ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název: Integrace Thunderbirdu s Exchange
Student: Karel Gudera
Vedoucí: Ing. Ondřej Guth, Ph.D.
Studijní program: Informatika
Studijní obor: Informační technologie
Katedra: Katedra počítačových systémů
Platnost zadání: Do konce letního semestru 2017/18

Pokyny pro vypracování

Navrhněte a implementujte podporu serveru Exchange v poštovním klientu Thunderbird. Podporu realizujte jako zásuvný modul, který bude fungovat pod OS Linux a Windows a bez toho, aby uživatel musel instalovat nebo spouštět další software. Funkčnost omezte na odesílání a přijímání mailů. Ke komunikaci s Exchange nepoužívejte protokol IMAP ani POP3, ale pouze MAPI nebo EWS. Modul bude dostupný jako svobodný software (opensource). Řešení vhodnými prostředky otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 18. listopadu 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalářská práce

Integrace Thunderbirdu s Exchange

Karel Gudera

Vedoucí práce: Ing. Ondřej Guth Ph.D.

9. května 2017

Poděkování

Chtěl bych poděkovat svému vedoucímu práce Ing. Ondřeji Guthovi, Ph.D., za vedení této práce, plnou podporu a nadšení při tvorbě. Dále chci poděkovat všem mým blízkým, kteří mě podporovali během celého studia. V neposlední řadě děkuji všem, kteří mi pomohli s testováním výsledného produktu a vyplněním příslušného online průzkumu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 9. května 2017 .....

Odkaz na tuto práci

Gudera, Karel. *Integrace Thunderbirdu s Exchange*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato práce se zabývá propojením (neboli integrací) poštovního klienta Mozilla Thunderbird s e-mailovým serverem Microsoft Exchange pomocí Exchange Web Services (EWS). Řešení je realizováno jako zásuvný modul, který se do klienta Thunderbird musí nainstalovat. Mezi hlavní cíle této práce patří uživatelská přívětivost při instalaci modulu i jeho používání. Dále by modul měl být nezávislá jednotka, tzn. nemělo by být nutné, aby uživatel musel instalovat nebo spouštět další software pro správný chod. Funkčnost modulu je omezena na přijímání a odesílání e-mailových zpráv. Modul by měl být dostupný pro operační systémy Microsoft Windows a Linux jako svobodný software s otevřeným kódem (opensource). Jak všech cílů bylo dosaženo je popsáno v této práci.

Klíčová slova poštovní klient, Mozilla Thunderbird, zásuvný modul, Microsoft Exchange, Exchange Web Services, integrace, uživatelská přívětivost

Abstract

This thesis deals with interconnection (or integration) of e-mail client Mozilla Thunderbird and e-mail server Microsoft Exchange via Exchange Web Services (EWS). Solution is implemented as a plugin (or add-on) which must be installed in Thunderbird. Main goals are user-friendliness during plugin installation and during it's usage. Plugin should be independent of other software, this means that user should not have to install anything else for the plugin to work properly. Functionality is limited to receiving and sending e-mail messages. Plugin should be available for Microsoft Windows and Linux as free opensource software. How all the goals have been achieved is described in this thesis.

Keywords e-mail client, Mozilla Thunderbird, plugin, Microsoft Exchange, Exchange Web Services, integration, user-friendliness

Obsah

Seznam obrázků

Úvod

Používáni e-mailu se v dnešní době stalo samozřejmostí asi každého z nás. Mnozí využívají e-mail prostřednictvím webového prohlížeče, ale najde se velká skupina lidí, kteří používají tzv. e-mailový klient jako je např. Mozilla Thunderbird. Chybí zde však funkčnost, která umožňuje komunikovat s e-mailovým serverem od firmy Microsoft. A právě zajištění této funkčnosti bude předmětem této bakalářské práce.

Práce bude prospěšná všem, kteří rádi používají e-mailový klient Mozilla Thunderbird a potřebují se spojit s výše zmiňovaným e-mailovým serverem od firmy Microsoft.

Toto téma jsem si zvolil, protože doposud neexistuje bezplatné řešení tohoto problému pro daný e-mailový klient. Zároveň chci uživatelům Mozilla Thunderbird poskytnout zcela jednoduchý způsob, jak docílit kýžené funkcionality pomocí zásuvného modulu, který je velmi jednoduché nainstalovat, popřípadě odinstalovat.

V práci se zabývám hlavně návrhem a implementací daného zásuvného modulu. Mimo jiné je součástí práce i testování, které by mělo potvrdit správné fungování modulu a jeho uživatelskou přívětivost.

Struktura práce

V první kapitole se seznámíme s principy fungování e-mailů a ukážeme, jak se liší e-mail Microsoft Exchange a jaké technologie používá.

Ve druhé kapitole provedeme analýzu současných řešení a prodiskutujeme jejich výhody a nevýhody.

Ve třetí kapitole se budeme zabývat samotným návrhem zásuvného modulu. To zahrnuje výběr vhodných technologií, které nám zajistí komunikaci se serverem Microsoft Exchange a napojení těchto technologií na zásuvný modul.

Ve čtvrté kapitole nastíníme implementaci řešení. Nebudeme však zabíhat do přílišné hloubky implementačních detailů.

Pátá a poslední kapitola se zabývá testováním správného fungování vytvořeného modulu.

Cíle práce

Cílem rešeršní části je vysvětlení principů z oblasti fungování tradičních e-mailových protokolů a poukázat na odlišnost mezi těmito protokoly a protokolem, který používá server Microsoft Exchange. Dále je cílem analýza současných řešení a prodiskutování jejich silných a slabých stránek. V neposlední řadě je cílem nastudování vhodných technologií, které nám pomohou vyřešit komunikaci s výše zmiňovaným serverem Microsoft Exchange.

Cílem praktické části je navrhnout, implementovat a vhodným způsobem otestovat zásuvný modul pro poštovní klient Mozilla Thunderbird. Důraz je kladen především na uživatelskou přívětivost a nezávislost na jiných programech. Zásuvný modul by měl umět přijímat a posílat e-mailové zprávy a měl by být podporován operačními systémy Microsoft Windows a Linux. Modul bude dostupný jako svobodný software s otevřeným kódem (opensource).

Kapitola 1

Seznámení s problematikou

1.1 Tradiční protokoly

Pro komunikaci mezi e-mailovým klientem a e-mailovým serverem už několik let slouží známé protokoly. Tyto protokoly jsou velice rozšířené a podporuje je většina klientů a serverů. Jedná se zejména o následující.

- **IMAP** [1]
- **POP3** [2]
- **SMTP** [3]

1.2 Exchange Web Services (EWS)

Na druhé straně máme EWS [4], což je služba poskytovaná výhradně serverem Microsoft Exchange. Tato služba je založená na protokolu SOAP (Simple Object Access Protocol), což je více méně protokol pro posílání XML zpráv přes HTTP nebo HTTPS [5]. Microsoft Exchange samozřejmě podporuje i tradiční protokoly, ale často se stává, že jsou vypnuty a nám nezbývá nic jiného než komunikovat se serverem pomocí EWS. Naneštěstí už ne všichni e-mailoví klienti podporují tuto funkcionalitu.

1.3 Messaging Application Programming Interface (MAPI)

MAPI [6] je další alternativou ke komunikaci s Exchange serverem. V posledních letech je MAPI vytlačováno EWS, proto se také v této práci zaměřím výhradně na EWS.

1.4 Název modulu

Jméno modulu jsem zvolil **MExInt**. Název vznikl ze slov **M**icrosoft **Ex**change **Int**egration.

Kapitola 2

Přehled současných řešení

V současné době existuje několik řešení pro komunikaci se serverem Microsoft Exchange pomocí EWS. Většina řešení je v podobě poštovních klientů pro operační systém Microsoft Windows a jen hrstka je dostupná pro Linux. Zde se pokusím stručně shrnout zmiňovanou menšinu.

2.1 ExQuilla

ExQuilla [7] je zásuvný modul pro Thunderbird. Pro komunikaci se serverem Exchange používá C++ knihovnu a propojení s modulem zajišťuje rozhraní XPCOM [8]. Modul umí pracovat s e-maily a navíc ještě s kontakty. Naše řešení bude mít nejvíce společného právě s ExQuillou. Nevýhodou je, že se jedná o placený produkt.

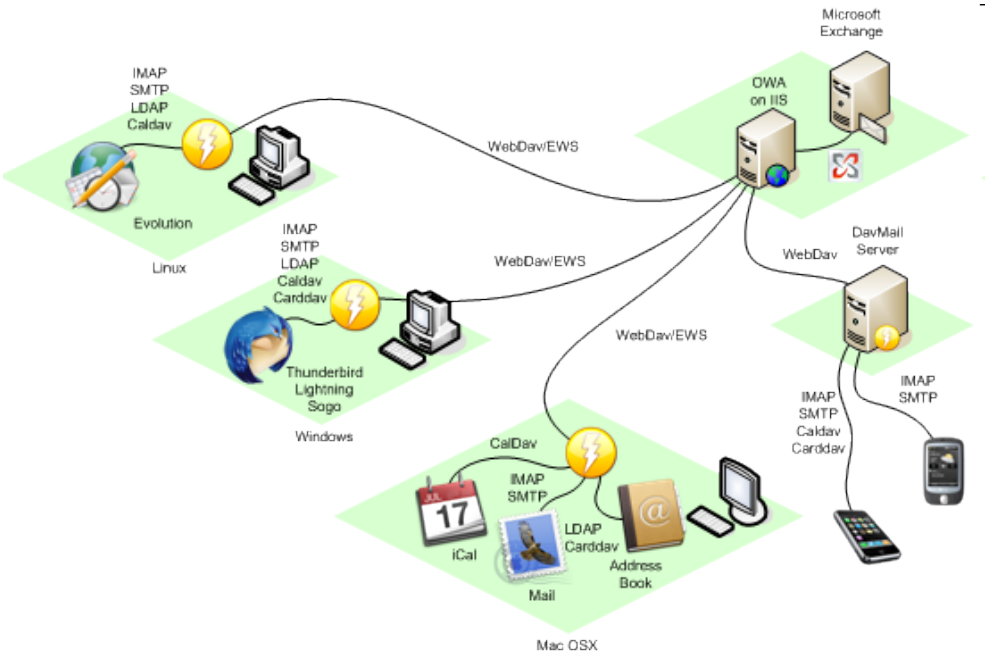
2.2 Evolution

Evolution [9] je e-mailový klient pro Linux, konkrétně pro grafické prostředí GNOME. Samozřejmě podporuje komunikaci se serverem Exchange a mimo práce s e-maily umí pracovat i s kalendářem a kontakty. Vcelku se jedná o velmi pěknou a uživatelsky přívětivou aplikaci. Jedinou nevýhodu tohoto klienta je, že je dostupný pouze pro zmiňované grafické prostředí GNOME.

