

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

13. November 2018

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	Projekt PGuard	4
2.2.1	Vorstellung	4
2.2.2	API-Anbindung für die Extension	4
2.3	Implementierung einer Google Chrome Extension	4
2.3.1	Eigenschaften	4
2.3.2	Funktionsumfang	5
2.3.3	Darstellung im Browser	5
2.4	Caching-Methoden	5
2.4.1	Welche Rolle spielt Performance?	5
2.4.2	Verwendete Methoden und deren Eigenschaften	5
3	Hauptteil	7
3.1	Erläuterung der Aufgabenstellungen	7
3.2	Aufgabe 1: Implementierung einer Browser-Extension zur Anzeige von Datenschutzinformationen im PlayStore	7
3.2.1	Anwendungsszenario	7
3.2.2	Anforderungsanalyse	8
3.2.2.1	Funktionale Anforderungen	8
3.2.2.2	Nichtfunktionale Anforderungen	8
3.2.3	Programmaufbau	10
3.2.4	Ergebnis	12
3.2.5	Diskussion	12
3.3	Aufgabe 2: Evaluierung von Caching Methoden einer Browser Extension	12

3.3.1	Anforderungen	12
3.3.2	Vorauswahl	14
3.3.3	Rahmenbedingungen	15
3.3.4	Vorgehensweise	16
3.3.5	Ergebnisse	16
3.3.6	Diskussion	16
4	Abschließende Diskussion	17
4.1	Konklusion	17
4.2	Fortsetzung der Forschung	17
5	Appendix	19
5.1	Derivations	19
5.1.1	Example Matlab Code	20

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. Können auch von außerhalb installiert werden zu Entwicklungszwecken oder wenn nicht auf der Plattform angeboten. 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 eindeutiger 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

2.2 Projekt PGuard

2.2.1 Vorstellung

2.2.2 API-Anbindung für die Extension

2.3 Implementierung einer Google Chrome Extension

2.3.1 Eigenschaften

Die Architektur einer 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 grundlegende 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. Externe Bibliotheken wie JQuery können ebenfalls eingebunden werden, müssen aber aufgrund der Policies 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. Mehr zu den APIs im entsprechenden Abschnitt. Der Aufbau wird unter „content scripts“ (BILD?) 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

2.3.3 Darstellung im Browser

2.4 Caching-Methoden

2.4.1 Welche Rolle spielt Performance?

2.4.2 Verwendete Methoden und deren Eigenschaften

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 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.

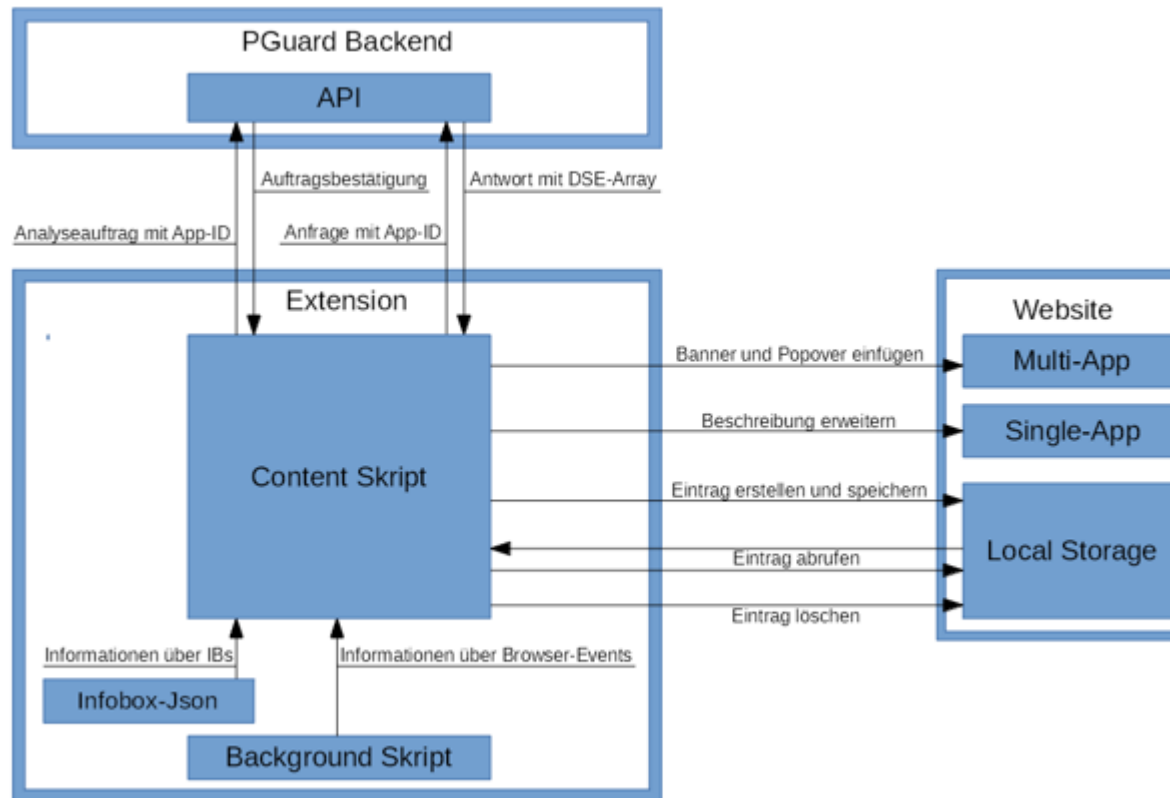
```

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                     pageUrl: { schemes: [ 'https' ], hostContains: 'play.google.com',
12                     pathContains: "/ store/apps" }
13                 })],
14                 actions: [new chrome.declarativeContent.ShowPageAction()]
15             }
16         ]);
17     });
18 });

```

Listing 3.2: background.js

3.2.4 Ergebnis



3.2.5 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

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, 13. November 2018