



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Милица Божић

Прилагођење интернет прегледача на DirectFB графичку библиотеку

МАСТЕР РАД

Нови Сад , 2024. године



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:	Мастер рад	
Аутор, АУ:	Милица Божић	
Ментор, МН:	Др Илија Башичевић	
Наслов рада, НР:	Прилагођење интернет прегледача на DirectFB графичку библиотеку	
Језик публикације, ЈП:	Српски / Ћирилица	
Језик извода, ЈИ:	Српски	
Земља публикавања, ЗП:	Република Србија	
Уже географско подручје, УГП:	Војводина	
Година, ГО:	2024. година	
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)	7/35/0/0/23/0/0	
Научна област, НО:	Електротехника и рачунарство	
Научна дисциплина, НД:	Примењене рачунарске науке и информатика - Инжењеринг информационих	
Предметна одредница/Кључне речи, ПО:	Chromium, Ozone платформа, DirectFB графичка библиотека	
УДК		
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, ВН:		
Извод, ИЗ:	Задатак овог рада јесте прилагођење интернет прегледача на произвољну графичку библиотеку, конкретно DirectFB. Овај задатак је сачињен од две целине, које је потребно испунити. У првој целини неопходно је развити тестну апликацију за приказивање графичког садржаја користећи DirectFB библиотеку. У другој целини потребно је прилагодити интернет прегледач, то јесте реализовати Ozone интеграцију, да користи DirectFB библиотеку за приказ садржаја на екран.	
Датум прихватања теме, ДП:		
Датум одбране, ДО:		
Чланови комисије, КО:	Председник:	
	Члан:	
	Члан, ментор:	Др Илија Башичевић
		Потпис ментора



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 :	Milena Božić	
Mentor, MN :	Goran Marinković, PhD., Graduate Geodetic Engineer	
Title, TI :	The adaptation of an internet browser to DirectFB graphics library	
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)	7/35/0/0/23/0/0	
Scientific field, SF :	Electrical Engineering	
Scientific discipline, SD :	Applied Computer Science and Informatics - Information Systems Engineering	
Subject/Key words, S/KW :	Chromium, Ozone platform, DirectFB graphics library	
UC		
Holding data, HD :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, N :		
Abstract, AB :	The task of this work is to adapt an internet browser to an arbitrary graphics library, specifically DirectFB. This task consists of two parts that need to be fulfilled. In the first part, it is necessary to develop a test application for displaying graphical content using the DirectFB library. In the second part, it is necessary to adapt the internet browser, i.e., to implement Ozone integration, to use the DirectFB library for displaying content on the screen.	
Accepted by the Scientific Board on, ASB :		
Defended on, DE :		
Defended Board, DB :	President:	
	Member:	
	Member, Mentor:	Dr Ilija Bašičević
		Mentor's sign

	УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Датум:
	ЗАДАТАК ЗА МАСТЕР РАД	Датум:

(Податке уноси предметни наставник - ментор)

СТУДИЈСКИ ПРОГРАМ:	Рачунарство и аутоматика
РУКОВОДИЛАЦ СТУДИЈСКОГ ПРОГРАМА:	Др Мирна Капетина, доцент

Студент:	Милица Божић	Број	Е2 18/2022
Област:	Електротехника и рачунарство		
Ментор:	Илија Башичевић		

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

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

НАСЛОВ МАСТЕР РАДА:

Прилагођење интернет прегледача на DirectFB графичку библиотеку

ТЕКСТ ЗАДАТКА:

Интернет прегледач заснован на Chromium пројекту мора бити у стању да ради на различитим платформама, као што су Linux и Android. Различите платформе могу имати различите графичке системе који се користе за приказ излаза прегледача. Chromium обезбеђује слој за интеграцију графике који се зове Ozone и који се користи за прилагођавање прегледача различитим графичким система.

Задаци:

1. Развити тестну апликацију за приказивање графичког садржаја користећи DirectFB библиотеку.
2. Прилагодити интернет прегледач (реализовати Ozone интеграцију) да користи DirectFB библиотеку за приказ садржаја на екран.

Руководилац студијског програма:	Ментор рада:

Примерак за: О - Студента; О - Ментора
--

Захвалност

Овом приликом изразила бих велику захвалност ментору професору Илији Башичевићу, који је својим сугестијама и примедбама помогао да овај рад буде на нивоу на ком се налази.

Такође, желим да се захвалим техничком ментору Тимуру Валееву и Мариу Радоњићу на посвећености, подршци и помоћи током израде овог пројекта, као и осталим колегама који су били са мном у тиму.

Посебну захвалност дугујем својој породици: тати Ненаду, мами Миленки, сестри Милени и веренику Игору на неизмерној љубави, подршци и разумевању пруженом током израде рада.

САДРЖАЈ

1.	Увод	2
2.	Теоријске основе	3
2.1	Интернет прегледачи	3
2.2	<i>Chromium</i>	5
2.2.1	Архитектура <i>Chromium</i> -а	5
3.	Концепт решења	8
3.1	Компоненте система	8
3.2	Платформа <i>Ozone</i>	8
3.3	Графичка библиотека <i>DirectFB</i>	10
3.4	<i>BUILD.gn</i>	11
4.	Програмско решење	12
4.1	Тестна апликација за приказивање графичког садржаја користећи <i>DirectFB</i> библиотеку	12
4.2	Прилагођење интернет прегледача да користи <i>DirectFB</i> библиотеку за приказ садржаја на екран	13
5.	Резултати.....	22
5.1	Тестна апликација за приказивање графичког садржаја користећи <i>DirectFB</i> библиотеку	22
5.2	Прилагођење интернет прегледача да користи <i>DirectFB</i> библиотеку за приказ садржаја на екран	22
6.	Закључак	26
7.	Литература	28

СПИСАК СЛИКА

Слика 1 Кориснички интерфејс интернет прегледача.....	3
Слика 2 Основне компоненте прегледача	4
Слика 3 Приказ мултипроцесне архитектуре <i>Chromium</i> -а.....	6
Слика 4 Вишеструки бекендови <i>Ozone</i> платформе	9
Слика 5 Приказ макро дефиниције.....	12
Слика 6 Приказ функције <i>Dump()</i>	13
Слика 7 <i>Build.gn</i> конфигурациони фајл <i>DirectFB</i> интеграције.....	14
Слика 8 Приказ методе иницијализације.....	15
Слика 9 Приказ функције <i>CreateWindow()</i>	16
Слика 10 Приказ класе <i>DirectFBPlatformWindow</i>	17
Слика 11 Приказ класе <i>DirectFBPlatformScreen</i>	17
Слика 12 Приказ функционалности класе <i>DirectFBSurface</i>	18
Слика 13 Приказ функционалности класе <i>DirectFBSurfaceFactory</i>	19
Слика 14 Приказ класе <i>GLOzoneEGLDFB</i>	20
Слике 15 Приказ синглтон шаблона	20
Слика 16 Приказ функције <i>InitializeUI()</i>	21
Слика 17 Приказ функције <i>InitializeGPU()</i>	21
Слика 18 Приказ резултата при покретању <i>DirectFB</i> апликације.....	22
Слика 19 Приказ учитавања <i>Google</i> странице.....	23
Слика 20 Резултат учитавања <i>Google</i> странице	23
Слика 21 Приказ учитавања <i>Yandex</i> странице.....	24
Слика 22 Приказ резултата учитавања <i>Yandex</i> странице	24
Слика 23 Приказ учитавања странице <i>Yandex.ru</i>	25

СКРАЋЕНИЦЕ

API – *Application Programming Interface*, програмски интерфејс

CE Linux Forum – *Consumer Electronics Linux Forum*, форум за потрошачку електронику базирану на Linux-у

CSS – *Cascading Style Sheets*, језик за описивање презентације *HTML* и *XML* документа.

DIP – *Device Independent Pixel*, јединица мере за величину елемената на екрану која не зависи од густине пиксела уређаја

DirectFB – *Direct Frame Buffer*, графичка библиотека

EGL – *Embedded-System Graphics Library*, библиотека за графику у уграђеним системима

GN - *Generated Ninja*, алат за иградљу софтвера

GPU – *Graphics Processing Unit*, јединица за графичку обраду

HTML – *HyperText Markup Language*, језик за креирање интернет страница

I/O – *Input/Output*, улаз/излаз, операције или комуникације између система и спољашњих уређаја

GPL – *Lesser General Public License*, општа јавна лиценца

OpenGL – *Open Graphics Library*, графичка библиотека

PDF – *Portable Document Format*, облик записа документа унутар датотеке

UI – *User Interface*, кориснички интерфејс

URL – *Uniform Resource Locator*, идентификатор ресурса

VSync – *Vertical Synchronization*, вертикална синхронизација

WebOS – *Web Operating System*, интернет оперативни систем

XML – *Extensible Markup Language*, прошириви (мета) језик за означавање текстуалних докумената

1. Увод

Брзи напредак интернет технологија и растућа потражња за високоперформансним интернет апликацијама истакли су важност ефикасних и прилагодљивих интернет прегледача. *Chromium*, пројекат интернет прегледача отвореног кода, постао је једна од најпопуларнијих платформи због своје архитектуре и широких могућности за прилагођавање. Међутим, разноврсност хардвера и оперативних система представља значајан изазов у осигуравању оптималних перформанси на свим платформама.

Задатак овог рада јесте прилагођење интернет прегледача на произвољну графичку библиотеку, конкретно *DirectFB*. Овај задатак је сачињен од две целине, које је потребно испунити. У првој целини неопходно је развити тестну апликацију за приказивање графичког садржаја користећи *DirectFB* библиотеку. У другој целини потребно је прилагодити интернет прегледач, то јесте реализовати *Ozone* интеграцију, да користи *DirectFB* библиотеку за приказ садржаја на екран. На тај начин, пружено је решење које омогућава да *Chromium*-у да ради ефикасно на уређајима са ограниченим графичким ресурсима.

Мотивација за овај пројекат произилази из потребе за оптимизацијом перформанси интернет прегледача на уграђеним системима, који често немају графичку обрадну моћ традиционалних десктоп окружења. *DirectFB* нуди ефикасан механизам за руковање графиком, што га чини идеалним кандидатом за такве уређаје. Прилагођавање *Chromium*-а да користи *DirectFB* може значајно побољшати перформансе рендеровања и корисничко искуство на овим платформама, отварајући пут за способније уграђене интернет апликације бржег одзива.

