

Universität Leipzig

Fakultät für Mathematik und Informatik
Institut für Informatik



– Bachelorarbeit –

PROGRAMMIERUNG EINER BROWSER-EXTENSION ZUR ANZEIGE VON DATENSCHUTZINFORMATIONEN IM PLAYSTORE SOWIE EVALUATION DER EXTENSION-PERFORMANCE

Author

Alexander Prull

ap62puny@studserv.uni-leipzig.de

Institut für Informatik

First Supervisor

Prof. Nummer 1

ggg@informatik.uni-
leipzig.de

Fancy Computer Science

Second Supervisor

Prof. Nummer 2

ttt@uni-leipzig.de

Institute of Rocket Science

External Supervisor

Extern Nummer 1

rrr.eee@uuu.com

Something AG

6. März 2019

Abstract

In dieser Arbeit wird sich mit den Eigenheiten der Browser Extension Programmierung auseinander gesetzt. Speziell geht es um Extension die Webseiten um bestimmte Informationen erweitern. Diese werden von einem Backend empfangen und zur Ladezeit der Seite eingespeist. Dabei setzt sich die Arbeit mit zwei Punkten auseinander. In erster Linie geht es darum die Ladezeit der Webseite durch das Anfordern von Informationen so wenig wie möglich zu beeinflussen. Also die Performance der Extension zu maximieren. Auf der anderen Seite wird durch die Nutzung der Extension von einer steigenden Nutzerzahl der Backend-Server mit einer steigenden Anzahl von Anfragen belastet. Um diese Probleme zu lösen werden in der Arbeit verschiedene Möglichkeiten zur Speicherung von Daten betrachtet und eine Auswahl der Methoden auf ihre Performance hin getestet.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Aufgabenstellung	1
1.2	Aufbau der Arbeit	2
2	Vorarbeit	3
2.1	Recherche zu Browser-Extensions	3
2.1.1	Extension-Programmierung allgemein	3
2.1.2	Existieren bereits vergleichbare Extensions	3
2.1.3	Vergleich führender Browser als Plattform für die Extension	4
2.2	PrivacyGuard	5
2.2.1	Vorstellung	5
2.2.2	API-Anbindung für die Extension	6
2.3	Implementierung einer Google Chrome Extension	6
2.3.1	Eigenschaften	6
2.3.2	Funktionsumfang und Richtlinien	7
2.3.3	Darstellung im Browser	8
2.4	Caching-Methoden	9
2.4.1	Anforderungen	9
2.4.2	Caching-Methoden und deren Eigenschaften	10
3	Hauptteil	15
3.1	Erläuterung der Aufgabenstellungen	15
3.2	Aufgabe 1: Implementierung einer Browser-Extension zur Anzeige von Datenschutzinformationen im PlayStore	15
3.2.1	Anwendungsszenario	15
3.2.2	Anforderungsanalyse	16
3.2.2.1	Funktionale Anforderungen	16
3.2.2.2	Nichtfunktionale Anforderungen	16
3.2.3	Aufbau der Website	18
3.2.4	Programmaufbau	18
3.2.5	Ergebnis	28
3.2.6	Diskussion	28

3.3	Aufgabe 2: Evaluierung von Caching Methoden einer Browser	
	Extension	28
3.3.1	Anforderungen	28
3.3.2	Vorauswahl	30
3.3.3	Rahmenbedingungen	31
3.3.4	Vorgehensweise	32
3.3.5	Ergebnisse	32
3.3.6	Diskussion	32
4	Abschließende Diskussion	33
4.1	Konklusion	33
4.2	Fortsetzung der Forschung	33
5	Appendix	35
5.1	Derivations	35
5.1.1	Example Matlab Code	36

Kapitel 1

Einleitung

Im Zeitalter der Digitalisierung werden große Mengen Informationen immer schneller und detaillierter verarbeitet. -?- Jeder, der im Internet unterwegs ist, hinterlässt dabei wertvolle, persönliche Daten. Dabei spielt der Datenschutz eine wichtige Rolle, denn nicht immer werden diese Daten freiwillig preisgegeben. So reguliert die Datenschutzerklärung, welche Nutzerdaten verarbeitet werden. Denn in dieser Erklärung muss jeder Dienstanbieter dem Nutzer zu Beginn des Nutzungsvorgangs über Art, Umfang und Zwecke der Erhebung und Verwendung personenbezogener Daten sowie über die Verarbeitung seiner Daten in Staaten außerhalb des Anwendungsbereichs [...] in allgemein verständlicher Form zu unterrichten”(Telemediengesetz Paragraph 13 1 1)

Das Projekt Privacy Guard hat sich damit beschäftigt, inwiefern diese Datenschutzerklärungen den Vorgaben entsprechen und im Speziellen analysiert, welche Applikationen unvollständige oder mangelhafte DSEs vorweisen. Im Rahmen dieses Projektes entstand die Idee, bereits vor der Installation von Anwendungen, deren Datenschutzerklärungen zu untersuchen und den Nutzer auf mögliche Bedenken hinzuweisen.

1.1 Aufgabenstellung

Die Arbeit befasst sich mit den folgenden Aufgaben:

1. **”Programmierung einer Browser-Extension zur Anzeige von Datenschutzinformationen im PlayStore”**
2. **”Evaluierung von Caching Methoden einer Browser Extension”**

Hauptaugenmerk ist die Erläuterung von Browser-Extensions, Umsetzung eines Beispiels und Limitationen. Welche Arten von Speicher stehen einer Extension zur Verfügung und welche Performance-Ersparnisse kann durch Abspeichern von

Daten die die Extension wiederholt benötigt eingespart werden. Welche Entlastung erfährt der Server mit Backend. Aufbau und Einbindung des ausgewählten Kandidaten

1.2 Aufbau der Arbeit

Zu Beginn werden Recherche Ergebnisse vorgestellt und ausgewertet. Aus den dadurch gewonnenen Resultaten die Aufgaben genauer Definiert. Auf Basis der Recherche entsteht im 1. Teil eine Extension wobei der Fokus darauf liegt, dass diese möglichst übersichtlich bleibt und zur Evaluierung von Speichermethoden dient. Anschließend werden verschiedene Testläufe präsentiert bei denen bestimmte Methoden zur lokalen Speicherung von Daten unter den gleichen Rahmenbedingungen verwendet werden. Die Ergebnisse werden verglichen und den Erwartungen gegenübergestellt. Zuletzt wird ein Fazit gezogen.

Kapitel 2

Vorarbeit

2.1 Recherche zu Browser-Extensions

2.1.1 Extension-Programmierung allgemein

Unter einer Extension versteht man ein Programm, welches den Browser um neue Funktionen ergänzt. Durch eigene Oberflächen oder Manipulation der Website erleichtern diese Erweiterungen das Nutzen des Browser.

Im Gegensatz zu Plug-Ins haben Extensions Zugriff auf Browser-spezifische Funktionen und sind in der Lage über die Webseite hinaus zu agieren. Plug-Ins werden direkt in eine Webseite eingebettet und sind auf diese beschränkt. Der Oberbegriff „Add-on“ wird heutzutage hauptsächlich als Synonym für Extension verwendet.

Jeder größere Browser stellt eine Plattform zur Verfügung auf denen Extensions angeboten und installiert werden können. In der Regel sind diese kostenlos. Wird eine Applikation nicht auf der Plattform angeboten oder dient sie zu Entwicklungszwecken, kann diese auch manuell aus externen Quellen installiert werden.

Extensions werden in HTML, JavaScript und CSS implementiert. Dabei können alle Bibliotheken verwendet werden, welche den Browserstandards für Extensions entsprechen. Kapitel 2.3.2 befasst sich genauer damit, welche Bedingungen für diese Bibliotheken in Google Chrome gelten.

Bekannte Beispiele sind Werbeblocker wie UBlock Origin und VPN-Anwendungen wie Hola.

2.1.2 Existieren bereits vergleichbare Extensions

Gesucht wurde nach einer Extension die auf der Play Store Seite den Nutzer datenschutzrelevante Informationen zu den angebotenen Apps liefert, eine Datenschutzwertung im Playstore vergibt oder den Nutzer Apps nach Berechtigungen

die Apps vorschlägt. Extensions werden nach ihrer Kurzbeschreibung in den Suchergebnissen überprüft und bei nicht eindeutig Aufgabenbeschreibung die Infoseite aufgerufen (Bsp. Safe.ad im Web Store „ecosystem“). Nur deutsche und englische Ergebnisse werden berücksichtigt.

Die Recherche hat ergeben, dass unter den genannten Suchkriterien keine Chrome oder Firefox Extension gefunden wurde die Aufgabenbereich der geplanten Extension abdeckt. Einige aufgeführte Beispiele implementieren einen Teil der geplanten Funktion (Umsortierung, Tracker checken), aber keine Extension erfüllt alle gewünschten Aufgaben.

2.1.3 Vergleich führender Browser als Plattform für die Extension

Die getroffene Auswahl des Browsers als Plattform für die Entwicklung der Extension basiert hauptsächlich auf den aktuellen Marktanteilen. Google Chrome führt mit ca. 71%, gefolgt von Mozilla Firefox mit 9,5%, Microsoft Internet Explorer mit ca. 5,7%, Apple Safari mit ca. 5%, Microsoft Edge mit 4,4% und Opera mit ca. 2,4%.

