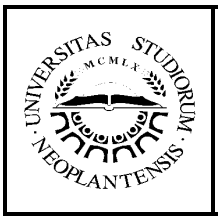


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: **Петар Тијанић**
Број индекса: **РА 138/2020**

Тема рада: **Једно решење Андроид прегледача интегрисаног са покретачем апликација**

Ментор рада: **др Илија Башичевић**

Нови Сад, јул 2024.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Завршни (Bachelor) рад
Аутор, **АУ**:		**Петар Тијанић**
Ментор, **МН**:		**др Илија Башичевић**
Наслов рада, **НР**:		**Једно решење Андроид прегледача интегрисаног са покретачем апликација**
Језик публикације, **ЈП**:		Српски / латиница
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2024.
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		7/28/0/5/21/0/0
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника
Предметна одредница/Кqучне речи, **ПО**:		Хромијум, Прегледач
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		Овај дипломски рад је посвећен креирању напредног Андроид прегледача који пружа корисничко искуство налик десктоп прегледачу, користећи Хромијум пројекат отвореног кода као своју основу. Прегледач имплементира функције као што су подршка за Прогресивне Веб Апликације (PWA) и веб апликације, дизајниране да пруже корисничко искуство које подсећа на традиционалне апликације за десктоп рачунаре. Интеграција овог прегледача са покретачем налик на десктоп ради побољшања интеракције крајњег корисника са Андроид уређајима. Истраживање ће се бавити техничким детаљима коришћења Хромијум пројекта отвореног кода за израду решења за оптимално искуство прегледа на Андроиду.
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		10.07.2024.
Чланови комисије, **КО**:	Председник:	Проф. др Мирослав Поповић
	Члан:	Проф. др Небојша Пјевалица
	Члан, ментор:	Проф. др Илија Башичевић

Потпис ментора

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:		
Identification number, **INO**:		
Document type, **DT**:		Monographic publication
Type of record, **TR**:		Textual printed material
Contents code, **CC**:		Bachelor Thesis
Author, **AU**:		**Petar Tijanić**
Mentor, **MN**:		**Ilija Bašičević, PhD**
Title, **TI**:		**Developing an Android Browser with Desktop-Like UX based on Chromium Open Source Project and Integration with a launcher**
Language of text, **LT**:		Serbian
Language of abstract, **LA**:		Serbian
Country of publication, **CP**:		Republic of Serbia
Locality of publication, **LP**:		Vojvodina
Publication year, **PY**:		2024.
Publisher, **PB**:		Author's reprint
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/28/0/5/21/0/0
Scientific field, **SF**:		Electrical Engineering
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, **S/KW**:		Chromium, Launcher
UC		
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, **N**:		
Abstract, **AB**:		This bachelor's thesis is dedicated to creating an advanced Android browser that provides a desktop-like experience, utilizing the Chromium open-source project as its foundation. The browser includes features such as Progressive Web App (PWA) support and web shortcuts, designed to deliver a user experience reminiscent of traditional desktop browsers. Integrating this browser with a desktop-like launcher for enhancing user interaction with Android devices. The research will delve into the technical intricacies of leveraging the Chromium open-source project to craft a solution for optimal browsing experiences on Android.
Accepted by the Scientific Board on, **ASB**:		
Defended on, **DE**:		10.07.2024.
Defended Board, **DB**:	President:	Miroslav Popović, PhD
	Member:	Nebojša Pjevalica, PhD
	Member, Mentor:	Ilija Bašičević, PhD

Menthor's sign

	УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА ЗАВРШНИ РАД	Датум:

(Податке уноси предметни наставник – ментор)

Студијски програм:	Рачунарство и аутоматика		
Студент:	Петар Тијанић	Број индекса:	РА 138/2020
Степен и врста студија:	Основне академске студије		
Област:	Електротехника и рачунарство		
Ментор:	проф. др. Илија Башичевић		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ЗАВРШНИ РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;

НАСЛОВ ЗАВРШНОГ РАДА:

Једно решење андроид прегледача интегрисаног са покретачем апликација

ТЕКСТ ЗАДАТКА:

Користећи Хромијум пројекат отвореног кода као основу, реализовати прегледач за Андроид оперативни систем са корисничким искуством налик на десктоп рачунар. Прегледач имплементира функције као што су подршка за прогресивне веб апликације (PWA).

Руководилац студијског програма:	Ментор рада:

Примерак за: o - Студента; o - Ментора

Захвалност

Захваљујем се институту РТРК на пруженој шанси и могућности за израду овог пројекта. Захваљујем се академском ментору проф. др Илији Башичевићу, техничком ментору Артјому Кузмицком као и свим колегама из тима који су радили на AndDesk пројекту на стручној помоћи и сарадњи током израде пројекта.

Посебно желим да се захвалим члановима породице и пријатељима на целокупној подршци током студирања.

САДРЖАЈ

Списак слика

Списак табела

Скраћенице

PWA – Progressive web application, прогресивна веб апликација

HTML – Hyper Text Markup Language, језик намењен опису веб страница

CSS – Cascading Style Sheets, језик форматирања помоћу ког се дефинише изглед елемената веб-странице.

Win - Microsoft Windows, Мајкрософт виндоус

URL - Uniform Resource Locators, уједначени локатор ресурса, веб адреса

1. Увод

Задатак је имплементирати напредни Андроид прегледач који ће крајњем кориснику пружити искуство налик десктоп прегледачу, користећи Хромијум пројекат отвореног кода као своју полазну основу.

Прва целина се односи на припрему окружења за изградњу прегледача као и добављање и изградња самог Хромијум прегледача и то последње стабилне верзије. У склопу прве целине такође улази и вршење неколико тестова након успешне изградње прегледача, као на пример колико добро наш прегледач подржава HTML5, пуштање разних аудио и видео формата, подршка за веб камеру и микрофон као и изглед прегледача и корисничко искутво када је 'Desktop site' опција укључена како би знали да дефинишемо опсег проблема којим ћемо се бавити током овог истраживања.

Друга целина обухвата постављање опције 'Desktop site' као подразумеване на нашем прегледачу, као и још пар ствари које ће омогућити корисничко искуство налик десктоп прегледачу, попут постављања односа пиксела и онемогућавања "mobile-user-agent" опције која је подразумевана на Андроид Хромијум прегледачима.

Трећа целина се односи на отклањање проблема који су се појавили након успешно извршене прве две целине, поправљање подршке за микрофон и немогућност учитавања одређених веб страница услед постављања 'Desktop site' опције и онемогућавање подразумеване "mobile-user-agent" опције. У склопу треће целине такође улази и постављање одређених уграђених Хромијумових опција (енг. "Switches") као подразумеваних што ће нам омогућити лакше коришћење а такође и убрзати сам прегледач.

