a program neuloží žádné změny. Grafické uživatelské rozhraní pro správu e-mailových účtů je zobrazeno na obrázku 1.

Po vybrání některého z e-mailových účtů v seznamu a kliknutí na tlačítko *Edit account* se otevře stejné okno jako v případě přidávání nového účtu. Rozdíly

oproti přidávání nového účtu jsou v tom, že vstupní pole v okně mají předvyplněné hodnoty z vybraného účtu a že kliknutí na tlačítko *OK* nepřidá nový e-mailový účet, ale aktualizuje údaje u vybraného účtu.

V grafickém uživatelském rozhraní pro změnu nastavení e-mailových účtů neprobíhá žádná kontrola uživatelského vstupu, případnou chybu v nastavení účtu odhalí až jádro e-mailového klienta. To pak uživateli zobrazí notifikaci s popisem chyby. Příklad notifikace s chybou v prostředí GNOME je na obrázku 2.

Po vybrání některého z e-mailových účtů v seznamu a po kliknutí na tlačítko *Remove account* se zobrazí dialogové okno s dotazem, zda chce uživatel skutečně účet odebrat. Po potvrzení dialogu dojde k odebrání e-mailového účtu z programu.



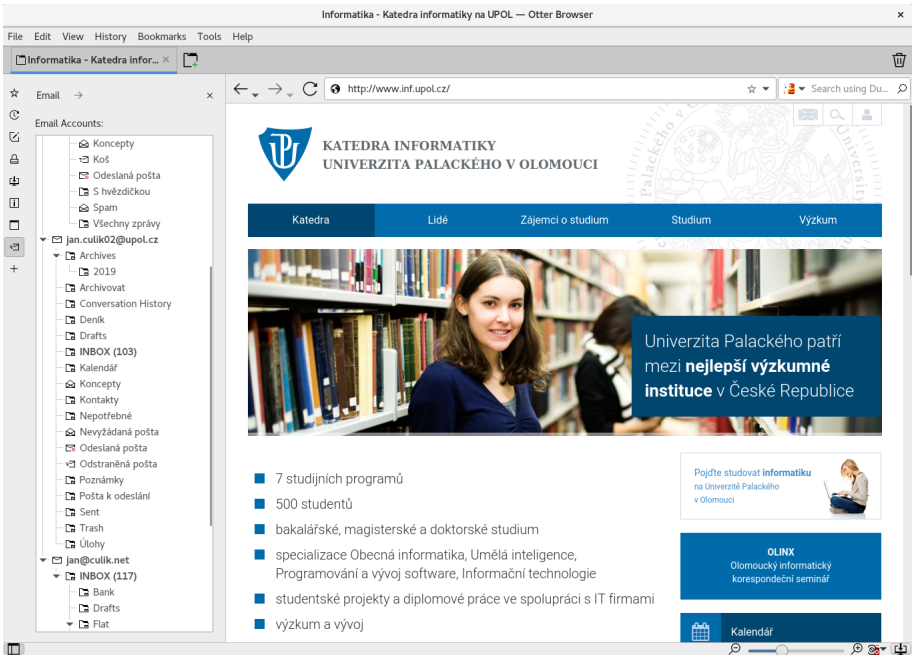
Obrázek 2: Notifikace s chybou o připojení k serveru

5.2 Boční panel s přehledem o e-mailových účtech

Hlavním místem pro zobrazení základních informací o záložkách, historii prohlížení, stahovaných souborech a informací o prohlížené stránce je v Otter Browseru i Opeře boční panel. Svůj základní přehled v bočním panelu v původní Opeře měly i další moduly a klienti pro e-mail, bittorent a IRC. V bočním panelu Otter Browseru pro e-mailový modul je zobrazena grafická komponenta s adresářovou strukturou IMAP účtů uživatele. Boční panel s přehledem e-mailových účtů zobrazuje obrázek 3.

Po prvním spuštění prohlížeče není boční panel zobrazen. Boční panel lze zobrazit nebo skrýt stisknutím klávesy *F4* nebo kliknutím na tlačítko zobrazení nebo skrytí bočního panelu v levém dolním rohu okna prohlížeče. Boční panel se pak zobrazí v levé části okna prohlížeče a obsahuje ikony pro záložky, historii prohlížení a dalších modulů. Aktivace některé z ikon boční panel rozšíří a ve volném místě se vykreslí grafická komponenta s informacemi o aktivním modulu. Ikona e-mailového modulu v bočním panelu není ve výchozím nastavení zobrazena. Pod ikonami všech modulů v bočním panelu je tlačítko *+*. Po kliknutí na tlačítko se zobrazí seznam všech modulů, které se dokáží vykreslit v bočním panelu. Vybráním položky *Email* dojde k přidání tlačítka s e-mailovým modulem do bočního panelu. Tlačítko e-mailového modulu má ikonu poštovní obálky.

V bočním panelu e-mailového modulu mají jména adresářů s nepřečtenými zprávami ve stromové struktuře tučný řez písma, za jménem adresáře je v závorkách napsaný počet nepřečtených zpráv. Informace o nepřečtených zprávách ve stromové struktuře adresářů se při změně automaticky aktualizují.



Obrázek 3: Boční panel s IMAP adresáři

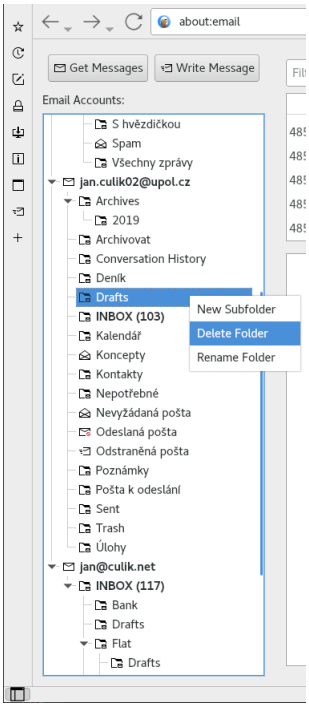
Stromová struktura s IMAP adresáři z e-mailových účtů uživatele se v prohlížeči používá na dvou místech: V bočním panelu pro e-mailový modul a v otevřené záložce s e-mailovým klientem. Chování komponenty se ale v obou případech liší.

Komponenta se stromem adresářové struktury v bočním panelu zobrazuje informace o adresářích v e-mailových účtech. Po kliknutí na některý z adresářů ve stromové struktuře program otevře novou záložku s plnohodnotným e-mailovým klientem. Původně jsem zamýšlel implementovat chování, kdy by prohlížeč po kliknutí na některý z adresářů zkontroloval právě otevřenou záložku v prohlížeči. Pokud by byl v záložce otevřený e-mailový klient v režimu čtení zpráv, prohlížeč by nastavil filtr zpráv tak, aby byly v seznamu zpráv zobrazeny zprávy pouze z tohoto adresáře. V opačném případě by prohlížeč otevřel novou záložku a v ní e-mailový modul se stejně nastaveným filtrem zpráv. Otter Browser ale neumí z bočního panelu zjistit, jaká záložka je aktuálně otevřena a boční panel ani neumí s otevřenými záložkami jednoduše komunikovat. Podle vývojářů Otter Browseru by takové chování možná šlo implementovat složitým a nestandardním způsobem.

Strom s adresářovou strukturou z e-mailových účtu tak zobrazuji i v záložce s e-mailovým klientem. Tam je komponenta se stromem adresářů umístěna na stejném formuláři jako komponenta pro čtení zpráv a adresářový strom už dokáže komunikovat se zbytkem záložky. Změna výběru adresáře ve stromu se pak projeví tak, že v seznamu zpráv budou vyfiltrované jen zprávy z tohoto adresáře.

Po kliknutí pravým tlačítkem myši na některý z IMAP adresářů ve stromu se zobrazí kontextová nabídka, přes kterou uživatel může vybraný adresář přejmenovat, smazat, nebo může vybranému adresáři vytvořit nový podadresář. Kontextová nabídka se zobrazuje jen v režimu, kdy je stromová struktura s

adresáři vykreslena v záložce s otevřeným e-mailovým klientem. Ukázka kontextové nabídky nad IMAP adresáři je na obrázku 4.



