UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2023 Ondřej Fialka

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Vývoj mobilní aplikace pro doporučení blízkého místa na zaparkování

Bakalářská práce

2023 Ondřej Fialka

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

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Ondřej Fialka**
Osobní číslo:	**I20084**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Vývoj mobilní aplikace pro doporučení blízkého místa na zaparkování**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem práce je implementace mobilní aplikace, která bude umožňovat uživateli zobrazit na mapě místa vhodná pro zaparkování.
Seznam parkovacích míst bude získáván z otevřených dat a bude je možné přednačíst a uložit do lokálního úložiště.
Uživatel bude mít možnost spravovat svoje preferovaná místa pro parkování.

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

Seznam doporučené literatury:

BOSE, S. – KUNDU, A. – MUKHERJEE, M. – BENERJEE, M. A comparative study: Java vs Kotlin programming in android application development. International Journal of advanced research in computer science, Volume 9, No. 3, ISSN 0976-5697
SIMS, Gary. Android authority [online]. Dostupné z: https://www.androidauthority.com/develop--android-apps-languages-learn-391008/
HARKIRAN, Kaur. Top Programming Languages for Android App Development Dostupné z: https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/

Vedoucí bakalářské práce: • **Ing. Martin Pozdílek, 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 vývoj mobilní aplikace pro doporučení blízkého místa na zaparkování 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 16. 05. 2023

Ondřej Fialka

PODĚKOVÁNÍ

Rád bych tímto poděkoval mému vedoucímu bakalářské práce Ing. Martinu Pozdílkovi, Ph.D. za odborné připomínky a vedení při vypracování této bakalářské práce.

ANOTACE

Cílem této bakalářské práce je vývoj mobilní aplikace pro platformu Android, která uživateli doporučí vhodná blízká místa na zaparkování a pokud k místu existuje trasa tak ji zobrazí. Tyto lokace bude možné uložit lokálně v zařízení pro možnost zobrazení parkovacích míst bez přístupu k internetu.

KLÍČOVÁ SLOVA

Android, React, JavaScript, node.js, Geolokace, Parkovací místa, zeměpisná délka, zeměpisná šířka

TITLE

Development of a mobile application for recommending a nearby parking place

ANNOTATION

The aim of this bachelor thesis is to develop a mobile application for the Android platform, which recommends suitable nearby parking places to the user and if there is a route to the place it will display it. These locations can be stored locally on the device for viewing parking places without internet access.

KEYWORDS

Android, React, JavaScript, node.js, geolocation, parking spaces, longitude, latitude

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

ART – Android Runtime

CSS – Kaskádové styly

HAL – Hardware abstraction layer

HTTP – Hypertext Transfer Protocol

JS – Java script

JSON – JavaScript Object Notation

NDK – Native Development Kit

NPM – Node Package Manager

NPX – Node Package eXecute

UI – user interface

XML – Extensible Markup Language

ÚVOD

Cílem bakalářské práce je realizovat mobilní aplikaci pro doporučení blízkého místa pro zaparkování. Aplikace bude fungovat jak v režimu offline tak online. Nicméně v obou případech vyžaduje přístup k lokaci uživatele. V režimu offline může uživatel zobrazit trasu k lokálně uloženým parkovacím místům. V režimu online pak kormě uložených parkovacích míst přibyde funkcionalita stahování parkovacích míst z open dat, které lze následně ukládat.

Jako zvolené vývojové prostředí jsem použil Visual studio code. K naprogramování aplikace jsem použil framewrok React Native.

1 Motivace

V rámci bakalářské práce, jsem chtěl vytvořit mobilní aplikaci, která by měla smysl a reálné využití. Napadlo mě tedy udělat aplikaci, která na mapě zobrazí blízká parkovací místa vhodná pro rychlé zaparkování. V dnešní době je parkování stále větší problém, zejména v městských oblastech, kde jsou parkovací místa s vysokou cenou a omezeným množstvím. Proto jsem se rozhodl vytvořit aplikaci, která by řidičům pomohla ušetřit čas a stres spojený s hledáním parkovacího místa. Zároveň jsem si chtěl vyzkoušet vývoj mobilních aplikací v React Native a zlepšit si své znalosti a schopnosti v programování JavaScriptu.

Aplikace je zaměřená na řidiče automobilů, motorek a dalších vozidel. Aplikaci jsem se rozhodl napsat v angličtině, jelikož overpass API, odkud parkovací místa stahuji má data parkovacích míst po celém světě.

2 Rešerše

2.1 Parkopedia Parking

Obrázek 1: Parkopedia [1]

Parkopedia parking je aplikace od firmy Parkopedia Ltd založené v roce 2007. Parkopedia poskytuje informace o více než 90 milionech parkovacích míst ve 20 tisících městech v 90 zemích. Aplikace nabízí možnost najití parkovacího místa a navigace k němu. Na rozdíl od mé aplikace Parkopedia drží informace o parkovištích, jako je cena, platební metody. Ve verzi Prémium i aktuální počet volných míst. Umožňuje vyhledávat místa, jak v okolí aktuální lokace, tak i parkovací místa v jiných městech po celém světě a lze si je ukládat do oblíbených. [1]

Tato aplikace funguje na spolupráci s majiteli parkovišť. Díky tomu má aplikace přístup k všem aktuálním datům o parkovišti jako je cena a počet volných míst. A majitel parkoviště má platformu, na které může své parkoviště propagovat. [1]

Aplikace také spolupracuje se značkami automobilů jako Audi, Škoda, BMW, Mercedes-Benz, Peugeot, Porsche a další. Díky této spolupráci mohou výrobci zabudovat vyhledávání parkovacích míst přímo do navigačního systému v automobilu. Řidiči mohou vidět všechna data o parkovišti a koupit si parkovací lístek přímo z vozidla. [1]

2.1.1 Porovnání aplikací

Parkopedia tedy disponuje mnoho výhodami oproti mé aplikaci. Nabízí aktuální informace o parkovištích, jako je cena, počet volných míst nebo provozní doba. Umožňuje koupi parkovacího lístku přímo z aplikace. Moje aplikace tyto funkce nenabízí. [1]

Výhodou mé aplikace je pak možnost najití většího množství parkovacích ploch v okolí, ať už se jedná o soukromé parkoviště, nebo stání u vlakového nádraží. Také podporuje stahování parkovacích míst do lokálního uložiště aplikace, takže je parkoviště možné zobrazit na mapě i bez přístupu k internetu. Má aplikace je pouze zdarma, nikoli pak v placeném módu, což limituje mé možnosti.

2.2 EasyPark

Obrázek 2: EasyPark [2]

Firma EasyPark byla založena v roce 2001. EasyPark drží data o parkovištích ve více než 20 zemích a 3200 měst v Evropě, USA a Austrálii a poskytuje své služby okolo 45 miliónům uživatelů.

Pomocí této aplikace uživatelé mohou jednoduše najít parkovací místo i s informacemi o něm. Aplikace je dostupná pro Android, ale i iOS. Aplikace také umožňuje najít dobíjecí stanice pro elektromobily. EasyPark také podporuje integraci do automobilů. Zatím mají integraci u automobilech značek Volvo a Mercedes-Benz. Nicméně aplikace je kompatibilní s Android Auto a Apple CarPlay. S využitím této možnosti stačí uživateli odjet z parkoviště a parkování se automaticky zastaví, takže se uživatelé nemusí bát, že budou přeplácet za parkování. [2]

Další funkcí EasyPark je služba Garage solutions pro majitele parkovišť. Jedná se o kamerový systém, který automaticky rozpoznává poznávací značky automobilů. Tento systém zlepšuje provoz parkoviště, jelikož stačí autu přijet k závoře od parkoviště a kamerový systém automaticky vyhodnocuje platbu na základě poznávací značky. [2]

2.2.1 Porovnání aplikací

Výhodou této aplikace je možnost najití velkého množství parkovišť a nabíjecích stanic pro elektromobily v Evropě, USA a Austrálii i s daty o parkovištích. Také možnost platit za parkování přímo z aplikace v mobilu a automatické ukončení parkovaní při odjezdu z parkoviště s pomocí Integrace aplikace v automobilu. Také poskytuje služby pro majitele parkovišť jako je automatizovaný kamerový systém. [2]

Výhodou mé aplikace je možnost najití parkovacích míst všude po světě a jejich uložení do mobilního zařízení a následné zobrazení offline. Má aplikace je pouze zdarma, nikoli pak v placeném módu, což limituje mé možnosti.

