Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Návrh a realizace mobilní aplikace s rozšířenou realitou

Bc. Tomáš Kratochvíl

Diplomová práce

2019

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš Kratochvíl**
Osobní číslo:	**I15213**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Návrh a realizace mobilní aplikace s rozšířenou realitou**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Cílem diplomové práce bude v teoretické části charakterizovat a popsat rozšířenou realitu a analyzovat její využitelnost v mobilních aplikacích. Součástí práce bude i přehled identifikačních a lokačních bezdrátových technologií potřebných pro umísťování digitálních objektů v obrazu reálného světa na displeji mobilního zařízení, identifikace jejich výhod a nevýhod. Druhým cílem v praktické části diplomové práce bude navrhnout a realizovat mobilní aplikaci (pro operační systém Android/iOS/Windows Phone (volitelně)) či multiplatformní mobilní aplikaci v Adobe PhoneGap. Ta bude využívat identifikačních (např. QR kódy) a lokačních (např. GPS v kombinaci s gyroskopem a akcelerometrem) bezdrátových technologií potřebných pro umísťování digitálních objektů v obrazu reálného světa na displeji mobilního zařízení.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

***SOOD, Raghav. Pro Android augmented reality. New York: Apress, c2012, xvi, 329 p. ISBN 143023945x.**
***ALAN B. CRAIG. Understanding augmented reality concepts and applications. Online-Ausg. Amsterdam: Morgan Kaufmann, 2013. ISBN 9780240824109.**
***GREGORY KIPPER, Joseph Rampolla. Augmented Reality an Emerging Technologies Guide to AR. 1st ed. Rockland, MA: Elsevier Science, 2012. ISBN 9781597497343.**

Vedoucí diplomové práce: **Ing. Jiří Kysela, Ph.D.**
Katedra informačních technologií

Datum zadání diplomové práce: **31. října 2016**

Termín odevzdání diplomové práce: **17. května 2017**

Ing. Zdeněk Němec, Ph.D.
děkan

Univerzita Pardubice
L.S.
Česká republika

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2016

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 23. srpna 2019

Bc. Tomáš Kratochvíl

PODĚKOVÁNÍ

Na tomto místě bych rád poděkoval svému vedoucímu práce Ing. Jiřímu Kyselovi, Ph.D. za poskytnutí tématu, možnosti zpracování pod jeho vedením včetně podnětných připomínek při tvorbě této práce. V neposlední řadě bych chtěl také poděkovat všem, kteří mě podporovali po celou dobu studia, ať už to byli rodiče, sourozenci, přítelkyně či spolužáci a kamarádi.

ANOTACE

Tato diplomová práce je zaměřena na charakterizaci rozšířené reality a její využitelnosti na mobilních platformách. Jsou popsány vybrané bezdrátové identifikační a lokační technologie, které jsou pro správou funkci rozšířené reality nejen na mobilních platformách klíčové. Rozebrány jsou majoritní mobilní platformy včetně aspektů mobilního vývoje. Provedena je analýza a návrh aplikace, kde jsou popsány požadavky na aplikaci a pomocí navržené metodiky vybrána mobilní platforma Android, pro kterou jsou popsány základní aspekty vývoje aplikací včetně vymezení požadovaných technologií pro vývoj rozšířené reality na této platformě. Závěreční část je věnována realizaci mobilní aplikace postavené na platformě Sceneform/ARCore a testování v reálném prostředí.

KLÍČOVÁ SLOVA

Rozšířená realita, bezdrátové identifikační a lokační technologie, mobilní platformy, aplikace, Android, Sceneform/ARCore

ANNOTATION

This diploma thesis is focused primarily on characterization of augmented reality and its usability on mobile platforms. Selected wireless identification and location technologies are described in detail, which are crucial for the right function of augmented reality not only on mobile platforms. Major mobile platforms are thoroughly analyzed, including certain aspects of mobile phone development. Analysis and application design are performed, wherein application requirements are carefully described. Then, using the designed methodics, the mobile platform Android is selected, for which the description of the basic aspects of application development, including definitions of required technologies for development of augmented reality, is present. The final part is devoted to implementation of the mobile application set on the platform Sceneform/ARCore and testing in authentic environment.

KEYWORDS

Augmented reality, wireless identification and location technology, mobile platforms, application, Android, Sceneform/ARCore

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

TZV.	Takzvané
AR	Augmented Reality
NAPŘ.	Například
3D	3-Dimension
POI	Point Of Interest
AV	Augmented Virtuality
MR	Mixed Reality
HAD	Head-Attached Displays
HW	Hardware
PC	Personal Computer
TV	Television
GPS	Global Positioning System
GLONASS	„Global Navigation Satellite System”
IEEE	Institute of Electrical and Electronics Engineers
WI-FI	Wireless Fidelity
SW	Software
RFID	Radio Frequency Identification
NFC	Near Field Communication
1D	1-Dimension
EAN	European Article Number
UPC	Universal Product Code
ISO	International Organization for Standardization

IEC	International Electrotechnical Commission
2D	2-Dimension
QR	Quick Response
iQR	„Information“QR
SQRC	Secret-function-equipped QR code
URL	Uniform Resource Locator
P2P	„Peer-to-peer“
JIS	Japanese Industrial Standards
MEMS	Micro Electro Mechanical Systems
NEMS	Nano Electro Mechanical Systems
IMU	Inertial Measurement Unit
LOS	Line Of Sight
WGS-84	World Geodetic System-84
S-JTSK	Systém Jednotné Trigonometrické Sítě Katastrální
NLOS	Non Line Of Sigh
A-GPS	Assisted GPS
IP	Internet Protocol
DNS	Domain Name Systém
2G	2„Generation“
GSM	Global System for Mobile Communications
3G	3„Generation“
UMTS	Universal Mobile Telecommunications System
LTE	Long Term Evolution
4G	4„Generation“

COO	Cell Of Origin
TOA	Time Of Arrival
OS	Operating System
OHA	Open Handset Alliance
SDK	Software Development Kit
Inc.	Incorporated
API	Application Programming Interface
ART	Android Runtime
JVM	Java Virtual Machine
DEX	Dalvik Executable
XML	Extensible Markup Language
XNU	„X is Not UNIX“
BSD	Berkeley Software Distribution
UI	User Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
MVC	Model-View-Controller
MVP	Model-View-Presenter
MVVM	Model-View-ViewModel
WPF	Windows Presentation Foundation
MVCC	Multiversion Concurrency Control
ACID	Atomic-Consistent-Isolated-Durable
HTTP	Hypertext Transfer Protocol
DI	Dependency Injection

JSON	JavaScript Object Notation
JPG	„Joint Photographic Experts Group“
IDE	Integrated Development Environment
SMS	Short Message Service
CRUD	Create-Retrieve-Update-Delete
UML	Unified Modeling Language
IoC	Inversion of Control
PNG	Portable Network Graphics
JPEG	„Joint Photographic Experts Group“
ASCII	American Standard Code For Information Interchange

ÚVOD

Smartphony, takzvané (tzv.) chytré telefony, jsou v současné moderní době trendem, bez kterého si nedokážeme prakticky představit život. S poměrně nízkými pořizovacími náklady se jeví jako optimální platforma pro přístup k informacím a nejrůznějším poskytovaným službám. Především s příchodem technologického rozmachu těchto zařízení v oblasti výpočetního výkonu, paměťových možností, velikostí/rozlišení zobrazovacího displeje, optimalizace spotřeby baterie a její kapacity a senzorických vlastností čelí tyto platformy expanzivním proměnám v jejich využití. Příkladem může být stále více se rozvíjející odvětví rozšířené reality, které se díky tomu více dostává do povědomí běžných uživatelů. Do nasnímaného pohledu reálného prostřední je vložen konvenční generovaný digitální objekt, který umožňuje mapovat libovolný bod zájmu uživatele a tím vizuálně zpříjemnit, zpřehlednit a v neposlední řadě případně optimalizovat vykonávání nejrůznějších volnočasových nebo pracovních aktivit.

Cílem diplomové práce je navrhnout na libovolnou mobilní platformu aplikaci využívající rozšířenou realitu pro zobrazení digitálních bodů zájmu reálného světa, za použití identifikačních a lokačních bezdrátových technologií. Aplikace řešená v rámci diplomové práce se sestává z prohlídky místopisných bodů zájmu, jako jsou historicky zajímavá a turisticky atraktivní místa, možnosti sportovního vyžití v dané lokalitě, přírodní zajímavosti a další informace, které zastupují jednotlivé body zájmu a pro ně jsou poskytnuty doplňující informace jak přes klasický mapový podklad, tak i s využitím rozšířené reality přes vestavěnou kameru daného mobilního zařízení. Potencionální využitelnost je tedy především v cestovním ruchu, kdy může uživatel získat jednoduše a pohodlně představu o geografické poloze všech vybraných bodů zájmu s možností grafické navigace.

První kapitola teoretické části diplomové práce je zaměřena na zasazení kontextu rozšířené reality a její základní kategorizace z různých úhlů pohledů. Ve druhé kapitole je nastíněn popis obecné architektury procesu realizace této reality na libovolném systému včetně obeznámení se čtyřmi hlavními podprocesy. Uvedena je také základní aplikovatelnost rozšířené reality na mnoha různých aplikačních doménách. Třetí kapitola seznamuje s rozšířenou realitou na mobilních platformách, především z pohledu technologií, které jsou využívány pro zajištění korektního umisťování digitálních objektů do reálné scény, detailněji je proto provedeno seznámení se základním rámcem identifikačních a lokačních bezdrátových technologií s popisem jejich výhod a nevýhod.

Závěrečná kapitola teoretické části je obohacena o aktuální srovnání majoritních mobilních platforem podle podílu na trhu včetně rámcového popisu historie jednotlivých platforem, charakteristických rysů, vývoje a distribuce aplikací. Představeny jsou nejpoužívanější přístupy mobilního vývoje aplikací a obecné varianty návrhových vzorů při realizaci aplikace.

První kapitola praktické části je věnována analýze a návrhu aplikace. Specifikovány jsou primární požadavky na vyvíjenou aplikaci, odůvodnění výběru přístupu mobilního vývoje a výběru optimální mobilní platformy, na kterou bude cíleno – Google Android. Dále je součástí volba architektury aplikace a z pohledu analýzy klíčový výběr technologií pro rozšířenou realitu na platformě Android. V neposlední řadě je proveden návrh uživatelského rozhraní aplikace, a také je zahrnut popis existujících „konkurenčních" aplikací z domény cestovního ruchu. V páté kapitole jsou rámcově popsány základní specifika při vývoji aplikace pro platformu Android. Následuje šestá kapitola, která si klade za cíl popsat strukturu aplikace a její jednotlivé části, ze kterých je složena. Komentovány jsou využité frameworky, hlavní metody a aplikované principy využité při vývoji aplikace. Nedílnou součástí praktické části je i testování výsledné aplikace na reálném zařízení a v reálném prostředí, což popisuje sedmá kapitola. Osmá kapitola obsahuje uživatelskou příručku pro vytvořenou aplikaci.

V závěrečné části práce jsou zhodnoceny dosažené výsledky a uvedeny možnosti budoucího rozšíření realizované aplikace.

1 ROZŠÍŘENÁ REALITA

„Rozšířenou realitu neboli augmentovanou realitu (také označovanou zkratkou AR) můžeme definovat jako pohled na skutečnou realitu prostřednictvím technického zařízení, které tuto okamžitou realitu pozorovateli rozšiřuje o počítačově vytvořené smyslové prvky.“

Definice uváděná Barborou Kohoutovou v článku [1]

1.1 Terminologie AR

Rozšířená realita (dále také uváděno jako AR[1]) je technologie, která je v posledních letech bezesporu moderním trendem, se kterým se více či méně každý setkává nebo se již mohl alespoň v nějaké míře setkat. Společnosti[2], zabývající se analýzami a průzkumy trhu a predikcí vývoje a trendů v oblasti informačních a komunikačních technologií, očekávají především v budoucích letech ještě intenzivnější vývoj a rozmach této technologie.

Obrázek 1: Rozšířená realita. Zdroj: vlastní.

Ačkoli se tedy může zdát AR jako nové odvětví, které se rozmohlo teprve s rozmachem počítačové a mobilní techniky, není tomu tak, a již minimálně od poloviny 20. století prochází neustálým technologickým rozvojem. Termín rozšířená realita byl ale obecně přijat až v počátcích 90. let, kdy byl pravděpodobně poprvé použit Tomem Caudellem a jeho kolegou Davidem Mizzelem[3]. Motivací pro zavedení této reality bylo zlepšení pracovního procesu v oblasti leteckého průmyslu [2].

---

[1] Rozšířená realita vychází z anglického termínu Augmented Reality (odtud také vžitá zkratka AR), v odborné literatuře a veřejně dostupných článcích se lze setkat s různými dalšími počeštělými názvy, jako je rozšiřující realita, obohacená realita, augmentovaná realita a další.

[2] Lze uvést společnosti, jako jsou International Data Corporation nebo Statista, jejichž vydané výzkumy jsou citovány v různých odborných článcích nejen o rozšířené realitě.

[3] V některých pramenech je uváděn pouze Tom Caudell.

Na první pohled by se mohlo zdát, že rozšířená realita je definovaná přesně danými předpoklady nebo dokonce požadavky. Přesto definovat AR není zcela jednoduché, jelikož pro ni neexistuje jednotná formulace – jedná se totiž o komplexní technologii, která nemusí být omezena pouze na vizuální[4] vnímání uživatele, ale i na ovlivňování všech lidských smyslů.

Takový popis ukazuje široké spektrum toho, jak je možné tuto realitu chápat, a nejspíše až čas a vývoj této technologie ukáže, jaká definice se mezi odborníky a veřejností ustálí.

Mezi všeobecně kladené požadavky[5] lze zejména zařadit:

- *kombinace reálné reality a digitálních senzorických objektů;*
- *definice a operativnost ve 3D prostředí;*
- *interaktivnost v reálném čase;*
- *smysluplný kontext digitálních objektů a příslušných dat.*

V současné době existuje nespočet digitálních medií a mobilních/počítačových aplikací, které jsou počítačově vylepšovány, a přesto nespadají do oboru AR, protože nesplňují výše uvedené požadavky. Příkladem může být úprava fotografií v grafickém editoru nebo nejrůznější filmové žánry, které využívají generované objekty pro zlepšení vizuální/zvukové scény. Různé pohledy na AR a jejich definice s požadavky lze doplňkově vyhledat například (např.) v [2], [3] a [4].

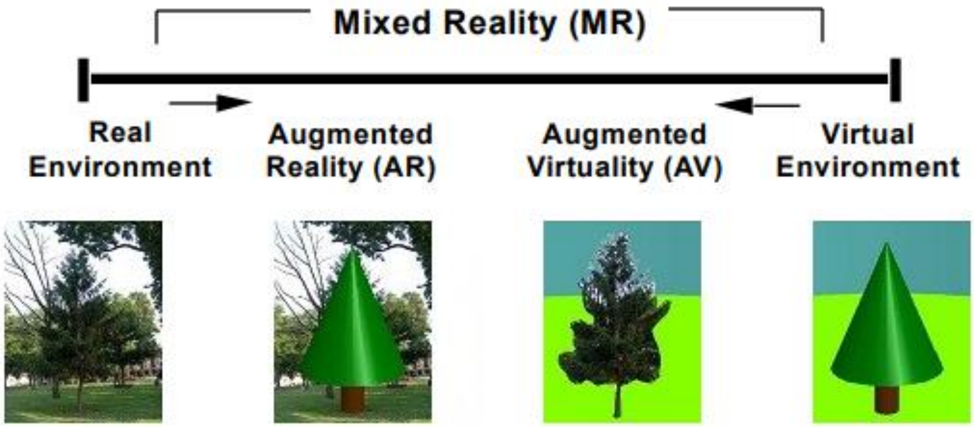
„Podstatou rozšířené reality (také označovanou AR) je koncept, aplikovaný v různých doménách působnosti, který umožňuje ve smysluplném kontextu, v rámci tzv. bodů zájmu[6] uživatele, obohatit reálnou scénu o interaktivní[7], počítačově generované smyslové objekty[8] v reálném čase prostřednictvím nejrůznějšího technického vybavení.“

Vlastní definice pro rozšířenou realitu

Někdy bývá rozšířená realita mylně označována jako virtuální realita, která si klade za cíl vytvoření plně syntetického prostředí, zatímco rozšířená realita pouze doplňuje skutečnou realitu, než aby ji plně nahrazovala.

[4] V současnosti je veřejnosti bezesporu nejvíce známá.
[5] Požadavky AR jsou dány tzv. kompromisy mezi navrhovanou a „ideální“ definicí jednotlivých autorů.
[6] Alternativně jsou jednotlivé body zájmu nazývány anglickým termínem Point Of Interest neboli POI.
[7] Generované objekty umožňují přímý vstup uživatele do vytvořené reality.
[8] Těmito objekty mohou být obrázky, texty, videa, případně dokonce 3D modely a jiné specifické prvky.

Obrázek 2 definuje diagram reálně-virtuálního kontinua vyjadřující vztah mezi různými realitami, kde klasifikace jednotlivých prostředí je dána množstvím použitého digitálního obsahu.



Obrázek 2: Reálně-virtuální kontinuum. Zdroj: [5] a [6].

Rozšířená virtualita (dále také označováno jako AV[9]) se od AR liší ve využívaném prostředí reality. Do této rozšířené reality nebo rozšířené virtuality jsou pak v závislosti stupně obohacení dané scény vkládány počítačově generované objekty nebo naopak reálné objekty. Kombinace AV a AR tvoří spektrum realit označované jako mixovaná realita[10] [6].

1.2 Kategorizace AR

Nejednoznačnost definice, a především komplexnost rozšířené reality, která využívá nejrůznější kombinace technologií, umožňuje vznik existence několika možností, jak kategorizovat rozšířenou realitu z různých úhlů pohledů, které mohou na první pohled vypadat podobně, ale vzhledem k povaze mohou být zcela odlišné – mající různé možnosti aplikovatelnosti, technologických principů nebo technického vybavení.

Nastíněná kategorizace bude lépe umožňovat zasadit koncept rozšířené reality ve smyslu chápání základních principů a jejího dalšího členění. Uvedeno je méně známé, ale z pohledu AR logické, rozdělení podle lidských smyslů dle [4] a dále jsou představena technická rozdělení uváděná v [7] a [8] již pro konkrétní AR, a to „vizuální“, na kterou je tato práce cílena.

[9] Rozšířená virtualita vychází z anglického termínu Augmented Virtuality (zkratka AV).
[10] Mixovaná realita vychází z anglického termínu Mixed Reality (zkratka MR).

1.2.1 Rozdělení podle lidských smyslů

AR nemusí být explicitně vyjádřena pouze jako vizuální, protože již z logického úsudku může být aplikovatelná na libovolné lidské smyslové orgány neboli receptory. Zde je ale možné polemizovat, zda se ještě jedná o AR nebo ne. Toto lze jednoduše pozorovat na sluchové rozšířené realitě, kdy by libovolný generovaný zvuk mohl být zasazen do kontextu AR. V tomto případě však nejsou splněny výše kladené základní požadavky.

Jednotlivé typy tedy jsou:

- *Vizuální AR (Visual AR)*[11] – Jedná se o nejčastější a veřejnosti nejvíce známý typ rozšířené reality, který je založen na využití objektů počítačové grafiky vložené do kontextu reálné scény.
- *Sluchová AR (Audio AR)*[12] – Tato rozšířená realita využívá oproti vizuální AR uměle generované virtuální zvuky nebo reálné zachycené zvuky.
- *Hmatová AR (Haptic AR)*[13] – Tento typ AR je založen na dotyku/cítění generovaných objektů v reálné scéně, pro které je simulován hmatový podnět.
- *Čichová AR (Olfactory AR) a Chuťová AR (Gustatory AR)*[14] – Cílem takovýchto realit je navození receptorů čichových nebo chuťových buněk uživatele přes nasnímání objektů, pro které generuje požadovaný zážitek.

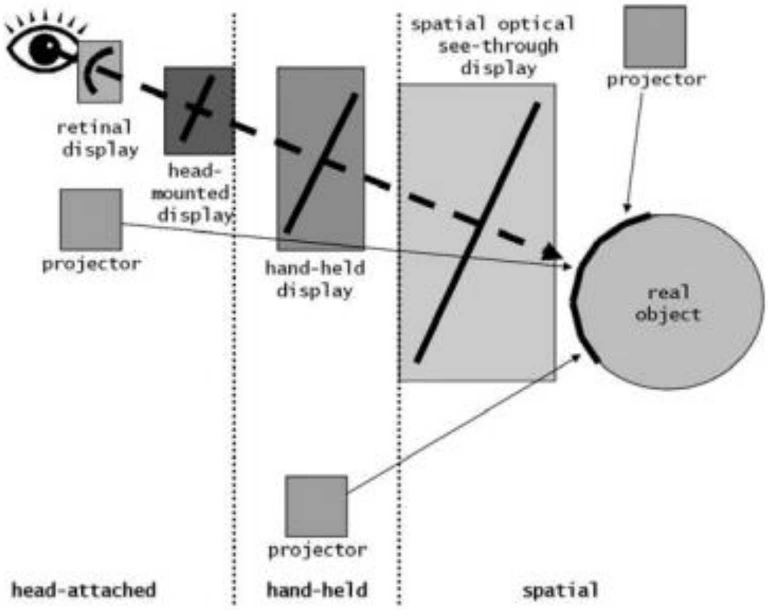
Obrázek 3: Rozdělení AR podle lidských smyslů. Zdroj: [9].

[11] Příklad aplikace Vizuální AR dostupný z: https://www.youtube.com/watch?v=SWtDeeXtMZM.
[12] Příklad aplikace Sluchové AR dostupný z: https://www.youtube.com/watch?v=jEIcNlO75UY.
[13] Příklad aplikace Hmatové AR dostupný z: https://www.youtube.com/watch?v=mBRFc5FmBGg.
[14] Příklad aplikace Čichové a Chuťové AR dostupný z: https://www.youtube.com/watch?v=3GnQE9cCf84.

Jednotlivé udávané typy rozšířených realit by mělo být možné mezi sebou libovolně kombinovat[15] a tím tak vytvořit plnohodnotný vjemový zážitek, který by měl snížit přechod mezi reálným světem a tím rozšířeným.

1.2.2 Rozdělení podle technického vybavení

Nejjednodušší systém vizuální rozšířené reality může být složen ze snímače scény neboli kamery, výpočetní jednotky, libovolné zobrazovací jednotky, případně je doplněný různými podpůrnými senzory a samozřejmě odpovídajícím softwarovým vybavením.



Obrázek 4: Rozdělení AR podle technického vybavení. Zdroj: [7].

Primárním cílem systému AR je vždy poskytnout co nejdokonalejší vizuální zážitek, a proto je následující možná kategorizace podle technického vybavení podávána především z pohledu zobrazovacích jednotek, které jsou specifické pro různé aplikace těchto systémů.

Různé další pohledy na tato rozdělení lze nalézt např. v [6]. Zajímavé je rozdělení s ohledem na pozici zobrazovací/snímací jednotky pozorovatele vůči uživatelskému bodu zájmu udávané v [7] zobrazené na obrázku 4. Každá tato kategorie zahrnuje specifická technologická zařízení, která využívá a je jimi definována.

[15] Z principiálního hlediska je patrné, že bude v majoritních aplikacích kombinována s vizuální AR.

1.2.2.1 Zobrazovací zařízení typu „Head-Attached“

HAD (Head-Attached Displays) kategorie využívá pro vizualizaci AR zařízení, která jsou přichycena na hlavě uživatele tak, že při jeho pohybu včetně pohybu hlavy, je výsledný rozšířený obraz ekvivalentně prezentován vzhledem k jeho očím. V současnosti je nejčastěji ve formě brýlí, či helem, ale z futuristického hlediska miniaturizace hardware (HW) komponent je možné předpokládat inteligentní čočky nebo jiná specifická zařízení.

Na obrázku 5 lze pozorovat inteligentní brýle[16] od předních společností v oblasti informačních a komunikačních technologií Google a Microsoft. Portfolio společnosti Apple zatím takové zařízení neobsahuje.


Obrázek 5: Google Glass Enterprise Edition 1 a Microsoft HoloLens 1. Zdroj: [10] a [11].

Rozšířená realita, postavená na HAD zařízeních, nabízí oproti klasickým mainstreamovým displejům popsaným níže lepší vizuální zážitek v podobě hlubší imerze do vytvořené reality. Hlavní nevýhodou je cenová nedostupnost tohoto řešení.

1.2.2.2 Zobrazovací zařízení typu „Hand-Held“

Z pohledu rozšířené reality se jedná se o uživatelsky nejpoužívanější a nejrozšířenější kategorii z uváděných. Tato kategorie v sobě zahrnuje především mobilní telefony, označované jako smartphony[17], nebo případně i tablety[18].

Mezi výhody tohoto řešení patří cenová dostupnost kompletní technologie a rozvíjející se základna využitelnosti jednotlivých aplikací. Hlavní nevýhodou pro uživatele je nízký stupeň vnoření do rozšířeného světa.

[16] V současné době existuje již druhá generace.
[17] Smartphone je zařízení s pokročilým operačním systémem – s definovaným aplikačním rozhraním a dalšími specifickými funkcemi.
[18] Dnes je většinou jediný rozdíl mezi tabletem a smartphone velikost displeje.

Dále je možné se setkat i s „hybridními“ zařízeními označovanými jako tablet PC[19].

Obrázek 6: AR Stickers na zařízení Google Pixel. Zdroj: [12].

1.2.2.3 Zobrazovací zařízení typu „Spatial“

Rozšířenou realitu lze také samozřejmě vizualizovat i na klasickém monitoru PC nebo displeji notebooku. Z hlediska uživatelů, kteří se mohou zúčastnit projekce AR, by byla tato kategorie značně omezená, a proto je možné zařadit i zařízení, jako jsou dataprojektory nebo TV.

Obrázek 7: Vizualizace AR na klasickém monitoru. Zdroj: [13].