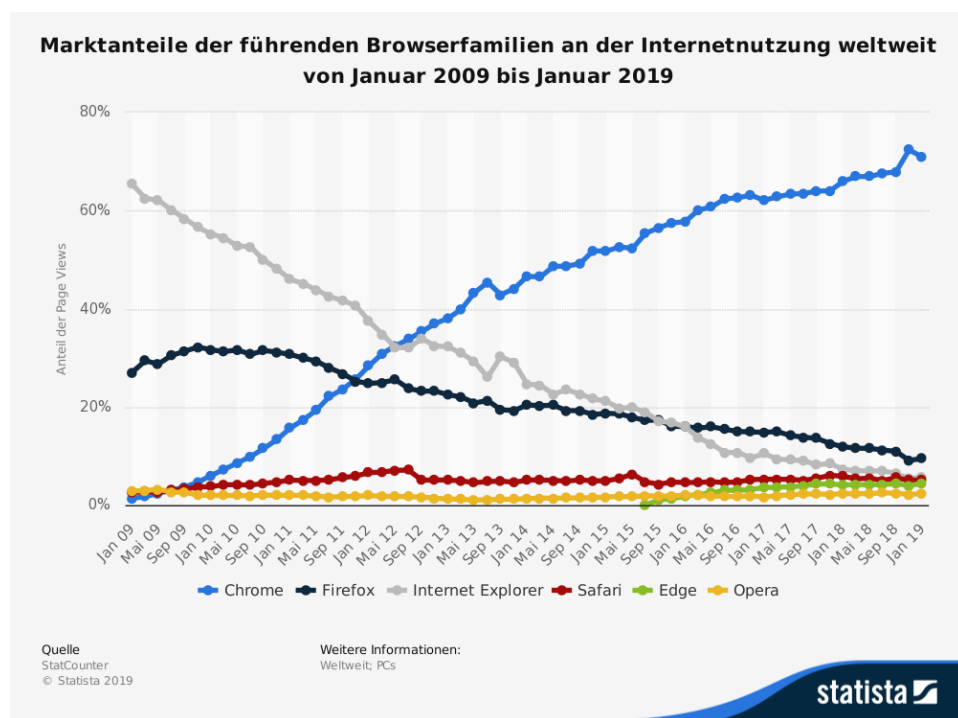


Abbildung 2.1: StatCounter. n.d. Marktanteile der führenden Browserfamilien an der Internetnutzung weltweit von Januar 2009 bis Januar 2019. Statista. Zugriff am 4. März 2019. Verfügbar unter <https://de.statista.com/statistik/daten/studie/157944/umfrage/marktanteile-der-browser-bei-der-internetnutzung-weltweit-seit-2009/>.

Aufgrund mangelnder Relevanz der Extension für Safari-Nutzer, sowie der Obso-

leszenz des Internet Explorers, wurden diese Browser nicht weiter berücksichtigt.

Google Chrome ist den verbleibenden Alternativen Mozilla Firefox, Microsoft Edge und Opera im Punkt Marktanteile weit voraus und somit die gewählte Plattform zur Entwicklung der Extension.

Unabhängig der Implementierung bieten sowohl Mozilla¹, als auch Edge² eine intuitive Lösung zur Portierung der fertigen Google Chrome-Extension.

2.2 PrivacyGuard

2.2.1 Vorstellung

Das Forschungsprojekt Privacy Guard³ wurde im Januar 2016 durch das Bundesministerium für Bildung und Forschung ins Leben gerufen. Das Institut für Angewandte Informatik⁴, die mediaTest digital GmbH⁵, die Quadriga Hochschule Berlin⁶ und die selbstregulierung informationswirtschaft e.V.⁷ haben gemeinsam Möglichkeiten entwickelt, Verbraucher auf die Verarbeitung ihrer Daten durch Handy-Applikationen aufmerksam zu machen. Dabei werden Vor- und Nachteile einzelner Aspekte der Datenverarbeitung erläutert und, bei Bedarf, Gegenmaßnahmen empfohlen.

„Ziel [...] ist die Erleichterung des Selbstdatenschutzes für Verbraucher auf mobilen Endgeräten.“ (<https://datenschutz-scanner.de/das-projekt.html>, Stand 4.3.2019)

Die Ergebnisse des Projekts teilen sich in 3 Kategorien ein. Im Rahmen der Datenbeschaffung entstanden verschiedene Werkzeuge um Informationen über Apps zu extrahieren. Besonderer Wert wurde hier auf verlinkte Datenschutzerklärungen im Playstore und in der später installierten App auf dem Handy gelegt. Desweiteren wurden die Datenschutzerklärungen mittels eines entwickelten Pre-Tagging Tools, aber auch manuell annotiert, um die Verarbeitung der Texte zu optimieren.

Zur Datenverarbeitung entstand ein Backend⁸, welches auf Anfrage der Bundle-ID einer App alle analysierten Daten übergibt. Dieses Backend bildet die Grundlage für die Datenvisualisierung von PGuard und ist die Schnittstelle der Informationen für die Browser-Extension.

¹https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Porting_a_Google_Chrome_extension

²<https://docs.microsoft.com/en-us/microsoft-edge/extensions/guides/porting-chrome-extensions>

³<https://datenschutz-scanner.de/das-projekt.html>

⁴<https://infai.org/>

⁵<https://appvisory.com/company>

⁶<https://www.quadriga-hochschule.com/>

⁷<https://sriw.de/>

⁸<https://pgadmin.datenschutz-scanner.de/api/docs.html>

Mit dem Projektabschluss im Juni 2018 stellte PrivacyGuard eine Webseite zur Analyse von Datenschutzerklärungen und Apps zur Verfügung⁹. Als Prototypen entstanden zusätzlich eine App zur Analyse aller auf dem Handy installierten Applikationen auf ihren Datenschutz und die in dieser Arbeit behandelte Browser-Extension zur Visualisierung der, durch das Projekt gewonnenen, Informationen über Apps im PlayStore.

2.2.2 API-Anbindung für die Extension

Sämtliche, von der Extension visualisierte, Daten werden über die Schnittstelle des Backends angefragt. Dazu benötigt die API mindestens eine Bundle-ID der App, Priorität und die Ausführlichkeit der Antwort.

Je höher die Anfrage priorisiert ist, um so eher wird sie vom Backend verarbeitet. Bei einer hohen Ausführlichkeit umfassen die angeforderten Informationen komplette Datenschutzerklärungen und alle Metainformationen zur Datenbeschaffung. Dagegen beinhaltet eine Antwort mit niedriger Ausführlichkeit lediglich Sprache, Quelle und Extraktionsdatum der Datenschutzerklärung, sowie die Nummer der geltenden Infofelder mit jeweiligen Textpassagen aus der Datenschutzerklärung.

Für den Anwendungsfall der Browser-Extension besteht eine hohe Priorität um Wartezeiten möglichst gering zu halten. Dagegen reicht eine niedrige Ausführlichkeit zur Darstellung der nötigen Informationen für den Verbraucher. Zusätzlich bleibt so der Datenverkehr der Extension eher gering.

Die Infofelder sind der Hauptinformationsträger der Schnittstelle. Sie sind in 31 Eigenschaften unterteilt und können eine sogenannten rote Linie sein. Das bedeutet, dass bei Besitz dieser Eigenschaft, die entsprechende App potentiell gegen ein Gesetz verstößt. Folgenden Infofelder sind im Rahmen des PrivacyGuard Forschungsprojekts zur Beurteilung von Apps entstanden (siehe Abbildung 2.1 und Abbildung 2.2).

2.3 Implementierung einer Google Chrome Extension

2.3.1 Eigenschaften

Die Architektur einer Google Chrome Extension stellt ein Paket aus mehreren Dateien dar und ist vergleichbar mit anderen Web-Technologien wie zum Beispiel Webseiten.

Grundvoraussetzung für eine funktionierende Extension ist die `manifest.json`, welche nötigen Informationen für den Browser bereitstellt und festlegt mit welchen Dateien und Rechten die Extension aufgebaut ist. Hinzu kommt mindestens eine HTML-Datei zur Darstellung der Inhalte und mindestens ein Skript zur Umsetzung der Funktionalität. Erweitert werden diese oft durch CSS-Dateien.

⁹<https://dseanalyser.pguard-tools.de/>

Externe Bibliotheken wie JQuery können ebenfalls eingebunden werden, müssen aber aufgrund der Policies¹⁰ von Google Chrome vollumfänglich lokal vorliegen. Mehr dazu Im nächsten Abschnitt.

Die Manifest-Datei ist im JSON-Format aufgebaut und beinhaltet sämtliche Informationen über die Extension. Wichtige Punkte sind Name der Extension, Beschreibung, Rechte und Aufbau. Unter Rechten oder „permissions“ werden alle APIs aufgelistet, welche die Extension benötigt um ordnungsgemäß zu funktionieren. Bevor ein Nutzer später die Extension installiert, muss er diesen „permissions“ zustimmen.

Der Aufbau wird im Punkt „content scripts“ in drei Eigenschaften unterteilt: unter welchem URL sind die Skripte aktiv, welche Skripte sind dort aktiv und welche CSS-Dateien werden dort von der Extension eingesetzt.

HTML-Dateien werden als „User-Interface Elemente“ zusammengefasst und beinhalten im Normalfall eine popup.html zur Darstellung des Fensters der Extension in der oberen rechten Ecke des Browser-Fensters (BILD?). Je nach Funktionsumfang können weitere UI-Elemente eingebunden sein, um zum Beispiel die besuchte Webseite zu erweitern.

Die vorhandenen Skripte werden normalerweise in zwei Kategorien eingeteilt. Das sogenannte „Background-Skript“ dient als Event-Handler und kommuniziert zwischen Extension und Browser. Alle restlichen Skripte sind „Content-Skripte“. Sie beinhalten die eigentliche Funktionalität der Extension.

2.3.2 Funktionsumfang und Richtlinien

Google setzt verschiedene Qualitätsansprüche an die Entwicklung einer Extension. Den Leitfaden bildet dabei das sogenannten „single purpose“-Prinzip¹¹. Das heißt, jede Anwendung muss auf sich entweder auf ein bestimmtes Thema fokussieren, wie zum Beispiel Datenschutzerklärungen und darf zu diesem Thema verschiedene Funktionen anbieten. Oder die Extension konzentriert sich auf eine bestimmte Funktion des Browser, wie zum Beispiel die Startseite und darf dort Inhalte zu mehreren Themen implementieren.

Ein weiterer Aspekt, der mit dem „single purpose“-Prinzip zusammenhängt, ist die Reichweite der Extension. Google unterscheidet hier zwischen „page-action“ und „browser-action“.

„page-action“ bedeutet, dass das Icon der Extension lediglich auf einer bestimmten Seite aktiv ist und soll den Nutzer darauf hinweisen, wo sich die Erweiterung einschaltet. Icons mit „browser-action“ sind permanent aktiv und zeigen somit, dass die Extension seitenübergreifende Funktionen hat.

¹⁰https://developer.chrome.com/webstore/program_policies

¹¹<https://developer.chrome.com/extensions/single-purpose>

Auch inhaltlich gibt es bestimmte Richtlinien¹² zu beachten. Die „Content Policies“ untersagen die Einbindung von Material mit sexuell expliziten Inhalten, Gewaltdarstellungen, „Hate Speech“, Identitätsbetrug, Urheberrechtsverletzung, Schadsoftware, Glücksspiel, Spam oder anderen illegalen Aktivitäten.

Ist Werbung ein Bestandteil der Erweiterung, unterliegen diese Inhalte ebenfalls strengen Auflagen. Der Nutzer muss in der Lage sein, die Werbung ohne Probleme zu entfernen, zum Beispiel durch Deinstallation der Extension. Werbeanzeigen dürfen keine Programmfunktionen versperren, imitieren oder andere schädliche Absichten verfolgen. Sind Anzeigen von externen Webseiten geschaltet, muss das klar ausgewiesen sein.

