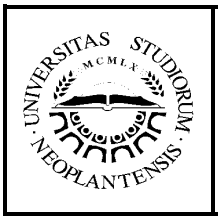


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

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: **Жељана Ракић**
Број индекса: **РА 29/2020**

Тема рада: **Једно рјешење интеграције покретача апликација са основним Андроид подсистемима за побољшано корисничко искуство**

Ментор рада: **проф. 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**:		Жељана Ракић
Ментор, **МН**:		проф. др Илија Башичевић
Наслов рада, **НР**:		Једно рјешење интеграције покретача апликација са основним Андроид подсистемима за побољшано корисничко искуство
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)		7/47/0/3/30/0/0
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника
Предметна одредница/Кључне речи, **ПО**:		Покретач апликација, Андроид подсистеми
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		Овај рад се бави истраживањем интеграције Андроид покретача налик на десктоп са основним Андроид подсистемима укључујући подешавања, управљање звуком, системска обавјештења и управљање корисницима. Интеграција има за циљ да унаприједи цјелокупно Андроид корисничко искуство.
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		10.7.2024.
Чланови комисије, **КО**:	Председник:	проф. др Небојша Пјевалица
	Члан:	проф. др Мирослав Поповић
	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**:		Željana Rakić	
Mentor, **MN**:		Ilija Bašičević, PhD	
Title, **TI**:		Integrating a Desktop-Like Launcher with Core Android Subsystems for elevated user experience	
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)		7/47/0/3/30/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		Launcher, core Android subsystems	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This bachelor thesis explores the integration of a desktop-like Android launcher with core Android subsystems, including settings, audio manager, notification system, and user management. The project endeavors to incorporate efficient quick settings, audio management, keyboard management, user notification systems and user management. The integration aims to elevate the overall Android user experience. This research delves into the technical intricacies of these integrations for optimal functionality and customization.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:		7/10/2024	
Defended Board, **DB**:	President:	Nebojša Pjevalica, PhD	
	Member:	Miroslav Popović, PhD	Menthor's sign
	Member, Mentor:	Ilija Bašičević, PhD	

	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:	Жељана Ракић	Број индекса:	РА 29/2020
Степен и врста студија:	Основне академске студије		
Област:	Електротехника и рачунарство		
Ментор:	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 рјешење интеграције покретача апликација са основним Андроид подсистемима за побољшано корисничко искуство

ТЕКСТ ЗАДАТКА:

Реализовати интеграцију Андроид покретача налик на десктоп са основним Андроид подсистемима укључујући подешавања, управљање звуком, системска обавјештења и управљање корисницима. Интеграција има за циљ да унаприједи цјелокупно Андроид корисничко искуство.

Руководилац студијског програма:	Ментор рада:

Примерак за: o - Студента; o - Ментора

Захвалност

Захваљујем се породици, момку и пријатељима на константној подршци током школовања.

Захваљујем се Институту РТРК на пруженој прилици, академском ментору проф. др Илији Башичевићу, а посебно техничком ментору Артјому Кузмицком, као и свим колегама из тима на сарадњи и стручној помоћи током израде рада.

САДРЖАЈ

Списак слика

Списак табела

Скраћенице

AOSP	- *Android Open-Source Project,* Андроид отворени код
UI	- *User Interface,* корисничка спрега
SDK	- *Software Development Kit,* развојни комплет софтвера
IDE	- *Integrated Development Environment,* интегрисано развојно окружење
ART	- *Android Runtime,* Андроид извршно окружење
API	- *Application Programming Interface,* апликативна програмска спрега
APK	- *Android Package,* формат у који се пакује Андроид апликација
CPU	- *Central Processing Unit,* централни процесор
GPU	- *Graphics Processing Unit,* графички процесор

1. Увод

Задатак овог рада је интеграција Андроид покретача (engl. launcher) налик на десктоп са основним Андроид подсистемима као што су подешавања, управљање звуком, управљање корисницима, системска обавјештења и управљање језиком уноса.

За реализацију пројекта коришћени су програмски језици Котлин и Јава. Андроид покретач је заправо Андроид апликација писана у Котлину са библиотеком Jetpack Compose која се користи за креирање корисничке спреге (engl. User Interface - UI). Програмски језик Јава је коришћен за реализацију интеграције унутар Андроид отвореног кода (engl. Android Open-Source Project - AOSP).

Циљ интеграције покретача налик на десктоп са основним Андроид подсистемима је да се побољша цјелокупно корисничко искуство.

У оквиру другог поглавља наведене су теоријске основе које представљају базу за реализацију пројекта. Такође, дате теоријске основе су неопходне за разумијевање концепта рјешења самог пројекта.

Треће поглавље представља концепт рјешења у оквиру којег се описује цјелокупна идеја пројекта са објашњењима које интеграције су имплементиране, зашто је због неких интеграција морао да се мијења код Андроид оперативног система као и то како је дио рјешења реализован само у склопу Андроид покретача.

У четвртом поглављу приказано је програмско рјешење.

Пето поглавље представља резултате пројекта на различитим Андроид оперативним системима и на различитим платформама.

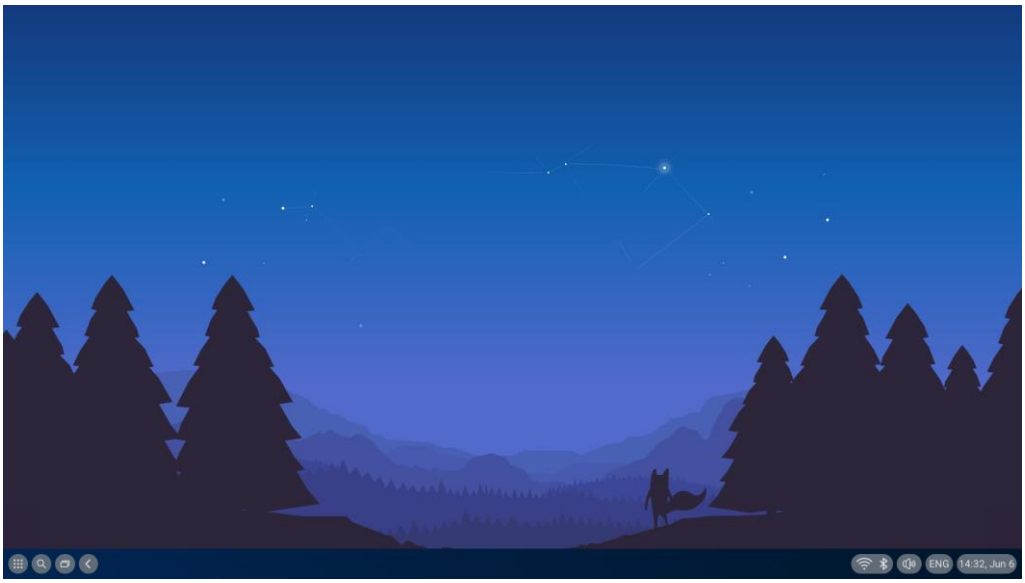
У шестом поглављу је изнесен закључак са сумираним резултатима имплементације рјешења интеграције Андроид покретача са основним Андроид подсиситемима.

2. Теоријске основе

У овом поглављу дате су теоријске основе архитектуре Андроид оперативног система и његових кључних компоненти и подсистема. Такође, описан је Андроид отворени код – AOSP, архитектура Андроид оперативног система, Андроид апликације као и имплементација покретача апликација налик на десктоп.

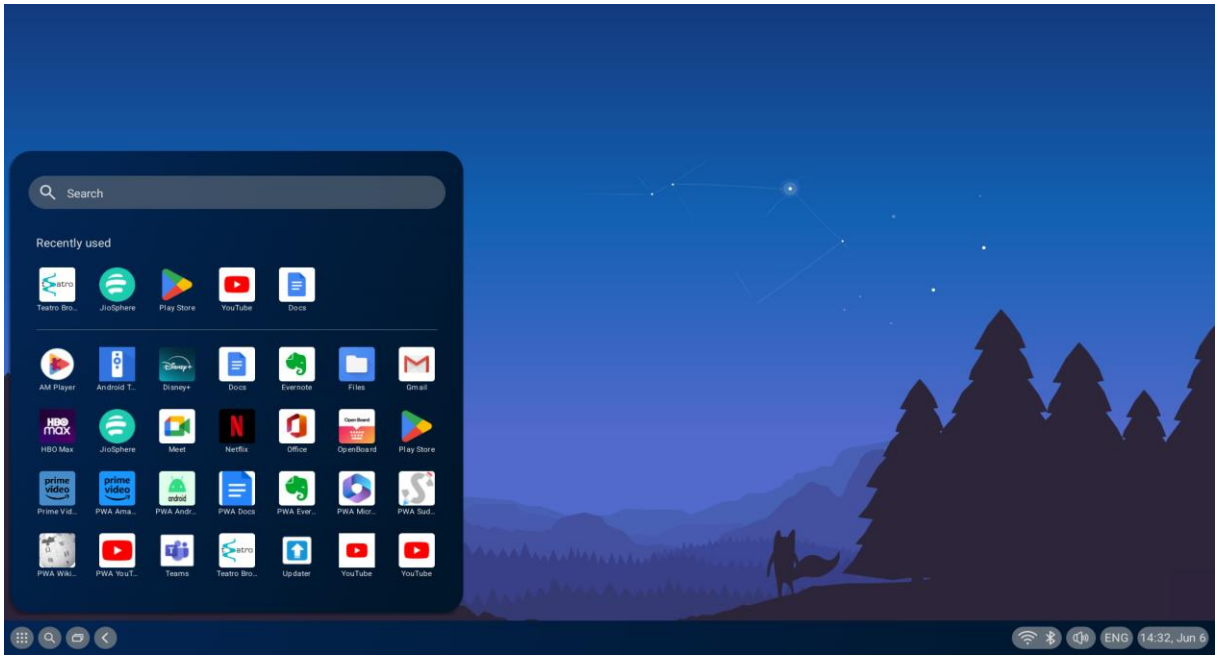
2.1 Андроид покретач апликација налик на десктоп

Андроид покретач апликација (engl. launcher) је апликација која омогућава корисницима да прилагођавају почетни екран, покрећу друге апликације и управљају основним функцијама уређаја. То је спрега између корисника и оперативног система, омогућава приступ апликацијама и информацијама на интуитиван начин. Покретач апликација игра кључну улогу у корисничком искуству на Андроид уређају.

