VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

VÝVOJ ALGORITMU ZACHOVÁVAJÍCÍHO SOUKROMÍ UŽIVATELE PRO ADAPTIVNÍ ORGANIZACI ZÁLOŽEK V PROHLÍŽEČI NA ZÁKLADĚ UŽIVATELSKÝCH VZORCŮ CHOVÁNÍ

TOWARDS PRIVACY BY DESIGN ALGORITHM FOR ADAPTIVE BROWSER TAB ORGANISATION BASED ON BROWSING PATTERNS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DOMINIK SAJKO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. LIBOR POLČÁK, Ph.D.**
SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

163495

Ústav:	Ústav informačních systémů (UIFS)
Student:	**Sajko Dominik**
Program:	Informační technologie
Název:	**Vývoj algoritmu zachovávajícího soukromí uživatele pro adaptivní organizaci záložek v prohlížeči na základě uživatelských vzorců chování**
Kategorie:	Uživatelská rozhraní
Akademický rok:	2024/25

Zadání:

1. Seznamte se s implementací prohlížeče Chromium, respektive Avast Secure Browser. Zaměřte se na správu záložek a komponenty interagující s uživatelem prohlížeče.
2. Nastudujte metody analýzy uživatelských vzorců chování aplikovatelných na používání webového prohlížeče a jeho záložek. Seznamte se s principem privacy by design (PbD). Analyzujte nastudované metody a zjistěte, zda jsou s principem PbD kompatibilní.
3. S využitím principu PbD navrhněte algoritmus, který na základě analýzy uživatelských vzorců chování (např. frekvence návštěv stránek, čas strávený na stránce) automaticky uspořádá záložky webového prohlížeče tak, aby reflektovaly měnící se zájmy a potřeby uživatele.
4. Implementujte algoritmus v prohlížeči Chromium, respektive Avast Secure Browser.
5. Implementaci otestujte.
6. Vyhodnoťte dosažené výsledky a navrhněte možná pokračování projektu.

Literatura:

- Benz, D.; Tso, K. H. L. & Schmidt-Thieme, L. (2006), Automatic Bookmark Classification - A Collaborative Approach, *in* 'Proceedings of the 2nd Workshop in Innovations in Web Infrastructure (IWI2).
- Chris Staff and Ian Bugeja. 2007. Automatic classification of web pages into bookmark categories. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '07). Association for Computing Machinery, New York, NY, USA, 731–732.
- The European Data Protection Board, Guidelines 4/2019 on Article 25 Data Protection by Design and by Default, Version 2.0 (2020). Dostupné online https://www.edpb.europa.eu/sites/default/files/files/file1/edpb_guidelines_201904_dataprotection_by_design_and_by_default_v2.0_en.pdf

Při obhajobě semestrální části projektu je požadováno:
Vypracování technické zprávy popisující splnění prvních 3 bodů zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Polčák Libor, Ing., Ph.D.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1.11.2024
Termín pro odevzdání:	14.5.2025
Datum schválení:	21.1.2025

Abstrakt

Táto práca sa zameriava na návrh a implementáciu algoritmu, ktorý dokáže adaptívne upravovať štruktúru záložiek vo webových prehliadačoch založených na jadre prehliadača Chrome podľa meniacich sa potrieb užívateľa. Algoritmus je navrhnutý v súlade s princípmi Privacy by Design za účelom zvýšenia bezpečnosti používaných osobných údajov užívateľa. Implementácia algoritmu je realizovaná prostredníctvom rozšírenia pre Chrome, ktoré zároveň využíva lokálny jazykový model pre spracovanie potrebných dát.

Abstract

This thesis focuses on the design and implementation of an algorithm that can adaptively adjust the structure of bookmarks in web browsers based on the Chrome browser core according to the changing needs of the user. The algorithm is designed in accordance with the principles of Privacy by Design in order to increase the security of the user's personal data. The implementation of the algorithm is carried out as a Chrome extension, which also uses a local language model for processing the necessary data.

Kľúčové slová

Rozšírenia pre Chrome, Privacy by Design, záložky, ochrana osobných údajov, súkromie, GDPR

Keywords

Chrome extensions, Privacy by Design, bookmarks, personal data protection, privacy, GDPR

Citácia

SAJKO, Dominik. *Vývoj algoritmu zachovávajícího soukromí uživatele pro adaptivní organizaci záložek v prohlížeči na základě uživatelských vzorců chování.* Brno, 2025. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Libor Polčák, Ph.D.

Vývoj algoritmu zachovávajícího soukromí uživatele pro adaptivní organizaci záložek v prohlížeči na základě uživatelských vzorců chování

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Libora Polčáka, Ph. D. Ďalšie informácie mi boli poskytnuté spoločnosťou Gen Digital v spolupráci, s ktorou bola táto práca realizovaná. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Dominik Sajko
13. mája 2025

Poďakovanie

Rád by som poďakoval pánovi Ing. Liborovi Polčákovi, Ph. D za konštruktívnu kritiku, cenné rady a ochotu pri konzultáciach. Taktiež by som chcel poďakovať pánovi Ondřejovi Malačkovi, ktorý mi počas implementácie práce poskytol cenné rady. Počas procesu písania boli na redakčné úpravy a jazykovú korektúru využité nástroje ChatGPT a Gemini.

Obsah

Kapitola 1

Úvod

Webové prehliadače sú neoddeliteľnou súčasťou každodenného života, pretože nám umožňujú rýchlo vyhľadávať informácie, komunikovať a prístupovať k rôznym online službám. Záložky, ktoré slúžia na ukladanie obľúbených stránok, sú nástrojom, ktorý užívateľom pomáha zorganizovať a zjednodušiť prístup k často navštevovaným webovým stránkam. S rastúcim počtom záložiek však môže byť ich organizácia neprehľadná, čo vedie k strate efektivity a samotného významu existencie záložiek. Aj, keď väčšina moderných webových prehliadačov obsahuje funkciu záložiek, ich organizácia často zostáva zanedbaná. Tradičné metódy organizácie záložiek pozostávajú z ich manuálneho premiestňovania, pridávania a odoberania, čo môže byť časovo náročné a nepraktické. Tieto metódy často vyžadujú, aby užívateľ investoval značné úsilie do správnej kategorizácie a udržiavania prehľadnosti, pričom nezohľadňujú dynamické zmeny v užívateľských preferenciách a návykoch.

Cieľom tejto práce je vylepšiť funkciu záložiek vo webovom prehliadači prostredníctvom algoritmu, ktorý na základe užívateľských vzorcov chovania automaticky upráví štruktúru záložiek. Tento algoritmus automatizuje manuálne procesy spravovania záložiek s využitím nástrojov umelej inteligencie, ktoré dokážu čiastočne nahradiť úlohu človeka, pričom bude zároveň dbať na zachovanie súkromia osobných dát užívateľa.

V kapitole 2 sa popisuje rozšírenie pre Chrome, pomocou ktorého je implementované navrhnuté riešenie. Táto kapitola je venovaná primárne popisu jednotlivých častí rozšírenia pre Chrome a dostupných Chrome API, ktoré sú dôležité pre implementáciu algoritmu riešenia. V kapitole 3 sú rozoberané princípy Privacy by Design, v súlade s ktorými je riešenie práce implementované. Taktiež sa v tejto kapitole popisujú aj návrhové vzory ochrany súkromia, ktoré pomáhajú riešiť problémy spojené s ochranou užívateľských dát. Kapitola 4 sa venuje návrhu riešenia tejto práce a jednotlivých funkcií, ktoré sú jeho súčasťou. V kapitole 5 je detailne popísaná štruktúra implementovaného riešenia spolu s výberom konkrétneho jazykového modelu, ktorý je v práci využitý. Kapitola 6 popisuje spôsoby, akými bola overená funkčnosť výsledného riešenia. Súčasťou tejto kapitoly je aj vyhodnotenie, ako riešenie spĺňa ciele zadania práce. Záverečná kapitola 7 je zhrnutím tejto práce ako celku, spolu s ďalšími návrhmi na jej možné pokračovanie.

Kapitola 2

Rozšírenia v prehliadači Chrome

Rozšírenia pre Chrome sú softvérové balíčky, ktoré je možné nainštalovať do webových prehliadačov založených na jadre prehliadača Chrome. Medzi ich hlavné funckie patrí prispôsobenie užívateľského prostredia webového prehliadača, monitorovanie udalostí v prehliadači alebo úprava zdrojového kódu navštívených webových stránok. Pre ich vytváranie sa využívajú rovnaké technológie ako pre vývoj webových aplikácií, konkrétne HTML, CSS a JavaScript. Rozšírenia môžu byť následne po ich vytvorení zverejnené a distribuované prostredníctvom Webového obchodu Chrome [18].

Na začiatku kapitoly popisujem jednotlivé časti, ktoré tvoria rozšírenia pre Chrome. Následne sa zameriavam na analýzu dostupných API, ktoré sú k dispozícii pre implementáciu týchto rozšírení. Rozoberám ich funkcionalitu z pohľadu vývojára rozšírení a uvádzam spôsoby ich využitia pri vývoji. V závere kapitoly sa venujem bezpečnostným zásadám, ktoré je potrebné dodržiavať pri tvorbe rozšírení, aby sa zabezpečila ich ochrana a správna implementácia.

2.1 Časti rozšírení pre Chrome

Rozšírenia pre Chrome sa skladajú z viacerých častí, pričom každá plní špecifickú funkciu a spoločne tvoria funkčný celok, avšak pri implementácii konkrétneho rozšírenia nemusia byť všetky tieto časti nutne zahrnuté. Výber jednotlivých častí závisí predovšetkým od požiadaviek a funkcionality daného rozšírenia. V nasledujúcich sekciách kapitoly popisujem jednotlivé časti rozšírení pre Chrome.

Manifest

Manifest je povinným súborom každého rozšírenia pre Chrome. Je písaný vo formáte JSON a nachádza sa v koreňovom adresári rozšírenia. Manifest obsahuje dôležité informácie o štruktúre a správaní rozšírení, ako sú názov, potrebné oprávnenia a požadovaná verzia Chrome [20].

Service worker

Service worker je špeciálna časť kódu v jazyku JavaScript zabezpečujúca obsluhu udalostí vyvolávaných webovým prehliadačom alebo rozšírením pre Chrome. Pre využívanie funkcií service workera v rozšírení je potrebné v súbore manifest zadefinovať novú položku typu

kľúč-hodnota, kde kľúčom je `"service_worker"` a hodnotou cesta k súboru, v ktorom sa nachádza kód service workera [25, 24].

Servie worker počas svojho životného cyklu typicky prechádza medzi aktívnym a neaktívnym stavom. Prechody medzi jednotlivými stavmi sú vykonávané systémom, na základe aktivity prichádzajúcich udalostí. Ich účelom je optimalizácia spotreby zdrojov systému. V prípade deaktivácie service workeru sa všetky dáta uložené v globálnych premenných stratia. Na tento fakt je potrebné brať ohľad, pričom dáta vyžadujúce trvalé uloženie je vhodné uložiť do perzistetnej pamäte napríklad pomocou storage API, ku ktorému sa vyjadrujem v sekcií 2.2.2 [26].

V riešení práce je service worker možné využiť na získavanie, spracovanie a perzistentné uloženie informácií, zasielaných pomocou content scriptov. Jedná sa o špecifické dáta, ktoré nie je možné získať iným spôsobom než použitím content scriptov.

Content script

Ak nie je uvedené inak, všetky informácie v tejto podkapitole boli prevzaté z oficiálnej dokumentácie Chrome [17]. Content scripty sú podobne, ako service workery špeciálne časti kódu v jazyku JavaScript, avšak narozdiel od service workerov slúžia pre manipuláciu s navštívenými webovými stránkami. Hlavnou výhodou content scriptov je ich schopnosť čítať a priamo upravovať zdrojový kód navštívenej webovej stránky pomocou rozhrania Document Object Model, ďalej len DOM.

Pre využívanie content scriptov je ich potrebné v rámci rozšírenia zaregistrovať spolu s cieľovou webovou stránkou, na ktorú majú byť nahrané. Registrácia content scriptu môže byť realizované ľubovoľným z troch nasledujúcich spôsobov:

1. Statickou deklaráciou, teda vytvorením JSON objektu v súbore manifest, ktorý bude obsahovať URL cieľovej webovej stránke, cestu k súboru obsahujúcemu content script a v prípade potreby cestu k CSS súboru, ktorý má byť spolu s content scriptom na cieľovú webovú stránku nahraný. V prípade navštevy cieľovej webovej stránky bude na content script automaticky nahraný hneď po jej načítaní.

