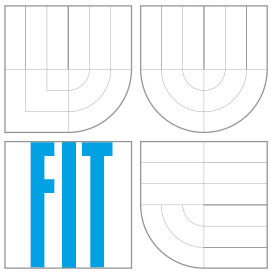
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DOPLNĚK PRO PROHLÍŽEČE PRO DETEKCI A ZPRACOVÁNÍ AUDIO A VIDEO STREAMŮ

BROWSER EXTENSION FOR AUDIO/VIDEO STREAM PROCESSING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB FEDOR

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. IGOR SZŐKE, Ph.D.

BRNO 2016

Zadání bakalářské práce/15826/2015/xfedor03

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Fedor Jakub**
Obor: Informační technologie
Téma: **Doplněk pro prohlížeče pro detekci a zpracování audio a video streamů**
Browser Extension for Audio/Video Stream Processing
Kategorie: Zpracování řeči a přirozeného jazyka

Pokyny:

1. Nastudujte tvorbu doplňků pro Firefox a Chrome. Najděte a porovnejte doplňky specializující se na snadné stahování multimediálního obsahu z webových stránek.
2. Navrhněte a implementujte doplněk, který bude detekovat multimediální obsah a na akci uživatele jej nechá zpracovat službou Audeliver.com. Zaměřte se na robustnost (detekce co nejvíce typů obsahu na širokém spektru stránek) a také na UX (uživatelskou zkušenost) doplňku.
3. Doplněk otestujte na vhodném vzorku beta-testerů
4. Pokračujte v implementaci. Získejte zpětnou vazbu od uživatelů. Doplněk publikujte.
5. Zhodnoťte výsledky a navrhněte směry dalšího vývoje.
6. Vyrobte A2 plakátek a cca 30 vteřinové video prezentující výsledky vaší práce.

Literatura:

- Dle pokynů vedoucího

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Szőke Igor, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Bakalárska práca sa zaoberá návrhom a tvorbou rozšírenia pre prehliadače a detekciou multimediálneho obsahu na webových stránkach. Súčasťou práce je analýza existujúcich riešení a testovanie. Celková práca je implementovaná v programovacích jazykoch HTML, CSS a JavaScript. Finálne rozšírenie spolupracuje so službou Audeliver.com. Výsledky práce sú voľne dostupné pod kľúčovým slovom Audeliver, v internetových obchodoch Google Web Store, Mozilla Add-ons a Opera Add-ons.

Abstract

Bachelor's Thesis describes design and development of extension for browsers and detection of multimedial content on web pages. Thesis includes analysis of existing extensions and testing. Whole work is implemented in programming languages HTML, CSS and JavaScript. The final extension cooperates with service Audeliver.com. The results of thesis are freely available under keyword Audeliver, at internet stores Google Web Store, Mozilla Add-ons and Opera Add-ons.

Kľúčové slová

rozšírenie, chrome, mozilla, opera, audeliver, prehliadač, multimédia, audio, video, podcast

Keywords

extensions, chrome, mozilla, opera, audeliver, browser, multimedia, audio, video, podcast

Citácia

FEDOR, Jakub. *Doplněk pro prohlížeče pro detekci a zpracování audio a video streamů.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Szőke Igor.

Doplněk pro prohlížeče pro detekci a zpracování audio a video streamů

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Igora Szőkeho, Ph.D.. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Jakub Fedor
18. mája 2016

Poďakovanie

Rád by som sa poďakoval vedúcemu práce Ing. Igorovi Szőkemu, Ph.D. za odbornú pomoc, konzultácie, usmernenie a cenné rady pri riešení bakalárskej práce.

Obsah

Kapitola 1

Úvod

V každom odvetí na svete existujú produkty, ktoré si užívateľ sám prispôsobí. U niektorých zakúpených produktov sa s touto možnosťou ráta už od začiatku príprav vývoja. Či už sa jedná o nový notebook alebo mobilný telefón, každý si rád zmení pozadie na ploche alebo minimálne upraví základné nastavenia. S príchodom internetovej revolúcie a s ňou spojených prehliadačov tomu nebolo ináč.

Prehliadače sa vo svojej podstate snažia zaistiť minimálne požiadavky pre užívateľa pri činnostiach spojených s prezeraním webu. Pozeranie videí, prezeranie textových dokumentov, počúvanie hudby, to je len zlomok toho, čo súčasný prehliadač zvládne. Avšak aj keď súčasný prehliadač zvláda desiatky vecí a miliónom užívateľom to stačí, vývojári nezabudli a poskytli iným užívateľom alebo vývojárom ďalšie možnosti ako ovplyvniť, prispôsobiť, možno vylepšiť chod prehliadača. S týmito cieľmi vznikli prvé rozšírenia a tak môžeme povedať, že rozšírenia sú malé programy, ktoré menia alebo obohacujú prehliadač. Podpora prehrávania videí, či čítanie PDF dokumentov nebolo natívnou súčasťou prehliadačov v minulosti. Tým pádom by sme mohli s istotou povedať, že rozšírenia môžu indikovať čo môže prísť ako ďalšia vlastnosť novej verzie prehliadača.

V tejto bakalárskej práci sa budem zaoberať vývojom doplnku pre prehliadače, ktorý je schopný detegovať multimediálny obsah na internete, je schopný komunikovať so vzdialeným serverom, ktorý obsah spracuje, tvorbou UI (užívateľského rozhrania) a UX (užívateľskou skúsenosťou). Tento doplnok najprv získa odkaz na audio alebo video stream, neskôr kontaktuje službu Audeliver.com, ktorá odkaz spracuje. Spracované odkazy budú pre užívateľa dostupné cez doplnok alebo priamo cez webové rozhranie Audeliver.com vo formáte samostatných nahrávok alebo vo formáte podcastu.

Text je rozčlenený do siedmich hlavných kapitol, kde prvá je úvod. Druhá kapitola sa zaoberá existujúcimi rozšíreniami s podobnou problematikou, ich porovnaním a zdôvodnením, prečo práve toto rozšírenie poskytuje viac ako ostatné. V tretej kapitole je rozberaná teoretická časť vývoja doplnku. Aké webové technológie boli použité, pre aké prehliadače sa doplnok vyvíjal, aké servery sa spracovávali a ako funguje nasadenie na oficiálne stránky pre správu doplnkov. Štvrtá kapitola sa zaoberá návrhom riešenia. Piata kapitola popisuje konkrétny vývoj od front endu po back end. Predposledná, šiesta kapitola patrí testovaniu a posledná, siedma kapitola je záver, kde sa zhrnú vyriešené problémy, k akým záverom sa došlo a kam v budúcnosti by mohla táto práca smerovať.

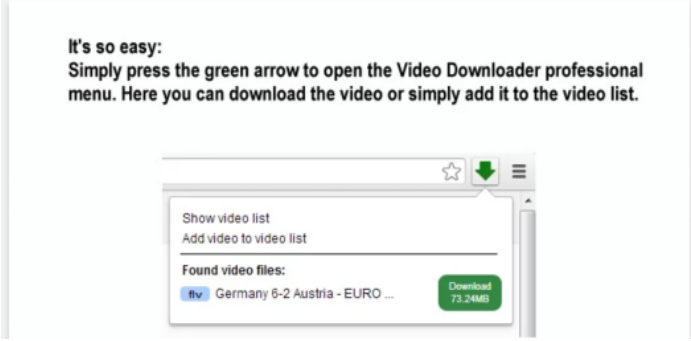
Kapitola 2

Existujúce rozšírenia

Existuje mnoho doplnkov a rozšírení pre prehliadače, ktoré sa zaoberajú problematikou vyhľadávania streamov, audiovizuálneho obsahu alebo detekciou multimédií. Táto kapitola predstavuje viacero doplnkov s podobnou tematikou, ktoré sú momentálne dostupné na trhu. Na konci kapitoly je uverejnené porovnanie týchto doplnkov.

2.1 Google Chrome

Internetový obchod Google (Google Webstore) ponúka mnoho rozšírení na detekciu audio a video obsahu. Je nutné si ujasniť, že väčšina týchto rozšírení nepodporuje niektoré servery ako Youtube, pretože vlastníkom tohto serveru je priamo spoločnosť Google[1].

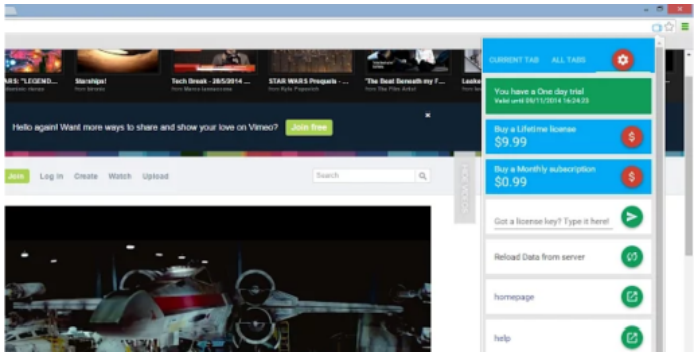


Obr. 2.1: Video Downloader professional[2]

Video Downloader professional je jeden z najpopulárnejších nástrojov na Google Webstore, kde si ho už stiahlo viac ako dva a pol milióna užívateľov. Ponúka možnosť stiahnutia videí na pevný disk alebo vytvorenie zoznamu videí, ktoré užívateľ už skôr navštívil. Najnovšia verzia ponúka užívateľovi možnosť prezerania videí cez Google Chromecast. Nepodporuje YouTube ako aj väčšina rozšírení na internetovom obchode Google.

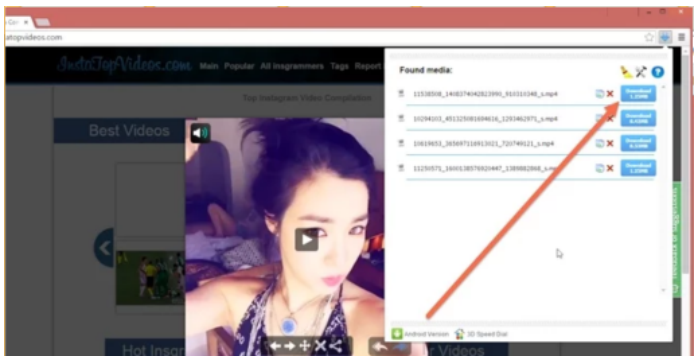
[1] https://www.youtube.com/yt/about/

[2] https://chrome.google.com/webstore/detail/video-downloader-professi/elicpjhcidhpjomhibiffojpinpmmpil



Obr. 2.2: Download Vimeo Videos, Premium[3]

Download Vimeo Videos, Premium je, ako z názvu vyplýva, doplnok, ktorý sa špecializuje na ukladanie videí zo servera Vimeo na pevný disk užívateľa. Je schopný stiahnuť videa nie len pomocou priameho odkazu na video, ale dokáže detegovať videá aj v embedded forme, napríklad na blogoch alebo iných stránkach. Hlavnou nevýhodou tohto rozšírenia je, že ponúka iba trial verziu na jeden deň. Plná verzia rozšírenia je pre užívateľa už spoplatnená.



