VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

VIDEOKONFERENCE V HTML 5

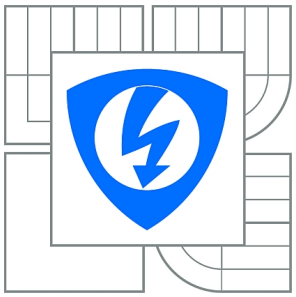
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE DAVID WIESNER
AUTHOR

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

VIDEOKONFERENCE V HTML 5

VIDEO CONFERENCE IN HTML 5

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

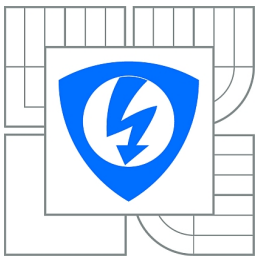
AUTOR PRÁCE — DAVID WIESNER

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR ČÍKA, Ph.D.

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	David Wiesner	***ID:***	121037
Ročník:	3	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Videokonference v HTML 5

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte možnosti uskutečnit hlasový i video hovor z webového prohlížeče bez nutnosti instalace jakéhokoli dalšího softwaru na koncový počítač. Pro signalizaci použijte protokol SIP. Vyberte vhodnou ústřednu a proxy server podporující technologii WebRTC. Softwarový klient bude kromě standardních funkcí umožňovat textový chat a propojení s adresářem dané struktury. Funkčnost aplikace ověřte testováním.

DOPORUČENÁ LITERATURA:

[1] WEBRTC. WebRTC [online]. 2011 [cit. 2013-04-16]. Dostupné z: http://www.webrtc.org/
[2] HTML5 Multimedia: Develop and Design. London: Peachpit Press, 2011. 1. ISBN 978-0321793935.

Termín zadání:	11.2.2013	***Termín odevzdání:***	5.6.2013

Vedoucí práce: Ing. Petr Číka, Ph.D.
Konzultanti bakalářské práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato bakalářská práce se věnuje návrhu a realizaci webové aplikace typu klient, která využívá protokol SIP pro signalizaci při multimediální komunikaci. Ta je přenášena přes protokol TCP, přesněji podprotokol WebSocket. V první části jsou popsány základní vlastnosti protokolu SIP, WebSocket a taky základní informace o projektu webRTC. Hlavní částí práce je popis návrhu a implementace webové aplikace pro audio-video komunikaci a pro textovou komunikaci v rámci posílání zpráv. V této části se nachází obeznámení s postupy, které byly použity během vývoje. Je popsán také výběr knihoven k tomu použitých. Na konci byla aplikace podrobena testování.

KLÍČOVÁ SLOVA

SIP, WebRTC, videokonference, WebSocket, officeSIP, WebStorm, JsSIP, Google Chrome

ABSTRACT

This bachelor thesis follow proposal and realization of client-type web application, which uses SIP protocol for multimedia communication signalization. The communication is relayed via TCP protocol, more precisely via WebSocket. There are described basic properties of SIP and WebSocket protocols together with information about webRTC in first part of this thesis. The main part of thesis describes proposal and implementation of web application for AV and text communication. The briefing about progress reached during development is mentioned here as well as used libraries. Application was tested after finishing.

KEYWORDS

SIP, WebRTC, videoconference, WebSocket, officeSIP, WebStorm, JsSIP, Google Chrome

WIESNER, David *Videokonference v HTML 5*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 42 s. Vedoucí práce byl Ing. Petr Číka, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Videokonference v HTML 5“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Petru Číkovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Videokonference je moderní způsob komunikace. Přenáší se při ní obraz i zvuk, účastníci si během přenosu mohou vyměňovat textové zprávy. Využití videokonference se našlo v mnoha oborech a činnostech lidského bádání. Jako například při poradách managementu různých organizací, také v medicíně, pro přenos průběhu operace. Ale také při běžné volnočasové přátelské komunikaci. Hlavní výhody jsou úspora finančních i časových nákladů, větší operativnost a také lepší kontakt mezi komunikujícími účastníky hovoru, kteří nepotřebují ke komunikaci klasický telefon. Úvod do problematiky videokonferencí přes rozhraní webového prohlížeče a také předvedení záměrů projektu webRTC popisují následující kapitoly.

Pro připojení účastníků do videokonference je nejprve potřeba vytvořit spojení mezi koncovým zařízením a serverem, toto spojení obsluhovat a také případně ukončit. K tomu se využívá tzv. signalizačních protokolů, jako je protokol SIP. Podrobněji o něm dále v textu. Pro přenos skutečných dat se používá protokol TCP, respektive jeho podprotokol WebSocket. Více o něm v následujících kapitolách. Celou problematikou přenosu obrazu a zvuku přes webové prohlížeče, bez potřeby instalace jakýchkoliv pluginů a doplňků, se zabývá projekt webRTC. Vývoj probíhal v prostředí WebStorm, což je velice moderní IDE (Integrated development environment) pro tvorbu v HTML, CSS a javaskriptu. Prostředí nabízí mnoho funkcí, které velkou měrou usnadňují vývojáři práci. Cílem této práce je návrh a následná implementace aplikace, která bude umožňovat volání přes Internet s využitím pouze webového prohlížeče. V práci budou popsány knihovny, které byly využity při vývoji aplikace. Součástí je popis architektury a návrh jednotlivých funkcí a také popis testování aplikace.

1 TEORETICKÉ MINIMUM

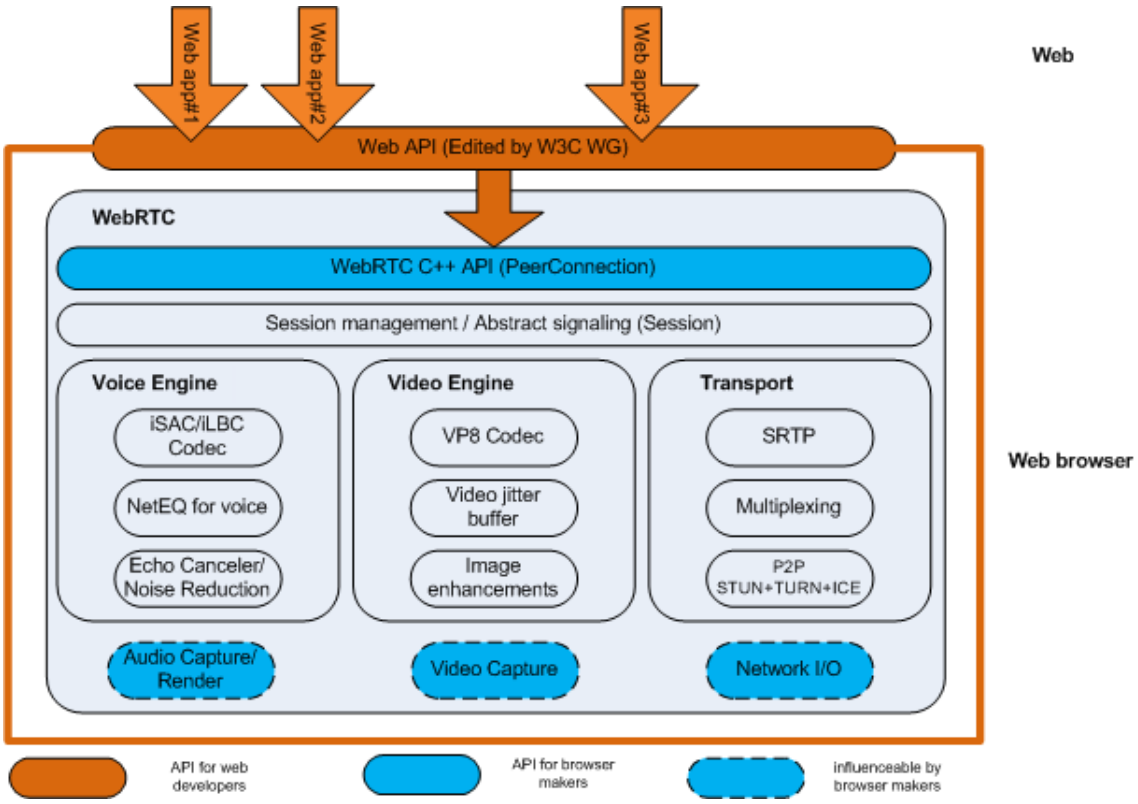
1.1 WebRTC

Jak definuje [1], WebRTC je otevřený a volně šířitelný projekt, který umožňuje přes webový prohlížeč komunikaci v reálném čase, ve zkratce RTC (Real-Time Communications). Vše je realizováno pomocí relativně jednoduchého API (Application Programming Interface) rozhraní jazyka javaskript. Všechny WebRTC komponenty byly optimalizovány tak, aby co nejlépe sloužily tomuto účelu.

Hlavním úkolem tohoto projektu je dosáhnout bohatých a vysoce kvalitních RTC aplikací, které budou vytvořeny díky jednoduchým javaskriptovým API podle specifikace HTML 5 [4].

Tento projekt je podporován prohlížeči Google Chrome, Mozilla a Opera.

