Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Tímový projekt 1

Zdieľaná identita vo WebRTC doménach

Dokumentácia pre zimný semester

Číslo tímu: 7
Členovia tímu: Bc. Šimon Šiplák, Bc. Maroš Medvec, Bc. Marián Sekeráš, Bc. Adam Števko, Bc. Roman Zelenaj
Vedúci tímového projektu: Ing. Ján Murányi
Ročník, typ štúdia: 1, inžinierske štúdium
Akademický rok: 2012/2013

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

Skratka	Anglický výraz
SSO	Single Sign On
WebRTC	Web Real Time Comunication
SIP	Session Initiation Protocol
HTML	HyperText Markup Language
XRI	Extensible Resource Identifier
API	Application programming interface
W3C	World Wide Web Consortium
IETF	Internet Engineering Task Force
P2P	Peer to Peer
RTP	real-time transport protocol
TCP	Transmission Control Protocol
CSS	Cascading Style Sheets
XHTML	Extensible HyperText Markup Language
PHP	HyperText Preprocessor
DOM	Document Object Model
JSON	JavaScript Object Notation
URL	Uniform Resource Locator

Tab. č. 0.1 Zoznam použitých skratiek v dokumente

1 Úvod

Táto kapitola obsahuje informácie o zadaní tímového projektu, o účele, rozsahu dokumentu a tiež o skratkách použitých v dokumente. Účelom tohto dokumentu je zachytenie všetkých fáz pri vývoji architektúry a systému pre tímový projekt.

1.1 Zloženie tímu

Pedagogický vedúci	Ing. Ján Murányi
Členovia tímu	Bc. Roman Zelenaj
	Bc. Maroš Medvec
	Bc. Marián Sekeráš
	Bc. Adam Števko
	Bc. Roman Zelenaj

Tab. č. 1.1 Zoznam členov tímu

1.2 Zadanie projektu

WebRTC je štandard, ktorý otvára webovým prehliadačom RTP protokol a s reláciami založené v reálnom čase na RTP protokole. Cieľom projektu je vytvoriť architektúru, ktorá umožní využívať zdieľanú identitu používateľa (OpenID) na využívanie služieb v doménach WebRTC.

V rámci tímového projektu je potrebné pripraviť platformu WebRTC na využívanie služieb s použitím zdieľanej identity a zabezpečenie autorizácie, autentifikácie a účtovania smerom k domácemu poskytovateľovi služieb, pripraviť prípady použitia a vyhodnotiť dosiahnuté výsledky.

Prirodzenou súčasťou práce je administrácia platformy a vytvorenie nielen dokumentácie k projektu, ale aj celej platformy vo vhodnom formáte.

1.2.1 Ciele projektu

Cieľom tohto tímového projektu je analyzovať technológiu WebRTC, RTP protokol a ďalšie príbuzné technológie, na základe ktorých bude navrhnutá a vytvorená architektúra, ktorá umožní využívať zdieľanú identitu používateľa na využívanie hlavne multimediálnych služieb v doménach WebRTC. Pre potrebnú analýzu budú musieť byť tiež preskúmané už existujúce riešenia, ktoré danú technológiu využívajú. Výstupom projektu by mala byť platforma WebRTC umožňujúca využívanie služieb s použitím

zdieľanej identity, v ktorej bude zabezpečená autorizácia, autentifikácia a účtovanie smerom ku koncovému poskytovateľovi služieb. Finálna platforma bude musieť byť tiež otestovaná a pri jej vyhodnotení budú musieť byť použité vhodné prípady použitia.

1.3 Účel a rozsah dokumentu

Tento dokument vznikol ako výsledok práce na predmete Tímový projekt 1. počas zimného semestra k 16.11.2012. Dokument obsahuje správu k projektu s názvom „Zdieľaná identita vo WebRTC doménach“ a je určený vedúcemu projektu, posudzujúcemu tímu, ako aj pre všetkých, ktorí sa o danú problematiku zaujímajú.

Účelom tohto dokumentu je priblížiť čitateľovi analýzu problémovej oblasti a opísať možnosti ako realizovať architektúru, ktorá umožní využívať zdieľanú identitu používateľa (OpenID) na využívanie služieb v doménach WebRTC.

1.3.1 Prehľad dokumentu

Dokument má nasledovnú štruktúru:

- Kapitola 1: Úvod do problematiky a štruktúra dokumentu
- Kapitola 2: Analýza problémovej oblasti
- Kapitola 3: Špecifikácia požiadaviek
- Kapitola 4: Návrh systému
- Prototyp 5: Prototyp
- Kapitola 6: Literatúra

2 Analýza

Táto časť projektu sa zaoberá analýzou problematiky spojenou so zdieľanou identitou, doménami používajúcimi technológiu WebRTC a hlavne analýzou zvolených technológií pre tento projekt, ktoré budeme využívať pri vytváraní finálneho produktu tímového projektu.

2.1 Analýza problematiky

V súčasnosti je veľká časť obsahu internetových portálov prístupná takmer každému používateľovi surfujúcemu po Internete. Čoraz viac sú však poskytované špeciálne služby pre používateľov, ktorí sa určitým spôsobom zaviažu, že opätovne navštívia alebo využijú služby danej stránky, respektíve organizácie. Aby si anonymní klienti vytvorili určitú identitu pod ktorou sú známi, vytvárajú si používateľské kontá. Tento jednoduchý spôsob evidencie umožňuje organizácií jednoduché rozdelenie sprístupnených služieb pre rôznych ľudí. Základom procesu autentifikácie je teda meno a heslo , uložené v centrálnej databázovej štruktúre, voči ktorému sa daný používateľ autentifikuje. Bežnou praxou bolo, že každá organizácia používala vlastnú databázu svojich klientov. Heslá používateľov boli kvôli dodržaniu bezpečnosti zašifrované. Ak si chcel klient napríklad pozrieť obsah svojej poštovej schránky, musel zadať meno a heslo. Následne došlo k overeniu zadaných prihlasovacích údajov a ak bol tento proces úspešný, tak ho systém presmeroval do zóny, kde nemal bežný neprihlásený používateľ prístup. Tento prístup separátnych služieb s centralizovaným lokálnym prihlasovaním sa používa u veľkého množstva internetových portálov.

Z hľadiska bezpečnosti však nie veľmi vhodné riešenie, ak sa používateľ prihlasuje na centralizovaný systém a odosiela mu svoje bezpečnostné heslo. Útočník sa môže pri nedostatočne dobre zabezpečených komunikačných sieťach ľahko dostať ku heslám používateľa. Problém nastáva taktiež pri periodických zmenách hesiel a vytváraní komplexných hesiel. Takáto politika môže viesť až do stavu, že používateľ stratí prehľad o používaných prihlasovacích údajoch (používateľské meno a heslo). Ak chce byť teda používateľ exkluzívnym členom nejakého portálu, je nutné, aby si vytvoril nejakú identitu, pod ktorou je známy. Tak môže vzniknúť obrovské množstvo používateľských účtov, pre ktoré je nutné vymyslieť prihlasovacie meno a heslo. Často sa stáva, že používateľ vo vidine uľahčenej práce a ľahšom zapamätaní si hesla použije na viaceré účtu portálov rovnaké heslo. Ak nastane hore spomínaný prípad odcudzenia hesla z databázy organizácie, ktorá ju mala nedostatočne zabezpečenú, tak útočník zrazu získa prístup ku veľkému počtu účtov daného používateľa.

Takýmto problémom bolo nutné predísť najmä pri firmách, ktoré často obsahovali citlivé informácie. Vznikla potreba ochrany údajov pri zaobchádzaní s používateľským menom a heslom. Keďže v rámci jednej organizácie môže byť veľa aplikácií alebo zdrojov, ktoré sú zabezpečené heslom, bolo nutné vytvoriť nejakú centrálnu identitu a zminimalizovať počet prenosov citlivých údajov cez sieť. Bolo treba taktiež redukovať miesta, na ktorých boli uložené prihlasovacie údaje. Väčšina veľkých firiem a spoločností má teda nejakým spôsobom riešený Single Sign On (SSO), kde identita používateľa je centralizovaná a s jediným prihlásením do podnikovej siete používateľ získava prístup ku všetkým IT zdrojom a aplikáciám v rámci danej siete. Čoraz viac používatelia využívajú zdroje a aplikácie tretích strán, ktoré sú mimo firemnej siete, v dôsledku čoho sa tradičné systémy na správu identít používateľov začínajú rúcať.

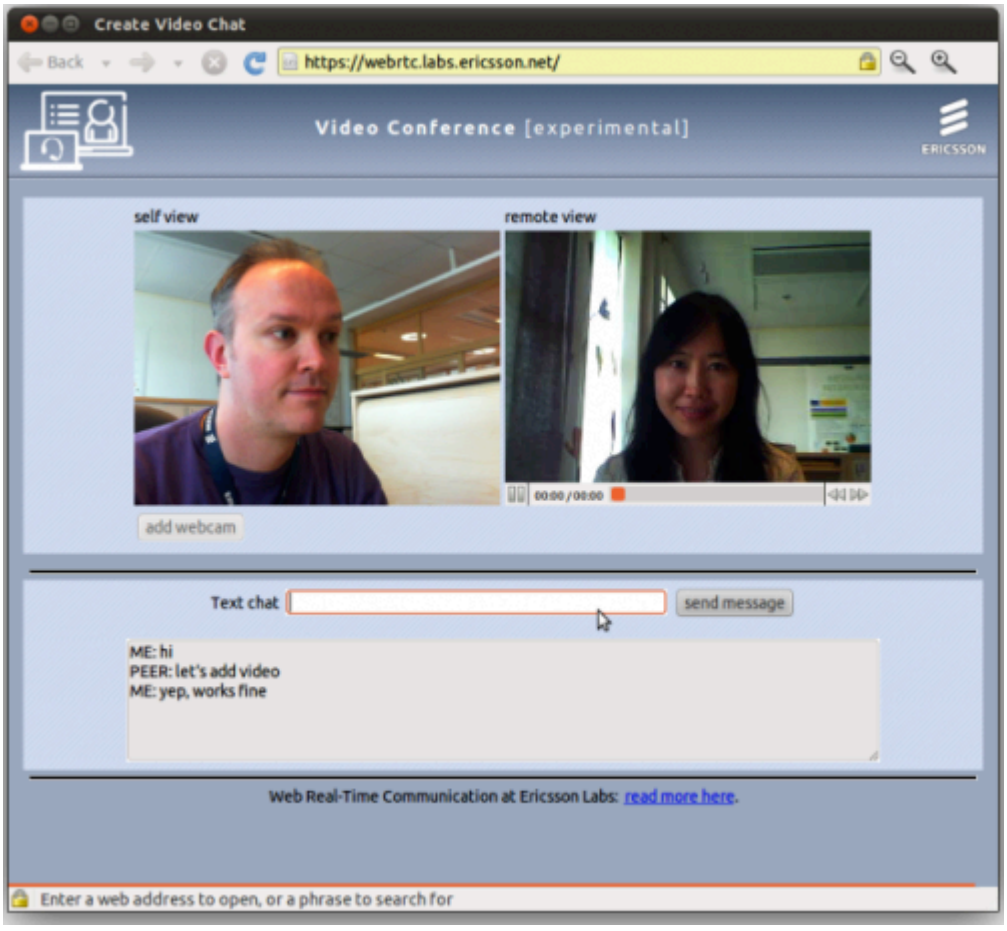
Riešením tohto problému je decentralizovaný systém OpenID. Na prvý pohľad sa môže zdať, že je to identické riešenie ako je SSO. Kým základná filozofia SSO spočíva v prihlásení sa na jednom mieste, pričom dôjde ku automatickej autentifikácii na ostatných lokalitách, u OpenID ide o delegovanie autentifikácie OpenID poskytovateľovi, aby bol používateľ následne schopný sa prihlásiť na rôzne stránky s jednou sadou poverení (credentials). Čoraz väčší počet organizácií začína využívať toto perspektívne autentifikačné riešenie. Medzi tých najvýznamnejších OpenID poskytovateľov sa zaradili aj portály ako Google, Yahoo!, PayPal, AOL, MySpace, Orange a mnoho ďalších.