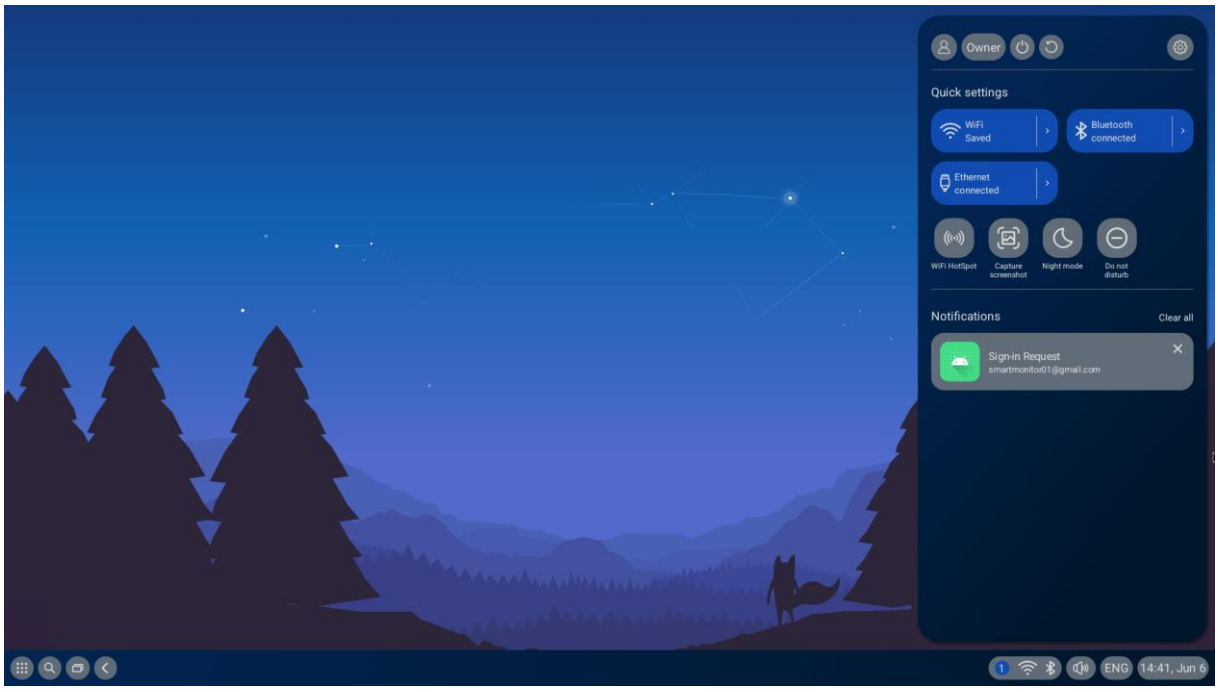


Слика 2.1 Покретач Андроид апликација налик на десктоп

На слици је приказан почетни екран (engl. Home Screen) покретача апликација налик на десктоп који садржи: палету послова (engl. Taskbar) помоћу кога се даље могу отварати приказ апликација (engl. App Drawer), приказ покренутих апликација (engl. Task Menu), брза подешавања и обавјештења (engl. Notifications and Quick Settings), управљач звуком и језиком уноса, календар и сат.



Слика 2.2 App Drawer



Слика 2.3 Notifications and Quick Settings

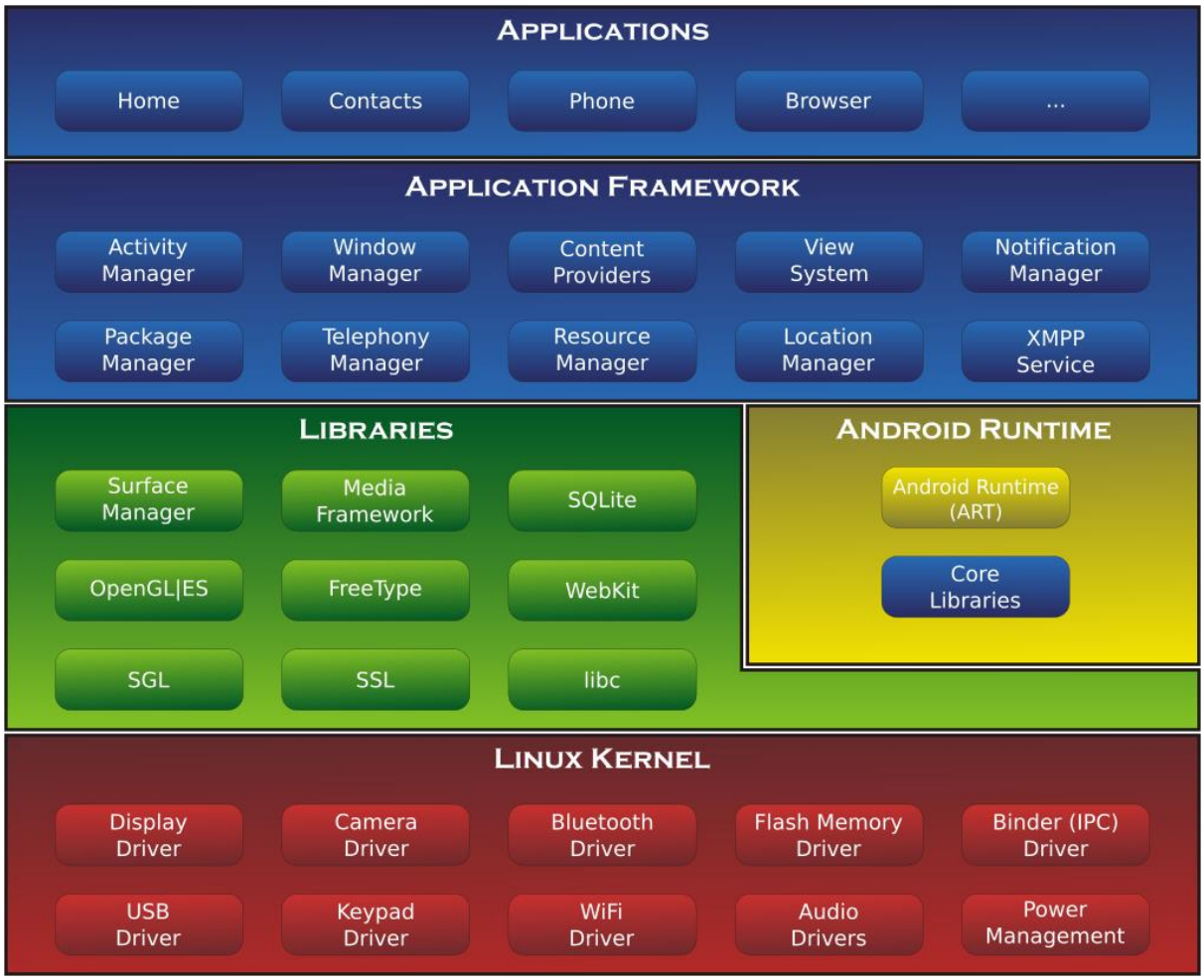
2.2 Андроид

Андроид је оперативни систем заснован на Линукс језгру, који је развила и одржава компанија Google. Првобитно је био намјењен за паметне телефоне и таблете, али је касније проширен на бројне друге уређаје, укључујући паметне телевизоре, сатове, аутомобиле и друге уређаје. Андроид је данас најзаступљенији оперативни систем за мобилне уређаје широм свијета.

2.2.1 Архитектура Андроид оперативног система

Андроид оперативни систем је сложена платформа која обухвата више слојева. Ова архитектура омогућава флексибилност и моћну интеграцију са хардвером и апликацијама. Основни слојеви архитектуре су:

1. **Линукс језгро**: Основа Андроид оперативног система је Линукс језгро које управља основним системским сервисима као што су безбједност, управљање меморијом, управљање процесима и мрежна комуникација. Линукс језгро такође омогућава апстракцију хардвера, чиме олакшава рад са различитим хардверским компонентама.
2. **Библиотеке (engl. Libraries)**: Андроид користи низ C/C++ библиотека које су специфичне за различите функције. На примјер, библиотеке као што су Surface Manager, Media Framework и WebKit обезбјеђују основне функције за графику, репродукцију медија и прегледање веб садржаја.
3. **Андроид Runtime (ART)**: Андроид Runtime је окружење за извршавање апликација. ART користи Just-In-Time (JIT) и Ahead-Of-Time (AOT) компилацију за побољшање перформанси и ефикасности апликација. Такође укључује Garbage Collection (GC) механизме за аутоматско управљање меморијом.
4. **Апликативни оквир (engl. Application Framework)**: Апликативни оквир пружа апликативне програмске спреге (engl. Application Programming Interface - API) које програмери користе за развој апликација. Овај слој садржи менаџере за различите услуге, укључујући Activity Manager, Resource Manager и Notification Manager, који омогућавају апликацијама да користе ресурсе система и комуницирају са корисницима.
5. **Системске апликације**: Андроид долази са низом системских апликација које пружају основну функционалност и често служе као примјери за развој других апликација.



Слика 2.4 Архитектура Андроид оперативног система [5]

2.2.2 Основни Андроид подсистеми

Основни Андроид подсистеми укључују различите компоненте које обезбјеђују функционалност оперативног система:

1. **Подешавања (engl. Settings):** Омогућавају корисницима да прилагоде различите аспекте уређаја, укључујући мрежне опције, звук, управљање апликацијама и друге.
2. **Управљач звуком (engl. Audio Manager):** Управља звуком на уређају, укључујући контролу јачине звука, аудио излазе и интеграцију са различитим аудио уређајима.
3. **Систем обавјештења (engl. Notifications):** Омогућава апликацијама да шаљу корисницима обавјештења која се приказују у статусној линији и могу укључивати различите акције.

4. **Управљање корисницима (engl. User Manager):** Омогућава креирање и управљање различитим корисничким профилима на истом уређају, што је посебно корисно за уређаје који се дијеле између више корисника.
5. **Управљање језиком уноса (engl. Input Manager):** Омогућава да корисници могу подесити више језика уноса и брзо се пребацивати између њих помоћу пречица са тастатуре.

2.2.3 Андроид отворени код – AOSP

AOSP је пројекат креиран од стране компаније Google са циљем да обезбједи слободно доступну, отворену платформу и представља основу за Андроид оперативни систем. Андроид отворени код омогућава програмерима и произвођачима уређаја да приступе изворном коду Андроид оперативног система, што им даје могућност да модификују, прилагођавају и дистрибуирају Андроид софтвер у складу са својим потребама. AOSP доприноси стварању разноврсног екосистема уређаја и апликација. Ово не само да подстиче иновације, већ и омогућава развој уређаја који су приступачни широј публици.

2.2.4 Андроид апликације

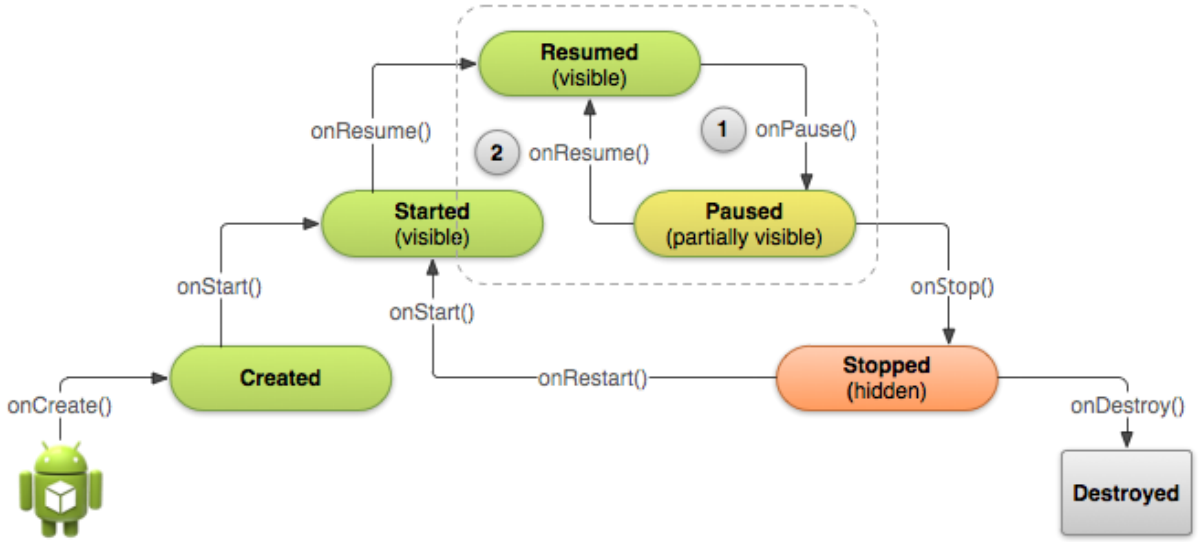
Андроид апликације се обично пишу у програмским језицима Јава и Котлин уз коришћење Андроид SDK-a (engl. Software Development Kit). Андроид SDK садржи алате и библиотеке неопходне за развој апликација, као и емулаторе за тестирање.

Након што је апликација написана и тестирана, Андроид SDK алати преводе изворни код, заједно са свим подацима и потребним датотекама, у Андроид пакет (engl. Android Package). Овај пакет има екстензију .apk и представља архивску датотеку која садржи све компоненте неопходне за инсталацију и извршавање апликације на Андроид уређајима. Корисници могу преузети .apk датотеку са различитих извора, најчешће са Google Play продавнице (engl. Google Play Store), и инсталирати је на своје уређаје. Основне компоненте Андроид апликација:

- **Активности (engl. Activities):** Активности су једна од основних компоненти Андроид апликација. Оне представљају један екран са корисничком спрегом. Апликација може имати једну или више активности, гдје свака активност представља различит дио функционалности апликације. Комбинација више активности омогућава кориснику да интерагује са апликацијом на различите начине.

- **Сервиси (engl. Services):** Сервиси су компоненте које раде у позадини без директне интеракције са корисником. Омогућавају апликацијама да извршавају дуготрајне операције, као што је пуштање музике, прикупљање података са интернета или обрада података, чак и када корисник не користи активно апликацију.
- **Добављачи садржаја (engl. Content Providers):** Ова компонента омогућава дијељење података између различитих апликација. Подаци могу бити сачувани у различитим форматима, као што су систем датотека и базе података.
- **Примаоци емитованих порука (engl. Broadcast Receivers):** Омогућавају апликацијама да реагују на системске поруке (engl. broadcasts). Могу реаговати на различите догађаје, као што су промјене у стању батерије, промјена мреже, промјена звука и друге.

Свака компонента у Андроид апликацији има свој специфичан животни циклус који дефинише како се компонента креира, користи и уништава. Разумјевање животног циклуса компоненти је кључно за правилно управљање ресурсима и обезбјеђивање добре перформансе апликације.



Слика 2.5 Животни циклус Андроид апликације [1]

Покретач Андроид апликација налик на десктоп је заправо Андроид апликација, има своје активности, сервисе и примаоце емитованих порука и инсталира се као системска апликација како би добила све неопходне дозволе за приступ осјетљивим подацима и ресурсима. Ове дозволе се наводе у AndroidManifest.xml датотеци

апликације. Такође, све компоненте апликације се морају дефинисати у AndroidManifest.xml датотеци, јер прије него што Андроид платформа може да покрене неку компоненту апликације, мора да сазна да та компонента постоји, а то сазнаје читајући управо ову датотеку.

2.3 Алати за писање и развој Андроид покретача

За развој Андроид покретача кориштен је Андроид Студио. Андроид Студио је званично интегрисано развојно окружење (engl. Integrated development environment - IDE) за Андроид апликације. Пружа свеобухватан скуп алата и функционалности неопходних за креирање, тестирање и дистрибуцију Андроид апликација.

2.3.1 Jetpack Compose

Jetpack Compose је модеран алат за креирање корисничких спрега у Андроид апликацијама. Заснован је на декларативном програмском моделу што значајно поједностављује и убрзава процес развоја корисничких спрега, омогућавајући програмерима да лакше и брже креирају интерактивне апликације. Корисничка спрега се описује позивајући низ функција које трансформишу податке у хијерархију корисничке спреге

Jetpack Compose долази са богатим скупом уграђених компоненти као што су текстуални уноси, листе, редови, колоне, дугмад и друге. Compose апликација је направљена од Compose функција – регуларне функције означене са *@Composable* које могу да позивају друге Compose функције.

```
@Composable
fun ColumnArrangement(){
    Column(modifier = Modifier.fillMaxHeight().fillMaxWidth(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment  =  Alignment.End
    ) {
        Text(text = "Text 1",Modifier.background(Color.Red))
        Text(text = "Text 2",Modifier.background(Color.White))
        Text(text = "Text 3",Modifier.background(Color.Green))
    }
}
```

Слика 2.6 Примјер једноставне Compose функције

2.3.2 Рефлексија

Рефлексија је моћна карактеристика програмских језика која омогућава програмерима да прегледају или модификују понашање програма током његовог извршавања. У контексту Андроид апликација, рефлексија се често користи за динамичко учитавање класа, приступ приватним пољима и методама, као и за позивање метода у вријеме извршавања, што иначе не би било могуће коришћењем статичког програмског кода.

У контексту покретача апликација, рефлексија се користи за приступ приватним пољима и методама самог Андроид оперативног система (односно његовог апликативног оквира) које нису доступне коришћењем конвенционалног кода.

Важно је нагласити да Google Play продавница има строге смјернице када је у питању коришћење рефлексије у Андроид апликацијама. Иако рефлексија сама по себи није забрањена, њена употреба може бити подложна додатним провјерама и ограничењима због потенцијалних безбједносних и перформансних проблема. Апликације које користе рефлексију за приступ приватним или системским API-јима могу бити одбијене ако се сматра да угрожавају безбједност корисника или система.

```
try {
    val privateField: Field = MyClass::class.java.getDeclaredField("privateFieldName")
    privateField.isAccessible = true
    val value: Any? = privateField.get(myClassInstance)
    println("Value: $value")
} catch (e: NoSuchFieldException) {
    e.printStackTrace()
} catch (e: IllegalAccessException) {
    e.printStackTrace()
}
```

Слика 2.7 Примјер рефлексије за приступ приватном пољу

3. Концепт рјешења

Интеграција покретача апликација са основним Андроид подсистемима представља значајан изазов у циљу побољшања корисничког искуства. Овај концепт рјешења се фокусира на детаљан опис интеграције, разрађујући основне елементе и модуле који су од значаја.

3.1 Општи преглед компоненти система

Општа архитектура система обухвата главне компоненте и њихове међусобне односе. Компоненте система су:

- Покретач апликација: Централна компонента која, преко корисничке спреге, пружа корисницима приступ апликацијама и разним услугама.
- Подсистем подешавања: Пружа корисницима конфигурацију различитих аспеката уређаја као што су повезивање на мрежу, конекција са другим уређајима, ноћни режим, приступна тачка, режим „не узнемиравај“ и друге.
- Подсистем управљања звуком: Управља свим аспектима звука на уређају.
- Подсистем обавјештења: Кључан за комуникацију између система и корисника, а такође омогућава пријем обавјештења од других апликација инсталираних на уређају.
- Подсистем управљања корисницима: Омогућава креирање и управљање корисничким профилима.
- Подсистем управљања језиком уноса: Обухвата све методе уноса текста као и методе за промјену језика уноса.

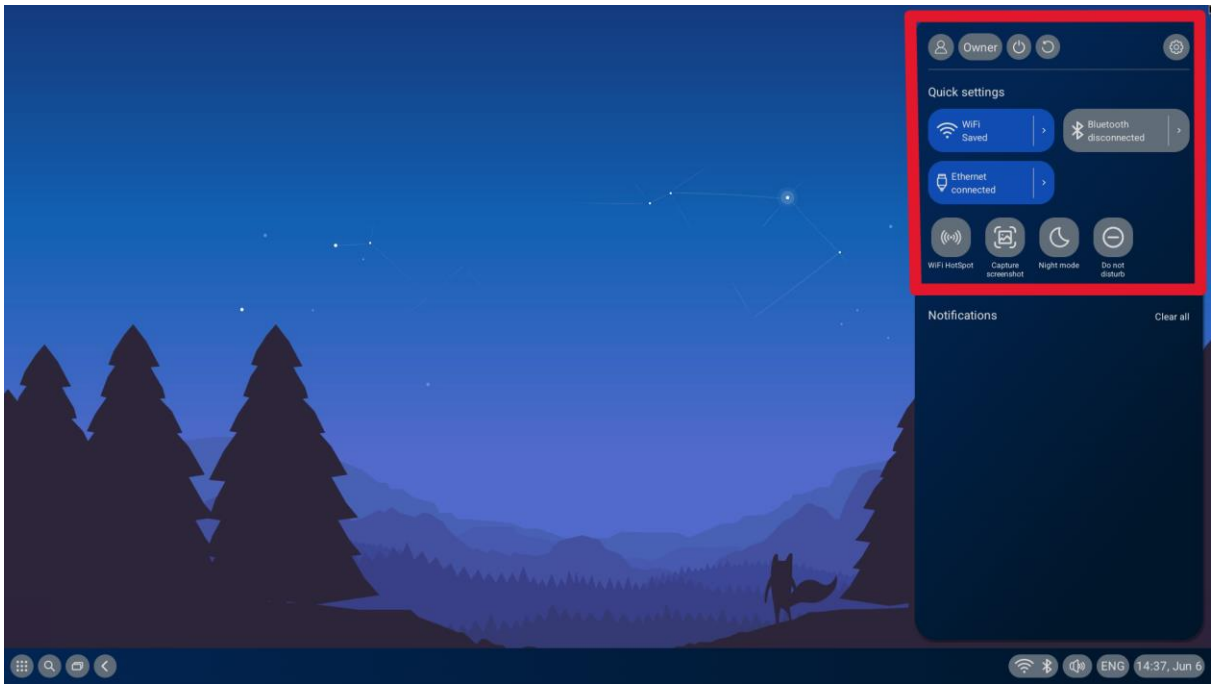
3.2 Интеграција са основним Андроид подсистемима

Покретач апликација омогућава корисничку спрегу преко које корисници могу да на интуитиван начин управљају Андроид подсистемима на уређају. У овој цјелини је описана интеграција са сваким појединачним подсистемом.

3.2.1 Подсистем подешавања

Интеграција са подсистемом подешавања омогућава корисницима да брзо приступе важним подешавањима директно из покретача.

На слици је приказана корисничка спрега за брзи приступ основним подешавањима.



Слика 3.1 Брза подешавања

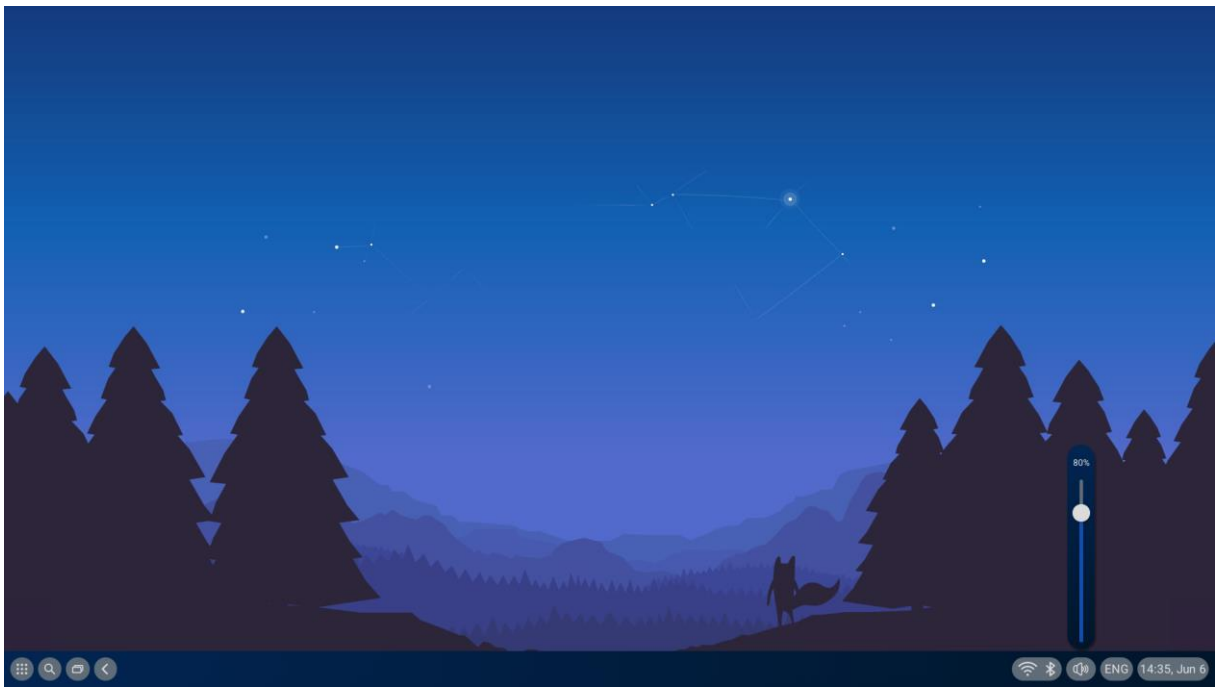
Као што се може видјети на слици, корисницима је омогућено да управљају бежичним интернетом (engl. Wi-Fi), бежичним преносом података (engl. Bluetooth), мрежном везом (engl. Ethernet). Такође, корисник може да укључи/искључи приступну тачку (engl. Wi-Fi Hotspot), ноћни режим (engl. Night Mode) и режим „не узнемиравај“ (engl. Do not disturb) који блокира сва обавјештења, а може и да слика екран коришћењем опције Capture screenshot. Корисничка спрега јасно показује корисницима која брза подешавања су тренутно укључена, а која нису.

Поред приступа брзим подешавањима корисник може да отвори системска подешавања Андроида притиском на иконицу у горњем десном углу, а у истом реду је и приступ опцијама за искључивање (engl. Power off) и поново покретање (engl. Restart) уређаја.

3.2.2 Подсистем управљања звуком

Интеграција са управљачем звука омогућава корисницима да брзо и лако мијењају јачину звука директно из покретача апликација. Пошто је покретач налик на десктоп окружење, једна јачина звука важи за све звукове на уређају. То значи да нема засебних опција за контролу музике, звона и системских звукова као што је случај код класичних покретача Андроид апликација намјењених за мобилне уређаје. Оваква интеграција поједностављује управљање звуком и пружа конзистентно корисничко искуство, прилагођено корисницима који су навикли на десктоп окружења.

На слици је приказана корисничка спрега за контролу јачине звука из покретача апликација.



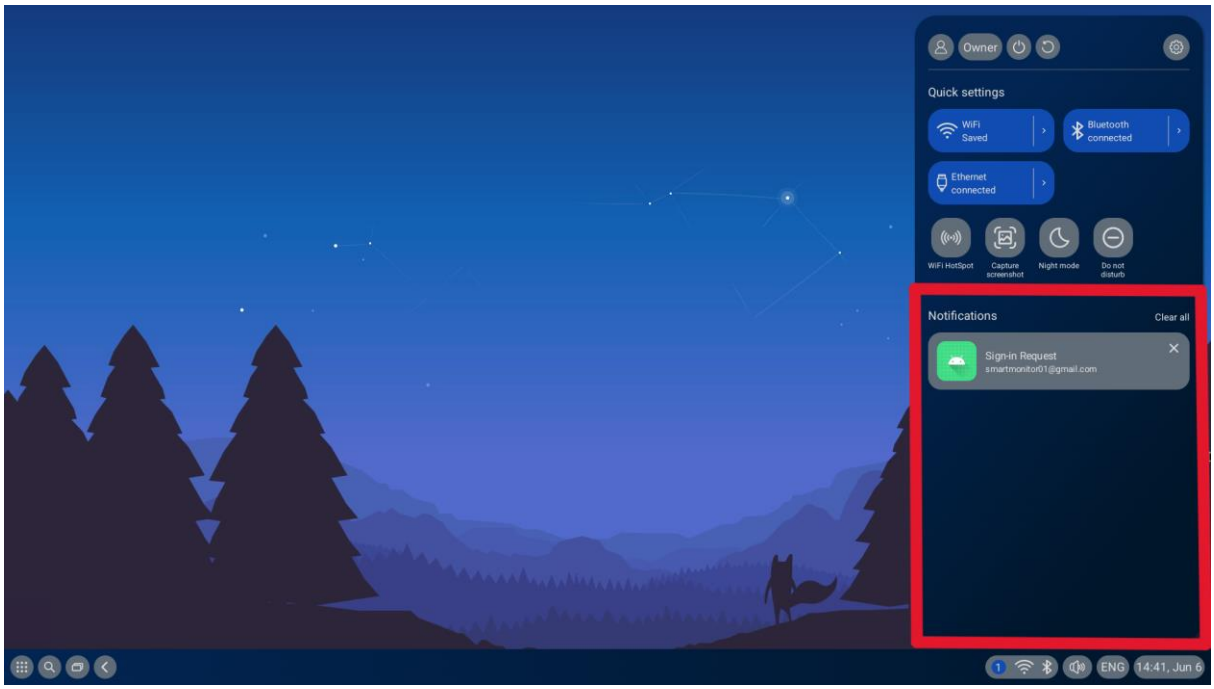
Слика 3.2 Клизач за контролу јачине звука

Имплементација омогућава промјену јачине звука кретањем клизача и приказује вриједност јачине звука у процентима. Такође, корисничка спрега прецизно реагује на промјене јачине звука, било да се оне врше директно на уређају или преко спољашњих уређаја као што су слушалице. Ова функционалност обезбјеђује тачну и правовремену

визуелну репрезентацију тренутне јачине звука, без обзира на извор управљања. Систем континуирано прати и синхронизује статус јачине звука, што корисницима омогућава интуитивно и ефективно управљање параметрима звука. Оваква интеграција побољшава корисничко искуство пружајући јасну и прецизну повратну информацију о тренутној јачини звука.

3.2.3 Подсистем обавјештења

Подсистем обавјештења је кључан за комуникацију између система и корисника. Интеграција покретача са овим подсистемом омогућава покретачу да пресреће и обрађује обавјештења не само од самог Андроид оперативног система већ и од других апликација. Овај процес омогућава покретачу да обавјештења корисницима представља на интуитиван и приступачан начин, обезбјеђујући да корисници буду благовремено информисани о свим важним догађајима и активностима у систему.



Слика 3.3 Приказ обавјештења

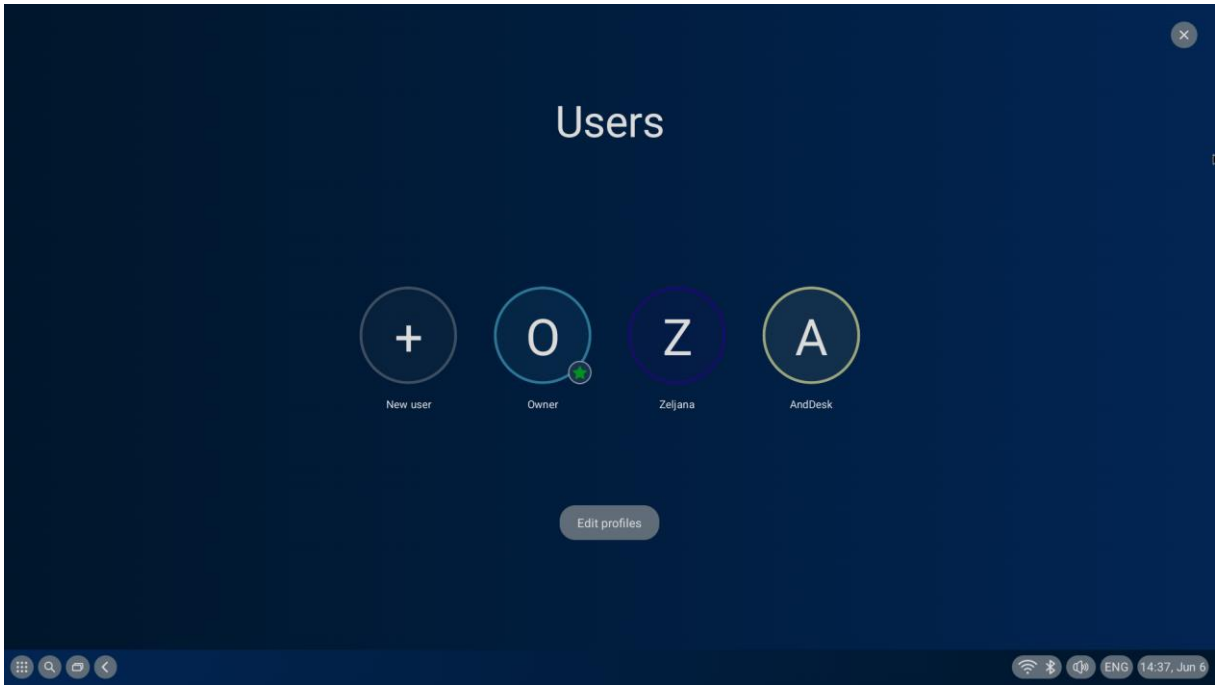
На слици је приказан визуелни приказ једног системског обавјештења. Корисник може одмах да уочи да ли је обавјештење системско или потиче од неке апликације на уређају. У случају да је обавјештење од апликације, приказ обавјештења укључује и икону конкретне апликације, а ако је обавјештење системско онда укључује препознатљиву икону Андроида. Систем омогућава кориснику да уклони појединачна обавјештења, као и да избрише сва обавјештења уколико их има више. Поред тога,

корисник може да види број пристиглих обавјештења на палети послова уколико је дијалог који приказује брза подешавања и обавјештења затворен. Притиском на одређено обавјештење, апликација задужена за то обавјештење се одмах отвара, или у случају системског обавјештења, корисник се преусмјерава на одговарајући дио Андроид подешавања.

3.2.4 Подсистем управљања корисницима

Интеграција са управљањем корисницима омогућава креирање и управљање корисничким профилима директно из покретача. Ово омогућава лако пребацивање између профила и прилагођавање искуства сваком кориснику.

У оквиру овог система, максималан број корисника, односно корисничких налога, ограничен је на пет. Ово ограничење је имплементирано како би се осигурало оптимално функционисање система и одржала стабилност оперативног окружења. Усложњавање управљања већим бројем корисничких налога може довести до повећања потрошње системских ресурса, што би могло негативно утицати на перформансе и брзину одзива система. Због тога је максималан број корисничких налога постављен на пет, како би се обезбиједила ефикасност и стабилност система у свим условима коришћења. Како би се добила подршка за више корисника, извршене су минималне промјене у оквиру Андроид SDK-а за циљну платформу.



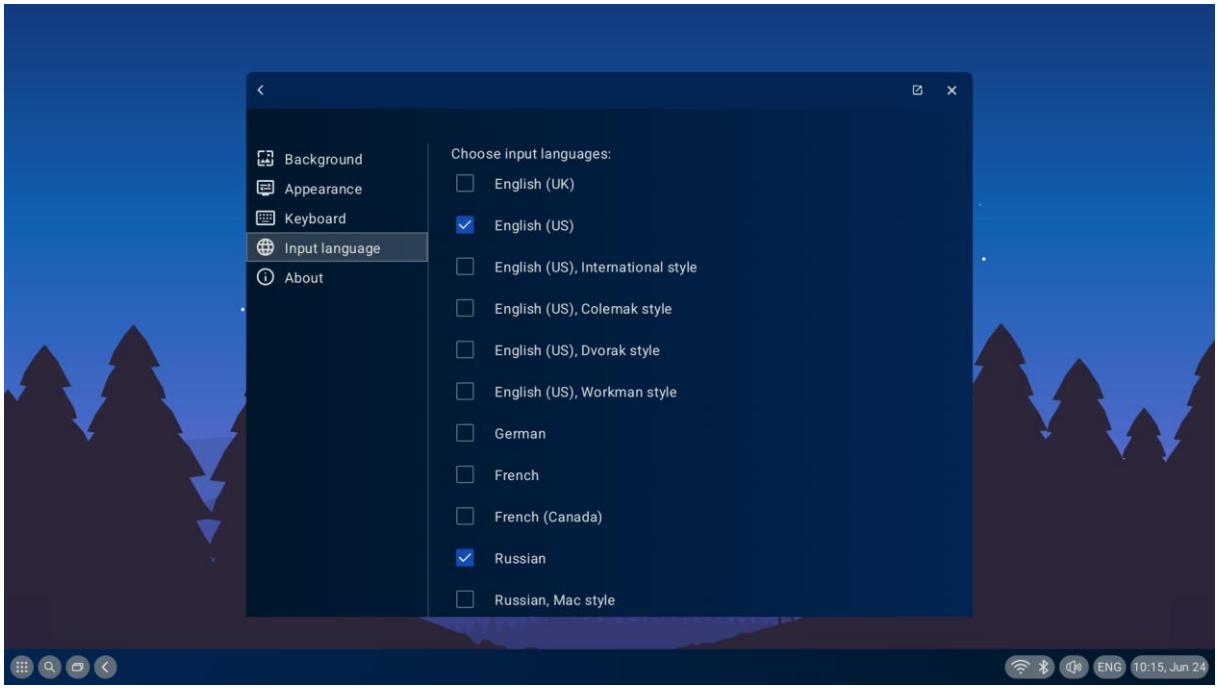
Слика 3.4 Приказ корисничких налога

На слици може да се види приказ корисничких налога, конкретно три корисничка налога. Тренутни активни корисник је системски администартор (engl. Owner) који има овлашћења да додаје и брише друге корисничке налоге. Кориснички налози морају да имају различита имена. Корисничка спрега пружа функционалност додавања нових корисника, уређивање и брисање већ постојећих корисника, лако пребацивање између корисника кликом на жељеног корисника као и интуитиван приказ тренутно активног корисничког налога.

3.2.5 Подсистем управљања језиком уноса

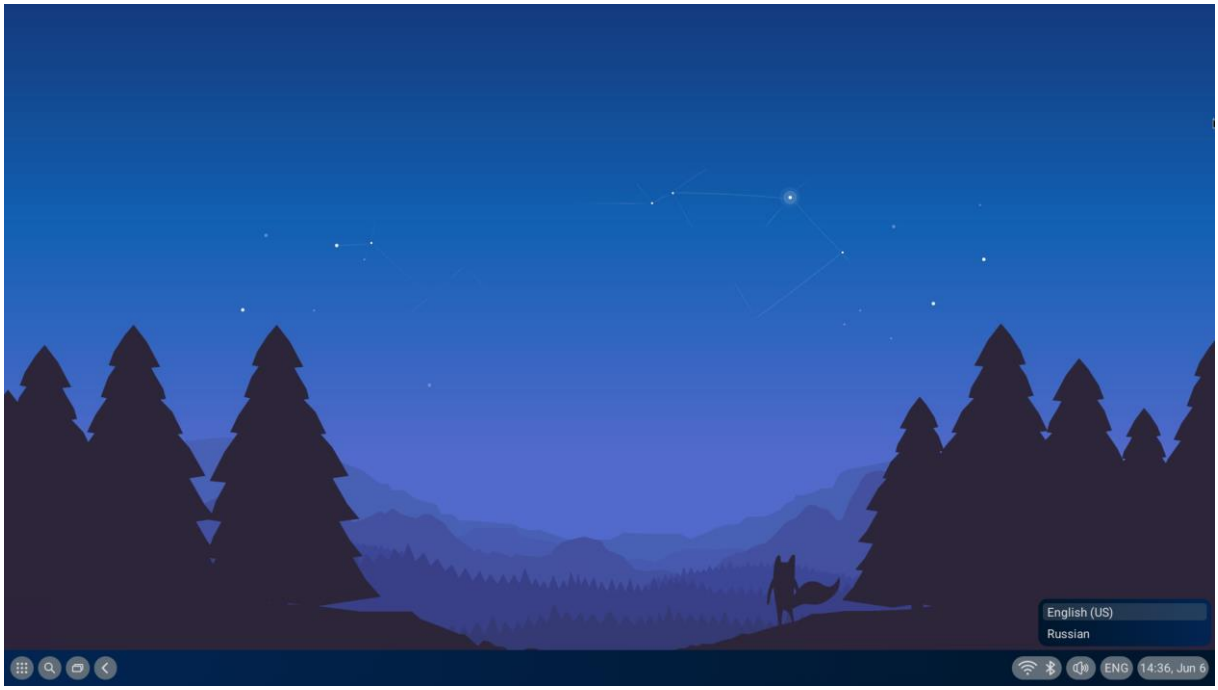
Интеграција покретача апликација са подсистемом управљања језиком уноса доноси бројне бенефите, који значајно унапређују корисничко искуство и функционалност система.

Корисници могу брзо и лако мијењати језике уноса директно из покретача, без потребе за сложеним навигацијама кроз системска подешавања. Ово је посебно корисно за кориснике који често прелазе између више језика. Такође, корисници могу одабрати максимално пет језика уноса које најчешће користе и лако се пребацивати између њих коришћењем пречица са тастатуре као што су: *Ctrl+Space* и *Windows key+Space.*



Слика 3.5 Корисничка спрега за избор језика уноса

Као што је приказано на Слици 3.5, корисник може да изабере жељене језике уноса у подешавањима самог покретача која отвара помоћу десног клика на позадину покретача и бирањем опције *Персонализација.* На том мјесту се налазе сви подржани језици уноса и корисник може да означи оне које жели или најчешће користи. Изабрани језици се приказују у дијалогу језика уноса ради што бржег приступа.



Слика 3.6 Дијалог језика уноса

На слици је приказан дијалог за управљање језиком уноса који јасно указује на тренутно активан језик уноса. Корисник може лако и брзо да мијења језик уноса једноставним притиском на одабрани језик у дијалогу или коришћењем пречица на тастатури. Овај дијалог пружа интуитиван и брз приступ разним језицима уноса, омогућавајући корисницима да оптимизују своје радне процесе. Поред тога, на палети послова се приказују прва три слова тренутног језика уноса, што омогућава кориснику да одмах зна који је језик активан, без потребе да улази у подешавања.

Дати концепт рјешења пружа основни преглед компоненти покретача апликација које су интегрисане са основним Андроид подсистемима и тако омогућавају корисницима да управљају датим подсистемима.

4. Програмско рјешење

У овом дијелу је представљено програмско рјешење које се бави интеграцијом покретача апликација са основним Андроид подсистемима. Овдје је стављен фокус на детаљно објашњење израде функционалности и техничких аспеката који омогућавају рад овог система. Главни алати и технологије коришћени за развој укључују Андроид Студио за развојно окружење, програмске језике Јава и Котлин за писање покретача као и алат Jetpack Compose за изглед корисничке спреге.

4.1 Имплементација функционалних дијелова Андроид покретача

Ова цјелина описује основне апликативне програмске спреге као и основне функције помоћу којих је остварена жељена функционалност система.

4.1.1 Управљач звуком (engl. Audio Manager)

Audio Manager је класа обезбијеђена од стране Андроид система и може се користити за контролу јачине звука на Андроид уређају. Функције ове класе које су кориштене у имплементацији су: *getStreamVolume* и *setStreamVolume*.

Метода *getStreamVolume* је доступна од првог нивоа API-ја у Андроиду и омогућава апликацијама да добију информације о тренутној јачини звука за различите типове аудио токова. Метода прихвата један параметар типа int који одређује тип аудио тока чију јачину звука желимо добити. Андроид систем дефинише неколико константи за различите типове аудио токова:

- **AudioManager.STREAM_VOICE_CALL**: За јачину звука гласовних позива.

- **AudioManager.STREAM_SYSTEM**: За системске звукове.
- **AudioManager.STREAM_RING**: За звукове звона.
- **AudioManager.STREAM_MUSIC**: За музичке токове.
- **AudioManager.STREAM_ALARM**: За аларме.
- **AudioManager.STREAM_NOTIFICATION**: За обавјештења.

Метода *setStreamVolume* је, такође, доступна од првог нивоа API-ја у Андроиду и користи се за подешавање нивоа јачине звука за различите типове аудио токова.

```
fun getCurrentVolumeLevel(): Float {
    return audioManager.getStreamVolume(AudioManager.STREAM_MUSIC).toFloat()
}
private fun setSystemVolume(volumeLevel: Float) {
    audioManager.setStreamVolume(AudioManager.STREAM_MUSIC, volumeLevel.toInt(), flags: 0)
}

fun updateVolume() {
    _volume.value = getCurrentVolumeLevel()
}
```

Слика 4.1 Примјер функција за контролу јачине звука

Ако се јачина звука мијења директно на уређају или преко слушалица, а не путем покретача, неопходно је правовремено ажурирати корисничку спрегу која приказује клизач за контролу звука. Ово је потребно како би клизач тачно одражавао актуелну вриједност јачине звука. Имплементација ове функционалности је омогућена коришћењем примаоца емитованих порука.

```
class VolumeChangeReceiver() : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        val viewModel = AudioViewModel.ViewModelProvider.provideAudioViewModel()
        if (intent.action == "android.media.VOLUME_CHANGED_ACTION" || intent.action == AudioManager.RINGER_MODE_CHANGED_ACTION) {
            viewModel.updateVolume()
        }
    }
}
```

Слика 4.2 Прималац емитованих порука за промјену јачине звука

4.1.2 Корисничка спрега за управљање звуком

Корисничка спрега за управљање звуком имплементирана је коришћењем Compose функције која приказује изглед клизача за контролу јачине звука.

```
@Composable
fun VolumeControl(context: Context) {
    val audioViewModel = AudioViewModel.ViewModelProvider.provideAudioViewModel()
    val currentVolumeState = audioViewModel.volume.collectAsState()
    val audioManager = context.getSystemService(Context.AUDIO_SERVICE) as AudioManager
    val maxVolume =  remember {
        audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC)
    }.toFloat()
    val volumePercentage = (currentVolumeState.value / maxVolume * 100).toInt()

    Column(
        modifier = Modifier
            .clip(CircleShape)
            .padding(OVERLAY_SHADOW_WIDTH) // for shadow
            .shadow(
                elevation = OVERLAY_SHADOW_WIDTH,
                shape = RoundedCornerShape(OVERLAY_ROUNDED_CORNERS_RADIUS),
            )
            .teatroBackground( angle: 90f)
            .fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) { this: ColumnScope
        VolumeDisplay(volumePercentage)
        VolumeSlider(
            currentVolume = currentVolumeState.value,
            onVolumeChange = { it: Float
                audioViewModel.setVolume(it) },
            maxVolume = maxVolume,
            modifier = Modifier.weight(8f)
        )
    }
}
```

Слика 4.3 Compose функција клизача

4.2 Сервис за слушање обавјештења (engl. Notification Listener Service)

Сервис за слушање обавјештења је напредни механизам у Андроид оперативном систему који омогућава апликацијама да прате и реагују на системска обавјештења. Овај сервис омогућава апликацији да добија информације о новим обавјештењима, уклањању постојећих обавјештења и промјенама у рангирању обавјештења.

Notification Listener Service је декларисан у манифесту покретача апликација са дозволом *Manifest.permission.BIND_NOTIFICATION_LISTENER_SERVICE*.

То осигурава да само апликације које имају ову дозволу могу користити сервис за слушање обавјештења.

```
<service
    android:name=".services.NotificationListener"
    android:exported="false"
    android:label="Notification Listener Service"
    android:permission="android.permission.BIND_NOTIFICATION_LISTENER_SERVICE">
    <intent-filter>
        <action android:name="android.service.notification.NotificationListenerService" />
    </intent-filter>

    <meta-data
        android:name="android.service.notification.default_filter_types"
        android:value="conversations|alerting" />
    <meta-data
        android:name="android.service.notification.disabled_filter_types"
        android:value="ongoing|silent" />
</service>
```

Слика 4.4 Декларација сервиса за слушање обавјештења

Функције које су кориштене у сервису су: *onNotificationPosted* и *onNotificationRemoved.*

Метода *onNotificationPosted* омогућава апликацијама да сазнају о новим обавјештењима чим их апликације објаве. Ова метода је кључна за имплементацију сервиса за слушање обавјештења у Андроиду. Параметар који прима дата метода је структура података која енкапсулира оригинални објекат обавјештења, као и информације које га идентификују (ознака - tag и идентификација - id) и извор (име пакета апликације која је објавила обавјештење).

Метода *onNotificationRemoved* омогућава апликацијама да прате и реагују на уклањање обавјештења, што је корисно за одржавање ажурности и релевантности приказаних података. Прима исти улазни параметар као и метода *onNotificationPosted*.

Комбиновањем ових метода, апликације могу да понуде значајно побољшано корисничко искуство кроз правовремено и ефикасно управљање обавјештењима, што доприноси бољој интеракцији корисника са системом и омогућава флексибилност у обради и приказивању обавјештења. Оваква интеграција не само да побољшава функционалност апликација већ и осигурава њихову релевантност и корисност у динамичном окружењу Андроид оперативног система.

```
class NotificationListener : NotificationListenerService() {
    private val viewModel: NotificationViewModel by lazy {
        NotificationViewModel.ViewModelProvider.provideNotificationViewModel()
    }

    override fun onNotificationPosted(sbn: StatusBarNotification?) {
        super.onNotificationPosted(sbn)

        if (sbn != null){
            // ignore foreground notification if it's from teatro service
            if (sbn.notification != null) {
                if (sbn.notification.channelId == TeatroosService.TEATRO_CHANNEL_ID) {
                    return
                }
            }

            if (viewModel.checkIfNotificationNew(sbn)) {
                viewModel.updateNotification(sbn)
            } else {
                // Only push notifications if Do-Not-Disturb is disabled
                if (currentInterruptionFilter == INTERRUPTION_FILTER_ALL)
                    viewModel.addNotification(sbn)
            }
        }
    }

    override fun onNotificationRemoved(sbn: StatusBarNotification?) {
        super.onNotificationRemoved(sbn)
        if (sbn != null) {
            viewModel.removeNotification(sbn)
        }
    }
```

Слика 4.5 Имплементација сервиса за слушање обавјештења

4.2.1 Приказ једног обавјештења

Корисничка спрега која приказује појединачно обавјештење имплементирана је као Compose функција. Основне компоненте ове спреге укључују текст, ред и колону, које заједно омогућавају структурисан и интуитиван приказ информација. Компонента текста служи за приказивање наслова и главног садржаја обавјештења. Компонента реда се користи за хоризонтално распоређивање елемената, што омогућава паралелни приказ иконе апликације и текста, док компонента колоне омогућава вертикално распоређивање, што је корисно за сложеније обрасце обавјештења који захтијевају приказ више линија текста. Икона апликације која шаље обавјештење приказује се у оквиру реда, што визуелно асоцира обавјештење са апликацијом.

```
Column(modifier = Modifier.fillMaxWidth()) { this: ColumnScope
    Row(
        horizontalArrangement = Arrangement.SpaceBetween,
        verticalAlignment = Alignment.Bottom,
        modifier = Modifier.fillMaxWidth()
    ) { this: RowScope
        if (notificationTitle != null) {
            Text(
                text = notificationTitle,
                style = MaterialTheme.typography.subtitle1,
                color = getContentColor(),
                fontSize = MaterialTheme.typography.caption.fontSize,
                overflow = TextOverflow.Ellipsis,
                maxLines = 1
            )
        }
    }
    if (notificationText != null) {
        Text(
            text = notificationText,
            style = MaterialTheme.typography.body1,
            color = getContentColor(),
            fontSize = MaterialTheme.typography.overline.fontSize,
            overflow = TextOverflow.Ellipsis,
            maxLines = 1
        )
    }

}
```

Слика 4.6 Поједностављена Compose функција за приказ обавјештења

4.3 Управљач језиком уноса (engl. Input Manager)

Класа *InputManager* пружа свеобухватне информације о улазним уређајима као и о доступним језицима уноса у Андроид систему. За имплементацију управљања језиком уноса у покретачу апликација, коришћене су методе ове класе које нису дио јавног API-ја, чиме се омогућава дубља интеграција и боље управљање уносом.

Будући да многе методе које пружа *InputManager* нису доступне кроз стандардни јавни API, приступ овим методама из кода покретача је остварен коришћењем рефлексије.

Важно је напоменути да се на циљној Андроид 11 платформи налазе ТВ системска подешавања у оквиру којих не постоји никакво подешавање за језике уноса физичке тастатуре, стога је комплетна функционалност за управљање језиком уноса додана у покретач апликација, што доводи до веће комплексности реализације и потребе за методама које нису доступне кроз стандардни јавни API.

Функција из Input Manager класе	**Опис функције**
setCurrentKeyboardLayoutForInputDevice	Поставља тренутни језик уноса (распоред тастатуре) за одређени улазни уређај.
addKeyboardLayoutForInputDevice	Додаје нови језик уноса за одређени улазни уређај.
removeKeyboardLayoutForInputDevice	Уклања постојећи језик уноса за одређени улазни уређај.
getCurrentKeyboardLayoutForInputDevice	Враћа тренутни језик уноса за специфични улазни уређај.
getEnabledKeyboardLayoutsForInputDevice	Враћа све омогућене језике уноса за одређени улазни уређај.

Табела 4.1 Методе за управљање језиком уноса

Посљедње двије методе из табеле као улазни параметар прихватају идентификатор улазног уређаја, док методе за постављање, додавање и уклањање језика уноса поред идентификатора улазног уређаја као параметар примају и опис распореда тастатуре (engl. descriptor). Опис распореда тастатуре пружа информацију о мапи карактера на тастере (engl. Key Character Map - KCM). KCM одређује како се притисци одређених тастера на тастатури преводе у текстуалне карактере. Такође, омогућава подршку за различите језике и распоред тастатура. На примјер, исти тастер

на тастатури може да производи различите карактере у зависности од тога да ли је изабран енглески, француски или неки други распоред тастатуре.

KCM датотеке су текстуалне датотеке које садрже мапирања између тастера и карактера. Ове датотеке се налазе у системским директоријумима у Андроид уређају и обично имају наставак *.kcm*.

За извршавање метода за постављање, додавање и уклањање језика уноса потребна је дозвола *SET_KEYBOARD_LAYOUT* у оквиру AndroidManifest.xml датотеке апликације.

```
fun setCurrentKeyboardLayoutForInputDevice(
    context: Context,
    descriptor: String,
    vendorId: Int,
    productId: Int,
    keyboardLayoutDescriptor: String
) {
    try {
        val inputManager = context.getSystemService(Context.INPUT_SERVICE) as InputManager

        val inputDeviceIdentifierClass = Class.forName(INPUT_DEVICE_IDENTIFIER_CLASS)

        val identifier = inputDeviceIdentifierClass.getConstructor(
            String::class.java, Int::class.javaPrimitiveType, Int::class.javaPrimitiveType
        ).newInstance(descriptor, vendorId, productId)
        val inputManagerClass = inputManager::class.java
        val method: Method = inputManagerClass.getDeclaredMethod(
            name: "setCurrentKeyboardLayoutForInputDevice",
            inputDeviceIdentifierClass,
            String::class.java
        )

        method.isAccessible = true
        method.invoke(inputManager, identifier, keyboardLayoutDescriptor)
    } catch (e: Exception) {
        e.printStackTrace()
    }
}
```

Слика 4.7 Примјер методе за постављање језика уноса

Покретач апликација садржи сопствену базу података која чува информације о језицима уноса које је корисник додао користећи одговарајућу методу за додавање језика уноса. Језици који су сачувани у овој бази података се приказују у дијалогу изабраних језика уноса, омогућавајући кориснику да лако мијења језике уноса притиском на жељени језик у дијалогу или помоћу пречица на тастатури.

Када корисник дода нови језик уноса, тај језик се одмах чува у бази података. Ако корисник одлучи да обрише неки од изабраних језика, тај језик се одмах уклања из базе података и више се не приказује у дијалогу изабраних језика уноса све док га корисник поново не дода. Једна од кључних предности коришћења базе података за чување информација о језицима уноса је та што ово рјешење омогућава корисницима да задрже своја подешавања језика уноса чак и након гашења или поновног покретања уређаја.

4.3.1 Дијалог изабраних језика уноса

На слици је приказана имплементација корисничке спреге за приказ дијалога изабраних језика уноса.

```
@Composable
fun EnabledLanguages(selectedLanguagesViewModel: SelectedLanguagesViewModel) {
    val selectedLanguages by selectedLanguagesViewModel.getSelectedLanguages()
        .collectAsState(initial = emptyList())
    val currentLanguage = selectedLanguagesViewModel.currentLanguage.value

    LazyColumn(
        modifier = Modifier
            .clip(RoundedCornerShape(ROUNDED_CORNER_SHAPE_AMOUNT))
            .teatroBackground()
            .wrapContentHeight()
            .width(INPUT_LANGUAGE_WIDTH)
            .padding(MODIFIER_PADDING_5),
        verticalArrangement = Arrangement.spacedBy(MODIFIER_PADDING_5)
    ) { this: LazyListScope
        items(selectedLanguages) { this: LazyItemScope languageItem ->
            EnabledLanguageItem(
                language = languageItem.language,
                currentLanguage = if (selectedLanguages.size > 1) currentLanguage.toString() else ""
            )
        }
    }
}
```

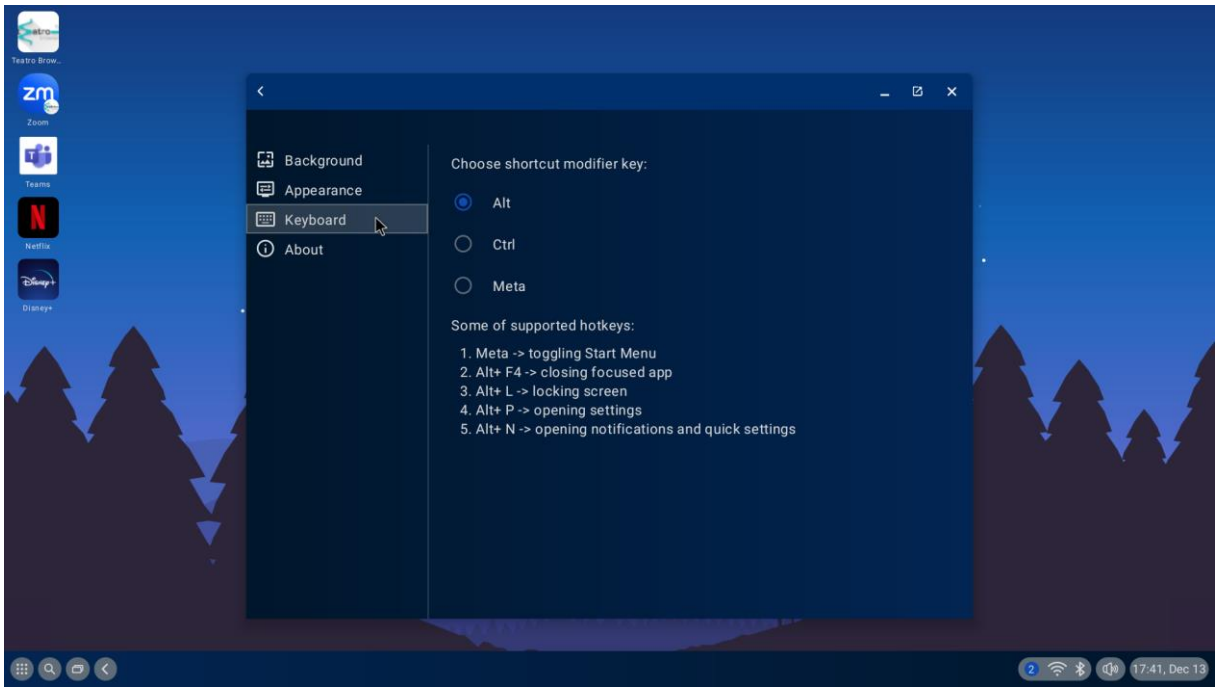
Слика 4.8 Compose функција дијалога језика уноса

4.4 Имплементација пречица са тастатуре

Пречице са тастатуре (engl. hotkeys) су имплементиране како би побољшале корисничко искуство и омогућиле корисницима да помоћу комбинације тастера са тастатуре могу да отворе Андроид подешавања и разне дијалоге у покретачу, да затварају апликације као и да итерирају кроз отворене апликације.

Пречице су имплементиране унутар Андроид SDK-а за циљну платформу и то конкретно у оквиру *PhoneWindowManager* класе. Имплементација је извршена тако што дата класа шаље емисионе поруке покретачу које он прима помоћу примаоца емисионих порука и на тај начин обавља жељене акције.

У подешавањима самог покретача корисник може да изабере модификатор који ће у комбинацији са другим тастерима чинити одређену пречицу са тастатуре. Тастери који могу бити модификатори су: *Alt*, *Ctrl* и *Meta (Windows key)*.



Слика 4.9 Подешавања пречица са тастатуре

На слици је приказано како корисник може да изабере жељени модификатор и у складу са изабраним модификатором корисник има списак омогућених пречица са тастатуре.

У сљедећој табели се налази списак свих имппплементираних пречица са тастатуре као и опис њихових функција.

Пречица са тастатуре	**Функционалност**
Windows тастер	Отвара приказ инсталираних апликација (engl. Start Menu).
Alt + F4	Затвара фокусирану апликацију.
Alt + Tab	Отвара приказ свих покренутих апликација.
Esc	Затвара отворени дијалог покретача.
(Alt, Ctrl, Windows key) + N	Отвара дијалог обавјештења и брзих подешавања.
(Alt, Ctrl, Windows key) + P	Отвара Андроид подешавања.
(Alt, Ctrl, Windows key) + L	Закључава екран.

Табела 4.2 Пречице са тастатуре

```
} else if (keyCode == KeyEvent.KEYCODE_ESCAPE && !isDreaming()) {
    if (down) {
        hotkeyIntent.putExtra(ACTION_TYPE, "closeOverlay");
        mContext.sendBroadcastAsUser(hotkeyIntent, UserHandle.CURRENT);
    }
} else if (isModifierPressed && keyCode == KeyEvent.KEYCODE_N && !isDreaming()) {
    if (down) {
        hotkeyIntent.putExtra(ACTION_TYPE, "toggleNotificationOverlay");
        mContext.sendBroadcastAsUser(hotkeyIntent, UserHandle.CURRENT);
    }
} else if (isModifierPressed && keyCode == KeyEvent.KEYCODE_P && !isDreaming()) {
    if (down) {
        hotkeyIntent.putExtra(ACTION_TYPE, "toggleSettings");
        mContext.sendBroadcastAsUser(hotkeyIntent, UserHandle.CURRENT);
    }
} else if (isModifierPressed && keyCode == KeyEvent.KEYCODE_L && !isDreaming()) {
    if (down) {
        hotkeyIntent.putExtra(ACTION_TYPE, "lock_screen");
        mContext.sendBroadcastAsUser(hotkeyIntent, UserHandle.CURRENT);
    }
}
```

Слика 4.10 Дио имплементације пречица у *PhoneWindowManager* класи

```
when (intent.getStringExtra( name: "type")) {
    // META button
    "toggleAppDrawer" -> teatroosService.toggleDrawer()
    // ALT + TAB
    "openTaskView" -> {
        if (taskViewModel.openedApps.value.isNotEmpty()) {
            teatroosService.removeAllOverlays()
            teatroosService.createTaskMenuOverlay()
            taskViewModel.setTaskFocus(true)
        }
    }
    // ALT + F4
    "closeFocusedApp" -> Util.closeFocusedApp(teatroosService)
    // ESCAPE button
    "closeOverlay" -> closeOverlay(teatroosService)
    // modifier key + N
    "toggleNotificationOverlay" -> teatroosService.toggleNotificationsOverlay()
    // modifier key + P
    "toggleSettings" -> toggleSettings(teatroosService)
    // modifier key + L
    "lock_screen" -> {
        TeatroosAccessibilityService.instance!!.performGlobalAction(AccessibilityService.GLOBAL_ACTION_LOCK_SCREEN)
        teatroosService.removeAllOverlays()
    }

    else -> return
}
```

Слика 4.11 Дио примаоца емисионих порука из покретача

4.5 Управљач корисницима (engl. User Manager)

Класа *UserManager* омогућава управљање корисницима и њиховим детаљима у оквиру вишекорисничког система. У основи, постоје двије главне категорије корисника: потпуно прилагодљиви корисници са сопственим пријавама и профили који дијеле радни простор са повезаним корисником.

Корисници се разликују од налога, којима управља *AccountManager*. Сваки корисник може имати свој скуп налога, што омогућава потпуну персонализацију и независност у коришћењу апликација и сервиса.

Методе које служе за управљање корисницима нису дио јавног API-ја, због тога је за приступ њима из покретача коришћена рефлексија.

Поред класе *UserManager* за имплементацију је коришћена и класа *UserHandle* која служи за представљање корисника на уређају.

Како би се омогућио вишекорисnички систем, извршене су промјене унутар Андроид SDK-ја за циљну платформу. Дате промјене су осигурале да систем подржава максимално пет корисника.

Функција из UserManager класе	**Опис фунцкије**
getUsers()	Враћа листу објеката типа *UserInfo*, који представљају све кориснике на уређају.
removeUser(int userId)	Омогућава брисање корисника и његових профила заједно са свим асоцираним подацима.
setUserName(int userId, String name)	Служи за ажурирање имена корисника.
getMaxSupportedUsers()	Враћа максималан број корисника који се могу креирати на датом уређају.

Табела 4.3 Методе за управљање корисницима

Поред метода из табеле, коришћена је метода *startUserInBackground* класе *ActivityManagerService*. Дата метода служи за покретање корисника система.

За пребацивање између корисника коришћена је метода *switchUser* из класе *ActivityManager*, а за креирање корисника метода *createProfileForUserWithThrow* из класе *UserManagerService*.

```
fun startUser(userId: Int) {

    val activityManagerService =
        ActivityManager::class.java.getMethod( name: "getService").invoke(null)

    activityManagerService::class.java.getMethod(
        name: "startUserInBackground", Integer::class.javaPrimitiveType
    ).invoke(activityManagerService, userId)
}

fun getMaxUsers(): Int {
    val maxNumberOfUsers =
        UserManager::class.java.getMethod( name: "getMaxSupportedUsers").invoke(null) as Int
    return maxNumberOfUsers.coerceAtMost( maximumValue: 5)
}
```

Слика 4.12 Метода за покретање корисника и метода за добијање максималног броја корисника

```
/* The box in which the users themselves are placed. */
@Composable
fun UserListView(usersViewModel: UsersViewModel, userManagementState: Int) {
    val users = usersViewModel.users.collectAsState()

    LazyRow(horizontalArrangement = Arrangement.spacedBy(SPACE_BETWEEN_USER_AVATARS)) { this: LazyListScope
        if (userManagementState == TeatroUserManager.STATE_USER_SELECTION) {
            // New user button - should be displayed only on user selection screen
            item { this: LazyItemScope
                UserView(
                    user = null, userManagementState = userManagementState
                ) // new user (+) button
            }
        }
        items(users.value.size) { this: LazyItemScope  it: Int
            // display all other users
            // @todo Remove this check if and when we decrease max number of users in SDK
            if (it <= MAX_NUMBER_OF_USERS_TO_DISPLAY) {
                UserView(user = users.value[it], userManagementState = userManagementState)
            }
        }
    }

}
```

Слика 4.13 Compose функција за приказ листе корисника

4.6 Брза подешавања

Имплементација плочица за брза подешавања (engl. tiles) је омогућена тако што је у покретачу направљена једна класа *AbstractBaseTile* коју затим насљеђује свака класа за појединачно подешавање.

У покретачу су имплементиране сљедеће класе за брза подешавања: *BluetoothTile*, *WifiTile*, *WifiHotspotTile*, *EthernetTile*, *ScreenshotTile*, *NightModeTile*, *DoNotDisturbTile*. Методе *AbstractBaseTile* класе:

- **updateState(state: Int)** – ажурира стање плочице које може бити *STATE_INACTIVE*, *STATE_ACTIVE* и *STATE_WIFI_SAVED*.
- **isEnabled()** – враћа вриједност „тачно“ ако је стање активно или ако је WiFi сачуван.
- **onClick()** – регулише клик на плочицу појединачног подешавања
- **refresh()** – мијења стање плочице

```kotlin
data class Tile(
    var state: Int = 0,
    val label: String,
    var secodaryLabel: String? = null,
    val contentDescription: String? = null,
    var icon: Icon? = null,
) {
    companion object {
        val STATE_INACTIVE = 1
        val STATE_ACTIVE = 2
        val STATE_WIFI_SAVED = 10
    }
}

abstract class AbstractBaseTile() {
    protected abstract val _baseTile: MutableStateFlow<Tile>
    abstract val baseTile: StateFlow<Tile>

    fun updateState(state: Int) {
        val currentValue = _baseTile.value
        val updated = currentValue.copy(state = state)
        _baseTile.value = updated
    }

    fun isEnabled(): Boolean {
        return baseTile.value.state == STATE_ACTIVE || baseTile.value.state == STATE_WIFI_SAVED
    }

    abstract fun onClick()
    abstract fun refresh()
}
```

Слика 4.14 Abstract Base Tile

4.6.1 Управљач енергијом (engl. Power Manager)

Методе класе PowerManager су кориштене како би било омогућено гашење и поновно покретање уређаја директно из покретача. Метода за гашење уређаја није дио јавног API-ја и зато је за њену имплементацију кориштена рефлексија док за методу која поново покреће уређај то није било потребно.

