Izrada dodatka za preglednik

Kossi, Patrick

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:045982

Rights / Prava: Attribution-NonCommercial 3.0 Unported/Imenovanje-Nekomercijalno 3.0

Download date / Datum preuzimanja: **2024-04-24**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Patrick Kossi

IZRADA DODATKA ZA PREGLEDNIK

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Patrick Kossi

Matični broj: 45976/17–R

Studij: Informacijski sustavi

IZRADA DODATKA ZA PREGLEDNIK

ZAVRŠNI RAD

Mentor:

Matija Kaniški, mag. inf.

Varaždin, veljača 2022.

Patrick Kossi

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

U ovome završnom radu, objasnit će se povijest web preglednika, kada su nastali, kako su se razvijali te njihovo trenutno stanje. Ukratko će biti opisane glavne tehnologije koje su potrebne za razvoj dodataka web preglednika. Nadalje opisat će se mikrokernel uzorak dizajna za razvoj dodataka. Kroz programski kod i metodama modeliranja prikazat će se arhitektura i glavne komponente potrebne za rad dodataka. Web preglednik sadrži svoj API kojem je moguće pristupiti putem dodataka i u radu će biti opisano njegovo korištenje. Dio rada će se baviti načinima pronalaska i ispravljanje grešaka. Dodatak zahtjeva korisničko sučelje te će biti opisano i dizajniranje korisničkog sučelja. Također opisat će se traženje i potvrđivanje dozvola od strane korisnika radi mogućnosti dodatka da upravlja određenim komponentama web preglednika. Napredna prilagodba dodatka prema potrebama korisnika. Spomenut će se DOM i BOM model web preglednika i njihovo korištenje za manipulaciju sadržaja web stranice. Bit će opisani i prikazani razvoj dodatka API-a odabranog preglednika i zaštita od neovlaštenog korištenja. Završni rad se sastoji i od praktičnog dijela u kojem će se razviti dodatak u odabranom web pregledniku.

Ključne riječi: preglednik, dodatak, API, HTML, CSS, JavaScript, razvoj

Sadržaj

1. Uvod…….. 1
2. Metode i tehnike rada 2
2.1. Alati……. 2
2.2. Tehnologije 6
2.2.1. Struktura 6
2.2.2. Prezentacija 7
2.2.3. Korisničko sučelje 8
2.2.4. Aplikacijsko programsko sučelje 10
2.2.4.1. API Chrome dodataka 10
2.2.4.2. WebExtension API 12
3. Web preglednici 15
3.1. Povijest web preglednika 15
3.2. Trenutno stanje web preglednika 17
3.2.1. Sučelje web preglednika 17
3.2.2. Arhitektura web preglednika 19
4. Web dodaci za preglednik 21
4.1. Mikrokernel uzorak 21
4.2. Povijest web dodataka 23
4.3. Korisni web dodaci za preglednik 24
4.4. Vrste web dodataka 25
4.5. Dokumentno objektni model dokumenta i objektni model preglednika 26
5. Razvoj web dodatka za upravljanje lozinkama 27
5.1. Arhitektura dodataka i direktorija 29
5.2. Manifest 31
5.3. Struktura 32
5.4. Prezentacija 33

5.5. Korisničko iskustvo 35
5.5.1. Glavne funkcionalnosti 36
5.5.2. Dodatne funkcionalnosti 41
5.5.3. Pozadinske funkcije 48
5.6. Uklanjanje pogrešaka 48
5.7. Zaštita, objava dodataka i upravitelj zadataka 50
5.8. Testiranje i upravljanje greškama 55
6. Zaključak.. 56
Popis literature 57
Popis slika…. 59
Popis tablica.. 61

1. Uvod

Internet je u zadnjih nekoliko godina sve više prisutan. Korisnici provode sve više vremena na njemu i sve veća je potražnja za novim stvarima, novim web stranicama i sve veća je potreba za nesmetanim „surfanjem po internetu“. Uz to povećao se je i broj stranica kojima korisnici pristupaju i tako svaka stranica traži neki način autentifikacije korisnika. Veliki broj stranica želi samo da se korisnik registrira i prijavi kako bi mogao koristiti njihove usluge. Vrlo često korisnici imaju jednu e-mail adresu i tu istu e-mail adresu koriste na više različitih web stranica. Također korisnici dosta često koriste jednu te istu lozinku za pristup svim tim stranicama, a tu nastaje problem. Teško je pamtiti novu lozinku kod svake registracije, pa korisnici odlučuju koristiti istu lozinku svugdje. To ih čini ranjivim. U trenutku kada neka zlonamjerna osoba dođe do lozinke od jednog korisničkog računa nekog korisnika (na svim ostalim stranicama na kojima je korisnik koristio tu kombinaciju e-mail-a i lozinke) on postaje ranjiv. Iz tog razloga dolazi do potrebe za korištenjem nekog alata za pamćenje svih tih lozinki. Jedan od načina kako to učiniti je uz pomoću preglednika upotrebom dodatka za preglednik. To je bilo razmišljanje koje je dovelo do izrade ovoga rada.

Teorijski dio rada obradit će arhitekturu, aplikacijsko programsko sučelje web preglednika, tehnologije i alati potrebni za razvoj takvog web dodatka za preglednik. Proći će se kroz povijest navedenih tehnologija i načine na koji one omogućuju sam razvoj dodatka. Spomenut će se povijest web preglednika, trenutno stanje web preglednika, kako izgledaju i način na koji rade. Sljedeće obradit će se mikrokernel uzorak dizajna, opisati način rada web dodataka za preglednik i njihova povijest. Nadalje navest će se neki popularni web dodaci za preglednik i vrsta web dodataka za preglednik koje je moguće razvijati. Za kraj obradit će se još i DOM te BOM model web preglednika.

Praktički dio rada opisat će razvoj dodataka i korisničkog sučelja dodatka za preglednik Google Chrome. Izradit će se web dodatak koji će pamtiti lozinke korisnika, kako bi se njemu omogućilo nesmetano uživanje i sigurno korištenje interneta.

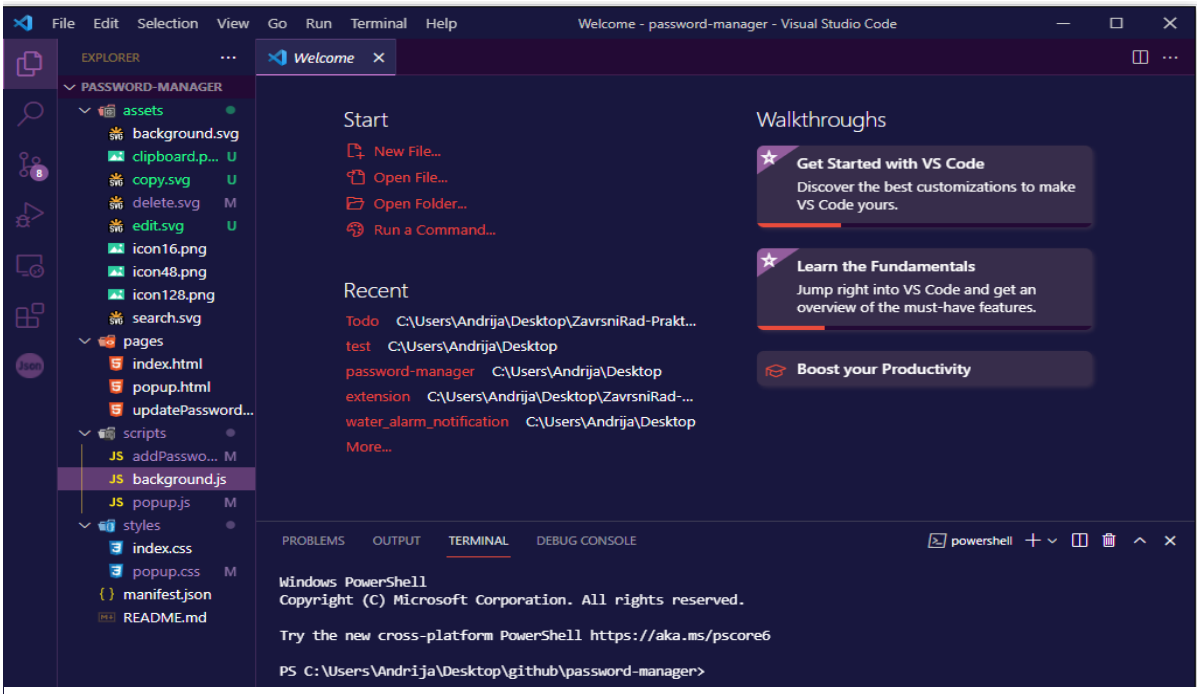
2. Metode i tehnike rada

U ovom poglavlju ukratko će se opisati tehnologije i programski alati koji su bili korišteni za izradu primjera dodataka za web preglednik.

Web dodatak ili drugim riječima proširenje je „mala aplikacija“ koja pregledniku dodaje određenje funkcionalnosti i značajke. Za web dodatak postoje dvije definicije a to su umetak (*eng. Plugin*) i proširenje (*eng. Extension*). Glavna razlika je u tome što su proširenja gotovo uvijek samo izvorni kod koji se dodaje u preglednik. Umetak se unaprijed kompilira i u preglednik se dodaju izvršne datoteke (eng. *Executables*). Za umetke treba više vremena da se razvije jednostavna aplikacija i imaju više sigurnosnih problema od proširenja. Radi toga popularni web preglednici od 2021. godine ne podržavaju umetke, već samo proširenja. Kroz ovaj rad kada se kaže web dodatak misliti će se na proširenje.

2.1. Alati

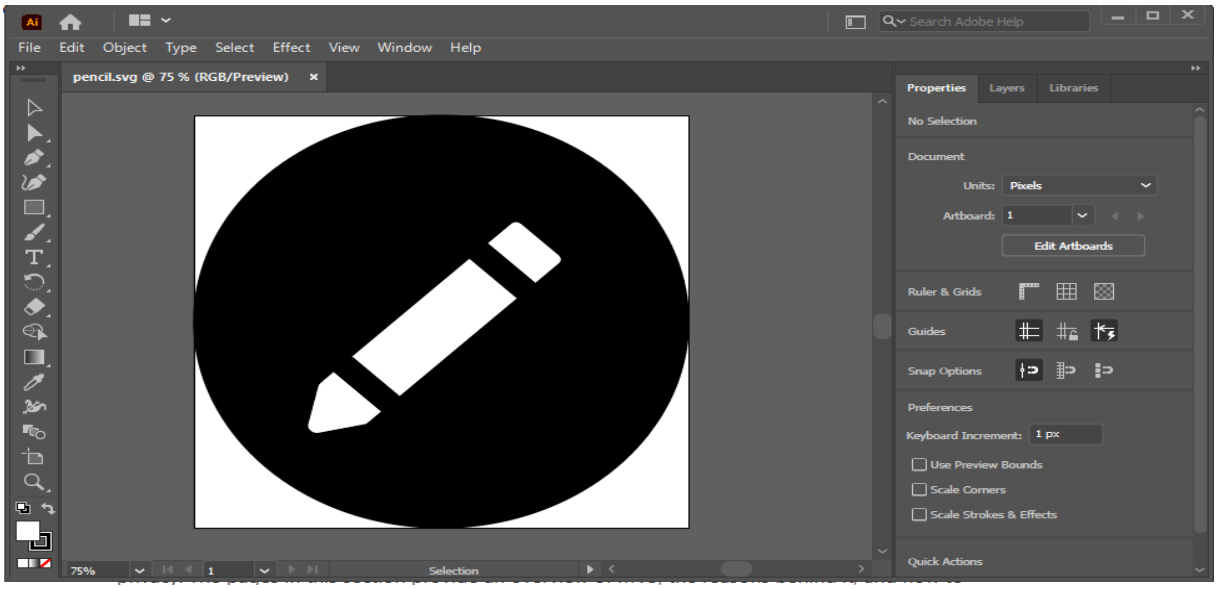
Visual Studio Code - integrirano razvojno okruženje i moćan uređivač izvornog koda za operacijske sustave: Windows, MacOS i Linux. Ima ugrađenu podrška za JavaScript, TypeScript i Node.js te podržava korištenje i drugih jezika.



Slika 1: Sučelje alata Visual Studio Code

Moguće ga je preuzeti na ovoj web stranici „https://code.visualstudio.com/Download“. U ovom radu Visual Studio Code (kraće VSCode ili Code) koristiti će se za razvoj dodatka. Na slici 1 vidi se sučelje alata.

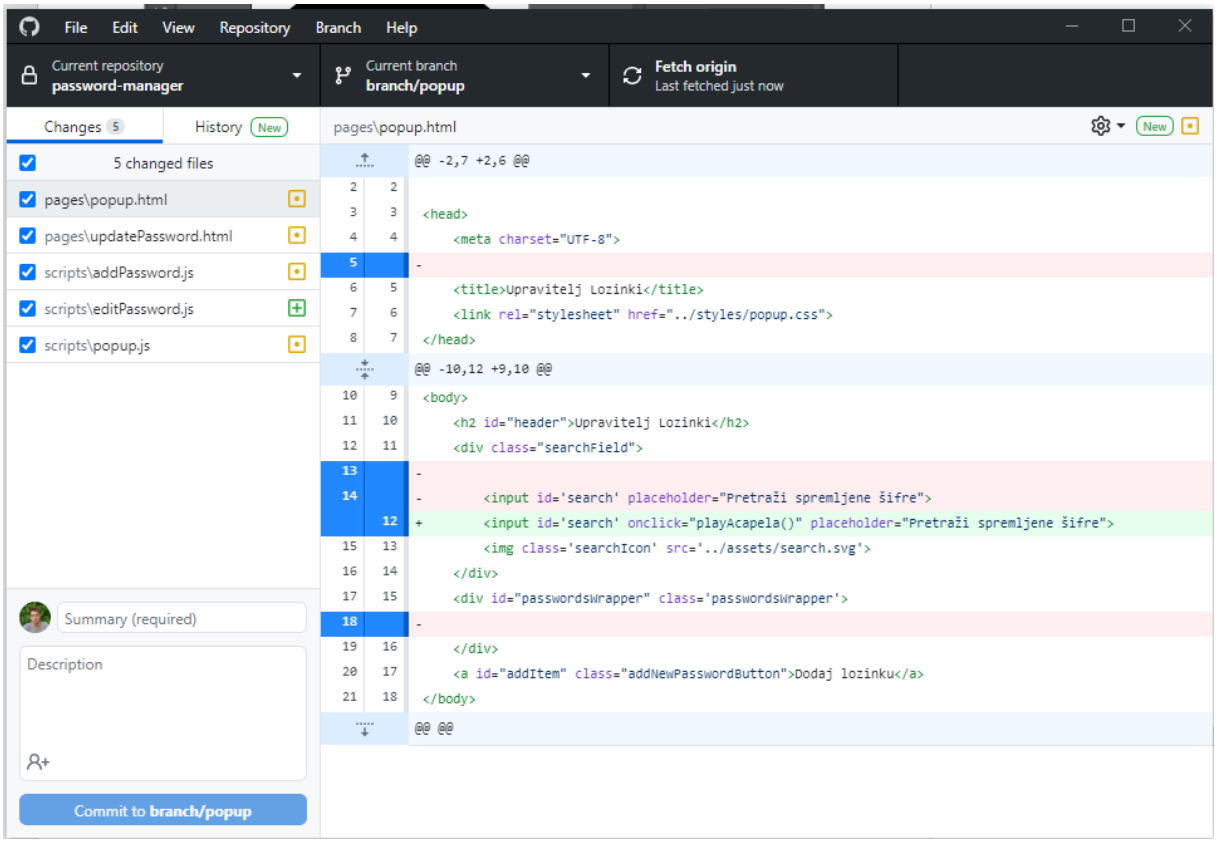
Adobe Illustrator 2021 - profesionalna je aplikacija za dizajn i crtanje zasnovan na vektorima. Illustrator omogućuje kreiranje pojedinačnih elemenata dizajna, dijagrame, grafove, logotipe i ostalo. U ovom radu je korišten za kreiranje pozadinskih slika za web dodatak te za kreiranje ikona od aplikacije dimenzija: 16x16px, 48x48px, 128x128px. Omogućuje spremanje datoteka u png (eng. *Portable Network Graphics*) formatu, koji je koristan kada se želi prikazati ikona koja nema vlastitu pozadinu. Isto tako omogućuje spremanje datoteka u svg (eng. *Scalable Vector Graphics*) formatu, koji je odličan za dizajniranje ikona i web stranica koje primjenjuju responzivan dizajn.



Slika 2: Sučelje alata Adobe Illustrator

Adobe Illustrator 2021 je moguće preuzeti sa ove poveznice https://www.adobe.com/products/illustrator.html. Na slici 2 prikazano je korisničko sučelje aplikacije Illustrator. Također se vidi jedna od ikona tijekom izrade. Na lijevoj strani ekrana nalaze se svi alati koji su potrebni za uređivanju i izradu ikone, a s desne strane se vide dodatne postavke za svaki dodani element.

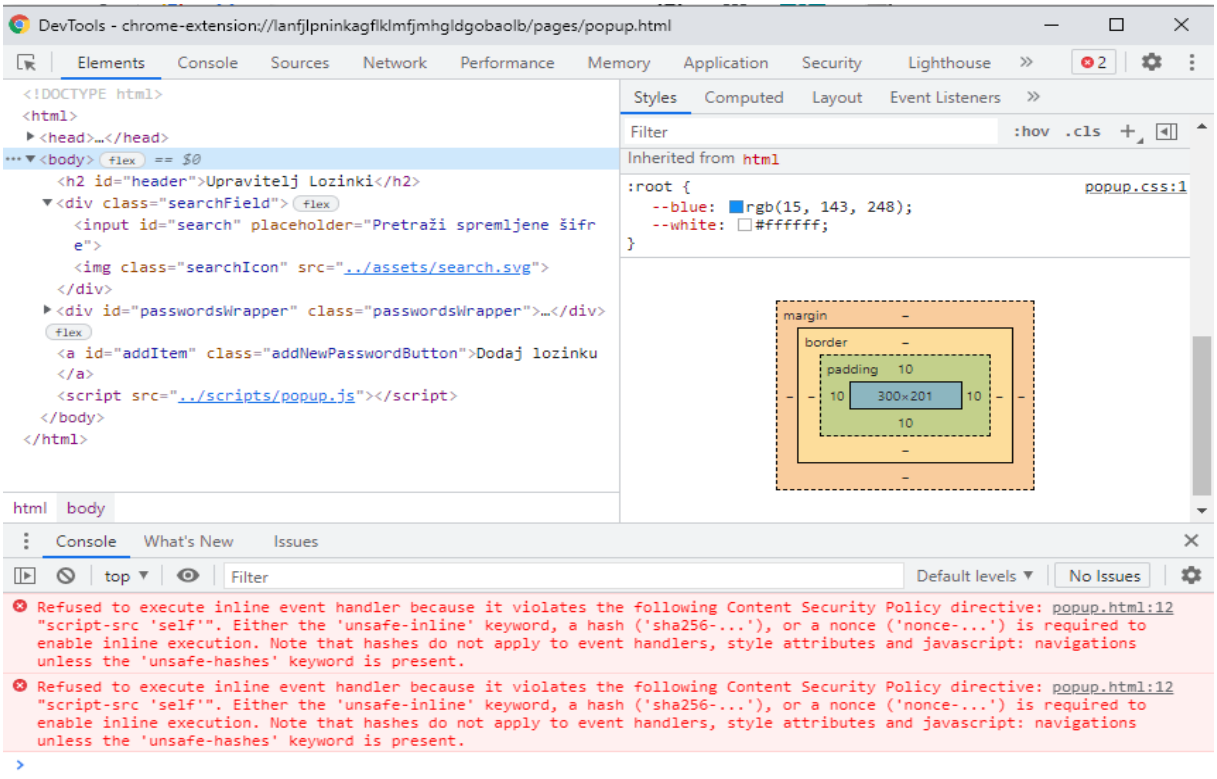
GitHub Desktop - aplikacija koja omogućuje interakciju s GitHub repozitorijem koristeći GUI (eng. *Graphical User Interface*) umjesto naredbenog retka ili web preglednika. U ovom radu je korištena za verzioniranje i stvaranja sigurnosne kopije izvornog koda. Alat je moguće preuzeti sa poveznice „https://desktop.github.com/“.



Slika 3: Sučelje alata GitHub Desktop

Na slici 3 prikazano je korisničko sučelje aplikacije, sa inicijaliziranim repozitorijem web dodatka napravljenog u sklopu ovog rada. Na lijevo strani vide se datoteke unutar kojih su napravljene promjene od zadnje spremljene verzije. Na centralno desnoj strani vide se koje su promjene napravljene. Crveno označeni redci predstavljaju obrisane retke, a redci sa zelenom pozadinom predstavljaju nove retke.