2.3 DavMail Gateway

Jak už název napovídá, DavMail [10] je brána, která převádí protokol používaný serverem Exchange na protokoly SMTP, IMAP a jiné. Hezké znázornění, jak DavMail funguje, můžete vidět na obrázku 2.1. Aplikace je napsána v jazyce Java, tudíž je zcela multiplatformní. Toto řešení má jen malou obtíž v tom, že je potřeba mít aplikaci spuštěnou na pozadí vždy, když chceme používat e-mailový klient. Dále se může zdát malinko komplikovanější na konfigu-

raci, a to zejména pro méně znalé uživatele, kteří nejsou seznámeni s termíny jako je např. port nebo protokol.



Obrázek 2.1: DavMail architektura [11]

2.4 Exchange EWS Provider

Exchange EWS Provider [12] je zásuvný modul pro Thunderbird, který umožňuje komunikovat se serverem Exchange. Bohužel modul přináší jen podporu pro kontakty a kalendář Lightning, který musí být nainstalovaný zvlášť jako další modul. Kalendář Lightning už je v novějších verzích Thunderbirdu automaticky předinstalován.

2.5 exchange-ews-thunderbird

Tento modul jsem objevil po delší době na stránkách Github [13]. Jedná se vlastně o tři moduly poskytující práci s e-mailem, kalendářem a kontakty pomocí EWS. Modul je svou strukturou velice podobný ExQuille, tedy využívá nativní C++ knihovny a rozhraní XPCOM. Projekt je ovšem neudržovaný a nefunguje s nejnovější verzí Thunderbirdu. Osobně se mi podařilo modul zprovoznit s verzí Thunderbird 38, což je jediná podporovaná verze. Na novější verze Thunderbirdu modul nelze nainstalovat. Zkusil jsem upravit soubor

„install.rdf“, nastavit maximální podporovanou verzi na 45, která je aktuální, a nainstalovat modul. I když se modul podařilo nainstalovat, tak nepracoval správně. Potíž je v tom, že se modul snaží načítat knihovny, které už v nových verzích Thunderbirdu nejsou, jedná se například o knihovnu „mozalloc“. Modul ve staré verzi Thunderbirdu podporuje většinu základních operací (přijímání a odesílání zpráv, práce se složkami, přesouvání, kopírování, ...).

Kapitola 3

Návrh řešení

Při návrhu vnitřní struktury zásuvného modulu se vyskytlo více možností řešení. Jedná se hlavně o výběr vhodné knihovny, která nám zajistí komunikaci se serverem Exchange. S tím souvisí problém napojení knihovny na náš modul. V této kapitole se pokusíme shrnout všechna pro a proti jednotlivých přístupů.

3.1 Přehled knihoven

3.1.1 ews-cpp

Jedná se o knihovnu napsanou v jazyce C++. Implementuje základní operace, jako je načítání e-mailových zpráv ze serveru, jejich mazání a posílání. Mimo jiné jsou zde i některé operace pracující s kalendářem a kontakty. Chybí zde však např. možnost práce se složkami nebo kopírování a přesouvání zpráv. Knihovna funguje jen se serverem Microsoft Exchange 2013. Více o ews-cpp můžete najít na stránkách Github [14].

3.1.2 ews-javascript-api

Ews-javascript-api [15] je knihovna napsaná v jazyce Javascript. Nejde však o čistý Javascript, ale o variantu podporovanou softwarovým systémem Node.js [16]. Tato knihovna je velice rozsáhlá a podporuje skoro všechny operace pro komunikaci se serverem. Další velkou výhodou je podpora Microsoft Exchange 2007–2016 a Office 365.

3.1.3 Ostatní

Existuje ještě celá řada dalších knihoven napsaných např. v jazyce Java či Python. Takovými se zde ale zabývat nebudeme, jelikož odporují požadavku nezávislosti modulu na jiných programech.

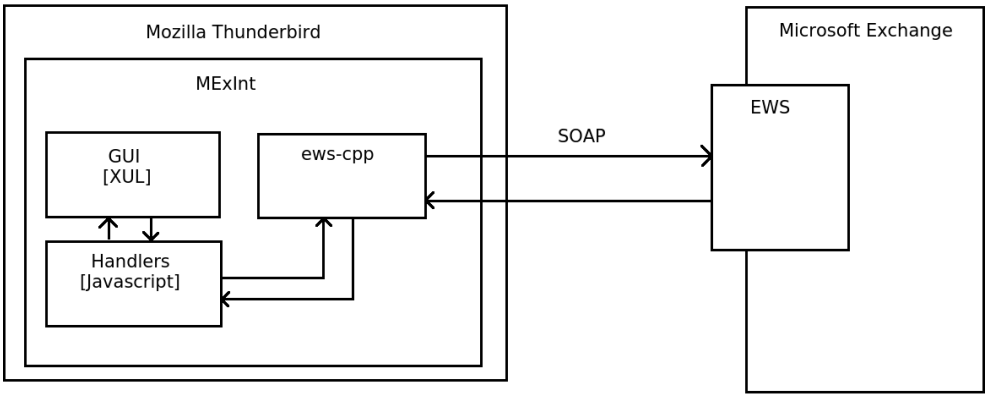
3.2 Napojení knihovny na modul

3.2.1 Na co vlastně napojujeme

Ještě před samotným popisem technik napojení je potřeba vyjasnit jaké technologie jsou použity pro vývoj zásuvného modulu. Mozilla Thunderbird používá jazyk XUL [17] pro tvorbu uživatelského rozhraní a Javascript pro zpracování požadavků z tohoto rozhraní. Naším cílem je tedy umožnit používání některé z knihoven právě z Javascriptu.

3.2.2 Napojení ews-cpp

Jelikož je implementačním jazykem C++, je nutné vytvořit nativní knihovnu pro každý operační systém zvlášť („.so“ pro Linux a „.dll“ pro Windows). Jakmile máme knihovnu hotovou a připravenou k použití, nastává otázka jak ji začít používat pomocí Javascriptu. Jedno řešení je už zmiňovaný XPCOM, což je rozhraní umožňující načíst a používat knihovní C/C++ funkce přímo v Javascriptu. Jedná se ale o zastaralou technologii od společnosti Mozilla. Mozilla doporučuje v současnosti používat technologii js-ctypes [18], která je jednodušší na používání. Jak vypadá architektura modulu při použití této knihovny můžete vidět na obrázku 3.1.

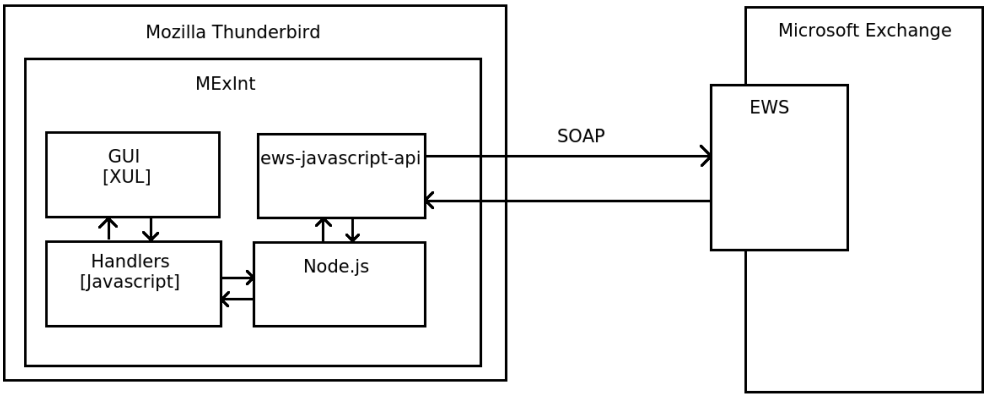


Obrázek 3.1: Architektura modulu (ews-cpp)

3.2.3 Napojení ews-javascript-api

Mohlo by se zdát, že použití Javascriptové knihovny v Javascriptu je na první pohled transparentní a jasné. Opak je ale pravdou. Knihovna je určena pro Node.js, což je jiný systém, než systém, který používá Mozilla Thunderbird. V současné době neexistuje žádné jednoduché řešení, které by dovolovalo používat knihovny určené pro Node.js v modulech pro Thunderbird nebo Firefox

[19]. Budeme tedy muset tento problém obejít. Jediná možnost je zahrnout interpret Node.js do modulu, tak aby byl jeho součástí. Tím dodržíme požadavek, aby byl modul nezávislou jednotkou. Vždy, když bude potřeba komunikovat se serverem Exchange, zavoláme z Javascriptu interpret Node.js a jako argument mu předáme skript, který má vykonat, popřípadě i jiné argumenty. Data, která vyprodukuje interpret můžeme získávat přes jeho standardní výstup. Obdobně si můžete prohlédnout architekturu modulu při použití této knihovny na obrázku 3.2.



Obrázek 3.2: Architektura modulu (ews-javascript-api)

3.3 Srovnání

Zde stručně shrneme výhody a nevýhody obou přístupů.

3.3.1 ews-cpp

+ Možnost pracovat s knihovnou přímo v Javascriptu
– Menší množství podporovaných operací
– Podpora pouze Microsoft Exchange 2013
– Nenabízí velký prostor pro budoucí rozšíření
– Nutnost kompilace pro cílové platformy

3.3.2 ews-javascript-api

+ Velké množství podporovaných operací
+ Podpora Microsoft Exchange 2007–2016 a Office 365

+ Lepší vyhlídky pro budoucí rozšíření modulu
+ Binární soubor interpretu je připraven pro všechny platformy (navíc i pro Mac OS X)
+ Veškerý napsaný kód je zcela multiplatformní, resp. závislý na Thunderbirdu a Node.js
– Nutnost volání nového procesu při použití knihovny
– Nutnost vložit binární soubor interpretu do modulu pro všechny cílové platformy

3.4 Zvolené řešení

Z důvodu převažujících kladů jsem se rozhodl použít ews-javascript-api.

