Nativní Android aplikace s využitím rozšířené reality zobrazující informace o zájmových bodech

Bc. Martin Mlýnek

Diplomová práce
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav počítačových a komunikačních systémů

Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Martin Mlýnek**
Osobní číslo:	**A17243**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Počítačové a komunikační systémy**
Forma studia:	**Prezenční**
Téma práce:	**Nativní Android aplikace s využitím rozšířené reality zobrazující informace o zájmových bodech**
Téma práce anglicky:	**A Native Augmented Reality Android Application for Displaying Information about Points-of-Interest**

Zásady pro vypracování

1. Nastudujte a popište možnosti nativního vývoje mobilních aplikací pro platformu Android a zaměřte se na nástoje pro práci s rozšířenou realitou.
2. Sepište funkční a nefunkční požadavky na reálnou aplikaci, která bude vhodně demonstrovat využití rozšířené reality v mobilním zařízení.
3. Navrhněte aplikaci tak aby využívala detekce bodů zájmu a za využití kamery poskytovala další uživatelské informace v rámci rozšířené reality.
4. Proveďte praktickou implementaci vycházející ze stanovených požadavků a případů užití.
5. Prakticky otestujte aplikaci na mobilním zařízení s operačním systémem Android.

Rozsah diplomové práce:
Rozsah příloh:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. PHILLIPS, Bill. Android Programming: The Big Nerd Ranch Guide. 3rd ed. Atlanta (Georgia): Big Nerd Ranch, 2017. ISBN 9780134706054.
2. MURPHY, Mark. The Busy Coder's Guide to Android Development. FINAL version. United States: CommonsWare, 2019. ISBN 9780981678009.
3. Google Inc., Android Developers. [online]. Mountain View [cit. 2019-11-27]. Dostupné z: https://developer.android.com/
4. MEW, Kyle. Android Design Patterns and Best Practice. Birmingham (United Kingdom): Packt Publishing, 2016. ISBN 9781786465917.
5. SMYTH, Neil. Android Studio 3.0 Development Essentials – Android 8 Edition. Scotts Valley: CreateSpace Independent Publishing Platform, 2017. ISBN 9781977540096.
6. LACKO, Ľuboslav. Mistrovství – Android. Brno: Computer Press, 2017. ISBN 9788025148754.
7. LINOWES, Jonathan. Augmented Reality for Developers: Build practical augmented reality applications with Unity, ARCore, ARKit, and Vuforia. Birmingham (United Kingdom): Packt Publishing, 2017. ISBN 9781787286436.

Vedoucí diplomové práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **13. prosince 2019**
Termín odevzdání diplomové práce: **29. května 2020**

doc. Mgr. Milan Adámek, Ph.D.
děkan

Ing. Miroslav Matýsek, Ph.D.
ředitel ústavu

Ve Zlíně dne 9. prosince 2019

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 14. 8. 2020 Bc. Martin Mlýnek, v. r.

ABSTRAKT

Tato diplomová práce se zabývá vývojem mobilní aplikace pro operační systém Android využitelné pro vzdělávací účely, a také jako motivace k fyzické aktivitě jejích uživatelů. Práce je rozdělena na dvě části. Teoretická část si dává za cíl popsat operační systém Android, jeho architekturu a možnosti vývoje aplikací pro tento systém. Část práce se také věnuje problematice rozšířené reality, její využití a možnosti. Praktická část je věnována realizaci takové aplikace dle stanovených požadavků a popisu vzhledu a funkcí.

Klíčová slova: Android, vývoj mobilních aplikací, rozšířená realita

ABSTRACT

This Master's thesis deals with mobile application development for operating systém Android used for education purposes, and also as a motivation for physical aktivity of its users. The thesis is dived into two parts. The theoretical part aims to describe operating systém Android, its architecture and the options of mobile application development for this systém. Part of this thesis is devoted to problematics of augmented reality, its usage and options. The practical part is dedicated to the realization of such an application according to the set requirements and a description of its design and functions.

Keywords: Android, mobile application development, augmented reality

Tímto bych rád poděkoval vedoucímu této práce Ing. Radku Valovi, Ph.D. za pomoc a trpělivost při vypracovávání této diplomové práce.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Během posledních několika let zaznamenaly mobilní zařízení obrovský vzestup popularity a využití. Smartphony neboli chytré telefony již téměř úplně nahradily klasické telefony a poskytují svým uživatelům dříve na těchto zařízeních nevídané možnosti jako mobilní platby, ovládání chytrých domovů, navigace či práce s dokumenty a neomezují se tak pouze na uskutečnění hlasových hovorů a posílání SMS zpráv.

Jsou to však nejen chytré telefony, které za těmito změnami stojí. Svým dílem přispěly také tablety, chytré hodinky, čtečky knih a další zařízení využívající některý z mobilních operačních systémů. Tato zařízení téměř vždy poskytují také neustálé připojení k internetu, což velmi přispívá k možnostem jejich využití. Zařízení již nejsou omezena pouze pravidelnými aktualizacemi, nýbrž dovedou využívat informace uložené na úložišti vzdáleném třeba i půlku světa a nejen to, dokážou i reagovat na změny těchto dat téměř v reálném čase.

Tyto nové technologie však mají i své temné stránky, především v oblasti socializace a nedostatečné fyzické aktivity člověka. Je to způsobeno především tím, že spoustu věcí člověk zvládne zařízením na dlani své ruky.

Cílem této práce je vyvinout takovou aplikaci pro operační systém Android, která umožňuje zobrazit edukační informace o vizuálně detekovaných zájmových bodech. Toto zobrazení probíhá v zábavnou formou v rámci rozšířené reality. Aplikace dále využitím technik gamifikace uživatele motivuje k jejímu používání, což vede ke vzdělávání zábavnou formou.

Teoretická část této práce se zabývá obecným popisem samotného operačního systému Android, pro nějž je aplikace vyvíjena. Dále jsou popsány typy vývoje mobilních aplikací pro Android a jsou mezi sebou porovnány. Poté se část věnuje popisu základních komponent aplikace a vybraným komponentám architektury. Zbytek teoretické části je věnován popisu virtuální a rozšířené reality a jejich vývojových nástrojů.

Praktická část začíná definováním funkčních a nefunkčních požadavků na vyvíjenou aplikaci s využitím rozšířené reality a dále popisuje její návrh a implementaci na základě získaných poznatků z teoretické části dle těchto požadavků. Závěr praktické části je věnován praktickému testování aplikace.

I. TEORETICKÁ ČÁST

1 OPERAČNÍ SYSTÉM ANDROID

Chytré telefony jsou mobilní zařízení, které kombinují telefonní a výpočetní funkce v jediném, přenosném zařízení. Od obyčejných telefonů se odlišují tím, že na nich běží operační systém a také tím, že jejich hardware je podstatně výkonnější.

Historie chytrých telefonů sahá až do první poloviny 90. let. V roce 1992 vyšel IBM Simon, který v mnohém předběhl svou dobu. Ten byl následován řadou telefonů nejčastěji se systémy Symbian, Windows Mobile a Palm OS. Adopce chytrých telefonů širokou veřejností však začala až s příchodem operačních systémů iOS od společnosti Apple v roce 2007 a Android od společnosti Google roku 2008. [1]

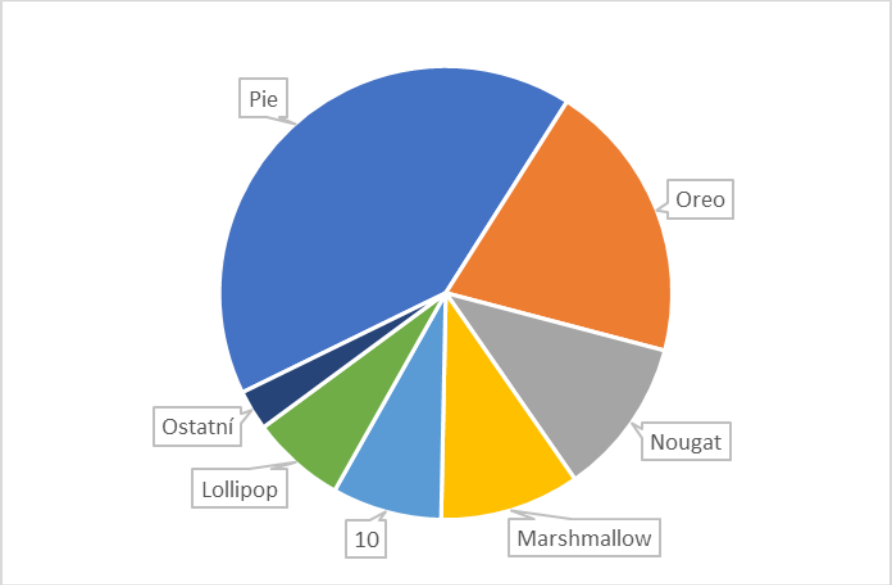
Od té doby se Android stal nejen nejpoužívanějším operačním systémem na mobilních telefonech, ale s podílem 39,67 % předčí dokonce Microsoft Windows a je tak nejpopulárnějším uživatelským operačním systémem vůbec. [2] Tomu vděčí z malé části také svým rozšířením na další platformy, jako televize, tablety, chytré hodinky aj.

Android byl původně vyvíjen společností Android Inc. jako chytřejší náhrada pro systémy digitálních fotoaparátů. Po převzetí firmou Google Inc. se projekt začal vyvíjet jako otevřený neboli open-source projekt pod názvem AOSP (Android Open Source Project). [3]

Tento projekt však ještě není jeho verzí tak jak jej známe. Slouží výrobcům zařízení dodávaných s Androidem, kteří si jej uzpůsobují své vizi.

1.1 Podíl verzí Androidu na trhu

Android, za svých dvanáct let existence, již vyšel v mnoha verzích. Každá další verze staví na předchozí a přidává nové funkce, vylepšuje zabezpečení a dalšími změnami vylepšuje uživatelův požitek z jejího užívání. Při tvorbě aplikace je nutné předem zvolit minimální podporovanou verzi systému. Toto by mělo vycházet z požadavků na funkcionalitu samotné aplikace, aby tato verze nebyla zbytečně vysoká a větší, než nutný počet uživatelů by tak aplikaci nemohl využívat.



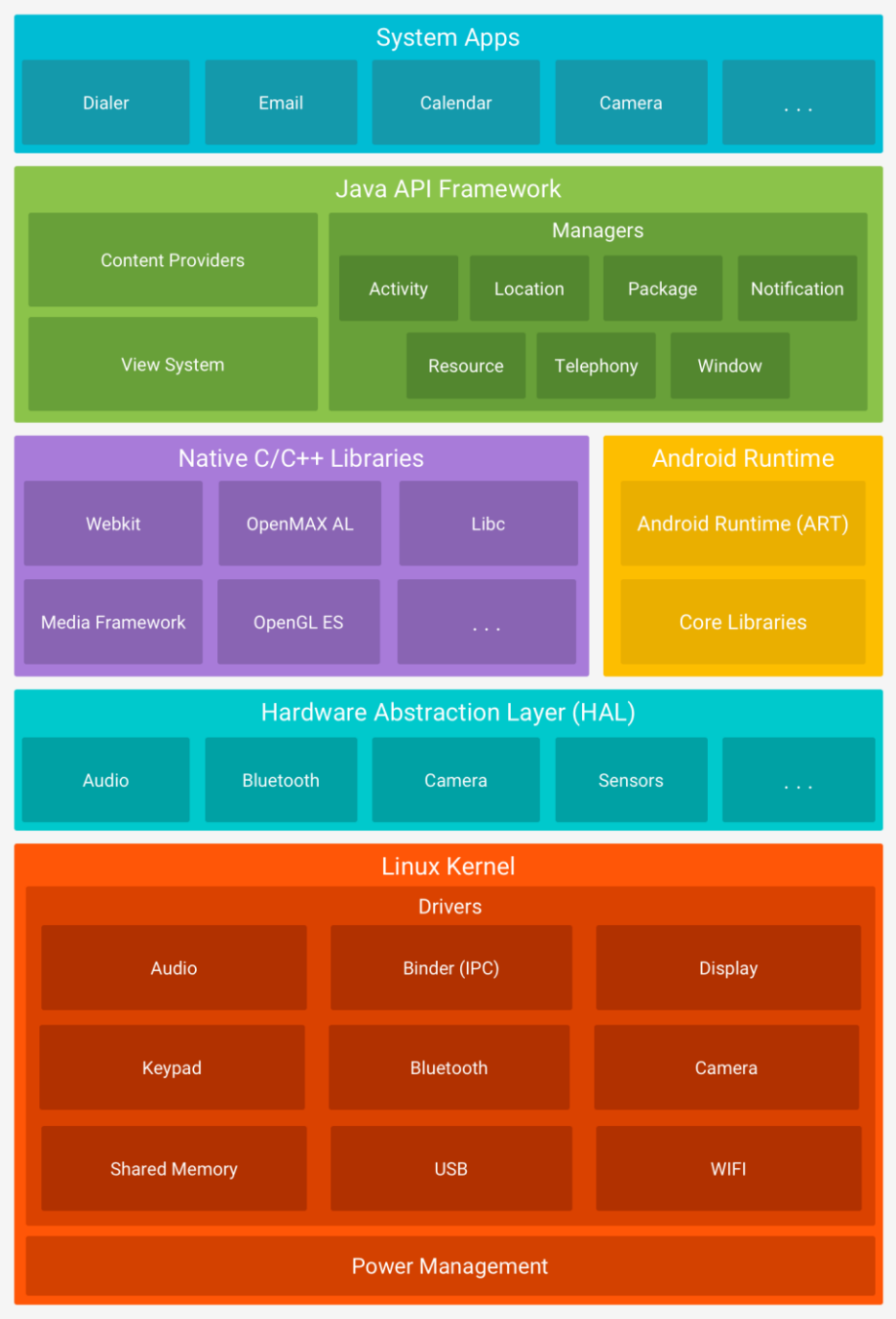
Obr. 1. Procentuální zastoupení verzí Androidu na trhu [4]

Tab. 1. Procentuální zastoupení verzí Androidu na trhu [4]

Verze	Kódové jméno	API	Zastoupení
10.0	10	29	7,84 %
9.0	Pie	28	41,22 %
8.1	Oreo	27	12,15 %
8.0		26	7,94 %
7.1	Nougat	25	4,58 %
7.0		24	6,67 %
6.0	Marshmallow	23	9,99 %
5.1	Lollipop	22	5,39 %
5.0		21	1,35 %
Other			2,87 %

1.2 Architektura

Android má velkou výhodu v tom, že může stavět na dlouholetých zkušenostech při vývoji operačních systémů na Linuxovém jádře. Android byl tedy vyvinut s modulární architekturou, což mu dodává silné zabezpečení proti bezpečnostním hrozbám ať již záměrným či nezáměrným a také adaptibilitu, poněvadž jednotlivé moduly, ze kterých se Android skládá, jsou vyměnitelné. Tyto moduly se shlukují do vrstev, z nichž každá má svůj účel.



Obr. 2. Architektura operačního systému Android [5]

1.2.1 Linux Kernel

V jádru samotného systému se nachází Linux Kernel. Jako u každého operačního systému, jeho jádro se stará o služby na nízké úrovni jako správa paměti, vláken a napájení. Kernel také umožňuje Androidu využít jeho klíčových bezpečnostních vlastností. Tato vrstva komunikuje přímo s fyzickými prvky zařízení.

1.2.2 Hardware Abstraction Layer (HAL)

Tato vrstva poskytuje standardní rozhraní, které dávají přístup vysokoúrovňovému Java API Frameworku k hardwarovým možnostem zařízení. HAL se skládá z množství modulů knihoven, z nichž každá implementuje rozhraní ke specifické hardwarové komponentě jako kamera, Bluetooth nebo akcelerometr. Když Java API Framework si vyžádá přístup k některé komponentě tak Android načte její příslušný modul. [5]

1.2.3 Android Runtime

Android Runtime (ART) komponenta je navržena tak, aby spouštěla všechny aplikace a služby. Skládá se z běhového prostředí a Core Java knihoven. Tyto knihovny obsahují většinu funkcionality Javy a jsou nutné pro Java API.

Dřívější verze Androidu používaly místo ART komponenty Dalvik. Hlavní rozdíl mezi těmito komponentami spočíval v tom, že Dalvik pro kompilaci Java bytekódu do strojového využíval proces jménem Just-In-Time (JIT), který tuto kompilaci prováděl vždy přispuštění aplikace. V kontrastu s tím ART využívá kompilaci Ahead-Of-Time (AOT), která tuto kompilaci provede jednou, a to při instalaci aplikace. Toto zrychlí čas potřebný pro spuštění aplikace, ale prodlouží čas pro její instalaci. [5]

1.2.4 Native C/C++ Libraries

Spousta systémových Android komponent a služeb jako ART a HAL jsou postaveny na nativním kódu, který vyžaduje nativní knihovny v jazycích C a C++. Některé tyto knihovny, jako například OpenGL a SSL, lze využít i při vývoji uživatelské aplikace. Je to výhodné především pro velmi výpočetně náročné úlohy jako kryptografie nebo 3D grafika.

1.2.5 Java API Framework

Všechny Android funkce, které nejsou poskytovány standardně knihovnami Java jsou obsaženy v této vrstvě. Aplikace jsou vyvíjeny s výrazným využitím množství lehce nahraditelných, zaměnitelných a znovu použitelných komponent implementovaných touto vrstvou. Jsou zde obsaženy moduly jako *ResourceManager, ContentProvider, NotificationManager, ActivityManager* aj. [5]

1.2.6 System Apps

Nejvyšší vrstvou architektury jsou System Apps. Zde jsou obsaženy dva typy aplikací. Prvním typem jsou aplikace již předinstalované na zařízení a většina je jich nutná k jeho použití a typicky je nelze odinstalovat. Spadá sem například aplikace pro nastavení nebo telefonování. Druhým typem jsou uživatelské aplikace. To mohou být hry, navigace aj. Tento typ není vyobrazen na Obr. 2. protože se nejedná o nutnou část architektury. Všechny aplikace jsou spouštěny uvnitř vlastního virtuálního prostředí.

2 VÝVOJ APLIKACÍ PRO ANDROID

Aby byl každý operační systém určený pro širokou veřejnost kompetitivní, je nutné umožnit třetím stranám vyvíjet pro ně aplikace. K tomuto účelu vzniklo několik možností vývoje – od nejsnazších až po složitější.

2.1 Typy vývoje mobilních aplikací

Mobilní aplikace jsou softwarové aplikace vytvořené za cílem použití na jednom nebo více mobilních operačních systémů pohánějících mobilní telefony, hodinky, tablety, autorádia, televize aj. Protože těchto systémů i zařízení je více typů, existuje také více možností vývoje aplikací. Ty se dělí do kategorií Web aplikace, Hybridní aplikace a Nativní aplikace.

2.1.1 Web aplikace

Na rozdíl od všech ostatních aplikací jsou webové vyvinuty pomocí webových technologií jako HTML5, CSS a JavaScript. Web aplikace jsou vhodné pro velmi jednoduché použití, protože se příliš neliší od obyčejné webové stránky – nelze využít funkce telefonu. Tuto nevýhodu se snaží minimalizovat nový typ zvaný Progresivní Webové aplikace. Ten umožňuje některé funkce telefonu využít, nicméně je jejich výběr velmi omezený, a ne všechny funkce fungují na všech operačních systémech. [6]

2.1.1.1 Výhody web aplikací:

- nízká nákladnost
- snadný vývoj aplikace
- není je třeba instalovat
- multiplatformní

2.1.2 Hybridní aplikace

Hybridní aplikace jsou aplikace tvářící se z vnějšího pohledu jako aplikace nativní, nicméně používá i prvky shodné mezi platformami. Jsou to vlastně web aplikace zaobalené do nativního kódu, který má možnost využít omezený počet funkcí daného zařízení. Jsou tedy takovým mezikrokem mezi web aplikacemi a nativními aplikacemi. Je vhodné je využít pro jednoduché aplikace, kdy však už možnosti webových nestačí. [7]

2.1.2.1 Výhody hybridních aplikací

- z většiny multiplatformní kódová báze – není tak třeba, v případě vývoje pro více platforem, udržovat velké množství kódu
- nižší obtížnost vývoje

2.1.3 Nativní aplikace

Nativní aplikace je aplikace specificky navržená pro daný operační systém, která musí být nainstalována přímo na zařízení. To znamená, že aplikace vyvinuta pro Apple iOS nefunguje na OS Android a naopak.

2.1.3.1 Výhody nativních aplikací:

- snadný přístup k hardwaru a jiným funkcím zařízení jako kamera, geolokace, push notifikace
- využití nejnovějších funkcí OS
- rychlé a plynulé uživatelské prostředí
- rozsáhlé možnosti monetizace aplikace
- podpora náročných grafických prvků
- obsáhlá dokumentace

Ačkoliv je nejnáročnější a ostatní postupy udělaly velký pokrok během posledních několika let, je nativní postup nejlepší volbou pro vývoj většiny aplikací, především pro aplikace využívající funkce pokročilé grafiky a virtuální nebo rozšířené reality. Z tohoto důvodu se zbytek této kapitoly i této práce bude věnovat vývoji nativní aplikace.

2.2 Základní komponenty Android aplikace

Každá nativní aplikace operačního systému Android obsahuje komponenty, které zprostředkovávají funkcionalitu nebo design. Tato kapitola se věnuje vybraným komponentám, které najdou využití v téměř každé aplikaci.

2.2.1 Aktivita

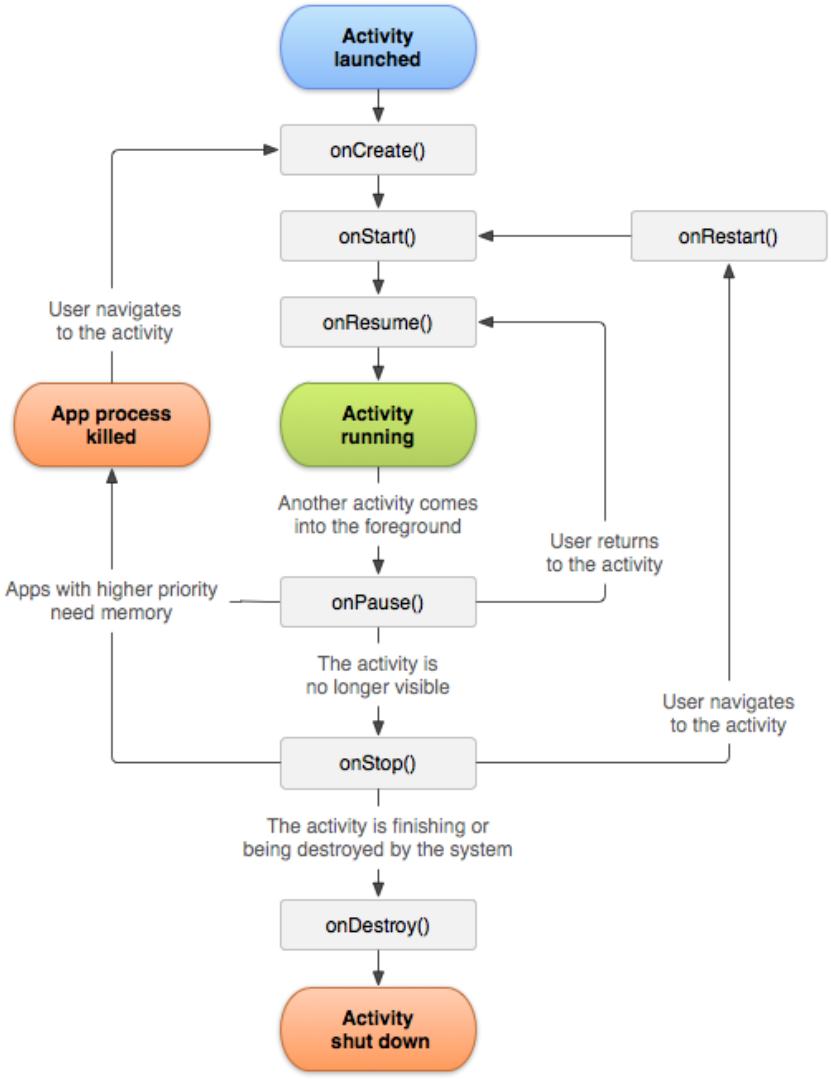
Nejčastěji využívanou komponentou Androidu je Aktivita. Každá aplikace obsahuje minimálně jednu aktivitu a je to vždy první věc která se po otevření aplikace spustí. Jedna aktivita představuje jednu obrazovku s uživatelským rozhraním. Například aplikace emailového klienta obsahuje jednu aktivitu jako seznam přijatých emailů a další aktivita zprostředkovává

psaní emailu. Aktivita uživateli zobrazuje tzv. Views, s jejichž interakcí komunikuje s aktivitou. Pokud se aplikace skládá z více aktivit, je při spuštění nové aktivity stávající aktivita zastavena a uložena na začátek zásobníku paměti, přičemž aktivita nová ji nahradí na popředí. Stará aktivita čeká na zničení buď systémem, vypnutím aplikace nebo na její opětovné vyvolání do popředí uživatelem. Jednotlivé aktivity na sobě nejsou závislé.

Každá aktivita má tyto tři hlavní stavy:

- Resumed – aktivita běží na popředí a přebírá instrukce od uživatele
- Paused – jiná aktivita je na popředí, ale tato je stále viditelná
- Stopped – aktivita je zastavena a není viditelná

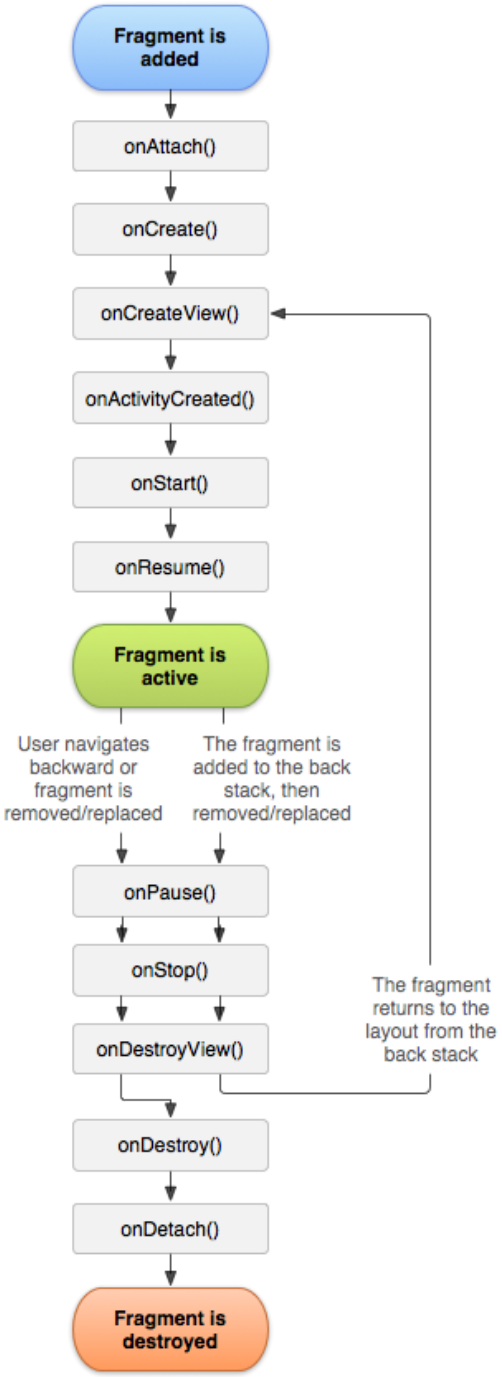
Životní cyklus aktivity a těchto a dalších stavů je na Obr. 3. Aktivita při přechodu do každého stavu spouští k němu korespondující metodu. Implementací těchto metod a prací s nimi se zajišťuje správný a očekávaný chod aplikace. Například před zničením aktivity je vhodné uložit data, aby o ně uživatel nepřišel. [7]



Obr. 3. Diagram životního cyklu aktivity [7]

2.2.2 Fragment

Fragmenty se v mnohém podobají aktivitám. Fragment představuje modulární a vícekrát použitelnou část uživatelského rozhraní uvnitř aktivity. Jeho životní cyklus je tak svázán s životním cyklem aktivity, nicméně i fragment má cyklus vlastní a pro správný chod aplikace a zabránění ztráty dat je nutné pracovat s jeho metodami. [8] Životní cyklus fragmentu je na Obr. 4.



Obr. 4. Diagram životního cyklu fragmentu [8]

2.2.3 Layout, View a ViewGroup

Vizuální struktura aplikace bývá zpravidla definována layoutem. Layout definuje rozložení jednotlivých prvků uživatelského rozhraní. Lze jej vytvořit buď za běhu aplikace programově, nebo spíše deklarovat XML souborem.

View je třída obsahující všechny jednotlivé vizuální prvky ze kterých se skládá uživatelské rozhraní jako Button, TextView, EditText atd.

ViewGroup je podtřída třídy View. Její instance fungují jako kontejnery pro více instancí třídy View a definují některé jejih vlastnosti. Mezi typické příklady ViewGroup patří LinearLayout, ConstrantLayout a FrameLayout.

2.2.4 Intent

Kvůli sandboxování aplikací a jejich komponent je nutné mít způsob pro komunikaci mezi procesy nebo i komponentami v rámci jedné aplikace. Pro tento účel vznikly Intenty. Jde o jednoduché objekty nesoucí jednoduchá data. Existují dva typy intentů – implicitní a explicitní.

Implicitní intenty nejmenují specifickou komponentu, která je má použít, nýbrž obecně deklarují akci určenou k provedení. Například pokud chce vývojář ukázat lokaci na mapě, může použít implicitní intent který vyvolá aplikaci, v tomto případě mapy, schopné akci specifikovanou tímto intentem vyvolat.

Explicitní intenty specifikují přímo danou aplikaci nebo komponentu která je má využít. Nejčastěji se využívají v rámci jedné aplikace, především k otevírání nových aktivit.

2.2.5 Service

Service neboli služba je řešení pro provádění operací na pozadí aplikace. Jsou velmi vhodné pro úlohy, které nevyžadují interakci s uživatelem a zároveň běží relativně delší dobu. Mají úplně jiný životní cyklus než např. aktivity a nejsou na jejich cyklech závislé. Služba běží na p Typickým použitím je stahování dat, monitorování změn v geolokaci atp.

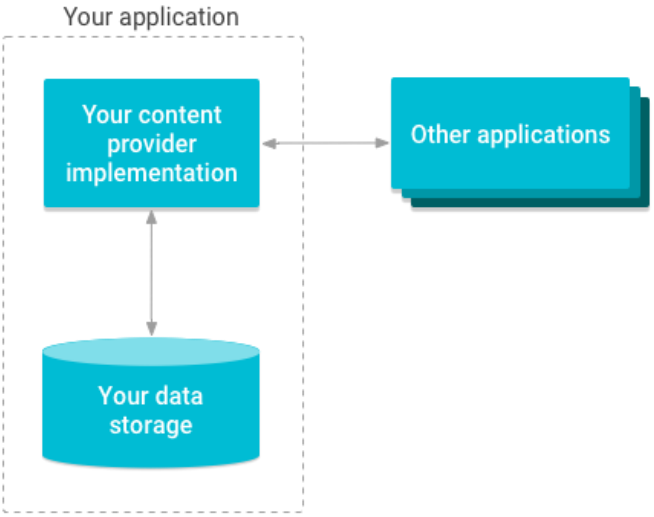
2.2.6 Broadcast Receiver

Broadcast Receiver je komponenta, kterou lze považovat za jistý systém zasílání zpráv. S jejím použitím lze monitorovat události systému jako úspěšné načtení systému a jiné implicitní intenty vyslané aplikacemi, jako například ukončení spuštěné služby pro stahování dat.

2.2.7 Content Provider

V operačním systému Android jsou data silně zabezpečena. S výjimkou externích úložišť jako SD karta, databází a souborů patřících přímo k dané aplikaci nelze snadno přistupovat k datům ostatních aplikací. K tomuto účelu existuje právě Content Provider. Jde o rozhraní

propojující data v jednom procesu s kódem běžícím v druhém procesu. Lze jej využít pro bezpečný přístup jiných aplikací k datům dané aplikace. [9]



Obr. 5. Diagram, jak Content Provider spravuje přístup k úložišti [9]

2.2.8 Context

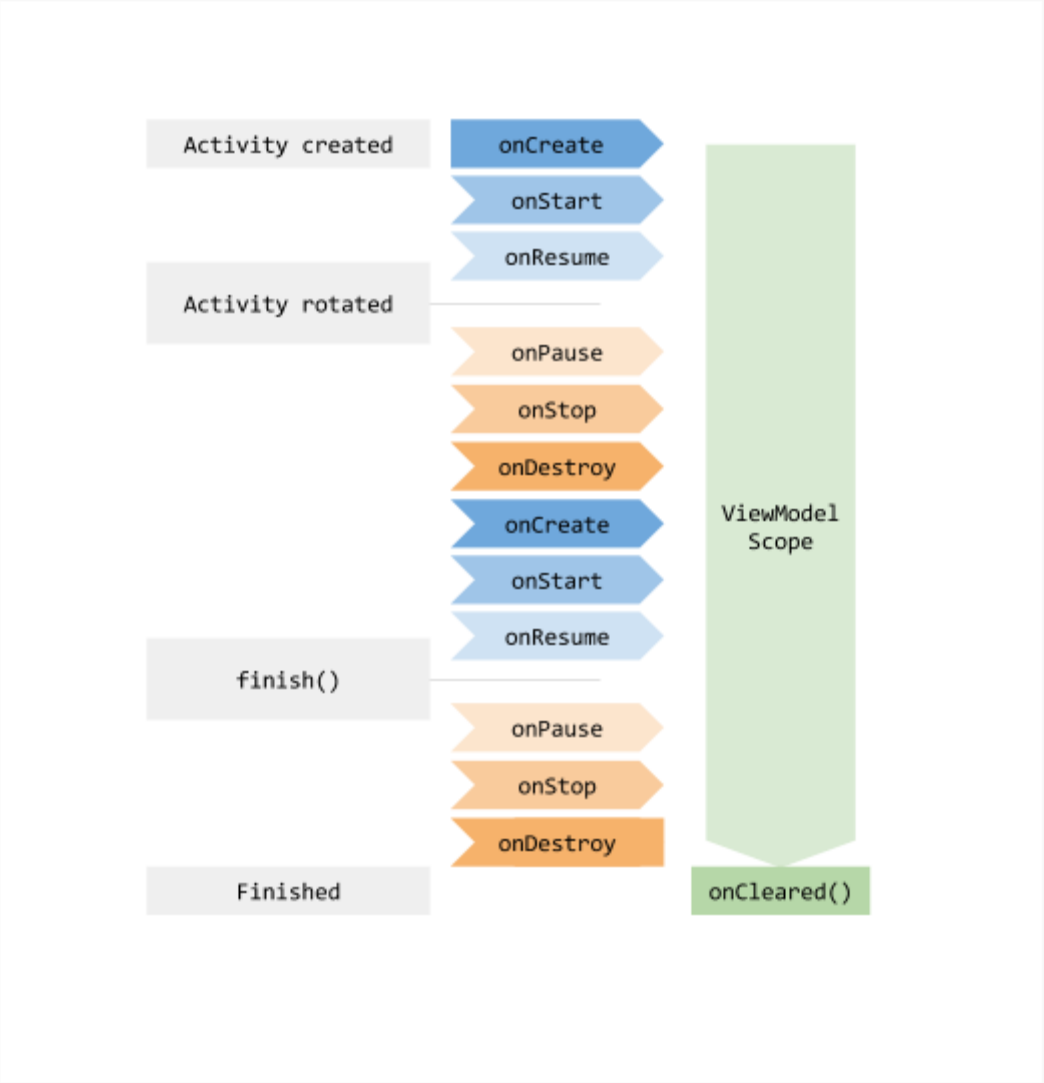
Context je velmi obsáhlá abstraktní třída obsahující metody které umožňují přístup k informacím o globálním aplikačním prostředí, různým zdrojům a podtřídám. Instance Contextu poskytují spojení se systémem. Context se využívá např. pro spouštění aktivit, služeb, zjištění velikosti obrazovky atd.

2.3 Android Jetpack Architecture components

V roce 2018 společnost Google Inc. představila Android Jetpack. Jedná se o velkou sadu knihoven a nástrojů, které mají zjednodušit vývoj převážně většiny aplikací pro Android, a také snížit závislost na využívání těchto věcí vyvinutých třetí stranou. V této kapitole se přiblížíme ty nejzásadnější architektonické komponenty z nich.

2.3.1 ViewModel

ViewModel je třída navržená pro ukládání a správu dat s ohledem na životní cykly komponent jako Aktivita nebo Fragment. ViewModel nezávisle na stavu životního cyklu tato data uchová, dokud není komponenta zničena. Tento postup výrazně zjednodušuje práci se zobrazenými daty, které by tak mohly být ztraceny například při změně konfigurace. [10]



Obr. 6. Životní cyklus ViewModelu [10]

2.3.2 DataBinding

Knihovna DataBinding je knihovna umožňující svázat komponenty uživatelského rozhraní (UI) ke zdrojům dat deklarativně namísto programově. V praxi to znamená, že již není nutné programově hledat referenci na widget jako na ukázce v Kódu 1.

```
findViewById<TextView>(R.id.ukazka).apply {
    text = viewModel.textProperty
}
```

Kód 1. Příklad svázání dat s widgetem bez knihovny DataBinding

Užitím databindingu lze velice jednoduše přiřadit data tomuto widgetu v daném souboru layout, viz ukázka Kód 2.

```
<TextView android:text="@{viewmodel.textProperty}" />
```

Kód 2. Příklad svázání dat s widgetem s knihovnou DataBinding

Tento postup sníží dotazy na uživatelské rozhraní, množství boilerplate kódu a zjednoduší jeho čitelnost. [11]

2.3.3 LiveData

LiveData je třída pozorovatelných (observable) dat. Na rozdíl od běžných pozorovatelných dat ale LiveData jsou si vědoma životních cyklů dalších komponent jako aktivity nebo fragmenty. Díky tomu LiveData aktualizuje pouze ty pozorovatele (observery), které jsou aktivní.

Výhody využití LiveData:

- **Vždy aktuální data**
  Nezávisle na tom, zdali komponenta obsahující observer přešla z neaktivity do aktivity nebo byla zničena a znovu vytvořena například změnou orientace, vždy tato komponenta získá poslední informace při její aktivaci do popředí
- **Žádné úniky paměti**
  Protože všichni pozorovatelé jsou svázáni s životním cyklem dané komponenty tak při jejich zničení jsou jejich data uklizena.
- **Pouze vhodné aktualizace UI**
  S pomocí LiveData bude UI aktualizováno pouze pokud jsou změněna data [12]

2.3.4 Navigation

Navigace v obecném smyslu slova v kontextu aplikace znamená interakce uživatele umožňující přechod do jiné části aplikace nebo zpět. Knihovna Navigation pomáhá

implementovat navigaci ať již obyčejným tlačítkem nebo komplexní logikou aplikace tak, aby byla intuitivní pro uživatele a konzistentní v rámci aplikace.

Knihovna se skládá ze tří klíčových částí:

- **Navigation graph**
  Jde o XML zdroj obsahující všechny navigační informace v dané aplikaci. To znamená všechny možné destinace aplikace i cesty k nim.
- **NavHost**
  Prázdný tzv. kontejner který zobrazuje destinace z navigačního grafu.
- **NavController**
  Jde o objekt, který spravuje navigaci v aplikaci v rámci NavHostu. NavController zprostředkovává přepínání mezi destinacemi v aplikaci

Funguje to tedy tak že, uživatel vydá pokyn k navigaci do dané destinace specifickou cestou, tento pokyn přebere NavController a nahradí stávající destinaci uvnitř NavHostu destinací novou. [13]

2.3.5 WorkManager

WorkManager je programovací rozhraní, které usnadňuje plánování odložitelných, asynchronních úloh, které mohou běžet i po zavření aplikace nebo restartu zařízení.

Mezi hlavní výhody této knihovny patří:

- **Robustní flexibilní plánování**
  Úloha může být spuštěna, když je zařízení pouze na wifi, opakovaně či pouze jednou a také znovu pokud spuštění selhalo. WorkManager také zajistí spuštění úlohy i po restartu zařízení.
- **Integrovaná interoperabilita vláken**
  WorkManager lze bezproblémově integrovat s knihovnami RxJava, Coroutines nebo i vlastním asynchronním API [14]

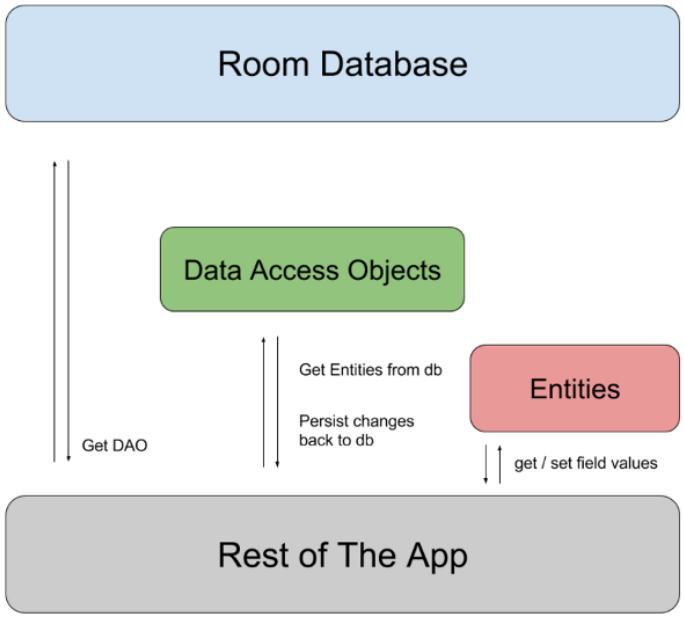
2.3.6 Room

Room je knihovna poskytující abstraktní vrstvu nad databází SQLite. Umožňuje robustnější přístup do databáze při zachování všech možností SQLite. Room umožňuje vytvoření tzv. cache neboli vyrovnávací paměti, do které jsou uloženy informace, ke kterým má pak uživatel přístup i v případě, že nemá v danou chvíli připojení k internetu. Mezi další výhody

patří validace SQL dotazu při kompilaci a zjednodušení přístupu k databází redukcí tzv. boilerplate kódu. [15]

Room má tyto tři hlavní části:

- **Database**
  Obsahuje holder databáze a slouží jako hlavní přístupový bod k perzistentním, relačním datům.
- **Entity**
  Třída představující tabulku databáze.
- **DAO (Data Access Objects)**
  Mapování SQL dotazů k funkcím. Obsahuje metody využívané k přístupu do databáze. [15]



Obr. 7. Vztahy mezi komponentami databáze Room [15]

3 VIRTUÁLNÍ A ROZŠÍŘENÁ REALITA

Koncept jiné reality není žádnou novinkou, naopak je již dlouho stálicí v představivosti autorů sci-fi literatury a dalších médií. Ale až rapidní progres v oblastech imerzivní zábavy, miniaturizace hardwaru a zvyšování výpočetního výkonu dovoluje vývoj zařízení a softwaru, který umožňuje uživateli zažít nové typy realit. S dalším vývojem těchto technologií došlo k jejich rozdělení na virtuální realitu, rozšířenou realitu a smíšenou realitu.

3.1 Virtuální realita

Virtuální realita neboli VR, popisuje technologii, která vytváří počítačem generované prostředí a zážitky, se kterými mohou lidé interagovat tak, jak by to dělali v reálném světě. K tomuto účelu softwarové a hardwarové prostředky musí spolupracovat tak, aby perfektní replikací způsobu rozpoznání okolí jedince dokázaly toto okolí nahradit. Jinými slovy tato technologie má za cíl úplné ponoření uživatele do virtuálního světa.

Uživatel této technologie by měl přirozeně a intuitivně ovládat všechny dostupné prvky k tomuto určeny a virtuální realita by měla na tyto akce patřičně reagovat. VR se snaží zaměřením na smysly jako zrak, dotek, zvuk, rovnováha aj. zmenšovat povědomí o umělosti světa. Například zrak lze ošálit dostatečně jemným a širokým (minimálně 180°) obrazem.

Na obrázku níže je pro příklad sestava pro virtuální realitu sestávající se z polohových senzorů umístěných na stativech, dvou ovladačů a samotného náhlavního displeje virtuální reality.

Obr. 8. Sestava pro virtuální realitu [16]

3.2 Rozšířená realita

Rozšířená realita, známá také pod názvem augmentovaná (AR) je technologie, která umožňuje zobrazit výpočetní technikou generované virtuální informace jako další vrstvu přes přímé nebo nepřímé zobrazení skutečné reality v reálném čase. Příklad přímého zobrazení je při využití například speciálních brýlí nebo HUD displeje, kdežto nepřímě zobrazení využívá kamery jistého zařízení a zabíraný obraz se promítá, spolu s virtuální informací, na obrazovku. Příklad nepřímého zobrazení AR navigace vozidla je na obrázku níže.

Obr. 9. Rozšířená realita [17]

3.3 Smíšená realita

Smíšená realita neboli Mixed reality (MR) je nový typ reality, který spojuje dohromady virtuální a rozšířenou realitu, tedy prvky z reálného a prvky z digitálního světa. Ve smíšené realitě uživatel interaguje a manipuluje s fyzickými i virtuálními objekty a prostředími. Hranice mezi rozšířenou a smíšenou realitou je mnohdy velmi úzká a mnohé zdroje tyto pojmy zaměňují. [18]

3.4 Vývojové nástroje pro tvorbu rozšířené reality pro mobilní telefony

Nástrojů pro vývoj aplikací s rozšířenou realitou existuje více. Rozlišuje je především platforma, pro kterou je daný nastroj určený a cena za jejich použití v komerční aplikaci. Z těchto nástrojů se ty nejvýznamnější zpravidla dají rozdělit na nástroje určené pro jednu platformu a nástroje multiplatformní. Nástroje pro specifickou platformu jsou zdarma, multiplatformní spadají pod placenou licenci za komerční využití.

3.4.1 ARCore

ARCore je platforma vyvinutá společností Google pro tvorbu aplikací s využitím rozšířené reality. Využívá tři základní schopnosti pro integraci virtuálních informací s reálným světem snímaným přes kameru zařízení:

- **Snímání pohybu**

Umožňuje sledovat relativní polohu používaného zařízení v reálném světě díky procesu zvaném souběžná odometrie a mapování. ARCore detekuje vizuálně odlišné body snímané kamerou a používá je, v kombinaci s informacemi ze senzorů zařízení, k výpočtu polohy

- **Rozpoznání okolního prostředí**
  ARCore neustále hledá shluky významných bodů ve snímaném prostředí. Shluk takových bodů v prostoru může například detekovat jako plochu, jejíž pozici a rozměry vývojáři nabídne pro umístění zobrazené informace.
- **Odhad světelných podmínek okolí**
  ARCore umí detekovat informace o světelných podmínkách okolí poskytnout je vývojáři aplikace. Ten pak může tyto informace využít po úpravu virtuálních objektů před jejich zobrazením tak, aby působily více realisticky.

ARCore je navržený pouze pro operační systém Android od verze 7.0. Jeho hlavní výhodou je jeho volná licence a jednoduchost použití. [19]

3.4.2 ARKit

ARKit je sada nástrojů od společnosti Apple určených pro vývoj aplikací pro operační systém iOS a v mnohém se podobá ARCore. Mezi hlavní funkcionality patří:

- Detekce okolního prostředí
- Odhady ambientního osvětlení
- Trakce pohybu
- Trakce obličeje
- Okluze lidí

ARKit je dostupný pouze pro zařízení s iOS ve verzi 11.0 a novější. [20] [21]

3.4.3 Vuforia

Vuforia je nejpopulárnější volbou pro vývoj komerčních aplikací s rozšířenou realitou. Je tomu tak především proto že je nejstarší a také, na rozdíl od nativních řešení jako ARCore a ARKit podporuje zpětně také starší verze operačního systému. Hlavní výhody Vuforie jsou schopnost rozpoznat a sledovat jakékoli objekty, které byly předem naskenovány, a rozpoznání textu a prostředí. [20]

3.4.4 Wikitude

Stejně jako Vuforia, jde o multiplatformní nástroj pro vývoj AR aplikací se zpětnou kompatibilitou se staršími systémy. Jeho hlavní výhodou je využití algoritmů typu SLAM (souběžná lokalizace a mapování) které umožňuje zobrazit a použít objekty v AR bez předem naskenovaných značek v prostoru a podpora OS Windows. [21]

3.4.5 Spark AR a Lens Studio

Spark AR a Lens studio jsou řešení Facebooku a Snapchatu pro velmi jednoduché aplikace, které ani nevyžadují instalaci, protože informace se zobrazí ve stávající hlavní aplikaci těchto společností. Spark AR a Lens studio jsou nabízeny zdarma.

II. PRAKTICKÁ ČÁST

4 POŽADAVKY NA APLIKACI

Účelem praktické části této diplomové práce je vytvořit nativní Android aplikaci zobrazující zajímavosti a informace o identifikovaných zájmových bodech. Tyto informace jsou následně zobrazeny v prostoru rozšířené reality. Tato kapitola se věnuje popisu jednotlivých požadavků na takovou aplikaci.

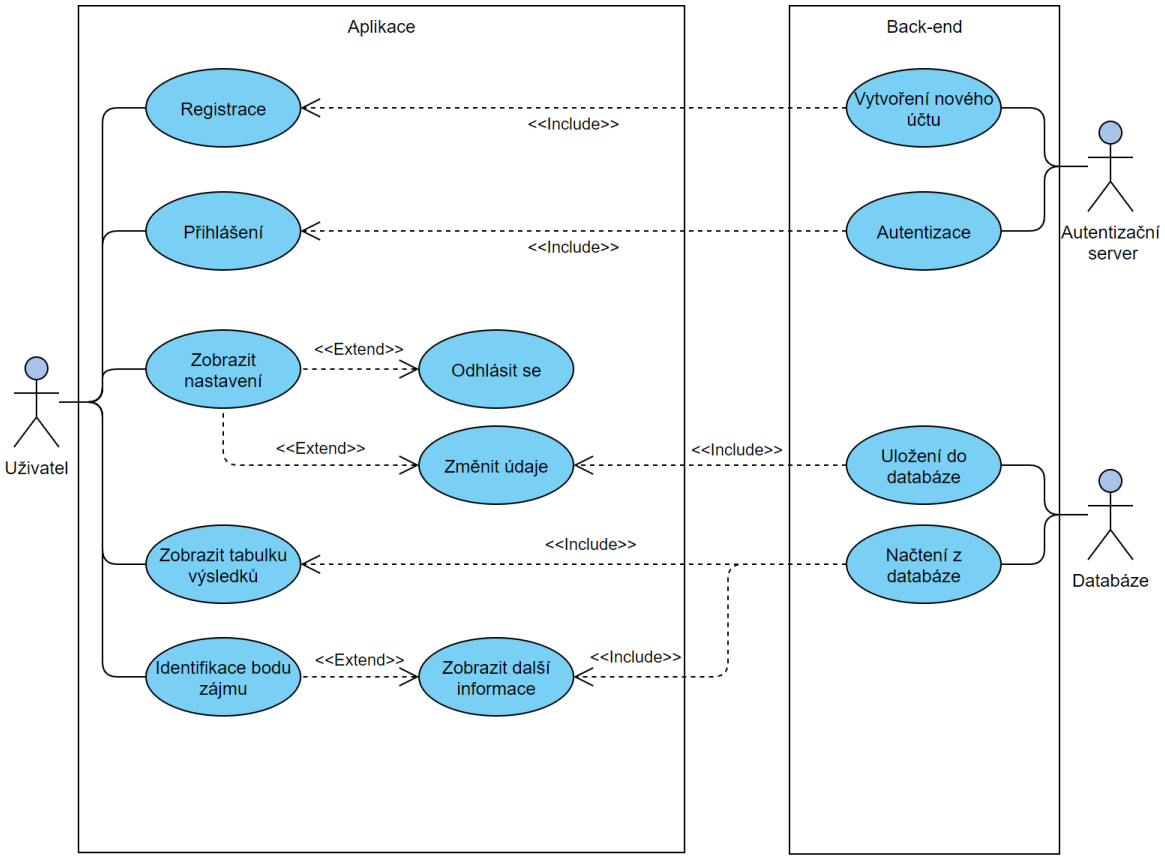
4.1 Funkční požadavky

Funkční požadavky jsou takové požadavky, které popisují jednotlivé funkce vyvíjeného programu. Udávají samotný účel aplikace a její možnosti. Funkční požadavky byly zvoleny následovně:

- Možnost registrace uživatele
- Autentizace identity uživatele
- Snadná identifikace zájmového bodu
- Zobrazení informací o zájmovém bodě v rámci rozšířené reality
- Volitelné zobrazení dodatečných informací z internetu
- Tabulka výsledků porovnávající aktivitu uživatelů aplikace
- Možnost úpravy uživatelských dat
- Funkce odhlášení z aplikace

4.1.1 Use Case Diagram

Dle funkčních požadavků jsem vytvořil *Use Case Diagram*. Diagram je zobrazen na Obr. 10 a vystupuje v něm několik aktérů, kteří se dělí na aktivní a pasivní. Aktivní aktérem je uživatel, který využívá prakticky všechny možnosti samotné mobilní aplikace, kdežto pasivní aktéři využívají v back-endové části tohoto projektu, autentizaci a práci s databází.



Obr. 10. Use case diagram

4.2 Nefunkční požadavky

Zatímco funkční požadavky popisují, co aplikace dělá, nefunkční požadavky popisují aplikaci z vnějšího pohledu, tedy její vlastnosti jako např. dostupnost, bezpečnost atd. Popsání a dodržení těchto požadavků je důležitou součástí úspěchu a popularity každé vyvíjené aplikace. Ačkoli tyto požadavky nebyly dány zadáním této práce, řada z nich plyne ať již z funkčních požadavků nebo z dobrých zvyklostí vývoje každého softwaru. Nefunkční požadavky byly stanoveny následovně:

- **Zabezpečení dat**

  Pro přístup do aplikace je nutné k ní vlastnit uživatelský účet a být s jeho pomocí přihlášen. Toto přihlášení probíhá přes vzdálenou, šifrovanou autentizaci. Veškerá uživatelská data, s výjimkou vyrovnávací paměti aplikace v telefonu, jsou uložena bezpečně na vzdáleném serveru a každý uživatel má přístup pouze k jeho vlastní datům. Cizí aplikace k těmto datům taktéž nemají přístup.

- **Lokalizace**

  V tuto chvíli je aplikace dostupná pouze v českém jazyce, nicméně nic nebrání jejímu rozšíření o další jazyky

- **Snadnost použití**

  Používání aplikace nesmí být pro uživatele frustrující záležitostí. Proto aplikace využívá prvky, se kterými je prakticky každý uživatel operačního systému Android již seznámen a dovede tak aplikaci ovládat okamžitě po její instalaci.

- **Efektivní využití paměti**

  Zvláště u mobilních zařízení, ve kterých je dostupná paměť omezená a sdílená s dalšími aplikacemi je nutné efektivně navrhnout její využití. Vyvinutá aplikace není nikterak náročná na paměť. Díky využití vzdálené databáze pro ukládání dat a použití některých prvků jako např. *RecyclerView* bylo využití paměti sníženo na nutné minimum.

- **Optimalizace využití baterie**

  Software určený pro použití na zařízeních s omezeným zdrojem energie by nikdy neměl spotřebovat více baterie, než je nezbytně nutné. V tomto ohledu není vyvinutá aplikace nijak náročná, protože nikdy neprovádí žádné instrukce na pozadí, když není zrovna aktivně využívána.

- **Síťové požadavky**

  Pro použití aplikace k jejímu hlavnímu účelu je vyžadováno připojení k internetu. Nezávisí na tom, jakou technologií je zařízení připojeno (Wi-Fi, 2G, 3G, LTE, 5G aj), aplikace funguje se všemi a poradí si také v případě přechodu mezi nimi.

- **Spolehlivost**

  Vyvinutá aplikace by měla být spolehlivá v celém rozsahu jejího využití. V ideálním případě by počet chyb měl směřovat k nule. Testováním aplikace a následnými opravami kódu se k tomuto ideálu vývojář snaží přiblížit.

- **Podpora verzí OS**

  Aby aplikaci mohlo využívat co nejvíce uživatelů, je vhodné zajistit podporu pro co nejvíce verzí operačního systému. U vyvíjené aplikace toto bylo mírně omezeno využitými technologiemi, a tak lze aplikaci v tuto chvíli použít na přibližně 83,4 % zařízení s Androidem (viz. kapitola 6).

5 GRAFICKÉ ROZHRANÍ

Pro každou aplikaci je nutné navrhnout její rozhraní co nejvíce uživatelsky přívětivě z funkční i grafické stránky. Vývojář by se měl držet zvyklostmi a pravidly dané platformy tak, aby aplikaci dokázal běžný uživatel bez problému ovládat.

5.1 Popis jednotlivých obrazovek

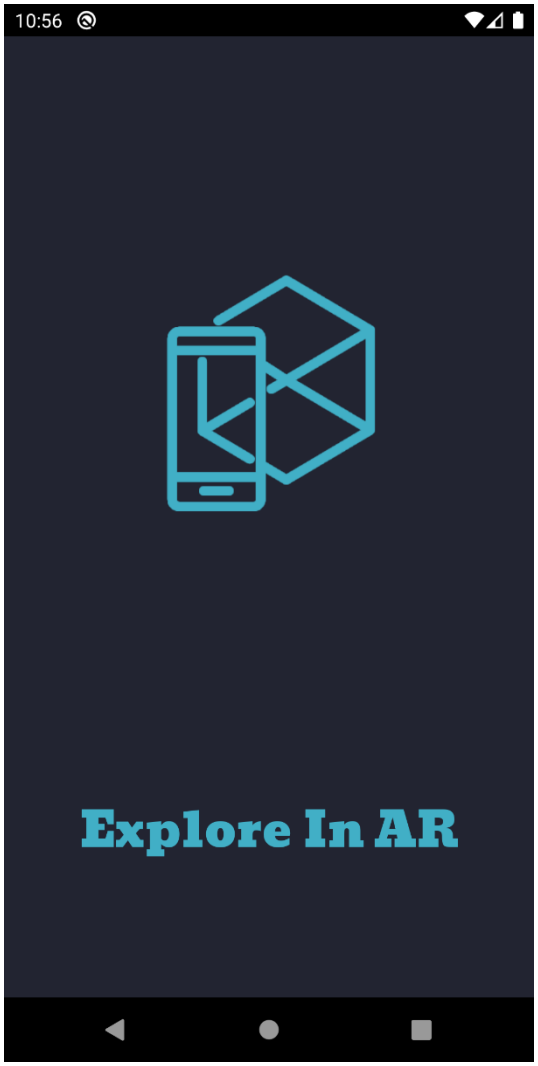
Každá obrazovka aplikace se skládá z prvků a widgetů, jejichž konfigurací ovlivňujeme celkový vzhled obrazovky. Tato kapitola se věnuje popisu jednotlivých obrazovek vyvinuté aplikace.

5.1.1 Vstupní obrazovka

Při spuštění aplikace je uživatel přivítán aktivitou se vstupní obrazovkou, na níž se nachází efektní animace, viz Obr. 11. Tato obrazovka se však zobrazí pouze tomu uživateli, který není do aplikace přihlášen. Na pozadí tato aktivita totiž zjišťuje, zda je uživatel přihlášen či ne. Je to tak provedeno proto, aby animace zbytečně neprodlužovala každé otevírání jejímu opakovanému uživateli.

Tato aktivita se skládá pouze ze dvou jednoduchých prvků:

- **ImageView**
  Jedná se o widget zobrazující obrázek. Tento obrázek je možné vybrat programově, i přímo v XML souboru rozvržení. Zde zobrazuje logo aplikace.
- **TextView**
  Jde o jednoduchý widget, jehož funkcí je zobrazit a případně stylizovat na obrazovce text. V této aktivitě vypisuje název vyvinuté aplikace.



Obr. 11. Vstupní obrazovka

5.1.2 Přihlašovací obrazovka

Po uběhnutí animace ve vstupní obrazovce se uživateli automaticky ukáže přihlašovací obrazovka zobrazená na Obr. 12 (A). Zde má uživatel několik možností. Může se přihlásit ke svému, již existujícímu účtu s pomocí emailové adresy a hesla, pokračovat dále na registrační obrazovku, nebo využít přihlášení pomocí Google účtu, který každý uživatel Androidu musí mít. Aplikace mu také nabídne výběr konkrétního účtu, případně přidání dalšího, viz Obr. 12 (B). Tato volba je pro uživatele jednoznačně nejpřívětivější, protože je nejen nejrychlejší, provede registraci zároveň s přihlášením, ale také není nutné si pamatovat přihlašovací údaje.

Tato obrazovka se skládá z několika prvků, jimiž jsou:

- **ImageView**

  Tento widget slouží opět pro zobrazení loga aplikace.
- **TextInputLayout**

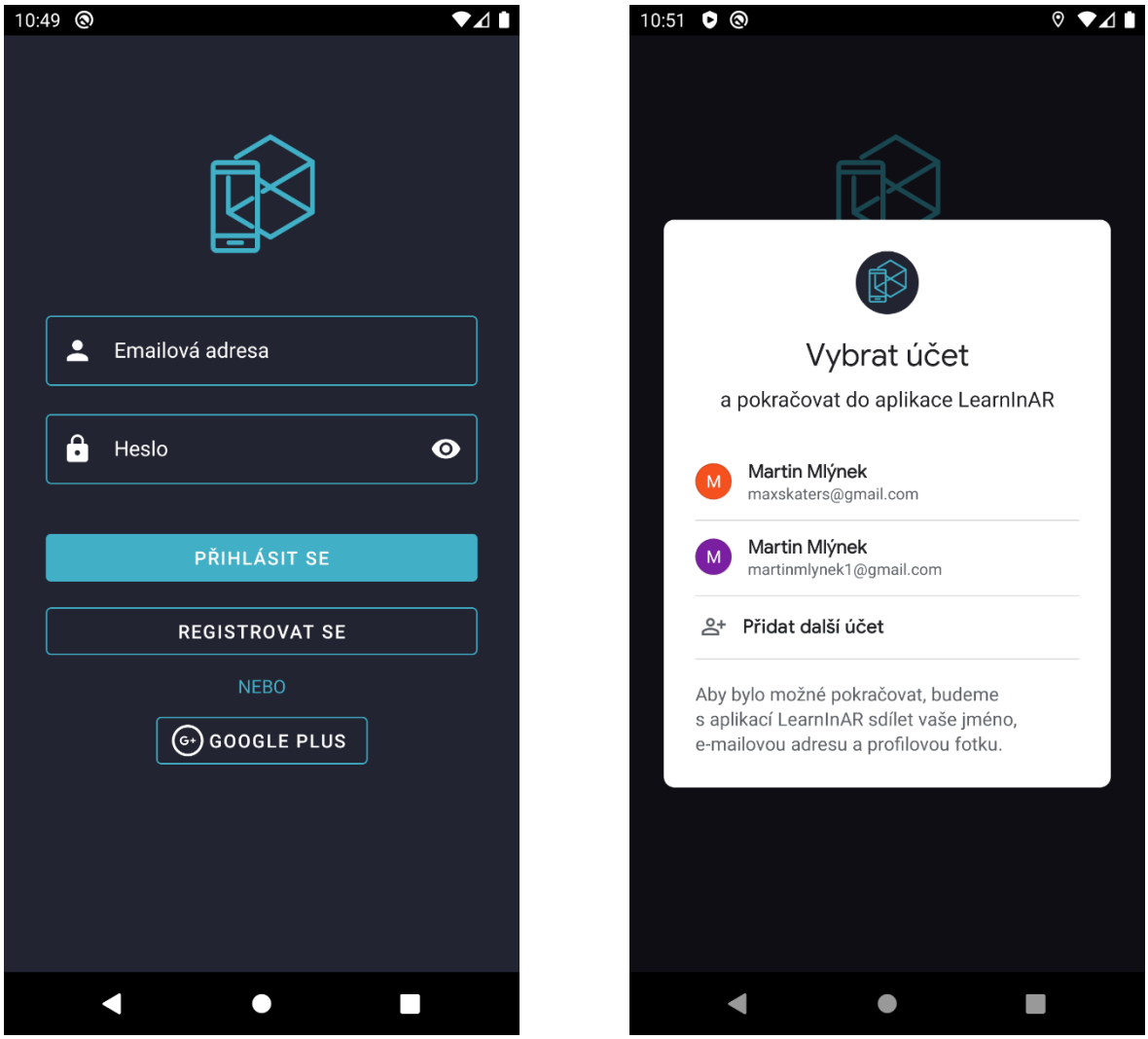
  Tento layout v kombinaci s widgetem **TextInputEditText** tvoří textové pole, do nějž uživatel vepisuje své údaje. Oproti obyčejnému EditText widgetu tento nabízí široké možnosti nastavení jeho vzhledu. Jedná se o widget z oficiální knihovny pro Material Design.
- **Button**

  Button je jednoduchý widget představující tlačítko. Na této obrazovce se nachází tři. Pro přihlášení, pro registraci a pro využití Google účtu.
- **TextView**

  Zobrazuje text „nebo“.
- **ProgressBar**

  Tento, na Obr. 12. neviditelný widget se zobrazí pouze ve chvíli, kdy probíhá autentizace uživatele. Jde o Animaci představující načítání.

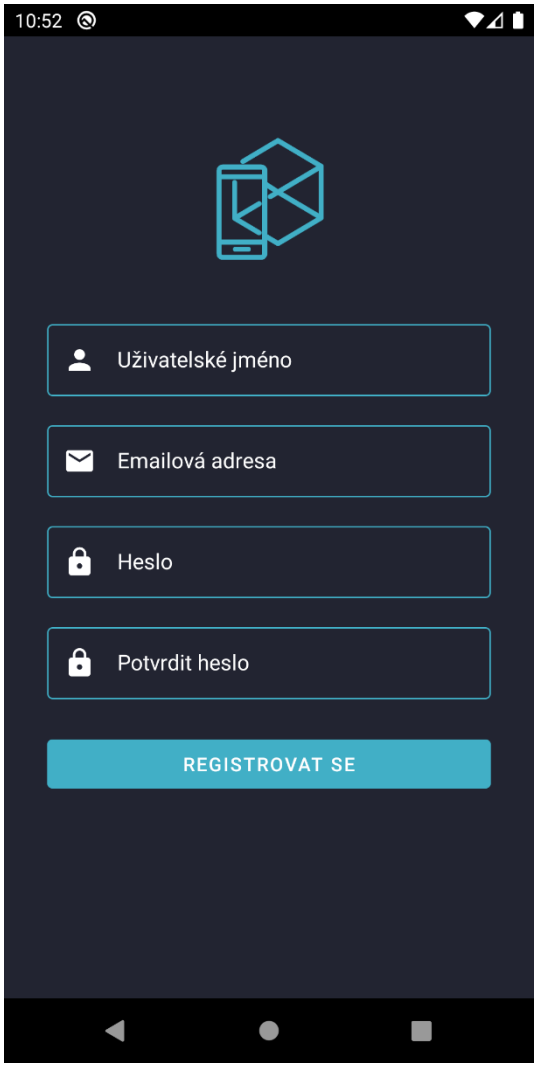
A) B)



Obr. 12. Přihlašovací aktivita (A) a první spuštění a přihlášení pomocí Google (B)

5.1.3 Registrační obrazovka

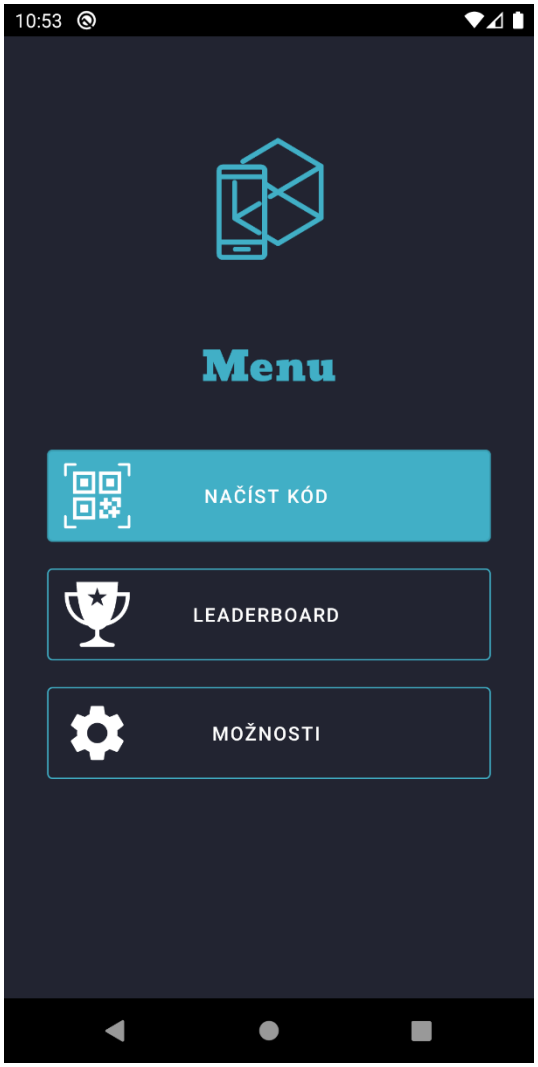
V případě, že uživatel na předchozí obrazovce vybere možnost „Registrovat se“, objeví se mu právě tato obrazovka. Je v mnohém podobná obrazovce předchozí, pouze vyžaduje navíc uživatelské jméno a potvrzení hesla. Tyto vstupy jsou, stejně jako v předchozí obrazovce, kontrolovány pro jejich validitu. Tedy pole nesmí být prázdné, emailová adresa musí být ve správném tvaru, heslo musí mít minimálně šest znaků apod.



Obr. 13. Registrační obrazovka

5.1.4 Hlavní menu

Po úspěšné autentizaci, či otevření aplikace v níž se uživatel již dříve přihlásil, ho uvítá obrazovka hlavního menu, viz Obr. 14. Na této obrazovce se nachází *ImageView* opět s logem a tři tlačítka *Button*. První tlačítko vede k identifikaci zájmového bodu, další na obrazovku s tabulkou výsledků a poslední do nastavení.



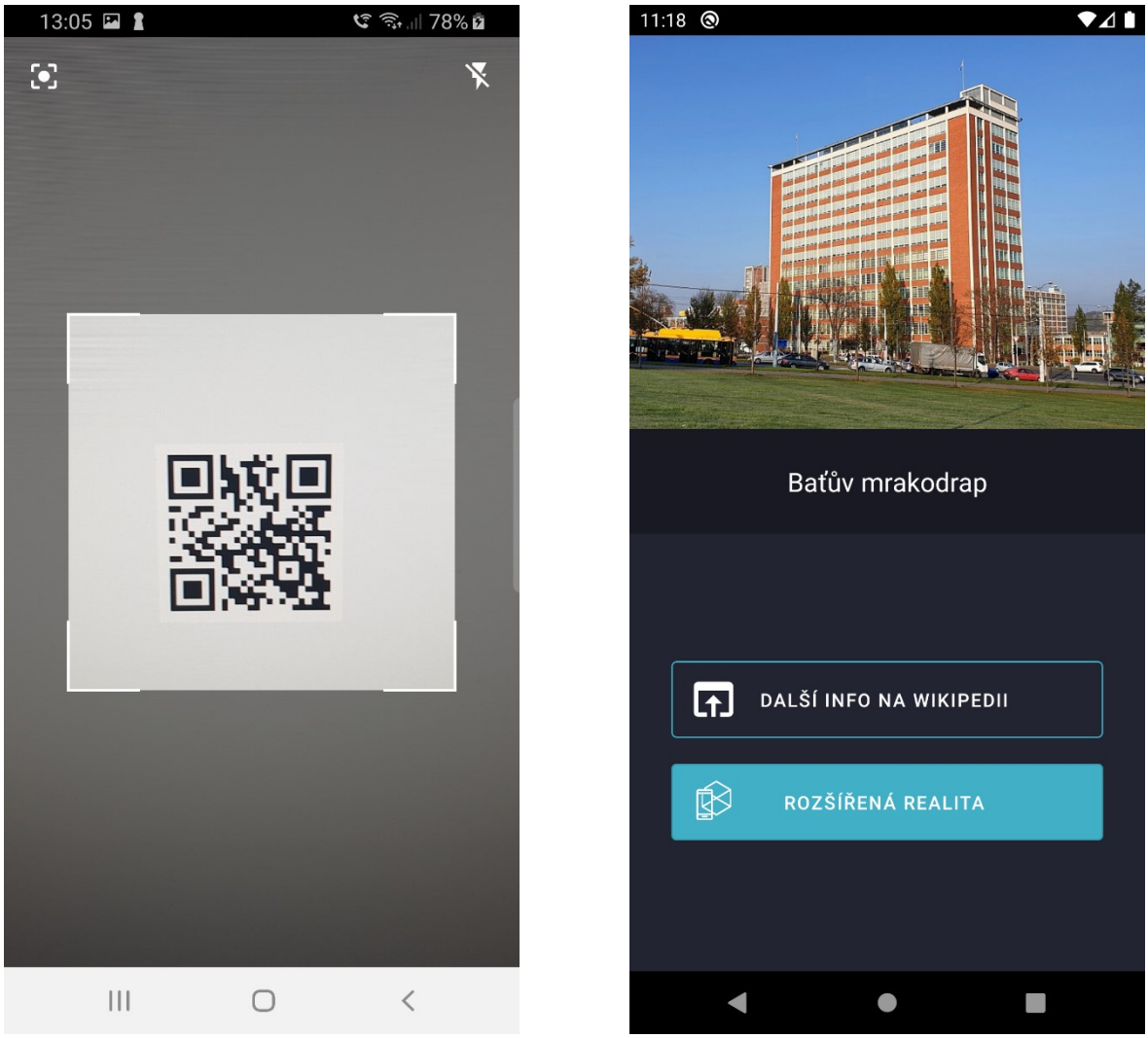
Obr. 14. Hlavní menu

5.1.5 Obrazovka načtení kódu

Při výběru první možnosti v hlavním menu, tedy „Načíst kód“, se v závislosti na tom, zda uživatel již někdy tuto možnost vybral nebo ne, mohou stát dvě věci. Pokud je toto poprvé, aplikace se ho zeptá, zdali jí povolí přístup ke kameře nebo nepovolí. Pokud jí přístup odmítne, dostane se mu upozornění že tento přístup ke pro tuto akci nutný a může to zkusit znovu. Pokud přístup povolí, otevře se obrazovka zachycená na Obr. 15 (A).

Tato obrazovka obsahuje hledáček kamery a pár dalších prvků, které mají za účel usnadnit uživateli čtení QR kódu jako přisvícení a zámek automatického ostření.

A) B)



Obr. 15. Hledáček skeneru QR kódu (A) a obrazovka po jeho načtení (B)

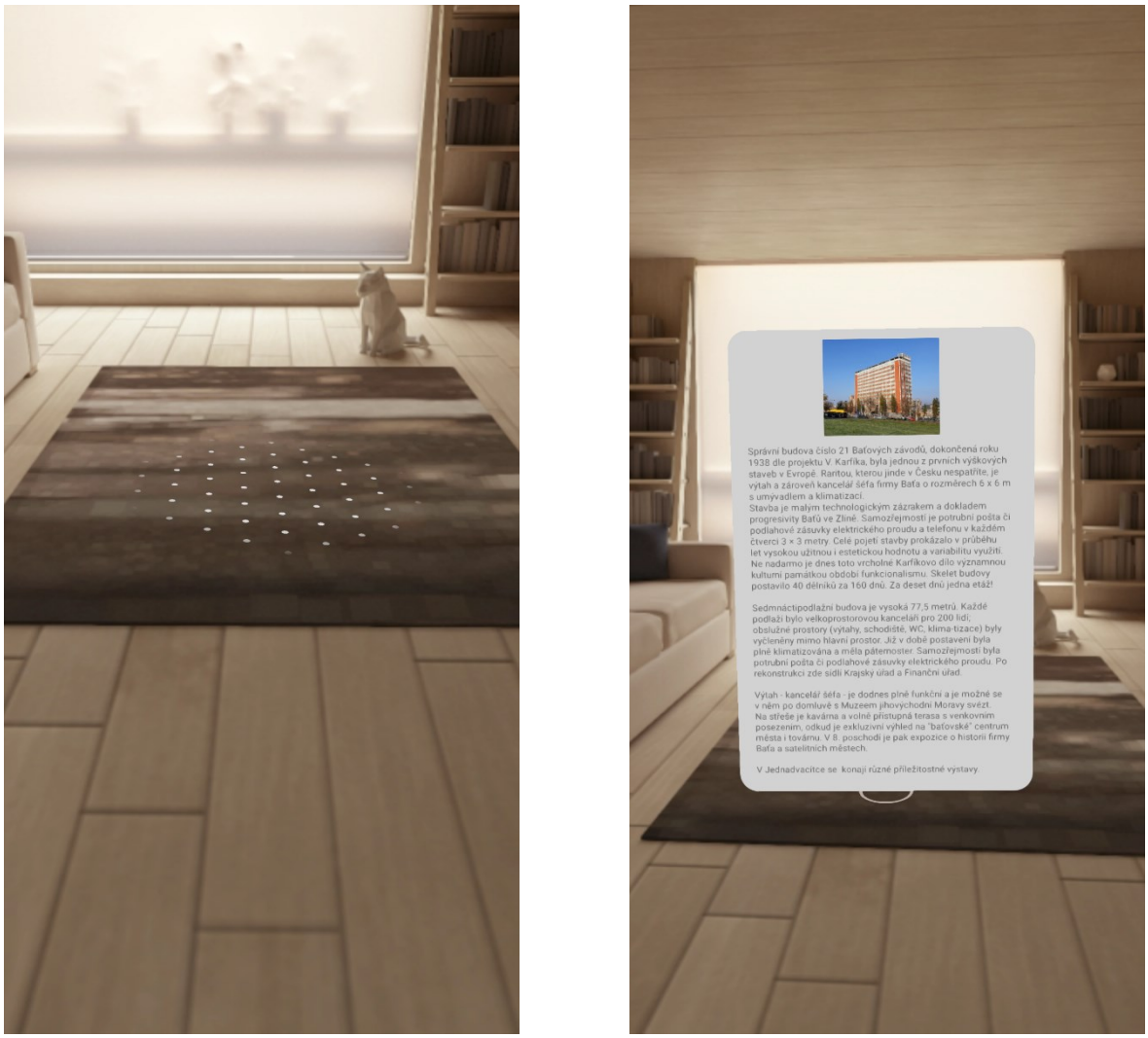
Po úspěšném načtení QR kódu, se uživatel ocitne na obrazovce zobrazené na Obr. 15 (B). Tato jednoduchá obrazovka poskytuje fotku objektu, jehož QR kód byl naskenován, jeho název a dvě tlačítka *Button*, z nichž první otevře s pomocí implicitního Intentu internetový prohlížeč s článkem o daném zájmovém objektu na Wikipedii k volitelnému dalšímu čtení. Druhé tlačítko vede do rozšířené reality.

5.1.6 Obrazovka rozšířené reality

Na této obrazovce se podobně jako při čtení QR kódu v předchozí obrazovce ukáže hledáček kamery. Okamžitě po jejím načtení začne program hledat plochy vhodné pro umístění

informací o zájmovém objektu. Tyto plochy se po jejich nalezení zobrazí jako bílé tečky, jak je vidět na Obr. 16 (A). Poté již zbývá uživateli jen klepnout na obrazovku a tyto informace se zobrazí na virtuálním objektu umístěném v prostoru, viz Obr. 16 (B). Tento objekt lze přetažením za jeho spodní část posunout po ploše jinam, či ho lze postavit na jiném místě opětovným klepnutím, za předpokladu, že na tom místě byla nalezena plocha.

A) B)



Obr. 16. Hledáček rozšířené reality (A) a hledáček po umístění objektu do prostoru (B)

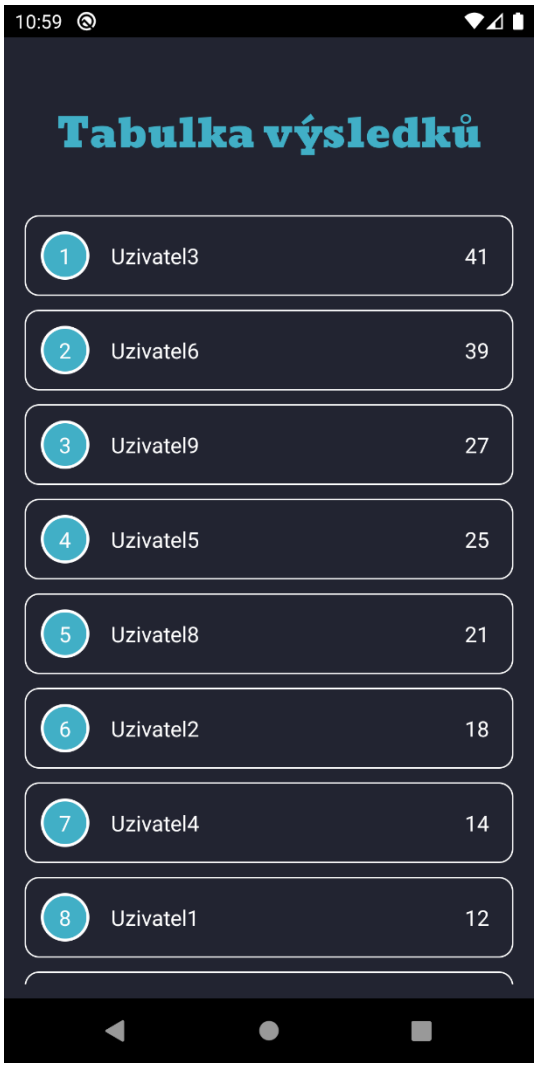
5.1.7 Obrazovka tabulky výsledků

Tato obrazovka dostupná z hlavního menu obsahuje tabulku, kde si každý uživatel aplikace může prohlédnout své skóre a porovnat si ho s ostatními. Skóre udává aktivitu, respektive

počet nalezených objektů zájmu. Za každý jeden objekt získá uživatel bod. Tabulka se také sama aktualizuje v reálném čase.

Obrazovka využívá tyto widgety:

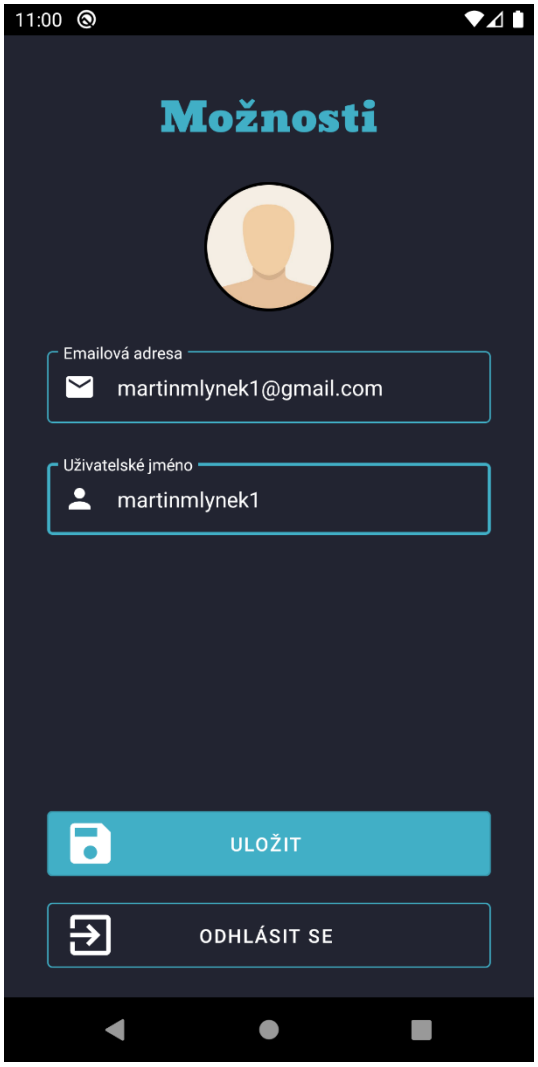
- **TextView**
  Zobrazuje nadpis obrazovky.
- **RecyclerView**
  Jedná se o speciální widget, který používá ty samé prvky rozložení tak, aby efektivně využíval operační paměť zařízení. Více se mu budeme věnovat v kapitole implementace.



Obr. 17. Tabulka výsledků

5.1.8 Obrazovka nastavení

Tato obrazovka je taktéž přístupná z hlavního menu. S pomocí widgetů na vkládání textu a tlačítka na uložení si může uživatel změnit své uživatelské jméno. Dále je zde také možnost se odhlásit z aplikace po jejímž zvolení je uživatel přesměrován zpět na začátek, tedy na vstupní obrazovku.



Obr. 18. Obrazovka nastavení

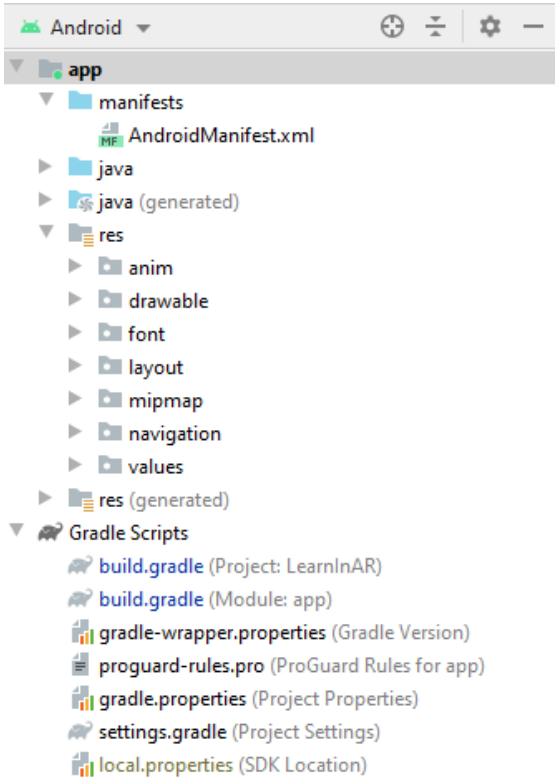
6 REALIZACE

Z důvodu požadavku na využití rozšířené reality byl vybrán nativní postup vývoje aplikace. Aplikace využívá knihovnu ARCore, jejíž podpora začíná na verzi 7.0 Nougat operačního systému Android. Dle tabulky Tab. 1. to znamená, že aplikace je spustitelná na 83,4 % všech zařízení s Androidem. Toto procento však je jistě nižší, protože ne všechny telefony s minimálně touto verzí jmenovanou knihovnu podporují. Dalo by se však téměř říci, že všechny telefony, které v době jejich vydání měly tuto nebo vyšší verzi jsou již podporovány.

Pro vývoj nativní aplikace se nejčastěji využívají vývojová prostředí Android Studio, IntelliJ IDEA a Visual Studio Code. Pro tento projekt bylo zvoleno Android Studio.

6.1 Android Studio

Android Studio je integrované vývojové prostředí určené pro vývoj nativních aplikací pro OS Android.



Obr. 19. Struktura aplikace v Android Studiu

Na Obr. 19 je vyobrazena struktura každé aplikace. Tato struktura se skládá ze čtyř hlavních částí:

- manifests
- java
- res
- gradle

Ve složce *manifests* se nachází jediný soubor – AndroidManifest.xml. Tento soubor obsahuje spoustu důležitých informací o aplikaci jako:

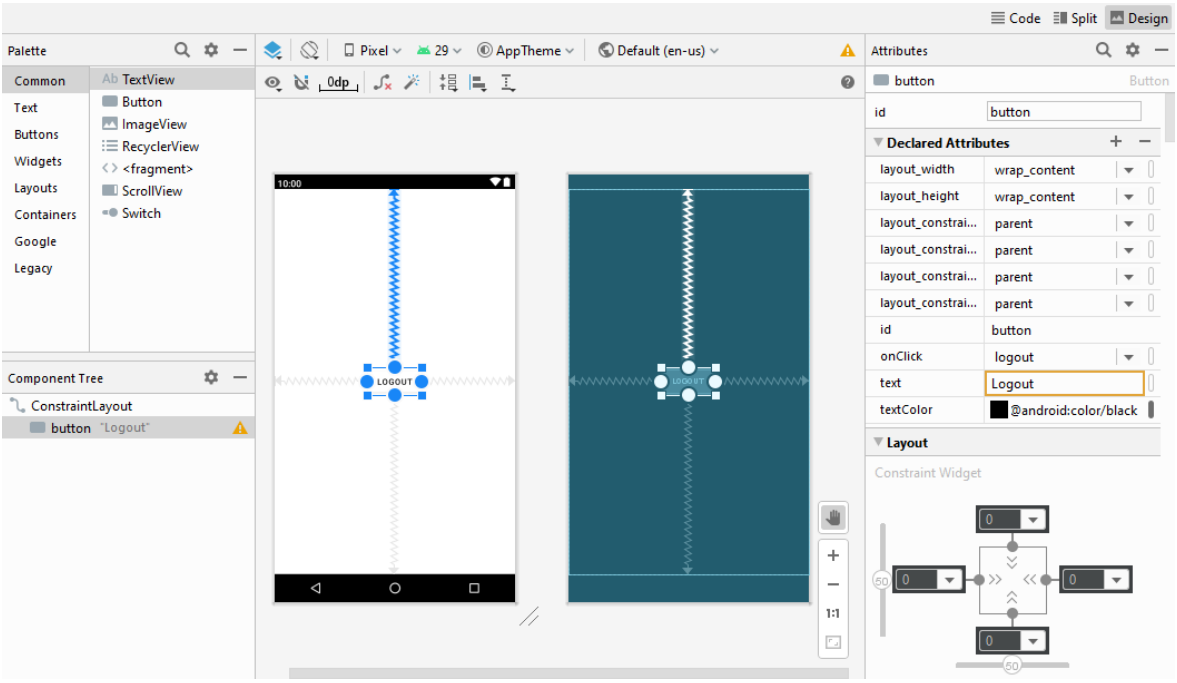
- vyžadované povolení
- jednotlivé XML elementy aktivit, služeb, BroadcastReceiverů a ContentProviderů aplikace
- název a id aplikace
- filtry intentů

Ve složce *java* jsou zdrojové soubory obsahující logiku celé aplikace a jednotkové testy. Tyto zdrojové kódy mohou být napsány v jazycích Java nebo Kotlin.

Složka *res* obsahuje zdroje využívané aplikací jako obrázky, animace, a rozvržení obrazovek.

Gradle je sestavovací program. Vezme Android projekt a sestaví ho do APK (Android Package Kit) souboru. Soubory Gradle se používají pro konfiguraci projektu a jeho sestavení. S pomocí Gradle lze také použít knihovny třetích stran.

Nedílnou součástí Android Studia je i editor layoutu neboli rozvržení aplikace. V editoru lze toto rozvržení psát ve formě XML kódu, umísťováním prvků do náhledu nebo kombinací obojího. Lze zde také vyzkoušet, jak se bude rozvržení chovat na jiných verzích API nebo konfiguracích obrazovek.

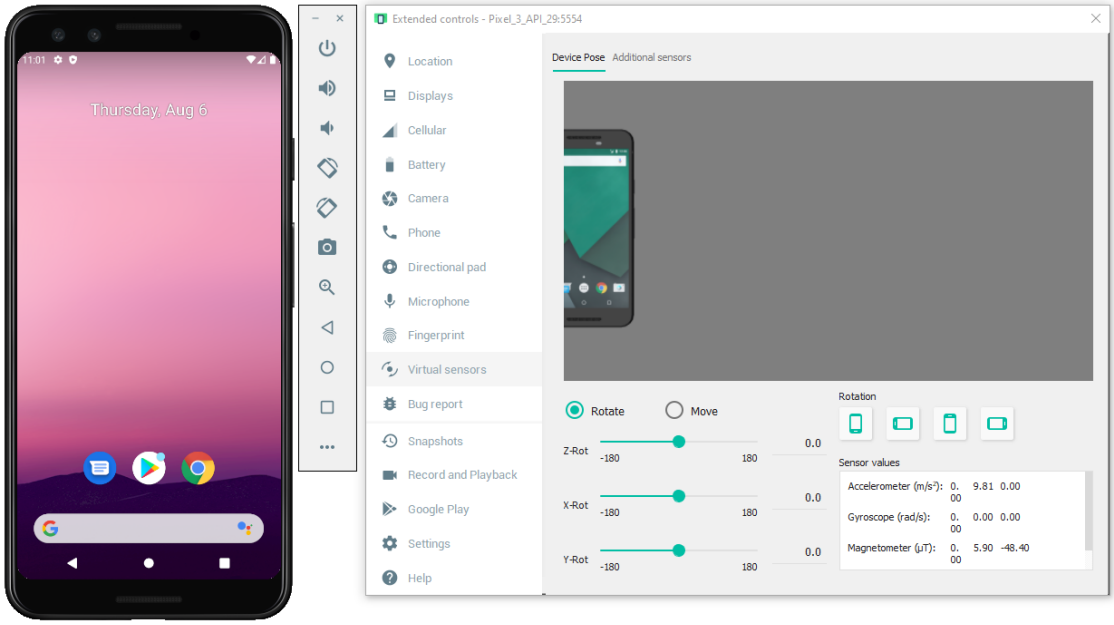


Obr. 20. Android editor rozvržení

6.1.1 Emulátor

Při vývoji Android aplikace je vhodné neustále zkoušet, zda se implementovaná funkcionalita chová dle očekávání či ne. K tomuto účelu je vhodné využít Android Emulátor. Jedná se o virtuální zařízení s operačním systémem Android, které se snaží přiblížit co nejvíce zařízení reálnému jeho simulací. Tato simulace se neomezuje pouze na mobilní telefony, ale lze také simulovat další zařízení s operačním systémem Android jako televize, autorádio, chytré hodinky a tablety. V konfiguraci emulovaného zařízení lze vybrat např. verzi systému, rozměry a rozlišení obrazovky, dostupné senzory aj.

Samotné zařízení lze ovlivňovat také změnami stav zařízení jako přístup k síti nebo změnou hodnot snímaných simulovanými senzory jako jsou geolokační, akcelerometr, světelný senzor aj. Lze tak třeba vyzkoušet, jak se aplikace zachová, pokud zařízení ztratí připojení k internetu.



Obr. 21. Android Emulátor

6.1.2 Programovací jazyky

K vývoji aplikace ve vývojovém prostředí Android Studio se používají tyto programovací jazyky:

6.1.2.1 Java

Java je objektově orientovaný programovací jazyk a výpočetní platforma vyvinutá společností Sun Microsystems v roce 1995. [23] Java byla až do roku 2017 jediným hlavním programovacím jazykem pro programování logiky aplikace kdy byla doplněna jazykem Kotlin. [24]

6.1.2.2 Kotlin

Kotlin je relativně nový programovací jazyk vyvinutý jako alternativa pro jazyk Java vyvinutý v roce 2011 společností JetBrains. Kód pro Android napsaný v kotlinu se stále kompiluje do Java bytekódu a běží v JVM (Java Virtual Machine). [25] Od roku 2019 je Kotlin oficiálně preferovaným jazykem pro vývoj aplikací pro Android [26]

6.1.2.3 XML

XML neboli Extensible Markup Language je rozšiřitelný značkovací jazyk který byl vyvinut konsorciem W3C. [26] Tento jazyk má široké využití, avšak v Android vývoji se používá

pro návrh rozvržení prvků na obrazovce, definici zdrojů jako animací a obrázků nebo konfiguračního souboru AndroidManifest.

6.2 Implementace

Aplikace byla implementována s využitím poznatků popsaných v předchozích kapitolách. Tato kapitola popisuje vybrané části zdrojového kódu.

6.2.1 Registrace uživatele

Pro registrace i přihlašování uživatelů bylo využito služby Firebase Authentication. Pro její využití je nutné nejprve zkontrolovat a upravit vstupy uživatele. Toto ukazuje Kód 3.

```
btnRegister.setOnClickListener {
    var txtName = regName.text.toString().trim()
    var txtMail = regMail.text.toString().trim()
    var txtPass = regPass.text.toString().trim()
    var txtPassConf = regPassConf.text.toString().trim()

    if(TextUtils.isEmpty(txtName)) {
        regName.error = "Jméno je vyžadováno"
        return@setOnClickListener
    }
    if(TextUtils.isEmpty(txtMail) || !HelperFunctions.isEmailVa-
lid(txtMail)) {
        regMail.error = "Email je vyžadován"
        return@setOnClickListener
    }
    if(TextUtils.isEmpty(txtPass) || txtPass.length < 6 || txtPass.length
> 15) {
        regPass.error = "Heslo musí mít 6-15 znaků"
        return@setOnClickListener
    }
    if(!TextUtils.equals(txtPass, txtPassConf)) {
        regPassConf.error = "Heslo není shodné"
        return@setOnClickListener
    }

    //Register the user
    registerUser(txtName, txtMail, txtPass)

}
```

Kód 3. Kontrola vstupů pro registraci

Následně je zavolána funkce, kde již probíhá žádost o registraci uživatele na Firebase Authentication server, viz Kód 4. Pokud žádost proběhla úspěšně, je uživatel přesměrován na hlavní aktivitu pomocí startActivity(), která obsahuje hlavní menu jako Fragment. Starrá aktivita je zavřena metodou finish().

```
private fun registerUser(name: String, mail: String, pass: String) {
    progressBar.visibility = View.VISIBLE

    fAuth.createUserWithEmailAndPassword(mail, pass).addOnCompleteListener
{
        if(it.isSuccessful) {
            Toast.makeText(activity, "Účet vytvořen",
Toast.LENGTH_SHORT).show()

            val intent = Intent(context, MainActivity::class.java)
            intent.putExtra("userName", name)
            startActivity(intent)

            activity?.finish()
        } else {
            Toast.makeText(context, "Error ! " + it.exception?.message,
Toast.LENGTH_SHORT).show()
            progressBar.visibility = View.GONE
        }
    }

}
```

Kód 4. Žádost o registraci

6.2.2 Přihlášení a odhlášení uživatele

Přihlášení probíhá velmi podobně jako registrace. Nejprve proběhne kontrola vstupů a poté je zavolána funkce s parametry pro emailovou adresu a heslo uživatele. Tyto jsou na straně serveru porovnány a v případě shody je uživatel přesměrován do menu.

```
private fun loginUser(mail: String, pass: String) {
    progressBar.visibility = View.VISIBLE
    fAuth.signInWithEmailAndPassword(mail, pass).addOnCompleteListener {
        if(it.isSuccessful) {
            Toast.makeText(context, "Úspěšné přihlášení",
Toast.LENGTH_SHORT).show()
            startActivity(Intent(context, MainActivity::class.java))
            activity?.finish()
        } else {
            Toast.makeText(context, "Error ! " + it.exception?.message,
Toast.LENGTH_SHORT).show()
            progressBar.visibility = View.GONE
        }
    }
}
```

Kód 5. Žádost o přihlášení

Odhlášení je taktéž velmi snadné, díky použití Firebase Authentication. Po odhlášení je uživatel přesměrován na začátek aplikace.

```
private fun logoutUser() {
    fAuth.signOut()
    googleSignInClient.signOut()
    startActivity(Intent(context, SplashActivity::class.java))
    activity?.finish()
}
```

Kód 6. Odhlášení uživatele

6.2.3 Načítání QR kódů

Pro načítání QR kódu byla implementována knihovna k tomu určena. Nejprve je inicializován view a instance CodeScanneru. Jakmile je přečten nějaký QR kód, tak se nejprve zjistí, zda je validní. Pokud ano, je uživatel přesměrován na další obrazovku. Pokud ne, je uživatel na toto upozorněn funkcí Toast. Uživatel má také možnost dotykem na obrazovku vynutit zaostření.

```
val scannerView = view.findViewById<CodeScannerView>(R.id.scanner_view)
val activity = requireActivity()
codeScanner = CodeScanner(activity, scannerView)
codeScanner.decodeCallback = DecodeCallback {
    activity.runOnUiThread {
        if(it.text.take(13) == "{ExploreInAr,") {
            val bundle = bundleOf("qrCodeData" to it.text)
            findNavController().navigate(R.id.action_scannerFrag-
ment_to_infoFragment, bundle)
        } else {
            Toast.makeText(activity, "Neznámý kód",
Toast.LENGTH_LONG).show()
            findNavController().popBackStack()
        }
    }
}

// focus
scannerView.setOnClickListener {
    codeScanner.startPreview()
}
```

Kód 7. Načtení QR kódu

6.2.4 RecyclerView

RecyclerView je widget, kterého lze s výhodou využít, pokud naše aplikace obsahuje takový prvek jedné obrazovky, který se často opakuje. Typicky se tak může jednat o různé seznamy např. kontaktů. V případě vyvíjené aplikace se nabízí využití pro tabulku výsledků. Hlavní funkcionalitu tohoto widgetu zajišťuje jeho adaptér. Adaptér vytváří

v onCreateViewHolder() ViewHoldery, které drží tzv. prázdné inflated views. Jakmile jsou jednou vytvořeny tak jsou drženy ve vyrovnávací paměti pro další použití. Po zavolání onBindViewHolder(), kterému je ViewHolder předán jako parametr, jsou jeho parametry plněny daty modelové třídy UserScoreData.

```
private class UserFirestoreRecyclerAdapter internal constructor(options:
FirestoreRecyclerOptions<UserScoreData>) : FirestoreRecyclerAdapter<UserS-
coreData, UserViewHolder>(options) {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
UserViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.lay-
out.item_score, parent, false)
        return UserViewHolder(view)
    }

    override fun onBindViewHolder(userViewHolder: UserViewHolder, posi-
tion: Int, userScoreData: UserScoreData) {
        userViewHolder.setUserName(userScoreData.userName)
        userViewHolder.setUserScore(userScoreData.score)
        userViewHolder.setUserPosition(position+1)
    }
}
```

Kód 8. Adaptér RecyclerView

```
private class UserViewHolder internal constructor(private val view: View)
: RecyclerView.ViewHolder(view) {

    internal fun setUserName(userName: String) {
        val tvName = view.findViewById<TextView>(R.id.tv_name)
        tvName.text = userName
    }

    internal fun setUserScore(score: Long) {
        val tvScore = view.findViewById<TextView>(R.id.tv_score)
        tvScore.text = score.toString()
    }

    internal fun setUserPosition(position: Int) {
        val tvPosition = view.findViewById<TextView>(R.id.tv_position)
        tvPosition.text = position.toString()
    }
}
```

Kód 9. Plnění RecyclerView daty

6.2.5 Ukládání dat do Firestore databáze

Pro ukládání dat bylo využito NoSQL Firestore databáze. Ukládání probíhá na více místech v aplikaci, zde si ukážeme, jak se vytváří nová uživatelova tabulka po jeho registraci.

```
private fun userIsNew(name: String) {
   val docRef = fStore.collection("users").document(userId)
    docRef.get()
        .addOnSuccessListener { document ->
            if (document != null) {
                Log.d(TAG, "User doc found")
                val mail = fAuth.currentUser!!.email.toString()
                val userMap : HashMap<String, Any> = HashMap()
                val completed = emptyList<Int>()
                userMap.put("userName", name)
                userMap.put("email", mail)
                userMap.put("score", 0)
                userMap.put("completed", completed)
                docRef.set(userMap).addOnSuccessListener {
                    Log.d(TAG, "onSuccess: profile created in firestore
for " + userId)
                }.addOnFailureListener {
                    Log.d(TAG, "onFailure " + it.toString())
                }
            } else {
                Log.d(TAG, "No such user doc")
            }
        }
        .addOnFailureListener { exception ->
            Log.d(TAG, "get failed with ", exception)
        }
}
```

Kód 10. Vložení dat novému uživateli do Firestore databáze

Nejprve se získá nutná reference na dokument v kolekci *users*. Ten se získá metodou get() a pokud byl nalezen, tak metodou set() této reference vložíme data do databáze.

6.2.6 Načítání dat z Firestore databáze a Firebase Storage

Načítání dat z Firestore databáze má podobný průběh jako její ukládání. Nejdříve získáme referenci na požadovaný dokument, získáme jej metodou get() a následně plníme získaná data do modelové třídy InfoData.

```
docRefInfo.get()
    .addOnSuccessListener { document ->
        if (document != null) {
            loadSuccess = true
            infoData = InfoData(document.getString("name")!!, document.ge-
tString("wiki")!!, document.getString("image")!!, document.getString("in-
foText")!!.replace("\\n", System.getProperty("line.separator")!!, false))
            displayData()
        } else {
            Log.d(TAG, "No such document")
            findNavController().popBackStack()
        }
    }
```

Kód 11. Načítání dat z Firestore databáze

Dále bylo pro pouhé ukládání obrázku využito cloudového úložiště Firebase Storage. Hlavní výhodou volby této služby bylo to, že spolupracuje s Firebase Authentication, není tedy potřeba dalšího kódu pro to, aby mohli pouze uživatelé této aplikace toto úložiště využívat. Na ukázce kódu Kód 12 je stažení obrázku z tohoto úložiště a jeho následné zobrazení knihovnou Glide uvnitř widgetu ImageView jménem iv_info.

```
val gsReference = storage.getReferenceFromUrl(infoData.infoImageUrl)
Glide.with(this)
    .load(gsReference)
    .centerCrop()
    .into(iv_info)
```

Kód 12. Stažení a zobrazení obrázku z Firebase Storage

6.2.7 Zobrazení objektu v rozšířené realitě

Jak již bylo zmíněno v kapitole 3.4.1, pro rozšířenou realitu na operačním systému Android je oficiální knihovnou ARCore. Ta tedy byla využita i pro zobrazení informací v této aplikaci. Nejprve aplikace čeká na vstup uživatele ve formě dotknutí se obrazovky. Pokud je v tuto chvíli také nalezena vhodná rovina pro umístění objektu je v tomto místě vytvořeno jeho ukotvení a pokračuje se s jeho vykreslením, viz Kód 13.

```
arFragment.setOnTapArPlaneListener { hitResult: HitResult, plane: Plane,
motionEvent: MotionEvent ->
    if (plane.type != Plane.Type.HORIZONTAL_UPWARD_FACING) {
        return@setOnTapArPlaneListener
    }
    val anchor = hitResult.createAnchor()
    placeObject(arFragment, anchor)
}
```

Kód 13. Zpracování vstupu uživatele a vyhodnocení roviny

Aplikace dále vytvořením objektu k vykreslení. Je mu nastaven View metodou setView() a škálování velikosti metodou setSizer(). Poté jsou do jeho widgetů vložena data.

```
ViewRenderable.builder()
        //bind layout
    .setView(fragment.context, R.layout.viewrenderable_info)
    .setSizer(DpToMetersViewSizer(300))
    .build()
    .thenAccept {
        it.isShadowCaster = false
        it.isShadowReceiver = false
        it.view.findViewById<TextView>(R.id.tv_ar_info).text = in-
foData.infoText
        val iv_info = it.view.findView-
ById<ImageView>(R.id.iv_ar_info_image)

        val storage = FirebaseStorage.getInstance()
        val gsReference = storage.getReferenceFromUrl(infoData.in-
foImageUrl)
        //load image
        Glide.with(this)
            .load(gsReference)
            .into(iv_info)

        addNode(it, anchor, fragment)
    }
```

Kód 14. Tvorba objektu k vykreslení

Nakonec proběhne samotné umístění vytvořeného objektu do prostoru na vybranou pozici uživatelem, viz Kód 15.

```
private fun addNode(renderable: Renderable, anchor: Anchor, arFragment:
ArFragment ) {
    val anchorNode = AnchorNode(anchor)
    val transformableNode = TransformableNode(arFragment.transformation-
System)
    transformableNode.renderable = renderable
    transformableNode.setParent(anchorNode)
    arFragment.arSceneView.scene.addChild(anchorNode)
}
```

Kód 15. Umístění objektu do prostoru

6.3 Testování aplikace

Každou aplikaci je nutné před jejím vydáním otestovat tak, aby se minimalizovala šance nečekaných chyb způsobených konfiguracemi různých zařízení nebo nezvyklým používáním. Hlavními možnostmi praktického testování Android aplikací jsou:

- **Emulátory**
  Mimo oficiální Android Emulátor existují i další jako např. *Genymotion*, z nichž některé nabízí i emulovat přímo zařízení jednotlivých výrobců
- **Vzdálený pronájem reálného zařízení**
  Existují také služby, které nabízí možnost otestovat aplikaci vzdáleně na reálném zařízení. Jmenovitě mezi ně patří např. Samsung Remote Test Lab, Google Firebase Test Lab a Amazon AWS Test Farm
- **Využití fyzického zařízení**

Hlavní testování aplikace probíhalo na emulátoru z integrovaného vývojového prostředí Android Studio. Jedná se o nejrychlejší možnost, jak otestovat určitou funkci při vývoji. Otestovány byly všechny podporované verze API až po nejnovější.

Dále byla také aplikace reálně otestována reálnými uživateli na těchto zařízeních:

Tab. 2. Tabulka otestovaných zařízení

Název zařízení	Verze OS Android	Verze API
Samsung Galaxy S8	9.0	28
LG G6	8.1	27
Oneplus Nord	10	29
Xiaomi Mi A3	10	29
Samsung Galaxy A8 (2018)	9.0	28

ZÁVĚR

Čtenář této práce byl seznámen se specifiky vývoje aplikací pro operační systém Android. Teoretická část se zabývala obecnými informacemi o tomto operačním systému, možnostmi vývoje mobilních aplikací a popisu základních komponent z nichž se skládá. Dále teoretická část popisuje pojmy jako virtuální a rozšířená realita a věnuje se také jejich vývojovým nástrojům.

Praktická část definuje funkční a nefunkční požadavky na vyvíjenou aplikaci. Poté se zabývá ukázkou a popisem jednotlivých obrazovek, jejich funkcemi a komponentami z nichž se skládá. Následně popisuje nástroje použité k vývoji aplikace a programovací jazyky. Hlavní částí však je popis úryvků zdrojového kódu, který obstarává hlavní funkce aplikace. Závěr praktické části je věnován praktickému testování aplikace.

Hlavním cílem této diplomové práce bylo vyvinout nativní Android aplikaci s využitím rozšířené reality umožňující zobrazit informace o detekovaných zájmových bodech dle zadání této práce a funkčních a nefunkčních požadavků stanovených v praktické části. Tento cíl byl splněn. Aplikace byla vyvinuta s využitím moderních technologií a postupů, přičemž části vývoje s ukázkami obrazovek a úryvků kódů byly popsány v praktické části. Aplikace byla taktéž prakticky otestována a nic tedy nebrání její publikaci na vybrané platformě pro distribuci aplikací pro operační systém Android.

Aplikace může být taktéž v rozšířena. Mezi možnosti budoucího vylepšení patří například přidávání zájmových bodů uživateli přímo z aplikace, podpora dalších jazyků, zobrazování dalších informací o zájmovém bodu v rozšířené realitě nebo možnost zobrazit lokaci všech bodů na mapě.

SEZNAM POUŽITÉ LITERATURY

[1] WIKIPEDIA. Smartphone. Wikimedia Foundation. [online]. 2020 [cit. 2020-01-20]. Dostupné z: https://en.wikipedia.org/wiki/Smartphone

[2] STATCOUNTER. Operating System Market Share Worldwide [online]. 2020 [cit. 2020-02-03]. Dostupné z: https://gs.statcounter.com/os-market-share

[3] CALLAHAM, John. The history of Android OS: its name, origin and more. Android Authority [online]. 2019 Android Authority [cit. 2020-02-03]. Dostupné z: https://www.androidauthority.com/history-android-os-name-789433/

[4] STATCOUNTER. Mobile & Tablet Android Version Market Share Worldwide [online]. 2020 [cit. 2020-03-08]. Dostupné z: https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide

[5] GOOGLE INC. Platform Architecture. Android Developers [online]. 2020 [cit. 2020-05-09]. Dostupné z: https://developer.android.com/guide/platform

[6] KALTOUN, Jan. Is Native, React Native, Flutter or PWA Right for You? STRV [online]. 2020 [cit. 2020-05-09]. Dostupné z: https://www.strv.com/blog/is-native-react-native-flutter-or-pwa-right-for-you-engineering-business

[7] STARDUST. Hybrid Apps: An overview of Advantages, Limitations & Consequences for your Testing Phases. [online]. 2020 [cit. 2020-05-09]. Dostupné z: https://www2.stardust-testing.com/en/blog-en/hybrid-apps

[8] GOOGLE INC. Fragments. Android Developers [online]. 2020 [cit. 2020-05-09]. Dostupné z: https://developer.android.com/guide/components/fragments

[9] GOOGLE INC. Content Providers. Android Developers [online]. 2020 [cit. 2020-05-09]. Dostupné z: https://developer.android.com/guide/topics/providers/content-providers.html

[10] GOOGLE INC. ViewModel Overview. Android Developers [online]. 2020 [cit. 2020-05-09]. Dostupné z: https://developer.android.com/topic/libraries/architecture/viewmodel

[11] GOOGLE INC. Data Binding Library. Android Developers [online]. 2020 [cit. 2020-05-09]. Dostupné z: https://developer.android.com/topic/libraries/data-binding

[12] GOOGLE INC. LiveData Overview. Android Developers [online]. 2020 [cit. 2020-05-09]. Dostupné z: https://developer.android.com/topic/libraries/architecture/livedata

[13] GOOGLE INC. Navigation. Android Developers [online]. 2020 [cit. 2020-05-20]. Dostupné z: https://developer.android.com/guide/navigation

[14] GOOGLE INC. Schedule tasks with WorkManager. Android Developers [online]. 2020 [cit. 2020-05-20]. Dostupné z: https://developer.android.com/topic/libraries/architecture/workmanager

[15] GOOGLE INC. Save data in a local database using Room. Android Developers [online]. 2020 [cit. 2020-05-20]. Dostupné z: https://developer.android.com/training/data-storage/room

[16] HTC CORPORATION. Veryfying your setup. Vive [online]. 2020 [cit. 2020-05-25]. Dostupné z: https://www.vive.com/eu/support/vive-pro-hmd/category_howto/verifying-your-setup.html

[17] MATNEY, Lucas. Phiar raises $3 million for an AR navigation app for drivers. Techcrunch [online]. 2018 [cit. 2020-05-25]. Dostupné z: https://techcrunch.com/2018/11/28/phiar-nabs-3-million-for-an-ar-navigation-app-for-drivers/

[18] INTEL CORPORATION. Demystifying the Virtual Reality Landscape [online]. 2020 [cit. 2020-05-25]. Dostupné z: https://www.intel.com/content/www/us/en/tech-tips-and-tricks/virtual-reality-vs-augmented-reality.html

[19] GOOGLE INC. ARCore overview. Android Developers [online]. 2020 [cit. 2020-05-30]. Dostupné z: https://developers.google.com/ar/discover

[20] JAFFE, Josh. Comparing ARKit vs ARCore vs Vuforia: The Best Augmented Reality Toolkit. Bluewhale [online]. 2020 [cit. 2020-05-30]. Dostupné z: https://bluewhaleapps.com/blog/comparing-arkit-vs-arcore-vs-vuforia-the-best-augmented-reality-toolkit

[21] KOVACH, Nadia. Best AR SDK for development for iOS and Android. Thinkmobiles [online]. 2020 [cit. 2020-05-09]. Dostupné z: https://thinkmobiles.com/blog/best-ar-sdk-review/

[22] GOOGLE INC. ARCore supported devices. Android Developers [online]. 2020 [cit. 2020-08-10]. Dostupné z: https://developers.google.com/ar/discover/supported-devices

[23] ORACLE. What is Java technology and why do I need it? Oracle [online]. 2020 [cit. 2020-08-10]. Dostupné z: https://java.com/en/download/faq/whatis_java.xml

[24] MILLER, Paul. Google is adding Kotlin as an official programming language for Android development. The Verge [online]. 2017 [cit. 2020-08-10]. Dostupné z: https://www.theverge.com/2017/5/17/15654988/google-jet-brains-kotlin-programming-language-android-development-io-2017

[24] WIKIPEDIA. Kotlin (programming language). Wikimedia Foundation. [online]. 2020 [cit. 2020-08-10]. Dostupné z: https://en.wikipedia.org/wiki/Kotlin_(programming_language)

[25] LARDINOIS, Frederic. Kotlin is now Google's preferred language for Android app development. Techcrunch [online]. 2019 [cit. 2020-08-10]. Dostupné z: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/

[26] WIKIPEDIA. Smartphone. Wikimedia Foundation. [online]. 2020 [cit. 2020-08-10]. Dostupné z: https://cs.wikipedia.org/wiki/Extensible_Markup_Language

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AOSP	Android Open Source Project
AOT	Ahead Of Time
API	Application Programming Interface
APK	Android Package Kit
AR	Augmentovaná Realita
ART	Android RunTime
CSS	Cascade Style Sheets
DAO	Data Access Object
HAL	Hardware Acccess Layer
HTML	HyperText Markup Language
JIT	Just In Time
JVM	Java Virtual Machine
MR	Mixed Reality
OS	Operační systém
SSL	Secure Sockets Layer
UI	User Interface
VR	Virtuální Realita
XML	eXtensible markup language

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM KÓDŮ

SEZNAM PŘÍLOH

P I CD-ROM s textem této práce, instalačním souborem vyvinuté aplikace, jejími zdrojovými kódy a QR kódy k jejímu použití