ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická

Bakálářská práce

Progresivní webová aplikace pro plánování a sdílení výletů

Aneta Czerneková

Vedoucí práce: Bc. Petr Huťťák
Obor: Softwarové inženýrství a technologie
Srpen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Czerneková** Jméno: **Aneta** Osobní číslo: **478064**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Webová aplikace pro plánování a sdílení výletů

Název bakalářské práce anglicky:

Web application for trip planning and sharing

Pokyny pro vypracování:

Sestavte návrh a proveďte implementaci aplikace pro plánování a sdílení výletů. Aplikace bude sloužit především k propojení jednotlivců, kteří mají zájem o společně cestování.
Nejprve proveďte analýzu podobných řešení na trhu. Následně identifikujte funkční požadavky a znázorněte strukturu a fungování aplikace pomocí vybraných UML diagramů. Dále proveďte rešerši a vyberte vhodné technologie pro implementaci aplikace.
Na základě výstupů z analýzy navrhněte prototyp a proveďte uživatelské testování, jehož výsledky zapracujte do analýzy. Poté naimplementujte aplikaci za použití zvolených technologií a následně realizujte závěrečné uživatelské testování.
Při implementaci aplikace se primárně zaměřte na mobilní zařízení a využijte technologie spjaté s Progresivními webovými aplikacemi.

Seznam doporučené literatury:

[1] Gaurav Kaushik. Progressive Web App – The future of Web Development. International Journal for Research in Applied Science and Engineering Technology. 2019, 7 495-498. DOI 10.22214/ijraset.2019.7077
[2] Progressive web apps (PWAs). 2020. https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps.
[3] The starting point for learning TypeScript. https://www.typescriptlang.org/docs/home
[4] Morten Hertzum. Usability Testing: A Practitioner's Guide to Evaluating the User Experience. Synthesis Lectures on Human-Centered Informatics. 2020, 1 i-105. DOI 10.2200/S00987ED1V01Y202001HCI045.
[5] Chauncey Wilson. Interview techniques for UX practitioners: a user-centered design method. Morgan Kaufmann, an imprint of Elsevier, 2014.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Bc. Petr Hurťák, katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **07.09.2020** Termín odevzdání bakalářské práce: **05.01.2021**

Platnost zadání bakalářské práce: **19.02.2022**

Bc. Petr Hurťák		prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ _____

Datum převzetí zadání Podpis studentky

Poděkování

Chtěla bych poděkovat především svému vedoucímu panu Bc.Petru Hurťákovi za trpělivost a pomoc při vypracovávání této práce. Dále děkuji všem, kteří se zúčastnili uživatelského testování.

Prohlášení

Prohlašuji, že jsem tuto práci vypracovala samostatně za použití uvedené literatury a zdrojů v souladu s Metodickým pokynem č. 1/20.

Abstrakt

Cílem této práce je provedení návrhu a následná implementace aplikace pro plánování a sdílení výletů. Při implementaci je kladen důraz na použití technologií spjatých s Progresivními webovými aplikacemi. Práce se zabývá analýzou stávajících řešení, výběrem vhodných technologií a popisem implementace a uživatelského testování aplikace.

Klíčová slova: Progresivní webová aplikace, plánování výletů, analýza požadavků, Node.js, TypeScript

Abstract

The aim of this work is to design and implement a trip planning and sharing application. The implementation heavily focuses on the usage of technologies related to Progressive Web applications. The thesis itself covers research of existing solutions, selection of suitable technologies and the description of the implementation and user testing of the application.