WebRTC nabízí vývojářům webových aplikací schopnost psát multimediální aplikace v reálném čase (video, chat) na webu bez nutnosti instalace jakýchkoliv pluginů. Jeho smyslem je pomoci vybudovat silnou RTC platformu, která bude pracovat ve většině webových prohlížečů. Celková architektura je vidět na obrázku 1.1.



Obr. 1.1: schéma postupu vývoje

.

Jsou zde dva odlišné způsoby použití:

- vývojáře prohlížečů bude zájímat WebRTC C++ API
- vývojáře webových aplikací bude zájímat Web API

1.1.1 Web App

Vývojáři webových aplikací(třetí strana vývojářů) s možnostmi AV hovorů a chatu, kteří jsou vyzbrojeni, díky webRTC, webovým rozhraním API pro tuto komunikaci v reálném čase.

1.1.2 Web API

API, které má být použito vývojáři třetích stran pro vývoj webových multimediálních aplikací.

1.1.3 WebRTC C++ API

Vrstva API, která umožňuje tvůrcům prohlížečů snadno implementovat návrh Web API.

1.1.4 Transport/Session

Komponenty přenosu jsou sestaveny z komponentů `libjingle`. Libjingle je sbírka kódů v C++ a také ukázkových aplikací, které umožňují vytvoření peer-to-peer aplikací. Kód slouží k vytvoření síťového spojení(přes NAT, firewally, relay servery a proxy servery).

RTP Stack

Síťový zásobník pro Real Time Protocol.

STUN/ICE

Komponenta umožňující používat STUN a ICE mechanismy k navazování spojení přes různé typy sítí. STUN je sada pomocných internetových standardů, které slouží k umožnění komunikace skrz NAT, podobně jako ICE.

Session Management

Abstraktní vrstva relace, která umožňuje nastavení hovoru a spravování vrstvy. Toto nechává rozhodnutí ohledně implementace protokolu na vývojářích aplikací.

1.1.5 VoiceEngine

VoiceEngine je framework pro přenos audia ze zvukové karty do sítě. Framework je softwarová struktura, která slouží jako podpora při programování a vývoji aplkací(knihovny API, návrhové vzory, doporučené postupy).

iSAC / iLBC / Opus

- **iSAC:** širokopásmový a super širokopásmový kodek pro VoIP a streamování audia. iSAC používá 16 kHz nebo 32 kHz vzorkovací frekvenci s adaptivní a variabilní přenosovou rychlostí 12 až 52 kbps.
- **iLBC:** úzkopásmový hlasový kodek pro VoIP a streamování audia. Používá 8 kHz vzorkovací frekvenci s datovým tokem 15,2 kbps pro 20ms rámce a 13,33 kbps pro 30ms rámce. Definován IETF RFC 3951 a 3952.
- **Opus:** Podporuje konstantní i variabilní datový tok pro kódování od 6 kbit/s do 510 kbit/s, při velikosti rámce od 2,5 ms do 60 ms, a různé vzorkovací rozsahy od 8 kHz (s šířkou pásma 4 kHz) až do 48 kHz (s 20 kHz šířkou pásma, kde je obsažen rozsah slyšitelnosti lidského sluchového systému). Definovány IETF RFC 6176.

1.1.6 NetEQ for Voice

Dynamický jitter buffer a algoritmus odstraňování chyb používaný pro skrývání negativních účinků síťového rušení (jitteru) a ztrátovosti paketů. Udržuje zpoždení tak nízké, jak je to možné při zachování nejvyšší kvality zvuku.

Acoustic Echo Canceler (AEC)

Acoustic Echo Canceler je komponent zpracovávající signál, který odstraní v reálném čase akustické echo vyplývající ze zvuku, který přichází do aktivního mikrofonu.

Noise Reduction (NR)

NR je software založený na zpracování signálu, který odstraňuje některé druhy šumu v pozadí obvykle spojené s běžnou komunikací(fanoušci v pozadí atd.)

1.1.7 VideoEngine

VideoEngine je framework pro přenos videa z kamery do sítě a ze sítě na obrazovku.

VP8

Video kodek z projektu WebM. Dobře se hodí pro RTC, protože je určen pro nízkou latenci.

Video Jitter Buffer

Dynamic Jitter Buffer, pomáhá zakrýt účinky chvění a ztráta paketů na celkovou kvalitu obrazu.

Image enhancements

Odstraňuje obrazový šum ze záběrů kamery.

1.2 Protokol SIP

Signalizační protokol zajišťující inicializaci, změnu a ukončení interaktivní multimediální relace se označuje SIP (Session Initiation Protocol). Fakt, že se jedná o signalizační protokol, znamená, že popisuje pouze relaci samotnou ovšem ne data, která jsou během ní vyměňována. V rámci této multimediální relace, kterou SIP protokol zajišťuje, se může jednat buď o audio, nebo audio-video hovor a to jak mezi dvěma, tak i více účastníky. Další využití protokolu SIP je posílání krátkých zpráv IM (Instant Messaging).

Protokol SIP je protokolem textovým a může být přirovnán k protokolu HTTP (Hypertext Transfer Protocol) a SMTP (Simple Mail Transfer Protocol). Znamená to, že funguje na základě posílání žádostí a k nim relevantních odpovědí. Definice protokolu SIP lze nalézt v doporučení RFC (Request for Comments) 3261 [5]. Protokol je dál vyvíjen a přidávají se k němu nové funkce a možnosti.

Pro vytvoření relace je nezbytné, aby existovalo síťové spojení mezi klienty. Navržení protokolu SIP vychází z faktu, že má být nasazen v síti Internet, která má vrstvový model TCP/IP. V sítovém modelu je protokol SIP umístěn na aplikační vrstvě.

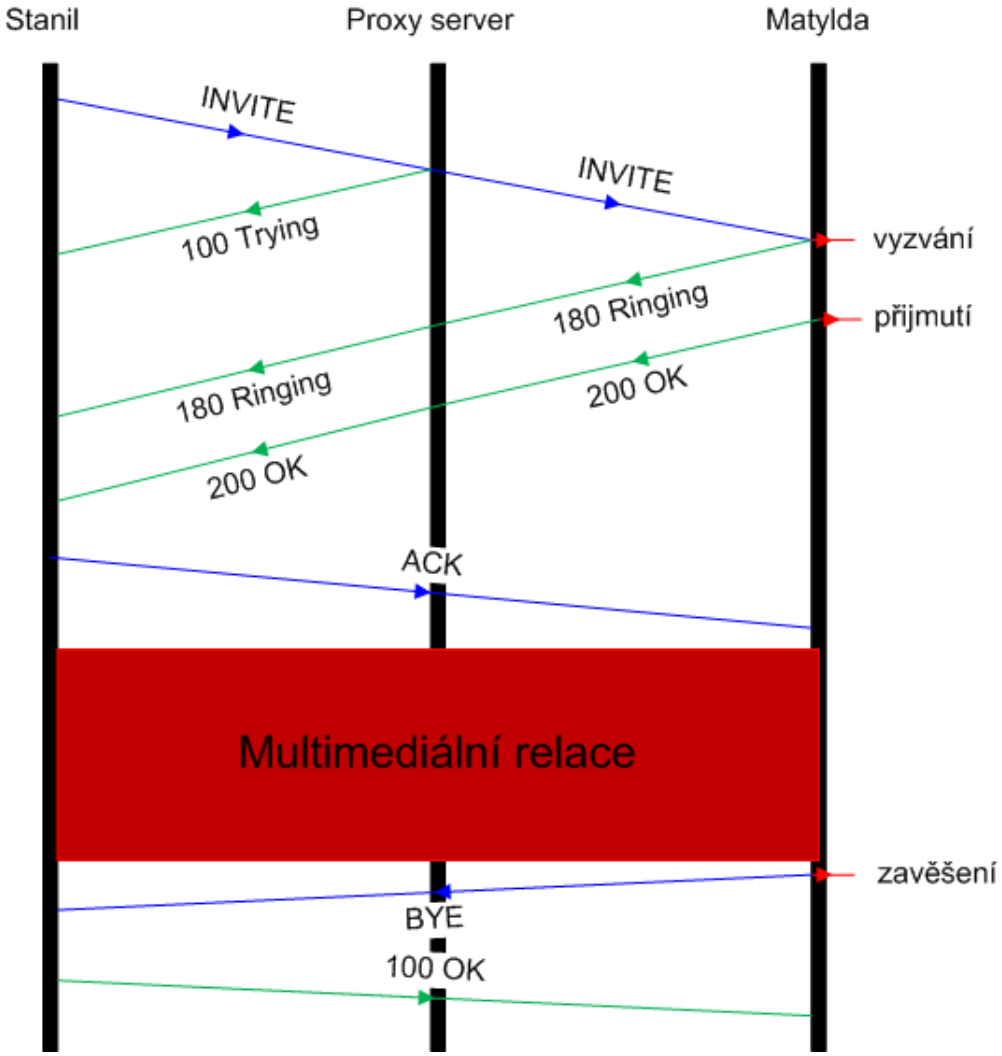
Protokol TPC (Transmission Control Protocol) a protokol UDP (User Datagram Protocol) jsou dvě možnosti, které je možné využít pro přenos SIP zpráv. Jedinou povinnou součástí architektury SIP je klient, zatímco ostatní části (Proxy servery, Redirect server apod.) již povinné nejsou, z čehož vyplývá, že uskutečnění relace je možné i bez nich.