2.3 Inrix ParkMe

Obrázek 3: Irix ParkMe [3]

Aplikace ParkMe byla vytvořena společností Inrix, která se od roku 2004 zabývá analýzou dat ze silničních senzorů a vozidel [3]. ParkMe poskytuje komplexní data o parkovacích místech ve městech, jako je cena za hodinu parkování, den či za celí měsíc. Také ukazuje počet volných parkovacích míst. Mimo parkovací domy a placené parkoviště také drží data o parkování v ulicích. Ve filtraci umožňuje nastavení příjezdu a odjezdu. Dle toho ukáže vhodná parkoviště a vypočítá cenu parkování. Aplikace také umožňuje platbu nebo rezervaci parkovacího místa přímo z aplikace. [4]

2.3.1 Porovnání aplikací

Aplikace ParkMe má mnoho výhod. Uchovává detailní informace o parkovacích místech jako je cena a počet volných míst. [4]

Opět výhodou mé aplikace je možnost nalezení parkovacích míst všude po světě a jejich uložení do mobilního zařízení a následné zobrazení offline. Má aplikace je pouze zdarma, nikoli pak v placeném módu, což limituje mé možnosti.

3 Použité technologie

3.1 React Native

React Native je JavaScriptový framework od společnosti Meta pro psaní mobilních aplikací pro Android a iOS. Díky dalším knihovnám lze využít i pro psaní webových a desktopových aplikací. Jeho hlavní výhodou je, že jeden kód lze použít jak pro platformu Android, tak i iOS a případně i další platformy. Není tedy třeba psát dvě aplikace zvlášť nebo aspoň většina kódu může být znovu použitelná. React Native funguje na principu univerzálních komponent. Při sestavení aplikace pro jednotlivé platformy jsou pak tyto komponenty převedeny na komponentu dané platformy. [5] [25]

React Native jsem si vybral hlavně z důvodu předchozí zkušenosti s JavaScriptem z vývoje webových aplikací. Také jsem si chtěl vyzkoušet novou technologii, která by se mi mohla nadále v praxi hodit.

React Ntive je vhodný pro psaní jednoduchých aplikací. Při psaní velkých komplexních aplikací je lepší zvolit nativní přístup pro lepší výkon aplikace. [6]

Tabulka 1: Výhody a nevýhody React Native

Výhody	**Nevýhody**
Multiplatformní 90% znuvupoužitelnost kódu	Výkon oproti nativním aplikacím
Změny za běhu aplikace	Špatné ladění
Velké množství pluginů	Není vhodný pro rozsáhlé aplikace
Modulární architektura	

Zdroj: [6]

3.2 Alternativy React Native

3.2.1 Kotlin

Kotlin je multiplatformní jazyk vytvořený společností JetBrains v roce 2011. Od roku 2019 se využívá pro vývoj aplikací pro platformu Android. Nicméně s technologií Kotlin Miltiplatform se dá využít i pro psaní aplikací pro další platformy jako iOS, Web nebo desktopové aplikace. [7] [6]

Tabulka 2: Výhody a nevýhody Kotlinu

Výhody	Nevýhody
Kompatibilita s jazykem Java	Vysoké náklady na údržbu
Konzistentní a čistý kód	Dlouhá doba kompilace
Bezpečný a spolehlivý	
Kotlin Multiplatform	

Zdroj: [6]

3.2.2 Flutter

Flutter byl vytvořen společností Google v roce 2017. Stejně jako React Native je to framework pro vytváření multiplatformních aplikací z jednoho kódu. Jedná se o framework postavený na jazyce Dart, který je též vytvořený společností Google. [8] Je využíván společnostmi jako BMW, eBay, Square a další. [6]

Tabulka 3: Výhody a nevýhody Flutteru

Výhody	Nevýhody
Open Source	Není velice rozšířený
Jednoduchost použití	Malé množství pluginů
Změny za běhu	Sestavená aplikace je veliká
Multiplatformní	

Zdroj: [6]

3.3 Použité komponenty

Pro vytvoření UI jsem využíval základní komponenty React Native, ale také komponenty z balíčků komunity. Dále jsem vytvořil vlastní komponenty, abych se vyvaroval zbytečné duplicitě kódu.

3.3.1 Použité komponenty React Native

View

View je základní komponentou React Native. Jedná se o kontejner, do kterého zabaluji další komponenty. V sestavení aplikace pro platformu Android se pak mapuje jako android.view. [9]

Text

Jedná se o komponentu pro zobrazování textu. V React Native se musí každý text vkládat do komponenty Text. [9]

TextInput

Jedná se o vstup pro zadávání textu z klávesnice [9]. V aplikaci ji používám jak při přihlášení pro zadávání emailu a hesla tak i pro registraci.

StatusBar

Komponentu StatusBar využívám ve všech obrazovkách aplikace k nastavení horní části obrazovky [9]. V aplikaci především nastavuji její pozadí, aby barva ladila s pozadím ostatních částí obrazovky.

Image

Jedná se o komponentu pro zobrazování obrázku v aplikaci [9]. V aplikaci ji využívám na přihlašovací a registrační obrazovce pro zobrazení obrázku.

ActivityIndicator

Tato komponenta slouží k zobrazení načítacího kolečka. V aplikaci ji používám ve všech obrazovkách, aby uživatel věděl, že probíhá načítání či jiná akce.

TouchableHighlight

Tato komponenta se využívá jako obal pro kontejner View. Reaguje na dotyk prstu na obrazovku. Při kliknutí na tuto komponentu se sníží jas této komponenty a všech vnořených [9]. Využívá se například pro vlastní tlačítka. Já ji v aplikaci používám pro bloky v seznamu parkovacích míst.

TouchableOpacity

Tato komponenta má stejné využití jako komponenta TouchableHighlight. Rozdíl je, že při kliknutí na komponentu TouchableHighlight se komponenta pouze ztmaví, ale při kliknutí na TouchableOpacity se zvýší její průhlednost. Komponentu využívám pro vlastní komponentu FloatingButton.

Button

Button je komponenta pro vytvoření tlačítka. Je optimalizována tak, aby vypadala dobře na všech platformách. Tato komponenta má limitované možnosti úpravy. Proto jsem v aplikaci především vytvářel vlastní tlačítka pomocí komponenty TouchableHighlight. Další možností

je komponenta Pressable, která umožňuje ještě větší úpravy [9]. Nicméně pro tlačítka plus a mínus ve filtru pro vyhledávání parkovacích míst jsem si vystačil s touto komponentou.

Alert

Tato komponenta spouští dialogové okno. Lze zde nastavit nadpis a zprávu která bude v dialogovém okně zobrazena. Dále lze přidat tlačítko se specifickým textem a vlastní funkcí. V aplikaci jí využívám při získávání lokace uživatele. V případě, že uživatel nepotvrdil práva pro získání lokace nebo má vypnuté GPS zobrazí se příslušné dialogové okno.

3.3.2 Použité komponenty z balíčků komunity

Navigator

Balíček react-navigation a react-navigation-stack využívám k vytvoření navigace pro obrazovky aplikace. Navigace funguje jak pro Android tak iOS. Vytvoření navigace s těmito balíčky je jednoduché a lehce nastavitelné. [10]

MapView

MapView je komponenta z balíčku react-native-maps. Slouží k integrování Google Map do aplikace React Native. V aplikaci jí používám v obrazovce mapy i offline mapy pro zobrazení lokace uživatele a parkovacích míst. Pomocí parametru initialRegion nastavuji mapu na pozici uživatele. Tento parametr je objekt obsahující: [11]

- Latitude, longitude – pro zeměpisnou šířku a délku
- latitideDelta, longitudeDelta – pro vzdálenost oddálení/přiblížení mapy

MapViewDirections

Komponenta je z balíčku react-native-maps-directions a slouží pro zobrazené trasy na mapě. Je vytvořena pro komponentu MapView. Vkládá se jako vnořené komponenta do komponenty MapView. Komponenta má několik parametrů: [12]

- origin – objekt s atributy: latitude (pro zeměpisnou šířku), longitude (pro zeměpisnou délku). Jedná se o geografický bot pro začátek trasy
- destination– objekt s atributy: latitude (pro zeměpisnou šířku), longitude (pro zeměpisnou délku). Jedná se o geografický bot pro cíl trasy
- apiKey – klíč pro Google Map API
- stokeColor – barva trasy na mapě
- strokeWidth – šířka trasy na mapě
- onError – callback pro funkci při chybě

- onReady – callback pro funkci při najití trasy

Marker

Jedná se o komponentu z balíčku react-native-maps. Slouží pro zobrazení bodu na mapě. Má jediný parametr coordinate pro objekt s atributy: latitude (pro zeměpisnou šířku), longitude (pro zeměpisnou délku) [11]. V aplikaci využívám marker pro označení lokace uživatele a vybraného parkovacího místa.