3.5 Nezávislost

Mohlo by se zdát trochu pochybné, zda je modul opravdu nezávislý, protože Node.js je binární soubor a pro svůj běh potřebuje dynamicky linkované knihovny. Nicméně se jedná o opravdu základní knihovny, které bývají nainstalované na všech operačních systémech. Pokud by ovšem i toto byl problém, tak je možné binární soubor Node.js sestavit ze zdrojového kódu staticky linkovaný.

Zde je výstup programu „ldd“, který na Linuxu vypíše seznam knihoven potřebných pro běh programu. Pokud použijeme příkaz „ldd“ na binární soubor Thunderbirdu, dostaneme úplně stejný seznam.

```
linux-vdso.so.1 =>  (0x00007ffcf17af000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f8c7fba2000)
librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00007f8c7f99a000)
libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f8c7f617000)
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f8c7f30e000)
libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f8c7f0f8000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f8c7eeda000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f8c7eb11000)
/lib64/ld-linux-x86-64.so.2 (0x0000564e7f4c6000)
```

Obdobně, zde je seznam knihoven potřebných pro správný chod programu na operačním systému Windows. Všechny knihovny jsou standardní součástí operačního systému [20].

- WS2_32.DLL

- KERNEL32.DLL
- WINMM.DLL
- ADVAPI32.DLL
- USER32.DLL
- GDI32.DLL
- IPHLPAPI.DLL
- PSAPI.DLL
- USERENV.DLL

3.6 Bezpečnost

3.6.1 Zabezpečení spojení

Jelikož ke komunikaci používáme protokol SOAP, který je založen protokolu HTTP, musí být spojení zabezpečeno pomocí SSL/TLS (HTTPS). Jak ews-javascript-api, tak Microsoft Exchange HTTPS podporují.

3.6.2 Autentizace

Microsoft Exchange

Microsoft Exchange nabízí řadu možností pro autentizaci.

- HTTP Basic Authentication
- HTTP Digest Authentication
- NTLM
- Kerberos
- OAuth

ews-javascript-api

Ews-javascript-api implementuje jen některé z těchto možností.

- HTTP Basic Authentication – Zabudováno přímo v knihovně.
- NTLM – Je možné za použití tzv. NTLM XHRApi adaptéru [21]. Ews-javascript-api je nyní ve verzi 0.8.0 a předpokládá se, že ve verzi 1.0 bude podpora NTLM integrována přímo v něm.
- OAuth – Vyžaduje dodatečnou implementaci.

Řešení

Pro naše účely se omezíme na HTTP Basic Autentizaci na zabezpečeném (SSL/TLS) spojení. Tato možnost poskytuje dostatečné zabezpečení a dále je výhodná, protože ji podporují všechny verze Microsoft Exchange (2007–2016) a také Office 365.

3.6.3 Kontrola certifikátu

Při každém požadavku na server provádí ews-javascript-api resp. Node.js kontrolu bezpečnostního certifikátu serveru. Pokud je certifikát neplatný nebo jeho řetězec důvěry nevede k věrohodné certifikační autoritě, není možné se serverem navázat spojení.

Node.js má seznam tzv. Well-Known certifikačních autorit zakompilován přímo v sobě, takže je nemožné s ním nějak manipulovat. Z toho plyne, že je nutné Node.js průběžně aktualizovat. Protože se jedná o první implementaci, vystačíme si se seznamem, který poskytuje Node.js, ale v budoucnu by bylo hezké k poskytovanému seznamu přidat certifikační autority, které nabízí samotný Thunderbird.

Pro účely lokálního testování jsem přidal možnost kontrolu certifikátu zcela vypnout. Předpokládám, že uživatelé modulu budou komunikovat s dobře nakonfigurovanými servery, které mají platné certifikáty, a že v případě vypnutí této kontroly si budou vědomi rizik, která jsou s tím spojeny. Největší bezpečnostní riziko spjaté s vypnutím této kontroly je útok typu „Man in the middle“.

Kapitola 4

Implementace řešení

Implementaci řešení jsem rozdělil do čtyř částí. První část se zabývá přidáním účtu Microsoft Exchange. Ve druhé části řešíme načítání e-mailových zpráv ze serveru. Ve třetí části se věnujeme odesílání zpráv. A nakonec se ve čtvrté části řeší odebrání účtu.

Subprocess

V naší implementaci hraje velkou roli spouštění externího procesu (podprocesu) z našeho modulu. Thunderbird pro tento účel nabízí „Subprocess“, který nám danou funkcionalitu zajistí. Jelikož se jedná o klíčový prvek v implementaci, přidal jsem jednoduchou ukázku kódu 4.1.

Velkou výhodou Subprocessu je, že je volán asynchronně a pokud nemusíme čekat na jeho dokončení, tak neblokuje další vykonávání programu, tím pádem nám „nezamrzne“ uživatelské rozhraní. Další výhodou je, že funkce „stdout“, resp. „stderr“, přes kterou získáváme standardní výstup, resp. standardní chybový výstup se volá několikrát během vykonávání procesu a data nám vrací po menších částech. Toto chování je velmi výhodné, protože tak můžeme např. ukládat zprávy do lokálního úložiště postupně a přitom uživatele informovat, která zpráva se aktuálně stahuje. Subprocess je spolu s dalšími rozhraními součástí Add-on SDK [22].

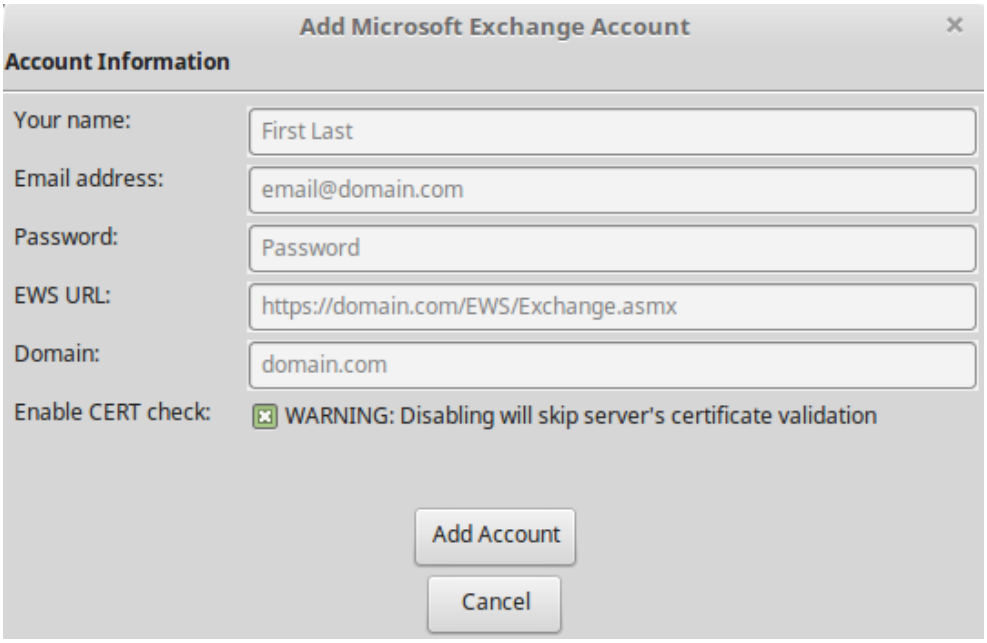
4.1 Přidání účtu

4.1.1 Uživatelské rozhraní

Jedná se o jediný větší návrh uživatelského rozhraní. Vše ostatní je řešeno v tzv. backendu. Uživatel si zvolí možnost přidat účet Microsoft Exchange a následně se mu objeví okno, do kterého vyplní informace potřebné k přihlášení k serveru (viz 4.1).

```
1  const subprocess = require('sdk/system/child_process/subprocess')
       ;
2  Components.utils.import('resource://gre/modules/FileUtils.jsm');
3
4  // path to node.js executable
5  var nodePath = FileUtils.getFile("ProfD", ["extensions", "
       mexint@karel.gudera", "components", "node"]);
6
7  // path to node.js script
8  var scriptPath = FileUtils.getFile("ProfD", ["extensions", "
       mexint@karel.gudera", "server", "script.js"]);
9
10 var exitCode;
11 var stdout = "";
12 var stderr = "";
13
14 var p = subprocess.call({
15     command: nodePath.path,
16     arguments: [scriptPath.path],
17     environment: ["FOO=BAR"],
18     charset: "UTF-8",
19     workdir: "/home/foo",
20
21     stdin: function (stdin) {
22         stdin.write("Write some data to stdin of process");
23         stdin.close();
24     },
25
26     stdout: function (data) {
27         stdout += data;
28     },
29
30     stderr: function (data) {
31         stderr += data;
32     },
33
34     done: function (result) {
35         exitCode = result.exitCode;
36     },
37
38     mergeStderr: false
39 });
40 p.wait(); // optionally wait for the subprocess to finish
```

Ukázka kódu 4.1: Volání podprocesu z modulu



Obrázek 4.1: Okno pro přídání účtu

Uživatel musí zadat své jméno, e-mailovou adresu a heslo. Dále je nutné vyplnit doménu a URL k Exchange Web Services, typicky má URL tvar „https://doména/EWS/Exchange.asmx“. Nakonec je zde ještě možnost vypnout kontrolu certifikátu, což vypne ověření, zda je certifikát serveru platný nebo podepsaný důvěryhodnou certifikační autoritou. Tato možnost je zde hlavně pro účel testování.

4.1.2 Backend

Pokud má uživatel vyplněné všechny informace a klikne na tlačítko „přidat účet“, veškerá další práce se přesouvá do pozadí a uživatel musí počkat, zda se podaří, či nepodaří účet přidat. V obou případech je o tom informován.

Následující kroky popisují tok programu při vytváření účtu.

1. Nejprve získáme informace z uživatelského rozhraní.

2. Poté pomocí Subprocess zavoláme Node.js a jako argumenty mu předáme informace z uživatelského rozhraní a jednoduchý skript, který vyšle požadavek na server a zjistí, zda jsou informace v pořádku. Pokud je něco špatně, tak skript vrátí chybovou hlášku.

3. Počkáme na dokončení procesu a pokud zjistíme, že nastala chyba, oznámíme to uživateli a končíme, jinak pokračujeme dál.

4. Pomocí rozhraní pro práci s účty načteme již existující účty a porovnáme je s účtem, který chce přidat uživatel. Pokud by uživatel chtěl přidat účet, který již existuje, oznámíme mu chybu a končíme, jinak pokračujeme.