Zprávy jsou zabalené do paketů (žádné kódování). V případě, že se použije protokolový analyzátor(WireShark), který slouží k zachycení provozu, zachycené pakety budou přímo čitelné (nezašifrované).

Základní vlastnosti a možnosti protokolu SIP jsou:

- **Vyhledání účastníka** – při inicializaci spojení vyhledá volaného. Podporována je taktéž mobilita uživatele.
- **Určení schopností uživatele** – umožňuje specifikovat vlastnosti jednotlivých zařízení (použitelné audio/video kodeky atd.).
- **Dostupnost uživatele** – přenos informací o dostupnosti uživatele.
- **Nastavení relace** – možnost nastavit a stanovit prostředky při zahájení relace.
- **Změna během relace** – během relace je možné relaci modifikovat, přesměrovat nebo ukončit.

Adresa SIP identifikuje uživatele v síti. Je definována pomocí URI – obdobný zápis jako u e-mailové adresy. Tvar SIP URI je vždy `sip:uzivatel@domena(host):port`. Pole uživatel může být například jméno nebo telefonní číslo, specifikace portu je volitelná, implicitně je číslo portu nastaveno na 5060. Příklad SIP URI je následující: `sip:emil.nejezdil@vutbr.cz`



Obr. 1.2: sestavení a ukončení relace

.

1.2.1 Síťové komponenty pro SIP

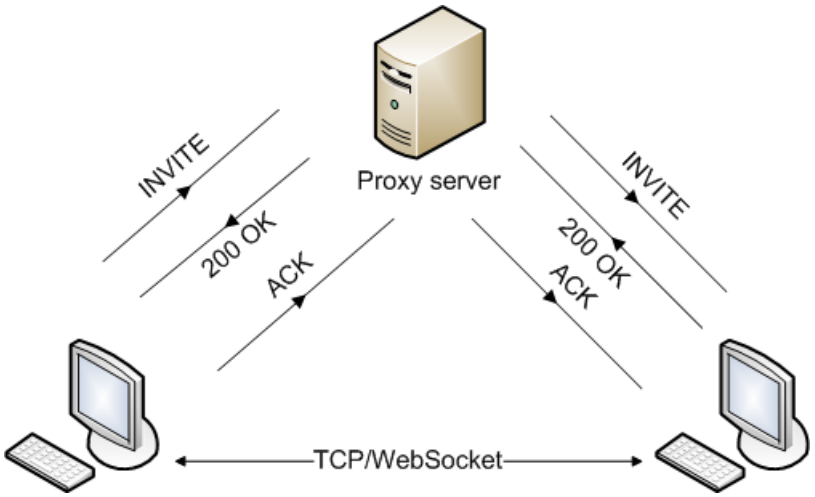
User Agent

Má jak funkci klient, tak funkci server:

- UAC (User Agent Client) – generuje a posílá žádosti serveru, přijímá a zpracovává odpovědi od serveru.
- UAS (User Agent Server) – přijímá a zpracovává žádosti od klienta, generuje odpovědi pro klienta.

Proxy server

Proxy server je mezilehlá entita, které zprostředkovává přeposílání žádostí na cílový UAS nebo odpovědi na UAC. V SIP sítích je tedy základní funkcí proxy serveru směrování, dalšími funkcemi jsou autentizace a autorizace uživatelů. Vidno na obrázku 1.3.



Obr. 1.3: schéma komunikace s proxy serverem

.

Redirect server

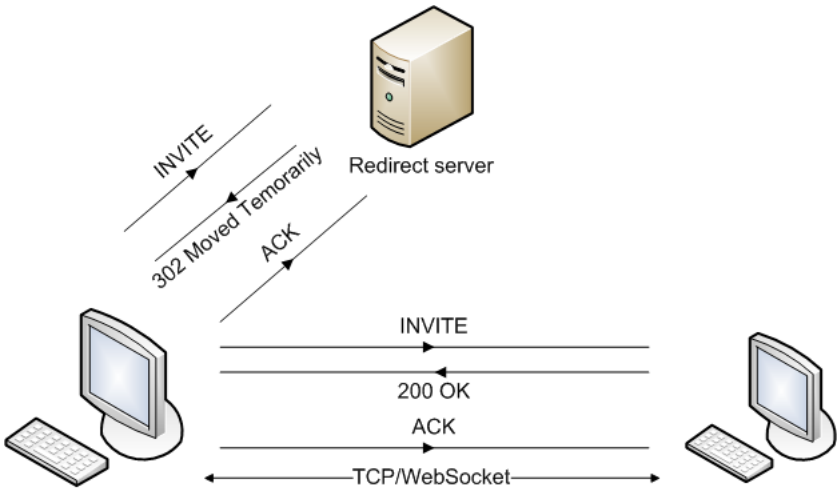
Pro generaci odpovědi třídy 3xx, které obsahují adresu volaného, slouží speciální typ UAS Redirect server. Hlavním úkolem Redirect serveru je poslat zpět UAC adresu, na které se volaný nachází. Jak lze spatřit na obrázku 1.4.

Registrar server

Funkcí Registrar serveru je přijímání žádostí o registraci SIP REGISTER od UAC a podle nich aktualizace lokalizační databáze koncových UA, které se nachází ve stejné doméně. V databázi se uchovává i informace o IP adrese. Vidno na obrázku 1.5.

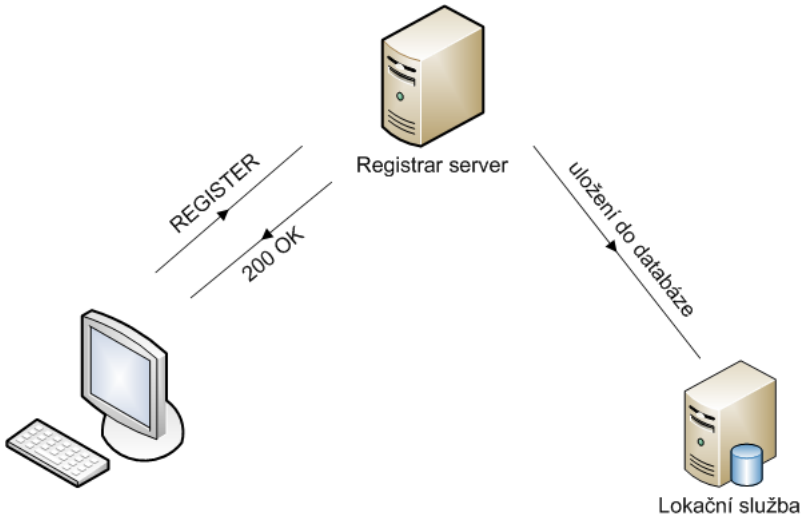
1.2.2 Zprávy SIP

K signalizaci v SIP dochází díky výměně žádostí a odpovědí. Skladba žádosti je následující - název metody s požadovaným URI a verzí SIP, několika poli hlavičky a volitelného těla zprávy. Odpovědi se skládají ze stavového řádku, zde je verze



Obr. 1.4: schéma komunikace redirect serveru

.



Obr. 1.5: schéma komunikace registrar serveru

.

SIP spolu s kódem odpovědi a textovým popisem, a dále znovu z pole hlavičky a volitelného těla.

Žádosti SIP

Nejčastěji používané jsou tyto:

- **INVITE** – vytvoření spojení,
- **BYE** – ukončení navázaného spojení,

- **OPTIONS** – vyžádání schopností koncového bodu,
- **ACK** – potvrzení, že UA dostal finální odpověď na INVITE,
- **REGISTER** – registrace umístění UA (nejčastěji přiřazení IP adresy k URI),
- **CANCEL** – ukončuje spojení ještě před vyřízením žádosti.

Příklad žádosti SIP INVITE:

```
INVITE sip:david@goolst.cz SIP/2.0
Via: SIP/2.0/UDP redmond.microsoft.com:5060;branch=z9hG4bKf0591c7ff61
Max-Forwards: 70
To: David <sip:david@goolst.cz>
From: Bill <sip:bill@microsoft.com>;tag=ccb158e3
Call-ID: 155ffd70f62abce5ec286877d29e1f0e@redmond.microsoft.com
CSeq: 1 INVITE
Contact: Bill <sip:bill@redmond.microsoft.com:5060;transport=udp>
Content-Type: application/sdp
Content-Length: 161
15
v=0
o=bill 0 0 IN IP4 redmond.microsoft.com
s=subject
c=IN IP4 100.101.102.103
t=0 0
m=audio 18404 RTP/AVP 3 0 8 101
a=rtpmap:0 PCMU/8000
```

Odpovědi SIP

Odpovědi jsou číslovány čísly 100 až 699 a dají se rozdělit do skupin, přičemž každá skupina vyjadřuje jeden typ odpovědi. Další dvě čísla označují konkrétní odpověď (např. 200 OK). Skupina 1xx jsou dočasné odpovědí, ostatní skupiny jsou odpovědi finální.