Pokrok bol taktiež zaznamenaný aj pri prenášaní multimediálneho obsahu medzi viacerými účastníkmi. Donedávna platilo, že kto chcel naplno využiť potenciál komunikačných služieb typu Skype či ICQ potreboval desktopový klientsky softvér. Ich online verzie často vyžadujú rozšírenie pre webový prehliadač. Ak už je online verzia nejakej komunikačnej služby založená čisto na webových technológiách, tak podporovala striktne len textovú komunikáciu. Pre uskutočnenie komunikácie medzi viacerými používateľmi nie len v textovej forme, bolo nutné použitie softvéru tretej strany (napr. Flash player). Ten však spôsobuje často nestabilitu alebo nie je kompatibilný. Ďalší problém je v náročnosti flash playera, ten totiž využíva vo veľkej miere dostupné prostriedky počítača a vyžaduje väčšiu výpočtovú silu. Riešením tohto problému je zavedenie technológie WebRTC.

Táto technológia je implementovaná v samotnom webovom prehliadači a umožňuje plnohodnotnú komunikáciu medzi používateľmi. Nahrádza tak samostatné aplikácie, ktoré bolo nutné nainštalovať. Súčasná situácia WebRTC je taká, že niektoré webové prehliadače ho už plne podporujú. Preto začína vznikať veľa rôznych experimentálnych chatovacích portálov, ktoré poskytujú prenos nielen zvuku, ale aj obrazu prostredníctvom WebRTC technológie. Príklad typického chatu na báze WebRTC

je ilustrovaný na obrázku č. 2.1.

Potenciál projektu WebRTC v oblasti komunikácie naznačuje v praxi aj projekt spoločnosti Doubango Telecom. Tá vytvorila prvý SIP klient založený na WebRTC. Pri vývoji bol použitý iba JavaScript a HTML 5. Tento projekt je určený pre zákazníkov VoIP operátorov, ktorí poskytujú služby na báze protokolu SIP. Prostredníctvom experimentálneho SIP klienta je možné realizovať klasické volania alebo video hovory. K dispozícii je aj podpora pre viacero telefónnych liniek, pridržanie hovoru či používateľských účtov.



Obr. č. 2.1 Príklad video chatu v prostredí webového prehliadača prostredníctvom technológie WebRTC.

2.2 Existujúce riešenia v danej oblasti

V oblasti poskytovania multimediálnych video služieb sa začína na internete objavovať čoraz väčšie množstvo organizácií, ktoré ponúkajú registrovaným klientom po zaplatení určitej sumy pozrieť si film, videozáznam alebo nejakú reláciu on-line bez nutnosti vlastnenia televízie. Takéto internetové video archívy majú niekoľko výhod oproti klasickej televízií.

- Prístup ku odvysielanej relácií alebo seriálu má používateľ aj potom ako bola odvysielaná. Tieto tituly sú uskladnené vo video archívoch. Používateľ nie je teda odkázaný na presný časový harmonogram televízie.

- Takéto portály často poskytujú širokú škálu filmov a relácií. Fungujú vo veľkej miere aj ako video požičovňa. Používateľ uhradením určitej finančnej sumy (či je už tarifikácia nastavená na platbu za jednotlivé tituly alebo paušálne sa platí mesačne) si môže pozrieť požadované video.

- Poskytované vysielania majú nastaviteľnú kvalitu, čo znamená, že používateľ si sám môže nastaviť v akej chce kvalite vidieť daný seriál, film alebo reláciu. Výhodou je to najmä vtedy, ak má používateľ pomalšie pripojenie k Internetu alebo obmedzené množstvo dát na využitie.

- Používateľ pri takýchto portáloch si väčšinou platí len za veci, ktoré aj naozaj pozerá. V prípade televízie totiž platí paušálny poplatok za veci, ktoré ho často nezaujímajú.

Okrem vyššie spomínaných výhod sa však objavujú aj nevýhody, ktoré prináša tento systém on-line vysielania.

- V prvom rade je potrebné mať pripojenie na Internet. Najviac sa odporúča mať neobmedzený tok dát, lebo často dôjde k prenosu veľkého množstva dát. Dôležitá je aj prenosová rýchlosť. Pri pomalých spojeniach musí klient dlho čakať kým sa mu načíta zvolené video.

- Webové prehliadače pri poskytovaní takejto služby musia mať nainštalovanú aktuálnu verziu softvéru tretej strany na zobrazenie videa. Zvyčajne ide o flash

player. Toto býva najväčšie negatívum, keďže často sú používatelia stránky na zariadenia ktorými nie sú vlastníkmi, a tam ak nemajú práva na inštalovanie nového softvéru, tak nemôžu využívať služby týchto portálov.

Existuje už niekoľko komerčných prevádzok, ktoré poskytujú služby takéhoto typ napríklad webové portály Huste.tv alebo Voyo.sk.

2.3 Analýza technológií

Pri implementovaní OpenID a WebRTC je potrebných niekoľko hlavných technológií, ktoré nám umožnia bezproblémové nasadenie do reálneho prostredia Internetu. Tieto technológie sú použité aj v terajších implementáciách podobného charakteru.

2.3.1 OpenID

OpenID je decentralizovaná technológia, ktorá slúži na správu identít. Pomocou OpenID je možné používať jeden profil, a ten využívať vo webových službách, ktoré OpenID podporujú, tým pádom nie je potrebná opätovná registrácia. OpenID je štruktúra, ktorá rieši komunikáciu medzi správcom identity a OpenID akceptovanou stranou. OpenID správy môžu mať rôzne atribúty, napríklad meno, priezvisko, vek atď.

Komunikácia pri OpenID prebieha medzi koncovým používateľom, napríklad prostredníctvom webovej stránky a OpenID poskytovateľom. Na obrázku č. 2.2 je možné vidieť priebeh komunikácie s OpenID poskytovateľom.

Komunikovať je možné v dvoch režimoch:

- checkid_immediate - pri tomto režime poskytovateľ OpenID nekomunikuje priamo s koncovým používateľom, ale komunikácia prebieha pomocou prostredníka a tým je webová stránka, ktorú používateľ navštívil a chcel využiť OpenID bez toho, aby bolo o tom notifikovaný

- checkid_setup - v tomto režime sa vytvorí komunikácia priamo medzi používateľom a poskytovateľom OpenID

Autentifikácia prebieha tak, že medzi poskytovateľom OpenID a službou, ktorá využíva OpenID, sa vymení zdieľané tajomstvo - kľúč, ktorý poznajú len tieto dve strany. Webová služba môže vyžiadať heslo alebo napríklad smart card.

Používateľ musí potvrdiť, či webovej službe verí a môže spracovať tieto údaje. Ak používateľ zamietne, je automaticky jeho pokus o prihlásenie prehlásený za neplatný. Pokiaľ používateľ potvrdí dôveru, potom služba využívajúca OpenID vyšle požiadavku OpenID poskytovateľovi. Vierohodnosť odpovede od OpenID poskytovateľa je možné overiť z predchádzajúcej komunikácie pomocou overenia zdieľaného tajomstva, ktoré sa ukladá. Tým pádom je možné overiť, či poskytovateľ OpenID je ten, za koho sa vydáva. Po úspešnom overení totožnosti je používateľ na službu využívajúcu OpenID prihlásený s údajmi, ktoré poskytol poskytovateľ OpenID.

Identifikátory

Účty OpenID sú reprezentované internetovou adresou. Pre získanie tejto adresy je potrebné sa zaregistrovať u poskytovateľa OpenID identít, alebo si zriadiť vlastný OpenID server. Tento identifikátor je potom možné použiť na prihlasovanie na internetové služby, ktoré OpenID podporujú.

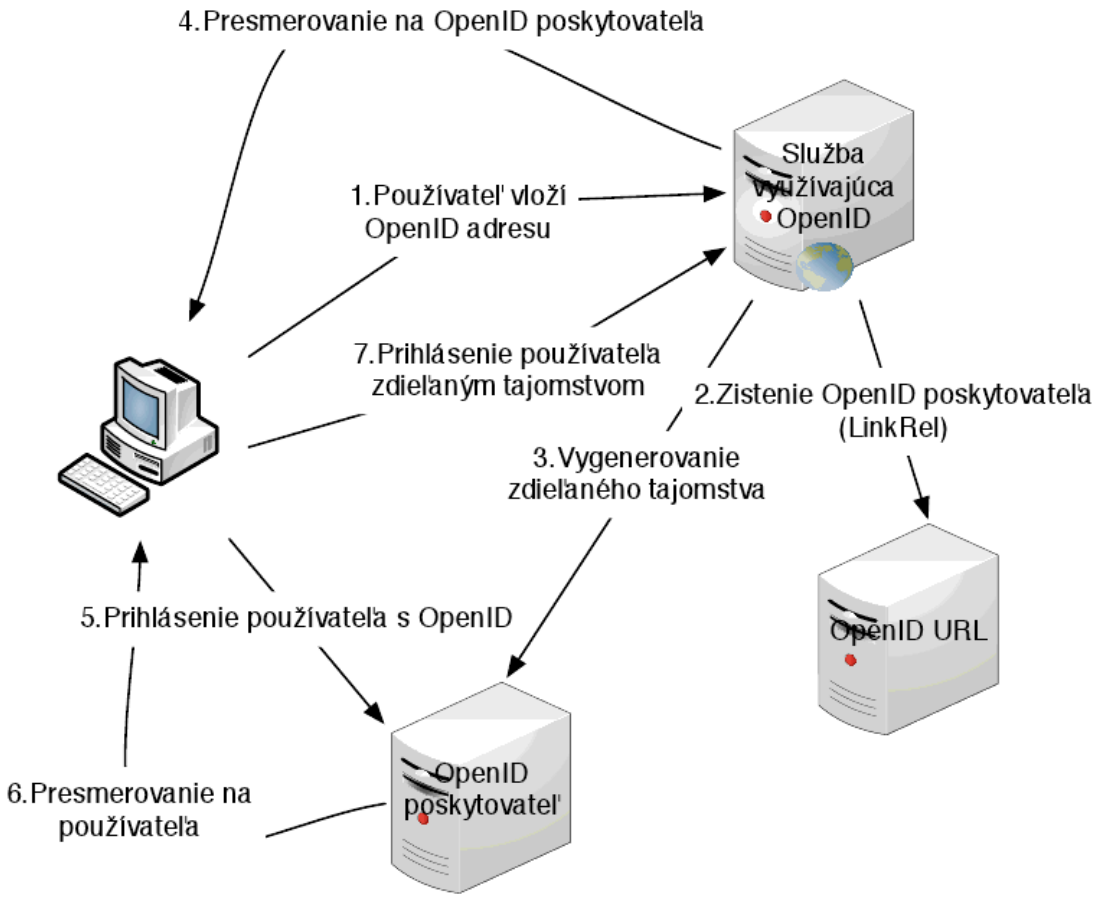
Extensible Resoruce Identifier - XRI

XRI je nová forma internetového identifikátora, ktorá je navrhnutá tak, aby nemohli nastať problémy so zneužitím URL, napríklad pri prebratí DNS záznamu. Funguje to systémom dvoch identifikátorov:

- i-meno - slúži na identifikáciu účtu, má formu čo najľahšie zapamätateľného reťazca, napríklad na identifikáciu osobného účtu =Peter, =Peter.Kovac, alebo na identifikáciu organizácie @FIIT, @team.seven

- i-číslo - tento identifikátor rieši problém perzistentnej identity pri presunoch medzi doménami a aj keď sa zmení doména, stále je identita jasne identifikovateľná pomocou i-čísla.

