

**SPECIALIZAREA SISTEME DISTRIBUITE ÎN INTERNET**

Lucrare de Disertație

Filtrarea conținutului web pe baza intereselor utilizatorului

**Coordonator științific**

Lect. Dr. Adriana Guran

**Absolvent**

Andreea-Sorana Ciobănița

**2021**

# Cuprins

[Cuprins 1](#_Toc78229409)

[Listă de figuri 2](#_Toc78229410)

[1. Introducere 3](#_Toc78229411)

[1.1 Prezentare generală 3](#_Toc78229412)

[1.2 Motivație 4](#_Toc78229413)

[1.3 Structura lucrării 4](#_Toc78229414)

[2. Domeniul lucrării 5](#_Toc78229415)

[2.1 Filtrarea conținutului web 5](#_Toc78229416)

[2.2 Extensii Google Chrome 7](#_Toc78229417)

[3. Tehnologii folosite 9](#_Toc78229418)

[3.1 HTML, CSS 9](#_Toc78229419)

[3.2 JavaScript 11](#_Toc78229420)

[3.3 ReactJS 12](#_Toc78229421)

[3.4 C# și .NET Core 12](#_Toc78229422)

[4. Fundamente teoretice 15](#_Toc78229423)

[4.1 Soluții existente 15](#_Toc78229424)

[4.2 Rezultate teoretice 15](#_Toc78229425)

[5. Soluția propusă 17](#_Toc78229426)

[5.1 Descrierea aplicației 17](#_Toc78229427)

[5.2 Detalii de implementare 18](#_Toc78229428)

[5.3 Evaluarea rezultatelor 18](#_Toc78229429)

[6. Concluzii 20](#_Toc78229430)

[Bibliografie 21](#_Toc78229431)

# Listă de figuri

[**Figura 2.1.** Structura unei extensii Google Chrome [7] 7](#_Toc78229452)

[**Figura 3.1.** Structura unei pagini HTML 9](file:///D:\FMI\SDI\practica_disertatie\my-extension\Disertatie.docx#_Toc78229453)

[**Figura 3.2.** DOM-ul real și cel virtual 12](#_Toc78229454)

[**Figura 5.1.** Fereastra pop-up a extensiei implementate 17](#_Toc78229455)

# Introducere

## Prezentare generală

Filtrarea conținutului web este reprezentată de tehnicile de blocare a afișării diferitelor elemente din paginile web, în funcție de diferite criterii.

În primul rând, necesitatea unei filtrări la nivelul conținutului paginilor web a apărut odată cu facilitarea accesului la internet, deoarece persoane de toate vârstele au învățat să îl folosească în viața de zi cu zi. Din această cauză, a apărut nevoia de implementare a unor metode ce să poată oferi un control parental, ajutând astfel părinții să își protejeze copiii împotriva conținutului nepotrivit ce poate fi foarte ușor accesibil pe internet.

O altă utilitate a filtrării este folosirea acesteia în instituții și companii, pentru a limita site-urile la care un utilizator să aibă acces în cadrul rețelei interne de internet. Acest lucru ajută, de exemplu, în rețelele folosite în școli pentru a împiedica elevii să acceseze site-uri cu conținut nepotrivit sau diferite platforme care ar putea afecta calculatoarele instituției prin instalarea unor viruși. De asemenea, companiile pot folosi un sistem de filtrare pentru a împiedica angajații să petreacă prea mult timp navigând pe diferite pagini non-productive, astfel fiind posibilă o creștere a eficienței și a productivității în timpul programului de lucru.

De asemenea, o altă utilitate importantă este posibilitatea de a filtra conținutul pe baza intereselor utilizatorilor. Datorită creșterii foarte mari din ultimii ani a internetului, au apărut foarte multe informații ușor accesibile, însă uneori când căutăm anumite lucruri de care suntem interesați primim și foarte multe reclame care de cele mai multe ori acoperă conținutul paginilor. Uneori ceea ce primim ca și reclamă ar putea fi ceva de interes pentru noi, mai ales că în ultima perioadă și sistemele de recomandare și afișare a reclamelor s-au dezvoltat, fiind capabile să prezinte cât mai mult lucruri din aria de interes a utilizatorilor, însă de cele mai multe ori când navigăm pe internet am dori să ne putem concentra atenția pe lucrurile pe care le căutăm în momentul respectiv. Din acest motiv s-au dezvoltat diferite aplicații software de tip ad-blocker pentru a ascunde cât mai eficient reclamele nedorite ce apar în timpul navigării pe internet.

## Motivație

Deși ad-blocker-ele au adus un mare avantaj în navigarea pe internet, acestea nu reușesc întotdeauna să ascundă toate reclamele care apar. În plus, filtrarea conținutului pe baza intereselor utilizatorilor poate fi extinsă să însemne mai mult decât ascunderea unor reclame. Există situații în care ne putem dori să ascundem dintr-o anumită pagină și altfel de elemente, care nu au cum să fie incluse în cele ce sunt filtrate de un ad-blocker.

De exemplu, daca navigăm pe un site de știri și suntem interesați de cele dintr-un anumit domeniu, ar fi mai ușor dacă am avea posibilitatea ca în momentul în care găsim o știre din afara domeniului nostru de interes să avem opțiunea de a o ascunde din pagină. Astfel, în final putem vizualiza mult mai ușor ceea ce ne interesează de fapt, fără să ne fie distrasă atenția cu alte lucruri. Un alt exemplu din această categorie ar putea fi diferite animații care se mișcă în continuu undeva pe pagina pe care o vizualizăm și am dori să putem să le ascundem.

În acest sens, propunem o extensie Google Chrome care să ne ofere posibilitatea de a selecta și a ascunde elementele pe care nu dorim să le vedem afișate în paginile pe care navigăm. Odată instalată în browser, extensia va adăuga o opțiune numită ”Remove Element” în meniul afișat în momentul în care dăm click dreapta pe un element din pagină, iar aceasta va ascunde elementul selectat. Mai târziu, în interfața extensiei va fi afișată o listă cu elementele care au fost ascunse și va exista posibilitatea ca acestea să fie scoase din listă și astfel să fie afișate din nou în pagină la locul lor. De asemenea, ne propunem să detectăm anumite keywords din textul elementelor pe care le ascunde un utilizator și după câteva utilizări să putem oferi sugestii în legătură cu posibile elemente ce nu par a fi de interes pentru acesta și ar putea fi ascunse din pagina pe care navighează.

## Structura lucrării

În capitolul doi sunt descrise aspectele de interes pentru domeniul lucrării, și anume ce reprezintă filtrarea conținutului web, cu accent pe filtrarea la nivel de client și ce sunt extensiile pentru browser-ul Google Chrome.

În capitolul trei sunt prezentate în mod teoretic tehnologiile care au fost folosite în dezvoltarea aplicației

Capitolul patru detaliază partea de cercetare a domeniului. Aici am descris câteva dintre soluțiile existente ce au fost analizate și rezultatele teoretice obținute în urma analizării lor.