Erhebt, speichert oder verarbeitet die Extension sensitive Nutzerdaten, muss der Entwickler das entsprechend deklarieren und trägt Verantwortung für die Sicherheit der Daten. In diesem Zusammenhang benötigt die Anwendung eine eigene Datenschutzerklärung.

2.3.3 Darstellung im Browser

Die Extension wird an mehreren Stellen im Browser integriert. Dabei besitzt jedes installierte Programm ein Icon in der Adresszeile des Browsers (Abbildung 2.2). Dieses Element dient als Steuerelement für Funktionen der Erweiterung, wie zum Beispiel das Aktivieren und Deaktivieren der Extension. Ein ausgegrautes Icon bedeutet dabei, dass die Extension inaktiv ist, bzw. dass die zugehörige Webseite nicht in diesem Tab geöffnet ist.

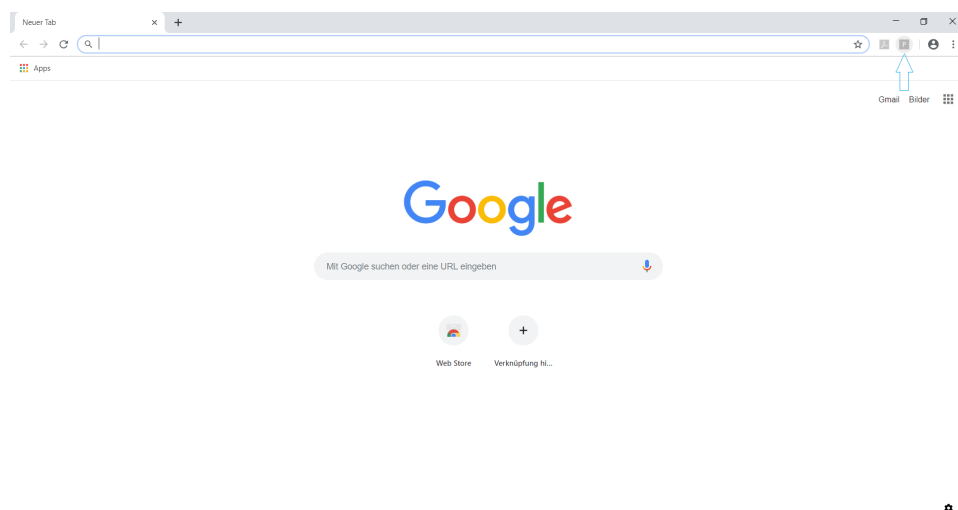


Abbildung 2.2: Browser-Extension Icon in der Adresszeile

¹²https://developer.chrome.com/webstore/program_policies

Eine weitere Möglichkeit Inhalte und Funktionen der Erweiterung zu verwalten, ohne dafür Template des Icons zu verwenden, ist die dedizierte Optionenseite¹³. Diese kommt vor allem bei umfangreicheren Browser-Extensions vor.

Alle restlichen Elemente werden direkt in die angezeigte Webseite integriert und erweitern die Anzeige beliebig nach Funktion der Extension.(REF)

2.4 Caching-Methoden

2.4.1 Anforderungen

Diese Arbeit beschäftigt sich damit, welche Caching-Methoden für Browser-Extensions geeignet sind. Der Fokus liegt hierbei auf den in Aufgabe 1.1 beschriebenen Anwendungsfällen. Daraus ergeben sich folgende Anforderungen an den Speicher:

Performanz spielt eine große Rolle, da dem Nutzer alle gespeicherten Daten der Extension zeitgleich zum Abschluss der Ladezeit der Webseite zur Verfügung stehen sollen. Werden Daten mit Verzögerung auf der Webseite dargestellt oder verlängert der Prozess sogar die initiale Ladezeit, hat das eine beträchtliche Auswirkung auf die Nutzererfahrung. So zeigt eine Studie von Google Research auf dem Jahr 2017¹⁴, dass die Absprungrate sich bei Ladezeiten von über zwei Sekunden stark erhöht.

Aufgrund der Schnelllebigkeit der Informationen, müssen diese nicht über einen längeren Zeitraum abgespeichert werden. Außerdem verarbeitet die Browser-Extension keine Daten weiter. Verlorene Einträge können so durch eine erneute Anfrage verlustfrei ersetzt werden.

Lastverteilung ist eine der Hauptanforderungen an die Caching-Methoden in dieser Arbeit. Da alle Daten über ein zentrales Backend abgerufen werden, besteht die Gefahr des Flaschenhalseffekts. Zudem sollen allgemein möglichst wenig Anfragen an externe Quellen gestellt werden, um so das Risiko zu vermeiden, auf Antworten von Server zu warten.

Alle Daten, die im Cache abgelegt werden sind durch PrivacyGuard gewonnene Informationen über die einzelnen Applikationen und beziehen sich nicht auf den Nutzer der Extension. Es werden also keine sensiblen Daten im Cache abgelegt. Somit besteht kein Bedarf an Sicherheitsmaßnahmen wie etwa die Verschlüsselung der Daten.

Die Datenvisualisierung der Extension ist überschaubar und auch die zu Grunde liegende Datenstruktur der API ist einfach aufgebaut. Dadurch kann auf komplexe Datenbankstrukturen oder zusätzliche Frameworks verzichtet werden.

¹³<https://developer.chrome.com/extensions/options>

¹⁴<https://www.thinkwithgoogle.com/marketing-resources/data-measurement/mobile-page-speed-new-industry-benchmarks/>

2.4.2 Caching-Methoden und deren Eigenschaften

Um zu evaluieren, welche Caching-Methode am besten für den beschriebenen Anwendungsfall geeignet ist, befasst sich dieses Kapitel mit der Auflistung aller möglichen Speicherfunktionen für Browser-Extensions. Die dazu aufgelisteten Eigenschaften dienen als Vergleich für die spätere Vorauswahl der Methoden. Als Quelle für Teile der Recherche diente das Buch von Raymond Camden „Client-Side Data Storgae“

Alle Daten der Extension sollen im Browser gespeichert werden. Datenbanken auf externen Servern bieten keinen Mehrwert, da die Informationen vom Backend nicht noch verarbeitet, personalisiert, oder gesichert werden müssen. In diesem Fall dient das Backend als Datenbank die durch lokale Speicherprozesse entlastet werden soll.

Cookies sind kleine Textspeicher die normalerweise von Webseiten in Browsern genutzt werden, um Informationen über den Besucher zu speichern. Sie besitzen meist eine kurze Lebensdauer und werden über HTTP-Header übertragen. Auch Browser-Extensions können Cookies nutzen¹⁵, um Informationen zu speichern. Das Buch verweist darauf, dass man bis zu 50 Cookies mit einer Gesamtgröße von 4KB pro Domäne ohne bedenken nutzen kann. Jedoch genießen Cookies, unter anderem aufgrund der EU-Richtlinie¹⁶, keinen guten Ruf. Webseiten müssen sich in der Regel vor Gebrauch die Genehmigung des Nutzers einholen. Dies könnte also auch bei Extension zu rechtlichen Problemen führen. Außerdem blockieren viele Nutzer diese Cookies eben aus datenschutzrechtlichen Gründen.

Die Web Storage API¹⁷ ist eine einheitlich Methode von mehreren Browser, Daten lokal zu speichern. Pro Domäne verfügt der Web Storage über 5-10MB, je nach Browser. Dabei besteht jede Einheit aus einem key-value-Paar, welches Daten in Form von Strings speichern kann. Die Web Storage API ist unterteilt in „local storage“ und „session storage“. Während beim session storage alle Daten nur solange gespeichert sind, bis der Browser wieder geschlossen wird, bleiben die Daten beim local storage solange bestehen, bis sie manuell überschrieben oder gelöscht werden.

Mit der IndexedDB API¹⁸ ist die Extension in der Lage eine domainspezifische Datenbank zu erstellen. Diese Datenbank besteht aus sogenannten „Object Stores“, welche Tabellen mit flexiblen Datentypen beinhalten können. Die Speicherkapazität wird nicht exakt definiert und ist abhängig von der Größe der Festplatte¹⁹. So liegt der verfügbare Speicher zwischen 10MB und 50% des freien Festplattenspei-

¹⁵<https://developer.chrome.com/extensions/cookies>

¹⁶<https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0058:de:HTML>

¹⁷https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API

¹⁸<https://developer.mozilla.org/de/docs/IndexedDB>

¹⁹https://developer.mozilla.org/de/docs/IndexedDB/Browser_storage_limits_and_eviction_criteria

chers. IndexedDB wird nicht noch komplett unterstützt und gilt als vergleichsweise komplexe API.

Ergänzend zu erwähnen ist die SQLite basierte Datenbank Web SQL. Seit 2010 wird diese Spezifikation von dem World-Wide-Web-Konsortium allerdings nicht mehr zur Implementierung empfohlen und gilt seit jeher als veraltet.

Nr.	Bezeichnung	rote Linie
1	Ihre Zahlungsdaten werden unverschlüsselt übermittelt	Ja
2	Die App hat Internetzugriff	Nein
3	Die App benennt keine Kontaktmöglichkeit für datenschutzrechtliche Anliegen	Ja
4	Ihre Login-Daten werden unverschlüsselt übermittelt	Ja
5	Die App stellt keine Datenschutzerklärung auf Deutsch bereit	Nein
6	Die Datenschutzerklärung verwendet ungenaue Formulierungen	Nein
7	Die Verschlüsselung der App ist unsicher	Nein
8	Die App nutzt Standortdaten	Nein
9	Die Datenschutzerklärung kann geändert werden, ohne Sie hierüber zu informieren	Ja
10	Das über Sie erstellte Profil wird durch öffentliche Informationen über Sie ergänzt	Nein
11	Ihre Daten werden durch Dienstleister verarbeitet	Nein
12	Die App integriert Werbenetzwerke	Nein
13	Die App erhebt eine Vielzahl an Geräteinformationen	Nein
14	Die App stellt die Datenschutzerklärung erst nach Start der App bereit	Nein
15	Die App hat Zugriff auf Ihr Adressbuch	Nein

Tabelle 2.1: Übersicht der Infofelder von PrivacyGuard

Nr.	Bezeichnung	rote Linie
16	Die App enthält Malware	Ja
17	Die App übermittelt Daten an Dritte	Nein
18	Die App erhebt statische Gerätekennungen	Nein
19	Die App verarbeitet Daten, die für die Funktion der App nicht erforderlich sind	Nein
20	Die App verarbeitet Daten, die ausdrücklich ausgeschlossen wurden	Ja
21	Die App ermöglicht einer Vielzahl von Drittanbietern Zugriff auf Ihre Nutzungsdaten	Nein
22	Es wird ein Profil über Sie erstellt	Nein
23	Ihre Daten werden in der Unternehmensgruppe geteilt	Nein
24	Die App kann sich im Hintergrund unbemerkt aktualisieren	Nein
25	Die Herkunft der App ist unbekannt	Nein
26	Die App stellt keine Datenschutzerklärung bereit	Ja
27	Ihre Daten werden für personalisierte Werbung genutzt	Nein
28	Die App klärt nicht ordnungsgemäß über Datenverarbeitungen im Ausland auf	Nein
29	Die App stellt unterschiedliche Datenschutzerklärungen in der App und im App-Store bereit	Ja
30	Ihre Daten werden über die App veröffentlicht	Nein
31	Die Sprachsteuerung ist dauerhaft im Hintergrund aktiv	Nein

Tabelle 2.2: Übersicht der Infofelder von PrivacyGuard

Kapitel 3

Hauptteil

This chapter will first outline the problems that constitutes the main portion of this work. Each problem is described separately starting with the available data sources followed by a detailed description of the proposed solutions. After that, the proposed solutions are evaluated by empirical means and the results are presented. Performance studies are conducted to provide suitable recommendations concerning the real world application.