Четврта целина обухвата додавање подршке за Прогресивне Веб Апликације (енг. Progressive web application) као и Веб пречице (енг. Web shortcuts), и интеграцију са покретачем.

2. Теоријске основе

2.1: Chromium:

Chromium веб прегледач је пројекат отвореног кода (енг. open source project) који подржава Google Chrome прегледач. Будући да је пројекат отвореног кода, Google као и многи други могу да користе Chromium изворни код, што је и главни разлог зашто је толико популаран. Захваљујући овој отворености, код је доступан свима за преузимање, компајлирање и прилагођавање.

Нека статистика:

Total Lines:	37,705,703	Code lines:	28,528,607	Percent code lines:	75%
Number of languages:	41	Total comment lines:	4,304,833	Percent comment lines:	11,4%
		Total blank lines:	4,872,263	Percent blank lines:	12,9%

Табела 1 Хромијум статистика

CPU ARCHITECTURES:	**POPULAR CHROMIUM-BASED BROWSERS**:	**OPERATING SYSTEMS**
-x86	Vivaldi	Android
-x64	Microsoft edge	Chromeos
arm	Opera	Ios
arm64	Brave	Mac
mipsel	yandex	Win
mips64el		
s390x		
ppc64		
riscv32		
riscv64		
e2k		
loong64		

Табела 2 Списак подржаних процесорских архитектура и оперативних система

Language breakdown:

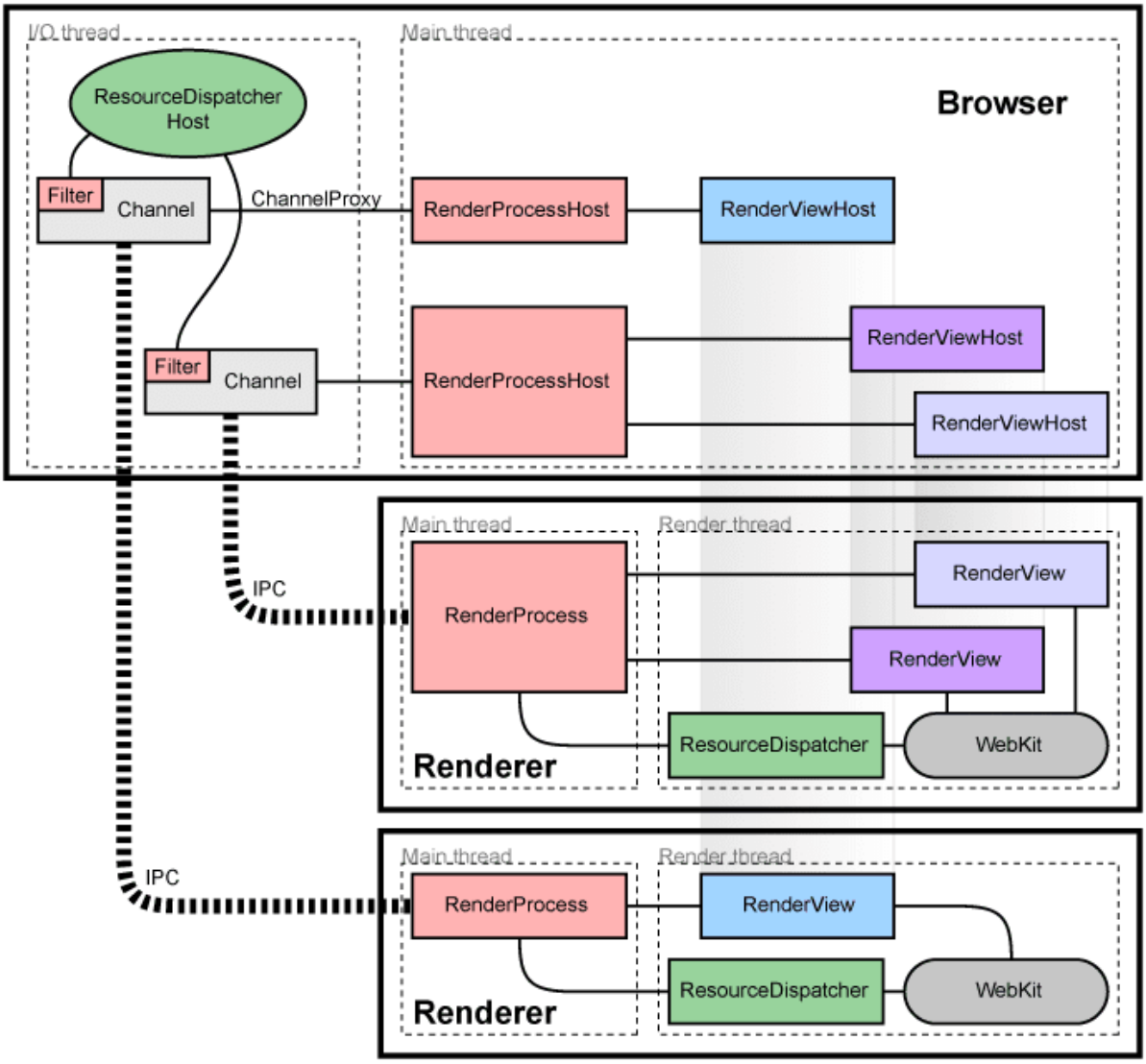
Language	**Code lines**	**Comment lines**	**Comment ratio**	**Blank lines**	**Total lines**	**Total percentage**
C++	14,767,936	2,520,977	14,6%	3,176,373	20,465,265	54,3%
HTML	3,449,314	71,152	2,0%	318,802	3,839,268	10,2%
XML	3,001,250	22,712	0,8%	135,953	3,159,915	8,4%
JavaScript	2,190,204	659,400	23,1%	360,845	3,210,449	8,5%
Java	1,186,304	293,830	20,1%	228,919	1,714,061	4,5%

C	946,715	192,720	16,9%	159,635	1,299,070	3,4%
Objective-C	722,962	103,906	12,6%	146,481	973,349	2,6%
Python	678,111	206,505	23,3%	160,455	1,714,061	2,8%
Rust	543,776	75,831	12,2%	37,574	1,299,070	1,7%
TypeScript	393,040	92,936	19,1%	76,490	973,349	1,5%

Табела 3 Списак програмских језика и написаних линија кода за сваки језик

Једна од кључних карактеристика Хромијума је коришћење више процеса ради повећања безбедности. Главни процес, познат као browser процес, управља корисничким интерфејсом и иницирањем renderer процеса који обрађују и приказују веб садржај. Сваки renderer процес је одвојен и задужен за једну картицу, што смањује могућност напада и побољшава стабилност прегледача.