- **1xx** – žádost ještě není vyřízena, ale je zpracovávána. Tato odpověď má informativní charakter, příkladem může být 180 Ringing, 100 Trying.
- **2xx** – označuje úspěšně vyřízenou žádost (např. 200 OK).
- **3xx** – přesměrování. Posílá informaci o novém umístění uživatele (např. 301 Moved Permanently) nebo o alternativní službě, která by měla být schopna uskutečnit hovor.
- **4xx** – indikuje chybu a znamená, že příjemce nemohl žádost zpracovat (např. 400 Bad Request).

- **5xx** – indikuje chybu na serveru (např. 500 Internal Server Error).
- **6xx** – příjemce nemohl být z nějakého důvodu kontaktován, například je zaneprázdněn (600 Busy Everywhere).

Příklad SIP odpovědi 200 OK:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP redmond.microsoft.com:5060;branch=z9hG4bKf0591c7ff61
;received=100.101.102.103
From: Bill <sip:bill@microsoft.com>;tag=ccb158e3
To: David <sip:david@goolst.cz>;tag=1e442909
Call-ID: 155ffd70f62abce5ec286877d29e1f0e@redmond.microsoft.com
CSeq: 1 INVITE
Contact: David <sip:david@goolst.cz>
Content-Type: application/sdp
Content-Length: 155
v=0
o=linus 15964 15964 IN users.freesoftware.org
s=subject
c=IN IP4 111.112.112.113
t=0 0
m=audio 10948 RTP/AVP 3 0 8 101
a=rtpmap:0 PCMU/8000
```

1.3 OfficeSIP server

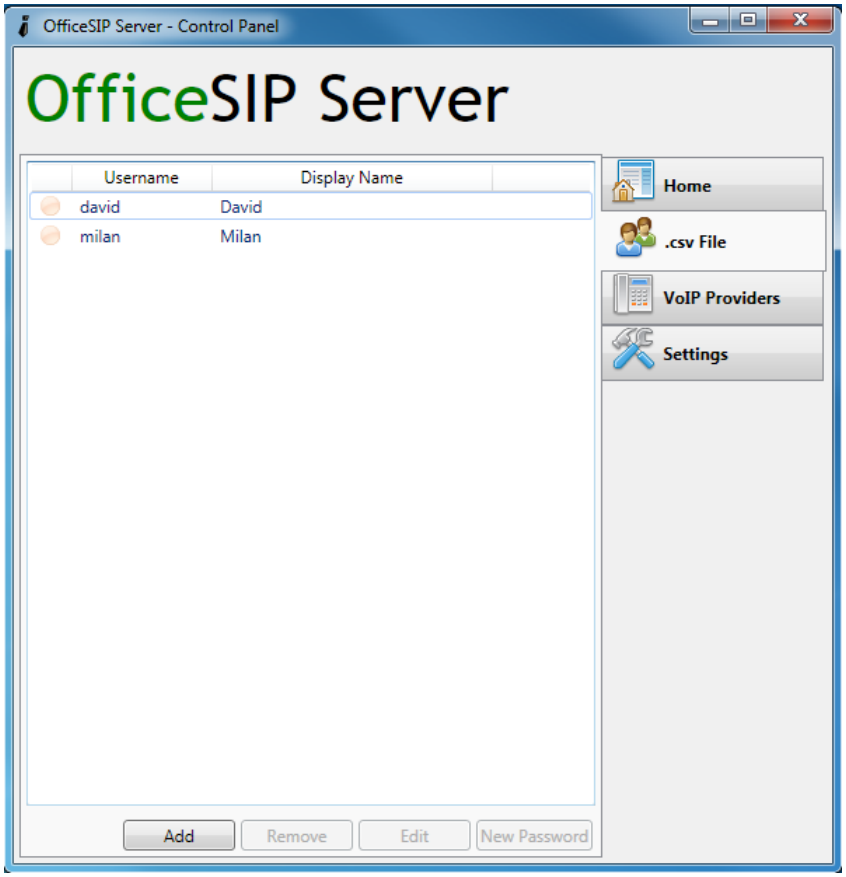
Serverová aplikace vytvořená za účelem volného volání v síti internet. Program je navržen tak, aby jej uživatelé co nejméně konfigurovali a aby měli co nejmenší práci s údržbou. V podstatě jej stačí pouze nainstalovat a začít používat.

Vlastnosti serveru:

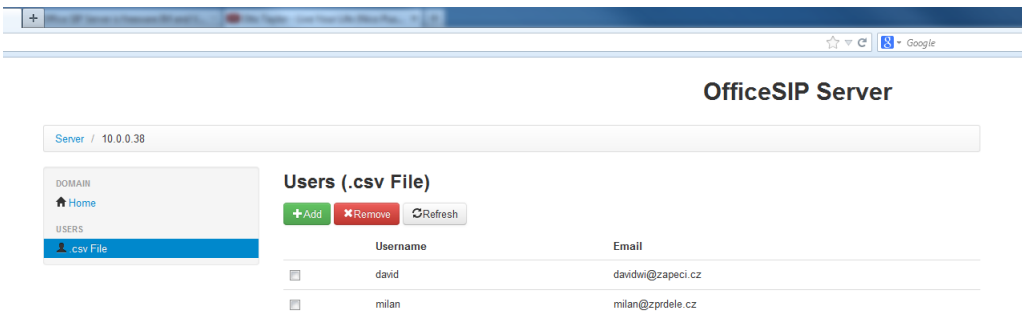
- plná podpora volné komunikace mezi počítači,
- umožňuje volání v aplikacích Windows Messenger, X-Lite a podobných aplikacích pracujících na protokolu SIP,
- jednoduchá práce s aplikací.

OfficeSIP Server je určen pro IM a VoIP. Umožňuje komunikaci v SIP-kompatibilními software a hardware klienty. OfficeSIP Server PC s Windows. Není zde ovladání přes příkazový řádek, ani žádné konfigurační soubory. OfficeSIP nabízí krom grafického

uživatelského rozhraní 1.6, také rozhraní webové 1.7. OfficeSIP Server vyžaduje minimální konfiguraci a je prakticky bezúdržbový. OfficeSIP podporuje, pro nás důležitou, technologii WebSocket.



Obr. 1.6: ukázka grafického prostředí OfficeSIP



Obr. 1.7: ukázka webového prostředí OfficeSIP

1.4 WebSocket

Podle [3] se v zásadě jedná o síťový socket, což je komunikační spojení, navázané mezi dvěma procesy, pomocí IP technologie. Socket je prakticky popsán dvojicí IP adres, protokolem a portem, který je pro komunikaci použit. Služby na vyšší úrovni tento socket používají k obousměrnému přenášení dat.

WS (WebSocket) je obdoba této technologie, přenesená do prostředí webových aplikací. WS je webová technologie, která umožňuje full-duplex komunikaci přes jedno TCP spojení. Protokol WebSocket byl standardizován IETF jako RFC 6455 v roce 2011 a WebSocket API je standardizován podle W3C (The World Wide Web Consortium). Pomocí WebSocket můžeme napsat webovou aplikaci, kde klient (aplikace v prohlížeči) může navázat obousměrné spojení se serverem, a po tomto spojení si mohou vyměňovat informace v reálném čase. WebSocket je navržen tak, aby byl implementován ve webových prohlížečích a na webových serverech, ale lze jej použít pro libovolného klienta, nebo server. WS zjišťuje, zda je v cestě nějaký proxy server a automaticky nastavuje tunel pomocí HTTP příkazu CONNECT. Tento požadavek otevře TCP/IP spojení s určeným serverem na určeném portu. Jakmile je takto vybudován tunel, mohou být zprávy volně přenášeny. Na podobném principu jsou vybudovány i WSS (WebSocket over SSL) tunely. WS nabízí jednoduché rozhraní pro navázání spojení a vzájemnou výměnu zpráv mezi klientem a serverem. Na straně klienta jsou WebSocket implementován v JavaScriptu jako třída WebSocket. Programátor může vytvořit instanci této třídy, a tím navázat spojení se serverem (pokud server tuto technologii podporuje).

```
var myWebSocket = new WebSocket("ws://www.server.com/service");
```

Pro WebSocket URL se používá prefix `ws://` nebo `wss://`. Při vytvoření nového objektu třídy WebSocket je navázáno spojení a je možno posílat zprávy. Zprávy jsou u WS prosté textové řetězce.
Objekt třídy WebSocket nabízí tři události:

- onopen - volaná při otevření spojení. Může sloužit jako příznak toho, že již lze posílat zprávy.
- onclose - oznamuje uzavření spojení.
- onmessage - slouží pro vlastní výměnu dat, je zavolána ve chvíli, kdy ze serveru přijde zpráva. Tělo zprávy je předáno v atributu data předané události.

```
myWebSocket.onopen = function(e) { alert("Connection open..."); };
myWebSocket.onmessage = function(e) {
alert( "Incoming message: " + e.data); };
myWebSocket.onclose = function(e) { alert("Connection closed."); };
```

Posílání zpráv má na starosti následující metoda. Jejím parametrem je řetězec, který má být poslán.