2. Dynamickou deklarácou, kde sa využíva metóda `registerContentScripts()` rozhrania Scripting API. Využitie dynamickej deklarácie je vhodné v prípadoch, kde nie je možné vopred určiť konkrétne webové stránky, na ktoré má byť content script nahraný alebo nie je možné vopred vybrať správny content script, ktorý má byť nahraný do cieľovej webovej stránky. Zavolaním metódy `registerContentScripts()` sa zvolený content script zaradí do zoznamu dynamicky registrovaných content scriptov a jeho správanie bude následne rovnaké, ako v prípade statickej deklarácie [23].

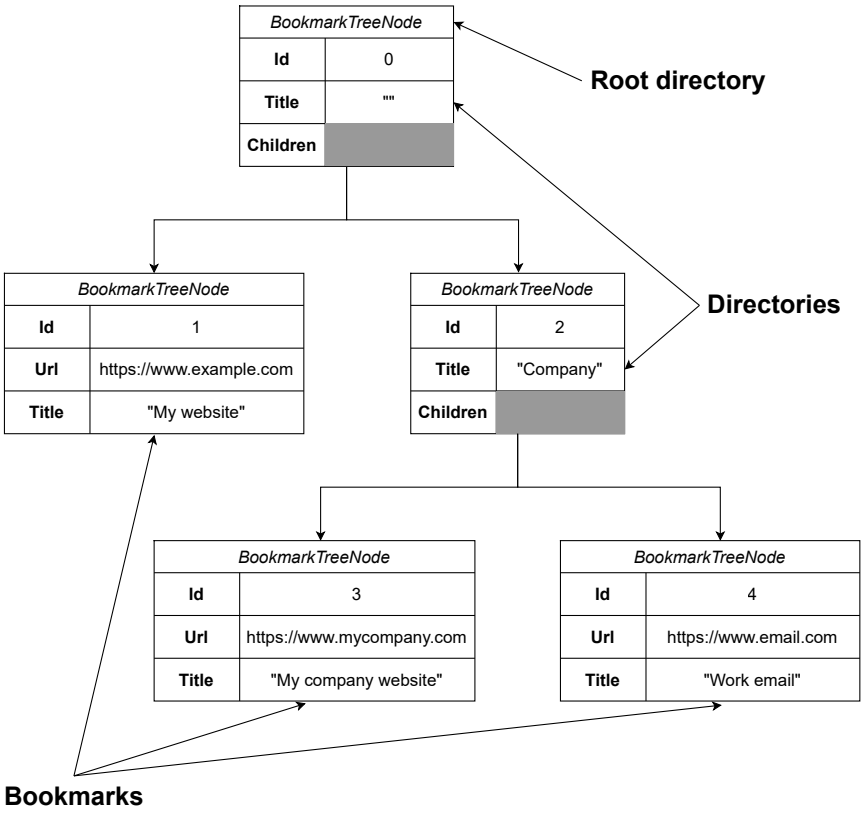
3. Programovým vložením, ktoré rovnako, ako Dynamické vloženie využíva Scripting API, avšak namiesto metódy `registerContentScripts()` využíva metódu `executeScript()`. Vybraný content script bude vložený do aktuálne zobrazenej webovej stránky v okamih zavolania metódy `executeScript()`. Programové vloženie je výhodné využiť v situáciách, kde chceme, aby bol content script nahraný do webovej stránky až po zachytení konkrétnej udalosti alebo vykonaní určitej časti kódu.

2.2 Chrome API

Webový prehliadač Chrome ponúka vývojárom rozšírení množstvo API, pomocou ktorých je možné spravovať rôzne prvky prehliadača, ako napríklad záložky, históriu, úložisko alebo užívateľské rozhranie. V nasledujúcich podkapitolách sa zaoberám popisom konkrétnych Chrome API relevantných pre túto prácu.

2.2.1 Bookmark API

Záložky v prehliadači Chrome sú uložené záznamy obsahujúce informácie o webových stránkach, ktoré umožňujú jednoduchý prístup na webovú stránku pomocou jedného kliknutia bez toho, aby užívateľ musel zadávať URL webovej stránky do adresového riadku. Prehliadač Chrome usporiadáva záložky do stromovej štruktúry, kde uzly reprezentujú záložky alebo pomenované adresáre. Každý uzol reprezentujúci záložku má svoj názov a URL, na ktorú odkazuje. Uzly reprezentujúce pomenované adresáre obsahujú názov adresára a množinu odkazov na ďalsie uzly [16]. Názorný príklad stromovej štruktúry záložiek je možné vidieť na Obrázku 2.1.



Obr. 2.1: Popis štruktúry záložiek v prehliadači Chrome

2.2.2 Storage API

Ak nie je uvedené inak, všetky informácie v tejto podkapitole boli prevzaté z oficiálnej dokumentácie Storage API [28]. Mnohé rozšírenia pre Chrome vyžadujú perzistentné ukladanie dát za účelom zachovania svojho stavu, preferencií užívateľa a podobne. Jednou z možností pre perzistentné ukladanie dát rozšírenia je využitie rozhrania Web storage API, avšak táto možnosť nie je v praxi odporúčaná [39]. Hlavnými dôvodmi sú obmedzená dostupnosť rozhrania pre service workery, absencia bezpečného prístupu k uloženým dátam z content scriptov a automatické odstránenie uložených dát pri vymazaní histórie prehliadača. Výhodnejšou alternatívou je použitie Storage API, ktoré poskytuje hneď niekoľko možností ukladania dát.

- storage.local – Trvalé lokálne úložisko dát s kapacitou obmedzenou na 10 MB. Veľkosť úložiska môže byť v prípade potreby rozšírená vyžiadaním oprávnenia `"unlimitedStorage"` v súbore manifest. Typ úložiska je vzhľadom na jeho charakter vhodný pre ukladanie veľkého množstva dát.
- storage.managed – Úložisko určené výhradne pre čítanie. Slúži pre uloženie konfiguračných súborov administrátorom rozšírenia.
- storage.session – Dáta v tomto type úložiska sú udržiavané len počas aktívnej relácie prehliadača a po jej ukončení sa automaticky vymažú. Maximálna kapacita úložiska je obmedzená na 10 MB. Session.store je odporúčané využívať pre ukladanie dát, ktoré sú spracovávané pomocou service workerov.
- storage.sync – V prípade zapnutej synchronizácie sú dáta uložené v storage.sync automaticky prenášané medzi všetkými zariadeniami, na ktorých je užívateľ do webového prehliadača Chrome prihlásený. Prenášanie dát medzi zariadeniami je zabezpečené pomocou Chrome Sync's Model API, ku ktorému nie je možné z rozšírenia priamo pristupovať [5]. Z toho dôvodu je potrebné, aby rozšírenie na každom synchronizovanom zariadení zistilo a spracovalo zmenu dát samo. Prenos synchronizovaných dát prebieha cez internet, preto je dôležité zabezpečiť ochranu citlivých údajov prostredníctvom šifrovania ukladaných dát.

2.2.3 Runtime API

Hlavným účelom Runtime API je zabezpečenie interakcie medzi jednotlivými časťami rozšírenia a správa reakcií na udalosti životného cyklu rozšírenia. Runtime API dokáže zachytávať udalosti spustenia, reštartovania a aktualizácie rozšírenia, ktoré môžu rozšíreniu poskytnúť informácie o aktuálnom stave a určiť nasledovný vývoj udalostí [22].

Komunikácia medzi časťami rozšírenia

Jednotlivé časti rozšírenia pre Chrome pracujú v rôznych kontextoch v závislosti od ich typu. Preto priame volanie metód alebo prístup k premenným medzi rôznymi časťami nie je možný. Pre komunikáciu medzi jednotlivými časťami je potrebné vytvoriť komunikačný kanál, cez ktorý sa nadviaže spojenie a bude možné zasielať správy. Runtime API poskytuje pre tento účel metódy `sendMessage()`, `sendResponse()` a `addListener()`. Názornú ukážku použitia týchto metód je možné vidieť v kóde 2.2.3, ktorý zobrazuje príklad spôsobu komunikácie medzi content scriptom a service workerom [22, 21].

```
// Content script
(async () => {
  const response = await chrome.runtime.sendMessage({greeting: "hello"});
  // do something with response here, not outside the function
  console.log(response);
})();
```

```
// Service worker
chrome.runtime.onMessage.addListener(
  function(request, sender, sendResponse) {
    console.log(sender.tab ?
                "from a content script:" + sender.tab.url :
                "from the extension");
    if (request.greeting === "hello")
      sendResponse({farewell: "goodbye"});
  }
);
```

Kód 2.1: Príklad kódu zabezpečujúceho komunikáciu medzi content scriptom a service workerom

2.2.4 History API

Webový prehliadač Chrome obsahuje zabudovanú funkciu správy histórie vyhľadávania, ktorá umožňuje užívateľom retrospektívne sledovať navšívené webové stránky spolu s ďalšími časovými údajmi o ich návštevách. History API umožňuje správu a prístup k histórii vyhľadávania priamo z rozšírenia pre Chrome.

Pre využívanie History API je nevyhnutné definovať oprávnenie `"history"` v manifeste rozšírenia, pričom jeho kľúčovou funkcionalitou pre túto prácu je metóda `search()`. Metóda `search()` získa položky histórie na základe zadaných vyhľadávacích parametrov. Každá položka histórie obsahuje údaje ako URL navštívenej webovej stránky, počet navštívení alebo čas od posledného navštívenia [19].

Toolbar action API

Webový prehliadač Chrome zobrazuje nainštalované rozšírenia v panely nástrojov vedľa vyhľadávacieho poľa. Jednotlivé rozšírenia sú zobrazené ako malé ikony. Pomocou Tooblar action API je možné upravovať parametre ikony rozšírenia, ako napríklad zobrazený obrázok, názov rozšírenia alebo okno, ktoré sa má po kliknutí na inkonu rozšírenia v prehliadači zobraziť [15].

2.3 Bezpečnosť rozšírení a dát

Rozšírenia pre Chrome môžu udržiavať rôzne citlivé informácie o užívateľoch a pomocou oprávnení deklarovaných v súboroch manifest pristupovať k dodatočným funkciám webových prehliadačov. Práve tieto vlastnosti robia z rozšírení potenciálne ciele pre útočníkov [27, 29].

V tejto kapitole sa zameriávam na rôzne spôsoby zabezpečenia rozšírení proti ich kompromitácií.

Použitie HTTPS

Všetky sieťové požiadavky, ktoré rozšírenie vykonáva by mali byť uskutočnené výhradne pomocou protokolu HTTPS, ktorý zabezpečuje šifrovaný prenos dát. V prípade využitia nešifrovaného protokolu HTTP je nutné predpokladať, že prenášané správy môžu byť odchytené a upravené útočníkom [27].

Oprávnenia rozšírenia

Prístup rozšírenia pre Chrome k údajom a funkcionalite je obmedzený oprávneniami uvedenými v súbore manifest. S narastajúcim množstvom oprávnení sa zvyšuje aj riziko potenciálneho zachytenia informácií útočníkom. Rozšírenie by malo povinne vyžadovať len oprávnenia potrebné pre svoju základnú funkcionalitu. Zvyšné oprávania je možné zahrnúť do poľa `"optional_permissions"` v súbore manifest, ako je uvedené v kóde 2.2. Voliteľné oprávnenia budú od užívateľa vyžiadané len v prípade využitia funkcií rozšírenia, ktoré tieto oprávnenia potrebujú. [27, 29].

```
1 "optional_permissions": [ "tabs", "bookmark", ]
```

Kód 2.2: Pole definujúce voliteľné oprávnenia rozšírenia

Získavanie a skladovanie užívateľských údajov

Množstvo zbieraných a udržiávaných dát o užívateľovi by malo byť obmedzené na minimum. Dôvodom je minimalizácia uniknutých dát užívateľa v prípade skompromitovania rozšírenia. Správne ukladanie dát je ďalším faktorom, ktorý môže zvýšiť bezpečnosť rozšírenia. Dáta uložené v úložiskách spomínaných v kapitole 2.2.2 nie sú šifrované, a teda tieto úložiská nie sú vhodné pre ukladanie citlivých užívateľských údajov [29].

Zraniteľnosti content scriptov

Hoci content scripty bežia v izolovanom prostredí, do ktorého nemajú prístup iné rozšírenia ani webové stránky, v ktorých sú content scripty vykonávané, nerobí ich to imúnnymi voči útokom [17]. Keďže content scripty pracujú priamo so zdrojovým kódom webovej stránky je možné pozmeniť ich očakávané správanie úpravou DOM elementov webovej stránky. Ďalšiu zraniteľnosť content scriptov predstavujú side channel útoky, ako je napríklad Spectre, ktorý dokáže získať obsah vyrovnávacej pamäte procesoru [37]. V prípade, že content script pracuje s citlivými údajmi, môže dôjsť k ich úniku [27].