Када корисник отвори нову картицу у Хромијум прегледачу, browser процес креира нови или користи постојећи renderer процес. Овај процес преузима и обрађује садржај веб странице, извршава JavaScript код и рендерује HTML и CSS. Изолованост renderer процеса омогућава да у случају пада једне картице, остале картице и прегледач остају функционални. Browser процес управља свим картицама, надгледа њихово стварање и затварање, и ослобађа ресурсе када се затвара Хромијум. Ова архитектура обезбеђује сигурност, стабилност и ефикасност у раду са веб садржајем.



Слика 1 Архитектура прегледача[1]

2.2: PWA – Progressive Web Application:

Прогресивна веб апликација (PWA) је врста апликативног софтвера који се испоручује путем интернета, направљен коришћењем уобичајених веб технологија као што су HTML, CSS и JavaScript. Намењена је за рад на било којој платформи са прегледачем који подржава стандарде, укључујући десктоп и мобилне уређаје.

Пошто је прогресивна веб апликација врста веб странице или веб сајта позната као веб апликација, није потребно посебно паковање или дистрибуција. Програмери могу једноставно објавити веб апликацију на интернету, осигурати да испуњава основне захтеве за инсталацију и обезбедити да корисници могу додати апликацију на свој почетни екран. Објављивање апликације на дигиталним дистрибутивним системима као што су Apple App Store или Google Play је опционо.

PWA (Progressive Web App)	*Native-app*
Једна апликација за све уређаје, што доприноси бржем изласку на тржиште.	**Различите апликације за различите оперативне системе (IOS, Android).**
Нижа цена развоја и одржавања.	**Скупљи развој и одржавање.**
Нема потребе за постављањем апликације на продавницу апликација. (Свакако, уколико програмер то жели могуће је.)	**Процес објављивања апликације на продавницу апликације тотално зависи од програмера, самим тим не знамо када ће и да ли ће апликација бити објављена.**
Инсталациони процес се одвија скроз путем прегледача, што повећава број инсталација.	**Инсталациони процес је сачињен од више корака, и сваки додатни корак ће резултовати смањењем корисника.**
Ажурирања се врше аутоматски, нема потребе за објављивањем нових промена путем продавнице апликација.	**Ажурирања се врше системски.**
Корисници могу лако наћи PWA на интернету.	**Нису доступне на интернету.**
PWA апликације могу бити директно додате на почетни екран путем прегледача.	**Потребно је пронаћи и преузети путем продавнице апликација.**
Обавештења (енг. Push notifications) су доступна.	**Обавештења (енг. Push notifications) су доступна.**

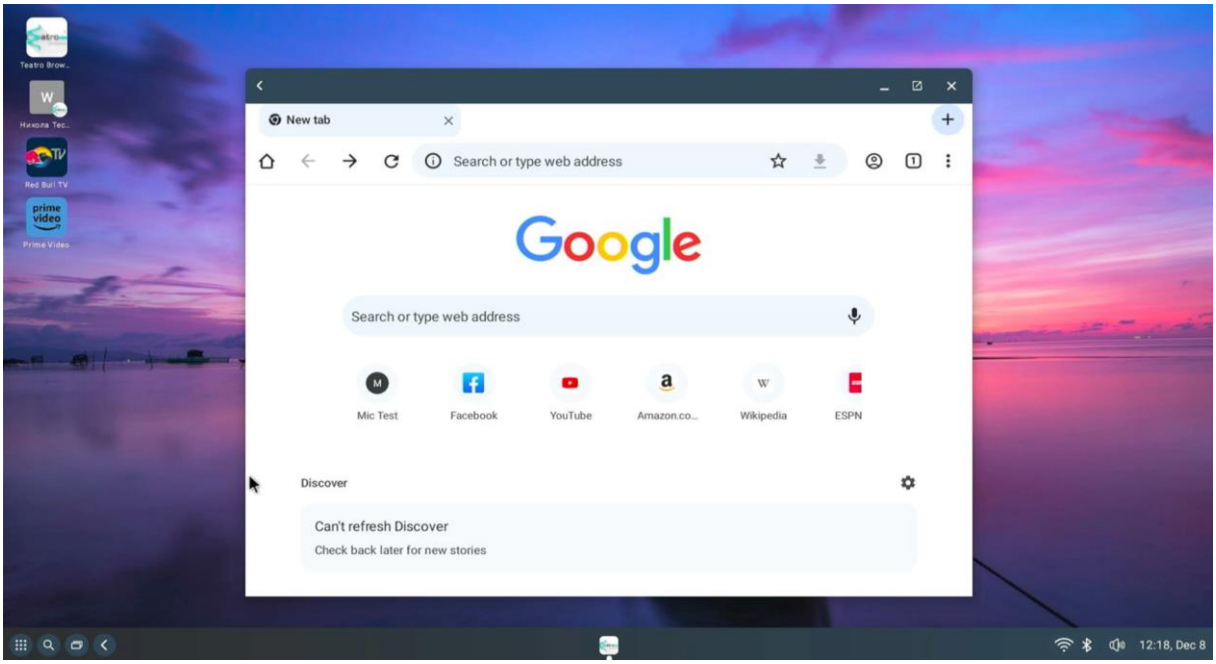
Табела 4 Разлика између PWA и нативних апликација

Критеријуми потребни за успешну инсталацију прогресивне веб апликације:

- Веб апликација није већ инсталирана
- Испуњава хеуристике корисничког ангажовања:
    - Корисник мора да је кликнуо или додирнуо страницу барем једном (у било ком тренутку, чак и током претходног учитавања странице)
    - Корисник мора да је провео најмање 30 секунди гледајући страницу (у било ком тренутку)
- Бити доступна преко HTTPS-а
- Укључује веб апликацијски манифест који укључује:
    - short_name или name
    - иконице - морају укључивати икону од 192px и 512px,
    - start_url
    - display - мора бити један од fullscreen, standalone, или minimal-ui
    - prefer_related_applications не сме бити присутан, или мора бити false

2.3: TeatroOS launcher:

На сликама је представљен Андроид покретач (енг. launcher) који је развијен као засебна андроид апликација у програмском језику Kotlin и његовом радном оквиру Jetpack Compose-у. Покретач изгледом подсећа на рачунарски оперативни систем (енг. desktop layout), има подршку за покретање више прозора у исто време(енг. Multiwindow), за промену величине прозора (енг. resizing) као и за више корисника.



Слика 2 Интеграција прегледача са покретачем

Слика 3 Изглед прегледача

3. Концепт решења

Концепт решења се састоји из четири главне компоненте. Прва се односи на припрему окружења за изградњу прегледача као и сама изградња прегледача. Друга компонента обухвата промену корисничког искуства налик десктоп прегледачу. Трећа компонента се односи на отклањање проблема који су се појавили након успешно извршене прве две целине, као и оптимизација самог прегледача. Четврта целина обухвата додавање подршке за инсталирање веб апликација путем прегледача.