Obrázek 4: Boční panel s IMAP adresáři a kontextovou nabídkou

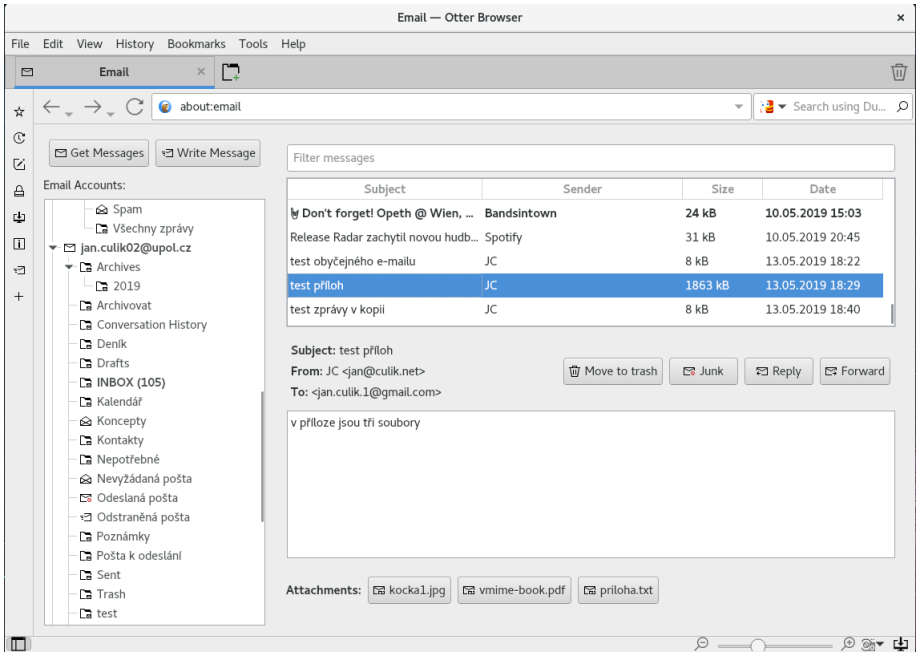
V záložce s e-mailovým klientem pak nad stromem adresářů ještě zobrazuji dvě tlačítka. Tlačítko *Get Messages* zavolá rutinu pro zkontrolování nových zpráv v e-mailových účtech uživatele a tlačítko *Write Message* naviguje uživatele do komponenty pro napsání nového e-mailu.

5.3 Komponenta pro čtení zpráv

Komponenta pro čtení zpráv se skládá z pěti hlavních částí: z tabulky se seznamem zpráv, z textového pole pro filtrování zpráv, z panelu, který obsahuje podrobnější informace o vybrané zprávě a z tlačítek akcí nad zprávou, z komponenty pro zobrazení obsahu zprávy a z panelu s přílohami. Okno s otevřenou záložkou e-mailového modulu a komponentou pro čtení zpráv je zobrazeno na obrázku 5.

5.3.1 Tabulka se seznamem zpráv a filtr zpráv

Tabulka zobrazuje seznam zpráv z toho adresáře, který si uživatel vybral ve stromové struktuře účtů a adresářů v bočním panelu. V tabulce jsou zobrazeny základní informace o zprávách, jako je předmět, odesílatel, velikost zprávy a datum odeslání.



Obrázek 5: Prohlížení zpráv v Otter Browseru

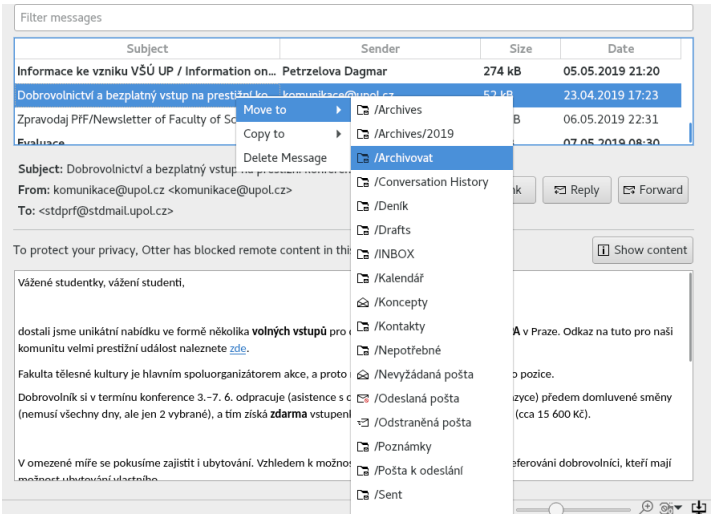
Po kliknutí na záhlaví sloupce tabulky klient seřadí zprávy v tabulce podle sloupce vzestupně, další kliknutí myší seřadí seznam zpráv sestupně.

Sloupce pro odesílatele, velikost a datum odeslání zprávy mají přednastavenou výchozí šířku tak, aby se do nich přesně vešly jejich hodnoty. Sloupec pro předmět pak vyplňuje zbytek šířky tabulky. Uživatel ale může ručně upravovat šířku sloupců podle sebe. Šířka sloupců se taky dynamicky mění při změně velikosti okna tak, aby se všechna data do tabulky vešla bez nutnosti horizontálního scrollování oknem. V klientu není implementovaná možnost volitelně zobrazovat nebo skrývat sloupce.

Po napsání textu do filtru zpráv tabulka zobrazí jen ty zprávy, kde předmět, obsah, datum a čas odeslání nebo kontakty z odesílatele, příjemců, příjemců v kopii nebo z příjemců pro odpověď zprávy se shodují s hledaným textem.

Po vybrání zprávy z tabulky klient v ostatních grafických komponentách zobrazí podrobnější informace o zprávě a obsah zprávy. Klient se snaží obsah zprávy zobrazit z databáze. Pokud chce uživatel zobrazit zprávu, jejíž obsah ještě nebyl stažen a uložen do databáze, klient nejdříve obsah zprávy stáhne ze serveru.

Po kliknutí pravým tlačítkem myši na nějakou zprávu v tabulce se seznamem zpráv se zobrazí kontextová nabídka. Vybranou zprávu lze v rámci účtu zkopírovat nebo přesunout do jiného IMAP adresáře, případně jde zprávu přesunout do koše, nebo z koše smazat. Akce vyvolané kontextovou nabídkou nejdříve provedou změnu na poštovním serveru. V lokální databázi se změna provede až pokud změna proběhla úspěšně i na serveru. Kontextová nabídka zprávy je zobrazena na obrázku 6.



Obrázek 6: Kontextová nabídka zprávy

5.3.2 Panel s informacemi o vybrané zprávě a tlačítka akcí

Panel se nachází uprostřed okna prohlížeče mezi tabulkou se seznamem zpráv a komponentou zobrazující obsah vybrané zprávy. Panel zobrazuje podrobnější informace o odesílateli, příjemcích, příjemcích v kopii a předmětu, které se celé nevejdou do tabulky se seznamem zpráv. Panel dále obsahuje tlačítka pro smazání zprávy, označení zprávy jako spam, přesunutí zprávy do archivu, odpovědění na zprávu a na přeposlání zprávy.

Tlačítka *Reply* pro odpověď na zprávu a *Forward* pro přeposlání zprávy se zobrazují vždy. Tlačítko *Reply All* pro odpovědění všem klient zobrazí pouze když má zpráva příjemce v kopii.

Tlačítko *Junk* klient zobrazuje jen když účet obsahuje adresář s příznakem *junk* a uživatel zrovna tento adresář neprohlíží. Příznak *junk* říká, že adresář slouží k uložení nevyžádané pošty (spamu). Kliknutí na tlačítko zprávu přesune do tohoto adresáře nejdříve na serveru, potom i v lokální databázi. Tlačítko *Archive* se chová stejně jako tlačítko *Junk*, jen s tím rozdílem, že pracuje s adresářem, který má příznak *archive* pro archivování pošty.

Posledním z tlačítek akcí je tlačítko pro smazání zprávy. V případě, že v e-mailovém účtu není žádný adresář s příznakem *trash*, má tlačítko text *Delete*. Kliknutí na tlačítko zobrazí uživateli dialogové okno pro potvrzení akce a pak zprávu rovnou smaže. Jinak má tlačítko text *Move to trash* a přesouvá zprávu do adresáře s košem. V adresáři s košem má pak tlačítko zase text *Delete* a kliknutí na tlačítko zprávu smaže. Uživatel může zprávu z koše přesunout do jiného adresáře přes kontextové menu v tabulce se seznamem zpráv.

5.3.3 Komponenta pro zobrazení obsahu zprávy

Vlastní obsah zprávy klient zobrazuje v *QTextBrowser* komponentě z Qt frameworku. Tato komponenta umí vykreslit jen omezenou podmnožinu HTML a

CSS, takže obsah některých složitějších HTML e-mailů vypadá zdeformovaně. Vykreslovací jádro této komponenty je součástí Qt frameworku a Qt ho primárně využívá pro vykreslování formátovaného textu. *QTextBrowser* rovněž nepodporuje javascript.

Plnohodnotnou komponentu webového prohlížeče s vykreslovacími jádry Blink nebo WebKit jsem nepoužil hlavně z bezpečnostních důvodů, kdy jednoduché vykreslovací jádro Qt frameworku ve srovnání s plnohodnotným jádrem webového prohlížeče trpí minimem bezpečnostních problémů. Zobrazovat obsah zprávy v jednoduché *QTextBrowser* komponentě doporučují i vývojáři Otteru.