În capitolul cinci este prezentată soluția propusă, o descriere generală a extensiei și a felului în care funcționează, apoi sunt oferite detalii despre implementare, explicând cum au fost folosite tehnologiile prezentate în capitolul anterior în procesul de dezvoltare. Ulterior se prezintă evaluarea rezultatelor obținute.

Capitolul șase prezintă concluziile și învățăturile câștigate în urma acestui studiu, alături de câteva idei de posibile direcții viitoare de dezvoltare a aplicației.

# Domeniul lucrării

## Filtrarea conținutului web

Conținutul web este reprezentat de ceea ce vedem atunci când navigăm pe internet. Acesta poate cuprinde text, imagini, animații, video sau sunete. În principal, conținutul web se împarte în două categorii: textual și multimedia [1]. Odată cu dezvoltarea internetului, a apărut necesitatea de a putea preveni utilizatorii să vadă conținut nepotrivit, ofensator, de a crește securitatea și productivitatea în cadrul unei rețele.

Filtrarea conținutui web este o tehnică folosită în cadrul unui program software pentru a ascunde elemente din paginile web sau pentru a bloca anumite URL-uri în întregime, în funcție de diferite criterii. Există mai multe metode de a filtra conținutul web, precum Platform for Internet Content Selection (PICS), blocarea URL-urilor, filtrarea pe baza cuvintelor cheie sau analiza inteligentă a conținutului [2].

Platform for Internet Content Selection (PICS) este un set de specificații pentru un sistem de evaluare a conținutului cu ajutorul unui rating oferit site-urilor web. Strategia de rating presupune atașarea unei etichete pentru un site, conținând o descriere a acestuia [3]. Pe baza acestor etichete se stabilește ulterior care pagini pot fi accesate și care trebuie blocate.

Pentru blocarea pe baza URL-ului, există două metode: fie se vor folosi liste predefinite cu elemente sau URL-uri unde utilizatorului îi este permis să navigheze (whitelist), fie liste cu elemente care nu sunt permise (blacklist). De obicei este mai folosită metoda blacklist-urilor, pentru că în general sunt mai puține elemente ce ar trebui să fie blocate decât cele ce ar trebui să rămână accesibile. Metoda whitelist-urilor este folosită în principal în instituții unde nu se dorește oferirea accesului decât la un număr limitat de site-uri web.

Filtrarea pe baza cuvintelor cheie este o metodă destul de intuitivă și simplă, blocând accesul la anumite elemente sau site-uri web dacă în conținutul acestora apar cuvintele definite în setul de cuvinte cheie de mai mult de un număr stabilit de ori. Acest tip de filtrare poate uneori să restricționeze accesul excesiv, din cauza faptului că anumite cuvinte cheie pot avea însemnătate diferită în funcție de contextul în care sunt utilizate, iar metoda nu poate să țină cont de acest fapt.

O soluție pentru rezolvarea problemei blocării excesive este folosirea de metode de analiză inteligente a conținutului. Acestea folosesc de multe ori rețele neuronale pentru a învăța din context și pentru a se adapta pe parcurs, astfel învățând să deosebească elementele ce conțin text nepotrivit care ar trebui blocat de cele ce ar putea conține text asemănător însă cu un înțeles inofensiv din contextul în care este folosit. Pentru a putea obține rezultate bune, astfel de rețele trebuie să fie antrenate cu un număr suficient de mare de exemple, atât pozitive cât și negative.

Din punct de vedere al tehnicii folosite, filtrele web se împart de asemenea în mai multe categorii, precum filtrele la nivel de client, filtre pentru browser, rețea sau motoare de căutare.

Filtrele web bazate pe browser sunt aplicații independente sau extensii pentru un anumit browser. Acestea sunt preferate de utilizatori datorită faptului că sunt ușor de folosit, de cele mai multe ori sunt extensii integrate în browser și nu mai este nevoie de alte aplicații adiționale, însă principalul dezavantaj este faptul ca pot fi configurate pentru un browser specific, iar dacă utilizatorul va începe să folosească altul își va pierde configurațiile extensiilor folosite în browser-ul precedent [4].

Filtrele pentru motoarele de căutare blochează rezultatele cu conținut nepotrivit atunci când utilizatorii caută informații prin intermediul acestora. De obicei, motoarele de căutare au activate astfel de filtre în mod implicit, însă orice utilizator poate să le dezactiveze, fără să aibă nevoie de drepturi administrative. Un dezavantaj este faptul că dacă utilizatorul cunoaște URL-ul direct al anumitor site-uri ce sunt de obicei blocate prin filtru, tot îl va putea accesa [5].

Filtrele client-side sunt programe instalate și configurate individual direct pe computerele utilizatorilor. Sunt de obicei întâlnite în școli și companii mici, însă din cauză că trebuie configurate pe fiecare dispozitiv în parte nu sunt folosite și în companiile mari deoarece nu sunt eficiente din punct de vedere al timpului [5]. De asemenea, fiind configurate individual, nu poate exista un sistem central de administrare a filtrelor, iar în cazul în care ar fi nevoie de modificări acestea ar trebui făcute peste tot pe rând.

Filtrele bazate pe rețea sunt implementate la nivelul de transport sau la cel de aplicație a unei rețele de internet. Acestea acționează ca un intermediar între datele pe care utilizatorii le cer și informațiile pe care le primesc ca și răspuns [4]. În plus, un astfel de filtru poate crește securitatea, întrucât în afara rețelei va fi disponibilă doar adresa IP a aplicației filtru, ascunzând celelalte adrese ale dispozitivelor interne.

Un alt tip de filtru existent în anumite țări ce doresc să cenzureze conținutul web este content-limited ISP. Acesta este oferit direct de furnizorii serviciilor de internet și blochează implicit anumite site-uri pentru toți utilizatorii lor.

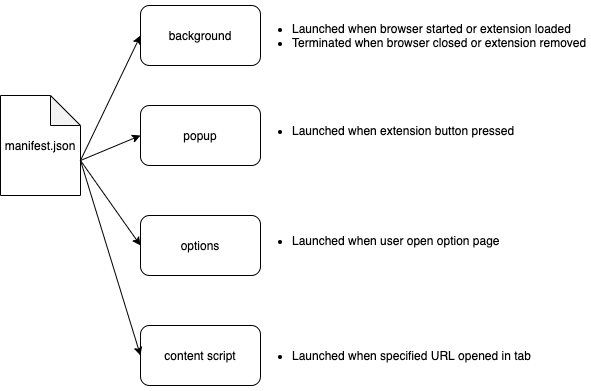
## Extensii Google Chrome

Extensiile pentru un browser web sunt programe care rulează în contextul browser-ului. Cu ajutorul acestora se pot adăuga funcționalități noi, extinzând caracteristicile existente ale browser-ului și permițând astfel utilizatorilor să facă mai multe lucruri în același timp [6]. Browser-ul Google Chrome a început să suporte extensii odată cu versiunea 4, din 2010.

Pentru a crea o extensie Google Chrome, este nevoie de tehnologiile HTML, CSS și JavaScript. HTML și CSS se vor folosi pentru a crea interfața grafică a extensiei, iar JavaScript pentru logica aplicației. De asemenea, se folosește formatul JSON pentru a crea un fișier ce va oferi informații despre extensie browser-ului Google Chrome.