3.1: Изградња прегледача:

Потребно је дефинисати циљну платформу (енг. Target platform), припремити окружење за изградњу, и након тога изградити прегледач са официјалног Хромијумовог репозиторијума. Након успешне изградње, потребно је урадити разне тестове везане за функционалност прегледача, како би се дефинисао опсег проблема на којима ће се радити у наредним целинама.

3.2: Претварање у рачунарски прегледач:

Потребно је трансформисати традиционални Андроид прегледач који је подразумевано мобилни прегледач у потпуно фунцкионални рачунарски прегледач (енг. like-a-desktop) који ће бити прилагођен великим екранима. Како би то успешно урадили, у склопу ове компоненте потребно је постављање опције ‘Desktop site’ као подразумеване на нашем прегледачу, као и још пар ствари које ће омогућити корисничко искуство налик десктоп прегледачу, попут постављања

односа пиксела (енг. Pixel ratio) и онемогућавања “mobile-user-agent” опције која је подразумевана на Андроид Хромијум прегледачима.

3.3: Отклањање познатих проблема уочених током прве две фазе:

Како је већ наведено у првој фази, након успешне изградње и тестирања прегледача очекује се да ће се појавити проблеми у неким функционалностима прегледача с тога је у овој фази потребно поправити те функционалности. Како је андроид прегледач традиционално мобилни прегледач, за очекивати је појаву проблема и након успешно завршене друге фазе када ћемо га претворити у рачунарски прегледач. У склопу ове целине такође улази и постављање одређених уграђених Хромијумових опција (енг. “Switches”) као подразумеваних што ће нам омогућити лакше коришћење а такође и убрзати сам прегледач.

3.4: Додавање подршке за инсталирање веб апликација и интеграција са покретачем:

У склопу ове целине потребно је додавање подршке за Прогресивне Веб Апликације (енг. Progressive web application) као и Веб пречице (енг. Web shortcuts), и интеграцију са покретачем.

4. Програмско решење

Први корак у имплементацији овог дела задатка је дефинисање циљне платформе (енг. Target platform) и изградња Хромијумовог прегледача за дефинисану архитектуру. Одлучено је да се за ово решење користи amlogic s905x4 2gb/16gb сет-топ-бокс, који има 32-битну архитектуру.

4.1: Инсталација и подешавања алата за Хромијум за Андроид:

4.1.1 Инсталирање depot_tools

Клонирајте репозиторијум depot_tools:

```
git clone https://chromium.googlesource.com/chromium/tools/depot_tools.git
```

Додајте depot_tools у PATH (обавезно унесите ово у ваш ~/.bashrc или ~/.zshrc):

```
export PATH="$PATH:/path/to/depot_tools"
```

4.1.2 Преузимање кода

Направите директоријум за Chromium и пређите у њега:

```
mkdir ~/chromium && cd ~/chromium
```

Преузмите код за Android:

```
fetch --nohooks android
```

Ако не желите пуну историју репозиторијума, користите заставицу `--no-history`.

Након завршетка, пређите у `src` директоријум:

```
cd src
```

4.1.3 Конвертовање постојеће Linux инсталације

Додајте Android подршку у .gclient датотеку:

```
echo "target_os = [ 'linux', 'android' ]" >> ../.gclient
```

Синхронизујте нове зависности:

```
gclient sync
```

4.1.4 Инсталација додатних зависности за компилацију

Покрените скрипту за инсталацију зависности:

```
build/install-build-deps.sh --android
```

4.1.5 Покретање hooks

Покрените Chromium-specific hooks:

```
gclient runhooks
```

4.1.6 Подешавање система за превођење

Chromium користи Ninja као главни алат за превођење, а GN за генерисање .ninja датотека.

Направите директоријум за превођење и подесите параметре:

```
gn args out/Default
```

Уредите `args.gn` датотеку са следећим параметрима:

```
target_os = "android"
target_cpu = "arm"
use_remoteexec = true
```

Компајлирање Chromium-a:

```
autoninja -C out/Default chrome_public_apk
```

Уколико је успешно прошло компајлирање, можемо инсталирати апликацију на уређај:

```
out/Default/bin/chrome_public_apk install
```

Апликација ће се појавити на уређају као "Chromium".

Након успешно изршеног инсталирања апликације, потребно је дефинисати тестове везане за функционалност прегледача, како би се дефинисао опсег проблема на којима ће се радити у наредним целинама.

4.1.7 Списак провера:

1. **HTML5 подршка**
   - Тестирати колико добро прегледач подржава HTML5 стандард.
   - Користити HTML5 тест алате који укључују следеће аспекте:
     - **JavaScript:** Проверити компатибилност са различитим JavaScript функцијама и библиотекама.
     - **Распоред елемената:** Проверити правилно приказивање и понашање HTML5 елемената при различитим дизајнима странице.
     - **CSS:** Проверити подршку за нове CSS3 функције и стилове.
     - **SVG:** Тестирати приказивање и интеракцију са SVG графиком.
     - **Canvas:** Проверити перформансе и функционалност canvas елемента кроз различите canvas игре и апликације.
   - Коришћење алата као што су HTML5Test за мерење подршке за HTML5 и Modernizr за проверу компатибилности са новијим веб технологијама.
2. **Репродукција**
   - Тестирати репродукцију различитих типова видео и аудио формата.
   - Проверити функције као што су пауза, наставак, брзо премотавање унапред и уназад.
   - Проверити репродукцију на различитим резолуцијама и брзинама интернета.
   - Проверити да ли се поднаслови и титлови правилно приказују и синхронизују са садржајем.
3. **Снимање видео и аудио садржаја**
   - Тестирати снимање видео и аудио садржаја користећи интегрисане или екстерне камере и микрофоне.
   - Проверити да ли је снимљени садржај доброг квалитета и синхронизован.
   - Проверити опције за подешавање квалитета снимања и коришћење различитих формата за чување.
   - Тестирати различите сценарије, као што су снимање у различитим осветљењима и нивоима буке.
4. **Режим прегледача за радну површину**
   - Проверити функционалност прегледача у режиму за радну површину.
   - Тестирати компатибилност са различитим веб сајтовима и апликацијама.
   - Проверити да ли се странице правилно приказују и да ли су све функције доступне као у стандардном режиму.
   - Тестирати прелаз између мобилног и desktop режима, и проверити да ли се све сесије и подешавања корисника чувају.