3.1 Erlaeuterung der Aufgabenstellungen

3.2 Aufgabe 1: Implementierung einer Browser-Extension zur Anzeige von Datenschutzinformationen im Play-Store

3.2.1 Anwendungsszenario

QUELLE STATISTIK?

Während vor einigen Jahren Applikationen hauptsächlich auf eigenen Webseiten zum Download angeboten wurden, haben sich die AppStores mittlerweile durchgesetzt. Vorteile für diese Plattformen sind unter anderem: erleichterter Zugang, Vergleiche mit anderen Applikationen und individuelle Empfehlungen. Bei der Wahl für eine bestimmte Applikation achten Nutzer auf Aspekte, wie Preis, Anzahl der Downloads und Bewertungen von anderen Nutzern.

Immer wichtiger aber auch die Frage: Welche Daten gebe ich der Applikation frei und wie werden diese verarbeitet. Der PlayStore bietet zwar einen groben Überblick, welche Daten eine Applikation von dem Handy ausliest, aber nicht wie diese vom Anbieter verarbeitet werden.

Dadurch entstehen beim Nutzer Fragen, welche der Playstore nicht beantwortet:

1. Handhabung der Daten: Wie werden die Daten verarbeitet und an wen werden diese weitergeleitet? Wird ein Profil anhand der Daten erstellt? Welche Sicherheit besteht bei der Übertragung der Daten?
2. Vor- und Nachteile der Datenverarbeitung: Kann der Anbieter die Applikation dadurch komfortabler gestalten? Wird Werbung in der Applikation personalisiert? Besteht Gefahr vor Missbrauch der Daten?
3. Kontrolle über die Daten: Welche Möglichkeiten stehen zu Verfügung im Falle von Nichteinverständnis? Ist der Umgang mit den Daten nach der Installation noch einschränkbar. Kann der Nutzer die Verwendung der Daten verbieten und trotzdem die App weiterhin nutzen?

3.2.2 Anforderungsanalyse

3.2.2.1 Funktionale Anforderungen

Aus den Fragen die bei dem Anwendungsszenario entstanden sind werden funktionale Anforderungen gebildet um konkrete Aufgaben für die Extension zu schaffen.
-Anforderungen in TEXTFORM-

-Direkt bei Apps in Anforderungen erwähnen

/F10/ Erweiterung der Informationen im PlayStore: Der Nutzer hat die Möglichkeit im Browserfenster per Aktivierung bzw. Deaktivierung der Extension zusätzliche Datenschutzinformationen zu den angezeigten Applikationen ein- bzw. auszublenzen.

/F20/ Anzahl von bedenklichen Eigenschaften einer Applikation: Zu jeder Applikation erhält der Nutzer ein Feedback von der Extension, wieviele Bedenken vorliegen.

/F30/ Darstellung von kritischen Eigenschaften einer Applikation: Eigenschaften einer Applikation, welche einen erheblichen Nachteil für den Nutzer darstellen oder einen möglichen Gesetzesverstoß beinhalten werden hervorgehoben.

/F40/ Abrufen von Details zu den Bedenken: Wird ein Bedenken angezeigt, kann der Nutzer direkt Erläuterung, Handlungsempfehlung sowie Vor- und Nachteile zu diesem Bedenken abrufen.

/F50/ Empfehlung bei Suchanfragen: Basierend auf den Bedenken einer Applikation kann der Nutzer die Suchanfrage so anpassen, dass ihm unbedenkliche Applikationen priorisiert angezeigt werden.

3.2.2.2 Nichtfunktionale Anforderungen

Das Programm richtet sich in erster Linie an Nutzer, denen keine besonderen informatischen Kenntnisse abverlangt werden Extensions zeichnen sich durch

ihre Einfachheit aus. Nutzer wissen vor der Installation, welche Funktionen diese Programme haben. Die Extension soll auf den ersten Blick klar machen, welche Komponenten des Browsers erweitert oder verändert wurden. -QUELLE- Was ist eine Nichtfunktionale Anforderung + kurze Auflistung (BUCH)

- /NF10/ Darstellung und Einbindung der Informationen Darstellung und Einbindung spielen bei Browser-Extensions eine wichtige Rolle. Hier wird keine grundlegend neue Oberfläche gestaltet sondern eine bereits vorhandene erweitert. Der Fokus fällt darauf, die bestehende Oberfläche so zu verändern, dass alle Elemente der Extension an der richtigen Stelle eingebaut werden. Der Nutzer soll auf den ersten Blick erkennen welche neuen Informationen zu welchen bereits bestehenden Teilen der Website gehören.
- /NF20/ Persistenz der Website Im Gegensatz zu NF10 darf die Website nicht so verändert werden, dass sie in ihrem Aussehen und ihren Funktionen zu stark von ihrem Originalzustand abweicht. Gerade bei Seiten auf denen viele Elemente automatisch generiert werden, verursachen kleine optische Veränderungen schon Probleme beim Aufbau der Website. Entsprechend müssen Informationen so subtil wie möglich eingebettet werden. So wird verhindert, dass der Nutzer die Extension nur aufgrund der Optik wieder deinstalliert.
- /NF30/ Handhabung In der Extension werden viele und vor allem auch umfangreiche Informationen angeboten. Diese dürfen den Nutzer nicht überwältigen. Dennoch müssen sämtliche Punkte vgl pguard informationen an der richtigen Stelle zur Verfügung stehen.
- /NF40/ Skalierbarkeit def Skalierbarkeit? Hier bezieht sich der Begriff Skalierbarkeit vor allem auf Anfragen an das Backend. Angenommen die Extension erreicht eine hohe Nutzerzahl. Dadurch steigt das Risiko auf Überlastung des Servers. Um das zu verhindern werden bei der Informationsgewinnung zwei Aspekte besonders wichtig. Zum Ersten wie aktuell die Informationen sein sollen. Je aktueller, desto öfter müssen Anfragen gesendet werden. Zum Anderen die Relevanz. Wie schnell müssen welche Informationen vorhanden sein und welche Informationen, die eine Analyse erfordern, werden erst auf spezielle Anfrage des Nutzers angefragt. Diese Anforderung stellt einen zentralen Punkt in der Entwicklung der Extension dar und wird in Aufgabe 2 detailliert behandelt.
- /NF50/ Datenschutz Bei allen Webdiensten spielt der Datenschutz eine wichtige Rolle. Auch in diesem Programm sollen Daten gespeichert werden um die Anforderung /NF40/ zu unterstützen. Um Datenschutzbedenken auszuschließen muss das Format der Daten so gewählt werden, dass diese nicht personalisiert werden und nach Möglichkeit komplett lokal gespeichert werden.
- /NF60/ Korrektheit der Daten Alle Informationen zu Applikationen die dieses Programm darstellt werden extern von einem Server des privacy guard-Projekts

eingespeist. Dieser gewinnt die Daten hauptsächlich auf automatischen Textmining-Verfahren. Ein Problem bei diesen Verfahren ist die fehlende Validierung der Informationen. Ursachen wie das heterogene Format von Datenschutzerklärungen und Mehrfachverlinkungen von Datenschutzinformationen können zu bei dieser Methode zu Fehlern oder Ungenauigkeiten führen. Aus diesem Grund muss dem Nutzer verdeutlicht werden, dass alle Angaben als Empfehlungen zu betrachten sind und keine verbindlichen Aussagen über Applikationen getätigt werden.

3.2.3 Aufbau der Website

Singlepage, Multipage, Kacheln, APP-ID

3.2.4 Programmaufbau

Die Struktur des Programms orientiert sich an der beschriebenen Architektur in Kapitel 2.3.3. Dieser Abschnitt befasst sich mit der Umsetzung der Anforderungen, welche im vorangegangenen Kapitel aufgesetzt wurden. Dabei werden wichtige Eigenschaften aus an Teilen des Quellcodes betrachtet und getroffene Entscheidungen begründet. Außerdem werden kritische Stellen beleuchtet, welche für die Diskussion und fortführende Arbeit relevant sind.

```

1 {
2   "name": "PGuard AppRating",
3   "description": "rates PlayStore Applications based on 30 data safety criteria",
4   "version": "0.1",
5   "page_action": {
6     "default_popup": "popup/popup.html"
7   },
8   "manifest_version": 2,
9   "permissions": ["storage", "declarativeContent", "activeTab", "tabs"],
10  "content_scripts": [
11    {
12      "matches": ["*://play.google.com/store/apps*"],
13      "js": ["lib/js/jquery-3.3.1.min.js", "lib/js/popper.js", "lib/js/bootstrap.min.js",
14            "lib/js/fontawesome-all.js", "pguard.js", "popup/popup-controller.js"],
15      "css": ["lib/css/bootstrap.min.css", "lib/css/multiapp.css"]
16    }
17  ],
18  "background": {
19    "scripts": ["background.js"],
20    "persistent": false
21  },
22  "web_accessible_resources": [
23    "lib/data/*.json",
24    "lib/templates/*"
25  ]
26 }
```

Listing 3.1: Aufbau der manifest.json

Das Manifest stellt die grundlegenden Zusammenhänge der Extension dar. Unter anderem den gewählten Entwicklungsnamen der Extension PGuard AppRating und die entsprechende Beschreibung. Weiterhin sind die nötigen Berechtigungen aufgeführt.