Extensiile sunt plasate în browser-ul Google Chrome, ceea ce permite executarea izolată a codului sursă, în așa fel încât dacă există mai multe extensii instalate în browser acestea nu vor avea conflicte între ele, ci vor putea rula independent în același timp. [6]

O extensie poate fi scrisă de asemenea folosint React, deoarece acesta este o librărie JavaScript. În figura 2.1 este reprezentată structura generală a unei extensii.



**Figura 2.1.** Structura unei extensii Google Chrome [7]

Fișierul *manifest* descrie ceea ce conține pachetul sursă. Se va specifica pentru browser unde se găsesc toate celelalte fișiere necesare: *background, content script, popup* și *options*. De asemenea, tot în acest fișier se specifică și permisiunile de care are nevoie extensia. Codul din *background* rulează mereu, cât timp extensia este instalată, acesta terminându-se doar când browser-ul este închis sau extensia este eliminată. Cu ajutorul acestui cod se pot accesa API-urile Chrome, însă nu se poate accesa DOM-ul. *Popup*-ul reprezintă interfața grafică a extensiei și apare în momentul în care utilizatorul apasă iconița extensiei din bara de acțiuni a browser-ului. *Content script* este codul JavaScript care rulează în tab-urile cu anumite URL-uri, specificate în *manifest.* Cu ajutorul acestui cod se poate accesa DOM-ul paginii. *Options* este o parte opțională a unei extensii. Aceasta este utilizată pentru a configura interfața grafică atunci când sunt lucruri mai complexe de afișat.

# Tehnologii folosite

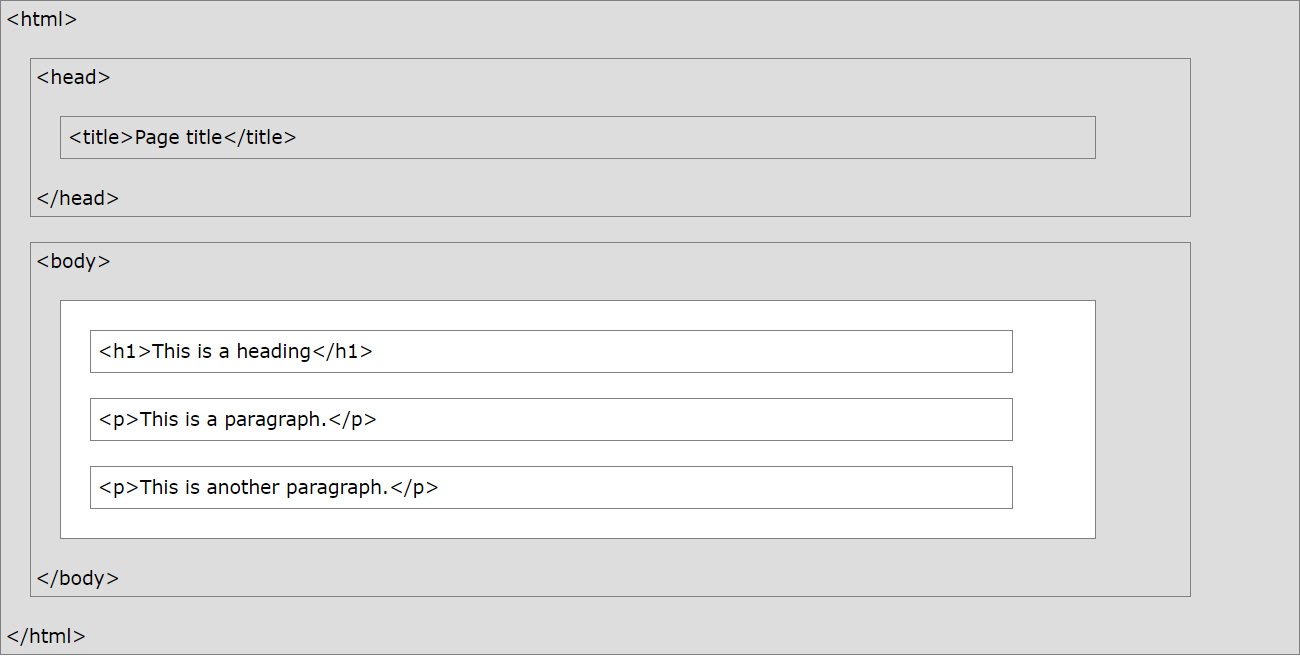
## HTML, CSS

HTML (*HyperText Markup Language*) este un limbaj standard pentru a crea pagini sau aplicații web și reprezintă scheletul oricărei pagini, deoarece descrie formatul primar în care documentele sunt vizualizate și distribuite pe Internet. HTML este un limbaj descriptiv care prezintă, cu ajutorul unor etichete (tag-uri) elementele structurale ale paginii web, precum titluri, paragrafe, liste, tabele sau legatări cu alte pagini, dar și aspectul vizual al paginii, cu ajutorul etichetelor ce indică browser-ului felul în care elementele trebuie să fie așezate în pagină [8].

Majoritatea limbajelor de programare folosite pentru front-end au la bază HTML. Întrucât aplicația dezvoltată este o extensie pentru browser-ul Google Chrome, este inevitabliă folosirea HTML pentru scrierea interfeței grafice a acesteia.

Deoarece este un limbaj descriptiv, nu unul de programare, acesta nu folosește variabile, expresii, tipuri de date, ci doar etichete (marcaje) ce pot fi de două tipuri:

* etichete container – sunt de forma <tag> bloc de text </tag>. Cele doua etichete de început și de final specifică formatul în care textul încadrat între ele va fi afișat.
* etichete vide – au forma <tag> și nu se referă la formatul textelor, ci doar introduc anumite elemente, cum ar fi început de paragraf, sfârșit de linie, linie orizontală și altele. Astfel dau indicații browser-ului despre element și cum trebuie acesta să fie așezat în pagină. [8]

Structura unui document HTML, ilustrată în Figura 3.5, începe cu eticheta <html> și se termină cu </html>, conține un antet cuprins între două etichete de forma <head> și </head>, în interiorul cărora pot să apară diferite elemente precum titlul paginii sau referințe către pagini externe, iar apoi urmează corpul propriu-zis al paginii web, cuprins între etichetele <body> și </body>. Se pot adăuga comentarii în codul HTML, cuprinzându-le între etichetele <!-- și -->. Majoritatea etichetelor dispun de atribute ce pot fi specificate prin sintaxa nume\_atribut=”valoare”, aceste atribute fiind utile pentru stilizarea elementelor.

**Figura 3.1.** Structura unei pagini HTML

Atributele disponibile pentru elemente în HTML, oferă mai mult posibilitatea de a descrie aspecte legate de culoare sau mărime, însă este mai dificil să se stilizeze tot aspectul paginii și modul în care elementele vor fi aliniate. Pentru a da paginilor web o formă mai concretă și expresivă, intervine partea de stilizare gestionată mai bine de CSS.

CSS (*Cascading Style Sheets*) este un limbaj de design ce ajută la simplificarea procesului de prezentare a paginii web. Cu ajutorul acestuia, se pot controla aspecte precum culorile textelor, stilul fonturilor, spațierea, dimen-siunile și așezările în pagină sau imaginile. Cel mai frecvent, CSS este combinat cu limbajele de marcare HTML sau XHTML [9].