Слика 2 Хромијум опција за режим за радну површину

4.1.8 Закључак

Након успешно извршених тестирања, закључено је да је HTML5 веома добро подржан, велики број аудио и видео формата је такође подржан, и Video capturing ради исправно. Грешке које су уочене и које треба поправити у наредним верзијама су:

- Desktop site mode треба да буде подразумевани режим прегледача.
- Немогућност отварања Excel и Word докумената путем прегледача.
- Аудио capturing не ради, не чује се звук када причамо, то је потребно поправити.
- Када пређемо у нови прозор прегледача, аудио или видео репродукција се паузира.
- Потребно је целокупно корисничко искуство претворити да што више подсећа на традиционалне рачунарске прегледаче.
- Потребно је убрзати коришћење прегледача и оптимизовати потрошњу ресурса.

4.2: Претварање у рачунарски прегледач:

Трансформација традиционалног Андроид прегледача у потпуно функционални рачунарски прегледач представља кључан корак ка унапређењу корисничког искуства, посебно за кориснике са великим екранима. Постављање „Desktop site“ опције као подразумеване:

```
// Request Desktop Site secondary settings for Android; including display
// setting and peripheral setting.
BASE_FEATURE(kRequestDesktopSiteAdditions,
             "RequestDesktopSiteAdditions",
             base::FEATURE_ENABLED_BY_DEFAULT);
```

Слика 3 Макро BASE_FEATURE

```
void RegisterPrefs(PrefRegistrySimple* registry) {
  RegisterClipboardAndroidPrefs(registry);
  webauthn::authenticator::RegisterLocalState(registry);
}

void RegisterUserProfilePrefs(user_prefs::PrefRegistrySyncable* registry) {
  NotificationPlatformBridgeAndroid::RegisterProfilePrefs(registry);
  // TODO(shuyng): Use PrefRegistrySimple for RDS prefs registration.
  registry->RegisterBooleanPref(prefs::kDesktopSitePeripheralSettingEnabled,
                                true);
  registry->RegisterBooleanPref(prefs::kDesktopSiteDisplaySettingEnabled,
                                true);
}

}  // namespace android
```

Слика 4 Функција RegisterPrefs

Прва и најзначајнија промена је постављање опције „Desktop site“ као подразумеване. Ово осигурава да све веб странице буду приказане у десктоп режиму, пружајући корисницима потпуни садржај и функционалност, без обзира на то што је наш циљни уређај препознат као мобилни уређај. Ова промена је нарочито важна за уређаје са већим екранима, где мобилни изгледи често не користе у потпуности расположиви простор.

Прилагођавање односа пиксела:

```
private void updateCommon(Point size, float density, float xdpi, float ydpi, Display display, int densityDpi) {
    if (hasForcedDIPScale()) density = sForcedDIPScale.floatValue();

    float max_teatroos_density = (float)densityDpi / DisplayMetrics.DENSITY_220;
    if (DeviceFormFactor.isTeatroOSMode() && density > max_teatroos_density) {
        density = max_teatroos_density;
    }

    boolean isWideColorGamut = false;
    // Although this API was added in Android O, it was buggy.
    // Restrict to Android Q, where it was fixed.
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
        isWideColorGamut = ApiHelperForO.isWideColorGamut(display);
    }

    int pixelFormatId = PixelFormat.RGBA_8888;

    // Note: getMode() and getSupportedModes() can return null in some situations - see
    // crbug.com/1401322.
    Display.Mode currentMode = display.getMode();
    Display.Mode[] modes = display.getSupportedModes();
    List<Display.Mode> supportedModes = null;
    if (modes != null && modes.length > 0) {
        supportedModes = Arrays.asList(modes);
    }

    super.update(size, density, xdpi, ydpi, bitsPerPixel(pixelFormatId),
            bitsPerComponent(pixelFormatId), display.getRotation(), isWideColorGamut, null,
            display.getRefreshRate(), currentMode, supportedModes, getHdrSdrRatio(display));
}
```

Слика 5 Функција UpdateCommon

Да би се обезбедило да сви елементи на страницама изгледају јасно и правилно, неопходно је прилагодити однос пиксела (енг. Pixel ratio) за велике екране. Ово побољшава визуелни квалитет и читање текста, чинећи искуство прегледања пријатнијим и ефикаснијим.

Онемогућавање „mobile-user-agent“ опције:

```
/**
 * A utility class that has helper methods for device configuration.
 */
public class DeviceUtilsImpl {
    private DeviceUtilsImpl() {}

    public static void addDeviceSpecificUserAgentSwitch() {
        try (StrictModeContext ignored = StrictModeContext.allowDiskReads()) {
            if (!DeviceFormFactor.isTablet() && !DeviceFormFactor.isTeatroOSMode()) {
                CommandLine.getInstance().appendSwitch(ContentSwitches.USE_MOBILE_UA);
            }
        }
    }

    public static void updateDeviceSpecificUserAgentSwitch(boolean isTablet) {
        if (DeviceFormFactor.isTeatroOSMode()) {
            return;
        }

        try (StrictModeContext ignored = StrictModeContext.allowDiskReads()) {
            if (isTablet) {
                CommandLine.getInstance().removeSwitch(ContentSwitches.USE_MOBILE_UA);
            } else {
                CommandLine.getInstance().appendSwitch(ContentSwitches.USE_MOBILE_UA);
            }
        }
    }
}
```

Слика 6 Функција UpdateDeviceSpecificUserAgentSwitch

Подразумевана опција „mobile-user-agent“ на Андроид Хромијум прегледачима омогућава веб страницама да препознају уређај као мобилни и прикажу мобилну верзију. Међутим, за постизање десктоп искуства, неопходно је онемогућити ову опцију, омогућавајући прегледачу да се понаша као десктоп прегледач у сваком погледу.

4.3: Отклањање познатих проблема уочених током прве две фазе:

Како је већ наведено у концепту решења, након успешне изградње и тестирања прегледача, као и након претварања прегледача у рачунарски прегледач, очекује се да ће се појавити

проблеми у неким функционалностима прегледача с тога је у овој фази потребно поправити те функционалности.

Грешке које су уочене и које треба поправити су:

- Немогућност отварања Excel и Word докумената путем прегледача .
- Аудио capturing не ради, не чује се звук када причамо, то је потребно поправити.
- Када пређемо у нови прозор прегледача, аудио или видео репродукција се паузира
- Потребно је убрзати коришћење прегледача и оптимизовати потрошњу ресурса.