5. Vytvoříme účet a přidáme mu dodatečné atributy, abychom později poznali, že se jedná o účet Microsoft Exchange.

6. Uložíme heslo, které uživatel zadal pomocí rozhraní pro bezpečné ukládáni hesel (v šifrované podobě).

7. Oznámíme uživateli, že účet byl úspěšně přidán.

4.1.3 Použitá rozhraní

Thunderbird umožňuje modulům přístup k rozhraním, která jsou implementována v C++. Přístup je zajištěn pomocí XPCOM. Jedná se o nejrůznější rozhraní, která poskytují možnost práce s uživatelskými účty, složkami, zprávami atd. Více o rozhraních, která Thunderbird nabízí můžete najít v dokumentaci nebo na stránkách Github [23, 24].

- nsIMsgAccountManager – Rozhraní poskytující přístup k uživatelským účtům, jejich modifikaci a vytváření.
- nsIMsgAccount – Rozhraní reprezentující uživatelský účet, skládá se navíc z rozhraní nsIMsgIdentity a nsIMsgIncomingServer (viz. níže).
- nsIMsgIdentity – Rozhraní reprezentující uživatelskou identitu, ukládá informace jako je jméno, příjmení či e-mail.
- nsIMsgIncomingServer – Rozhraní udržující informace potřebné ke komunikaci s e-mailovým serverem. Dále slouží přímo ke komunikaci se serverem a stará se o ukládání dat do lokálního úložiště. Toto rozhraní může být typu IMAP, POP3 atd.
- nsILoginManager – Rozhraní pro bezpečné ukládání hesel do databáze (v šifrované podobě).
- nsIPromptService – Poskytuje vyskakovací okna sloužící k informování uživatele.

Za zmínku stojí, že ve skutečnosti jsme nevytvořili opravdový účet Microsoft Exchange. Rozhraní pro práci s účty dovoluje vytvoření pouze účtů IMAP, POP3 a jiné. Já vybral účet typu POP3, protože je velice benevolentní a

dovoluje práci v offline módu. Účet IMAP nedovoluje žádnou akci bez perzistentního připojení k e-mailovému serveru, což je pro náš účel problém. Exquilla toto řeší tak, že si implementuje vlastní rozhraní v C++ (XPCOM komponenta) a to poté používá. Toto řešení je určitě dobré, ale zároveň velmi náročné a zdlouhavé.

4.2 Načítání zpráv

4.2.1 Uživatelské rozhraní

Standardně pokud klikneme pravým tlačítkem na nějaký účet v Thunderbirdu, objeví se nám nabídka, kde je možnost přijmout zprávy. Uživatelské rozhraní v tomto případě řeší zachycení této události a pokud se jedná o námi vytvořený Exchange účet, přesměruje obsluhu této události do našeho backendu.

4.2.2 Backend

Ještě před samotným popisem programu pro načítání zpráv je nutno zmínit, že server Exchange uchovává tzv. UniqueId pro každý objekt uložený v databázi, čímž ho jednoznačně identifikuje. Toho budeme využívat k jednoduché synchronizaci našich lokálně uložených zpráv se zprávami uloženými na serveru. Naše lokálně uložené zprávy mají atribut „messageId“.

Tok programu je následující.

1. Nejprve získáme všechny údaje z účtu potřebné ke komunikaci se serverem (e-mail, heslo, URL k EWS, ...).
2. Pomocí Subprocess zavoláme Node.js, předáme mu dané informace a skript, který vyšle požadavek na server a získá UniqueId všech zpráv uložených na serveru. Pokud by se nepodařilo na server připojit (například kvůli špatnému síťovému připojení), tak skript vrátí chybovou hlášku, další vykonávání programu ukončíme a uživatele informujeme o chybě.
3. Načteme messageId všech lokálně uložených zpráv.
4. Porovnáme seznam UniqueId načtených ze serveru se seznamem načteným lokálně a vytvoříme dvě množiny identifikátorů.
   - Množina identifikátorů, které jsou na serveru a nejsou v lokálním úložišti, tedy je potřeba je ze serveru načíst.
   - Množina identifikátorů, které nejsou na serveru a jsou v lokálním úložišti, takže je možné je z lokálního úložiště smazat.
5. Zprávy, které je možné smazat, smažeme.

6. Zprávy, které je potřeba načíst ze serveru, načteme. Opět tak, že zavoláme Node.js a jako argumenty mu předáme seznam identifikátorů zpráv, které je potřeba stáhnout, a skript, který stáhne MIME obsah daných zpráv.

7. Zprávy získané ze serveru uložíme pomocí metody, která umí zpracovat MIME obsah zprávy a uložit ji. Nově uložené zprávě nastavíme atribut messageId, který odpovídá UniqueId zprávě na serveru.

4.2.3 Použitá rozhraní

Jelikož je nutné získat informace o uživatelském účtu, jsou zde opět použita rozhraní, která už byla popsána výše. Navíc jsou zde použita následující rozhraní.

- nsIMsgFolder – Rozhraní reprezentující složku, nsIMsgIncomingServer obsahuje atribut „rootFolder“, což je kořenový adresář každého účtu.
- nsIMsgLocalMailFolder – Rozhraní poskytující metodu „addMessage“, která umí zpracovat MIME obsah, vytvořit z něj objekt reprezentující zprávu a uložit ji do dané složky.
- nsIMsgDBHdr – Rozhraní, které drží nejdůležitější informace o uložené zprávě. Jedná se např. o messageId, předmět a různé „flagy“. Pokud chceme ke zprávám přistupovat, musíme přes atribut „messages“, který se nachází v nsIMsgFolder a vrací enumerátor typu nsIMsgDBHdr na zprávy uložené ve složce resp. databázi.
- nsIActivityManager – Jde o rozhraní poskytující možnost posílat notifikace o různých aktivitách, např. kolikátá zpráva se aktuálně stahuje. Notifikace se zobrazují v levém dolním rohu Thunderbirdu.

4.3 Odesílání zpráv

4.3.1 Uživatelské rozhraní

Obdobně jako u načítání zpráv, uživatelské rozhraní slouží k zachycení události odesílání zprávy a přesměrování obsluhy do našeho backendu.

4.3.2 Backend

Zde narážíme na drobná omezení, která ews-javascript-api přináší. Naneštěstí neumí posílat e-mail s přílohami pro verzi Exchange menší než 2013. Ovšem toto omezení se dá obejít tak, že místo abychom předávali všechny údaje zvlášť (příjemce, předmět, kopie, příloha 1, příloha 2, ...), tak si necháme vygenerovat celý MIME obsah zprávy a ten předáme k odeslání. Thunderbird obsahuje

metodu „createRFC822Message“, která zmiňovaný MIME obsah umí vygenerovat.

Zde je popsáno, jak odesílání zprávy probíhá.

1. Získáme všechny potřebné údaje k odeslání zprávy a pro komunikaci se serverem.
2. Z opatřených údajů vygenerujeme MIME obsah e-mailové zprávy pomocí „createRFC822Message“.
3. Použijeme Subprocess pro zavolání Node.js a předáme mu MIME obsah zprávy spolu s údaji pro komunikaci se serverem a skriptem, který zprávu odešle.
4. Počkáme na dokončení procesu a pokud skončil úspěšně, informujeme uživatele, že zpráva byla odeslána. V opačném případě informujeme uživatele o chybě.

4.3.3 Použitá rozhraní

Zde opět popíši jen rozhraní, která doposud nebyla zmíněna.

- nsIMsgCompose – Rozhraní, přes které se přistupuje k dalším rozhraním jako jsou nsIEditor a nsIMsgCompFields. Jde o hlavní rozhraní při kompozici zprávy.
- nsIEditor – Rozhraní reprezentující textový editor. Nejdůležitější metoda, kterou toto rozhraní nabízí, je „outputToString“, která umí převést text napsaný v editoru do HTML formátu a vrátit příslušný řetězec.
- nsIMsgCompFields – Toto rozhraní udržuje informace o aktuálně sestavované zprávě. Jedná se např. o seznam příjemců, předmět zprávy, přílohy atd.
- nsIMsgAttachment – Rozhraní reprezentující přílohu. K přílohám se dá přistupovat přes atribut „attachments“ rozhraní nsIMsgCompFields, který vrací enumerátor typu nsIMsgAttachment.
- nsIMsgSend – Rozhraní poskytující metodu „createRFC822Message“, která umí vygenerovat MIME obsah a uložit ho do souboru. Ukládání do souboru probíhá asynchronně. K vygenerování obsahu používá nsIMsgCompFields a řetězec těla zprávy vygenerovaný pomocí „outputToString“.
- nsIMsgSendListener – Rozhraní použité k zachycení události, že zpráva byla uložena do souboru.
- nsIFile – Rozhraní reprezentující soubor.

- nsIFileInputStream – Rozhraní použité k načítání obsahu souboru.
- nsIBinaryInputStream – „Wrapper“ kolem nsIFileInputStream, který zjednodušuje načtení binárního obsahu souboru.

4.4 Odebrání účtu

4.4.1 Uživatelské rozhraní

Uživatelské rozhraní zde opět slouží k zachycení události odebírání účtu a přesměrování obsluhy směrem k nám.

4.4.2 Backend

Na rozdíl od předchozích případů, kde jsme zamezili výchozímu chování a přesměrovali celou obsluhu události k nám, tentokrát necháme Thunderbird, aby odebral účet sám. Náš úkol je pouze se postarat o řádné odstranění uživatelského hesla z databáze.

Odebrání účtu probíhá velice jednoduše.

1. Získáme informace o účtu, který se chystáme odebrat.
2. Necháme Thunderbird provést standardní odebrání účtu, při kterém je uživatel dotázán, zda chce účet opravdu odstranit.
3. Zkontrolujeme, zdali byl účet opravdu odebrán.
4. Pokud byl účet opravdu odebrán a zároveň se jednalo o námi vytvořený Exchange účet, odstraníme heslo z databáze.

4.4.3 Použitá rozhraní

Zde jsou použita pouze rozhraní pro práci s uživatelskými účty a hesly.

4.5 Rozšíření

Výše uvedená implementace řešení splňuje všechny požadavky zadání. Jedná se o opravdu základní funkčnost, tj. možnost stahovat zprávy pouze z „Inboxu“ a odesílat zprávy. Z příznivých časových důvodů jsem se rozhodl implementovat dodatečnou funkcionalitu. Zde stručně shrnu, o jakou funkcionalitu se jedná.