Рад се састоји из 7 поглавља:

- У овом поглављу је направљен увод у рад.
- У другом поглављу се налазе теоријске основе. Састоји се од интернет прегледача и *Chromium*-а.
- У трећем поглављу се налази концепт решења, као план решавања проблема.
- У четвртном поглављу је сама имплементација решења.
- У петом поглављу су резултати имплементације.
- У шестом поглављу се налази закључак са идејом о даљим корацима.
- У седмом поглављу је списак литература.

2. Теоријске основе

У овом поглављу ће бити представљени теоријски описи кључних компонената рада.

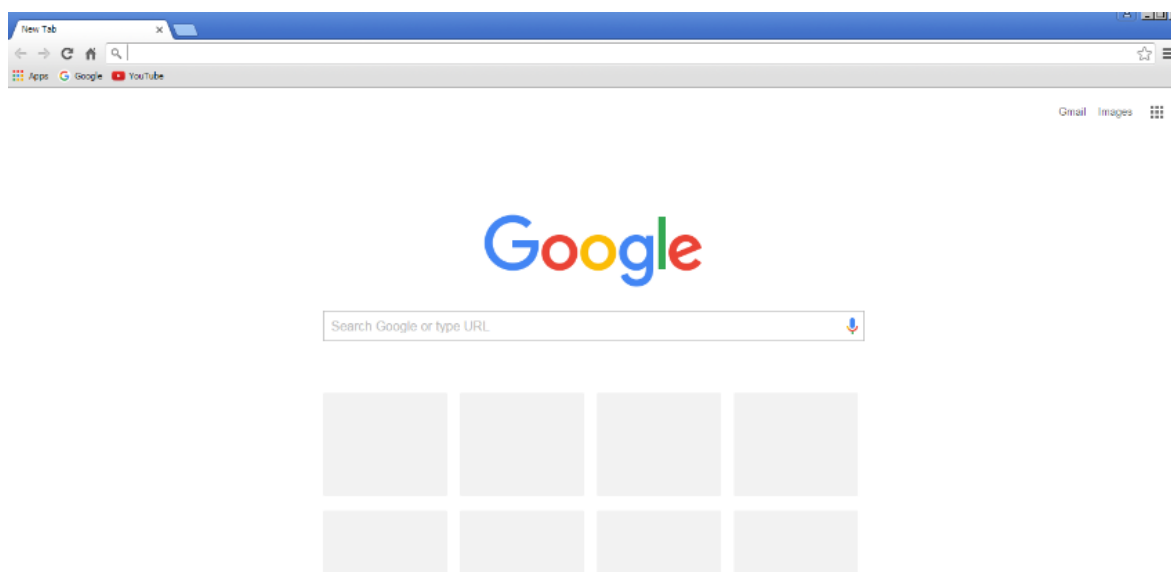
2.1 Интернет прегледачи

Интернет прегледачи су основне софтверске апликације које корисницима омогућавају приступ, преузимање и интеракцију са садржајем на *World Wide Web*-у. Они служе као интерфејс између корисника и огромних ресурса доступних на мрежи.

Главна функција интернет прегледача јесте приказивање интернет ресурса, захтевајући га од сервера, у интернет прозору. Ресурс је најчешће *HTML* документ. Лоциран је на месту означеном *URL* адресом.

Кориснички интерфејси прегледача имају заједничке елементе као што су (Слика 1):

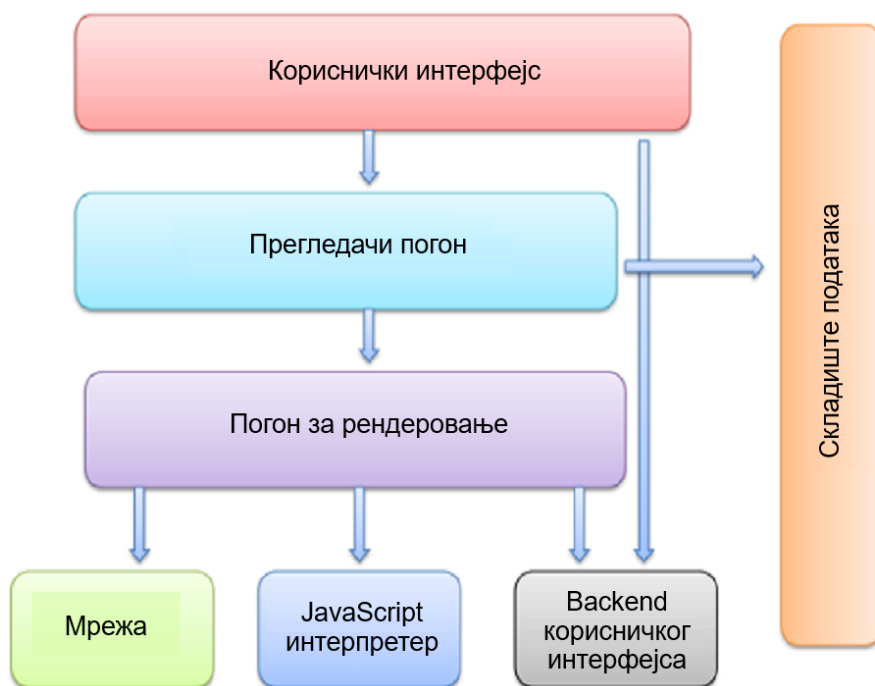
- адресна трака за уношење *URL*-а,
- напред и назад дугме,
- обележавање страница,
- дугме за освежавање и дугме за заустављање учитавање тренутачног садржаја и
- дугме за почетну страницу.



Слика 1 Кориснички интерфејс интернет прегледача

Основне компоненте прегледача су (Слика 2):

1. Кориснички интерфејс: ово укључује адресну траку, дугме назад и напред, мени за обележавање и слично. Сваки део приказа прегледача осим прозора у ком се види тражена страница.
2. Прегледачки погон: управља радњама између корисничког интерфејса и погон за рендеровање.
3. Погон за рендеровање: одговоран за приказивање траженог садржаја. На пример, ако је тражени садржај *HTML*, погон за рендеровање парсира *HTML* и *CSS*, и приказује парсирани садржај на екрану.
4. Мрежа: за мрежне позиве као што су *HTTP* захтеви, користећи различите имплементације за различите платформе иза платформски независног интерфејса.
5. *Backend* корисничког интерфејса: користи се за цртање основних додатака као што су комбо пакети и прозори. Овај *backend* излаже генерализовани интерфејс који није специфичан за платформу. Испод тога користи методе корисничког интерфејса оперативног система.
6. *JavaScript* интерпретер: користи се за парсирање и извршавање *JavaScript* кода.
7. Складиште података: ово је слој за презентацију. Прегледач може да сачува разне врсте података локално, као што су колачићи. Прегледачи такође подржавају механизме складиштења као што су *localStorage*, *IndexedDB*, *WebSQL* и *FileSystem*.



Слика 2 Основне компоненте прегледача

Најпознатији интернет прегледачи укључују *Google Chrome*, *Mozilla Firefox*, *Microsoft Edge*, *Apple Safari* и *Opera*. Сваки од ових прегледача заузео је значајан удео на тржишту захваљујући својим јединственим карактеристикама, перформансама и корисничким интерфејсима. *Google Chrome* се посебно истиче као један од најпопуларнијих интернет прегледача на свету. Изграђен је на *Chromium* пројекту отвореног кода, који служи као основа за неколико других прегледача.

2.2 Chromium

Chromium је пројекат интернет прегледача отвореног кода који служи као основа за многе популарне интернет прегледаче, укључујући *Google Chrome*, *Microsoft Edge* и *Brave*. Развио га је и одржава га пројекат *Chromium*, који води *Google*, али укључује доприносе различитих појединаца и организација.

У својој сржи, *Chromium* је прегледач који је дизајниран да приказује интернет странице и извршава задатке у вези са интернетом. Обезбеђује основне компоненте неопходне за приказивање *web* локација, руковање корисничким интеракцијама и управљање различитим *web* технологијама. Механизам прегледача у *Chromium*-у је познат као *Blink*, који је настао од *WebKit* погона 2013.

Један од кључних аспеката *Chromium*-а је његова природа отвореног кода, што значи да је његов изворни код слободно доступан јавности. Ово омогућава програмерима из целог света да прегледају, модификују и допринесу пројекту, промовишући транспарентност и заједнички развој. Природа *Chromium* отвореног кода довела је до стварања бројних пројеката прегледача који користе његову базу кода.

Chromium је изграђен на вишепроцесној архитектури, где се различитим задацима рукују различити процеси како би се побољшала стабилност и безбедност. Свака картица у прегледачу ради у оквиру сопственог заштићеног процеса, обезбеђујући изолацију између интернет страница и спречавајући злонамерни код да утиче на цео прегледач или оперативни систем.

Поред основног прегледача, *Chromium* пружа широк спектар функција и функционалност. Подржава различите *web* стандарде и технологије, укључујући *HTML*, *CSS*, *JavaScript* и *WebGL*, омогућавајући модерним *web* апликацијама да раде без проблема. *Chromium* такође укључује *JavaScript* погон под називом *V8*, који је познат по својој брзини и ефикасности у извршавању *JavaScript* кода.