ScrollView

Tato komponenta je z balíčku react-native-gesture-handler. Tento balíček slouží pro řízení gest pomocí UI vlákna namísto systému JS responder. Díky tomu je interakce s aplikací plynulá a spolehlivá [13]. Komponentu ScrollView pak používám pro zobrazení seznamu parkovacích míst.

Toast

Tato komponenta je z balíčku react-native-toast-message. Slouží pro zobrazování animovaných zpráv. [14] V aplikaci ji používám pro zobrazování oznámení uživateli. Informuji tím například že parkovací místo bylo lokálně uloženo nebo odstraněno. Nebo když dojde k nějaké chybě.

3.3.3 Vlastní komponenty

PrimaryButton

Tuto komponentu jsem vytvořil, jelikož jsem chtěl vlastní styl tlačítka a komponenta Button neumožňovala dostatečné možnosti úprav. Jelikož tlačítko využívám ve více obrazovkách vytvořil jsem ho jako samostatnou komponentu, kterou pak do obrazovky přidávám. Komponenta se skládá z několika vnořených komponent. Nejprve komponenta TouchableHighlight slouží jako obal pro ostatní komponenty a vytváří animaci při kliknutí na tlačítko. Dále komponenta View a uvnitř komponenta Text.

Komponenta má dva parametry, a to parametr Text a OnPress. Parametr text slouží pro text, který má tlačítko obsahovat. Parametr OnPress je funkce, která se spustí při kliknutí na

Tabulka 4: Kód komponenty PrimaryButton

```
const PrimaryButton = ({ onPress, text }) => {
  return (
    <TouchableHighlight style={mainStyles.buttonOuter} onPress={onPress}>
    <View style={mainStyles.buttonPrimary}>
      <Text style={mainStyles.buttonTextLight}>{text}</Text>
    </View>
  </TouchableHighlight>
  );
};
```

FloatingButton

Tato komponenta je opět komponenta tlačítka. Tuto komponentu využívám v obrazovkách map. Komponenta se opět skládá z několika vnořených komponent React Native. První z nich je komponenta TouchableOpacity, která obaluje všechny ostatní komponenty. Dále kontejner View a uvnitř je komponenta Icon nebo ActivityIndicator, to záleží na nastavení tlačítka.

Komponenta má sedm parametrů:

- onPress – funkce která se spustí po kliknutí
- icon – text pro jméno ikony
- left – vzdálenost od levé strany obrazovky nebo null
- right – vzdálenost od pravá strany obrazovky nebo null
- top – vzdálenost od horní strany obrazovky nebo null
- bottom – vzdálenost od spodní strany obrazovky nebo null
- active – boolean dle kterého se uvnitř tlačítka ukazuje buď ikona nebo se točí kolečko

Tabulka 5: Kód komponenty FloatingButton

```
const FloatingButton = ({onPress, icon, left, right, top, bottom, active}) =>
{
  let floatingButtonStyles = {...mainStyles.floatingButton};
  right ? (floatingButtonStyles.right = right) : null;
  left ? (floatingButtonStyles.left = left) : null;
  top ? (floatingButtonStyles.top = top) : null;
  bottom ? (floatingButtonStyles.bottom = bottom) : null;

  return (
    <TouchableOpacity style={floatingButtonStyles} onPress={onPress}>
      <View style={{justifyContent: 'center', alignItems: 'center'}}>
        {active ? (
```

```
          <Icon name={icon} size={40} color="white" />
        ) : (
           <ActivityIndicator size="large" color={mainStyles.colors.secondary}
/>
        )}
      </View>
    </TouchableOpacity>
  );
};
```

3.3.4 Další použité balíčky

Firebase

Jedná se o Firebase SDK pro JavaSrcipt. V aplikaci ho využívám pro přihlašování a registraci uživatelů. Abych nemusel inicializovat firebase v každé obrazovce, ve které chci Firebase použít. Vytvořil jsem si soubor firebase.js. V tomto souboru mám několik funkcí, které exportuji:

- getApp() - Tato funkce slouží k inicializaci Firebase aplikace.
- getAuthObj() – Inicializuje autentizační objekt pro autentizaci skrze Firebase Authentication

Geolocation

Geoloaction z balíčku react-native-community/geolocation. Využívám ji pro získání geolokace uživatele.

react-native-android-open-settings

Tento balíček používám pro otevření nastavení zařízení Android přímo z aplikace. Já ho využívám pro otevření nastavení GPS v případě, že uživatel má GPS vypnuté.

Axios

Balíček axios slouží pro vytváření http dotazů. V aplikaci ho využívám pro httpd GET dotaz na získání dat parkovacích míst. Dále ho využívám pro http dotaz na Google Maps API pro získání adresy parkovacího místa.

async-storage

Jedná se o balíček pro ukládání lokálních data v aplikaci. V paměti si ukládá ve formátu klíč hodnota. Obě hodnoty jsou ukládány jako typ string. AsyncStorage v aplikaci využívám pro ukládání parkovacích míst. Parkovací místa ukládám jako objekt obsahující atribut spots.

Atribut spots je pole objektů parkovacích míst. Jelikož se do AsyncStorage musí hodnota ukládat jako string využívám metody JSON.stringify() pro převedení objektu do typu string.

PermissionAndroid

PermissionAndroid je přímo z balíčku react-native. Nabízí přístup k systému oprávnění pro platformu Android. Využívám ho pro získání oprávnění k přístupu k lokaci uživatele. Bez udělení tohoto oprávnění se uživatel nedostane k mapě.

3.4 Firebase

Firebase je platforma od společnosti Google typu Backend as a service. Poskytuje tedy backendové řešení přímo v cloudu. Administrace backendu je dostupná přes jejich webové rozhraní, které je velmi jednoduché a intuitivní. Firebase je zaměřená především na mobilní a webové aplikace. Nabízí spoustu služeb jako: [15]

- Firebase authnetication
- NoSQL uložiště Firestore a Realtime database
- A/B testing, Remote config
- Google Analytics, Crashlytics a spoustu dalších

Díky těmto službám vývojáři nemusí řešit správu backendových komponent. Vše běží v cloudu a je spravováno týmem Firebase. [15]

3.4.1 Firebase Authentication

Pro svou práci jsem využil služby Firebase authentication. Tato služba slouží k řízení identity uživatelů. Pomocí této služby můžeme vytvářet v aplikaci registrace a přihlašování uživatelů a následně řídit autorizaci. Služba podporuje mnoho metod ověřování: [15]

- nativní metody: email/heslo, telefon, anonym
- metody třetích stran: Google, Facebook, Play Games, Game Center, Apple, GitHub, Microsoft, Twitter, Yahoo
- vlastní metody (pouze s Identity Platform): openID Connect, SAML

Pro implementaci služby do aplikace lze využít předpřipravených UI knihoven přímo od Firebase nebo je možné službu implementovat pomocí jejich SDK. [15] Já jsem službu implementoval za pomocí Firebase SDK pro JavaScript.

Pro ověřování uživatelů jsem zvolil metodu email/heslo. Uživatelé se mohou přihlašovat a registrovat emailovou adresou ze všech domén, ale ve Firebase lze omezit domény, ze kterých se uživatel může přihlašovat. Heslo pak musí být dlouhé minimálně 6 znaků.

Službu jsem zvolil z důvodu předešlých zkušeností s platformou Firebase. Služba Firebase Authentication i další služby Firebase jsou velmi jednoduše implementovatelné a nastavitelné. Bylo pro mě tedy jednoduché službu do aplikace implementovat. Získal jsem tím zabezpečenou metodu ověřování uživatelů bez nutnosti vytvářet vlastní databázový server, řešit jeho správu a zabezpečení.

Přihlášení

Přihlášení jsem implementoval pomocí metody signInWithEmailAndPassword(). Ta má parametry:

- auth – Instance autentizačního objektu
- email – emailová adresa
- password – heslo účtu

Tabulka 6: Metoda přihlášení uživatele

```
const LogIn = () => {
  setErrorMessage('');
  setLoading(true);
  signInWithEmailAndPassword(getAuthObj(), email, password)
    .then(userCredential => {
      const user = userCredential.user;
      navigation.navigate('HomeStack');
    })
    .catch(error => {
      const errorCode = error.code;
      const errorMessage = error.message;
      setErrorMessage('wrong email or password!');
      setLoading(false);
    });
};
```

Nejprve se nastaví chybový text na prázdný text. Poté se změní stav načítání na true. Díky tomu se objeví na obrazovce načítací animované kolečko z komponenty ActivityIndicator. Následně proběhne pokus o přihlášení pomocí metody signInWithEmailAndPassword(). Pokud je přihlášení úspěšné, probíhá přesměrování na obrazovku mapy. V opačném případě se nastaví chybová hláška a je zobrazena uživateli.