4.5.1 Základní složky

Jak jsem uvedl výše, zprávy je možné stahovat pouze z „Inboxu“. Toto rozšíření přidává možnost stahovat zprávy z dalších základních složek, které jsou pro práci s e-mailem téměř nezbytné. Jedná se o následující složky.

- Drafts (Koncepty)
- Sent (Odeslaná pošta)
- Junk (Nevyžádaná pošta)
- Trash (Smazaná pošta)
- Outbox (Odchozí pošta)

Co se týká implementace, je nutné udělat pár změn.

1. Při vytváření účtu vytvořit dodatečné složky.
2. Při přijímání zpráv správně detekovat, do které složky chce uživatel zprávy stahovat.
3. Upravit skript pro stahování zpráv tak, aby věděl, z které složky má zprávy stahovat. Tuto informaci jednoduše předáme jako argument.

4.5.2 Mazání zpráv

Dále jsem přidal možnost mazání e-mailových zpráv. Princip je zde velice jednoduchý.

1. Z uživatelského rozhraní získáme seznam zpráv, které jsou označené a uživatel je chce smazat (jde o pole typu nsIMsgDBHdr).
2. Pro každou zprávu získáme její messageId, které se rovná UniqueId zprávy uložené na serveru.
3. Seznam získaných identifikátorů spolu s dalšími údaji a skriptem, který smaže všechny zprávy podle daných identifikátorů, předáme Node.js procesu. Mimo těchto informací předáváme navíc příznak, zda se jedná o tzv. hard delete, který zprávu smaže navždy. Tento příznak předáváme jen když mažeme zprávy ze složky smazaná pošta.
4. Pokud proces skončil úspěšně, smažeme dané zprávy z lokálního úložiště, pokud ne, informujeme uživatele, že je problém s připojením k serveru.

4.5.3 Podpora odchozí pošty

Někdy se stane, že chceme poslat e-mail, ale nemáme internetové připojení. V takovém případě je škoda e-mail zahodit, místo toho je možné si ho uložit do odchozí pošty a někdy v budoucnu stačí kliknout na tlačítko „Odeslat neodeslané zprávy“ a všechny zprávy ve složce odchozí pošta se odešlou. Toto je jedna z dalších funkcionalit, kterou jsem se rozhodl implementovat.

Opět je nutné udělat pár změn v dosavadní implementaci.

1. Při odesílání zprávy je nutno rozlišit, zda chceme zprávu odeslat hned nebo ji uložit pro pozdější odeslání.

2. Pokud chce uživatel zprávu uložit pro pozdější odeslání, je nutné ji uložit do „Outboxu“ a nastavit příznak, že se zpráva nemá mazat (jelikož je uložena jen lokálně a na serveru neexistuje, tudíž by se podle dosavadní implementace měla zpráva smazat).

3. Musíme přidat podmínku, aby se zprávy s tímto příznakem z lokálního úložiště nemazaly.

Nyní máme složku „Outbox“, kde mohou být zprávy uložené na serveru i zprávy uložené pouze lokálně. Teď je potřeba pouze implementovat operaci „Odeslat neodeslané zprávy“.

1. Získáme všechny zprávy z „Outboxu“ a rozdělíme si je na dvě množiny.

   - Zprávy, které jsou uložené na serveru.
   - Zprávy, které jsou uložené pouze lokálně.

2. Ze zpráv, které jsou uložené na serveru, získáme messageId, resp. UniqueId a z lokálně uložených zpráv získáme jejich MIME obsah.

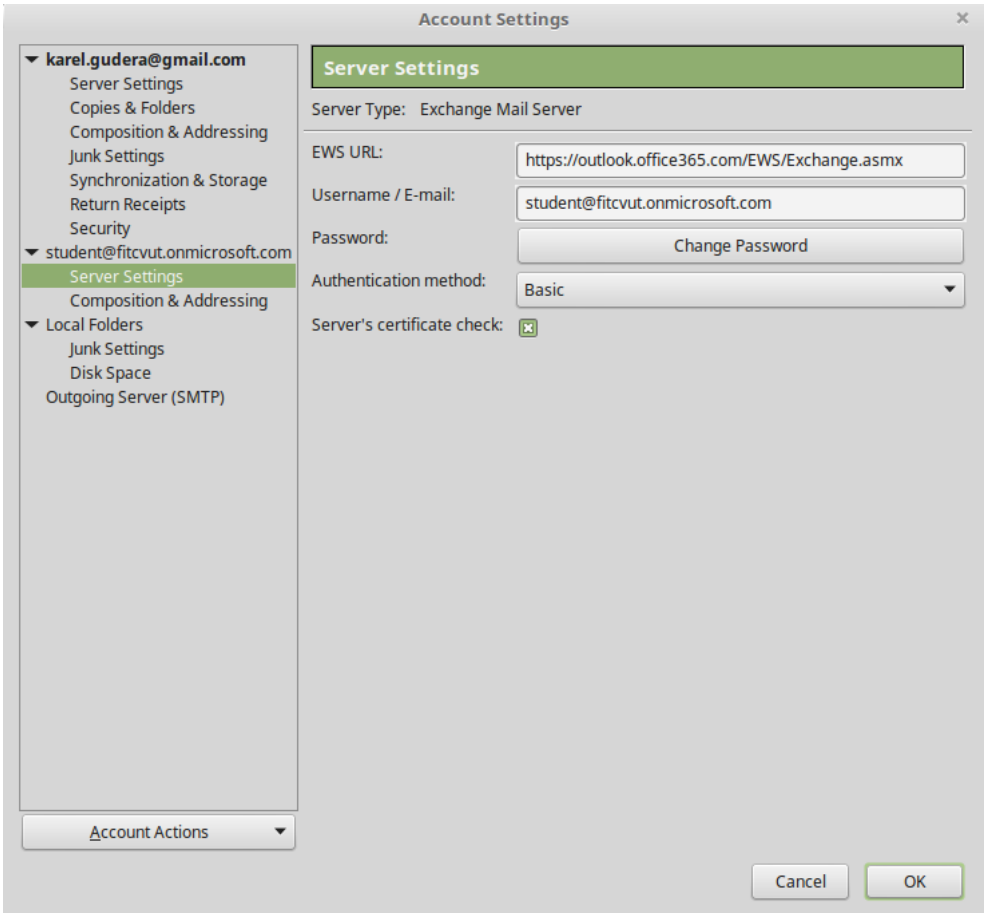
3. Zavoláme Node.js a předáme mu všechny potřebné informace (seznam identifikátorů a MIME obsah zpráv) a skript zodpovědný za odeslání.

4. Skript se pomocí identifikátorů připojí k daným zprávám a odešle je. Pro zprávy uložené lokálně vytvoří pomocí MIME obsahu novou zprávu, kterou také odešle.

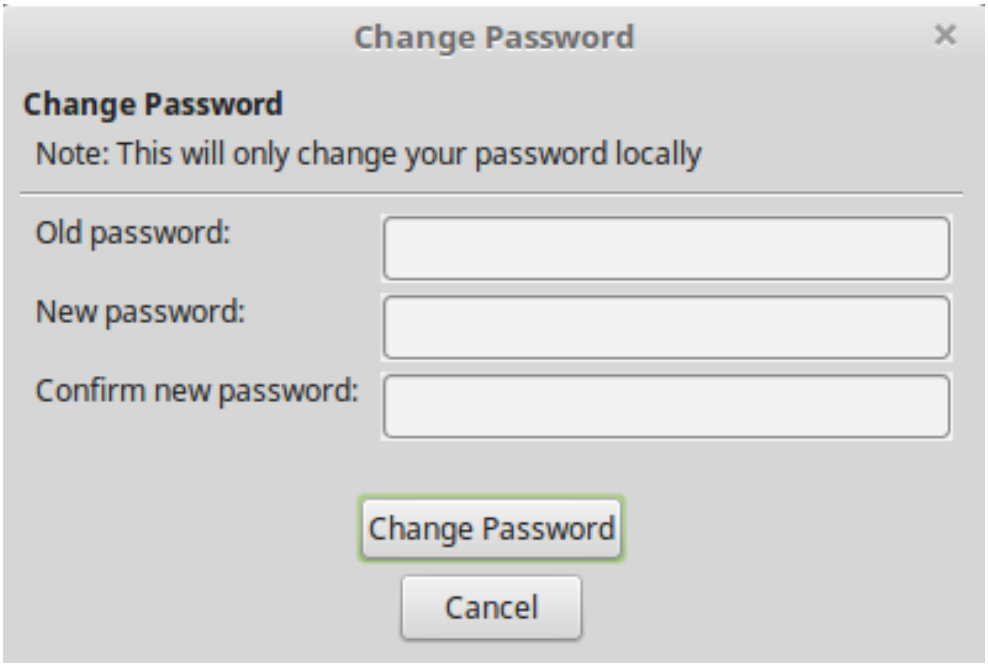
5. Pokud proces odesílání skončil úspěšně, smažeme všechny zprávy z „Outboxu“, jinak uživatele informujeme o chybě.

4.5.4 Uživatelské rozhraní správce účtů

Původně bylo uživatelské rozhraní ponecháno v originálním stavu, tzn. uživatel viděl nastavení pro POP3 účet. Jakékoliv provedené změny neměly na funkčnost modulu žádný vliv. Toto by ovšem mohlo vést ke zmatení některých uživatelů, proto jsem se rozhodl původní uživatelské rozhraní nahradit. V nově vytvořeném rozhraní je možné změnit EWS URL, uživatelské jméno, heslo či kontrolu certifikátu. Jak uživatelské rozhraní vypadá můžete vidět na obrázcích 4.2 a 4.3.



Obrázek 4.2: Správce účtů



Obrázek 4.3: Změna hesla

Kapitola 5

Testování

Vzhledem k tomu, že modul je zaintegrován přímo v Thunderbirdu a nepřináší žádné velké změny v uživatelském rozhraní, budeme se v testování soustředit hlavně na testování funkčnosti. Naším cílem je, aby modul bezproblémově fungoval na co nejvíce platformách.

Testování zahrnuje jednoduchý scénář použití a dále několik souvisejících otázek.

K testování byl použit 30 denní zkušební účet Microsoft Office 365 [25], který dovoluje použití EWS.