Komponenta ve výchozím nastavení v HTML e-mailech nezobrazuje externí obrázky a kaskádové styly. Klient nad komponentou s obsahem zprávy zobrazuje panel s informací, že v rámci ochrany soukromí uživatele externí objekty nestáhl a nezobrazil. Po kliknutí na tlačítko *Show Content* ale klient obrázky dodatečně stáhne a v obsahu zprávy zobrazí.

5.3.4 Panel s přílohami

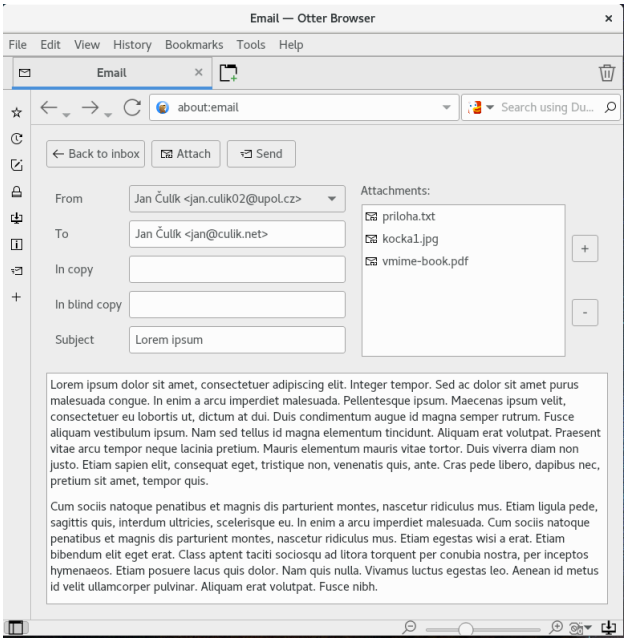
V případě, že právě zobrazená zpráva obsahuje přílohy, je pod komponentou s obsahem zprávy zobrazen i panel s řádkem tlačítek, které reprezentují jednotlivé soubory příloh. Každé tlačítko obsahuje text se jménem přiloženého souboru. Po kliknutí na tlačítko s přílohou se otevře standardní systémový dialog pro uložení souboru a uživatel si v něm zvolí cestu, kam se soubor uloží.

5.4 Komponenta pro psaní zprávy

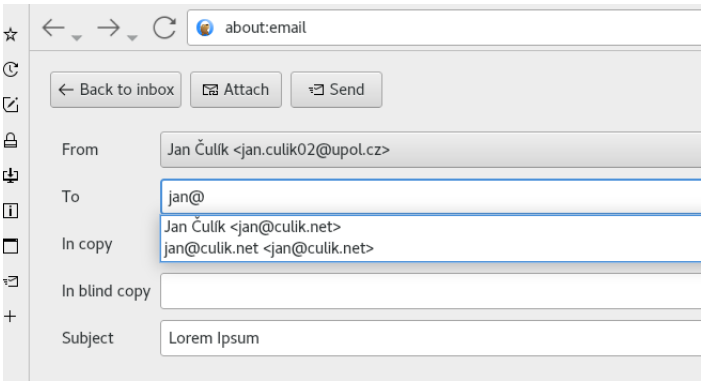
Komponenta pro psaní zprávy poskytuje rozhraní pro napsání nového e-mailu v neformátovaném textu s přílohami, zároveň se používá i k odpovědi na přijatou zprávu nebo k přeposlání přijaté zprávy. V případech užití s odpovídáním na zprávu nebo s přeposíláním zprávy klient podporuje i HTML e-maily. Prostředí pro psaní zprávy je zobrazeno na obrázku 7.

Komponenta obsahuje rozbalovací seznam pro výběr účtu, ze kterého bude e-mail odeslán. Dále obsahuje textový vstup pro psaní neformátovaného textu zprávy a textové pole pro předmět.

Samostatnou skupinou pak jsou textová pole pro příjemce, příjemce v kopii a příjemce ve skryté kopii. E-mailový klient podporuje psaní informací o kontaktech ve formátu *Jméno <jmeno@server>*, kde jméno a špičaté závorky kolem e-mailové adresy nejsou povinné. U zpráv pro větší množství příjemců je oddělovačem mezi jednotlivými kontakty znak čárky. Každé z těchto polí taky obsahuje našeptávač se seznamem všech kontaktů, které má uživatel uloženy v databázi v polích *From*, *To*, *In Copy* a *Reply To* u metadat všech přijatých zpráv. Našeptávač umí kontakty filtrovat podle jména i podle e-mailové adresy. Ukázka našeptávače je na obrázku 8.



Obrázek 7: Psaní e-mailu v Otter Browseru



Obrázek 8: Našeptávač při psaní kontaktů

5.4.1 Práce s přílohami

Po kliknutí na tlačítko *Attach* se otevře systémový dialog pro výběr souboru z disku. Po vybrání souboru nebo souborů a potvrzení dialogu si klient načte soubory do paměti. Zároveň se napravo od rozbalovacího seznamu s výběrem odesílatele a textovými poli pro předmět a příjemce zobrazí seznam přiložených souborů. Tlačítkem + u seznamu souborů lze přidat novou přílohu, tlačítkem – lze vybraný soubor v seznamu z příloh odebrat. Pokud uživatel všechny přiložené soubory odebere, seznam příloh se automaticky skryje.

Žádný standard nepopisuje, jaká je maximální možná velikost příloh v e-mailu. Někteří poskytovatelé e-mailových služeb maximální velikost příloh omezují, ale horní hranice používaná poskytovateli se liší. V e-mailovém klientu maximální velikost příloh neomezuji a klient se pokusí odeslat všechny přílohy vlože-

né uživatelem. Existuje riziko, že se program zhroutí, když se uživatel pokusí přidat do příloh soubory o větší velikosti, než je množství volné operační paměti na jeho počítači.

5.4.2 Kontrola zprávy před odesláním a odesílání zprávy

Po kliknutí na tlačítko *Send* e-mailový klient zkontroluje uživatelem vložená data zprávy. Pokud jsou všechny údaje validní, klient zneaktivní tlačítko *Send* a spustí vlákno, ve kterém se pokusí zprávu odeslat. Když se zpráva úspěšně odešle, klient komponentu pro psaní zprávy skryje a zobrazí komponentu pro prohlížení příchozí pošty uživatele. Jinak klient zobrazí notifikaci s podrobnostmi o chybě a udělá tlačítko *Send* opět aktivním.

Klient kontroluje pole pro předmět, které nemůže být prázdné a pole pro vložení informací o příjemcích zprávy. Pole pro příjemce v kopii nebo skryté kopii můžou být prázdná, pole pro příjemce zprávy musí obsahovat alespoň jednu validní e-mailovou adresu.

Kontrola kontaktů v polích pro příjemce, příjemce v kopii a příjemce ve skryté kopii probíhá tak, že klient vezme text napsaný uživatelem, ten rozdělí na podřetězce podle znaku čárka a v každém takovém podřetězci se pokusí načíst e-mailovou adresu regulárním výrazem. Pokud regulární výraz nenajde v některém z podřetězců e-mailovou adresu, klient zobrazí chybovou hlášku a zprávu neodešle.

5.4.3 Psaní odpovědi na zprávu

Komponenta pro psaní zprávy se zobrazí i potom, co uživatel při čtení pošty u vybrané zprávy klikne na tlačítko *Reply* (odpovědět) nebo *Reply All* (odpovědět všem). Z původní zprávy se v komponentě předvyplní obsah zprávy a příjemce, nastaví se přílohy a v rozbalovacím seznamu s výběrem účtu, ze kterého se odpověď odešle bude vybraný ten účet, kterému původní zpráva přišla. Uživatel má možnost po vzoru klienta M2 z Opery změnit účet, ze kterého se odpověď pošle. Jako předmět zprávy se nastaví předmět původní zprávy doplněný o prefix *Re:* .

V poli pro příjemce jsou předvyplněné kontakty z hlavičky *Reply-To* původního e-mailu. Pokud původní e-mail nemá hlavičku *Reply-To* nastavenou, jako příjemce zprávy se nastaví odesílatel původní zprávy. V případě akce odpovědět všem se do pole s příjemci doplní i kontakty, které byly v původní zprávě uvedeny v kopii.

Způsob zobrazení původní zprávy a psaní odpovědi na zprávu se u e-mailů s neformátovaným textem a u e-mailů s HTML obsahem liší.

V případě odpovědi na e-mail v neformátovaném textu je obsah původní zprávy zobrazen v *QTextEdit* komponentě, do které bude uživatel zároveň i psát odpověď. Před obsah původní zprávy je vložen nový řádek s textem ve formátu *On 01.01.2019 12:00, Odesílatel <odesilatel@server> wrote:*. Obsah původní zprávy pak je upraven tak, aby každý řádek textu měl maximálně 78 znaků.