Keywords: Progressive web app, trip planning, requirements analysis, Node.js, TypeScript

Title translation: Progressive web application for trip planning and sharing

Obsah

Slovníček pojmů

Pojem	Význam
HTTP	hypertextový přenosový protokol
JSON	javasriptový objektový zápis
PWA	progresivní webová aplikace
SQL	strukturovaný dotazovací jazyk
UX	user experience
IDE	integrované vývojové prostředí

Kapitola 1
Úvod

1.1 Představení aplikace

Stěžejní myšlenkou aplikace je umožnění uživatelům cestovat s cizími lidmi a částečně s nimi sdílet některé náklady(např. pronájem domu, či auta). Nápad na tuto aplikaci vznikl na základě podobných aplikacích postavených na sdílené ekonomice, jako např. AirBnB či různé carsharingové platformy. Aplikace však cílí i na uživatele, kteří chtějí cestovat pouze s přáteli a nemají zájem využít možnosti propojení s cizími lidmi. Pro tyto uživatele je v aplikaci k dispozici možnost vytvoření privátního výletu. V následujících kapitolách jsou popsány detaily analýzy a implementace aplikace, průběh jejího testování a výčet technologií spjatých s Progresivními webovými aplikacemi.

Kapitola 2
Analýza trhu

Na trhu mobilních aplikací již existuje poměrně velké množství aplikací zaměřených na cestovatele. Z těch nejpopulárnějších byly vybrány tři, které byly následně důkladně otestovány z hlediska poskytovaných funkcionalit. V každé aplikaci byla vyzkoušena tvorba vlastního itineráře a přidání jednotlivých položek. Dále byly prozkoumány možnosti sdílení itinerářů s přáteli či s ostatními uživateli. Veškeré testování probíhalo pouze na neplacených verzích pro Android.

2.1 TripIt

Jedná se pravděpodobně o nejpopulárnější aplikaci pro plánování výletů, která měla již v roce 2016 13 milionů uživatelů [1]. Je dostupná pouze ve formě mobilní aplikace, ve verzích pro iOS i pro Android, a byla vyvinuta firmou SAP Concur.

Aplikace nabízí řadu funkcionalit, mezi ty primární patří:

- automatické generování itineráře na základě emailů s elektronickými vstupenkami při povolení přístupu k emailové schránce
- možnost sdílení itineráře (včetně povolení jeho úprav) s konkrétními uživateli
- vysoká flexibilita – aplikace nabízí velké množství typů položek, které lze přidat do itineráře

Mezi slabé stránky aplikace patří nepřehlednost itineráře, ve kterém chybí lepší vizuální oddělení jednotlivých dnů.

TripIt dle všeho cílí především na cestovatele, kteří již mají svůj výlet naplánovaný a hledají nástroj pro uchování všech informací o výletu na jednom místě.

2.2 Sygic Travel

Sygic Travel byla vyvinuta slovenskou společností Sygic, která se primárně zabývá vývojem navigačních systému pro mobilní přístroje[2]. Aplikace je dostupná nejen v mobilní verzi pro iOS a Android, ale i ve verzi webové. V dubnu 2020 měla aplikace v Google PlayStore přes 1 milion stáhnutí.

Aplikace nabízí tyto funkcionality:

- jednoduché přidání položky do itineráře, díky vlastní databázi atrakcí, hotelů, restaurací a dalších turisty navštěvovaných míst
- automatické přidání dopravy mezi jednotlivé položky v itineráři – typ dopravy je možné dodatečně specifikovat a aplikace poskytuje podrobné informace, včetně trasy a instrukcí
- přístup k cestovatelským příručkám a k mapám městské hromadné dopravy
- sdílení itineráře s ostatními pomocí odkazu

Hlavní nevýhodou aplikace je její nízká flexibilita – uživatel nemá možnost přidání vlastní položky, či přidání doplňujících informací k jednotlivým položkám.

Cílovou skupinou aplikace jsou uživatelé, kteří již mají vybranou lokalitu svého výletu a potřebují výlet důkladněji naplánovat. Neslouží však k ukládání podrobných informací o dopravě, hotelu, či lístcích do muzea.

2.3 Roadtrippers

Poslední testovanou aplikací je Roadtrippers. Stejně jako Sygic Travel je i tato aplikace dostupná ve verzích pro iOS, Android a web a má přes 1 milion stáhnutí v Google PlayStore. Aplikace má sloužit pro plánování „roadtripů“, tedy cest autem.