Komunikácia medzi jednotlivými časťami rozšírenia pre Chrome prebieha prostredníctvom komunikačného kanála, ktorý som bližšie popísal v kapitole 2.2.3. Správy prichádzajúce z content scriptov môžu obsahovať neočakávané údaje na základe spomínaných zraniteľností. Aj, keď sa jedná o správy posielané v rámci rovnakého rozšírenia je nutné ich obsah po prijatí skontrolovať. V oficiálnej dokumentácii Chrome je spomínané len základné

overovanie pomocou pridania novej položky do zasielanej správy (viď. kód 2.3). Ďalšími spôsobmi kontroly prichádzajúcich dát môžu byť využitie JSON schémy, regulárnych výrazov alebo overenie rozsahu pri číselných dátových typoch [41], [27].

```
1 // Formát príjmanej správy: { allowedAction: true, ...}
2 chrome.runtime.onMessage.addListener(function(request, sender, sendResponse) {
3   if (request.allowedAction)
4   {
5     ...
6   }
7 });
```

Kód 2.3: Základné overenie správy príjmanej pomocou Runtime API

Priama ochrana voči side channel útokom nie je možná, pretože content scripty musia byť vykonávané v rovnakom vykresľovacom procese, ako webové stránky, aby mohli pristupovať k DOM elementom. Operácie, ktoré využívajú rôzne Chrome API je vhodné implementovať pomocou service workerov, a tak nevystavovať oprávnenia rozšírenia v content scripte. Dáta, zasielané content scriptom pomocou iných častí rozšírenia môžu byť získané účoníkom. Preto nie je vhodné zasielať žiadne citlivé dáta [27].

Režim inkognito

Režim inkognito sľubuje užívateľom nezanechanie žiadnych stôp, čo by malo byť rovnako rešpektované rozšíreniami pre Chrome. V prípade, aktivovaného režimu inkognito v prehliadači užívateľa by rozšírenie malo zvážiť ukladanie akýchkoľvek dát spojených s navštívenými webovými stránkami [29].

Kapitola 3

Privacy by Design

Ako sa technológie v súčastnosti aktívne menia a množstvo dát o užívateľoch na internete pribúda, únik citlivých dát predstavuje čoraz vyššie riziko. Podľa globálneho prieskumu sa až 88% ľudí obáva, kto každý má prístup k ich súkromným dátam [46].

Privacy by Design je prístup k vývoju technológií, s cieľom začleniť súkromie užívateľa už do najskorších fázy životného cyklu vývoja za účelom zvýšenia bezpečia súkromných údajov užívateľa. Tento koncept bol vytvorený Hanou Cavoukinovou v deväťdesiatych rokoch dvadsiateho storočia a prijatý Európskym parlamentom [2].

V tejto kapitole sa zameriavam na popis Privacy by Design tak, ako je zahrnutý v Európskej legislatíve. Následne popisujem všeobecné princípy Privacy by Design a v závere kapitoly sa zaoberám implementáciou princípov Privacy by Design v praxi.

3.1 Právne aspekty ochrany užívateľských údajov

V roku 2018 prijal Európsky parlament regulačné opatrenia vo forme General Data Protection Regulation, ďalej len GDPR, ktoré predstavujú jeden z najprísnejších svetových zákonov o ochrane súkromia a bezpečnosti [12]. Tento zákon zaväzuje organizácie, ktoré spracúvajú osobné údaje užívateľov, k prevzatiu zodpovednosti za prípadné porušenie pravidiel. Súčasťou GDPR je princíp Privacy by Design, ktorý je špecifikovaný v článku 25 [10].

Aj, keď Európska legislatíva od organizácií vyžaduje implementáciu princípov Privacy by Design, zároveň im poskytuje flexibilitu pri výbere konkrétnych ochranných opatrení, ktoré majú byť prijaté. Príklady ochrany osobných údajov uvedené v odôvodnení 78 zahŕňajú minimalizáciu spracovávaných osobných dát, pseudonimizáciu mien alebo autentifikáciu užívateľa, avšak žiadne ďalšie podrobnosti uvedené nie sú [11, 10]. GDPR zároveň zohľadňuje, že pri výbere vhodných prostriedkov na ochranu užívateľských dát je nevyhnutné prihliadať aj na náklady spojené s ich implementáciou.

Z pohľadu GDPR, Privacy by Design nie je akýsi súbor pravidiel, ktoré majú byť za každú cenu striktne dodržané, ale skôr prístup k vývoju riešenia s dôrazom na ochranu súkromia užívateľa. Týmto spôsobom nahliadania na vývoj riešení sa zabezpečí vyššia úroveň ochrany súkromia užívateľov, pričom nedochádza k výraznému zvýšeniu nákladov na implementáciu zo strany organizácií.

3.2 Princípy Privacy by Design

V pôvodnom dokumente, ktorý popisuje Privacy by Design je spomenutých 7 základných príncípov [3]. Nejedná sa o princípy aplikovateľné len v oblasti informačných technológií, ale aj v rôznych ďalších oblastiach, ako napríklad fyzický dizajn, organizačné praktiky a ďalšie. Môj popis základných princípov Privacy by Design je z pohľadu programátora, ktorý vyvíja aplikáciu. Informácie v nasledujúcich podkapitolách sú prevzané z dokumentu [3].

Proaktívne, nie reaktívne

Pri vývoji aplikácie sa je potrebné zamerať na opatrenia, ktoré zabránia neoprávnenému zásahu do súkromia užívateľa. Koncept Privacy by Design neposkytuje prostriedky, ktoré riešia únik citlivých informácií užívateľa, ale naopak, snaží sa úniku citlivých informácií predísť. Opatrenia, pre zvýšenie bezpečnosti dát užívateľa majú byť príjmané nad rámec povinných opatrení vyžadovaných legislatívou.

Ochrana súkromia, ako predvolené nastavenie

V prípadoch, kedy potreba aplikácie pracovať s osobnými údajmi užívateľa nie je nevyhnutná, je vhodné uplatniť, čo najstriktnejšie opatrenia pri práci s osobnými údajmi. Získavané osobné údaje o užívateľovi majú byť obmedzené na minimálne množstvo potrebné pre správne fungovanie aplikácie. Ak je to možné, aplikácia má používať pre svoje fungovanie len údaje, ktoré s jej užívateľom nie je možné spojiť. V prípade, že aplikácia osobné údaje užívateľa potrebuje, ich udržiavanie má byť odmäzené na minimálnu možnú dobu a po ukončení práce majú byť údaje odstránené.

Zahrnutie ochrany súkromia dát do dizajnu aplikácie

Ochrana súkromia nemá byť len akýmsi rozšírením hlavnej funkcionality programu, ale kľúčovým komponentom zahrnutým do základu aplikácie. Už od počiatočných fázach vývoja aplikácie je potrebné príjmať opatrenia zabezpečujúce ochranu súkromia užívateľa. Rovnako pri rozširovaní funkcionality aplikácie treba zvážiť riziká ohrozujúce súkromie dát užívateľa. Vždy, keď je to možné, by sa malo vykonať a zverejniť podrobné posúdenie vplyvu a rizika na súkromie, v ktorom sa jasne zdokumentujú riziká pre súkromie a všetky opatrenia prijaté na zmiernenie týchto rizík vrátane zváženia alternatív. Zahrnutím opatrení pre ochranu súkromia dát už v počiatočných fázach vývoja aplikácie sa môžu odhaliť bezpečnostné riziká a predísť úniku citlivých dát.

Úplná funkcionalita

Začlenenie ochrany súkromia do vyvíjanej aplikácie by nemalo ovplyvniť jej plnú funkčnosť. Koncept Privacy by Design sa snaží poukázať na fakt, že ochrana súkromia a funkcionalita aplikácie nie sú protiklady, ktoré sú v konflikte, ale môžu existovať súčasne. Zahrnutie opatrení pre ochranu súkromia užívateľa už v počiatočných fázach vývoja môže predísť konfliktom medzi bezpečnoťou užívateľských dát a funcktionalitou aplikácie. Akékoľvek kompromisy môžu byť často zbytočné a mali by byť nahradené hľadaním riešení, ktoré podporujú multifunkčnosť.

Ochrana dát počas celého životného cyklu

Bezpečnosť osobných údajov užívateľa má byť zabezpečená počas ich celého životného cyklu. To znamená od získania dát, počas ich používania až po vymazanie dát zo systému. Je potrebné uplatňovať bezpečnostné štandardy, ktoré zabezpečia dôvernosť, integritu a dostupnosť údajov počas ich životného cyklu [2]. Princíp ochrany dát počas celého životného cyklu zdôrazňuje dôležitosť nepretržitej ochrany a zodpovednosti, čím sa zabezpečí, že v ochrane údajov nebudú žiadne medzery.

Viditeľnosť a transparentnosť

Viditeľnosť a transparentnosť sú kľúčovými faktormi k nadobudnutiu dôvery užívateľov voči vývojárom. Koncept Privacy by Desing zahŕňa jasné, konzistentné a transparentné dokumentovanie a informovanie o tom, ako vývojári nakladajú so súkromnými údajmi užívateľov. Vývojári by mali pre správne uplatnenie princípu viditeľnosti a transparentnosti dodržiavať nasledujúce zásady.

- **Zodpovednosť** – Zodpovednosť za všetky zásady a postupy súvisiace s ochranou súkromia musia byť zdokumentované a oznámené užívateľovi. Pri prenose osobných údajov tretím stranám bude zabezpečená rovnocenná ochrana súkromia prostredníctvom zmluvných alebo iných prostriedkov.
- **Otvorenosť** – Informácie o tom, ako vývojári nakladajú so súkromnými údajmi, by mali byť pre užívateľa jednoducho prístupné.
- **Dodržiavanie pravidiel** – Užívateľ by mal mať zabezpečenú možnosť podať sťažnosť týkajúcu sa nevhodného nakladania s jeho osobnými údajmi. V prípade podania sťažnosti by mal byť informovaný o prijatých opatreniach a krokoch vedúcich k náprave.

Rešpekt súkromia užívateľa

Ústrednou postavou konceptu Privacy by Design je užívateľ. Umožnenie užívateľovi spravovať prístup aplikácie k jeho osobným dátam je účinnou kontrolou proti neoprávnenému použitiu osobných údajov. Pred zhromažďovaním, používaním alebo zverejňovaním osobných údajov užívateľa musí užívateľ udeliť svoj osobný súhlas, ktorý má mať možnosť neskôr odvolať. Osobné údaje požadované aplikáciou by mali byť jasne a presne špecifikované, aby užívateľ pri udeľovaní súhlasu so spracovaním osobných údajov disponoval jednoznačnými informáciami o ich povahe a účele. Čím sú osobné údaje citlivejšie, tým precíznejšie a jasnejšie by mali byť definované pri požadovaní súhlasu so spracovaním, aby bola zabezpečená informovanosť užívateľa.

3.3 Návrhové vzory ochrany súkromia

Pôvodný dokument popisujúci Privacy by Design neobsahuje konkrétne kroky ani podrobný návod pre jeho implementáciu. Na túto medzeru reagovala výskumná skupina CUPS (CyLab Usable Privacy and Security Laboratory), ktorá vypracovala návrhové vzory zamerané na integráciu ochrany súkromia užívateľov do procesu vývoja systémov. Pojem návrhový vzor je známy z Objektovo orientovaného softvérovného dizajnu, kde prestavuje všeobecný

spôsob riešenia problému, ktorý sa opakovane vyskytuje. Návrhové vzory v kontexte Privacy by Design predstavujú abstraktné riešenia typických problémov spojených s ochranou užívateľských dát [45, 8].

V súčasnosti existuje množstvo návrhových vzorov zameraných na ochranu súkromia. Pre účely tejto práce som vybral a analyzoval tie, ktoré sú najrelevantnejšie vzhľadom na jej zameranie.

Onion routing

Onion routing (cibuľové smerovanie) je pokročilá metóda anonymizácie a zabezpečenia dátovej komunikácie, ktorá funguje na princípe viacvrstvového šifrovania [43].

Metóda onion routing využíva asymetrické šifrovanie, ktoré pracuje s dvojicou kľúčov. Verejný kľúč, ktorý je verejne zdielaný. Tento kľúč v kontexte onion routing slúži pre zašifrovanie obsahu odosielanej správy. Druhým kľúčom je súkromný kľúč. Súkromný kľúč je uchovávaný v tajnosti a slúži pre dešifrovanie správy, ktorá bola zašifrovaná pomocou verejného kľúča. Pred odoslaním je správa zašifrovaná verejnými kľúčmi sieťových uzlov, cez ktoré bude prechádzať. Ak správa postupne prechádza sieťovými uzlami U_1, U_2, ..., U_n je zašifrovaná verejnými kľúčmi uzlov U_n, U_{n-1}, ..., U_1. Pred pridaním novej vrstvy šifrovania je k správe priradená adresa nasledujúceho sieťového uzlu, aby bolo možné správu doručiť až do požadovaného cieľa. Týmto spôsobom je zaručené, že žiaden sieťový uzol nepozná celú komunikačnú cestu správy, pretože disponuje len informáciou o uzle, z ktorého bola správa odoslaná a uzle, ktorý ma byť správa odoslaná [4].