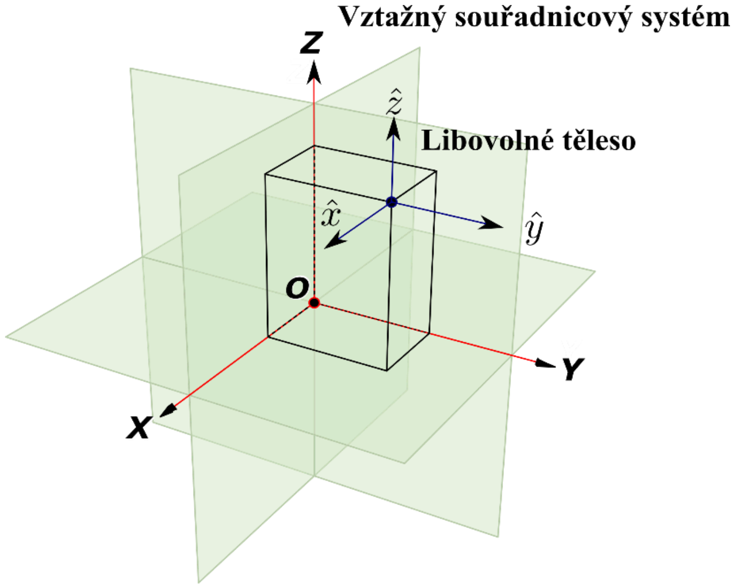
Mezi hlavní výhody těchto zařízení patří jejich cenová dostupnost, rozšířenost u běžných uživatelů a potencionálně[20] kvalitní vizuální prezentace rozšířené reality. Nevýhodou je opět nízké vnoření do vytvořené reality, stejně jako u Hand-Held zařízení, a většinou jsou takové systémy z hlediska vlastní funkčnosti statické k určitému prostředí.

[19] Příkladem je technologie Surface od společnosti Microsoft, která umožňuje zařízení používat jako plnohodnotný notebook nebo tablet.
[20] Systém rozšířené reality je samozřejmě závislý nejen na kvalitě zobrazovací jednotky, ale i na výpočetní jednotce, snímacím zařízení a jiných dalších faktorech.

1.2.3 Rozdělení podle sledovací technologie pro získání polohy a orientace

Primárním úkolem systému vizuální AR je obohacovat realitu o digitální objekty, které je nutné legitimně umístit neboli registrovat jak do venkovních, tak vnitřních prostorů reálného prostředí. Z toho pohledu je tedy nutné se správně orientovat neboli trackovat v pracovním prostoru zařízení neboli vyhledávat „uživatelský nebo přímo systémový" zorný úhel a jeho polohu k poloze a orientaci pozorovaného bodu zájmu.

Poloha a orientace ideálního tuhého tělesa ve třírozměrném prostoru je dána tzv. 6[21] stupni volnosti, kde jednotlivé parametry jsou závislé na předem zvolené parametrizaci, exemplárně třemi souřadnicemi posunu (translace) [x, y, z] tělesa a třemi úhly natočení (rotace) [α, β, γ] vzhledem ke vztažnému souřadnicovému systému soustavy [14]. Na základě takto získaných informací lze vyhledat informace závislé na poloze a směru pohledu uživatele, nebo případně celého systému AR k poloze a orientaci bodu zájmu. Takto získané souřadnice polohy a orientace je nutné dále transformovat[22] do koordinačního souřadnicového systémů zobrazovacího zařízení.



Obrázek 8: Vztažný třírozměrný prostor s libovolným tělesem. Zdroj: vlastní.

Tyto trackovací „sledovací" technologie jsou tedy využívány pro zajištění co nejvěrohodnějšího vnoření uživatele do rozšířené reality – umožněno je i překrývat body zájmu.

[21] Jedná se o minimální počet nezávislých parametrů, které určují jednoznačný stav tělesa v systému.
[22] Mezi základní transformace lze zejména zařadit posunutí, změnu měřítka, otočení nebo zkosení. Dále mezi běžné operace lze zařadit i různé projekce neboli způsoby zobrazení scény.

Pozorovat je možné i systémy, které trackovací technologie využívají minimálně, protože je generovaný objekt rozšířené reality ve výsledné scéně registrován na libovolné nebo předem definované místo, ale vzniká zase otázka, zda se jedná o AR či nikoliv.

Existují různé možnosti, jak trackovací technologie kategorizovat, ale v případě prvotního začlenění do kontextu rozšířené reality je plně dostačující klasifikace podle zdroje [8], kde je uvedeno rozdělení na aktivní, pasivní nebo hybridní trackovací „sledovací" systémy. Do těchto tří kategorií jsou informativně zasazeny nejpoužívanějších senzory, zařízení nebo dokonce plnohodnotné sledovací systémy, které daný systém AR pro orientaci v prostoru může potencionálně využít či dokonce vhodně libovolně kombinovat.

Alternativně lze využít případně rozdělení uváděné v [15], které kategorizuje inerciální senzorové systémy do vlastní skupiny.

1.2.3.1 Aktivní sledovací systémy

Aktivní sledovací systémy, nebo přímo jednotlivé technologie, jsou založeny na existenci předem uměle připravených prostředích. Tato prostředí jsou pro dané specifické využití co nejlépe zkalibrována, aby bylo dosaženo co možná nejpřesnější určení polohy objektu.

Do této skupiny lze zařadit nejrůznější mechanické systémy, kdy je měřena pozice a orientace pomocí mechanického spojení mezi cílovým bodem a bodem počátečním. Dále zde mohou být obsaženy elektromagnetické systémy, které využívají změn polohy snímače uvnitř vytvořeného elektromagnetického pole, akustické (ultrazvukové) systémy využívající přijímače a vysílače ultrazvukových vln nebo případně optické systémy založené na detekci emitovaného elektromagnetického paprsku[23].

Obrázek 9: Příklad aktivních sledovacích systémů. Zdroj: [16].

[23] Primárně se jedná o infračervený paprsek.

V neposlední řadě lze mezi aktivní trackovací systémy zařadit geolokační služby jako americký globální satelitní navigační systém GPS, ruský GLONASS nebo případně evropský systém GALILEO. Dále je možné zařadit navigační systémy v rámci bezdrátových sítí založené na oblíbeném standardu IEEE 802.11[24], alternativně je možné využít i chytré sítě založené na technologiích, jako je např. Bluetooth, ZigBee nebo případně ANT/ANT+. Samozřejmostí je zjišťování polohy v rámci celulárních mobilních sítí.

1.2.3.2 Pasivní sledovací systémy

Na rozdíl od aktivních sledovacích systémů jsou pasivní sledovací systémy založeny na využívání přirozeně[25] se vyskytujících systémů a fyzikálních jevů.

Rozdělení tohoto typu tedy může obsahovat nejrůznější inerciální systémy a senzory jako akcelerometr, gyroskop, magnetometr a jiné, které umožňují za jistých podmínek zjistit správnou orientaci systému v prostoru na základě měření různých fyzikálních veličin. Jako pasivní technologii lze také označit obor počítačové vidění, které může být potencionálně zaměřeno jak na detekci nejrůznějších snadno detekovatelných značek, případně obrazců, tak na rozpoznání různých reálných objektů, dokonce i těch lidských.

Obrázek 10: Příklad pasivních sledovacích systémů. Zdroj: [16].

Počítačové zpracování obrazu jsou sice výpočtově náročné operace, ale vzhledem ke svým přednostem z pohledu jednoduchosti HW řešení, kdy „nejsou“ potřebné další podpůrné systémy jako GPS, jde o velice oblíbenou technologii trackovacího systému.

[24] Známe spíše pod označením Wi-Fi (Wireless Fidelity).

[25] Není tedy zejména nutné budovat satelitní, mobilní, senzorovou síť nebo dokonce speciální „dalo by se říci až studiová“ prostředí.

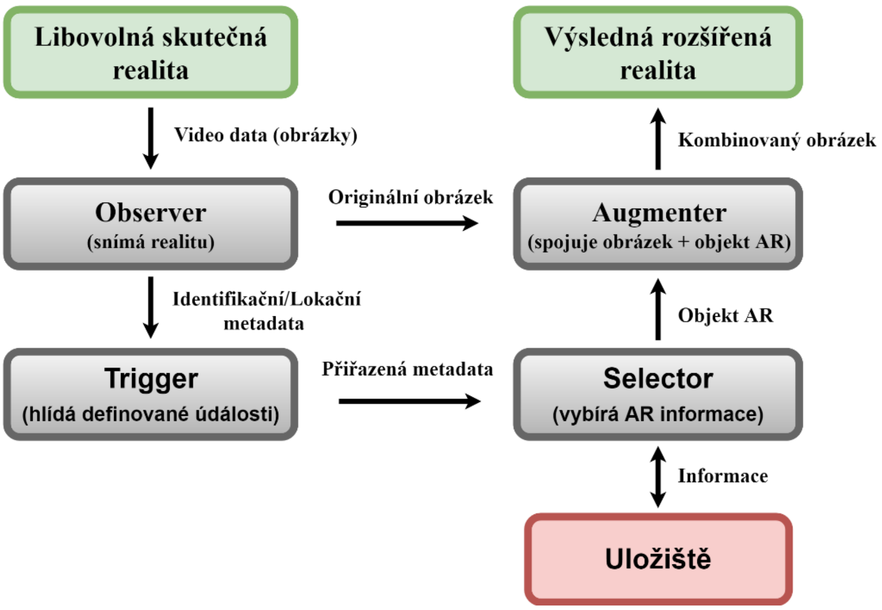
1.2.3.3 Hybridní sledovací systémy

Hybridní sledovací systémy kombinují aktivní a pasivní technologii, u kterých je snaha využít jejich jednotlivé výhody, a především eliminovat nebo případně alespoň kompenzovat jejich jednotlivé více či méně významné nevýhody. V podstatě tato kategorie hybridních systémů není omezena libovolností a počtem kombinací sledovacích systémů, a tak může vzniknout skutečně plnohodnotný systém rozšířené reality.

Příkladem takového systému může být kombinace počítačového vidění a geolokačních služeb případně i inerciálních senzorů, se kterou se lze hojně setkat v tzv. „browserch AR“ neboli prohlížečích rozšířené reality použitých především na mobilních platformách.

1.3 Architektura procesu realizace AR

Nejjednodušší systémy rozšířené reality jsou složeny ze snímače scény, výpočetní jednotky, libovolné zobrazovací jednotky a případně disponují několika podpůrnými inerciálními senzory. Tyto systémy také mohou využívat specializované SW technologie počítačové grafiky, identifikačních/lokačních technik a jiných specifických informačních technologií. Je patrné, že rozšířená realita je tedy kombinací nejrůznějších HW a SW technologií a z tohoto pohledu je nutné zasadit základní koncept tvorby AR neboli tzv. obecnou architekturu procesu, kterou lze najít v [17]. Takto získaný jednoduchý nadhled nad procesem tvorby AR může být využit pro vybudování opravdu komplexního systému rozšířené reality.



Obrázek 11: Architektura procesu AR. Zdroj: vlastní.

Ze senzoru scény neboli kamery jsou v reálném čase získávána video-data, pro které jsou získány rozšiřující informace. V případě využití lokačních technologií jde o metadata pro zjištění polohy a zorného pole systému AR vůči potencionálnímu bodu zájmu. Naopak v případě použití identifikačních technologií jde nejčastěji o metadata získaná z technik zpracování obrazu při detekci uživatelsky definovaných značek, případně i libovolných objektů.

V případě hybridního systému jsou získána kombinovaná metadata. Informace obsažené v metadatech jsou následně zaslány ke kontrole jejich legitimnosti. Z výsledku tohoto rozhodnutí může být vyvolána relevantní akce neboli dochází ke generování objektu rozšířené reality. Následně lze vygenerovaný objekt na základně metadat správně umístit v získaných video-datech scény a opětovně zobrazit na využívané zobrazovací jednotce.

Při procesu realizace rozšířené reality lze tedy pozorovat několik požadovaných podprocesů, které musí zajistit realizovaná aplikace pro její správné fungování. Uživateli musí být výsledky rozšíření prezentovány včasně a především přesně. Pouze takto lze docílit autentičnosti a příjemného uživatelského zážitku.

Tabulka 1: Podprocesy rozšířené reality. Zdroj: vlastní.

Proces č.	Název procesu	Popis procesu
1.	Snímání	Kontinuální získávání video-dat libovolného reálného prostředí skrze snímací jednotku.
2.	Sledování (Tracking)	Nepřetržitá reakce na změnu zorného pole a polohy systému vůči libovolnému objektu zájmu z důvodu zajištění legitimnosti generovaných digitálních objektů s reálným prostředím.
3.	Vkládání (Registrace)	Finální rozšíření získaných video-dat scény o generovaný digitální objekt/objekty v závislosti zjištěných informací z podprocesu sledování.
4.	Zobrazení	Zobrazení výsledné rozšířené reality na libovolné využité zobrazovací jednotce.

1.4 Aplikační domény AR

V dnešní době je rozšířená realita již integrována do všech možných oblastí působnosti, kde plní jednu nebo více specifických funkcí. Rozšířená realita je zajímavá především tím, že se neomezuje pouze na jednu určitou doménu působnosti, ale snaží se proniknout čím dál více do lidského podvědomí, jak je patrné např. z [3] nebo [6].

Na různých příkladech domén AR lze pozorovat její široké uplatněné AR:

- *Business*[26] – Oblast, která by se dala v uvozovkách označit za hnací motor rozšířené reality. Již od počátku je snaha redukovat provozní náklady firem a z tohoto pohledu vznikají zajímavé aplikace řešící efektivní problematiku výrobních procesů.
- *Reklama*[27] – Další významná doména, do kterého se AR snaží integrovat v podobě nejrůznějších kampaní. Zákazníky se snaží nalákat skrze 3D náhledy nabízených produktů, které lze zobrazit přes vlastní mobilní telefon nebo specializované zařízení.
- *Vzdělání*[28] – Vývoj informačních technologií umožňuje vylepšit vzdělávací programy o interaktivní výuku přes názorné vizuálně interaktivní příklady.
- *Navigace*[29] – Rozšířená realita je v tomto oboru hojně využívána, a to v případě automobilových nebo mobilních navigacích. Vznikají aplikace typu „průvodce“, které umožňují lidem přehledně v reálném čase zobrazovat body zájmu, nebo cestovní aplikace, které se zaměřují na zjednodušení jízdy v dopravních prostředcích.
- *Zábava a hry*[30] – Zábavní a herní průmysl je masivním producentem aplikací postavených na AR především díky růstu a dostupnosti mobilních technologií.
- *Architektura a umění*[31] – Neméně významnou oblastí je architektura, kde exemplárními příklady mohou být aplikace na simulaci návrhů exteriérů/interiérů bytových jednotek nebo zahrad. Aplikovatelnost v umění také prochází vývojem – lze dnes pozorovat umělce, kteří takto prezentují svoje díla.

Dalšími typickými oblastmi pro využití je především armáda, lékařství nebo zemědělství.

[26] Příklad aplikace Business AR dostupný z: https://www.youtube.com/watch?v=0m67O1Em7dY.
[27] Příklad aplikace Reklama AR dostupný z: https://www.youtube.com/watch?v=vDNzTasuYEw.
[28] Příklad aplikace AR Vzdělání dostupný z: https://www.youtube.com/watch?v=nkkMNaY0_Qw.
[29] Příklad aplikace Navigace AR dostupný z: https://www.youtube.com/watch?v=hl21nXCDIKE&t=1s.
[30] Příklad aplikace Zábava a hry AR dostupný z: https://www.youtube.com/watch?v=IeHKVC0XLe4.
[31] Příklad aplikace Architektura a umění AR dostupný z: https://www.youtube.com/watch?v=whUg6ozdGlY.

2 TRACKOVACÍ TECHNOLOGIE

Mobilní platforma[32] je specifické zařízení, které se neustále rozvíjí. Především z hlediska cenové dostupnosti a zvyšujícího se výkonu jsou to v dnešní době optimální zařízení pro různou škálu jejich využití. Vzhledem k těmto předpokladům se stávají snadno dostupnými pro běžné uživatele i vývojáře. V případě využití moderních mobilních zařízení jako systému rozšířené reality jsou již minimální HW požadavky splněny, kdy tato zařízení disponují dostatečným výpočetním výkonem, kvalitním integrovaným snímačem scény, displejem s dostatečným rozlišením a relativně přesnými doplňujícími senzory a moduly. V případě využití kamery z protější strany displeje může uživatel pohodlně nahlížet přímo přes displej na reálnou realitu, případně i tu obohacenou.

Z pohledu vývoje vizuální rozšířené reality na mobilní platformě není rozdělení podle technologie pro získání uživatelské/systémové polohy a zorného úhlu v pracovním prostoru dostačují a je dále nutná její rozšiřující rešerše. Tohoto rozšíření je především v kontextu mobilní AR dosaženo v podobě zasazení dvou kategorizací, které se dají při realizaci využít. Z tohoto hlediska vzniká úplně divergentně odlišné rozdělení[33], které se opírá o tzv. Identifikační bezdrátové technologie (označované Identification-based AR, které by mohlo být v případně omezení pouze na technologie počítačové vidění nazýváno jako „Recognition-based AR nebo alternativně Marker AR[34]“) neboli technologie založené na identifikaci předem definovaných objektů v reálném prostředí nejčastěji za pomoci oboru počítačového vidění a Lokační bezdrátové technologie (označované Location-based AR nebo případně jako „Markerless AR[34]“) neboli technologie, které jsou zaměřeny na využití nejrůznějších inerciálních soustav a dostupných geolokačních systémů. S výhodou lze tyto obě představené technologie u mobilní AR kombinovat a může takto vzniknout skutečně reálně využitelný komplexní systém rozšířené reality.

Doplňující informace a různé pohledy, jak dále alternativně rozšířenou realitu rozdělit lze např. vyhledat v [4], [18] a [19].

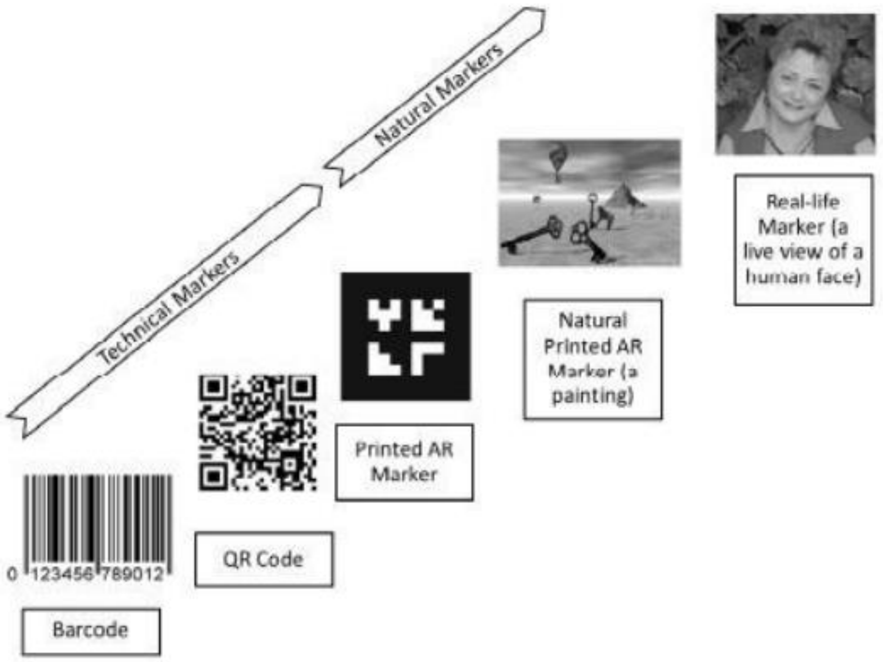
---

[32] Mobilní platforma je pro účely této práce nejen samotný mobilní přístroj neboli smartphone (zařízení s pokročilým operačním systémem s definovaným aplikačním rozhraním a dalšími specifickými funkcemi), ale i vývojovým prostředím a nástroji pro ladění/testování aplikací, prostředím distribuce aplikací, podporou a v neposlední řadě i uživatelskou komunitou a jiné.
[33] Rozdělení na identifikační a lokační bezdrátové technologie není zažité diktum, ale nabízí odlišný pohled na problematiku využití vizuální AR na mobilní platformě.
[34] Toto pojmenování je přinejmenším zavádějící, protože tato kategorizace je v některých pramenech uváděná spíše jako podmnožina kategorie počítačového vidění nebo přímo optických trackovacích „sledovacích“ systémů.

2.1 Identifikační bezdrátové technologie

Identifikační bezdrátové technologie při realizaci požadavků uživatele na získání informací o identitě[35] objektu jsou obvykle v systému rozšířené reality na mobilní platformě založeny na technologii počítačového vidění neboli detekci různých známých detekovatelných objektů[36], dokonce i často standardizovaných přímo přes integrovanou kameru daného zařízení. Legitimně je tak umožněno ze získaných informací interpretovat v reálném čase požadované objekty AR přímo do získaných video-dat, nejčastěji pak přímo před detekovaný prvek nebo dokonce prvky.

Dalšími potencionálními technologiemi, se kterými se lze také setkat a nejsou založené na počítačovém vidění, jsou radiofrekvenční technologie RFID (Radio Frequency Identification) nebo NFC (Near Field Communication) technologie. Tyto záměrně vytvořené technologie pro identifikaci objektů jsou jakousi evolucí[37] čárových kódů a identifikační kategorizaci dále utvářejí a definují.



Obrázek 12: Příklad historického vývoj značek v AR. Zdroj: [4].

[35] Umožňují poskytovat např. informace k aktuální poloze uživatele nebo dokonce identifikovat, o jaký požadavek má uživatel vlastně zájem.
[36] V současné době je detekce směřována a optimalizována k rozpoznávání reálných objektů.
[37] Tyto technologie si z podstaty vývoje nekladou za cíl nahradit čárové kódy, ale spíše je rozšířit o další technologický stupeň. V řadě aplikací je výhodnější použít tyto technologie v jejich vzájemné kombinaci.

V přiložené tabulce 2 lze pozorovat hlavní výhody a nevýhody v případě využití technologií počítačového vidění a rádio frekvenčních identifikačních technologií v systému AR.

Tabulka 2: Výhody a nevýhody identifikačních bezdrátových technologií. Zdroj: vlastní.

Výhody	Nevýhody	
Aplikace nemusí v případě komplexnějšího rozpoznávání být omezena směrem snímání ani max. rozlišovací vzdálenosti od objektu – objekt může být otáčen, pohybován nebo škálován v různých velikostech.	Správná detekce objektů je silně závislá nejen v závislosti úhlu pozorovatele a rozlišovací vzdáleností, ale i na lokálních světelných podmínkách a složitosti jednotlivých objektů.	Počítačové vidění (Computer vision)
Snadné vytvoření vlastní detekovatelné značky včetně její distribuce v interiéru i exteriéru. V případě použití standardizovaných značek, lze jednoduše zajistit přenositelnost informací bez komunikačních kanálů a napájení.	V případě detekce vzorů je nutné udržovat a aktualizovat (neměnný obsah) vytvořenou síť před poškozením a zneužitím (mohou obsahovat škodlivý obsah, jako je přesměrování a jiné útoky).	
Aplikace nepotřebuje pro správnou činnost tzv. přímou viditelnost mezi jednotlivými zařízeními, která jsou specifická používanou rádiovou frekvencí, komunikačními módy, svým pracovním dosahem a např. přenosovou rychlostí.	Účinná komunikační vzdálenost je ovlivněna fyzikálními hledisky (existence parazitních magnetických polí, útlum signálu a jiné).	RFID/NFC
Existence standardizovaných pasivních (zařízení bez přímého zdroje napájení) a aktivních (zařízení s vlastním zdrojem napájení) zařízení.	Pořizovací náklady jsou vždy vyšší než u technologie počítačového vidění, kde lze detekované objekty pouze vytisknout.	

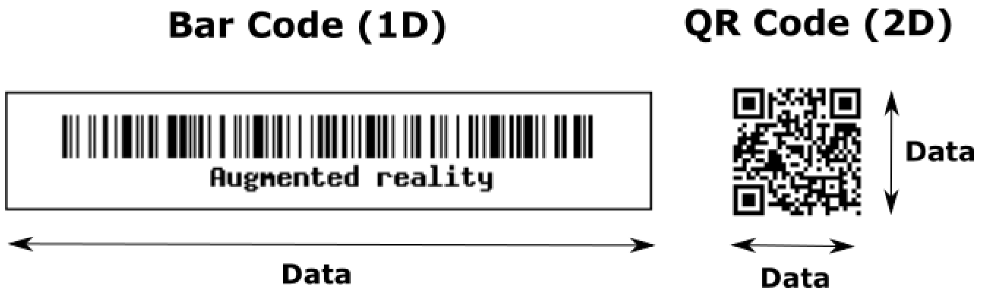
Technologie počítačového vidění (čárové kódy) a technologie radiofrekvenčních štítků (RFID a NFC) mohou být dále disjunktně porovnávány z různých úhlů pohledů zahrnujících datovou kapacitu a např. dekódovací/komunikační vzdálenost, jak je uvedeno ve zdroji [20], který poskytuje zajímavý pohled na čárové kódy pro aplikovatelnost v mobilních zařízeních.

2.1.1 Značky typu „Technical Markers“

Z grafického pohledu nejsou technické značky nic jiného, než jasně viditelné geometrické vzory s definovanou strukturou, které jsou optimalizované pro zakódování co největšího množství dat, snadnou čitelnost s možnou korekcí chyb a v neposlední řadě přenositelnost informací bez příslušných komunikačních kanálů a zdroje napájení. Ke čtení a dekódování uložených informací je nutné použít příslušný detekční snímač.

Takto definované značky umožňují rychlou a přesnou odezvu na získání požadovaných dat, ale nevypadají tak uživatelsky přívětivě jako přírodní objekty a je nutná jejich mechanická údržba.