Obr. 2.3: Flash Video Downloader[4]

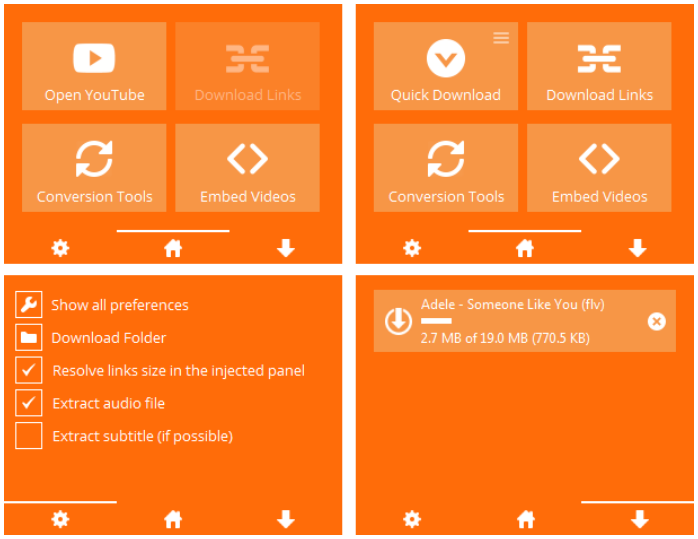
Flash Video Downloader je ďalší z rady doplnkov, ktoré sa špecializujú na download videí. Používa ho vyše milión užívateľov a je schopný sťahovať okrem základných video formátov aj videá alebo animácie vo formáte Flash. Jeho hlavnou výhodou je to, že sa nešpecializuje na jeden server, ale pokúša sa o robustnú detekciu multimediálneho obsahu.

2.2 Mozilla Firefox

Doplnky pre prehliadač Mozilla Firefox sú dostupné v Add-ons Mozilla centre. Mozilla Add-ons disponuje širokým sortimentom rozšírení pre detekciu a sťahovanie audio a video obsahu. Hlavným plus tohto centra a celkovo Firefox je, že povoľujú vo väčšine prípadov aj iné servery ako Google Chrome.

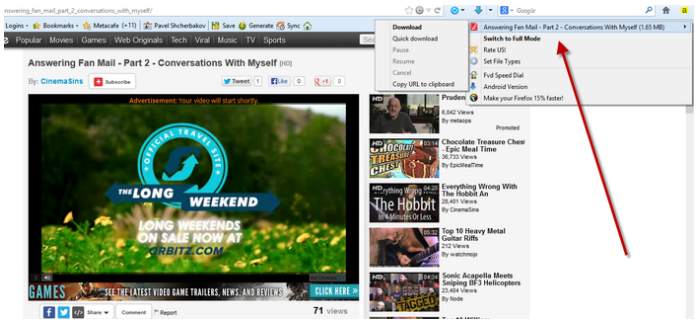
[3] https://chrome.google.com/webstore/detail/download-vimeo-videos-pre/phpaiffimemgakmakpcehgbophkbllkf

[4] https://chrome.google.com/webstore/detail/flash-video-downloader/aiimdkdngfcipjohbjenkahhlhccpdbc



Obr. 2.4: Youtube Video and Audio Downloader[5]

Youtube Video and Audio Downloader používa približne sedemsto tisíc užívateľov, tým pádom ho môžeme zaradiť do kategórie populárnych rozšírení. Dokáže stiahnuť video zo serveru YouTube vo všetkých dostupných kvalitách a formátoch. Disponuje taktiež knižnicou, pomocou ktorej vie konvertovať video do audio formátu. To ho stavia do popredia medzi ostatnými doplnkami pre Mozilla Firefox. Veľkým mínus je nulová podpora iných video serverov.



Obr. 2.5: Flash Video Downloader - YouTube HD Download [4K][6]

Flash Video Downloader poskytuje širokú podporu serverov, čo ho stavia do veľmi výhodnej pozície. Podporuje napríklad YouTube, Facebook alebo Metacafe. Disponuje taktiež stiahnutím Flash videí alebo animácií a videí v rozlíšení 4K[7].

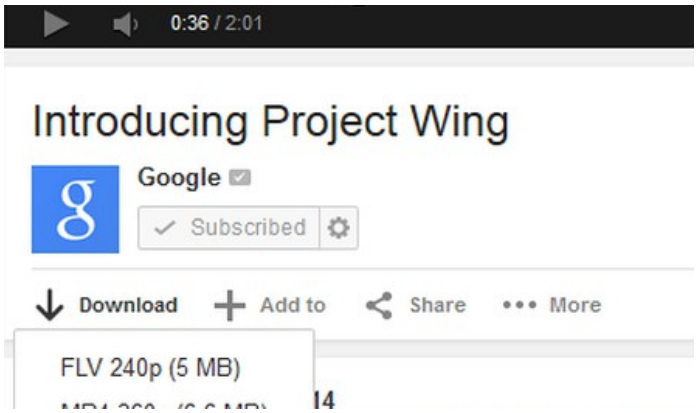
[5] https://addons.mozilla.org/cs/firefox/addon/youtube-video-and-audio-dow/
[7] Nový štandard pre rozlíšenie obrazu vo filme a počítačovej grafike
[7] https://addons.mozilla.org/cs/firefox/addon/flash-video-downloader/

Obr. 2.6: Ant Video Downloader[8]

Tento doplnok, Ant Video Downloader, podporuje viacero serverov s multimediálnym obsahom. Užívateľ jednoducho, pár klikmi, stiahne video ktoré práve sleduje. Bohužiaľ nepodporuje audio servery ako SoundCloud.

2.3 Opera

Opera addons poskytuje mnoho rozšírení, ktoré uľahčia užívateľovi download súborov z video serverov.



Obr. 2.7: Download YouTube Videos as MP4[9]

Rozšírenie Download YouTube Videos as MP4, ako už z názvu vyplýva, má k dispozícii sťahovanie YouTube videí do formátu MP4. Má jednoduché užívateľské rozhranie, ktoré je integrované priamo do HTML kódu stránky navštíveného videa.

[8] https://addons.mozilla.org/cs/firefox/addon/video-downloader-player/
[9] https://addons.opera.com/cs/extensions/details/download-youtube-videos-as-mp4/

2.4 Porovnanie rozšírení

	VDP[10]	DVV[11]	FVD[12]	YVAD[13]	FVD[14]	AVD[15]	DYV[16]
Podpora viacero serverov	✓	X	✓	X	✓	✓	X
Konverzia videa na iný formát	X	X	X	✓	X	X	✓
Rozšírenie je zadarmo	✓	X	✓	✓	✓	✓	✓
Podpora audio serverov	X	X	✓	X	X	X	X

Tabuľka 2.1: Porovnanie dostupných rozšírení

Zistil som, že väčšina rozšírení sa zameriava na robustnú detekciu multimédií, no stále je tu mnoho ďalších, čo sú špecializované na jeden server. Drvivá väčšina doplnkov nie je schopná získané médium konvertovať do iného formátu ako originál. Väčšina rozšírení sú v základnej forme zadarmo a máloktoré podporujú audio servery.

Týmto zistením som prišiel na to, že **existuje diera** na trhu pre doplnok, ktorý vie robustne detegovať multimédia na audio/video serveroch a následne vie s takýmto získaným obsahom narábať. Funkcionalita väčšiny doplnkov po získaní odkazu na stream končí.

V tejto práci bude doplnok konvertovať získaný obsah do audio formátu, kde bude spolupracovať so službou Audeliver.com. Za pomoci tejto služby budeme mať rozšírenie, ktoré momentálne na trhu **nemá veľkú konkurenciu**. Vo finálnej fáze, si užívateľ bude môcť tieto konvertované multimédia vypočuť vo forme nahrávok alebo podcastov, pomocou spomínanej služby Audeliver.com.

[10] Video Downloader professional
[11] Download Vimeo Videos, Premium
[12] Flash Video Downloader
[13] Youtube Video and Audio Downloader
[14] Flash Video Downloader - YouTube HD Download [4K]
[15] Ant Video Downloader
[16] Download YouTube Videos as MP4

Kapitola 3

Použité technológie

Táto kapitola má za úlohu objasniť a vysvetliť s akými technológiami budeme pracovať. Objasní situáciu ohľadom moderných používaných prehliadačov, ukáže ktoré to sú a akú majú podporu pre rozšírenia. Ktoré potenciálne servery s multimediálnym obsahom by malo toto rozšírenie v ideálnom prípade podporovať. Ukáže špecifikácie pri vývoji doplnkov pre jednotlivé platformy a v neposlednej rade ozrejmí proces nasadzovania a publikovania rozšírení na internete respektíve na oficiálnych platformách, ktoré poskytujú prehliadače vývojárom.

3.1 Webové technológie

Pri tvorbe doplnkov budeme potrebovať webové technológie ako HTML, CSS a JavaScript. Budeme ich potrebovať práve preto, pretože moderné vývojové postupy pri tvorbe rozšírení považujú rozšírenie ako takú webovú stránku, ktorá je napísaná v HTML, dizajn je nadefinovaný v CSS a funkcionalita je zabezpečená pomocou jazyka JavaScript. V nasledujúcich podkapitolách budú práve tieto jazyky podrobne, ale stručne vysvetlené a objasnené čitateľovi.

3.1.1 HTML

HyperText Markup Language alebo HTML, je textový značkovací jazyk používaný pre tvorbu webových stránok, ktoré môžu a v drvivej väčšine sú prepojené hypertextovými odkazmi. Každá webová stránka sa skladá aspoň z minimálneho množstva kódu jazyka HTML, inak by to nebola webová stránka. Jazyk HTML je vhodný pre definovanie významu obsahu stránok. Stránky jazyka HTML sú textové súbory, vďaka čomu je ich možné jednoducho upravovať.

Jazyk HTML vznikol na počiatku deväťdesiatych rokov minulého storočia v podobe stručného dokumentu popisujúceho niekoľko elementov používaných pre tvorbu webových stránok. Mnoho z týchto elementov popisovala rôzne časti webovej stránky; napríklad záhlavie, odseky alebo zoznamy. Číslo verzie jazyka HTML sa postupne zvyšovalo, ako sa tento jazyk rozrastal o nové elementy a prispôsoboval nových potrebám. Najnovšou verziou tohto jazyka je HTML5.

HTML5 sa prirodzene vyvinul zo starších verzií jazyka HTML, pričom sa snaží reflektovať potreby súčasných ale aj budúcich webových stránok. Tento jazyk zdedil veľkú časť vlastností svojich predchodcov. To znamená, že pokiaľ ste už v minulosti vyvíjali v jazyku HTML, tak s príchodom do novšej verzie jazyka HTML5 ste nemali problém, pretože ste

už poznali väčšiu časť tohto jazyka. Ďalším dôsledkom HTML5 je to, že jeho prevažná časť funkcií bude fungovať aj na starších webových prehliadačoch. Spätná kompatibilita bola kľúčovou vlastnosťou návrhu jazyka HTML5.

Najnovšia verzia jazyka samozrejme prináša mnoho nových funkcií. Niektoré sú z nich veľmi jednoduché ako doplnkové elementy k popisu obsahu:

- section (časť)
- figure (obrázok)
- article (článok)
- a ďalšie

Iné funkcie sú zložitejšie a pomáhajú s tvorbou prepracovanejších webových aplikácií. Jazyk HTML5 taktiež po novom zavádza podporu prehrávania zvukových súborov a video súborov priamo v moderných webových prehliadačoch a to bez nutnosti inštalácie alebo pridania doplnkov.[1]

3.1.2 CSS

CSS je skratkou pre kaskádové štýly alebo v angličtine *Cascading Style Sheets*. Jedná sa o jazyk pre popis spôsobu zobrazenia elementov na stránkach vytvorených pomocou jazyka HTML. Prvé verzie jazyka CSS sa objavili až niekoľko rokov po vzniku jazyka HTML, pričom oficiálne sa jazyk CSS presadil až v roku 1996. Najnovšou verziou tohto jazyka je CSS3.

Vzťah medzi jazykom CSS3 a jeho starším verziami je analogický k vzťahu jazyka HTML5 s jeho predošlými verziami - jazyk CSS3 je prirodzeným rozšírením svojich starších verzií. CSS3 mnohom silnejší ako jeho predchodcovia. Zavádza totiž niekoľko vizuálnych efektov, zaoblené rohy alebo prechody. Celkový vzhľad je definovaný v šablóne štýlov alebo po anglicky v *stylesheet*.