```
myWebSocket.send("Hello server!");
```

Na konci běhu aplikace je možno spojení uzavřít, k čemuž slouží metoda `disconnect()`. Na straně serveru je zapotřebí HTTP server, který podporuje technologii WebSocket.

1.4.1 Navázání spojení - handshake

WebSocket protokol je nezávislý protokol založený na protokolu TCP. Je podobný k HTTP, díky tomu, že jeho handshake je interpretován HTTP servery. Pro navázání spojení WebSocket, klient odešle požadavek WebSocket handshake, a server odešle WebSocket handshake reakci, jak je uvedeno v následujícím příkladu:

```
GET /mychat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

Odpověď serveru:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

Jakmile je spojení navázáno, klient a server může odeslat WebSocket data tam a zpět v plně duplexním režimu. Data jsou minimálně rámcované, s krátkou hlavičkou a následuje payload(zatížení). WebSocket přenosy jsou popsány jako zprávy, kde jednu zprávu je možné případně rozdělit do několika datových rámců. To může umožnit zasílání zpráv, které mají počáteční údaje k dispozici, ale kompletní délka zprávy je neznámá (odešle jeden datový rámec po sobě, dokud není dosaženo konce a je označen FIN bit). Protokolu může být také použit pro multiplexování několika toků paralelně.

1.4.2 SIP Over WebSockets

Dokument `draft-IETF sipcore-sip-websocket` [2] definuje způsob, jak používat WebSocket pro transport SIP zpráv. Nelze, ale nahlížet na WebSocket jako tunel pro přenos SIP zpráv. Obzvláště SIP User Agenti a proxy servery se musí chovat lehce odlišně, jestliže se používá WS namísto protokolu TCP nebo UDP. WebRTC popisuje způsob, jakým se prohlížeč stane koncovým bodem komunikace, ale nikoliv jako koncový bod SIP. Existují implementace SIP napsané v JavaScriptu, které používají WebSocket přenos pro vytvoření WebRTC relace, které správně přizpůsobí proxy servery, tak, aby byly schopny komunikovat s těmito klienty.

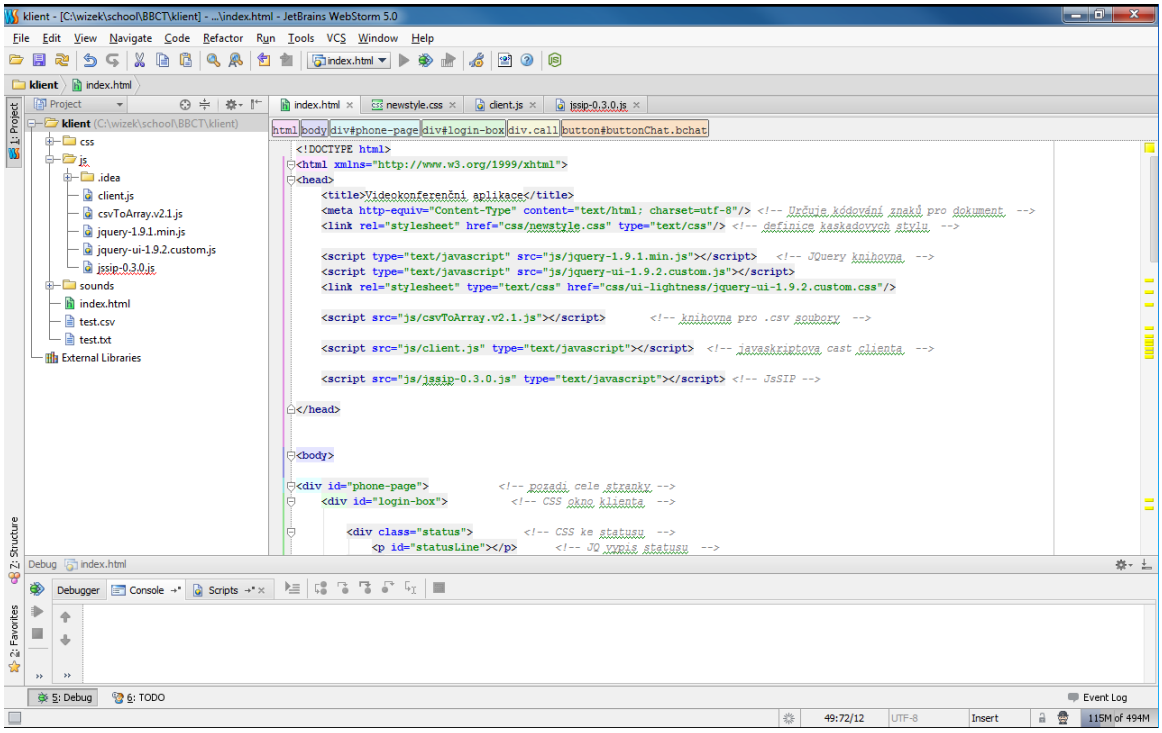
WebSocket SIP sub-protokol byl navržen pro provádění SIP žádosti a na nich odpovídajících reakcí. WebSocket zprávy lze přepravovat buď v UTF-8 textových, nebo binárnéch rámcích. Proto SIP WebSocket klienti a SIP WebSocket servery musí přijímat textové i binární rámce.

Každá zpráva SIP musí být provedena v rámci jedné WebSocket zprávy a zpráva WebSocket nesmí obsahovat více než jednu SIP zprávu. WebSocket přenos zachovává velikost zpráv, proto použití záhlaví Content-Length(udává délku zprávy) v SIP zprávách je volitelné. To zjednodušuje analyzování SIP zpráv pro klienty a servery. Není nutné stanovit velikost pomocí záhlaví Content-Length, stejně jako u protokolů UDP a SCTP (Stream Control Transmission Protocol).

1.5 Vývojové prostředí WebStorm

WebStorm je vývojové prostředí pro HTML, CSS a javascript. Podporuje Frameworky jako JQuery, Yahoo UI, Prototype, DoJo, MooTools, Qooxdoo, and Bindows. Podporuje nově i HTML 5. Umí automaticky kompletovat javaskriptový kód, proměnné, parametry a funkce. Podporuje nastavení webových prohlížečů (například Internet Explorer, Firefox, Chrome). Podporuje důmyslnou navigaci a vyhledávání v kódu. Je obsažena funkce „Quick-Fix“, která opravuje pravděpodobné chyby v kódu (například při použití syntaxe, která odpovídá starší normě). Poporuje technologie CoffeeSript a TypeScript, které mají tyto funkce:

- snadná orientace a dokončování kódu,
- stylizování kódu,
- podbarvení určitých syntaxí,
- kontrola chyb,
- hledání využití,
- automatická kompilace kódu.



Obr. 1.8: ukázka vývojového prostředí WebStorm

2 NÁVRH A ŘEŠENÍ KLIENTA

Soubory, které jsou součástí této práce:

- `index.html` - hlavní soubor v jazyce HTML. Obsahuje umístění prvků na stránce.
- `newstyle.css` - kaskádové styly, soubor obsahuje informace o celkové vzhledu aplikace.
- `client.js` - hlavní soubor v javaskriptu. Obsahuje všechny funkce a zajišťuje interaktivitu stránky.
- `jssip-0.3.0.js` - stěžejní knihovna, která obsahuje potřebné funkce pro uskutečnění hovoru, chatu atd.
- `jquery-1.9.1.js` - je javascriptová knihovna na interakci mezi JavaScriptem a HTML.
- `jquery-ui-1.9.2.js` - je javascriptový framework pro implementaci efektů a elementů do stránek.
- `csvToArray.v2.1.js` - je javascriptová knihovna, jejíž funkce umí převádět CSV (Comma-separated values) řetězce do dvourozměrného pole atd.

2.1 Třídy knihovny JsSIP

V knihovně se nachází vícero tříd. Popsány jsou jen ty nejdůležitější pro vyřešení problematiky.

2.1.1 JsSIP.UA

Metody

- **start** - Připojuje se k WS serveru a obnoví předchozí stav, pokud byl předtím zastaven. Při úplně novém začátku zaregistruje SIP domény, pokud je register parametr v konfiguraci UA nastaven na hodnotu true.
- **stop** - Uloží aktuální stav registrace a odpojí se od WS serveru. Zruší a ukončí aktivní relaci, pokud existuje.
- **register** - Zaregistruje UA.
- **unregister** - Odregistruje UA.
- **call** - Generuje odchozí multimediální hovor.
- **sendMessage** - Odešle zprávu chatu, která využívá SIP MESSAGE metody.
- **isRegistered** - Vrácí hodnotu true, pokud je UA registrován, jinak false.
- **isConnected** - Vrací hodnotu true, pokud je WS spojení navázáno, jinak false.