- *Jednodimenzionální (1D) čárové kódy* – Standardy čárových kódů jsou jedny z nejrozšířenějších uměle vytvořených grafických prvků a lze je považovat za předchůdce moderních značek. V případě standardizace tyto kódy podléhají normám[38], které specifikují jejich základní vlastnosti a pravidla. Technologie jednodimenzionálních čárových kódů je založena na detekci algoritmů zakódovaných do definovaných atomických elementů kódů[39]. Příkladem této symboliky je rozšířený standardizovaný kód pro identifikaci, sběr dat a jejich sdílení EAN (European Article Number)/UPC (Universal Product Code), vytvořený v 70. letech 20. století, definovaný mezinárodní normou ISO/IEC 15420.
- *Dvoudimenzionální (2D) čárové kódy* – Technologická struktura 2D kódu je založena nejčastěji na detekci víceřádkových (stacked) nebo maticových (matrix) algoritmů zakódovaných do definovaných elementů kódu, které dokáží nést poměrně velké množství dat na malém prostoru.



Obrázek 13: Porovnání čárových kódů z pohledu ukládání dat. Zdroj: vlastní.

---

[38] Příslušná norma nebo normy specifikují jednotlivé typy kódů, příslušné metody kódování, formáty symbolů, rozměrové charakteristiky, pravidla pro opravu chyb, referenční dekódovací algoritmus, požadavky na kvalitu produkce a uživatelsky volitelné aplikační parametry.

[39] Elementem kódu jsou nejčastěji v 1D čárových kódech jednotlivé čáry a mezery nebo jejich definovaný sled.

Příkladem maticové symboliky je průmyslově i veřejností oblíbený standard QR (Quick Response) kód, vytvořený japonskou firmou Denso-Wave v roce 1994, definovaný mezinárodní normou ISO/IEC 18004. PDF417 je příkladem víceřádkového kódu definovaného normou ISO/IEC 15438, vytvořeného v 90. letech 20.století.

- *Vícedimenzionální čarové kódy* – Snaha zvýšit informační obsah umožňuje evoluci již zmíněných kategorií čárových kódů přidáním dalšího rozměru. Toho lze dosáhnout intuitivně v podobě barvy[40], kdy jde o tzv. barevné čárové kódy, nebo výšky a podobně. Za pomocí prostorových prvků nebo nerovností (tzv. „bumpy" čárové kódy) je dosaženo diferenční odrazivosti jednotlivých elementů na místo kontrastu mezi různými barvami těchto elementů.
- *Tištěné značky* – Jedná se nejčastěji opět o černobílé obrazce neboli šablony, které jsou jednoduše detekovatelné, ale lze s nimi efektivně pracovat.

Obrázek 14: Příklad tištěné technické značky. Zdroj: [21].

Svým vzhledem se mohou velmi podobat vizuálním vzorům QR kódu, ale většinou jde o nestandardizované, uživatelsky vymyšlené značky.

- *Digitální značky* – Tento typ značek může kombinovat černobíle i reálné objekty, které nejsou z podstaty k dispozici ve fyzickém světě, ale pouze jsou generované na displeji fyzického zobrazovacího zařízení. Z pohledu rozšířené reality jsou ale tyto značky diskutabilní, spíše by měli být zařazeny ke značkám využívaných v mixované realitě.

Technologický rozvoj v oblasti detekovatelných značek je patrný u již zmiňovaných QR kódů, kdy tyto kódy procházejí neustálým vývojem a lze se setkat s minimalizovanou verzí Micro QR kódu, iQR kódem s vyšší kódovací schopností, SQRC kódem vybaveným restrikční funkcí nebo Frame QR kódem zajišťujícím lepší uživatelskou přívětivost z hlediska designu [22].

[40] V některých pramenech nebývají tzv. barevné čárové kódy označované jako vícedimenzionální, ale pouze jako jejich rozšíření z pohledu grafického designu, kdy nenesou žádnou přídavnou informaci.

2.1.2 Značky typu „Natural Markers“

Přírodní značky jsou také ze své podstaty sled geometrických vzorů, ale nemají přesně definovanou strukturu pro optimalizovanou detekci. Na rozdíl od technických značek je detekce takovýchto vzorů z hlediska využitých principů a algoritmů výpočtově náročnější. Snímány jsou specifické rysy objektů, jako jejich významné body, hrany, plochy nebo dokonce i barvy a další jiné parametry.

- *Tištěné značky* – Existují také přírodní tištěné značky, které oproti technickým tištěným značkám vycházejí z přirozených libovolných objektů. Tyto značky jsou jednoznačným technologickým předvojem reálných značek.

  Na obrázku 15 lze pozorovat názorný příklad tištěné značky, která je využita v rámci rozšířené reality v zábavním odvětví.

Obrázek 15: Aplikace Color and Play využívající tištěné značky. Zdroj: [23].

- *Reálné značky* – Pro rozšířenou realitu by bylo velmi limitující, kdyby se omezovala pouze na snímání tištěných grafických značek. Na scénu tedy vystupuje reálné rozpoznávání, které umožňuje snímat přímo libovolné objekty.

2.1.3 RFID/NFC technologie

Jedná se o „novou“ generaci identifikátorů navazující na předcházející technické a přírodní značky, které jsou založeny na radiofrekvenčních identifikačních štítcích s programovatelnými čipy a integrovanou anténou. Obsahem jsou veškeré potřebné informace pro aplikační sběr dat a jejich následné použití všude tam, kde je potřeba co nejrychlejší a nejpřesnější identifikace jednotlivých objektů.

Tento proces je podobný jako u čárových kódů s tím rozdílem, že nepotřebuje přímou viditelnost mezi komunikujícími RFID/NFC zařízeními. Detailnější informace lze dohledat přímo v jednotlivých normách nebo např. ze zdroje [24].

- *RFID* – Souhrnná bezdrátová technologie RFID je definována v nejrůznějších formách aplikačního použití, je předepisována mezinárodními ISO normami. V závislosti využití aktivních/pasivních[41] RFID zařízení komunikují na různých definovaných kmitočtových pásmech, jako jsou 125kHz, 13,56MHz, 960MHz a 2,45GHz. Výsledkem jsou různé technické charakteristiky zařízení a jejich přenosové parametry komunikace včetně využitých komunikačních módů[42]. RFID technologie je určena převážně pro identifikaci osob, zvířat a zboží, zatímco technologie NFC je definována na přenos libovolných dat.

  Z pohledu rozšířené reality realizované na mobilním zařízení je zajímává především technologie NFC, která je v nespočtu zařízení přímo integrovaná a podporována.

- *NFC* – Technologie NFC je tvořena sadou standardů (opírá se o standardy technologie RFID) využívaných pro bezdrátovou výměnu dat mezi dvěma zařízeními na frekvenci 13,56MHz s velmi krátkým dosahem v řádech jednotek centimetrů.

Obrázek 16: NFC zařízení (controller) pracující ve třech módech. Zdroj: [25].

[41] Pasivní RFID čip je vysílací zařízení s přijímačem (transpondér), neobsahující přímý zdroj napájení. Pro svůj chod využívají přijatou energii z iniciátoru komunikace neboli čtecího zařízení. Aktivní transpondér obsahuje vlastní integrovaný zdroj energie (baterii) a je schopen vysílat, aniž by musel být poblíž iniciačního zařízení. Případně lze také zařadit složitější zařízení označované jako semi-pasivní (hybridní), která obsahují zdroj napájení pouze k obsluze připojených periferií a jiným specifickým účelům, ale stále využívá pro vzájemnou komunikaci přijatou energii.

[42] Umožňuje i tzv. hromadné snímání objektů.

Mohou být provozovány pomocí aktivních/pasivních zařízení podobně jako u technologie RFID s různou velikostí kapacity pro uložení dat, maximální komunikační rychlostí v závislosti na použitém režimu komunikace (reader/writer[43], peer-to-peer neboli P2P[44] nebo card emulation[45]) a dalších technických parametrů daných zařízení a komunikace.

Aktivní rozšíření a případnou novou specifikaci NFC technologie zastřešuje nezisková organizace NFC Forum, kde lze také nalézt doplňující informace o NFC [26]. Toto společenství spojuje globálně nespočet industriálních společností, které použitím klíčových prvků ve stávajících a uznávaných normách, jako jsou normy ISO/IEC 18092, ISO/IEC 14443-2,3,4, japonská norma JIS X6319-4 nebo také ISO/IEC 15693, harmonizuje a rozšiřuje stávající bezkontaktní standardy NFC.

Na mobilních telefonech se lze setkat i s kombinovanou formou, kdy je exemplárně NFC použito pro rychlé spárování dvou zařízení a následně je pro přenos využitá optimálnější bezdrátová technologie především z pohledu přenosových vlastností.

2.2 Lokační bezdrátové technologie

Oproti Identifikačním bezdrátovým technologiím jsou Lokační bezdrátové technologie primárně založeny na využívání vestavěných senzorických modulů nebo přímo na geolokačních systémech a službách. Z uživatelského hlediska nabízejí flexibilnější využití, kdy nejsou závislé pouze na snímání předem definovaných objektů. Oproti tomu jsou z takovýchto technologií získávána pouze technická data pro určení pozice, zatímco identifikační technologie mohou aktuální polohu včetně doplňující informace přímo obsahovat.

[43] Reader/writer režim slouží k přenosu nejčastěji nezabezpečených dat mezi iniciující NFC zařízením a NFC transpondéry. Např. telefon s podporou NFC v blízkosti NFC transpondéru dokáže načíst adresu URL (Uniform Resource Locator) a přejít na odpovídající webové stránky.

[44] Peer-to-peer neboli P2P režim podporuje rychlejší přenos dat mezi dvěma aktivními NFC zařízeními. Příklad je synchronizace kontaktů mezi mobilními telefony.

[45] Card emulation režim umožňuje NFC zařízení se chovat jako standardní bezkontaktní čipová karta lze např. provádět platby. Emulace čipových karet obvykle funguje v pasivním režimu se zabezpečeným přenosem a uložením citlivých dat.

Takto definovaná kategorizace není omezena pouze na mainstreamové technologie, které jsou nejčastěji využívány při tvorbě AR na mobilní platformě, ale i na složitější systémy.

Tabulka 3: Výhody a nevýhody lokačních bezdrátových technologií. Zdroj: vlastní.

Výhody	Nevýhody
Vytvořený systém není závislý na vytvořené síti značek počítačového viděné a RFID/NFC zařízení. Uživateli nabízí v jistých směrech příjemnější ovládání – uživatel nemusí hledat a skenovat definované značky.	Funkčnost je závislá na přesnosti použitých senzorů (lze se setkat s tzv. kumulativním procesem hromadění chyb pro, které je nutné provádět nejrůznější korekce, filtrace nebo odhady) a geolokačních technologií.
Vhodnost použití je především v exteriéru z hlediska použití GPS systému, ale lze využít alternativní metody i pro použití v interiéru, jako jsou různé senzorové sítě.	Dostupnost geolokačních služeb (v případě GPS musí být zařízení v režimu přímé viditelnosti se satelity navigačního systému a obvykle je nutná internetová konektivita).

2.2.1 Potencionálně využitelné systémy

Kategorie potencionálně využitelných systémů umožňuje případně zařadit složitější technologické systémy, které jsou z pohledu mobilního vývoje AR pro běžné mainstreamové využití nevhodné již z podstaty funkce, cenové dostupnosti či samotného realizování[46].

- *Mechanické snímaní* – Získání pozice a orientace v mechanickém sledovacím systému je realizována pomocí mechanického spojení s potřebnou snímací elektronikou (přenos informací lze provádět i bezdrátovou formou) mezi cílovým a referenčním bodem.
- *Elektromagnetické snímání* – Správná detekce pozice a orientace je založena na změnách polohy snímací jednotky uvnitř uměle vytvořeného elektromagnetického pole.
- *Akustické (ultrazvukové) snímání* – Sledovací systém tohoto typu je nejčastěji založen na změně vlastností vysílačem generovaných ultrazvukových vln detekovaných na soustavě přijímačů neboli mikrofonů.

[46] Některé potencionálně využitelné systémy nacházejí využití spíše v laboratorních podmínkách a speciálních ateliérech (především pro vývoj počítačových her nebo filmů). Tyto techniky jsou označované jako Motion capture.

- *Optické snímání* – Detekce polohy a orientace je v optickém sledovacím systému prováděna skrze emitovaný elektromagnetický paprsek zachycený na detekčním zařízení, které pracuje ve frekvenčních pásmech, jako je infračervená nebo ultrafialová.

Tyto systémy mohou být velmi přesné, ale bohužel trpí různými omezeními technického nebo fyzikálního rázu. Detailnější informace lze např. vyhledat v [27] nebo [28].

2.2.2 Inerciální systémy

Inerciální sledovací systémy pro mobilní platformy jsou složeny z lehkých, kompaktních a v poměru výkon/cena efektivních senzorů[47] včetně vyhodnocovacích obvodů a logiky, které jsou majoritně vyráběny jako technologie MEMS (Micro Electro Mechanical Systems) případně NEMS (Nano Electro Mechanical Systems). Výstupem mohou být digitální i analogové veličiny, které lze měřit s dostatečnou přesností, a především rychlou odezvou na změny, a to umožňuje využití těchto senzorů pro shromažďování údajů o pohybu nositele.

Existuje několik druhů akcelerometrů, gyroskopů a magnetometrů od různých výrobců, které jsou determinovány technickými vlastnostmi a s tím spojenou cenou. Především cena určuje vhodnost jejich využití. Detailnější informace lze nalézt v [29] nebo přímo u výrobců[48].

- *Akcelometr* – Senzor pro měření dynamického nebo statického zrychlení. Využívají se i pro účely měření setrvačných/odstředivých sil, určování pozice tělesa včetně naklonění a detekce vibrací.
- *Gyroskop* – Umožňuje determinovat úhlovou rychlost, to znamená údaj o tom, jak se měřený objekt rychle otáčí nebo přímo úhel mezi aktuální a výchozí pozicí.
- *Magnetometr* – Senzor, který umožňuje měřit intenzitu i směr magnetického pole. Jedna z efektivních aplikací magnetického senzoru je využití jako kompasu.

Trackovací systémy založené pouze na inerciálních MEMS senzorech ale bohužel v mobilních zařízeních nedosahují takové přesnosti[49] a rozlišení jako běžné konstrukce známých typů akcelerometrů, gyroskopů a magnetometrů.

[47] V případě akcelerometru, gyroskopu nebo magnetometru se jedná o samostatný modul nebo případnou součást inerciální jednotky mobilního zařízení.
[48] Zejména se jedná o společnosti, jako jsou STMicroelectronic, Analog Devices a mnohé další.
[49] U inerciálních systémů je nutné provádět nejrůznější automatické/manuální korekce, filtrace nebo odhady (lze především pozorovat kumulativní proces hromadění chyb) pro správnou lokalizaci, protože měření veličin je vždy zatíženo nějakou chybou.

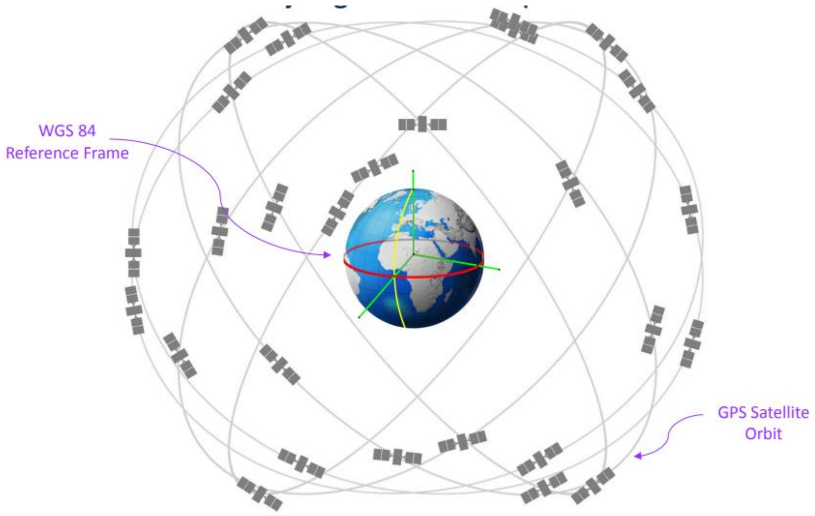
Tato degradace může být např. způsobena konstrukčními nepřesnostmi při výrobě jednotlivých součástek nebo celého MEMS modulu. Dále na senzory působí různé další parazitní vlivy, které mohou vést ke zvýšení šumu na výstupu nebo dokonce nestability celého zařízení.

V současnosti se pro měření pohybu používají nejčastěji tzv. inerciální jednotky neboli IMU (Inertial Measurement Unit), které jsou např. složené z 3-osého akcelerometru, z 3-osého gyroskopu a 3-osého magnetometru a umožňují tak komplexní měření [30].

2.2.3 Geolokační systémy a služby

Geolokační systémy a služby představují další způsob určení polohy zařízení/uživatele. Na rozdíl od inerciálních sledovacích systémů poskytují přímou informaci o poloze v podobě geografických souřadnic[50], případně lze získat i doplňková data. Nejznámější zástupcem geolokačních systémů je americký GPS.

- *GPS* – Základní navigační systém využívaný pro zjišťování polohy s přesností centimetrů až metrů, založený na vybudované satelitní síti.



Obrázek 17: GPS. Zdroj: [31].

Přijímač sleduje vždy minimálně čtyři satelity, ze kterých jsou získány potřebné informace. Pro svoji funkčnost GPS vyžaduje tzv. „přímou viditelnost“ označovanou jako LOS (Line Of Sight)[51] mezi přijímačem a navigačními satelity.

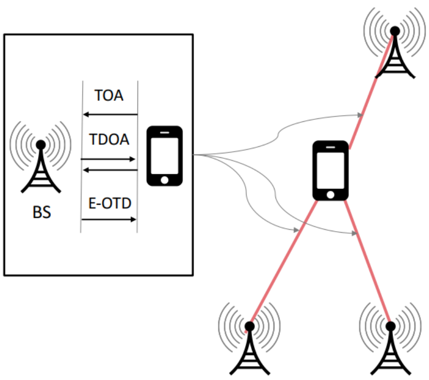
---

[50] Tyto souřadnice jsou definovány vůči vhodně zvolenému souřadnicovému systému, jako je geodetický systém WGS-84, S-JTSK a jiné.
[51] Opakem je tzv. „nepřímá viditelnost“ NLOS (Non Line Of Sigh).

V případě využití technologie A-GPS (Assisted GPS) je proces inicializace prvotního zaměření urychlen s využitím informací o stavu navigačního systému z asistenčního centra [31].

Jistými stávajícími nebo budoucími alternativami k americkému GPS už jsou GLONASS (Rusko), Galileo (Evropa) nebo Beidou (Čínská lidová republika).

- *Mobilní síť* – Zjišťování přibližné polohy zařízení je dosaženo prostřednictvím mobilní celulární sítě[52], která vychází z informací získaných ze známé polohy[53] základnové (přístupové) stanice – technicky se může jednat o zaměření jedné nebo více stanic. Více informací o jednotlivých typech sítí lze vyhledat, popřípadě v [32] a doplňující informace k jednotlivým využitým metodám lze vyhledat v [33].



Obrázek 18: Zjišťování polohy v rámci celulární sítě. Zdroj: [32].

- *Internetová síť* – Existuje řada geolokačních metod využívaných pro stanovené geografické polohy uživatelské stanice v rámci internetové sítě. Tyto metody lze rozdělit do dvou základních skupin na pasivní metody[54] a aktivní metody[55]. Pasivní geolokace umožňuje např. využívat IP (Internet Protocol) adresu, DNS (Domain Name System) nebo informace z Wi-Fi sítí. Naopak aktivní metody měří nejčastěji zpoždění a zkoumá se cesta přenášených dat. Detailnější pohled poskytuje zdroj [34].

Geolokační systém lze za jistých předpokladů provozovat i na bezdrátových technologiích, jako je Bluetooth, ZigBee, ANT/ANT+ nebo dalších jiných.

---

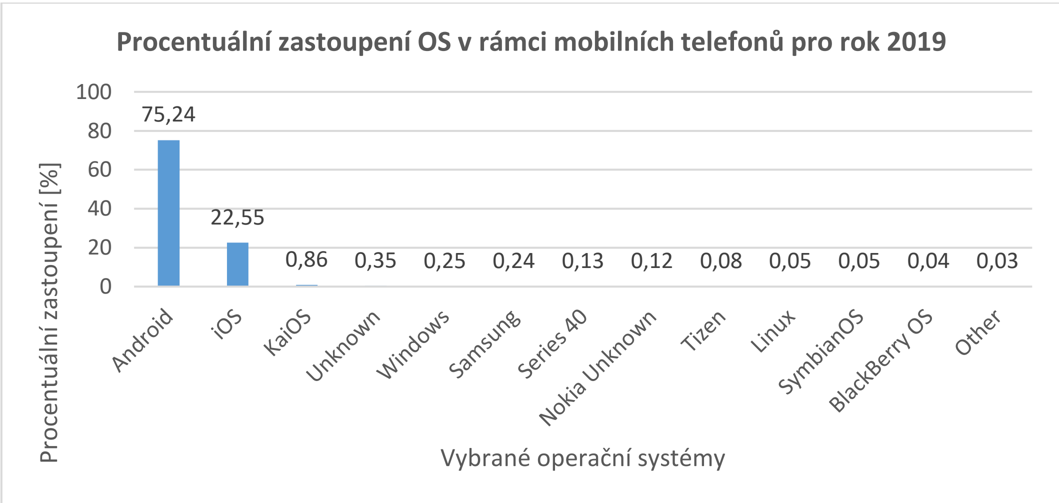
[52] Mezi potencionálně nejznámější sítě lze zařadit 2G sít GSM (Global System for Mobile Communications), 3G sítě UMTS (Universal Mobile Telecommunications System) a LTE (Long Term Evolution) nebo popřípadě 4G síť (LTE-Advanced).
[53] Mezi nejznámější lze zařadit COO (Cell Of Origin) nebo TOA (Time Of Arrival).
[54] Lokalizační informace jsou získávány na základě dostupných informací o síťových zařízení.
[55] Predikce polohy stanice je založena na základě informací z datového přenosu.

3 MOBILNÍ VÝVOJ APLIKACÍ

V současné době je v mobilních zařízeních používáno několik různých operačních systémů[56] v závislosti na výrobci zařízení, které jsou diferenčním procentuálním způsobem zastoupeny na trhu a mohou být pozorovány např. v průzkumech společnosti StatCounter[57]. Z pohledu rozšířenosti a možné budoucí oblíbenosti budou v následujících kapitolách rámcově[58] popsány operační systémy Android a iOS z pohledu historie, základních rysů a možnosti vývoje nativních aplikací včetně jejich distribuce (vycházeno je z průzkumu viz obrázek 19). V neposlední řadě budou popsány možné přístupy mobilního vývoje aplikací a možnosti mobilní architektury aplikací.

3.1 Mobilní platforma

Hlavní charakteristikou mobilního zařízení není pouze technické vybavení, ale i jeho operační systém (dále také označováno jako OS). Tyto systémy jsou specifické svojí HW a SW architekturou, vývojovým prostředím a nástroji pro ladění/testování aplikací, prostředím distribuce aplikací, podporou a také uživatelskou komunitou. Majoritně také bývají úzce spjaté s hardwarovým vybavením, na kterém fungují. Z tohoto důvodu výrobci primárně dodávají svoje zařízení pouze s jedním OS, což umožňuje logickou disjunktní kategorizaci těchto zařízení.



Obrázek 19: Zastoupení OS v rámci mobilních telefonů pro rok 2019. Zdroj: [35].

[56] Souhrn SW vybavení zařízení, které vytváří rozhraní mezi aplikačními programy a hardwarovým vybavením.
[57] Globální statistiky StatCounter jsou založeny na počtu zobrazených stránek webových prohlížečů a umožnují znázornit různé statistiky i časových horizontech.
[58] Detailnější informace lze dohledat přímo na oficiálních stránkách jednotlivých platforem.

Na obrázku 19 je celosvětová statistika pro rok 2019[59], která platí pouze pro klasická mobilní zařízení označovaná jako smartphone, nezapočítávají se tablety, hodinky a jiná specifická zařízení.

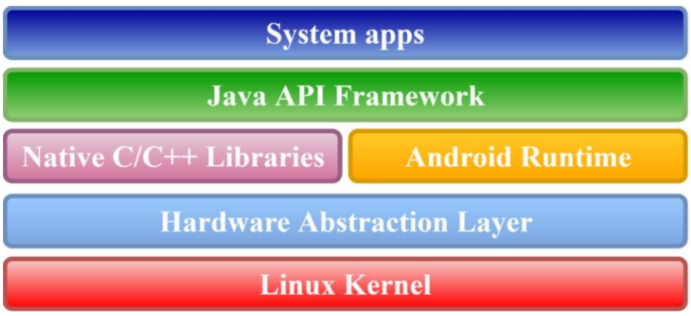
3.1.1 Android

Operační systém Android je open-source[60] operační systém, který v současnosti zastřešuje společnost Google Inc[61] a dále se také stará o její šíření a neustálý rozvoj. V současnosti je aktuálně vydaná verze 9 Pie (nově uvolněna verze Android Q Beta).

V roce 2007 vzniklo konsorcium technologických společností Open Handset Alliance (OHA) vedené společností Google, které v současnosti obsahuje 84 firem. Hlavním společným cílem bylo vytvoření platformy Android jako open-source operačního systému neboli vytvořit platformu, která bude otevřena komukoliv, a urychlení inovací v mobilním světě.

Komerčně byla tato platforma představena v roce 2008 v rámci prvního mobilní zařízení s Android OS ve verzi 1.0 T-Mobile G1 (HTC Dream). Vydáno bylo také oficiální Android 1.0 SDK (Software Development Kit). Zajímavostí je od verze 1.5 (Cupcake) pojmenovávání nových významnějších verzí po sladkostech, dle vzestupného abecedního pořadí.

Z pohledu celosvětové rozšířenosti, oblíbenosti a počtu vydaných aplikací na oficiálním distribučním kanále jednotlivých platforem má na trhu tato platforma konkurenci pouze v podobě mobilní platformy iOS od společnosti Apple, přestože se jako potencionální[62] konkurent jevila i platforma Windows 10 „Mobile“ od společnosti Microsoft.



Obrázek 20: Architektura platformy Android. Zdroj: [36].