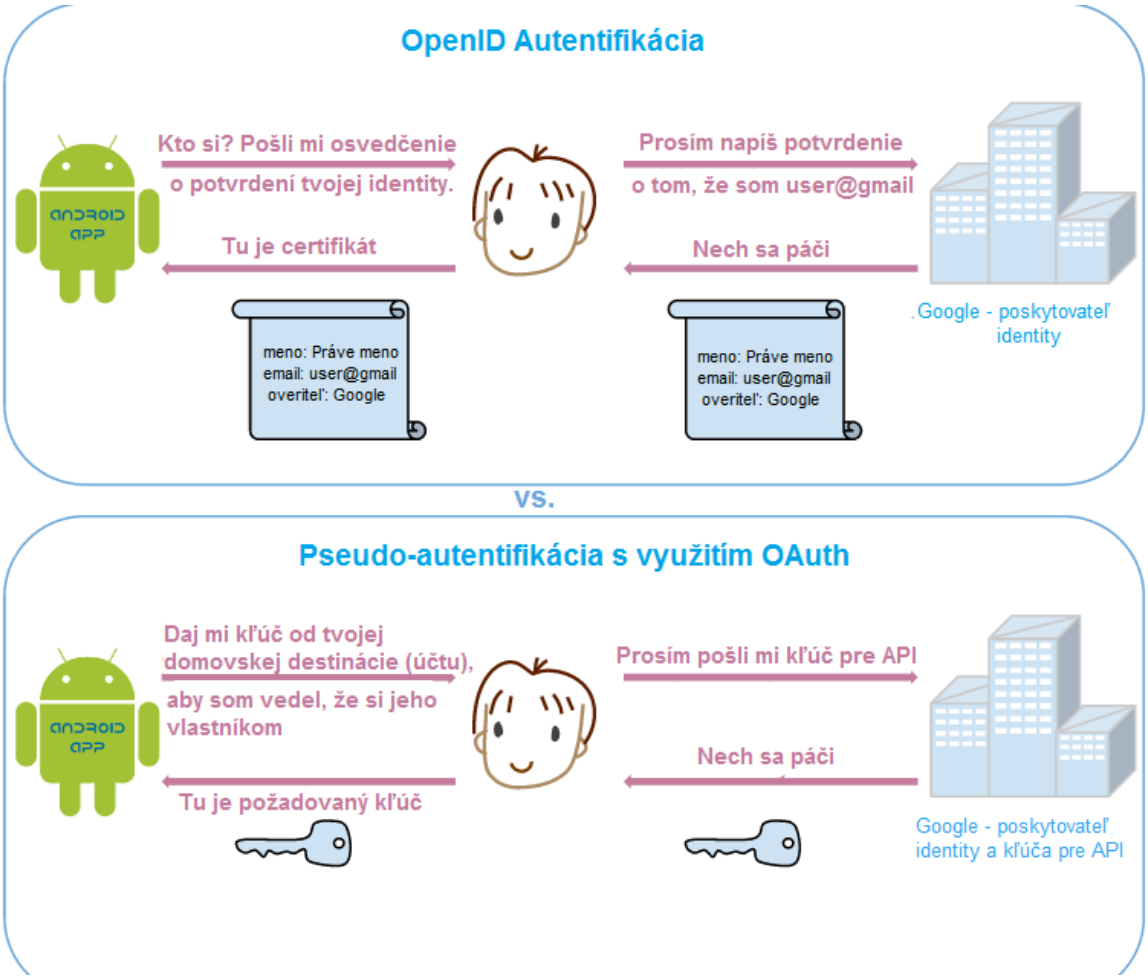
Tieto dva identifikátory sú registrované naraz a ako synonymá.



Obr. č. 2.2 Priebeh komunikácie s OpenID poskytovateľom.

Často je OpenID mylne spájané s technológiou autorizácie OAuth. Ide však o otvorený štandard, ktorý umožňuje používateľom zdieľať svoje osobné zdroje medzi rôznymi stránkami bez toho, aby si vymieňali ich poverenia (credentials). Na miesto toho sa používajú tokeny.

V nasledujúcej schéme (Obr. č. 2.3) [10] je znázornený rozdiel medzi OpenID a OAuth. OpenID proces sa zvyčajne začína tým, že aplikácia požiada používateľa, aby zadal svoju identitu, pričom OAuth aplikácia požiada o token pre prístup ku API. Ak používateľ povolí prístup, tak aplikácia je schopná získať jedinečný kľúč pre vytvorenie profilu, teda identity prostredníctvom API. Pri OpenID je povolený používateľovi vstup na základe overenej identity od OpenID poskytovateľa. V prípade OAuth je udelený prístup používateľovi na základe toho, že API poskytovateľ dôveruje svojim vlastným kľúčom.



Obr. č. 2.3 Porovnanie procesu autentifikácie OpenID s OAuth

2.3.2 HTML 5

V posledných rokoch sa často diskutuje o tom, aká technológia bude nástupcom pri vytváraní web stránok a aplikáciách po zastaranom html. Práve HTML 5 sa ukázala ako potencionálne vhodná technológia pre viaceré vývojárske firmy.

HTML 5 je nová generácia jazyka HTML, ktorá prináša sadu nových elementov, ktoré do veľkej miery zjednodušujú štruktúru stránok. Medzi najvýznamnejšie novinky patrí API, ktoré dovoľuje tvorbu grafiky priamo v prehliadači a tiež nové elementy ako <audio> a <video>, prostredníctvom ktorých je veľmi jednoduché prehrávať multimediálny obsah na strane prehliadača, bez nutnosti inštalácie akéhokoľvek softvéru vyrobeného treťou stranou. Avšak najväčšou nevýhodou danej technológie je paradoxne slabá podpora zo strany viacerých webových prehliadačov. Preto najväčšou výhodou, ale

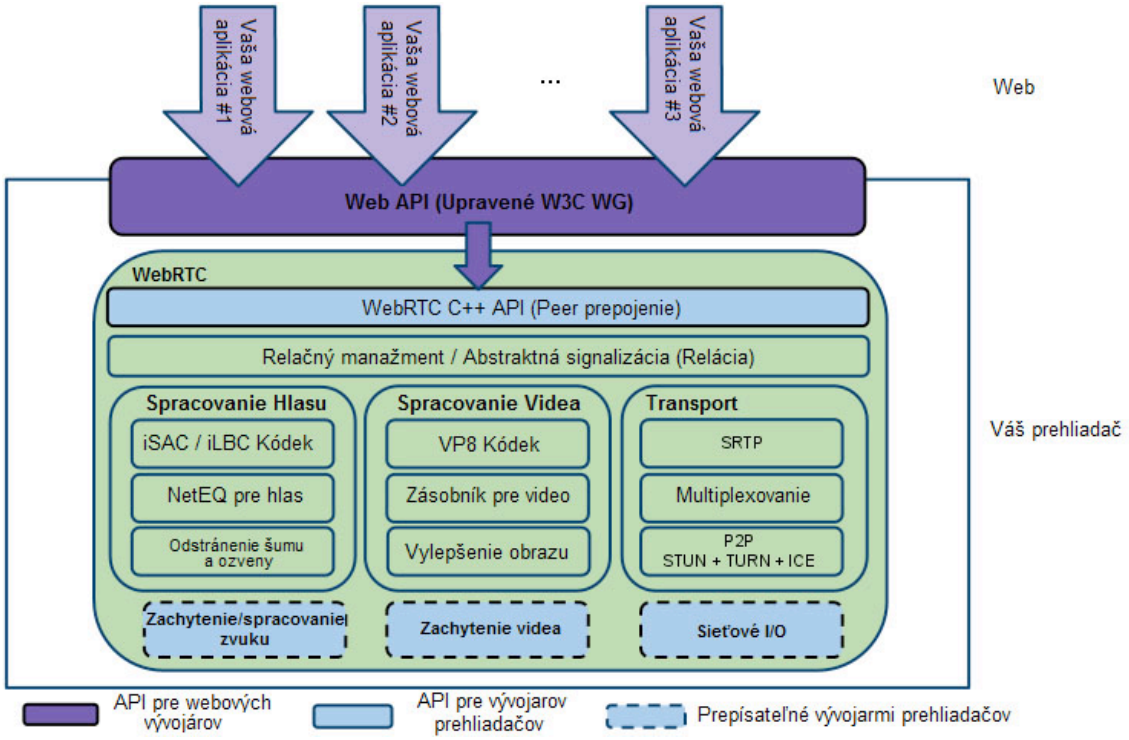
aj nevýhodou, je už skôr spomínaná nezávislosť od externého prehrávača, rozšírenia.

2.3.3 WebRTC

WebRTC je API definované organizáciou W3C a je v procese štandardizácie pracovnou skupinou IETF, čiže je aj veľmi premenlivé. Cieľom WebRTC je umožniť webovým prehliadačom komunikáciu v reálnom čase bez použitia akýchkoľvek rozšírení. V súčasnosti nám WebRTC umožňuje prenášať video telefonovanie a pre prenos zvuku. V pláne je štandardizovať P2P prenos súborov medzi jednotlivými účastníkmi.

WebRTC je podporované takmer každou firmou, ktorá ma čo do činenia s webom. Spoločnosti Google, Microsoft, Opera, Ericsson alebo Mozilla Foundation patria k vývojárom a veľkým podporovateľom tohto štandardu. O tom hovorí aj fakt, že WebRTC získalo veľmi rýchlo na obľube a dostalo sa do každého používanejšieho webového prehliadača. Jediná nevýhoda je to, že niektoré implementácie sú ešte považované za vysoko experimentálne a treba ich ručne povoliť. Toto však už neplatí pre posledné verzie Google Chrome a Opera, kde je podpora WebRTC štandardne zapnutá.

Architektúra WebRTC

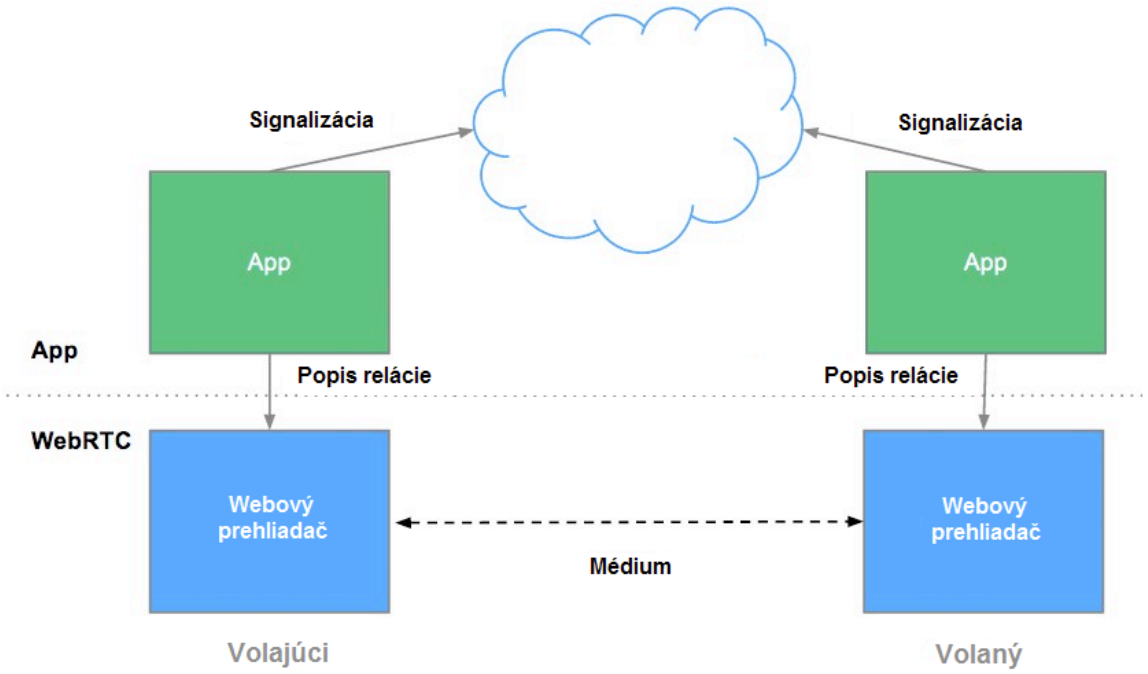


Obr. č. 2.4 Architektúra WebRTC

Na obrázku č. 2.4 [1] môžeme vidieť architektúru WebRTC. Obsahuje dve prístupové vrstvy:

- *WebRTC C++ API*: využívané vývojármi prehliadačov, poskytuje priamy prístup k hlavným funkciám WebRTC.
- *Web API*: napísané pre vývojárov webových aplikácií, v Javascripte. Dá sa povedať, že je to "wrapper" pre C++ API.