Šablóna štýlov je obyčajný textový súbor, ktorý obsahuje jedno alebo viacero pravidiel, ktoré prostredníctvom svojich vlastností a hodnôt určujú, ako by sa mali určité elementy zobrazovať. Jazyk CSS ponúka vlastnosti pre formátovanie textu, vlastnosti pre definíciu rozvrhnutia umiestnenia elementov a tiež vlastnosti pre riadenie tlače.

Najlepšie podporovanou verziou jazyka CSS naprieč všetkými webovými browsermi je verzia CSS2. Výbornou správou je to, že takmer s každou novou verziou prehliadačov sa rozširuje podpora novšej verzie CSS respektíve CSS3.[1]

3.1.3 JavaScript

JavaScript je dynamický, netypovaný, interpretovaný skriptovací jazyk navrhnutý pre interakciu s webovými stránkami. Nemá nič spoločné s Javou, názov bol zvolený v minulosti iba pre rozruch pri jazyku Java a tak sa tvorcovia chceli zviesť na tomto druhu ošiaľu. Prvýkrát sa objavil v roku 1995, kde slúžil len ako validátor vstupov užívateľa, pretože dovtedy vstupy nevalidoval klient ale server. S pomalým internetom, sa úkony spravené už na klientskej strane zdali ako veľmi výhodné.

V súčasnosti je JavaScript veľmi dôležitou súčasťou každého moderného browsera. JavaScript je v súčasnosti schopný pracovať s celým oknom prehliadača a jeho obsahom. V dnešnej dobe je tento jazyk považovaný za plnohodnotný programovací jazyk, schopný komplexných výpočtov, vrátane uzáverov (closures), anonymných lambda funkcií alebo dokonca

metaprogramovania. JavaScript patrí k trom základným technológiám pre tvorbu webu v modernom prehliadači.

Je to jazyk, ktorý má jednoduché základy, ale pretože je oproti ostatným jazykom príliš voľný, trvá roky dokým sa programátor vypracuje na experta. Jazyk v minulosti nemal žiaden platný štandard, to znamenalo, že bol ešte voľnejší. V 1997 bol JavaScript 1.1 navrhnutý Európskej asociácii výrobcov počítačov (ECMA), kde bolo vydané prvé štandardizovanie jazyka.

Je zložený z týchto troch častí:

- ECMAScript
- DOM
- BOM

ECMAScript, je jazyk definovaný v ECMA-262, ktorý nie je nutne spojený s webovými prehliadačmi. V skutočnosti ECMA-262 definuje tento jazyk ako bázu na ktorej môžu byť postavené iné jazyky. ECMAScript špecifikuje syntax, typy, kľúčové slová, výrazy, rezervované slová, operátori a objekty. V súčasnosti je podporovaná stabilná verzia ECMAScript o čísle 6. Siedma edícia je práve vo vývojovom štádiu.

DOM je skratka pre *Document Object Model*, čo je rozhranie pre programovanie aplikácii v XML, ktoré bolo funkčne rozšírené v HTML. Document object model mapuje celú web stránku ako hierarchiu uzlov. Tieto uzly vo výsledku tvoria akoby strom, ktorý reprezentuje dokument. DOM poskytuje vývojárom obrovskú kontrolu nad obsahom a štruktúrou dokument. Uzly môžu byť mazané, pridávané, vymenené ale pozmenené jednoducho pomocou DOM aplikačného rozhrania. Je potrebné poznamenať, že DOM nie je špecifický iba pre JavaScript a bol implementovaný aj v iných jazykoch. Pre webové prehliadače bol implementovaný pomocou ECMAScript a dnes tvorí veľkú časť jazyka JavaScript.

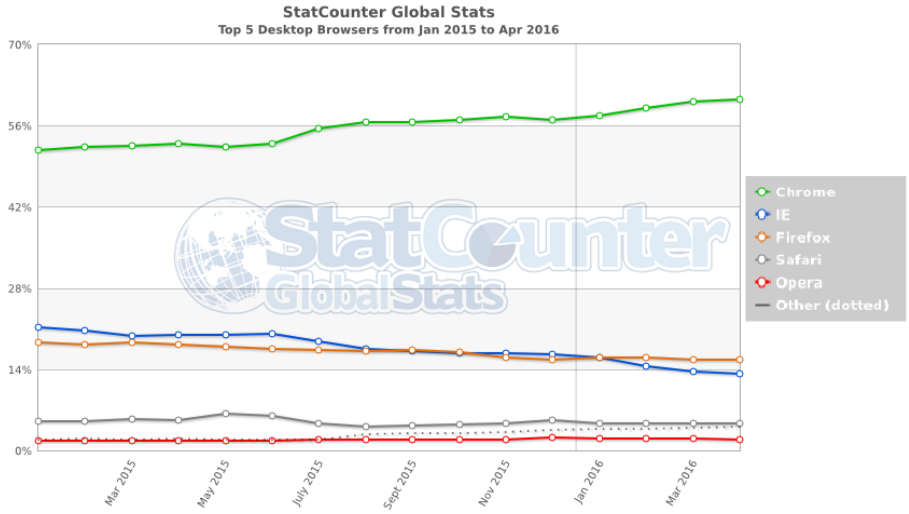
BOM je skratka pre *Browser Object Model*, ktorý je súčasťou jazyka JavaScript a umožňuje prístup a manipuláciu s oknom prehliadača. S použitím Broswer Object Modelu, môžu vývojári pracovať s prehliadačom aj mimo kontext zobrazenej stránky. Primárne, BOM je používaný k práci s oknami a rámami prehliadača, no v dnešnej situácii môžeme každé rozšírenie pre prehliadač, ktoré používa JavaScipt, považovať za časť Browser Object Model[5].

Funkcionalita týchto rozšírení je schopná napríklad:

- Zobraziť nové okno v prehliadači
- Pohybovať, meniť veľkosť alebo zatvárať okná prehliadača
- Podporovať cookies
- Vytvárať objekty ako XMLHttpRequest
- Používať objekty ako location, screen alebo navigator

3.2 Webové prehliadače

Webové prehliadače sú aplikácie, ktoré slúžia užívateľovi ako nástroj pre zobrazenie a interakciu s HTML dokumentmi. Moderné webové prehliadače sú schopné interpretovať aj



Obr. 3.1: Graf popularity prehliadačov za obdobie Január 2015 - Apríl 2016 [1]

ďalšie technológie ako JavaScript alebo kaskádové štýly. Medzi najznámejšie prehliadače patria: Google Chrome, Firefox, Internet Explorer, Opera a Safari.

Ako v grafe môžeme vidieť, najpopulárnejší prehliadač je **Google Chrome**, ktorý je výrazne vpredu pred Internet Explorerom. Internet Explorer sa zdá byť druhý, no roky potvrdzujú jeho klesajúcu tendenciu. S príchodom roku 2016, Firefox predbehol Internet Explorer a stal sa druhým najpopulárnejším browserom na trhu. Prehliadač Safari vďaka tomu, že je výhradne inštalovaný len na platformu Apple ostáva na predposlednom mieste, za ktorým nasleduje Opera.

V tejto práci, už zo zadania, budem pracovať s prehliadačmi Google Chrome a Firefox. Keďže Opera má podobné rozhranie pre vývoj rozšírení ako Google Chrome, rozhodol som sa tento prehliadač, respektíve podporu pre tento prehliadač, zahrnúť do tejto bakalárskej práce.

3.2.1 Google Chrome

Google Chrome je webový prehliadač spoločnosti Google s minimalistickým dizajnom. Ponúka štandardné funkcie ako zmeny veľkosti písma, správu hesiel alebo záložky. Na trhu je ako najpopulárnejší prehliadač s podielom vyše 60%. Prvá verzia prehliadača vyšla v roku 2008, no jednotkou na trhu sa stal až v roku 2012[2]. Je dostupný na všetkých operačných systémoch v hlavnom prúde, ako Windows, OSX a Linux. Taktiež je dostupný na mobilných zariadeniach. Je freeware a má výbornú podporu JavaScriptu, čo je dôležité pre tento projekt, pretože väčšina back-endu pri vývoji rozšírení, prebieha v JavaScripte. Disponuje možnosťou vytvorenia účtu pod Google Chrome, kde si prehliadač zapamätá záložky, nainštalované rozšírenia alebo heslá. To uľahčí užívateľovi prechod z jedného počítača na druhý. Podpora pre vývoj rozšírení je centralizovaná na Chrome Webstore.

[1] http://gs.statcounter.com/#desktop-browser-ww-monthly-201501-201604

3.2.2 Firefox

Firefox alebo Mozilla Firefox je open-source webový prehliadač, ktorý je taktiež zdarma. Je vyvíjaný spoločnosťou Mozilla Foundation. Je dostupný na operačných systémoch ako Windows, OS X alebo Linux. Tak isto ako aj Google Chrome, aj Firefox je dostupný na mobilných zariadeniach. Prehliadač bol zverejnený v roku 2004, kde sa hneď zapísal ako úspešný konkurent vtedajšieho Internet Explorer 6. Bol bezpečnejší, rýchlejší a taktiež podporoval oveľa viac rozšírení[3]. Začiatok roka 2016 sa stal druhým najúspešnejším prehliadačom na trhu s podielom cca 16%. Spoločnosť Mozilla bola zvolená ako najdôveryhodnejšia internetová spoločnosť, čo taktiež nahráva sympatie užívateľom. Firefox sa poslednými verziami zapísal ako najrýchlejší prehliadač, kde v benchmark testoch predbehol Chrome o 70 bodov[2]. Podpora pre rozšírenia je sústredená na Mozilla Add-ons.

3.2.3 Opera

Opera je webový prehliadač vyvinutý spoločnosťou Opera Software. Je dostupný pre operačné systémy Windows, OS X alebo Linux. Niektoré verzie Opery sú schopné fungovať aj na FreeBSD systémoch. Na mobilných zariadeniach má Opera vyhradené produkty ako Opera Mobile, Opera Mini a Opera Coast. Opera je známa adaptovaním vlastností, ktoré ostatné prehliadače nasadili až neskôr a to blokovanie pop-up okien, speed dial, sessions v prehliadači, súkromné surfovanie po webe alebo prítomnosť tabov[3]. Na trhu je Opera umiestnená až za Safari, čiže na piatom mieste. No s klesajúcou tendenciou Internet Exploreru a kvalitným produktom, sa môže v nasledujúcich rokoch vyšvihnúť na štvrté miesto. Vývoj a publikovanie rozšírení pre Operu je dostupný na Opera add-ons. Tu zase Opera dokazuje svoju silnú stránku v napredovaní v trendoch. Pri vývoji je možné použiť aplikačné rozhranie, ktoré používa aj Google Chrome. Tým pádom, pre vývojára je jednoduchšie vyvíjať rozšírenie pre Operu, ak ho už vyvinul pre Google Chrome.

3.3 Servery s multimediálnym obsahom

V tejto časti sa rozoberie jedna z najdôležitejších častí tohto rozšírenia a to detekcia multimédií zo serverov, ktoré poskytujú multimediálny obsah. Konkrétne sa bude jednať o problematiku získavania odkazu na stream audio alebo video súboru. Taktiež sa v krátkosti predstavia servery, ktoré tento obsah poskytujú.