Stilurile pot fi aplicate unui document în mai multe moduri. Există stilizarea la nivel de element, atunci când sunt folosite atribute pentru etichete, iar în acest mod va fi afectat doar elementul respectiv. Această modalitate nu este însă prea des folosită, deoarece în loc să facă documentul să fie mai simplu și mai lizibil, ar produce exact contrariul, încârcând codul cu foarte multe informații. O metodă puțin mai eficientă ar fi utilizarea stilurilor încapsulate. Acestea vor fi incluse în documentul asupra căruia se dorește aplicarea, în secțiunea <head>, folosind marcajul <style>. Ultima și cea mai bună metodă este utilizarea fișierelor separate de stil cu extensia .css. În aceste fișiere poate fi definit orice este nevoie legat de stilizare, iar în documentele HTML avem posibilitatea de a face legătura cu acestea cu ajutorul etichetei <link>. Avantajul folosirii fișierelor externe este că același stil poate fi aplicat mai multor documente, fără să mai fie nevoie să rescriem cod. Pe de altă parte, în cadrul aceluiași document, pot exista stiluri diferite pentru aceleași elemente.

Formatul general al unei declarații de stil creată cu CSS este:

selector {

proprietate1: valoare;

proprietate2: valoare;

...

}

Selectorii sunt structuri folosite pentru a identifica în cadrul documentului HTML care sunt acele elemente pentru care trebuie aplicată stilizarea declarată. Cei mai folosiți sunt:

* .class - selectează toate elementele ce au clasa egala cu ”class”
* #id – selectează toate elementele ce au id-ul egal cu ”id”
* \* - toate elementele
* element – selectează toate elementele de tipul ”element”

## JavaScript

JavaScript este un limbaj de programare care este folosit pentru a adăuga interactivitate și comportamente specifice în site-urile web [10]. Acesta este un limbaj la nivel de client, ceea ce înseamnă că va rula la nivelul computer-ului utilizatorului, ci nu pe un server, ceea ce îl face să fie potrivit pentru utilizarea în implementarea extensiilor pentru browser. Din cauza acestui fapt, JavaScript este dependent de browser-ul folosit și de setările acestuia, întrucât utilizatorii pot alege să dezactiveze anumite funcționalități sau este posibil chiar ca anumite lucruri să nu fie suportate de toate browser-ele.

JavaScript este un limbaj dinamic, deci nu este nevoie să fie rulat prin intermediul unui compilator care să interpreteze codul. Browser-ul este capabil să îl citească și să îl interpreteze pe parcursul navigării [10]. O altă caracteristică a acestui limbaj este faptul că este *loosely typed*, adică nu este nevoie să specificăm tipul variabilelor deoarece acesta va fi dedus din valoarea pe care o vom asigna.

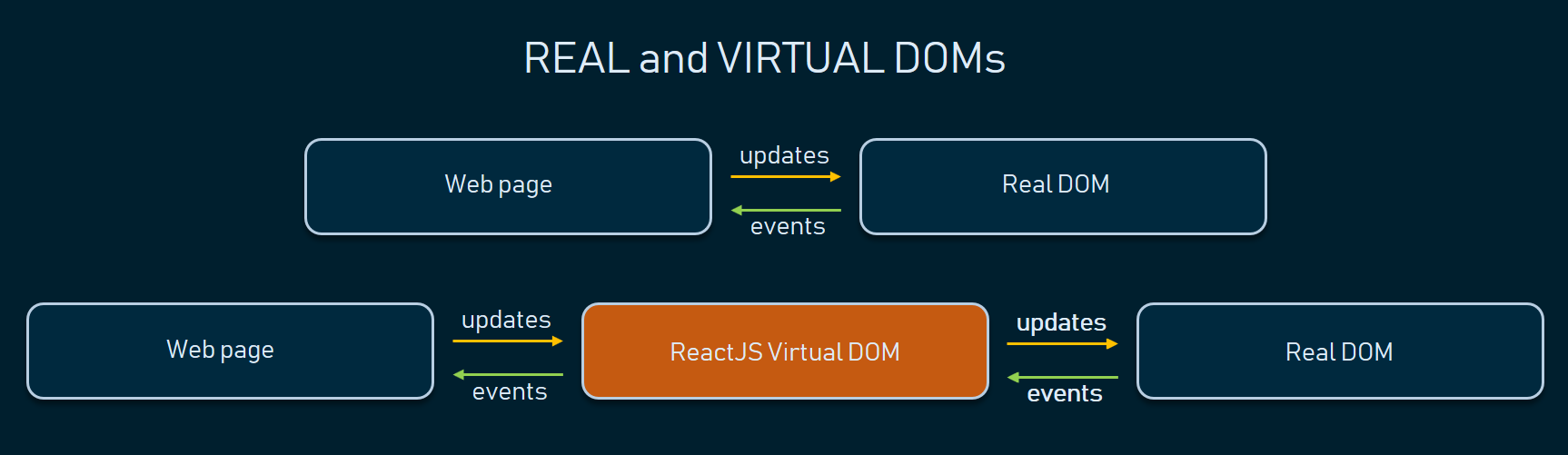
În principal, JavaScript este folosit ca să adăugăm logica de interactivitate în paginile web. După cum am văzut, folosim HTML ca să definim structura unei pagini și CSS pentru stilizarea elementelor, iar JavaScript intervine mai departe pentru a adăuga funcționalitate. De exemplu, cu ajutorul său putem defini ce să se întâmple în momentul în care utilizatorul apasă un anumit buton sau completează anumite input-uri. Includerea scripturilor JavaScript în paginile web se face în același mod în care se poate introduce și codul CSS. Astfel, putem folosi cod încapsulat scris direct în fișier sau putem folosi fișiere separate ce vor avea extensia *.js*. Pentru ambele metode, în fișierul HTML va fi adăugat un element <script>, iar în funcție de caz vom scrie codul directîn interiorul sau, sau vom specifica atriburul *src* ce va reprezenta calea către fișierul extern ce conține codul sursă. Așadar, sintaxa de adăugare a codului JavaScript într-un fișier HTML poate arăta în aceste moduri:

|  |  |
| --- | --- |
| <script>  ...cod JavaScript  </script> | <script src=”script.js”>  </script> |

## ReactJS

ReactJS este o bibliotecă JavaScript declarativă, bazată pe componente, eficientă și flexibilă. A fost dezvoltată de Facebook și lansată în 2013. Aceasta este foarte folosită pentru front-end, fiind capabilă să fie folosită cu orice tip de tehnologii back-end, dacă acestea expun un API.

O caracteristică specifică React-ului este conceptul de *Virtual DOM (Document Object Model)*, fiind o reprezentare în memorie a DOM-ului paginii sub forma unui arbore de noduri. De fiecare dată când se produce o schimbare la nivelul nodurilor din DOM browser-ul trebuie să refacă layout-ul paginii și să o reafișeze. Folosind React, atunci când ceva se modifică în pagină Virtual DOM-ul se va schimba, iar browser-ul va putea să găsească difetențele dintre cel vechi și cel nou, astfel modificând DOM-ul real doar acolo unde există schimbări, fără să refacă toată pagina.