WebRTC používa PeerConnection API na transport dát, ale potrebuje aj mechanizmus na zabezpečenie signalizácie. Signalizácia nie je v WebRTC štandarde špecifikovaná, preto sa využívajú iné technológie ako WebSockety alebo XMLHttpRequest (obr. č. 2.5 [2])



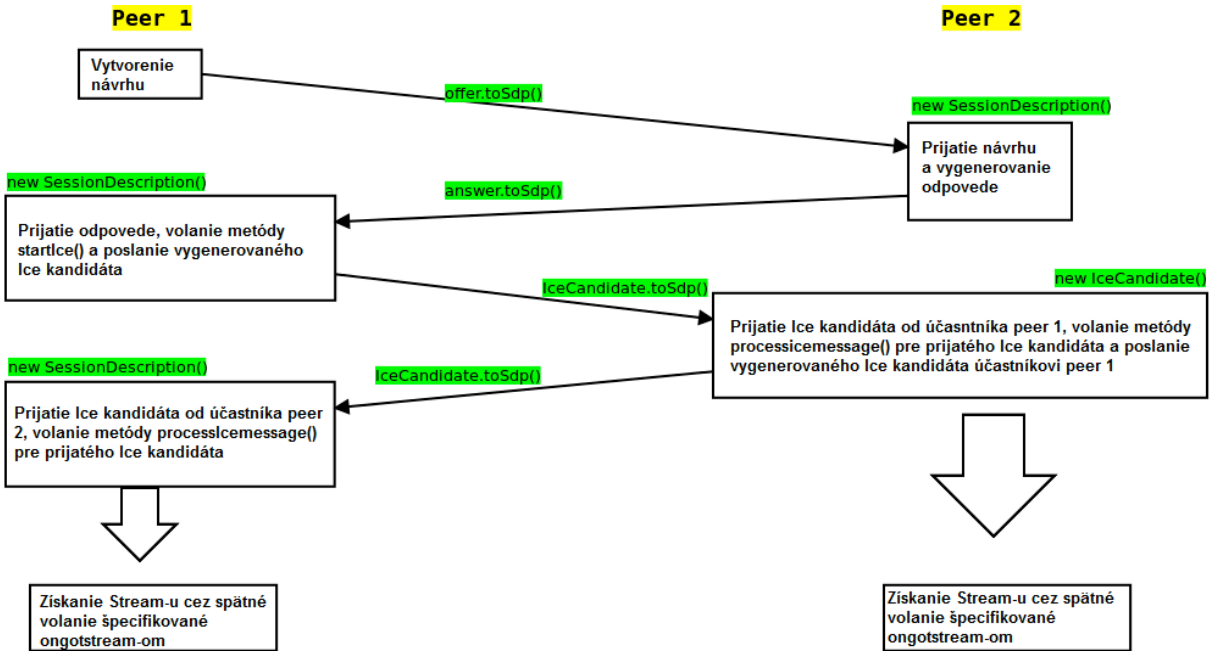
Obr. č. 2.5 Signalizácia a WebRTC

Komponenty WebRTC

WebRTC je zložené z troch základných komponentov:

- *Network Engine*
- *Video Engine*
- *Audio (Voice) Engine*

Hlavnou náplňou práce Network Enginu je sprostredkovanie peer-to-peer spojenia medzi dvoma používateľmi. Na prenos dát sa využívajú komponenty knižnice libjingle, avšak bez použitia xmpp alebo jingle protokolu. Hlavným jadrom je Real Time Protocol (RTP), ktorý poskytuje end-to-end transport dátového toku v reálnom čase, ako aj kompenzáciu jitteru a vysporiadanie sa s poprehadzovanými dátami. Na nasledujúcom obr. č. 2.6 [8] je zobrazené vytvorenie spojenia medzi dvoma účastníkmi (peer 1 a peer 2) s využitím Ice v sieťach s NAT/firewallmi.



Obr. č. 2.6 Vytvorenie spojenia dvoch účastníkov v sieti s NAT/firewallmi

Video Engine je založený na kodeku VP8, predstavenom v roku 2010, ktorý je súčasťou WebM projektu. Zameriava sa na nízku odozvu videa, obsahuje techniky na zakrytie chýb spôsobených stratou paketov, vyhladzovanie, snímanie obrazu a to všetko

na viacerých platformách.

Audio (Voice) Engine poskytuje okrem základných kodekov umožňujúcich hlasovú komunikáciu aj techniky ako sú automatické prispôsobenie hlasitosti, odstraňovanie ozveny, redukcia a potlačovanie šumu a umožňuje aj prístup k hardvéru, taktiež na viacerých platformách. Dostupný je kodek pre úzkopásmovú komunikáciu s dátovým tokom do 15.2 kbps a kodek pre širokopásmovú komunikáciu s premenlivou bitovou rýchlosťou, od 12 do 52 kbps.

Nedostatky WebRTC

WebRTC má aj napriek svojim kladom niekoľko záporov, ktoré nie je možné úplne ignorovať. Prvým nedostatkom je výber video kodeku - VP8 namiesto H.264. Je pravda, že H.264 je obklopená veľkým množstvom patentov, ktoré nie sú zadarmo, kdežto VP8 je "royalty free", t. j. za jeho použitie netreba platiť. Problémom je však hardvérová podpora - keďže VP8 je relatívne nový štandard, nestihol ešte získať hardvérovú akceleráciu, čo značí menšie rozlíšenie ako H.264, čo značne vplýva na celkový "user experience".

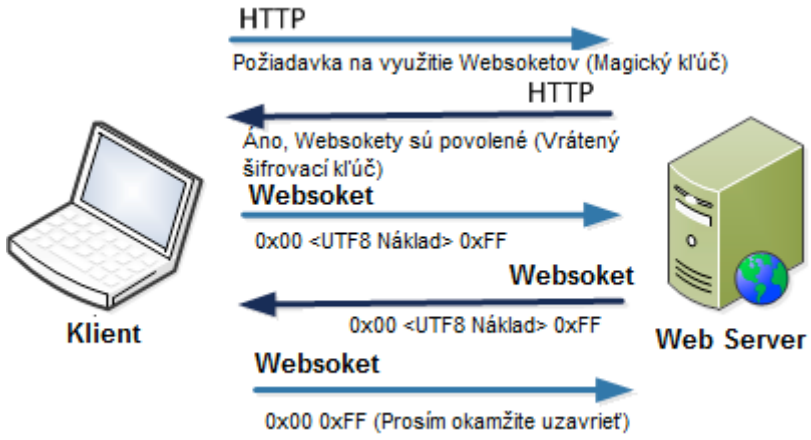
Ďalším takýmto problémom je absencia na mobilných zariadeniach. Google sa rozhodol zamerať najprv na klasické počítače pred tým, než začne uvažovať nad Android verziou. Toto predstavuje problém hlavne na smartfónoch a tabletoch, kde táto technológia chýba. Navyše používatelia platformy Apple budú tiež ochudobnení, pretože Apple nemá veľký záujem o túto technológiu.

Aj napriek všetkým týmto nedostatkom vyzerá WebRTC ako technológia, ktorá je krokom vpred a ktorá istotne nájde v dnešnej dobe uplatnenie.

2.3.4 WebSocket

WebSocket je nová webová technológia, ktorá si za svoj cieľ kladie umožnenie obojsmernej komunikácie nad jedným TCP spojením. WebSocket je štandardizovaný ako RFC 6455.

WebSockety sú navrhnuté, aby ich podporu implementovali webové prehliadače a webové servery, ale sú použiteľné aj na klient-server komunikáciu. WebSocket ako protokol je nezávislý TCP protokol. S HTTP má spoločné iba to, že si klient a server dohodnú použitie WebSocketov. Celá nasledujúca komunikácia prebieha cez port 80 (pokiaľ sa nedohodlo ináč), čo môže byť výhodou v sieťach, kde sa blokuje komunikácia firewallmi.



Obr. č. 2.7 Komunikácia cez WebSockety

Na obr. č. 2.7 [3] je znázornený prechod z obyčajného HTTP spojenia na WebSockety. Najprv pošle iniciátor správu s tzv. *Magickým kľúčom (Magic Key)*, čo je reťazec zakódovaný v base64. Následne pošle druhý účastník odpoveď, ktorá pozostáva z pôvodného reťazca, ku ktorému sa pridá *magický reťazec* 258EAFA5-E914-47DA-95CA-C5AB0DC85B11, ktorý sa potom zašifruje algoritmom SHA-1 a zakóduje sa do base64 a odošle sa iniciátorovi. Následne sa dáta vymieňajú už bez HTTP hlavičiek, kým sa spojenie neuzavrie.

Ako je to s každou novou technológiou, aj WebSockety majú svoje nedostatky. Problémom sú niekedy proxy servery, ktoré sú realizované na báze HTTP, pretože po dohodnutí spojenia sa spojenie razom stáva už len TCP spojením, čo sa niektorým proxy serverom nepáči, a tak zrušia spojenie. Podobne je to pri niektorých antivírusoch a firewalloch, avšak snáď s rastúcou popularitou budú tieto prekážky prekonané.

WebSockety sa ihneď od svojho predstavenia stali populárnou technológiou, o čom značí aj fakt, že sú dnes implementované v každom populárnejšom jazyku a prehliadači (okrem Internet Explorera).

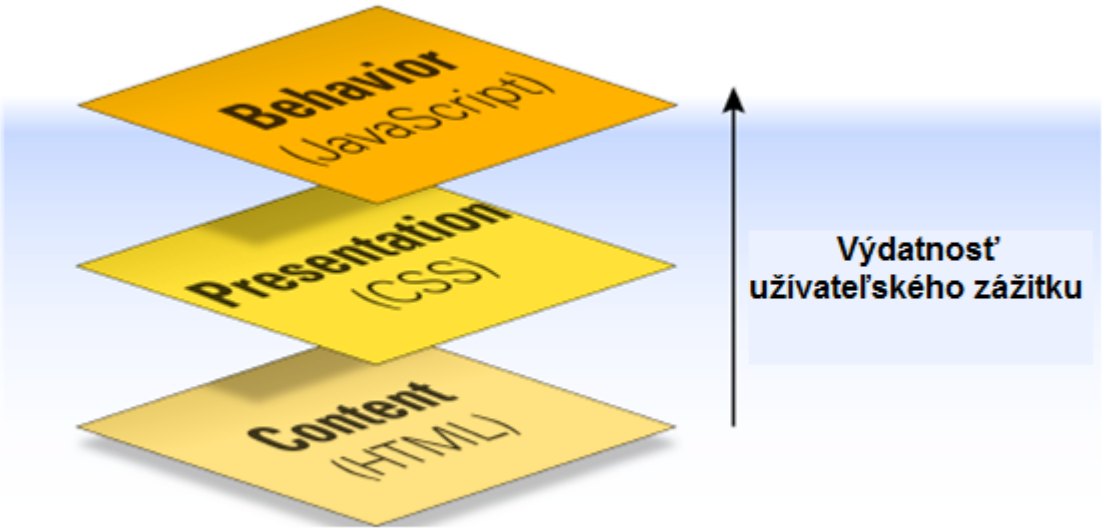
2.3.5 Client-Side JavaScript

Prostredníctvom webového prehliadača je možné pristupovať ku rôznym webovým stránkam a prezerať si ich obsah. Tento proces prebieha tak, že klient pošle žiadosť na server, na ktorom sa nachádza požadovaná stránka. Server vyhodnotí požiadavku klienta a sprístupní mu žiadanú stránku tým, že mu ju odošle na jeho

zariadenie. Následne webový prehliadač načíta získanú webovú stránku a zobrazí používateľovi jej obsah. Typicky je tento obsah tvorený troma základnými technológiami: HTML, CSS a JavaScript.