Události

- **connected** - Aktivována, když je navázáno WebSocket spojení.
- **disconnected** - Aktivována, když se pokus o WS připojení se nezdaří.
- **registered** - Aktivována při úspěšné registraci.
- **unregistered** - Aktivována při zrušení registrace. Tato událost je aktivována v důsledku volání žádosti `UA.unregister()`.
- **registrationFailed** - Aktivována při selhání registrace.
- **newRTCSession** - Aktivována při příchozí nebo odchozí relaci.
- **newMessage** - Aktivována při příchozí nebo odchozí žádosti MESSAGE.

2.1.2 JsSIP.RTCSession

Atributy

- **direction** - Řetězec označující, kdo začal relaci. Možné hodnoty jsou "incoming", když je relace zahájena vzdáleným uživatelem nebo "outgoing", když je relace zahájena místním uživatelem.
- **local identity** - Koresponduje s hodnotou záhlaví `INVITE From`, pokud je směr "outgoing". A s hodnotou záhlaví `To` , pokud je směr "incoming".
- **remote identity** - Koresponduje s hodnotou záhlaví `INVITE To`, pokud je směr "outgoing". A s hodnotou záhlaví `From`, pokud je směr "incoming".
- **start time** - udává dobu, kdy došlo k zahájení relace. Bere hodnotu v okamžiku aktivování události `started`.
- **end time** - udává dobu, kdy došlo k ukončení relace. Bere hodnotu v okamžiku aktivování události `ended`.

Metody

- **answer** - odpovídá na příchozí relaci.
- **terminate** - ukončí aktuální relaci, bezohledu na směr. V závislosti na stavu relace, tato funkce může odeslat žádost `CANCEL` nebo `BYE`. A na ně odpověď `2xx final`.
- **getLocalStreams** - Vrací sekvenci `MediaStream` objektů, které představují data poslaná v objektu ze třídy `RTCSession`.
- **getRemoteStreams** - Vrací sekvenci `MediaStream` objektů, které představují data přijatá v objektu ze třídy `RTCSession`.

Události

Tato třída definuje události, které umožňují reagovat spouštěním daných funkcí na podněty uživatele.

- **progress** - Aktivována, když přijímá odpovědi 1xx a na ně odesílá INVITE žádosti.
- **started** - Aktivována při přijetí hovoru.
- **ended** - Aktivována při ukončení spojení.
- **failed** - Aktivována, když nebylo možno ustavit spojení.

2.1.3 JsSIP.Message

Metody

- **send** - Posílá zprávy skrz WS spojení. Pouze pro odchozí zprávy.
- **accept** - Odpovídá kladně na příchozí zprávu. Potvrzuje přijetí pouze pro příchozí zprávy.
- **reject** - Odpovídá negativně na příchozí zprávu. Oznamuje nedoručení zprávy.

Události

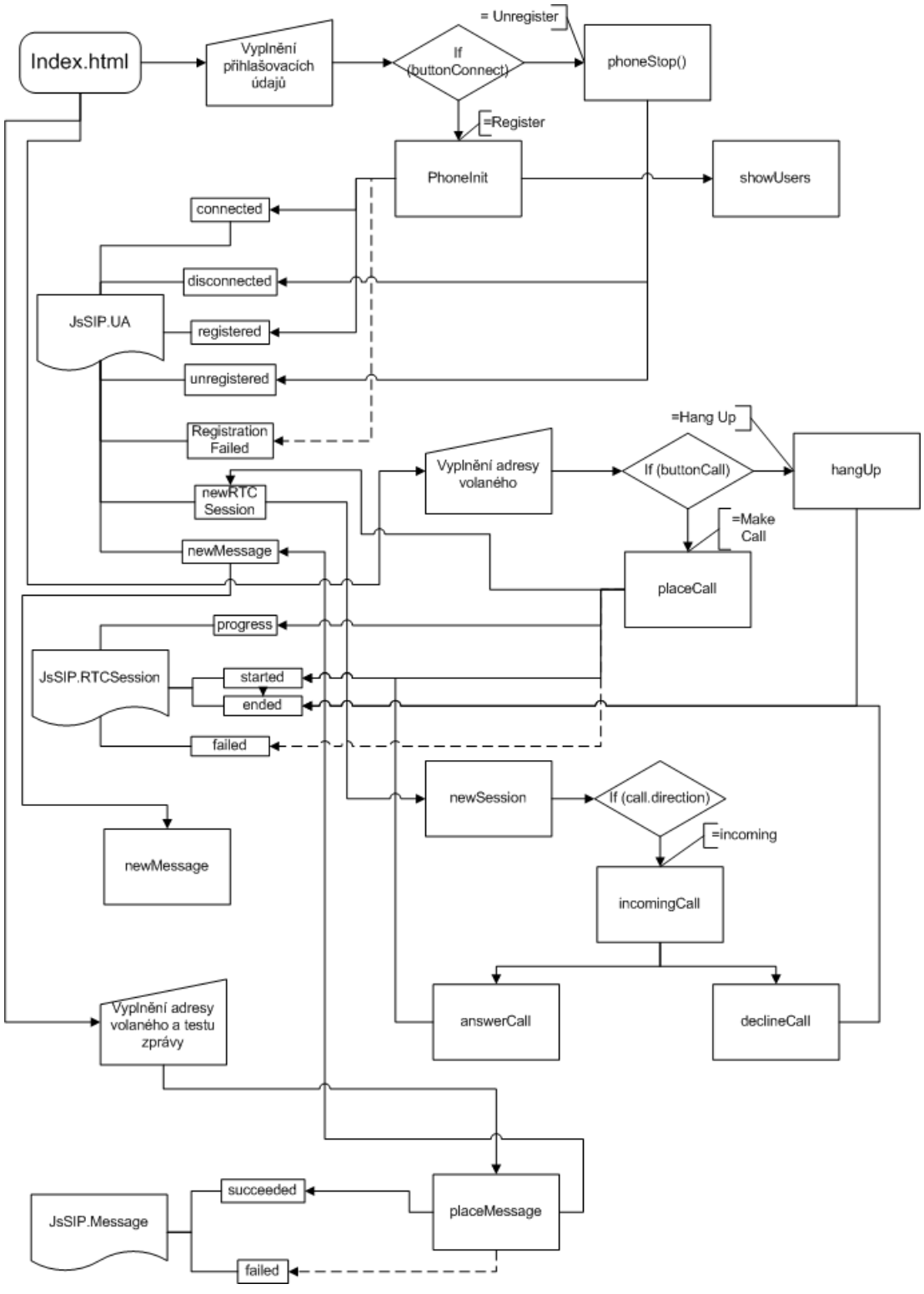
- **succeeded** - Aktivována při přijetí pozitivní odpovědi na žádost MESSAGE.
- **failed** - Aktivována, při přijetí negativní odpovědi na žádost MESSAGE.

2.2 Funkce klienta - architektura

Na obrázku 2.1 je znázorněno schéma podle které se volají jednotlivé funkce a události aplikace.

Zaregistrování uživatele

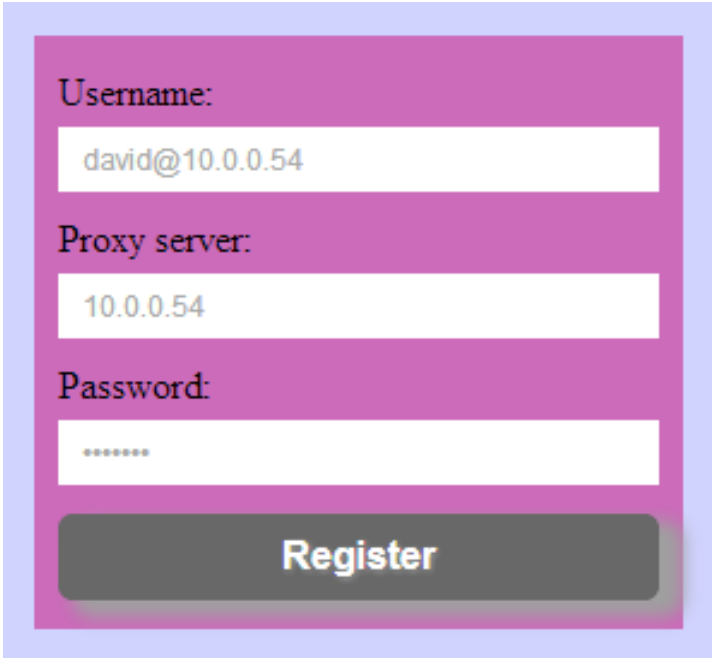
Na obrázku 2.2 vidíme jak vypadá přihlašovací okno na výchozí stránce `index.html`. Zde se vyplní základní přihlašovací údaje. Jako Username ve tvaru `david@10.0.0.54`, Proxy server ve tvaru `10.0.0.54` a heslo, které je zobrazeno skrytě a reprezentováno puntíky. Při kliknutí na tlačítko Register se zobrazí stav přihlášení například `SIP registered as david@10.0.0.54`. A také se zobrazí ovládací prvky jako na obrázku 2.3. A algoritmus zavolá funkci `phoneInit` 2.2.1. Vyvolají se události `connected` a `registered`. Zavolá se funkce `showUsers` 2.2.12. Také se může v případě neúspěšného přihlášení aktivovat událost `registrationFailed`.