Google Chrome - preglednik za koji je dodatak razvijen. Omogućava instalaciju vlastitog dodatka. Tijekom razvoja, promjene unutar aplikacijskog koda su u istom trenu vidljive unutar instaliranog dodatka. To uvelike olakšava razvojni proces. Preko Chrome-ih alata za razvojne programe (eng. *DevTools*) moguće je pratiti razvoj web aplikacija unutar ugrađenih alata. verzija 96.0.4664.93. Preglednik je moguće preuzeti sa sljedeće poveznice „https://www.google.com/intl/hr_HR/chrome/“.



Slika 4: Sučelje Chrome DevTools alata

Na slici 4 prikazani su Chrome-ovi alati za razvojne programe. Jedan od alata je konzola i ona ispisuje sve pogreške programa i to olakšava proces uklanjanja pogrešaka (pozicionirana dolje). Također je moguće vidjeti elemente korisničkog sučelja (pozicionirani lijevo). Na taj način je lakše provjeriti njihove vrijednosti i prikazuju li se svi elementi kako je programer želio (pozicija desno).

JavaScript Obfuscator - je program koji JavaScript programski kod pretvara u novi nečitljivi programski kod. Taj programski kod koji je teže čitati, kopirati i koristiti bez poznavanja izvornog koda. Ovaj program je besplatan i dostupan online na stranici „obfuscator.io“.

Slika 5: Sučelje JavaScript Obfuscator alata

Programski kod je moguće unijeti na dva načina, kao tekst ili kao datoteku i pritiskom na gumb „Obfuscate“, aplikacija unutar par sekundi generira nečitljivi kod.

2.2. Tehnologije

Za razvoj dodataka koriste se iste tehnologije kao i za razvoj web aplikacija. Osnovne tehnologije koje se koriste su HTML (eng. *HyperText Markup Language*), CSS (eng. *Cascading Style Sheets*) i JS (eng. *JavaScript*). Za razvoj je moguće koristiti i različite JS okvire kao što su React, Vue.js. U ovom radu korištene su samo osnovne tehnologije te dodatno API (eng. *Application Programming Interface*) preglednika Google Chrome.

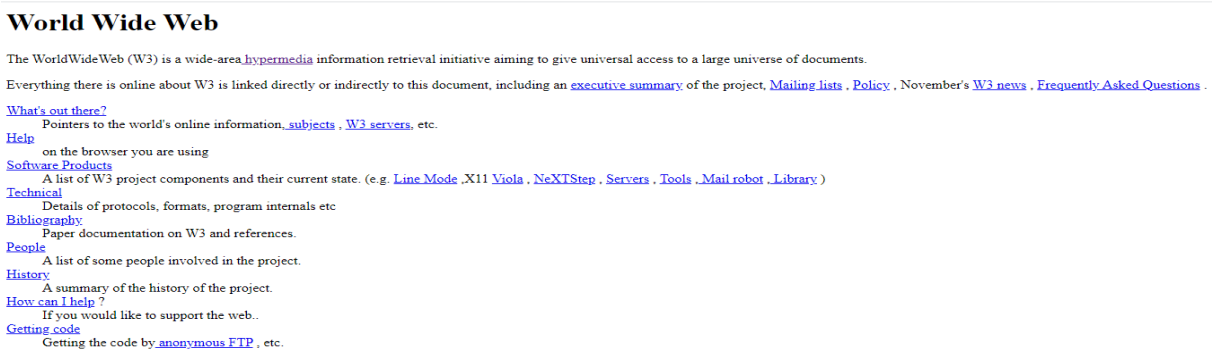
2.2.1. Struktura

HTML se sastoji od oznaka koje web programeri koriste za izradu web stranica. To je standardni format dokumenta za web stranice, definiran od strane IETF-a (*eng. Internet Engineering Task Force*). Svaka web stranica sadrži HTML oznake (*eng. Tag*) ugrađene u izvorni kod koji definiraju strukturu stranice, fontove i hipertekstualne veze. Veza sadrži URL (*eng. Uniform Resource Locator*) druge web stranice na tom istom poslužitelju ili bilo kojem poslužitelju u svijetu [1, str. 3-9]. Dakle, HTML dokument, može se izvršavati u pregledniku i nije potreban poslužitelj. HTML oznake također definiraju grafičke elemente na stranici, od kojih je svaki zasebna datoteka na lokalnom ili udaljenom poslužitelju. Odnosno, HTML se koristi kako bi se stranici dodala struktura i kako bi se grupirali elementi.

U prvih pet godina (1990. - 1995.) HTML je prošao kroz brojne revizije i mnoga proširenja, uglavnom dostupne od strane CERN-a (eng. *European Council for Nuclear Research*), a zatim od IETF-a. Osnivanjem W3C-a (eng. *The World Wide Web Consortium*) organizacijom zaduženom za razvoj web standarda, sljedeći razvoj HTML-a bio je ponovo proširenje. Prvi neuspjeli pokušaj proširenja HTML-a 1995. godine nazvan je HTML 3.0, a zatim je došlo do novog boljeg proširenja pod nazivom HTML 3.2, koja je dovršena 1997. godine. HTML4 ubrzo se pojavio kasnije iste godine.

Sljedeće godine W3C je odlučio prestati razvijati HTML te je umjesto toga počeo istraživati ekvivalent temeljen na XML-u (eng. *Extensible Markup Language*) nazvan XHTML (eng *Extensible HyperText Markup Language*). Krenuli su sa pretvorbom HTML4 u XML, nazvanu XHTML 1.0, a dovršen je 2000. godine. Nakon XHTML 1.0, fokus W3C-a bio je olakšati drugim radnim skupinama proširenje XHTML-a (modularnosti XHTML-a). U isto vrijeme, W3C je također proučavao novi jezik koji nije kompatibilan s ranijim HTML i XHTML jezicima (nazvan XHTML2).

W3C je 2006. godine izrazio interes za sudjelovanje u razvoju HTML5, a 2007. godine osnovana je radna skupina koja će odobriti suradnju s WHATWG-om (eng. *Web Hypertext Application Technology Working Group*). Tvrtke Apple, Mozilla i Opera dopustili su W3C-u da objavi specifikacije pod W3C autorskim pravima i zadrži manje restriktivnu verziju na web stranici WHATWG. Timovi W3C i WHATWG surađuju već dugi niz godina. Međutim, 2011. godine ove su grupe zaključile da imaju različite ciljeve. W3C je htio objaviti "gotovu" verziju "HTML5", a WHATWG je htio nastaviti raditi na standardu HTML–a. U 2019. godini WHATWG i W3C potpisali su ugovor o suradnji na verziji HTML-a.



Slika 6: Prva Web Stranica [2]

Na slici 6 vidi se kako je izgledala prva objavljena web stranica. Web stranica jedino je sadržavala oznake za naslov i paragrafe unutar kojih su postavljene hiperveze. Stranica ne primjenjuje stilsko oblikovanje (danas moguće uz pomoću CSS-a), te ne sadrži funkcionalnosti poboljšanog korisničkog sučelja (danas moguće uz pomoć JavaScript-a).

2.2.2. Prezentacija

CSS je kratica koja u prijevodu znači kaskadni stilski listovi. HTML se koristi za izradu web dokumenata (definiranje sadržaja kao što su naslovi i odlomci te omogućuje umetanje slika, videozapisa i drugih multimedijskih sadržaja). CSS određuje prezentaciju dokumenta (svi rasporedi elemenata, boje i fontovi su uređeni pomoću CSS). HTML predstavlja strukturu (sadržaj) stranice, a CSS izgled (prezentaciju) stranice. O korištenju CSS-a više će biti govora u 5. poglavlju.

CSS je prvi put spomenut 1994. godine, upravo kad je Internet počeo dobivati na popularnosti [3, str. 1-2]. U to vrijeme preglednici su korisnicima pružali podršku CSS-u, poput postavki prezentacije u Mozaik-u. Svakom korisniku omogućilo se je da definira vrste fontova, veličine i boje za svaki element. Razvojni programeri sve što su mogli učiniti je označiti dio sadržaja kao odlomke, naslove na određenoj razini, unaprijed formatirani tekst ili neke od drugih vrsta elemenata. Cilj CSS-a je pružiti jednostavan stil dizajna koji je fleksibilan za programera.

Kako novoosnovana radna skupina za CSS napreduje u CSS2, preglednici nastoje implementirati CSS1 na interoperabilan način. Ti bi problemi mogli potpuno prekinuti CSS, ali na sreću, primijenjeni su neki razumni prijedlozi i preglednici su se počeli sinkronizirati. Radna skupina CSS-a dovršila je CSS2 specifikaciju početkom 1998. godine. Nakon što je CSS2 dovršen, odmah se pristupilo razvoju CSS3, kao i poboljšane verzije CSS2.1. U skladu s vremenom, CSS3 je kreiran kao niz neovisnih modula, a ne kao jedna specifikacija.

Do početka 2012. godine tri su CSS3 modula dosegla status potpuno preporučeni (*eng. Full Recommendation Status*): CSS razina boje 3 (*eng. CSS Color Level 3*), nazivni prostor CSS-a (*eng. CSS Namespaces*) i razina selektora 3 (*eng. Selectors Level 3*). Istodobno, sedam modula ima status preporuke kandidata (eng. *Candidate Recommendation Status*). Prethodno su se metode, boje, selektori i nazivni prostori morale čekati. Nakon što su se svi drugi dijelovi specifikacije dovršili ili izbrisali mogli su postati dio potpune specifikacije [4].

2.2.3. Korisničko sučelje

JavaScript je programski jezik se koristi i na strani klijenta i na strani poslužitelja. Omogućava interaktivnost sa web stranicom [5]. HTML i CSS pružaju strukturu i prezentaciju web stranicama, dok JavaScript web stranici pruža interaktivnu komponentu. Neki primjeri JavaScript-a su: pretraživanje na web trgovini npr. „Amazon“, dohvaćanje i prikaz videozapisa s vijestima ugrađene u novinarski portal npr. „Jutarnji List“ ili dohvaćanje i prikaz vijesti sa društvene mreže npr. „Twitter“. Primjena JavaScript-a poboljšava korisničko iskustvo web stranice pretvaranjem web stranice sa statičke stranice u dinamičku (interaktivnu) stranicu. JavaScript predstavlja sigurnosni problem. JavaScript se i može isključiti u pregledniku. Sve u svemu, JavaScript dodaje interakciju web stranicama. O korištenju JavaScript-a više se može pročitati u 5. poglavlju.

Do sredine 1990-ih godina bilo je važno vrijeme za internet. Ključne tvrtke poput Netscape-a i Microsoft-a bili su usred “ratova” s preglednicima (Netscape-ovim Navigator-om i Microsoft-ovim Internet Explorer-om). U rujnu 1995. godine, programer Netscape-a Brandan Eich razvio je novi skriptni jezik u samo 10 dana. Na početku se je zvao Mocha, ali je brzo postao poznat kao LiveScript, a kasnije i JavaScript. U 1997. godini, zbog brzog rasta JavaScript-a, postalo je jasno da će se jezik morati održavati po nekim normama. Stoga je posao Netscape-a razvoj jezične specifikacije predao Europskom udruženju proizvođača računala (eng. *European Computer Manufacturer's Association* - ECMA). ECMA je osnovana s ciljem standardizacije računalstva. ECMA specifikacije bile su označene kao ECMA-262, a jezici ECMAScript uključivali su JavaScript, JScript i ActionScript.

Između 1997. i 1999. godine ECMA-262 je tri puta revidiran. Gotovo 10 godina kasnije, četvrto izdanje je napušteno zbog različitih mišljenja o smjeru jezika i njegovim predloženim funkcijama. Zanimljivo je da su mnoge od ovih kontroverznih značajki, poput generatora, iteratora i dodjele destrukcije tj. brisanja, uključene u novu specifikaciju ECMAScript.

Ispostavilo se da je 2005. godina bila sjajna godina za JavaScript. Jesse James Garrett predstavlja AJAX (eng. *Asynchronous JavaScript And XML*), revolucionarni tehnološki paket koji uključuje JavaScript. AJAX uvelike poboljšava korisničko iskustvo stvarajući dojam da se web stranice više ponašaju kao stolne (eng. *Desktop*) aplikacije. Na taj način je JavaScript kao programski jezik stavljen u središte pozornosti.

Inačica	Službeni naziv	Opis
ES1	ECMAScript 1 (1997)	Prva verzija (eng. First Edition)
ES2	ECMAScript 2 (1998)	Uredničke promjene (eng. Editorial changes)
ES3	ECMAScript 3 (1999)	Dodani regularni izrazi (eng. regular expressions) Dodan probaj/ulovi (eng. try/catch) Dodana skretnica (eng. switch) Dodan radi dok (eng. do-while)
ES4	ECMAScript 4	Nikad objavljen
ES5	ECMAScript 5 (2009)	Dodan strog način (eng. strict mode) Dodana podrška za JSON Dodana funkcija String.trim() Dodana funkcija Array.isArray() Dodani načini iteracije kroz polja Dopušta prateće zareze za literale objekata (eng. trailing commas)
ES6	ECMAScript 2015	Dodan let i const za deklaraciju varijabli Dodane zadane vrijednosti parametra (eng. default) Dodana funkcija Array.find() Dodana funkcija Array.findIndex()
	ECMAScript 2016	Dodan operator za eksponencije (**) Dodana funkcija Array.includes()
	ECMAScript 2017	Dodano ispunjavanje stringova (eng. string padding) Dodana funkcija Object.entries() Dodana funkcija Object.values() Dodane asinkrone funkcije (eng. async) Dodana djeljenja memorija (eng. shared memory)
	ECMAScript 2018	Dodani svojsta za odmor i širenje (eng. rest/spread) Dodana asinkrona iteracija Dodana funkcija Promise.finally() Dodane funkcije regularnim izrazima

Slika 7: Prikaz verzija ECMAScript-a [6]

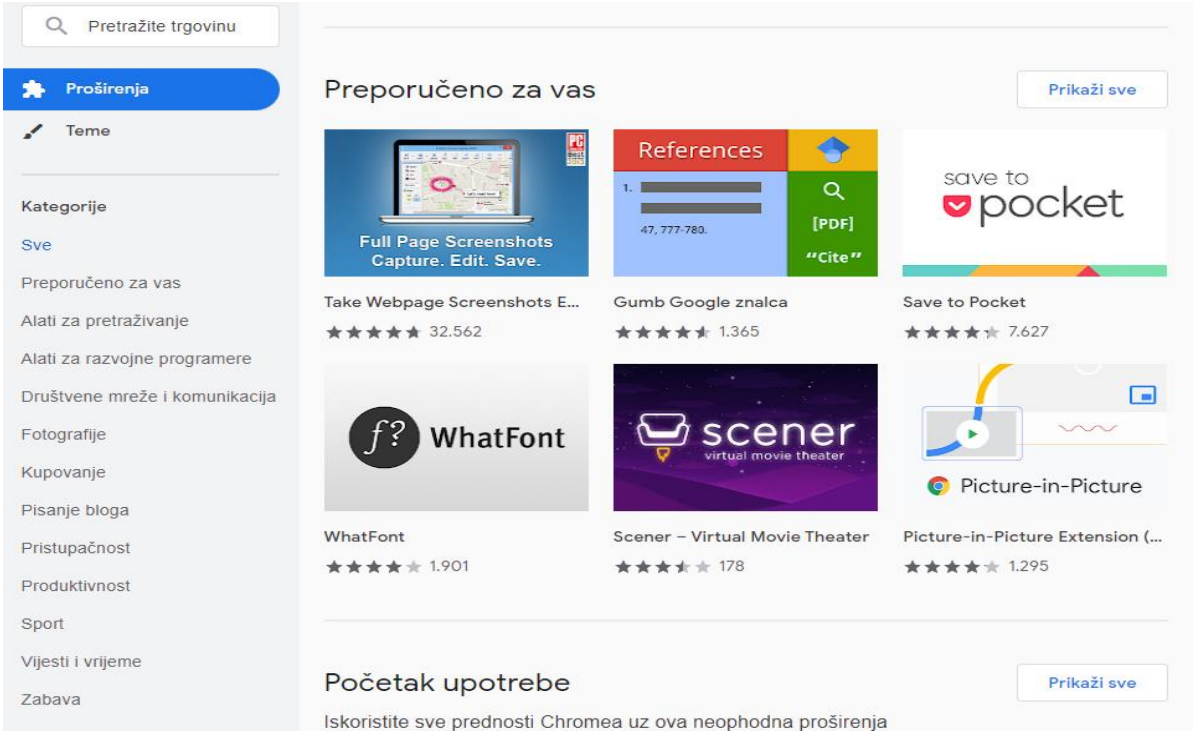
Na slici 7 vidimo sve verzije ECMAScript-a kroz godine i opise što je došlo s njima. JavaScript ES6 donosi novu sintaksu i nove sjajne značajke kako bi kod bio moderniji i čitljiviji. ES6 dolazi s mnogim sjajnim značajkama kao što su funkcije strelice (*eng. Arrow functions*), nizovi predložaka, destrukcija klasa, moduli i ostalo. Praktični dio ovog završnog rada razvijao se je u najnovijoj verziji ES6.

2.2.4. Aplikacijsko programsko sučelje

Aplikacijsko programsko sučelje (API) omogućuje da dvije aplikacije međusobno komuniciraju radi pristupa podacima. Svaka radnja koju izvršavamo na webu putem našeg mobilnog uređaja (poput slanja poruke ili pregledavanja rezultata nogometne utakmice) ili stolnog računala koristi API-je za pristup i dohvaćanje tih podataka. Većina web mjesta pružaju API koji programeri mogu iskoristiti da izrade vlastite aplikacije koje komuniciraju s tim web mjestom [7]. Primjerice, Google pruža API koji omogućuje programerima da implementiraju autentikaciju ili prikaz karte zemlje u vlastitoj aplikaciji koristeći njihove sustave.

2.2.4.1. API Chrome dodataka

Google Chrome pruža mnogo različitih API-ja za programere svojih dodataka. Na primjer: dodavanja novih stranica, skočni prozori, kreiranje obavijesti, postavljanja zadane tražilice, pa čak i kreiranja stavki kontekstnog izbornika (izbornik koji se pojavljuje kada se desnom tipkom miša pritisne stranica). Mogućnosti su beskrajne. Počevši od dodataka koje prikazuje poruku "Pozdrav svijete!" do dodatka koje nam omogućuje snimanje zaslona web stranice. Unutar Chrome web-trgovine (dostupno na ovoj web stranici „https://chrome.google.com/webstore“) moguće je pronaći dodatke uz pomoć pretrage ili pretraživanje po kategorijama. Kategorije koje se nude su vezane uz uređivanje fotografija, kupovinu, produktivnost, sport, vrijeme, društvene mreže i zabavu (vidljivo na slici 8).



Slika 8: Chrome web-trgovina

Metode u Chrome API-ju su asinkrone, vraćaju se odmah, bez čekanja da se operacija dovrši. Ako treba vratiti rezultat operacije, u metodu se prosljeđuje funkciju povratnog poziva (eng. *Callback Function*).

Sa najnovijom verzijom manifesta (v3) moguće je koristiti obećanja (eng. *Promises*) prilikom korištenja Chrome API-ja. U suštini, obećanje je vraćeni objekt kojemu se prilažu funkcije povratnog poziva, umjesto prosljeđivanja povratnih poziva u funkciju. Obećanja su pojednostavila upravljanje greškama, dala su mogućnost programiranja u sinkronom stilu pozivanje asinkronih funkcija i sa jednostavnijom sintaksom su olakšale pozivanje istodobnih funkcija [8].

Korištenja obećanja rezultira čišćim kodom koji je lakši za održavanje. Vrijeme kada je poželjno koristiti obećanja je kada se želi urediti kod da izgleda više kao sinkroni. Kada bi upravljanje pogreškama bilo puno jednostavnije nego sa funkcijama povratnog poziva.

Ispod je vidljiv kod koji pokazuje razliku u načinu pozivanja funkcija:

```
function successCallback(result) {
  console.log("Email succesfully sent to: " + result);
}
function failureCallback(error) {
  console.error("Error sending email: " + error);
}

sendEmailAsync(sendTo, successCallback, failureCallback);

sendEmailAsync(sendTo).then(successCallback, failureCallback);
```

U posljednja dva retka ovog koda, vidi se razlika pozivanja, na prvi način prosljeđuju se funkcije povratnog poziva dok na drugi način se spajaju koristeći ključnu riječ „then“.

Ovo također omogućava povezivanje više funkcija jednu za drugom. Dok bi korištenjem funkcija s povratom poziva, ako ih se hoće više povezati, napravila takozvana „piramida“, korištenjem obećanja ispao bi „lanac“, vidljivo u kodu ispod:

```
// Piramida
nekaFunkcija(function(rezultat) {
    nekaDrugaFunkcija(rezultat, function(noviRazultat) {
      nekaTrecaFunkcija(noviRazultat, function(finalniRezultat) {
        console.log('Finalni rezultat je: ' + finalniRezultat);
      }, failureCallback);
    }, failureCallback);
  }, failureCallback);
```

```
// Lanac
nekaFunkcija()
.then(rezultat => doSomethingElse(rezultat))
.then(noviRazultat => doThirdThing(noviRazultat))
.then(finalniRezultat => {
  console.log(`Finalni rezultat je: ${finalniRezultat}`);
})
.catch(failureCallback);
.error("Error sending email: " + error);
}
```

Na taj način kod je puno čitljiviji i lakši za održavanje.