Použitím tohto návrhového vzoru sa dosahuje vysoká miera anonymity, čím sa eliminuje možnosť prepojenia odosielateľa a príjemcu správy.

Voľba využívaných funkcií

Aplikácie často zhromažďujú osobné údaje užívateľov, ktoré nakoniec nie sú využité. Tento stav je dôsledkom rozsiahlej funkcionality aplikácií, ktorú používateľ nevyužíva v jej plnom rozsahu. Preferencie používateľov ohľadom využívaných funkcií aplikácie sa môžu líšiť, rovnako ako osobné údaje, ktoré jednotlivé funkcie vyžadujú. Poskytnutím užívateľovi možnosti vybrať, ktoré funkcie si nepraje používať alebo ktorým nie je ochotný poskytnúť svoje osobné údaje, je možné znížiť množstvo zhromažďovaných osobných informácií, čím sa zároveň znižuje riziko ich úniku [42].

Obmedzenie užívateľských údajov

Inžiniersky proces je často orientovaný na vývoj architektúr zameraných na systém, kde sú dáta zhromažďované a spracovávané v jednom centralizovanom uzle. Užívatelia sú nútení dôverovať organizáciám a zdielať svoje potencionálne citlivé osobné informácie. Riešením tohto problému je presunúť spracovanie osobných údajov z centrálneho uzla na lokálne zariadenie užívateľa. Užívateľ sa tak nemusí naďalej spoliehať na dôveryhodnosť centrálneho uzla pri zabezpečení jeho osobných údajov [44].

Kapitola 4

Návrh riešenia adaptívnej organizácie záložiek

Táto kapitola sa zaoberá návrhom algoritmu, ktorý na základe analýzy užívateľských vzorcov chovania automaticky usporiadáva a spravuje záložky v prehliadači Chrome. Zameriavam sa na popis technológií, ktoré som zvolil pre implementáciu praktickej časti tejto práce. Následne popisujem jednotlivé funkcie, ktoré má výsledné riešenie obsahovať a v závere kapitoly sa venujem konkrétnym krokom algoritmu organizácie záložiek.

V návrhu algoritmu organizácie záložiek zahŕňam princípy ochrany súkromia spomínané v kapitole 3.1 a návrhové vzory ochrany súkromia z kapitoly 3.3 s cieľom zabezpečiť, čo najvyššiu úroveň ochrany užívateľských dát. Tento prístup aplikuje techniky, ako je minimalizácia zhromažďovania súkromných dát užívateľa a spracovanie dát na lokálnom zariadení. Zároveň aplikujem metódy, ktoré umožňujú analyzovať správanie používateľa, čím sa snažím zabezpečiť, aby algoritmus fungoval efektívne a dosahoval čo najpresnejšie výsledky. Cieľom je dosiahnuť rovnováhu medzi ochrannými opatreniami a vysokou presnosťou riešenia, čím sa zabezpečuje optimálna funkcionalita s maximálnym rešpektovaním ochrany súkromia.

4.1 Použité technológie

Technológie, ktoré som zvolil pre vývoj algoritmu adaptívnej organizácie záložiek sú rozšírenia webových prehliadačov, respektíve rozšírenia pre Chrome a jazykové modely. Pri výbere technológií som sa zameriaval na ich schopnosť splniť špecifické požiadavky zadania, konkrétne automatizáciu organizovania záložiek a ochranu súkromia dát užívateľa. Zvolena kombinácia technológií umožňuje vytvoriť systém, ktorý automaticky prispôsobuje štruktúru záložiek prehliadača aktuálnym potrebám používateľa, pričom zohľadňuje dynamické zmeny v jeho správaní.

Rozšírenie Pre Chrome

Pri vývoji algoritmu adaptívnej organizácie záložiek bolo nutné nájsť spôsob, ktorým by bolo možné interaktívne pracovať s webovým prehliadačom. Rozšírenia pre Chrome sa ukázali byť vhodnou technológiou pre tento účel.

Rozšírenia pre Chrome v spolupráci s Chrome API poskytujú vhodný spôsob získavania užívateľských údajov potrebných pre túto prácu rovnako, ako spôsob upravovania štruktúry

záložiek v prehliadači [31]. Pomocou rozšírení pre Chrome je taktiež možné vytvoriť grafické užívateľské rozhranie, ktoré zabezpečí:

1. Informovanie užívateľa o zmenách vykonávaných rozšírením.
2. Možnosť úpravy nastavení rozšírenia a preferencií užívateľa.
3. Žiadosti o potvrdenia vykonávaných akcií rozšírenia užívateľom.

Podrobnejší popis rozšírení pre Chrome vrátane Chrome API relevantných pre túto prácu je možné nájsť v kapitole 2.

Jazykový model

Jazykový model je výpočtový systém navrhnutý tak, aby dokázal pracovať s prirodzeným jazykom. Jeho hlavným cieľom je správne pochopiť a analyzovať význam slov, na základe ktorých vygeneruje relevantný text odpovede [13].

Vďaka rozsiahlej škále využití našli jazykové modely uplatnenie v rôznych odvetviach. Organizácia IBM vo svojich materiáloch uvádza niekoľko konkrétnych príkladov ich využitia [36]:

- Generovanie textu za účelom vytvárania obsahu správ elektronickej pošty a dokumentov.
- Sumarizácia dlhých článkov či hlásení.
- Preklad textov do rôznych jazykov.
- Asistencia zákazníkom prostredníctvom Chatbotov.
- Automatické generovanie kódu a odhaľovanie bezpečnostných rizík v rámci vývoja aplikácií.

V riešení tejto práce je jazykový model využívaný pre automatizovanie úloh, ako napríklad vypĺňanie názvu záložky alebo triedenie záložiek do kategórií. Jedná sa o úlohy, ktoré by za bežných okolností musel užívateľ vykonávať manuálne. Alternatívne riešenie, ktoré by namiesto využitia jazykového modelu riešilo tieto typy úloh tradičnými metódami, ako sú ručne vytvorené pravidlá alebo preddefinované šablóny, by bolo výrazne komplikovanejšie. Takýto prístup by si vyžadoval rozsiahlu analýzu rôznych typov URL a vytváranie špecifických algoritmov pre každý typ domény webovej stránky.

4.2 Funkcionalita riešenia

Webový prehliadač Chrome obsahuje už vo svojej základnej verzii funkciu záložiek, ktoré slúžia pre zjednodušenie prístupu na webové stránky. So záložkami je možné manuálne vykonávať operácie ako ich pridávanie, upravovanie, odstraňovanie alebo organizovanie v rámci stromovej štruktúry priečinkov [33].

Výsledné riešenie tejto práce slúži ako nadstavba nad základnou funkcionalitou už obsiahnutou v prehliadači Chrome, ktorú rozširuje a automatizuje. Primárne sa zameriavam na oblasť organizácie záložiek, avšak vzhľadom na úzku súvislosť sú v návrhu riešenia zahrnuté aj funkcie automatického pridávania a odstraňovania záložiek.

Automatická organizácia záložiek

Hlavnou funkciou výsledného riešenia je automatické organizovanie záložiek. Záložky v prehliadači Chrome sú ukladané v priečinkoch, ktoré vytvárajú stromovú štruktúru. Táto funkcia sa zaoberá organizáciou záložiek do hierarchickej štruktúry priečinkov tak, aby novo vytvorená štruktúra priečinkov umožnila užívateľovi, čo najjednoduhší prístup k najviac využívaným záložkám. Funkcia automatickej organizácie záložiek zároveň upravuje názvy priečinkov tak, aby jasne vystihovali svoj obsah a v prípade potreby vytvára nové priečinky alebo odstraňuje existujúce priečinky.

Jazykový model zohráva kľúčovú úlohu v rámci tejto funkcie. Na základe existujúcej štruktúry záložiek priradí jednotlivé záložky do pomenovaných kategórií. Pri vytváraní kategórií sú uprednostňované názvy odvodené z existujúcich priečinkov. Výstup jazykového modelu slúži následne ako predloha pre vytvorenie novej štruktúry záložiek v prehliadači, kde priečinky odpovedajú pomenovaným kategóriám. Pridávanie záložiek do nových priečinkov prebieha postupne od najpoužívanejších záložiek, k tým najmenej používaným, čím sa vytvorí štruktúra, kde v každom priečinku budú najpoužívanejšie záložky budú zobrazené ako prvé.

Automatické pridávanie záložiek

V základnom prostredí prehliadača Chrome existuje mnoho spôsobov pridávania záložiek. Medzi užívateľsky najprívetivejšie spôsoby pridania záložiek patrí pridanie záložky kliknutím na ikonu hviezdy, ktorá sa nachádza v pravej strane adresového riadku alebo pravým kliknutím na panel záložiek a vybraním možnosti pridať stránku. V oboch prípadoch je potrebné vyplniť informácie o názve záložky a priečinku, do ktorého má byť záložka uložená. V prípade pridávania záložky, cez panel záložiek je potrebné zadať aj URL cieľovej stránky, na ktorú bude záložka odkazovať.

Automatické pridávanie záložiek odbremeňuje užívateľa od vypĺňania vyššie spomínaných údajov a samo ponúka užívateľovi relevantné webové stránky pre uloženie do záložiek. Výber vhodných webových stránok je vykonávaný na základe frekvencie návštev konkrétnych webových stránok za určité obdobie. Získanie názvu záložky a priečinku, do ktorého má byť záložka uložená prebieha podobne, ako v prípade automatickej organizácie záložiek. Teda prostredníctvom jazykového modelu.

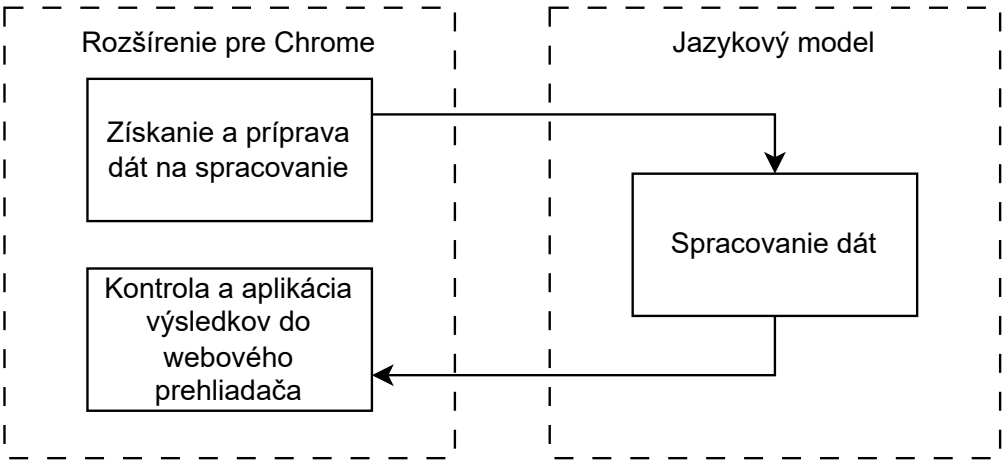
Automatické odstraňovanie záložiek

V prípade automatického odstraňovania záložiek sa nie je potrebné zaoberať problémom získavania informácií o záložkách. Dôležité je však určiť, za akých okolností má byť záložka odstránená. K tomuto účelu sa využíva časový údaj o poslednej návšteve webovej stránky, na ktorú záložka odkazuje. Záložka odkazujúca na webovú stránku, ktorá nebola užívateľom navštívená určitú dobu je považovaná za ďalej nepotrebnú a následne odstránená.

4.3 Návrh algoritmu

Každá z funkcií výsledného riešenia potrebuje pre správne fungovanie dáta, na základe ktorých rozhoduje o svojej činnosti. Po získaní dát je tieto dáta potrebné pripraviť na spracovanie a spracovať, čím sa získajú zmysluplné a využiteľné výsledky, ktoré sa aplikujú do webového prehliadača Chrome vo forme úprav štruktúry záložiek. Tento algoritmus je možné rozdeliť do troch ucelených častí: získanie a príprava dát na spracovanie, spracovanie

dát, kontrola a aplikácia výsledkov do webového prehliadača. Celistvejší náhľad na jednotlivé časti algoritmu a ich prepojenie s využitými technológiami je možné vidieť na obrázku 4.1.



Obr. 4.1: Diagram častí algoritmu a ich prepojenia s využitými technológiami.

Vo zvyšku kapitoly sa venujem podrobnému popisu jednotlivých častí algoritmu, v ktorých zahŕňam princípy Privacy by Design.