Trojvrstvový model štruktúry webovej stránky

Na vytvorenie webovej stránky sa teda používajú HTML (HyperText markup language), CSS (Cascading Style Sheets) a JavaScript. Tieto tri prvky tvoria základný trojvrstvový model štruktúry webovej stránky (obrázok č. 2.8) [9].



Obr. č. 2.8 Trojvrstvový model štruktúry webovej stránky

Pri vytváraní webovej stránky je dôležité dodržať rozdelenie jednotlivých vrstiev. To znamená, že HTML, CSS a JavaScript sa uložia do troch rôznych súborov. Aplikácia tohto modelu prináša veľké množstvo výhod, ako je napríklad možnosť zdieľať externé súbory CCS a JavaScript medzi rôznymi stránkami. Takáto štruktúra poskytuje aj spätnú kompatibilitu, čo znamená, že prehliadače, ktoré nie sú schopné použiť technológiu CSS a JavaScript, tak zobrazia aspoň jej základný obsah a rozloženie.

- *Štruktúrna vrstva (obsahová)* – je to vrstva, na ktorej je uložený celý obsah žiadaný klientom. Patrí sem text, obrázky a multimediálny obsah. Na vytvorenie požadovanej štruktúry obsahu webového dokumentu sa používa HTML alebo XHTML.

- *Prezentačná vrstva* (*výzorová*)– alebo vrstva štýlu určuje vzhľad dokumentu alebo

ako bude stránka vyzerať pri prezeraní čitateľom. Rôzne formy zobrazenia zabezpečuje CSS. Jeho úlohou je taktiež rozdeliť štýly pre zobrazenie obsahu na rôznych typoch médií (na monitore, na zariadení podporujúce Braillovo písmo, ...).

- *Funkčná vrstva (správania)* – je vrstva, ktorá rozširuje vytvorenú webovú stránku o interaktivitu a dynamické správanie. Na tento účel sa najčastejšie používa JavaScript.

JavaScript

JavaScript je skriptovací programovací jazyk s objektovo orientovanými vlastnosťami, ktorý je implementovaný vo webových prehliadačoch na klientovej strane (client-side JavaScript) a umožňuje vytvoriť rozšírené používateľské rozhranie a dynamické webové stránky. Formalizovaný bol v jazykovom štandarde ECMAScript. Jadro jazyku JavaScript je syntaxou podobné C, C++ a jazyku Java. Obsahuje elementy ako podmienkové bloky (if), cykly s podmienkou (while) alebo operátor &&. Táto podobnosť sa vyskytuje len v rámci syntaxe. JavaScript je jazyk s voľným typom, čo znamená, že pri definovaní premenných sa neuvádza žiadny typ. Kľúčové prvky dizajnu prebral JavaScript od programovacích jazykov Self a Scheme.

JavaScript nie je Java

Originálne bol jazyk JavaScript vyvinutý v Netscape Brendanom Eichom. Na začiatku bol známy pod názvami Mocha a LiveScript a v roku 1995 vyšiel v beta verziách Netscape Navigator 2.0. Ku premenovaniu na JavaScript došlo v roku 1995 na základe marketingovej spolupráce firmy Sun a Netscape. Avšak formulácia takéhoto názvu je veľmi zavádzajúca, keďže veľké množstvo ľudí si myslí, že JavaScript je akási „light“ verzia programovacieho jazyka Java, ktorý vyvinula firma Sun Microsystems. Tieto dva jazyky pritom majú iba čiastočne spoločný názov. V prípade JavaScriptu sa jedná o skriptovací jazyk podporovaný webovými prehliadačmi a Java je programovací jazyk.

JavaScript knižnice

Postupným rozširovaním jazyku JavaScript sa začali zvyšovať aj nároky jednotlivých používateľov a vývojárov. Začali sa hľadať nové spôsoby, ako by bolo možné jednoducho vytvoriť dynamické rozhranie podľa programátorových požiadaviek. Inšpiráciou čerpanou z iných programovacích jazykov začali postupne vznikať knižnice. Medzi najznámejšie JavaScript knižnice alebo farmeworky patria: Prototype, jsPHP,

MooTools a jQuery. Niektoré JavaScript knižnice umožňujú jednoduchšiu integráciu JavaScriptu s inými webovými technológiami ako CSS, PHP, Ruby a Java.

- **Prototype** – je to JavaScript konštrukcia, ktorá poskytuje množstvo funkcií pre tvorbu JavaScript aplikácií. Obsahuje taktiež väčšinu funkcií pre prácu s XMLHttpRequest.

- **jsPHP** – knižnica, ktorá sprístupňuje PHP API v prostredí JavaScriptu. Obsahuje funkcie pre polia, výpočty, manipuláciu objektov/tried, prácu s chybovými výstupmi a reťazcami.

- **MooTools (My Object-Oriented Tools)** – objektovo orientovaná JavaScript konštrukcia (Ajax štruktúra).

- **jQuery** – JavaScript knižnica vytvorená pre zjednodušenie skriptovania na strane klienta. Obsahuje množstvo preddefinovaných funkcií pre prácu s objektmi, vytváranie animácií, spracovanie udalostí a vývoj Ajax aplikácií.

AJAX

Potreba vytvárania interaktívnych stránok viedla ku vytvoreniu nových prístupov pri budovaní webových stránok. Ajax (Asynchronous JavaScript + XML) predstavuje súhrnné označenie pre technológie vývoja interaktívnych webových aplikácií, ktoré umožňujú meniť obsah stránok bez potreby ich opätovného kompletného načítania zo serveru. Príkladmi takýchto aplikácií sú on-line chat, textový procesor, editor obrázkov alebo vizuálny editor. Ajax nie je nová technológia, ale kombinuje už existujúce technológie pre tvorby stránok.

- Pre úpravu vzhľadu stránky používa HTML a CSS

- DOM (Document Object Model) spolu s JavaScriptom pre dynamické zobrazenie a interakciu s prezentovanou informáciou. Prostredníctvom objektovo orientovanej reprezentácie dokumentu je totiž možné objekty pridávať, mazať, editovať ich obsah alebo vzhľad. Základom teda je, že pri spracovaní dokumentu HTML stránky webovým prehliadačom sa vytvorí stromová štruktúra, v ktorej sú jednotlivé uzly hierarchicky usporiadané.

- Metódy pre komunikáciu medzi webovým prehliadačom a serverom. Dáta sú vymieňané bez nutnosti neustáleho obnovovania stránky. Často sa používa

XMLHttpRequest.

- Pre výmenu dát sa používajú formáty najmä XML, JSON (JavaScript Object Notation), HTML alebo sú dáta posielané ako jednoduchý text.

JavaScript a bezpečnostná politika

Bezpečnostný model implementovaný v jazyku JavaScript je veľmi podobný modelu použitému pri Java apletoch. Aplety predstavovali veľké potencionálne nebezpečenstvo, keďže umožňovali spustenie Java kódu na klientovom zariadení. Takéto aplikácie mohli mať rovnaké prístupové práva do systému ako mali lokálne programy. Preto bol vytvorený bezpečnostný mechanizmus sandbox (pieskovisko). Tento mechanizmus vytvára bezpečnostnú zónu, v ktorej môže aplet operovať. JavaScript pracuje na rovnakej logike. Neumožňuje čítanie ani zápis do lokálnych súborov zariadenia okrem špeciálnych súborov zvaných cookies. Dokonca JavaScript implementoval prísnejšiu politiku ako v prípade apletu. Dôvod je ten, že JavaScript pracuje v prostredí prehliadača a je nutné zaistiť bezpečnosť osobných dát. Preto napríklad JavaScript, ktorý beží v jednom otvorenom okne prehliadača nemôže monitorovať informácie vrátane URL, ktoré sú prijímané v druhom otvorenom okne prehliadača. Bežnou praxou je taktiež dodržiavanie separácie JavaScriptu a ostatného HTML kódu.

2.3.6 Node.js (Server- Side JavaScript)

Node.js je asynchrónny, udalosťami riadený programovací jazyk, ktorý uzrel svetlo sveta v roku 2009. Jeho autorom je Ryan Dahl, zamestnaný spoločnosťou Joyent, ktorá financuje jeho rozvoj a vo veľkom používa vo svojich produktov napr. Cloud Analytics alebo Joyent Public Cloud. Primárne sa používa na vytváranie škálovateľných a reálno-časových sieťových aplikácií, najmä aplikačných webových serverov.

Node.js tvorí niekoľko častí:

- **V8 JavaScriptový engine od Google** - používa sa vo webovom prehliadači Google Chrome. V8 patrí k jednému z najrýchlejších javascriptových enginov vôbec. Svoju rýchlosť získava najmä tým, že javascriptový kód kompiluje priamo do strojového kódu namiesto toho, aby ho kompiloval do bytecode alebo ho interpretoval. Taktiež používa niekoľko výkonnostných optimalizácií, ktoré ho robia rýchlejším.

- **libUV** - je platformová vrstva, ktorej cieľom je zjednotiť IOCP rozhranie na Windowse a libev na unixových operačných systémoch. Všetky platformové závislé časti kódu potrebné pre beh Node.js sú obsiahnuté v tejto knižnici.

- **niekoľko vstavaných knižníc**

Node.js ako svoju syntax používa JavaScript. Ryan si JavaScript vybral, pretože ešte nebolo zadefinované žiadne I/O rozhranie. Toto mu dovolilo vytvoriť neblokujúce, udalosťami riadené I/O rozhranie. Na rozdiel od klasického JavaScriptu, ktorý beží vo webovom prehliadači, Node.js beží na serveri a preto ho mnohí nazývajú aj serverový JavaScript.

Vďaka tomu, že Node.js používa JavaScript, je práca s ním veľmi jednoduchá. Skutočne každý tvorca webových aplikácií môže v Node.js robiť bez väčších problémov. Jediná nevýhoda je, že pochopenie udalosťami riadeného systému môže chvíľu trvať. Ďalšou veľkou výhodou Node.js je jednoduchosť inštalácie modulov cez npm a ich veľké množstvo. Ďalším faktorom, ktorý hovorí o Node.js, je rýchlo rastúca komunita okolo neho. Za 3 roky svojej existencie získal Node.js veľmi veľkú popularitu medzi tvorcami webových aplikácií a webových spoločností ako LinkedIn, Microsoft, Joyent, Yahoo! alebo Walmart a jeho obľuba rastie. Node.js bol označený za budúcnosť webových aplikácií najmä vďaka jeho vlastnostiam.

V rámci nášho projektu budeme používať Node.js na programovanie portálu, ktorý bude overovať používateľov pomocou ich OpenID konta a prehrávať multimediálny obsah pomocou WebRTC. Node.js sme si vybrali pre to, že v rámci tímu s ním máme skúsenosti, jednoduchosť a množstvo dostupných knižníc, ktoré nám pomôžu portál implementovať.

PassportJS

PassportJS je autentifikačná konštrukcia pre Node.js. Jediným cieľom PassportJS je autentifikácia požiadaviek. PassportJS využíva moderné autentifikačné prostriedky, ktoré umožňujú jednoduchú implementáciu do hotových aplikácií a umožniť tak, využívanie existujúcich účtov iných služieb umožňujúcich túto funkcionalitu, alebo napríklad OpenID. Passport využíva štruktúru Connect.

Express

Express je webová konštrukcia postavená na Node.js. Express je veľmi podobný

webovej konštrukcii Sinatra (Ruby), ktorým boli jeho vývojári inšpirovaní. Express uľahčuje prácu pri vytváraní webových aplikácií. K dispozícií je dokumentácia na oficiálnej stránke projektu a je veľmi prehľadná. V rámci našej témy budeme Express používať na programovanie frontendu pre portál.

2.3.7 CouchDB

CouchDB je databázový systém, ktorý ukladá všetky údaje v JSON formáte, a celá komunikácia s databázou prebieha pomocou HTTP požiadaviek. CouchDB umožňuje master-master replikáciu, s automatickou detekciou kolízií. CouchDB má priamu podporu Node.js, preto je vhodnou službou na ukladanie perzistentných údajov.