Dodaci mogu koristiti lokalno spremište (eng. *Local Storage*) preglednika za trajnu pohranu podataka. Pomoću Chrome-ovih ugrađenih JSON funkcija moguće je pohraniti složene strukture podataka. Web dodaci mogu pohraniti do 5 MB podataka u lokalno spremište. Chrome implementira SQL baze podataka na strani klijenta za web dodatke koji koriste SQL (eng. *Structured Query Language*) upite.

Chrome nakon preuzimanja web dodatak automatski istog instalira. Međutim, ako programer stavi novu verziju web dodatka na web-trgovinu, on će se automatski preuzeti u pozadini. Web dodaci također mogu komunicirati s web servisima. Kako bi se unapredila kvaliteta korisničkog iskustva u Chrome-u od 19. prosinca 2013. godine uvedeno je pravilo prema kojem Chrome-ovi dodaci moraju imati samo jednu svrhu.

To znači da dodaci moraju imati jedinstvenu svrhu koja je lako razumljiva. Ako su dva dijela funkcionalnosti jasno odvojena, potrebno ih je staviti u dva različita web dodatka. Korisnici bi trebali imati mogućnost zasebnog instaliranja i deinstaliranja web dodatka. Osim toga, početkom svibnja 2014. godine, obavezno je da se dodaci u Chrome-u za Windows operacijski sustav, nalaze u Chrome web-trgovini. MacOs operacijski sustav isto tako je s Chrome verzijom 44 počeo zahtijevati da se od srpnja 2015. godine svi web dodaci nalaze u Chrome web-trgovini [9].

2.2.4.2. WebExtension API

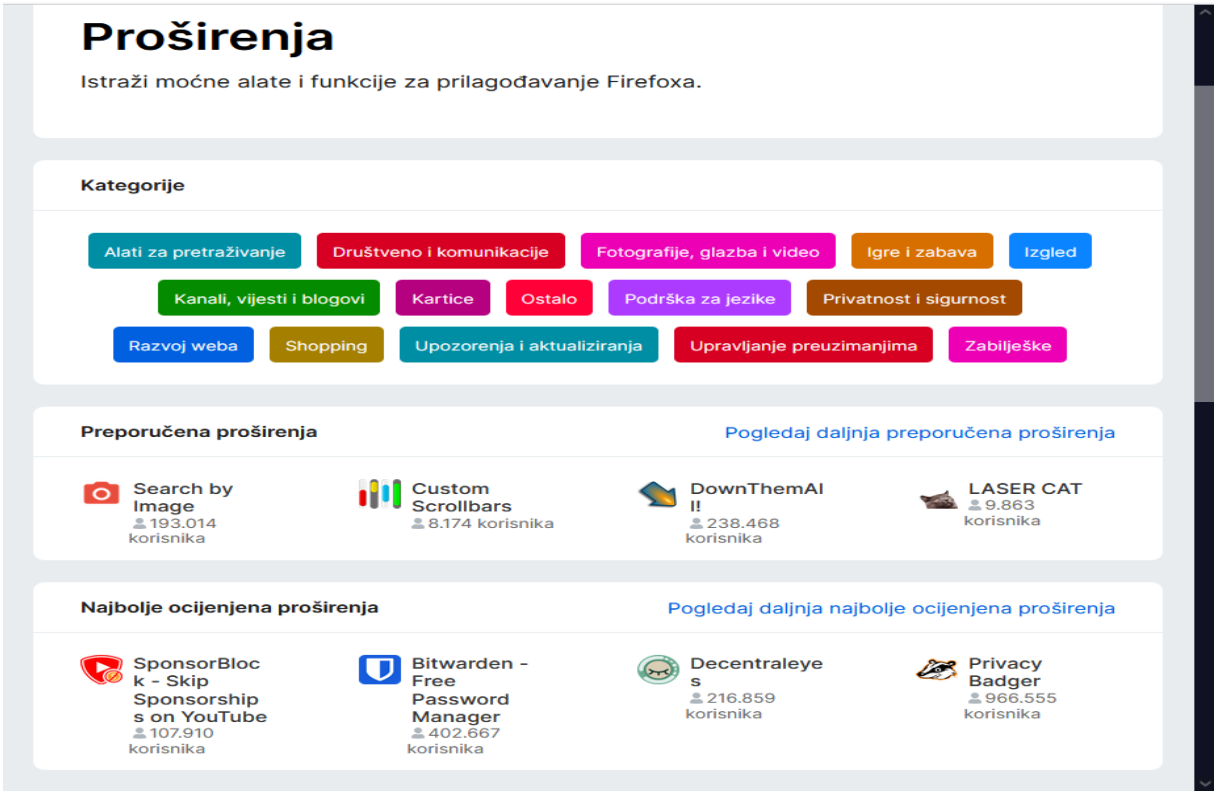
WebExtension API pruža web-platformu za razvoj dodatka za proširenje funkcija Firefox-a i ostalih preglednika. Prema [10] uvođenje API-ja dodataka preglednika stvorilo je jedinstvenu okolinu za razvoj proširenja preglednika. Međutim, postoje razlike u implementacijama API-ja i opsegu pokrivenosti među preglednicima koji koriste API proširenja (glavni su Chrome, Edge, Firefox, Opera i Safari). Povećanje dosega dodataka preglednika znači razvoj za najmanje dva preglednika, a moguće i više.

Postoji šest područja na koja je potrebno obratiti pažnju kada se izrauju proširenja za više preglednika:

- API prostor imena– postoje dva glavna prostora imena koja se koriste a to su prostor imena „browser.*“ predloženi standard za API proširenja koje koriste preglednici Firefox i Safari. Postoji još i prostor imena „chrome.*“ koji koriste preglednici Chrome, Opera i Edge. Firefox također podržava prostor imena „chrome.*“ za API-je koji su kompatibilni s preglednikom Chrome-om, prvenstveno za pomoć pri prijenosu (eng. *port*). Međutim, preferira se korištenje prostor imena „browser.*“. Osim što je predloženi standard, „browser.*“ koristi obećanja.
- API asinkrono rukovanje događajima – postoje dva asinkrona načina rukovanja događajima, koji su ranije spomenuti, a to su funkcije sa povratnim pozivom i obećanja i preporuća se korištenje obećanja radi pojednostavljanja programa
- Pokrivenost API funkcija - razlike u API funkcijama koje se nude u svakom od glavnih preglednika dijele se u tri kategorije:
    - Nedostatak podrške za cijelu funkciju. Na primjer, u vrijeme pisanja ovog rada, preglednik Edge ne podržava funkciju postavke preglednika (eng. *browserSettings*).
    - Varijacije u podršci za značajke unutar funkcije. Na primjer, preglednici Safari i Firefox trenutno ne podržavaju metodu obavijesti na pritisak gumba (eng. *notification.onButtonClicked*), dok je preglednik Firefox jedini preglednik koji podržava metodu na prikaz obavijesti (eng. *notification.onShown*).
    - Vlasničke (eng. *propriatery*) funkcije koje podržavaju značajke specifične za preglednik. Na primjer, u vrijeme pisanja, spremnici su bili značajka specifična za preglednik Firefox koju podržava funkcija kontekstualni identiteti (eng. *contextualIdentities*). Jedan od načina za suočavanje s ovim problemom je da se koriste funkcije koje podržavaju svi preglednici ili korištenjem drugačijih implementacija funkcije.
- Manifest ključne riječi - razlike u ključnim riječima datoteke „manifest.json“ koje podržavaju glavni preglednici dijele se uglavnom u tri kategorije:
    - Atributi informacija proširenja. Preglednici Firefox i Opera uključuju ključnu riječ „developer“ za pojedinosti o programeru proširenja.
    - Značajke proširenja. Preglednik Chrome trenutno ne podržava ključnu riječ „browser_specific_settings“.

- Obaveznost ključnih riječi. Općenito su samo "manifest_version", "version" i "name" obavezne ključne riječi.

- Zapakiravanje dodataka - za distribuciju kroz trgovine dodataka preglednika relativno je jednostavno. Preglednici Firefox, Chrome, Edge i Opera koriste jednostavan zip format koji zahtijeva da datoteka „manifest.json“ bude u korijenu zip paketa. Preglednik Safari zahtijeva da proširenja budu zapakirana na sličan način kao i aplikacije.
- Objava dodataka - svaki od glavnih preglednika posjedue trgovine dodataka za preglednik. Svaka trgovina također pregledava dodatke kako bi provjerila ima li sigurnosnih propusta.



Slika 9: Firefox dodaci za preglednik

Kako postoji Chrome web-trgovina za Chromium bazirane preglednike, tako postoji i stranica za Firefox dodatke (eng. *Firefox Browser Add-Ons*). Dostupno na web stranici „https://addons.mozilla.org“. Obje web trgovine nude slične kategorije i dodatke koji su dostupni za svoje preglednike (vidljivo na slici 9).

3. Web preglednici

Web preglednik je program koji poznaje sintaksu HTML jezika i na temelju prepoznatih elemenata iz učitanog HTML dokumenta prikazuje na ekranu njima pripadajuće objekte. Prema [11] trenutno najpoznatiji i najkorišteniji web preglednici su „Google Chrome“ i „Apple Safari“.

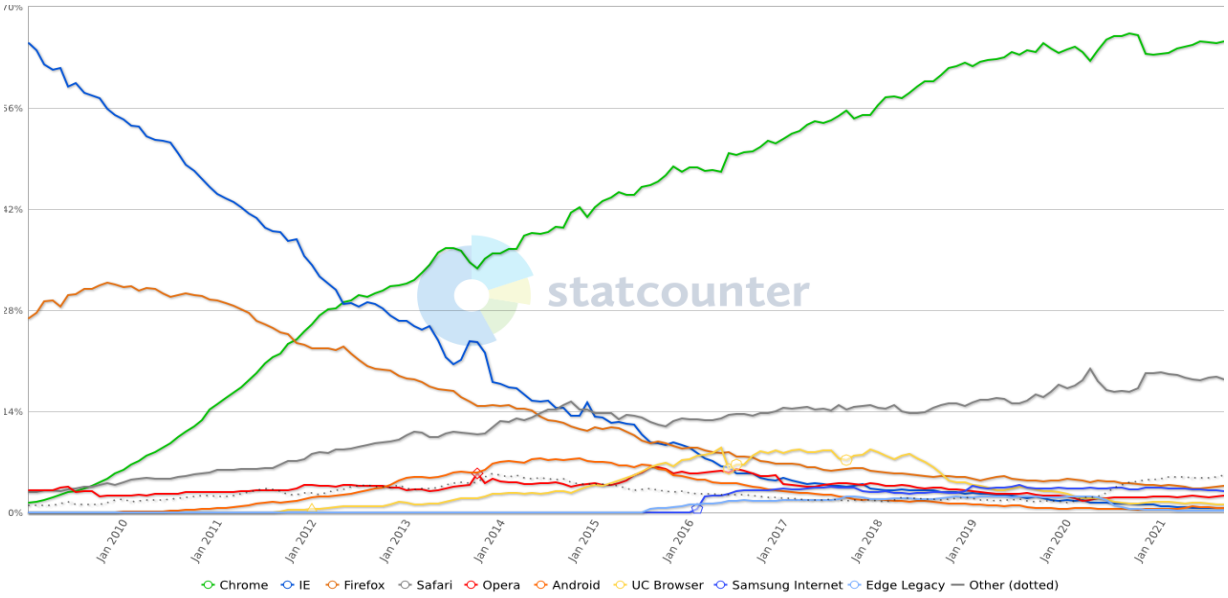
Iako nisu popularni kao i ostali, postoje preglednici kojima je fokus na sigurnosti i privatnosti korisnika. Oni odmah blokiraju skripte i resurse koje prate rad i ponašanje korisnika. Tako reklame nisu vidljive korisniku i nemaju pristup njegovoj Internet aktivnosti. Primjer takvih preglednika su Tor preglednik (od Tor Projecta) i preglednik Brave (od Brave Softwarea).

3.1. Povijest web preglednika

Iako se osobna računala proizvode od sedamdesetih godina dvadesetog stoljeća, potreba za web preglednikom nije postojala sve do 1991. godine (skoro 15 godina kasnije kada je Internet odnosno World Wide Web postao dostupan široj javnosti). Najznačajnije godine web preglednika prema [12] su:

- 1990. godina - Pojavio se je prvi preglednik pod nazivom The WorldWideWeb i bio je jedini način pristupu Internetu. Razvio ga je Tim Berners-Lee (ista osoba koja se smatra izumitelj Interneta). Budući da je ovo bio prvi web preglednik to je također bio i jedini način za pristup „World Wide Webu“. Kako ne bi dolazilo do zabune, kasnije je ovaj preglednik preimenovan u „Nexus“.
- 1992. godina - Na tržište je izašao Lynx i MidasWWW. Ni jedan ni drugi nisu imali mehanizam za prikaz grafičkog sadržaja.
- 1993. godina - Iako je zadnja verzija izašla 1997. godine, prije dvadeset i četiri godine, ovom pregledniku se pripisuje popularizacija korištenja web preglednika.
- 1995. godina - Microsoft je razvio vlastiti preglednik za web, poznat pod nazivom Internet Explorer. On je i dan danas najpoznatiji web preglednik temeljem svoje integracije unutar operacijskog sustava Windows. Microsoft je odlučio odustati od njega zbog novijih, bržih i stabilnih preglednika. Tako da 15. lipnja 2022. godine ovaj web preglednik odlazi u „mirovinu“.
- 1996. godina - Norveško poduzeće objavljuje prvu javno dostupnu verziju web preglednika Opera. Ovaj web preglednik je započeo kao znanstveno istraživanje norveškog poduzeća „Telenor“. 1995. godine se odvaja u vlastito poduzeće pod nazivom „Opera Software“.

- 2003. godina - Apple objavljuje vlastiti web preglednik Safari i sva Macintosh računala unaprijed imaju predinstaliran isti (do sad su dolazila sa Netscape Navigator-om).
- 2004. godina - Izlazi prvi web preglednik otvorenog izvornog koda (Mozilla Firefox).
- 2008. godina - Google je razvio vlastiti web preglednik Chrome koji je s vremenom postao najpopularniji među dostupnim preglednicima.
- 2015. godina - Microsoft razvio je Edge kako bi vratio popularnost koju je nekada imao Internet Explorer.
- 2020. godina - Microsoft je odustao od prijašnjeg Microsoft Edge-a koji je bio napravljen na EdgeHTML mehanizmu i prebacili su se na Chromium.



Slika 10: Grafički prikaz korištenja preglednika u proteklih 12 godina [13]

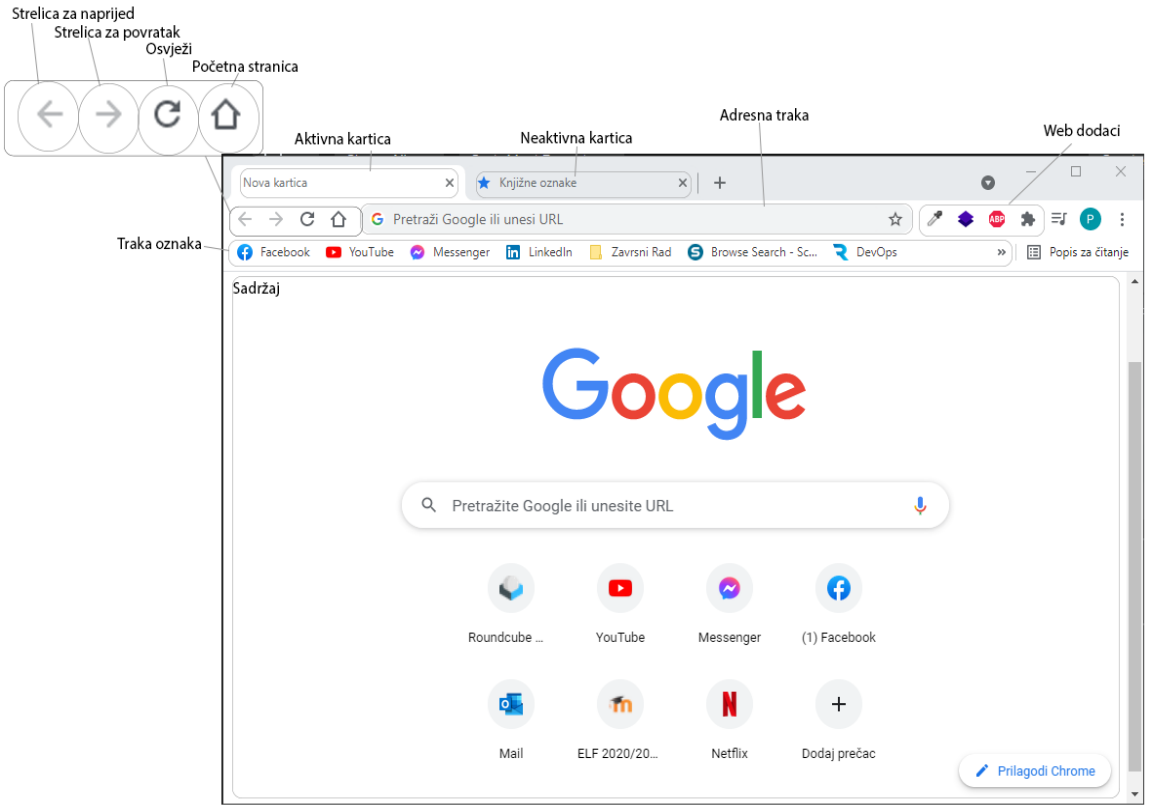
Na slici 10 moguće je vidjeti kako se mijenjala popularnost web preglednika. Nekad je dominirao Internet Explorer, ali s vremenom je zbog njegovog zaostajanja s novim funkcionalnostima za ostalim preglednicima, izgubio tu popularnost. Vidimo da je prijelazom iz 2012. godine na 2013. godinu, Google Chrome preuzeo vodstvo i dan danas je najpopularniji preglednik. Jedan od bitnih razloga zašto je na mobilnim uređajima najpopularniji Chrome je to što koriste Android operacijski sustav koji je razvio Google.

3.2. Trenutno stanje web preglednika

Prema [13] trenutno tržištem dominira Google Chrome kako na stolnim računalima (pretežito sa Windows i Linux operativnim sustavima) tako i na pametnim telefonima (Android). Za razliku na svim Apple računalima i telefonima prevladava Safari. Chromium je Google-ov preglednik s izvorno dostupnim kodom. Na njemu je bazirana većina drugih popularnih preglednika kao što je Opera, Edge, Brave (preglednik kojemu je glavna stvar sigurnost i privatnost korisnika) [14]. Samsung Internet preglednik dostupan samo za pametne telefone. Tako da je glavna „bitka“ web preglednika između Safari-a i Chrome-a.

3.2.1. Sučelje web preglednika

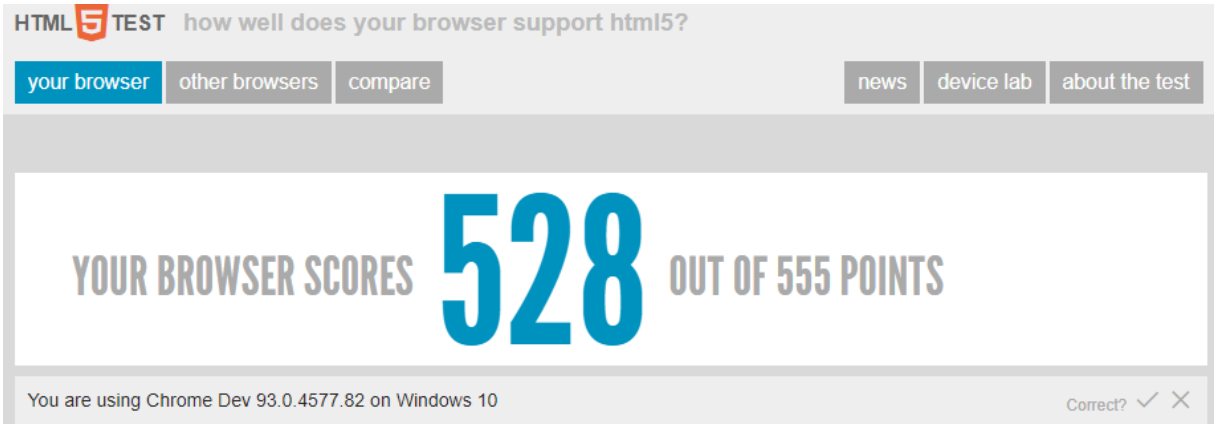
Korisničko sučelje preglednika (eng *Browser User Interface* - BUI) je metoda interakcije s aplikacijom, koja se obično nalazi na poslužitelju, putem komponenata predstavljenih u web pregledniku. Na slici 11 vidi se sučelje Chrome web preglednika. Većina web preglednika ima slično grafičko korisničko sučelje.



Slika 11: Sučelje Chrome web preglednika

Chrome-ovi alati za razvojne programere se otvaraju koristeći tipku F12 na tipkovnici.