3.3.1 Všeobecne

V dnešnej dobe existuje mnoho serverov, ktoré disponujú audio/video obsahom. No získanie odkazu nie je vždy jednoduché. Každý väčší server neposkytuje odkazy priamo v zdrojovom kóde stránky. K odkazom sa dostávame cez oficiálne poskytnuté aplikačné rozhranie, cez rôzne skripty, ktoré pracujú s DOM[4] modelom stránky alebo z prvotnej stránky vyberieme potrebné dáta, napríklad vo formáte JSON[5] alebo XML[6] a tie ďalej spracovávame. Môžeme naraziť na prehrávače, ktoré zdrojové dáta nečítajú z jedného súboru a výsledné video je súčtom mnoho súborov s malou kapacitou. V jednoduchosti sa dá tvrdiť, že neexistuje

[2] https://www.mozilla.org/en-US/firefox/desktop/
[3] http://www.opera.com/docs/history/presto/#o40
[4] Document Object Model
[5] JavaScript Object Notation
[6] Extensibe Markup Language

univerzálny prostriedok, na detekciu audia a videa na internete. Každý doplnok, čo toto presne tvrdí je zoskupením mnoho skriptov, kde je väčšinou jeden skript určený jednému serveru, no vo výsledku sa užívateľovi javí takýto doplnok ako niečo univerzálne a jasné.

3.3.2 Soundcloud, Vimeo, Tedtalks, vid.me, dailymotion

Táto časť predstaví populárne servery, ktoré by rozšírenie malo potenciálne spracovať. Jedná sa o servery, ktoré poskytujú buď len audio obsah po servery aj s video obsahom. Výber serverov bol vykonaný podľa ich relevantnosti v dnešnej dobe.

Soundcloud.com je platforma pre zvukové záznamy a hudbu. Momentálne v tejto kategórii pôsobí ako svetový líder. Na tejto stránke môže hocikto zdieľať svoju hudbu, podcasty, relácie alebo iné zvukové záznamy. Okrem zdieľania môže užívateľ propagovať a tvoriť nové zvukové záznamy. Momentálne má okolo 175 miliónov unikátnych mesačných poslucháčov[7].

Vid.me je v súčasnosti jeden z najprogresívnejších upload serverov. Ľudí zaujal hlavné svojím minimalistickým dizajnom a takzvanou čistotou, ktorú ponúka voči, už dnes chaotickému YouTube.

TedTalks.com je server, ktorý sústreďuje prednášky, ktoré boli prezentované na platforme TED (Technology, Entertainment, Design). Je známy kvalitným obsahom, ktorý hlavne tvoria zaújimaví prednášajúci. Na serveri môžeme nájsť mnoho náučných videí, ktoré sú prednášané špecifickým, viac na bežnú verejnosť, orientovaným štýlom.

Vimeo.com bolo v roku 2004 založené skupinou filmárov, ktorý chceli zdieľať svoje osobné momenty alebo kreatívne diela. Postupom času sa okolo Vimeo vytvorila veľká komunita ľudí s podobnými filmárskymi záľubami. Bola to prvá stránka, ktorá podporovala, dnes už bežné, videá vo vysokom rozlíšení. Momentálne Vimeo navštevuje vyše 213 miliónov unikátnych užívateľov a viac ako 45 miliónov registrovaných užívateľov[8].

Dailymotion.com je francúzky server, na ktorom môžu užívatelia nahrávať, pozerať a zdieľať video obsah. Jedna sa o jedného z najväčších gigantov v tomto priemysle, hneď vedľa YouTube a Vimeo. Dnes je lokalizovaný do mnoha jazykov a taktiež ponúka obsah z lokality, ktorá je užívateľovi blízka.

3.4 Špecifikácie pri vývoji doplnkov

Vývoj doplnkov pre prehliadače v dnešnej dobe zatiaľ ešte nie je štandardizovaný. V tejto kapitole sú rozobraté jednotlivé platformy a ich prípadne výnimky, ktoré sa vymykajú štandardu iných platforiem.

3.4.1 Google Chrome extensions

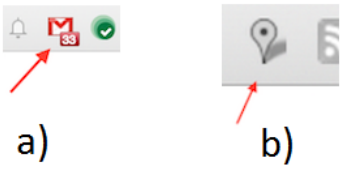
Rozšírenie pre Google Chrome je archív so zabalenými súbormi. Môže obsahovať HTML, CSS, JavaScript, obrázky a všetko zvyšné, čo môže vývojár k implementácii potrebovať ak chce vylepšiť vlastnosti Google Chrome. Tieto rozšírenia sú v jednoduchosti webové stránky, ktoré môžu využiť API poskytované prehliadačom, od XMLHttpRequestov[link] cez JSON po HTML5. Rozšírenia môžu pracovať s webovými stránkami alebo servermi

[7] http://blogs.wsj.com/digits/2014/12/09/soundclouds-valuation-could-top-1-2-billion-with-new-\tabularnewlinefundraising/

[8] http://iac.com/brand/vimeo

pomocou content skriptov alebo cross-origin XMLHttpRequestov. Rozšírenia sú schopné manipulovať taktiež s vlastnosťami browsera ako záložky alebo okná.

Chrome pridáva užívateľské rozhranie do rozšírenia pomocou takzvaných browser actions alebo page actions. Každé rozšírenie musí byť buď browser action alebo page action. Browser action sa hodí vtedy ak je rozšírenie relevantné k väčšine stránok, page action sa vyberie vtedy, ak sa rozšírenie hodí len ku špecifickým stránkam.



Obr. 3.2: a) rozšírenie s browser action b) rozšírenie s page action [9]

Rozšírenia implementované pre Google Chrome, môžu prezentovať užívateľské rozhranie aj v iných formách. Užívateľské rozhranie sa môže objaviť aj ako context menu, ako stránka nastavení. Každé rozšírenie obsahuje nasledujúce súbory:

- Súbor manifest
- Jeden alebo viac HTML súborov (pokiaľ rozšírenie nie je iba téma)
- Jeden alebo viac JavaScript súborov (nepovinné)
- Ďalšie súbory, ktoré rozšírenie potrebuje ako napríklad obrázky (nepovinné)

Všetky tieto súbory, ktoré patria k rozšíreniu, sa pred odovzdaním zabalia do špeciálneho ZIP archívu s príponou .crx. Po vytvorení archívu môžeme toto rozšírenie distribuovať ďalej.

V súbore manifest (konkrétne manifest.json), sú špecifikované základné informácie o rozšíreniu ako verzia, názov, popis a ikona. Taktiež sa v ňom špecifikujú skripty, ktoré sa používajú spolu s ostatnými súbormi.

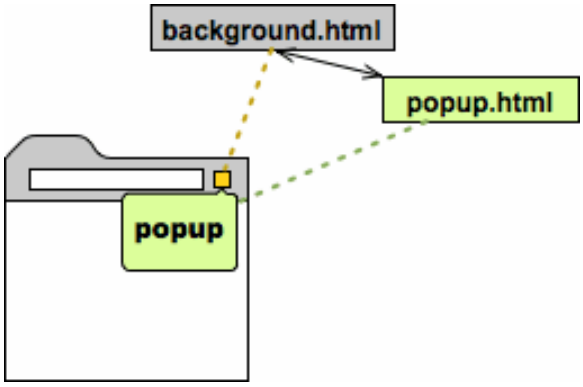
Architektúra rozšírenia v Google Chrome väčšinou minimálne pozostáva z takzvanej ***background page***. Je to stránka, ktorá drží po hromade hlavnú logiku doplnku. Samozrejme, rozšírenie môže obsahovať ďalšie súbory, pomocou ktorých vytvorí ***užívateľské rozhranie***. Ak chce vývojár narábať a pracovať z kódom konkrétnej stránky, tak použije ***content script***.

Background page je stránka, ktorá nie je viditeľná užívateľovi rozšírenia a je definovaná v background.html, kde môže obsahovať JavaScript kód, ktorý môže ovplyvňovať správanie rozšírenia. Existujú dva typy background stránok a to *Event pages* a *persistent background pages*. *Event pages* sú otvorené a zatvorené podľa potreby ako ich je treba. *Persistent background pages*, ako z názvu vyplýva, sú stránky, ktoré sú perzistentné. Pokiaľ, nepotrebujeme aby background page bežala stále, preferujeme *Event page*.

Užívateľské rozhranie tvoria HTML stránky, ktoré sa zobrazujú užívateľovi. Napríklad browser action rozhranie má popup okno, ktoré je implementované pomocou HTML. Každé rozhranie môže mať aj stránku nastavení, kde si užívatelia môžu kustomizovať ako

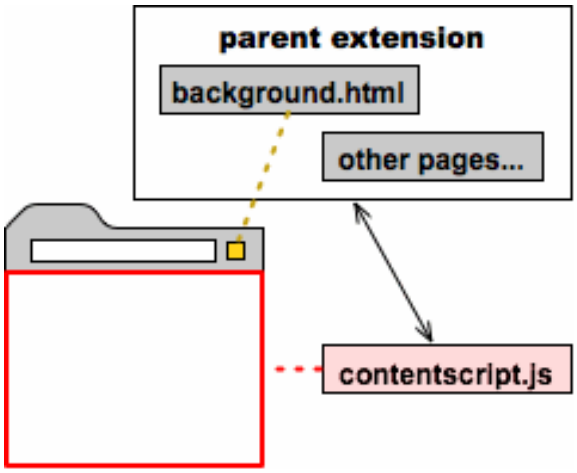
[9] https://developer.chrome.com/extensions/overview

má doplnok fungovať. Nasledujúci obrázok ukazuje architektúru pri browser action ak používa popup okno:



Obr. 3.3: Princíp fungovania popup okna s background page

Popup okno sa zobrazí ako web stránka, pretože je taktiež formátu HTML. Na tomto obrázku 3.3, rozšírenie obsahuje aj background page. Ak by vývojár chcel aby mala popup stránka podobnú funkcionalitu respektíve zdieľala logiku s background page, tak nemusí kopírovať časti tejto page, ale je schopný naviazať komunikáciu medzi popup a background page.



Obr. 3.4: Princíp fungovania content scriptov s background page[10]

Ak doplnok potrebuje pracovať s aktuálnou stránkou, tak v tom prípade potrebuje **content skript**. Content script je ľubovoľný JavaScript kód, ktorý sa vykoná v kontexte stránky, ktorá sa pravé načítala v prehliadači. Dalo by sa povedať, že content skript je ďalšou súčasťou práve načítanej stránky a nie pravou súčasťou rozšírenia.

Content skript je schopný čítať detaily o stránke, ktorú práve navštívil a tak je schopný prevádzať zmeny na tejto stránke. Nie je schopný pristupovať k DOM background page. Je však zdatný komunikovať s background page pomocou správ. Napríklad content skript zistí niečo čo sa udialo na stránke, pošle to background page a tá to môže poslať popup oknu. Vo výsledku môže popup okno zobrazovať informácie, ktoré obdŕžal content script.

[10] https://developer.chrome.com/extensions/overview

3.4.2 Mozilla Add-ons

Doplnky pre Mozillu Firefox sa delia do dvoch hlavných kategórií. Tie, ktoré rozširujú funkcie prehliadača sú takzvané ***Extensions***, zatiaľ, čo druhá kategória, ***Themes***, dokáže meniť užívateľské prostredie prehliadača. Keďže s Themes má tento projekt nulovú spojitosť, budeme sa zaoberať iba Extensions.

Extensions sú v preklade rozšírenia, čiže rozširujú funkcionalitu aplikácií ako Firefox alebo Thunderbird. To znamená, že tak isto ako rozšírenia v Chrome, dokážu pridať napríklad nové spôsoby ako spravovať taby alebo meniť obsah stránky.

Momentálne existujú štyri hlavné techniky, ako vytvoriť rozšírenie pre Firefox. Tri z nich sú oficiálne spustené pre vývojárov, jedna je ešte vo fáze testovania a stále sa čaká na prvé oficiálne zverejnenie. Prvé tri sú:

- **Add-on SDK extensions**
- **Restartless extensions**
- **Overlay extensions**

Posledná, ale nie menej dôležitá je metóda tvorenia pomocou **WebExtensions API**.

Overlay extensions sú rozšírenia, pri ktorých sa na vývoj rozhrania používa XUL overlay[11] a API pomocou ktorého môžeme mať prístup k tabom alebo JavaScript modulom. Tento spôsob je zastaraný, ale je potrebné vedieť, že Overlay extensions položili základ *Restartless* a *Add-on SDK* rozšíreniam[12].