Mezi nabízené funkcionality patří:

- zobrazení zajímavých míst, hotelů nebo restaurací (dle výběru uživatele) podél plánované trasy
- možnost přidání zastávek do itineráře – ve verzi zdarma je však možné přidat maximálně 5 položek
- zobrazení detailů místa, včetně recenzí
- sdílení itineráře pomocí odkazu

Největší nevýhodou aplikace je opět nízká flexibilita. Aplikace nedovoluje žádnou úpravu nabízených položek, či alespoň přidání vlastní poznámky. Navíc je zaměřená výhradně na cestování autem, tudíž nenabízí možnost přidání jiné dopravy do itineráře.

Cílovou skupinu tentokrát tvoří uživatelé, kteří plánují cestu autem, mají již naplánovanou trasu a hledají vhodná místa k přespání nebo aktivity, kterými by mohli cestu doplnit.

2.4 Závěr

Ačkoliv všechny aplikace nabízejí řadu zajímavých funkcionalit a zároveň umožňují sdílení itinerářů s přáteli, jejich koncept se značně liší od aplikace implementované v rámci této práce. Aplikace neumožňují veřejné sdílení itinerářů ani přidání ceny k položkám v itineráři a pozvaní přátelé nemohou pozvání potvrdit, či odmítnout. Na druhou stranu nabízejí řadu funkcionalit, které v rámci této práce implementovány nebudou – např. automatický import cestovních plánů z emailů, či plánování trasy.

Kapitola 3
Progresivní webové aplikace

Aplikace byla vytvářena s důrazem na uživatele mobilních zařízení, nicméně lze spustit na všech platformách, která disponují webovým prohlížečem. Aby bylo možné sdílet kód napříč různými platformami, byla aplikace implementována jako PWA. PWA jsou aplikace, které využívají různé technologie a návrhové vzory, které aplikacím poskytují určité funkcionality typické pro nativní aplikace. Aplikace pak například umožňují zasílání push notifikací uživateli nebo přístup k některým částem aplikace i pokud je uživatel offline.[3] Aplikace tak částečně dokáže simulovat UX nativních aplikací. Výhodou pro koncové uživatele oproti nativním aplikacím je především to, že aplikaci není nutné před prvním použitím instalovat. Uživatel tak může aplikaci nejprve vyzkoušet a instalovat ji až při dalším použití.[4] Díky tomu je aplikace schopna obsloužit i ty uživatele, které od používání aplikací odrazuje nutnost instalace, a má tak větší dosah.

3.1 Vlastnosti PWA

Ačkoliv se nejedná o formalizovaný standard, PWA by měly splňovat určité vlastnosti. Mezi ty klíčové podle Google patří:[5]

- instalovatelnost – aplikaci je možné nainstalovat a spustit ji z domovské obrazovky, není to však nutností pro její používání
- responzivita – uživatelské prostředí aplikace je uzpůsobeno všem velikostem obrazovky
- nezávislost na připojení – aplikace používá cachovací mechanismy, které umožňují její omezené použití i pokud uživatel není připojen
- rychlost – aplikace by měla bez prodlev reagovat na akce provedené uživatelem
- dohledatelnost – aplikace používá správné meta značky a je tak jednoduše dohledatelná pomocí webových vyhledávačů

3.1.1 Webový aplikační manifest

Manifest je JSON soubor obsahující informace, které jsou nutné k umožnění stáhnutí a instalace aplikace. Mezi tyto informace patří například jméno a autor aplikace, ikony pro různé typy zařízení, či barevné schéma aplikace.[6]

3.2 Service worker

Service worker patří mezi webové workery. Webový worker je javascriptový objekt, který umožňuje spuštění javascriptového kódu v jiném vlákně, než na kterém běží hlavní kód aplikace. Může tak na pozadí provádět úkoly bez toho, aby ovlivňoval chod aplikace.[7] Samotný service worker funguje jako proxy, která umožňuje převzít kontrolu nad tím, jak budou zpracovány HTTP požadavky. Díky tomu je možné specifikovat, jak se bude aplikace chovat, pokud uživatel nebude připojen k síti a zajistit přístup k vybraným