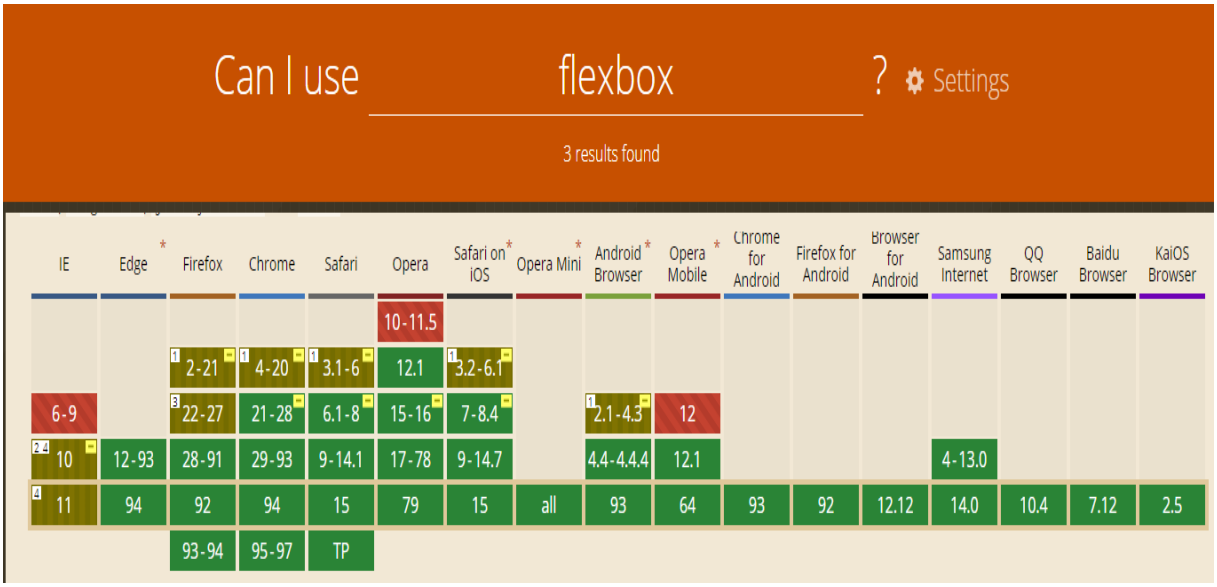
„HTML5test“ je web stranica za procjenu implementacije HTML5 standarda, WebSQL baze podataka (razvijen od World Wide Web Consortium) i WebGL standarda (razvijen od Mozilla Foundation i Khronos Group).



Slika 12: HTML5 podrška preglednika [15]

HTML5test procjenjuje podršku korištenog preglednika korisnika koji je posjetio stranicu za: pohranu podataka, W3C Geolocation API, specifične HTML5 elemente (uključujući element canvas) i druge značajke. Ne ocjenjuje usklađenost preglednika s drugim web standardima, kao što su: CSS, ECMAScript, skalabilna vektorska grafika (SVG) ili dokumentno objektni model (DOM). Maksimalni broj bodova je 555, a korišteni preglednik Chrome je ostvario 528 bodova (vidljivo na slici 12) [15].

Web stranica "Can I use" pruža ažurirane izvještaje podrške u korištenju naprednih web tehnologija za web preglednike stolnih računala i mobilnih uređaja.



Slika 13: CSS podrška preglednika [16]

Web stranicu je izradio Alexis Deveria i održava je uz pomoć zajednice za razvoj web stranica (eng. *Web Development Community).* Za dizajn je zaslužan Lennart Schoors, koji se koristi od 2014. godine [16]. Na slici iznad vidi se podržanost flexbox-a u različitim web preglednicima. Zelene vrijednosti predstavljaju podržanost u tim verzijama preglednika. Žute vrijednosti omogućuju korištenje flexbox-a uz navođenje postavke (potrebno je dodati osobinu „webkit“). Crvene vrijednosti ne podržavaju korištenje flexbox-a u tim verzijama preglednika.

3.2.2. Arhitektura web preglednika

Arhitektura web preglednik se prema [17] sastoji od:

- Korisničko sučelje (*eng. User Interface*): je prostor putem kojeg korisnik komunicira s preglednikom. Uključuje: adresnu traku, tipke za naprijed i nazad te gumb za početnu stranicu, tipke za osvježavanje, opcije, oznake. Osim prozora koji prikazuje traženu stranicu, svi ostali dijelovi su ispod njega.
- Stroj preglednika (*eng. Browser Engine*): Preglednik djeluje kao posrednik između korisničkog sučelja i stroja za iscrtavanje. Na temelju unosa s različitih korisničkih sučelja, traži i manipulira strojem za iscrtavanje.
- Stroj za iscrtavanje (*eng. Rendering Engine*): Kao što naziv govori, on je odgovoran za iscrtavanje tražene web stranice na zaslonu preglednika. Stroj interpretira HTML, XML dokumente i slike oblikovane pomoću CSS-a i generira izgled za prikaz na korisničkom sučelju. Različiti preglednici koriste različite strojeve za iscrtavanje. Internet Explorer koristi Trident. Firefox i drugi Mozilla preglednici koriste Gecko. Chrome i Opera 15+ koriste Blink. Chrome i Safari koriste Webkit.
- Mrežna komponenta (*eng. Networking*): Komponenta preglednika koja koristi uobičajene internetske protokole HTTP ili FTP za dohvaćanje URL-ova. Mrežna komponenta upravlja svim aspektima Internetske komunikacije i sigurnosti. Mrežna komponenta može unaprijed pohraniti (*eng. Cache*) preuzete dokumente kako bi se smanjio mrežni promet.
- JavaScript prevoditelj (*eng. Javascript Interpreter*): Ovo je komponenta preglednika koja se koristi za prevođenje i izvršavanje Javascript koda ugrađenog u web mjesto. Kompilirani rezultat priprema se za prikaz. Ako je skripta vanjska, prvo se preuzme izvor s mreže. Prevoditelj je zaustavljen sve dok se skripta ne izvrši.
- Pozadina korisničkog sučelja (*eng. User Interface Backend*): Pozadina korisničkog sučelja koristi se za crtanje osnovnih grafičkih elemenata (*eng. Widgeta*), poput kombiniranih okvira i prozora. Pozadina korisničkog sučelja nudi generičko sučelje koje nije specifično za platformu.

- Postojanost/pohrana podataka (*eng. Data Persistance / Data Storage*): Preglednik podržava mehanizme za pohranu kao što su „localStorage“, „IndexedDB“, „WebSQL“. „WebSQL“ je mala baza podataka kreirana u pregledniku računala. Omogućuje rad sa korisničkim podacima kao što su: predmemorija, kolačići, knjižne oznake i postavke preglednika.

Slika 14: Komponente preglednika [17]

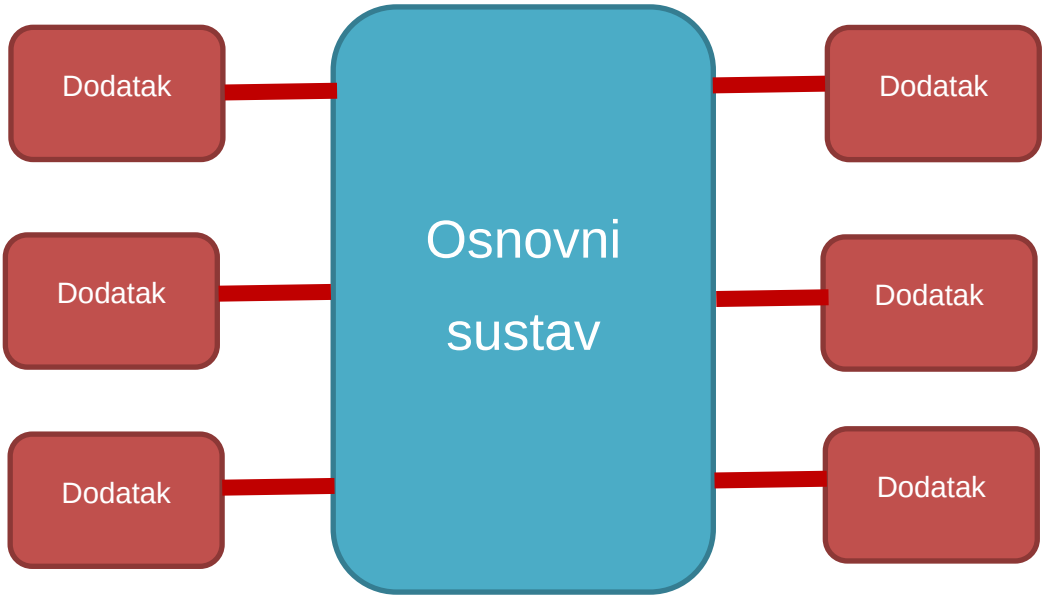
Na slici 14 vidimo komponente preglednika i njihovu međusobnu interakciju. Može se vidjeti da je većina web preglednika izgrađena u sedam komponenti, od kojih svaki ima svoje odgovornosti.

4. Web dodaci za preglednik

Ukratko, web dodatak za preglednik je mali dio funkcionalnosti koji omogućava korisniku personalizaciju vlastitog iskustva prilikom korištenja Interneta. Oni omogućavaju korisnicima promjenu izgleda svake web stranice tako da na primjer mogu dodati svoju najdražu sliku kao pozadinu te stranice. Osim toga, moguće je dodavanje i drugih značajki u svoj preglednik (primjerice sakrivanje neželjenog sadržaja). Prema [18] programski kod za web dodatke preglednika je razvijen u web tehnologijama (HTML, CSS i JavaScript). HTML služi za definiciju njegove strukture. CSS je zadužen za definiciju njegovog izgleda. JavaScript se primjenjuje za poboljšanje korisničkog iskustva.

4.1. Mikrokernel uzorak

Prema [19] mikrokernel uzorak dizajna (eng. *Microkernel Design Pattern*) je uzorak za implementaciju aplikacija. Web dodaci za preglednike su sastavljeni kao neovisne komponente. Osmišljene su za poboljšanje ili proširenje osnovnog sustava (eng. *Core System*). Mikrokernel uzorak omogućuje dodavanje novih značajki (eng. *Plug-in Component*) u obliku dodataka osnovnog sustava. On pruža proširivost, kao i odvajanje i izolaciju značajki.



Slika 15: Mikrokernel uzorak dizajna [19]

Na slici 15 vidimo mikrokernel uzorak dizajna. U centru je osnovni sustav (web preglednik), ostatak su komponente dodataka (web dodaci) preglednika koji proširuju funkcionalnosti preglednika.

Mikrokernel uzorak dizajna pruža veliku podršku evolucijskom dizajnu i inkrementalnom razvoju. Najprije je potrebno izgraditi osnovni sustav, a zatim dodati komponente dodatka (značajke i funkcije) kako se aplikacija razvija bez većih promjena u sustavu jezgre. Osnovni sustav kod mikrokrernel uzorka sadrži samo minimalne funkcionalnosti potrebne za rad sustava. Osnovni sustav često se sastoji od glavne poslovne logika bez prilagođenog koda za posebne slučajeve, posebna pravila ili složenu uvjetnu obradu.

Tablica 1: Analiza uzorka [19]

Karakteristika	Ocjena	Analiza
Ukupna agilnost	Visoka	Sposobnost brzog reagiranja na promjenu okruženja. Mogućnost brzih promjena putem dodataka. Mikrokernel uzorak dizajna omogućava da osnovni sustav postane stabilan vrlo brzo i kao takav zahtijeva mali broj promjena tijekom vremena.
Jednostavnost implementacije	Visoka	Ovisno o tome kako je uzorak implementiran, dodaci se mogu dinamički dodavati na osnovni sustav tijekom izvođenja, smanjujući vrijeme zastoja tijekom implementacije.
Testiranje	Visoka	Dodaci se mogu testirati izolirano i lako se oponaša strana osnovnog sustava kako bi se izradila određena značajka bez promjena u osnovnom sustavu.
Izvođenje	Visoka	Većina aplikacija izgrađenih pomoću mikrokernel uzorka dizajna radi dobro jer se mogu prilagoditi i pojednostaviti tako da uključuju samo one značajke koje su potrebne. Moguće je smanjiti aplikaciju da koristi samo one značajke koje su potrebne, uklanjajući nepotrebne dodatke koji pretjerano troše memoriju, procesor i usporavaju izvođenje aplikacije.
Skalabilnost	Niska	Budući da je većina implementacija bazirana na proizvodu i općenito su manje veličine, implementiraju se kao pojedinačne jedinice i stoga nisu visoko skalabilne.
Lakoća razvoja	Niska	Arhitektura mikrokernela zahtijeva promišljen dizajn i upravljanje ugovorom, što je čini prilično složenom za implementaciju.

Mikrokernel uzorak dizajna preporuča se koristiti kod aplikacija gdje će se vremenom dodatne nove značajke. Ako se naknadno pokaže da ovaj dizajn sustava ne može zadovoljiti sve zahtjeve uvijek je moguće promijeniti arhitekturu sustava [19]. Web dodaci su samostalne, neovisne komponente koje sadrže specijaliziranu obradu, dodatne funkcije i posebna pravila. Osmišljeni su za poboljšanje ili proširenje osnovnog sustava. Općenito, web dodaci bi trebali biti neovisni o drugim web dodacima. Moguća je Interakcija sa drugim web dodacima. U svakom slučaju, važno je svesti interakciju između dodataka na minimum kako bi se izbjegao problem prevelike ovisnosti. Pokazalo se da je ovakav dizajn sustava vrlo fleksibilan.

Najbolji primjer arhitekture mikrokernela je Eclipse IDE. Preuzimanjem ovog razvojnog okruženja programer preuzima samo moderan uređivač. Međutim, kada se počinju dodavati moduli, postaje vrlo prilagodljiv i koristan. Web preglednici su još jedan uobičajeni primjer proizvoda koji koriste arhitekturu mikrokernela. Različiti web dodaci dodaju nove značajke pregledniku koje nisu uključene prilikom instalacije preglednika.

Programeri mogu pojedinačno testirati dodatne prototipe i provjeriti probleme s performansama bez utjecaja na osnovni sustav. Također važno je napomenuti da ovo omogućava korisnicima izradu vlastitih dodataka za personalizaciju svog iskustva u pregledniku.

4.2. Povijest web dodataka

Prema [20] prvi dodatak za preglednik nastao je 1999. godine unutar Microsoft-ovog Internet Explorer-a i zvao se „Explorer Bar“. Omogućavao je korisnicima da unutar prozora preglednika imaju manji prozor unutar kojeg su mogli prikazivati dodatne informacije i upravljati preglednikom.

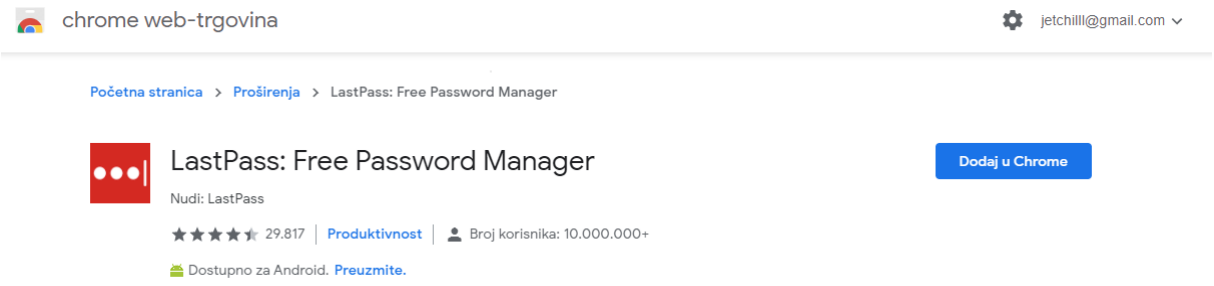
Nakon Internet Explorer-a, prvi koji je omogućio dodatke za preglednike je bio Mozilla Firefox (2004. godine). U beta verziji (v0.9), omogućavao je korisnicima da instaliraju različite dodatke i mijenjaju temu svog preglednika (godinama prije ostale konkurencije). Alpha 10.5 verzija Opere, 2010. godine dolazi sa svojim dodacima za preglednik i sa svojom stranicom za preuzimanje dodataka pod nazivom „Opera extensions“. Google Chrome je također iste godine dodao podršku za vlastite dodatke, kao i Apple Safari.

Slika 16: Prvi web dodatak [20]

Na slici 16 vidimo prvi web dodatak. Bio je napravljen samo za Internet Explorer. Zvao se „Explorer bar“ i bilo ga je moguće postaviti na lijevu stranu ili na donju stranu preglednika. Unutar manjeg prozora „Explorer bar“ moguć je prikaz dodatnih informacija na isti način kao u običnom prozoru preglednika.

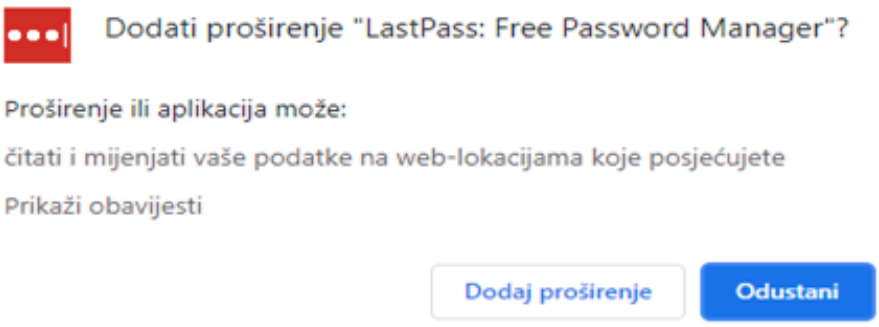
4.3. Korisni web dodaci za preglednik

Neki od korisnijih web dodataka u kontekstu praktičnog dijela završnog rada su „LastPass“ koji drži sve lozinke od korisnika na jednom mjestu. Samim klikom na njega korisnik može spremiti i dohvatiti lozinku i korisničko ime na nekoj stranici. Web dodatak „Honey“ korisnicima tijekom online kupovine klikom pronalazi kupone za popuste, od par kuna pa do većih iznosa. „Honey“ to radi na način da za stavke unutar košarice provjerava dostupne kupone na najpopularnijim web trgovinama. Nažalost trenutno ne radi na previše hrvatskih stranica, ali iz povratne informacije drugih korisnika ima dobre recenzije. Naravno, jedan od najpopularnijih i korištenijih web dodataka je „AdBlocker“ te sve njegove kopije i verzije. On omogućuje korisnicima automatsko sakrivanje reklama, blokiranje skočnih prozora i neželjenog sadržaja te prati i blokira aktivnosti zlonamjernog softvera na web stranicama.



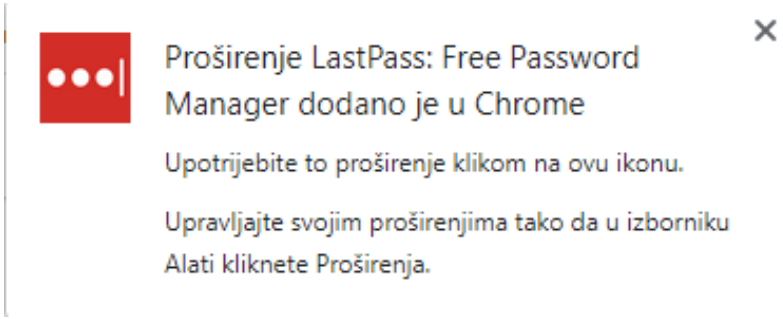
Slika 17: Chrome web-trgovina LastPass dodatka [21]

Kako bi se instalirali ti dodaci za preglednik Google Chrome, treba posjetiti Chrome web-trgovinu ili pritisnuti na poveznicu nakon naziva dodatka. Poveznice vode na stranicu koja izgleda kao na slici 17. Ovdje treba pritisnuti na gumb sa tekstom „Dodaj u Chrome“.



Slika 18: Skočni prozor LastPass dodatka

U slijedećem koraku pojavljuje se skočni prozor koji korisnika pita za dopuštenja koja su potrebna dodatku (slika 18). Nakon pritiska gumba za dodavanje proširenja, dodatak je instaliran u nekoliko trenutaka i korisnik ga može početi koristiti.