Restartless extensions sú špeciálne rozšírenia, ktoré na rozdiel od overlay extensions, nepoužívajú XUL overlay pre tvorbu užívateľského rozhrania. Tieto rozšírenia sa programovo vkladajú do aplikácie Firefox. To je uskutočnené pomocou špeciálneho skriptového súboru, ktorý je súčasťou rozšírenia. Tento skript obsahuje funkcie volané prehliadačom, ktoré sú schopné vykonať aby sa rozšírenie nainštalovalo, odinštalovalo, spustilo alebo vyplo. Všetko, čo prehliadač vykonáva je volanie tohto skriptového súboru; za pridávanie a odoberanie z užívateľského rozhrania alebo vypnutie a zapnutie je zodpovedné už rozšírenie a nie prehliadač[13]. Sú tak isto známe pod menom *Bootstrapped Extensions*.

WebExtensions API je nový spôsob ako vytvárať rozšírenia, ktorý ešte oficiálne nebol vydaný. Za ciele si kladie *ľahší prevod rozšírení* z iných prehliadačov, *jednoduchšie schvaľovanie rozšírení*, WebExtensions musia byť *kompatibilné s multiprocess Firefox*, *zmeny v kóde Firefox* (napríklad pri aktualizácii), by nemali rozhodiť chod rozšírenia a *ľahšie používanie*. Väčšina API je podobná API, ktoré využíva aj Google Chrome a Opera. To znamená, že prevod rozšírenia pre Opera alebo Google Chrome bude omnoho jednoduchší. Bude stačiť porovnať funkcie v API, ktoré WebExtensions podporujú a po jednoduchých úpravách doplnok pridať[14].

[11] Overlay XUL sú súbory, ktoré sa používajú pre popis navyše obsahu v užívateľskom rozhraní. XUL je jazyk založený na XML pre budovanie rozhraní

[12] https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL

[13] https://developer.mozilla.org/en-US/Add-ons/Bootstrapped_Extensions

[14] https://wiki.mozilla.org/WebExtensions

Add-on SDK extensions je ďalší a v súčasnosti odporúčaný spôsob ako pre Mozillu Firefox tvoriť doplnky. Pri tvorbe sa môžu využívať rôzne štandardné webové technológie ako JavaScript, HTML a CSS. SDK[15] obsahuje API, pomocou ktorého môže programátor tvoriť doplnky a taktiež nástroje pomocou ktorých môže tieto doplnky spúšťať, testovať alebo komprimovať.

Súčasťou Add-on SDK je nástroj príkazového riadku, ktorý je schopný inicializovať, spúšťať, testovať alebo archivovať. Tento nástroj sa nazýva **jpm** a je založený na Node.js. Nahradil starší nástroj s názvom cfx. Najzákladnejšie príkazy, ktoré sa pri vývoji používajú sú *jpm init*, *jpm run* a *jpm xpi*, kde **jpm init** vytvorí kostru pre rozšírenie, inými slovami, vytvorí počiatočné rozšírenie. **jpm run** spustí prehliadač Firefox s rozšírením, ktoré bolo vyvíjané. **jpm xpi** zabalí rozšírenie do archívu s príponou .xpi. Jedná sa o podobný spôsob distribúcie ako v Chrome.

Rozhranie pre programovanie doplnkov sa vo Firefox delí na *Low-Level API* a *High-Level API*:

Low-level API je rozdelené do troch kategórií, *fundamentálne utility* ako kolekcie, mnoho add-ons pravdepodobne používa moduly z tejto kategórie. *Stavebné kamene pre high-level moduly*, tieto sa používajú ak chceme budovať vlastné moduly. *Privilegované moduly*, ktoré povoľujú prácu s oknami alebo HTML requesty[16].

High-level API zahrňujú moduly pre tvorbu užívateľských rozhraní, prácu s webom a interakciu s prehliadačom[17].

Veľmi dôležité pre vývoj doplnkov pre Mozilla Firefox sú znalosti o typoch skriptov, ktoré využíva. Jedná sa o *inú architektúru ako Chrome*, no samozrejme existujú nejaké paralely medzi rozšírením v Google Chrome a Firefox. Skripty, ktoré sa používajú sa delia do dvoch kategórií[18]:

Add-on code je miesto, kde je sústredená hlavná logika rozšírenia, ktoré je implementované. Add-on je implementovaný ako kolekcia jedného alebo viacerých CommonJS[19] modulov. Každý modul je poskytovaný ako skript, ktorý je uložený pod adresárom *lib*. Minimum pri tvorbe doplnku, je potrebné mať jeden modul implementovaný v skripte *main.js*, ďalšie moduly sa importujú pomocou require().

Content skripty sú skripty, ktoré manipulujú webový obsah. Zatiaľ, čo main.js je povinný modul, content skript nie je. Môžu byť uložené vo forme súborov alebo priamo napísané v main.js ako reťazec znakov. **Hlavný rozdiel** medzi content skriptom v Google Chrome a Mozilla Firefox je to, že vo Firefox sú content skripty všetky skripty mimo hlavného addon skriptu main.js. V Google Chrome je za content skript považovaný len skript, ktorý bol špecifikovaný doménou, na ktorej sa má spustiť. Napríklad skript pre popup okno v Google Chrome nie je považovaný na content script.

[15] Software development kit

[16] https://developer.mozilla.org/en-US/Add-ons/SDK/Low-Level_APIs

[17] https://developer.mozilla.org/en-US/Add-ons/SDK/High-Level_APIs

[18] https://developer.mozilla.org/en-US/Add-ons/SDK/Guides/Two_Types_of_Scripts

[19] CommonJS je základná infraštruktúra vo Firefox na ktorej sa budujú rozšírenia, je to modul znovu použiteľného kódu JavaScript

3.4.3 Opera Add-ons

Vývoj rozšírení pre Opera je skoro totožný ako vývoj pre Google Chrome, pretože pri vývoji sa používa ta istá architektúra ako aj pri Google Chrome. Jeden z malých rozdielov je to, že výsledné rozšírenie má po zabalení príponu *.nex*.

3.5 Publikovanie doplnku

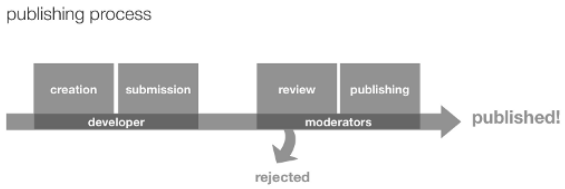
3.5.1 Chrome Web Store

Spoločnosť Google ponúka priestor pre publikovanie doplnkov na takzvanom Chrome Web Store. Jedná sa o online obchod, ktorý hosťuje rozšírenia a aplikácie, ktoré sú zadarmo ale taktiež aj tie, ktoré sú platené. Pre publikovanie je nutný účet vývojára (developer account) cez ktorý vývojár spravuje svoje doplnky alebo aplikácie. Po zverejnení doplnku, môže vývojár svoje rozšírenie aktualizovať novou verziou. Nová verzia sa automaticky stiahne užívateľom, ktorí dané rozšírenie už mali nainštalované. Malou nevýhodou pre developera môže byť zavedenie účtu, ktoré je spoplatnené sumou 5$.

3.5.2 Mozilla Add-ons

Mozilla Add-ons je oficiálny repozitár doplnkov spoločnosti Mozilla Foundation pre software Mozilla, ktorý zahrňuje Mozilla Firefox, Mozilla Thunderbird, Mozilla Sunbird a SeaMonkey. Účet pre vývojára je bezplatný, na rozdiel od Google Web Store. Tak isto ako aj Chrome Web Store, Mozilla Add-ons ponúka správu doplnkov alebo nahrávanie nových verzií. Ďalší rozdiel je to, že doplnok musí prejsť kontrolou cez automatizovaný skript na strane Mozilla Add-ons.

3.5.3 Opera - addons



Obr. 3.5: Proces publikovania doplnku pri platforme Opera[20]

Opera addons ponúka možnosť publikovať doplnky pre webový prehliadač Opera. Proces publikovania je trochu iný ako u predošlých repozitárov respektíve obchodov. Na rozdiel od ostatných centier pre rozšírenia, nestačí jednoduché publikovanie. Po nahraní najnovšej verzie doplnku, musí vývojár počkať určitú dobu (väčšina doplnkov je schválená do dňa), na schválenie moderátormi v Opera centre. Po úspešnom schválení je doplnok zverejnený a dostupný pre užívateľov na ďalšie použitie. Ak sa moderátorom zdá doplnok výnimočný a zároveň splňuje funkcionalitu, tak ho môžu moderátori zaradiť do kategórie s názvom *Odporúčané*.

[20] https://dev.opera.com/extensions/publishing-guidelines/

Kapitola 4

Návrh rozšírenia

Po objasnení situácie na trhu rozšírení a vysvetlení teoretického základu bude v tejto kapitole predstavená analýza zadania a konkrétne možnosti vývoja doplnkov pre prehliadače.

4.1 Analýza zadania

Zadaním bolo vytvoriť rozšírenie pre webové prehliadače, konkrétne minimálne pre Firefox a Google Chrome. Bolo potrebné zistiť aktuálnu situáciu na trhu doplnkov pre prehliadače a následne previesť ich porovnanie. Rozšírenie má detegovať multimediálny obsah a cez akciu, respektíve činnosť užívateľa ho nechá spracovať službou Audeliver.com. Rozšírenie by malo byť čím viac robustnejšie, čiže by malo detegovať čo najviac typov obsahov stránok. Zo zadania taktiež plynie, že dôležitou súčasťou úspešného riešenia je analýza užívateľskej skúsenosti (UX[1]). Tým pádom som sa rozhodol návrh rozčleniť do dvoch kategórií a to, užívateľské rozhranie a back-end.

4.2 Užívateľské rozhranie

Pri návrhu užívateľského rozhrania, som si najprv zistil informácie o rozdieloch pri vývoji rozhrania u jednotlivých prehliadačoch (Google Chrome, Mozilla Firefox, Opera). Keďže všetky prehliadače podporujú HTML a CSS pri vytváraní rozhrania, malo by sa jednať o prácu, ktorá po dokončení bude funkčná na všetkých prehliadačoch.

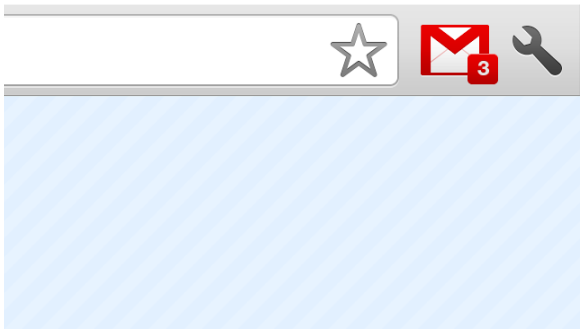
Zo zadanie plynie, že rozšírenie má detegovať obsah. Následne tento zdetegovaný obsah má, pričinením užívateľa, spracovať služba na inom serveri. To značí, že na tento obsah musí byť užívateľ nejakým spôsobom upozornený. Po preskúmaní iných rozšírení som sa rozhodol o upozornenie užívateľa cez takzvaný badge alebo odznak, ktorý sa objaví na ikone rozšírenia (viď. obrázok 4.1).

Tento odznak bude udávať hodnotu o počte nájdených streamov alebo iných audio/video odkazov. Ak hodnota bude väčšia ako 0, tento odznak sa objaví a tým pádom bude užívateľ oboznámený s tým, že na aktuálnej stránke existuje multimediálny obsah. To prirodzene bude smerovať k tomu, že užívateľ klikne na danú ikonu a zobrazí si streamy, ktoré sa našli.