Функција isHardwareKeyboardAvailable проверава доступност хардверске тастатуре за тренутно приказану картицу (eng. Tab). Ако је показивач на TabAndroid валидан, користи JNI да провери стање. Потребно је дату функцију проширити у случају да показивач на TabAndroid није валидан, како не би дошло до отказа картица приликом учитаванња Excel и Word докумената путем прегледача.

```
bool TabAndroid::isHardwareKeyboardAvailable(TabAndroid* tab_android) {
  if (tab_android) {
      JNIEnv* env = base::android::AttachCurrentThread();
      return Java_TabUtils_isHardwareKeyboardAvailable(
          env, tab_android->GetJavaObject());
  } else {
    // always return true in case of nullptr to TabAndroid(seems it's Chromium bug)
    // if it is not disabled explicitly
    char tmp[PROP_VALUE_MAX] = {};
    __system_property_get("persist.iwedia.teatroos_mode", tmp);
    std::string value(tmp);
    return !(value == "false" || value == "0");
  }
}
```

Слика 7 Функција IsHardwareKeyboardAvailable

Функција MakeAudioInputStream у класи AudioManagerAndroid креира нови аудио улазни стрим. Она узима три аргумента:

- **params** (тип AudioParameters): Параметри који дефинишу аудио конфигурацију.
- **device_id** (тип std::string): ИД уређаја који ће се користити за аудио улаз.
- **log_callback** (тип LogCallback): Повратни позив за логовање.

Функција MakeAudioInputStream креира нови аудио улазни стрим за Андроид уређај. Она проверава да ли тренутна процесна нит одговара тренутном задатку (енг. Task Runner), проверава да ли постоје други аудио улазни стримови и, у зависности од укључености TeatroOS режима и параметара, поставља аудио режим на MODE_IN_COMMUNICATION. Ова функција омогућава

флексибилно подешавање аудио улазних стримова за различите сценарије коришћења на Андроид платформама , и самим тим поправља Аудио capturing, након ових промена се чује звук када причамо.

```
AudioInputStream* AudioManagerAndroid::MakeAudioInputStream(
    const AudioParameters& params,
    const std::string& device_id,
    const LogCallback& log_callback) {
  DCHECK(GetTaskRunner()->BelongsToCurrentThread());
  bool has_no_input_streams = HasNoAudioInputStreams();
  AudioInputStream* stream = AudioManagerBase::MakeAudioInputStream(
      params, device_id, AudioManager::LogCallback());

  // By default, the audio manager for Android creates streams intended for
  // real-time VoIP sessions and therefore sets the audio mode to
  // MODE_IN_COMMUNICATION. However, the user might have asked for a special
  // mode where all audio input processing is disabled, and if that is the case
  // we avoid changing the mode.
  if (IsTeatroOSEnabled()) {
    if (stream && has_no_input_streams) {
      communication_mode_is_on_ = true;
      SetCommunicationAudioModeOn(true);
    }
  } else {
    if (stream && has_no_input_streams &&
        params.effects() != AudioParameters::NO_EFFECTS) {
      communication_mode_is_on_ = true;
      SetCommunicationAudioModeOn(true);
    }
  }

  return stream;
}
```

Слика 8 Функција MakeAudioInputStream

Функција ShouldCloseAppWithZeroTabs затвара апликацију када је последња картица затворена.

```
/**
 * @return Whether to close the app when the user has zero tabs.
 */
public static boolean shouldCloseAppWithZeroTabs() {
    if (DeviceFormFactor.isTeatroOSMode()) return true;

    return HomepageManager.isHomepageEnabled()
            && !UrlUtilities.isNTPUrl(HomepageManager.getHomepageUri());
}
```

Слика 9 Функција ShouldCloseAppWithZeroTabs

Функција ShouldIgnoreSwipeGesture онемогућава мењање картица путем додира из адресног бара.

```
boolean shouldIgnoreSwipeGesture() {
    if (DeviceFormFactor.isTeatroOSMode()) return true;
    if (mUrlHasFocus || mFindInPageToolbarShowing) return true;
    return mAppMenuButtonHelper != null && mAppMenuButtonHelper.isAppMenuActive();
}
```

Слика 10 Функција ShouldIgnoreSwipeGesture

Функција MaybeShowCursorInLocationBar аутоматски фокусира курсор у горњем делу екрана, како би одмах могли да унесемо жељени текст.

```
/**
 * Called whenever the NTP could have been entered (e.g. tab content changed, tab navigated to
 * from the tab strip/tab switcher, etc.). If the user is on a tablet and indeed entered the
 * NTP, we will check two cases:
 *   1. If a11y is enabled, we will request a11y focus on the omnibox (e.g. for TalkBack).
 *   2. If a keyboard is plugged in, we will show the URL bar cursor (without focus animations).
 */
private void maybeShowCursorInLocationBar() {
    if (!DeviceFormFactor.isNonMultiDisplayContextOnTablet(mActivity)) return;
    Tab tab = mLocationBarModel.getTab();
    if (tab == null) return;
    NativePage nativePage = tab.getNativePage();
    if (!(nativePage instanceof NewTabPage) && !(nativePage instanceof IncognitoNewTabPage)) {
        return;
    }

    if (ChromeAccessibilityUtil.get().isAccessibilityEnabled()
            && nativePage instanceof NewTabPage) {
        mLocationBar.requestUrlBarAccessibilityFocus();
    }

    if (mActivity.getResources().getConfiguration().keyboard == Configuration.KEYBOARD_QWERTY) {
        mLocationBar.showUrlBarCursorWithoutFocusAnimations();
    }
}
```

Слика 11 Функција MaybeShowCursorInLocationBar

Функција SetContentCommandLineFlags прилагођава командну линију за покретање прегледача на Андроид платформи на основу различитих услова и конфигурација. Она обезбеђује специфично понашање прегледача за различите сценарије, као што су рад у једном процесу, подршка за TeatroOS режим (режим у ком ми користимо прегледач) и оптимизације за уређаје са малом меморијом.

Постављање командних линија ако је TeatroOS омогућен:

(cc::switches::kNumRasterThreads, std::to_string(num_processors));

Ово поставља број нити за растеризацију на максималан број доступних процесора (језгара).

parsed_command_line->AppendSwitch(switches::kDisableRendererAccessibility);

Онемогућава приступачна подешавања (енг. Accessibility Settings) рендереру, што отклања проблем нестајања курсора миша приликом пуштања видео записа.

parsed_command_line>AppendSwitch(switches::kDisableBackgroundMediaSuspend);

Онемогућава суспендовање медија у позадини, што омогућава да када пређемо у нови прозор прегледача, аудио или видео репродукција се не паузира.

parsed_command_line->AppendSwitchASCII("enable-use-zoom-for-dsf", "false");

Поставља опцију аутоматског зумирања мањег садржаја странице на "false".

parsed_command_line->AppendSwitch(switches::kDisablePullToRefreshEffect);