Slika 19: Dodan dodatak u Google Chrome preglednik

4.4. Vrste web dodataka

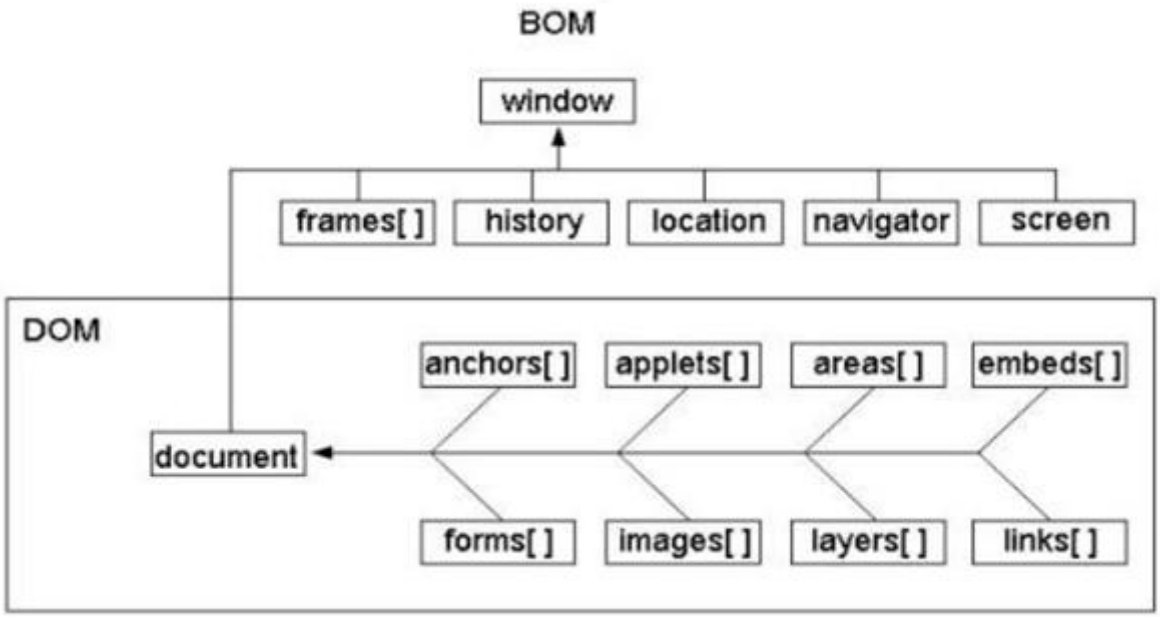
Neke od vrsta dodataka su [22]:

- Poboljšanja ili dopune web stranica (eng. *Enhancements or Additions to a Website*): Dodaci koji se koriste za pružanje dodatnih funkcija preglednika s web stranice. Prikupljanje detaljnih podataka sa stranica koje korisnici posjećuju kako bi se poboljšale usluge koje im se pružaju.
- Personalizacija (*eng. Personalization*): Proširenja preglednika mogu manipulirati web sadržajem. Na primjer, dopuštajući korisnicima da dodaju svoj omiljeni logotip ili sliku kao pozadinu svakoj stranici koju posjete. Proširenja također mogu omogućiti korisnicima da ažuriraju izgled korisničkog sučelja npr. preglednika Google Chrome na isti način kao i samostalni dodaci za teme.
- Dodavanje ili uklanjanje sadržaja s web stranica (*eng. Adding or Removing Content from Web Pages*): Pomoć korisnicima kako bi blokirati nametljive oglase sa web stranica. Preoblikuje sadržaj stranice kako bi se ponudio dosljedan doživljaj čitanja. Uz mogućnost pristupa i ažuriranja HTML-a i CSS-a stranice, proširenja mogu pomoći korisnicima da vide web na način na koji to žele.
- Dodavanje alata i novih značajki pregledavanja *(eng. Adding Tools and New Browsing Features*): Dodavanje novih značajki na ploču zadataka (eng. *Taskboard*) ili generirajte slike QR koda iz URL-ova, hiperveza ili teksta stranice. S fleksibilnim opcijama korisničkog sučelja i mogućnostima Chrome Extension API–a mogu se poboljšati značajke ili funkcionalnost bilo koje web stranice.
- Igre (*eng. Games*): Mogućnost podrške klasičnih računalnih igara sa „off-line“ komponentom. Na primjer, dodavanjem elemenata igara svakodnevnom pregledavanju.
- Dodavanje razvojnih alati (*eng. Adding Developer Tools*): Mogućnost dodavanja novih razvojnih alata u alate za razvojne programere.

4.5. Dokumentno objektni model dokumenta i objektni model preglednika

Dokumentno objektni model (eng. *Document Object Model - DOM*) je API za oblikovanje i interakciju s objektima u HTML, XHTML i XML dokumentima. Čvorovi (eng. *Nodes*) svakog dokumenta organizirani su u strukturi stabla (eng. *Tree Structure*) koja se naziva stablo DOM-a. Objekti u DOM stablu mogu se pristupiti i manipulirati metodama objekta.

Prema [23] objektni model preglednika (eng. *Browser Object Model - BOM*) ponaša se slično kao DOM. Sadrži dodatno informacije o korisniku preglednika koji se koristi. Za razliku od Dokumentno objektnog modela, on ne implementira standarde ili stroge definicije, pa su proizvođaći preglednika slobodni implementirati BOM na bilo koji način.



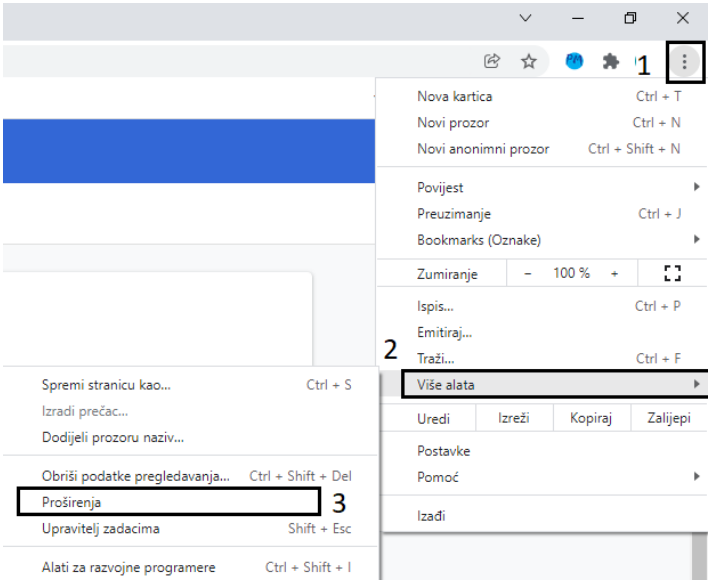
Slika 20: Dom i Bom [24]

Na slici 20 vidimo, da je DOM dio BOM-a. BOM je zapravo preglednik i on se sastoji od okvira (*eng. Frames*), povijesti (*eng. History*), lokacije (*eng. Location*), navigacije (*eng. Navigator*) i zaslona (*eng. Screen*). Naravno BOM se još sastoji i od dokumenta (*eng. Document*) koji je opisan uz pomoć DOM-a. DOM unutar sebe još sadrži sidra (*eng. Anchors*), obrasce (*eng. Forms*), jednostavne aplikacije (*eng. Applets*), slike (*eng. Images*), područja (*eng. Areas*), slojeve (*eng. Layers*), ugrađene element (*eng. Embeds*) i poveznice (*eng. Links*).

5. Razvoj web dodatka za upravljanje lozinkama

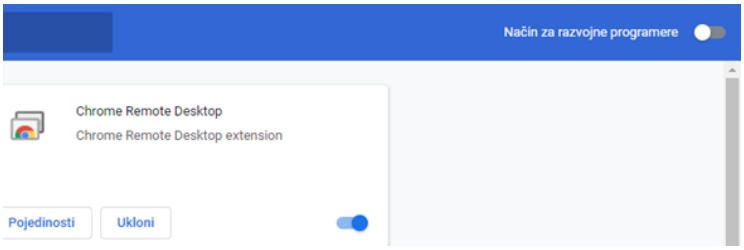
Razvoj web dodatka se je proveo u sljedećim fazama: planiranje, razvoj, testiranje, otklanjanje pogrešaka, implementaciju, nadogradnju i zaštitu dodataka. Za izradu ovog web dodatka koristit će se „Github“ za verzioniranje i stvaranje sigurnosne kopije. Za jednostavnije korištenje „Visual Studio Code“ u sebi sadrži dodatak za kontrolu izvora, odnosno upravljanje sa „Github repozitorijem“. Uz to koristi se „Github Desktop“ za kompliciranije naredbe. Alat „VS Code“ koristiti će se za pisanje izvornog koda i pomoći će sa sintaksnim pogreškama. Web dodatak će se na kraju testirati unutar Google Chrome preglednika koji omogućava jednostavno učitavanje otpakiranih dodataka.

Za učitavanje vlastitog dodatka, prvo što je potrebno je unutar Google Chrome preglednika postavke za proširenja.



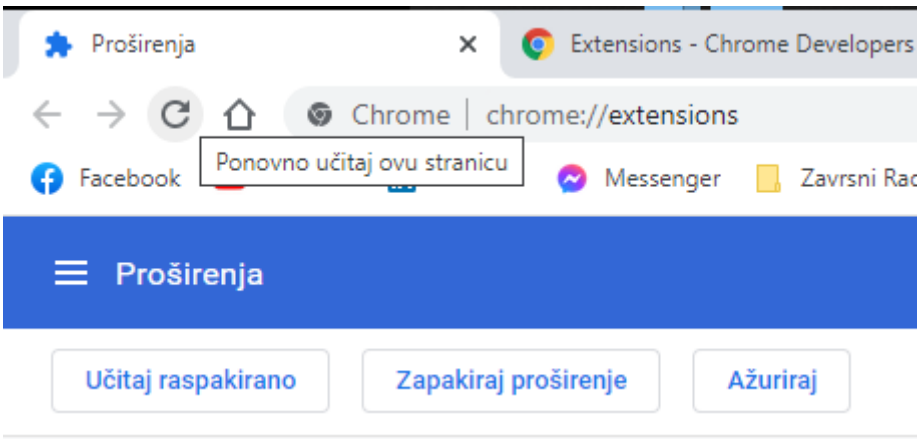
Slika 21: Otvaranje proširenja

Nakon toga potrebno je omogućiti „način za razvojne programere“ koji se nalazi u gornjem desnom kutu postavki za proširenja.



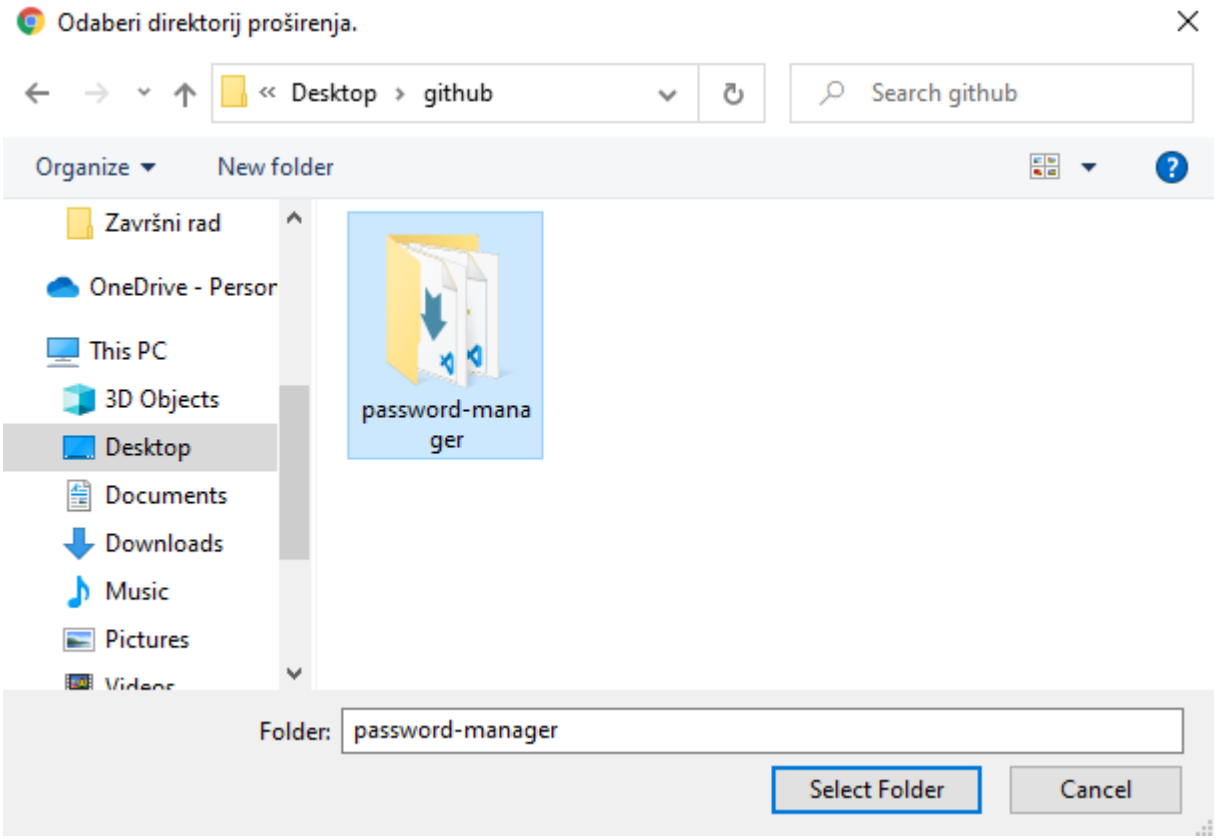
Slika 22: Način za razvojne programere

U gornjem lijevom kutu će se pojaviti tri gumba i pritiskom na prvi „učitaj raspakirano“, otvara se prozor za odabir direktorija.



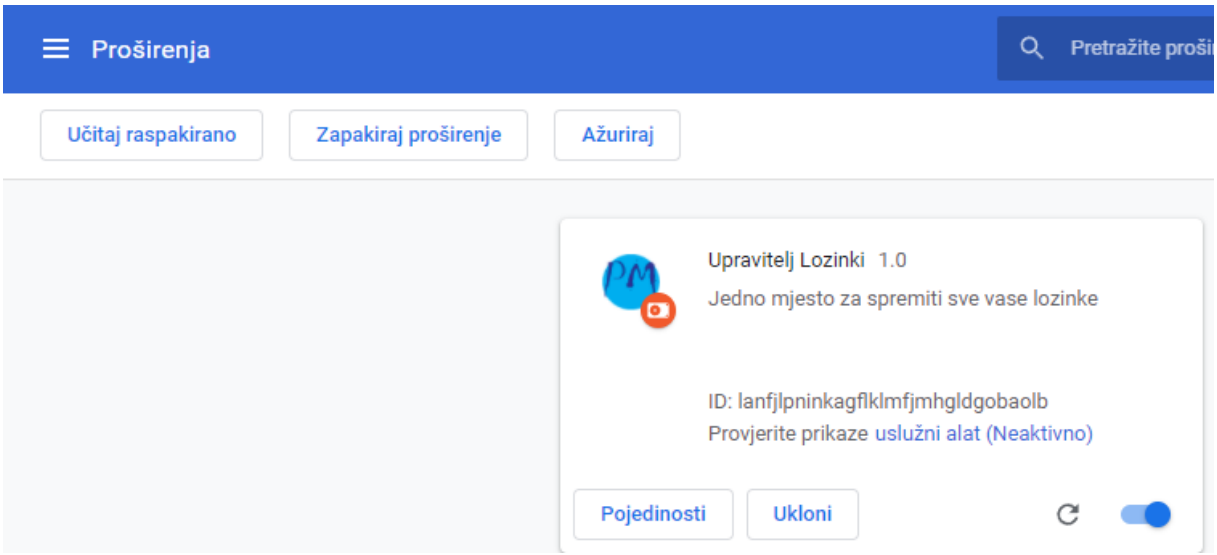
Slika 23: Proširenja - učitaj raspakirano

Potrebno je odabrati direktorij u kojem se nalazi datoteka „manifest.json“ i pritiskom na gumb za odabir direktorija (eng. *Select Folder*) učitan je dodatak.



Slika 24: Proširenja - Učitavanje direktorija

Dodatak je sada vidljiv uz ostala proširenja i moguće ga je testirati.



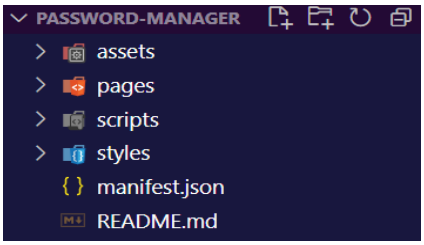
Slika 25: Proširenja - Učitan dodatak

Također, klikom na pojedinosti vidljive su nastale greške prilikom kompiliranja dodatka, te dodatne postavke.

Upravitelj lozinki je dodatak koji pohranjuje sve lozinke korisnika kako ne bi trebao pamtiti sve lozinke koje koristi. Ideja iza ovog dodatka je proizašla iz sve veće potrebe za sigurnosti na Internetu i zahtjeva od stranica za pohranu lozinka na siguran način. Korisnik je mogao koristiti proizvoljnu lozinku bez minimalnih zahtjeva za složenošću, ali s vremenom su korisnici i programeri primijetili kako drugi korisnici iskorištavaju to. Tako da se prijavljuju na račune drugih korisnika i rade zlonamjerne radnje. Iz tog razloga došlo je do potrebe za sve složenijim lozinkama, sve više različitih znamenki, što manje riječi sa značenjem i slično. Kako današnji korisnici svakodnevno koriste Internet, te veliki broj stranica ima mogućnost registracije i prijave, tako i potreba za pamćenjem tih kompliciranih lozinki raste. Iz tog razloga tu postoji „Upravitelj Lozinki“, koji sve lozinke sprema na jedno mjesto i korisnik može jednim klikom miša pristupiti svim svojim lozinkama. A u slučaju promjene lozinke na stranici na kojoj se koristi, također je moguće promijeniti lozinku unutar dodatka.

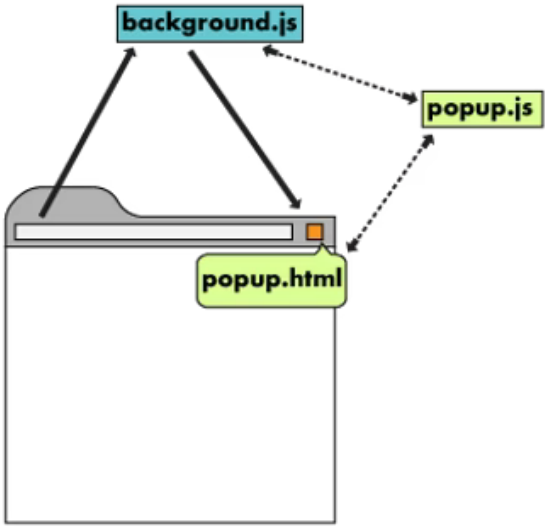
5.1. Arhitektura dodataka i direktorija

Kako bi izvorni kod bio pregledniji i lakše dostupan programeru, cijeli projekt je odvojen unutar poddirektorija (slika 26). Za razvoj je također korišten mikrokernel uzorak dizajna.



Slika 26: Arhitektura direktorija

Unutar „assets“ poddirektorija nalaze se sve slike, kao što su slike korištene za ikone i pozadine. Direktorij „pages“ uključuje za datoteke s nastavkom „.html“ odnosno stranice koje se prikazuju korisniku. Direktorij „scripts“ sadrži sve funkcionalne odnosno JavaScript datoteke. Unutar „styles“ direktorija se još nalaze CSS datoteke, koje određuju izgled svake stranice koju dodatak može otvoriti. Po potrebi moguće je dodati dodatne direktorije kako bi se bolje pojednostavio razvoj i kako sve datoteke ne bi bile na istom mjestu. Za izradu je korišten Google Extension API.



Slika 27: Arhitektura dodatka

Na slici iznad vidimo jednostavan prikaz arhitekture dodatka. Unutar datoteke „popup.html“ definirana je struktura samog dodatka. Unutar skripte „popup.js“ nalaze se funkcionalnosti koje se aktiviraju na zahtjev korisnika (pritiskom nekog gumba unutar proširenja). Unutar skripte „background.js“ nalaze se funkcionalnosti koje se odvijaju u pozadini, bez korisničke interakcije.

5.2. Manifest

Za početak treba napraviti direktorij u kojem će se razvijati web dodatak. Unutar direktorija prvo treba napraviti novu datoteku „manifest.json“. Ta datoteka sadrži ključne informacije potrebne web pregledniku kako bi znao pokrenuti dodatak i omogućiti razvojnom programeru ili korisniku korištenje istog. Pošto je manifest napisan u json-u koristi parove „ključ-vrijednosti“. Obavezni ključevi koji se koriste su „name“ - ime, „description“ - opis, „version“ - verzija, „manifest_version“ - verzija manifesta i „action“ - akcija (odnosno što se želi da dodatak radi). Izborni ključ je „permissions“ unutar kojega se upisuje polje sa svim potrebnim dozvolama kako bi dodatak radio. U primjeru su to „storage“ kako bi omogućili dodatku da pohranjuje podatke i „tabs“ koji dodatku daje prava da upravlja sa elementima na stranici i rad sa karticama.

Sadržaj datoteke manifest.json:

```
{
    "name": "Upravitelj lozinki",
    "description": "Jedno mjesto za spremiti sve vase lozinke!",
    "version": "1.0",
    "manifest_version": 3,
    "action": {
        "default_popup": "pages/popup.html"
    },
    "icons": {
        "16": "assets/icon16.png", 48": "assets/icon48.png",
        "128": "assets/icon128.png"
    },
    "background": {
        "service_worker": "scripts/background.js"
    },
    "permissions": [
        "storage", "tabs"
    ],
    "options_page": "pages/options.html"
}
```

5.3. Struktura

Idući korak je napisati HTML programski kod dodatka, kao što je prije spomenuto on određuje strukturu odnosno gdje se koji element nalazi. Ovdje je bitno uočiti kako svaki element sadrži ili „id“ ili „class“ atribut, to kasnije služi kako bi unutar CSS-a ili JavaScript-a bilo lakše upravljati sa tim elementima.