Získanie a príprava dát na spracovanie

Získanie a príprava dát na spracovanie je najcitlivejšou časťou algoritmu z pohľadu Privacy by Design, pretože práve v tejto časti sa rozhoduje s akými dátami výsledné riešenie pracuje. Pre minimalizovanie množstva ukladaných citlivých dát som sa rozhodol využiť dostupné Chrome API, ktoré umožňujú prácu s dátami uloženými prehliadačom Chrome. Zaniká tak potreba ukladania dát odvoditeľných z prvkov prehliadača, ako je história vyhľadávania alebo samotné záložky. Dátové úložiská popísané v kapitole 2.2.2 sú kvôli ich nezabezpečenej povahe používané výhradne pre účely ukladania konfigurácií výsledného riešenia a dodatočných údajov, ktoré nie je možné odvodiť z prvkov prehliadača.

Pri získavaní dát pomocou metód obsiahnutých v Chrome API dochádza k načítaniu nadbytočného množstva informácií vrátane osobných údajov, ktoré neprispievajú k zvýšeniu funkčnosti algroritmu. Tieto dáta sú vyfiltrované a odstránené za účelom nevystavovania citlivých údajov. Takto vyfiltrované dáta sa prevedú do štruktúrovaného formátu JSON, čím sa stávajú pripravenými na spracovanie.

Spracovanie dát

Spracovanie dát prebieha prostredníctvom jazykového modelu. Vstup jazykového modelu je tvorený dátovou štruktúrou vo formáte JSON spoločne s výzvou obsahujúcou informácie o spôsobe spracovania dát. V súvislosti s návrhovým vzorom obmedzenia užívateľských údajov som sa rozhodol pre využitie jazykového modelu, ktorý funguje priamo na lokálnom zariadení, čím sa zabráni vystavovaniu osobných údajov serverom tretích strán. Využitie lokálneho jazykového modelu prináša zvýšené výpočtové nároky na zariadenie, vyžaduje značné množstvo úložného priestoru a výsledná kvalita spracovaných dát môže byť nižšia

v porovnaní s veľkými jazykovými modelmi, ktoré sú prevádzkované na cloudových systémoch. Na výstupe modelu sa generuje štruktúrovaný objekt typu JSON, čím sa zjednodušuje proces overenia správnosti a aplikácie výsledkov.

Kontrola a aplikácia výsledkov do webového prehliadača

Spracované dáta, respektíve výstup jazykového modelu nemusí byť vždy úplný alebo správne spracovaný. Pred aplikovaním spracovaných dát je potrebné vykonať kontrolu správnosti formátu a obsahu dát. V prípade chybného formátu spracovaných dát je vhodné celý algoritmus opakovať znovu. Chybný obsah je možné napraviť pomocou dodatočného spracovania, konkrétnych vadných častí alebo manuálnou korekciou zo strany užívateľa.

Po vykonaní kontroly dát a opravení výstupov jazykového modelu sú zmeny aplikované do prostredia webového prehliadača pomocou metód poskytovaných Chrome API.

Kapitola 5

Implementácia riešenia

V tejto kapitole sa detailne venujem implementácii riešenia adaptívnej organizácie záložiek na základe správania užívateľa.

V prvej časti kapitoly sa venujem výberu konkrétneho jazykového modelu, ktorý zohráva kľúčovú úlohu pri vykonávaní jednotlivých funkcií riešenia. Porovnávam výhody a nevýhody jednotlivých spôsobov využitia jazykových modelov ako aj technické výzvy spojené s ich nasadením priamo v prostredí prehliadača. Výsledkom tejto časti je výber konkrétneho jazykového modelu a technológie, ktorá najlepšie spĺňa požiadavky zadania.

Druhá časť kapitoly je zameraná na implementáciu samotného rozšírenia pre Chrome, ktoré využíva zvolený jazykový model na pridávanie nových záložiek a navrhovanie reorganizácie záložiek. V tejto časti popisujem jednotlivé funkcie riešenia, spôsob interakcie s prehliadačom pomocou dostupných Chrome API, spracovanie dátových vstupov a výstupov ako aj logiku rozhodovania o presune či pomenovaní záložiek.

5.1 Výber jazykového modelu

Primárnym spôsobom, ktorým riešenie tejto práce spĺňa princípy Privacy by Design, je využitie lokálneho jazykového modelu namiesto cloudového riešenia. V prípade couldového riešenia by sa vstupné dáta jednotlivých funkcií tejto práce, teda URL webových stránok uložených v záložkách a navštšívených webových stránok museli odosielať na vzdialené servery tretích strán, čím by mohlo dôjsť k neželanému spracovaniu alebo úniku osobných údajov. V tomto riešení sa všetky operácie s využívanými dátami, vrátane klasifikácie a generovania odpovedí, vykonávajú priamo na zariadení užívateľa.

Knižnica Transformers v rámci rozšírenia pre chrome

Pôvodnou myšlienkou bolo využiť jazykové modely, ktoré sú dostupné v rámci knižnice Transformers od spoločnosti Hugging Face. Táto knižnica ponúka spôsob komunikácie so širokým spektrom predtrénovaných modelov rôznej veľkosti a účelu, vrátane modelov vhodných na klasifikáciu alebo generovanie textu [35].

Zvolené kandidátne modely podporované knižnicou Transformers sú uvedené v tabuľke 5.1. Pri výbere modelov boli zohľadnené viaceré kritériá. Konkrétne sa jedná o veľkosť modelu, počet parametrov a presnosť výsledkov modelu. Pri výbere boli preferované modely s veľkosťou v stovkách megabajtov, až jednotkách gigabajtov, aby ich bolo možné využívať na zariadení užívateľa, bez výrazného dopadu na výkon zariadenia. Počet parametrov určuje kapacitu jazykového modelu porozumieť vstupom a vygenerovať kvalitný text. Čím

je vyšší počet parametrov, tým kvalitnejšie výstupy je jazykový model schopný generovať, avšak s vyššími nárokmi na pamäť a výpočtový výkon zariadenia. Presnosť jazykového modelu predstavuje mieru zhody výstupov jazykového modelu s očakávanými výsledkami na štandardizovaných úlohách. Ide o indikátor toho, ako dobre model rozumie kontextu a generuje relevantné alebo správne výstupy pri konkrétnej úlohe.

Názov modelu	**Veľkosť modelu**	**Počet parametrov**	**Presnosť(MuSR)**
TinyLlama-1.1B	2,2 GB	1,1 miliardy	4,31%
Phi-2	5,6 GB	2,7 miliardy	13,84%
Gemma-2B	5,1 GB	2,5 miliardy	7,56%
GPT-2	0,5 GB	137 miliónov	15,35%

Tabuľka 5.1: Porovnanie kandidátnych jazykových modelov [48, 38, 14, 40, 34].

V počiatočných fázach vývoja bola testovaná funkčnosť knižnice Transformers priamo v prostredí rozšírenia pre Chrome. Ako pilotný prípad bol zvolený jednoduchý model `DistilBERT base uncased finetuned SST-2`, určený na sentimentovú analýzu. Tento model, s veľkosťou približne 283 MB a počtom parametrov 67 miliónov, sa podarilo úspešne spustiť a využívať v sandboxe rozšírenia pre Chrome, čo potvrdilo technickú uskutočniteľnosť základného konceptu. Model bol schopný spracovať vstupný text a vrátiť výstup bez potreby pripojenia na internet [7].

Pri pokusoch o implementáciu zložitejších generatívnych jazykových modelov z tabuľky 5.1 sa vyskytli zásadné technické prekážky. Tieto modely majú výrazne vyšší počet parametrov a vyžadujú stovky megabajtov až niekoľko gigabajtov pamäte zariadenia. Prostredie rozšírenia pre Chrome má obmedzenia týkajúce sa veľkosti súborov, pamäťových nárokov a správy výpočtových zdrojov, čo viedlo k zlyhaniu pri načítavaní alebo počas komunikácie s jazykovými modelmi. Problémy sa prejavili ako pády a zamrznutia rozšírenia pre Chrome v dôsledku nedostatku pamäte.

Tento fakt viedol k rozhodnutiu nepokračovať v ceste integrácie väčších jazykových modelov pracujúcich s kinžnicou Transformers priamo do rozšírenia pre Chrome, a namiesto toho preskúmať alternatívne spôsoby, ktoré by umožňovali lokálne spracovanie dát užívateľa. V ďalšej fáze bol skúmaný prístup s backend serverom bežiacim lokálne, ako aj možnosť využitia jazykového modelu Gemini Nano, ktorý je integrovaný v prehliadači Chrome v rámci balíčku ChromeAI.

Knižnica Transformers na lokálnom backend serveri

Alternatívnou možnosťou využitia jazykových modelov pomocou knižnice Transformers je ich spustenie na lokálnom backend serveri, ktorý beží na zariadení užívateľa. Tento server môže vystavovať jednoduché API, ako napríklad WebSocket, s ktorým môže rozšírenie pre Chrome komunikovať prostredníctvom Runtime API, ktoré som rozoberal v kapitole 2.2.3.

Výhodou tohto spôsobu pristupovania k jazykovému modelu je, že nie je obmedzené kapacitami a politikami rozšírenia pre Chrome. Výpočty prebiehajú v plne kontrolovanom prostredí a môžu byť prispôsobené podľa potrieb užívateľa. Umožňuje tiež využívať väčšie modely, ktoré by sa do prostredia prehliadača nezmestili alebo by vyžadovali špecifické optimalizácie.

Nevýhodou však je komplexita správy a synchronizácie medzi prehliadačom a backend serverom. Keďže backend server je na prehliadači nezávislý proces, je potrebné zabezpečiť, aby bol spustený práve vtedy, keď je spustený aj prehliadač Chrome s nainštalovaným

rozšírením. Samotné rozšírenie pre Chrome nemá oprávnenie spúšťať procesy na zariadení z bezpečnostných dôvodov, aby sa predišlo zneužitiu vo forme malvéru, čo znamená, že užívateľ musí backend server spravovať manuálne. Ak užívateľ zabudne server spustiť, rozšírenie nebude schopné komunikovať s modelom, čo bude viesť k nefunkčnému správaniu aplikácie. Naopak, ak server ostane spustený aj po ukončení prehliadača, dochádza k neefektívnemu využitiu systémových prostriedkov. Navyše, manuálna správa tohto procesu môže byť pre bežného užívateľa neintuitívna a znižuje celkovú užívateľskú prívetivosť riešenia. Z týchto dôvodov je použitie backend servera vhodnejšie pre technicky zdatnejších užívateľov alebo na účely prototypovania a vývoja.

ChromeAI a Gemini Nano

Riešením problému s využitím lokálneho jazykového modelu je súbor funkcií umelej inteligencie ChromeAI, ktorý je integrovaný priamo do webového prehliadača Chrome. ChromeAI poskytuje viacero API rozhraní, ktoré umožňujú prístup k modelom strojového učenia bez potreby externého serverového spracovania. Tieto modely sú navrhnuté tak, aby bežali efektívne priamo na užívateľovom zariadení, čím sa znižuje latencia, zvyšuje ochrana súkromia a znižuje sa spotreba dát [30].

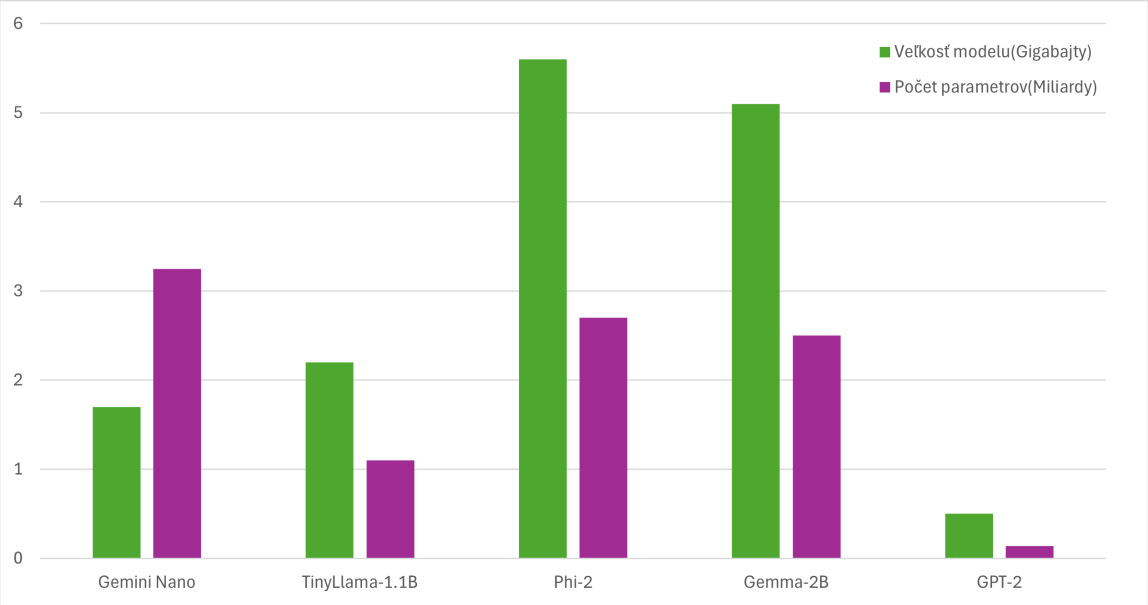
Konkrétnym API, ktoré je v tejto práci využité, je Prompt API. Jedná sa o experimentálne rozhranie, ktoré umožňuje vývojárom odosielať správy v prirodzenom jazyku a získavať odpoveďe od integrovaného jazykového modelu [32].