Nájdené streamy, respektíve užívateľské rozhranie bude v Chrome terminológii implementované ako *popup* okno. Popup sa objaví po kliknutí na ikonu, a jeho jadro je tvorené HTML obsahom (samozrejme sú povolené aj iné technológie) (viď. obrázok 4.2). V Google

[1] User experience

[2] https://chrome.google.com/webstore/detail/google-mail-checker/mihcahmgecmbnbcchbopgniflfhgnkff



Obr. 4.1: Rozšírenie s upozornením[2]

Chrome sa popup automaticky responzívne prispôsobuje. Vo Firefox terminológii je popup nazvaný ako *panel*. Panel sa vytvorí cez konštruktor Panel(). Panely alebo popup okná sú vhodné ako dočasné užívateľské zohrania. Popup alebo panel, sa po každom kliku načítava stále na novo, preto sa jedná o dočasné rozhranie. V tomto doplnku budem potrebovať imitovať užívateľovi situáciu, kde rozhranie je permanentné aj keď technicky bude stále na novo načítavané. Keďže Opera zdieľa väčšinu rozhrania s rozhraním, ktoré používa aj Google Chrome, vývoj v Opere bude skoro totožný ako aj v Chrome.

Obr. 4.2: Popup okno[3]

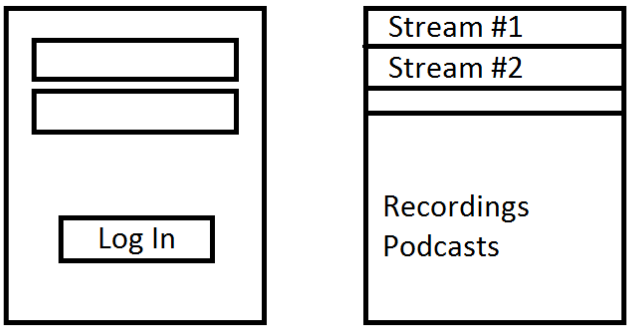
Funkcionalita užívateľského rozhrania bude zahrňovať prihlásenie a odhlásenie. Užívateľ si bude schopný zobraziť svoje nahrávky, s ktorými bude môcť manipulovať v zmysle presúvania do iných podcastov alebo mazania. Tak isto si užívateľ bude môcť zobraziť svoje podcasty, ktoré budú odkazovať na webovú stránku podcastu, ak sa bude jednať o verejný podcast. Ďalej bude môcť vytvárať a taktiež mazať vytvorené podcasty. V neposlednom rade, bude užívateľ schopný po detekcii streamu, pridať nájdené médium do svojich nahrávok.

Nasledujúce obrázky predstavujú prvotné návrhy rozmiestenia užívateľského rozhrania (obrázok 4.3) :

4.3 Analýza API

Návrh ako by mohli fungovať funkcie rozšírenia z programátorského hľadiska respektíve backend, začal po zverejnení rozhrania pre programovanie aplikácie (API) serveru *Audeli-*

[3] https://chrome.google.com/webstore/detail/google-mail-checker/mihcahmgecmbnbcchbopgniflfhgnkff



Obr. 4.3: Prvotný návrh pre prihlasovaciu obrazovku a zobrazenie streamov

ver.com. Ďalším krokom bola komplexná analýza tohto aplikačného rozhrania.

API s názvom *Swagger UI* poskytuje drvivú väčšinu funkcií, ktoré poskytuje aj webové rozhranie web stránky Audeliver.com. Jedná sa o funkcie napríklad ako mazanie nahrávok, tvorba podcastu alebo upravovanie užívateľského profilu. Celkovo je rozhranie Swagger UI rozdelené do troch kategórií:

- **User** - táto kategória zahrňuje funkcie spojené s profilom používateľa. Konkrétne sa jedná o funkciu pre zobrazenie dát o užívateľovi, cez ďalšiu funkciu je schopný pomocou API vytvoriť nového užívateľa a poslednou funkciou sme schopný upraviť respektíve vykonať update profilu užívateľa.
- **Recordings** - táto kategória disponuje piatimi operáciami, ktoré sme schopní použiť. Jedná sa o operácie ako zobrazenie dostupných nahrávok, tvorba novej nahrávky, vymazanie nahrávky zadaním identifikačného čísla, pod ktorým je nahrávka uložená, zobrazenie špecifickej nahrávky podľa identifikačného čísla a úprava respektíve update informácií o konkrétnej nahrávke taktiež zadaním čísla identifikácie na serveru.
- **Podcasts** - jedná sa o poslednú kategóriu v ktorej existuje päť operácií. Prvá operácia je zobrazenie podcastov, druhá je tvorba nového podcastu, tretia je zmazanie podcastu podľa identifikačného čísla, táto operácia zmaže okrem podcastu aj všetky nahrávky, ktoré boli jeho súčasťou, štvrtá funkcia je zobrazenie podcastu podľa čísla identifikácie a v neposlednom rade je tu piata operácia v ktorej je užívateľ schopný upraviť dáta alebo informácie o podcaste.

Všetky spomenuté operácie sú realizované pomocou HTML Request metód a to GET, POST, PUT a DELETE.

- **GET** - Metóda GET po zavolaní obdrží ľubovoľné informácie (vo forme entity), ktoré sú identifikovateľné podľa URI[4] žiadosti. Metoda GET bola navrhnutá na získavanie informácií zo serveru, na rozdiel od účelu metódy POST.
- **POST** - V metóde POST sa informácie o žiadosti nepodávajú pomocou URI, ale v tele žiadosti. Žiadosť s dátami musí server prijať a tak isto server musí vedieť ako skladovať prijaté dáta. Často sa táto metóda používa pri nahrávaní nového súboru na server alebo pri potvrdení vyplneného formulára, ktorý chceme zaslať na server.

[4]Uniform Resource Identifier

- **PUT** - Táto metóda žiada aby zaslané informácie boli uložené podľa hodnoty URI žiadosti. Ak dáta uložené pod zaslanou URI už na serveru existujú, tak sa zaslané informácie budú považovať za modifikovanú verziu starých dát. Ak dáta na serveru pod zaslanou URI neexistujú, metóda PUT bude schopná vytvoriť nový záznam pod hodnotou konkrétnej URI.

  Rozdiel medzi PUT a POST je v odlišnom význame Request-URI alebo URI žiadosti. URI v POST identifikuje zdroje, ktoré budú prijímať zaslané dáta. Bude sa jednať o proces, presmerovanie na iný protokol alebo nejaký podobný proces. URI v PUT identifikuje priamo entitu, ktorá sa bude ovplyvňovať. Server v tomto momente nesmie priradiť zdroje inej metóde.

- **DELETE** - Metóda DELETE žiada server o zmazanie určitých informácií, ktoré sú špecifikované pomocou URI žiadosti. Klient nie je schopný vedieť na istotu, či sa akcia mazania na serveru vykonala. Server avšak môže spätne zaslať informácie o mazaní, čo bude indikovať klientovi, že mazanie sa uskutočnilo úspešne.

Pri zaslaní žiadosti ľubovoľného typu na server Audeliver.com, musia všetky žiadosti prejsť autorizáciou. Server na autorizáciu používa základnú HTTP autentizáciu k prístupu alebo *HTTP Basic access authentication*, za pomoci protokolu HTTPS.

HTTP Basic access authentication je jedná z najjednoduchších autentizačných metód. Samostatne bez použitia iných pomocných technológií je táto metóda bezpečnostne slabá. Nechráni zaslanú entitu, a tak sú tým pádom informácie pri zaslaní reprezentované v čistom texte. HTTP neponúka dodatkové technológie pre autentizáciu alebo mechanizmy pre enkrypciu zasielaných dát. Najzásadnejší nedostatok v *Basic authentication* je spomínané zasielanie dát v čistom texte, čo sa môže prejaviť taktiež zaslaním hesla, ktoré sa potom dá ľahko detegovať pri sledovaní sieťového prenosu. Preto by sa táto autentizačná metóda nemala používať pri prenášaní takýchto citlivých dát[5].

Tento problém s bezpečnosťou na serveru je riešený pomocou *HTTPS protokolu* a zakódovaním pomocou kódovania *Base64*.

HTTPS alebo Hypertext Transer Protocol Secure je rozšírením protokolu HTTP (spomínaný v kapitole 3.1.1), ktorý je internetový komunikačný protokol chrániaci integritu a dôveryhodnosť užívateľských dát medzi serverom a užívateľským počítačom. Užívateľ v dnešnej dobe očakáva vysokú bezpečnosť a preto sa odporúča používať HTTPS všade tam, kde sú zadávané citlivé informácie. Dáta posielané pomocou HTTPS sú zabezpečené cez *Transport Layer Security* protokol, ktorý poskytuje tri vrstvy bezpečnosti a ochrany[6] :

- **Šifrovanie** - Šifrovanie informácií zabezpečuje to, že zatiaľ čo si užívateľ prehliada web stránky, nikto z vonkajšej strany nemôže v tom momente sledovať jeho konverzácie, aktivity alebo ukradnúť informácie o užívateľovi.

- **Integrita dat** - Integrita zabezpečuje to, že informácie nebudú behom prenosu pozmenené alebo poškodené, či už účelne alebo nie. Ak sa dáta behom prenosu pozmenia, bude to zdetegované.

- **Overenie** - Autentizácia dokazuje, že užívateľ komunikuje priamo s web stránkou s ktorou aj chcel komunikovať. Zabraňuje takzvaným *man-in-the-middle* [7] útokom.

[5] https://tools.ietf.org/html/rfc2617#page-19

[6] https://support.google.com/webmasters/answer/6073543?hl=env

[7] Man-in-the-middle - Jeden z najznámejších problémov v kryptografii a informatike. V základe ide o

Base64 je skupina podobných schém alebo algoritmov, ktoré šifrujú informácie z binárnej formy do textovej. Base64 kóduje vždy tri oktety binárnych dát pomocou štyroch ASCII znakov. Pri kódovaní používa 64 prvkovú znakovú sadu tvorenú veľkými aj malými písmenami anglickej abecedy, číslicami a znakmi ako plus a lomítko. Pokiaľ výsledný zašifrovaný text nie je deliteľný tromi, tak sa na konci textu pridajú jedno alebo dva znaky rovná sa.

Napríklad slovo *Jakub* bude zašifrované ako *SmFrdWI=*. Celkovo platí, že zašifrovaný vstupný text je kratší asi o tretinu oproti výstupnému textu[4].

4.4 Back end rozšírenia

O dizajnovom návrhu užívateľského rozhrania sa zaoberala kapitola XX. Táto časť rozoberá návrh celku, ktorý bežný užívateľ nepostrehne a to je **back end**.

Celkovú problematiku ohľadom back endu aplikácie a vlastne celej funkcionality rozšírenia by som rozdelil do dvoch základných kategórií. Prvou by bolo *vytvorenie funkčného užívateľského rozhrania*, ktoré akoby imituje funkcionalitu webovej stránky Audeliver.com, je schopné napríklad zobraziť nahrávky, pracovať s nimi alebo tvoriť podcasty. Druhá problematika, ktorá by vlastne mala byť vo finálnom riešení podskupinou prvej (vyhľadané streamy sa eventuálne zobrazia v užívateľskom rozhraní), je *funkčnosť vyhľadávacich skriptov*, ich spôsob vyhľadávania a problémy a riešenia spojené s nimi.

Výsledné rozšírenie musí fungovať aj v takej situácii, keď sa na aktuálnej stránke nebude nachádzať žiaden multimediálny obsah. To tým pádom určuje prioritu **najprv sfunkčniť užívateľské rozhranie** a až potom sa zaoberať problematikou streamov. V užívateľskom rozhraní bude jeden z problémov vyriešenie prihlasovania sa a odhlasovania sa.