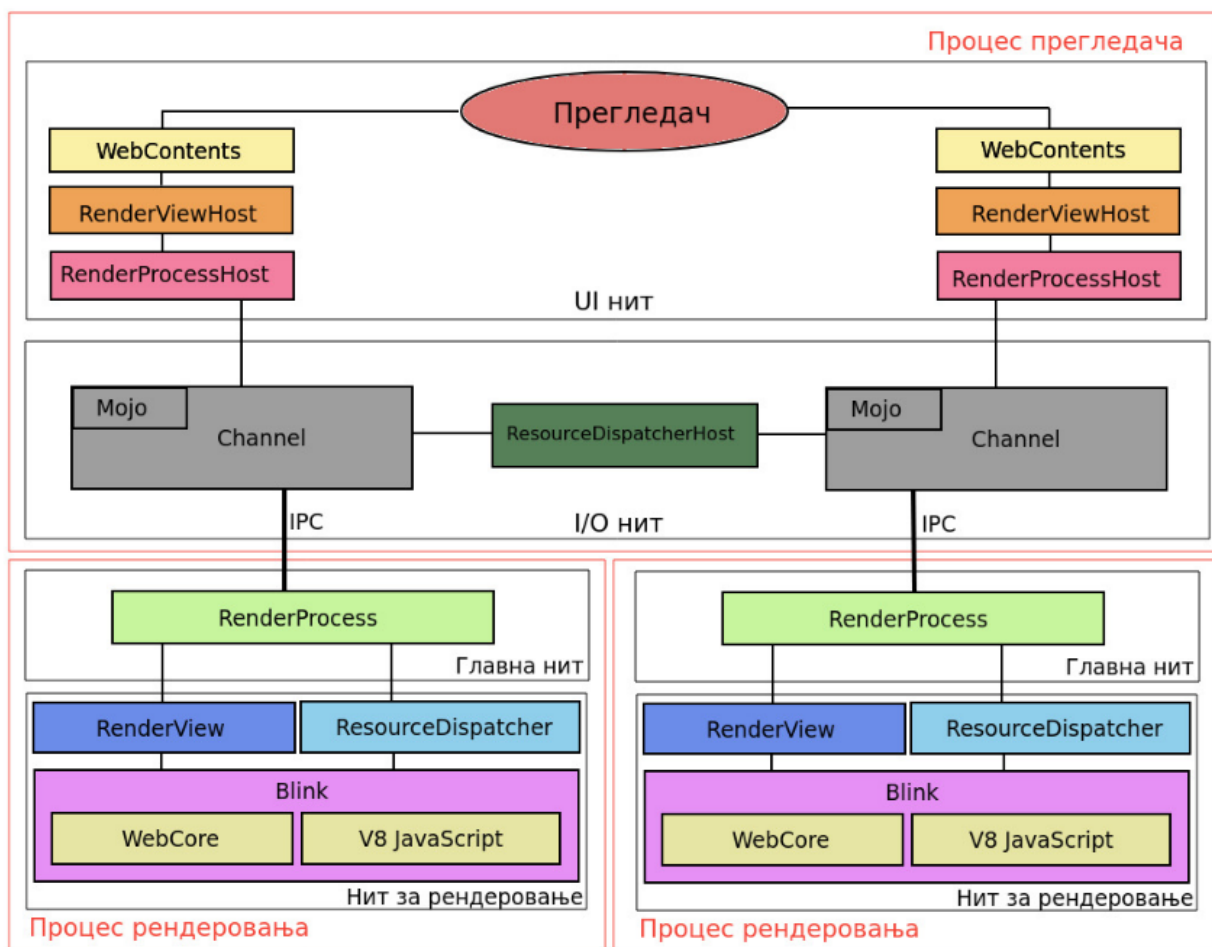
Chromium развојна заједница активно одржава и ажурира пројекат. Програмери доприносе пројекту тако што подносе извештаје о грешкама, закрпе и нове функције. Редовна ажурирања се објављују да би се решили безбедносни пропусти, побољшале перформансе и представиле нове функције.

Google Chrome, најпознатији прегледач заснован на *Chromium*-у, додаје додатне функције и услуге поврх *Chromium* базе. Ове функције често укључују синхронизацију са *Google* налозима, уграђени *PDF* прегледач, подршку за власничке медијске кодексе и интеграцију са *Google* услугама као што су *Google* преводилац и *Google* безбедно прегледање.

Све у свему, *Chromium* је пројекат прегледача отвореног кода који пружа основу за неколико популарних интернет прегледача. Комбинује моћан прегледач, вишепроцесну архитектуру и подршку за интернет стандарде, омогућавајући безбедно искуство прегледања интернета богато функцијама.

2.2.1 Архитектура Chromium-а

Архитектура *Chromium*-а је дизајнирана са моделом више процеса, који се углавном састоје од процеса прегледача и више процеса рендеровања (Слика 3). Може постојати само један процес прегледача и више процеса рендеровања.

Слика 3 Приказ мултипроцесне архитектуре *Chromium*-а

Процес прегледача, често називан једноставно прегледач, је главни за управљање процесима рендеровања и приказ корисничког интерфејса (UI). Овај процес ради са две главне нити: *UI* нити и *I/O* нити. *UI* нит је примарно одговорна за приказивање интернет страница на екрану, док *I/O* нит рукује међупроцесном комуникацијом (*IPC*) између процеса прегледача и процеса рендеровања, као и мрежним комуникацијама.

Објекти *UI* нити:

- *RenderProcessHost*: Сваки *RenderProcessHost* се повезује са једним процесом рендеровања, одржавајући однос један-према-један. Он шаље поруке специфичне за приказ на *RenderViewHost* и обрађује друге поруке које нису специфичне за приказ. Кључан је за једноставно идентифицирање *RenderView*-а у комбинацији са *RenderView ID*-ом, јер више *RenderView*-ова може постојати унутар једног процеса рендеровања.
- *RenderViewHost*: *RenderViewHost* прима поруке специфичне за приказ од *RenderProcessHost*-а. Однос овде је један-према-више јер процес рендеровања може генерисати више *RenderView*-а. *RenderViewHost* је одговоран за навигационе команде, обраду догађаја уноса и приказивање интернет странице.
- *WebContents*: Овај објекат представља *tab* који приказује интернет страницу. Он управља приказивањем интернет садржаја унутар правоугаоног подручја.
- *Browser*: Овај објекат представља прозор прегледача и може садржавати више инстанци *WebContents*-а, од којих свака одговара различитим *tab*-овима.

Објекти I/O нити:

- *Channel*: Овај објекат олакшава комуникацију кроз *IPC* цеви, осигуравајући ефикасну комуникацију између процеса прегледача и процеса рендеровања.
- *ResourceDispatcherHost*: Делујући као чувар мрежних захтева, овај објекат управља свим мрежним комуникацијама. Процеси рендеровања морају послати своје мрежне захтеве *RenderProcessHost*-у, који из затим прослеђује *ResourceDispatcherHost*-у за извршавање.

Процес рендеровања, познат као *renderer*, је посвећен конструкцији и приказивању интернет страница. Одржава две примарне нити: главну нит и нит за рендеровање. Главна нит процеса рендеровања углавном управља међунитном комуникацијом, док нит за рендеровање рукује стварном конструкцијом интернет страница.

Објекти главне нити:

- *RenderProcess*: Овај објекат олакшава комуникацију између нити за рендеровање где се налази *RenderView*, и I/O нити процеса прегледача.

Објекти нити за рендеровање:

- *ResourceDispatcher*: Када интернет страница треба преузети са сервера, ослања се на *ResourceDispatcher* да проследи захтев процесу прегледача путем *IPC*-а, јер процес рендеровања нема директан приступ интернету.
- *Blink*: Ово је основни погон за рендеровање у *Chromium*-у, одговоран за конструкцију *DOM*-а и распоред страница. *Blink* је *fork Webkit*-а и пружа боље могућности за оптимизацију перформанси, сигурности и развој нових функционалности. *Blink* се састоји од два главна дела: *WebCore*, који рукује основним функцијама распореда, и *V8*, где делује *JavaScript interpreter*.

Вишепроцесна архитектура *Chromium*-а ефикасно дели одговорности између процеса прегледача и више процеса рендеровања. Процес прегледача надзире приказ *UI*-а и управљање процесима, док се процеси рендеровања баве детаљном конструкцијом интернет страница. Ова подела рада побољшава перформансе и сигурност, осигуравајући стабилно и брзо искуство прегледања.

3. Концепт решења

У циљу логичког поједностављења имплементације и лакшег праћења током израде, рад је подељен на три целине, односно на три фазе израде. Свака фаза представља логички заокружену целину. Детаљан опис програмског решења се налази у следећем поглављу.

У првом сегменту потребно је подесити окружење и *Chromium* пројекат *Linux* платформи.

У другој фази неопходно је упознавање са постојећом документацијом и њена анализа везано за *Chromium Web Browser* и *DirectFB* графичку библиотеку.

У трећој фази потребно је постојеће *DirectFB* примере наместити и прилагодити *Linux* платформи тако да се могу покренути приказујући садржај. С обзиром да *Linux Ubuntu* има проблем са покретањем *DirectFB* апликација, резултат треба да буде *DirectFB* апликација која шаље податке на диск као слику.

У другом делу треће фазе потребно је креирати *BUILD.gn* скелет за нови порт платформе и имплементирати неопходне класе *DirectFB* графичке библиотеке тако да *Chromium* ради користећи графичку библиотеку *DirectFB*.

3.1 Компоненте система

Рад се састоји из две целине. Прва је *DirectFB* апликација која шаље податке на диск као слику. Друга целина је *Chromium* пројекат који ради користећи графичку библиотеку *DirectFB*.

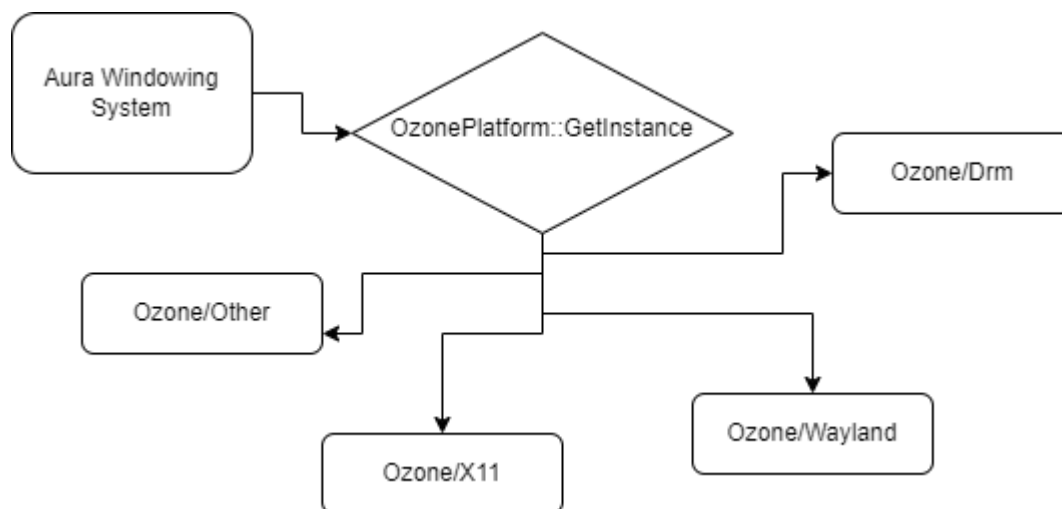
3.2 Платформа *Ozone*

Платформа *Ozone* је пројекат у оквиру *Chromium* екосистема који има за циљ да апстрахује основни код који је специфичан за платформу за различите оперативне системе и прозорске системе. Ова апстракција омогућава *Chromium*-у да ради у различитим окружењима без потребе за поновним писањем великих делова кода за сваку платформу.