[59] Získaná statistika je tvořena daty do července 2019.
[60] Zdrojové kódy jsou primárně poskytovány pod licencí Apache 2.0.
[61] Vyvíjen byl na počátku společností Android Inc, kterou v roce 2005 převzal Google Inc.
[62] Windows 10 Mobile je proprietární operační systém firmy Microsoft, který navazuje na systémy Windows Mobile a Windows Phone. Představen byl v roce 2015 a jedná se již o několikátý pokus o návrat mezi elitu mezi mobilními platformami z pohledu rozšířenosti mezi uživateli, která je neúspěšná.

Architektura platformy Android může být rozdělena do 5 vrstev:

- *Linux Kernel* – Základem nejnižší vrstvy architektury je modifikované[63] linuxové jádro, které poskytuje určitou úroveň abstrakce mezi HW a vyššími vrstvami OS.

  Zajišťuje základní funkce systému, jako řízení procesů, správu paměti, bezpečnost, spravuje systémové ovladače zařízení a další.

- *Hardware Abstraction Layer* – Tato vrstva definuje standardní rozhrání pro přístup z vyšších vrstev systému k samotným HW komponentám daného zařízení. Nemusí tak být implementované specifické přístupy pro ovládání různých HW komponent.

- *Libraries* – Tato vrstva obsahuje nativní knihovny napsané v C/C++, které jsou využívány různými komponentami OS nebo je funkcionalita některých knihoven také přístupná prostřednictvím Java framework API (Application Programming Interface). Obsahují knihovnu pro práci s grafikou OpenGL ES nebo knihovnu pro ukládání a práci s daty SQLite a další jiné.

- *Android Runtime* – Obsahem této vrstvy je optimalizovaný virtuální stroj[64] pro běh na mobilní platformě, který vytváří běhové prostředí pro Android aplikace. Dále jsou obsaženy základní Java knihovny, které se svým obsahem blíží Java Standard Edition.

- *Java API Framework* – Základní vrstva pro vývojáře, která zpřístupňuje funkcionalitu systému přes poskytnuté API v programovacím jazyku Java.

- *Systém Apps* – Poslední vrstva obsahuje přímo využitelné[65] aplikace, které jsou již standardně předinstalované nebo je lze případně doinstalovat z distribučního kanálu.

Pro snazší vývoj aplikací Google představil i oficiální vývojové prostředí Android Studio, které je poskytováno zdarma včetně všech potřebných nástrojů. Primárním programovacím jazykem je Java v kombinaci s „Android“ XML (Extensible Markup Language). Distribuční kanál je Google Play, kde je nutné zaregistrovat vývojářský účet zpoplatněný jednorázovou částkou 25 dolarů [36].

[63] Jádro je přizpůsobeno možnostem mobilní platformy.
[64] Do verze 4.4 KitKat je využit virtuální stroj Dalvik, od této verze je využit ART (Android Runtime), které se zásadně liší především z výkonnostního hlediska. Na rozdíl od JVM (Java Virtual Machine) nepracují tyto virtuální stroje s Java bytecode, ale s tzv. DEX (Dalvik Executable) soubory.
[65] Funkcionalita těchto aplikací může být využita i jinými aplikacemi (není nutná nadbytečná implementace).

3.1.2 iOS

Jedná se o proprietární mobilní operační systém, který je vyvíjen společností Apple Inc. pouze pro svoje interní zařízení iPhone, iPad a iPod touch. Pro tyto OS poskytuje téměř[66] vždy nejnovější aktualizace systému na rozdíl od platformy Android, kde jsou aktualizace závislé na jednotlivých výrobcích zařízení a vystupuje zde problematika tzv. „roztříštěnosti" platformy Android. V současnosti je aktuální verze tohoto operačního systému iOS 12. Komerčně byl OS představen na začátku roku 2007 společně s prvním mobilním zařízením iPhonem.

Současné pojmenování vzniklo až v roce 2010, dříve byl označován za jakousi variaci Mac OS X optimalizovanou[67] pro mobilní zařízení a známý byl pod označením iPhone OS.

Operační systém se dá rozdělit do čtyř abstraktních vrstev[68], které zajišťují základní funkčnost, a aplikace nekomunikují přímo s HW, ale prostřednictvím definovaných systémových rozhraní.

Obrázek 21: Vrstevnatá architektura iOS. Zdroj: [37].

Jednotlivé vrstvy iOS:

- *Core OS* – Základem nejnižší vrstvy je hybridní XNU („X is Not UNIX") jádro[69], které je přímo zodpovědné za všechny aspekty moderního OS. Obsahem jsou také prostředky pro práci s různým matematickými výpočty, týkající se bezpečnosti nejen dat, ale i přístupu k aplikaci nebo přístupu k externímu příslušenství a další jiné.
- *Core Services* – Vrstva zpřístupňuje základní systémové služby, které jsou poskytovány přímo aplikacím.

[66] Nejnovější systém nepodporuje 32 bitovou architekturu, ale pouze novější 64 bitovou.
[67] Neposkytuje plnou funkcionalitu Mac OS X.
[68] Na jednotlivých vrstvách jsou také definovány klíčové komponenty neboli technologie/frameworky, které když nejsou explicitně využívány přímo, tak mohou být využívány komponentami z jiných vrstev.
[69] Založený je na mikrojádře Mach, komponentách z BSD (Berkeley Software Distribution) a dalších různých technologií a komponentách, které zajišťují základní funkce OS, jako řízení procesů, správu paměti a jiné.

Poskytnuty jsou exaktní mechanismy pro práci s geografickou lokalizací, případně i práce s inerciálními senzory, nebo také definuje jeden ze základních klíčových frameworků Foundation[70] a spoustu dalších frameworků a technologií.

- *Media* – Mediální vrstva zpřístupňuje grafické, zvukové a video technologie, které si kladou za cíl dosažení lepšího multimediálního zážitku na daném zařízení. Obsahem je také klíčový framework ARKit pro realizaci rozšířené reality na platformě iOS.
- *Cocoa Touch (Application)* – Tato nejvyšší „aplikační“ vrstva vytváří rozhraní mezi uživatelem, aplikací a OS. Definuje základní infrastrukturu aplikace z pohledu vzhledu a mechanismů pro řízení a její správu. Mimo to také obsahuje další různé frameworky a technologie, nejznámější a další klíčový framework po ARKit je při realizaci aplikací UIKit[71].

Vývoj iOS aplikací může být postaven na programovacím jazyku Objective-C nebo novějším jazyku Swift. Oficiální vývojářské studio Xcode, které obsahuje všechny potřebné nástroje, je nabízené zdarma pro registrované členy. Pro umístění aplikací na App Store je však nutný individuální vývojářský účet, který je zpoplatněný částkou 99 dolarů za rok členství [37].

3.2 Přístupy mobilního vývoje aplikací

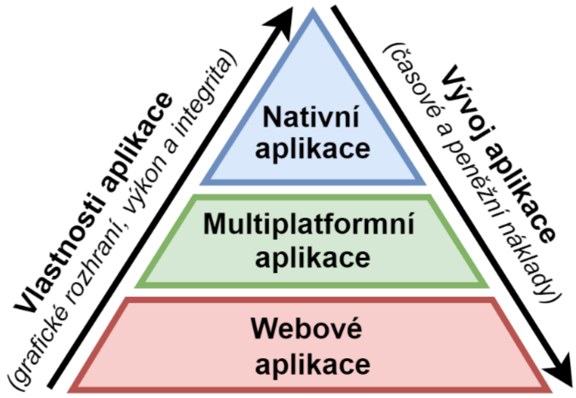
Při vytváření mobilních aplikací je nutné zohlednit, jaký přístup vývoje bude při realizaci využit. Na tyto přístupy lze nahlížet především z pohledu vývojářů nebo manažerů, kteří mají zcela odlišné role v rámci vývoje SW. Zatímco vývojové oddělení má za cíl splnění dané funkcionality projektu, manažerské oddělení zajímají především lidské a finanční zdroje vynaložené na projekt. Proto je nutné tyto aspekty zohlednit a hledat kompromisy, které tak umožňují vzniknout divergentně odlišným technologickým přístupům vývoje SW, které se v různých aspektech mohou zásadně lišit.

„Write once, run anywhere“

Potencionální slogan vystihující multiplatformní[72] vývoj aplikací.

[70] Poskytuje esenciální prostředky neboli infrastrukturu pro tvorbu aplikací – dále definuje datové typy a kolekce nebo poskytuje různé služby spojené s OS, jako je práce s vlákny, se souborovým systémem nebo se sítí.
[71] Poskytuje prostředky pro tvorbu UI (User Interface) nebo dále např. zajišťuje odpovídající mechanismy pro základní řízení a správu aplikace.
[72] Původně byl tento slogan pravděpodobně použit pro Java paradigma neboli zvýraznění benefitů multiplatformnosti této technologie a lze jej přeložit jako („Napiš jednou a spusť kdekoliv“).

Na obrázku 22 lze pozorovat vztah mezi mobilními nativními, multiplatformními a webovými aplikacemi z pohledu uživatelského rozhraní, výkonnosti a integrity, časových a peněžních nákladů na jejich realizaci, testování, nasazení a případnou aktualizaci.



Obrázek 22: Porovnání nativního, multiplatformního a webového vývoje. Zdroj: [38].

K dispozici je spousta technologických možností a neexistuje přímá jednoznačná odpověď na to, kterou technologii vyžít při vývoji SW, protože je nutné zohlednit i další aspekty, jako jsou výhled do budoucna, licenční podmínky, velikost vývojářské komunity a mnohé další.

Z pohledu těchto technologií není důležitá jejich komplexní rešerše a příslušné zařazení, ale spíše vytvoření základního nadhledu nad těmito přístupy mobilního vývoje. Pro takto definované předpoklady je dostačující rozdělení, které poskytuje zdroj [38], [39] případně [40].

3.2.1 Nativní vývoj

Jedná se o přístup založený na podpoře samotných vývojářů mobilních platforem, kterým „nezáleží“[73] na přenositelnosti vyvíjených aplikací. Nativní aplikace jsou nejčastěji realizovány v primárním jazyce za použití oficiálních vývojových prostředí a nástrojů dané platformy. Tyto aplikace jsou optimalizované z hlediska výkonu, grafického rozhraní a přinášejí nejlepší možnosti, jak využívat nejnovějších funkcionalit systému.

[73] Organizace Microsoft např. koupila společnost Xamarin, která se zaměřuje na multiplatformní mobilní vývoj aplikací – umožňuje nativně portovat aplikace na různé platformy (Android, iOS a Windows). Další eventuální možností je využít např. platformu Unity.

Tabulka 4: Výhody a nevýhody nativního vývoje. Zdroj: vlastní.

Výhody	Nevýhody
Výkon aplikací je optimalizován pro konkrétní platformu (aplikace je nainstalována a spouštěna přímo v systému dané platformy).	Nepřenositelnost mezi systémy (nutnost vyvíjet, testovat, aktualizovat nebo nasazovat stejné aplikace pro různé platformy).
Využívání nejnovějších funkcionalit systému skrze poskytované API.	Zdlouhavé testování aplikací přímo na fyzických zařízení nebo simulátoru.
Dodržování doporučených programových a designových návrhových vzorů konkrétní platformy.	Diferenčně odlišné programovací jazyky a doporučené programové a designové techniky na různých platformách.

3.2.2 Webový vývoj

Webový vývoj naopak umožňuje vyvíjet multiplatformní aplikace, které jsou zobrazeny jako optimalizovaná webová stránka v prohlížeči vyvíjená pomocí HTML, CSS a JavaScriptu neboli standardizovaných webových technologií.

Tabulka 5: Výhody a nevýhody webového vývoje. Zdroj: vlastní.

Výhody	Nevýhody
Vyvíjená aplikace je multiplatformní.	Nutná je internetová (datová) konektivita a prohlížeč.
Samotná aplikace je uložena na vzdáleném serveru (centrálně může proběhnout její nasazení, testování nebo aktualizace).	Nekonzistence grafického designu na různých platformách/prohlížečích (chybí nativní grafické UI elementy daných platforem).
Využití standardizovaných web technologií (není nutné znát specializované nativní technologie).	Síťový provoz aplikace (aplikace neustále komunikuje se vzdáleným serverem).
Složité výpočetní operace jsou zpracovávány přímo na vzdáleném serveru.	Absence využití plné funkcionality systému a omezený přístup k datům. (nepřístupnost k API).

Tyto technologie odstraňují nejslabší stránku nativního vývoje, a to multiplatformní aspekt, ale za cenu ztráty přístupu k nižším vrstvám OS. Funkce, jako jsou notifikace, ukládání dat do databáze a jiné, nejsou k dispozici.

Aplikace je uložena na vzdáleném serveru a s tím plyne nutný požadavek na internetové připojení[74]. Představitelem webového vývoje může být framework JQuery Mobile.

3.2.3 Hybridní vývoj

Aplikace označované jako hybridní využívají většinou kombinaci webového a nativního přístupu. Tyto technologie fungují na principu získání instance web kontejneru neboli prohlížeče, případně jsou obaleny tzv. wrapperem. Tento kontejner pak zobrazí optimalizovanou mobilní aplikaci, která je přímo nainstalována v daném zařízení.

Tabulka 6: Výhody a nevýhody hybridního vývoje. Zdroj: vlastní.

Výhody	Nevýhody
Vyvíjená aplikace je opět multiplatformní, stejně jako v případě webového vývoje.	Aplikace je více HW náročná než v případě nativního vývoje.
Využití standardizovaných web technologií (není nutné znát specializované nativní technologie).	Nekonzistence grafického designu (chybí nativní grafické UI elementy daných platforem).
Funkcionalita systému je poskytována např. přes tzv. pluginy neboli Javascriptové API.	Pravděpodobná nedostupnost nejnovějších nebo nestandardních API.
Není nutná internetová (datová) konektivita (aplikace je fyzicky nainstalována).	Omezenější možnosti při zpracování audia, videa a 2D/3D grafiky a dalších oblastech.

Odstraňuje opět multiplatformní aspekt a značně také kompenzuje i nevýhodu webového vývoje, který neumožňuje využít plnou funkcionalitu systému. Tato funkčnost je v hybridním vývoji nejčastěji poskytována přes tzv. pluginy neboli Javascriptové specializované API. Typický představitel je framework Adobe PhoneGap/Cordova.

[74] Některé aplikace mohou pracovat off-line v případě využití cache prohlížeče.

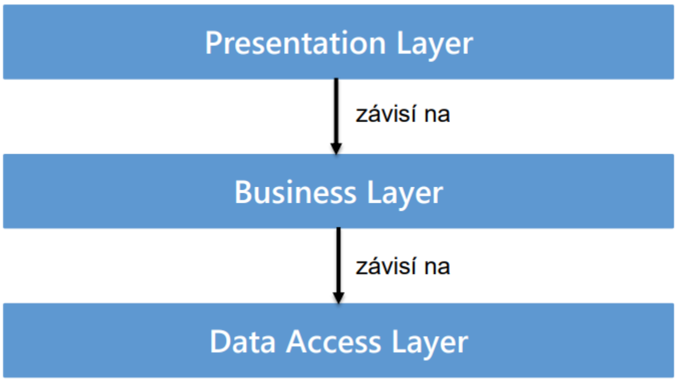
3.3 Architektura mobilních aplikací

Tato kapitola si klade za cíl představit obecné varianty nejčastěji využívaných návrhových architektonických vzorů 3vrstvé architektury, použité nejen při vývoji mobilních aplikací.

„Snaha o zefektivnění vývoje programů a „nevymýšlení“ dříve vymyšleného jde ještě dále. V posledních letech jsou stále populárnější tzv. návrhové vzory (anglicky design patterns), které bychom mohli charakterizovat jako návody na řešení některých typických, často se vyskytujících úloh.“

Charakteristika návrhových vzorů uváděná v literatuře [41]

Aplikován je koncept rozdělení odlišné funkcionality do logických separátních komponent neboli tzv. vrstev, který umožňuje lepší škálovatelnost[75], přehlednost[76] a znuvupoužitelnost a v neposlední řadě testovatelnost kódu aplikace. Na různých platformách lze pozorovat více či méně realizovaný mechanismus pro dekompozici zodpovědností funkcionality aplikace. Krajním případem může být delegování chování do jedné třídy, která obstarává veškeré dění v aplikaci. Dochází tak k míchání datové[77], business[78] a prezentační[79] vrstvy.



Obrázek 23: Vrstevnatý model architektury. Zdroj: vlastní.

Na první pohled by se mohlo zdát, že jsou tyto návrhové vzory pro libovolnou platformu unikátní, ale na různých platformách mohou být vysvětlovány divergentně, přestože principiálně vždy vychází ze stejného konceptu daného architektonického vzoru.

[75] Umožňuje rozšiřitelnost/udržovatelnost aplikace bez značného přepisování kódu, za určitých předpokladů umožňuje nahrazení celých vrstev.
[76] Problematika tzv. „špagetového kódu“.
[77] Reprezentuje doménová data, udržuje jejich stav a obstarává business logiku aplikace.
[78] Reprezentuje přenosovou vrstvu mezi prezentační a datovou vrstvou, a to včetně začlenění aplikační logiky.
[79] Reprezentuje uživatelské rozhraní, které je odpovědné za sběr a zobrazení doménových dat.

Detailnější rozbor jednotlivých obecných vzorů nabízí případně zdroj [42].

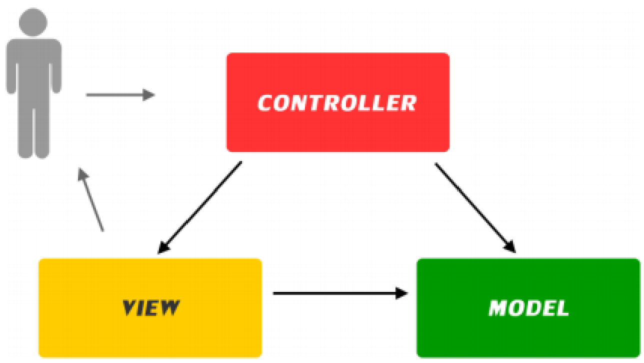
Otázkou zůstává, zda se jedná o architekturu, návrhový vzor, koncept nebo obecný přístup pro vývoj aplikací, protože není přesným dogmatem, jak mají tyto architektonické vzory na různých platformách fungovat.

3.3.1 MVC

MVC (Model-View-Controller) je tzv. „klasická“ 3vrstvá architektura, známá především díky nasazení při vývoji webových aplikací, kterou lze aplikovat i při mobilním vývoji. Jednotlivé komponenty mezi sebou spolupracují, každá má v rámci aplikace jinou úlohu, ale měly by být na sobě nezávislé.

Komponenty MVC architektury:

- *Model*[80] *(datová vrstva)* – Spravuje business logiku aplikace a doménová data. Zajišťuje tedy přístup k datům a operace s daty jako je načítaní, ukládání a zpracování.
- *View*[81] *(prezentační vrstva)* – Reprezentuje uživatelské rozhraní, které interpretuje data zastoupená Modelem.
- *Controller (aplikační vrstva)* – Funguje jako částečný nebo úplný[82] koordinátor mezi Modelem a View. Zpracovává požadavky od uživatele, zabezpečuje změny v Modelu nebo View.



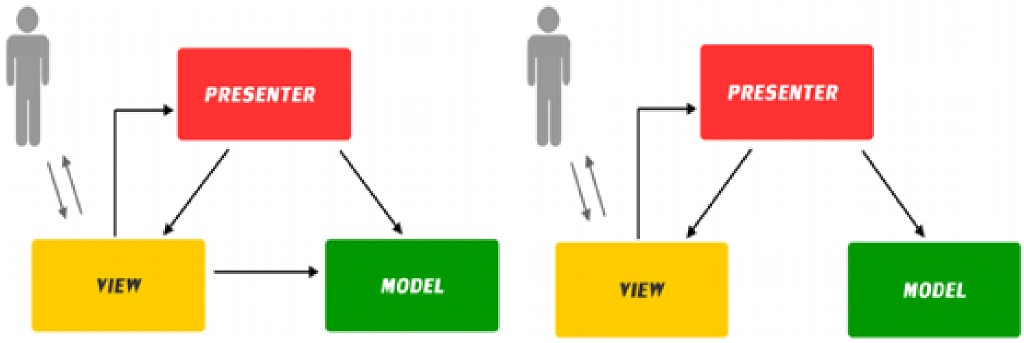
Obrázek 24: Architektura MVC. Zdroj: [42].

[80] Model je při návrhu aplikace chápán jako celistvá komponenta, při realizaci aplikace je tvořen množinou tříd, případně funkcí.
[81] Vlastnosti komponenty View jsou závislé na konkrétní aplikaci, může např. ošetřovat uživatelský vstup (tato vlastnost je typická u desktopových „widgetových“ aplikací). Jinak tuto funkcionalitu zajišťuje controller.
[82] Při určité variantě MVC předává View informaci o změně stavu modelu Controller. V obecném variantě je View komponenta notifikována při změně stavu Modelu přímo Modelem.

Všechny další popisované architektonické vzory definují objekty View a Model podobně, liší se ve vlastnostech třetí komponenty a dále ve vzájemném propojení vazeb mezi jednotlivými komponentami.

3.3.2 MVP

MVP (Model-View-Presenter) vychází z architektury MVC a snahou je kompletní oddělení View a Modelu pomocí vrstvy označované jako Presenter, která na rozdíl od Controlleru přebírá větší míru zodpovědnosti. View předává informaci o uživatelskému vstupu přímo Presenteru, který pak deleguje příslušnou akci přímo na Model. Takovéto pojetí architektury je typické pro desktopové aplikace, kde uživatelský vstup zpracovává přímo konkrétní View.



Obrázek 25: Architektura MVP s vazbou a bez vazby mezi View a Modelem . Zdroj: [42].

Specifickou variantou MVP architektury je koncept, kdy není vytvořena vazba mezi View a Modelem. V tomto případě veškerou komunikaci deleguje Presenter. Stejný koncept může být aplikován i u MVC.

Další alternativou k MVC a MVP architektuře je případně MVVM (Model-View-ViewModel), architektura známá z technologií WPF (Windows Presentation Foundation) a Silverlight, která je založená na mechanismu tzv. data bindingu. ViewModel je abstrakcí uživatelského rozhraní, který zpřístupňuje data Modelu a stav specifický pro View. Rozdíl je ve směru vazby, kde Presenter má přímo referenci na View, zatímco u MVVM má View referenci na ViewModel.

4 ANALÝZA A NÁVRH APLIKACE

Analýza je úvodním zamyšlením nad cílem diplomové práce neboli definuje požadavky na praktickou část, vycházející ze zadání diplomové práce. Dále je odůvodněn výběr přístupu vývoje mobilní aplikace včetně výběru podporované platformy, dále specifikuje architekturu aplikace a všechny další potřebné technologie, chování aplikace a uživatelského rozhraní. V neposlední řadě je nastíněna problematika rozšířené reality z aplikační domény.

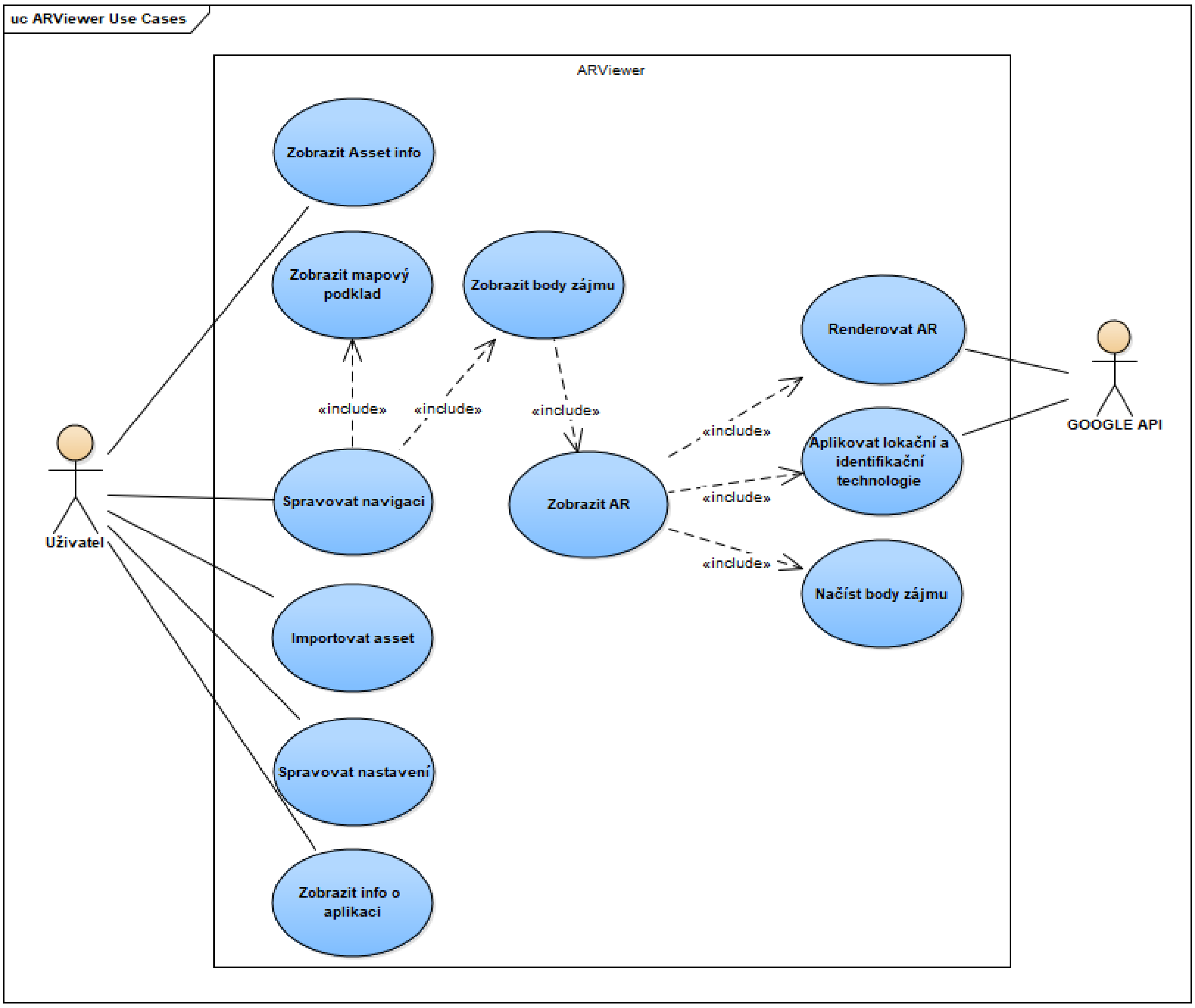
4.1 Požadavky na mobilní aplikaci

Navrhovaná aplikace by měla obsahovat následující klíčové funkcionality shrnuté v několika fundamentálních bodech:

1. *Rozšířená realita*
    - Aplikace bude využívat lokační bezdrátové technologie.
    - Aplikace bude využívat identifikační bezdrátové technologie.
2. *Zobrazení v reálné scéně*
    - Aplikace bude zobrazovat navigaci k jednotlivým bodům zájmu z definované množiny bodů zájmu a poskytne požadované doplňující informace v rámci zvolené vzdálenosti (10 metrů).
3. *Zobrazení na mapě*
    - Aplikace bude zobrazovat aktuální pozici uživatele.
    - Aplikace bude zobrazovat polohu bodu zájmu nebo vybrané množiny bodů zájmu.
4. *Body zájmu*
    - Aplikace bude umožňovat import bodů zájmu ze vzdáleného serveru ve vhodném formátu.
    - Aplikace bude umožňovat uložit body zájmu včetně případných grafických podkladů do perzistentního úložiště mobilního zařízení.