Registrace

Pro registraci jsem využil metody createUserWithEmailAndPassword(). Ta má parametry:

- auth – Instance autentizačního objektu
- email – nová emailová adresa
- password – heslo k novému účtu

Tabulka 7: Metoda registrace uživatele

```
const Register = () => {
  setErrorMessage('');
  setLoading(true);
  createUserWithEmailAndPassword(getAuthObj(), email, password)
    .then(async userCredential => {
      const user = userCredential.user;
      console.log(„adduser“);
      console.log(„adduserAdded“);
      navigation.navigate(‚HomeStack‘);
    })
    .catch(error => {
      console.log(‚registrace‘, error);
      setErrorMessage(‚wrong email or password (at least 6 chars)!‘);
      setLoading(false);
    });
};
```

Při registraci je nejprve nastavena chybová hláška na prázdný text. Poté se změní stav načítání na true. Díky tomu se objeví na obrazovce načítací animované kolečko z komponenty ActivityIndicator. Následně proběhne pokus o registraci pomocí metody createUserWithEmailAndPassword(). Pokud je registrace úspěšná, probíhá přesměrování na obrazovku mapy. V opačném případě se nastaví chybová hláška a je zobrazena uživateli.

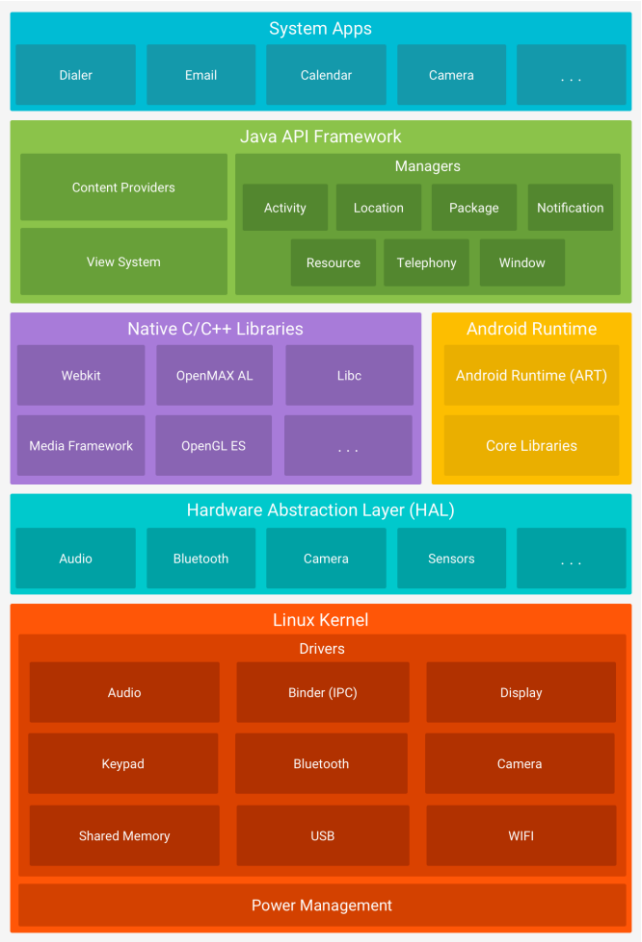
3.5 Android

Android je operační systém pro mobilní telefony, tablety, chytré hodinky a další zařízení. Je vytvořen na jádře Linuxu a je vyvíjen společností Google. Aktuálně je to nejrozšířenější operační systém v telefonech. Využívá ho většina značek chytrých mobilních telefonů. Samotní výrobci si pak software upravují dle své potřeby a přejmenovávají ho.

Svou práci jsem napsal pro platformu Android, jelikož sám mám mobilní zařízení s platformou Android. Nicméně jelikož jsem aplikaci napsal v multiplatformním frameworku React Native, šla by s menšími změnami v kódu sestavit i pro platformu iOS. [16]

3.5.1 Architektura

Architektura operačního systému Android se skládá z několika hlavních součástí. A to z jádra Linux Kernel, Hardware Abstraction Layer (HAL), nativních Knihoven C/C++, Android Runtime, Java API Framwork a Systémových aplikací. [16]



Obrázek 4: Architektura operačního systému Android [16]

Linuxové jádro Linux Kernel

Linuxové jádro je základem platformy Android. Stará se o základní funkce jako vlákno a nízko úrovňová správa paměti. Díky Linuxovému jádru mohou výrobci zařízení lehce vyvíjet ovladače. [16]

HAL

HAL poskytuje rozhraní pro Java API. Skládá se z několika modulů, kde každý je zodpovědný za rozhraní pro určitý typ hardwarové komponenty. [16]

Android Runtime

Android Runtime je běhové prostředí pro aplikace a systémové služby. Od verze Android 5.0 běží každá aplikace ve vlastním procesu a s vlastní instancí Android Runtime. [16]

Nativní knihovny C/C++

Nativní knihovny C/C++ jsou v systému, protože některé klíčové součásti systému jsou vytvořeny z nativního kódu. Například ART nebo HAL. K některým funkcím těchto knihoven lze přistupovat i pomocí Java API. Pokud aplikace vyžaduje kód v jazyce C nebo C++, lze se k těmto knihovnám dostat pomocí Android NDK. [16]

Java API framework

Všechny funkce, které Android OS poskytuje jsou přístupné skrze Java API framework. Toto API je, jak už název napovídá, napsáno v jazyce Java. Vývojáři aplikací ho můžou využívat pro přístup k funkcím Androidu. [16]

Systémové aplikace

Systémové aplikace fungují jako pro uživatele ale i vývojáře. Pokud vývojář potřebuje ve své aplikaci funkcionalitu, kterou už plní jiná systémová aplikace může využít tuto aplikaci aby funkcionalitu plnila za něj. Například pokud by chtěl poslat email může využít už nainstalované systémové emailové aplikace. [16]

3.6 Node.js

Node.js je javascriptové asynchronní runtime prostředí. Je potřeba pro běh javascriptových aplikací mimo webový prohlížeč, ve kterém je javascriptové prostředí zabudováno. Například v prohlížeči Google Chrome je zabudován V8. Javascriptový runtime je tedy nutný pro vývoj aplikací v React Native. [17]

3.6.1 npm

Npm je největší softwarový registr. Používá se pro sdílení a stahování balíčků kódu mezi vývojáři. V placené verzi lze mít i soukromé registry. Například pro firemní potřeby. Při instalaci Node.js se automaticky nainstaluje i npm. Má vlastní konzolové rozhraní, pomocí kterého lze jednoduše instalovat balíčky příkazem npm install a jméno balíčku. Při vývoji jsem ho používal pro stahování všech dodatečných balíčku. [18]

3.6.2 Npx

Npx slouží pro spouštění binárních souborů npm blaíčků. Umí spouštět jakýkoliv npm balíček, aniž by musel být nainstalován.

Pro React Native jsem ho použil při vytváření nového projektu React Native pomocí příkazu npx react-native@latest init AwesomeProject a spouštění aplikace pomocí příkazu npx react-native start

3.7 Overpass-turbo

Overpass turbo je webový nástroj pro získávání dat z OpenStreetMap. K získávání dat používá Overpass API dotazy. Jedná se o webové rozhraní, ve kterém je možné vytvářet API dotazy pro Overpass API. Tyto dotazy lze v rozhraní testovat a okamžitě na mapě vidět data z odpovědi. [19]

3.7.1 OpenStreetMap

OpenStreetMap je mapa celého světa, která je volně editovatelná a dobrovolníci po celém světě se snaží o vytvoření komplexních dat. Aktuálně už existuje několik hotových map jako: [20]

- Cyklistická mapa
- Mapa pro vozíčkáře
- Mapa pro venkovní aktivity
- Mapa veřejné dopravy
- Jezdecká a turistická mapa
- Požární mapa hydrantů a hasičských sborů

Tyto mapy jsou volně přístupné na internetu a lze je prohlížet stejně jako mapy Google nebo seznam mapy. Hlavní výhodou pro využít OpenStreetMap a ne Google Map je téměř volný přístup k datům. To je také cíl celého projektu. Vytvořit volná geografická data pro volné využití v různých projektech. [20]

3.7.2 Overpass API

Overpass API je API pro čtení dat z OpenStreetMap. Pomocí http dotazu se klient dotáže na určitá data a v odpovědi je dostává. API je optimalizováno pro klienty tak aby data obdrželi během chvilky. API je schopné klientovy odeslat až milióny dat během několika minut. [21]

Výsledná data jsou ve formátu XML, nicméně v dotazu lze definovat výstupní formát jako formát JSON. Já jsem v práci využil tohoto nastavení, jelikož práce s JSON formátem je v Javascriptu výrazně jednodušší. Dá se sním pracovat jako s objektem, a tedy tečkovou notací se dostat až na chtěné atributy objektu. Data jsou z dotazu vrácena jako pole těchto objektů. V mém případě objektů parkovacích míst. [21]

3.8 platforma Google Maps

Google Maps platform je sada API a SDK, které můžou vývojáři využít pro implementování Google Map do své aplikace nebo pro získání dat z Google Map. Poskytuje [22]:

- API a SDK pro mapy pro JavaScript Android i iOS.
- API tras
- API a SDK pro místa na mapě

Já jsem ve své práci využil balíček react-native-maps pro implementování Google Map do mé aplikace. Pro získání adres parkovacích míst jsem pak použil API pro reverzní geokódování. Toto API pomocí zeměpisné šířky a délky vrací adresu místa.

4 Práce s daty parkovacích míst

Po kliknutí na tlačítko Load se volá funkce loadSpots().

4.1 Stažení dat z overpass API

Pro stažení dat jsem vytvořil funkci fetchOverPassTurboParking(). Tato funkce má tři parametry:

- lat – zeměpisná šířka uživatele
- lon – zeměpisná délka uživatele
- radius – poloměr v kterém se vyhledají parkovací místa

Nejprve z těchto parametrů vytvořím dotaz pro overpass API. V něm specifikuji, jaká data a v jakém formátu je chci stáhnout.

Tabulka 8: HTTP dotaz na overpass API

```
`[out:json][timeout:25];
(node["amenity"="parking"](around:${radius},${lat},${lon});
way["amenity"="parking"](around:${radius},${lat},${lon});
relation["amenity"="parking"](around:${radius},${lat},${lon})
);out center;out body;>;out skel qt;`
```

Poté pomocí balíčku axios, který slouží pro http dotazy odešlu na " https://overpass-api.de/api/interpreter" tento dotaz. V odpovědi pak dostávám JSON nalezených parkovacích míst. Všechny parkovací místa projdu smyčkou foreach a zkontroluji, že mají zeměpisnou šířku a délku své pozice. Pokud mají, ukládám je do pole jako objekt s atributy name, Latitude a Longitude. Toto pole následně funkce vrací.

4.2 Seřazení dle vzdálenosti

Seřazení probíhá pomocí funkce quickSortByDistance(). Tato funkce má dva parametry:

- arr – pole objektů parkovacích míst
- obj – objekt uživatelovy pozice se zeměpisnou šířkou a délkou

Tato funkce implementuje algoritmus QuickSort pro seřazení všech parkovacích míst. Funkce začíná kontrolou délky pole. Pokud je pole prázdné nebo obsahuje jen jeden prvek, je již setříděné a funkce vrátí původní pole. V opačném případě se určí pivot pole a vzdálenost

uživatele k pivotu. Následně smyčkou procházím celé pole a pokud vzdálenost parkovacího místa k uživateli je menší než vzdálenost pivotu k uživateli, tak se parkovací místo vloží do levého pole. Pokud je vzdálenost delší, vloží se do pravého pole. Poté se rekurzivně zavolá funkce pro levé a pravé pole, dokud se nedostane k jednoprvkovým polím. Nakonec jsou všechny setříděné části spojeny do jednoho pole a funkce seřazené pole vrací.

4.3 Vzdálenost mezi dvěma geografickými body

Pro získávání vzdálenosti mezi dvěma geografickými body se zeměpisnou šířkou a délkou používám funkci getDistance(). Funkce má dva parametry čímž jsou dva objekty obsahující zeměpisnou šířku a délku. Funkce implementuje Haversinův vzorec. Ten slouží pro výpočet vzdálenosti mezi dvěma body na povrchu koule. Pro výpočet je nutné znát zeměpisnou šířku a délku obou bodů a také průměr samotné koule. [23]

$$
\begin{aligned}
d &= 2r \arcsin\left(\sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + (1 - \operatorname{hav}(\varphi_1 - \varphi_2) - \operatorname{hav}(\varphi_1 + \varphi_2)) \cdot \operatorname{hav}(\lambda_2 - \lambda_1)}\right) \\
&= 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \left(1 - \sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) - \sin^2\left(\frac{\varphi_2 + \varphi_1}{2}\right)\right) \cdot \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \\
&= 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right).
\end{aligned}
$$

Obrázek 5: Haversinův vzorec [23]

- d – je vzdálenost mezi dvěma body podél hlavní kružnice koule
- φ 1, φ 2 – jsou zeměpisná šířka bodu 1 a zeměpisná šířka bodu 2
- λ 1, λ 2 – jsou zeměpisná délka bodu 1 a zeměpisná délka bodu 2

4.4 Filtrace

Po seřazení všech parkovacích míst probíhá filtrace všech velmi blízkých parkovacích míst, jelikož overpass API občas vrací odkaz na stejné parkoviště vícekrát, jen s pár metry rozdílnou polohou. Pro to jsem vytvořil funkci filterCloseSpots(). Ta má jen jeden parametr:

- spots – seřazené pole objektů parkovacích míst

V této funkci pomocí smyčky for procházím celé pole parkovacích míst. Uvnitř této smyčky nejprve získám vzdálenost aktuálního a následujícího prvku pole pomocí funkce getDistance popsané v předchozím bodu. Pokud je vzdálenost menší než minimální vzdálenost 0.08km tak následující prvek z pole odeberu a snížím index i o 1 abych v dalším průběhu smyčky opět kontroloval stejný prvek s prvkem následujícím.

4.5 Získání adresy

Když jsou data parkovacích míst stažena, seřazena a vyfiltrována tak pro všechny stáhnu adresu. Projdu celé pole pomocí smyčky for uvnitř která stahuju adresu pro každé parkovací místo. Ke stažení adresy využívám API platformy Google Map. Pro dotázání opět používám balíček axios s metodou GET. Dotaz posílám na adresu:

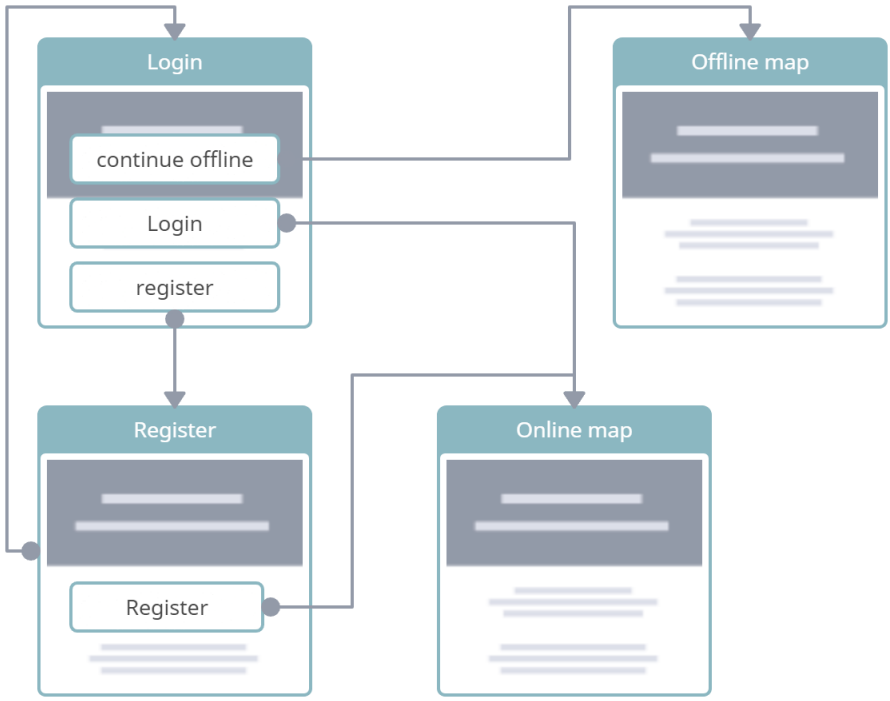
```
https://maps.googleapis.com/maps/api/geocode/json?latlng=${ele
ment.Latitude},${element.Longitude}&key=APIKey`
```

Do url adresy dosazuji zeměpisnou šířku a délku parkovacího místa. Následně atribut name objektu parkovacího místa nastavuji na adresu z odpovědi.

Poté jsou data kompletní a aplikace je může vypsat do seznamu nalezených parkovacích míst.

5 Navigace

Pro navigaci mezi obrazovkami jsem využil balíčky react-navigation a react-navigation-stack. Pomocí těchto balíčků vytvářím komponentu Navigator, která je zodpovědná za posloupnost a navigaci mezi jednotlivými obrazovkami. Navigace je nastavená tak, aby zprvu šlo přepínat mezi přihlašovací a registrační obrazovkou. Po přihlášení nebo registraci se navigace přepne na obrazovku mapy. Nebo pomocí odkazu continue offline se lze přepnout na offline mapu.

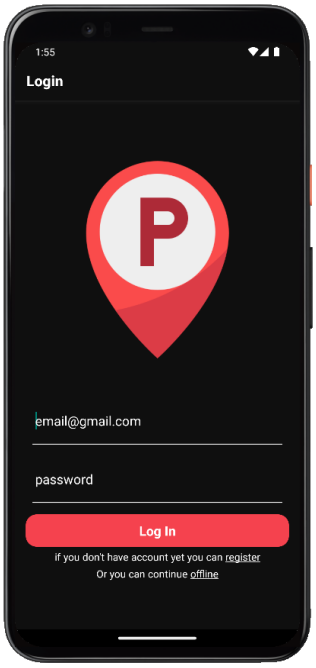


Obrázek 6: Navigace mezi obrazovkami

6 Obrazovky aplikace

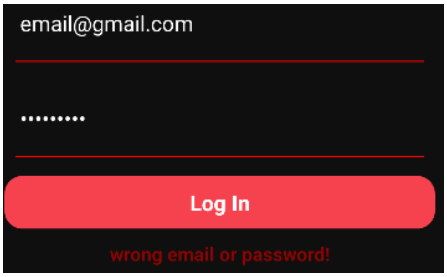
6.1 Přihlášení

Při spuštění aplikace se jako první obrazovka zobrazí právě přihlašovací obrazovka.



Obrázek 7: Přihlášení

Ta obsahuje přihlašovací formulář se dvěma poli. Prvním polem je pole pro email uživatelova účtu. Druhé pole je pak pro jeho heslo. Formulář lze potvrdit tlačítkem Log In. Po potvrzení formuláře se spouští metoda LogIn(). Metoda slouží k autentizaci skrze Firebase pomocí metody signInWithEmailPassword(authObj,email,password). Pokud tato metoda proběhla úspěšně, tak vrací objekt uživatele a probíhá přesměrování na obrazovku mapy. V případě selhání metody se vypíše chybová hláška a přesměrování neprobíhá.



Obrázek 8: Chyba přihlášení

V případě, že uživatel ještě nemá vytvořený učet nebo si chce vytvořit nový, může kliknout na text register. Po kliknutí na tento text dojde k přesměrování na obrazovku registrace.

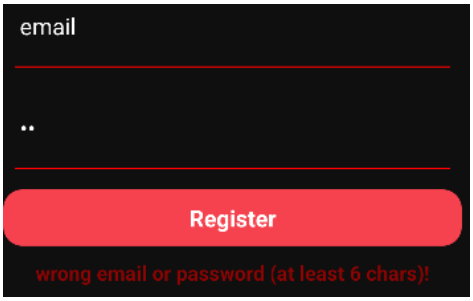
V případě, že uživatel nemá přístup k internetu může využít offline mapy. K přesměrování na obrazovku offline mapy doje po kliknutí na text offline.

6.2 Registrace

Do obrazovky registrace se dostaneme z přihlašovací obrazovky. Z ní se můžeme vrátit zpět na přihlašovací obrazovku kliknutím na šipku zpět v levém horním rohu. Registrační obrazovka vypadá následovně.

Obrázek 9: Registrace

Obrazovka obsahuje registrační formulář s dvěma poli. Stejně jako u přihlašovacího formuláře první pole slouží pro zadání emailu uživatelova účtu. Druhé pole pak slouží pro vložení hesla. Heslo musí být dlouhé minimálně 6 znaků. Formulář lze potvrdit tlačítkem Register, které spouští metodu Register(). V této metodě probíhá registrace ve Firebase Authentication pomocí metody createUserWithEmailPassword(authObj, email, password). Pokud registrace úspěšně proběhla, probíhá přesměrování na obrazovku mapy. V opačném případě se zobrazuje chybová hláška.



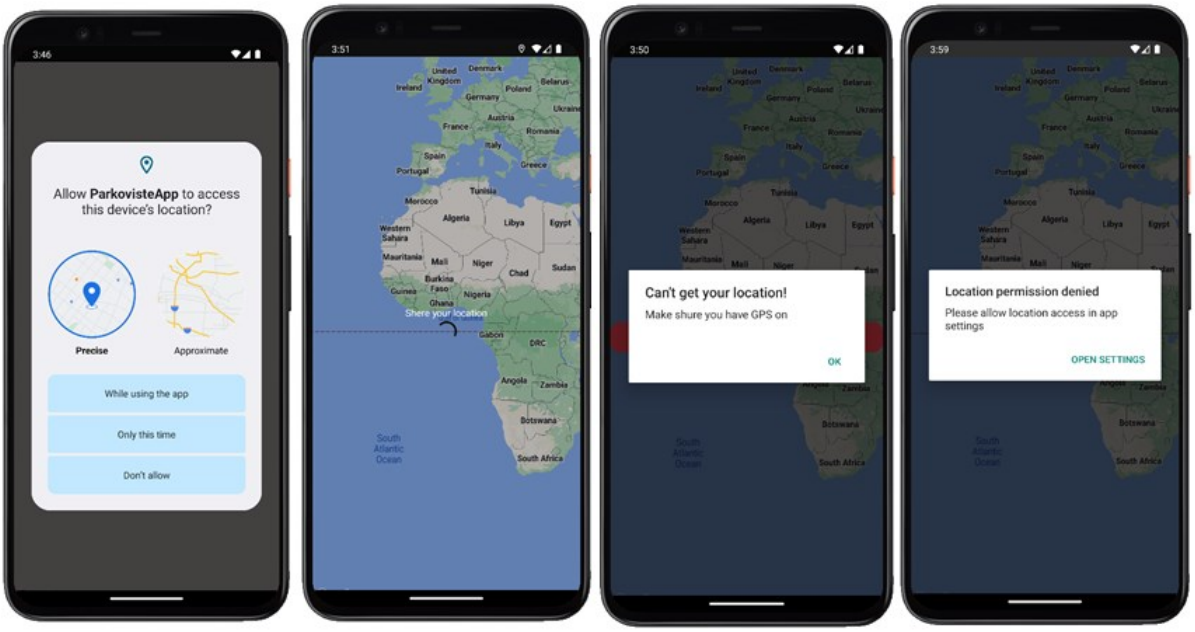
Obrázek 10: Chyba registrace

6.3 Mapa

Obrazovka mapy je hlavní obrazovkou této aplikace. Dostaneme se na ni z obrazovky přihlášení nebo registrace.

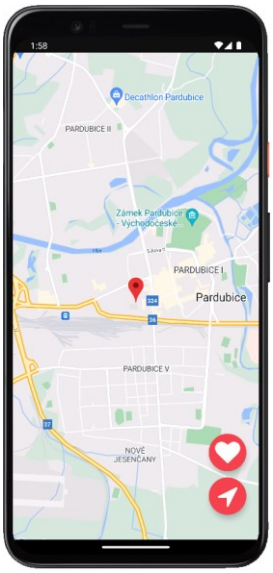
Předtím než se uživatel dostane k mapě musí být splněny určité požadavky, které aplikace ověřuje. Nejprve od uživatele žádá přístup k lokaci telefonu. To udělá pomocí funkce requestLocationPermission(). Ve které se volá funkce PermissionAndroid.request(PermissionsAndroid.PERMISSIONS.ACCESS_FINE_LOCATION). Pokud je přístup odepřen, aplikace dál uživatele nepustí a zobrazí mu okno se zprávou Please allow location access in app settings a tlačítkem pro otevření nastavení.

Pokud byl přístup přidělen, tak se aplikace snaží získat pozici zařízení pomocí balíčku react-native.community/geolocation. V případě, že selže, zobrazí okno se zprávou Make sure you have GPS on. S odkazem do nastavení GPS.