Ozone обезбеђује скуп *API*-ја и апстракција које омогућавају *Chromium*-у да комуницира са основним прозорским системом и графичким стеком различитих платформи. Ово олакшава одржавање и развој *Chromium*-а за више оперативних система, као што су *Linux*, *Windows* и *MacOS*.

Пројекат *Ozone* је посебно вредан за пренос *Chromium*-а на нове платформе или прилагођавање за рад у окружењима са специфичним захтевима. Помаже да основна

Chromium кодна база буде чиста и фокусирана на основне функције прегледача, док се детаљима специфичним за платформу рукује одвојено преко *Ozone* платформе. *Ozone* је само једна од многих компоненти које чине пројекат, доприносећи његовој флексибилности и могућностима на више платформама.



Слика 4 Вишеструки бекендови *Ozone* платформе

Ozone платформа има следеће карактеристике:

1. Апстракција платформе: *Ozone* обезбеђује слој апстракције који изолује код специфичан за платформу од остатка *Chromium*-ове кодне базе. Ова апстракција омогућава да се *Chromium* лако прилагоди и покрене на различитим оперативним системима, прозорским системима и графичким стековима без већих промена кода.
2. Вишеструки бекендови (Слика 4): *Ozone* подржава различите позадине, од којих сваки циља на одређену платформу. На пример, постоје одвојена позадина *Ozone* за платформе као што су *X11 (Linux)*, *Wayland (Linux)* и *Headless* (без графичког интерфејса). Ови бекендови дефинишу како *Chromium* реагује са основним прозорским системом, улазним уређајима и другим ресурсима специфичним за платформу.
3. Интеграција графичког стека: *Ozone* се повезује са графичким стеком основне платформе, омогућавајући *Chromium*-у да ефикасно рендерује и приказује садржај. Ова интеграција осигурава да је графички излаз прегледача оптимизован за могућности специфичне платформе.
4. Руковање уносом: *Ozone* управља догађајима уноса као што су унос мишем и тастатуром, интеракције са екраном осетљивим на додир и покрети. Ово обезбеђује да *Chromium* може да прима и обрађује уносе корисника на доследан начин на различитим платформама.
5. Управљање прозорима: *Ozone* управља креирањем прозора, манипулацијом и интеракцијом. Он обезбеђује да се *Chromium*-ови прозори понашају на одговарајући начин у контексту основног прозорског система.
6. Приступачност: *Ozone* такође узима у обзир функције приступачности и обезбеђује неопходне куке за алатке за приступачност за интеракцију са корисничким интерфејсом *Chromium*-а.
7. Изазови интеграције: *Ozone* иако поједностављује развој на више платформама, интеграција са различитим платформама може представљати изазове због разлика у *API*-јима, понашању и карактеристикама перформанси. *Ozone* има за циљ да минимизира ове изазов обезбеђивањем јединственог интерфејса.
8. Сарадња у заједници: као и друге компоненте *Chromium*-а, *Ozone* има користи од доприноса разноврсне заједнице програмера. Сарадници раде на побољшању

компатибилности *Ozone*-а са различитим платформама и на решавању проблема који произилазе из варијација специфичних за платформу.

9. Безбедност и стабилност: апстракција *Ozone*-а помаже да се осигура безбедност и стабилност изоловањем кода специфичног за платформу. Ово значи да је мања вероватноћа да ће проблеми или рањивости у вези са одређеном платформом утицати на целу *Chromium* кодну базу.

Све у свему, *Ozone* платформа је кључни део *Chromium*-ове архитектуре који му омогућава да одржи конзистентне перформансе и понашање у различитим оперативним системима уз прилагођавање јединствених аспеката сваке платформе. Он показује посвећеност пројекта отвореног кода модуларности, прилагодљивости и сарадњи.

3.3 Графичка библиотека *DirectFB*

DirectFB (*Direct Frame Buffer*) је софтверска библиотека са малим меморијским отиском која обезбеђује убрзање графике, руковање улазним уређајима и слој апстракције, као и интегрисани систем прозора са подршком за прозирне прозоре и више слојева приказа на врху *Linux* бафера оквира без потребе модификације језгра. *DirectFB* је бесплатан софтвер отвореног кода који подлеже условима *GNU* мање опште јавне лиценце (*GPL*).

Графичка библиотека *DirectFB* омогућава програмерима алтернативу пуном сервер *X Window System (X11)* који се користи у оперативним системима сличним *UNIX*-у. *DirectFB* омогућава апликацијама да комуницирају директно са видео хардвером преко директног *API*-ја, убрзавајући и поједностављујући графичке операције.

Често га користе програмери игара и уграђених система да би се заобили трошкови пуне имплементације сервера *X Window* система. Године 2006, верзија *DirectFB*-а укључена је у софтверски стог за Аудио/Видео графичку спецификацију *CE Linux foruma*.

DirectFB може да хостује *XDirectFB*, имплементацију *X* сервера без корена која користи *DirectFB* прозоре за прозоре највишег нивоа *X11*. *XDirectFB* је интерфејс који имитира *X11* интерфејс преко *DirectFB API*-ја да би поједноставио покретање апликација написаних за *X11* на *DirectFB*.

DirectFBGL је проширење за *OpenGL* за *DirectFB/XDirectFB* које користи *Direct Rendering Infrastructure (DRI)* у *Mesa 3D* библиотеци да подржи *OpenGL* хардверско убрзање.

Међу производима који користе *DirectFB* су *LinuxTV*, *Ben Nanonote*, необјављени *Palm Foleo* мобилни пратилац, *webOS* оперативни систем, *Panasonic Viera Connect* телевизори и *Philips* телевизори засновани на *jointSPACE*-у.

DirectFB је танка библиотека која обезбеђује хардверско убрзање графике, руковање улазним уређајима и апстракцију, интегрисани прозорски систем са подршком за прозирне прозоре и више слојева приказа, не само на врху *Linux Framebuffer* уређаја. То је потпуни слој апстракције хардвера са софтверским резервама за сваку графичку операцију коју основни хардвер не подржава, *DirectFB* додаје графичку снагу уграђеним системима и поставља нови стандард за графику под *Linuxom*.

DirectFB ради на врху *Frame Buffer Device*-а (*/dev/fb*) и користи хардверско убрзање са својим драјверима за чипсет. Такође ради на *VESA* баферу оквира.

DirectFB извршава само следеће задатке преко */dev/fb*:

- подешавање видео режима (резулација, дубина боје и времена),

- меморијско мапирање бафера оквира и *I/O* портова и
- промена области приказа (нпр. За двоструки бафер).

DirectFB користи сопствени управљачки програм чипсета да би користио хардверско убрзање преко меморијских мапираних *I/O* портова. Искључује функције убрзања драјвера бафера оквира.

3.4 *BUILD.gn*

GN систем за грађење је мета-систем за грађење који је развио *Google* првенствено за *Chromium* пројекат. Дизајниран је да генерише датотеке за грађење за *Ninja*, који је брз систем за грађење ниског нивоа. *GN* има за циљ да ефикасно рукује сложеношћу великих пројеката, пружајући баланс између флексибилности, брзине и лакоће коришћења.

BUILD.gn је конфигурациона датотека за грађење која се користи у *GN* систему за грађење. Примарна сврха *BUILD.gn* датотеке је дефинисање циљева грађења и управљање зависностима у пројекту. Она специфицира како би требало компајлирати и линковати изворни код да би се креирала финална извршна датотека или библиотека.

Кључне функционалности *BUILD.gn* укључују:

- Дефинисање циљева грађења – специфицира различите типове циљева грађења, као што су извршне датотеке, дељене библиотеке, статичке библиотеке и скупови извора.
- Управљање зависностима – управља зависностима између различитих компоненти пројеката, осигуравајући да свака компонента буде изграђена у исправном редоследу.
- Конфигурационе поставке – укључује разне конфигурационе поставке, као што су заставице компајлера, директоријум за укључивање и платформско-специфичне поставке, које контролишу процес компилације.

Типична *BUILD.gn* датотека организована је у неколико секција:

- Увози – користе се за укључивање других *GN* датотека или конфигурационих датотека,
- Конфигурације – дефинишу заједничке поставке које се могу применити на више циљева грађења,
- Циљеви – дефинишу стварна правила грађења за извршне датотеке, библиотеке и скупове извора.

4. Програмско решење

4.1 Тестна апликација за приказивање графичког садржаја користећи *DirectFB* библиотеку

Први задатак је био развити апликацију за приказивање графичког садржаја користећи *DirectFB* библиотеку. *Ubuntu* има проблем са покретањем *DirectFB* апликација, тако да би решење била *DirectFB* апликација која излазне податке чува на диску у виду слике.

На *Github*-у имамо примере једноставних *DirectFB* апликација које се могу користити за демонстрацију и тестирање различитих *DirectFB* функција, као што су: анимација пингвина, програм за тестирање функција графичког драјвера, пример матрице трансформације, пример мапирања текстуре, пример стека прозора и многе друге. У сврху решавања задатка узет је један пример, а то је *df_window*, пример стека прозора. Овај *DirectFB* пример показује како креирати, управљати и комуницирати са више прозора користећи *DirectFB* графичку библиотеку, руковањем догађајима за имплементацију основних функција прозора.