Онемогућава ефекат повлачења миша на доле да би се страница освежила.

```
void SetContentCommandLineFlags(bool single_process) {
  // May be called multiple times, to cover all possible program entry points.
  static bool already_initialized = false;
  if (already_initialized)
    return;
  already_initialized = true;

  base::CommandLine* parsed_command_line =
      base::CommandLine::ForCurrentProcess();

  if (single_process) {
    // Need to ensure the command line flag is consistent as a lot of chrome
    // internal code checks this directly, but it wouldn't normally get set when
    // we are implementing an embedded WebView.
    parsed_command_line->AppendSwitch(switches::kSingleProcess);
  }

  parsed_command_line->AppendSwitch(switches::kEnableViewport);
  parsed_command_line->AppendSwitch(switches::kValidateInputEventStream);
#if defined(OS_ANDROID)
  if (IsTeatroOSEnabled()) {
#if defined(ARCH_CPU_ARM_FAMILY) && !defined(ARCH_CPU_ARM64)
    int num_processors = base::SysInfo::NumberOfProcessors();
    parsed_command_line->AppendSwitchASCII(cc::switches::kNumRasterThreads, std::to_string(num_processors));
#endif // defined(ARCH_CPU_ARM_FAMILY) && !defined(ARCH_CPU_ARM64)
    parsed_command_line->AppendSwitch(switches::kDisableRendererAccessibility);
    parsed_command_line->AppendSwitch(switches::kDisableBackgroundMediaSuspend);
    parsed_command_line->AppendSwitchASCII("enable-use-zoom-for-dsf", "false");
    parsed_command_line->AppendSwitch(switches::kDisablePullToRefreshEffect);
    parsed_command_line->AppendSwitch(switches::kDisableOverscrollEdgeEffect);
  }
#endif // defined(OS_ANDROID)

  if (base::android::BuildInfo::GetInstance()->sdk_int() >=
      base::android::SDK_VERSION_MARSHMALLOW) {
    parsed_command_line->AppendSwitch(switches::kEnableLongpressDragSelection);
    parsed_command_line->AppendSwitchASCII(
        blink::switches::kTouchTextSelectionStrategy,
        blink::switches::kTouchTextSelectionStrategy_Direction);
  }

  // On legacy low-memory devices the behavior has not been studied with regard
  // to having an extra process with similar priority as the foreground renderer
  // and given that the system will often be looking for a process to be killed
  // on such systems.
  if (base::SysInfo::IsLowEndDevice())
    parsed_command_line->AppendSwitch(switches::kInProcessGPU);

  // Disable anti-aliasing.
  parsed_command_line->AppendSwitch(
      cc::switches::kDisableCompositedAntialiasing);
}
```

Слика 12 Функција SetContentCommandLineFlags

4.4: Додавање подршке за инсталирање веб апликација и интеграција са покретачем:

Од једноставног клика у прегледачу, до инсталирања целе апликације директно у позадини, можемо закључити да PWA чине сложене веб технолгије сачињене од неколико различитих Андроид компоненти. Намера ове целине је да се разуме цео ток од корисника који додирује дугме „Инсталирај апликацију“ до тренутка када је PWA инсталиран на покретачу.

Када корисник кликне на дугме „Додај на почетни екран“ или „Инсталирај апликацију“ као што је приказано у наставку, долазимо до кода који се налази у AddToHomeScreenDialogView класи.

```
/**
 * From {@link  ModalDialogProperties.Controller}. Called when a dialog button is clicked.
 *
 * @param model The dialog model that is associated with this click event.
 * @param buttonType The type of the button.
 */
@Override
public void onClick(PropertyModel model, int buttonType) {
    int dismissalCause = DialogDismissalCause.NEGATIVE_BUTTON_CLICKED;
    if (buttonType == ModalDialogProperties.ButtonType.POSITIVE) {
        mDelegate.onAddToHomescreen(mShortcutTitleInput.getText().toString());
        dismissalCause = DialogDismissalCause.POSITIVE_BUTTON_CLICKED;
    }
    mModalDialogManager.dismissDialog(mDialogModel, dismissalCause);
}
```

Слика 13 Функција OnClick

Та функција позива функцију из AddToHomescreenMediator класе, која позива нативни код што можемо закључити из назива “JNI” у наслову функције. JNI је механизам који омогућава Јава коду да позива и буде позван из нативних апликацијаи библиотека написаних у C, C++ и асемблеру.

```
@Override
public void onAddToHomescreen(String title) {
    if (mNativeAddToHomescreenMediator == 0) return;

    AddToHomescreenMediatorJni.get().addToHomescreen(mNativeAddToHomescreenMediator, title);
    destroyNative();
}
```

Слика 14 Функција OnAddToHomeScreen

Одмах затим долазимо до одговарајуће функције која се налази на путањи components/webapps/browser/android/add_to_homescreen_mediator.cc са следећим кодом:

```
void AddToHomescreenMediator::AddToHomescreen(
    JNIEnv* env,
    const JavaParamRef<jstring>& j_user_title) {

  if (!params_ || GetWebContents() == nullptr)
    return;

  bool teatroos_mode_enabled = IsTeatroOSEnabled();

  if (params_->app_type == AddToHomescreenParams::AppType::SHORTCUT) {
    if (teatroos_mode_enabled) {
      // in case of TeatroOS we force to run all webshortcuts in a browser tab
      params_->shortcut_info->display = blink::mojom::DisplayMode::kBrowser;
    }
    params_->shortcut_info->user_title =
        base::android::ConvertJavaStringToUTF16(env, j_user_title);
  } else if (params_->app_type == AddToHomescreenParams::AppType::WEBAPK) {
    if (teatroos_mode_enabled) {
      params_->shortcut_info->display = blink::mojom::DisplayMode::kStandalone;
      params_->shortcut_info->user_title =
        base::android::ConvertJavaStringToUTF16(env, j_user_title);
    } else {
      AppBannerManager* app_banner_manager =
          AppBannerManager::FromWebContents(GetWebContents());
      app_banner_manager->TrackInstallPath(/* bottom_sheet= */ false,
                                           params_->install_source);
    }
  }

  AddToHomescreenInstaller::Install(GetWebContents(), *params_,
                                    teatroos_mode_enabled, event_callback_);
}
```

Слика 15 Функција AddToHomeScreen

Овде можемо видети да код проверава објекат params да би открио да ли је захтевана инсталација за PWA или за веб пречице (енг. Web Shortcuts).