V mojom návrhu sa užívateľ bude **prihlasovať cez popup okno**. V predošlej kapitole XX, bolo spomínané, že popup okno sa každým kliknutím na ikonu zobrazuje nanovo a tým pádom všetky dáta sú získane až po kliku. Táto situácia vylučuje tým pádom ukladanie užívateľských informácií cez ľubovoľný skript, ktorý by bol importovaný do HTML kódu popup okna. Avšak po štarte rozšírenia sa v Google Chrome a v Opera spúšťa stále bežiaca background page a v Mozilla Firefox je to main.js. To by mohlo jednoducho umožniť **priebeh situácie s prihlasovaním** a vyriešilo by to tento počiatočný problém.

Tejto algoritmus sa bude opakovať pri každom otvorení popup okna:

1. **Užívateľ otvorí popup okno**
2. **Popup okno pomocou skriptu popup.js kontaktuje background page (alebo main.js), kde môžu nastať dva prípady:**
   a) Ak užívateľ nebude prihlásený, zadá prihlasovacie údaje a tým pádom popup skript pošle informácie o prihlásení background page
   b) Užívateľ bude prihlásený, dostane aktuálne dáta (to, či je užívateľ prihlásený zistí požiadaním o dáta od background page. V tomto prípade sa dáta nastavili pri prihlásení)

Po úspešnom prihlásení sa funkcionalita ohľadom klikania na odkazy, tlačidiel alebo textových polí v užívateľskom rozhraní, sústredí do popup skriptu. Ak bude popup skript

snahu odpočúvať konverzáciu medzi minimálne dvomi účastníkmi, a to tým spôsobom, že sa z nezvaného pasívneho účastníka stane aktívny účastník konverzácie.

potrebovať informácie z iných serverov alebo dáta ohľadom nájdených streamov, kontaktuje background page.

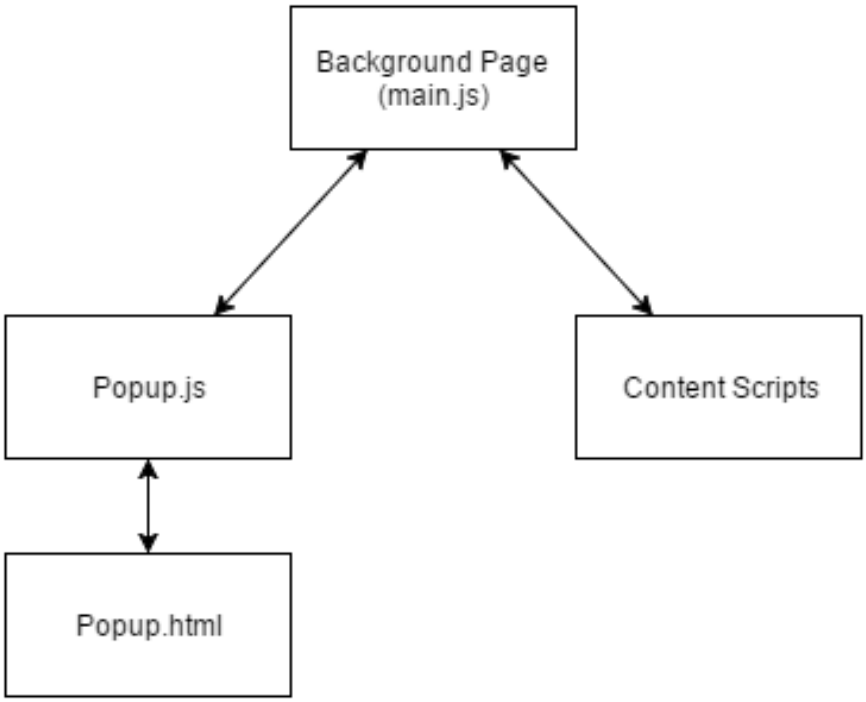
Background page (alebo main.js) bude obsahovať metódy a funkcie, ktoré budú kontaktovať Audeliver.com po vzoru Swagger UI API. Pri používaní uživateľského rozhrania implementujeme iba potrebné HTML metódy zo Swagger UI. Taktiež bude background page udržiavať informácie o prihlásení a odhlásení.

V neposlednom rade bude potrebné vyriešiť problematiku ohľadom odznakov (viď. obrázok 4.1) na ikone rozšírenia. To sa bude taktiež riešiť v background skripte.

Po úspešnom implementovaní užívateľského rozhrania bude ďalší krok, **implementácia skriptov pre vyhľadávanie streamov**. Budú to samostatné súbory, ktoré budú totožné ako pre Google Chrome, tak isto aj pre Mozillu Firefox alebo Opera. Budú rozdelené do dvoch kategórií a to špecializované a všeobecné.

Špecializované skripty budú skripty, ktoré sú špecifické pre daný server a spúšťajú sa iba vtedy ak sa URL aktuálneho okna rovná vzoru, do ktorého sa musí trafiť. Napríklad server soundcloud má REGEXP[8] vzor *https://soundcloud.com/** a to napríklad vyhovuje *https://soundcloud.com/langerhan/j-cole-type-beat-dont-save-her*.

Všeobecné skripty budú skripty, ktoré nie sú špecifické pre servery, tým pádom sa budú spúšťať stále. Budú vyhľadávať v HTML kóde stránky HTML5 tagy ako audio alebo video.



Obr. 4.4: Diagram zobrazujúci zovšeobecnenú komunikáciu v rozšírení

Celkovo by sa dalo zovšeobecniť a zhrnúť, že backend bude obsahovať tri druhy skriptov a to **popup skript** na manipuláciu a funkcionalitu popup okna, **background page skript**,

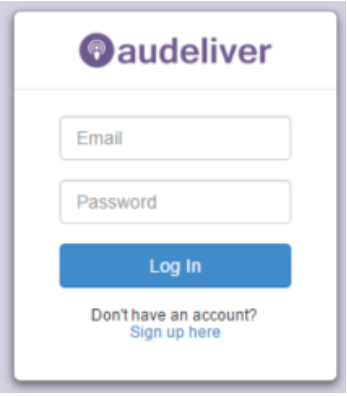
[8]Regular expression - regulárny výraz

ktorý bude držať hlavnú logiku a **content skripty**, ktoré budú získavať dáta o audio/video médiách na aktuálnej stránke. Všetky tieto skripty budú schopné medzi sebou buď priamo alebo nepriamo komunikovať.

Kapitola 5

Implementačná časť

V implementácii som prirodzene vychádzal z návrhu, ktorý som pripravil. Mojím prvým cieľom bolo vytvoriť rozšírenie, ktoré bude zvládať funkcie ako prihlásenie a odhlásenie a taktiež bude mať základný dizajn. Postupom času a vývoja som ďalšie funkcie vytvoril analogicky. V nasledujúcej časti budem vysvetľovať vývoj primárne pre Chrome a Opera, výnimky vo vývoji pre Firefox budú spomenuté taktiež.



Obr. 5.1: Prihlasovacie okno rozšírenia

Základné rozšírenie
Pri prvom základnom rozšírení ešte neexistoval dizajn. To znamenalo, že bolo potrebné vymyslieť štýl, akým sa bude celé rozšírenie uberať. Zvolil som pozadie podobnej farby, ale svetlejšieho odtieňu ako logo Audeliver.com, cez to som dal biely podklad, ktorý drží po hromade celkový obsah popup okna. Na to som pridal Audeliver logo, ktoré je prítomné na každej stránke a vytvára dojem hlavičky. Väčšinu tlačidiel, ktoré značia pohyb po užívateľskom rozhraní ako *prihlásenie* do systému ale *návrat* na predošlú stránku som zvolil modrej farby.

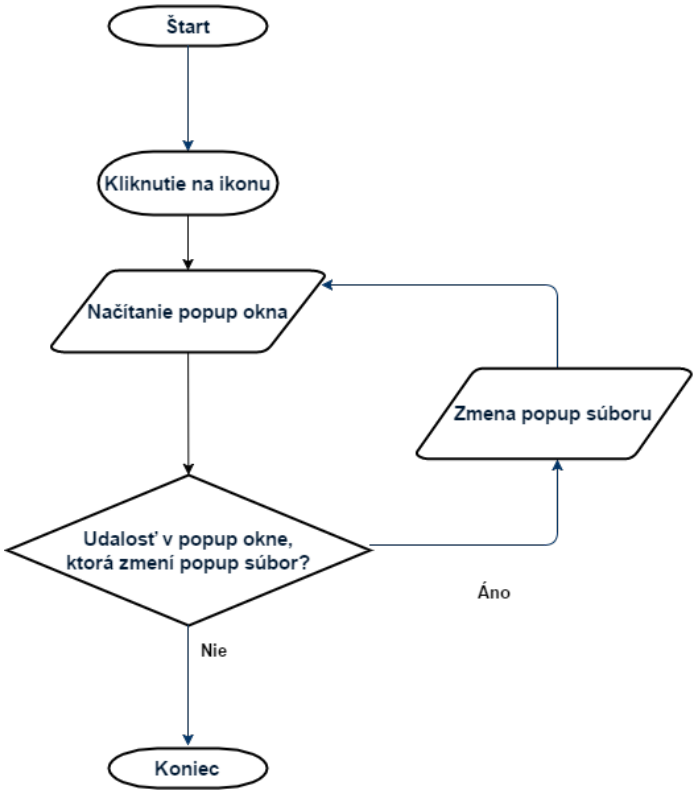
Po navrhnutí základného popup okna bolo potrebné pridať funkcionalitu pre prihlásenie sa užívateľa do systému. Preskúmaním Audeliver API som zistil, že pri každej žiadosti o dáta prebieha autentizácia. To znamená, že nie je možné vytvoriť dlhodobé spojenie medzi užívateľom a serverom. Pre takéto asynchrónne žiadosti sa v JavaScripte používa časť Jquery a to Ajax alebo v čistej verzii XMLHttpRequest.

XMLHttpRequest je API, ktoré poskytuje klientovi funkcionalitu pre prenos dát medzi klientom a serverom. Poskytuje jednoduchú cestu ako obdŕžať dáta z URL bez

nutnosti plného znova načítania stránky. To umožňuje web stránke aktualizovať časť obsahu bez narušenia činnosti užívateľa. XMLHttpRequest sa vo veľkej miere využíva v programovaní AJAX. Napriek názvu, XMLHttpRequest môže prijímať aj iné typy dát ako XML[1].

To či užívateľ zadal platné meno a heslo som implementoval pomocou metódy **GET user** zo Swagger UI cez XMLHttpRequest. Ak asynchrónna funkcia pošle spätne klientovi užívateľské informácie, jedná sa o úspešné prihlásenie, ak pošle chybný stav, jedná sa o zlé zadané heslo alebo problém so serverom. XMLHttpRequest nefungoval dokiaľ sa neupravila hlavička zaslanej žiadosti. Originálna curl žiadosť z Audeliver API obsahuje dve dodatkové informácie pre hlavičku, bez ktorých neprebehne spojenie a to `Accept: application/xml` a `Authorization: Basic base64 hodnota (meno : heslo)`. Každá žiadosť v tomto rozšírení obsahuje tieto dodatkové informácie.

Odpoveď zo servera je vo formáte XML, čo nie je pre bežného užívateľa najvhodnejší formát. To vynucuje vytvoriť funkcie ktoré tieto odpovede spracujú, prípadne sformátujú do prijateľnejšieho formátu ako napríklad HTML.



Obr. 5.2: Flowchart zmeny súboru, ktorý sa zobrazuje v popup okne

Po úspešnom prihlásení, popup skript obdrží informácie o tom, že prihlásenie prebehlo úspešne a zmení defaultný súbor popup na súbor frontpage, kde sa nachádza užívateľské menu a nájdené streamy. Túto zmenu obdrží aj background skript. Teraz, sa pri znova kliknutí na ikonu popup okna, už nebude zobrazovať stránka s prihlásením ale **pozmenená stránka**, v tomto prípade frontpage.html.