5.1 Scénář použití

1. Stáhněte si návod z `http://webdev.fit.cvut.cz/~guderkar/guide.pdf`
2. Podle návodu nainstalujte modul
3. Podle návodu přidejte účet Microsoft Exchange
4. Podle návodu stáhněte zprávy ze serveru
5. Podle návodu odešlete zprávu na některý svůj soukromý e-mail
6. Zkontrolujte, zda vám e-mail dorazil
7. Odešlete ze svého soukromého e-mailu zprávu na student@fitcvut.onmicrosoft.com
8. Opět stáhněte zprávy ze serveru a zkontrolujte, zda e-mail dorazil

Návod k použití můžete také nalézt v příloze B.

5.2 Vlastní testování

Zásuvný modul jsem osobně otestoval v následujících případech.

5.2.1 Testování proti Office 365 (online)

Testování proběhlo podle scénáře na následujících platformách.

- Linux Mint 18.1 Cinnamon 64-bit
- Debian 8.7.1 Cinnamon 64-bit
- Debian 8.7.1 Cinnamon 32-bit
- Gentoo Linux 4.5.2 KDE 64-bit
- Windows 7 Professional 64-bit
- Windows 7 Ultimate 32-bit
- Windows 8.1 Pro 64-bit
- Windows 10 Pro N 64-bit
- Mac OS X 10.7.5 64-bit

Výsledek

Vše proběhlo v pořádku.

5.2.2 Testování proti Exchange 2010 (online)

Bylo mi umožněno modul vyzkoušet na účtu České pojišťovny, která používá verzi Exchange 2010.

Výsledek

Byly nalezeny následující problémy.

- Exchange server podporoval pouze NTLM autentizaci.
- Autentizovat se dalo pouze přes „username“, nikoli e-mailovou adresu.

5.2.3 Testování proti Exchange 2010 (Virtualbox)

Nejvíce jsem ovšem modul otestoval na lokálním serveru Exchange 2010, který běžel ve virtuálním stroji.

Výsledek

Podle scénáře proběhlo testování v pořádku. Objevil jsem ale některé chyby, které nemusí být na první pohled viditelné.

- Při spuštěném Node.js procesu lze odhalit heslo pomocí příkazu „ps -ef | grep node“.
- Nefunkční skrytá kopie.
- Nefunkční vCard (vizitka) při kompozici nové zprávy.
- Nefunkční embedované obrázky při kompozici nové zprávy (špatný atribut „src“).
- Některé zprávy se stahují i přesto, že už existují v lokálním úložišti.

5.3 Testování v podobě online průzkumu

Mimo osobního testování byl vytvořen online průzkum, který zkoumá funkčnost podle uvedeného scénáře a je doplněn o několik otázek. Průzkumu se zúčastnilo 8 respondentů.

5.3.1 Otázky

1. Jaký operační systém jste k testování použili? Zkuste být co nejspecifičtější.
2. 32 nebo 64 bitová verze OS?
3. Proběhlo testování v pořádku? Pokud ne, popište problém, který se vyskytl.
4. Jak by jste hodnotili uživatelskou přívětivost modulu na stupnici od 1 do 10, kde 10 představuje nejlepší známku?
5. Máte nějaké připomínky nebo rady co zlepšit? Popřípadě popište co se vám líbilo, či nelíbilo.

5.3.2 Odpovědi

- Respondent 1
    1. Ubuntu 16.04 LTS
    2. 64-bit
    3. Ano, proběhlo.
    4. 9

5. Nevím, co bych mohl od takového modulu očekávat. Provedl jsem vše podle návodu a nebyl problém.

- Respondent 2
  1. Windows 10 Pro
  2. 64-bit
  3. Nepodařilo se přidat účet a hláška „Failed to add account“ je při hledání problému úplně k ničemu, zkoušel jsem jak připravený účet, tak firemní účet a ani jednou se mi nepodařilo přidat účet.
  4. 1
  5. Líbilo by se mi kdyby to fungovalo. Jinak to chce zlepšit hlášení chyb uživateli, aby si buď uživatel našel problém, nebo aby se dal reportovat.
- Respondent 3
  1. Windows 10
  2. 64-bit
  3. Testování proběhlo bez problémů, modul funguje spolehlivě.
  4. 9
  5. Oceňuji jednoduché uživatelské rozhraní, které nedává prostor ke zmatení uživatele. Také bych vypíchnul jasný a stručný instalační návod.
- Respondent 4
  1. Windows 10 Home, version 1607
  2. 64-bit
  3. Všechno proběhlo OK.
  4. 8
  5. Jelikož jsem Thunderbird nikdy nepoužil, tak se mi těžko řekne, co patří pod nový modul a co je původní Thunderbird, ale na letmý pohled mi nic nechybí a vypadá to v pohodě.
- Respondent 5
  1. Ubuntu 14.04
  2. 64-bit
  3. E-mail s obsahem „Super text, který určitě přijde.“ přišel, ale bohužel do spamu. Gmail říká „Zjistili jsme, že mnoho zpráv z domény fitcvut.onmicrosoft.com je spam.“. Typuji že při testování modulu došlo k mnoha testovacím zprávám a tato doména je již označená jako spam. Vše ostatní proběhlo v pořádku dle návodu.

4. 9
5. Nevyplněno.

- Respondent 6
  1. Windows 10 Pro
  2. 64-bit
  3. Testování proběhlo v pořádku.
  4. 10
  5. Modul mi přijde uživatelsky přívětivý a lehce použitelný.
- Respondent 7
  1. Windows 10
  2. 64-bit
  3. Ano.
  4. 10
  5. Nevyplněno.
- Respondent 8
  1. Windows 10
  2. 64-bit
  3. Ano.
  4. 7
  5. Nevím, Thunderbird nepoužívám – e-maily to posílá, takže OK.

5.3.3 Shrnutí

Množina operačních systémů, na kterých modul fungoval

- Windows 10 (Home, Pro) 64-bit
- Ubuntu (16.04, 14.04) 64-bit

Množina operačních systémů, na kterých modul nefungoval

- Windows 10 Pro 64-bit

Úspěšnost

- 7 z 8 = 87.5 %

Uživatelská přívětivost

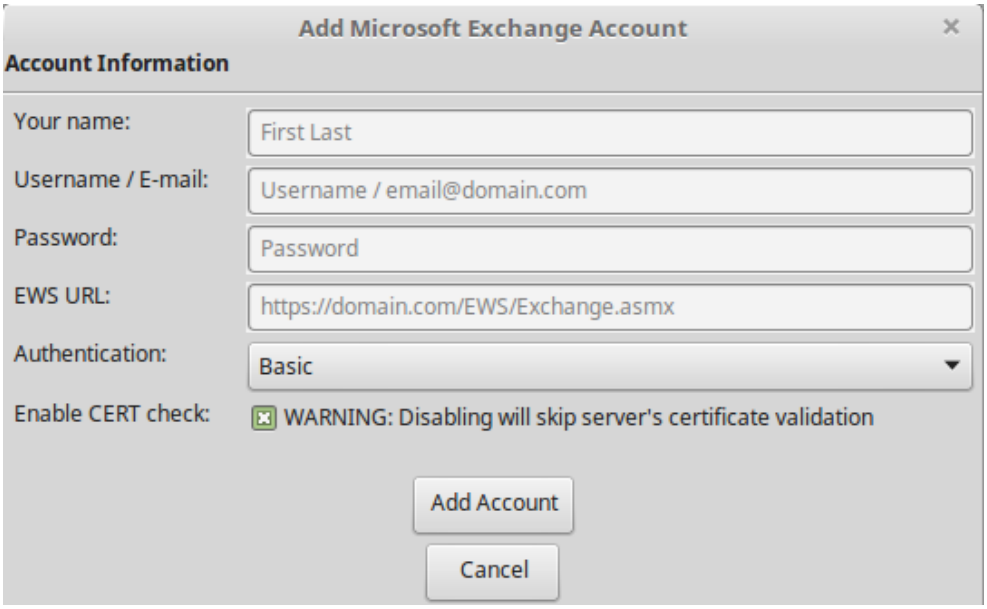
- 7.88 z 10

5.4 Závěr z testování

Testování nám odhalilo několik nedostatků, které by bylo dobré opravit. Mimo jiné jsem provedl další změny, které by měly vést k rozšíření funkčnosti a většímu komfortu při používání.

- Přidána podpora NTLM.
- Přidána možnost autentizace podle „username“.
- Citlivé informace jsou nyní předávány bezpečně na standardní vstup Node.js procesu namísto původního řešení, kde se předávaly jako argumenty.
- Opravena funkčnost skryté kopie.
- Opravena funkčnost vCard (vizitka).
- Opravena funkčnost embedovaných obrázků.
- Opravena chyba, kdy se znovu stahovaly už stažené zprávy.
- Zlepšení hlášení chyb uživateli a přidán návod na reportování problému.
- Odstraněno pole doména, protože ke komunikaci se serverem stačí EWS URL.
- Opravena nefunkční tlačítka pro načítání zpráv.

Nové uživatelské rozhraní pro přidání účtu, které už neobsahuje pole doména a obsahuje možnost výběru způsobu autentizace mezi HTTP Basic a NTLM si můžete prohlédnout na obrázku 5.1.



Obrázek 5.1: Nové okno pro přidání účtu

Závěr

Cílem této práce bylo analyzovat současný stav řešení problematiky komunikace mezi e-mailovým klientem a serverem Microsoft Exchange pomocí Exchange Web Services. Bylo nalezeno několik řešení, ale žádné ve formě bezplatného zásuvného modulu pro Mozilla Thunderbird, který by fungoval. Z tohoto důvodu bylo usouzeno, že se má smysl touto problematikou dále zabývat.

Dále bylo cílem daný zásuvný modul navrhnout, implementovat a vhodným způsobem otestovat. Modul měl umět přijímat a odesílat e-mailové zprávy a být dostupný pro operační systémy Linux a Microsoft Windows jako svobodný software s otevřeným kódem. Uživatel by neměl potřebovat spouštět nebo instalovat další software pro správné fungování modulu.

Při návrhu se vyskytlo více možností. Jelikož jsme v tomto směru omezeni dostupnou technologií pro komunikaci se serverem Exchange, vybral jsem z mého pohledu tu nejjednodušší možnost, která zároveň poskytuje dobrý výhled do budoucna, co se týče dalšího rozšíření modulu. Toto řešení taktéž dodržuje požadavek, aby uživatel nemusel spouštět či instalovat další software.