Jazykový model použitý v rámci rozhrania Prompt API je Gemini Nano, čo je malý, efektívny jazykový model vyvinutý spoločnosťou Google. Gemini Nano je navrhnutý špecificky pre beh na zariadeniach s obmedzenými zdrojmi, ako sú mobilné telefóny alebo notebooky, pričom využíva optimalizované architektúry a akceleráciu pomocou dostupného hardvéru. Gemini Nano obsahuje 3,25 miliady parametrov a zaberá 1,7 GB miesta, čo z neho robí výhodného kandidáta pre použitie v rozšírení pre Chrome. V porovnaní s inými jazykovými modelmi uvedenými v tabuľke 5.1 dosahuje lepší pomer medzi výpočtovým výkonom a veľkosťou modelu. Tento pomer parametrov a veľkosti je vizualizovaný v grafe na obrázku 5.1 [47].

Aby bolo možné jazykový model Gemini Nano využívať priamo vo webovom prehliadači Chrome, je nevyhnutné použiť verziu prehliadača minimálne 127. Používateľ musí manuálne aktivovať podporu pre model na zariadení prostredníctvom experimentálnych príznakov. Konkrétne ide o nastavenie príznaku `Prompt API for Gemini Nano` na hodnotu `Enabled` a príznaku `Enables optimization guide on device` na hodnotu `Enabled BypassPerfRe- quirement`. Tieto príznaky sú dostupné na stránke `chrome://flags/`. Následne je potrebné aktualizovať komponentu `Optimization Guide On Device Model`, ktorá sa nachádza v sekcii `chrome://components/`. Po správnom nastavení je možné overiť dostupnosť a aktiváciu modelu prostredníctvom konzoly prehliadača pomocou objektu `window.ai`, ktorý slúži ako API rozhranie pre komunikáciu s modelom [6].

5.2 Implementácia rozšírenia pre Chrome

Navrhnuté riešenie adaptívnej organizácie záložiek bolo implementované formou rozšírenia pre webový prehliadač Chrome. Z technického hľadiska je možné rozdeliť riešenie na dve oddelené časti:



Obr. 5.1: Graf pomeru veľkostí a počtu parametrov jazykových modelov.

1. **Grafické užívateľské rozhranie** zabezpečujúce interakciu užívateľa s rozšírením pre Chrome a zároveň spracúva všetky akcie, ktoré užívateľ v rozhraní manuálne vykoná. Ide napríklad o manuálne spúšťanie jednotlivých funkcií riešenia, úpravu nastavení rozšírenia či reakcie na notifikácie.

2. **Background script**, ktorý obsahuje hlavnú logiku riešenia, zachytáva udalosti vyvolávané webovým prehliadačom a vykonáva kľúčové operácie súvisiace s triedením a organizáciou záložiek

Funkcionalita riešenia je navrhnutá tak, aby bola primárne vyvolaná automaticky na základe špecifických udalostí v prostredí prehliadača a nie prostredníctvom priamej akcie používateľa v grafickom rozhraní. Z tohto dôvodu bolo potrebné implementovať background script, ktorý beží na pozadí a má schopnosť neustále monitorovať relevantné udalosti v prehliadači.

Na sprostredkovanie komunikácie medzi background scriptom a popup komponentom grafického používateľského rozhrania je využité rozhranie Runtime API spomínané v kapitole 2.2.3, ktoré umožňuje obojsmerné zasielanie správ medzi jednotlivými časťami rozšírenia. V prípade úspešného vykonania príslušnej funkcie odošle background script správu grafickému rozhraniu, ktoré následne zobrazí notifikáciu s výsledkami užívateľovi. Tento mechanizmus zabezpečuje, že aj keď logika prebieha na pozadí, užívateľ má o výsledkoch spracovania okamžitú spätnú väzbu prostredníctvom zobrazeného výstupu v rozhraní rozšírenia.

Grafické užívateľské rohranie

Grafické užívateľské rozhranie bolo navrhnuté s dôrazom na jednoduchosť a minimálne rušenie užívateľa pri práci s webovým prehliadačom. Jeho návrh je čiastočne inšpirovaný rozšírením Privacy Badger, ktoré po kliknutí na ikonu rozšírenia zobrazí užívateľovi zrozumiteľný prehľad blokovaných sledovacích prvkov na aktuálnej webovej stránke [9].

V prípade navrhnutého grafického užívateľského rozhrania sa po kliknutí na ikonu rozšírenia zobrazí panel, ktorý zobrazuje notifikácie s informáciami o vykonaných funkciách riešenia. Okrem notifikácií panel obsahuje aj interaktívne prvky, ktoré umožňujú používateľovi ovplyvniť, akým spôsobom sa majú jednotlivé odporúčané zmeny hierarchickej štruktúry záložiek aplikovať do webového prehliadača. Ide napríklad o možnosť prijať alebo odmietnuť navrhované úpravy, prípadne manuálne zasiahnuť do odporúčaného riešenia. Názornú ukážku zobrazenej notifikácie v rozšírení pre Chrome je možné vidieť na obrázku 5.2.

Obr. 5.2: Ukážka grafického uživateľského rozhrania.

Grafické užívateľské rozhranie taktiež obsahuje sekciu nastavení, v ktorej môže užívateľ prispôsobiť správanie celého riešenia podľa svojich preferencií. Konfigurovateľné sú napríklad prahové hodnoty, na základe ktorých sú generované notifikácie pre pridanie alebo odstránenie záložiek ako aj výber konkrétnych funkcií riešenia, ktoré si užívateľ praje spúšťať automaticky na pozadí bez jeho zásahu. Týmto prístupom sa napĺňa súlad s návhovým vzorom ochrany súkromia `Voľba využívaných funkcií` popísaným v kapitole 3.3, keďže užívateľovi dodáva plnú kontrolu nad výberom jednotlivých funkcií riešnia, ktoré si praje využívať.

Prítomnosť nových notifikácií je indikovaná malým bielym štvorčekom s číselným údajom o počte nových notifikácií v rohu ikony rozšírenia. Tento minimalistický spôsob signalizácie nových notifikácií bol zvolený, aby užívateľa pri bežnom používaní webového prehliadača zbytočne nerušil, no zároveň mu poskytoval jasnú informáciu o aktivite rozšírenia.

Background script

Druhou časťou implementovaného riešenia je background script, ktorý je implementovaný ako service worker. Tento skript je automaticky spúšťaný v reakcii na definované udalosti webového prehliadača, ako je napríklad aktualizácia rozšírenia pre Chrome alebo otvorenie novej karty prehiadača. Vďaka schopnosti reagovať na udalosti webového prehliadača bez potreby trvalého behu umožňuje background scriptu efektívne spracovanie užívateľských dát na pozadí pri minimálnej spotrebe systémových zdrojov.

Background script slúži ako centrálny koordinačný bod celého riešenia. Obsahuje implementáciu hlavných funkcií rozšírenia, spúšťaných na základe špecifických udalostí vo webovom prehliadači. Výsledkom vykonania týchto funkcií sú vygenerované notifikácie s odporú-

čaniami na reorganizáciu štruktúry záložiek v prehliadači Chrome, ktoré sú ďalej spracované grafickým užívateľským rozhraním.

Jednou z hlavných funkcií riešenia je automatizovaná organizácia záložiek. Účelom tejto funkcie je na základe aktuálnej štruktúry záložiek v prehliadači vytvoriť novú, prehľadnejšiu štruktúru záložiek, ktorá má užívateľovi zjednodušiť orientáciu v záložkách prostredníctvom ich preorganizovania. Táto funkcia je automaticky spustená pri inštalácii alebo aktualizácii rozšírenia v prehliadači Chrome.

Funkcia najprv prečíta a analyzuje aktuálnu štruktúru záložiek vrátane ich priečinkov a názvov. Na základe získaných údajov vygeneruje štruktúrovaný prompt (vid. kód 5.1), ktorý je následne odoslaný lokálnemu jazykovému modelu. Tento prompt obsahuje informácie o existujúcich záložkách a priečinkoch v špecifickom formáte, na základe ktorého jazykový model priradí záložky do nových priečinkov. Výstup jazykového modelu je prevedený na JSON objekt a prípadné nezrovnalosti v novo vytvorenej štruktúre záložiek sú napravené. Následne je užívateľ informovaný notifikáciou o návrhu novej štruktúry záložiek, ktorú môže schváliť alebo odmietnuť.

```
1  const prompt = `Task:
2  Group the following bookmarks into meaningful categories (maximum 7).
3  Input:
4  Each bookmark has a title and a URL.
5  Bookmarks: ${JSON.stringify(bookmarkArray)}
6  Existing categories (use these when possible): ${JSON.stringify(formatedFolders)}
7  Rules:
8  You must not create more than 7 categories total.
9  You must assign every single bookmark from the input to exactly one category.
10 You must not assign the same id to more than one category, and do not list the same
11 id multiple times within a single category.
12 Reuse category titles from the provided list if suitable.
13 If none fit, create a new category name.
14 If a bookmark doesn't clearly belong to any group, assign it to the category "Other".
15 Group bookmarks with similar topics under the same category.
16 Output:
17 JSON.stringify format only.
18 No extra text.
19 No explanations.
20 Example:
21 [
22 {
23     "categoryTitle": "Shopping",
24     "bookmarkId": ["123", "987"]
25 },
26 {
27     "categoryTitle": "News",
28     "bookmarkId": ["566"]
29 }
30 ]`;
```

Kód 5.1: Prompt funkcie organizácie záložiek.

V prípade, že používateľ reorganizáciu záložiek schváli, je nová štruktúra záložiek aplikovaná do webového prehliadača, čím sa okamžite aktualizuje ich usporiadanie. Ak používateľ

reorganizáciu odmietne, žiadne zmeny sa nevykonajú a pôvodná štruktúra záložiek zostáva zachovaná.

Ďalšou z implementovaných funkcií riešenia je automatické pridávanie záložiek, ktorého účelom je navrhnúť užívateľovi pridanie novej záložky do existujúcej hierarchickej štruktúry záložiek, na základe správania užívateľa počas používania webového prehliadača. Táto funkcia je aktivovaná vždy, keď užívateľ otvorí novú kartu vo webovom prehliadači, respektíve navštívi webovú stránku.

Počas aktivácie funkcie dochádza k vyhodnocovaniu viacerých podmienok. Primárne sa kontroluje, či sa aktuálna URL navštívenej webovej stránky:

1. už nachádza medzi existujúcimi záložkami používateľa,
2. nachádza v zozname webových stránok, ktoré používateľ v minulosti odmietol pridať do svojich záložiek.

Ak sa navštívená stránka nenachádza v žiadnom z uvedených zoznamov, funkcia automatického pridávania záložiek pokračuje v hodnotení ďalších parametrov, ktoré sú konfigurovateľné užívateľom v nastaveniach rozšírenia. Konkrétne ide o počet navštívení webovej stránky v definovanom časovom intervale, čo slúži ako indikátor záujmu užívateľa o pridanie webovej stránky do štruktúry záložiek.

V prípade, že sú všetky podmienky splnené, background script vygeneruje na základe preddefinovaných promptov (viď kód 5.2) požiadavky pre jazykový model, ktoré určia vhodný názov a priečinok, do ktorého má byť záložka pridaná. Výsledný návrh záložky je následne spracovaný a užívateľskému rozhraniu je odoslaná informácia o novej notifikácii.

```
1  const titlePrompt = `Create a short and descriptive bookmark title for this URL
2  "${currentUrl}". Print just the bookmark title.`;
3
4  const folderPrompt = `Classify the following bookmark with title:
5  "${bookmarkTitle}" and url: "${currentUrl}" into one of the following categories:
6  ${JSON.stringify(folderTitles)}. If you're not sure, return an empty string.
7  Print only the category.`;
```

Kód 5.2: Prompty funkcie pridávania záložiek.

Notifikácia obsahuje možnosť okamžitého pridania navrhovanej záložky do existujúcej štruktúry záložiek prehliadača. Alternatívne môže používateľ návrh odmietnuť. V takom prípade je daná URL webovej stránky zaradená do zoznamu odmietnutých stránok a v budúcnosti už nebude navrhovaná na pridanie medzi záložky, čím sa predchádza budúcemu zaťažovaniu užívateľa neželanými návrhmi.

Poslednou implementovanou funkciou riešenia je automatické odstraňovanie záložiek, ktorého účelom je identifikovať a navrhovať odstránenie zastaraných alebo dlhodobo nepoužívaných záložiek. Táto funkcia riešenia je aktivovaná pri spustení webového prehliadača.