- storage: Berechtigung zum Zugriff auf Speicherplatz, um Informationen aus Backend-Anfragen zu speichern. Details dazu in Kapitel 3.3.1.

- declarativeContent: Bereitstellung von Events, wie Seitenaufruf oder -änderung und damit zusammenhängende Regeln, wie das Ausführen von Content-Skripten. Diese API wird vom Background-Skript genutzt, welches die genannten Aufgaben umsetzt.
- activeTab und tab: Gibt an, ob sich der Nutzer gerade in einem Tab befinden auf dem die Extension aktiv ist.

Außerdem werden alle Dateien ihren Rollen zugewiesen.

- Content-Skript: Unter dem Punkte content scripts wird festgelegt, welche Skripte unter welchem URL aktiv sind. Der Ausdruck `*://play.google.com/store/apps*` bedeutet, dass die Extension auf jeder Playstore-Seite der Kategorie Apps und deren Unterverzeichnis aktiv ist. Da es sich um eine page action Extension handelt, wird lediglich eine Website als matchäufgeführt. Die beiden wichtigen Dateien hier sind pguard.js als das Content Skript für sämtliche Funktionen die die Erweiterung der Website betreffen und popup-controller.js für alle Funktionen des Popups. Hinzu kommen sämtliche Bibliotheken, welche von den Content-Skripten benötigt werden.
- Background-Skript Wie bereits erwähnt fungiert das Background-Skript `background.js` als Eventhandler der Extension ist deshalb separat im Manifest aufgeführt.
- web_accessible_resources Diese Ressourcen sind Dateien welche der Extension zur Verfügung stehen aber selber keine Skripte sind. Sie beinhalten ausgelagerte Informationen wie Fließtexte und Templates zum Bauen von HTML-Elementen. Die `popup.html` ist hier ein Sonderfall und wird direkt dem Popup zugewiesen.

Die Background.js besteht lediglich aus Callback-Funktionen der declarativeContent API. Hier wird zur Installation der Extension ein Listener eingebunden. Dieser funktioniert mit Regeln nach dem Konditionen-Aktionen-Prinzip. Zum Start des Aufrufs werden alle bereits vorhandenen Regeln des Listeners entfernt und anschließend die übergebenen Regeln installiert. Hier benötigt das Programm eine Regel. Die Kondition prüft ob der passende URL aufgerufen wurde. Dieser stimmt mit dem String aus der Manifest-Datei überein. Ist die Kondition erfüllt, aktiviert sich das Popup.

```
1  /**
2   * Created by Alexander on 04.04.2018.
3   */
4  chrome.runtime.onInstalled.addListener(function() {
5      //removeRules, braucht declarativeContent permission, 1 = liste an regeln(undefined = keine
        Eingabe), 2= function nach loeschen
6      chrome.declarativeContent.onPageChanged.removeRules(undefined, function() {
7          //siehe removeRules, wird am ende von remove rules aufgerufen.
8          chrome.declarativeContent.onPageChanged.addRules([
9              {
10                 conditions: [new chrome.declarativeContent.PageStateMatcher({
11                     pathContains: "/store/apps"
12                 })],
13                 actions: [new chrome.declarativeContent.ShowPageAction()]
14             }
15         ]
16     });
17 }
```

```

15     });
16 });

```

Listing 3.2: background.js

Das Contentskript pguard.js bildet den Hauptteil der Extension und dient zur Erfassung aller, auf der Webseite dargestellten Apps, der Einbettung von zusätzlichen Informationen durch das PGuard-Backend und optischen Anpassung der Webseite, um die neuen Inhalte ordentlich einzubinden.

Bei Skript-Aufruf werden zuerst die lokalen Bibliotheken der Extension geladen. Dazu gehören die IB texte.json sowie die HTML-Templates. Außerdem überprüft das Skript, ob lokaler Speicher zur Verfügung steht. (???)

Anschließend prüft die Funktion fillApps, ob die aktuelle Seite eine Single-App-Page oder Mutli-App-Page ist und ermittelt sämtliche Kandidaten, welche für die Einbettung der Informationen in Frage kommen. Dabei liest der JQuery-Selektor alle Elemente mit dem entsprechenden Klassnamen aus.

Mit der Funktion loadInfoPanels wird jeder so gefundene Kandidat auf seine APP-ID überprüft. Diese befindet sich entweder in dem Attribut "data-docid" oder "href". Mithilfe dieser ID durchsucht die Funktion "getStorageItem" den lokalen Speicher auf Einträge. Der Eintrag ist valide, falls er nicht leer ist und vor weniger als 3 Tagen angelegt wurde. Findet die Funktion keinen validen Eintrag im lokalen Speicher, wird eine neue Anfrage an das Backend, für die entsprechende APP-ID, erstellt.

BEISPIEL ANFRAGE

Liefert das Backend eine Antwort mit mindestens einem Ergebnis, speichert die Funktion die Informationen im lokalen Speicher ab. Bei mehr als einem Ergebnis, entscheidet eine Prioritätenliste abhängig von der zuverlässigsten Quelle, welcher Datensatz genutzt wird.

Der gewählte Datensatz wird der Funktion createPanel übergeben. Handelt es sich bei dem Kandidaten um eine App-Page(Single-App), wird das Popover aus den Informationen direkt erstellt. Dazu wird der Datensatz mit Hilfe der IB texte.json in den entsprechenden Text umgewandelt und in die Html-Template eingefügt. Bei App-Kacheln(Multi-app) baut die Funktion einen Banner in die Kachel ein. Auf diesem Banner wird mittels JQuery ein Event geladen, welches bei einem Klick auf den Banner das Popover erstellt.

BEISPIEL IB texte

```

1  /**
2   * Created by Alexander on 03.05.2018.
3   */
4  var today = new Date();
5  var anfragencounter = 0;
6  var ibJson = [];
7  var useLocalStorage = true;
8  var db;
9  //TODO set Storage by toggle in popup

```

```

10 var usedStorage = "indexedDB";
11 var ibCardTemplate = document.createElement("div");
12 var innerCollapseTemplate = document.createElement("div");
13 var trennzeichen = "|";
14 var dseOrigins = ["lcm", "playstore", "link.guesser", "app", "mtd", "manuell"];
15 var route = "get.infa.dataset.by.bundle_id";
16 var token = "uni.leipzig.ba-prull.2018.01.17.Aihungaem5ie7opheeme";
17 var verbosity = "3";
18 var priority = "5";
19 var trgDsePlay = "false";
20 var trgLinkGuess = "false";
21 var trgDseLcm = "false";
22 var trgDseMtd = "false";
23 var trgDseInApp = "false";
24 // var forceExec = "&force.execution=true";
25 var urlWholeDataSetNoRequest = "https://infaibackend.pguard-tools.de/" + route + "?api.token=" + token
26   + "&verbosity=" + verbosity + "&priority=" + priority + "&trigger.dse-playstore.download=" +
27     trgDsePlay
28   + "&trigger.link.guesser=" + trgLinkGuess + "&trigger.dse.lcm.download=" + trgDseLcm
29   + "&trigger.dse.mtd.download=" + trgDseMtd + "&trigger.dse.inapp.search=" + trgDseInApp
30   + "&bundle.id=";
31 var urlTriggerNewAnalysis = "https://infaibackend.pguard-tools.de/" + route + "?api.token=" + token
32   + "&verbosity=" + "3" + "&priority=" + priority + "&trigger.dse-playstore.download=" + "true"
33   + "&trigger.link.guesser=" + "true" + "&trigger.dse.lcm.download=" + "true"
34   + "&trigger.dse.mtd.download=" + "true" + "&trigger.dse.inapp.search=" + "true" + "&bundle.id=";
35 // Funktion von Mozilla die Browser auf localStorage-Verfuegbarkeit ueberprueft.
36 function storageAvailable(type) {
37   if (type === "indexedDB" && type in window) {
38     return true;
39   }
40   try {
41     var storage = window[type],
42       x = '__storage_test__';
43     storage.setItem(x, x);
44     storage.removeItem(x);
45     return true;
46   } catch (e) {
47     return e instanceof DOMException && (
48       // everything except Firefox
49       e.code === 22 ||
50       // Firefox
51       e.code === 1014 ||
52       // test name field too, because code might not be present
53       // everything except Firefox
54       e.name === 'QuotaExceededError' ||
55       // Firefox
56       e.name === 'NS_ERROR_DOM_QUOTA_REACHED') &&
57       // acknowledge QuotaExceededError only if there's something already stored
58       storage.length !== 0;
59   }
60 }
61
62 function getStorageItem(itemKey) {
63   var appInfo;
64
65   switch (usedStorage) {
66     case "none":
67       break;
68     case "localStorage":
69       appInfo = localStorage.getItem(itemKey);
70       break;
71     case "indexedDB":
72       var transaction = db.transaction(["apps"], "readonly");
73       var store = transaction.objectStore("apps");
74       appInfo = store.get(itemKey);
75       break;
76
77     default:
78       console.log("No storage method selected.");
79   }
80   return appInfo;
81 }
82
83 function setStorageItem(itemKey, itemValue) {
84   switch (usedStorage) {
85     case "none":
86       break;
87     case "localStorage":
88       localStorage.setItem(itemKey, itemValue);
89       break;
90     case "indexedDB":
91       var transaction = db.transaction(["apps"], "readwrite");