Řádky s větším množstvím znaků klient rozdělí v místě prvního bílého znaku před 78. pozicí. Nakonec klient před každý řádek textu přidá znak >. Doporučení o maximální délce řádku je součástí standardu RFC 2822[12] a implementuje ho většina e-mailových klientů.

U odpovědi na HTML zprávu uživatel píše do *QTextEdit* komponenty pouze vlastní odpověď. Pod touto komponentou je zobrazený *QTextBrowser*, který zobrazuje obsah původního HTML e-mailu. Obsah původní zprávy v *QTextBrowser* komponentě uživatel nemůže upravovat. Původní zprávu klient obalí HTML tagem * *, kterému na levé straně kaskádovými styly nastaví odsazení a zobrazení horizontální čáry. Před původní obsah zprávy dodá řádek s textem ve formátu *On 01.01.2019 12:00, Odesílatel <odesilatel@server> wrote:*. Při odesílání odpovědi pak klient spojí obsah obou textových komponent dohromady a ten nastaví jako HTML obsah odpovědi. Výsledná zpráva tak obsahuje odpověď a pod ní odsazený obsah původní zprávy.

5.4.4 Přeposílání zpráv

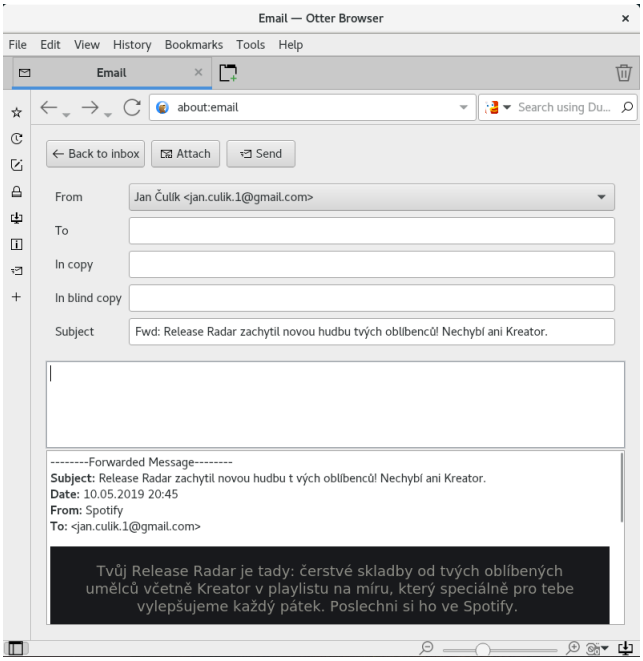
Přeposílání zpráv funguje až na pár rozdílů úplně stejně jako odpovídání na zprávy. Rozdíl je v nastavení předmětu, který obsahuje předmět původní zprávy s prefixem *Fwd:* . Zpráva taky nemá přednastavená pole s příjemci. Poslední rozdíl je v textu, který klient vkládá před obsah původní zprávy. Formát vloženého textu před přeposlanou zprávu popisuje zdrojový kód 6. Na obrázku 9 je ukázka přeposílání HTML zprávy.

```
1 --------Forwarded Message--------
2 Subject: Původní předmět
3 Date: 01.01.2019 12:00
4 From: Jméno Odesílatele <odesílatel@server>
5 To: <příjemce@server>
```

Zdrojový kód 6: Text vložený před odpověď na zprávu

6 Od prototypu k integraci do Otter Browseru

Když jsem začal pracovat na bakalářské práci, mé znalosti programovacího jazyka C++ byly na základní úrovni. S Qt frameworkem jsem neměl žádné předchozí zkušenosti. Proto jsem jako první začal psát prototyp e-mailového klienta. Projekt s prototypem jsem rozdělil na dvě části: na staticky linkovanou knihovnu, ve které bylo jádro e-mailového klienta postavené kolem knihovny *vmime* s mými datovými strukturami a na jednoduchou grafickou aplikaci připomínající Microsoft Outlook nebo Mozilla Thunderbird, která stavěla na této knihovně. Do integrace do Otter Browseru jsem se pustil v momentě, kdy jsem měl v prototypu vyzkoušenou a ověřenou většinu potřebné funkcionality. Na obrázku 10 je prototyp e-mailového klienta, zdrojové kódy prototypu jsou volně dostupné na Githubu.

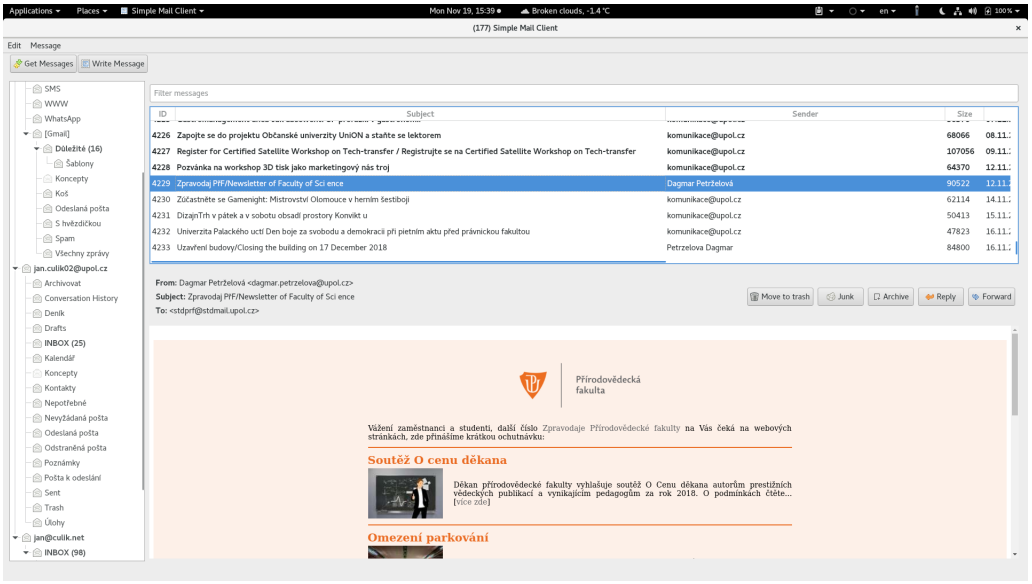


Obrázek 9: Přeposílání HTML zprávy

E-mailový modul jsem integroval do vlastní verze Otter Browseru. Jádro e-mailového klienta jsem do Otteru po domluvě s vývojáři vložil přímo, kód tak není rozdělený do samostatné knihovny jako v prototypu. Zdrojové kódy všech modulů Otter Browseru se nacházejí v adresáři *src/modules*, v tomto umístění jsem proto vytvořil adresář *mail*, do kterého jsem zkopíroval jádro původního prototypu.

V prohlížeči Otter je grafické uživatelské rozhraní všech tříd s formuláři vytvořeno v nástroji Designer, který je součástí vývojového prostředí QtCreator. Popis struktury uživatelského rozhraní vytvořeného v nástroji Designer je uložen v XML souboru, C++ objekty jednotlivých prvků uživatelského rozhraní pak Qt framework vytvoří během překladu programu. Třídy a formuláře s grafickým uživatelským rozhraním prohlížeče se nacházejí v adresáři *src/modules/windows*. Na stejné místo jsem proto postupně přidával zdrojové kódy a formuláře týkající se grafického uživatelského rozhraní pro práci s e-maily. Zatímco v prototypu jsem všechny prvky grafického uživatelského rozhraní vytvářel a nastavoval ručně v kódu, při implementaci v Otter Browseru jsem se snažil v maximální možné míře využít grafický nástroj pro tvorbu uživatelského rozhraní Designer.

Formuláře s grafickým uživatelským rozhraním Otter Browseru téměř vůbec neobsahují grafické kompomenty z čistého Qt frameworku, vývojáři prohlížeče si vytvořili svou vlastní sadu grafických komponent, která ale z komponent poskytovaných Qt frameworkem dědí. Ve většině případů je zděděná sada komponent obohacena jen o větší možnosti úprav vzhledu. Uživatel si totiž může vzhled prohlížeče upravit speciálním souborem s kaskádovými styly. Z tohoto důvodu v grafickém uživatelském rozhraní e-mailového klienta také používám sadu grafick-



Obrázek 10: Původní prototyp e-mailového klienta

ých komponent poskytovanou Otter Browserem.

6.1 Zaregistrování e-mailového modulu do prohlížeče

K úplné integraci nového modulu je potřeba přidat několik změn do jádra Otter Browseru. Bez nich by prohlížeč o přítomnosti e-mailového modulu nevěděl, při startu by nevytvořil struktury s e-mailovými účty a ani by nebyl schopen zobrazit grafické uživatelské rozhraní pro práci s elektronickou poštou.