Po aktivácii funkcia analyzuje históriu prehliadania pomocou History API, a z nej získa zoznam navštívených webových stránok za zvolené časové obdobie. Dĺžku tohto obdobia je možné konfigurovať užívateľom prostredníctvom nastavení v rozšírení pre Chrome. Následne sa vykoná filtrovanie záložiek, ktoré:

1. sa aktuálne nachádzajú v zozname záložiek používateľa,
2. a zároveň neboli medzi navštívenými stránkami v danom časovom intervale,
3. a zároveň neboli používateľom v minulosti manuálne označené ako záložky, ktoré si neželá odstrániť.

Výsledný zoznam potenciálne nepotrebných záložiek je prezentovaný užívateľovi prostredníctvom novej notifikácie. Používateľ má možnosť vybrať, ktoré konkrétne záložky si praje odstrániť, a ktoré chce zachovať. Záložky označené na odstránenie sú následne automaticky vymazané z prehliadača. Naopak, záložky, ktoré používateľ označí, že si želá ponechať, sú pridané do zoznamu záložiek vylúčených z budúcich návrhov na odstránenie. Tým sa zabezpečuje, že sa tieto záložky nebudú v budúcnosti opätovne objavovať v návrhoch na zmazanie, čím sa predchádza budúcemu zaťažovaniu užívateľa neželanými návrhmi.

Kapitola 6

Testovanie

Cieľom tejto kapitoly je overiť funkčnosť, presnosť a súlad riešenia s požiadavkami zadania. Testovanie bolo vykonané v prostredí webového prehliadača Google Chrome, pričom boli overované predovšetkým funkcie riešenia, ktoré závisia na výstupoch z jazykového modelu. Všetky experimenty a testovacie scenáre boli realizované na dátach v anglickom jazyku. Parametre prostredia, v ktorom bolo riešenie testované je možné vidieť v tabuľke 6.1.

Operačný systém	Windows 11
Verzia Chrome	136.0.7103.93
Operačná pamäť	16 GB
Procesor	Intel i5-1135G7
Grafická karta	Intel Iris Xe Graphics

Tabuľka 6.1: Parametre prostredia použitého pre testovanie výsledného riešenia.

V nadchadzajúcej časti kapitoly sú popísané vykonané experimenty, ktoré overujú správanie algoritmu v rôznych scenároch používania. Testovanie je následne doplnené o užívateľskú štúdiu vo forme dotazníka, v rámci ktorej mali respondenti zhodnotiť kvalitu výstupov implementovaného riešenia. Záver kapitoly je tvorený celkovým vyhodnotením výsledkov testovania a ich zhodnotením z pohľadu cieľov práce.

6.1 Experimenty

Táto časť kapitoly sa zameriava na overenie presnosti a spoľahlivosti riešenia v rôznych scenároch. Cieľom experimentov je otestovať, ako efektívne algoritmus triedi záložky, identifikuje tematickú orientáciu webových stránok a navrhuje ich zaradenie do vhodných kategórií. Súčasťou experimentov je aj analýza výkonnosti riešenia pri spracovaní rozlične veľkých a tematicky rôznorodých vstupov.

Prvé štyri experimenty sú zamerané na testovanie funkcie organizácie záložiek. Piaty experiment sa vzťahuje na schopnosť funkcie pridávania záložiek správne zaradiť záložku do existujúcej kategórie. Úlohou šiesteho experimentu je ohaliť limity zadania v zmyle maximálneho počtu záložiek, ktoré je funkcia organizácie schopná spracovať. V experimentoch sa sledujú dva hlavné parametre:

1. Správnosť zaradenia záložky do tematicky vhodnej kategórie, respektíve priečinka záložiek.

2. Čas potrebný na vykonanie testovanej funkcie.

Vstupom pre experimenty sú webové stránky a predpripravené kolekcie záložiek vo webovom prehliadači, ktoré zodpovedajú povahe konkrétneho experimentu. Správnosť zaradenia webovej stránky do príslušnej kategórie je overovaná pomocou porovnania názvu kategórie s relevantnými kľúčovými slovami získanými z meta údajov v hlavičke HTML dokumentu. V prípade, že sa názov kategórie, do ktorej bola záložka zaradená nachádza v kľúčových slovách metadát webovej stránky, stránka je označená ako úspešne zaradená. Pre účely experimentov boli preto volené výhradne webové stránky obsahujúce zmienený typ meta údajov.

Prvý experiment bol realizovaný na vzorke desiatich záložiek, ktoré odkazovali na webové stránky tematicky zamerané na oblasti športu a technológií. Cieľom experimentu bolo overiť schopnosť riešenia správne identifikovať tematický kontext jednotlivých webových stránok a zaradiť ich do zodpovedajúcich kategórií. Z celkového počtu záložiek bolo správne klasifikovaných 8, čo predstavuje úspešnosť 80%. Čas potrebný na spracovanie tejto kolekcie predstavoval 40,8 sekundy.

Druhý experiment vychádzal z rovnakej kolekcie desiatich záložiek ako v predchádzajúcom prípade, avšak štruktúra záložiek bola doplnená o preddefinované priečinky. Tieto priečinky niesli názvy zodpovedajúce očakávaným kategóriám šport a technológie, čo poskytlo algoritmu dodatočný kontext pre presnejšie zaradenie. Cieľom experimentu bolo overiť, do akej miery sa zlepší presnosť klasifikácie záložiek v prípade, že sú už vopred k dispozícii potenciálne cieľové kategórie. Za týchto podmienok bolo správne zaradených 9 z 10 záložiek, čo predstavuje úspešnosť 90%. Čas potrebný na vykonanie funkcie klesol oproti prvému experimentu na 18,7 sekundy.

Tretí experiment bol navrhnutý s cieľom overiť správanie riešenia pri spracovaní väčšieho objemu údajov. Testovacia kolekcia pozostávala z tridsiatich záložiek patriacich do piatich tematických oblastí: zdravie, spravodajstvo, vzdelanie, nákupy a počasie. Experiment sa zameriaval na hodnotenie kategorizácie a výkonu riešenia pri rozšírenej množine dát. Za týchto podmienok sa systému podarilo správne zaradiť 11 záložiek, čo poukazuje na náročnosť klasifikácie pri vyššej rozmanitosti tém a väčšom objeme vstupov. Celkový čas potrebný na spracovanie kolekcie bol 85,6 sekundy.

Štvrtý experiment nadväzoval na predchádzajúci a využíval identickú kolekciu tridsiatich záložiek aká bola použitá v treťom experimente. Rozdielom však bolo rozšírenie štruktúry záložiek o priečinky, ktorých názvy zodpovedali očakávaným kategóriám. Konkrétne zdravie, spravodajstvo, vzdelanie, nákupy a počasie. Cieľom experimentu bolo overiť, či prítomnosť explicitne pomenovaných kategórií prispeje k zlepšeniu presnosti zaradenia záložiek. V tomto prípade riešenie správne klasifikovalo 10 záložiek. Spracovanie vstupov trvalo 121,7 sekundy, čo naznačuje nárast časovej náročnosti pri väčšej množine vstupných dát.

Prehľad nameraných hodnôt z prvých štyroch experimentov je uvedený v tabuľke 6.2.

Experiment č.	Počet záložiek	Priečinky	Správne zaradené	Trvanie [s]
1	10	nie	8	40,8
2	10	áno	9	18,7
3	30	nie	11	85,6
4	30	áno	10	121,7

Tabuľka 6.2: Výsledky experimentov 1 až 4 zameraných na testovanie funkcie organizácie záložiek.

Piaty experiment sa zameriaval na overenie schopnosti riešenia správne kategorizovať jednotlivé webové stránky do vopred existujúcich priečikov na základe URL a vygenerovaného názvu. Testovacia vzorka pozostávala z 20 webových stránok a 10 preddefinovaných priečinkov, pričom každý priečinok reprezentoval jednu kategóriu, do ktorej mali byť zaradené práve dve webové stránky. Webové stránky boli spracovávané jednotlivo a nezávisle od seba. Priemerný čas potrebný na spracovanie jednej webovej stránky bol 3,4 sekundy. Riešeniu sa podarilo správne zaradiť 90% webových stránok do relevantného priečinka.

Posledný, šiesty experiment mal identifikovať limity jazykového modelu pri spracovaní veľkého počtu záložiek počas volania funkcie organizácie záložiek. Experiment prebiehal postupným zvyšovaním počtu vstupných záložiek, pričom cieľom bolo zistiť, pri akom množstve vstupných záložiek prestane model generovať výstup. Zlyhania algoritmu v podobe chyby na výstupe jazykového modelu sa začali prejavovať po prekročení hranice približne 80 záložiek. Napriek tomu sa v niektorých prípadoch podarilo úspešne spracovať aj vyšší počet, konkrétne v rozmedzí od 85 do 90 záložiek.

6.2 Užívateľský prieskum

Overenie presnosti riešenia bolo dodatočne realizované pomocou užívateľského prieskumu formou dotazníka. Táto metóda bola zvolená vzhľadom na povahu implementovaného riešenia, pri ktorom generované výstupy majú nedeterministický charakter a neexistuje pre ne jednoznačne správna odpoveď. Z tohto dôvodu nie je testovanie prostredníctvom klasických metód, kde sa výstupy porovnávajú s očakávanými hodnotami, dostatočné na dôkladné zhodnotenie kvality výsledkov. Hodnotenie pomocou spätnej väzby od reálnych používateľov tak umožňuje komplexnejšie zhodnotiť, či výstupy riešenia pôsobia logicky, zrozumiteľne a užitočne v kontexte očakávaného správania systému.

Prieskum bol zameraný na testovanie funkcie pridávania záložiek. Pozostával z 25 otázok, v ktorých mal dotazovaný určiť, ako veľmi je vhodný daný názov záložky pre vybranú webovú stránku. Webové stránky boli náhodne vybrané zo zoznamu 100 najnavštevovanejších webových stránok, vydaného organizáciou Ahrefs [1]. Respondenti mali vybrať jednu z piatich možností:

- Sám by som si zvolil/a takýto názov záložky.
- Celkom sa mi páči, ale trochu by som ho upravil/a.
- Je to v poriadku, ale nie je to nič výnimočné.
- Nepáči sa mi, názov nie je dostatočne výstižný.
- Najhorší možný názov. Určite by som si ho nevybral.

Užívateľského prieskumu sa zúčastnilo 9 respondentov vo veku od 20 do 24 rokov. Všetci respondenti sú študentmi vysokých škôl s technickým zameraním a primeranou úrovňou digitálnej gramotnosti.

Jednotlivým odpovediam boli lineárne priradené percentuálne hodnoty od 100% pri najpozitívnejšej odpovedi po 0% v prípade najviac negatívnej odpovede. Priemerné hodnotenie bolo vypočítané ako aritmetický priemer odpovedí od všetkých respondentov.

Z výsledkov prieskumu vyplíva, že zvolené názvy záložiek sú pre cieľovú skupinu prevažne uspokojivé, avšak nie ideálne. Priemerná spokojnosť na úrovni 79% naznačuje, že aj keď je výstup algoritmu vo všeobecnosti vnímaný pozitívne, stále existujú možnosti na zlepšenie a doladenie názvov pre optimálne prispôsobenie používateľským potrebám.

6.3 Vyhodnotenie dosiahnutých výsledkov

Implementované riešenie je schopné organizovať záložky v internetovom prehliadači na základe správania používateľa. Táto organizácia je zabezpečená súborom funkcií, ktoré zahŕňajú pridávanie, usporiadávanie a odstraňovanie záložiek. Konkrétne implementačné detaily týchto funkcií sú uvedené v kapitole 5.2.

Výsledky experimentov ukázali, že pri menšom počte záložiek algoritmus dosahuje vysokú mieru správneho zaradenia, no so zvyšujúcim sa počtom záložiek dochádza k poklesu presnosti. Tento jav možno pripísať limitom použitého jazykového modelu, ako aj zložitejšej štruktúre záložiek a rôznorodosti webových stránok pri väčších množstvách údajov. Pokles presnosti bol často spôsobený práve tendenciou jazykového modelu priradiť ku jednej záložke viacero kategórií. Pri následnom vyhodnocovaní boli tieto duplikáty z výstupov odstránené, avšak tento proces viedol aj k nechcenému odstráneniu záložiek z relevantných kategórií.