**Figura 3.2.** DOM-ul real și cel virtual

## C# și .NET Core

Platforma .NET de la Microsoft, împreună cu limbajul C# asociat acesteia, a fost introdusă formal în 2002 și a devenit rapid una din bazele dezvoltării software moderne [11].

C# este un limbaj de programare a cărui sintaxă este similară cu cea din Java, însă se poate considera că această construcție sintactică este modelată după mai multe aspecte similare și cu cele din Visual Basic sau C++. Acesta suportă noțiunea de proprietăți de clasă (ceea ce înlocuiește metodele clasice de set și get din alte limbaje), permite folosirea parametrilor opționali, suprascrierea și supraîncărcarea metodelor sau operatorilor, precum și crearea de structuri, enumerații și funcții recursive. Mai mult decât atât, cu ajutorul *LINQ (Language Integrated Query)*, putem folosi în C# numeroase construcții asemănătoare cu sintaxa SQL, care nu sunt disponibile și în alte limbaje. Putem spune că acest limbaj este un hibrid obținut din mai multe limbaje de programare, iar asta face ca rezultatul final să fie un produs cu o structură sintactică clară precum Java, simplu precum Visual Basic și dispunând de putere și flexibilitate asemănătoare cu C++ [11].

Câteva dintre caracteristicile și beneficiile C# sunt faptul că nu mai este nevoie să folosim pointeri ca în C++, gestionarea memoriei se face în mod automat, resursele neutilizate fiind astfel colectate automat de către garbage collection, avem o construcție sintactică formală pentru clase, interfețe, structuri și enumerații și sunt suportate adnotări, metode anonime, expresii lambda, membri de tip dinamic (dynamic) și tipuri generice.

Framework-ul .NET este o platformă pentru construirea sistemelor software în principal pe sistemul de operare Windows, dar totodată și pe alte sisteme non-Microsoft precum macOS, iOS, Android și Unix/Linux [11]. Framework-ul furnizează câteva caracteristici de bază precum interoperabilitatea cu cod existent, suport pentru mai multe limbaje de programare (C#, VB, F#, C++/CLI, JavaScript), un motor de rulare comun pentru toate limbajele ce pot folosi .NET, o bibliotecă de bază pentru clase ce este foarte cuprinzătoare, cu foarte multe tipuri predefinite și un model de implementare simplificat.

În iunie 2016 Microsoft a anunțat lansarea platformei .NET Core, o platformă revoluționară pentru dezvoltatorii de .NET, bazată pe C# și Framework-ul .NET [11]. Așa cum ne este prezentat de către Troelsen și Japikse, platforma .NET Core și-a propus inițial ca scop principal să ofere posibilitatea de rulare a programelor pe alte sisteme de operare în afară de Windows, însă, pe lângă asta, au mai fost adăugate și îndeplinite câteva:

* Suport multi-platformă - .NET Core poate rula pe Windows, Linux și macOS, iar aplicațiile pot fi construite peste platforme folosind Visual Studio. Includerea Xamarinului a adăugat Android și iOS la platformele suportate.
* Performanță - .NET Core are o performanță ce se află în mod constant aproape de partea superioară a tuturor graficelor de performanță relevante, iar fiecare nou release aduce îmbunătățiti.
* Biblioteci de clasă portabile, utilizabile pe toate rulările .NET - a fost introdus .NET Standard, o specificație formală pentru a stabili o uniformitate a comportamentului de rulare pentru .NET.
* Implementare portabilă sau de sine stătătoare – apicațiile pot fi implementate alături de framework sau folosind instalarea la nivel de mașină a .NET Core.
* Suport complet pentru linia de comandă
* Open source - .NET Core împreună cu documentația sa sunt complet open source.
* Interoperabilitatea cu întregul Framework .NET - .NET Core permite corelarea cu bibliotecile din .NET Framework.

# Fundamente teoretice

## Soluții existente

În lucrarea ”FilterPlus: A real-time content filtering extension for Google Chrome” [12] autorii prezintă extensia FilterPlus. Aceasta oferă o interfață simplă prin care utilizatorii pot filtra imagini, ferestrele de tip popup, codul JavaScript, Cookie-urile și notificările pe paginile web pe care navighează. Permițând blocarea imaginilor, a codului JavaScript și a popup-urilor este asigurat un aspect mai curat al paginilor și în același timp este crescută securitatea, deoarece numeroase programe de tip malware sunt instalate prin intermediul codului rulat fără știrea utilizatorilor. Pentru cookies sunt disponibile trei opțiuni: Allow, Session Only și Block, astfel oferind control utilizatorului asupra a ceea ce dorește sau nu să fie memorat, iar pentru notificări este posibil ca acestea să fie acceptate, blocate, sau acceptate doar după ce extensia va întreba înainte pentru fiecare în parte.

Thomas Chen și Victoria Wang prezintă în ”Web Filtering and Censoring” [13] aplicația Green Dam, care este o aplicație desktop ce filtrează conținutul web. Folosește un blacklist definit de utilizator pentru a filtra mai întâi URL-urile unde accesul nu este permis, apoi folosește analiza de text pentru a căuta și filtra posibile obscenități sau conținut sensibil politic. De asemenea, implementează și o metodă de filtrare a imaginilor pornografice.

” Dynamic Web Content Filtering based on User’s Knowledge” [14] descrie un sistem de filtrare pe baza conținutului, ca o soluție pentru problema filtrării după URL, deoarece aceasta nu este eficientă pentru site-urile web necunoscute încă și o listă completă de URL-uri ce nu ar trebui permise este imposibil de obținut. Pentru implementarea soluției se folosește metoda de achiziționare a cunoștințelor cu reguli multiple de clasificare - Multiple Classification Ripple-Down Rules (MCRDR), care permite unui expert în domeniu să mențină baza de cunoștințe fără să fie nevoie de ajutorul inginerilor. Sistemul este antrenat pe un anumit set pagini web pentru a învăța care sunt cele ce trebuie să fie blocate. Ca și exemplu, sistemul prezentat învață să clasifice un set de 283 site-uri web în pagini care au sau nu legătură cu jocurile de noroc.

## Rezultate teoretice

Prima soluție prezentată, FilterPlus, reușește cu succes să crească securitatea utilizatorilor oferind posibilitatea de blocare a codului JavaScript și a ferestrelor de tip popup și, în același timp, oferă și o filtrare a conținutului permițând ascunderea imaginilor sau a notificărilor. Preferințele utilizatorilor se reflectă însă doar în alegerea de a permite sau nu elementele pe care extensia le poate bloca, însă nu oferă posibilitatea de a alege altceva, de a filtra conținutul mai mult decât atât, pe baza altor reguli.

Aplicația Green blochează în mod eficient toate URL-urile aflate pe lista neagră, iar sistemul de filtrare pe baza analizei de text este destul de precis. Pe de altă parte, sistemul de blocare a imaginilor pornografice nu este la fel de eficient, deoarece acesta a clasificat ca și conținut nepotrivit și imagini greșite, precum imagini cu o pisică sau o farfurie cu friptură. De asemenea, filtrarea pe baza altor preferințe ale utilizatorilor nu este posibilă nici în cadrul aplicației Green.