```

```

92         var store = transaction.objectStore("apps");
93         var entry = {
94             appID: itemKey,
95             data: itemValue
96         };
97         var request = store.put(entry);
98         request.onerror = function (ev) {
99             console.log("Error", ev.target.error);
100             console.dir(ev.target);
101         };
102         request.onsuccess = function (ev) {
103             console.log("IndexedDB funkt");
104         };
105         break;
106     default:
107         console.log("No storage method selected.");
108     }
109 }
110
111 function deleteStorageItem(itemKey) {
112     switch (usedStorage){
113         case "none":
114             break;
115         case "localStorage":
116             localStorage.removeItem(itemKey);
117             break;
118         case "indexedDB":
119             break;
120         default:
121             console.log("No storage method selected.");
122     }
123 }
124
125 // Laedt neugewonnene Daten in den localStorage des Browsers
126 function castDseToStorageString(dse){
127     var extractionDate;
128     var ibString = "";
129     var freqCount = "1";
130     var ibs = [];
131     // Auf Tage runden.
132     extractionDate = Math.floor(today.getTime() / 86400000);
133
134     console.log("Wachle: ", dse.origin, dse.date_infobox_calculation_finished);
135     for(var o = 0; o < dse.infoboxes.length; o++){
136         // 20 und 28 werden ignoriert
137         if(dse.infoboxes[o].id !== 28 && dse.infoboxes[o].id !== 20){
138             // 13 und 21 setzten mehrere Elemente bei den jeweiligen Eigenschaften voraus
139             // TODO aktuell werden 13 und 21 ignoriert, da bei verbosity 3 die Information fehlt.
140             if((dse.infoboxes[o].id !== 13 && dse.infoboxes[o].id !== 21)){
141                 ibs.push("" + dse.infoboxes[o].id);
142             }
143         }
144     }
145
146     for(var i = 0; i < ibs.length; i++){
147         if(i === (ibs.length - 1)){
148             ibString += ibs[i];
149         } else {
150             ibString += ibs[i] + trennzeichen;
151         }
152     }
153
154     return "" + extractionDate + trennzeichen + freqCount + trennzeichen + ibString;
155 }
156
157 function getInfoForIB(attr, ib){
158     for(var i = 0; i < ibJson.length; i++){
159         if(ib === ibJson[i].id){
160             return ibJson[i][attr];
161         }
162     }
163     return "";
164 }
165
166 function createPopover(parentElement, ibArray){
167     var eleToRemove;
168     $(parentElement).attr({"data-toggle": "popover", "data-trigger": "focus"});
169     var cardContainer = document.createElement("div");
170     cardContainer.id = "accordion";
171     for(var i = 0; i < ibArray.length; i++){
172         // Template
173         var cardTemplate = ibCardTemplate.cloneNode(true);
174
175         // Red-Line Markierung
176         for(j = 0; j < ibJson.length; j++){

```

3.2. AUFGABE 1: IMPLEMENTIERUNG EINER BROWSER-EXTENSION ZUR ANZEIGE VON DATENSCHUTZINFORMATIONEN IM PLAYSTORE

23

```
176         if (ibArray[i] === ibJson[j].id && ibJson[j]["is_red_line"] === "true"){
177             var cardHeader = cardTemplate.getElementsByClassName("card-header")[0];
178             cardHeader.style.backgroundColor = "#ff8c8c";
179             break;
180         }
181     }
182
183     // Titel
184     var titel = cardTemplate.getElementsByClassName("ibtitel")[0];
185     titel.innerText = getInfoForIB("titel", ibArray[i]);
186
187
188     // Einzigartige collapse-id
189     $(titel).attr("href", "#collapse" + i);
190     $(titel.parentNode.parentNode.children[1]).attr("id", "collapse" + i);
191
192     // Beschreibung
193     if (getInfoForIB("description", ibArray[i]) !== "") {
194         eleToRemove = cardTemplate.getElementsByClassName("thead")[0];
195         eleToRemove.parentNode.removeChild(eleToRemove);
196         eleToRemove = cardTemplate.getElementsByClassName("description")[0];
197         eleToRemove.parentNode.removeChild(eleToRemove);
198     } else {
199         cardTemplate.getElementsByClassName("description")[0].children[0].innerText =
200         getInfoForIB("description", ibArray[i]);
201     }
202
203     // Pro
204     if (getInfoForIB("pros", ibArray[i]).length > 0) {
205         var pros = getInfoForIB("pros", ibArray[i]);
206         var proElement = cardTemplate.getElementsByClassName("pro")[0];
207         for (var j = 0; j < pros.length; j++) {
208             var row = document.createElement("tr");
209             var tData = document.createElement("td");
210             if (pros[j]["second_layer"] !== "") {
211                 tData.appendChild(innerCollapseTemplate.cloneNode(true));
212                 var firstLayerElement = tData.getElementsByClassName("firstlayer")[0];
213                 $(firstLayerElement).attr({"href": ("collapseSecondPro" + j + "_" + i)});
214                 firstLayerElement.innerText = pros[j]["first_layer"];
215
216                 $(firstLayerElement.parentNode.parentNode.parentNode).attr("id", "accordionSecondPro" + j +
217                 "_" + i);
218                 var secondLayerElement = firstLayerElement.parentNode.nextElementSibling;
219                 $(secondLayerElement).attr({
220                     "data-parent": ("accordionSecondPro" + j + "_" + i),
221                     "id": ("collapseSecondPro" + j + "_" + i)
222                 });
223                 secondLayerElement.children[0].innerText = pros[j]["second_layer"];
224             } else {
225                 tData.innerText = pros[j]["first_layer"];
226             }
227             row.appendChild(tData);
228             // $(row).insertAfter(proElement.children[proElement.children.length - 1]);
229             $(row).insertAfter(proElement);
230         }
231     } else {
232         eleToRemove = cardTemplate.getElementsByClassName("pro")[0];
233         eleToRemove.parentNode.removeChild(eleToRemove);
234     }
235
236     // Contra
237     if (getInfoForIB("cons", ibArray[i]).length > 0) {
238         var cons = getInfoForIB("cons", ibArray[i]);
239         var contraElement = cardTemplate.getElementsByClassName("contra")[0];
240         for (j = 0; j < cons.length; j++) {
241             var rowCon = document.createElement("tr");
242             var tDataCon = document.createElement("td");
243             if (cons[j]["second_layer"] !== "") {
244                 tDataCon.appendChild(innerCollapseTemplate.cloneNode(true));
245                 firstLayerElement = tDataCon.getElementsByClassName("firstlayer")[0];
246                 $(firstLayerElement).attr({"href": ("collapseSecondCon" + j + "_" + i)});
247                 firstLayerElement.innerText = cons[j]["first_layer"];
248
249                 $(firstLayerElement.parentNode.parentNode.parentNode).attr("id", "accordionSecondCon" + j +
250                 "_" + i);
251                 secondLayerElement = firstLayerElement.parentNode.nextElementSibling;
252                 $(secondLayerElement).attr({
253                     "data-parent": ("accordionSecondCon" + j + "_" + i),
254                     "id": ("collapseSecondCon" + j + "_" + i)
255                 });
256             }
```

```

255         secondLayerElement.children[0].innerText = cons[j]["second_layer"];
256     } else {
257         tDataCon.innerText = cons[j]["first_layer"];
258     }
259 }
260
261 rowCon.appendChild(tDataCon);
262 $(rowCon).insertAfter(contraElement);
263 }
264
265 } else {
266     eleToRemove = cardTemplate.getElementsByClassName("contra")[0];
267     eleToRemove.parentNode.removeChild(eleToRemove);
268 }
269
270 // Empfehlung
271 if (getInfoForIB("recommendations", ibArray[i]) === "") {
272     eleToRemove = cardTemplate.getElementsByClassName("recommendations")[0];
273     eleToRemove.parentNode.removeChild(eleToRemove.nextElementSibling);
274     eleToRemove.parentNode.removeChild(eleToRemove);
275 } else {
276     cardTemplate.getElementsByClassName("recommendations")[0].nextElementSibling.children[0].innerText
= getInfoForIB("recommendations", ibArray[i]);
277 }
278 cardContainer.appendChild(cardTemplate);
279 }
280 console.log("Karte erstellt: ", cardContainer);
281 return cardContainer;
282 }
283
284 //Baut den entsprechenden Banner fuer die Multiapp-Ansicht
285 function createPanel(parentNode, appDataArray, hasResults, isSinglePage){
286     var ibArray;
287     var popover;
288     //TODO parentNode umbenennen
289     if(isSinglePage){
290         if(hasResults){
291             ibArray = appDataArray.slice(2, appDataArray.length + 1);
292             var infoCard = document.createElement("div");
293             popover = createPopover(infoCard, ibArray);
294             infoCard.appendChild(popover);
295             parentNode.appendChild(infoCard);
296         }
297     } else {
298         var redLine = false;
299         var banner = document.createElement("div");
300         banner.classList.add("pguard");
301         var funde = document.createElement("span");
302
303         if(hasResults){
304             ibArray = appDataArray.slice(2, appDataArray.length + 1);
305             funde.innerText = "Funde: ";
306             var badge = document.createElement("span");
307             badge.classList.add("badge", "badge-secondary", "float-right");
308             badge.innerText = "" + ibArray.length;
309             funde.appendChild(badge);
310             for(var j = 0; j < ibArray.length; j++){
311                 for(var i = 0; i < ibJson.length; i++){
312                     if(ibArray[j] === ibJson[i].id && ibJson[i]["is_red_line"] === "true"){
313                         redLine = true;
314                         break;
315                     }
316                 }
317             }
318             if(redLine){
319                 banner.style.backgroundColor = "#FF3333";
320                 banner.style.color = "white";
321             } else {
322                 banner.style.backgroundColor = "#3BCCFF";
323                 banner.style.color = "white";
324             }
325
326             $(banner).popover({
327                 title: "Details",
328                 html: true,
329                 trigger: "click",
330                 content: function(){
331                     return createPopover(banner, ibArray);
332                 }
333             });
334         } else {
335             funde.innerText = "Keine Ergebnisse";
336             banner.style.backgroundColor = "#d1d1d1";