Као у свим примерима и у овом примеру кориштена је макро дефиниција приказана на слици (Слика 5). Овај макро је услужни програм за проверу повратне вредности *DirectFB* функција. Ако резултат није *DFB_OK*, избацује се порука о грешци.

```
/* macro for a safe call to DirectFB functions */
#define DFBCHECK(x)
do {
    DFBResult ret = x;
    if (ret != DFB_OK) {
        fprintf( stderr, "%s <%d>:\n\t", __FILE__, __LINE__ );
        DirectFBErrorFatal( #x, ret );
    }
} while (0)
```

Слика 5 Приказ макро дефиниције

За чување излазних података на диску у виду слике кориштена је функција *Dump()* (Слика 6). Функција *Dump()* је алатка за отклањање грешака која штампа информације о наведеној површини као што су њена величина, формат и други релевантни детаљи. Обично се користи током развоја за инспекцију и дијагнозу стања површина у графичком

корисничком интерфејсу. Тачан излаз функције *Dump()* зависи од имплементације *DirectFB* библиотеке.

```
DFBCHECK(layer->CreateWindow( layer, &wdsc, &window1 ));

DFBCHECK(dfb->CreateImageProvider( dfb, DATADIR"/dfblogo.dff", &provider ));
DFBCHECK(window1->GetSurface( window1, &window_surface1 ));
provider->RenderTo( provider, window_surface1, NULL );
window_surface1->SetDrawingFlags( window_surface1, DSDRAW_SRC_PREMULTIPLY );
window_surface1->SetColor( window_surface1, 0xFF, 0x20, 0x20, 0x90 );
window_surface1->DrawRectangle( window_surface1, 0, 0, wdsc.width, wdsc.height );
provider->Release( provider );

DFBCHECK(dfb->CreateImageProvider( dfb, DATADIR"/cursor_red.dff", &provider ));
provider->GetSurfaceDescription( provider, &sdsc );
DFBCHECK(dfb->CreateSurface( dfb, &sdsc, &cursor_surface1 ));
provider->RenderTo( provider, cursor_surface1, NULL );
DFBCHECK(window1->SetCursorFlags( window1, DWCF_NONE ));
DFBCHECK(window1->SetCursorShape( window1, cursor_surface1, 0, 0 ));
window_surface1->Dump(window_surface1, "Milica", NULL);
provider->Release( provider );
```

Слика 6 Приказ функције *Dump()*

4.2 Прилагођавање интернет прегледача да користи *DirectFB* библиотеку за приказ садржаја на екран

Ова целина детаљно описује имплементацију прилагођавања *Chromium* интернет прегледача за коришћење *DirectFB*-а кроз *Ozone* слој апстракције платформе.

Након подешеног развојног окружења први корак је био креирање скелета *BUILD.gn* за нови платформски порт (Слика 7). Ово је укључивало:

- Увоз конфигурационог фајла *ui_mode.gni*, кључног за *ChromeOS* специфичне *UI* поставке,
- Ограничавање видљивости на датотеке унутар одређене путање,
- Функција *assert* проверава да се грађење врши на *Linux* платформи,
- Дефинисање сетова извора за конкретно *DirectFB* код, који укључује листу изворних фајлова, зависности и неопходне библиотеке.

```

1  import("//build/config/chromeos/ui_mode.gni")
2
3  visibility = [ "ui/ozone/**" ]
4
5  assert(is_linux)
6
7  source_set("directfb"){
8    sources = [
9      "directfb_connector.cc",
10     "directfb_connector.h",
11     "directfb_platform_window.cc",
12     "directfb_platform_window.h",
13     "directfb_surface.h",
14     "directfb_surface.cc",
15     "directfb_surface_factory.cc",
16     "directfb_surface_factory.h",
17     "directfb_platform_screen.cc",
18     "directfb_platform_screen.h",
19     "ozone_platform_directfb.cc",
20     "ozone_platform_directfb.h",
21     "client_native_pixmap_factory_directfb.cc",
22     "client_native_pixmap_factory_directfb.h",
23   ]
24
25   defines = [ "OZONE_IMPLEMENTATION" ]
26
27   deps = [
28     "//base",
29     "ui/base/cursor",
30     "ui/base/ime",
31     "ui/gfx",
32     "ui/gfx/geometry",
33     "ui/platform_window",
34     "ui/platform_window/common",
35     "ui/platform_window/wm",
36     "ui/ozone/common",
37     "ui/ozone:ozone_base",
38   ]
39
40   libs = [
41     "directfb",
42   ]
43
44   lib_dirs = [
45     "/usr/local/lib",
46   ]
47
48   include_dirs = [
49     "/usr/local/include/directfb",
50   ]
51
52   cflags = [
53     "-Wno-register",
54   ]
55 ]
56
57 }

```

Слика 7 *Build.gn* конфигурациони фајл *DirectFB* интеграције

Срж софтверског решења укључивала је имплементацију компоненти специфичних за *DirectFB*:

- *DirectFBConnector*,
- *DirectFBPlatformWindow*,
- *DirectFBPlatformScreen*,
- *DirectFBSurface*,
- *DirectFBSurfaceFactory*,
- *GLOzoneEGLDFB*,
- *OzonePlatformDirectFB*.

DirectFBConnector класа служи као синглтон одговоран за иницијализацију и деиницијализацију *DirectFB* оквира. Управља главним *DirectFB* интерфејсом, слојем приказа, креирањем прозора и компонентама за руковање догађајима. Ова класа прати синглтон образац како би се осигурало да постоји једна инстанца одговорна за одржавање *DirectFB* стања током животног циклуса апликације. Она обухвата заплетености иницијализације *DirectFB*-а, креирања прозора и управљања ресурсима, пружајући чист и организован интерфејс за остатак апликације за интеракцију са *DirectFB* оквиrom.

Метода иницијализације подешава *DirectFB* систем иницијализацијом *DirectFB*-а, креирањем главног *DirectFB* објекта и добијањем примарног слоја приказа (Слика 8). Такође позива функцију *CreateWindow()* да постави *DirectFB* прозор.

```
44 void DirectFBConnector::Initialize(){
45     // Initialize the DirectFB system
46     if(is_init_)
47         return;
48
49     DFBCHECK(DirectFBInit(nullptr, nullptr));
50
51     // Create a DirectFB object
52     DFBCHECK(DirectFBCreate(&dfb_));
53
54     // The default <i>layer_id</i> is DLID_PRIMARY.
55     DFBCHECK(dfb_->GetDisplayLayer(dfb_, DLID_PRIMARY, &layer_));
56
57     DFBCHECK(layer_->SetCooperativeLevel( layer_, DLSCL_ADMINISTRATIVE ));
58
59     CreateWindow();
60
61     is_init_=true;
62 }
```

Слика 8 Приказ методе иницијализације

Метода *CreateWindow()* конфигурише и креира *DirectFB* прозор са одређеним атрибутима као што су величина и позиција. Обрађује аргументе командне линије да би динамички поставио величину и позицију прозора, осигуравајући флексибилност и прилагодљивост. Креира прозор и његову површину и поставља непрозирност прозора. Овај процес осигурава да је *DirectFB* прозор исправно конфигурисан и спреман за коришћење (Слика 9).

```

void DirectFBConnector::CreateWindow() {

    // Set the flags for window creation
    wdsc_.flags = DFBSurfaceDescriptionFlags(DWDESC_WIDTH | DWDESC_HEIGHT | DWDESC_POSX | DWDESC_POSY);

    const base::CommandLine& command_line = *base::CommandLine::ForCurrentProcess();

    if (!command_line.HasSwitch(switches::kNativeWindowSize)) {
        base::CommandLine::ForCurrentProcess()->AppendSwitchASCII(switches::kNativeWindowSize, "800,600");
    }

    if (!command_line.HasSwitch(switches::kNativeWindowPosition)) {
        base::CommandLine::ForCurrentProcess()->AppendSwitchASCII(switches::kNativeWindowPosition, "0,0");
    }

    if (command_line.HasSwitch(switches::kNativeWindowSize)) {
        int width, height;

        std::string window_size = command_line.GetSwitchValueASCII(switches::kNativeWindowSize);
        if (ParseWindowSize(window_size, &width, &height)) {
            wdsc_.width=width;
            wdsc_.height=height;
        }
    }

    if (command_line.HasSwitch(switches::kNativeWindowPosition)) {
        int posx, posy;

        std::string window_position = command_line.GetSwitchValueASCII(switches::kNativeWindowPosition);
        if (ParseWindowSize(window_position, &posx, &posy)) {
            wdsc_.posx=posx;
            wdsc_.posy=posy;
        }
    }

    // Create a window
    DFBCHECK(layer_->CreateWindow(layer_, &wdsc_, &window_));

    // Create a surface for the window
    DFBCHECK(window_->GetSurface(window_, &primary_));
    primary_->GetSize(primary_, &width_, &height_);
    bitmap_size_.SetSize(width_, height_);

    // Offscreen surface
    dsc_.flags = DFBSurfaceDescriptionFlags(DSDESC_CAPS | DSDESC_WIDTH | DSDESC_HEIGHT | DSDESC_PIXELFORMAT);
    dsc_.caps = DFBSurfaceCapabilities(DSCAPS_PREMULTIPLIED);
    dsc_.width = bitmap_size_.width();
    dsc_.height = bitmap_size_.height();
    dsc_.pixelformat = DSPF_ARGB;
    dfb->CreateSurface(dfb_, &dsc_, &offscreen_);
    offscreen_->Clear(offscreen_, 0, 0, 0, 0);
    offscreen_->Lock(offscreen_, DSLF_WRITE, (void**)&pointer_, &pitch_);
    offscreen_->Unlock(offscreen_);

    // Set the opacity of the window
    window_->SetOpacity(window_, opacity_);

    // Set the input device to the default device
    DFBCHECK(dfb_->GetInputDevice(dfb_, DIDID_KEYBOARD, &input_device_));

    // Create an event buffer
    DFBCHECK(window_->CreateEventBuffer(window_, &event_buffer_));
}

```

Слика 9 Приказ функције *CreateWindow()*

Класа укључује неколико корисничких метода које манипулишу видљивошћу прозора, непрозирношћу и ажурирају његов садржај. Методе *Hide()* и *Show()* прилагођавају непрозирност прозора да би га сакрили или приказали. Метода *Update()* освежава садржај прозора преврћући примарне површине, осигуравајући да се приказ ажурира с најновијим

садржајем. Додатно, метода *SaveAsImage()* снима тренутни садржај прозора као слику у фајл.