V implementaci jsem se opět soustředil na co možná nejjednodušší způsob provedení, který zároveň bude pracovat korektně. Mimo přijímání a odesílání zpráv bylo navíc implementováno i mazání zpráv, podpora základních složek, funkčnost „Outboxu“ a uživatelské rozhraní pro správce účtů.

Testování proběhlo jak lokálně tak online. Mimo toho byl vytvořen průzkum, který zjišťoval funkčnost a uživatelskou přívětivost modulu. Až na výjimečný případ modul fungoval bez problému a uživatelé si jej chválili, ba dokonce nemohli rozeznat, co je původní Thunderbird a co modul, což je z hlediska integrace úspěch.

Mimo operačních systémů Linux a Microsoft Windows, je modul dostupný taktéž pro Mac OS X. To, že je modul dostupný i pro tuto platformu je příznivý důsledek zvoleného řešení při návrhu modulu.

Modul a celý jeho zdrojový kód je dostupný na stránkách Github (`https://github.com/guderkar/MExInt`). Verzi pro daný operační systém je možné

nalézt ve složce xpi.

Současná verze modulu určitě není dokonalá a stále bude co zlepšovat. Bylo by pěkné v budoucnu rozšířit funkcionalitu o kopírování/přesouvání e-mailových zpráv, synchronizaci ostatních složek existujících na serveru, jejich vytváření a mazání. Nicméně si myslím, že pokud někdo hledá bezplatnou alternativu k již existujícímu řešení, tak **MExInt** je správná volba.

Literatura

[1] Mark R. Crispin: IMAP4 RFC 3501. [cit. 8.2.2017]. Dostupné z: https://tools.ietf.org/html/rfc3501

[2] Marshall Rose: POP3 RFC 1081. [cit. 8.2.2017]. Dostupné z: https://tools.ietf.org/html/rfc1081

[3] Jonathan B. Postel: SMTP RFC 821. [cit. 8.2.2017]. Dostupné z: https://tools.ietf.org/html/rfc821

[4] Microsoft: Exchange Web Services. [cit. 8.2.2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/office/dd877045(v=exchg.140).aspx

[5] W3C: Simple Object Access Protocol. [cit. 8.2.2017]. Dostupné z: https://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[6] Microsoft: Messaging Application Programming Interface. [cit. 8.2.2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/aa142548(v=exchg.65).aspx

[7] R. Kent James: Exquilla. [cit. 10.2.2017]. Dostupné z: https://exquilla.zendesk.com/hc/en-us

[8] Mozilla: XPCOM. [cit. 10.2.2017]. Dostupné z: https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XPCOM

[9] Evolution: Evolution. [cit. 10.2.2017]. Dostupné z: https://wiki.gnome.org/Apps/Evolution

[10] DavMail: DavMail Gateway. [cit. 10.2.2017]. Dostupné z: http://davmail.sourceforge.net

[11] DavMail: DavMail Architecture. [cit. 10.2.2017]. Dostupné z: http://davmail.sourceforge.net/images/davmailArchitecture.png

[12] Ericsson: Exchange EWS Provider. [cit. 10.2.2017]. Dostupné z: https://github.com/Ericsson/exchangecalendar

[13] Jingnan Si: exchange-ews-thunderbird. [cit. 21.2.2017]. Dostupné z: https://github.com/stonewell/exchange-ews-thunderbird

[14] otris software AG: ews-cpp. [cit. 15.2.2017]. Dostupné z: https://github.com/otris/ews-cpp

[15] Gautam Singh: ews-javascript-api. [cit. 15.2.2017]. Dostupné z: https://github.com/gautamsi/ews-javascript-api

[16] Node.js Foundation: Node.js Documentation. [cit. 15.2.2017]. Dostupné z: https://nodejs.org/en/docs

[17] Mozilla: XUL Reference. [cit. 19.2.2017]. Dostupné z: https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL/XUL_Reference

[18] Mozilla: js-ctypes. [cit. 15.2.2017]. Dostupné z: https://developer.mozilla.org/en-US/docs/Mozilla/js-ctypes

[19] Stack Overflow: Using core Node.js modules in Firefox addons. [cit. 21.2.2017]. Dostupné z: http://stackoverflow.com/questions/36114813/using-core-nodejs-modules-in-firefox-addons

[20] Microsoft: Microsoft Windows DLL List. [cit. 9.5.2017]. Dostupné z: http://www.win7dll.info

[21] Gautam Singh: NTLM XHRApi. [cit. 2.3.2017]. Dostupné z: https://gist.github.com/gautamsi/28211eda711e3e7dc04c

[22] Mozilla: Add-on SDK. [cit. 12.3.2017]. Dostupné z: https://developer.mozilla.org/cs/Add-ons/SDK

[23] Mozilla: Mozilla Thunderbird Documentation. [cit. 19.3.2017]. Dostupné z: http://doxygen.db48x.net/mozilla-full/html/classes.html

[24] Mozilla: Mozilla Thunderbird Github. [cit. 19.3.2017]. Dostupné z: https://github.com/mozilla/releases-comm-central

[25] Microsoft: Microsoft Office 365. [cit. 25.3.2017]. Dostupné z: https://products.office.com/en-US

[26] Mozilla: Mozilla Thunderbird DXR. [cit. 19.3.2017]. Dostupné z: https://dxr.mozilla.org/mozilla-central/source

[27] Mozilla: Mozilla Thunderbird MDN. [cit. 19.3.2017]. Dostupné z: https://developer.mozilla.org/en-US/docs/Mozilla/Thunderbird

[28] Mozilla: Mozilla Thunderbird HowTos. [cit. 19.3.2017]. Dostupné z: http://mdn.beonex.com/en/Extensions/Thunderbird/HowTos.html

[29] Mozilla: Mozilla Thunderbird Google Groups. [cit. 19.3.2017]. Dostupné z: https://groups.google.com/forum/#!forum/mozilla.dev.apps.thunderbird

[30] Microsoft: Microsoft Exchange MSDN. [cit. 25.3.2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/office/mt674770.aspx

Příloha A

Seznam použitých zkratek

EWS Exchange Web services

MAPI Messaging Application Programming Interface

XML Extensible Markup Language

XUL XML User Interface Language

IMAP Internet Message Access Protocol

POP Post Office Protocol

SMTP Simple Mail Transfer Protocol

SOAP Simple Object Access Protocol

HTTP Hypertext Transfer Protocol

HTTPS HTTP Secure

SSL Secure Socket Layer

TLS Transport Layer Security

NTLM NT LAN Manager

XPCOM Cross Platform Component Object Model

URL Uniform Resource Locator

MIME Multipurpose Internet Mail Extensions

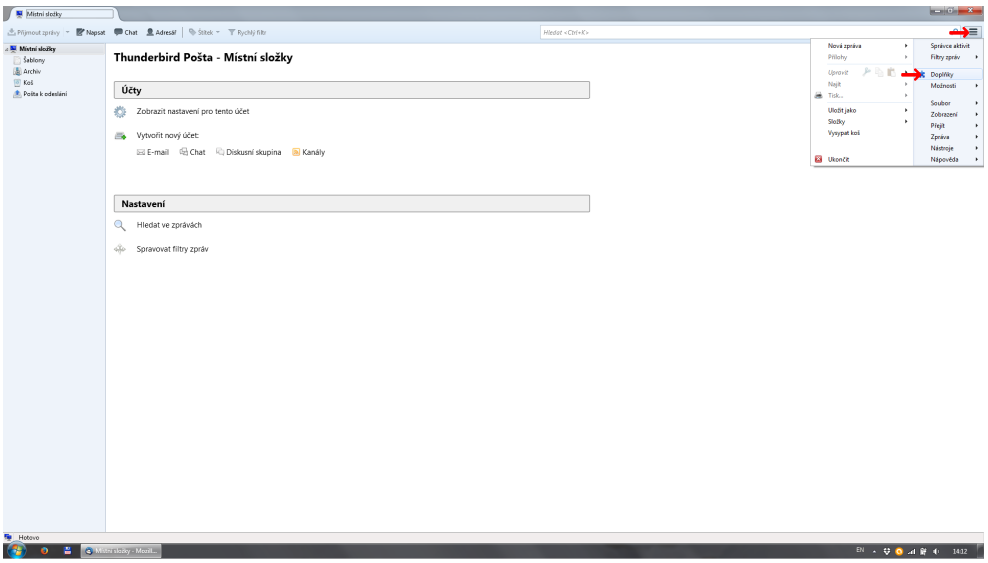
OS Operační systém

Příloha B

Uživatelská příručka

B.1 Instalace modulu

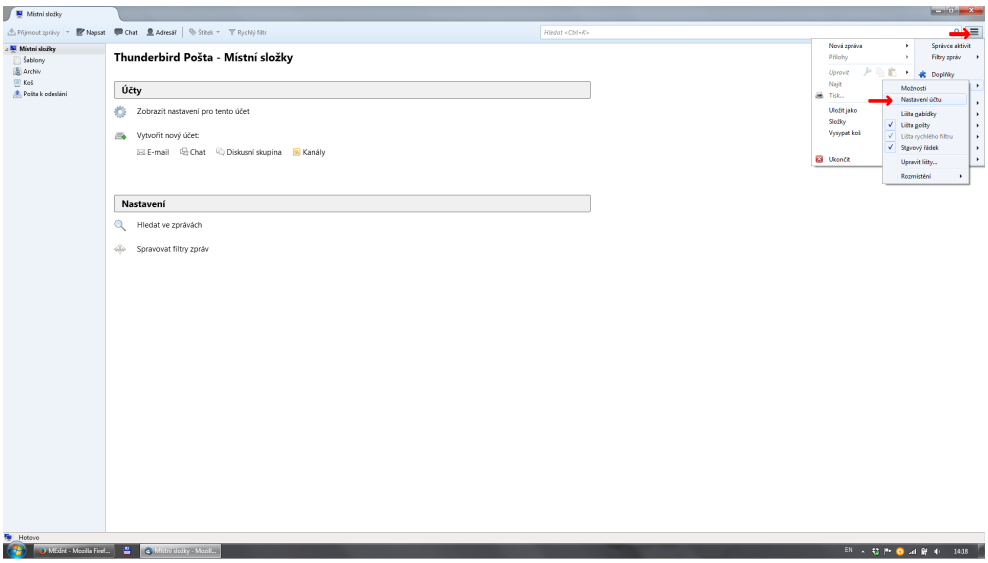
1. Stáhněte si modul z `http://www.webdev.fit.cvut.cz/~guderkar/`.
2. Spusťte Thunderbird.
3. Dále postupujte podle obrázků.