2.4 Zhodnotenie analýzy

Naša analýza sa skladá z troch hlavných častí, z ktorých nadobudnuté informácie poslúžili na návrh architektúry nášho systému.

Prvá časť spočívala v analýze celkovej problematiky a tiež oboznámenie sa s danou témou. Boli analyzované dve hlavné prvky nášho projektu, a to OpenID a WebRTC. V tejto časti sme získali potrebné znalosti o týchto technológiách, ktoré nám pomôžu pri návrhu a implementácii našej architektúry.

V druhej časti analýzy sme sa zaoberali podobnými existujúcimi riešeniami. Keďže WebRTC ešte nie je veľmi rozšírenou technológiou a okrem video rozhovorov neexistuje veľa podobných implementácií, tak sme sa zamerali hlavne na služby, ktoré poskytujú multimediálny obsah. Práve analýzou týchto služieb sme dospeli k záveru, že implementácia WebRTC spolu s týmito službami, môže byť do určitej miery "novinkou na trhu". Keďže momentálne ešte neexistujú podobne riešenia a WebRTC ešte nie je plne podporovaný všetkými prehliadačmi, tak implementácia tejto architektúry môže byť o to zložitejšia.

V poslednej časti analýzy sme sa zamerali na technológie, pomocou ktorých je možné danú architektúru implementovať. Dané technológie sme pozorne analyzovali, aby sme v neskorších fázach vedeli naplno využiť ich potenciál pri tomto projekte.

Z našej analýzy sme dospeli k záveru, že daný systém sa dá pomocou daných nástrojov implementovať a jeho finálny výstup môže rozšíriť ponúkane služby na Internete, kde by mal vysoký potenciál na úspech.

3 Špecifikácia riešenia

Táto kapitola obsahuje špecifikáciu navrhovaného riešenia.

3.1 Funkcionálne požiadavky

Cieľom tímového projektu bude vytvoriť jednoduchý webový portál, ktorý bude slúžiť na prezeranie multimediálneho (video) obsahu. Používateľ bude schopný sa jednoducho prihasiť na daný webový portál a následne si vybrať obsah, ktorý chce zobraziť. Tento obsah by mu mal byť v určitých intervaloch fakturovaný.

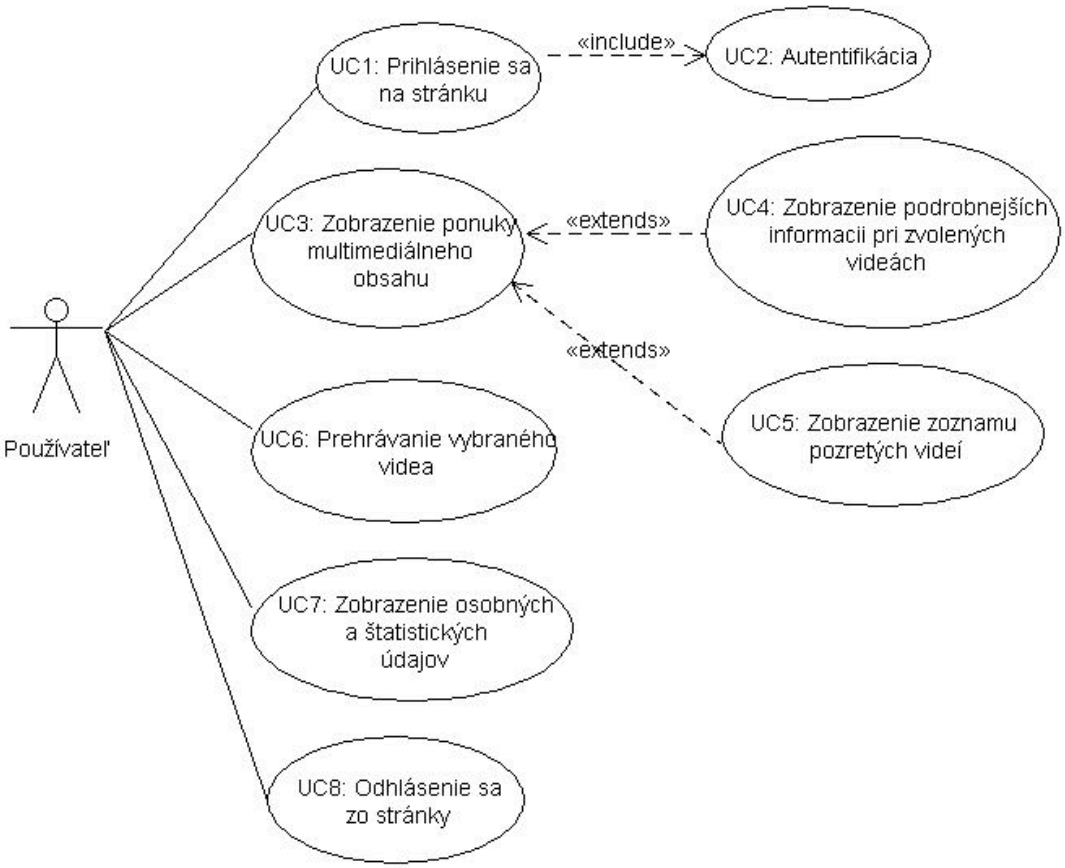
Prostredie bude realizované ako webová aplikácia. Používateľ si bude môcť vybrať ľubovoľný multimediálny obsah, ktorý chce v tej chvíli na svojej strane zobraziť.

Pre prehrávanie obsahu na danom portáli bude musieť byť v čase prehrávania aj korektne prihlásený. Prihlasovanie bude možné tiež pomocou technológie OpenID, pri ktorej sa používateľ nemusí na danom portáli registrovať, čo ušetrí jeho čas a tiež prípadne nároky na systém. V prípade, že používateľ nemá vlastné OpenID konto, bude mu ponúknutá možnosť registrácie priamo na portáli.

Po prehraní vybraného obsahu bude používateľ upozornený na počet hodín, ktoré má zaplatiť za vybrané časove obdobie. Tiež bude možne aby si pozrel históriu videní jednotlivého multimediálneho obsahu. Používateľ nebude priamo platiť portálu a faktúra za poskytnuté služby bude vystavená jeho poskytovateľom internetu.

3.2 Prípady použitia

Na obr. č. 3.1 sú zobrazené prípady použitia navrhovaného systému. Vo vytváranom systéme vystupuje iba jediný aktér a to samotný používateľ, ktorý využíva funkcie systému.



Obr. č. 3.1 Diagram prípadov použitia

3.3 Nefunkcionálne požiadavky

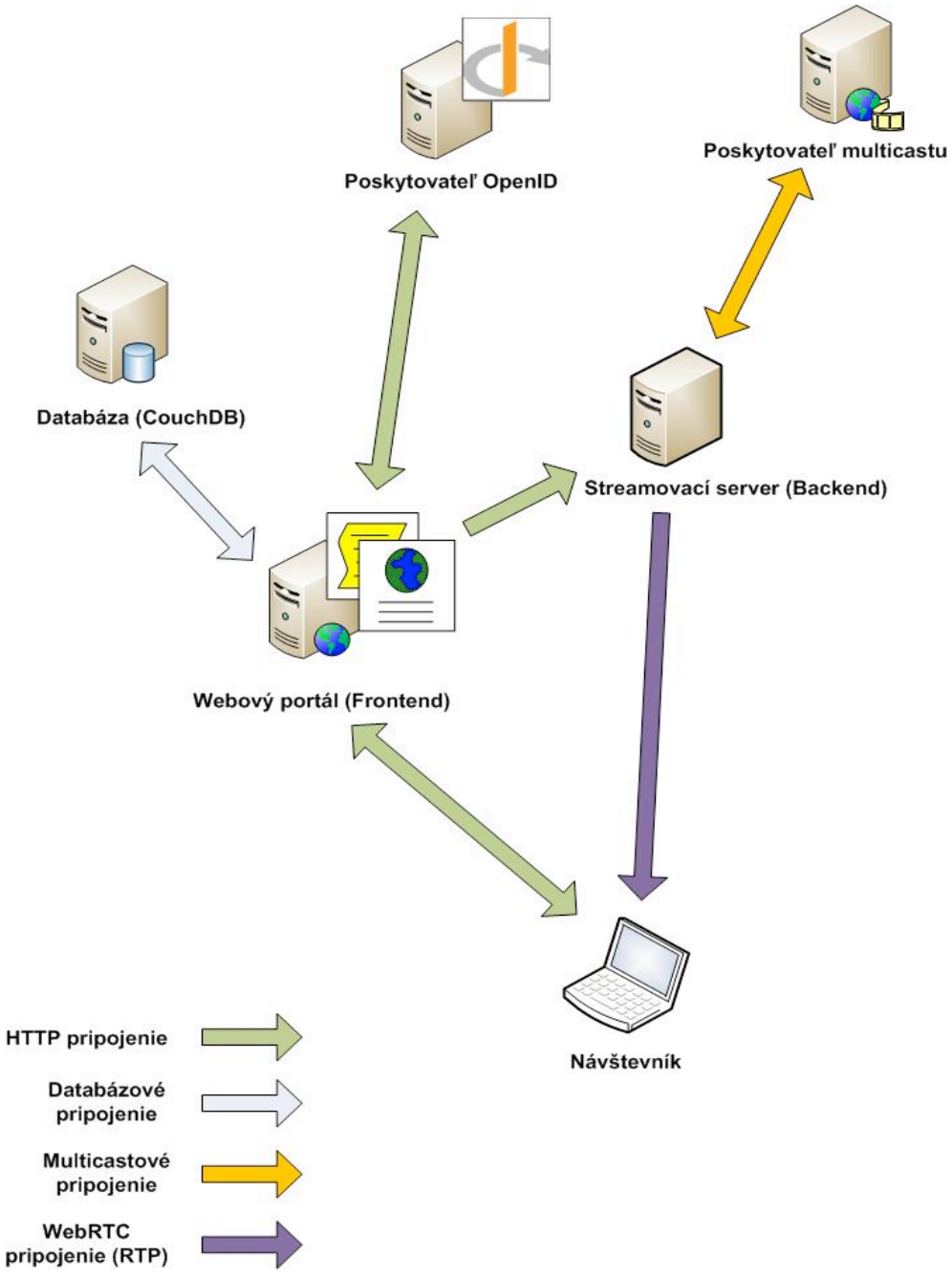
Na vyvíjaný systém sú kladené nasledovné nefunkcionálne požiadavky:

- Intuitívna práca s webovým portálom
- Prehľadné používateľské prostredie

Hlavnou úlohou je vytvoriť webový portál, v ktorom sa bude vedieť používateľ jednoducho orientovať a intuitívne ho bude vedieť ovládať.

4 Návrh architektúry

Architektúra systému sa skladá z 3 hlavných častí a celkový náhľad na návrh je možné vidieť na obrázku č. 4.1.



Obr. č. 4.1 Návrh celkovej architektúry systému

- webový portál
- databáza
- multimediálny portál

4.1 Webový portál

Návštevník príde na webový portál a môže sa prihlásiť. Klient sa prihlási svojím OpenID kontom, ktoré sa následne autentifikuje u poskytovateľa zdieľanej identity OpenID. Ak autentifikácia prebehla úspešne, tak je používateľ prihlásený na obsah webového portálu. Po úspešnej autentifikácií sa vygeneruje náhodný identifikátor, ktorý identifikuje webové sedenie klienta. Tento identifikátor sa zapíše do databázy a je tam pokiaľ je klient prihlásený.

Po overení svojej identity sa klientovi zobrazí aktuálna dostupná ponuka multimediálneho obsahu. Klient si po kliknutí na video môže zobraziť náhľad a ďalšie dodatočné informácie o videu. Metadáta k videu sú uložene v databáze. Potom, ako sa klient rozhodol, že si nechá obsah zobraziť, sa do databázy zapíše informácia o tom, že ktoré video si vybral. Táto zaznamenaná skutočnosť bude použitá pri zaslanom výpise o pozretom multimediálnom obsahu, ktorý klient dostane spolu s vyúčtovaním za poskytnutie tohto obsahu od svojho poskytovateľa internetového pripojenia. Tieto informácie sa budú odosielať raz mesačne poskytovateľom internetového pripojenia.