Na obrázku 26 lze pozorovat zjednodušený doprovodný[83] diagram případu užití, který mapuje chování aplikace z pohledu účastníků[84] uživatele a Google API. Uživatel je motivován k vykonávání různých funkcí nad aplikací, ale z pohledu diplomové práce je klíčová funkce rozšířené reality.

Rozšířená realita zahrnuje funkce načtení množiny bodů zájmu, získání polohy uživatele a systému a následné renderování bodu zájmu do reálné scény.



Obrázek 26: Zjednodušené zobrazení případů užití. Zdroj: vlastní.

[83] Pro dosažení cílů diplomové práce není stěžejní poskytnout komplexní diagram užití s definovanými scénáři, ale spíše vytvořit nadhled nad vyvíjenou aplikací.
[84] Reprezentuje entitu mimo systém, která se systémem interaguje.

4.2 Výběr přístupu mobilního vývoje a podporovaných platforem

Ze získaných rámcových přehledů základních přístupů mobilního vývoje je možné určit, jaký přístup by měl být využit pro aplikaci s rozšířenou realitou včetně výběru podporované mobilní platformy nebo platforem.

Webový a hybridní multiplatformní přístup (Adobe PhoneGap/Cordova a jiné) byl zamítnut především z hlediska práce s grafickými knihovnami, kde tyto přístupy nejsou tak vhodné jako nativní přístup.

Pro výběr podporované platformy byla zvolena tři hlavní kritéria[85]:

1. *Platforma musí být cenově* dostupná *a rozšířená*[86].
2. *Platforma nesmí být zastaralá a musí mít výhled do budoucnosti*[87].
3. *Platforma se aktivně zajímá o rozvoj rozšířené reality.*

Z tabulky 7 je možné vypozorovat, že všechna stanovená kritéria splňuje platforma Android. Z těchto důvodů byla tato platforma zvolena pro vytvoření aplikace.

Tabulka 7: Kritéria pro výběr platformy. Zdroj: vlastní.

	Platforma		
Výběrová kritéria	Windows	Android	iOS
1.			
2.			
3.			

Platforma Windows pro mobilní zařízení je momentálně na ústupu, ale z pohledu rozšířené reality se zdá být zajímavou především díky zařízením označovaným jako tablet PC, která v sobě integrují plnohodnotný notebook a tablet.

[85] Řádky tabulky 7 odpovídají kritériu stejného čísla, zelená barva znamená splnění, oranžová částečné splnění/nesplnění, červená nesplnění pro jednotlivé platformy uvedené ve sloupcích.
[86] Porovnání jsou prováděna skrze přední české internetové obchody – Alza.cz a CZC.cz.
[87] Porovnání jsou prováděna především z hlediska rozvoje platformy, jako jsou vydané verze OS, vývojové nástroje, velikosti vývojářské komunity a podpory.

Platforma iOS je sice potencionálně zajímavá nejen pro rozšířenou realitu, ale bohužel z pohledu rozšířenosti a dostupnosti zařízení nedokáže konkurovat platformě Android. Na rozdíl od většiny Android zařízení ale mají zařízení iOS plnou podporu[88] aktualizací operačního systému.

4.3 Výběr architektury

Nezávisle na typu aplikace je architektura aplikace stěžejním bodem před samotnou realizací. Snahou vrstevnaté architektury je především, aby dopad realizovaných změn v aplikaci byl na ostatní části co nejmenší – jedná se o problematiku tzv. lokalizace změn[89].

Použití architektonického vzoru je do značné míry dáno platformou, na které je aplikace vyvíjena. Ze získaného přehledu obecných návrhových architektonických vzorů je možné určit, jakou architekturu by měla aplikace realizovat.

Architekturu MVC je obtížné realizovat na většině moderních platforem pro vývoj desktopových nebo mobilních aplikací, protože tok událostí od uživatele je vyvolán na grafických komponentách neboli přímo v daném View.

V těchto případech se používá spíše architektura MVP a z těchto důvodů byla také zvolena pro realizaci aplikace. Poslední architektura je oblíbená při vývoji na platformě WPF, která poskytuje různé ulehčujících mechanismy.

4.4 Výběr technologií pro rozšířenou realitu na platformě Android

Z hlediska vývoje aplikace s rozšířenou realitou je nutné počítat s různými technologiemi, které budou využity a různě kombinovány pro dosažení požadovaných cílů diplomové práce.

U navrhované aplikace se předpokládá práce s 2D případně 3D grafikou pro rozšíření reálné scény o navigaci a doplňující informace k bodům zájmu, perzistentním úložištěm, mapovým podkladem pro zobrazení aktuální polohy uživatele a množiny bodů zájmu. Nedílnou součástí jsou další podpůrné technologie, knihovny, případně frameworky využité pro realizaci aplikace. V poslední řadě rozbor možností testovacího serveru pro aplikační použití.

[88] Výrobci Android zařízení umožňují aktualizovat OS pouze minimálně (obvykle pouze na další novou verzi).
[89] Lze např. přidat nebo změnit grafické komponenty.

4.4.1 ARCore/SceneForm

ARCore je nová platforma od společnosti Google, navazující na projekt Tango[90], pro mobilní AR, která si klade za cíl zjednodušení realizace na systému Android. Detailněji je opět vysvětlena na developerských stránkách [36] a v praktické části diplomové práce.

V současné době je tato platforma pro AR ve verzi 1.10.0[91], kdy je možné ARCore aplikovat pouze na oficiálně podporovaná[92] zařízení.

Obrázek 27: Platforma ARCore. Zdroj: [36].

Využívá tří klíčových technologií pro integraci virtuálního obsahu do reálného světa pomocí:

- *sledování pohybu;*
- *mapování okolního prostředí;*
- *vlivu světelných podmínek.*

Porozuměním a kombinací těchto technologií je umožněno poskytnout co nejdokonalejší vizuální zážitek a vytvořit tak plnohodnotný systém AR pouze přes vestavěné HW komponenty mobilního zařízení. Vkládané objekty by měly být bezproblémově integrovány do skutečného světa. Na libovolnou pozici lze vložit objekt zájmu, který lze prohlížet z různých úhlů a správná pozice je determinována i v případě opuštění a navrácení do pracovního prostoru.

Současné předpoklady pro vývoj ARCore aplikací:

- *Android Studio verze min. 3.1 s Android SDK Platform verze min. 7.0 (API level 24);*
- *fyzické podporované zařízení;*
- *ARCore SDK pro Android.*

[90] ARCore na rozdíl od platformy Tango funguje bez specializovaného HW – dalo by se říci, že je odpovědí na framework ARKit od společnosti Apple, který již rozšířenou realitu na svých zařízení podporuje.
[91] Představeno bylo oficiální SDK včetně dokumentace API pro různá vývojová prostředí.
[92] Zatím se jedná o vybrané produktové řady, jako jsou Google Pixel a některé další. Později by měly být podporované libovolné telefony s operačním systémem Android ve verzi 7 a vyšší.

Pro vývoj aplikace s rozšířenou realitou je dále nutné se seznámit nejen se samotným vývojem pro Android zařízení, ale i s grafickou knihovnou OpenGL, která je v ARCore využita.

Pro realizaci byl zvolen 3D framework SceneForm, který usnadňuje[93] vykreslování 3D scén v rozšířené realitě[94] bez nutnosti znalosti OpenGL.

Základní charakteristika SceneForm:

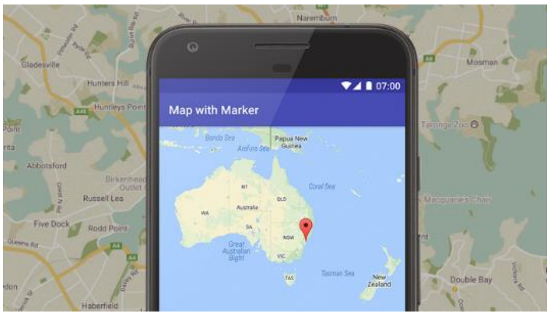
- *pro vykreslení scény je využit koncept grafu[95];*
- *založen je na renderovacím engine Filament;*
- *zjednodušuje práci s 3D objekty poskytuje plugin.*

4.4.2 Realm databáze

Multiplatformní databázová open-source platforma Realm je alternativou k oblíbené databázi SQLite. Realm databáze je MVCC (Multiversion Concurrency Control) mobilní úložiště v souladu s vlastnostmi ACID (Atomic-Consistent-Isolated-Durable) založené na vlastním ukládacím mechanismu – nejedná se pouze o knihovnu nad SQLite databází. Data mohou být vystavena přímo jako live[96] nativní objekty, nad kterými je možné provádět transakce.

4.4.3 Google Maps

Google Maps Android API umožňuje přidat do aplikace mapový podklad, který je založený na Google Maps. API automaticky spravuje přístup k serverům Google Maps, stahuje data, zobrazuje mapu a reaguje na uživatelská gesta.



Obrázek 28: Google Maps. Zdroj: [36].

[93] Většina požadovaných konceptů je již zpracována interně.
[94] Lze využít v aplikacích, které nepracují s AR.
[95] Graf obsahuje uzly, které jsou umístěné ve scéně.
[96] Realm database je reaktivní, to znamená ,že může pracovat s aktuálními daty.

Podporuje také přidání vlastních bodů zájmu neboli značek, křivek, polygonů, různých překryvů, definici vlastních akcí, upravování celkového vzhledu a jiné. Využívání API je podmíněné vygenerováním identifikačního klíče (API Key), který je poskytován zdarma [36].

4.4.4 Výběr knihoven

Tato sekce popisuje oblíbené[97] použité knihovny na platformě Android pro zefektivnění vývoje aplikace a jejich uplatnění v diplomové práci.

Pro vývoj byly vybrány tyto knihovny:

- *Butter Knife*[98] – Knihovna umožňuje zjednodušené provázání UI komponent pomocí anotačního mechanismu.
- *Picasso*[99] – Knihovna poskytuje podporu pro jednoduché stahování a cachování obrázků z definované URL.
- *Retrofit 2*[100] – HTTP klient, který zabezpečuje komunikaci s testovacím serverem.
- *Dagger 2*[101] – Knihovna pro podporu vkládání závislostí – Dependency Injection (DI).
- *Rxjava 2* (Reactive Java)[102] – Knihovna pro podporu reaktivního programování.

4.5 Definice uživatelské rozhraní

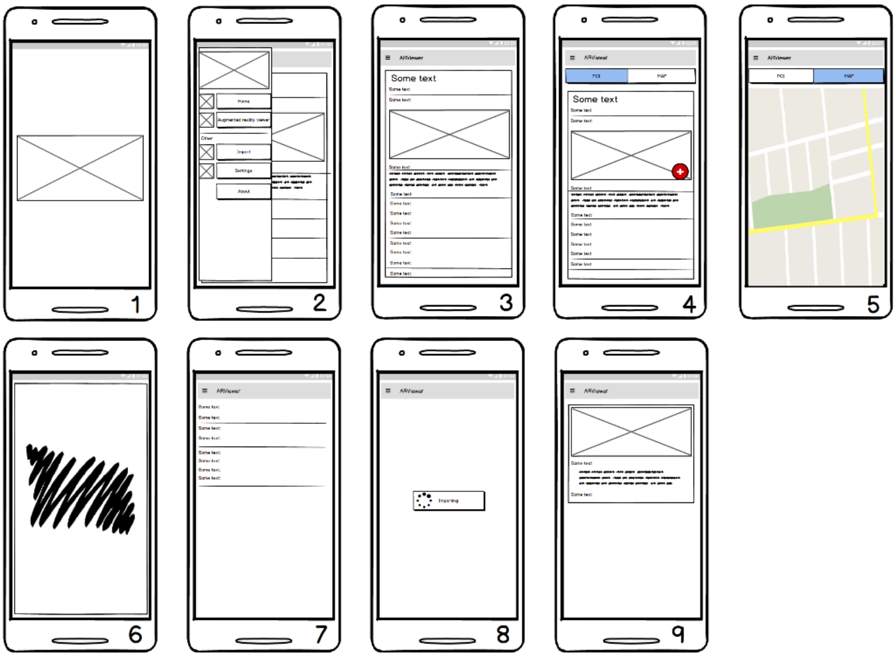
Před zahájením realizace samotné aplikace je vhodné vytvořit wireframy[103], ze kterých je patrné rozložení prvků, způsob průchodu aplikací jednotlivých oken[104] aplikace. Jednotlivé wireframy aplikace byly vytvořeny v programu Balsamiq[105].

Jednotlivé obrazovky budou detailněji vysvětleny v dalších kapitolách týkající se realizace a uživatelské příručky.

[97] Popisovány nejsou defaultní knihovny pro platformu Android.
[98] Github zdroj dostupný z: https://github.com/JakeWharton/butterknife.
[99] Github zdroj dostupný z: https://github.com/square/picasso.
[100] Github zdroj dostupný z: https://square.github.io/retrofit.
[101] Github zdroj dostupný z: https://github.com/google/dagger.
[102] Github zdroj dostupný z: https://github.com/ReactiveX/RxJava.
[103] Jedná se o návrhové modely, které umožňují zobrazit grafický náhled aplikace.
[104] V případě Android aplikací se jedná o layouty Activit případně Fragmentů.
[105] Aplikace je dostupná v Trial verzi z: https://balsamiq.com.

Legenda jednotlivých oken aplikace podle číselného označení v obrázku 29:

1. *úvodní obrazovka;*
2. *obrazovka hlavního menu;*
3. *obrazovka pro informace o assetu;*
4. *obrazovka bodů zájmu (POI);*
5. *obrazovka mapových podkladů (Mapa);*
6. *obrazovka pro rozšířenou realitu;*
7. *obrazovka pro nastavení aplikace;*
8. *obrazovka pro import assetu;*
9. *obrazovka pro informace o aplikaci.*



Obrázek 29: Wireframy jednotlivých oken. Zdroj vlastní.

4.6 Testovací server

Testovací server je vytvořen z důvodu vlastní funkčnosti[106] Android aplikace, kdy by měl umožnit stahovat veškeré informace k bodům zájmu a případné grafické podklady. Získávány budou přes webové služby postavené nad protokolem HTTP.

[106] Pro aplikaci neexistuje plnohodnotný server.

Přenesené informace budou ve formě JSON (JavaScript Object Notation) souboru a mapové podklady ve formátu JPG („Joint Photographic Experts Group“).

Pro realizaci byla zvolena technologie Spring Boot. Jedná se o zjednodušenou vrstvu postavenou nad frameworkem Spring, která umožňuje jednoduchý a rychlý vývoj serveru.

4.7 „Konkurenční“ mobilní aplikace

Oblíbenost rozšířené reality předurčuje tuto technologii k tomu, že vzniká nemalé množství aplikací na rozmanité domény lidské činnosti. Na mobilní platformě se nejčastěji může uživatel setkat s aplikacemi typu browserů. Tyto browsery fungují jako prohlížeče rozšířené reality, které obsahují různé definovatelné vrstvy s libovolně použitelnými informacemi. Lze se setkat i s aplikacemi, kde si uživatel tyto jednotlivé vrstvy může sám definovat a může se tak jednoduše stát tvůrcem.

Mezi nejznámější aplikace patří:

- *Layar*;
- *Wikitude*;
- *Junaio*.

V přiložené tabulce 8 můžeme pozorovat hlavní výhody a nevýhody browserů.

Tabulka 8: Výhody a nevýhody hybridního vývoje. Zdroj: vlastní.

Výhody	**Nevýhody**
Snadné masové použití ve všech oborech působnosti, nejen v oboru cestovního ruchu.	Browsery jsou placené a mezi sebou v majoritních případech nespolupracují (nelze migrovat vytvořené vrstvy).
Snadné tvoření vrstev, usnadnění práce jiným vývojářům a bezplatné a snadné stáhnutí koncovým uživatelem.	Dříve se browsery zaměřily výhradně na Location-based AR. Postupem času spíše integrují Marker AR.

5 ZÁKLADNÍ ASPEKTY VÝVOJE APLIKACÍ PRO MOBILNÍ PLATFORMU ANDROID

Všechny potřebné nástroje pro vývoj aplikací na platformu Android mohou být staženy zdarma z oficiálních developerských stránek. Oficiálně podporovaným vývojovým prostředím je unifikované IDE (Integrated Development Environment) Android Studio, dostupné pro operační systémy Windows, Mac OS i Linux, které umožňuje vyvíjet aplikace na všechny známé druhy Android zařízení. Pro snažší vytváření aplikací také obsahuje nejrůznější testovací/ladící nástroje[107] a vestavěné funkce[108].

Tato kapitola si klade za cíl rámcově zasadit do kontextu základní programové vybavení a aspekty vývoje aplikací na platformě Android. Detailnější informace opět poskytují oficiální developerské stránky [43].

5.1.1 Aplikační komponenty

Na rozdíl od klasických desktop aplikací neobsahuje Android aplikace klasický vstupní bod, typicky se jedná o statickou metodu main s různými parametry, ale složena je z více na sobě nezávislých aplikačních komponent. Tyto fundamentální komponenty jsou při vývoji aplikace klíčové, plní určitou danou specifickou funkci aplikace, umožňují její běh a interakci s operačním systémem.

Existují čtyři typy aplikačních komponent:

- *Activities* – Tvoří tzv. vstupní bod pro interakci „aplikace“ s uživatelem. Vytváří uživatelský zážitek, který je rozdělen na jednotlivé nezávislé obrazovky s definovanými funkcemi a grafickým rozhraním.

  V aplikaci bývají volně vázány mezi sebou – obecně existuje jedna „primární“ a více „sekundárních“ aktivit.

- *Services* – Neposkytují uživatelské rozhraní na rozdíl od aktivit, ale umožňují provádět výpočetně náročné operace na pozadí nebo přistupovat k nejrůznějším vzdáleným zdrojům a službám. Služba např. umožňuje přehrávat hudbu na pozadí, i když se uživatel nachází již v jiné aplikaci nebo aktivitě.

[107] Obsahuje Profiling tools, emulátor pro simulaci fyzických zařízení s OS Android a další jiné nástroje.
[108] Podporuje funkci Inteligentní kód editor, Instant Run a jiné funkce.

- *Broadcast receivers* – Umožňují aplikaci reagovat na různé vzniklé události v systému nebo přímo v aplikaci. Příkladem může být příchod SMS (Short Message Service) zprávy, přijatý hovor nebo uživatelsky definována zpráva.
- *Content providers* – Umožňují spravovat přístup k aplikačním datům[109], na kterých je možné případně v závislosti na uděleném oprávnění provádět základní CRUD (Create-Retrieve-Update-Delete) operace. Poskytují mechanismus neboli vytvářejí abstrakční vrstvu pro sdílení obsahu napříč aplikacemi[110].

Tři ze čtyř komponent (Activities, Services a Broadcast receivers) spolu interagují přes tzv. Intents[111] neboli záměry, zatímco pro přístup k datům poskytovaným přes Content provider je definován Content Resolver[112].

5.1.2 Manifest soubor

AndroidManifest.xml je primární soubor pro specifikaci vlastností Android aplikace, které jsou poskytovány operačnímu systému. Deklarovány jsou parametry jako identifikátor aplikace, použité komponenty, systémová oprávnění/požadavky, seznam využívaných knihoven a další jiné. Definována je i minimální podporovaná[113] verze Android API.

5.1.3 Aplikační zdroje

Aplikace pro Android neobsahuje pouze fundamentální komponenty, ale vyžaduje různé další zdroje, jako jsou obrázky, zvukové soubory nebo definice vzhledu aplikace[114], které jsou separátní od zdrojového kódu. Tímto je dosažena optimalizace aplikace pro divergentní konfigurace[115] zařízení a je usnadněna aktualizace aplikace bez nutnosti zásahu do aplikační logiky.

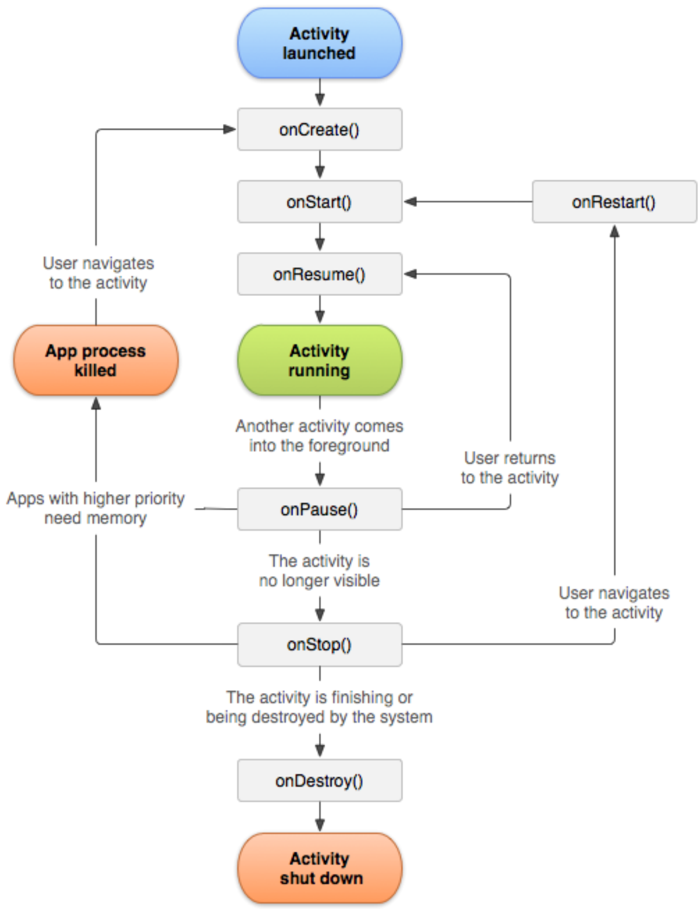
---

[109] Přímo data aplikace nebo data ostatních aplikací.
[110] Operační systém izoluje jednotlivá data aplikací. Tato data mohou být uložena např. v rámci souborového systému, webového kanálů nebo jakéhokoliv perzistentního úložiště, ke kterému má daná aplikace přístup.
[111] Poskytují explicitní/implicitní asynchronní mechanismus pro komunikaci mezi komponentami aplikace nebo přímo jinými aplikacemi v rámci systému, např. umožňují zaslat zprávu pro přepnutí mezi aktivitami.
[112] Content Resolver zasílá požadavky Content provideru, který následně provede danou operaci a vrátí výsledek.
[113] Umožňuje omezit/podporovat cílová zařízení aplikace.
[114] Definovat lze mimo uživatelských rozhraní dále např. různé animace, menu, styly a barvy.
[115] Android systém je obsažen v zařízeních, které mají nejen různé technické vlastnosti, ale i např. jazykové mutace.

5.1.4 Životní cyklus aplikace

Všechny součásti spuštěné aplikace ve výchozím nastavení běží ve stejném samostatném procesu[116] neboli vlastní instanci virtuálního stroje Android Runtime nebo Dalvik v závislosti na použité verzi systému s definovaným UI vláknem[117].

Alokování paměti či vytváření podpůrných procesů je výhradně řešeno za běhu. Operační systém v případě nedostatku systémových zdrojů automaticky ukončuje aplikace, které uživatel dlouhodobě nevyužívá – rozhodnutí bývá úzce spjato se stavem jednotlivých komponent procesu. Ukončování procesů probíhá tedy podle jejich priority, kdy ty s nejnižší prioritou jsou ukončeny nejdříve. Operační systém sám tedy určuje životní cyklus aplikace, tento mechanismus je patrný např. u primární komponenty aktivita, kde definuje stav a události, kterými daná aktivita prochází, viz obrázek 30.



Obrázek 30: Životní cyklus komponenty aktivita. Zdroj: [43].

[116] Komponenty aplikace lze definovat i v separátních procesech. Libovolně lze vytvářet tzv. pracovní vlákna.
[117] Toto vlákno by nemělo být blokováno dlouhými operacemi, jako jsou např. databázové dotazy nebo síťové operace. Zablokování může způsobit zamrznutí aplikace nebo je dokonce nutné aplikaci ukončit.

6 REALIZACE APLIKACE

Tato kapitola je zaměřena na popis realizace aplikace a vše, co se realizace[118] týká. Popsána bude architektura aplikace, popis návrhu grafického designu aplikace, stěžejní principy práce s rozšířenou realitou včetně všech potřebných součástí aplikace.

6.1 Aplikační rámec aplikace

Aplikace je cílená na zařízení s operačním systémem Android verze minimálně 7.0[119] (Nougat) a novější, toto omezení je dáno závislostí využití platformy SceneForm (ARCore) pro rozšířenou realitu.

Každá aplikace je v rámci OS identifikována jménem balíků neboli package uvedeným v manifestu aplikace. V případě této aplikace bylo zvoleno jméno aplikace cz.upce.arviewer.

Vytvořená je základní třída pro zachování globálního stavu celé aplikace ARViewer, která je potomkem třídy Application.

```
public class ARViewer extends Application … //Generalizace{
… //Obsah třídy
}
```

Současně musí být tato třída uvedena v manifestu aplikace, aby ji bylo možné používat.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cz.upce.arviewer">
        … //Ostatní atributy a elementy
        <application
            android:name=".ARViewer"
            … //Ostatní atributy a elementy
        </application>
</manifest>
```

V aplikaci je využívána především pro inicializaci různých komponent aplikace. Detailněji bude na tyto inicializace upozorněno až v případě jejich využití.

Aplikace je vystavěna na využívání nejrůznějších modulů a senzorů, vestavěných přímo v mobilním telefonu a jejich vzájemném kombinování. Zejména se jedná o videokameru, GPS, nejrůznější polohové senzory a využívání mobilního připojení k internetu.

[118] Uvedeny jsou i části kódu, které jsou pro snažší čitelnost upraveny.
[119] Nougat (API level 24).

Pro využití těchto zdrojů je nutné deklarovat v manifestu aplikace přístupová práva tzv. permissions, které musí uživatel v případě používání aplikace z důvodu bezpečnosti odsouhlasit.

Každé takové oprávnění má nastavenou úroveň ochrany a v závislosti na této úrovni je nutné toto oprávnění ošetřit[120].V manifestu jsou tedy definovány permission elementy use-permission[121], které aplikace potřebuje pro svoji funkci. V rámci realizace je nutné dále ošetřit situace, kdy uživatel v průběhu využívání aplikace zruší dané oprávnění.

Konkrétně se jedná o tato oprávnění:

- *CAMERA* – Oprávnění, které umožňuje využívat kameru daného zařízení.
- *ACCESS_FINE_LOCATION* – Oprávnění, které determinuje přístup aplikace k aktuální poloze zařízení. Lze přistupovat např. skrze poskytovatele služby GPS.
- *INTERNET* – Oprávnění umožňuje aplikaci otevřít síťové sockety – možnost využívat připojení k internetu.
- *ACCESS_NETWORK_STATE* – Oprávnění umožňuje zjišťovat informace o dostupných přístupových sítích.

```
<usus-permission android:name="android.permission.CAMERA" />
… //Ostatní uses-permission
```

Dále jsou v manifestu definovány elementy uses-feature pro filtraci zařízení, která nesplňují požadavky na HW a SW vybavení.

```
<uses-feature
    android:name="android.hardware.camera.ar"
    android:required="true" />
… //Ostatní uses-feature
```

Nedílnou součástí manifestu aplikace je definování aplikačních komponent, které aplikace využívá a definování samostatných atributů aplikace a meta-dat aplikace.

[120] Oprávnění uživatele nemusí nijak ovlivňovat, nebo musí dané oprávnění potvrdit.

[121] Operační systém Android od verze 6.0 obsahuje změnu s oprávněními, které lze měnit za běhu aplikace – na rozdíl od dřívějších verzí, kde byla práva povolována při instalaci aplikace.

```
<meta-data
    android:name="com.google.ar.core"
    android:value="required" />
… //Ostatní meta-data
```

```
<activity
    android:name=".mvp.view.activity.MainActivity"
    android:label="@string/title_activity_main"
    android:configChanges="locale|orientation|screenSize"
    android:screenOrientation="portrait"
    android:theme="@style/AppTheme.NoActionBar" />
… //Ostatní aplikační komponenty
```

6.2 Vývojové prostředí a sestavovací systém

Aplikace byla po celou dobu vyvíjena ve vývojovém prostředí Android Studio s postupnými aktualizacemi[122]. Jeho součástí je také sestavovací systém Gradle, který v aplikaci obsahuje minimálně dva[123] konfigurační soubory, na základě kterých může být sestavena aplikace. První soubor je specifický pro aplikaci a druhý pro kompletní projekt.

```
apply plugin: 'com.android.application'
apply plugin: 'realm-android'

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "cz.upce.arviewer"
        minSdkVersion 24
        targetSdkVersion 28
     … //Ostatní atributy
    }
   … //Ostatní atributy
}
```

V závislosti využití se pak mohou konfigurovat závislosti jednotlivých součástí projektu nebo závislosti pro celý projekt.

```
dependencies {
    //Google Sceneform
    implementation "com.google.ar.sceneform.ux:sceneform-ux:1.10.0"
    //Google ARCore
    implementation 'com.google.ar:core:1.10.0'
  … //Ostatní závislosti
}
```

[122] Poslední aktualizace byla 3.4.2.
[123] Projekt se může skládat i z více souborů (někdy také označováno jako moduly).

6.2.1 Testovací prostředí

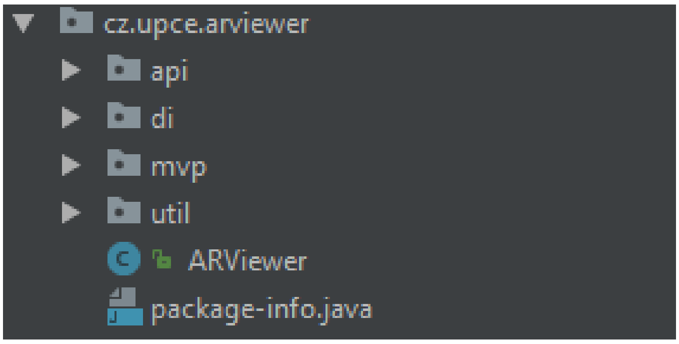
Testovacím prostředím jsou zde myšlena testovací zařízení a testovací server, nikoliv testování z pohledu ověřování kvality zdrojového kódu.

Aplikace byla vyvíjena a průběžně testována na fyzickém zařízení Nokia 6.1, a to především z důvodu cenové dostupnosti zařízení pro SceneForm (ARCore).

Dále jako testovací prostředí lze označit i vytvořený testovací server, který byl spouštěn pouze v lokální domácí síti. Tento server umožňuje simulovat požadované webové služby potřebné pro chod mobilní aplikace.

6.3 Architektura projektu

Pro přehlednost a snadnou rozšiřitelnost jsou součásti aplikace rozděleny do několika samostatných balíčků s určitou funkcí, které jsou ve většině případů nezávislé, a proto není nutné při změně funkcionality upravovat celou vytvořenou aplikaci. Základní strukturu lze pozorovat na obrázku 31.



Obrázek 31: Základní struktura projektu aplikace. Zdroj: vlastní.

Balíček api definuje aplikační třídy, zatímco balíček di obsahuje třídy pro Dependency Injection. Třídy spojené s MVP architekturou jsou obsaženy v balíčku mvp. V neposlední řadě je obsažen balíček util pro tzv. podpůrné třídy aplikace.

6.3.1 Architektura aplikace

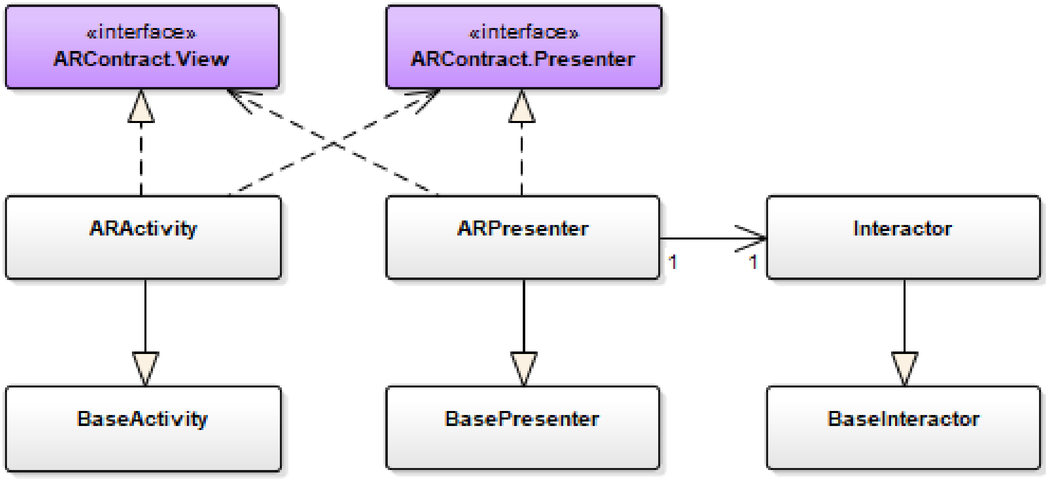
Vytvořená aplikace je postavena nad architekturou MVP. Tato architektura je realizována u každého View, které zobrazuje aplikační data nebo případně vyžaduje nějakou aplikační logiku, v aplikaci se tedy jedná o aktivity, zejména pak fragmenty.

Komunikace mezi komponentou View a komponentou Presenter je dána speciální dohodou přes tzv. „Contract“. Tímto je definována obousměrná komunikace přes definovaná rozhraní, která obsahují příslušné akce.

```
public interface ARActivityContract {
    interface ARActivityView {
        … //Obsah rozhraní    }
    interface ARActivityPresenter {
        … //obsah rozhraní
    }
}
```

Presentery, které vyžadují business logiku, dále využívají třídy typu Interactor, které obstarávájí dodání požadovaných aplikačních dat. Na tyto Interactory jsou pak v závislosti využití dodávány závislosti tříd vykonávající vybranou business logiku[124].

Příkladem lze uvést aktivitu ARActivity, která implementuje rozhraní ARContract.View. Presenter ARActivityPresenter naopak implementuje rozhraní ARContract.Presenter.



Obrázek 32: Zjednodušený diagram tříd MVP architektury aplikace. Zdroj: vlastní.

UML (Unified Modeling Language) diagram těchto vztahů popisuje obrázek 32, ze kterého je dále patrné napojení na různé interaktory[125] se specifickou funkcionalitou. U všech ostatních View je situace podobná, liší se pouze tím, jaké akce jsou realizované mezi View a daným Presenterem.

---

[124] Případně může být logika vykonávána přímo v daném Interactoru.
[125] V diagramu tříd je použita pouze fiktivní třída Interactor, která by měla mapovat asociaci presenteru s libovolným interactorem.

V aplikaci jsou tyto třídy typu Interactor:

- *DatabaseInteractor* – Interactor je využíván pro práci s databází Realm.
- *FileSystemInteractor* – Interactor obsluhuje funkcionalitu spojenou s ukládáním a načítáním grafických podkladů z perzistentního úložiště.
- *LocationInteractor* – Interactor umožňuje získávat data spojené s lokalizací zařízení.
- *NetworkInteractor* – Interactor je využíván pro práci se sítí.
- *WebServiceInteractor* – Interactor pro komunikaci se vzdáleným serverem.
- *SensorsInteractor* – Interactor zajišťující spolupráci s třídami odpovědnými za spolupráci se sensory daného zařízení.

6.3.2 Dependency Injection

Aplikace je dále postavena na využívání principů techniky[126] Dependency Injection (dále také DI), které si klade za cíl redukci pevných vazeb mezi komponentami neboli závislostí jednotlivých komponent na konkrétní realizaci. Tyto principy umožňují větší přehlednost kódu, a především jeho lepší testovatelnost a znuvupoužitelnost kódu.

Cílem této kapitoly není seznámit s celým procesem tvorby vkládání závislostí, ale ukázat na jakých principech je aplikace vystavena.

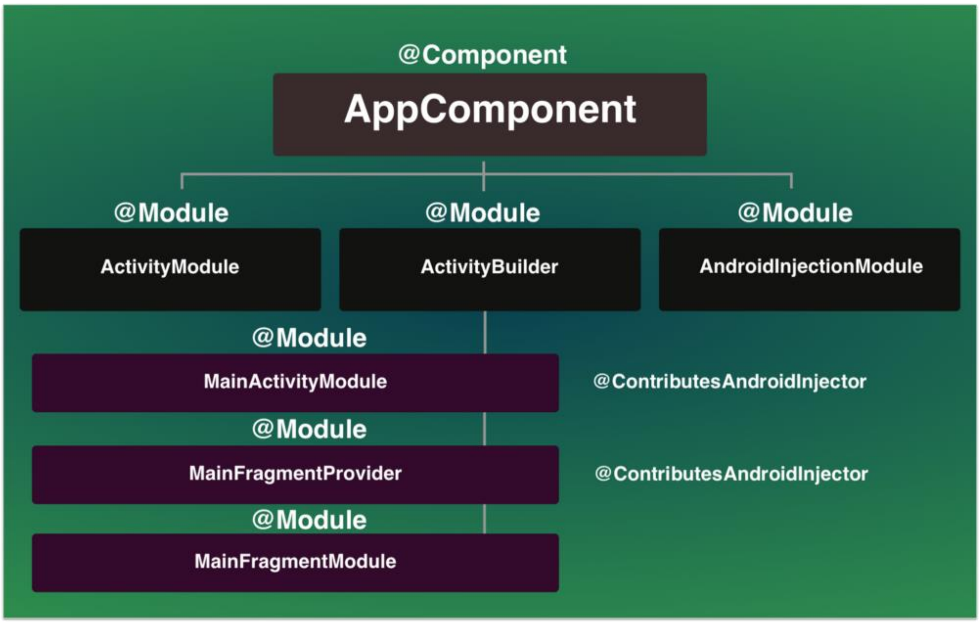
Techniky DI jsou v aplikaci řešeny pomocí frameworku Dagger 2 (dále také označováno jako Dagger), který je zjednodušeně založen na generování kódu potřebného pro vkládání závislostí ze statické[127] analýzy založené na předdefinovaných Java anotací[128]. Detailnější informace lze dohledat přímo v [44].

Pro využití Dagger v aplikaci je nutné zahrnout závislosti do konfiguračního souboru.

```
compile 'com.google.dagger:dagger:2.14.1'
annotationProcessor 'com.google.dagger:dagger-compiler:2.14.1'
annotationProcessor 'com.google.dagger:dagger-android-processor:2.14.1'
compile 'com.google.dagger:dagger-android-support:2.14.1'
```

[126] „Návrhový vzor" z rodiny IoC (Inversion of Control).
[127] Analýza není prováděna za běhu aplikace.
[128] Dagger definuje moduly označené anotací @Module, které obsahují poskytovatele obsahu neboli metody označené @Provides, Dále je definováno rozhraní označené anotací @Component, které zpřístupňuje závislosti poskytované moduly a umožňuje vkládat závislosti do označených míst anotací @Inject. V neposlední řadě je definován rozsah označený anotací @Scope, který vymezuje životnost závislostí.

Na obrázku 33 může být pozorován zjednodušený diagram využití Dagger na platformě Android. Hlavní a nejvýše položenou komponentou je AppComponent, která umožňuje definovat moduly nutné pro provedení vkládání závislostí.



Obrázek 33: Zjednodušený diagram využití Dagger na platformě Android. Zdroj [45].

Definován je modul AppModule, který poskytuje požadované závislosti sdílené napříč aplikací, modul ActivityBuilder, který umožňuje mapovat[129] všechny aktivity aplikace a případné fragmenty. Poslední modul AndroidInjectionModule je interní modul Dagger frameworku.

Následně jsou vytvořeny moduly pro activity a fragmenty, ty poskytují požadované závislosti[130], nejčastěji se jedná o instance View a presenteru z MVP architektury.

```
@Module
public class ARActivityModule {
    @PerActivityScope
    @Provides
    ARActivityContract.ARActivityView provideARView(…){
    … //
    }
    … //Ostatní závislosti
}
```

[129] Využita je anotace @ContributesAndroidInjector se specifikací modulů.
[130] Díky DI všechny závislosti aktivity nebo fragmentu existují pouze tehdy, když je aktivita nebo fragment aktivní. Jakmile se aktivita nebo fragment ukončí, dojde k uvolnění alokovaných závislostí.

Dále jsou vytvořeny moduly pro business logiku, definované v AppModule:

- *DatabaseModule* – Modul poskytuje závislosti související s databází Realm.
- *FileSystemModule* – Modul poskytuje závislosti související se souborovým systémem.
- *LocationModule* – Modul poskytuje závislosti související s lokací.
- *NetworkModule* – Modul poskytuje závislosti související se sítí.
- *WebServiceModule* – Modul poskytuje závislosti související s webovými službami.
- *SensorsModule* – Modul poskytuje závislosti související se senzory zařízení.

Následně může být provedena výsledná inicializace Dagger komponenty v aplikační třídě ARViewer[131].

```
DaggerAppComponent
    .builder()
    .application(this)
    .build()
    .inject(this);
```

Dále je nutné provést AndroidInjection.inject v metodě životního cyklu onCreate[132].

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    AndroidInjection.inject(this);
    super.onCreate(savedInstanceState);
    … //Ostatní akce
}
```

Poté je možné vložit požadovanou závislost, např. lze využít Presenter pro konkrétní View neboli injektovat přímo do aktivity/fragmentu. Daná třída se nemusí starat o inicializaci, a lze tak snadno provádět změny. O inicializaci se stará ten, kdo objekt poskytuje.

```
public class ARActivity extends BaseActivity implements … //Generalizace{
    @Inject
    ARActivityPresenter arActivityPresenter;
    … //Obsah třídy
}
```

[131] Tato aplikační třída musí implementovat rozhraní HasActivityInjector.

[132] V případě, že aktivita obsahuje fragmenty je nutné v aktivitě definovat rozhraní HasFragmentInjector a následně provést AndroidInjection.inject v daném fragmentu v metodě onAttach.

6.3.3 Reaktivní programování

Knihovna RxJava je implementací funkcionálně reaktivního rozhraní ReactiveX v jazyce Java, která umožňuje programování s asynchronními datovými proudy[133] neboli asynchronní operace a události. ReactiveX je kombinací nejlepších návrhových vzorů Observer, Iterator a funkcionálního programování. Doplňující informace lze nalézt na oficiálních stránkách [46].

Využita byla druhá verze knihovny, pro kterou je opět nutné definovat závislosti.

```
implementation 'io.reactivex.rxjava2:rxjava:2.2.10'
implementation 'io.reactivex.rxjava2:rxandroid:2.1.1'
```

Tato knihovna je postavena na definici tzv. poskytovatelů[134], konzumentů[135]a operátorů[136]. Význam jednotlivých komponent je zřejmý. Poskytoval plní roli tzv. „mluvčího“, konzument roli „posluchače“ a operátor plní roli „překladače“.

V aplikaci jsou využity[137] nejčastěji poskytovatelé typu Flowable, Single a případně Completable. Flowable je objekt, který nabízí regulaci emitace zpráv. Pro případ emitace jediné zprávy je určený Single. Poslední využitý poskytovatel je využíván pouze pro emitování stavů aplikace. V ukázce kódu níže je zobrazen databázový dotaz, jehož výsledek je emitován přes objekt typu Flowable, zatímco typ konzumenta je v podobě speciálního objektu Disposable.

```
public Flowable<Asset> getAsset() {
    … //Databázový dotaz pro asset
    return Flowable.just(asset);
}
```

Konzument nejprve registruje poskytovatele metodou add, případně lze definovat sled prováděných datových proudů metodou andThen. Konzument dále definuje několik metod, které např. umožňují definovat typ vláken, nad kterými bude asynchronní datový proud prováděn. Dále mohou být obsaženy operátory neboli transformační funkce, prováděné nad daty datového proudu.

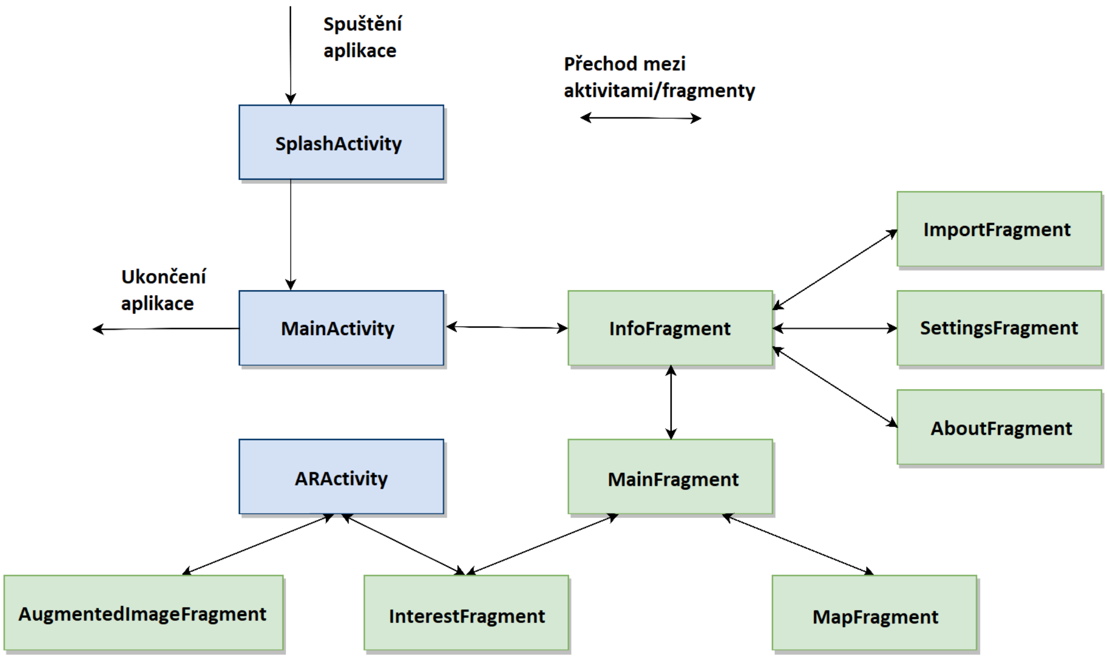
[133] Datový proud je sekvence dat obsahující různé prvky. Mohou to být proměnné, vstupy od uživatele a další.
[134] Označovaný také jako Observable.
[135] Označovaný také jako Observer.
[136] Označovaný také jako Operator.
[137] RxJava zavádí základní typy pro vytváření asynchronních datových proudů: Flowable, Observable, Single, Completable, Subject, Completable a Maybe.

V neposlední řadě je provedena metoda subscribe, která zpracovává emitované hodnoty včetně případných chyb. Následně může být provedeno zneplatnění konzumenta.

```
@Override
public void fetchAsset() {
    getDisposables().add(databaseInteractor.getAsset()
        .subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(asset -> {
            //Ostatní akce
        }, throwable -> {
            //Chybové akce
        })
    );
}
```

6.4 Uživatelské rozhraní

Základním požadavkem na uživatelské rozhraní je jeho přehlednost, jednoduchost a případná rozšiřitelnost. Definovány jsou čtyři aktivity a devět fragmentů, které tvoří grafické rozhraní aplikace[138]. Na obrázku 34 lze pozorovat diagram přechodu aktivit.



Obrázek 34: Diagram přechodu aktivit a fragmentů. Zdroj: vlastní.

[138] UI neboli grafické rozhraní, které je viditelné přímo uživateli, tvoří pouze tři aktivity a osm fragmentů. Poslední aktivita a fragment slouží ke generalizaci daných rozhraní.

Význam jednotlivých aktivit na obrázku 34 je následující:

- *SplashActivity* – Vstupní aktivita aplikace, která tvoří úvodní obrazovku po spuštění aplikace s vlastním logem aplikace.
- *MainActivity* – Aktivita je rozcestníkem aplikace, umožňující veškerou správu aplikace.
- *ARActivity* – Aktivita, která zajišťuje veškeré akce spojené s rozšířenou realitou.
- *BaseActivity* – Aktivita v podobě abstraktní třídy, která slouží ke generalizaci ostatních aktivit.

Z pohledu grafického rozhraní je zajímavá především aktivita MainActivity, která definuje pro svoji funkcionalitu několik odlišných fragmentů[139] se specifickými rolemi v aplikaci.

Význam fragmentů v aplikaci je následující:

- *AboutFragment* – Fragment poskytuje staticky definované informace o aplikaci.
- *ImportFragment* – Fragment umožňuje importovat body zájmu ve formě assetu a příslušné grafické podklady pro zajištění funkčnosti aplikace.
- *InfoFragment* – Fragment zobrazuje informační data ze získaného assetu.
- *MainFragment* – Fragment zajišťuje grafický a funkcionální podklad pro vnořené fragmenty InterestsFragment a MapFragment.
  - *InterestsFragment* – Fragment zobrazuje body zájmu ve formě seznamu s dodatečnými informacemi a s možností přechodu do rozšířené reality.
  - *MapFragment* – Fragment spravuje mapové podklady.
- *SettingsFragment* – Fragment umožňuje měnit nastavení aplikace.
- *AugmentedImageFragment*[140] – Fragment zajišťuje akce spojené s rozšířenou realitou.
- *BaseFragment* – Fragment v podobě abstraktní třídy, který slouží ke generalizaci ostatních fragmentů.

[139] Stejné funkcionality by bylo možné dosáhnout použitím několika aktivit, od tohoto řešení bylo upuštěno, neboť vytvářet aktivity pro jednotlivé funkce by mohlo být méně efektivní z výkonnostního hlediska. Fragmenty ale do aplikace mohou vnášet zcela jistě menší přehlednost v projektu a složitosti kódu.

[140] Tento fragment je využit v ARActivity, ostatní fragmenty jsou využity v MainActivity.

6.5 Mapové podklady

Mapové podklady Google Maps tvoří nedílnou součást aplikace. Zobrazují aktuální polohu uživatele a jednotlivé body zájmu ve formě definovaných značek. Pro využití mapových podkladů je nutným požadavkem opět definovat potřebné závislosti do konfiguračního souboru.

```
implementation 'com.google.android.gms:play-services-maps:17.0.0'
```

Nedílnou součástí je také proces vygenerování API klíče[141] a následné využití v manifestu aplikace. Z pohledu využití map je provedena základní inicializace a přidávání bodů zájmu na mapu ve formě grafických značek[142].

```
//Iterace vybranými body zájmu
for (Interest interest : event.getInterests()) {
    //Aktuální poloha bodu zájmu
    LatLng = new LatLng(interest.getLocation().getLatitude(),
                        interest.getLocation().getLongitude());
    //Titulek bodu zájmu
    String title = interest.getName();
    //Vytvoření markeru s požadovanými vlastnostmi
    MarkerOptions markerOptions = MapHelper.createMarker(…);
    //Přidání aktuálního markeru do mapového podkladu
    googleMap.addMarker(markerOptions);
}
```

Na ukázce kódu výše lze pozorovat přidávání bodu zájmu do mapového podkladu. Tyto mapové podklady jsou zobrazeny ve fragmentu MapFragment[143].

6.6 Lokační technologie

Využití lokačních služeb tvoří páteř celé aplikace. Ze získané polohy zařízení a okamžité orientace zařízení, lze následně provést transformace pro výsledné renderování rozšířené reality.

6.6.1 Geolokace zařízení

Geolokace vyžaduje opět definovat závislosti pro využití v aplikaci.

```
implementation 'com.google.android.gms:play-services-location:17.0.0'
```

[141] Proces generování klíče může být dohledán na oficiálních stránkách Android.
[142] Zvoleny byly defaultní značky se změněnou barvou z důvodu lepší přehlednosti a zažité grafické kultury.
[143] Potřebné geolokační souřadnice jsou získávány z LocationService nebo jsou získána přímo z bodů zájmu.

Definován je LocationModule pro poskytování potřebných závislostí[144] a příslušný LocationInteractor, který realizuje příslušné lokační akce[145] nad API LocationService. LocationService API obstarává periodicky geolokační data[146], přesněji je využita pouze poslední známá lokace. Následně je možné využít LocationInteractor v příslušném Presenteru[147].

API LocationService obsahuje hlavní metodu subscribe, která umožňuje periodické emitování aktuálních lokací pomocí reaktivního programování.

```
@Override
public void subscribe(FlowableEmitter<Location> emitter) {
    //Vytvoření LocationCallback pro záchyt lokací
    LocationCallback mLocationCallback = new LocationCallback() {
        @Override
        public void onLocationResult(LocationResult locationResult) {
            //Seznam dostupných lokací
            List<Location> locationList = locationResult.getLocations();
            //Kontrola stavu kontejneru lokací
            if (locationList.size() > 0) {
                //Zjištění poslední lokace
                Location location = locationResult.getLastLocation();
                //Emitace poslední lokace
                emitter.onNext(location);
            }
        }
    };
    //Vytvoření požadavku na aktualizaci polohy
    fusedLocationClient.requestLocationUpdates(locationRequest,
                                               mLocationCallback,
                                               Looper.myLooper());
    emitter.setCancellable(() -> {
        //Zneplatnění locationRequest
        this.locationRequest = null;
        //Zneplatnění LocationCallback pro záchyt lokací
        fusedLocationClient.removeLocationUpdates(mLocationCallback);
    });
}
```

Pro zjištění polohy zařízení je využit FusedLocationProviderClient, který je doporučovanou metodou pro aplikace pracující s geolokací. FusedLocationProviderClient umožňuje vytvořit požadavek na aktualizaci polohy a následnou obsluhu vyvolané události neboli emitace poslední známé lokace přes LocationCallback. V případě, kdy není aktuální poloha již požadována, dojde k zneplatnění požadavku a LocationCallback.

[144] Poskytován je FusedLocationProviderClient a LocationManager.
[145] Zjišťuje aktuální lokaci a stav lokačních služeb.
[146] Geolokační data jsou získávána přímo v podobě geografický souřadnic.
[147] MapFragmentPresenter nebo ARActivityPresenter.

6.6.2 Okamžitá orientace zařízení

Zjištění správné orientace zařízení v reálném světě není na rozdíl od geolokačních systémů poskytována přímo – získání není tak přímočaré jako v případě aktuální polohy.

Definován je SensorsModule pro poskytování potřebných závislostí[148] a příslušný SensorsInteractor, který realizuje příslušné akce pro zjištění orientace zařízení nad API SensorsService, která obstarává periodicky získává data ze senzorů daného zařízení[149]. API SensorsService obsahuje opět hlavní metodu subscribe, která umožňuje periodické emitování aktuálních dat pro určení orientace pomocí reaktivního programování.

```
@Override
public void subscribe(FlowableEmitter<Orientation> emitter) {
    //Získání senzoru vybraného typu
    sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR);
    … //Kontrola senzoru
    //Definování listeneru pro daný senzor
    SensorEventListener sensorEventListener = new SensorEventListener() {
        @Override
        public void onSensorChanged(SensorEvent event) {
            if (event.sensor.getType() == Sensor.TYPE_ROTATION_VECTOR) {
                //Filtrace dat ze senzoru
                //Získání rotační matice
                SensorManager.getRotationMatrixFromVector(rotationMatrix,
                                                          event.values);
                //Přetransformování rotační matice
                SensorManager.remapCoordinateSystem(…);
                //Výpočet orientace z rotační matice
                SensorManager.getOrientation(updateRotationMatrix,
                                             orientationValues);
                //Získání požadovaných lokačních informací
                Orientation orientation = new Orientation();
                orientation.setAzimuth((float)
                                   Math.toDegrees(orientationValues[0]));
                orientation.setPitch((float)
                                   Math.toDegrees(orientationValues[1]));
                orientation.setRoll((float)
                                   Math.toDegrees(orientationValues[2]));
                //Emitování lokačního objektu
                emitter.onNext(orientation);
            }
        }
        … //Metoda OnAccuracyChanged
        };
        //Registrace listeneru pro příjem dat
        sensorManager.registerListener(…);
        … //Zrušení emitace a odregistrace listeneru
    }
}
```

[148] Poskytován je SensorManager a WindowManager.
[149] Data jsou nejprve filtrována pomocí filtru typu dolní propusti následně je získána relativní orientace v prostoru pomocí tří úhlů: yaw, roll a pitch (definice je nejčastěji vysvětlována na 3D modelu letadla).

Pro zjištění orientace je využit SensorManager, který umožňuje reagovat na změnu událostí vybraných senzorů. Lze tedy vypočítat z přetransformované rotační matice lokační informace.

6.7 Identifikační technologie

ARCore framework obsahuje technologii Augmented Images, která umožňuje být využita jako identifikační technologie. Lze tak provádět reakce v závislosti na staticky detekovaném objektu.

V aplikaci musí být definována tzv. image databáze[150] pro detekovatelné objekty, které mohou být pouze ve formátech PNG nebo JPEG a názvu složeného z ASCII znaků. Databázi lze vytvořit přes nástroj arcoreimg tool[151]. Pro využití byl vytvořen QR kód v generátoru dostupném na [47], který nabízí jednoduché generování.

Obrázek 35 Detekovatelný objekt ve formě QR kódu. Zdroj: vlastní.

Vygenerovaný obrázek lze zkontrolovat na škále 0-100[152] kvalitu obrázku pro použití v Augmented Images, viz obrázek 36, ze kterého je patrné, že vygenerovaný obrázek splňuje požadavek na použití.

```
C:\Users\Tomas-PC\Documents>arcoreimg.exe eval-img --input_image_path=ARViewerQRCode.png
100
```

Obrázek 36: Ověření kvality obrázku pro Augmented Images. Zdroj: vlastní.

Následně je možné vytvořit databázi viz obrázek 37. Využití databáze pro rozšířenou realitu bude komentováno v následujících kapitolách.

```
C:\Users\Tomas-PC\Documents>arcoreimg.exe build-db --input_image_list_path=image_list_file.txt
--output_db_path=ARimages.imgdb
Image database generated at: ARimages.imgdb
Image list file generated at: ARimages.imgdb-imglist.txt
```

Obrázek 37: Generování databáze pro Augmented Images. Zdroj: vlastní.

[150] Databázi lze vytvořit v programu nebo načíst existující. Zvolena byla druhá možnost.
[151] Spouštění v příkazovém řádku.
[152] Hodnocení 0 znamená nevhodné, zatímco hodnocení 100 znamená nejvíce vhodné.

6.8 Webové služby

Aplikace umožňuje načíst Asset požadovaných bodů zájmu z testovacího serveru, kde je příslušný asset uložen ve formátu JSON souboru a grafické podklady ve formátu JPG.

Pro komunikaci se serverem je použita knihovna Retrofit neboli klient HTTP pro Android, který umožňuje namapovat HTTP API do rozhraní programovacího jazyka Java. HTTP response body obsahuje veškeré potřebné informace. Retrofit analyzuje data a umožňuje namapovat JSON data přímo do Java objektů.

Pro využití Retrofitu je nutné deklarovat několik závislostí, které jsou definovány v lokálním souboru Gradle.

```
implementation 'com.squareup.retrofit2:converter-gson:2.3.0'
implementation 'com.squareup.retrofit2:retrofit:2.3.0'
implementation 'com.squareup.okhttp3:logging-interceptor:3.3.0'
implementation 'com.squareup.retrofit2:adapter-rxjava2:2.3.0'
```

Definován je WebServiceModule pro poskytování potřebných závislostí[153] a příslušný WebServiceInteractor, který realizuje příslušné webové akce[154] nad API WebService, která umožňuje stažení assetu přes aplikační rozhraní[155].

```
public interface WebService {
    @GET("/{asset}")
    Flowable<Asset> getAsset(@Path("asset") String asset);
}
```

Ze získaného assetu lze následně stáhnout příslušný grafický podklad assetu a jednotlivých bodů zájmu. Pro stahování grafických podkladů je využita knihovna Picasso[156].

Pro využití Picasso je nutné opět definovat potřebnou závislost.

```
implementation 'com.squareup.picasso:picasso:2.71828'
```

WebServiceInteractor obsahuje metodu downloadAssetImage, která umožňuje stahování grafických podkladů.

153 Poskytován je Retrofit, OkHttpClient, GsonConverterFactory a další.
154 Zajišťuje stažení assetu a stažení grafických podkladů.
155 Síťový provoz lze provádět pomocí asynchronních operací a událostí.
156 Picasso manažer umožňuje nejen stahovat obrázky, ale provádět transformace a cache operace.

```
public Single<Pair<Bitmap, String>> downloadAssetImage(String url, String
                                                               name) {
    return Single.create(emitter -> {
        … //Target
        Picasso.get()
            .load(url)
            .into(target);
    });
}
```

Následně je vytvořen Target objekt, který představuje libovolný posluchač pro načtení grafický podkladů.

```
Target target = new Target() {
    @Override
    public void onBitmapLoaded(Bitmap bitmap, Picasso.LoadedFrom from) {
        //Emitace stažené bitmapy
        emitter.onSuccess(new Pair<>(bitmap, name));
    }
    … //Metoda onBitmapFailed
    … //Metoda onPrepareLoad
};
```

6.9 Perzistentní úložiště

Všechny body zájmu, které aplikace umožňuje zobrazit včetně doplňkových informací, jsou uloženy v lokální databázi Realm. V případě uložení grafických podkladů je využito interní úložiště mobilního zařízení. Tímto lze sice dosáhnout snížení datových přenosů, ale za cenu plýtvání zdrojů[157] mobilního zařízení.

Opět je nutné definovat závislosti pro využití v aplikaci. Na rozdíl od dříve definovaných závislostí je nutné definovat pro Realm databázi závislost přímo v globálním souboru Gradle.

```
buildscript {
    … //Ostatní konfigurace
    dependencies {
        classpath "io.realm:realm-gradle-plugin:5.13.0"
      … //Ostatní závislosti
    }
}
```

Následně je možné aplikovat realm-android plugin v lokálním souboru Gradle.

```
apply plugin: 'realm-android'
```

[157] Nespornou výhodou je, že se aplikace dá částečně využít i v případě, kdy není internetová (datová) konektivita dostupná. Díky kompatibilitě této databáze by neměl být problém v případném budoucím rozšiřování.

Jakmile jsou definovány všechny potřebné závislosti, lze pak vybudovat schéma databáze, které je tvořeno modelovými třídami.

V rámci aplikace jsou definovány tři základní objektové modely:

- *Asset*[158] – Bázová třída, která v sobě mapuje všechny obsažené body zájmu.
- *Interest* – Třída reprezentuje jednotlivý bod zájmu.
- *Location* – Třída zastupuje lokační informace bodu zájmu.

Databáze Realm ukládá (mapuje) stažený asset a příslušné body zájmu přímo v modelových objektech[159]. Jednotlivé modelové třídy obsahují pouze příslušné akce pro změnu stavu jednotlivých atributů, u kterých jsou dále definovány doplňující anotace[160].

```
public class Asset extends RealmObject {… //Obsah třídy }
public class Interest extends RealmObject {… //Obsah třídy}
public class Location extends RealmObject {… //Obsah třídy }
```

V níže uvedených tabulkách lze pozorovat atributy jednotlivých modelových tříd. Z návrhu je patrné, že systém pracuje pouze s body zájmu, které mají pevnou datovou strukturu.

Tabulka 9: Atributy pro datový model Interest. Zdroj: vlastní.

Název	Datový typ	Význam
ID	String	Jednoznačný identifikátor bodu zájmu
NAME	String	Název bodu zájmu
IMAGE_PATH	String	Cesta ke grafickému podkladu
DESCRIPTION	String	Popis
LOCATION	Location	Zeměpisné souřadnice
ALTITUDE	String	Nadmořská výška
ALTITUDE	String	Zařazení bodu zájmu do kategorie
UPDATE	String	Datum poslední aktualizace

158 Asset navíc obsahuje atribut SYMBOL_IMAGE_PATH, který není prozatím využit.
159 Aby bylo možné uložit požadované informace, musí být třída potomkem třídy RealmObject.
160 Anotace nejsou využity jen pro potřeby Realm databáze, ale i např. jsou využity pro mapování z JSON souboru.

Tabulka 10: Atributy pro datový model Asset. Zdroj: vlastní.

Název	Datový typ	Význam
ID	String	Jednoznačný identifikátor assetu
NAME	String	Název bodu zájmu
INFO_IMAGE_PATH	String	Cesta ke grafickému podkladu
DESCRIPTION	String	Popis
COUNTRY	String	Stát
REGION	String	Kraj
DISTRICT	String	Okres
POPULATION	String	Počet populace
AREA	String	Rozloha
LOCATION	Location	Centrální zeměpisné souřadnice
ALTITUDE	String	Centrální nadmořská výška
WEB	String	Webové stránky
TELEPHONE_NUMBER	String	Telefonní číslo
EMAIL	String	Email
ADDRESS	String	Kontaktní adresa
UPDATE	String	Datum poslední aktualizace
INTERESTS	RealmList<Interest>	Seznam bodů zájmu assetu

Tabulka 11: Atributy pro datový model Location. Zdroj: vlastní.

Název	Datový typ	Význam
LATITUDE	String	Zeměpisná šířka
LONGITUDE	String	Zeměpisná délka

Nad takto vybudovaným schématem databáze je vytvořena třída DatabaseModule a DatabaseService, která poskytuje pouze závislost pro globální inicializaci databáze Realm v aplikační třídě ARViewer. Jednotlivé transakce nad databází jsou prováděné v DatabaseInteractor[161], který realizuje veškeré databázové dotazy.

Z ukázky kódu níže je patrné, že je před každým importovaným assetem provedeno vymazání databáze a pak následně uložen aktuálně stažený asset.

```
public class DatabaseInteractor {
    … //Proměnné a metody konstruktorů
    public void restoreAsset(Asset asset) {
        //Získání instance Realm
        try (Realm realm = Realm.getDefaultInstance()) {
            realm.executeTransactionAsync(realm1 -> {
                //Vymazání databáze
                realm1.deleteAll();
                //Znovuobnovení databáze
                realm1.copyToRealmOrUpdate(asset);
            });
        }
    }
 … //Ostatní metody
}
```

Následně jsou definovány operace pro získání již uloženého assetu, filtrování/vyhledávání bodů zájmu nebo libovolného bodu zájmu. Na ukázce kódu níže lze pozorovat metodu getAsset pro získání assetu.

```
public Flowable<Asset> getAsset() {
    … //Inicializace
    try {
        //Získání instance Realm
        realm = Realm.getDefaultInstance();
        //Databázový dotaz pro asset
        RealmQuery<Asset> query = realm
                .where(Asset.class);
        //Provedení databázového dotaz
        Asset asset = realm.copyFromRealm(query.findFirst());
        //Emitace assetu
        return Flowable.just(asset);
    } catch (Exception e) {
        //Emitace chyby
    } finally {
        … //Obsah bloku finally
        //Ukončení spojení Realm
        realm.close();
    }
}
```

[161] Komunikace s definovaným Presenterem opět probíhá přes reaktivní programování.

Dále je nutné zařídit ukládání grafických podkladů. Tyto podklady nejsou ukládány přímo v databázi, ale v lokálním úložišti mobilního zařízení. Definován je FileSystemModule pro poskytování potřebných závislostí a příslušný FileSystemInteractor, který realizuje příslušné akce na perzistentním úložišti.

Definovány jsou operace pro uložení, načtení a případné vymazání všech grafických podkladů. Na ukázce kódu níže lze pozorovat metodu loadBitmap pro načtení bitmapy.

```
public class FileSystemInteractor {
    … //Proměnné a metody konstruktorů a ostatní metody
    public Single<Pair<Integer, Bitmap>> loadBitmap(int index,
                                                     String name) {
        return Single.create(emitter -> {
            //Inicializace proměnných
            try {
                //Bytově orientovaný vstup ze souboru
                fileInputStream = new FileInputStream(getFile(name));
                //Vytvoření bitmapy
                bitmap = BitmapFactory.decodeStream(fileInputStream);
            } catch (FileNotFoundException e) {
                //Emitace prázdné bitmapy
                emitter.onSuccess(new Pair<>(index, null));
            } finally {
                try {
                    … //Ukončení bytově orientovaného vstupu
                    //Emitace bitmapy
                    emitter.onSuccess(new Pair<>(index, bitmap));
                } catch (IOException e) {
                    … //Emitace chyby
                }
            }
        });
    }
}
```

6.10 Rozšířená realita s využitím SceneForm/ARCore

Google ARCore[162] je platforma pro budování rozšířené reality na platformě Android, který společně s 3D frameworkem SceneForm umožňuje vytvářet zajímavé aplikace postavené na rozšířené realitě bez nutnosti znalostí OpenGL.

Nutné je opět deklarovat závislosti do Gradle souboru. V případě globálního Gradle souboru přístup k repozitářům Google a SceneForm plugin Google Sceneform Tools. Poté je možné aplikovat SceneForm plugin[163] v lokálním souboru Gradle.

[162] Realizace rozšířené reality pomocí SceneForm/ARCore byla inspirována příklady dostupnými z developerských stránek.
[163] Plugin Google Sceneform Tools (beta verze) lze importovat přímo přes nastavení Android Studia.

```
buildscript {
    … //Ostatní konfigurace
    dependencies {
        … //Ostatní závislosti
        classpath 'com.google.ar.sceneform:plugin:1.10.0'
    }
}
allprojects {
    repositories {
        google()
        … //Ostatní repositáře
    }
}
```

A pro lokální soubor Gradle přímo závislost pro ARCore a Sceneform UX.

```
//Google Sceneform
implementation "com.google.ar.sceneform.ux:sceneform-ux:1.10.0"
//Google ARCore
implementation 'com.google.ar:core:1.10.0'
```

Případně je nutné pro SceneForm provést další kompilační nastavení z důvodu využití Java verze 8.

```
compileOptions {
    targetCompatibility JavaVersion.VERSION_1_8
    sourceCompatibility JavaVersion.VERSION_1_8
}
```

Samozřejmostí je prerekvizita na minimální verzi Android systému.

```
android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "cz.upce.arviewer"
        minSdkVersion 24
        … //Ostatní konfigurace
    }
    … //Ostatní konfigurace
}
```

Prerekvizitou je, že musí aplikace splňovat určité požadavky pro využití SceneForm (ARCore), které musí být zaneseny do manifestu aplikace.

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature
    android:name="android.hardware.camera.ar"
    android:required="true" />
<meta-data
    android:name="com.google.ar.core"
    android:value="required" />
```

6.10.1 Renderování rozšířené reality

Aplikace je postavená na využití identifikační technologie, která umožňuje detekci definovaných značek v reálném světě a následné registraci modelů do reálné scény.

ARCore využívá tzv. session pro správu stavu systému AR a zpracování životního cyklu neboli je vstupním bodem k přístupu ARCore API – lze tak vytvořit relaci, nakonfigurovat ji, případně spustit nebo zastavit, ale nejdůležitější je umožnit přijímání tzv. frame snímků[164].

V případě využití SceneForm je poskytnut ArFragment[165], který automaticky zpracovává session a nezbytná oprávnění[166] pro fungování aplikace. Dále kontroluje, zda je nainstalován v mobilním zařízení ARCore[167] neboli Google Play Services pro AR.

Následně je možné využívat SceneForm (ARCore) v aplikaci. Lze přiřadit listener OnUpdateListener, který umožňuje spravovat jednotlivé frame snímky před aktualizací scény v metodě onUpdateFrame.

```
//Přiřazení listeneru OnUpdateListener pro Sceneform Scene
arFragment.getArSceneView().getScene().addOnUpdateListener(
                                              this::onUpdateFrame);
```

Následuje proces tvorby rozšířené reality neboli renderování rozšířené reality pomocí SceneForm (ARCore), který by se dal kategorizovat do několika významných bodů:

- *vytvoření potřebných 2D a 3D modelů;*
- *detekce Augmented Images;*
- *finální renderování scény.*

6.10.1.1 Vytvoření 2D a 3D modelů

Pro dosažení požadavků aplikace byl vytvořen jeden 3D model a dva 2D modely, které mají v aplikaci určité funkce. SceneForm umožňuje vytvářet tzv. Renderable[168] modely, které obsahují vše potřebné pro renderování ve scéně.

[164] Umožňují přístup k obrazu kamery.
[165] Aplikace využívá ArFragment pro generalizaci vlastního fragmentu.
[166] Povolení kamery umožňuje využít rozšířenou realitu na daném zařízení.
[167] Umožňuje využít rozšířenou realitu na daném zařízení.
[168] ViewRenderable pro standardní android widgety a ModelRenderable pro 3D modely v různých formátech, které je nutné převést na formáty SceneForm.

Vytvořené jsou tyto modely:

- *2D objekt obsahu detailu* – Objekt reprezentuje informační tabuli.
- *2D objekt obsahu navigace* – Objekt reprezentuje navigační tabuli.
- *3D objekt[169] navigace* – Objekt reprezentuje ukazatel k bodu zájmu.

Modely mohou být následně vytvořeny pomocí ModelRenderable.builder nebo ViewRenderable.builder. V závislosti objektu je vrácen objekt CompletableFuture pro ModelRenderable nebo případně ViewRenderable.

```
… //Ostatní modely
//Vytvoření renderable 2D View
CompletableFuture<ViewRenderable> noticeStage =
    ViewRenderable.builder()
        .setView(this, R.layout.ar_content_notice)
        .build();
//Vytvoření renderable 3D objektu
CompletableFuture<ModelRenderable> arrowStage =
    ModelRenderable.builder()
        .setSource(this, Uri.parse(AppConstants.AR_3D_MODEL))
        .build();
```

Poté je provedeno reaktivní načtení těchto zdrojů. Po dokončení mohou být jednotlivé objekty použity pro ovlivnění scény.

```
CompletableFuture.allOf(
    … //Ostatní Renderable
    arrowStage)
    .handle(
            (notUsed, throwable) -> {
                … //Chybové akce
                try {
                    //Čekání na dokončení CompletableFuture
                    noticeRenderable = noticeStage.get();
                    //Ostatní Renderable
                    //Příznak pro povolení renderování
                    enableRendering = true;
                } catch (Exception e) {
                    … //Chybové akce
                }
                return null;
            });
```

169 Využit byl 3D objekt ukazatele dostupný z: https://clara.io/view/942deb6f-a75a-4d08-bf20-4bd8dad0bcbe.

6.10.1.2 Detekce Augmented Images

Augmented Images spolupracují s obrázky ze skutečného světa neboli ARCore umožňuje určit, kde jsou tyto obrázky fyzicky umístěny v reálném světě.

Pro propojení SceneForm s Augmented Images API je nutné provést inicializaci session v ArFragment. Session je přiřazena databáze[170] referenčních obrázků, které má vyhledávat.

```
protected Config getSessionConfiguration(Session session) {
    //Získání konfigurace ze session
    Config config = super.getSessionConfiguration(session);
    //Získání AugmentedImageDatabase
    AugmentedImageDatabase ARDatabase =
            ARHelper.getAugmentedImagesDatabase(getContext(), session);
    … //Kontrola stavu AugmentedImageDatabase
    //Přiřazení konfigurace session
    config.setAugmentedImageDatabase(ARDatabase);
    return config;
}
```

Následně je možné v metodě onUpdateFrame vyhledávat referenční objekty z aktuálního frame snímku a reagovat na stavy, ve kterých se může referenční objekt nacházet[171].

```
private void onUpdateFrame(FrameTime frameTime) {
    … //Ostatní akce
    //Získání aktuálního frame snímku
    Frame frame = arFragment.getArSceneView().getArFrame();
    … //Kontrola stavu frame a stavu trackování
    … //Kontrola různých příznaků
    //Kontejner AugmentedImage z aktuálního frame
    Collection<AugmentedImage> updatedAugmentedImages =
                    frame.getUpdatedTrackables(AugmentedImage.class);
    //Iterace kontejneru AugmentedImage
    for (AugmentedImage augmentedImage : updatedAugmentedImages) {
        //Výběr stavu trackování obrázku
        switch (augmentedImage.getTrackingState()) {
            … //Stav PAUSED
            case TRACKING:
                … //Ostatní akce pro renderování
                break;
            … //Stav STOPPED
        }
    }
}
```

[170] Databáze referenčních obrázků byla vytvořena v nástroji arcoreimg tool.
[171] Nejdůležitější je stav TRACKING. Detekovatelný objekt lze sledovat a mohou být k němu připojeny kotvy (Anchors). ARCore neustále upravuje informace o okolním světě, proto je potřeba definovat kotvu, která zajistí, že vložený model zůstane i po aktualizaci informací na legitimním místě.

6.10.1.3 Finální renderování scény

SceneForm (ARCore) je založen na konceptu grafu scény neboli uzlů, které jsou umisťovány do vykreslené scény. Vytvořen je rodičovský AugmentedImageNode, na který je možné mapovat další uzly[172] scény.

```
//Vytvoření renderovacího node AugmentedImageNode
AugmentedImageNode node = new AugmentedImageNode(this, actualInterest,
                                                  arFragment);
node.setImage(augmentedImage, noticeRenderable, navigationRenderable,
              arrowRenderable, lastLocation, direction);
//Asociace renderovacího Node s AR scénou
arFragment.getArSceneView().getScene().addChild(node);
```

Rodičovskému uzlu je nutné před umístěním modelu do scény definovat na detekovaném objektu tzv. kotvu. V případě, že je využito Augmented Images lze použít souřadnice středu detekovatelné značky.

```
//Přiřazení anchor (kotvy) z pozice středu obrázku
setAnchor(image.createAnchor(image.getCenterPose()));
```

Následně lze vytvořit další uzel, který je mapovaný na rodičovský uzel AugmentedImageNode. Uzlu je možné přiřadit definovaný model Renderable nebo nad tímto uzlem lze provádět nejrůznější transformace posunu, změny měřítka a další. Této skutečnosti je např. využito v případě navigačního objektu, kdy je uzlu změněna rotace k aktuálnímu bodu zájmu.

```
//Vytvoření Node informační tabule
Node modelTable = new Node();
modelTable.setLocalScale(
                        new Vector3(scaledSize, scaledSize, scaledSize));
modelTable.setLocalRotation(rotation);
modelTable.setParent(this);
modelTable.setRenderable(renderable);
```

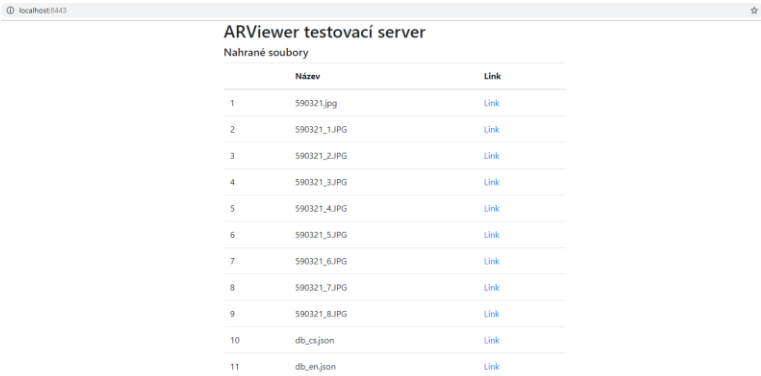
V případě, že se jedná o ViewRenderable může měnit i jeho obsah. Této skutečnosti je využito pro informační notifikování uživatele o vzdálenosti bodu zájmu nebo detailu.

```
//Získání renderable 2D View
View navigationView = navigationRenderable.getView();
//Přiřazení aktuálního jména
TextView interestName =
navigationView.findViewById(R.id.ar_nav_interest_name);
interestName.setText(actualInterest.getName());
```

[172] Tento uzel je vytvořen až v rámci detekovaného Augmented Images objektu.

7 TESTOVÁNÍ

Cílem práce bylo vytvořit aplikaci zaměřenou na navigaci pomocí rozšířené reality. Pro účel testování byl vytvořen testovací server provozovaný pouze v lokální domácí síti, který umožňuje poskytovat body zájmu[173] ve formě JSON souboru a příslušné grafické podklady ve formátu JPG.



localhost:8443

ARViewer testovací server

Nahrané soubory

	Název	Link
1	590321.jpg	Link
2	590321_1.JPG	Link
3	590321_2.JPG	Link
4	590321_3.JPG	Link
5	590321_4.JPG	Link
6	590321_5.JPG	Link
7	590321_6.JPG	Link
8	590321_7.JPG	Link
9	590321_8.JPG	Link
10	db_cs.json	Link
11	db_en.json	Link

Obrázek 38: Testovací server. Zdroj: vlastní.

Aplikace byla testována přímo v terénu, kdy se podařilo ověřit, že navigace k jednotlivým bodům zájmu odpovídá „skutečné“ vzdušné vzdálenosti reálných objektů. Aktuální poloha a orientace nebyla v některých případech zcela přesná vlivem nepřesností senzorů nebo chybou výpočtu orientace, ale pro účel navigace zcela dostačující. Dále občas dochází k zákmitům[174] vložených objektů vlivem nepřesnosti senzorů daného zařízení. Ověřeno také bylo, že ve vzdálenosti 10 metrů dochází ke změně navigace na detailní informace o bodu zájmu.

Obrázek 39: Testování v terénu (pořízeno v obci Blatnice). Zdroj: vlastní.

[173] Informace obsažené v bodech zájmu jsou získány z obecních webových stránek Blatnice dostupné z: http://www.obec-blatnice.cz/ nebo jsou vytvořeny podle uvážení autora.
[174] Zákmity je myšlen neřízený pohyb vložených objektů.

8 UŽIVATELSKÁ PŘÍRUČKA

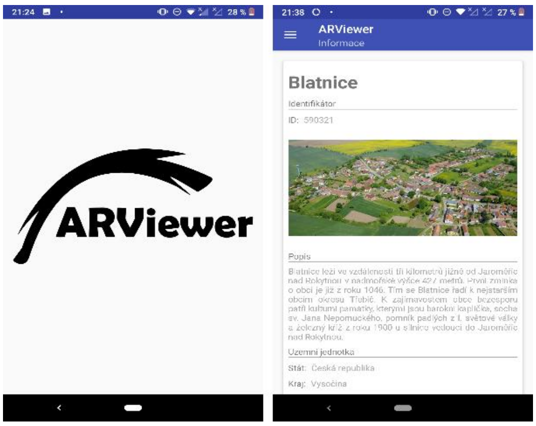
Tato kapitola popisuje aplikaci z uživatelského hlediska z pohledu samotného ovládání[175] aplikace. K využití funkčnosti aplikace je nezbytné využít testovací server[176], který pro potřeby diplomové práce obsahuje veškerá potřebná data.

8.1 Úvodní obrazovka

Při spuštění aplikace je nejprve zobrazena úvodní obrazovka s definovaným časovým zpožděním pro zobrazení vlastního loga aplikace.

8.2 Obrazovka pro informace o assetu „rozcestník aplikace“

Po úspěšném spuštění je uživateli zobrazena obrazovka, která obsahuje informační data z assetu a dále je rozcestníkem neboli tvoří vstupní bod aplikace[177]. Obrazovka umožňuje veškerou správu aplikace přes definované navigační menu.



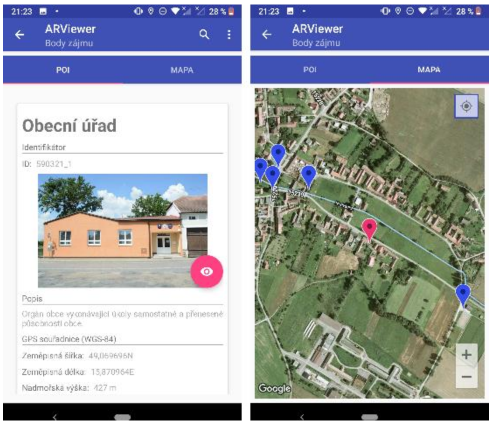
Obrázek 40: Úvodní obrazovka a obrazovka Informace. Zdroj: vlastní.

8.3 Obrazovka bodů zájmu a mapových podkladů

Tato obrazovka zobrazuje body zájmu včetně možnosti filtrace a vyhledávání ve formě seznamu s dodatečnými informace a možností přechodu do samostatné navigace realizovanou přes rozšířenou realitu.

[175] Nutné prerekvizity pro spuštění aplikace byly komentovány v dřívějších kapitolách.
[176] Testovací server stačí pouze spustit a zjistit URL adresu pro komunikaci.
[177] Vždy je proveden návrat do této obrazovky při uživatelském stisku návratu.

Dále obrazovka umožňuje přechod do mapových podkladů, kde je zobrazena aktuální poloha[178] zařízení včetně vybrané množiny bodů zájmu.



Obrázek 41: Obrazovka POI a obrazovka MAPA. Zdroj: vlastní.

8.4 Obrazovka pro rozšířenou realitu

Obrazovka je spuštěna pouze z vybraného bodu zájmu a zajišťuje veškeré akce spojené s rozšířenou realitou. Následně je možné detekovat[179] definovanou značku a rozšířit realitu o definované objekty v závislosti polohy[180] zařízení k poloze vybraného bodu zájmu.

Obrázek 42: Obrazovka „Rozšířená realita“. Zdroj: vlastní.

178 Nutné je mít povolené oprávnění pro polohu zařízení včetně zapnutí této funkce – umožněno je fungovat i bez zapnuté polohy.
179 Nutné je mít povolené oprávnění pro fotoaparát zařízení a oprávnění pro polohu včetně zapnutí této funkce. Další prerekvizitou je také správné natočení vůči detekovatelné značce.
180 Do vzdálenosti větší jak 10 metrů je poskytován objekt navigace, při menší vzdálenosti je zobrazen objekt detailu bodu zájmu.

8.5 Obrazovka pro import assetu

Aplikace umožňuje jednoduše stáhnout[181] z „testovacího“ serveru, libovolný asset s pevně definovanou strukturou ve formě JSON souboru, který je následně uložen do interního úložiště zařízení. Samostatné stažení je provedeno skrze získaná data z nastavení, kde jsou uloženy všechny potřebné informace o serveru.

8.6 Obrazovka pro nastavení aplikace

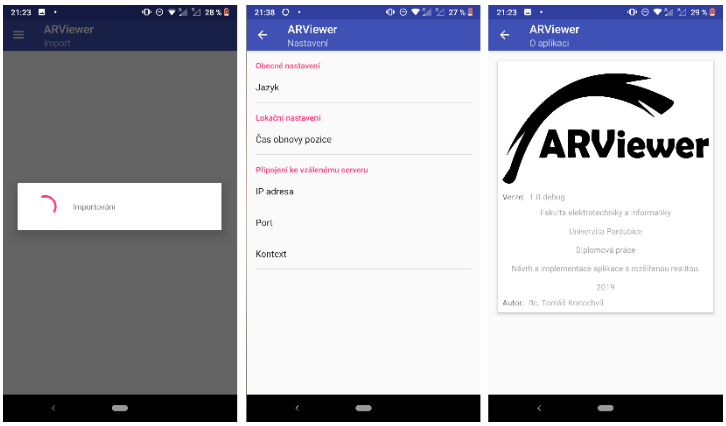
Nastavení aplikace obsahuje tři základní kategorie nastavení. V případě prvotního spuštění je v aplikaci nutné provést jejich nastavení.

Kategorie nastavení:

- *Obecné nastavení* – Nastavení definuje jazykovou lokalizaci aplikace.
- *Lokační nastavení* – Nastavení časové konstanty pro získávání lokace.
- *Připojení ke vzdálenému serveru* – Nastavení URL adresy včetně portu a kontextu serveru.

8.7 Obrazovka pro informace o aplikaci

Aplikace umožňuje získat základní rámec informací týkající se přímo realizované aplikace. Poskytována jsou předem definovaná statická data.



Obrázek 43: Obrazovka Import, Nastavení a O aplikaci. Zdroj: vlastní.

[181] Nutná je internetová (datová) konektivita.

ZÁVĚR

Předložená diplomová práce se skládá ze dvou částí. Cílem teoretické části bylo především seznámení s problematikou rozšířené reality a její využitelnosti nejen na mobilních platformách. Detailněji bylo provedeno seznámení se základním rámcem identifikačních a lokačních technologií, které jsou pro správnou funkci rozšířené reality klíčové, především z pohledu legitimního umisťování digitálních objektů v obrazu reálného světa na displeji mobilního zařízení. Cílem praktické části bylo navrhnout a realizovat mobilní aplikaci pro libovolnou mobilní platformu nebo multiplatformní aplikaci, která bude využívat identifikační a lokační technologie pro realizaci rozšířené reality.

V rámci této práce byla navržena a vytvořena uživatelsky příjemná aplikace ARViewer určena pro mobilní platformu Google Android. Aplikace umožňuje prohlídku místopisných bodů zájmu, jako jsou historicky zajímavá a turisticky atraktivní místa, možnosti vyžití v dané lokalitě, přírodní zajímavosti a další informace, které zastupují jednotlivé body zájmu za pomoci mapového podkladu a rozšířené reality přes vestavěnou kameru daného mobilního zařízení. Vytvořen byl systém pro navigaci k jednotlivým bodům zájmu včetně zobrazení doplňujících informací v závislosti vzdálenosti od bodu zájmu. Výsledná aplikace může tedy sloužit jako navigace nebo i jako turistický průvodce.

Realizovaná aplikace je postavena na architektuře MVP a technik Dependency Injection. Pro vlastní realizaci rozšíření reality na platformě Android byl zvolen framework SceneForm (ARCore), který usnadňuje vytvoření aplikací postavené na rozšířené realitě. Tento framework zahrnuje možnost využít přímo identifikační technologie, přes které lze vkládat legitimně libovolné vytvořené digitální objekty přímo do reálné scény na displeji mobilního zařízení, ale neumožňuje přímé využití lokačních technologií, především z pohledu lokace zařízení, které bylo nutné zkombinovat s tímto frameworkem.

Řešená aplikace by měla umožňovat navigaci k libovolným bodům zájmu, z tohoto důvodu neobsahuje aplikace staticky uložené body zájmu, ale je nutné tyto body naimportovat ze vzdáleného serveru. Protože se nepodařilo zajistit napojení na aplikační server poskytující potřebná data, byl vytvořen testovací server, který tato veškerá potřebná data poskytuje.

V další fázi práce by bylo možné se zaměřit na nasazení aplikace do ostrého provozu. Vytvořena by tak mohla být robustnější síť definovaných bodů zájmu.

Pro zvýšení komfortu by mohla aplikace navíc zahrnovat funkci pro ovládání hlasem. Přidány by mohly být další funkce, jako je navigace k více bodům zájmu současně, čímž by se ale změnil smysl vytvořené aplikace. Přesto ale do budoucna přemýšlím nad touto změnou aplikační logiky, aby mobilní aplikace fungovala jako multifunkční navigace.

Vyvinutá Android aplikace včetně testovacího serveru splnila stanovené cíle a jsou plně funkční pro další vývoj. Přínosem vytvořené aplikace je především nahrazení fyzických rozcestníků neboli směrových ukazatelů a informačních tabulí tisknutelnou značkou v daných lokalitách. Tato aplikace tedy nachází využití především v cestovním ruchu, kdy je uživatel jednoduše a pohodlně informován o geografické poloze všech vybraných bodů zájmu s možností grafické navigace. Tato práce může být také využita potencionálními čtenáři jako návod na použití rozšířené reality na jejich mobilních telefonech, pokud by chtěli, takovou aplikaci zrealizovat.

LITERATURA

[1] KOHOUTOVÁ, Barbora. *Neviditelná vizualita: rozšířená realita v umění a současná mediální ekologie* [online]. 2017 [cit. 2019-08-01]. Dostupné z: https://cemolid.blogspot.com/2017/04/neviditelna-vizualita-rozsirena-realita.html.

[2] KIPPER, Gregory a Joseph RAMPOLLA. *Augmented reality: An Emerging Technologies Guide to AR*. Boston, MA: Syngress/Elsevier, c2013. ISBN 9781597497336.

[3] AZUMA, Ronald T. *A Survey of Augmented Reality* [online]. Teleoperators and Virtual Environments 6, 1997 [cit. 2019-08-07]. Dostupné z: http://www.cs.unc.edu/~azuma/ARpresence.pdf.

[4] GEROIMENKO, Vladimir. Augmented Reality Technology and Art: The Analysis and Visualization of Evolving Conceptual Models. In: *2012 16th International Conference on Information Visualisation* [online]. IEEE, 2012, 2012, s. 445-453 [cit. 2019-08-06]. DOI: 10.1109/IV.2012.77. ISBN 978-1-4673-2260-7. Dostupné z: http://ieeexplore.ieee.org/document/6295852/.

[5] SMITH, Robert E. OPERATIONALIZING BIG DATA. In: *FORTUNA'S CORNER*[online]. 2017 [cit. 2019-08-05]. Dostupné z: https://fortunascorner.com/2017/10/16/operationalizing-big-data/.

[6] MILGRAM, Paul, Haruo TAKEMURA, Akira UTSUMI a Fumio KISHINO. *Augmented Reality: A class of displays on the reality-virtuality continuum* [online]. SPIE Vol. 2351, Telemanipulator and Telepresence Technologies. 1994, s. 282-292 [cit. 2019-08-07]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.6861&rep=rep1&type=pdf.

[7] BIMBER, Oliver a Ramesh RASKAR. *Spatial augmented reality: merging real and virtual worlds*. Wellesley, Mass.: A K Peters, c2005. ISBN 1568812302.

[8] BILLINGHURS, Mark. *COSC 426: Augmented Reality: Lecture3: AR Tracking* [online]. In: . 2012 [cit. 2019-08-07].
Dostupné z: https://www.slideshare.net/marknb00/426-lecture3-ar-tracking.

[9] BILLINGHURST, Mark a Bruce THOMAS. *BEYOND REALITY (2027): THE FUTURE OF VIRTUAL AND AUGMENTED REALITY* [online]. In: . 2017 [cit. 2019-08-06].
Dostupné z: https://www.slideshare.net/marknb00/beyond-reality-2027-the-future-of-virtual-and-augmented-reality.

[10] GLASS: DISCOVER GLASS ENTERPRISE EDITION [online]. 2019 [cit. 2019-08-05].
Dostupné z: https://www.google.com/glass/start/.

[11] HoloLens 2: A new vision for computing [online]. 2019 [cit. 2019-08-05].
Dostupné z: https://www.microsoft.com/en-us/hololens/hardware.

[12] PIXEL: More reasons to love your Pixel phone. In: Google: blog [online]. 2017 [cit. 2019-08-05].
Dostupné z: https://www.blog.google/products/pixel/more-reasons-love-your-pixel-phone/.

[13] Augmented Reality Applications. In: *Archinect* [online]. 2009 [cit. 2019-08-07].
Dostupné z: https://archinect.com/forum/thread/87105/re-augmented-reality-applications.

[14] SMUTNÝ, Vladimír. *Robotika: Úvod do kinematiky* [online]. [cit. 2019-08-07].
Dostupné z: http://cmp.felk.cvut.cz/cmp/courses/ROB/roblec/kinematika-notecz.pdf.

[15] SUYA YOU, U. NEUMANN a R. AZUMA. Hybrid inertial and vision tracking for augmented reality registration. In: *Proceedings IEEE Virtual Reality (Cat. No. 99CB36316)* [online]. IEEE Comput. Soc, 1999, s. 260-267 [cit. 2019-08-07]. DOI: 10.1109/VR.1999.756960. ISBN 0-7695-0093-5.
Dostupné z: http://ieeexplore.ieee.org/document/756960/.

[16] PRAWITA, Fat'hah Noor. *Augmented Reality: Sejarah dan Perkembangan* [online]. In: . 2016 [cit. 2019-08-07].
Dostupné z: https://www.slideshare.net/phatcool/sejarah-dan-perkembangan-augmented-reality.

[17] SINGH, Mona a Munindar P. SINGH. Augmented Reality Interfaces. *IEEE Internet Computing* [online]. 2013, 17(6), 66-70 [cit. 2019-08-06]. DOI: 10.1109/MIC.2013.107. ISSN 1089-7801.
Dostupné z: http://ieeexplore.ieee.org/document/6682933/.

[18] BILLINGHURS, Mark. *COSC 426: Augmented Reality: Lecture3: AR Tracking* [online]. In: . 2013 [cit. 2019-08-07].
Dostupné z: https://www.slideshare.net/marknb00/2013-lecture3-ar-tracking.

[19] PAPAGIANNAKIS, George, Gurminder SINGH a Nadia MAGNENAT-THALMANN. A survey of mobile and wireless technologies for augmented reality systems. *Computer Animation and Virtual Worlds*[online]. 2008, 19(1), 3-22 [cit. 2019-08-07]. DOI: 10.1002/cav.221. ISSN 15464261.
Dostupné z: http://doi.wiley.com/10.1002/cav.221.

[20] KATO, Hiroko, Douglas CHAI a Keng T. TAN. Barcodes for mobile devices. New York: Cambridge University Press, 2010. ISBN 0521888395.

[21] MACDONALD, Alistair. JAVASCRIPT AUGMENTED REALITY. In: *Bocoup* [online]. 2011 [cit. 2019-08-05].
Dostupné z: https://bocoup.com/blog/javascript-augmented-reality.

[22] *QR code* [online]. 2019 [cit. 2019-08-05].
Dostupné z: https://www.qrcode.com/en/.

[23] *Disney Color & Play* [online]. 2019 [cit. 2019-08-05].
Dostupné z: https://www.colorandplaybooks.com/.

[24] ROSENBERG, Martin a Tomáš MERTLÍK. Technologie NFC – popis, bezpečnost a využití. *Elektrorevue* [online]. 2013, 2013(2), 139-146 [cit. 2019-08-07].
Dostupné z: http://www.elektrorevue.cz/cz/download/technologie-nfc---popis--bezpecnost-a-vyuziti/.

[25] *ST25 NFC guide: TN1216 Technical note* [online]. In: . 2016, s. 1-38 [cit. 2019-08-07].
Dostupné z: https://www.st.com/content/ccc/resource/technical/document/technical_note/f9/a8/5a/0f/61/bf/42/29/DM00190233.pdf/files/DM00190233.pdf/jcr:content/translations/en.DM00190233.pdf.

[26] *NFC Forum* [online]. 2019 [cit. 2019-08-05].
Dostupné z: https://nfc-forum.org/.

[27] CRAIG, Alan B. *Understanding augmented reality: concepts and applications*. Amsterdam: Morgan Kaufmann, [2013]. ISBN 9780240824086.

[28] SHARMA, Ashish, MUKESHAGARWAL, Anima SHARMA a PANKHURIDHURIA. *MOTION CAPTURE PROCESS, TECHNIQUES AND APPLICATIONS* [online]. 2013 [cit. 2019-08-07].
Dostupné z: http://www.ijritcc.org/download/IJRITCC_1350.pdf.

[29] SKULA, David. *DATOVÁ FÚZE PRO URČOVÁNÍ ROTACE*. BRNO, 2008. Diplomová práce. Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav automatizace a měřicí techniky.

[30] HÖFLINGER, Fabian, Rui ZHANG, Leonhard M. REINDL, Jörg MÜLLER a Wolfram BURGARD. *A Wireless Micro Inertial Measurement Unit (IMU)* [online]. 2012 [cit. 2019-08-07].
Dostupné z: http://ais.informatik.uni-freiburg.de/publications/papers/hoeflinger13tim.pdf.

[31] The Global Positioning System. In: *GPS* [online]. 2017 [cit. 2019-08-07].
Dostupné z: https://www.gps.gov/cgsic/meetings/2016/wong.pdf.

[32] *Geolocation methods in mobile networks* [online]. In: . 2016 [cit. 2019-08-07].
Dostupné z: https://svs.informatik.uni-hamburg.de/publications/2016/2016-12-27-CCC-Geolocation_methods_in_mobile_networks.pdf.

[33] Ana Roxin, Jaafar Gaber, Maxime Wack, Ahmed Nait Sidi Moh. Survey of Wireless Geolocation Techniques. 50th IEEE Globecom07, Nov 2007, Washington DC, United States. ffhal-00701118f.

[34] VERNER, Lukáš a Dan KOMOSNÝ. Geolokace síťových zařízení v internetových sítích. *Elektrorevue* [online]. 2011, 2011(3), 1-7 [cit. 2019-08-07].
Dostupné z: http://www.elektrorevue.cz/cz/clanky/komunikacni-technologie/0/geolokace-sitovych-zarizeni-v-internetovych-sitich/.

[35] Mobile Operating System Market Share Worldwide. In: *Statcounter* [online]. 2019 [cit. 2019-08-07].
Dostupné z: http://gs.statcounter.com/os-market-share/mobile/worldwide/#yearly-2019-2019-bar.

[36] *Google Developers* [online]. 2019 [cit. 2019-08-03].
Dostupné z: https://developers.google.com.

[37] *Developer* [online]. 2019 [cit. 2019-08-03].
Dostupné z: https://developer.apple.com/.

[38] LACHGAR, Mohamed a Abdelmounaïm ABDALI. *Decision Framework for Mobile Development Methods* [online]. In: . 2017, s. 110-118 [cit. 2019-08-07]. DOI: Decision Framework for Mobile Development Methods.
Dostupné z: https://pdfs.semanticscholar.org/6dfa/5e4ab2140892638f0d67a8db918adde93013.pdf.

[39] VÁCLAVÍK, Jan. [online]. In: . 2015 [cit. 2019-08-07].
Dostupné z: http://janvaclavik.cz/jak-vyvijet-mobilni-aplikace/.

[40] XANTHOPOULOS, Spyros a Stelios XINOGALOS. A comparative analysis of cross-platform development approaches for mobile applications. In: *Proceedings of the 6th Balkan Conference in Informatics on - BCI '13* [online]. New York, New York, USA: ACM Press, 2013, 2013, s. 213- [cit. 2019-08-07]. DOI: 10.1145/2490257.2490292. ISBN 9781450318518.
Dostupné z: http://dl.acm.org/citation.cfm?doid=2490257.2490292.

[41] PECINOVSKÝ, Rudolf. *Myslíme objektově v jazyku Java: kompletní učebnice pro začátečníky*. 2., aktualiz. a rozš. vyd. Praha: Grada, 2009. ISBN 978-80-247-2653-3.

[42] ERLEBACH, Michal. *Tvorba WWW aplikace s využitím relační databáze pro oddíl SK Studenec*. Pardubice, 2011. Bakalářská práce. UNIVERZITA PARDUBICE Fakulta elektrotechniky a informatiky.

[43] *Android Developers* [online]. 2019 [cit. 2019-08-03].
Dostupné z: http://developer.android.com/index.html.

[44] *Dagger* [online]. 2019 [cit. 2019-08-05].
Dostupné z: https://dagger.dev/.

[45] The New Dagger 2 Android Injector. In: *MindOrks* [online]. 2017 [cit. 2019-08-03]. Dostupné z: https://blog.mindorks.com/the-new-dagger-2-android-injector-cbe7d55afa6a.

[46] *ReactiveX* [online]. 2019 [cit. 2019-08-03]. Dostupné z: http://reactivex.io.

[47] *QR Code Generator* [online]. [cit. 2019-08-05]. Dostupné z: https://www.the-qrcode-generator.com/.

OBSAH CD

Obsah přiloženého CD je následující:

- *elektronická verze práce dp_Kratochvil_AR.pdf;*
- *kompletní projekt aplikace ARViewer obsahující veškeré zdrojové kódy;*
- *projekt testovacího serveru včetně souborů pro testování;*
- *spustitelná aplikace ARViewer.*