```

```
337     }
338
339     banner.appendChild(funde);
340     if($(parentNode).hasClass("card")){
341         banner.classList.add("multiapp");
342         parentNode.insertBefore(banner, parentNode.children[0]);
343     } else {
344         banner.classList.add("similarapp");
345         $(banner).insertBefore(parentNode);
346     }
347 }
348
349 }
350
351 function getNewestDseFromData(data){
352     var newestDse = null;
353     //Dummy-DSEs haben Vorrang. Ansonsten wird die neuste DSE verwendet. Gibt es 2 DSE zum gleichen
354     //Analyse-Datum wird die aus der bevorzugten Quelle gewachlt.
355     for (var i = 0; i < data.dses.length; i++) {
356         //Dummy
357         if(data.dses[i].origin === "dummy"){
358             newestDse = data.dses[i];
359             break;
360         }
361         //Erste gefundene
362         if(!newestDse){
363             newestDse = data.dses[i];
364         } else {
365             //Vergleich mit vorher gefundener DSE
366             var candidateTimeStamp = new Date(data.dses[i].date_infobox_calculation_finished);
367             var dseTimeStamp = new Date(newestDse.date_infobox_calculation_finished);
368             if(candidateTimeStamp.getTime() > dseTimeStamp.getTime()){
369                 newestDse = data.dses[i];
370             } else if (candidateTimeStamp.getTime() === dseTimeStamp.getTime()){
371                 if(dseOrigins.indexOf(data.dses[i].origin) <
372                    dseOrigins.indexOf(newestDse.origin)){
373                     newestDse = data.dses[i];
374                 }
375             }
376         }
377     }
378     return newestDse;
379 }
380
381 //Erstellt die Header fuer die Multiapp-Ansicht
382 function loadInfoPanels(parentNode, isSinglePage) {
383     var appID;
384     if(isSinglePage){
385         var currentURL = new URL(location.href);
386         appID = currentURL.searchParams.get("id");
387     } else {
388         if($(parentNode).attr("data-docid")){
389             appID = $(parentNode).attr("data-docid");
390             console.log("appID gefunden", appID);
391         } else {
392             appID =
393             $(parentNode.children[0].children[0].children[3].children[0]).attr("href").split("id=")[1];
394         }
395     }
396     //Wurde eine App-ID gefunden?
397     if (appID) {
398         var appDataString = getStorageItem(appID);
399
400         var future = function(param) {
401
402             var appDataArray = [];
403             //Prueft ob bereits Daten im localStorage vorhanden und aktuell sind.
404             if(useLocalStorage && param && param.split(trennzeichen)[0]){
405                 var lastUpdate = new Date(param.split(trennzeichen)[0] * 86400000);
406                 if((lastUpdate.getTime() + 259200000) >= today.getTime()){
407                     appDataArray = param.split(trennzeichen);
408                 } else {
409                     console.log("Aktualisiere Daten fuer " + appID);
410                 }
411             }
412             //Falls Daten vorhanden baue das Element daraus
413             if(useLocalStorage && appDataArray.length === 1){
414                 console.log("STORAGE OHNE ERGEBNISSE GEFUNDEN: ", appID, appDataArray);
415                 createPanel(parentNode, appDataArray, false, isSinglePage);
416             } else if(useLocalStorage && appDataArray.length > 1){
417                 console.log("STORAGE GEFUNDEN: ", appID, appDataArray);
418                 createPanel(parentNode, appDataArray, true, isSinglePage);
419             }
420         }
421     }
422 }
```

```

418 //Ansonsten lade die Informationen aus dem Backend
419 } else {
420   console.log("Frage (erstmalig) Daten ab fuer " + appID);
421   anfragencounter ++;
422   $.ajax({
423     url: urlWholeDataSetNoRequest + "" + appID,
424     method: "POST",
425     success: function (response) {
426       var data = response.data;
427       console.log(data);
428       //Wurden bereits DSEs fuer die App gefunden?
429       if (data.dses && data.dses.length > 0) {
430         var storageString = castDseToStorageString(getNewestDseFromData(data));
431         setStorageItem(appID, storageString);
432         console.log("Neuer Eintrag angelegt: ", appID, storageString);
433         appDataArray = storageString.split(trennzeichen);
434         createPanel(parentNode, appDataArray, true, isSinglePage);
435       } else {
436         console.log("KEINE DSE VORHANDEN", appID);
437         setStorageItem(appID, "" + Math.floor(today.getTime() / 86400000));
438         $.ajax({
439           url: urlTriggerNewAnalysis + appID,
440           method: "POST",
441           success: function () {
442             console.log("Anfrage erfolgreich: ");
443           },
444           dataType: "json"
445         });
446         createPanel(parentNode, [], false, isSinglePage);
447       }
448     },
449     dataType: "json"
450   });
451   console.log("Anfragen bisher: ", anfragencounter);
452 }
453 };
454
455 // Promise
456 if (typeof appDataString === 'object') {
457   appDataString.onsuccess = function () {
458     if (appDataString.result) {
459       future(appDataString.result.data);
460     } else {
461       future();
462     }
463   };
464 }
465
466 } else {
467   future(appDataString);
468 }
469 }
470 }
471
472 function fillApps() {
473   //Prueft, ob auf Single-App-Page oder Multi-App-Page
474   if (document.getElementsByClassName("card")[0]) {
475     $(".card").each(function () {
476       loadInfoPanels(this, false);
477       //createPanel(this, [], false, false)
478     });
479   } else {
480     if (document.getElementsByClassName("JHTxhe")[0]) {
481       loadInfoPanels(document.getElementsByClassName("JHTxhe")[0], true);
482     }
483     $(".Vpfgmd").each(function () {
484       loadInfoPanels(this, false);
485     });
486   }
487 }
488
489 function testStorageCap() {
490   localStorage.clear();
491   localStorage.setItem("1", "test");
492   var counter = 2;
493   while (localStorage.getItem("1")) {
494     console.log(counter);
495     localStorage.setItem("" + counter,
496       "ABCDEFGHIIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMN" + counter);
497     counter++;
498   }
499 }
500 //Laedt lokale Json-Bibliothek fuer
501 $.getJSON(chrome.extension.getURL("lib/data/IB_texte.json"), function (input) {

```



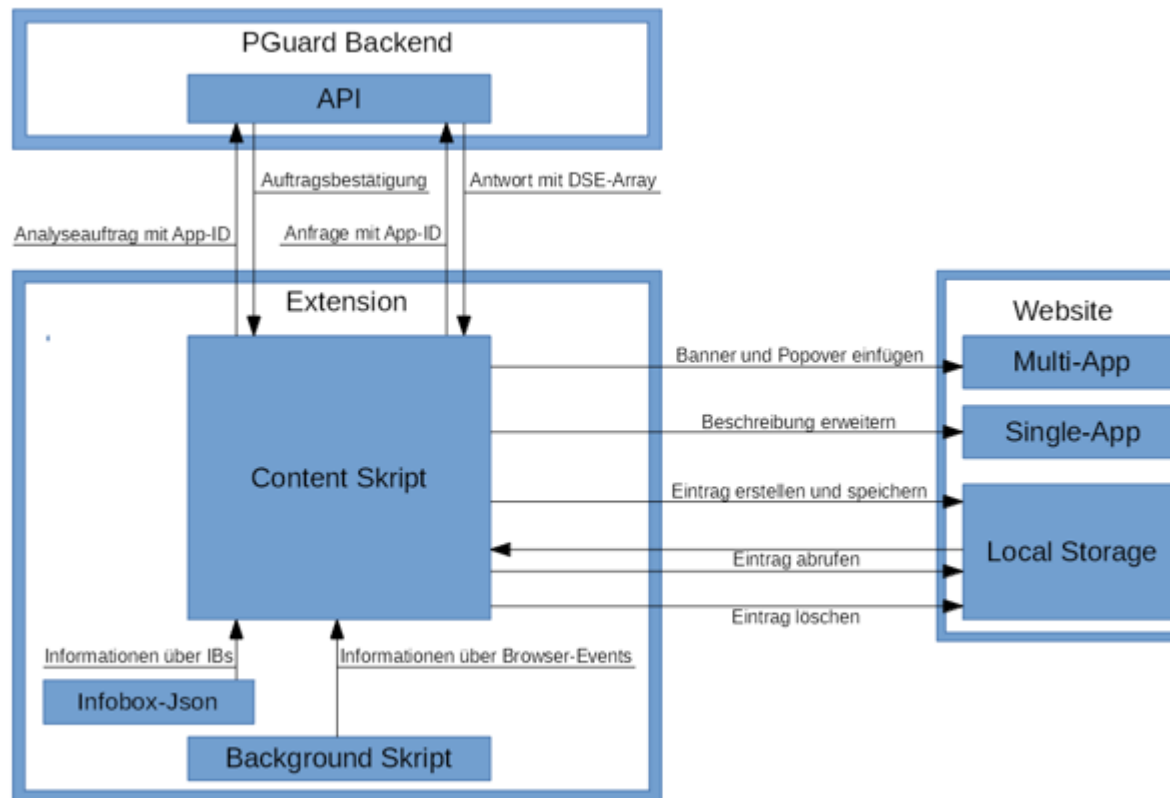
```

501     ibJson = input;
502     $(ibCardTemplate).load(chrome.extension.getURL("lib/templates/ibCardTemplate.html"), function
503         (data) {
504
505         $(innerCollapseTemplate).load(chrome.extension.getURL("lib/templates/innerCollapseTemplate.html"),
506             function() {
507                 if((usedStorage === "localStorage" || usedStorage === "indexedDB") &&
508                     storageAvailable(usedStorage)){
509                     // for (var z = 0; z < localStorage.length; z++){
510                     //     console.log("Key " + z + " : " , localStorage.key(z), " Value: ",
511                         localStorage.getItem(localStorage.key(z)));
512                     // }
513                     // testStorageCap();
514                     // console.log(localStorage.length);
515                     if(usedStorage === "indexedDB"){
516                         var openRequest = indexedDB.open("pguardExt",1);
517
518                         openRequest.onupgradeneeded = function(e){
519                             var thisDB = e.target.result;
520                             console.log("running onupgradeneeded");
521                             if(!thisDB.objectStoreNames.contains("apps")){
522                                 var appsOS = thisDB.createObjectStore("apps",{keyPath: "appID"});
523                             }
524                         };
525
526                         openRequest.onsuccess = function (ev) {
527                             console.log("DB bereit");
528                             db = ev.target.result;
529                             fillApps();
530                         };
531
532                         openRequest.onerror = function (ev) {
533                             console.log("onerror!");
534                             console.dir(ev);
535                         }
536                     }
537                 } else {
538                     console.log("kein Storage verfuegbar.");
539                 }
540             });
541         console.log("Lokale Json geladen.");
542     });

```

Listing 3.3: Aufbau der pguard.js

3.2.5 Ergebnis



3.2.6 Diskussion

3.3 Aufgabe 2: Evaluierung von Caching Methoden einer Browser Extension

3.3.1 Anforderungen

Extension erweitert die Landingpage. Diese in Kategorien unterteilt. Kategorien werden bei jedem Besuch wieder aufgerufen. Spiele mit Vorregistrierung. Stellt über längere Zeit gleiche Apps dar. New + Updated Games und Top-Bewertung. Spielereihenfolge liefern jedes Mal ähnliche Ergebnisse = überlappende Informationen. Empfehlungen für dich und "Das könnte dir gefallen" passen sich vorherigen Suchen an und liefern daher auch redundante Ergebnisse. Selbe App oft mehrmals in der Übersicht vertreten. Konklusion: viel Redundanz. Ergänzende Informationen werden mehrfach benötigt.

Verarbeitet diese nur zu benutzerfreundlichen Format. Informationen müssen von

externen Quelle entnommen werden Dadurch entstehen Anfragen an einen Server mit Antworten Antworten und Anfragen durch // redundant.

Thesen: Nutzung der Extension nach "Veröffentlichung" würde hohen Traffic verursachen mit vielen wiederholten Anfragen. Anfragen könnten ab einer bestimmten Nutzerzahl Server überlasten Performance der Extension leidet unter dieser Art der Informationsbeschaffung Bei Ausfall der Quelle, bietet Extension keinen Mehrwert für den Nutzer mehr.