Ultima soluție descrisă este un sistem destul de eficient de filtrare pe baza preferințelor utilizatorului, întrucât acesta poate alege ce fel de pagini dorește să fie blocate în funcție de conținutul lor. Un dezavantaj ar fi faptul că este necesar un set de câteva pagini pentru învățare, pentru ca sistemul să fie capabil mai apoi să clasifice site-urile nou accesate.

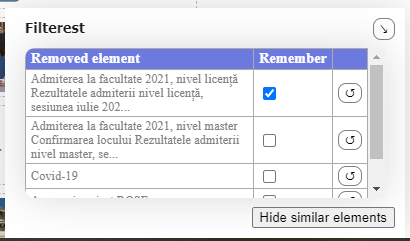
Toate cele trei soluții prezentate au atât avantaje cât și dezavantaje și se potrivesc pentru anumiți utilizatori, în funcție de nevoile lor. În timp ce primele două nu permit o personalizare a conținutului ce se dorește a fi blocat, ultima oferă această posibilitate. Pe de altă parte, chiar dacă a treia soluție implementează filtrarea pe baza preferințelor utilizatorului, aceasta blochează întreaga pagină web, nefiind posibilă ascunderea doar pentru elementele ce nu sunt de interes.

# Soluția propusă

## Descrierea aplicației

Soluția propusă este o extensie pentru browser-ul Google Chrome, care va permite utilizatorilor să aleagă elementele din pagini web de care nu sunt interesați și să le ascundă. Spre deosebire de celelalte soluții descrise, extensia implementată nu are anumite filtre predefinite pentru ce fel de elemente să fie blocate, precum imaginile sau textul având conținut nepotrivit, ci oferă posibilitatea utilizatorului să aleagă orice element din pagină de care nu este interesat și să îl ascundă. De asemenea, extensia nu va bloca întreaga pagină web, ci doar va ascunde elementele selectate de utilizator care nu sunt de interes.

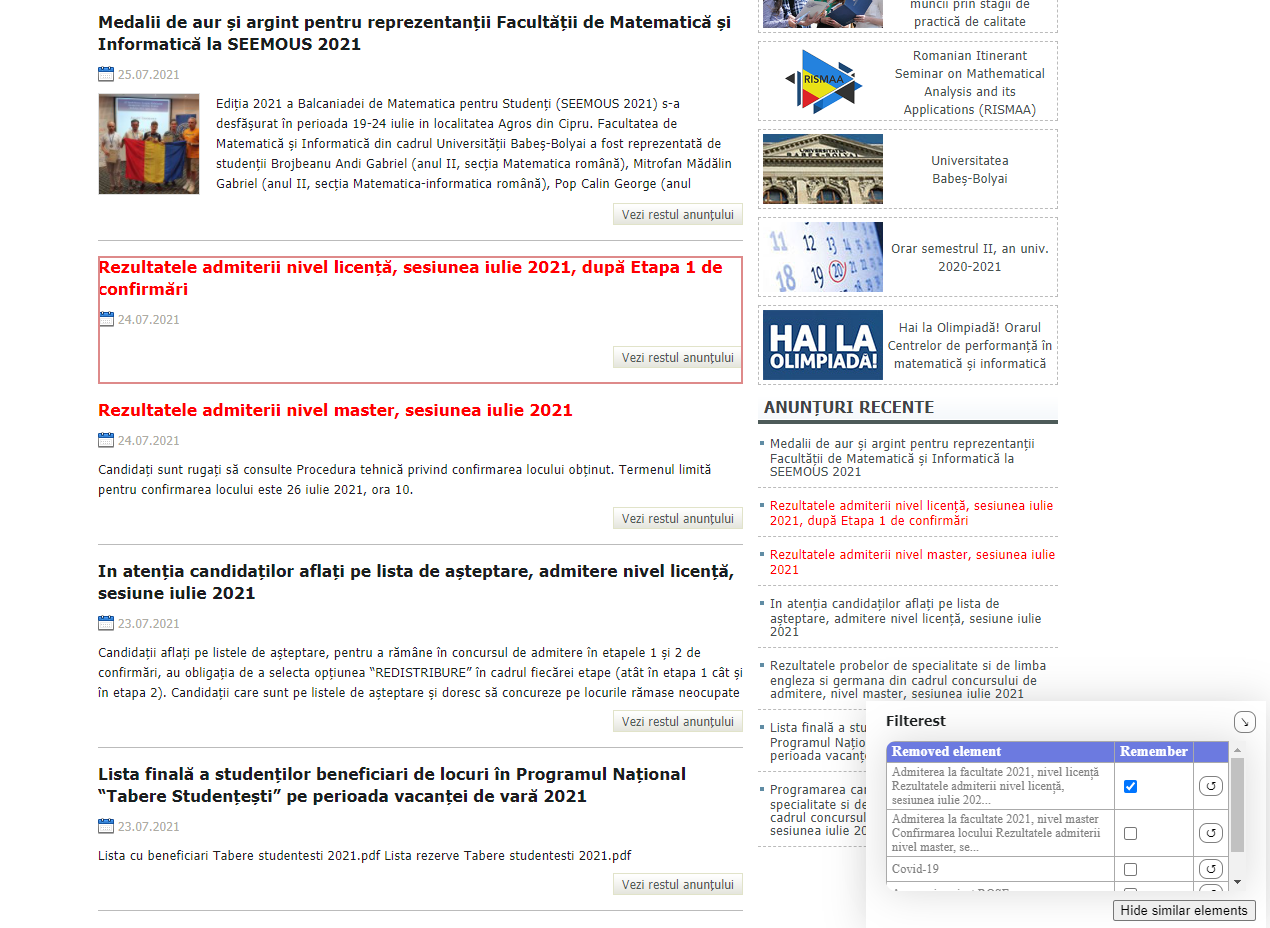
Extensia adaugă o mică fereastră în colțul din dreapta jos al paginilor web în care vor fi afișate elementele ascunse într-un tabel. De asemenea, atunci când extensia este activă, în momentul în care mutăm cursorul peste elementele din pagini acestea vor avea o margine setată, astfel fiind mai usor vizibil elementul. Avand astfel elementul evidențiat, atunci când utilizatorul dă click pe acesta, el va fi ascuns din pagină iar informațiile sale vor apărea în tabelul din popup-ul extensiei. Pentru fiecare element va fi disponibilă de asemenea opțiunea de a-l șterge din listă, astfel acesta va apărea din nou în pagină în locul său inițial. Elementele ce au fost ascunse pot fi memorate bifând această opțiune, astfel încât la o nouă accesare a aceleași pagini utilizatorul să nu fie nevoit să le ascundă din nou.