Základem při integraci je vytvoření singleton třídy *EmailAccountsManager*. Tato třída si drží seznam e-mailových účtů, SQLite databázi a stará se o načtení konfigurace e-mailových účtů při startu programu, nebo o zapsání změn při změně konfigurace. K inicializaci singletonu dochází v hlavní třídě Otter Browseru *Application* a instance této třídy je společná pro všechna okna prohlížeče. Stejným způsobem je v Otter Browseru implementován například správce záložek, správce doplňků, jádro čtečky RSS kanálů a další moduly. Ze všech dalších míst prohlížeče se dá k datům o e-mailových účtech dostat zavoláním metody *EmailAccountsManager::getInstance()*, která vrátí ukazatel na instanci singletonu.

6.2 Integrace bočního panelu a záložky pro čtení a psaní e-mailů

Základem grafického uživatelského rozhraní pro práci s e-maily je boční panel (sidebar). Ten obsahuje stromovou strukturu s adresáři a informacemi o nepřečtených zprávách v e-mailových účtech uživatele. Stejně důležitá je i komponenta pro čtení e-mailů, která se v prohlížeči otevírá jako nová záložka (tab). Komponenta pro čtení e-mailů obsahuje seznam zpráv s filtrem a dále zobrazuje zák-

ladní informace o vybrané zprávě, obsah vybrané zprávy a tlačítka se základními akcemi nad vybranou zprávou. Ve stejné záložce s komponentou pro čtení zpráv se vykresluje i komponenta pro psaní zprávy.

Vnitřně je boční panel implementován i s obsahem záložky v jednom formuláři. Otter Browser pro tyto účely obsahuje bázovou třídu *ContentsWidget*, kterou programátor při tvorbě nového modulu zdědí. Třída *ContentsWidget* obsahuje metodu *isSidebarPanel()*, která vrací hodnotu *true*, když je komponenta vykreslena v bočním panelu, jinak vrací hodnotu *false*. Programátor na základě této informace v konstruktoru své komponenty skryje tu část formuláře, kterou v bočním panelu nebo v záložce nechce zobrazovat. Při otevření nové záložky s e-mailovým klientem se v záložce vykreslí komponenta pro boční panel a komponenta pro čtení zprávy, komponenta pro psaní zprávy je skryta. Při přechodu do režimu psaní zprávy se komponenta s bočním panelem a komponenta pro čtení zprávy skryjí a zůstane zobrazena jen komponenta pro psaní zprávy.

Programátor v potomkovi třídy *ContentsWidget* ještě musí implementovat několik metod, z nichž nejdůležitější jsou *getTitle()*, *getUrl()* a *getIcon()*. Metoda *getTitle()* vrací jméno záložky (tabu). Metoda *getUrl()* vrací textový řetězec, po jehož napsání do adresního řádku se v aktuální záložce obsah widgetu vykreslí. Pro e-mailový modul jsem použil řetězec *about:email*. A nakonec metoda *getIcon()* vrací ikonu, která se zobrazí v levé části bočního panelu a v otevřené záložce s e-mailovým modulem. E-mailový klient vrací ikonu poštovní obálky.

Aby se komponenta dokázala zobrazit v bočním panelu nebo v obsahu záložky, je nutné ji na dvou místech Otter Browseru zaregistrovat: V konstruktoru třídy *AddonsManager* a v metodě *createContentsWidget()* třídy *WidgetFactory*.

6.3 Integrace nastavení e-mailového klienta

O vykreslení dialogového okna s nastavením programu se stará třída *PreferencesDialog* Otter Browseru. Okno se dá vyvolat z hlavního menu programu *Tools* pod položkou *Preferences*. Okno s nastavením programu je členěno do záložek, kde každá ze záložek sdružuje nějakou skupinu nastavení. Vlastní záložka s nastavením e-mailových účtů je implementována jako obyčejný *QWidget* z Qt frameworku, aby ale Otter Browser záložku do okna s nastavením programu automaticky přidal, je nutné ji v konstruktoru třídy *PreferencesDialog* zaregistrovat. V té samé třídě je ještě u metody *currentTabChanged()* potřeba nastavit, aby se komponenta s nastavením e-mailových účtů při aktivování záložky vykreslila.

7 Struktura e-mailového klienta

Tato kapitola popisuje základní třídy a metody klienta pro práci s e-mailovými účty, parsery pro zpracování stažených dat a soubory, do kterých si e-mailový klient Otter Browseru ukládá pomocná data a informace o nastavení e-mailových účtů.

E-mailový modul používá dva soubory. Do textového souboru ve formátu JSON si ukládá nastavení o e-mailových účtech uživatele a v souboru s SQLite databází si udržuje informace o zprávách a adresářích z e-mailových účtů.

Třída *EmailAccount* zapouzdřuje údaje o nastavení e-mailového účtu a poskytuje rozhraní pro stahování pošty, odesílání zpráv a provádění operací, které změní stav na IMAP serveru.

7.1 SQLite databáze

Jádro e-mailového klienta ukládá data stažená z IMAP serveru do lokální SQLite databáze. Soubor s databází je uložen ve výchozím adresáři pro ukládání dat aplikací podle platformy. Soubor je pojmenován *emailDatabase.sqlite*. Na operačních systémech unixového typu je soubor uložen na cestě *~/.config/otter/emailDatabase.sqlite*.

E-mailový modul si vystačí se dvěmi SQL tabulkami. Tabulka *Folders* slouží k uchování informací o adresářích z IMAP účtů. Strukturu SQL tabulky *Folders* popisuje tabulka 1. Tabulka *MessageData* obsahuje informace o příchozích zprávách. Strukturu SQL tabulky *MessageData* popisuje tabulka 2.

Pravdivostní hodnoty u dat z příznaků (flagů) jsou v databázi uloženy jako celá čísla, protože SQLite nemá datový typ *boolean*.

Přílohy a vložené objekty HTML zpráv jsou uloženy jako pole bajtů. Pokud je ve zprávě více než jedna příloha nebo více než jeden vložený objekt, tak jsou data těchto objektů uložena za sebou v jednom poli bajtů, e-mailový modul při deserializaci umí data správně rozdělit.

Informace o kontaktech jsou v databázi uloženy jako text ve formátu JSON. Každý kontakt má povinnou e-mailovou adresu a nepovinné jméno. V tabulce *MessageData* sloupec *sender* obsahuje jeden JSON objekt s informací o odesílateli, sloupce *recipients*, *copyRecipients* a *replyToRecipients* obsahují pole JSON objektů s kontakty. Data o kontaktech jsem původně uchovával jako pole bajtů, ukládání souboru s databází u většího množství zpráv ale bylo velmi pomalé a způsobovalo velký počet zápisů na disk. Výkon databáze se po přechodu na textovou reprezentaci dat o kontaktech výrazně zlepšil.

Databáze běží v hlavním vlákně programu, které je sdílené i s grafickým uživatelským rozhraním. Nevýhodou tohoto řešení je, že při zápisu velkého množství dat do databáze v jeden okamžik může grafické uživatelské rozhraní na krátkou dobu zamrznout. Tohoto problému si uživatel může všimnout při přidávání nového e-mailového účtu do programu, kdy si modul stáhne metadata všech zpráv z IMAP serveru a pak je ukládá do databáze. Při zápisu databáze na starší SSD disk po stažení metadat k 15 000 zprávám grafické rozhraní zamrzne přibližně na 5 sekund, na pomalém plotnovém disku nebo při přidání účtu s větším množstvím zpráv může být odezva mnohem delší. Původně jsem chtěl databázi přepsat tak, aby běžela ve vlastním vlákně, ale třída *QSqlTableModel* z Qt frameworku, kterou používám k naplnění seznamu zpráv v komponentě pro čtení pošty vyžaduje k objektu s databází přímý přístup. Přesunutí databáze

Tabulka 1: Tabulka Folders

Jméno sloupce	SQL datový typ	Popis
id	`integer primary key`	jednoznačný identifikátor adresáře v databázi
emailAddress	`text`	e-mailová adresa IMAP účtu, ve kterém se adresář nachází
path	`text`	jméno IMAP adresáře včetně cesty od kořene
isAllMessages	`integer`	příznak allMessages: reprezentuje, zda adresář obsahuje všechny zprávy
isArchive	`integer`	příznak isArchive: reprezentuje, zda adresář slouží k archivaci zpráv
isDrafts	`integer`	příznak drafts: reprezentuje, zda adresář slouží k uchování konceptů
isJunk	`integer`	příznak junk: reprezentuje, zda adresář slouží k uchování nevyžádané pošty
isSent	`integer`	příznak sent: reprezentuje, zda adresář slouží k uchování odeslané pošty
isTrash	`integer`	příznak trash: reprezentuje, zda je adresář koš.