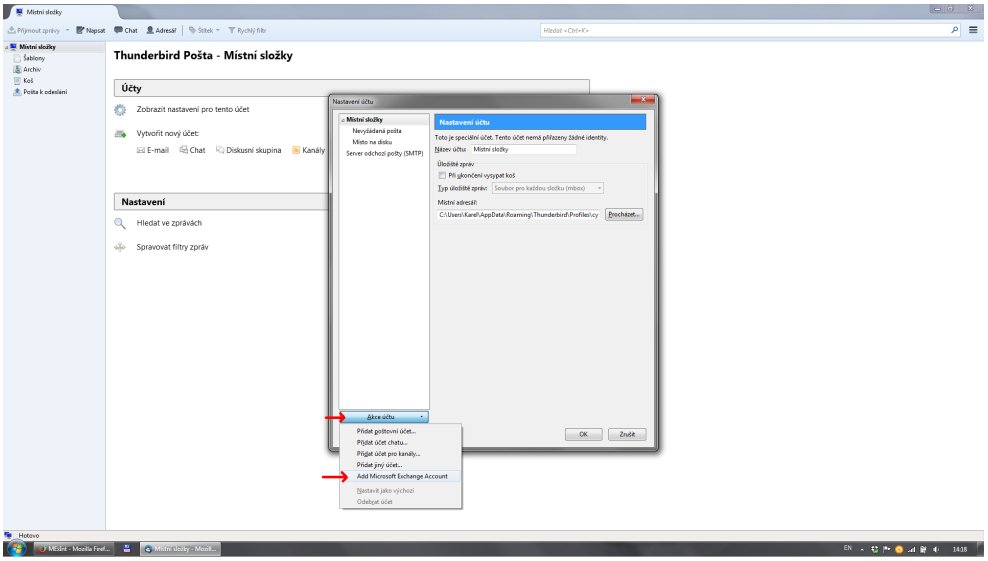
Obrázek B.1: Instalace modulu 1

Obrázek B.2: Instalace modulu 2

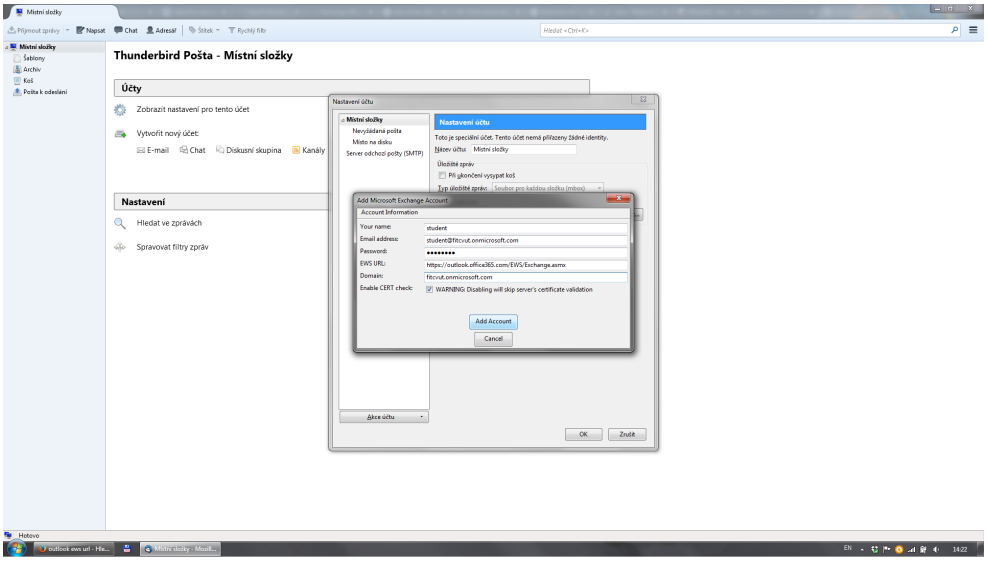
B.2 Přidání účtu



Obrázek B.3: Přidání účtu 1



Obrázek B.4: Přidání účtu 2

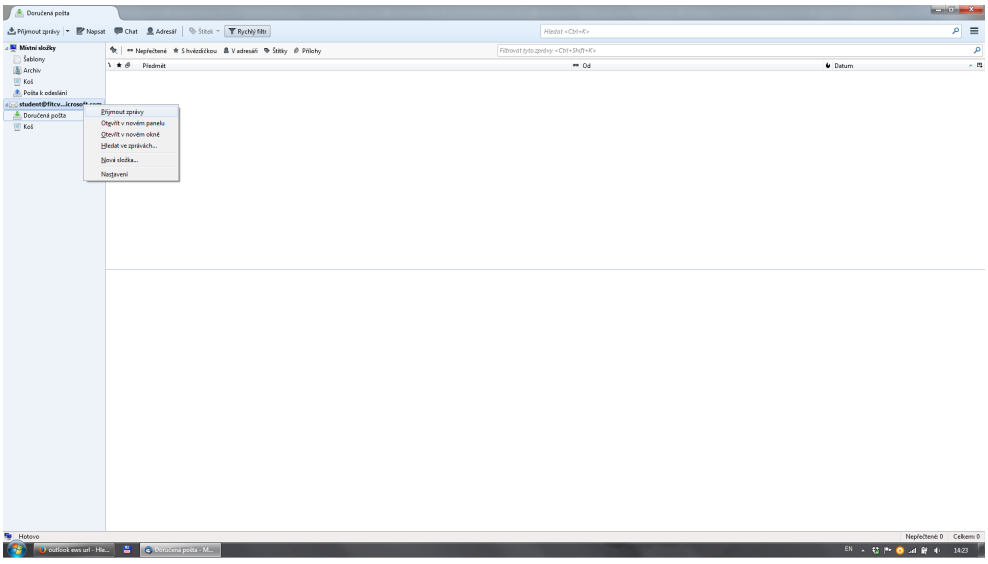


Obrázek B.5: Přidání účtu 3

Informace k testovacímu účtu ke zkopírování

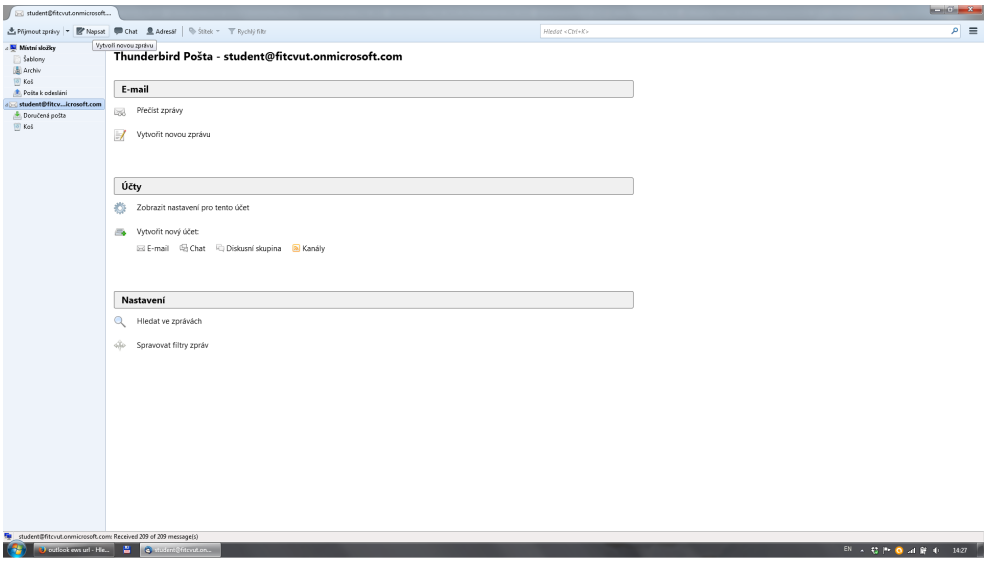
1. student
2. student@fitcvut.onmicrosoft.com
3. Qoca0756
4. https://outlook.office365.com/EWS/Exchange.asmx
5. fitcvut.onmicrosoft.com

B.3 Přijímání zpráv

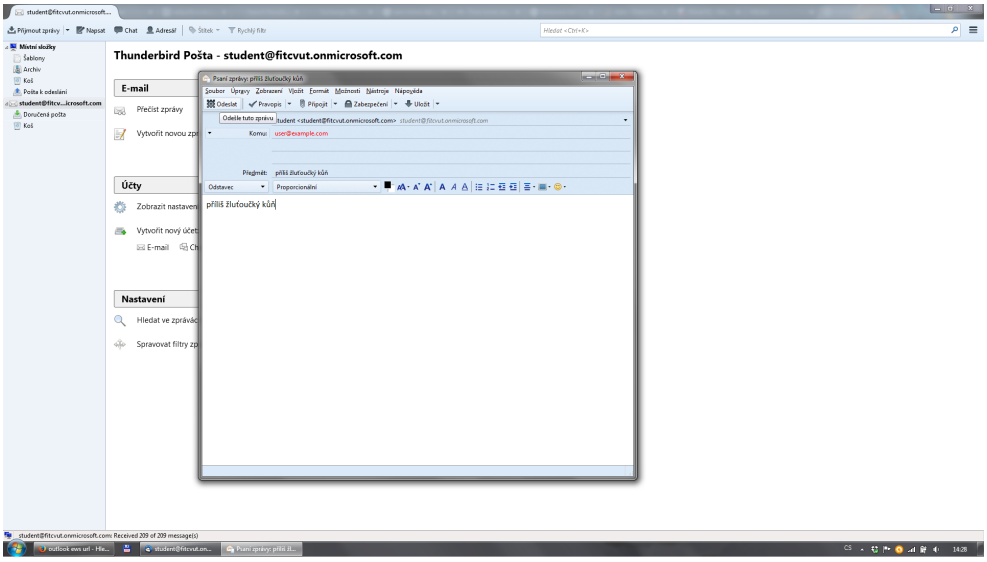


Obrázek B.6: Přijímání zpráv 1

B.4 Odesílání zpráv



Obrázek B.7: Odesílání zpráv 1



Obrázek B.8: Odesílání zpráv 2

Příloha C

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `xpi` ..... adresář s implementací modulu pro různé platformy
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS