UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2024 Dominik Lopauer

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Tvorba aplikace pro Android s využitím OpenData

Bakalářská práce

2024 Dominik Lopauer

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Dominik Lopauer**
Osobní číslo:	**I20267**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Tvorba aplikace pro Android s využitím OpenData**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem práce bude vypracovat funkční aplikaci pro OS Android s využitím veřejně dostupných dat. Student získá data z vybraného portálu (např. dopravní data) a ta zobrazí na mapě. V rámci aplikace dojde k vytvoření databáze implementované v mobilním zařízení. Aplikace bude naprogramována v Kotlinu

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

BAYLISS, Darryl, Tom BLANKENSHIP, Fuad KAMAL a Namrata BANDEKAR. *Android apprentice: beginning android developement with Kotlin*. Second edition. [McGaheysville]: Razeware, [2019]. ISBN 978-1-942878-77-3.

SMYTH, Neil. *Android Studio 4.0: development essentials : Kotlin edition*. [místo vydání není známé]: Neil Smyth / Payload Media, [2020]. ISBN 978-1-951442-20-0.

Vedoucí bakalářské práce: **Ing. Jan Panuš, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **16. prosince 2022**
Termín odevzdání bakalářské práce: **12. května 2023**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2023

Prohlašuji:

Práci s názvem tvorba aplikace pro Android s využitím OpenData jsem 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., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, 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.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10. 5. 2024 Dominik Lopauer

PODĚKOVÁNÍ

Děkuji Ing. Janu Panušovi, Ph.D., který mi velmi pomohl s vypracováním této práce. Děkuji také své rodině, která mě podporovala po celou dobu studia.

ANOTACE

Tato bakalářská práce zkoumá vývoj mobilních aplikací v Android Studiu. Hlavní zaměření této práce je na mapy a open data. V práci jsou popsány open data, teoretické principy vývoje včetně UI/UX designu, detailní popis aplikace Android Studio, mapy, open data a rozdíly mezi Javou a Kotlinem. Tyto teoretické zásady jsou využity v implementaci aplikace v programovacím jazyce Kotlin.

KLÍČOVÁ SLOVA

vývoj mobilních aplikací, Java, Kotlin, open data, UI/UX design, programovací jazyk, mapy, implementace aplikace

TITLE

Development of an Android Application utilizing OpenData

ANNOTATION

This Bachelor’s thesis explores the development of mobile applications in Android Studio. The main focus of this thesis is on maps and open data. The thesis describes open data, theoretical principles of development including UI/UX design, detailed description of the Android Studio application, maps, open data, and differences between Java and Kotlin. These theoretical principles are utilized in the implementation of the application in the Kotlin programming language.

KEYWORDS

mobile application development, Java, Kotlin, open data, UI/UX design, programming language, maps, implementation of the application

OBSAH

SEZNAM ILUSTRACÍ A ZDROJOVÝCH KÓDŮ

SEZNAM ZKRATEK A ZNAČEK

XML	Extensible Markup Language
API	Application Programming Interface
SQL	Structured Query Language
ORM	Object-Relational Mapping
IDE	Integrated Development Environment
VCS	Version Control System
JVM	Java Virtual Machine
UI	User Interface
SDK	Software Development Kit

TERMINOLOGIE

Toast	Krátká zpráva na displeji
Overlay	Prvek, který překrývá hlavní obsah
Aktivita	Jedno okno aplikace
Handler	Objekt, který reaguje na událost

ÚVOD

Tato bakalářská práce se zaměřuje na popsání teoretických základů vývoje mobilních aplikací pro systém Android v programu Android Studio. V teoretické části jsou představeny základní pojmy v oblasti programování, open data, UI/UX design, trendy a budoucnost mobilních aplikací, vývoj mobilních aplikací včetně porovnání Javy a Kotlinu a na závěr jsou představeny mapy včetně historie jejich vývoje.

V praktické části je popsána aplikace včetně designu a inspirace z jiných aplikací. Dále je zde podrobně rozepsána konkrétní implementace řešení včetně rozebrání každého adresáře v projektu. Část této práce je také věnována problémům s implementací a jejich řešením. Další kapitolou praktické části je rozebrání funkcionalit aplikace včetně databáze. Poslední část je věnována možným rozšířením aplikace.

Aplikace má otestovat implementaci Open Street Map, načítání open dat a následné přidávání těchto dat do mapy. V této aplikaci je také možnost přidávat vlastní parkovací místa pomocí vybírání bodů na mapě. Při prvním zapnutí aplikace jsou uživateli načtena základní data, která obsahují parkovací místa.

1 TEORETICKÁ ČÁST

1.1 Základní pojmy

1.1.1 Program

Program lze popsat jako sekvenci přesně definovaných kroků, které jsou napsány v programovacím jazyce a jejich cílem je řešení úkolů od jednoduchých matematických výpočtů až po složité aplikace a systémy. Jednotlivé instrukce jsou kompilovány do strojového kódu, který je počítačem snadno pochopitelný.[1]

Samotný počítačový program je možné si představit, jako abstraktní entitu, která získává konkrétní podobu a funkčnost až ve spojení s počítačovým hardwarem. K tomuto stavu dojde až při kompilaci. Kompilace je stav, kdy kompilátor překládá zdrojový kód do strojového kódu, který je vykonatelný počítačem.[1], [5], [18]

Programy mohou být rozděleny na více modulů a knihoven, čímž je umožněno opakované využívání kódu a je usnadněna správa a údržba.[4], [5]

U programů se vyskytují 2 módy. Prvním z nich je interaktivní mód, kdy uživatel může interagovat s aplikací v reálném čase a přijímá vstupy a výstupy. Tento mód je používán, pokud uživatel potřebuje okamžitou zpětnou vazbu. Druhým je dávkový mód, který nepotřebuje interakci v reálném čase. Provádějí se automatické úlohy, dokud nejsou splněny, nebo nenastane chyba. Tento mód je rychlý, efektivní, zvládne velký objem dat a je možné provádět více úloh najednou. Příkladem takové úlohy je například čtení souboru.[2], [3]

1.1.2 Programovací jazyk

Programovací jazyk je soubor syntaktických a sémantických pravidel, která určují, jak psát zdrojový kód programu. Programovací jazyky se dají dělit podle míry abstrakce na nižší a vyšší.[6]

Nižší programovací jazyky se vyznačují tím, že jsou strojově orientované. To znamená, že umožňují přímý přístup k hardwaru počítače a pracují na úrovni procesoru. Programátorovi také umožňují přístup k paměti a registrům procesoru. To umožňuje manipulaci s daty na úrovni bitů, čímž umožňují tvorbu efektivního a optimalizovaného kódu, který je přímo prováděn procesorem. Mezi nižší programovací jazyky se řadí například C, ve kterém je napsán operační systém Linux.[7], [8]

Vyšší programovací jazyky mají větší míru abstrakce, což znamená, že by kód měl být srozumitelnější a měl by se přiblížit se problémům v reálném světě. Vyšší programovací jazyky jsou srozumitelnější a lépe čitelné než nižší. Jsou také méně náchylné k chybám, právě díky abstrakci. Mezi další výhody se řadí rychlejší vývoj, a hlavně snadnější učení. Nevýhodami je nižší výkon a omezená kontrola nad hardwarovými zdroji, protože neumožňují přímý přístup k hardwaru. Mezi nejznámější programovací jazyky se řadí Python, Kotlin, Java, C#, Javascript, Ruby, PHP a další.[9], [10]

Programovací jazyky je dále možno dělit do různých paradigmat. Paradigma představuje základní koncepty, principy a vzory. Pomocí nich určujeme způsob, jakým je kód strukturován a organizován. Existuje několik hlavních paradigmat programování, z nichž každé má své vlastní charakteristiky, přístupy a výhody. Toto dělení nám pomáhá vybrat vhodné programovací jazyky pro konkrétní účely. Základní dělení je mezi procedurálními a neprocedurálními paradigmaty. Jeden jazyk může podporovat více paradigmat, a to i mimo tyto základní kategorie.[6], [9]

Procedurální programování se zaměřuje na sekvenci příkazů, které jsou vykonávány postupně. Hlavním prvkem jsou procedury nebo funkce. Ty definují kroky, které je třeba provést k dosažení požadovaného výsledku. Nejdůležitějšími charakteristikami procedurálního paradigmatu jsou čitelnost kódu, snadná modifikace a možnost znovu používat funkce. Procedurální paradigmata je možno dělit na strukturované a objektově orientované programovaní.[11]

Strukturované programování se zaměřuje na vytváření programů pomocí jasně strukturovaných bloků kódu. Hlavním cílem je zvýšení čitelnosti a spolehlivosti kódu. Hlavní myšlenou je to, že by program měl být rozdělen do menších částí, které jsou snadno pochopitelné. Tyto části jsou vzájemně propojeny pomocí struktur například podmínky, cykly, funkce a metody. Mezi jazyky, které toto paradigma podporují se řadí C, Basic a Pascal.[6], [9], [12],

Objektově orientované programování se vyznačuje používáním objektů. Tyto objekty obsahují vlastnosti a metody. Nejlepším vysvětlením tohoto objektu je například objekt člověk, který má atribut jméno, příjmení, věk a podobně. Vytvořila by se mu také metoda „toString()“, která by vypsala „Dobrý den“. Využívá se dědičnosti a polymorfismu, což je volání metody se stejným jménem, ale s jinou implementací. Dědičnost je možné si představit u objektu „člověk“. Například objekt žák by zdědil všechny atributy, co má člověk. Navíc by se mu vytvořil atribut, který určuje jeho studentské číslo. Dále se mu vytvoří metoda „toString()“, která vypíše „Jsem

student!“. Tato metoda využívá polymorfismus. Mnoho jazyků umožňuje objektově orientované programování, například C#, C++, Java, PHP, Python, Kotlin a mnoho dalších.[10], [13]

Neprocedurální programování se nezaměřuje na specifikaci konkrétních kroků, ale na popis výsledku, který má být dosažen. Plně nechává interpret, nebo kompilátor vybrat kroky k jeho dosažení. Rozděluje se na funkcionální a logické programování.[14]

Funkcionální programování se zaměřuje na vytváření funkcí jako základních bloků programu. Funkce nemění stav programu, nebo okolního prostředí a vrací stejný výstup pro stejný vstup. U funkcionálního programování se často spoléhá na rekurzi a manipulaci s funkcemi vyšší úrovně. Jednoduše řečeno, je to situace, kdy funkce volá sebe sama. Příkladem použití je počítání faktoriálu, kdy je volána metoda factorial(int cislo) pokaždé znovu s hodnotou n-1 a je násobena pomocí n, dokud není dosaženo požadovaného výsledku. Manipulace s funkcemi vyšší úrovně ve zkratce znamená, že může předávat funkce jiným funkcím. Příkladem je vracení funkce jako výsledku. Toto programování nabízejí jazyky FP, Haskell, Miranda a další.[15], [16]

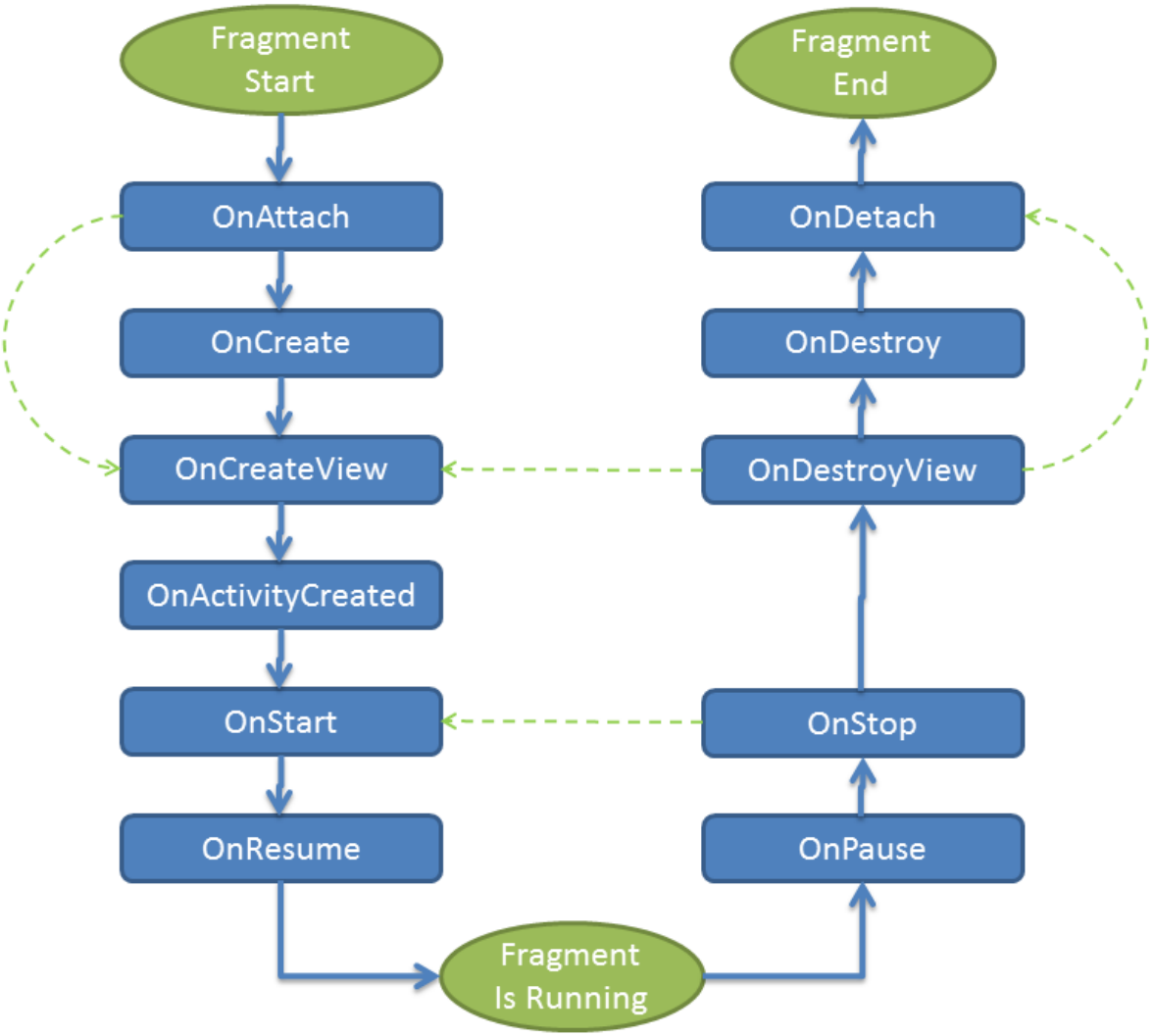
Logické programování je specifické tím, že je založeno na matematické logice. Logické programování se využívá například v oblasti umělé inteligence, protože se zde hledají řešení v databázích. Využívá ho například jazyk Prolog.[17]

1.1.3 Fragment

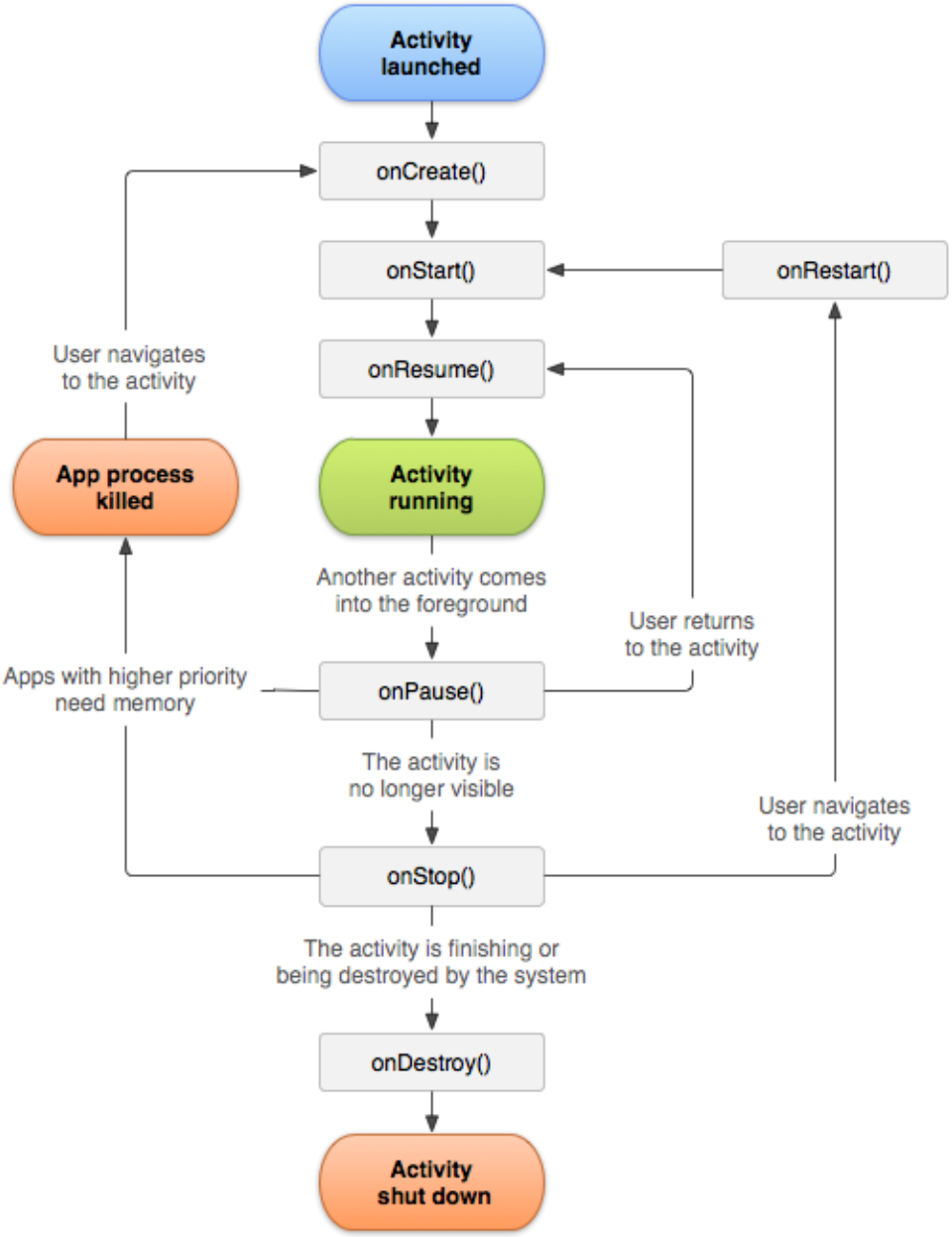
Fragment je samostatná část uživatelského rozhraní, které může být dynamicky zobrazena, nebo skryta uvnitř části aktivity. Fragmenty jsou obvykle používány na rozdělení uživatelského rozhraní na menší znovupoužitelné části.[20],[21]

Existují 2 možné způsobí použití. Prvním je pevné přidání do aplikace pomocí XML kódu. Druhá možnost je snazší, protože využívá Kotlin případně Javu a přidává se až za běhu programu.[20]

Ačkoliv jsou fragmenty podobné aktivitám, mají jiný životní cyklus. Na rozdíl od aktivit, mají mimo funkcí onCreate(), onStart(), onResume(), onPause(), onStop() a onDestroy() další funkce, pomocí kterých můžeme interagovat s aplikací. Jsou to metody onAttach(), onCreateView(), onViewCreated(), onActivityCreated(), onDestroyView(), onDetach().[21]



Obrázek 1: Životní cyklus fragmentu (Zdroj: 20)



Obrázek 2: Životní cyklus aktivity (Zdroj 19)

Nejčastější je u fragmentu použití funkcí onCreateView(), onCreate() a onActivityCreated(). Všechny 3 funkce se spouštějí při startu aplikace. OnCreate() je spuštěna ve chvíli, kdy se vytváří fragment. Používá se pro inicializaci dat fragmentu. OnCreateView() se aktivuje, když se vytváří uživatelské rozhraní fragmentu. Nastavují se zde prvky UI. OnActivityCreated() se zavolá po vytvoření aktivity, ve které je fragment umístěn. Když je tato metoda aktivována, znamená to, že je aktivita úspěšně vytvořena a inicializována a můžeme k ní přistupovat.[21]

Motivací pro použití fragmentů je hlavně přizpůsobení uživatelského rozhraní různým zařízením a různým velikostem obrazovek. To zaručuje jednoduché vytvoření responzivity.

Dalším důvodem je možnost znovu používat části uživatelského rozhraní a logiky, což zvyšuje modularitu kódu.[21]

1.1.4 Relační databáze

Databáze je základem většiny moderních informačních systémů, které umožňují efektivní správu, ukládání a manipulaci s rozsáhlým množstvím dat. Tato část bude podrobněji zaměřena na relační databáze, které jsou jedny z nejrozšířenějších typů databázových systémů používaných v dnešní době.

Relační databáze představují strukturovaný způsob ukládání dat, kde jsou informace organizovány do tabulek skládajících se z řádků a sloupců. Každý sloupec tabulky reprezentuje určitou vlastnost nebo atribut a každý řádek obsahuje konkrétní hodnoty těchto atributů. V databázích jsou klíčovými prvky klíče, které definují vztahy mezi různými tabulkami. Díky těmto klíčům je možné spolu tabulky propojovat a sdílet data mezi nimi.[22], [24], [25]

Manipulace s daty se v databázích provádí pomocí dotazovacího jazyka, který se nazývá SQL. Tento jazyk nám umožňuje provádění různých operací. Tyto operace se označují jako CRUD a řadí se sem tyto metody:[23]

- Create

  Tato metoda je používána pro vytvoření tabulky v databázi. Při vytváření se přidávají názvy sloupců a jejich datový typ (Char, int, double a další).[26]

- Read

  Pro tuto operaci se používá operace SELECT, která vybírá konkrétní záznamy z vybrané tabulky. Příkladem použití je například SELECT * from Studenti, která vybere veškeré záznamy z tabulky „Studenti“.[27]

- Update

  Operace UPDATE upravuje již existující záznamy v tabulce. Používá se s přidáním podmínky WHERE, protože bez ní by se upravovaly všechny záznamy v tabulce, což může být nežádoucí. [28] Příkladem použití je příkaz UPDATE Studenti SET Name=“Dominik“ WHERE st=64521. Použitím tohoto příkazu je upraveno jméno u studenta, který má st64521 na jméno Dominik.

- Delete

  Při operaci DELETE se opět používá podmínka WHERE, a to ze stejného důvodu.[29] Příkaz může vypadat následovně: DELETE FROM Studenti WHERE st=64521. Tento příkaz smaže studenta, který má st64521 z tabulky.

Interakce s aplikacemi je dalším důležitým bodem pří používání databáze. Aplikace komunikují s relačními databázemi prostřednictvím API, které poskytuje rozhraní pro provádění operací s daty. Mezi tyto operace se řadí vytváření, čtení, aktualizace a mazání záznamů. ORM nástroje umožňují aplikacím pracovat s daty v databázi pomocí objektů namísto SQL dotazů, což tuto práci zjednodušuje.[30]

1.1.5 Framework

Framework je strukturovaný soubor nástrojů, knihoven, pravidel a standardů, který umožňuje programátorům rychleji a efektivněji vytvářet aplikace a řešit různé druhy problémů. Tento soubor poskytuje prostředky a metodiky, které poskytují hotové komponenty. Framework je možné si představit jako kostru, na které je postaven zbytek kódu.[31]

Frameworky jsou vytvořeny pro různé potřeby. Dají se používat na webové aplikace, frontend, backend a mnoho dalšího.[31] Dále je nutné zmínit frameworky pro vývoj mobilních aplikací. Tyto frameworky představují některé z nejlepších nástrojů pro vývoj aplikací v jazyce Kotlin:

- Ktor:

  Ktor je framework pro tvorbu webových služeb a API v jazyce Kotlin.[32]

- Koin:

  Koin je framework pro správu závislostí v jazyce Kotlin. Nabízí jednoduché způsoby definice a vkládání závislostí do aplikace. Koin je navržen tak, aby byl jednoduchý k použití a minimalizoval zbytečné kódy a konfigurace.[33]

- Exposed:

  Exposed je framework pro práci s relačními databázemi v jazyce Kotlin. Poskytuje deklarativní způsob definice databázových schémat, dotazování dat a manipulaci s databázovými záznamy v Kotlinu.[34]

1.2 OpenData

Open data nabízejí nevyčerpatelný zdroj informací, které jsou volně dostupné, snadno zpracovatelné a jsou poskytované ve strojově čitelném formátu.[35] Open data existují v různých formách a typech například hospodářství, regiony, životní prostředí, doprava a mnoho dalšího.[36]

Následující část bude popisovat analýzu Open dat a budou představena základní význam jejich využívání v oblasti vývoje. Bude zkoumáno, jak využít dostupné zdroje, formáty dat a jejich vliv na funkcionalitu.

1.2.1 Definice

Open data jsou informace, které jsou veřejně dostupné a volně použitelné bez omezení autorských práv a licencí. Tato data jsou poskytována různými organizacemi, vládami, městy a dalšími. Mohou zahrnovat různorodé informace, například statistiky, geografická data, zdravotnické informace, parkovací místa a mnoho dalších věcí.[35]

1.2.2 Zdroje

Zdroje otevřených dat jsou různé. Řadí se mezi ně vládní portály, webové stránky, neziskové organizace a další. Data jsou většinou poskytnuta ve strojově čitelných formátech, jako jsou CSV, JSON, XML, GeoJson.[36] Velké množství dat je dostupné přímo na webových stránkách, například opendata.praha.eu, avšak mnohdy se stává, že tato data jsou dostupné pouze pod API k čemuž bývá potřeba registrace.[37]

1.2.3 Význam v mobilních aplikacích

Otevřená data mají význam i v mobilních aplikacích, protože umožňují programátorům integrovat aktuální a relevantní informace do aplikací. Mobilní aplikace mohou využívat různorodá data pro zobrazení počasí, dopravních informací, kulturních událostní, polohových informací a mnoho dalšího. Takto mohou mobilní aplikace poskytovat užitečné služby a informace uživatelům a zlepšit jejich uživatelský zážitek.[38]

1.3 UI/UX design

Návrh uživatelského rozhraní (UI) a uživatelského zážitku (UX) je klíčovým při vývoji úspěšných a efektivních mobilních aplikací. Dnes je kladen velký důraz na jednoduchou použitelnost aplikace. „In a survey by Qualtrics, 80% of the 1000+ surveyed customers claimed that they value customer experience above all other aspects of the technological products and services they purchase.“[39] Z tohoto textu vyplívá, že 80% uživatelů si vybere aplikaci podle dobrého uživatelského zážitku.

Tato kapitola bude určena klíčovým principům UI/UX designu. Důraz bude kladen na jejich aplikaci v mobilních aplikacích. Budou probrány estetické a vizuální aspekty, a hlavně funkčnost a efektivita.

1.3.1 Definice

UI/UX design zahrnuje komplexní proces navrhování a vytváření uživatelského rozhraní. Začíná se vciťováním do uživatelova pohledu, zkoumáním základním požadavků a zajištěním nejlepšího řešení. Dále se definuje, co uživatel potřebuje v aplikaci, pokračuje se vymýšlením různých řešení, ze kterých se pak vybírají ty, které nejvíce vyhovují. Poté se tato řešení testují a vítězné řešení je nakonec použito.[40], [42]

1.3.2 Estetika a Vizuální design

Součástí UI/UX designu je také vizuální design, který se zabývá estetikou. Je potřeba stále dbát na to, že aplikace se uživatelům musí i líbit. Design zahrnuje volbu barev, typografií, ikon, obrázků a dalších prvků, které vytvářejí jedinečný vizuální styl aplikace. Při designování aplikace je dobré prohledávat konkurenci nejen jako inspiraci, ale i pro vylepšení stávajících chyb, kterých se konkurence dopustila. Je dobré udělat si wireframe, který nám pomůže hlavně u UX. Android studio má funkci na zobrazení pouze wireframe našich prvků na obrazovce. Když je hotový, stačí už jen aplikaci doladit vhodnými barevnými styly.[39], [41], [42]

1.4 Trendy a budoucnost mobilních aplikací

Trendy a budoucnost mobilních aplikací představují klíčovou oblast zkoumání, kterou je výhodné odhadnout. Tato kapitola se blíže zaměřuje na zkoumání současných trendů. S rychlým pokrokem technologie a stále častějším používám mobilních zařízení je důležité znát trendy a budoucnost technologického průmyslu.

Tato část bude analyzovat klíčové trendy a nové technologie, které ovlivňují mobilní vývoj, včetně umělé inteligence, rozšířené reality, blockchainu a edge computingu.

1.4.1 Umělá inteligence v mobilních aplikacích

AI se stále více používá v mobilních aplikacích a její vliv se rozšiřuje do různých oblastí. Mezi tyto oblasti patří například bankovnictví, zdravotnictví a další. Implementace AI do mobilních aplikací přináší mnoho výhod, například možnost detekce anomálií v chování uživatelů a poskytování vyšší úroveň ochrany.[43] Samozřejmostí je i automatické odpovídání v reálném čase. Bezpečnostní metody, jako například rozpoznávání obličeje se dají vylepšit pomocí AI.[44]

1.4.2 Rozšířená a virtuální realita

Rozšířená realita (AR) a virtuální realita (VR) se stávají stále populárnějšími a očekává se, že tento trend bude nadále postupovat. AR umožňuje propojení digitálního obsahu s reálným světem a přináší vylepšené zážitky, jako například tréninkové aplikace, simulace a další. Oproti tomu VR nabízí uživatelům čistě virtuální prostředí.[45] Dá se říct, že vytvářejí jiný svět, kde je možné cestovat, vzdělávat se a vlastně vše ostatní co v reálném světě. Podniky, které se rozhodnou do těchto technologií investovat, mohou získat konkurenční výhodu na trhu.[46]

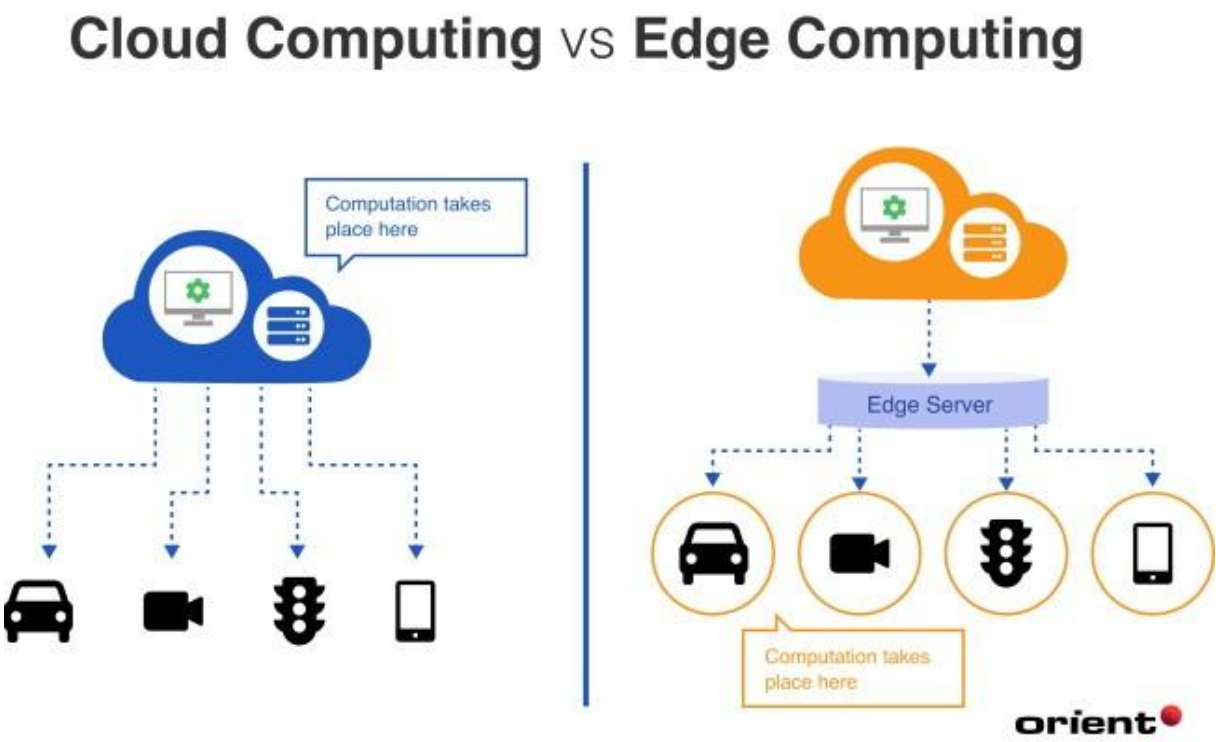
1.4.3 Blockchain a bezpečnost mobilních aplikací

Blockchain je známý převážně díky kryptoměnám (převážně Bitcoinu). V současné době, kdy je kladen důraz na bezpečnost, start-upy tuto technologii využívají k zajištění bezpečnosti důležitých uživatelských dat, ale i kvůli urychlení transakcí. Blockchain je nyní vnímán jako univerzální databáze, která zajišťuje transparentnost transakcí. Implementace blockchainu do aplikace usnadňuje digitální transakce a zvyšuje spolehlivost, což je pro start-upy velkou výhodou v boji o zákazníky. V mobilním vývoji může být blockchain využit pro zabezpečení platebních systémů, ověření identity uživatelů a vytváření decentralizovaných aplikací, které jsou odolné vůči manipulaci a podvodům. Tyto trendy naznačují, že blockchain bude hrát stále významnější roli v oblasti bezpečnosti a důvěry v mobilních aplikacích.[47]

1.4.4 Edge computing

Edge computing, jak už název napovídá, se zabývá hranovým výpočtem, který přenáší výpočetní funkčnost přímo k uživatelům. Cílem je zpracovávat data produkovaná a spotřebovávaná co nejblíže zdroji, nejlépe v reálném čase. Tento přístup snižuje zatížení sítě a zlepšuje výkon aplikací.[48] V porovnání s cloud computingem, kde všechny procesy běží na cloudu, edge computing distribuuje výpočetní sílu do koncového zařízení a minimalizuje komunikaci mezi klientem a serverem, čímž nejen zrychluje, ale hlavně zajišťuje větší odolnost proti útokům. Navzdory tomu není edge computing konkurencí cloudu, ale spíše jeho vylepšením, které využívá cloudové výhody a přispívá k efektivitě.[49]

Edge computing také zajišťuje efektivní využívání zdrojů i v podmínkách špatné, nebo dokonce žádné konektivity k internetu. Toto naznačuje, že edge computing bude hrát stále důležitější roli v oblasti mobilních aplikací.[48]



Obrázek 3: Cloud a Edge computing (Zdroj 48)

1.5 Vývoj mobilních aplikací

Při vytváření aplikací pro Android je důležité promyslet technické i designové řešení, které bude uživatelům vyhovovat. Aby aplikace byla konkurenceschopná, je důležité chápat principy vývoje a vybrat vhodný program. První část se zaměřuje na základy vývoje aplikací pro systém Android. Budou zde popsány i na návrhové vzory používané při vývoji mobilních aplikací a poslední část bude věnována životnímu cyklu aplikací.

1.5.1 Software

Pro vývoj mobilních aplikací existuje celá řada softwarů a platforem, které umožňují vytvářet aplikace pro různé operační systémy. Pro vývoj na systém iOS se řadí Xcode a pro Android je to Android Studio. Mezi další oblíbené programy patří Flutter, React Native a Swiftic. Tyto nástroje poskytují prostředí pro vývoj, testování a ladění aplikací. Nabízejí také širokou škálu dalších funkcí a nástrojů pro efektivní vývoj mobilních aplikací.[50]

Android Studio je oficiální IDE pro vývoj aplikací pro platformu Android. Bylo představené společností Google v roce 2013. Poskytuje kompletní sadu nástrojů pro programátory. Mezi hlavní prvky patří editor kódu, emulátor mobilního zařízení, nástroje pro návrh uživatelského rozhraní a integrované nástroje pro testování a ladění aplikace. Android Studio je založeno na IntelliJ IDEA a je optimalizováno na vývoj aplikací pro Android.[51]

Xcode je IDE vyvinuté společností Apple pro vývoj aplikací pro platformu iOS. Je to hlavní nástroj pro programátory, kteří chtěli vytvářet aplikace pro iPhone, iPad, Apple Watch a další zařízení vytvoření společností Apple. Xcode poskytuje prostředí pro psaní kódu ve 2 jazycích. Jsou jimi Swift a Objective-C. Vizuálně je Xcode podobný Android Studiu a obsahuje všechny nástroje zmíněné u Android Studia.[52]

Výhod použití těchto specializovaných vývojových softwarů je více. Jedním z hlavních důvodu je již připravené optimalizované prostředí pro konkrétní platformu. Vývojáři mají jednodušší přístup k dokumentaci, což je také velkou výhodou. V novějších verzích těchto programů se dokonce vyskytuje i umělá inteligence, kterou je při vytváření aplikací možné využívat.[51],[52]

1.5.2 Návrhové vzory

Návrhové vzory jsou důležité pro vytvoření uživatelsky přívětivých a intuitivních uživatelských rozhraní. Jejich správné použití přináší mnoho výhod a zajišťuje konzistentní vzhled a chování aplikace pro všechny zařízení a platformy.

Nejčastěji používané návrhové vzory jsou Material Design pro aplikace na systém Android a Human Interface Guidelines pro aplikace na systém iOS. Tyto vzory definují soubor pravidel a doporučení pro používání barev, typografií, ikon, navigací a dalších prvků, které přispívají k jednotnému a konzistentnímu vzhledu aplikace a zlepšují uživatelskou zkušenost.[53], [54]

Material Design, který je vyvinutý společností Google, klade důraz na realistické stínování, pohybování a barevné vrstvy.[53] Human Interface Guidelines jsou oproti tomu vytvořené společností Apple a definují standardy pro design uživatelského rozhraní. Jejich cílem je zajištění konzistentní a intuitivní uživatelské zkušenosti na iOS.[54]

Výhod při používání návrhových vzorů je více. Zajišťují konzistentní vzhled a chování aplikací, což zvyšuje uživatelovu důvěru a snižuje šanci na zmatení uživatele. Dále umožňují jednodušší vývoj aplikace, protože jsou zde poskytnuta hotová řešení pro běžné designové problémy. Také se snaží zjednodušit uživatelskou zkušenost tím, že usnadňují orientaci v aplikaci.[55]

Celkově lze říct, že používat návrhové vzory je pro nás výhodné. Jejich správné použití totiž vede k vytvoření aplikace, které vizuálně uživatele osloví, ale také mu pomůže s orientací v aplikaci.

1.5.3 Životní cyklus aplikací

Životní cyklus aplikace představuje dynamický proces, který popisuje různé fáze, kterými aplikace prochází od spuštění až po ukončení. Tento cyklus zajišťuje správné fungování aplikace, a ta pomocí něho může reagovat na různé události a situace. Životní cyklus se skládá z těchto částí:

- Inicializace aplikace:

  Tato fáze označuje počáteční inicializaci aplikace, kdy jsou načteny všechny potřebné zdroje a prostředky a spouští se hlavní komponenty aplikace. Mezi tyto komponenty patří například aktivity, služby a fragmenty. Během této fáze můžeme provádět inicializační operace (načítání konfigurace a nastavení globálních proměnných).

- Spuštění:

  Fáze spuštění se spustí ve chvíli, kdy je zobrazeno hlavní uživatelské rozhraní, ale uživatel s ním stále nemůže interagovat. Tato fáze bývá rychlá.

- Interakce s uživatelem:

  Zde je hlavní část životního cyklu. Uživatel aktivně interaguje s aplikací a provádí různé operace, na které aplikace reaguje. Během této fáze jsou uživateli zobrazovány obrazovky a okna, zpracovávají se vstupy a aktualizuje se stav aplikace podle akcí uživatele.

- Pozastavení:

  Pokud je aplikace překryta jinou aplikací, nebo je shozena na pozadí, dostane se aplikace do tohoto bodu. Během této fáze může aplikace ukládat svůj stav a pozastavovat provádění některých operací z důvodu šetření baterie.

- Obnovení:

  Když je aplikace znovu aktivována, neboli je vrácena na obrazovku, přejde se do této fáze, kdy aplikace obnovuje svůj stav a pokračuje v provádění operací.

- Ukončení:

  Pokud je aplikace ukončena, přechází do této fáze, kde uvolňuje používané zdroje a provádí závěrečné úkony. Těmito úkony zpravidla bývá ukládání stavu, nebo mazání dočasných souborů.[19]

1.6 Android studio

Android Studio představuje IDE vyvinuté přímo pro tvorbu mobilních aplikací pro platformu Android. Toto prostředí poskytuje komplexní sadu nástrojů, které pomáhají spravovat projekt, editovat zdrojový kód, navrhovat uživatelské rozhraní, ladit, testovat a nasazovat.

Android Studio je postaveno na platformě IntelliJ Idea a nabízí programátorům klasické funkce, jako je zvýrazňování syntaxe, automatické doplňování kódu, refaktoring, debugging, integraci s VCS a mnoho dalšího. Aplikace také disponuje vlastním emulátorem a kartami pro designování aplikace.[51]

1.6.1 Základy vývoje pro Android

Android Studio představuje nezbytný nástroj pro programátory, kteří chtějí vytvářet aplikace pro operační systém Android. Toto vývojové prostředí poskytuje široké spektrum nástrojů a funkcí, které usnadňují každou část vývoje aplikace.

Základním krokem je vytvoření nového projektu a správná konfigurace aplikace. Při vytváření projektu má programátor na výběr i verzi Androidu a nastavení základních vlastností projektu. Programátor má i možnost vybrat si aktivitu, která může mít předdefinované fragmenty nebo funkce. Příkladem je aktivita, která má připravenou Google mapu, do které stačí jen přidat API klíč a je plně funkční bez dalšího nastavení.

Po vytvoření projektu je programátorovi k dispozici panel nástrojů, editor kódu, správce projektu a další užitečné panely. Důležité jsou i jiné nástroje a funkce, a to například správa závislostí a knihoven pomocí Gradle, což je nástroj na automatizaci.

1.6.2 Uživatelské rozhraní a interakce

Uživatelské rozhraní Android Studia je navrženo tak, aby programátorům poskytlo prostředí, které je intuitivní, efektivní a podporuje jejich potřeby při vývoji aplikací pro Android.

- Hlavní okno:

  Po spuštění Android Studia se uživateli zobrazí hlavní okno, které obsahuje panel nástrojů, menu a hlavní pracovní prostor.

- Panel nástrojů:

  Panel nástrojů poskytuje rychlý přístup k důležitým funkcím a nástrojům IDE. Jsou zde projekty, verze kódu, návrhy rozložení, konzole a další.

- Editor kódu:

  Hlavní pracovní prostor obsahuje editor kódu, kde programátoři píšou, upravují, a hlavně debugují svůj kód.

- Návrhový editor:

  Pro tvorbu uživatelského rozhraní je k dispozici návrhový editor, který umožňuje vytvářet rozložení obrazovek pomocí přetahování prvků UI, jako jsou tlačítka, textová pole, obrázky a další.

- Správce projektů:

  Android Studio zahrnuje správce projektů, který umožňuje uživatelům prohlížet, organizovat a spravovat soubory projektu, včetně zdrojových souborů, knihoven, obrázků a dalších.

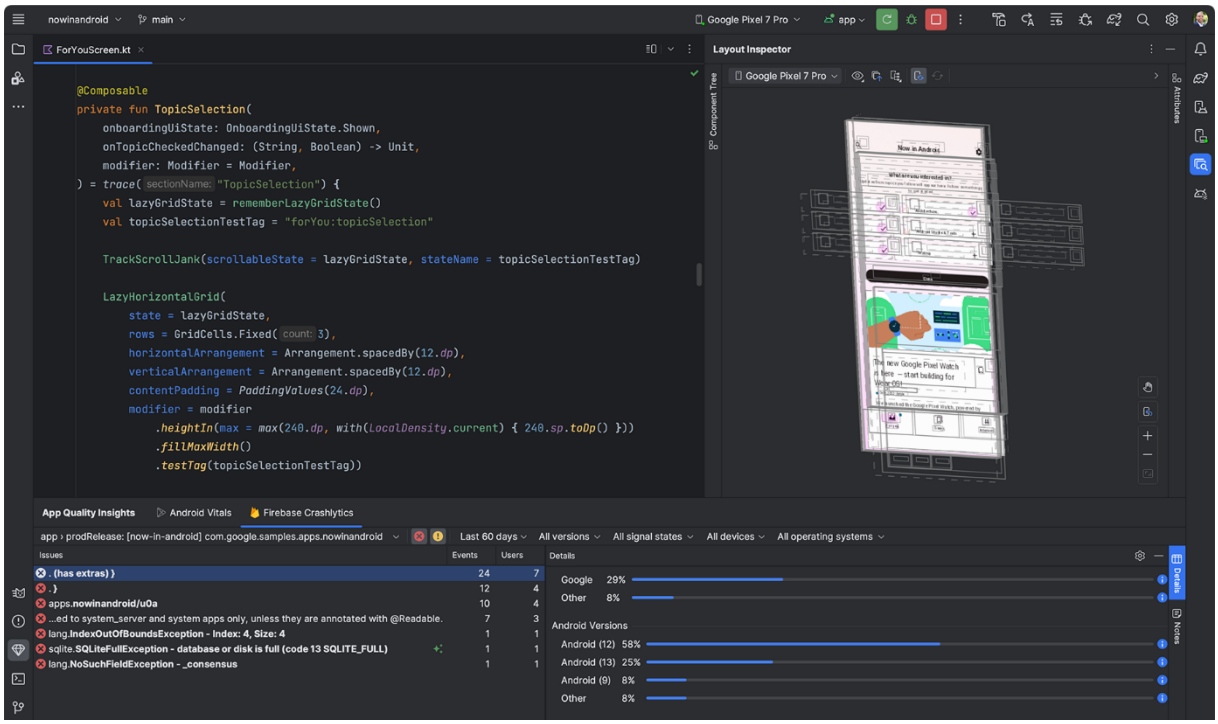
- Integrace s emulátorem a zařízeními:

  Uživatelé mohou spouštět a testovat své aplikace pomocí vestavěného emulátoru nebo připojit fyzická zařízení k vývojovému počítači a testovat své aplikace v reálném prostředí.

- Nástroje pro ladění a debugování:

  Android Studio poskytuje širokou škálu nástrojů pro ladění a debugování aplikací, včetně sledování proměnných, zásobníků volání, logování událostí a dalších.

Celkově Android Studio nabízí uživatelům komplexní a integrované prostředí pro vývoj aplikací pro platformu Android, které podporuje jejich potřeby od návrhu a psaní kódu až po testování a ladění aplikací.[51]



Obrázek 4: Android Studio (Zdroj 51)

1.6.3 Java a Kotlin

Java je dlouho používaným programovacím jazykem pro vývoj mobilních aplikací pro platformu Android. Její dlouhá historie a široká podpora od nástrojů a knihoven z ní dělají stabilního hráče ve vývoji mobilních aplikací. Java je spolehlivý programovací jazyk, který umožňuje vývojářům vytvářet mobilní aplikace se širokým spektrem funkcí a možností.

Kotlin je moderní programovací jazyk pro platformu JVM a byl oficiálně přijat jako podporovaný jazyk pro vývoj aplikací pro Android. Díky jeho přehledné syntaxi, snadné čitelnosti kódu a bezpečnější konstrukci nabízí nové možnosti a přínosy. Kotlin přináší moderní funkce, které usnadňují programátorům práci.[56], [58] Příkladem této funkce je například „range“ funkce. Zde je příklad kódu: 4 in 1..4. Tento kód vrátí „true“, protože je to dotaz, jestli tento rozsah obsahuje číslo 4.[57] Kotlin je kompatibilní s Javou, proto je programátorům umožněno přecházení z existujícího kódu, který je v Javě na nový kód v Kotlinu. Android Studio má dokonce funkci, pomocí které můžeme převést kód v Javě do Kotlinu. Rozhodnutí by mělo záviset na konkrétních potřebách a preferencích projektu a vývojářů. Zatímco některé projekty mohou nadále preferovat Javu kvůli její stabilitě, jiné by mohly využít výhody Kotlinu ve formě snadnějšího a bezpečnějšího vývoje. Ale dá se předpokládat, že Kotlin bude v budoucnu v této oblasti převládat.[56]

1.7 Mapy

Mapy jsou nenahraditelným prvkem mnoha moderních mobilních aplikací. Usnadňují uživatelům navigaci, zobrazení okolních míst a získání prostorového kontextu. Jejich využití není omezeno pouze na mapové aplikace, ale rozšiřuje se i do různých oblastí, jako jsou cestování, sociální sítě a další. Mapy jsou schopné uživatelům poskytnout přesnou polohu, polohu přátel, informace o dopravě a další funkce.[38]

Integrace map do mobilních aplikací vyžaduje vhodný výběr mapových technologií a API, které umožní přizpůsobit mapové funkce potřebám aplikace. Důležitou součástí je také návrh uživatelského rozhraní, které umožňuje uživatelům snadnou a intuitivní interakci s funkcemi a daty. Efektivní využití map přispívá k lepší uživatelské zkušenosti a zvyšuje oblíbenost aplikace. Zároveň musíme brát v úvahu i různé aspekty, jako jsou náklady na použití těchto API a ochranu soukromí uživatelů. Díky mapám se mobilní aplikace stávají užitečnějšími a atraktivnějšími pro uživatele.[38], [59], [60]

1.7.1 Význam map v mobilních aplikacích

V mobilních aplikacích mají mapy klíčový význam, protože umožňují uživatelům navigovat a získávat prostorový kontext. Jsou důležité hlavně pro mapové navigace, aplikace pro cestování a podobně. Mapy poskytují uživatelům informace o aktuální poloze, trasách, okolních zajímavých místech a mnoho dalšího.[60]

1.7.2 Technologie mapových služeb

Pro programátora je důležité mít přístup k vhodným mapám, které odpovídají jeho cílům a potřebám. Existuje více populárních API a platforem, které poskytují mnoho funkcí a možností pro implementaci map do mobilních aplikací. Nejznámější je pravděpodobně Google Maps API, které nabízí pokročilé funkce, jako je geokódování, vyhledávání tras a interaktivní mapové prvky. Dalšími možnostmi jsou například Mapbox, OpenStreetMap a další.[59], [60] Každé API má jiné ceny. Obvykle jsem se setkal s tím, že jsou mapy placené buď podle počtu uživatelů, které tuto aplikaci používají, nebo podle provedených úkonů na mapě, jako je například přiblížení, posouvání a další.

1.7.3 Historie

Jednou z prvních mapových aplikací byla MapQuest Mobile, která byla uvedena na trh v roce 2006. Tato aplikace, ačkoli nabízela základní funkce, jako bylo vyhledávání a navigace, byla velmi omezena oproti dnešním mapám.[61] Průlom v oblasti mobilních mapových aplikací přineslo uvedení aplikace Google Maps pro iPhone v roce 2007. Tato aplikace, vytvořená

společností Google, přinesla výrazné vylepšení v oblasti mapování na mobilních zařízeních. Díky interaktivním mapám, vyhledávání míst a navigaci prostřednictvím GPS se stala velmi populární.[62]

2 EXPERIMENTÁLNÍ ČÁST

2.1 Popis aplikace

2.1.1 Účel aplikace

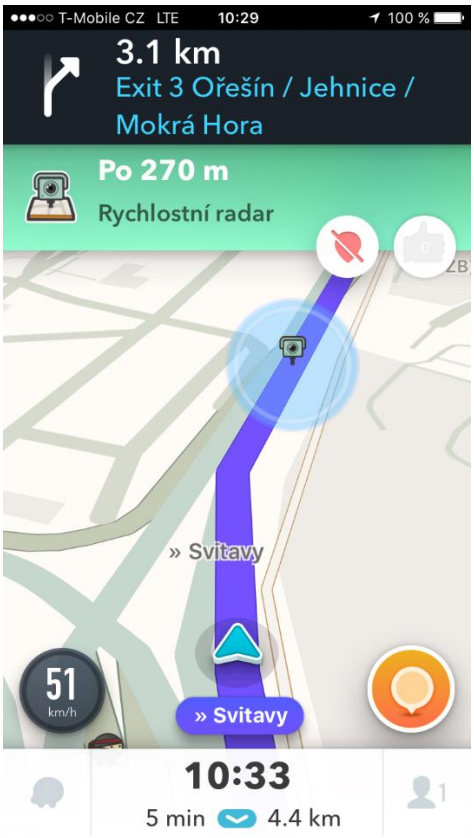
Aplikace zobrazuje uživateli parkovací místa na mapě, která si může libovolně pojmenovat. Každý uživatel má svoji lokální databázi, do které jsou při prvním spuštění nahrána data základních parkovacích míst. Uživatel může přidávat vlastní parkovací místa pomocí vybírání bodů přímo na mapě. Při vytváření parkovacího místa se objeví menu, ve kterém jsou 3 možnosti: vytvořit parkovací místo, smazání posledního bodu, smazání všech vytvořených bodů. Při vytváření je nutné mít alespoň 3 body, jinak není možné místo vytvořit. Dále se dají importovat další parkoviště ze souboru ve formátu GeoJson a také lze smazat všechna parkovací místa.

2.1.2 Inspirace

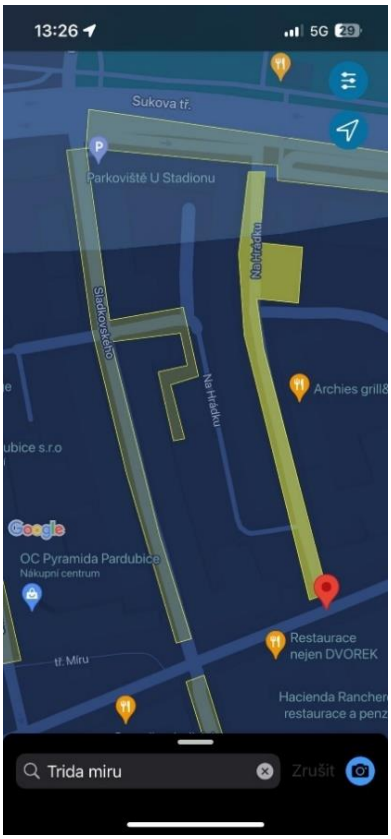
Při tvorbě návrhu a designu aplikace bylo intenzivně čerpáno inspirace z aplikace „Waze“. Tato aplikace se stala pro mě jakýmsi průvodcem v procesu tvorby uživatelsky orientovaného rozhraní. Zvláště bylo zaměřeno na umístění spodních ovládacích prvků, protože jejich rozložení bylo vnímáno jako efektivní a intuitivní pro uživatele.

Dalším významným zdrojem inspirace byla aplikace „MPLA“, která se specializuje na zjednodušení parkování. Inspirace pro tuto práci byla zejména z implementací polygonů pro vizualizaci dostupných parkovacích míst. Tento koncept byl následně upraven pro potřeby projektu a byl integrován do designu, aby uživatelé měli snadný přehled o volných parkovacích místech v okolí.

Nicméně, cílem práce nebylo pouze pouhé opisování existujících řešení. Důraz byl kladen na jejich důkladné zkoumání a následnou úpravu podle konkrétních potřeb a cílů aplikace. Cíle bylo zajistit, aby uživatelé mohli používat aplikaci s komfortem a efektivitou, a proto bylo vybráno a kombinováno těch nejlepších prvků a konceptů z různých zdrojů, aby bylo dosaženo co nejlepší uživatelské zkušenosti.



Obrázek 5-a: aplikace Waze (Zdroj 63)



Obrázek 5-b: aplikace MPLA (Zdroj 64)

2.1.3 Design

Na obrázku 6-a je vidět, jak vypadá aplikace ihned po startu aplikace. Pokud je povoleno sdílení polohy, uvidíme postavičku, která tuto polohu zobrazuje. V dolní části obrazovky jsou 2 ikony: hlavní menu a přidání parkovacího místa.

Při kliknutí na tlačítko pro přidaní parkovací místa vyjíždí ze shora nabídka na vytváření polygonů a Toast informace o minimálním počtu bodů pro vytvoření parkovacího místa, která je zobrazena na obrázku 6-b.

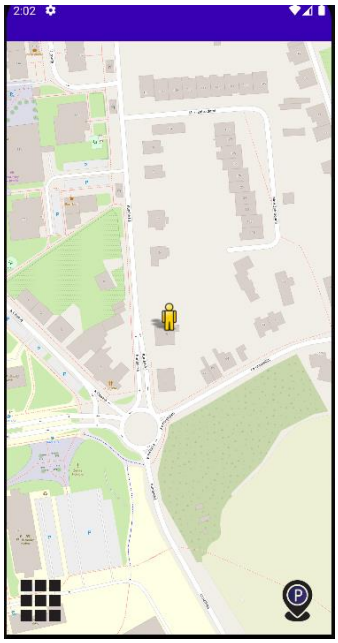
Na obrázku 6-c je menu hlavní nabídky obsahující 2 tlačítka, které vyjíždí ze zdola. Ikona menu se při kliknutí změní na křížek sloužící k zavření menu. Také není možné otevřít obě tyto nabídky současně, proto je zde kontrolováno, zda je toto tlačítko aktivní a pokud ano, spustí animaci zavření. Oba obdélníky, které jsou u každého menu mají zaoblené rohy a nastavený modrofialový gradient. Tento barevný motiv je použit pro celou aplikaci.

```
val blue=0xFF222354.toInt()
val purple= 0xFF3A2B72.toInt()
val gradientDrawable = GradientDrawable()
gradientDrawable.shape = GradientDrawable.RECTANGLE
gradientDrawable.cornerRadius = 32f
gradientDrawable.colors = intArrayOf(blue,purple)
gradientDrawable.orientation = GradientDrawable.Orientation.LEFT_RIGHT
rectangle.background = gradientDrawable
```

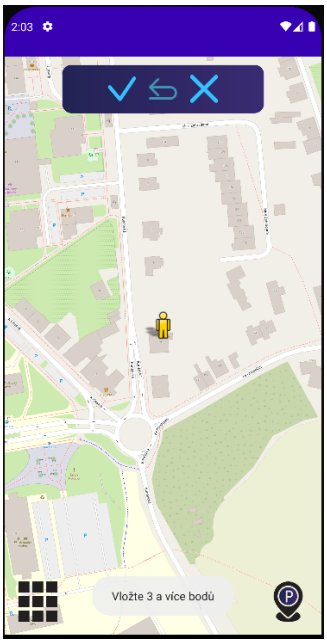
Zdrojový kód 1: Nastavení obdélníkového menu (Zdroj vlastní)

Pro aktivitu na import parkovacích míst je zvolen klasický Android design, který je intuitivní a jednoduchý viz. obrázek 6-d. Uživatel má možnost vybrat soubory, které jsou uloženy lokálně, ale i soubory, které má na cloudu.

Při načítání aktivit, kterým trvá načtení delší dobu (importování parkovišť, zjišťování polohy) je používáná načítací obrazovka, která je zobrazena na obrázku 6-e.

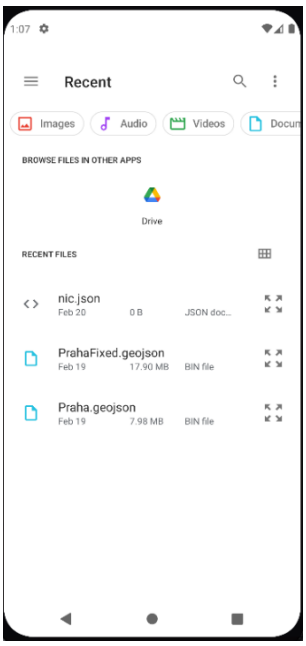


Obrázek 6-a (Zdroj vlastní)

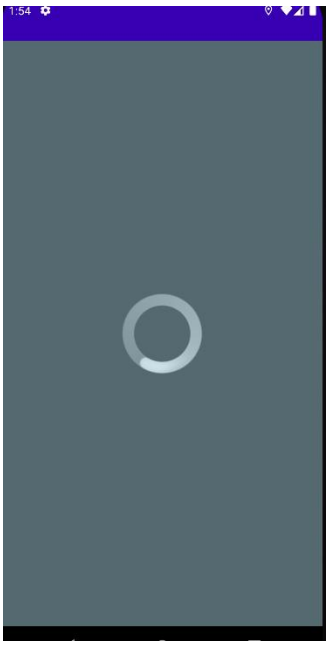


Obrázek 6-b (Zdroj vlastní)

Obrázek 6-c (Zdroj vlastní)



Obrázek 6-d (Zdroj vlastní)



Obrázek 6-e (Zdroj vlastní)

2.2 Implementace

2.2.1 Programovací jazyk a framework

Pro vývoj této práce byl zvolen programovací jazyk Kotlin ve spojení s frameworkem Android SDK, který je poskytovaný společností Google. Tato kombinace byla vybrána kvůli svému modernímu přístupu k vývoji, vysoké produktivitě a práci s nástroji pro vývoj aplikací pro Android.

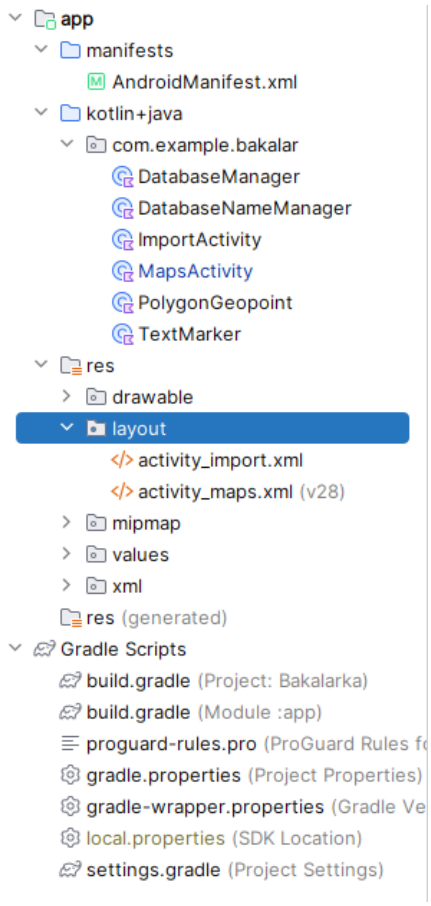
Programovací jazyk Kotlin nabízí řadu výhod, jako je čitelnější kód, snadná syntaxe a kompatibilita s Javou. Díky těmto vlastnostem je vývoj v Kotlinu plynulejší a efektivnější. Kotlin také umožňuje psát méně kódu, a přitom dosáhnout stejné funkcionality jako Java.

Framework Android SDK poskytuje kompletní sadu nástrojů, knihoven a API pro vývoj aplikací pro Android. Tyto nástroje jsou napsány v jazyce Java, ale díky kompatibilitě Kotlinu jsou tyto nástroje dostupné. Navíc Kotlin umožňuje jednoduchou integraci s existujícím Java kódem, což usnadňuje migraci již existujících aplikací na Kotlin.

Jednou z hlavních výhod Kotlinu a Android SDK je jejich integrace s Android Studiem. Android Studio poskytuje podporu pro vývoj v Kotlinu, včetně automatických importů, refaktoringu a dalších funkcí. Díky tomu je vývoj v Kotlinu v aplikaci Android Studio velmi efektivní.

Možností byl i vývoj v Javě s použitím Android SDK, ale nakonec jsem se rozhodl pro Kotlin, kvůli modernímu přístupu k vývoji a vysoké efektivitě, kterou Kotlin a Android Studio poskytují. Navíc Kotlin je již oficiálně podporován společností Google pro vývoj aplikací pro Android, což zajišťuje dlouhodobou udržitelnost a kompatibilitu s budoucími verzemi Androidu.[56]

2.2.2 Struktura kódu



Obrázek 7: Struktura kódu (Zdroj 51)

V Android Studiu jsou soubory rozděleny do složek manifests, kotlin + java a res. Dále se zde nacházejí Gradle scripty a konfigurační soubory, které nastavují implementace a konfiguraci celé aplikace.

2.2.2.1 Manifests

Ve složce manifests je soubor AndroidManifest.xml, který obsahuje deklarativní popis všech komponent aplikace. Deklarativní popis je způsob popisu, ve kterém je specifikováno, co má být provedeno, aniž by se specifikoval přesný postup. Pouze se popisují stavy, vlastnosti, chování a nástroje. Také jsou zde uvedeny povolení, která jsou v aplikaci potřebné. V aplikaci je to například internet, nebo zapisování a čtení souborů. V ukázce kódu je zobrazeno základní nastavení ikony a vytvoření aktivity na import souborů.

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.Bakalarka"
    tools:targetApi="31">
    <activity
        android:name="com.example.bakalar.ImportActivity"
        android:exported="false">
        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
    </activity>
```

Zdrojový kód 2: Ukázka AndroidManifest.xml souboru (Zdroj vlastní)

2.2.2.2 Kotlin+Java

V této složce se nachází podsložka s názvem aplikace (v mém případě „bakalar“). V podsložce jsou umístěny všechny zdrojové soubory napsané v jazyce Kotlin a Java. V Android studiu je možné psát zároveň v Javě i Kotlinu, a to dokonce souběžně. Android Studio umí i automaticky převádět kód z jednoho jazyku do druhého. V aplikaci je použit pouze Kotlin, a to z důvodu lepší přehlednosti.[56] Třídy se dají rozdělit do dalších balíčků, například pro pomocné třídy. Tato možnost není v práci použita, protože je v ní méně zdrojových souborů. Při malém počtu tříd je struktura stále dobře čitelná, tudíž není důvod přidávat další balíčky.

2.2.2.3 Res

Zde se nachází více podsložek. Složka drawable, která obsahuje obrázky pro aplikaci. Na obrázku 7 jsou skryty z důvodu lepší přehlednosti. Tyto obrázky mohou být operačním systémem upravovány a komprimovány.

Layout obsahuje jednotlivé rozložení stránek. Obvyklé je, že pro každou aktivitu, která má unikátní rozložení existuje vlastní layout. Tyto soubory jsou ve formátu XML a poskytují přehled o tom, jak jsou na stránce rozloženy objekty. Objekty je možné přidávat i přímo z kódu a nemusí být zapsané v layoutu, ale z důvodu přehlednosti je lepší je zapsat. Jednotlivé layout soubory nám zobrazují i náhled rozložení stránky.[65] Tento náhled se neaktualizuje, pokud něco změníme v kódu, tudíž je lepší měnit většinu věcí zde, aby měl programátor představu o aplikaci už při prohlížení layoutů. V uvedeném kódu 3 je zobrazeno nastavení tlačítka, které se nachází v menu pro vytváření parkoviště.

```
<ImageButton
    android:id="@+id/undo"
    android:layout_width="35dp"
    android:layout_height="35dp"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:background="@null"
    android:scaleType="fitCenter"
    android:src="@drawable/back" />
```

Zdrojový kód 3: Nastavení ImageButton (Zdroj vlastní)

Složka mipmap obsahuje ikony aplikace, do které ale žádné ikony nebyly přidány, protože jsem žádnou nevytvořil. Složka by měla obsahovat ikony v různých rozlišeních. Tyto soubory nikdy nemohou být upraveny operačním systémem, proto mají svojí vlastní složku a nejsou spojeny s drawable, které upravované být mohou.[66]

Values obsahuje jednotlivé stringové zdroje, které je lepší používat například u popisu obrázků. Jsou zde specifikované i barvy a barevná schémata aplikace. Důležitou kartou jsou styly, kde se dají definovat vzhledy a rozložení aplikace.

Poslední složkou je xml, která může obsahovat různé konfigurační a datové soubory, případně layouty. V této práci zde žádný soubor není, avšak vhodným vylepšením aplikace by mohlo být přidání automatické zálohy při pádu aplikace.[67]

2.2.2.4 Gradle scripts

Build.gradle(Project) obsahuje základní implementaci Androidu a Kotlinu. Dá se zde nastavit verze Androidu i Kotlinu, kterou chceme používat.

Build.gradle(Module) implementuje jednotlivé soubory a nastavuje jejich konfiguraci. Zde se také nastavuje namespace aplikace. Důležitý je zde i výběr SDK a JVM verze.

Local.properties obsahuje lokální proměnné, například sdk.dir, která určuje, kde se nachází SDK. Při využívání Google map by se zde vytvářela proměnná s klíčem k API, která je vytvořena pod uživatelským účtem na Googlu. Tímto je účet propojen s mapami a Google má informace o využívání map a může je zpoplatnit podle skutečně použitých služeb. Open Street Mapy nic takového nevyužívají, protože jsou zcela zdarma.

2.2.3 Využité knihovny a nástroje

Mezi nejdůležitější knihovny, které byly implementovány se řadí tyto:

```
implementation 'com.google.android.gms:play-services-location:21.2.0'
implementation 'org.osmdroid:osmdroid-android:6.1.18'
implementation 'com.github.MKergall:osmbonuspack:6.9.0'
implementation 'org.osmdroid:osmdroid-shape:6.1.3'
```

Zdrojový kód 4: Implementace knihoven (Zdroj vlastní)

Knihovna com.google.android.gms:play-services-location je součástí služeb Google Play a poskytuje rozhraní pro získávání polohových informací v aplikacích pro Android. Tato knihovna poskytuje i širokou škálu funkcí pro práci s polohovými daty, mezi které patří i Fused Location Agent, která vrací aktuální polohu uživatele.[68]

Knihovna org.osmdroid:osmdroid-android:6.1.18 implementuje mapu a nástroje pro práci s mapou v aplikacích pro Android. Mezi hlavní služby knihovny se řadí zobrazování mapových dlaždic, interakce s mapou, přizpůsobování mapy a podpora overlayů.[69]

Další implementace byla na knihovnu com.github.MKergall:osmbonuspack:6.9.0, která přidává bonusové ovládací prvky Open Street Map. Mezi hlavní prvky patří zobrazování trasy, hledání významných bodů na mapě, zobrazování informací o trase, podpora vícevrstvé mapy a zobrazování značek na mapě.[70] Z těchto prvků byla využita jediná věc, a to sice vícevrstvá mapa, protože bylo potřeba vkládat na mapu parkoviště.

Poslední import org.osmdroid:osmdroid-shape:6.1.3 zajišťuje možnost vytváření tvarů a jejich zobrazování. Dovoluje také interaktivní manipulaci s těmito tvary. Přidává i možnost vytvoření nové proměnné GeoPoint, která obsahuje zeměpisnou výšku a šířku.[71]

2.2.4 Problémy s implementací

2.2.4.1 Implementace mapy

Původním plánem bylo použití Google Map, z důvodu jednoduché implementace (v Android Studiu přímo existuje předdefinovaná aktivita s již nainstalovanou mapou). Problém nastal při používání této mapy, protože určité funkce jsou zpoplatněny. Proto byla vybrána Open Street mapa, která je zcela zdarma. K základním mapám, bylo potřeba importovat „OSM Bonus pack“, který obsahuje polygony, markery a overlaye, které jsou v aplikaci potřeba. Při použití Google Map by se nic takového řešit nemuselo, protože všechny výše jmenované věci, jsou již implementovány v základu.[72]

Dalším problémem bylo povolení „internetu“, protože bez tohoto povolení by se nestáhla mapa. Stahování mapy funguje přes jednotlivé tiles, které se pak skládají vedle sebe a vytvářejí mapu.

```
<uses-permission android:name="android.permission.INTERNET" />
```

Zdrojový kód 5: Povolení použití internet (Zdroj vlastní)

Po přidání povolení se mapa zobrazila, ale byla zobrazena vícekrát vedle sebe. Tato chyba byla způsobena špatným nastavením limitů posouvání. Tyto limity byly nastaveny na zeměpisnou šířka a výšku. Nastaveno bylo i maximální a minimální přiblížení mapy na hodnoty 20 a 4.

```
map.maxZoomLevel = 20.0
map.minZoomLevel = 4.0
map.setScrollableAreaLimitLatitude(
    MapView.getTileSystem().maxLatitude,
    MapView.getTileSystem().minLatitude,
    0
)
map.setScrollableAreaLimitLongitude(
    MapView.getTileSystem().minLongitude,
    MapView.getTileSystem().maxLongitude,
    0
)
```

Zdrojový kód 6: Nastavení limitů mapy (Zdroj vlastní)

2.2.4.2 Implementace tlačítek

Při vytváření obrázkových tlačítek nastal problém se zobrazováním. Kvůli špatnému typu, který měla mapa nastavený základně, se tlačítka nezobrazovala. Řešením byla změna z LinearLayout na RelativeLayout. Linear layout umožňuje jednotlivé prvky zarovnávat v řadě, nebo sloupci. Toto bylo využito při zarovnávání tlačítek vedle sebe, ale pro zarovnání mapy a tlačítek byl použit Relative layout, který umožňuje zarovnávat relativně k jiným prvkům a rodičům.[73]

2.2.4.3 Jména polygonů

Největším problémem, který se v této aplikaci objevil, bylo pojmenování polygonů. Bylo předpokládáno, že aplikace bude mít předdefinovanou možnost nastavit jméno, ale bohužel měla jen možnost přidat polygonu popis, který se zobrazí při kliknutí na vybrané parkoviště. Při kliknutí je nastavený vlastní handler, který otevře dialog pro editaci jména. Z tohoto důvodu není možné popis zobrazit.

```
polygon.setOnClickListener { _, _, _ ->
    val input = EditText(map.context)

    val alertDialog = AlertDialog.Builder(map.context)
        .setTitle("Název Polygonu")
        .setView(input)
        .setPositiveButton("OK") { _, _ ->
            val enteredText = input.text.toString()
            databaseNameManager.insertPolygon(polygon.id.toInt(),
enteredText)
            map.overlays.removeAll { it is TextMarker }
            loadMarks()
            map.invalidate()
        }
        .setNegativeButton("Zrušit", null)
        .create()


    alertDialog.show()
    false
}
```

Zdrojový kód 7: Vytvoření handleru pro polygon (Zdroj vlastní)

Pro vytvoření textu bylo prvotním plánem použití obyčejného markeru, který sice má možnost nastavení textIcon, ale velikost textu nejde změnit, protože to není implementováno v této třídě. Proto byla vytvořena vlastní třída TextMarker, která dědí ze třídy Marker a přidává možnost nastavení textu.

```
class TextMarker(private val mapView: MapView, text: String, center:
GeoPoint) : Marker(mapView) {
    private var markerText: String = text
    private val markerTextSize: Float = 44f
```

Zdrojový kód 8: Vytvoření třídy TextMarker (Zdroj vlastní)

2.2.4.4 Ukládání polygonů do databáze

Při ukládání polygonů do databáze bylo potřeba převádět data do jiných tříd a funkcí. Bohužel třída Geopoint neobsahuje žádnou proměnnou, kterou by bylo možné použít pro id, proto bylo vhodné vytvořit jednoduchou třídu na převod zeměpisné výšky, šířky a id polygonu s názvem PolygonGeopoint. Tato třída pomáhá převádět proměnné, a proto není potřeba je převádět postupně, což může vytvořit méně čitelný kód.

```
class PolygonGeopoint(val latitude:Double,val longitude:Double,val
polygonId:Int)
```

Zdrojový kód 9: Vlastní třída pro body Polygonu (Zdroj vlastní)

2.2.4.5 Načítání polygonů ze souboru

Dialog pro načítání souboru má podle dokumentace možnost zjistit MIM typ a podle toho určit možnost vybrání určitých souborů. To znamená, že by nemělo být možné vybrat například

obrázek, pokud chceme vybírat pouze Geojson soubory. Při tomto nastavení bohužel není možné vybrat žádný soubor. Tudíž bylo nutné toto ověření doprogramovat.

2.3 Funkcionality aplikace

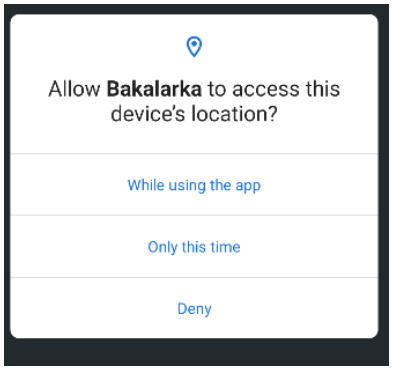
2.3.1 Hlavní funkce

Aplikace při startu zavolá funkci createDb(), která zkontroluje, zda již databáze existuje a pokud ne, tak načte základní data aplikace. Prvotní plán byl přidat tlačítka na určitá města a uživatel by si vybral, ve kterém městě chce načíst parkovací místa. Z důvodu uživatelské přívětivosti byl zvolen tento způsob, aby uživatel nemuset při prvním spuštění vybírat města. Také bylo zvažováno, kdy načítat tato města, protože uživatel může chtít mít například prázdnou mapu. Tudíž byla zvolena možnost, která data načte pouze při neexistující databázi (pouze první spuštění).

```
private fun createDb() {
    val databaseFile =
File(this@MapsActivity.getDatabasePath(DatabaseManager.DATABASE_NAME).pa
th)
    databaseManager = DatabaseManager.getInstance(this@MapsActivity)
    databaseNameManager =
DatabaseNameManager.getInstance(this@MapsActivity)
    if (!databaseFile.exists()) {
        val files = arrayOf(
            File(this@MapsActivity.filesDir,
"startData/parkingFirst.geojson"),
            File(this@MapsActivity.filesDir,
"startData/parkingSecond.geojson")
        )
        for (file in files) readGeoJson(Uri.fromFile(file))
    }
}
```

Zdrojový kód 10: Funkce kontrolující existenci databáze (Zdroj vlastní)

Dále se při každém spuštění zkontroluje, zda má uživatel povoleno sdílení polohy. Pokud má nastavenou možnost „vždy se zeptat“ je mu zobrazen dialog vyobrazený na obrázku 8.



Obrázek 8: Dialog pro zjištění polohy (Zdroj vlastní)

Uživatel může mít sdílení polohy vypnuté a v té chvíli mu je zobrazen dialog, který mu při kliknutí „povolit“ otevře nastavení se sdílením polohy viz obrázek 9. Pokud uživatel klikne na zrušit, je dialogové okno zavřeno a je mu zobrazena mapa a zbytek aplikace je funkční, protože poloha není vyžadována pro správné fungování celé aplikace.

```
if (ContextCompat.checkSelfPermission(
        this,
        Manifest.permission.ACCESS_FINE_LOCATION
    ) == PackageManager.PERMISSION_GRANTED
) {
    fusedLocationClient.getCurrentLocation(
        Priority.PRIORITY_HIGH_ACCURACY,
        object : CancellationToken() {
            override fun onCanceledRequested(p0:
OnTokenCanceledListener) =
                CancellationTokenSource().token

            override fun isCancellationRequested() = false
        }
    ).addOnSuccessListener { location: Location? ->
        if (location != null) {
```

Zdrojový kód 11: Zjišťování aktuální polohy uživatele (Zdroj vlastní)

Obrázek 9: Dialog pro přechod do nastavení (Zdroj vlastní)

Když je zjišťování polohy povoleno je proveden následující kód, který na souřadnice aktuální polohy přidává ikonu aktuální polohy (obrázek 6-a).

```
val actualPosition = GeoPoint(location.latitude, location.longitude)
map.controller.setCenter(actualPosition)
val marker = Marker(map)
marker.position = actualPosition
marker.icon = ContextCompat.getDrawable(
    this@MapsActivity,
    org.osmdroid.library.R.drawable.person
)
marker.setAnchor(Marker.ANCHOR_CENTER, Marker.ANCHOR_BOTTOM)
map.overlays.add(marker)
```

Zdrojový kód 12: Přidání ikony na aktuální pozici (Zdroj vlastní)

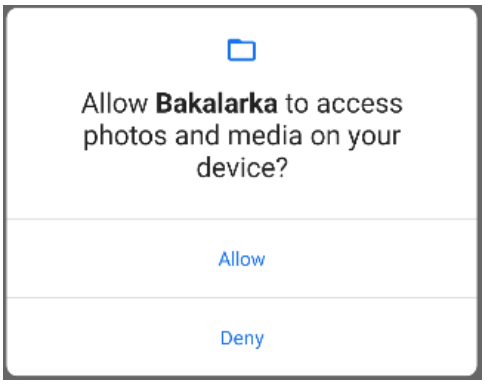
Když se poloha nastavuje a mapa se načítá, jsou uživateli veškeré funkce zablokovány a je mu zobrazena pouze načítací obrazovka (obrázek 6-e), která je řešena skrýváním a zobrazováním overlaye.

```
private fun initViews() {
    loadingOverlay = findViewById(R.id.loadingOverlay)
    progressBar = findViewById(R.id.progressBar)
}
private fun showLoadingOverlay() {
    loadingOverlay.visibility = View.VISIBLE
}
private fun hideLoadingOverlay() {
loadingOverlay.visibility = View.GONE
}
```

Zdrojový kód 13: Zapnutí načítacího overlaye (Zdroj vlastní)

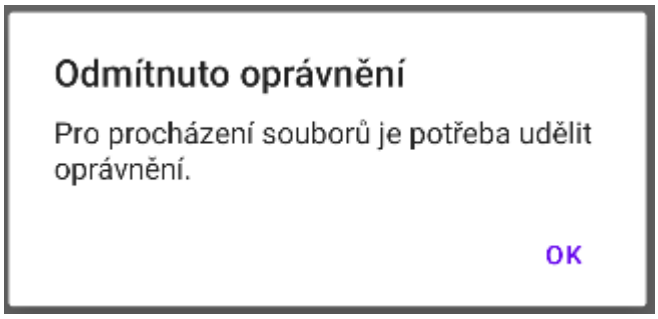
Po načtení je uživateli zobrazena mapa a může používat veškeré funkce aplikace. Pokud má povoleno zjišťování polohy, mapa je přiblížena na místo, kde se uživatel nachází (obrázek 6- a). Pokud má zjišťování vypnuto, mapa zůstává na základním bodě, který se nachází na zeměpisné šířce 0 a zeměpisné výšce 0.

Uživatel má také možnost vytvářet parkovací místa. Tato parkovací místa může vytvářet dvěma způsoby. Prvním je importování dat pomocí souboru ve formátu GeoJson. Uživatel je při prvním spuštění pomocí dialogu dotázán na přístup k souborům.



Obrázek 10: Dialog pro povolení přístupu k souborům (Zdroj vlastní)

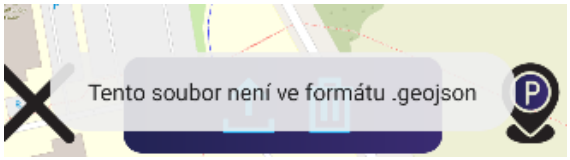
Pokud uživatel souhlasí, je mu povolen přístup k aktivitě pro výběr dat, avšak pokud nesouhlasí, je mu zobrazen následující dialog.



Obrázek 11: Dialog při zakázání přístupu (Zdroj vlastní)

Byl vybrán jiný druh dialogu, protože povolení k souborům je nezbytné pro importování souborů. Při kliknutí na „ok“ je uživatel přesunut do nastavení povolení a může zde oprávnění změnit. Připadným řešením by bylo stejné dialogové okno, jako u polohy (obrázek 9), kdy by tlačítko „zrušit“ vracelo uživatele na aktivitu s mapou.

Dále je potřeba řešit, zda uživatel vybere správný soubor, protože aktivita počítá pouze se vstupem ve formátu geoJson. Uživatel může vybrat i jiné typy souboru, ale dostane chybovou hlášku ve formě Toast.



Obrázek 12: Toast s chybou (Zdroj vlastní)

Toto ověření je řešeno pomocí jednoduché funkce, která načte začátek datového souboru a kontroluje, zda obsahuje typické prvky pro geoJson soubor. Pokud obsahuje alespoň jeden prvek, je pravděpodobné, že se jedná o geoJson.

```
return fileContent.contains("\"type\": \"FeatureCollection\"") ||
        fileContent.contains("\"type\": \"Feature\"") ||
        fileContent.contains("\"type\": \"Polygon\"") ||
        fileContent.contains("\"type\": \"Point\"")
```

Zdrojový kód 14: Kontrola Geojson formátu (Zdroj vlastní)

Při vkládání parkovacích míst je také samozřejmě kontrolováno, zda už se nenachází v databázi. Hlavně při vkládání pomocí souboru totiž hrozí vložení více parkovacích míst na stejné místo, to s sebou nese 2 problémy. Prvním je to, že uživatel zbytečně zaplňuje místo parkovištěm, které se už na mapě vyskytuje a druhým je to, že se stejné parkovací místo na mapě vloží na to předchozí. Zde nastává problém, protože polygon má nastavenou alfu na hodnotu 72 a když se 2 místa přidají přes sebe bude barva na tomto místě tmavší, než na ostatních místech. Tento jev

je vidět i u ohraničení parkovišť, protože obrys má stejnou barvu, ale překrývá výplň. Proto je přímo v databaseManageru vytvořena funkce na kontrolu duplicity.

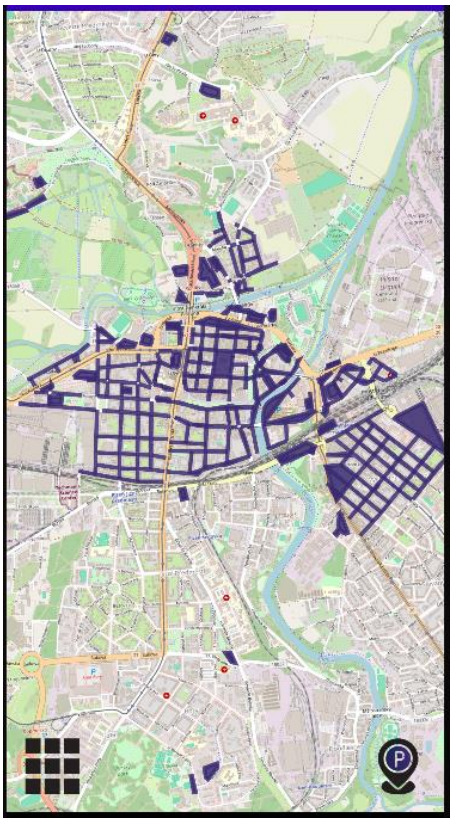
Druhá možnost vytváření parkovacích míst je přímo na mapě. Uživatel musí kliknout na ikonu pro vytváření parkovišť a otevře se mu horní nabídka, která pomáhá při tvorbě. Uživatel musí vždy vybrat alespoň 3 body na mapě (obrázek 13) a pokud je vybere a vytvoří toto místo, ihned se přidá na mapu.

```
fun isPolygonInDatabase(polygonPoints: List<Pair<Double, Double>>):
Boolean {
    val maxPolygonId = getMaxPolygonId()

    for (currentPolygonId in 1..maxPolygonId) {
        val polygonPointsInDatabase =
selectAllPolygonPoints(currentPolygonId)

        if (polygonPointsInDatabase != null && polygonPointsInDatabase
== polygonPoints) {
            return true
        }
    }

    return false
}
```

Zdrojový kód 15: Funkce na kontrolu existenci polygonu v databázi (Zdroj vlastní)

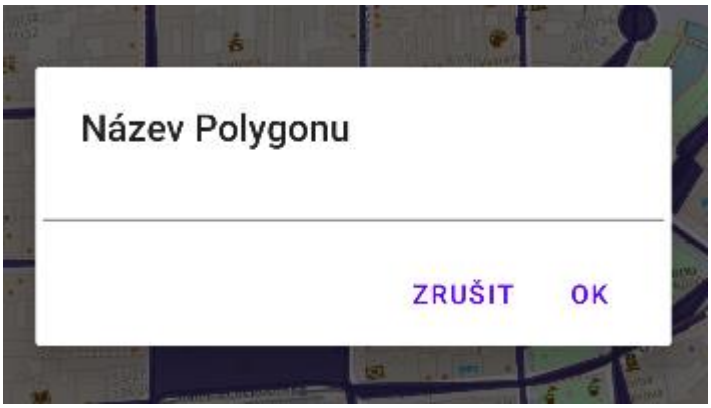
Obrázek 13: Vybraná místa pro parkoviště (Zdroj vlastní)

Každé parkoviště má výplň i obrys nastaven na modrofialovou barvou (#D73A2B s alfa hodnotou 72). Tato barva se hodí k designu celé aplikace. Parkovací místa na mapě jsou zobrazena na obrázku 14.



Obrázek 14: Parkovací místa na mapě (Zdroj vlastní)

Polygony při načtení nebo vytvoření nejsou nijak pojmenované, uživatel si je však může pojmenovat podle sebe, pokud potřebuje. Při kliknutí na polygon je zobrazen dialog (obrázek 15), při vyplnění jména a zmáčknutí tlačítka „ok“ je polygonu nastaveno jméno do jeho středu. Název má bílou barvu a velikost 44f (obrázek 16).



Obrázek 15: Pojmenování polygonu (Zdroj vlastní)

Obrázek 16: Parkovací místo se jménem (Zdroj vlastní)

Vložení názvu na mapu je vytvořeno přes objekt Marker, který je zarovnán do středu polygonu. Polygony obsahují proměnnou infoWindowLocation, která obsahuje střed polygonu ve formatů GeoPoint. Touto proměnnou se práce velmi zjednodušila, protože není potřeba počítat střed ručně.

```
private fun drawText(canvas: Canvas) {
    val paint = Paint()
    paint.color = Color.WHITE
    paint.textSize = markerTextSize
    paint.textAlign = Paint.Align.CENTER
    paint.typeface = Typeface.DEFAULT_BOLD

    val mapViewPoint = mapView.projection.toPixels(position, null)
    val x = mapViewPoint.x.toFloat()
    val y = mapViewPoint.y.toFloat()

    canvas.drawText(markerText, x, y, paint)
}
```

Zdrojový kód 16: Zobrazení názvu polygonu na mapu (Zdroj vlastní)

K implementaci názvů polygonů byla přidána podmínka, která dynamicky ovládá jejich zobrazení na mapě v závislosti na úrovni přiblížení. Tato inovace zvyšuje uživatelskou přívětivost aplikace tím, že zajišťuje, že názvy polygonů jsou viditelné pouze tehdy, pokud je uživatel dostatečně blízko k mapě. Tímto způsobem se eliminuje zbytečný nepořádek na mapě a zajišťuje se optimální uživatelský zážitek. Pro dosažení této funkcionality byla použita dynamická hodnota přiblížení, konkrétně byla stanovena hodnota 15, která byla určena jako nejvhodnější pro optimální zobrazení názvů polygonů.

```
override fun draw(canvas: Canvas?, mapView: MapView?, shadow: Boolean) {
    if (canvas != null && !shadow) {
        val zoomLevel = mapView?.zoomLevelDouble
        if (zoomLevel != null) {
            if (zoomLevel >= 15) {
                drawText(canvas)
                super.draw(canvas, mapView, false)
            }
        }
    }
}
```

Zdrojový kód 17: Kontrola oddálení mapy (Zdroj vlastní)

2.3.2 Databáze

V aplikaci byla použita databáze SQLite, která je vhodná pro méně komplexní data, je zdarma[74] a dostačující pro účely aplikace. Aplikace disponuje dvěma databázemi, které jsou sdílené pro celou aplikaci skrze všechny aktivity.

Sdílení databází je vyřešeno třídami DatabaseManager a DatabaseNameManager. Tyto třídy obsahují funkci getInstance(), která při existující databázi vrací instanci této databáze. V opačném případě je vytvořena nová databáze.

První databáze obsahuje samotné polygony. Polygony jsou ukládány postupně po jednotlivých bodech (obrázek 17). Každý bod obsahuje zeměpisnou výšku, zeměpisnou šířku a id polygonu. Při načítání polygonu se pak vybírají určité body podle tohoto id. Při ukládání se pak najde nejvyšší id, přičte se k němu hodnota 1 a nové záznamy jsou uloženy do databáze.

```
fun getInstance(context: Context): DatabaseManager {
    return instance ?: synchronized(this) {
        instance ?: DatabaseManager(context.applicationContext).also {
instance = it }
    }
}
```

Zdrojový kód 18: Vracení instance databáze (Zdroj vlastní)

id	polygon_id	latitude	longitude
1	1	49.74715777	13.37837574
2	1	49.74707442	13.37978521
3	1	49.74707898	13.37990352
4	2	49.74639715	13.37815253
5	2	49.74633393	13.3782441
6	2	49.74623201	13.37959888

Obrázek 17: Databáze polygonů (Zdroj vlastní)

Druhá databáze je jednodušší, protože obsahuje pouze id polygonu a jméno (obrázek 18). Vzhledem k tomu že při každém vytváření polygonu je mu do jeho proměnné id uložena hodnota je ukládání jednoduché. Stačí načíst polygon.id a jméno, které je zadáno v dialogu. Při načítání jmen do mapy je použita funkce find(), která prohledává polygony podle id a pokud ho najde nastaví do středu polygonu nový marker s textem.

id	polygon_id	name
1	154	test 1
2	121	moje par...
3	321	dum
4	319	skola

Obrázek 18: Databáze jmen polygonů (Zdroj vlastní)

2.4 Možné rozšíření aplikace

Nejlepším rozšířením aplikace by bylo přidání funkce, která po zjištění aktuální polohy najde nejbližší dostupné město v souborech a přidá parkovací místa z tohoto města. Ulehčilo by to databázi i samotné mapě, protože kdyby bylo více souborů na základní načítání dat, tak databáze i mapa bude vytížena načítáním.

Dalším případným vylepšením by bylo načítání pouze parkovacích míst v určitém okolí středu mapy, kde ji má uživatel otevřenou, toto s sebou nese problém častého načítání a mazání dat při pohybu uživatele.

Ideální rozšíření by bylo přidání uživatelského účtu se jménem a dalšími daty například oblíbená a vytvořená místa.

Velkým, ale zajímavým rozšířením by také mohlo být sdílení databáze s ostatními uživateli. Pří této změně by bylo ale potřeba optimalizovat databázi i mapu, protože při velkém objemu dat by mohla být aplikace nestabilní. U Open Street Map je doporučený výkonnostní limit 10000 polygonů na mapě.[69] Při větším objemu lidí, kteří sdílí tato data, by byl pravděpodobně limit překonán a aplikace by byla neoptimalizovaná.

Posledním vhodným vylepšením je menu, které při kliknutí na polygon zobrazí možnost pojmenování, editace a mazání. Editace by odemknula posouvání rohů polygonu a bylo by možné ho upravit.

ZÁVĚR

V této bakalářské práci byla nejprve implementována Open Street Mapa v aplikaci Android Studio. Jsou zde přidělány funkce na načítání geojson souborů, které obsahují parkoviště. Také je zde možnost přidávání parkovacích míst pomocí klikání na mapě. Mezi další hlavní funkci patří pojmenovávání polygonů. Polygony i jména jsou ukládány do databází a načítány při jejím spuštění.

Při práci se vyskytlo mnoho problémů, které byly úspěšně vyřešeny. Prvním problémem byla samotná implementace mapy, protože Google mapy mají zpoplatněny funkce, které jsou pro funkčnost aplikace nezbytné. Proto jsou používány Open Street Mapy, kde se také vyskytly problémy, a to hlavně se zobrazením více map vedle sebe. Dalším problémem byla implementace tlačítek. Tato tlačítka se nezobrazovala a bylo potřeba předělat jejich „kontejner“. Problém nastal i u jmen polygonů, protože bylo potřeba upravovat handler, který tyto objekty mají již nastavený. Objevily se i problémy s ukládáním a načítáním polygonů.

Při tvorbě této aplikace byly využity znalosti získané při studiu na této škole, a to hlavně z předmětů objektově orientované programování, základy programování a mobilní aplikace. Bylo potřeba i nastudovat manuály v Kotlinu, a hlavně dokumentaci Open Street Map, protože zde byly dobře popsány všechny funkce, které jsou použity v aplikaci.

Na rozšíření aktuální aplikace by bylo vhodné pracovat, a to i přesto, že je aplikace funkční. Při používání může nastat problém s překročením doporučeného výkonnostního limitu (10000 polygonů) a aplikace bude zasekaná a pomalá. Tento problém by se dalo vyřešit více způsoby. První možností je použití jiných map (Google mapy), druhým, ne úplně jednoduchým by bylo upravení mapy, a to tím stylem, že by se načítaly pouze polygony, které jsou v blízkosti uživatele, nebo polohy, kam se zrovna uživatel dívá.

POUŽITÁ LITERATURA

[1] "Počítačový program." V: Wikipedie [online]. Dostupné z: https://cs.wikipedia.org/wiki/Počítačový_program [cit. 8. dubna 2024]

[2] "Computer Program." GeeksforGeeks. Dostupné z: https://www.geeksforgeeks.org/what-is-a-computer-program/ [citováno: 8. dubna 2024]

[3] "Batch Processing in Operating System." GeeksforGeeks. [online]. Dostupné z: https://www.geeksforgeeks.org/batch-processing-operating-system/ [Citováno: 8. dubna 2024]

[4] Wikipedia contributors. "Modular programming." Wikipedia, The Free Encyclopedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/Modular_programming [Citováno: 16 Apr. 2024]

[5] Abelson, Harold, Gerald Jay Sussman, and Julie Sussman. Structure and Interpretation of Computer Programs. 2nd ed., The MIT Press, 1996. ISBN 978-0262510875

[6] "Programovací jazyk." Wikipedie, otevřená encyklopedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Programovací_jazyk [Citováno: 8. dubna 2024]

[7] Scott, Michael L. "Programming Language Pragmatics." Morgan Kaufmann, 2015. ISBN 978-0124104099

[8] Wikipedie. "Nižší programovací jazyk." Wikipedie, otevřená encyklopedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Nižší_programovací_jazyk [Citováno: 16. dubna 2024]

[9] Sebesta, Robert W. Concepts of Programming Languages. Pearson, 2015. ISBN 978-133943023

[10] "Vyšší programovací jazyk." Wikipedie: Otevřená encyklopedie. Wikimedia Foundation, Inc. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Vyšší_programovací_jazyk. [Datum přístupu: duben 17, 2024]

[11] "Imperativní programování." Wikipedie, otevřená encyklopedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Imperativní_programování [Citováno: 8. dubna 2024]

[12] "Strukturované programování." Wikipedie, otevřená encyklopedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Strukturované_programování [Citováno: 8. dubna 2024]

[13] "Objektově orientované programování." Wikipedie, otevřená encyklopedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Objektově_orientované_programování [Citováno: 8. dubna 2024]

[14] "Deklarativní programování." Wikipedie, otevřená encyklopedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Deklarativní_programování [Citováno: 8. dubna 2024]

[15] "Java: Rekurze." ITnetwork.cz. [online]. Dostupné z: https://www.itnetwork.cz/java/zaklady/zdrojove-kody/tutorial-java-rekurze. [Datum přístupu: duben 17, 2024]

[16] "Funkcionální programování." Wikipedie, otevřená encyklopedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Funkcionální_programování [Citováno: 8. dubna 2024]

[17] "Logické programování." Wikipedie, otevřená encyklopedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Logické_programování [Citováno: 8. dubna 2024]

[18] "Překladač." Wikipedie, otevřená encyklopedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Překladač [Citováno: 8. dubna 2024]
[19] "Activity lifecycle" [Online]. Dostupné z: https://developer.android.com/guide/components/activities/activity-lifecycle [Citováno: 16. dubna 2024]
[20] "Fragment Lifecycle in Android." GeeksforGeeks. [online]. Dostupné z: https://www.geeksforgeeks.org/fragment-lifecycle-in-android/ [Citováno: 8. dubna 2024]
[21] "Creating and Using Fragments." CodePath Android Guides. [online]. Dostupné z: https://guides.codepath.com/android/Creating-and-Using-Fragments [Citováno: 8. dubna 2024]
[22] Date, C.J. "SQL and Relational Theory: How to Write Accurate SQL Code." O'Reilly Media, 2009. ISBN 978-1449316402
[23] [7] Wikipedie. "CRUD." Wikipedie, otevřená encyklopedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/CRUD [Citováno: 16. dubna 2024]
[24] Wikipedia contributors. "Relational database." Wikipedia, The Free Encyclopedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/Relational_database [Citováno: 16. dubna 2024]
[25] [9] Exasol. "What is a Relational Database?" Exasol. [online]. Dostupné z: https://www.exasol.com/resource/what-is-a-relational-database/ [Citováno: 16. dubna 2024]
[26] W3Schools. "SQL CREATE TABLE Statement." W3Schools. [online]. Dostupné z: https://www.w3schools.com/sql/sql_create_table.asp [Citováno: 16. dubna 2024]
[27] W3Schools. "SQL CREATE TABLE Statement." W3Schools. [online]. Dostupné z: https://www.w3schools.com/sql/sql_create_table.asp [Citováno: 16. dubna 2024]
[28] W3Schools. "SQL SELECT Statement." W3Schools. [online]. Dostupné z: https://www.w3schools.com/sql/sql_select.asp [Citováno: 16. dubna 2024]
[29] W3Schools. "SQL DELETE Statement." W3Schools. [online]. Dostupné z: https://www.w3schools.com/sql/sql_delete.asp [Citováno: 16. dubna 2024]
[30] LinkedIn. "How Can ORM Improve Communication & Collaboration?" LinkedIn. [online]. Dostupné z: https://www.linkedin.com/advice/0/how-can-orm-improve-communication-collaboration-dsluf [Citováno: 16. dubna 2024]
[31] Codecademy. "What Is a Framework?" Codecademy Blog. [online]. Dostupné z: https://www.codecademy.com/resources/blog/what-is-a-framework/ [Citováno: 16. dubna 2024]
[32] Ktor. "Welcome to Ktor Documentation." Ktor. [online]. Dostupné z: https://ktor.io/docs/welcome.html [Citováno: 16. dubna 2024]
[33] InsertKoin.io. [online]. Dostupné z: https://insert-koin.io [Citováno: 16. dubna 2024]
[34] JetBrains. "JetBrains/Exposed: Kotlin SQL Framework." GitHub. [online]. Dostupné z: https://github.com/JetBrains/Exposed [Citováno: 16. dubna 2024]
[35] Otevřená data. "Informace o Otevřených datech." Otevřená data. [online]. Dostupné z: https://opendata.gov.cz/informace:start [Citováno: 16. dubna 2024]
[36] Otevřená data Praha. [online]. Dostupné z: https://opendata.praha.eu [Citováno: 16. dubna 2024]
[37] Otevřená data Praha. [online]. Dostupné z: https://opendata.praha.eu/datasets/https%3A%2F%2Fapi.opendata.praha.eu%2Flod%2Fcatalog%2F5c93758e-3f50-4293-afd9-cf11c12ba01d [Citováno: 16. dubna 2024]

[38] Google Maps Platform. "Three Unique Ways to Add Maps to Custom Native, No-Code Mobile Applications." Google Maps Platform Blog. [online]. Dostupné z: https://mapsplatform.google.com/resources/blog/three-unique-ways-add-maps-custom-native-no-code-mobile-applications/ [Citováno: 16. dubna 2024]
[39] Design Studio UI/UX. "Importance of UI/UX Design in Today's Digital World." Design Studio UI/UX Blog. [online]. Dostupné z: https://www.designstudiouiux.com/blog/importance-of-ui-ux-design-in-todays-digital-world/ [Citováno: 16. dubna 2024]
[40] Interaction Design Foundation. "The UX Design Process: A Beginner's Guide." Interaction Design Foundation. [online]. Dostupné z: https://www.interaction-design.org/literature/article/ux-design-process-guide [Citováno: 16. dubna 2024]
[41] Krug, S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. New Riders, 2014. ISBN 978-0321965516
[42] 99designs. "How to design an app: the ultimate guide." 99designs. [online]. Dostupné z: https://99designs.com/blog/web-digital/how-to-design-an-app/ [Citováno: 16. dubna 2024]
[43] Jones, M. (n.d.). The Impact of Artificial Intelligence (AI) on Mobile App Development. LinkedIn. [online]. Dostupné z: https://www.linkedin.com/pulse/impact-artificial-intelligence-ai-mobile-app-development-dxscc [Citováno: 16. dubna 2024]
[44] eLearning Industry. "How to Use Artificial Intelligence in Mobile Apps." eLearning Industry. [online]. Dostupné z: https://elearningindustry.com/how-to-use-artificial-intelligence-in-mobile-apps [Citováno: 16. dubna 2024]
[45] Codigee. "Introduction to AR in Mobile Apps: The Future of Mobile Technology." Codigee Blog. [online]. Dostupné z: https://codigee.com/blog/introduction-to-ar-in-mobile-apps-the-future-of-mobile-technology [Citováno: 16. dubna 2024]
[46] LinkedIn Pulse. "Augmented Reality (AR) vs. Virtual Reality (VR) in Mobile." LinkedIn. [Online]. Dostupné z: https://www.linkedin.com/pulse/augmented-reality-ar-virtual-vr-mobile [Citováno: 16. dubna 2024]
[47] Startup Grind. "How Blockchain Technology is Transforming Mobile App Development." Startup Grind Blog. [online]. Dostupné z: https://www.startupgrind.com/blog/blockchain-technologymobile-app-development/ [Citováno: 16. dubna 2024]
[48] SynergixServ Private Limited. "Edge vs. Cloud Computing." LinkedIn Pulse. [online]. Dostupné z: https://www.linkedin.com/pulse/edge-vs-cloud-computing-synergixserv-private-limited [Citováno: 16. dubna 2024]
[49] Master. "Edge Computing slibuje nižší latenci." Master Blog. [online]. Dostupné z: https://www.master.cz/blog/edge-computing-slibuje-nizsi-latenci/ [Citováno: 16. dubna 2024]
[50] MOR Software JSC. "Top 10 Mobile App Development Frameworks for 2024." LinkedIn Pulse. [online]. Dostupné z: https://www.linkedin.com/pulse/top-10-mobile-app-development-frameworks-2024-mor-software-jsc-9tlkc [Citováno: 16. dubna 2024]
[51] Android Developers. "Android Studio" [online]. Dostupné z: https://developer.android.com/studio [Citováno: 16. dubna 2024]
[52] Apple Developer. "Xcode" [online]. Dostupné z: https://developer.apple.com/xcode/ [Citováno: 16. dubna 2024]

[53] "M3 - Material Design Component System." [online]. Dostupné z: https://m3.material.io. [Citováno: 16. dubna 2024]
[54] Apple Developer. "Human Interface Guidelines" [online]. Dostupné z: https://developer.apple.com/design/human-interface-guidelines [Citováno: 16. dubna 2024]
[55] Cogtix. "The Importance of Design Patterns in Software Development." [online]. Dostupné z: https://www.linkedin.com/pulse/importance-design-patterns-software-development-cogtix [Citováno: 16. dubna 2024]
[56] Kotlin vs Java: A Comprehensive Comparison." [online]. Dostupné z: https://www.imaginarycloud.com/blog/kotlin-vs-java/#Java [Citováno: 16. dubna 2024]
[57] "Ranges." Kotlin. [online]. Dostupné z: https://kotlinlang.org/docs/ranges.html [Citováno: 16. dubna 2024]
[58] Kotlin. [online]. Dostupné z: https://kotlinlang.org [cit. 1. dubna 2024]
[59] "Google Maps API Alternative - 7 Free Alternatives to Google Maps." Storemapper. [online]. Dostupné z: https://www.storemapper.com/blog/google-maps-api-alternative [Citováno: 16. dubna 2024]
[60] "Datarade.ai's Top Lists: Best Map APIs." Datarade.ai. [online]. Dostupné z: https://datarade.ai/top-lists/best-map-apis [Citováno: 16. dubna 2024]
[61] "MapQuest." Wikipedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/MapQuest [Citováno: 16. dubna 2024]
[62] "Google Maps." Wikipedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/Google_Maps [Citováno: 16. dubna 2024]
[63] "Waze Wiki - Hlavní strana." In: Waze Wiki [online]. Dostupné z: https://www.waze.com//wiki/Czech/Hlavní_strana [cit. 1. dubna 2024]
[64] „MPLA“ mpla.io. [online]. Dostupné z: https://www.mpla.io/en/ [Citováno: 17. dubna 2024]
[65] Tanmay Deo (2021). "What Are Layouts in Android: Everything You Need to Know." Medium. [online]. Dostupné z: https://medium.com/@tanmanydeo321/what-are-layouts-in-android-everything-you-need-to-know-2ecb70bcae67 [Citováno: 17. dubna 2024]
[66] Manu Aravind (2021). "Difference Between Drawables and Mipmaps." Medium. [online]. Dostupné z: https://medium.com/@manuaravindpta/difference-between-drawables-and-mipmaps-ec1d3c11525e [Citováno: 17. dubna 2024]
[67] Android Developers "Auto backup for apps." Android Developers. [online]. Dostupné z: https://developer.android.com/guide/topics/data/autobackup [Citováno: 17. dubna 2024]
[68] Android Developers "Retrieve the device's location." Android Developers. [online]. Dostupné z: https://developer.android.com/develop/sensors-and-location/location/retrieve-current [Citováno: 17. dubna 2024]
[69] "osmdroid“ [online]. Dostupné z: https://github.com/osmdroid/osmdroid [cit. 1. dubna 2024]
[70] "MKergall. Osmbonuspack“ [online]. Dostupné z: https://github.com/MKergall/osmbonuspack [cit. 1. dubna 2024]
[71] "Lassana. osmdroid-shape-extension“ [online]. Dostupné z: https://github.com/lassana/osmdroid-shape-extension [cit. 1. dubna 2024]

[72] Google Developers "Polygon tutorial." Google Developers. [online]. Dostupné z: https://developers.google.com/maps/documentation/android-sdk/polygon-tutorial [Citováno: 17. dubna 2024]
[73] GeeksforGeeks. "Difference between LinearLayout and RelativeLayout in Android." [online]. Dostupné z: https://www.geeksforgeeks.org/difference-between-linearlayout-and-relativelayout-in-android/ [cit. 1. dubna 2024]
[74] SQLite "SQLite." [online]. Dostupné z: https://www.sqlite.org [Citováno: 17. dubna 2024]