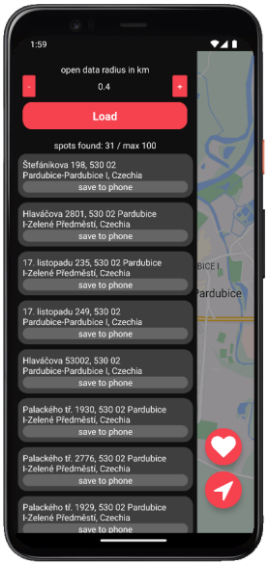
Obrázek 11: Proces získávání lokace zařízení

V případě, že se povedlo získat lokaci zařízení, dostáváme se k mapě. Mapa je hlavní obrazovkou této aplikace. Slouží k načítání parkovacích míst z overpass API, ukládání, mazání a načítání parkovacích míst z lokálního uložiště aplikace a k vyhledání trasy k parkovacímu místu.



Obrázek 12: Mapa

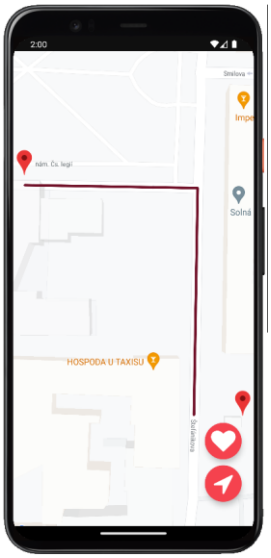
Zprvu je na mapě zobrazen pouze bod ukazující lokaci uživatele. Také obsahuje dvě tlačítka. První tlačítko s ikonou šipky zobrazuje menu s filtrem a seznamem doporučených parkovacích míst z overpass API. Druhé tlačítko slouží pro zobrazení seznamu uložených parkovacích míst v zařízení. Při kliknutí na tlačítko se šipkou se zobrazí postranní menu.



Obrázek 13: Menu s doporučenými parkovacími místy

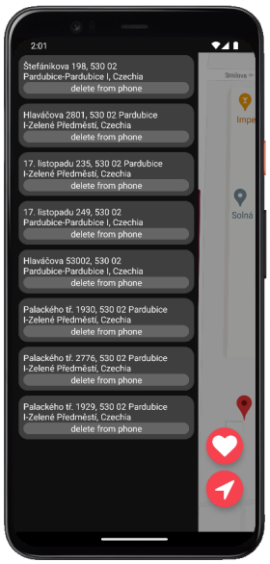
Postranní menu obsahuje filtr. Filtrem nastavujeme vzdálenost, ve které chceme parkovací místa vyhledat. Vzdálenost lze měnit tlačítky plus a mínus. Vzdálenost se mění o 0.2 kilometru a nejmenší možná vzdálenost je 0.2 kilometru. Poté tlačítkem Load lze filtr potvrdit. Zavolá se funkce loadSpots() a spustí se vyhledávání parkovacích míst. Nejprve se místa stáhnou z overpass API ve formátu JSON. Poté probíhá srovnání parkovacích míst dle vzdálenosti od aktuální lokace uživatele. Následně jsou místa filtrována, jelikož overpass-tubo

API občas vrací odkaz na stejné parkoviště vícekrát. Proto proběhne vyfiltrování míst, která jsou blízko sebe a následuje stažení jména, respektive adresy parkovacích míst. To probíhá skrze Google Map API. Po úspěšném stažení parkovacích míst jsou vypsány do seznamu v postranním menu. Odkazy na parkoviště obsahují adresu parkoviště a tlačítko save to phone. To slouží k uložení parkovacího místa do lokálního uložiště. Po kliknutí na toto tlačítko se postranní menu zavře a zobrazí se oznámení, že parkovací místo bylo uloženo. V případě, že už parkovací místo je uloženo, zobrazí se oznámení, že parkovací místo už uložené je a znova se neukládá.



Obrázek 14: Mapa s trasou k parkovacímu místu

Při kliknuti na samotný odkaz parkovacího místa se opět zavře postranní menu a na mapě se vykreslí bod ukazující na parkovací místo a trasa k parkovacímu místu. K vyhledání trasy opět slouží Google Maps API. V případě, že trasa nabyla nalezena, zobrazí se oznámení.

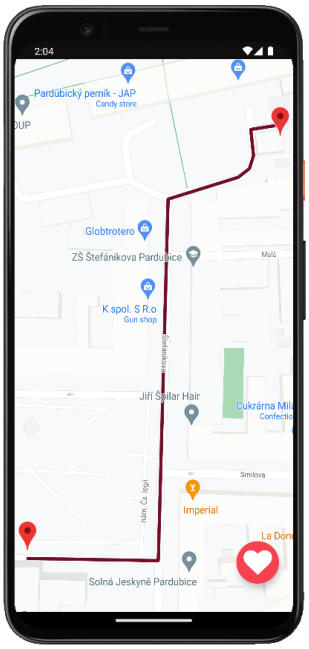


Obrázek 15: Menu s uloženými parkovacími místy

Druhé tlačítko se srdcem pak slouží pro stažené parkovací místa. Při kliknutí na toto tlačítko se zobrazí druhé postranní menu, které obsahuje seznam uložených parkovacích míst. Odkaz na parkovací místo opět obsahuje tlačítko, tentokrát však pro odstranění parkovacího místa z lokálního uložiště. Při kliknutí na toto tlačítko se zavře postranní menu a zobrazí se oznámení, že parkovací místo bylo úspěšně odstraněno. Při kliknutí na samotný odkaz parkovacího místa se zobrazí ukazatel místa na mapě a opět pomocí Google Maps API proběhne pokus o získání trasy k parkovacímu místu. V případě úspěchu se trasa vykreslí na mapu. V opačném případě se zobrazí oznámení, že trasa nebyla nalezena.

6.4 Offline Mapa

V případě, že v přihlašovací obrazovce klikneme na text „or continue offline“ dostaneme se k obrazovce offline mapy. Nejprve stejně jako u online mapy proběhne získání lokace. V případě, že získání lokace bylo úspěšné dostáváme se přímo k mapě. Zde je ukazatel aktuální lokace uživatele na mapě a jedno tlačítko. Tlačítko se srdcem opět slouží k zobrazení postranního menu s uloženými parkovacími místy. Jednotlivé odkazy na parkovací místa opět obsahují tlačítka pro odstranění parkovacího místa z lokálního uložiště. Při kliknutí na odkaz parkovacího místa se na mapě zobrazí ukazatel pozice parkovacího místa.



Obrázek 16: Offline mapa

Offline mapa stále implementuje funkcionalitu vyhledání trasy. V případě, když se uživatel nechce přihlašovat, může k obrazovce offline mapy pokračovat i s přístupem k internetu a v tomto případě se na mapu vykreslí i samotná trasa k parkovacímu místu. Nicméně uživatel ztrácí možnost vyhledávat nová parkovací místa. Tato funkcionalita funguje jen pro přihlášené uživatele.

7 Styly aplikace

Styly v React Native se píší obdobně jako ve vývoji webových stránek pomocí CSS. Hlavní rozdíl je v tom, že se píší velbloudí notací. Například píšeme backgroundColor místo background-color. Předávají se komponentě jako parametr style. Styly se předávají jako objekt, ale lze předat i pole objektů. [24]

Vytvořil jsem si tedy soubor styles.js, ze kterého exportuju objekt styles. Vytvořil jsem ho, abych mohl importovat styly do jednotlivých obrazovek a znovu je využívat. Díky tomu je nemusím neustále kopírovat, a pokud chci udělat nějakou změnu, stačí ji udělat na jednom místě.

Nejprve si definuji primární a sekundární barvu: const primary = "#f5424e"; a const secondary = "#0f0f0f";

Tyto barvy pak dál využívám ve stylech. Objekt styles obsahuje jednotlivé objekty stylů pro různé komponenty.

```
colors:{

    primary: primary,
    secondary: secondary,
}
```

Objekt colors obsahuje právě primární a sekundární barvu. Takže když v aplikaci někde vytvářím styly pro komponentu, můžu je použít a barvy jsou jednoduše změnitelné a všude stejné.

```
floatingButton: {
    position: 'absolute',
    backgroundColor: primary,
    borderRadius: 50,
    height: 60,
    width: 60,
    justifyContent: 'center',
    alignItems: 'center',
    elevation: 5,
    zIndex: 100,
  }
```

Dále například nastavení stylů pro komponentu FloatingButton. Komponenta má absolutní pozici, nastavení barvy pozadí a velikosti tlačítka. Poté obsahuje položku elevation pro vytvoření stínu tlačítka a zIndex, aby tlačítko bylo vždy nahoře.