Пратећи позив функције AddToHomescreenInstaller::Install(GetWebContents(), *params_, event_callback_), где параметар GetWebContents() враћа HTML document тренутне странице, стижемо до следећег парчета кода:

```
namespace webapps {

// static
void AddToHomescreenInstaller::Install(
    content::WebContents* web_contents,
    const AddToHomescreenParams& params,
    bool teatro_mode_enabled,
    const base::RepeatingCallback<void(Event, const AddToHomescreenParams&)>&
        event_callback) {
  if (!web_contents) {
    event_callback.Run(Event::INSTALL_FAILED, params);
    return;
  }

  event_callback.Run(Event::INSTALL_STARTED, params);
  switch (params.app_type) {
    case AddToHomescreenParams::AppType::NATIVE:
      InstallOrOpenNativeApp(web_contents, params, event_callback);
      break;
    case AddToHomescreenParams::AppType::WEBAPK:
      if (teatro_mode_enabled) {
        WebappsClient::Get()->InstallShortcut(web_contents, params);
      } else {
        WebappsClient::Get()->InstallWebApk(web_contents, params);
      }
      break;
    case AddToHomescreenParams::AppType::SHORTCUT:
      WebappsClient::Get()->InstallShortcut(web_contents, params);
      break;
  }
  event_callback.Run(Event::INSTALL_REQUEST_FINISHED, params);
}

// static
```

Слика 16 Функција Install

Овде видимо да уколико је PWA инсталација, да се позива WebappsClient::Get()->InstallWebApk(web_contents, params). Погледајмо код за то:

```
void ChromeWebappsClient::InstallWebApk(content::WebContents* web_contents,
                                        const AddToHomescreenParams& params) {
  WebApkInstallService::Get(web_contents->GetBrowserContext())
      ->InstallAsync(web_contents, *(params.shortcut_info), params.primary_icon,
                     params.install_source);
}
```

Слика 17 Функција InstallWebApk

Структура shortcut_info садржи све потребне информације за инсталираше апликација, попут имена, логоа, УРЛ-а на којој се налази страница као и начина приказивања апликације.
Пар позива касније коначно стижемо до следећег дела кода:

```
void WebApkInstaller::InstallAsync(content::WebContents* web_contents,
                                   const webapps::ShortcutInfo& shortcut_info,
                                   const SkBitmap& primary_icon,
                                   FinishCallback finish_callback) {
  DCHECK(!install_from_webapk_service_);
  install_duration_timer_ = std::make_unique<base::ElapsedTimer>();

  web_contents_ = web_contents->GetWeakPtr();
  install_shortcut_info_ =
      std::make_unique<webapps::ShortcutInfo>(shortcut_info);
  install_primary_icon_ = primary_icon;
  short_name_ = shortcut_info.short_name;
  finish_callback_ = std::move(finish_callback);
  source_ = install_shortcut_info_->source;
  manifest_url_ = install_shortcut_info_->manifest_url;
  task_type_ = INSTALL;

  if (!server_url_.is_valid()) {
    OnResult(webapps::WebApkInstallResult::SERVER_URL_INVALID);
    return;
  }

  CheckFreeSpace();
}
```

Слика 18 Функција InstallAsync

Функција CheckFreeSpace() само проверава да ли има довољно простора за инсталирање и позива следећу функцију уколико је претходни услов испуњен. Шаље се захтев Google серверу, који је задужен за генерисање и инсталирање PWA.

```
void WebApkInstaller::SendRequest(
    const net::NetworkTrafficAnnotationTag& traffic_annotation,
    const std::string& serialized_proto) {
  DCHECK(server_url_.is_valid());

  timer_.Start(
      FROM_HERE, base::Milliseconds(webapk_server_timeout_ms_),
      base::BindOnce(&WebApkInstaller::OnResult, weak_ptr_factory_.GetWeakPtr(),
                     webapps::WebApkInstallResult::REQUEST_TIMEOUT));

  auto request = std::make_unique<network::ResourceRequest>();
  request->url = server_url_;
  request->method = "POST";
  request->load_flags = net::LOAD_DISABLE_CACHE;
  request->credentials_mode = network::mojom::CredentialsMode::kOmit;
  loader_ =
      network::SimpleURLLoader::Create(std::move(request), traffic_annotation);
  loader_->AttachStringForUpload(serialized_proto, kProtoMimeType);
  loader_->DownloadToStringOfUnboundedSizeUntilCrashAndDie(
      GetURLLoaderFactory(browser_context_),
      base::BindOnce(&WebApkInstaller::OnURLLoaderComplete,
                     weak_ptr_factory_.GetWeakPtr()));
}
```

Слика 19 Функција SendRequest

5. Резултати

У овом одељку ћемо испитати колико се наше решење андроид десктоп прегледача разликује од неких других сличних комерцијалних решења. Као прегледач са којим ћемо поредити тестне случаје коришћен је JioSphere андроид десктоп прегледач који је развијен за потребе компаније Jio. Поређење ће бити по питању подржаних функционалности, у смислу који прегледач пружа крајњим корисницама боље корисничко искуство и који више подсећа на рачунарски прегледач по питању корисничког искуства.

Описи функционалности су дати у табели. Одлучено је да се за тестирсање резултата такође користи amlogic s905x4 2gb/16gb сет-топ-бокс, који има 32-битну архитектуру.

Редни број:	**Функционалност:**	**Подржава Teatro Browser**	**Подржава JioSphere Browser**
1.	Након преласка у нову картицу, сав аудио садржај који се тренутно пушта се паузира.	Подржава.	Не подржава.
2.	Када пређемо у нову картицу, курсор се аутоматски фокусира како би одмах могли да унесемо текст и претражујемо интернет.	Подржава.	Не подржава.
3.	Укинута је могућност за прелазак између картица помоћу превлачења миша у адресном бару.	Подржава.	Не подржава.
4.	Отклоњен је проблем нестанка курсора миша приликом пуштања видео садржаја	Подржава.	Не подржава.
5.	Укинута је могућност превлачења миша ка доле како би се тренунта веб страница поново учитала.	Подржава.	Не подржава.
6.	Апликација се затвара када се последња картица затвори.	Подржава.	Не подржава.

Табела 5 Резултати

6. Закључак

У овом раду је описано једно решење напредног андроид прегледача, који ће крајњем кориснику пружити искуство налик десктоп прегледачу, користећи Хромијум пројекат отвореног кода као своју полазну основу. Такође додата је подршка за инсталирање апликација преко прегледача и интеграција са самим покретачем.

Извршено је поређење са већ постојећим решењем на тржишту, у смислу који прегледач пружа крајњим корисницама боље корисничко искуство и који више подсећа на рачунарски прегледач по питању корисничког искуства.

Даљи правци развоја могу укључивати смањење потрошње меморије, оптимизација перформанси и убрзање рада прегледача. Такође, могуће је и развити засебан сервер који ће генерисати инсталирање апликација путем прегледача, како би се избегла зависност и комуницирање са Google Play серверима.

7. Литература

[1]https://www.chromium.org/developers/design-documents/multi-process-architecture посећено 29.5.2024.

[2]https://blog.takemyhand.xyz/2023/10/analyzing-webapks-in-chrome-for-android посећено 1.6.2024.

[3]https://www.computerworld.com/article/1717405/googles-chromium-browser-explained.html посећено 29.5.2024.

[4]https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps посећено 4.6.2024.

[5] https://www.lifewire.com/chromium-web-browser-4171288 посећено 9.6.2024.