DirectFBPlatformWindow класа обухвата функционалност која се односи на управљање прозором платформе унутар *DirectFB* окружења (Слика 10). Повезује се са *DirectFB* конектором и управља различитим операцијама у вези са прозором, као што су приказивање, скривање, промена величине и снимање уноса. Ова класа игра централну улогу у управљању корисничким интерфејсом, делујући као мост између прозорске логике апликације и *DirectFB* платформе. Такође управља конверзијом између пиксела независних од уређаја (*DIP*) и пиксела, служећи се разноврсним екранима које *DirectFB* подржава.

```
DirectFBPlatformWindow::DirectFBPlatformWindow(PlatformWindowDelegate* delegate,
                                                DirectFBConnector* connection,
                                                PlatformWindowInitProperties properties): connection_(connection),
                                                delegate_(delegate),
                                                bounds_(delegate->ConvertRectToPixels(properties.bounds)),
                                                weak_ptr_factory_(this) {
    window_id_ = (intptr_t)connection->GetWindow();
    delegate->OnAcceleratedWidgetAvailable(window_id_);
}

DirectFBPlatformWindow::~DirectFBPlatformWindow(){
    delete connection_;
}

void DirectFBPlatformWindow::Show(bool inactive){
    if(connection_){
        connection_->Show();
    }
}

void DirectFBPlatformWindow::Hide(){
    if(connection_){
        connection_->Hide();
    }
}

void DirectFBPlatformWindow::SetOpacity(float opacity){
    if(connection_){
        connection_->SetOpacity(opacity);
    }
}
```

Слика 10 Приказ класе *DirectFBPlatformWindow*

DirectFBPlatformScreen класа игра кључну улогу у графичком подсистему *DirectFB* окружења (Слика 11). Она служи као интерфејс за управљање и одржавање информација повезаних са приказом (*display*) унутар платформе. Ова класа је од суштинске важности за руковање више екрана и добијање информација о примарном екрану.

```
DirectFBPlatformScreen::DirectFBPlatformScreen() : displays_({display::Display::GetDefaultDisplay()}) {}
DirectFBPlatformScreen::~DirectFBPlatformScreen() {}

const std::vector<display::Display>& DirectFBPlatformScreen::GetAllDisplays() const {
    return displays_;
}
```

Слика 11 Приказ класе *DirectFBPlatformScreen*

DirectFBSurface класа обухвата функционалност која се односи на креирање и управљање површином за цртање помоћу *DirectFB* графичке библиотеке. Ова класа служи као мост између *DirectFB* конектора и графичког корисничког интерфејса. Она је одговорна за руковање креирањем и променом величине платна за цртање, представљање платна на екрану и управљање *VSync* провајдером. Класа се ослања на *Skia* графичку

библиотеку за руковање основним графичким операцијама, обезбеђујући слој апстракције за беспрекорну интеграцију са *DirectFB* оквиром.

Ова класа подржава следеће функције (Слика 12):

- *GetCanvas* – преузима тренутно *Skia* платно повезано са површином,
- *ResizeCanvas* – мења величину платна површине на одређену величину приказа,
- *PresentCanvas* – приказује садржај платна на екрану и позива методу *Update()* како би приказао ажурирани садржај.

```
SkCanvas* DirectFBSurface::GetCanvas() {
    if(!surface_){
        return nullptr;
    }
    return surface_>getCanvas();
}

// Resizes the canvas of the surface.
// Updates the SkSurface with the new size.
// Creates a new SkSurface with the provided SkImageInfo, pixel buffer, and row bytes.
void DirectFBSurface::ResizeCanvas(const gfx::Size& viewport_size, float scale) {
    int width = connection_>bitmap_size().width();
    int height = connection_>bitmap_size().height();

    SkImageInfo info = SkImageInfo::Make(width, height, kN32_SkColorType, kPremul_SkAlphaType);
    surface_ = SkSurface::MakeRasterDirect(info, connection_>pixel(), info.minRowBytes());
}

// Presents the canvas to the screen.
// Updates the DirectFBConnector to display the updated content
void DirectFBSurface::PresentCanvas(const gfx::Rect& damage){
    connection_>Update();
}
```

Слика 12 Приказ функционалности класе *DirectFBSurface*

DirectFBSurfaceFactory класа делује као фабрика за креирање инстанци класе *DirectFBSurface*. Она је одговорна за иницијализацију неопходних компоненти, као што је *DirectFB* конектор, и обезбеђивање јединственог интерфејса за креирање површина за цртање. Класа такође управља интеграцијом *OpenGL*-а кроз класу *GLOzoneEGLDFB*, обезбеђујући да се одговарајућа *OpenGL* површина креира на основу наведених захтева. Ова фабрички образац побољшава проширивост кодне базе, омогућавајући лаку интеграцију нових стратегија креирања површина у будућности.

Ова класа подржава следеће функције (Слика 13):

- *GetAllowedGLImplementations* – враћа вектор дозвољених *GL* имплементација, а у овом случају дозвољена је само *EGL* базирана *GLES2* имплементација,
- *GetGLOzone* – враћа одговарајућу *GLOzone* имплементацију на основу пружене *GL* имплементације, у овом случају инстанцу *GLOzoneEGLDFB*,
- *GetAllowedGLImplementations* – креира *SurfaceOzoneCanvas* објекат за одређени додатак (*widget*) и уколико постоји конекција враћа јединствени показивач на *DirectFBSurface* објекат.

```

// Returns a vector of allowed GL implementations.
// In this case, it only allows EGL-based GLES2 implementation.
std::vector<gl::GLImplementationParts> DirectFBSurfaceFactory::GetAllowedGLImplementations() {
    return std::vector<gl::GLImplementationParts>{
        gl::GLImplementationParts(gl::kGLImplementationEGLGLES2),
    };
}

// Returns the GLOzone implementation based on the given GL implementation.
// In this case, it returns the GLOzoneEGLDFB instance for EGL-based GLES2 implementation.
GLOzone* DirectFBSurfaceFactory::GetGLOzone(const gl::GLImplementationParts& implementation) {
    switch (implementation.gl) {
        case gl::kGLImplementationEGLGLES2:
            return egl_implementation_.get();
        default:
            return nullptr;
    }
}

// Creates a SurfaceOzoneCanvas object for the specified widget.
// It returns a unique_ptr to the DirectFBSurface object if a connection exists.
std::unique_ptr<SurfaceOzoneCanvas> DirectFBSurfaceFactory::CreateCanvasForWidget(gfx::AcceleratedWidget widget) {
    if (!connection_)
        return nullptr;

    return std::make_unique<DirectFBSurface>(connection_>GetInstance());
}

```

Слика 13 Приказ функционалности класе *DirectFBSurfaceFactory*

Класа *GLOzoneEGLDFB* је специјализација класе *GLOzoneEGL*, дизајнирана да олакша интеграцију *OpenGL* рендеровања са *DirectFB* платформом. Ова класа замењује методе за креирање *OpenGL* површина за приказ и ван екрана, обезбеђујући компатибилност са *DirectFB* окружењем. Имплементацијом неопходних везива и руковања за *EGL* и *OpenGL*, омогућава ефикасно приказивање унутар *DirectFB* контекста. Класа игра кључну улогу у омогућавању хардверски убрзане графике на системима који подржавају *DirectFB*.

GLOzoneEGLDFB класа је угнеждена у оквиру имплементације *DirectFBSurfaceFactory*. Она проширује *GLOzoneEGL* и пружа методе за креирање *OpenGL* површина (Слика 14):

- *CreateViewGLSurface* – креира *GLSurface* за рендеровање на приказу,
- *CreateOffscreenGLSurface* – креира *offscreen GLSurface* за рендеровање,
- *GetNativeDisplay* – враћа приказ платформе којим се управља,
- *LoadGLES2Bindings* – поставља *GL* везе за површину.

```

class GLOzoneEGLDFB : public GLOzoneEGL{
public:
    GLOzoneEGLDFB() {}
    ~GLOzoneEGLDFB() override {}
    GLOzoneEGLDFB(const GLOzoneEGLDFB&) = delete;
    GLOzoneEGLDFB& operator=(const GLOzoneEGLDFB&) = delete;

    // GLOzone:
    // Creates a GLSurface for rendering on the view.
    scoped_refptr<gl::GLSurface> CreateViewGLSurface(gl::GLDisplay* display, gfx::AcceleratedWidget window) override{
        return gl::InitializeGLSurface(new gl::NativeViewGLSurfaceEGL(display->GetAs<gl::GLDisplayEGL>(), window, nullptr));
    }

    // Creates an offscreen GLSurface for rendering.
    scoped_refptr<gl::GLSurface> CreateOffscreenGLSurface(gl::GLDisplay* display, const gfx::Size& size) override{
        gl::GLDisplayEGL* egl_display = display->GetAs<gl::GLDisplayEGL>();
        if (egl_display->IsEGLSurfacelessContextSupported() && size.width() == 0 &&
            size.height() == 0) {
            return gl::InitializeGLSurface(new gl::SurfacelessEGL(egl_display, size));
        } else {
            return gl::InitializeGLSurface(new gl::PbufferGLSurfaceEGL(egl_display, size));
        }
    }

protected:
    // Returns native platform display handle and platform type as per
    // EGL platform extensions.
    // This is used to obtain the EGL display connection for the native display.
    gl::EGLDisplayPlatform GetNativeDisplay() override {
        return gl::EGLDisplayPlatform(EGL_DEFAULT_DISPLAY);
    }

    // Sets up GL bindings for the native surface.
    bool LoadGLS2Bindings(const gl::GLImplementationParts& implementation) override {
        return LoadDefaultEGLGLS2Bindings(implementation);
    }
};

```

Слика 14 Приказ класе *GLOzoneEGLDFB*

Класа *OzonePlatformDirectFB* служи као специјализована имплементација *Ozone* платформе, прилагођена конкретно за *DirectFB* окружење. Интегрише различите компоненте неопходне за управљање прозорима, површинама, методама уноса и приказима, чиме омогућава беспрекорну графичку и улазни интерфејс за апликације.