Dôležitým aspektom riešenia bolo dodržanie princípov Privacy by Design, ktoré sa prejavili predovšetkým v tom, že všetky údaje sú spracovávané výhradne na zariadení používateľa. Pre tento účel bol použitý lokálny jazykový model, ktorý umožňuje vykonávať kategorizáciu záložiek bez potreby odosielania údajov na vzdialené servery. Uchovávanie citlivých údajov používateľa bolo zredukované na minimum prostredníctvom dostupných rozhraní Chrome API, ktoré umožňujú získať potrebné informácie priamo z prostredia webového prehliadača bez potreby ich dlhodobého ukladania. Prvú výnimku pri uchovávaní dát tvoria notifikácie generované funkciami riešenia, ktoré je potrebné uchovávať do momentu, kedy užívateľ danú notifikáciu spracuje. Druhou výnimkou je zoznam webových stránok, ktoré užívateľ v rámci funkcie pridávania záložiek označil, ako neželané pre pridanie do zoznamu záložiek. Tento údaj je uchovávaný za účelom zamedzenia opätovného dotazovania používateľa.

Adaptívnosť riešenia je realizovaná pomocou sledovania užívateľských návykov, na základe ktorých algoritmus rozhoduje, kedy majú byť webové stránky pridávané medzi záložky a za akých okolností majú byť záložky odstránené. Pridanie novej záložky sa uskutoční len v prípade, že určitá stránka bola navštívovaná užívateľom opakovane počas stanoveného časového obdobia. Naopak, odstránenie záložky nastáva vtedy, keď systém zistí, že užívateľ danú webovú stránku nenavštívil počas definovaného obdobia. Tieto parametre sú nastaviteľné v užívateľskom rozhraní, čo umožňuje prispôsobenie správania algoritmu individuálnym potrebám.

Celkovo možno skonštatovať, že implementované riešenie spĺňa stanovené ciele a poskytuje funkčný základ pre inteligentnú a zároveň súkromie rešpektujúcu správu záložiek.

V priebehu testovania boli síce identifikované určité obmedzenia, avšak vzhľadom na charakter riešenej problematiky a použité technologické prostriedky sú tieto limity pochopiteľné.

Kapitola 7

Záver

Cieľom tejto práce bolo navrhnúť a implementovať algoritmus, ktorý adaptívne organizuje záložky vo webových prehliadačoch založených na jadre prehliadača Chrome v súlade s princípmi Privacy by Design.

V úvodnej fáze práce sa bolo potrebné oboznámiť s fungovaním rozšírení pre prehliadač Chromium, konkrétne Google Chrome, ktorý je kompatibilný s Avast Secure Browser. Vzhľadom na cieľ práce bola pozornosť venovaná predovšetkým správe záložiek a komponentom, ktoré umožňujú interakciu s užívateľom. Bolo potrebné naštudovať Chrome API, ktoré zohrávajú kľúčovú úlohu pri práci so záložkami, ako aj pri interakcii s ďalšími komponentmi, ako sú bookmark API, storage API, history API a runtime API.

Pred návrhom riešenia bolo nevyhnutné podrobne preskúmať koncept Privacy by Design, ktorý kladie dôraz na ochranu súkromia užívateľov už od počiatočných fáz vývoja systémov a aplikácií. Dôkladné pochopenie princípov Privacy by Design bolo kľúčové pre zabezpečenie, aby navrhovaný algoritmus prirodzene rešpektoval súkromie užívateľa a minimalizoval riziko zneužitia dát už vo svojej podstate.

Na základe naštudovaných teoretických poznatkov bol navrhnutý a implementovaný algoritmus riešenia, ktorý s pomocou lokálneho jazykového modelu automatizuje rutinné úlohy spojené so správou záložiek, ktoré užívateľ vykonáva manuálne. Konkrétne, riešenie automatizuje pridávanie nových záložiek na základe frekvencie návštev webových stránok, organizáciu záloziek do štruktúrovaných kategórií a priebežné odstraňovanie nepoužívaných záložiek.

Výsledná implementácia bola otestovaná sériou experimentov zameraných na určenie presnosti a výkonnosti pri vykonávaní automatizovaných úloh. Súčasťou testovania bola aj užívateľská štúdia, ktorá poskytla spätnú väzbu v situáciách, kde nebolo možné riešenie overiť bežnými metódami testovania.

Výstupy testovania poukázali na nedostatky využitia lokálneho jazykového modelu najmä pri spracovaní rozsiahlych textových dát. Tieto obmedzenia sa prejavili v zníženej presnosti organizácie záložiek, čo je dôsledkom limitov modelov určených pre lokálne nasadenie. Zistené nedostatky zároveň poskytujú cenné poznatky pre ďalší vývoj a optimalizáciu algoritmu, s dôrazom na zvýšenie jeho efektivity pri práci s rôznorodejším obsahom.

V rámci možného pokračovania projektu by sa mohla pozornosť venovať vývoju a trénovaniu vlastného modelu umelej inteligencie. Použitie špecializovaného modelu, optimalizovaného priamo pre účely správy záložiek a rozpoznávanie kontextu webových stránok, by mohlo prispieť k dosiahnutiu vyššej presnosti v porovnaní s výsledkami všeobecného jazykového modelu.

Literatúra

[1] Ahrefs. *Top 100 most visited websites in the world* [online]. 2025. Navštívené 2025-5-12. Dostupné z: https://ahrefs.com/websites.

[2] Carbide Secure. *The Seven Principles of Privacy by Design* [online]. 2024. Navštívené 2024-12-31. Dostupné z: https://carbidesecure.com/resources/the-seven-principles-of-privacy-by-design/.

[3] Cavoukian, A. *Privacy by Design: The 7 Foundational Principles – Implementation and Mapping of Fair Information Practices.* Information & Privacy Commissioner Ontario, Canada, máj 2010.

[4] Chaum, D. L. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM.* New York, NY, USA: Association for Computing Machinery, február 1981, zv. 24, č. 2, s. 84–90. ISSN 0001-0782. Dostupné z: https://doi.org/10.1145/358549.358563.

[5] Chromium Developers. *Sync Model API* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://www.chromium.org/developers/design-documents/sync/model-api/.

[6] debugtheworldbot. *Chromegemini* [online]. 2025. Navštívené 2025-5-12. Dostupné z: https://github.com/debugtheworldbot/chromegemini.

[7] DistilBERT community. *DistilBERT base uncased finetuned SST-2* [online]. 2025. Navštívené 2025-05-07. Dostupné z: https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english.

[8] Doty, N. a Gupta, M. Privacy Design Patterns and Anti-Patterns: Patterns Misapplied and Unintended Consequences. In: *Proceedings of the 9th Symposium on Usable Privacy and Security (SOUPS 2013).* New York, NY, USA: Association for Computing Machinery, 2013, s. 45–60. Dostupné z: https://cups.cs.cmu.edu/soups/2013/trustbusters2013/Privacy_Design_Patterns-Antipatterns_Doty.pdf.

[9] Electronic Frontier Foundation. *Privacy Badger* [online]. 2025. Navštívené 2025-5-12. Dostupné z: https://privacybadger.org/.

[10] European Parliament and Council of the European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation).* 2016. Dostupné z:

https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=celex:32016R0679. Official Journal of the European Union.

[11] GDPR-INFO.EU. *Privacy by Design* [online]. 2024. Navštívené 2024-12-31. Dostupné z: https://gdpr-info.eu/issues/privacy-by-design/.

[12] GDPR.EU. *What is GDPR?* [online]. 2024. Navštívené 2024-12-31. Dostupné z: https://gdpr.eu/what-is-gdpr/.

[13] GOODFELLOW, I.; BENGIO, Y. a COURVILLE, A. *Deep Learning*. MIT Press, 2016. Dostupné z: http://www.deeplearningbook.org. Book in preparation for MIT Press.

[14] GOOGLE. *Gemma 2B* [online]. 2025. Navštívené 2025-05-07. Dostupné z: https://huggingface.co/google/gemma-2b.

[15] GOOGLE DEVELOPERS. *Action API Reference* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.chrome.com/docs/extensions/reference/api/action.

[16] GOOGLE DEVELOPERS. *Bookmarks API Reference* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.chrome.com/docs/extensions/reference/api/bookmarks.

[17] GOOGLE DEVELOPERS. *Content Scripts in Chrome Extensions* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.chrome.com/docs/extensions/develop/concepts/content-scripts.

[18] GOOGLE DEVELOPERS. *Get Started with Chrome Extensions* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.chrome.com/docs/extensions/get-started.

[19] GOOGLE DEVELOPERS. *History API Reference* [online]. 2024. Navštívené 2024-12-30. Dostupné z: https://developer.chrome.com/docs/extensions/reference/api/history.

[20] GOOGLE DEVELOPERS. *Manifest File Format* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.chrome.com/docs/extensions/reference/manifest.

[21] GOOGLE DEVELOPERS. *Messaging in Chrome Extensions* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.chrome.com/docs/extensions/develop/concepts/messaging.

[22] GOOGLE DEVELOPERS. *Runtime API Reference* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.chrome.com/docs/extensions/reference/api/runtime.

[23] GOOGLE DEVELOPERS. *Scripting API Reference* [online]. 2024. Navštívené 2024-12-30. Dostupné z: https://developer.chrome.com/docs/extensions/reference/api/scripting.

[24] GOOGLE DEVELOPERS. *Service Workers Basics* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.chrome.com/docs/extensions/develop/concepts/service-workers/basics.

[25] Google Developers. *Service Workers in Chrome Extensions* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.chrome.com/docs/extensions/develop/concepts/service-workers.

[26] Google Developers. *Service Workers Lifecycle* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.chrome.com/docs/extensions/develop/concepts/service-workers/lifecycle.

[27] Google Developers. *Stay Secure in Chrome Extensions* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.chrome.com/docs/extensions/develop/security-privacy/stay-secure.

[28] Google Developers. *Storage API Reference* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.chrome.com/docs/extensions/reference/api/storage.

[29] Google Developers. *User Privacy in Chrome Extensions* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.chrome.com/docs/extensions/develop/security-privacy/user-privacy.

[30] Google Developers. *Built-in AI* [online]. 2025. Navštívené 2025-05-07. Dostupné z: https://developer.chrome.com/docs/ai/built-in.

[31] Google Developers. *Develop Extensions* [online]. 2025. Dostupné z: https://developer.chrome.com/docs/extensions/develop. Navštívené 2025-01-11.

[32] Google Developers. *The Prompt API* [online]. 2025. Navštívené 2025-5-7. Dostupné z: https://developer.chrome.com/docs/extensions/ai/prompt-api.

[33] Google Support. *Create, view & edit bookmarks* [online]. 2025. Navštívené 2025-01-10. Dostupné z: https://support.google.com/chrome/answer/188842?hl=en&co=GENIE.Platform%3DDesktop.

[34] Hugging Face. *Open LLM Leaderboard* [online]. 2025. Navštívené 2025-05-07. Dostupné z: https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard.

[35] Hugging Face. *Transformers Documentation* [online]. 2025. Navštívené 2025-05-07. Dostupné z: https://huggingface.co/docs/transformers/index.

[36] IBM. *Large Language Models: Understanding and Leveraging AI for Business Innovation* [online]. 2025. Navštívené 2025-01-10. Dostupné z: https://www.ibm.com/think/topics/large-language-models.

[37] Kocher, P.; Horn, J.; Fogh, A.; Genkin, D.; Gruss, D. et al. Spectre Attacks: Exploiting Speculative Execution. In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, s. 1–19.

[38] Microsoft. *Phi-2* [online]. 2025. Navštívené 2025-05-07. Dostupné z: https://huggingface.co/microsoft/phi-2.

[39] Mozilla Developers. *Web Storage API* [online]. 2024. Navštívené 2024-11-13. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API.

[40] OpenAI community. *GPT-2* [online]. 2025. Navštívené 2025-05-07. Dostupné z: https://huggingface.co/openai-community/gpt2.

[41] OWASP Foundation. *Input Validation Cheat Sheet* [online]. 2024. Navštívené 2024-12-30. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html.

[42] Privacy Patterns. *Enable/Disable Functions* [online]. 2024. Navštívené 2024-12-31. Dostupné z: https://privacypatterns.org/patterns/Enable-Disable-Functions.

[43] Privacy Patterns. *Onion routing* [online]. 2024. Navštívené 2024-12-31. Dostupné z: https://privacypatterns.org/patterns/Onion-routing.

[44] Privacy Patterns. *User Data Confinement Pattern* [online]. 2024. Navštívené 2024-12-31. Dostupné z: https://privacypatterns.org/patterns/User-data-confinement-pattern.

[45] Rest, J.; Boonstra, D.; Everts, M.; Rijn, M. van a Paassen, R. Designing Privacy-by-Design. In:. Január 2014, s. 55–72. ISBN 978-3-642-54068-4.

[46] Spiekermann, S. The Challenges of Privacy by Design. *Communications of The ACM - CACM*, Júl 2012, zv. 55, s. 38–40.

[47] Team, G.; Anil, R.; Borgeaud, S.; Alayrac, J.-B.; Yu, J. et al. *Gemini: A Family of Highly Capable Multimodal Models*. 2024. Dostupné z: https://arxiv.org/abs/2312.11805.

[48] TinyLlama. *TinyLlama-1.1B* [online]. 2025. Navštívené 2025-05-07. Dostupné z: https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0.