На сљедећим сликама је приказана реализација ових функција унутар Compose функције која покреће дијалог задужен за потврђивање да ли корисник жели да угаси/поново покрене уређај или не.

```
// power off
TeatroCircledIconButton(
    drawableId = R.drawable.ic_shutdown,
    modifier = Modifier.padding(NAS_BUTTON_INNER_ICON_PADDING)
)
{
    val pm = context.getSystemService(Context.POWER_SERVICE) as PowerManager
    val method = pm.javaClass.getMethod(
        name: "shutdown",
        Boolean::class.javaPrimitiveType,
        String::class.java,
        Boolean::class.javaPrimitiveType
    )
    if (service.powerView?.windowToken == null || service.powerView == null) {
        service.createPowerDialog(
            text = powerOffText,
            action = {
                method.invoke(
                    pm,
                    false,
                    "User action for shutdown",
                    false
                )
            })
        service.removeOverlay(service.notificationView)
```

Слика 4.15 Примјер за гашење уређаја

```
// reboot
TeatroCircledIconButton(
    drawableId = R.drawable.ic_reboot,
    modifier = Modifier.padding(NAS_BUTTON_INNER_ICON_PADDING)
) {
    if (service.powerView?.windowToken == null || service.powerView == null) {
        val pm = context.getSystemService(Context.POWER_SERVICE) as PowerManager
        service.createPowerDialog(
            text = rebootText,
            action = { pm.reboot( reason: "user action for reboot") })
        service.removeOverlay(service.notificationView)
    } else {
        service.powerView?.visibility = View.VISIBLE
    }
}
```

Слика 4.16 Примјер за поновно покретање

5. Резултати

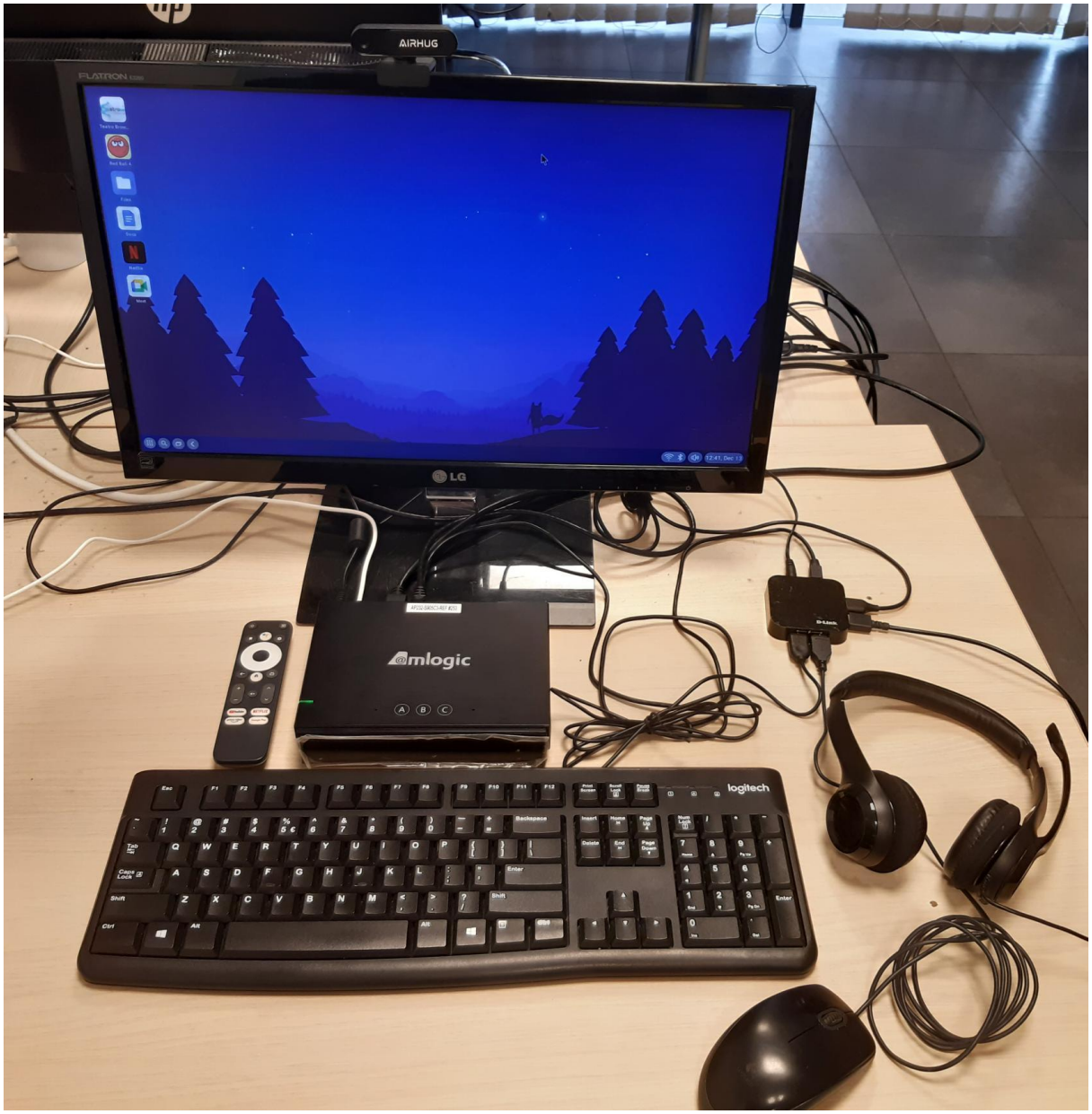
У овом поглављу представљени су резултати интеграције покретача апликација са основним подсистемима Андроид оперативног система. Презентовани резултати обухватају функционалност покретача на различитим Андроид платформама, као и на различитим верзијама Андроид оперативног система. Приказана су побољшања у корисничком искуству и стабилности система, те су анализиране перформансе покретача у контексту различитих хардверских конфигурација и верзија Андроид ОС-а.

5.1 Резултати на циљној Андроид платформи

Циљна Андроид платформа коришћена за овај пројекат је *Amlogic*. Односно, коришћене су *Amlogic Oppen (S905Y4)* и *Amlogic Ohm (S905X4)* плоче. Њихове техничке спецификације су:

- CPU: четворојезгарни ARM Cortex-A35 процесор
- GPU: ARM Mali-G31 MP2 графичка процесорска јединица
- Меморија: 32-битне DDR3, DDR4, DDR3L LPDDR3/4
- Видео декодирање: подржавају AV1, H.265, VP9 и AVS2-P2 кодеке до 4k резолуције при 60 кадрова у секунди.
- Видео кодирање: S905X4 подржава H.265 и JPEG кодирање до 1080p при 60 кадрова у секунди, док S905Y4 користи софтверске кодере.
- HDMI-TX: подржавају 4k резолуцију при 60Hz.
- AB излаз: CVBS (композитни видео излаз)

Имплементација рјешења је првобитно тестирана на Андроиду 11, с циљем да интеграција покретача са основним Андроид подсистемима буде стабилна и поуздана на овој верзији оперативног система. Током тестирања, посебна пажња је посвећена компатибилности и перформансама како би се осигурало да све функционалности покретача раде беспрјекорно и да корисничко искуство буде оптимално.



Слика 5.1 Развојна платформа повезана са спољашњим уређајима

На слици је приказана развојна платформа на коју су повезани миш, тастатура, слушалице и камера. Дати спољашњи уређаји омогућавају тестирање имплементације рјешења интеграције покретача са Андроид подсистемима.

Мануелним тестирањем је показано да на циљној платформи са Андроидом 11 интеграција покретача са сваким појединачним Андроид подистемом функционише исправно. То значи сљедеће:

- Корисник може успјешно да промијени јачину звука из покретача и да јасно на слушалицама чује да се јачина звука промијенила, а такође ако мијења јачину звука преко слушалица може у покретачу видјети како се клизач звука мијења.
- Могуће је додавање више корисника, сваки корисник може да персонализује Андроид покретач, а да његове промјене остану сачуване.
- Корисник може да прима обавјештења са разних апликација укључујући и друштвене мреже.
- Корисник може да изабере највише пет језика уноса и да се брзо пребацује између њих користећи пречице са тастатуре или дијалог језика уноса.
- Такође, све имплементиране пречице са тастатуре су успјешно тестиране и показано је да раде.
- Брза подешавања су функционална и вјерно одражавају стање уређаја.
- Корисник може да гаси/поново покреће уређај директно из покретача.

Након низа тестирања и унапређивања, како у погледу функционалности тако и корисничког искуства, постигнуто је стабилно рјешење интеграције покретача апликација са основним Андроид подсистемима. Овај процес је укључивао темељну евалуацију и оптимизацију свих компоненти, што је резултирало унапријеђеним и поузданим системом који задовољава високе стандарде квалитета и перформанси.

5.1.1 Прилагођавање рјешења за Андроид 14

Рјешење интеграције покретача апликација са основним Андроид подсистемима је успјешно прилагођено и за Андроид 14 верзију на *Amlogic* платформи. Иако рјешење није стабилно као на Андроиду 11, функционалност је задржана. Прилагођавање је захтијевало адаптацију промјена, извршених унутар Андроид SDK-а, новој верзији Андроида, као и додатне промјене специфичне за Андроид 14 SDK које нису биле потребне на Андроиду 11.

5.2 Прилагођавање рјешења другим платформама

Рјешење је поред *Amlogic* плоче тестирано и на другим плочама. Андроид платформе на којима је рјешење тестирано укључују *Realtek STB, Realtek TV, Brooklyn, Haier*. На свим овим платформама постигнута је жељена функционалност покретача уз минималне промјене и прилагодбе специфичне за сваку конкретну платформу. Ово указује на висок степен преносивости и флексибилности рјешења, омогућавајући интеграцију покретача апликација са основним Андроид подсистемима на различитим хардверским конфигурацијама уз очување кључне функционалности и корисничког искуства.

6. Закључак

У овом раду имплементирана је интеграција покретача апликација са основним Андроид подсистемима. Описан је првенствено концепт самог рјешења, а потом и програмско рјешење са кориштеним класама и функцијама, а описане су и промјене извршене унутар самог Андроид SDK-а за циљну платформу.

Имплементација је започела детаљном анализом постојећих Андроид подсистема и дефинисањем захтјева за интеграцију са покретачем апликација. Посебна пажња је посвећена корисничком искуству и интуитивности корисничке спреге, што је постигнуто кроз пажљиво дизајнирање и тестирање корисничке спреге. Кориштење савремених програмских метода и алата омогућило је развој стабилног и функционалног рјешења које задовољава све постављене захтјеве.

Резултати тестирања показали су да је рјешење способно за стабилну и функционалну интеграцију на разним Андроид платформама уз минималне прилагодбе. Ово указује на висок степен преносивости и флексибилности рјешења, омогућавајући његову примјену на различитим хардверским конфигурацијама.

Један од значајних доприноса овог рада је и прилагођавање рјешења за новије верзије Андроид оперативног система, конкретно за Андроид 14. Упркос изазовима који су се јавили због промјена унутар Андроид SDK-а, рјешење је успјешно прилагођено и стабилно функционише на циљној платформи.

Даљи правци развоја могу укључивати побољшање корисничког искуства, односно корисничке спреге, унапређивање већ постојећих функционалности покретача, а такође и додавање нових функционалности. Примјер за нову функционалност која би могла бити додана је могућност закључавања екрана тако да сваки корисник има своју шифру. Ово би омогућило већу сигурност и приватност за кориснике, посебно у сценаријима гдје више корисника дијели један уређај.

Закључно, рад је пружио детаљан преглед и анализу процеса интеграције покретача апликација са основним Андроид подсистемима, показујући како савремене методе и алати могу бити кориштени за развој флексибилних и преносивих софтверских рјешења. Резултати овог рада могу послужити као основа за будућа истраживања и развој у области интеграције софтвера са оперативним системима за мобилне уређаје.

7. Литература

[1] Android Developer: https://developer.android.com/ , учитано 29.6.2024.

[2] Kotlin Docs: https://kotlinlang.org/docs/home.html , учитано 29.6.2024.

[3] Kotlin Programming Language: https://kotlinlang.org/education/ , учитано 29.6.2024.

[4] Jetpack Compose: https://developer.android.com/develop/ui/compose , учитано 29.6.2024.

[5] Wikipedia: https://en.wikipedia.org/wiki/Android_(operating_system), учитано 29.6.2024.

[6] Karim Yaghmour: *"Embedded Android: Porting, Extending and Customizing"* O'Reilly Media, 2013

[7] Dawn Griffiths, David Griffiths: *"Head First Kotlin: A Brain-Friendly Guide"* O'Reilly Media, 2019

[8] Dawn Griffiths, David Griffiths: *"Head First Android Development: A Brain-Friendly Guide"* O'Reilly Media, 2017