ZÁVĚR

Aplikaci jsem ozkoušel na svém mobilním zařízení a splňuje požadované funkce. Nepochybně by se dala po vzoru konkurence dále rozvíjet a vylepšovat, ale to by vyžadovalo čas a prostředky, které profesionální aplikace získávají z poplatků a reklamy. Jako hlavní prostor pro vylepšení aplikace bych viděl získání dat o placených parkovacích místech, metodách placení a případné navázání na platební brány. Toto jsou ovšem funkce, které vyžadují spolupráci s již existujícími organizacemi a nelze je řešit bez finančního zajištění.

POUŽITÁ LITERATURA

[1] *Parkopedia: Welcome to Parkopedia!* [online]. c2023 [cit. 2023-04-25]. Dostupné z: https://www.parkopedia.com/about-us/

[2] *EasyPark: Making Cities More Livable* [online]. c2023 [cit. 2023-04-25]. Dostupné z: https://easyparkgroup.com/

[3] *Inrix: about* [online]. c2023 [cit. 2023-04-25]. Dostupné z: https://inrix.com/about/

[4] *Parkme* [online]. c2023 [cit. 2023-04-25]. Dostupné z: https://www.parkme.com/

[5] META. *React Native: Learn once, write anywhere* [online]. c2023 [cit. 2023-05-12]. Dostupné z: https://reactnative.dev/

[6] INDEX. *Index Blog: Flutter vs React Native vs Kotlin: Which will drive app development in 2023?* [online]. c2023 [cit. 2023-05-12]. Dostupné z: https://www.index.dev/post/flutter-vs-react-native-vs-kotlin-which-will-drive-app-development-in-2023

[7] JETBRAINS. *Kotlin* [online]. c2023 [cit. 2023-05-12]. Dostupné z: https://kotlinlang.org/

[8] GOOGLE. *Flutter: Build apps for any screen* [online]. c2023 [cit. 2023-05-13]. Dostupné z: https://flutter.dev/

[9] META. *React Native: Core Components and APIs* [online]. c2023 [cit. 2023-05-13]. Dostupné z: https://reactnative.dev/docs/components-and-apis#basic-components

[10] *React Navigation: Routing and navigation for Expo and React Native apps* [online]. c2023 [cit. 2023-05-15]. Dostupné z: https://reactnavigation.org/

[11] GITHUB. *React-native-maps* [online]. c2023 [cit. 2023-05-13]. Dostupné z: https://github.com/react-native-maps/react-native-maps

[12] GITHUB. *React-native-maps-directions* [online]. c2023 [cit. 2023-05-13]. Dostupné z: https://github.com/bramus/react-native-maps-directions

[13] NPM. *React-native-gesture-handler* [online]. c2023 [cit. 2023-05-13]. Dostupné z: https://www.npmjs.com/package/react-native-gesture-handler

[14] NPM. *React-native-toast-message* [online]. c2023 [cit. 2023-05-14]. Dostupné z: https://www.npmjs.com/package/react-native-toast-message

[15] FIREBASE. *Documentation: Learn the fundimentals* [online]. c2023 [cit. 2023-05-14]. Dostupné z: https://firebase.google.com/docs

[16] GOOGLE. *Android: Platform architecture* [online]. c2023 [cit. 2023-05-14]. Dostupné z: https://developer.android.com/guide/platform

[17] OPENJS FOUNDATION. *Nodejs: About Node.js* [online]. c2023 [cit. 2023-05-14]. Dostupné z: https://nodejs.org/en/about

[18] NPM. *Documentation: About npm* [online]. c2023 [cit. 2023-05-14]. Dostupné z: https://docs.npmjs.com/about-npm

[19] OPEN STREET MAP. *Overpass turbo* [online]. c2023, aktualizováno 23. 10. 2022 [cit. 2023-05-14]. Dostupné z: https://wiki.openstreetmap.org/wiki/Overpass_turbo

[20] OPEN STREET MAP. *About OpenStreetMap* [online]. c2023, aktualizováno 13. 4. 2022 [cit. 2023-05-14]. Dostupné z: https://wiki.openstreetmap.org/wiki/About_OpenStreetMap

[21] OPEN STREET MAP. *Overpass API* [online]. c2023, aktualizováno 1. 5. 2023 [cit. 2023-05-15]. Dostupné z: https://wiki.openstreetmap.org/wiki/Overpass_API

[22] GOOGLE. *Google Maps Platform: Build awesome apps with Google's knowledge of the real world* [online]. c2023 [cit. 2023-05-15]. Dostupné z: https://developers.google.com/maps

[23] WIKIPEDIA. *Haversine formula* [online]. [cit. 2023-05-15]. Dostupné z: https://en.wikipedia.org/wiki/Haversine_formula#cite_note-Gade2010-9

[24] META. *React Native: Style* [online]. c2023 [cit. 2023-05-15]. Dostupné z: https://reactnative.dev/docs/style

[25] WARD, Dan. *React Native Cookbook: Recipes for solving common React Native development problems*. 2nd ed. Packt Publishing, 2019. ISBN 978-1788991926.

SEZNAM PŘÍLOH

PŘÍLOHA A*:* Funkce pro stažení parkovacích míst

```
async function fetchOverpassTurboparking(lat, lon, radius) {
  const query = `[out:json][timeout:25];
    (
node["amenity"="parking"](around:${radius},${lat},${lon});
way["amenity"="parking"](around:${radius},${lat},${lon});
relation["amenity"="parking"](around:${radius},${lat},${lon});
); out center; out body; >; out skel qt;`;
return await axios
    .post('https://overpass-api.de/api/interpreter', query)
    .then(response => {
      let pom = [];
      response.data.elements.forEach((element, key) => {
        if (
          (element.type === 'way' && element.center === undefined) ||
          (element.type === 'node' &&
            (element.lat === undefined || element.lon === undefined))
        ) {
          return;
        }
        let obj = {
          name: key,
          Latitude: null,
          Longitude: null,
        };
        if (element.type === 'way') {
          obj.Latitude = element.center.lat;
          obj.Longitude = element.center.lon;
          pom.push(obj);
        } else if (element.type === 'node') {
          obj.Latitude = element.lat;
          obj.Longitude = element.lon;
          pom.push(obj);
        }
      });
        return pom;
    })
    .catch(error => {
      return error;
    });
}
```

popis přílohy A, zdroj.

PŘÍLOHA B*:* Funkce pro srovnání parkovacích míst

```
function quickSortByDistance(arr, obj) {
  if (arr.length <= 1) {
    return arr;
  }

  const pivotIndex = Math.floor(arr.length / 2);
  const pivot = arr[pivotIndex];
  const left = [];
  const right = [];

  for (let i = 0; i < arr.length; i++) {
    if (i === pivotIndex) {
      continue;
    }

    const distanceToPivot = getDistance(pivot, obj);
    const distanceToCurrent = getDistance(arr[i], obj);

    if (distanceToCurrent < distanceToPivot) {
      left.push(arr[i]);
    } else {
      right.push(arr[i]);
    }
  }

  return quickSortByDistance(left, obj).concat(
    pivot,
    quickSortByDistance(right, obj),
  );
}
```

PŘÍLOHA C*:* Funkce pro získání vzdálenosti mezi dvěma body

```
function getDistance(obj1, obj2) {
  const earthRadius = 6371; // in km

  const latRad1 = (obj1.Latitude * Math.PI) / 180;
  const latRad2 = (obj2.Latitude * Math.PI) / 180;
  const deltaLat = ((obj2.Latitude - obj1.Latitude) * Math.PI) / 180;
  const deltaLon = ((obj2.Longitude - obj1.Longitude) * Math.PI) / 180;

  const a =
    Math.sin(deltaLat / 2) * Math.sin(deltaLat / 2) +
    Math.cos(latRad1) *
      Math.cos(latRad2) *
      Math.sin(deltaLon / 2) *
      Math.sin(deltaLon / 2);

  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
  const distance = earthRadius * c;
  return distance;
}
```

PŘÍLOHA D*:* Funkce pro filtrování blízkých míst

```
function filterCloseSpots(spots) {
  for (let i = 0; i < spots.length - 1; i++) {
    const distance = getDistance(spots[i], spots[i + 1]);
    if (distance <= 0.08) {
      spots.splice(i + 1, 1);
      i--;
    }
  }
}
```