Sadržaj datoteke popup.html:

```
<!DOCTYPE html>
<head>
    <meta charset="UTF-8">
    <title>Upravitelj Lozinki</title>
    <link rel="stylesheet" href="../styles/popup.css">
</head>
```

U dijelu koda iznad, postavlja se format znakova na odgovarajuću vrijednost, koja je u ovom slučaju „UTF-8“ (eng. *Unicode Transformation Format*) i on omogućuje prikaz hrvatskih dijakritičkih znakova unutar dodatka. Unutar oznaka „link“ postavlja se poveznica na CSS datoteku, koja sadrži stilove potrebne za pravilnu prezentaciju dodatka.

```
<body>
    <h2 id="header">Upravitelj Lozinki</h2>
    <div class="searchField">
        <input id='search' placeholder="Pretraži spremljene šifre">
        <img class='searchIcon' src='../assets/search.svg'>
    </div>
```

U drugom dijelu programskog koda definirani su elementi koji se prikazuju korisniku. Unutar oznake „h2“ (eng. *Heading level 2*) postavljen je naziv dodatka i prikazan je korisniku. Nakon toga definirana je oznaka „div“ (eng. *division*) unutar kojeg su postavljene oznake „input“ i „img“ (eng. *Image*). Oznaka „input“ se koristi za pretraživanje spremljenih lozinki i unutar sebe atribut „placeholder“ u koji korisniku daje doznanja da se radi o unosu za pretraživanje, a oznaka „img“ sadrži ikonu povećala koje korisniku daje dodatno osiguranje da se radi o unosu za pretraživanje.

```
    <div id="passwordsWrapper" class='passwordsWrapper'>
    </div>
    <a id="addItem" class="addNewPasswordButton">Dodaj lozinku</a>
```

```
</body>
<script src="../scripts/popup.js"></script>
</html>
```

U posljednjem dijelu programskog koda definirana je oznaka „div“ u kojem će se kasnije uz pomoć JS-a prikazivati spremljene lozinke. Oznaka „a“ (eng. *Anchor*) se koristi kao gumb. Pritiskom na gumb će se unutar JS-a definirati otvaranje nove kartice unutar koje će biti moguće dodati novu lozinku.

Na kraju programskog koda se nalazi oznaka „script“ u kojem se nalazi poveznica na JS datoteku popup.js koja sadrži funkcionalnosti. Kako bi se dodatku dodale funkcionalnosti potrebno je to napraviti u vanjskoj datoteci jer dodatak ne podržava JS unutar HTML datoteke.

Slika 28: Dodatak HTML

Kao što se vidi na slici 28, dodatak izgleda jako jednostavno, obično i neprivlačno. Ako se tako objavi dodatak i ako postoji neki drugi slični dodatak s ljepšim sučeljem korisnik će brzo izgubiti zainteresiranost. Zbog toga je jako važno paziti da korisničko sučelje koje korisniku daje zadovoljstvo primjene web dodatka. Iz toga razloga postoji CSS.

5.4. Prezentacija

Kako je to praktički mini web stranica, za poboljšanje dizajna možemo koristiti CSS. U primjeru se koristi vanjska CSS datoteka, kako bi kod bio pregledniji i jednostavniji za razumjeti. Kada se radi dizajn odnosno izgled bilo koje aplikacije ili u ovom slučaju dodatka, važno je osvrnuti se ne samo na korisničko sučelje već i na korisničko iskustvo (*eng*. *User interface/User experience*) [3, str. 856-882].

Korisničko sučelje u ovom kontekstu odnosi se na izgled aplikacije, odnosno odgovara na korisnikovo pitanje „Je li lijepo?“ i „Sviđa li mi se?“. Ova dva pitanja su jako subjektivna i ovise od osobe do osobe i njihovim preferencijama, ali kako nije moguće dizajnirati nešto što bi se svima svidjelo, postoje pravila kojih se dizajneri drže. Neka od tih pravila su konzistencija

dizajna, jednostavnost boja, uvijek treba birati boje koje si odgovaraju i nikada se ne bi trebala koristiti prevelika količina boja. Korisničko iskustvo odgovara na pitanja „Uživam li u korištenju ovog dodatka?“. Tijekom razvoja, uvijek je potrebno razmišljati sa stajališta krajnjeg korisnika. Česta greška koju programeri rade je što misle da ako je nešto njima jednostavno i intuitivno da će tako biti i korisnicima. Ova pogreška često nastaje kada programer veliku količinu vremena provodi testirajući i koristeći aplikaciju, a pošto ju je on izradio, njemu sve ima smisla i jednostavno mu je za koristiti.

Radi toga neka od pravila kojih bi se trebalo držati prilikom proširenja web preglednika, kratak i jasan tekst ili još bolje ikone za koje svi znaju što znače. Kao što su ikona „diskete“ koja predstavlja spremanje, ikona „povećala“ koji predstavlja pretraživanje, „x“ zatvaranje i mnoge druge. Važno je korisniku naznačiti da dok je nešto napravio unutar aplikacije da je aplikacija to prepoznala, preko skočnog prozora ili slično. Također važno je ne zamarati korisnika sa previše mogućnosti na početku korištenja.

Kada se koristi CSS-a, pametno je na početku datoteke postaviti neka početna pravila stranice. U ovom primjeru koristi se selektor „*“ koji se odnosi na sve elemente unutar dodatka i pomoću njega se određuje da će svi ostali elementi imati početne vrijednost razmaka i podloge nula i font koji će se koristiti je „Lucida Sans“.

Sadržaj datoteke popup.css

```
* {
    margin: 0;
    padding: 0;
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
}
```

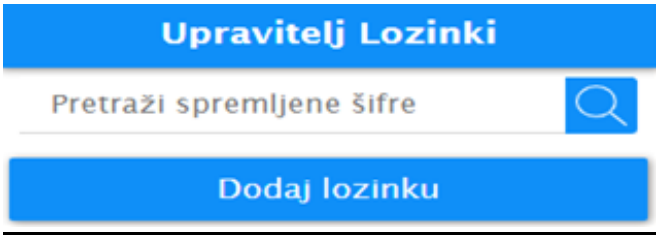
Kako se radi o dodatku unutar kojeg se dodaju nove lozinke i dolazi do potrebe za dinamičnom promjenom veličine. Tu je korištenje Flexbox-a izrazito korisno. Flexbox je predstavljen 2009. godine kao novi sustav rasporeda, sa ciljem da pomogne u izgradnji responzivnih web stranica i jednostavnoj organizaciji elemenata.

```
body {
    width: 300px;
    display: flex;
    padding: 10px;
    flex-direction: column;
    justify-content: center;
    align-items: center;
```

```
}
```

Kao što je prije spomenuto, kako bi korisničko iskustvo bilo bolje i jasnije, važno je korisniku prikazati da ima interakciju sa dodatkom. Ovdje je to implementirano na dva načina, jedan od njih je da prelaskom preko gumba. Gumb „iskače“ i daje korisniku informaciju da je to gumb na koji se može pritisnuti. Drugi je da se kursor miša promjeni u pokazivač. To je moguće sa „:hover“ pseudo klasom. Pseudo klasa u CSS-u je ključna riječ dodana selektoru koja specificira posebno stanje odabranih elemenata. Unutar tog selektora koristimo „box-shadow“ i „cursor“.

```
.addNewPasswordButton:hover {
    box-shadow: 1px 2px 5px rgba(0, 0, 0, 0.7);
    cursor: pointer;
}
```



Slika 29: Dodatak CSS

Kada se svi stilovi dodaju skupa, dodatak izgleda kao na slici 29. Na dodatku je sada vidljiva prezentacija. Naslov dodatka ima plavu pozadinsku boju i bijelu boju teksta kako bi tekst bio čitljiviji. Elementu za pretraživanje su maknuti svi osim donjih obruba i izgleda minimalistički. Gumb za dodavanje lozinke je isto uređen sa istim stilom kao i naslov. Ali ovaj dodatak još nema nikakve funkcionalnosti i ne radi ono što je rečeno da radi, iz tog razloga je tu JavaScript.

5.5. Korisničko iskustvo

Kako bi se implementirala logika koja pokreće cijeli dodatak koristi se JavaScript. JS je trenutno jedan od najtraženijih i najkorištenijih programskih jezika. Titulu je zaslužio sa svojom jednostavnošću, velikom količinom dostupnih materijala za učenje i naravno sve većom potrebom za izradom web stranica sa što više „fora“ funkcionalnosti. Kao i za web stranice, JS se unutar dodatka koristi kako bi korisniku omogućio interakciju s dodatkom. Dodatak prihvaća JS na dva načina, za korištenje na prednjem kraju *(eng. Foreground)* gdje korisnik ima direktnu

interakciju s njim i korištenje na zadnjem kraju (*eng. Background)* gdje radi u pozadini i aktivira se na neki postavljeni događaj.

5.5.1. Glavne funkcionalnosti

Glavne funkcionalnosti ove aplikacije su dodavanje lozinki i korištenje istih. Kako bi se to omogućilo, potrebno je koristiti rukovatelje događaja (*eng. Event Listener*). Rukovatelj događajima je funkcija povratnog poziva koja radi asinkrono i obrađuje ulaz (događaj). U ovom slučaju, događaji su važni elementi aplikacijskih informacija iz temeljnog razvojnog okvira. U smislu grafičkog sučelja, događaji uključuju pritiske tipki, aktivnosti miša, operacije odabira ili odbrojavanje vremena.

Na stranici za unos događaji uključuju otvaranje ili zatvaranje datoteka i tokova podataka, čitanje podataka itd. Dodaci podržavaju samo rukovatelje događaja iz vanjskih datoteka. Tako da je potrebno implementirati cijelu skriptu i unutar nje koristiti metodu „addEventListener“ za svaki događaj koji se želi izvršiti.

Sadržaj datoteke popup.js:

```
function loadList() {
    array = [];
    chrome.storage.sync.get(['list'], function(result) {
        result.list ? array = result.list : array = [];
        loadElements(array);
    });
}
```

Prva važna funkcija koja je „loadList“ i ona uz pomoć Chrome API-a dohvaća sve spremljene lozinke i preko funkcije „loadElements“ ih prikazuje u HTML formatu. Ova funkcija radi na način da prvo makne sve lozinke koje se trenutno prikazuju, nakon toga za svaku lozinku koja postoji se napravi novi skup elemenata koji zajedno čine jednu cjelinu potrebnu za pravilan prikaz.

```
function loadElements(array) {
    document.getElementById("passwordsWrapper").innerHTML = '';
    array.forEach(element => {
        document.getElementById("passwordsWrapper").innerHTML += `
      <div id='item${element.id}' class='singlePasswordContainer'>
```

```
            <div class="singlePasswordNameUser">
                <p class="singlePasswordName">${element.name}</p>
                <p class="singlePasswordUser">${element.email}</p>
            </div>
            <div class="tooltip">
                <p class="tooltiptext">Kopiraj lozinku</p>
                <img class='icon copyIcon' src='../assets/copy.svg'>
            </div>
            <div class="tooltip">
            <p class="tooltiptext">Uredi lozinku</p>
            <img class='icon editIcon' src='../assets/edit.svg'>
            </div>
            <div class="tooltip">
            <p class="tooltiptext">Obriši lozinku</p>
            <img class='icon deleteIcon' src='../assets/delete.svg'>
            </div>
        </div>`;
});
```

U programskom kodu iznad, funkcija generira izgled lozinke u HTML obliku za svaki dohvaćeni element proslijeđen funkciji. Prvo što učini je da obriše sve unutarnje elemente, elementa s id-om „passwordsWrapper“. Nakon toga koristeći petlju „forEach“ prolazi se kroz svaku spremljenu lozinku u polju i postavljaju se vrijednosti pojedinog elementa iz polja. Uz pomoć znaka dolara ($) i vitičastih zagrada ({}), jednostavno se ubacuju vrijednosti iz elementa.

```
var elements = document.getElementsByClassName("deleteIcon");
Array.from(elements).forEach(function(element) {
    element.addEventListener('click', function(params) {
        var id = params.path[2].id[4];
        deleteElement(id);
    });
});
```

U dijelu programskog koda iznad, postavljaju se osluškivač događaja na gumbe u gornjem dijelu funkcije. Za početak dohvaća sve elemente sa klasom „deleteIcon“ i sprema ih u polje, te za svaki element dodaje osluškivač događaja koji se aktivira na pritisak. „Id“ pojedinog elementa dohvaćamo iz „id“. Ovaj „id“ predstavlja jedinstveni broj svake spremljene lozinke. Funkciji deleteElement prosljeđujemo taj „id“.

```
function deleteElement(id) {
    var remove;
    array.forEach((element, index) => {
        if (element.id == id)
            remove = index;
    });
    array.splice(remove, 1);
    document.getElementById('search').value = '';
chrome.storage.sync.set({
        list: array
    }, function() {
        loadList();
    });
}
```

Iznad vidimo funkciju „deleteElement“ koja kao parametar prihvaća id elementa koji želimo izbrisati (element je u ovom kontekstu lozinka). Prvo prolazimo kroz cijelo polje koje sadrži sve lozinke uz pomoć „forEach“ petlje. Tamo uspoređujemo dok ne pronađemo element s odgovarajućim id-om. U pomoćnu varijablu spremamo indeks odgovarajućeg elementa. Koristeći funkciju „splice“ na polju iz polja brišemo odabrani element. Nakon toga uz pomoć Chrome API-a spremamo novo polje u „storage“ i pokrećemo funkciju „loadList“ koja osvježava dodatak.

```
var elements = document.getElementsByClassName("copyIcon");
    Array.from(elements).forEach(function(element) {
        element.addEventListener('click', function(params) {
            var id = params.path[2].id[4];
            var text = array[id].password;
            var name = array[id].email;
            copyPassword(text, name);
        });
    });
```

Na isti način kao za brisanje dodajemo, funkciju za kopiranje. Funkcija za kopiranje služi kako bi se kopirala vrijednost lozinke. Koju je kasnije moguće zalijepiti u unos za lozinke na željenoj stranici. Funkcija isto kao i za brisanje, prvo dohvaća id elementa, preko id-a

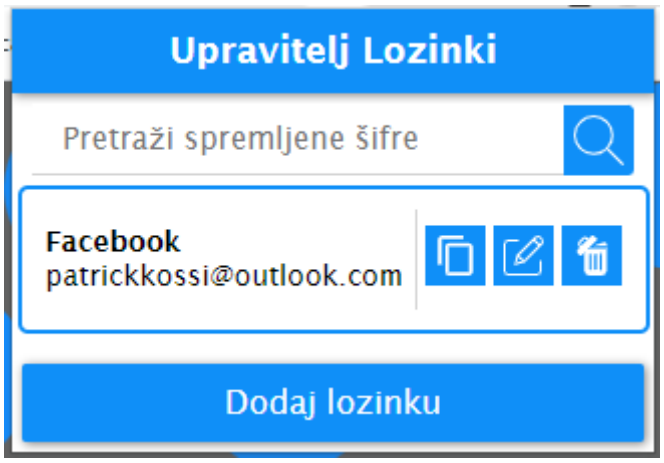
dohvaća spremljenu lozinku i spremljeni email ili korisničko ime elementa s tim id-em i prosljeđuje ih funkciji „copyPassword“.

```
async function copyPassword(text, name) {
    try {
        await navigator.clipboard.writeText(text);
        alert("Copied the password for: " + name);
    } catch (err) {
        console.error('Failed to copy: ', err);
    }
}
```

Funkcija „copyPassword“ (iznad) je asinkrona funkcija koja u međuspremik pohranjuje vrijednost lozinke preko navigator API-a i prikazuje skočni prozor kako bi korisnika obavijestila o kojem računu se radi. U slučaju greške u konzolu se ispisuje poruka, ona je tu radi lakšeg testiranja i otklanjanja grešaka.

```
var elements = document.getElementsByClassName("editIcon");
    Array.from(elements).forEach(function(element) {
        element.addEventListener('click', function(params) {
            var id = params.path[2].id[4];
            chrome.tabs.create({
                url: '../pages/updatePassword.html?id=' + id
            });
        });
    });
}
```

Kako za brisanje i kopiranje, tako i za uređivanje dodajemo osluškivače događaja na gumb za uređivanje. Na isti način dohvaćamo elemente i na isti način dohvaćamo id elementa koji želimo. Uz pomoć „Chrome Extension API-a“ otvara se stranica za uređivanje lozinke. Više o njoj kasnije.



Slika 30: Upravitelj lozinki

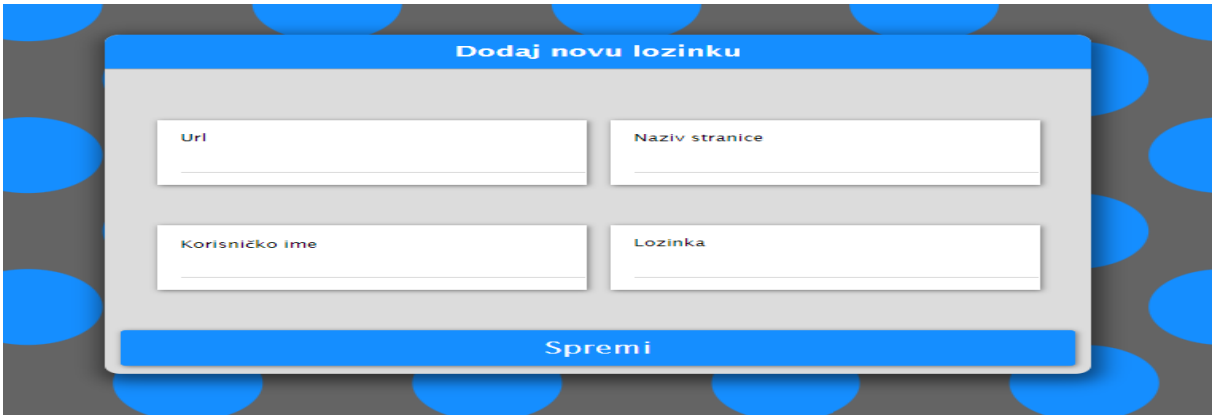
Pritiskom na gumb „Dodaj lozinku“ (slika 30) izvršava se funkcija koja pokreće novi prozor i otvara stranicu za dodavanje lozinke. Ovdje se koristi „chrome.tabs“ API i on omogućava kreiranje novih i upravljanje postojećim prozorima. Stranica se otvara na URL-u koji započinje sa „chrome-extension://“ i završava sa imenom stranice u ovom slučaju „index.html“.

```
document.getElementById('addItem').addEventListener('click', function(e) {
    e.preventDefault();
    chrome.tabs.create({
        url: '../pages/index.html'
    });
});
```

U programskom kodu iznad vidimo implementaciju toga. Važno je napomenuti da kako bi ovo funkcioniralo obavezno treba u manifest.json dodati dozvolu unutar „permissions“ opcija za rad s karticama (*eng. Tab*).

5.5.2. Dodatne funkcionalnosti

Unutar stranice za dodavanje lozinke, nalazi se obrazac (eng. *Form*) koji omogućava dodavanje nove lozinke. Ova stranica ima vlastitu HTML, CSS i JS datoteku. Tako da je za potrebe razvoja lakše pronaći greške i razvija se kao zasebna web stranica. Web stranica na ovakav način podržava „Chrome Extension API“. Tako da je vrlo jednostavno dodati mogućnosti koje sadrži dodatak unutar nje. Sastoji se od funkcije za dodavanje novog elementa koja unutar sebe poziva dohvaćanje ostalih spremljenih lozinki i zatvaranje trenutnog prozora. Spremanje lozinke radi se na način da se svaka posebna i nova lozinka sprema unutar JSON objekta. Dodaje se na kraj polja i to polje su na kraju sprema putem Chrome API-a u memoriju preglednika.



Slika 31: Dodavanje nove lozinke

Na slici 31 vidimo kako izgleda stranica za dodavanje lozinke, sastoji se od obrasca koji se sastoji od naslova, četiri elemenata za unos i gumba za spremanje. U nastavku vidimo HTML kod ove stranice (index.html):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="../styles/index.css">
    <title> Dodaj Lozinku </title>
</head>
```

Početak stranice je standardnog formata, postavljaju se metapodaci za pravilno učitavanje stranice. Učitava se CSS datoteka i postavlja se naslov (*eng. Title*) koji je u ovom slučaju „Dodaj novu lozinku“.