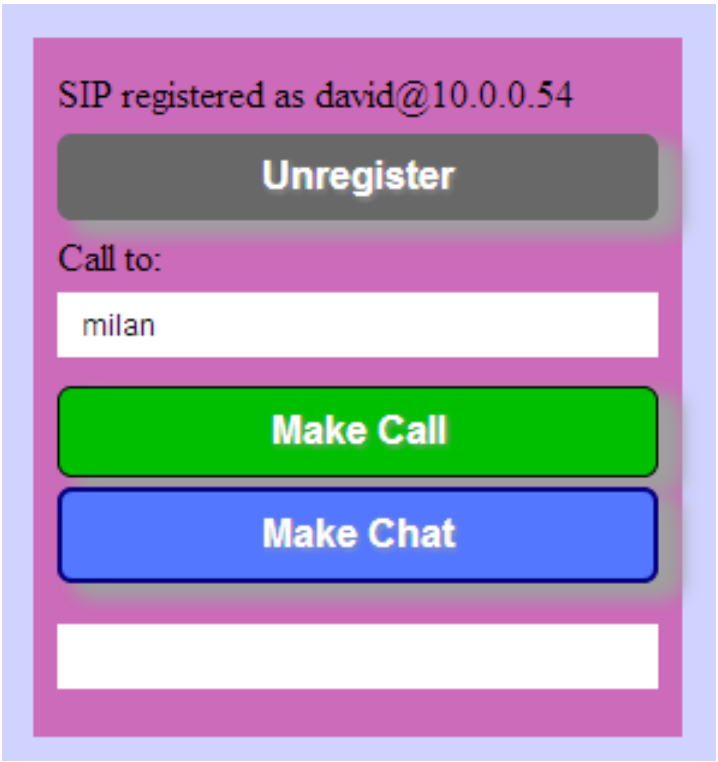
Obr. 2.1: diagram aplikace

Odregistrování uživatele

Při kliknutí na tlačítko Unregister se spustí funkce `phoneStop` 2.2.2. Aktivují se události `disconnected` a `unregistered`.



Obr. 2.2: ukázka přihlašování na hlavní stránce



Obr. 2.3: základní menu klienta

Nyní je aplikace připravena na zahájení příchozího nebo odchozího hovoru, potažmo na příchozí nebo odchozí zprávu.

Odchozí hovor

Pro zahájení odchozího hovoru, se vybere ze seznamu jméno uživatele a zadá se do vstupu `Call to:` jméno volaného. Klikne se na tlačítko Make Call. Tím se zavolá funkce `placeCall` 2.2.3. Což zavolá událost `newRTCSession`. Ta zavolá funkci `newSession` 2.2.5. Během vytáčení je zavolána událost `progress`. Jakmile volaný účastník „zvedne“ hovor, tak se vyvolá událost `started`, ve které se zobrazí okna pro video vzdálené i lokální. A spustí se jejich „streamování“ jako na obrázku 2.4. Při vytáčení i během hovoru může také dojít k vyvolání události `failed`.

K ukončení hovoru slouží tlačítko "Hang Up", které vyvolá funkci `hangUp` 2.2.11. A také událost `ended`, která schová elementy vzdáleného i lokálního videa.

Obr. 2.4: klient v průběhu relace

Příchozí hovor

Při příchozím hovoru se zavolá opět událost `newRTCSession`, která zavolá funkci `newSession` 2.2.5. Ta zavolá v případě příchozího hovoru funkci `incomingCall` 2.2.8. Při přijetí hovoru hovoru se zavolá funkce `answerCall` 2.2.9. Při odmítnutí hovoru se zavolá funkce `declineCall` 2.2.10. Příchozí hovor se také dá zrušit za průběhu, stejně jako odchozí hovor.

Odchozí zpráva

Pro odeslání zprávy se vybere ze seznamu jméno uživatele a zadá se do vstupu `Call to:` jméno volaného(chatovaného). A vepíše se do vstupu pod tlačítkem pro chat znění zprávy. Klikne se na tlačítko 'Make Chat'. Jak je vidět na obrázku 2.5. Tím se zavolá funkce `placeMessage` 2.2.4. Což zavolá událost `newMessage`. Ta zavolá funkci `newMessage` 2.2.6.

Příchozí zpráva

Při příchozím hovoru se zavolá opět událost `newMessage`, která zavolá funkci `newMessage` 2.2.6. Ta zavolá v případě příchozího hovoru funkci `incomingMessage` 2.2.7.

2.2.1 funkce phoneInit

V této funkci (metodě) se vstupní parametry uloží do proměnných, které jsou součástí konfigurace k inicializování objektu coolPhone ze tridy JsSIP.UA. Dále se aktivují všechny události této třídy. Nakonec se zavolá metoda start ze objektu coolPhone, zadáním příkazu `coolPhone.start`.

2.2.2 funkce phoneStop

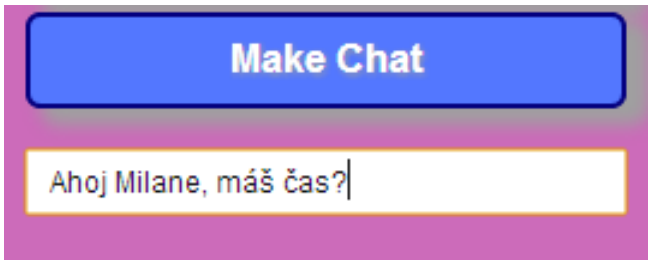
Pokud probíhá relace, tak ji ukončí příkazem `coolPhone.stop` a odregistruje uživatele příkazem `coolPhone.unregister`. Schovají se elementy stránky pro volání a objeví se přihlašovací elementy. Jako na obrázku 2.2 s vypísem hlášky `SIP unregistered`.

2.2.3 funkce placeCall

Nejprve se uloží adresa volaného do proměnné. Dále se uloží do proměnných konfigurace pro příkaz `coolPhone.call`, který zahájí odchozí hovor. Tlačítko se změní na Hang Up a zčervená.

2.2.4 funkce placeMessage

Nejprve se uloží adresa příjemce zprávy a tělo zprávy do proměnné. Aktivují se události třídy `JsSIP.Message`. Dále se uloží do proměnných konfigurace pro příkaz `coolPhone.sendMessage`, který odešle zprávu. Při úspěšném poslání zprávy vyvolá událost `succeeded`. Při nezdařilém odeslání vyvolá událost `failed`.



Obr. 2.5: ukázka zadávání textu zprávy

2.2.5 funkce newSession

Kde se opět uloží do proměnných všechny potřebné atributy spojené s hovorem. Pokud jde o příchozí hovor tak se zavolá funkce `incomingCall` 2.2.8. Aktivují se události třídy `JsSIP.RTCSession`.

2.2.6 funkce newMessage

Kde se opět uloží do proměnných všechny potřebné atributy spojené se zprávou. Pokud jde o příchozí zprávu, tak se zavolá funkce `incomingMessage` 2.2.7. A spustí se zvuk příchozí zprávy.

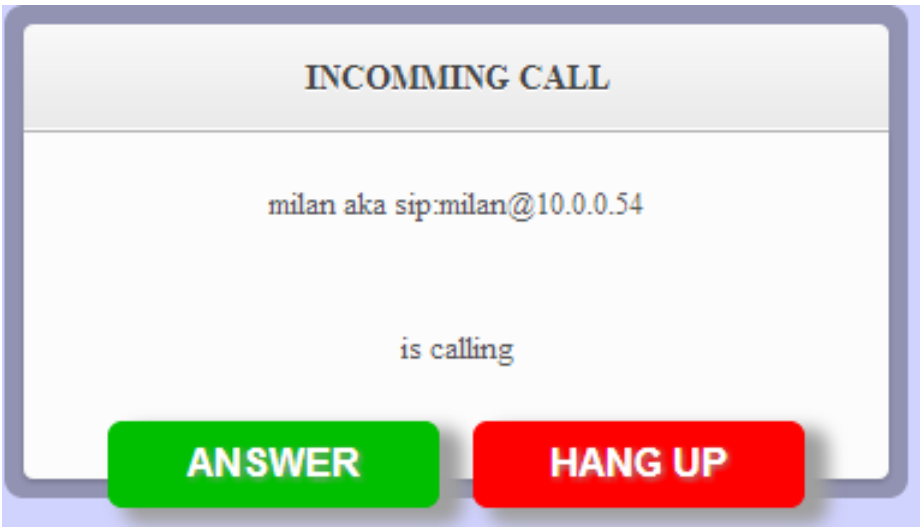
2.2.7 funkce incomingMessage

Tato funkce zobrazí element `chat div`, což je okno pro zobrazení příchozí zprávy. A v tomto okně zprávu zobrazí, názorně na 2.6

Obr. 2.6: okno s příchozí zprávou

2.2.8 funkce incomingCall

Na začátku této funkce se objeví element `answerCallWindow`. Vypíše do něj jméno a adresu volajíciho jako na obrázku 2.7. A spustí vyzvánění.