Tabulka 2: Tabulka MessageData

Jméno sloupce	SQL datový typ	Popis
id	`integer primary key`	jednoznačný identifikátor zprávy v databázi
folderId	`integer`	id adresáře z tabulky Folders, ve kterém se zpráva nachází
uid	`text`	jednoznačný číselný identifikátor zprávy na IMAP serveru
isSeen	`integer`	příznak seen: reprezentuje, zda zpráva byla zobrazena
isDraft	`integer`	příznak draft: reprezentuje, zda je zpráva konceptem
date	`text`	reprezentuje datum a čas, kdy byla zpráva odeslána
sender	`text`	obsahuje jméno a e-mailovou adresu odesílatele
size	`text`	obsahuje velikost zprávy na IMAP serveru
subject	`text`	obsahuje předmět zprávy
recipients	`text`	obsahuje jména a e-mailové adresy příjemců zprávy
copyRecipients	`text`	obsahuje jména a e-mailové adresy příjemců zprávy v kopii
replyToRecipients	`text`	obsahuje jména a e-mailové adresy příjemcům odpovědi na zprávu
plainTextContent	`text`	obsahuje plaintextovou část multipart zprávy
htmlContent	`text`	obsahuje HTML část multipart zprávy
attachments	`blob`	obsahuje přílohy zprávy
embeddedObjects	`blob`	obsahuje embedded objekty z HTML části multipart zprávy

do vlastního vlákna a přepsání třídy s modelem pro seznam zpráv by mohlo přinést výkonnostní problémy na jiných místech a zvýšilo by se riziko dalších chyb v programu. V ostatních případech užití výkon databáze uživatele nijak neomezuje.

7.2 JSON soubor s nastavením e-mailových účtů

E-mailový klient kromě databáze používá i jednoduchý textový soubor, do kterého si ukládá všechny informace o nastavení e-mailových účtů uživatele. Soubor se nachází na cestě *~/.config/otter/emailAccounts.json* na operačních systémech unixového typu. Obsahuje textovou reprezentaci JSON pole, kde každý objekt v poli popisuje jeden e-mailový účet. Uživatel by nikdy neměl mít potřebu upravovat tento soubor přímo.

Každý e-mailový účet v konfiguračním souboru je definovaný jménem uži-

```
[
   {
      "contactName": "Jan Čulík",
      "emailAddress": "jan.culik.02@upol.cz",
      "imapHost": "outlook.office365.com",
      "imapPassword": "heslo",
      "imapPort": "993"
      "imapUserName": "culija01@upol.cz",
      "isImapConnectionSecured": true,
      "isSmtpConnectionSecured": true,
      "smtpHost": "smtp.office365.com",
      "smtpPassword": "heslo",
      "smtpPort": "465",
      "smtpUserName": "culija01"
   }
]
```

Zdrojový kód 7: ukázka souboru s nastavením e-mailových účtů

vatele (*contactName*), e-mailovou adresou (*emailAddress*), přihlašovacím jménem k IMAP a SMTP serverům (*imapUserName*, *smtpUserName*), hesly k IMAP a SMTP serverům (*imapPassword*, *smtpPassword*), adresami IMAP a SMTP serverů (*imapHost*, *smtpHost*) a porty (*smtpPort*, *imapPort*). Heslo je uloženo v nešifrované podobě, stejně jako v e-mailovém klientu M2 z Opery 12.

7.3 Třída EmailAccount

Nejdůležitější částí e-mailového klienta je třída *EmailAccount*. Objekty této třídy si drží informace o e-mailových účtech uživatele, stahují informace o zprávách z poštovních serverů, zapisují změny do SQLite databáze a zároveň poskytují prostředky pro veškerou práci se SMTP a IMAP servery.

Všechny metody této třídy, které komunikují přes síť s některým ze serverů jsou naprogramovány asynchronně: metoda si vytvoří lambda výraz, v jehož těle je předpis pro síťové volání. Všechny informace potřebné k provedení operace jsou lambda výrazu předány formálními argumenty. Metoda pouze spustí lambda výraz v novém vlákně a zaregistruje vláknu callback metodu. Síťové operace tak neblokuje hlavní vlákno programu a grafické uživatelské rozhraní. Výsledek síťové operace se pak zpracovává v hlavním vlákně programu v callback funkci.

Třída *EmailAccount* obsahuje metody, které mění stav e-mailového účtu na IMAP serveru. Všechny metody jsou implementovány tak, že e-mailový modul v novém vlákně nejdříve provede změnu na IMAP serveru. Výsledek operace je pak hlavnímu vláknu vrácen v callback funkci. V případě, že změna stavu na serveru byla úspěšná, program provede ty samé změny i nad daty v lokální databázi. Jinak program změny nad lokálními daty neprovede a uživateli je zobrazena notifikace s popisem chyby.

Destruktivní operace nad e-mailovými účty nefungují u některých posky-

tovatelů e-mailových služeb. Problém může být v chybách v použité externí knihovně *vmime* nebo v nestandardní implementaci protokolu IMAP ze strany poskytovatelů e-mailových služeb. Destruktivní operace fungují například s e-mailovými účty od Seznamu nebo na univerzitních Office 365 účtech, na vážné problémy jsem narazil například u Gmailu od Google.

7.3.1 Stahování a zpracování informací o adresářích a zprávách z IMAP serveru

O stažení a zpracování struktury IMAP účtu se stará metoda *fetchStoreContent()* třídy *EmailAccount*. Metoda spustí vlákno, které ze serveru rekurzivně stáhne seznam IMAP adresářů a ten předá callback funkci. Callback funkce zkontroluje stažené adresáře s databází a případně přidá nové adresáře, nebo z databáze odstraní adresáře, které už na serveru neexistují.

Dál callback funkce ten samý seznam adresářů použije k vytvoření dalšího vlákna, které pro každý adresář stáhne a zpracuje metadata všech zpráv.

Callback funkce se staženými a zpracovanými metadaty zpráv u nepřečtených zpráv v databázi zkontroluje, zda zprávy nebyly přečteny třeba na jiném počítači a případně stav zpráv v databázi aktualizuje. Callback funkce taky z databáze odebere zprávy, které se už na serveru nenacházejí a přidá do databáze nové zprávy. Nakonec callback funkce vyvolá signál s počtem nových zpráv, na který aplikace zareaguje zobrazením notifikace.

Metoda *fetchStoreContent()* se v programu volá z většího množství míst. Zavolá se nad všemi účty při spuštění programu, dále při přidání nového e-mailového účtu, po úpravě některého ze stávajících účtů, po kliknutí na tlačítko *Get messages* v bočním panelu e-mailového modulu a provolává se nad všemi účty každých patnáct minut při automatické kontrole nových zpráv.

Při chybě v kterékoliv části volání se provádění dalších akcí zruší a uživateli je zobrazena notifikace s informací o chybě.

7.3.2 Stahování a zpracování obsahu zprávy

O stažení a zpracování obsahu zprávy se stará metoda *fetchMessageContent()* třídy *EmailAccount*. Metoda vytvoří vlákno, které ze serveru stáhne a zpracuje neformátovaný a HTML text zprávy, případně i přílohy a vložené objekty, které pak v callback funkci vrátí do hlavního vlákna. Callback funkce pak uloží stažená data do databáze a vyvolá signál, který dá zbytku aplikace vědět, že je obsah zprávy uložený a připravený k dalšímu použití.

Tato metoda se volá v případě, kdy uživatel chce zobrazit zprávu, jejíž obsah ještě nebyl stažen. Při každé změně výběru zprávy v seznamu zpráv uživatelem program kontroluje, jestli je v databázi uložen obsah zprávy v neformátovaném a HTML textu. Pokud jsou u zprávy v databázi obě pole prázdná, program předpokládá, že zpráva ještě nebyla stažena a zavolá si pro její obsah na IMAP server.

7.3.3 Odesílání zpráv

Odeslání zprávy se provede zavoláním metody *sendMessage()*, kdy metoda jako argument bere objekt zapouzdřující všechny informace o zprávě k odeslání. Metoda odesílané zprávě přiřadí aktuální datum a čas.

K zavolání této metody dochází v případě, kdy uživatel při psaní e-mailu klikne na tlačítko pro odeslání zprávy. V případě chyby při odesílání zprávy aplikace zobrazí notifikaci se stručným popisem problému.

7.3.4 Mazání zpráv

O mazání zpráv se stará metoda *deleteMessage()*. Metoda se volá, když uživatel v grafickém uživatelském rozhraní u detailu vybrané zprávy klikne na tlačítko pro smazání zprávy, nebo když vybere smazání zprávy z kontextové nabídky po kliknutí pravým tlačítkem myši na zprávu v seznamu zpráv.

7.3.5 Kopírování zpráv

E-mailové zprávy kopíruje metoda *copyMessage()* třídy *EmailAccount*. K zavolání metody pro zkopírování zprávy dojde, když uživatel v kontextové nabídce po kliknutí pravým tlačítkem myši na nějakou zprávu v seznamu zpráv vybere v podnabídce pro kopírování zprávy adresář, do kterého se má zpráva zkopírovat.