Po odkliknutí na multimediálny obsah bude klient presmerovaný HTTP odpoveďou na multimediálny portál, kde sa vygeneruje unikátna URL, na ktorej nájde to, čo sa rozhodol si pozrieť. URL je platná len pre klienta, ktorý má identifikátor webového sedenia. URL bude mať časovo obmedzenú platnosť niekoľko hodín, po ktorej vypršaní už nebude aktívna. Portál začne tlačiť multimediálny obsah pomocou technológie WebRTC. Ako transportný protokol použijeme WebSockety, pretože signalizácia nie je štandardizovaná v rámci projektu WebRTC.

4.1.1 Autentifikácia

Autentifikácia môže prebiehať dvoma spôsobmi:

- Lokálna databáza prihlasovacích údajov - toto by vyžadovalo držanie vlastnej databázy používateľov, a s tým spojená potrebná réžia na údržbu a zálohovanie a navyše v zadaní bolo určené, že na autentifikáciu je potrebné použiť OpenID

- Využitie OpenID - tento spôsob bude využitý v našom projekte, keďže tento spôsob nevyžaduje uchovávanie dát o používateľských prihlasovacích údajov, lebo autentifikácia vždy prebieha cez domovský server, ktorý OpenID konto poskytuje a spravuje. Tento štandard v budúcnosti môže nahradiť všetky registrácie jediným prihlasovacím údajom a to OpenID identitou

Autentifikácia s využitím OpenID

Pri použití OpenID je potrebné len uchovávať lokálnu databázu, ktorá je potrebná na funkcionalitu a autorizáciu a nie je potrebné riešiť lokálne registrácie a s tým spojené činnosti. Toto bude riešené na strane servera, ktorý bude dostávať požiadavky na autentifikáciu a ten to bude potom preposielať poskytovateľovi OpenID identity, ten následne bude odpovedať, či používateľ je oprávnený sa prihlásiť pod poskytnutými prihlasovacími údajmi.

Následne, ak neexistuje lokálny profil pre daného používateľa, je mu vytvorený. Je mu vygenerovaný unikátny lokálny identifikátor, ktorý je viazaný na jeho OpenID profil. Na základe toho sa ďalej rieši autorizácia a účtovníctvo za poskytnuté služby.

Implementácia OpenID na webovom portáli

Prihlasovanie bude prebiehať na webovom portáli pomocou OpenID. Toto bude zabezpečovať framework pre Node.js, Passport. Bude tam vstupné pole, ktoré bude brané ako vstup pre OpenID identitu. Po úspešnom prihlásení je následne možné pristupovať k obsahu, čo bude zabezpečené pomocou cookies. Webový portál bude taktiež komunikovať s lokálnou databázou, ktorá uchováva informácie, ktoré sú potrebné na korektné fungovanie tejto služby.

4.2 Databáza

Uchovávanie perzistentných dát a profilov

Dáta potrebné na správnu funkcionalitu budú uchovávané pomocou CouchDB, ktorá je potrebná na uchovávanie transakcií, činnosti používateľov a všeobecnú prácu s účtami. Napriek tomu, že nie je potreba uchovávať prihlasovacie mená a heslá, je potrebné držať lokálny profil používateľa, aby bolo možné určiť, čo môže a čo nemôže používateľ spraviť - autorizácia. Databáza bude vždy zálohovaná a bude to riešené pomocou master-master replikácie, na ktorú je CouchDB stavané.

Budú potrebné minimálne dve databázy, ktoré budú mať rôzne účely:

Databáza identít - táto databáza bude uchovávať prepojenia medzi OpenID identitami a lokálnymi profilmi. Toto bude slúžiť aj na autorizáciu a následné prepojenie k účtovníctvu a fakturácii.

Databáza, ktorá bude uchovávať účtovníctvo, platby a s tým spojené údaje - tieto údaje búdu spracovávané oddelene a tak, aby bolo možné komunikovať s lokálnou databázou identít, alebo, aby vedela spolupracovať s externými inštitúciami, ktoré takúto komunikáciu podporujú a zároveň je možné využiť to na účtovníctvo a fakturáciu.

4.3 Multimediálny portál

Multimediálny portál sme identifikovali ako jeden z komponentov nami navrhovanej architektúry. Portál by mal byť schopný udržať vysielanie obsahu pre stovky klientov naraz.

Vysielanie obsahu

Pri návrhu multimediálneho portálu sme dbali na to, aby sa ako zdroj multimediálneho obsahu dalo použiť čo najviac možností: súbory na disku až po multicastové vysielanie. V našej implementácií uvažujeme nad oboma možnosťami.

Implementácia WebRTC v multimediálnom portáli

Portál budeme implementovať v Node.js a na implementáciu použijeme jeden z dostupných modulov, ktorý podporu pre WebRTC už implementuje, napr. WebRTC.io. Ako transportný protokol sme sa rozhodli použiť WebSockety pre vlastnosti, ktoré nám poskytuje (najmä multiplexovanie dát cez jedno TCP spojenie), čo pomôže so zvládnutím udržania veľkého počtu klientov.

Generovanie URL

Portál musí vedieť vygenerovať jedinečnú URL, na ktorej bude dostupný multimediálny obsah. Generovanie by malo byť dostupné cez minimalistické REST API, ktoré bude danú funkcionalitu implementovať. Webový portál bude posielať požiadavky vo formáte JSON, natívnom formáte pre Node.js.

5 Prototyp

Táto časť projektu sa venuje vytvorenému prototypu nášho systému. Je rozdelená na dve podkapitoly. V prvej podkapitole sú stanovené a bližšie popísané časti systému, ktoré budú vytvárané v rámci prototypu. Druhá podkapitola obsahuje dosiahnuté výsledky a bude hlavne obsahovať výslednú funkcionalitu prototypu, ktorú sa nám podarilo implementovať.

5.1 Ciele prototypovania

Cieľom prototypu bolo vytvorenie prvotného modelu systému a ukážka funkcionality vybraných častí navrhovaného systému. Dôležité bolo najmä identifikovanie prvkov, ktoré budú implementované v rámci prototypu. Išlo hlavne o to, aby sa dokázala správnosť navrhovaného riešenia a teda oddelenie úloh, ktoré len predstavujú implementačný problém od metód a myšlienok, ktoré je nutné overiť pred implementovaním celého systému, takzvaný „proof of concept“. Na finálnej podobe prototypu sa postupne dohodol náš tím v rámci tímových stretnutí spolu s vedúcim tímu. Hlavnou požiadavkou na prototyp bolo vytvorenie jednoduchého webového portálu, na ktorom by bolo možné prihlásenie sa prostredníctvom OpenID a následné vytvorenie spojenia medzi serverom a klientovým webovým prehliadačom prostredníctvom štandardu WebRTC. Ďalšou požiadavkou bolo zhotovenie prototypu v takej podobe, aby ho bolo možné prezentovať oponujúcemu tímu.

Keďže štandard WebRTC bol navrhnutý pre komunikáciu medzi dvoma webovými prehliadačmi klientov, tak zatiaľ nebolo implementované riešenie pre komunikáciu medzi serverom a webovým prehliadačom klienta. V prvom rade sa náš tím zameral na vytvorenie takzvanej „serverovej časti“, teda implementovanie mechanizmov štandardu WebRTC na serveri.

5.2 Dosiahnuté výsledky

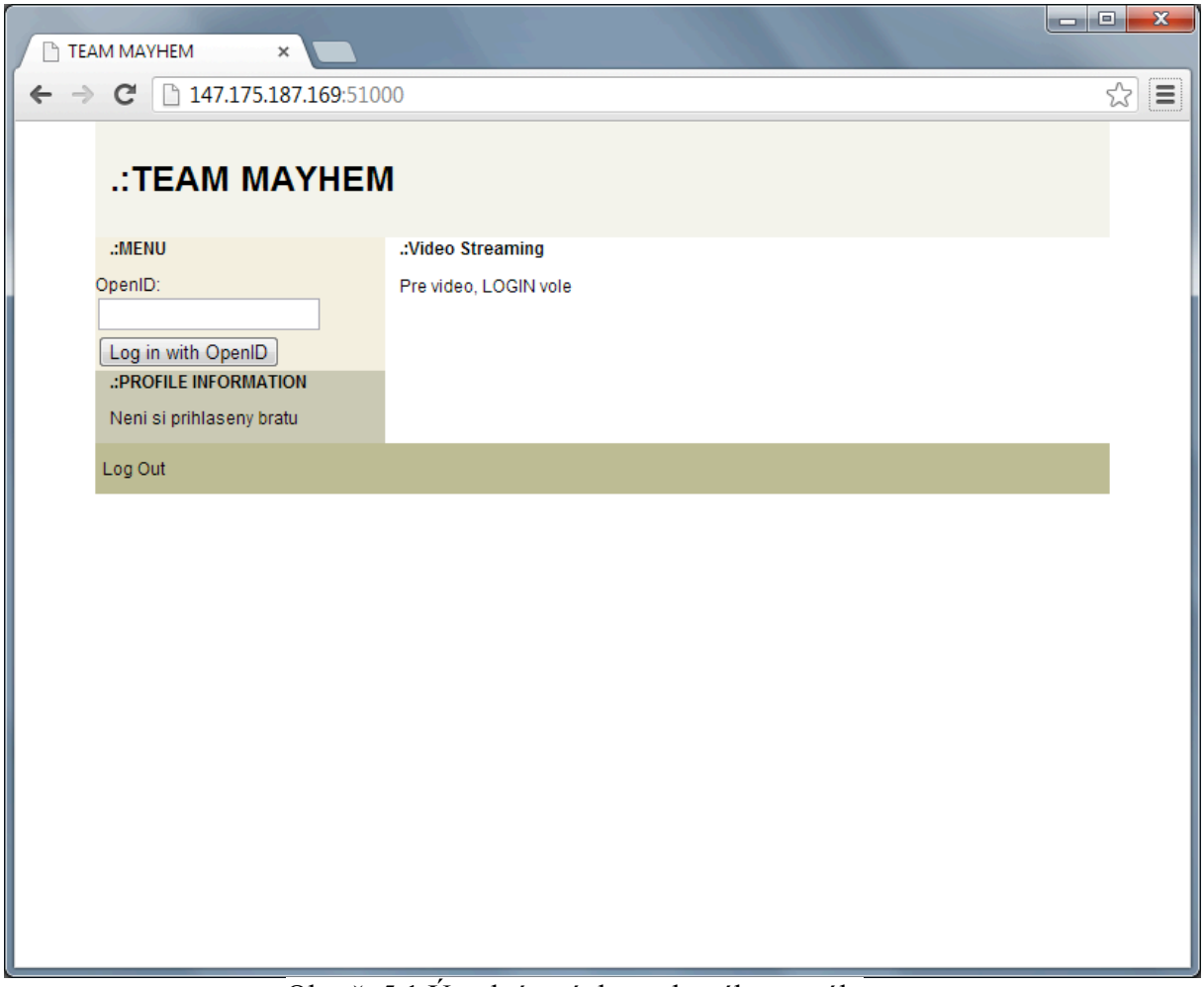
Vytvorený prototyp je možné rozdeliť na dva funkčné časti. Prvá časť predstavuje webový portál s možnosťou prihlásenia prostredníctvom OpenID a druhá časť predstavuje samotný mechanizmus pre nadviazanie spojenia webového prehliadača so serverovou stranou. Prototyp spĺňa teda všetky požiadavky, ktoré sme si stanovili pred implementovaním prototypu.

5.2.1 Autentifikácia na klientskej strane cez OpenID

Prototyp nášho webového portálu obsahuje len nevyhnutné prvky pre overenie

stanovenej funkcionality. Preto niektoré objekty ako napríklad zoznam dostupných videí, zobrazenie podrobnejších informácií vybraného videa nie sú v tejto fáze dôležité a budú doplnené v neskoršej fáze implementácie. Ide o predbežný návrh stránky, ktorý sa môže zmeniť, keďže je nutné dosiahnuť čo najjednoduchší dizajn a aby ovládanie danej stránky bolo čo najviac intuitívne.