Obr. 2.7: okno answerCallWindow

2.2.9 funkce answerCall

Na začátku této funkce se naplní proměnné důležité pro multimediální hovor a schová se element `answerCallWindow`. Objeví se okna pro „streamování“ obou videí. Příkazem `call.answer` se přijme hovor. Dále se změní tlačítko na 'Hang Up' a zčervená.

2.2.10 funkce declineCall

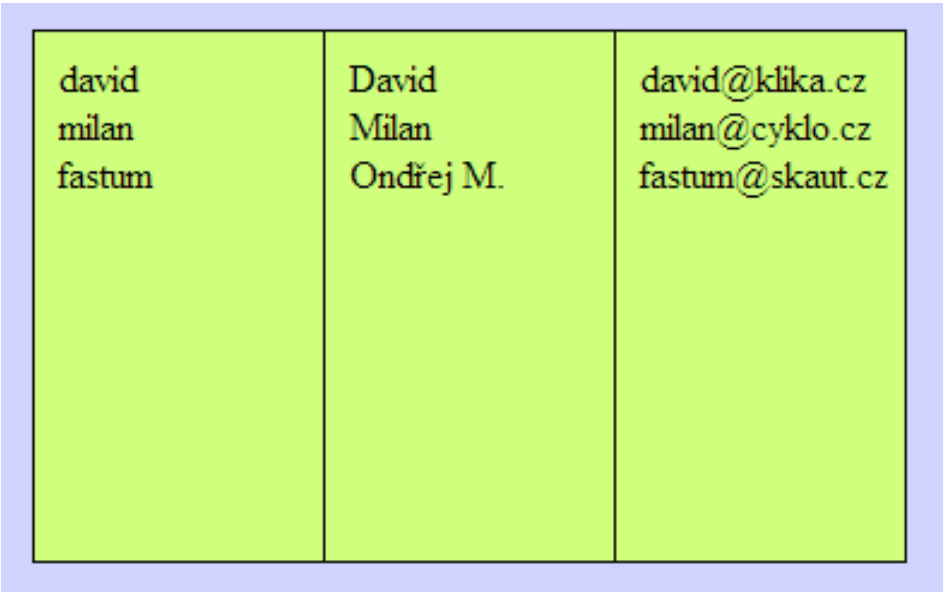
Na začátku této funkce se schová element `answerCallWindow`. Příkazem `call.terminate` se zruší hovor. Tlačítko zůstane ve stavu v jakém se nachází.

2.2.11 funkce hangUp

Tato funkce změní barvu tlačítka na zelenou a změní nápis na 'Make Call'. A příkazem `call.terminate` zruší hovor.

2.2.12 funkce showUsers

Nejdříve se zviditelní okno pro zobrazení adresáře. Kde figurují jednotlivé údaje o uživatelích přihlášených na stejný server. V této funkci probíhá načítání údajů o uživatelích ze souboru typu CSV, kde si ukládá tyto údaje server. Výsledky se následně vypisují na stránku jako na obrázku 2.8.



Obr. 2.8: okno se seznamem uživatelů stejné domény

3 TESTOVÁNÍ APLIKACE

U aplikací všeobecně, zvláště u aplikací tohoto druhu, se používají výpisy pro usnadnění ladění. Vypisují se buď na standardní výstup někde v aplikaci, a nebo do konzole webového prohlížeče. Jsou zde kvůli informování vývojáře, popřípadě uživatele, co aplikace zrovna provádí(výpis aktivování a volání události, SIP zprávy, chybová hlášení). Výpisy jsou nejdůležitější v době pádu aplikace, kde zjistíme, ve které části došlo k chybě nebo k neočekávatelné změně. Ovšem při běžném používání aplikace by došlo spíše k rušení uživatele, proto se hlášky vypisují pouze do konzole. A do aplikace se vypisují pouze nejdůležitější hlášení(stavy). V této kapitole se testuje aplikace jako celek, nikoliv jako dílčí funkce.

Aplikace, představující webovou stránku, byla v průběhu vývoje testována lokálně. Na počítači byl nainstalován operační systém Windows 7. A také na tomto počítači bylo nainstalováno vývojové prostředí WebStorm, kde se aplikace vyvíjela. Dále jsou uvedeny parametry při testování závěrečném.

Predispozice testování

Na testování je potřeba dvou počítačů s těmito požadavky:

- vstupní(mikrofon) a výstupní zařízení (reproduktory, sluchátka),
- webcamera(není nezbytně nutná),
- připojení k internetu nebo místní síti,
- nainstalovaný systém windows 7 a webový prohlížeč (podle webRTC můžeme použít Google Chrome nebo Mozilla Firefox),
- vytvořit výjimku v základním nastavení firewallu.

Dále je potřeba funkční proxy server, v tomto případě officeSIP server, s vytvořenými uživatelskými účty.

Průběh testování

Na jednom počítači se spustila aplikace s přihlašovacími údaji (Username:`david@10.0.0.54`, Proxy server:`10.0.0.54`, Password:`davidek`), na druhé počítači se spustila aplikace s těmito údaji (Username:`milan@10.0.0.54`, Proxy server:`10.0.0.54`, Password:`milanek`). Testuje se následujícím způsobem:

1. Uživatel david zavolá uživateli milan, navážou komunikaci, kterou ukončí uživatel david.
2. Uživatel david zavolá uživateli milan, který hovor nepřijme.
3. Uživatel milan zavolá uživateli david, navážou komunikaci, kterou ukončí uživatel david.
4. Vymění si navzájem každý textovou zprávu.

5. Oba uživatelé se odregistrují ze serveru.

Testování aplikace proběhlo úspěšně v prohlížeči Google Chrome, kdežto v prohlížeči Mozilla Firefox nikoliv. V tomto prohlížeči proběhlo zaregistrování na server neúspěšně.

4 ZÁVĚR

Obsahem práce bylo prostudovat možnosti uskutečnění hlasového i videohovoru z webového prohlížeče bez nutnosti instalace jakéhokoli dalšího softwaru na koncový počítač. Což se povedlo, protože bylo nalezeno API JsSIP, které splňuje všechny tyto požadavky.

Pro signalizaci bylo použito protokolu SIP. Byl vybrán proxy server officeSIP, který samozřejmě podporuje technologii WebRTC. Softwarový klient kromě standardních funkcí umožňuje textový chat a propojení s adresářem dané struktury. Funkčnost aplikace byla ověřena testováním.

Pro výslednou implementaci byl zvolen jazyk Javaskript, který zajišťuje interakci webové stránky. Ta je psána v jazyce HTML. Vývoj aplikace byl proveden v prostředí WebStorm, což se ukázalo jako velmi vhodné vzhledem k množství funkcí pro usnadnění při vývoji. Dále se dá považovat za zdařilé výběr knihoven (JsSIP, JQuery, csvToArray).

Dosaženo bylo všech cílů daných na začátku. Do budoucna by bylo vhodné určitě předělat celkový vzhled a ovládání aplikace. Ten vznikal na základě potřeb vývoje, ovšem s přibývajícími funkcemi se stalo uživatelské rozhraní neideální. Dala by se také doprogramovat možnost přenosu přes šifrovaný protokol WSS. Dále by bylo možné otestovat i jiné proxy servery, protože námi používaný officeSIP, nenabízí mnoho možností. Při zjištění, že aplikace funguje pouze v prohlížeči Google Chrome, je znát, že vývoj webRTC aplikací je stále ve vývoji a musíme počkat na její masové využití do budoucna.

LITERATURA

[1] WebRTC 1.0: Real-time Communication Between Browsers. http://www.w3.org/.

[2] Castillo, I. B.; Villegas, J. M.; Pascual, V.: The WebSocket Protocol as a Transport for the Session Initiation Protocol (SIP). RFC 3261 (Proposed Standard), Březen 2013, draft-ietf-sipcore-sip-websocket-08.
URL http://tools.ietf.org/html/draft-ietf-sipcore-sip-websocket-08

[3] Fette, I.; Melnikov, A.: The WebSocket Protocol. RFC 6455 (Proposed Standard), Prosinec 2011.
URL http://www.ietf.org/rfc/rfc6455.txt

[4] Lubbers, P.; Albers, B.; Salim, F.: *HTML5*. COMPUTER PRESS, 2011, ISBN 9788025135396, 304 s.

[5] Rosenberg, J.; Schulzrinne, H.; Camarillo, G.; aj.: SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), Červen 2002, updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141, 6665, 6878.
URL http://www.ietf.org/rfc/rfc3261.txt

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

SIP Session Initiation Protocol

IM Instant Messaging

HTTP Hypertext Transfer Protocol

SMTP Simple Mail Transfer Protocol

IETF Internet Engineering Task Force

RFC Request for Comments

TCP Transmission Control Protocol

UDP User Datagram Protocol

UA User Agent

UAC User Agent Client

UAS User Agent Server

URI Uniform Resource Identifier

RTC Real-Time Communications

API Application Programming Interface

WS WebSocket

WSS WebSocket over SSL

W3C The World Wide Web Consortium

SCTP Stream Control Transmission Protocol

CSV Comma-separated values

AV Audio-Video

IDE Integrated development environment

SEZNAM PŘÍLOH

A OBSAH CD