7.3.6 Přesouvání zpráv

Zprávy napříč adresáři v IMAP účtu přesouvá metoda *moveMessage()* třídy *EmailAccount*. K zavolání metody pro přesunutí zprávy dochází, když uživatel v kontextové nabídce po kliknutí pravým tlačítkem myši na nějakou zprávu v seznamu zpráv vybere v podnabídce pro přesunutí zprávy adresář, do kterého se má zpráva přesunout.

7.3.7 Vytváření IMAP adresářů

Nové IMAP adresáře vytváří metoda *createFolder()* ve třídě *EmailAccount*. K zavolání metody pro vytvoření adresáře dojde, když uživatel klikne pravým tlačítkem myši na některý z uzlů ve stromu s IMAP adresáři e-mailových účtu v a když v kontextové nabídce aktivuje volbu vytvoření nového adresáře.

7.3.8 Přejmenování IMAP adresářů

Existující adresáře přejmenuje metoda *renameFolder()* třídy EmailAccount. K zavolání metody pro přejmenování adresáře dojde, když uživatel klikne pravým tlačítkem myši na některý z uzlů ve stromu s IMAP adresáři e-mailových účtu a když v kontextovém menu aktivuje volbu přejmenování adresáře.

7.3.9 Mazání IMAP adresářů

O mazání IMAP adresářů se stará metoda *deleteFolder()* třídy *EmailAccount*. K zavolání metody pro smazání adresáře dojde, když uživatel klikne pravým tlačítkem myši na některý z uzlů ve stromu s IMAP adresáři e-mailových účtu a když v kontextové nabídce aktivuje volbu smazání adresáře.

7.3.10 Označení zprávy za přečtenou

Metoda *setMessageAsSeen()* třídy *EmailAccount* nastaví zprávě příznak, že byla přečtená uživatelem. Metoda pro změnu příznaku zprávy se volá v případě, kdy uživatel v grafickém uživatelském rozhraní v seznamu se zprávami vybere k zobrazení zprávu, kterou ještě nepřečetl.

7.4 Struktury pro práci s e-mailovými protokoly a parsery

Veškerý kód klienta, který komunikuje se servery protokolem IMAP je napsaný kolem externí knihovny *vmime*. Tato knihovna má jako závislost boost framework a používá jeho implementaci ukazatelů s automatickou správou paměti.

Knihovna z hlaviček i obsahu e-mailů vrací všechna data jako textové řetězce ve vlastním řetězcovém datovém typu, kde jsou znaky uloženy v 7bitovém ASCII. Znaky s vyšší hodnotou než 128 v ASCII tabulce jsou kódovány v BASE64. Přesně v takovém formátu je obsah celého e-mailu přenášen po síti protokoly POP3, IMAP a SMTP. Knihovna pak nabízí prostředky pro získání znakové sady přijatého e-mailu a umožňuje převést textové řetězce do textového řetězce z C++ ve znakové sadě, jakou si zvolí programátor. Všechny textové informace z přijatých e-mailů aplikace převádí do znakové sady UTF-8.

Knihovna *vmime* neumí načíst kořenové certifikáty ze systému, které jsou nutné k ověření důvěryhodnosti SSL spojení. Získat platformově nezávisle kořenové certifikáty ale dokáže Qt framework. V programu není implementovaná podpora přidávání vlastních self-signed certifikátů ani možnost dávat neplatným certifikátům výjimky. Program podporuje autentikované spojení k IMAP serverům, které může být nešifrované, nebo šifrované pomocí SSL.

Knihovna *vmime* sice oficiálně podporuje protokol POP3, při stahování zpráv tímto protokolem jsem ale narazil na spoustu problémů. Knihovna dokázala z hlavičky zprávy získat pouze informaci o předmětu, pokusy o stažení informací o příjemcích zprávy nebo o datu odeslání zprávy skončily výjimkou s textem *not implemented*. U multipart zpráv knihovna dokázala získat pouze první textovou část, takže se u zpráv s větším množstvím textu uživateli nezobrazí celý její obsah.

Po domluvě s vývojáři Otteru a vedoucím práce jsem se rozhodl podporu protokolu POP3 do e-mailového klienta nezahrnout. Protokol POP3 je v návrhu s ohledem na potřeby dnešní doby silně omezený, protokol IMAP řeší všechny jeho nedostatky. Dnes taky prakticky neexistuje žádný poskytovatel e-mailových služeb, který by protokol IMAP nepodporoval.

Na další velkou limitaci knihovny jsem narazil v závěru psaní bakalářské práce. Praktickou část práce jsem celou dobu psal v operačním systému Linux, kde s knihovnou nebyly problémy. Po zkompilování knihovny *vmime* na platformě Windows nejsou ve výsledném dll souboru s knihovnou exportované symboly tříd a metod, takže jakákoliv aplikace používající tuto knihovnu nejde přeložit. Autoři knihovny přitom deklarují, že je knihovna multiplatformní. Autorům knihovny jsem o problému psal na jejich oficiálním Google Groups fóru a na Githubu. V době odevzdání práce (přibližně dva měsíce od nahlášení problému) ale na problém nikdo z autorů knihovny nereagoval. Z tohoto důvodu jiný operační systém než Linux v bakalářské práci nepodporuji.

Zprávy protokolem SMTP v e-mailovém modulu odesílá knihovna *simple-mail*. Zprávy lze posílat autentikovaně se SSL šifrováním nebo bez šifrování komunikace. Knihovna *simple-mail* nekontroluje důvěryhodnost SSL certifikátů. Zprávy jsem původně posílal knihovnou *vmime*, knihovna ale má problém připojit se k jakémukoliv SMTP serveru z prostředí školní sítě. Na jinou síť, ze které by zprávy knihovnou *vmime* odeslat nešly jsem nenarazil. Knihovna *simple-mail* dokáže odesílat poštu i ze školní sítě.

7.4.1 Parser metadat zpráv

Parser metadat zpráv získává informace o zprávě přímo z hlavičky zprávy. Knihovna *vmime* má vlastní parser, který zprávu serializuje do C++ objektu, ze kterého se dá k informacím o zprávě přistupovat mnohem jednodušeji, serializace jednoho objektu ale trvá 3 až 4 sekundy. Tento parser se naprosto nehodí v situaci, kdy si e-mailový klient stáhne ze serveru metadata všech zpráv a ty pak zpracovává. Vytažení informací přímo z hlavičky e-mailu proběhne okamžitě, metadata zpráv proto získávám touto nízkoúrovňovější metodou. Vlastní parser metadat je v e-mailovém klientu implementovaný ve třídě *VmimeMessageMetadataParser*.

Knihovna *vmime* umí po získání hlavičky e-mailu vytvořit objekt s příznaky zprávy. Z tohoto objektu si můj parser získá informace o tom, jestli zpráva byla přečtena, smazána nebo zda je označena jako koncept. Z hlavičky zprávy parser dále získává univerzální identifikátor zprávy a její velikost.

Mírně složitější je situace u zpracování předmětu, odesílatele, seznamu kontaktů pro odpověď (reply-to) a u data a času odeslání zprávy. Knihovna vrací text s předmětem ve formátu *"Subject: Obsah předmětu"*, text s odesílatelem ve formátu *"From: jméno <jmeno@server>"* a seznam příjemců odpovědi ve formátu *"Reply-To: jméno1 <jméno1@server>, jméno2 <jméno2@server>, ..."* a to včetně uvozovek. Parser u každého z těchto polí odmaže jméno hlavičky a uvozovky, u odesílatele ještě vytvoří objekt s kontaktem a u příjemců odpovědi vytvoří seznam kontaktů.

Datum a čas odeslání zprávy knihovna vrátí ve formátu *"Date: Fri, 5 Aug 2011 05:22:45 -0700"*. I tady parser nejdříve odstraní uvozovky a jméno hlavičky, ze zbylého textového řetězce pak vytvoří a nastaví *QDateTime* objekt.

7.4.2 Parser obsahu zpráv

Parser obsahu zpráv pracuje už se zprávou naserializovanou do objektu knihovny *vmime*. Parser ze zprávy získává její obsah v HTML a v neformátovaném textu, přílohy, seznam příjemců zprávy, seznam příjemců zprávy v kopii a u HTML zpráv vložené objekty. V e-mailovém klientu je funkcionalita pro parsování zpráv implementována ve třídě *VmimeMessageContentParser*.

Parser získává obsah zprávy v neformátovaném textu a v HTML textu tak, že iteruje přes všechny textové části (text parts) multipart zprávy. Každá textová část obsahuje informaci o tom, zda se jedná o čistý text nebo HTML. HTML text zprávy získaný parserem je pak sjednocením obsahu všech textových částí, které jsou v HTML a neformátovaný textový obsah zprávy je sjednocení všech textových částí, které jsou v neformátovaném textu.