```
<body>
    <div class="formAddNew">
        <h2 class="header">Dodaj novu lozinku</h2>
        <div class="inputContainer">
            <div class="inputMiniContainer">
                <label>Url</label>
                <input id="url" type="url">
            </div>
            <div class="inputMiniContainer">
                <label>Naziv stranice</label>
                <input id="name" type="text">
            </div>
            <div class="inputMiniContainer">
                <label>Korisničko ime</label>
                <input id="email" type="text">
            </div>
            <div class="inputMiniContainer">
                <label>Lozinka</label>
                <input id="password" type="password">
            </div>
        </div>
        <a id="newItem" class="buttonAdd">Spremi</a>
    </div>
</body>
<script src="../scripts/addPassword.js"></script>
</html>
```

Ostatak programskog koda sastoji se od nekoliko „div“ oznaka i „label“ oznaka. Na kraju je dodan gumb koji pokreće skriptu iz „addPassword.js“ datoteke. U nastavku prvo vidimo datoteku index.css:

```
body {
    background-image: url("../assets/background.svg");
    background-attachment: fixed;
}
```

Ovaj dio CSS koda postavlja pozadinu stranice preko „background-image“ osobine na sliku napravljenu koristeći Adobe Illustrator. Osobina „background-attachment“ postavljamo na „fixed“ kako bi slika bila uvijek na jednom mjestu i da se ne pomakne tijekom približavanja stranice.

```
.formAddNew {
    position: absolute;
    height: 400px;
    width: 600px;
    background-color: rgb(220, 220, 220);
    top: calc(50% - 200px);
    left: calc(50% - 300px);
    border-radius: 10px;
    padding: 10px;
    box-shadow: 2px 5px 15px 5px rgba(0, 0, 0, 0.5);
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    align-items: center;
}
```

U ovom djelu koristimo selektor klase za obrazac. Važno je napomenuti osobinu „position“ koje postavljamo na „absolute“. To nam postavlja cijeli obrazac na mjesto koje želimo na stranici. Uz pomoć osobine „top“ i „left“ možemo koristiti funkciju „calc“ za izračunavanje na kojoj poziciji treba biti gornji lijevi rub kako bi obrazac uvijek bio u centru ekrana.

Osobina „border-radius“ koristimo kako bi rubovima obrasca dodali zaobljenje. Ovdje kao i unutar dodatka još koristimo flex-box, koji omogućuje raspodjelu svih elemenata po obrascu. Pomoćne osovine „flex-direction“, „justify-content“ i „align-items“ koristimo kako bi zadali prikaz koji želimo.

```
.header {
    color: white;
    border-bottom: solid 1px darkgray;
    border-top-left-radius: 10px;
    border-top-right-radius: 10px;
    width: 100%;
    margin-top: -10px;
    background-color: var(--blue);
    padding: 10px;
    text-align: center;
}
```

Selektorom klase „header“ uređuje se kako se želi da izgleda naslov na vrhu obrasca. Osobina „border-bottom“ se koristi kako bi samo na donju stranu dodali obrub. To služi kako bi se dodatno odvojio naslov od ostatka obrasca. „Margin-top“ je negativan broj kako bi naslov

postavili iznad postavljenih razmaka (eng. Margin). Unutar osobine „background-color“ koristimo CSS vrijednost koja je postavljena na točnu vrijednost plave nijanse koja se koristi kroz cijeli dodatak.

```
input:focus {
    outline: none;
}
```

Selektor „input:focus“ i CSS uputa „outline:none“ koristimo kako bi se sa elemenata za unos maknuli obrubi tijekom unosa.

```
.inputContainer {
    display: grid;
    grid-template-columns: 270px 270px;
    column-gap: 15px;
    row-gap: 50px;
}
```

Unutar selektora klase „inputContainer“ koristimo osobinu "display“ postavljeno na „gird“ kako bi unose prikazali unutar „grid“-a. Ostale pomoćne osobine postavljamo kako bi prikazali elemente na 2x2 mreži.

Sadržaj datoteke addPassword.js:

```
document.getElementById('newItem').addEventListener('click', function(e) {
    e.preventDefault();
    console.log(array);
    var id = array.length > 0 ? array[array.length - 1].id + 1 : 0;
    array.push({
        id: id,
        email: document.getElementById("email").value,
        name: document.getElementById("name").value,
        password: document.getElementById("password").value,
```

```
        site: document.getElementById("url").value,
    });
```

Isječkom programskog koda iznad dohvaća sa elemenat za unos vrijednosti i postavlja njihove vrijednosti u JSON objekt na odgovarajuća mjesta. Id se generira na način da se uzme posljednji element iz polja i na njegov id se doda broj jedan. U slučaju da još ne postoji element u polju, postavlja se na 0 kao početni id.

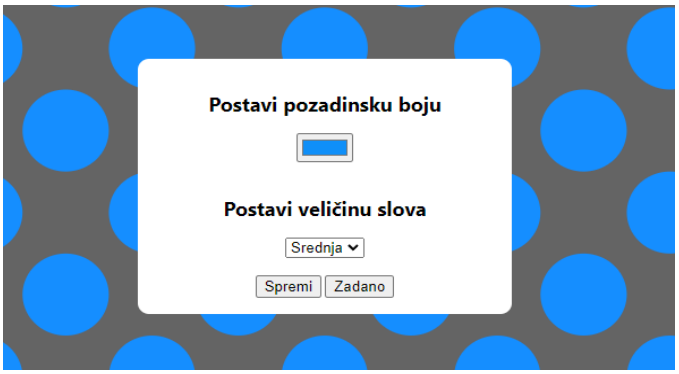
```
    chrome.storage.sync.set({
        list: array
    }, function() {
        closeCurrentTab();
    })
});
```

U programskom kodu iznad sprema se nova lozinka, na način da se prije dohvaćaju trenutne lozinke u lokalno polje. U to polje se dodaje nova lozinka i polje je spremljeno na mjesto starog polja. Nakon toga se poziva funkcija „closeCurrentTab“.

```
    async function closeCurrentTab() {
        let queryOptions = { active: true, currentWindow: true };
        let [tab] = await chrome.tabs.query(queryOptions);
        chrome.tabs.remove(tab.id);
    }
```

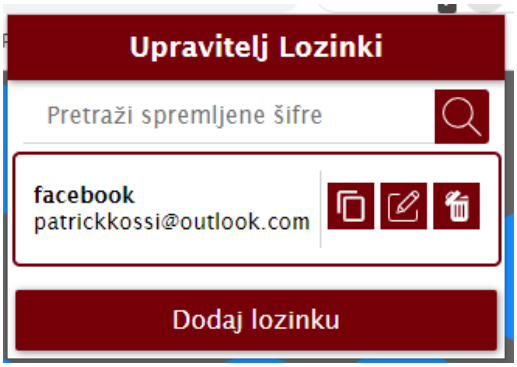
Asinkrona funkcija „closeCurrentTab“ koristi „chrome.tab“ API i pronalazi koji je prozor trenutno aktivan. U svakom slučaju to je prozor za spremanje lozinke. Prosljeđivanjem id-a trenutne kartice u funkciju „remove“, kartica se zatvara. Sada ako pokrenemo dodatak vidi se novo dodana lozinka.

Desnim klikom na dodatak moguće je otvoriti opcije dodatka i to vodi na novu stranicu „options.html“. Unutar nje je moguće postaviti primarnu boju dodatka, zadana primarna boja je svijetlo plava (0f8ff8).



Slika 32: Postavke

Na slici 32 vidimo unos za boju i unos za promjenu veličine slova, te dva gumba, gumb spremi pohranjuje odabranu boju i mijenja se primarna boja dodatka, a zadano ju vraća na nijansu plave koja je postavljena.



Slika 33: Nova boja dodatka

Na slici 33 vidi se kako izgleda dodatak kada se promjeni primarna boja u opcijama. U nastavku je prikazan JS programski kod koji je to omogućio.

Sadržaj datoteke options.js:

```
function save_options() {
    var color = document.getElementById('color').value;
    chrome.storage.sync.set({
        background: color,
    }, function() {
        var status = document.getElementById('status');
        status.textContent = 'Primarna boja promijenjena.';
        setTimeout(function() {
```

```
            status.textContent = '';
        }, 750);
    });
}
```

Funkcija „save_options“ postavlja varijablu „color“ na odabranu boju na stranici opcija. Nakon toga sprema odabranu boju unutar „storage“ i postavlja tekst u elementu „status“ na „primarna boja promijenjena“. Uz pomoć „setTimeout“ metode nakon 750 milisekundi briše tekst.

```
function restore_options() {
    chrome.storage.sync.get({
        background: "#0f8ff8"
    }, function(items) {
        document.getElementById('color').value = items.background;
    });
}
document.addEventListener('DOMContentLoaded', restore_options);
document.getElementById('save').addEventListener('click', save_options);
document.getElementById('default').addEventListener('click', function() {
    chrome.storage.sync.set({
        background: "#0f8ff8",
    }, function() {
        restore_options();
    });
});
```

Funkcija „restore_options“ dohvaća iz „storage“ postavljenu primarnu boju i mijenja vrijednost elementa za biranje boje na trenutno pohranjenu boju. Nakon toga osluškivaći događaja postavljaju događaje na elemente. Na gumb za zadanu boju, u „storage“ se postavlja zadana plava boja i na element za odabir boje se mijenja vrijednost na tu istu plavu boju.

Sadržaj datoteke popup.js:

```
chrome.storage.sync.get({
```

```
    background: "#0f8ff8"
}, function(items) {
    let root = document.documentElement;
    root.style.setProperty("--blue", items.background);
});
```

Ovaj dio programskog koda se dodaje na početak datoteke „popup.js“ i služi kako bi promijenio vrijednost CSS varijable „—blue“ u primarnu boju. Način na koji se to radi je da se prvo dohvati primarna boja iz „storage“-a i uz pomoć „setProperty“-a se mijenja.

5.5.3. Pozadinske funkcije

Ovaj web dodatak sadrži jednu pozadinsku funkciju. Ta funkcija, otvara stranicu za dodavanje lozinki prilikom prve instalacije. Koristeći „chrome.runtime“ API funkcija provjerava dali se radi o instalaciji i kada je to slučaj otvara novu karticu.

Sadržaj datoteke background.js:

```
chrome.runtime.onInstalled.addListener((reason) => {
    if (reason === chrome.runtime.OnInstalledReason.INSTALL) {
        chrome.tabs.create({
            url: 'index.html'
        });
    }
});
```

U programskom kodu iznad uz pomoć „chrome.runtime“ API-a postavljamo osluškivać koji čeka na događaj instalacije dodatka. Provjerava je li uvjet istinit i ako je uz pomoć „chrome.tabs“ API-a otvara stranicu.

5.6. Uklanjanje pogrešaka

Otklanjanje pogrešaka je postupak pronalaženja i uklanjanja postojećih i potencijalnih pogrešaka (*eng. Bug*) u programskom kodu koje mogu uzrokovati nenormalan rad ili rušenje dodataka. Ispravljanje pogrešaka koristi se za pronalaženje i rješavanje pogrešaka ili nedostataka. Kada su različiti podsustavi ili moduli usko povezani, ispravljanje pogrešaka postaje teže, jer svaka promjena u jednom modulu može uzrokovati više pogrešaka u drugom

modulu. Ponekad ispravljanje pogrešaka u dodatku zahtijeva više vremena nego programiranje dodatka.

Koristan alat za uklanjanje pogrešaka je ESLint [25]. On statički analizira kod kako bi brzo pronašao probleme. Ugrađen u većinu uređivača teksta, a može se lako dodati u VS Code preko proširenja. Greške koje ESLint pronađe mogu se riješiti automatski.

ESLint popravljanje razumije sintakse pa neće doći do pogrešaka koje uvode tradicionalni algoritmi za pronalaženje i zamjenu. ESLint koristi stablo apstraktne sintakse (*eng. Abstract Syntax Tree*) a to predstavlja stablo izvornog koda računalnog programa koji prenosi strukturu izvornog koda. Svaki čvor u stablu sintakse predstavlja konstrukt koji se nalazi u izvornom kodu.

Najčešće vrste pogrešaka koje su se dogodile tijekom razvoja aplikacije su sintaksne pogreške (*eng. Syntax Errors*). To su pogreške koje nastaju kada programer pogrešno napiše naziv funkcije ili varijable ili na pogrešno mjesto stavi točku, zarez, itd. One su također najlakše za razriješiti i VS Code samostalno primjećuje i obavještava programera na iste.

```
chrome.storage.sync.set({
        list: array
    }, function() {
        loadList();
    );
}
```

Slika 34: Sintaksna pogreška

Na slici 34 vidi se sintaksna pogreška. VS Code označava sa crvenom vijugavom crtom gdje je moguća greška i ako se strjelicom miša pređe preko nje napiše nam moguću pogrešku. U ovom slučaju nedostaje vitičasta zagrada da se zatvori funkcija.

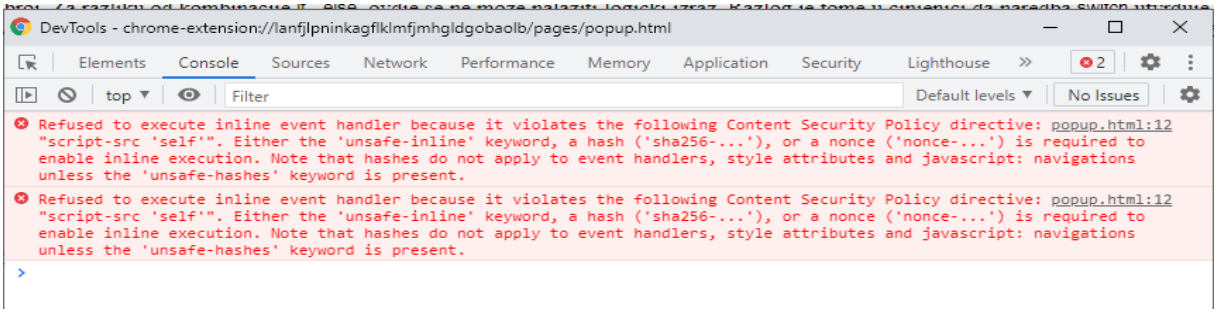
Logičke pogreške (*eng. Logic Errors*) koje je nekad teže pronaći od sintaksnih su također bila česta pojava. Uz pomoć konzole (unutar preglednika) jednostavno se rješava, na način da se ispiše svaki korak i vrijednost željene varijable u tom koraku.

```
array.forEach((element, index) => {
    if (element.id != id)
        remove = index;
});
```

Slika 35: Logička pogreška

Na slici 35 vidi se logička pogreška koja ne prikazuje nikakve obavijesti da postoji. Greška se ni ne primjećuje tijekom izvršavanja (ali postoji). Problem je što se traži element s jednakim id-em kao varijabla, ali unutar „if“ upit provjerava jesu li id-evi različiti i tu nastaje greška.

Posljednja greška koja se najmanje javljala je greška u izvođenju (*eng. Runtime Error*). Ona se događala tijekom pogrešnog pozivanja funkcija ili postavljanja poziva na pogrešno mjesto. Ove greške su onemogućile rad dodataka, ali su bile lako za pronaći, jer su se ispisivale samostalno unutar konzole. Prva greška koje se dogodila tijekom razvoja dodataka je bila greška u izvođenju. Dogodila se zato što se pokušao napraviti poziv funkcije unutar HTML-a, međutim dodatci to ne podržavaju.



Slika 36: Greška u izvođenju

Na slici 36 vidimo unutar konzole grešku tijekom izvođenja. Na svakoj grešci, sa desne strane vidimo poveznicu na datoteku u kojoj se nalazi greška i na kojoj liniji. Ako pročitamo što nam kaže možemo zaključiti što se radi. Ako ne znamo možemo lako upisati grešku u tražilicu i ona će nam ponuditi moguća rješenja. Ova greška se javlja kada se unutar HTML-a pokuša napraviti JS poziv funkcije.

5.7. Zaštita, objava dodataka i upravitelj zadataka

Kako bi se zaštitio dodatak i kako bi se zaštitio izvorni kod od krađe ili ponovnog iskorištavanja postoji tehnika koja se zove „prikrivanje“ (eng. Obfuscation). Prikrivanje znači učiniti nešto teško razumljivim.

Kriptiranje dijela ili cijelog programskog koda jedna je od metoda prikrivanja. Ostale metode uključuju uklanjanje potencijalno propuštenih metapodataka, zamjenu naziva klasa i varijabli besmislenim oznakama te dodavanje neiskorištenog ili besmislenog koda u skripte aplikacije. Alat koji se naziva „obfuscator“ automatski pretvara jednostavan izvorni kod u program koji radi na isti način, ali ga je teže čitati i razumjeti. Postoji mnogo online stranica

koje prikrivaju kod i teško ga je razumjeti. Primjer ovdje je funkcije „loadList()“, korištenjem metode prikrivanje, praktički je nemoguće razumjeti što radi taj dio koda.

```
function _0x49c2(_0x1b6e1c,_0x1d4f7c){var _0x2d2f63=_0x2d2f();return
_0x49c2=function(_0x49c279,_0x30e8ad){_0x49c279=_0x49c279-0x18d;var
_0xfa75c3=_0x2d2f63[_0x49c279];return
_0xfa75c3;},_0x49c2(_0x1b6e1c,_0x1d4f7c);}function _0x2d2f(){var
_0x2bac3f=['72UXLDPI','1644geTMvZ','3951qzghPs','sync','284976ZgYowf','3915
360xSJzpW','116382UxoKvK','get','list','734146cTXawa','1770324bmVQlb','3440
87nrJuMI','storage'];_0x2d2f=function(){return _0x2bac3f;};return
_0x2d2f();}(function(_0x2a8645,_0x465dae){var
_0x4a866b=_0x49c2,_0x539f43=_0x2a8645();while(!![]){try{var _0x2b3dee=-
parseInt(_0x4a866b(0x198))/0x1+-parseInt(_0x4a866b(0x193))/0x2+-
parseInt(_0x4a866b(0x18f))/0x3*(-
parseInt(_0x4a866b(0x18e))/0x4)+parseInt(_0x4a866b(0x192))/0x5+-
parseInt(_0x4a866b(0x197))/0x6+parseInt(_0x4a866b(0x196))/0x7+parseInt(_0x4
a866b(0x18d))/0x8*(-
parseInt(_0x4a866b(0x191))/0x9);if(_0x2b3dee===_0x465dae)break;else
_0x539f43['push'](_0x539f43['shift']());}catch(_0x39ffb7){_0x539f43['push']
(_0x539f43['shift']());}}}(_0x2d2f,0x6d1d1));function loadList(){var
_0x29cfab=_0x49c2;array=[],chrome[_0x29cfab(0x199)][_0x29cfab(0x190)][_0x29
cfab(0x194)]([_0x29cfab(0x195)],function(_0x126a5c){var
_0xb97c3e=_0x29cfab;_0x126a5c[_0xb97c3e(0x195)]?array=_0x126a5c['list']:arr
ay=[],loadElements(array);});}
```

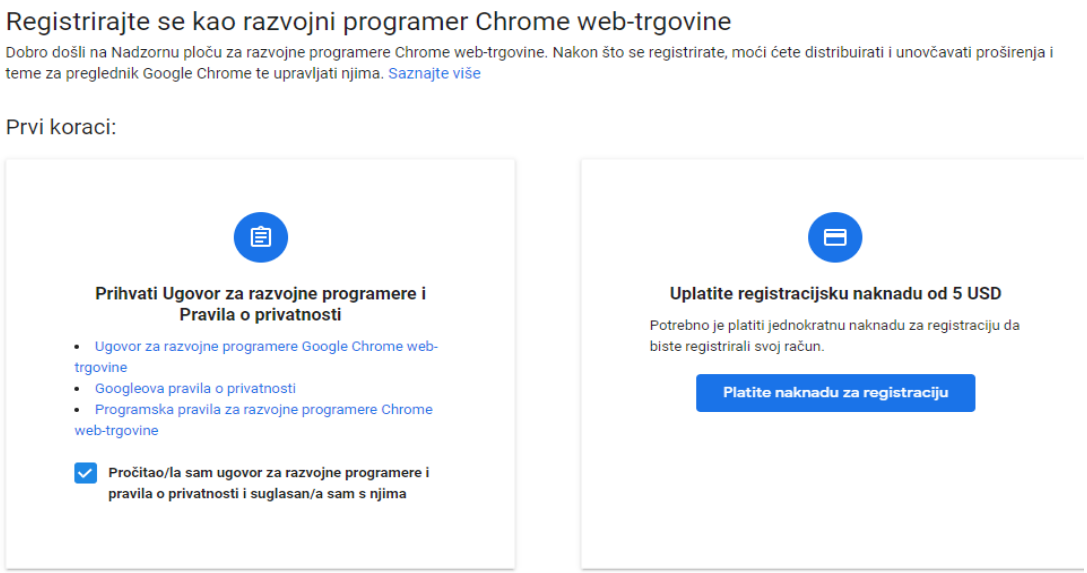
Ovaj način pisanja koda programera štiti od neovlaštenog korištenja i reverznog inženjerstva njegovog koda.

Originalni kod izgleda ovako:

```
function loadList() {
    array = [];
    chrome.storage.sync.get(['list'], function(result) {
        result.list ? array = result.list : array = [];
        loadElements(array);
    });
```

Prikriveni kod je jako teško vratiti u originalni, nema pouzdanih alata koji to mogu automatski napraviti, tako da tako nešto treba ručno raditi i to jako dugo traje.

Za objavu dodatka na Chrome Web trgovinu prvo što treba je napraviti programerski račun na njihovoj stranici. Tokom registracije potrebno je platiti pet američkih dolara (5$) i prihvatiti uvjete korištenja (slika ispod). U Republici Hrvatskoj to trenutno nije moguće jer prilikom naplate nije moguće odabrati Republiku Hrvatsku kao zemlju naplate.



Slika 37: Registracija programera

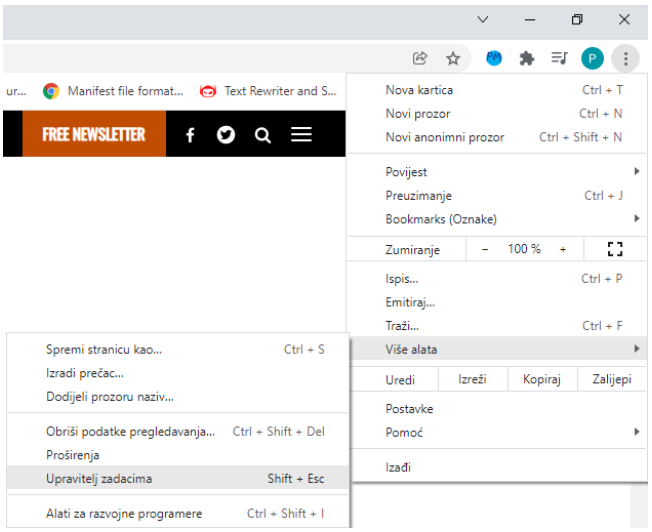
Nakon uplate i registracije, otvara se sučelje za razvojne programere, gdje je moguće pritisnuti na gumb za dodavanje nove stavke. Nakon toga moguće je prenijeti „zip“ datoteku proširenja na stranicu. Ako je manifest datoteka valjana, dodatak odlazi na pregled od strane Chrome-a. Pregled datoteke ovisi o njezinoj složenosti. Traje između nekoliko sati do nekoliko dana. Ako je pregled prošao uspješno, dodatak se može objaviti i nalaziti će se unutar Chrome web trgovine [26].

Svaki dan se preuzme 4 milijuna Chrome-ovih dodataka. Chrome web trgovina broji više od 250.000 dodataka i tema. Ne postoje dva ista Chrome preglednika. Od alata za produktivnost i učenje do zabave i kupovine. Chrome-ovi dodaci otvaraju novi svijet mogućnosti, omogućujući korisnicima da prilagode svoje iskustvo i pomaže im u obavljanju posla. Google vodi računa da dodaci koje su razvili programeri ispune očekivanja privatnosti i sigurnosti korisnika.

Neke od novih mogućnosti koje će dodati su [27]:

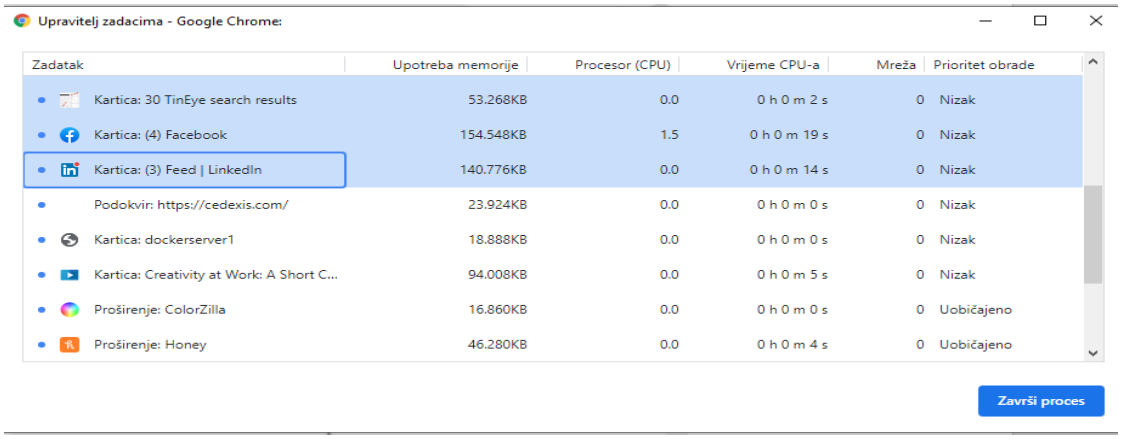
- Stroža pravila privatnosti i bolja kontrola podataka: Promijenit će način na koji dodaci pristupaju podacima i kako rade dopuštenja kada su dodaci instalirana. Korisnici će moći odrediti koje web stranice dodatak može posjetiti tijekom pregledavanja weba. Ova ažuriranja slijede druge promjene koje je Google napravio, kada je dodao ikonu slagalice na alatnoj traci. Svrha toga je da proširene kontrole učinili vidljivijim. Nakon što korisnik dodatku da dopuštenje za pristup podacima web stranice, ta se postavka može spremiti samo za tu domenu. Također može odlučiti odobriti dodatku pristup svim web stranicama koje posjećuje, ali to više nije zadana postavka.
- Transparentna upotreba podataka dodatka: Google je također poboljšao pravila za razvojne programere kako bi dodatke učinili transparentnijim. Svaki će dodatak javno prikazivati svoju praksu privatnosti te će koristiti jasne vizualne efekte i jednostavan jezik za objašnjenje podataka koje prikuplja i koristi. Također ograničava mogućnosti programera s podacima koje prikupljaju.
- Dodatna sigurnosna ažuriranja kako bi osigurali sigurnost korisnika: Google je ažurirao svoju sigurnosnu praksu kako bi lakše identificirali štetniji dodaci prije nego što uđu u Chrome web trgovinu. Na primjer, zahvaljujući njihovoj integraciji s Google-ovim sigurnim pregledavanjem, broj zlonamjernih dodataka koja Chrome onemogućuje radi zaštite ljudi povećan je za 81%. Google je ažurirao i sigurnosne provjere u Chrome-ovim postavkama kako bi pomogli korisnicima da brzo otkriju jesu li instalirani štetni dodaci i da nauče kako ih ukloniti.

Većina operacijskih sustava ima ugrađeni upravitelj zadataka ili monitor resursa koji omogućuje da pregled svih aktivnih procesa i programa koji se izvode na računalu. Web preglednik Chrome također sadrži upravitelj zadataka koji omogućuje zaustavljanje problematičnih kartice i proširenja.



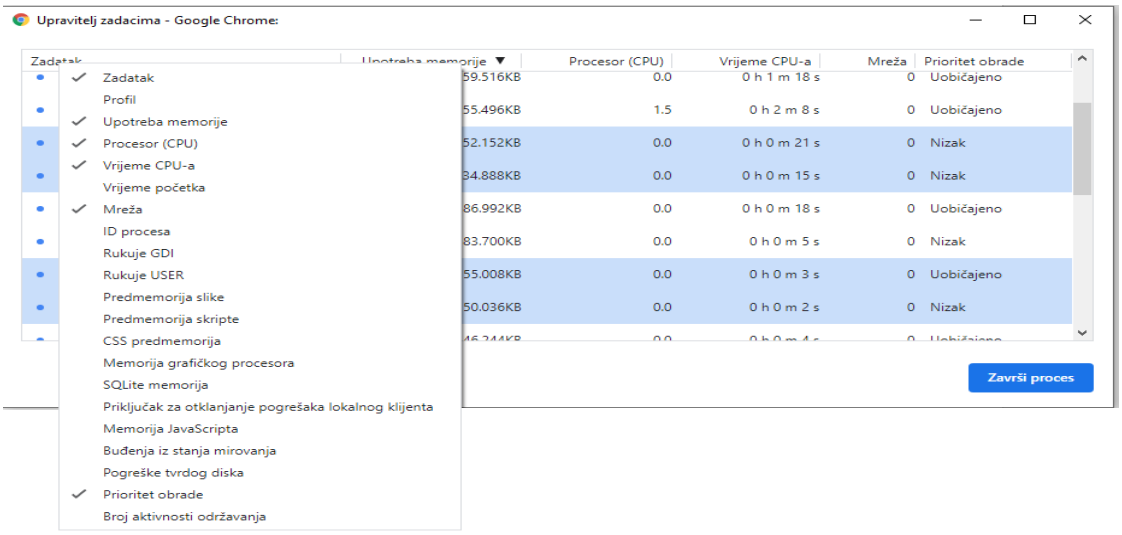
Slika 38: Otvoranje upravitelja zadataka

Na slici iznad, prikazano je kako doći do upravitelja zadataka preglednika, pritiskom na tri točkice u gornjem desnom kutu, odabirom „više alata“ unutar otvorenog prozora i tamo je vidljiv.



Slika 39: Upravitelj zadataka preglednika Chrome

Kada je sada otvoren Chrome-ov Upravitelj zadataka, moguće je vidjeti popis svih kartica, proširenja i procesa koji se trenutno izvode u pregledniku (slika 39). Moguće je završiti bilo koji od procesa iz ovog izbornika, što može biti od pomoći kada proširenje ili kartica prestane reagirati. Da bi se to učinilo, potrebno je odabrati proces, a zatim pritisnuti gumb "Završi proces". Moguće je zaustaviti više od jednog procesa u isto vrijeme držanjem tipke Shift ili Ctrl, označite više stavki s popisa, zatim pritiskom na gumb "Završi proces".



Slika 40: Kategorije statistike upravitelja zadataka

Chrome ima 21 kategoriju statistike koje je moguće dodati kao nove stupce (slika 40). Desnom tipkom miša pritisne se „zadatak“ i pojavit će se kontekstni izbornik s potpunim popisom dostupnih statistika koje je moguće izabrati. Pritiskom na bilo koju dodatnu kategoriju ona se dodaje Upravitelju zadataka. Kategorije koje imaju kvačicu pored sebe već su prikazane. Za uklanjanje određene statistike, potrebno je pritisnuti neželjenu statistiku ponovno. Moguće je i sortirati određene stupce klikom na naslov. Na primjer, klikom na stupac „upravljanje memorijom“, proces koji zauzima najviše memorije bit će razvrstan na vrh popisa.

5.8. Testiranje i upravljanje greškama

Testiranje softvera je metoda provjere ispunjava li stvarni softverski proizvod očekivane zahtjeve i osigurava da softverski proizvod ne sadrži nedostatke. Uključuje uporabu ručnih ili automatiziranih alata za izvršavanje softverskih/sistemskih komponenti za procjenu jednog ili više atributa od interesa. Svrha testiranja softvera je identificiranje pogrešaka prilikom izvršavanja. Kod testiranja moguće je pronaći greške koje se događaju s nepravilnim i neočekivanim korištenjem.

Kako se korisnik ne bi morao suočavati sa greškama i neočekivanim ishodima tijekom korištenja aplikacije, unutar JS-a se dodaju „try-catch“ metode. Naredba „try ... catch„ sastoji se od bloka try koji sadrži jedan ili više izraza i bloka „catch“ unutar kojeg se nalaze izrazi koji određuju što treba izvršiti kada se dogodi iznimka u bloku „try“. Ta naredba zaustavlja nepotrebno zaustavljanje dodatka i omogućava korisniku daljnje korištenje istog.

6. Zaključak

Izrada završnog rada omogućila mi je unapređenje vlastitog znanja o internetu i tehnologijama koje se koriste za razvoj web dodataka u pregledniku. Kroz rad sam shvatio kako je jednostavno započeti i baviti se razvojem web dodataka. Internet je bogat dokumentima i resursima za učenje razvoja web dodataka, tako da nije problem u vrlo kratkom roku naučiti sve što je potrebno za izradu web dodatka. Za web dodatak postoje dvije definicije a to su umetak i proširenje (dodatak). Razlika je da se za dodatak izvorni kod dodaje u preglednik, a za umetak unaprijed kompilirane datoteke, umetak se više ne koristi.

U radu su nabrojani alati i tehnologije koje se primjenjuju za razvoj dodatka. Svaki alat je detaljno opisan i prikazano je njegovo sučelje. Najvažniji i najkorišteniji alat za izradu dodatka je bio VS Code, koji je omogućio brz i jednostavan razvoj dodatka za preglednik Chrome. Tehnologije koje se koriste za izradu su HTML, CSS i JS. Dodatno je primijenjen API Chrome dodataka koji omogućuje da dodatak komunicira sa web preglednikom i na taj način razmjenjuju potrebne podatke. Web preglednik je program koji poznaje sintaksu HTML jezika i na temelju prepoznatih elemenata iz učitanog HTML dokumenta prikazuje na ekranu njima pripadajuće objekte. Trenutno najkorišteniji preglednik na većini platforma je Chrome. Web dodatak je razvijen po mikrokernel uzorku dizajna. Mikrokernel uzorak dizajna pruža veliku podršku evolucijskom dizajnu i inkrementalnom razvoju. Ovaj uzorak dizajna omogućuje dodavanje novih značajki u web preglednik bez mijenjanja izvornog koda preglednika. Razvoj web dodatka je bio proveden u nekoliko faza: planiranje, razvoj, testiranje, otklanjanje pogrešaka, implementaciju, nadogradnju i zaštitu dodataka. Za praktični dio rada napravljen je web dodatak upravitelj lozinki. To je dodatak koji pohranjuje lozinke korisnika kako ih ne bi trebao sve pamtiti. Glavne funkcionalnosti ove aplikacije su dodavanje lozinki, korištenje istih i promjena dizajna. Važna je zaštita izvornog kod dodatka od krađe ili ponovnog iskorištavanja i u svrhu se koristi tehnika prikrivanja. Web dodatak je nakon završetka razvoja moguće objaviti na Chrome web trgovini gdje on postaje dostupan za preuzimanje.

Po mojemu mišljenju ovaj rad mi se pokazao kao zanimljiv izazov i preko njega sam naučio jako puno o načinu rada preglednika, o web dodacima preglednika i o samom razvoju istih. Web dodaci preglednika su jako korisni i smatram da bi svaka osoba koja koristi preglednik trebala znati o njima i njihovom načinu rada.

Popis literature

[1] McLaughlin, B., What is HTML5?. O'Reilly Media, 2011.

[2] Cern., „Word Wide Web“ [Na Internetu]. Dostupno: http://info.cern.ch/hypertext/WWW/TheProject.html [Pristupano 20.6.2021.]

[3] Meyer E., Css: The definitive guide. O'Reilly Media, 2017.

[4] W3.org., „A brief history of CSS until 2016.“ [Na Internetu]. Dostupno: https://www.w3.org/Style/CSS20/history.html [Pristupano 20.6.2021.]

[5] Developer.mozilla.org., „What is JavaScript? - Learn web development“ [Na Internetu]. Dostupno: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript [Pristupano 1.7.2021.]

[6] W3schools.com., „JavaScript Versions.“ [Na Internetu]. Dostupno: https://www.w3schools.com/js/js_versions.asp [Pristupano 27.6.2021.]

[7] MuleSoft. „What is an API?“ [Na Internetu]. Dostupno: https://www.mulesoft.com/resources/api/what-is-an-api [Pristupano 15.6.2021.]

[8] Chrome Developers., „Using promises“ [Na Internetu] Dostupno: https://developer.chrome.com/docs/extensions/mv3/promises/ [Pristupano 4.1.2021.]

[9] Chrome Developers., „Extensions quality guidelines FAQ - Chrome Developers.“ [Na Internetu]. Dostupno:https://developer.chrome.com/docs/extensions/mv3/single_purpose/ [Pristupano 15.9.2021.]

[10] Developer.mozilla.org., „Building a cross-browser extension“. [Na Internetu]. Dostupno: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Build_a_cross_browser_extension [Pristupano 4.1.2021.]

[11] Gaubys J., „Most popular web browsers in 2021.“ [Na Internetu]. Dostupno: https://www.oberlo.com/statistics/browser-market-share [Pristupano 16.9.2021.]

[12] McPeak, A., „A Brief History of Web Browsers and How They Work“. [Na Internetu]. Dostupno: https://smartbear.com/blog/history-of-web-browsers/ [Pristupano 23.6.2021.]

[13] Statcounter Global Stats., „Browser Market Share Worldwide.“ [Na Internetu]. Dostupno: https://gs.statcounter.com/browser-market-share#monthly-200901-202109 [Pristupano 23.6.2021.]

[14] Martin, J., „Chrome, Firefox, Edge, Safari and more“ [Na Internetu]. Dostupno: https://www.techadvisor.com/test-centre/software/best-web-browsers-3635255/ [Pristupano 23.6.2021.]

[15] HTML5test.com. „The HTML5 test“. [Na Internetu]. Dostupno: https://html5test.com/ [Pristupano 19.9.2021.]

[16] Caniuse.com., „Can I use...“ [Na internetu]. Dostupno: https://caniuse.com/ [Pristupano 19.9.2021.]
[17] Raghuwanshi M., „How does web browsers work?.“ [Na Internetu]. Dostupno: https://medium.com/@monica1109/how-does-web-browsers-work-c95ad628a509 [Pristupano 16.7.2021.]
[18] Chrome Developers., „Extensions - Chrome Developers“ [Na Internetu]. Dostupno: https://developer.chrome.com/docs/extensions/ [Pristupano 16.7.2021.]
[19] Richards M. „Software Architecture Patterns“ [Na Internetu]. Dostupno: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch03.html [Pristupano 20.8.2021.]
[20] Mcleanbyron, “Create custom Explorer bars,”. [Na Internetu]. Dostupno: https://docs.microsoft.com/en-us/windows/win32/shell/band-objects [Pristupano 15.7.2021.]
[21] Google Chrome webstore, „LastPass“, [Na Internetu]. Dostupno: https://chrome.google.com/webstore/detail/lastpass-free-password-ma/hdokiejnpimakedhajhdlcegeplioahd?hl=hr [Pristupano 16.9.2021.]
[22] Developer.mozilla.org., „What are extensions?“ [Na Internetu] Dostupno: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/What_are_WebExtensions [Pristupano 10.9.2021.]
[23] Medium. „DOM & BOM Revisited.“ [Na Internetu]. Dostupno: https://medium.com/@fknussel/dom-bom-revisited-cf6124e2a816 [Pristupano 18.9.2021.]
[24] Stack Overflow., „What is the DOM and BOM in JavaScript?.“ [Na Internetu]. Dostupno: https://stackoverflow.com/questions/4416317/what-is-the-dom-and-bom-in-javascript [Pristupano 20.9.2021.]
[25] OpenJS Foundation, „ESLint“ [Na Internetu]. Dostupno: https://eslint.org/ [Pristupano 1.10.2021.]
[26] Chrome.google.com., „Chrome Web Store.“ [Na Internetu]. Dostupno: https://chrome.google.com/webstore/devconsole [Pristupano 29.9.2021.]
[27] Google. „Making Chrome extensions more private and secure.“ [Na Internetu]. Dostupno: https://blog.google/products/chrome/making-chrome-extensions-more-private-and-secure/ [Pristupano 29.9.2021.]

Popis slika

Slika 1: Sučelje alata Visual Studio Code 2
Slika 2: Sučelje alata Adobe Illustrator 3
Slika 3: Sučelje alata GitHub Desktop 4
Slika 4: Sučelje Chrome DevTools alata 5
Slika 5: Sučelje JavaScript Obfuscator alata 5
Slika 6: Prva Web Stranica [2] 7
Slika 7: Prikaz verzija ECMAScript-a [6] 9
Slika 8: Chrome web-trgovina 10
Slika 9: Firefox dodaci za preglednik 14
Slika 10: Grafički prikaz korištenja preglednika u proteklih 12 godina [13] 16
Slika 11: Sučelje Chrome web preglednika 17
Slika 12: HTML5 podrška preglednika [15] 18
Slika 13: CSS podrška preglednika [16] 18
Slika 14: Komponente preglednika [17] 20
Slika 15: Mikrokernel uzorak dizajna [19] 21
Slika 16: Prvi web dodatak [20] 23
Slika 17: Chrome web-trgovina LastPass dodatka [21] 24
Slika 18: Skočni prozor LastPass dodatka 24
Slika 19: Dodan dodatak u Google Chrome preglednik 25
Slika 20: Dom i Bom [24] 26
Slika 21: Otvaranje proširenja 27
Slika 22: Način za razvojne programere 27
Slika 23: Proširenja - učitaj raspakirano 28
Slika 24: Proširenja - Učitavanje direktorija 28
Slika 25: Proširenja - Učitan dodatak 29
Slika 26: Arhitektura direktorija 30

Slika 27: Arhitektura dodatka....30
Slika 28: Dodatak HTML....33
Slika 29: Dodatak CSS....35
Slika 30: Upravitelj lozinki....40
Slika 31: Dodavanje nove lozinke....41
Slika 32: Postavke....46
Slika 33: Nova boja dodatka....46
Slika 34: Sintaksna pogreška....49
Slika 35: Logička pogreška....49
Slika 36: Greška u izvođenju....50
Slika 37: Registracija programera....52
Slika 38: Otvoranje upravitelja zadataka....54
Slika 39: Upravitelj zadataka preglednika Chrome....54
Slika 40: Kategorije statistike upravitelja zadataka....55

Popis tablica

Tablica 1: Analiza uzorka [19].....22