[1] https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

Takto **analogicky fungujú všetky funkcie v background skripte**. Skript odpočúva a čaká na správy z popup.js. Po doručení správy, vo väčšine prípadov, požiada server Audeliver.com o dáta, ktoré chcel užívateľ v rozhraní. Nejedná sa len o žiadosti týkajúce sa žiadania informácii, taktiež ide o žiadosť o poslanie alebo modifikovanie informácii na serverovskej strane (napríklad vytvorenie novej nahrávky).

Funkcie implementované v background skripte su rozdelené do dvoch kategórii a to tie, ktoré *žiadajú a spracujú odpovede zo serveru* a tie, ktoré tieto odpovede *sformátujú do prijateľnejšej formy pre užívateľa*. Background skript okrem odpoučúvania popup skriptu (alebo respektíve akcií užívateľa), ešte odpočúva aj aktuálne taby. Ak sa na aktuálnom tabe objaví mediálny obsah, tak tab cez popup skript upozorní background page o nájdenom obsahu, čo bude značiť aktualizáciu hodnoty odznaku na ikone rozšírenia.

Skripty na vyhľadanie audio/video obsahu

Ďalšou nevyhnutnou časťou rozšírenia sú skripty na vyhľadávanie audio a video obsahu. Pre každý podporovaný server som implementoval samostatný skript, pretože každý mediálny server ma iný spôsob ukladania audia alebo videí. Niektoré videá sú prístupné cez API, iné sa nachádzajú v HTML kóde stránky. Tu bolo nutné čítať zdrojové HTML kódy stránok, hľadať kľúčové slová ako *mp4, video, flv a podobné*. Dalo by sa povedať, že táto časť spolu s portovaním na Firefox bola časovo najviac náročná.

Servery ako Vid.Me a Dailymotion mali cesty k mediálnym súborom ukryté v HTML súbore videa pod *určitým tagom* (väčšinou script), kde sa nachádzal nejaký kus neinterpretovaného JavaScript kódu alebo správa vo formáte JSON. K týmto dátam bolo potrebné pristupovať na začiatku ako k čistému textu, neskôr po jednoduchých úpravách sa väčšinou tento text musel parsovať do formátu JSON, s ktorým sa už dalo v JavaScripte pracovať.

Vimeo neposkytuje takýto spôsob ukladania URL videa (skryté v HTML) ako Vid.Me a Dailymotion. K získaniu linkov sa muselo pristúpiť cez *špecifickú API*, ktorá po žiadosti navrátila správu vo formáte JSON. Soundcloud.com zase neposkytoval verejnú API vôbec, preto bolo potrebné sa registrovať ako bežný užívateľ a vyžiadať si kľúč k používaniu API.

Zvyšok serverov bol prehliadavaný na tag *audio* a *video*, kde sa získavali atribúty *source* týchto tagov.

Zmeny v Mozilla Firefox

Pri vývoji pre Mozilla Firefox bolo potrebné zmeniť niektoré funkcie z Chrome, ktoré neboli natívne podporované. Pre vývoj som si vybral vývoj pod Add-on SDK z dôvodu, že je to v súčasnej dobe najodporúčanejšia metóda od Mozilla Firefox. Najväčšia zmena nastala po zistení, že hlavný skript, ktorý drží logiku rozšírenia, **nepodporuje cross-domain** dotazy na iné servery. To znamenalo presunutie veľkej časti logiky do popup okna, čo taktiež spomalilo chod rozšírenia. Najviac spomalilo chod rozšírenia to, že vo Firefoxe je odznak na ikone rozšírenia v experimentálnej fáze, to znamená, že neregistruje aktuálne aktívne okno ako v Chrome a tak akonáhle užívateľ preklikne na iný tab, prebieha kontrola hodnôt odznakov na všetkých taboch, čo už pri vyššom počte tabov spôsobuje problémy. Preto **odporúčam testovať funkčnosť riešenia v Chrome alebo Opera**. Riešenie v Mozilla Firefox je funkčné ale mnohonásobne viac pomalšie.

Dobrá správa je, že o pár mesiacov sa Firefox pokúsi prejsť na WebExtensions, kde je vývoj čistejší a zároveň bude podporovať Chrome API ako to podporuje Opera.

Zmeny v Opera

Pri vývoji nenastali žiadne zmeny, pretože Opera podporuje Chrome API.

Kapitola 6

Testovanie

Testovanie prebiehalo v dvoch formách. Prvá časť sa týkala testovania užívateľského rozhrania, druhá časť bola zameraná na testovanie skriptov pre nájdenie médií. Testovalo sa na 6 užívateľoch a X skriptov.

6.1 Priebeh testovania

Užívateľské rozhranie sa testovalo zadaním úloh pre užívateľov, položením otázok formou dotazníka vrátene napísania pripomienok. Beta-testeri dostali funkčné rozšírenie, v ktorom content script vedel rozoznať média iba zo servera Soundcloud. V úlohách sa mali **prihlásiť do rozšírenia** (prihlasovacie údaje som im pripravil skôr, pre ušetrenie času), mali si **prezrieť nahrávky**, **zmazať jednu nahrávku**, **vytvoriť podcast**, načítať web stránku zo servera soundcloud a **pridať nájdený stream do podcastov**. S prihlásením a prezeraním nahrávok alebo podcastov nemal problém žiadny tester. Taktiež mazanie nahrávok bolo bez problémov. Tvorbu nového podcastu zvládlo 83% testerov a pridávanie novej nahrávky zvládlo 50% testerov. Chyby pri tvorbe nového podcastu nastali len jednému užívateľovi. Chyby pri pridaní novej nahrávky zo soundcloud nastali v dvoch prípadoch tak, že užívateľ mal stále zobrazené okno s podcastmi a tak si myslel, že stream sa mu objaví v tomto okne. Druhá chyba bola neznámeho charakteru pravdepodobne na užívateľskej strane, pretože rozšírenie začalo fungovať po reštarte Google Chrome. Pri pripomienkach alebo navrhovaných vylepšeniach, piati zo šiestich súhlasili aby rozšírenie bolo funkčné aj bez prihlasovania a registrácie. To však momentálne nie je možné, pretože toto rozšírenie je priamo naviazané na registrovaný účet, na ktorom sú nahrávky, podcasty špecifické pre užívateľa.

Testovanie skriptov na vyhľadávanie obsahu prebiehalo vybratím skriptu a testovacích dát. Testovala sa reakcia vyhľadávacích skriptov na danú url. Pre každý skript boli pripravené testovacie dáta vo forme odkazov, pričom sa vo výsledku vypočítala percentuálna úspešnosť skriptu.

Pri testovaní sa vybrali náhodné odkazy na videá z podporovaných serverov. Zistilo sa, že pri bežných odkazoch väčšinou fungovali. Pri jednom odkaze na soundcloud.com,

	Soundcloud	**Vimeo**	**Vid.me**	**Dailymotion**	**TedTalks**
Úspesnosť	9/10	9/10	10/10	10/10	9/10

Tabuľka 6.1: Porovnanie funkčnosti skriptov na vyhľadávanie multimédií

sa jednalo o autorsky chránený obsah, pri Vimeo nastala taká istá situácia. Pri TedTalks nastal problém pri odkaze, ktorý smeroval na zoznam prednášok na stránke Ted.com.

Celkovo testy vyvodili informácie ako o užívateľskej skúsenosti pri tomto rozšírení, ktoré sa ďalej použili pri oprave chýb, to isté sa môže povedať tiež o teste skriptov.

Kapitola 7

Záver

Cieľom tejto práce bolo vytvorenie rozšírenia pre rôzne platformy prehliadačov, kde bolo hlavnou métou detekcia multimediálneho obsahu a zároveň zjednodušenie práce so serverom Audeliver.com, ktorý poskytoval API pre funkčné rozšírenie. Táto práca sa implementovala v jazykoch HTML, CSS a JavaScript. Po implementácii boli vykonané testy špecializované na užívateľskú prácu s rozšírením a taktiež testy, ktoré sledovali úspešnosť detekcie multimediálneho obsahu.

Práca na rozšírení začala analýzou existujúcich rozšírení, z ktorých po porovnávaní bolo usúdené, kde je na trhu diera a čo by mohlo toto rozšírenie vyriešiť. Následne bola prevedená analýza technológii s ktorými súvisel vývoj rozšírenia. Boli preskúmané webové prehliadače a ich rozdiely, predstavené základné servery, ktoré rozšírenie podporuje. Postupne boli analyzované spôsoby vývoja rozšírení pre jednotlivé prehliadače a taktiež bola preskúmaná publikácia doplnkov na oficiálnych internetových obchodoch.

Prvotný návrh riešenia začal analýzou zadania, čo neskôr pokračovalo k preskúmaniu API serveru Audeliver a k návrhu ako užívateľského rozhrania, tak isto aj back endu. Po návrhu nasledovala implementácia, kde sa využili všetky získané informácie od rozdielov pri vývoji pre jednotlivé prehliadače po prácu so servermi.

Po ukončení implementácie, boli rozšírenia nasadené do špecializovaných obchodov, kde boli dostupné užívateľom k testovaniu (Chrome Web Store[1], Mozilla Add-ons[2], Opera[3], prípadne pod kľúčovým slovom audeliver). Testovanie odhalilo pri práci užívateľa rôzne problémy ohľadom užívateľského zážitku. Druhá časť testovania sledovala úspešnosť skriptov, ktorá bola v celku pozitívna.

V budúcnosti pri tejto práci by sa vývojár mal zaoberať podporou viac serverov a zároveň údržbou tých predošlých. Mal by dbať na pripomienky užívateľov a tým pádom sledovať trendy (prispôsobovať sa trhu). Vylepšenie tejto práci by malo podporovať aj zmeny obsahu na stránkach, pretože sa čím ďalej viac serverov sústreďuje na vývoj webu, kde sa nemení celá stránka ale iba jej obsah (neruší to užívateľa). Služba Audeliver by mohla nadviazať oficiálny stály kontakt so servermi, ktoré poskytujú multimediálny obsah (zatiaľ rozšírenie funguje na voľne dostupných API, ktoré avšak nemusia byť stále).

[1] https://chrome.google.com/webstore/detail/audeliver/cefnfaaefmgieoiffljjaloempibogkn
[2] https://addons.mozilla.org/cs/firefox/addon/audeliver/
[3] https://addons.opera.com/en/extensions/details/audeliver/

Literatúra

[1] Elizabeth Castro, B. H.: *HTML5 a CSS3*. Computer Press, 2012, ISBN 978-80-251-3733-8.

[2] Fried, I.: *Be sure to read Chrome's fine print [online]*. CNET, 2008-10-07 [cit. 2016-05-17], [Online; navštívené 17.5.2016].
URL http://www.cnet.com/news/be-sure-to-read-chromes-fine-print/

[3] Jay, P.: *Curtains for Netscape [online]*. CNET, 2008-02-29 [cit. 2016-05-17], [Online; navštívené 17.5.2016].
URL http://www.cbc.ca/technology/technology-blog/2008/02/curtains-for-netscape.html

[4] Josefsson, S.: *The Base16, Base32, and Base64 Data Encodings [online]*. Network Working Group, 2006-10 [cit. 2016-05-17], [Online; navštívené 17.5.2016].
URL https://tools.ietf.org/html/rfc4648

[5] Zakas, N. C.: *Professional JavaScript for Web Developers, 3rd Edition*. John Wiley & Sons, Inc., 2012, ISBN 978-1-118-02669-4.

Prílohy

Zoznam príloh

Príloha A

Obsah CD

Priložené CD obsahuje:

- plagát
  - priečinok, v ktorom sa nachádza plagát
- source
  - zdrojové súbory
- dokumentacia
  - source - zdrojové súbory pre technickú správu
  - pdf - technická správa vo formáte pdf
- video
  - demonštračné video