Na začiatku sa používateľ prihlási na webovú stránku prostredníctvom OpenID. To dosiahne tým, že na úvodnej stránke (obr. č. 5.1) vloží do textového poľa svoje prihlasovacie údaje spolu s informáciou o OpenID poskytovateľovi. To vygeneruje žiadosť pre poskytovateľa identity. Pri potvrdení identity používateľom, je opätovne presmerovaný na náš webový portál. OpenID mechanizmus na našom serveri je vytvorený pomocou autentifikačnej konštrukcie PassportJS. PassportJS spracováva požiadavky generované používateľom a na základe spätnej odpovede od poskytovateľa identity zamietne alebo povolí používateľov ďalšie akcie s portálom. Na prácu s prijatými požiadavkami a odpoveďami sme použili aplikačný framework pre node.js Express. Prijatý identifikátor je zaznamenaný a uložený v jednoduchom cookie vo webovom prehliadači, aby nemusel sa používateľ neustále prihlasovať.



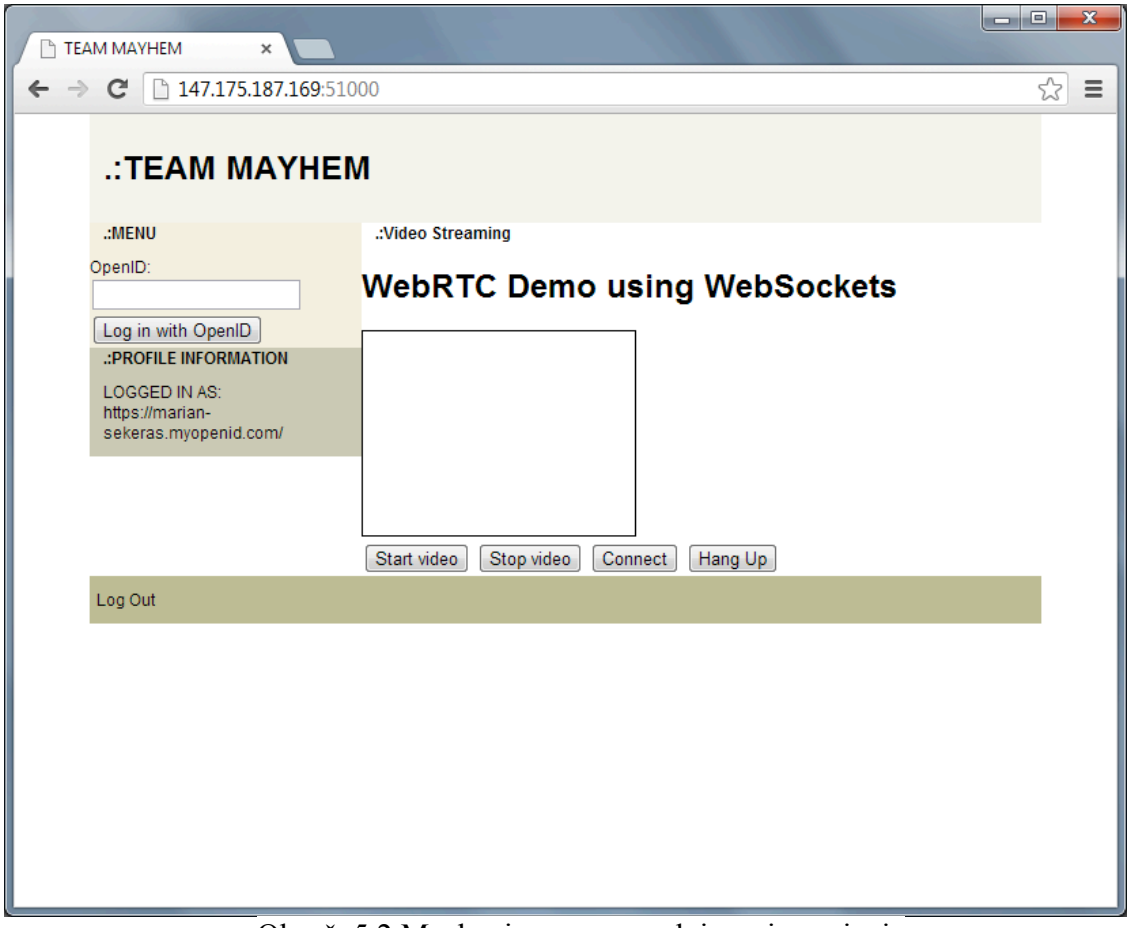
Obr. č. 5.1 Úvodná stránka webového portálu

5.2.2 Pripojenie sa na streamovací server

Najkomplikovanejšou a zároveň najpodstatnejšou úlohou bolo nastavenie potrebných atribútov na strane klienta a servera a vytvorenie spojenia medzi týmito dvoma zariadeniami. Druhá časť prototypu je práve zameraná na túto problematiku.

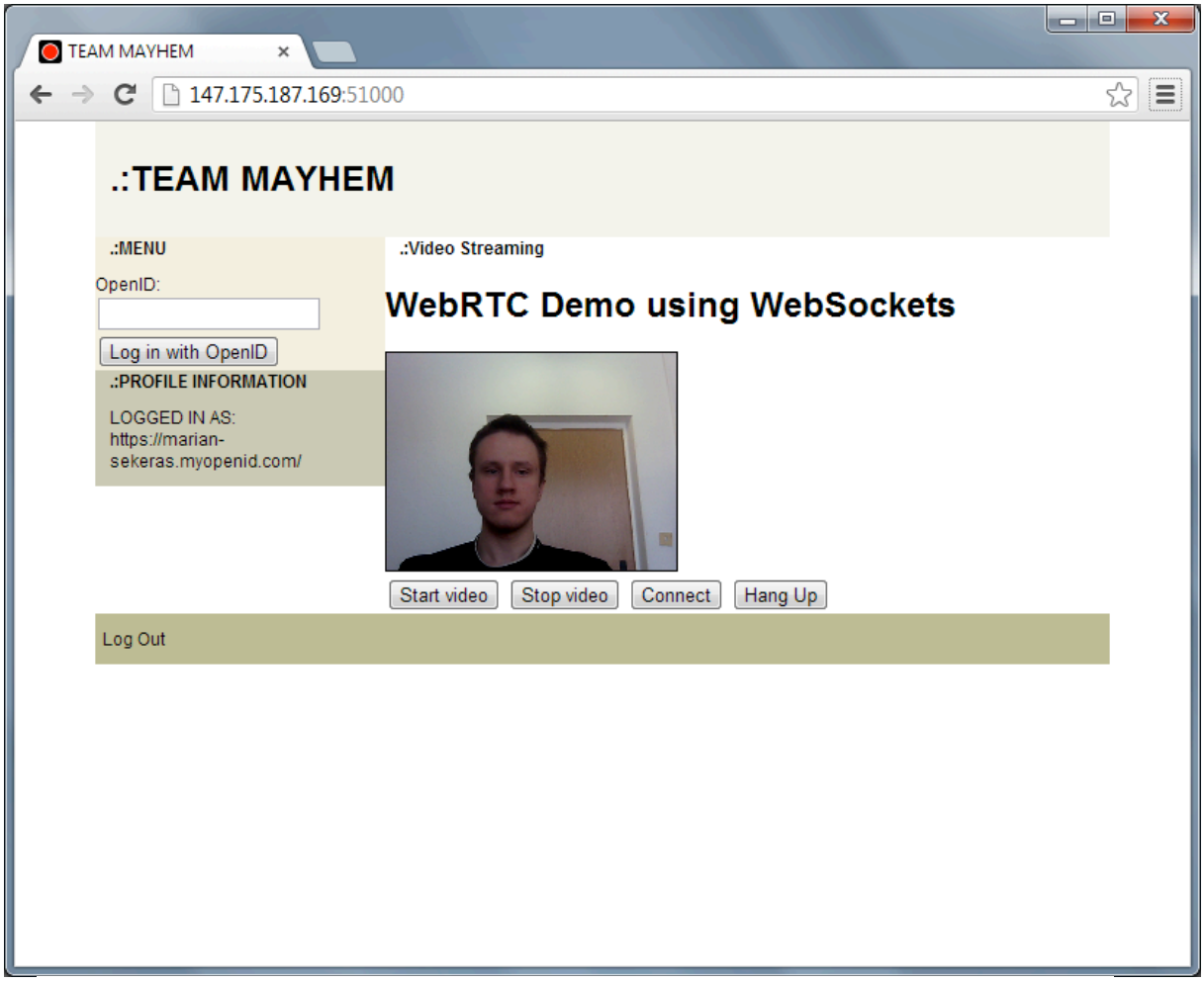
Časť zabezpečujúca spojenie klientov sa vo WebRTC nazýva PeerConnection. Spojenie sa vytvára na základe vymenenia si parametrov spojenia a ich vzájomným akceptovaním. Parametre sú špecifikované vo formáte, ktorý ustanovuje Session Description Protocol (SDP). Vytvárajú sa na základe multimediálnych zdrojov (audio, video), ktoré boli predtým pridané do objektu PeerConnection. Pretože štandard WebRTC nešpecifikuje, akým komunikačným kanálom má byť vytvorená SDP správa preposlaná, my pre tento účel využívame Websockety. Na strane servera je táto správa aktuálne vypísaná do konzoly. Následne sa pristupuje k nadviazaniu spojenia a začatiu posielania videa. Dohodnutie sa na konkrétnych parametroch prenosu a prenos samotného videa je potom už len implementačná záležitosť.

Na testovanie tejto funkcionality sme vytvorili na webovej stránke jednoduchý mechanizmus, ktorý sa objaví používateľovi, keď sa úspešne prihlási (obr. č. 5.2).



Obr. č. 5.2 Mechanizmus pre nadviazanie spojenia

Na začiatok používateľ stlačením tlačidla „start video“ zavolá funkciu getUserMedia, ktorá si vyžiada sprístupnenie zariadenia web kamery. Video zachytené cez web kameru je následne vykreslené do okna na webovej stránke (obr. č 5.3). Potom je možné pristúpiť ku samotnému nadviazaniu peerconnectionu prostredníctvom tlačidla „connect“. Vtedy je vygenerovaná správa SDP a poslaná prostredníctvom WebSocketu na streamovací server. Server v prípade žiadnych komplikácii prijme túto správu a vypíše nastavené atribúty.



Obr. č. 5.3 Vykreslenie zachyteného videa cez web kameru do okna webovej stránky

6 Zdroje

[1] Architektúra WebRTC. 10.11.2012. WebRTC
http://www.webrtc.org/reference/architecture

[2] Signalizácia a WebRTC. 10.11.2012. GETTING STARTED WITH WEBRTC
http://www.html5rocks.com/en/tutorials/webrtc/basics

[3] Komunikácia cez Websockety. 10.11.2012. Parallel, Asynchronous, and Real-Time Data Operations
http://ofps.oreilly.com/titles/9781449320317/ch_Parallelization.html

[4] OpenID: intro & howto for non-techies. 10.11.2012
http://www.consumingexperience.com/2006/12/openid-introduction.html

[5] CouchDB Official Page. 10.11.2012
https://couchdb.apache.org/

[6] PassportJS Development Page. 10.11.2012
http://passportjs.org/

[7] About JavaScript. 10.11.2012
https://developer.mozilla.org/en-US/docs/JavaScript/About_JavaScript

[8] Sahil Singh: WebRTC - using Media Stream and PeerConnection API. 8.12.2012
http://lihashgnis.blogspot.sk/2012/09/webrtc-using-media-stream-and.html

[9] Yanks Kevin, Cameron Adams: Simply JavaScript. 8.12.2012
http://www.itp.uzh.ch/~suzanne/ebooks/simplyJavascript.pdf

[10] OpenID: Wikipedia (autor obrázku). 8.12.2012
http://en.wikipedia.org/wiki/OpenID