OzonePlatformDirectFB класа дизајнирана је као синглтон, осигуравајући да постоји само једна инстанца током животног века апликације (Слика 15). Овај дизајн је кључан за централизовано управљање ресурсима и доследно одржавање стања широм апликације. Синглтон шаблон се спроводи постављањем инстанце у конструктору и спречавањем копирања конструкције и доделе.

```

// Singleton OzonePlatform implementation for Directfb platform.
class OzonePlatformDirectfb: public OzonePlatform, public OSEExchangeDataProviderFactoryOzone {
public:
    OzonePlatformDirectfb(){
        SetInstance(this);
    }
};

```

Слика 15 Приказ синглтон шаблона

Класа подржава две примарне методе за иницијализацију:

- *InitializeUI()* и
- *InitializeGPU()*.

Метода *InitializeUI()* поставља компоненте корисничког интерфејса, као што су механизам за распоред тастатуре, везу са екраном, менаџер прекривача (*overlay manager*) и фабрику курсора (Слика 16). Он је од суштинске важности за припрему платформе за руковање операцијама корисничког интерфејса.

```
bool InitializeUI(const InitParams& params) override {
    keyboard_layout_engine_ = std::make_unique<StubKeyboardLayoutEngine>();
    KeyboardLayoutEngineManager::SetKeyboardLayoutEngine(keyboard_layout_engine_.get());
    connection_ = DirectFBConnector::GetInstance();
    overlay_manager_ = std::make_unique<StubOverlayManager>();
    cursor_factory_ = std::make_unique<BitmapCursorFactory>();
    gpu_platform_support_host_.reset(CreateStubGpuPlatformSupportHost());

    return true;
}
```

Слика 16 Приказ функције *InitializeUI()*

InitializeGPU метод иницијализацију компоненте повезане са графичком јединицом (*GPU*), посебно *SurfaceFactory*, која је кључна за операције рендеровања (Слика 17).

```
void InitializeGPU(const InitParams& params) override {
    surface_factory_ = std::make_unique<DirectFBSurfaceFactory>(connection_);
}
```

Слика 17 Приказ функције *InitializeGPU()*

Ове класе заједно чине окосницу графичког и подсистема корисничког интерфејса дизајнираног за интеграцију са *DirectFB* платформом. Свака класа има специфичну улогу, доприносећи укупној функционалности и модуларности система. Употреба шаблона дизајна, као што су фабрички образац и синглтон образац, побољшава организацију кода и могућност одржавања. Интеграција графичких библиотека као што су *Skia* и *OpenGL* показује свестраност и проширивост система за руковање различитим графичким захтевима.

5. Резултати

5.1 Тестна апликација за приказивање графичког садржаја користећи *DirectFB* библиотеку

С обзиром да *Ubuntu* има проблем са покретањем *DirectFB* апликација, решење је била *DirectFB* апликација која излазне податке чува на диску у виду слике.

За резултат смо требали добити слику са *DirectFB* логом, што се и види на слици (Слика 18).



Слика 18 Приказ резултата при покретању *DirectFB* апликације

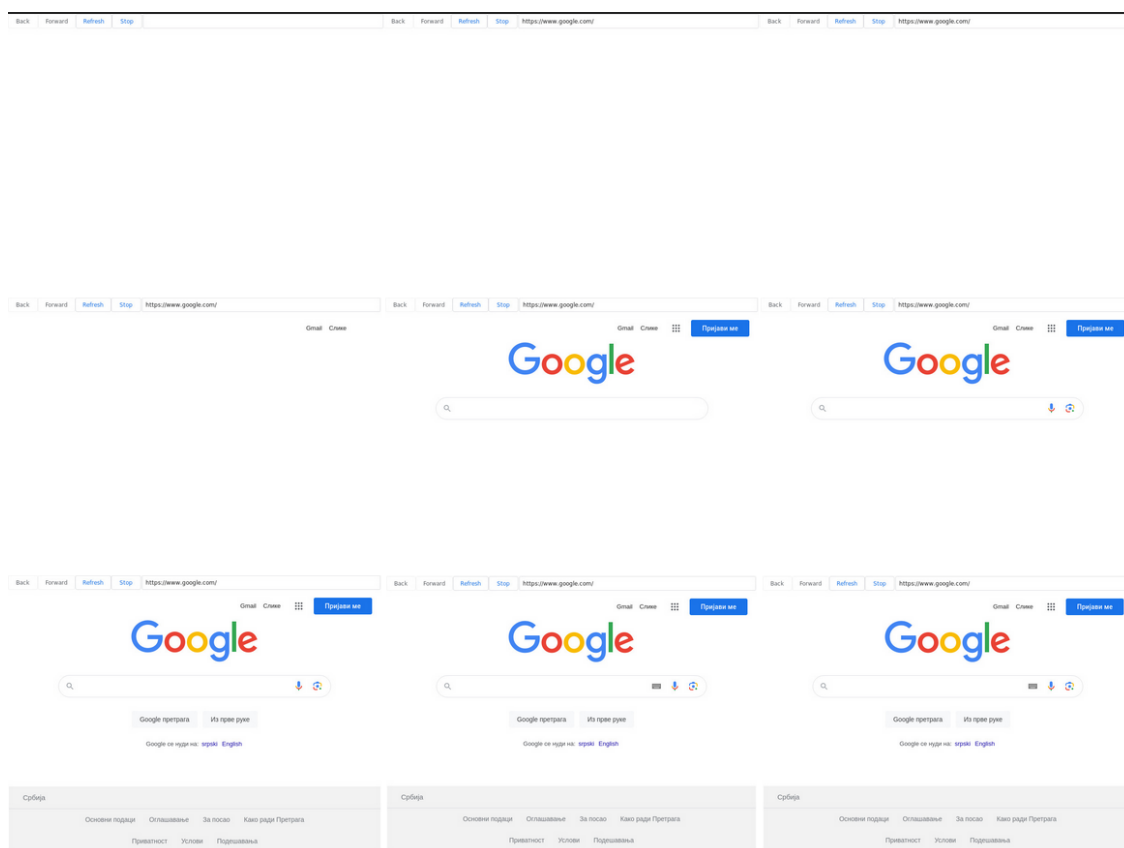
5.2 Прилагођење интернет прегледача да користи *DirectFB* библиотеку за приказ садржаја на екран

С обзиром да *Ubuntu* има проблем са покретањем интернет прегледача користећи *DirectFB* библиотеку, решење је био интернет прегледач који користи *DirectFB* библиотеку и који излазне податке чува на диску у виду слике.

У сврху верификације резултата, покренули смо различите странице и видели како се страница парсира, рендерује и приказује заправо.

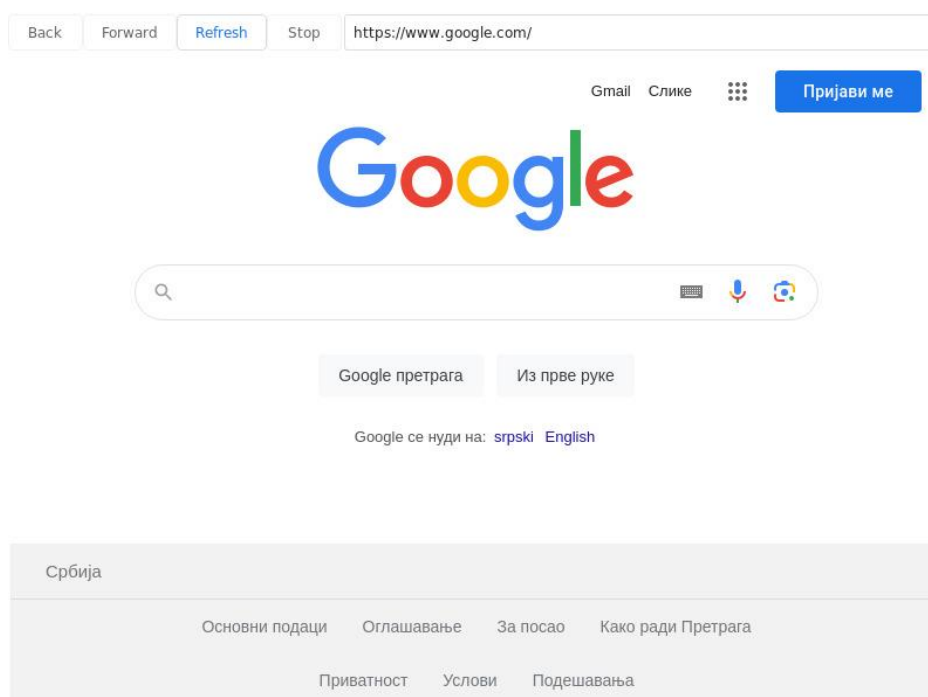
Кључну улогу има функција *SaveAsImage()* у класи *DirectFBConnector*. Она служи као алтернативни механизам за проверу графичког излаза приликом покретања *Chromium*-а на *Linux*-у са *DirectFB*-ом. Ова функција хвата тренутно стање *DirectFB* прозора и чува га као слику на диску.

При покретању *Google* странице можемо видети да кроз 10 слика се страница учита (Слика 19).

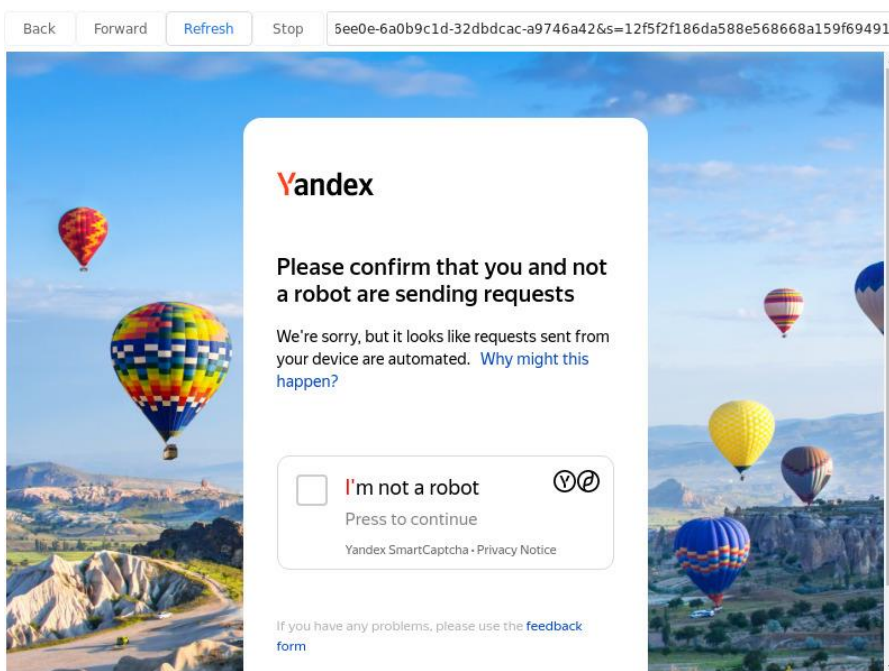
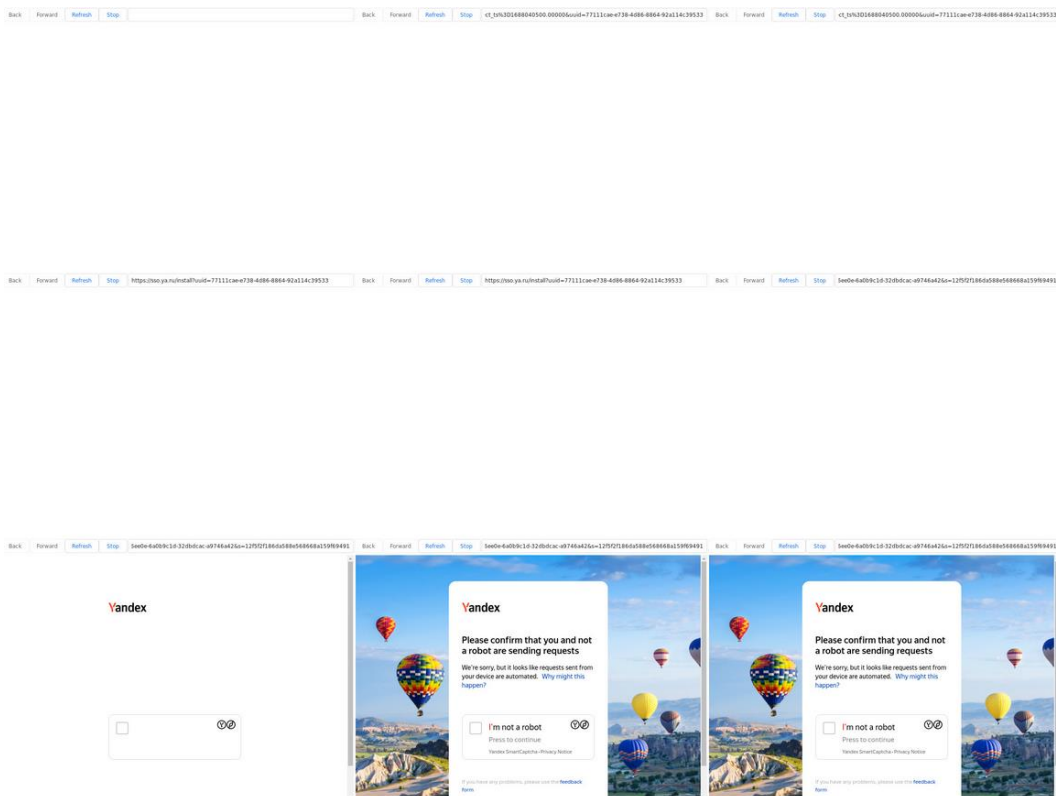


Слика 19 Приказ учитавања *Google* странице

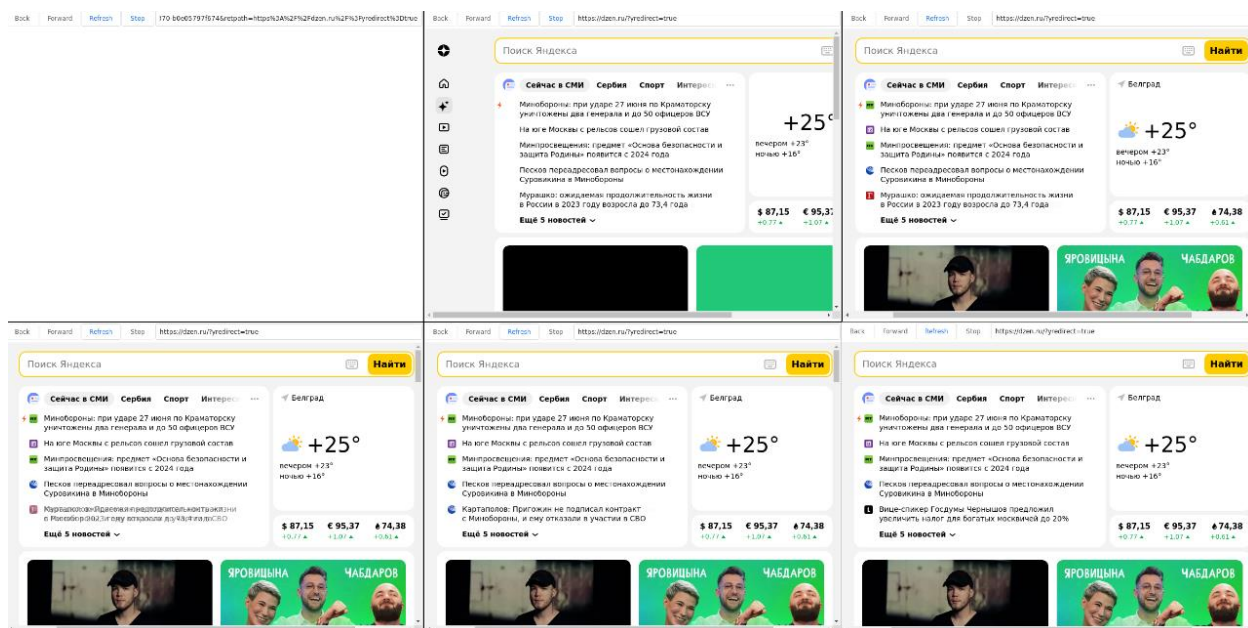
И као коначну учитану страницу видимо приказ на слици (Слика 20).



Слика 20 Резултат учитавања *Google* странице



Међутим постоје и комплекснији задаци, као што је учитавање странице која константно мења свој изглед због реклама које се смењују на страници. Као, на пример, код *Yandex.ru* странце. Страница се врло брзо учитавала али при томе је константо мењала изглед, те је сачувано чак 170 слика пре него је прекинуто. Учитавање је било успешно као што се може видети на сликама (Слика 23).



Слика 23 Приказ учитавања странице *Yandex.ru*

6. Закључак

Овај рад је за циљ имао да реши изазов прилагођавања *Chromium* интернет прегледач за коришћење *DirectFB*, лагане графичке библиотеке, кроз *Ozone* апстрактни слој платформе. Интеграција *DirectFB* у *Chromium* имала је за циљ да побољша перформансе прегледача на уграђеним системима са ограниченим могућностима графичке обраде. Систематским приступом овом задатку, демонстрирали смо изводљивост и предности таквих адаптација, доприносећи вредним увидима и практичним решењима у области развоја интернет прегледача.

Иницијална фаза укључивала је подешавање развојног окружења и изградњу *Chromium*-а за *Linux* платформу. Овај основни рад био је кључан за омогућавање накнадних модификација и напора за интеграцију. Детаљно истраживање архитектуре *Chromium*-а, са фокусом на *Ozone* платформу, пружило је неопходно разумевање за наставак интеграције *DirectFB*.

На основу овог знања, истражили смо *DirectFB* библиотеку, развијајући и покрећући разне примере како бисмо стекли практично искуство. Креирање новог *BUILD.gn* скелета за *DirectFB* платформу био је кључни корак, постављајући темеље за даљи развој. Процес је укључивао дефинисање сетова извора, модификацију постојећих конфигурационих датотека за грађење и имплементацију *DirectFB* класа.

Фаза имплементације укључивала је између осталог писање подкласа за *DirectFBPlatformWindow*, *DirectFBSurfaceFactory* и *DirectFBPlatformScreen*, користећи *DirectFBConnector* класу за обављање основних функција као што су приказивање, скривање и подешавање непрозирности прозора и имплементирањем класе *OzonePlatformDirectFB*, која оркестрира интеракцију између *Chromium*-а и *DirectFB*-а. Резултат је верзија *Chromium*-а која се може изградити и покренути са *DirectFB* графичком библиотеком.

Овај рад доприноси текућем развоју и оптимизацији интернет прегледача за разноврсне хардверске платформе. Успешна интеграција *DirectFB*-а у *Chromium* истиче флексибилност архитектуре *Chromium*-а, показујући његову прилагодљивост различитим графичким библиотекама. Увиди стечени из овог пројекта могу информисати будуће напоре за оптимизацију интернет прегледача за уграђене системе и друге специјализоване употребе.

Иако је овај рад постигао своје примарне циљеве, неколико области за будућа истраживања и развој остаје отворено. Додатна оптимизација интеграције *DirectFB*-а могла би још више побољшати перформансе, посебно у смислу побољшања ефикасности рендеровања. Нарочито би се могао више истражити проблем при покретању интернет прегледача користећи *DirectFB* библиотеку на *Linux* оперативном систему. Поред тога, проширење подршке за напредне графичке функције и побољшање компатибилности са ширим спектром хардвера могло би још више проширити применљивост овог рада.

7. Литература

- [1] DirectFB; URL: <https://elinux.org/DirectFB>; 22.06.2024.г.
- [2] Ozone Overview; URL: https://chromium.googlesource.com/chromium/src/+HEAD/docs/ozone_overview.md; 23.06.2024.г.
- [3] Wikipedia: DirectFB; URL: <https://en.wikipedia.org/wiki/DirectFB>; 22.06.2024.г.
- [4] Explore the Magic Behind Google Chrome; URL: <https://zicodeng.medium.com/explore-the-magic-behind-google-chrome-c3563dbd2739>; 24.06.2024.г.
- [5] Google's Chromium browser explained; URL: <https://www.computerworld.com/article/1717405/googles-chromium-browser-explained.html>; 23.06.2024.г.
- [6] Multi-process Architecture; URL: <https://www.chromium.org/developers/design-documents/multi-process-architecture/>; 24.06.2024.г.
- .