Lösung: Einrichtung von unabhängigen Speicher zur Aufbewahrung der gewonnen Informationen. Insbesondere viele/wiederholt genutzt Informationen sollen ohne erneute Anfrage zur Verfügung stehen. Neue Anfragen nur bei Veraltung der Informationen bzw. nur von neu aufgetauchten Apps Aufbau einer Struktur zur Abspeicherung der wichtigen Informationen

Verwendung des Speichers:

Welche Informationen stehen pro App zur Verfügung?(1) Welche Informationen werden pro App benötigt? (2) Wie wird der Speicher gepflegt?(3)

(1)

Wird bei Aufbau der App schon Beschrieben? Aynchron!

(2) Alter der Information: Ist die Information auf dem Aktuellen Stand wie die der Quelle? Ist die Information der Quelle veraltet? Wie oft wird so eine Information erneuert? (Neue DSE o.ä) =, Abspeichern des Analysedatums. Regelmäßige Überprüfungen(3 Tage), ob neue Information bei der Quelle vorhanden

Aufruf der App: Wie oft wird diese App aufgerufen? Informationen über die App im Speicher können nach gewisser Zeit gelöscht werden, wenn sie nicht erneut aufgerufen wurde. =, Frequency Count: Zähler im Speicher der bei jedem Aufruf erhöht wird. Regelmäßig wird der Speicher nach niedrigen Zählern durchsucht und diese Einträge gelöscht.

Informationen zur App: Inhalt entspricht Kennnummern der Infoboxen. Also Abspeichern der jeweilig vorhandenen Eigenschaft/Infobox

(3) LRU Konzept oder Fehler falls Speicher voll. Aufbau des Speichers von Speichermethode abhängig (Chrome =, key, value String) Möglichst kurze Zusammenfassung der benötigten Informationen: Alter der Information als Tageszähler seit festem Datum, da Tagesvergleich stattfindet Frequency Counter als einzelner Integer (ÜB: casten auf einstellige Zahl ProtectFaktor?) Abfolge von Kennnummer mit Infoboxen Trennzeichen zum korrekten Auslesen der Informationen. ÄLTER-TAGE(TRENNZEICHEN)FREQ(TRENNZEICHEN)IB(TRENNZEICHEN)IB ..."Bsp: "17500-3-1-2-3-4"

FREQ FÄLLT WEG???

tioniert nach key, value prinzip Alle Datentypen von JavaScript werden unterstützt. Kann indexiert werden um Suchen effizient zu machen. Verwendet Prinzip von Transaktionen Anfragen mit Rückgabewerten als Basis aller Operationen Verfolgt den NoSQL-Ansatz

Speicherlimit nach global Limit (1/2 Festplatte) und Gruppenlimit (1/5 von global Limit, min 10MB max. 2GB) Gruppenlimit voll = voll (Fehler) Global Limit voll = löschen bis wieder frei (Quellenabhängig komplette Elemente gelöscht)

Warum nicht IndexedDB?

Vorteile von IndexedDB: Abspeicherung von großen strukturierten Datenmengen. Nachteile: hoher Aufwand bei Implementierung. Overhead lohnt nicht bei kleinen Datenmengen. Transaktionen blockieren bei Fehlern eventuell den Datenabruf bzw. die Aktualisierung

Storage API von Chrome ausreichend Speicher und geringer aufwand bei der Implementierung. Lediglich Strings benötigt. Indices bei gewählten value-Struktur nicht notwendig.

Vorteil von Session Storage: Speicherpflege nicht notwendig, da 5MB groß genug für Anzahl(?) an App-Informationen während einer Session im PlayStore. Informationen immer auf Stand der Quelle Nachteil: Bei erstmaligen Öffnen des Stores in neuer Browsersession werden viele Anfragen losgeschickt für Apps die bereits in der letzten Session schon angefragt wurden. Bei Serverausfällen fehlen die Informationen Lediglich in einer Session mehrfach aufgerufene Apps ersparen erneute Anfragen. =; Speicherpflege fällt weg, dafür kaum Mehrwert bei Anfragen.

Vorteil von Lokal Storage: Apps werden einmal abgefragt und sind anschließend abgespeichert. Fällt der Server aus können die lokalen Informationen genutzt werden. Daten auch aus letzter Session bleiben vorhanden. Neue Anfragen werden nur dann geschickt wenn aktuelle Daten über 3 Tage alt sind. Nachteile: Speicherpflege notwendig. Dadurch wird die Information länger (Counter und Tag). Zusätzliche Rechenzeit für das Löschen von alten Informationen notwendig. Dadurch wird sichergestellt dass die 5MB nicht überschritten werden und somit Informationen ungewollt verloren gehen. Für Informationen mit hohem Counter muss regelmäßig überprüft werden, ob die Information noch aktuell ist, weil diese in der Regel lange im Speicher verweilt. =; Hohe Einsparung bei Anfragen an den Server möglich. Dafür müssen zusätzliche Operationen zur Speicherpflege und Prüfung der Informationen ausgeführt werden.

3.3.3 Rahmenbedingungen

Plattform: Windows 10 Rechner Build, Specs Chrome Details App Details Was wird gemessen? Limitierungen

Getestet auf:

Windows 10 Education 64 Bit Build 10.0.17134 Prozessor i7-6700K RAM: 16GB
GPU: Nvidia GTX 1070

Chrome 67.0.3396.99 64 Bit

3.3.4 Vorgehensweise

3.3.5 Ergebnisse

Storage: none Ladezeit: 1435ms Start der Extension-Funktionsaufrufe: 944ms Dauer:
491ms Anzahl der Anfragen an das Backend: 0

Storage: Local Storage Ladezeit: 1711ms Start der Extension-Funktionsaufrufe:
892ms Dauer: 819ms Anzahl der Anfragen an das Backend: 125

Storage: none Ladezeit: 1761ms Start der Extension-Funktionsaufrufe: 883ms Dauer:
878ms Anzahl der Anfragen an das Backend: 131

3.3.6 Diskussion

Kapitel 4

Abschließende Diskussion

4.1 Konklusion

4.2 Fortsetzung der Forschung

Kapitel 5

Appendix

5.1 Derivations

5.1.1 Example Matlab Code

Add a sourcefile directly into L^AT_EX

```

1 % simple Kalman Filter example:
2 % state "x" consists of position and velocity
3 % system model "F" is a cinematic model of constant velocity
4 % only the position is measured
5
6 clear % clear all matlab variables
7
8 %% declare matlab variables and assign default (randomly chosen) value
9 % simulation specifications
10 T = 1; % make a measurement every T steps. also called \Delta t
11 % i.e. every 1, 2, 3, ... seconds
12 real_x = [0; 10]; % "real world" state, only needed in simulation context
13 % also called ground truth. start: position=0, velocity=10
14
15 % model specifications
16 model_F = [1, T; % the model we have about the real world
17            0, 1]; % here: cinematic model of constant velocity
18 q = 9; % controls the amount of process noise. is usually unknown
19 model_Q = [T^4/4, T^3/2; % process noise
20            T^3/2, T^2] * q; % arises from the cinematic model
21
22 % estimations specifications
23 esti_x = [0; 10]; % estimated state: position and velocity
24 esti_P = [1, 0; % estimated covariance of esti_x. reflects
25            0, 2]; % the uncertainty about the estimated state esti_x
26
27 esti_z = 0; % estimated measured value. here: just depicting position-entry
28 % of esti_x since we are only interested in the position. Or
29 % maybe it is only possible to measure position, but not velocity
30 esti_S = [0, 0; % estimated covariance of esti_z. Will consist of process noise
31            0, 0]; % with added measurement noise
32
33 H = [1, 0; % observation matrix. we only measure position values
34      0, 0]; % this row could be left out, but then also modify R to 1x1
35 R = [1, 0; % measurement noise. is usually unknown. reflects the
36      0, 1]; % inaccuracy of the sensors
37 K = [0, 0]; % Kalman gain vector
38
39
40 %% Initialization
41 esti_x = [0; 10];
42 esti_P = [1, 0;
43            0, 2];
44
45 for step = 1:1000 % simulate for 1000 steps (simulate continuous time)
46     if mod(step, T) == 0 % if it is time to take a new measurement
47         % update the "real data". For simplicity: take the model F. But could be any
48         % other function, possibly non-linear.
49         % mvnrnd = multi variate normal random numbers
50         real_x = model_F * real_x + transpose(mvnrnd([0,0], model_Q));
51
52         %% Step 1: Prediction Step
53         esti_x = model_F * esti_x; % estimate the new state according to the
54         % system model since we do not have any
55         % control inputs, this term is left out
56         esti_P = model_F * esti_P * transpose(model_F) + model_Q; % update the
57         % covariance of estimated state esti_x
58         esti_z = H * esti_x; % depict position value from estimated state
59         esti_S = H * esti_P * transpose(H) + R; % estimation of the covariance of
60         % the estimated measured value. inherits model
61         % noise and measurement noise
62
63         %% make a measurement z
64         z = H * real_x + transpose(mvnrnd([0,0], R)); % make a noisy measurement
65
66         % Step 2: Innovation Step
67         K = esti_P * transpose(H) * esti_S^-1; % calculate Kalman gain vector by
68         % comparing model and measurement
69         % noise
70         esti_x = esti_x + K * (z - esti_z); % update the estimated state by an
71         % weighted sum of the measurement
72         % and the model-estimation
73         esti_P = esti_P - K * esti_S * transpose(K); % update covariance of
74         % estimated state
75     end
76 end

```

Listing 5.1: Simple example of a Kalman Filter in Matlab

Acknowledgement

First of all, I would like to express my gratitude to ... for the aspiring guidance, useful comments and invaluable support throughout the whole process of this Master Thesis. Furthermore, I would like to thank ..., ... and the other members of the research team for helpful discussions and constructive criticism. In addition, I would like to thank my University supervisor ... for all the helpful remarks, advises and discussions.

Also, I would like to thank my parents and ... who have supported me throughout the entire process by keeping me harmonious and motivated.

Last but not least, I like to thank ... for funding my research and providing me with the facilities being required.

Abbildungsverzeichnis

- 2.1 StatCounter. n.d. Marktanteile der führenden Browserfamilien an der Internetnutzung weltweit von Januar 2009 bis Januar 2019. Statista. Zugriff am 4. März 2019. Verfügbar unter <https://de.statista.com/statistik/daten/studie/157944/umfrage/marktanteile-der-browser-bei-der-internetnutzung->
- 2.2 Browser-Extension Icon in der Adresszeile 8

Literaturverzeichnis

Proclamation

Hereby I confirm that I wrote this thesis independently and that I have not made use of any other resources or means than those indicated.

Forname Surname, Place, 6. März 2019