**Figura 5.1.** Fereastra pop-up a extensiei implementate

Mai mult decât atât, în momentul ascunderii mai multor elemente ce conțin text într-o pagină web, aplicația va oferi posibilitatea utilizatorului să aleagă opțiunea de a lăsa aplicația să găsească niște similarități, alcătuind o listă de cuvinte întâlnite mai des în conținutul de care utilizatorul nu este interesat. Pe baza acestor keywords extensia va putea sugera alte elemente ce ar putea fi eliminate, iar utilizatorul poate alege dacă vrea să ascundă elementele sugerate.

În figurile 5.2 și 5.3 este ilustrat cum sunt evidențiate elementele în cadrul paginilor web atunci când mișcăm cursorul deasupra lor pentru a alege ce vrem să ascundem sau atunci când extensia ne sugerează un element asemănător pentru a fi ascuns.



**Figura 5.2.** Evidențierea elementelor la mouseover pentru a ști ce vom ascunde

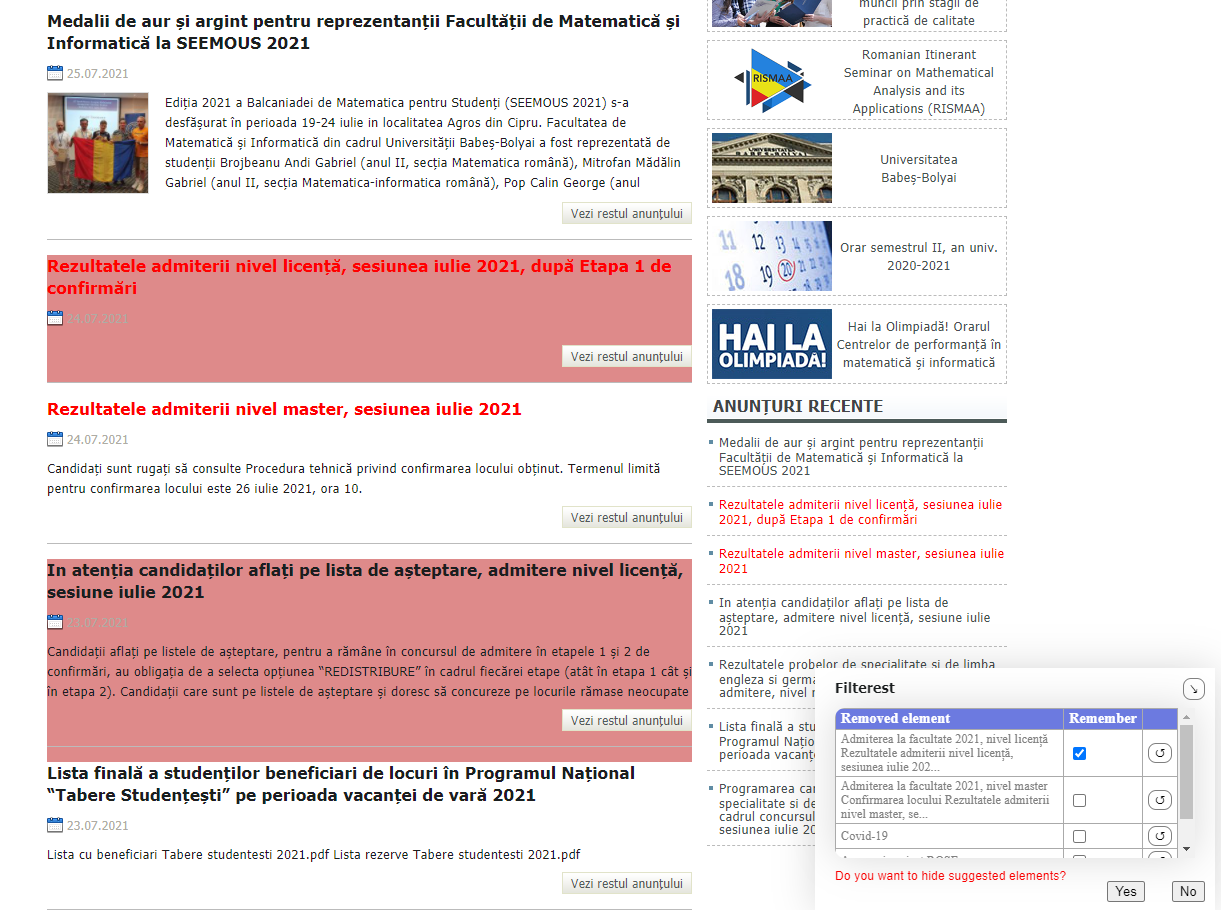


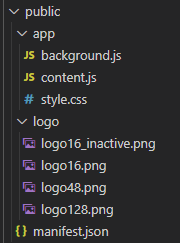
Figura 5.3. Evidențierea elementelor sugerate și afișarea opțiunii de a le ascunde sau nu

## Detalii de implementare

Interfața aplicației, și anume fereastra de tip popup în care vor fi afișate elementele ascunse de către utilizator a fost implementată cu ajutorul framework-ului React. Fiind vorba despre filtrarea la nivel de client, funcționalitățile aplicației sunt implementate exclusiv în JavaScript în cadrul extensiei. Pentru memorarea elementelor a fost folosit state-ul global al aplicației.

Pentru o mai bună performanță, pentru găsirea de similarități în text, alcătuirea listei de cuvinte cheie și sugerarea posibilelor elemente pe care utilizatorul ar putea dori să le ascundă a fost implementat un Web API în .Net, care va primi datele necesare din cadrul extensiei, va analiza textele și va trimite înapoi rezultate, fără să stocheze datele.

### Extensie

Aplicația implementată, numită *Filterest*, este o extensie pentru browser-ul Google Chrome, așadar structura sa este cea specifică acestui tip de aplicație, ilustrată în figura alăturată. 

**Figura 5.4.** Structura de fișiere pentru extensie

Fișierul principal care trebuie să fie întâlnit în orice extensie este *manifest.json*, deoarece acesta conține toate informațiile care definesc extensia, precum numele, versiunea și aspectele legate de funcționalitate (script-uri care vor rula în background, script-uri pentru interfața extensiei, elemente de stil, permisiuni și alte acțiuni ale browser-ului). Acest fișier este în format JSON și poate conține mai multe chei. Valorile posibile pentru aceste chei sunt:

* author
* background
* browser\_action
* browser\_specific\_settings
* chrome\_settings\_overrides
* chrome\_url\_overrides
* commands
* content\_scripts
* content\_security\_policy
* default\_locale
* description
* developer
* devtools\_page
* dictionaries
* externally\_connectable
* homepage\_url
* icons
* incognito
* manifest\_version
* name
* offline\_enabled
* omnibox
* optional\_permissions
* options\_page
* options\_ui
* page\_action
* permissions
* protocol\_handlers
* short\_name
* sidebar\_action
* storage
* theme
* theme\_experiment
* user\_scripts
* version
* version\_name
* web\_accessible\_resources

Dintre acestea, singurele care trebuie să fie obligatoriu prezente sunt *manifest\_version, version* și *name*. În cadrul fișierului aplicației implementate, au fost folosite mai multe dintre aceastea, precum:

* *background –* conține informații în legătură cu fișierele în care se află script-ul ce va rula în background. Fișierele trebuie enumerate într-o listă, în cazul nostru existând înregistrarea "scripts": ["app/background.js"]
* *content\_scripts –* cu ajutorul acestei înregistrări browser-ul va ști ce script-uri trebuie să încarce în paginile al căror URL se potrivește cu cele specificate. În cadrul acestui obiect pot exista mai multe chei pentru specificarea detaliilor. Cele mai importante sunt:
  + *”matches” –* o listă de pattern-uri cu ajutorul cărora se stabilește pentru care URL-uri ar trebui ca extensia să ruleze. Această cheie este obligatorie în interiorul *content\_scripts*
  + *”js” –* specifică o listă de fișiere JavaScript care vor fi incluse ca scripturi în cadrul paginilor,
  + *”css” –* o listă ce va conține fișierele CSS care se dorește să fie injectate în pagini
  + *”run\_at” –* determină momentul în care script-urile specificate în ”js” ar trebui să fie injectate în pagină. Cele trei valori posibile pentru aceasta setare sunt *document\_start* (atunci când DOM-ul încă se încarcă), *document\_end* (când DOM-ul s-a încărcat, însă e posibil ca alte resurse precum imagini sau script-uri încă se încarcă) și *document\_idle* (DOM-ul și toate resursele necesare s-au încărcat).

În cazul nostru, această înregistrare arată în felul următor:

"content\_scripts": [

{

"matches": ["http://\*/\*", "https://\*/\*"],

"js": ["app/content.js"],

"css": ["app/style.css"],

"run\_at": "document\_end",

"all\_frames": false

}

],

* *permissions –* această cheie este folosită pentru a defini permisiunile pe care le are extensia. Permisiunile sunt enumerate într-o listă și pot fi de trei tipuri: host permissions (pattern-uri care vor identifica URL-urile pentru care extensia are nevoie de permisiuni speciale), API permissions (acestea sunt definite prin niște cuvinte cheie folosite să identifice un WebExtensionAPI pe care extensia are nevoie să îl folosească) și permisiunea pentru *activeTab.*

### Web API

web api-ul

## Evaluarea rezultatelor

Pentru testarea extensiei s-a folosit site-ul Facultății de Matematică și Informatica, Universitatea Babeș-Bolyai, http://www.cs.ubbcluj.ro/.

Funcționalitatea de ascundere a elementelor selectate, întrucât este una destul de simplă, funcționează bine de fiecare dată. Așadar, ne vom axa pe evaluarea funcționalității de sugerare a elementelor similare cu cele ascunse ulterior de utilizator.

Pentru testare s-a încercat ascunderea anunțurilor cu informații despre admiterea la studiile de nivel licență și s-a verificat dacă elementele sugerate anterior fac parte din aceeași categorie. Primul anunț care a fost ascuns conținea informații despre organizarea concursului de admitere în luna aprilie 2021. În urma ascunderii acestui articol, a început formarea listei de cuvinte cheie, însă fiind doar primul element ascuns au rezultat 43 de cuvinte cu posibilitatea de a fi cuvinte cheie. Așadar, aplicația nu va genera sugestii după ce un singur element a fost ascuns. Al doilea articol ales conținea informații despre calendarul concursului de admitere, iar în urma extragerii de cuvinte cheie din acesta s-au obținut 41 de variante. Făcând însă intersecția între cele două liste de cuvinte, a rămas un set de 8 keywords care să fie utilizat pentru căutarea altor elemente ce ar putea fi ascunse.

Pe baza acestei liste de cuvinte, aplicația a verificat alte anunțuri pentru a sugera unele care e posibil să nu fie de interes. Un prim articol care a fost sugerat a fost cel cu rezultatele concursului de admitere la nivel licență, o sugestie conform așteptărilor. Întrucât printre cuvintele cheie se aflau ”calendar” și ”admitere”, a fost de asemenea sugerat și un anunț despre calendarul de admire la nivel de master. Chiar dacă acest anunț nu este legat de studiile la nivel de licență, nu putem spune ca sugestia a fost întocmai greșită, deoarece articolul conține o parte dintre cuvintele cheie.

Un dezavantaj al site-ului ales este faptul ca fiecare anunț este afișat cu titlu și un prim paragraf, urmând apoi un buton cu textul ”Vezi restul anunțului” pentru a accesa întregul articol. Din cauză că atunci când au fost ascunse articolele pentru testare acest buton a fost găsit în ambele, extensia a sugerat de asemenea ascunderea celorlalte butoane de acest tip.

Așadar, funcția de sugerare a unor elemente asemănătoare cu altele ascunse anterior de utilizatori pe baza unor cuvinte cheie este destul de precisă, însă din cauza faptului că se folosesc doar aceste cuvinte cheie rezultatul este unul foarte general. În acest fel, sugestiile nu vor putea fi cuprinse într-un anumit domeniu care nu este de interes, ci vor reprezenta orice alte elemente care conțin în textul lor o parte dintre cuvinte.

Extensia ar putea fi îmbunătățită schimbând modul în care sunt extrase cuvintele cheie, alegând grupuri de cuvinte în schimbul celor individuale, pentru a crește precizia. De asemenea, ar fi bine să fie luată în considerare și frecvența de apariție a cuvintelor în texte, iar daca un keyword nu apare de un anumit număr minim de ori să nu fie luat în considerare mai departe.

# Concluzii

# Bibliografie

|  |  |
| --- | --- |
| [1] | A. Baishya și S. Kakoty, „A Review on Web Content Filtering, Its Technique and Prospects,” *International Journal of Computer Science Trends and Technology,* 2019. |
| [2] | S. C. H. A. C. M. Fong, „Web Content Filtering Methods and Tools,” în *Multimedia Engineering - A practical guid for internet implementation*, 2006. |
| [3] | E. Bertino, E. Ferrari și A. Perego, „Web Content Filtering,” în *Web and Information Security*, 2006. |
| [4] | „Comodo Dome,” [Interactiv]. Available: https://cdome.comodo.com/what-is-web-filtering.php. |
| [5] | [Interactiv]. Available: https://www.calyptix.com/top-threats/types-of-web-filters-and-how-they-can-work-for-clients/. |
| [6] | P. Mehta, Creating Google Chrome Extensions, 2016. |
| [7] | A. Kumar, „Creating a Chrome Extension with React - https://javascript.plainenglish.io/creating-a-chrome-extension-with-react-d92db20550cb”. |
| [8] | Microsoft, Programarea web cu Microsoft .NET. |
| [9] | „TutorialsPoint - CSS,” [Interactiv]. Available: https://www.tutorialspoint.com/css/css\_tutorial.pdf. |
| [10] | J. N. Robbins, Learning Web Design, O'Reilly, 2018. |
| [11] | A. Troelsen și P. Japikse, Pro C# 7 With .NET and .NET Core, Apress, 2017. |
| [12] | B. Babu și M. Kumar, „FilterPlus: A real-time content filtering extension for Google Chrome,” 2014. |
| [13] | T. M. Chen și V. Wang, „Web Filtering and Censoring,” *IEEE Computer,* 2010. |
| [14] | N. Churcharoenkrung, Y. S. Kim și B. H. Kang, „Dynamic web content filtering based on user's knowledge,” *Proceedings of the International Conference on Information Technology: Coding and Computing.* |