Textová část zprávy ve formátu HTML může obsahovat i vložené objekty. Nejčastěji se jedná o obrázky, kaskádové styly, soubory s událostmi do kalendáře, nebo soubory s kontakty, které odesílatel posílá přímo jako součást e-mailu. E-mailový klient umí pracovat i s takovými objekty. Každý takový objekt kromě dat obsahuje i jméno, informaci o typu souboru (mediaType) a jednoznačný identifikátor *contentId*. Parser projde všechny HTML části multipart zprávy a vložené objekty uloží do seznamu. E-mailový klient pak při zobrazení HTML zprávy každý odkaz na vložený objekt v HTML nahradí jeho daty v BASE64 formátu a grafická komponenta pak obsah zprávy i s objektem správně vykreslí.

Parser dále ze serializovaného objektu zprávy knihovny *vmime* získává přílohy. Pro každý soubor s přílohou si parser uloží její jméno, data a typ souboru.

Získání informací o příjemcích zprávy a příjemcích zprávy v kopii je přímočaré. Knihovna *vmime* pro oba případy vrací kolekci s adresami, které reprezentují e-mailovou skupinu nebo kontakt, parser z těchto objektů získává jména a e-mailové adresy kontaktů.

7.4.3 Parser IMAP adresářů

O získání informací o adresáři z IMAP účtu se v e-mailovém klientu stará třída *VmimeInboxFolderParser*. Knihovna *vmime* pro každý adresář dokáže vrátit objekt s příznaky, jméno adresáře a odkaz na rodičovský adresář.

Z příznaků o adresáři parser získává informace o tom, zda má adresář funkci koše, archivu, konceptů, spamu nebo odeslané pošty.

Plnou cestu k adresáři od kořene parser získává přes odkazy na rodičovské adresáře rekurzí, kde postupně ze jmen adresářů skládá cestu.

7.4.4 Generování zprávy k odeslání

Zprávu pro odeslání z objektů e-mailového klienta skládá dohromady a odesílá metoda *sendMessage()* třídy *SimpleMailSmtpService*. Knihovna *simple-mail* obsahuje třídu *MimeMessage*, objektu této třídy kód e-mailového modulu nastaví

všechna data zprávy k odeslání. Připravenou zprávu pak odešle metoda *sendMail()* objektu třídy *Sender* z knihovny *simple-mail*.

Metoda *sendMessage()* pro posílání zprávy nijak nekontroluje správnost dat zprávy, o kontrolu se stará grafické uživatelské rozhraní, které v případě nekonzistentních dat uživateli neumožní zprávu vůbec odeslat.

Závěr

E-mailový klient pro prohlížeč Otter poskytuje uživateli základní funkcionalitu pro stahování a zobrazování e-mailů z poštovního serveru protokolem IMAP a pro psaní a odesílání zpráv protokolem SMTP.

Modul umí pracovat s větším množstvím e-mailových účtů zároveň. Klient umožňuje uživateli číst jednoduché zprávy v neformátovaném textu a složitější zprávy složené z více částí, které mohou mít i obsah ve formátu HTML. Klient si poradí i se zobrazením vložených objektů v HTML zprávách a dává uživateli možnost uložit si přílohy ze zpráv jako soubory na disk. Modul dále umí kopírovat a přesouvat zprávy napříč adresáři v IMAP účtu, mazat zprávy, označovat zprávy při zobrazení jako přečtené na serveru a vytvářet nebo mazat adresáře v IMAP účtu. E-mailový klient neumí k poště přistupovat starším protokolem POP3, který má oproti protokolu IMAP spoustu omezení.

E-mailový klient poskytuje prostředí pro psaní jednoduchých zpráv v neformátovaném textu. Posílaná zpráva může mít větší množství příjemců i adresátů v kopii a ve skryté kopii. Klient umí ke zprávě připojit přílohy. Dále je možné přeposílat zprávy a odpovídat na zprávy stažené z poštovního serveru, tady klient umí přeposlat i případný HTML obsah. Editační pole pro příjemce zprávy, příjemce zprávy v kopii a příjemce zprávy ve skryté kopii obsahují i našeptávač se seznamem všech kontaktů, které jsou z metadat zpráv uloženy v databázi.

Klient si ukládá kopie všech metadat zpráv z e-mailových účtů uživatele a obsah přečtených zpráv do SQLite databáze. Informace o konfiguraci e-mailových účtů jsou uloženy v JSON souboru. Oba dva soubory jsou uloženy na standardní cestě pro umístění dat aplikací operačního systému uživatele.

Otter Browser je vyvíjen multiplatformně, autoři na webu projektu nabízejí ke stažení instalační balíčky pro operační systémy Windows, Linux a Mac OS X. Prohlížeč má největší uživatelskou základnu na operačním systému Linux. Mou verzi prohlížeče a e-mailový modul jsem vyvíjel a testoval taktéž na operačním systému Linux. Má verze Otter Browseru s e-mailovým klientem kvůli chybám v používané knihovně třetí strany nepodporuje platformu Microsoft Windows. Na operačním systému Mac OS X není program otestovaný, předpokládá se ale, že má verze Otter Browseru bude fungovat i tam a i na ostatních operačních systémech unixového typu, na kterých jdou přeložit externí knihovny používané Otter Browserem a Qt framework.

Conclusions

E-mail module for Otter Browser provides the basic functionality for previewing e-mail messages fetched from e-mail servers via IMAP protocol and for composing and sending e-mail messages via SMTP protocol.

Module is able to work with multiple e-mail accounts at the same time. E-mail client allows user to read simple e-mail messages with the plain text content and more complex messages with multipart and HTML content. Client is capable to display embedded objects in HTML messages and allows user to save attached files from messages as files to the hard drive. E-mail module is able to move and copy e-mail messages across folders in the IMAP account, delete e-mail messages, mark e-mail messages as seen and to create, remove and rename IMAP folders. E-mail module doesn't support fetching messages via older and limited POP3 protocol.

E-mail client provides user interface for composing simple messages in plain text. Client can send composed e-mail message to multiple recipients, recipients in copy and recipients in blind copy. Client is able to attach files to the composed message. E-mail module is able to reply and forward e-mail messages with content in both plain text and HTML formats. Text fields for recipients, copy recipients and blind copy recipients in the user interface contain a completer with list of all contacts from user's accounts that are stored in the local database.

E-mail module stores copies of all messages metadata from all e-mail accounts and content of fetched messages in the local SQLite database. Information about e-mail accounts configuration is stored in the JSON file. Both files are stored on the standard path for storing application data on the user's platform.

Otter Browser is developed as a multiplatform application, there are available installation packages for Microsoft Windows, GNU/Linux and Apple Mac OS X platforms on the project's site. Otter Browser has the biggest user base on Linux based systems. My fork of Otter Browser with e-mail module was as well developed and tested on the Linux platform. My fork of Otter Browser is not able to run on Microsoft Windows platform due to bugs in used third-party libraries. Application is not tested on Mac OS X, but it is assumed that my fork of Otter Browser is able to run on every unix-based operating system, where it is possible to compile used third-party libraries and the Qt framework.

A Obsah přiloženého CD/DVD

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

bin/
: Spustitelný soubor programu OTTER-BROWSER. Program byl přeložen na linuxové distribuci Fedora 28 s Qt frameworkem ve verzi 5.10.1.

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 OTTER-BROWSER 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 OTTER-BROWSER, včetně všech požadavků pro jeho bezproblémový provoz.

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`.

Literatura

[1] *VMime C++ MIME Library*
https://www.vmime.org/.

[2] *SimpleMail Library*
https://github.com/cutelyst/simple-mail.

[3] *Simple Mail Transfer Protocol*
https://tools.ietf.org/html/rfc5321.

[4] *Post Office Protocol 3*
https://tools.ietf.org/html/rfc1939.

[5] *Internet Message Access Protocol*
https://tools.ietf.org/html/rfc3501.

[6] *RFC 821 - Simple Mail Transfer Protocol*
https://tools.ietf.org/html/rfc821.

[7] *RFC 5321 - Simple Mail Transfer Protocol*
https://tools.ietf.org/html/rfc5321.

[8] *RFC 1939 - Post Office Protocol - Version 3*
https://tools.ietf.org/html/rfc1939.

[9] *Multipurpose Internet Mail Extensions*
https://tools.ietf.org/html/rfc2045.

[10] *RFC 1064 - Interactive Mail Access Protocol - Version 2*
https://tools.ietf.org/html/rfc1064.

[11] *RFC 3501 - Interactive Mail Access Protocol - Version 4rev1*
https://tools.ietf.org/html/rfc3501.

[12] *RFC 2822 - Internet Message Format*
https://tools.ietf.org/html/rfc2822.