částem aplikace v offline módu. Ke zpřístupnění dat a souborů potřebných ke spuštění aplikace i bez přístupu k síti používá service worker cachovací mechanismy. Service worker také mimo jiné umožňuje posílání push notifikací uživatelům. Při psaní kódu pro service worker je třeba myslet na to, že kód neběží v prohlížeči a nemá tedy přístup k DOMu či k localStorage.[8]

3.3 HTTPS protokol

Aplikace musí být obsluhována zabezpečeným protokolem HTTPS, který umožňuje bezpečnou výměnu dat mezi klientem a serverem. Použití této technologie je nutnou podmínkou pro fungování service workerů.[9]

3.4 Instalace aplikace

Aby PWA co nejvěrněji imitovala nativní mobilní aplikace, umožňují novější prohlížeče její instalaci. Nejedná se však o instalaci v pravém slova smyslu, kdy dochází k instalaci souborů do zařízení, nýbrž jen o pouhé přidání URL odkazu aplikace do sekce aplikací na uživatelově zařízení. Tento odkaz se tak tváří jako opravdová aplikace, ta nicméně stále běží v prohlížeči, nikoliv na uživatelově zařízení. Aby byla aplikace instalovatelná musí splnit následující podmínky:

- obsahuje soubor manifest.json, jehož atribut *display* je nastaven na jednu z následujících hodnot: *fullscreen*, *standalone*, or *minimal-ui*
- obsahuje service worker, který implementuje fetch handler
- běží v protokolu HTTPS
- obsahuje ikony nutné pro instalaci aplikace [10]

Na Android zařízeních je instalace podporována všemi moderními prohlížeči, na iOS zařízeních je prozatím zprostředkována pouze v prohlížeči Safari.[11]

3.5 Push Notifikace

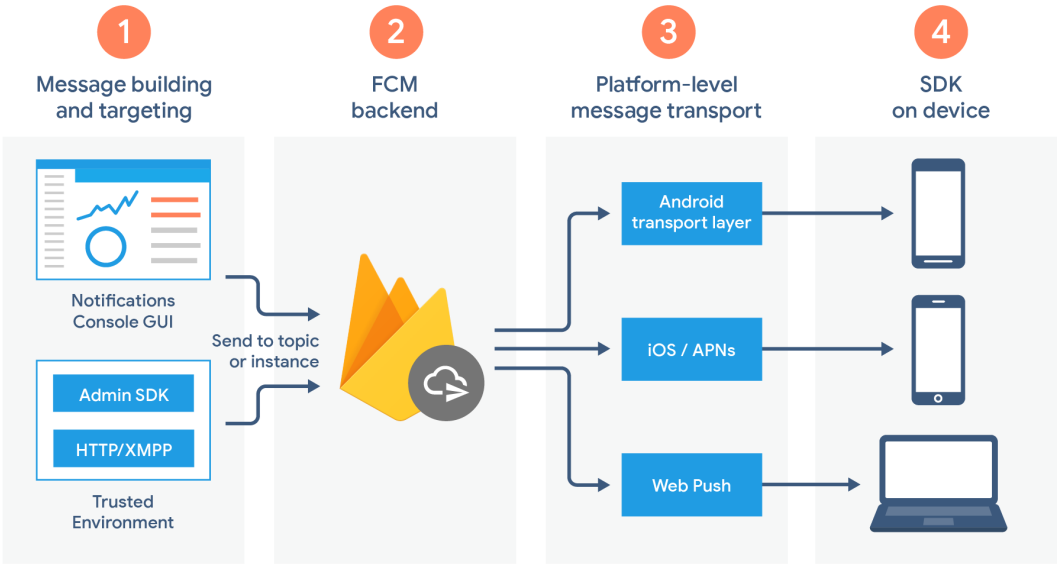
PWA částečně podporují zasílání push notifikací uživatelům aplikace. Využívají k tomu dvě různé API:

- Notifications API Tato API umožňuje aplikaci zobrazit uživateli systémové zprávy, za předpokladu, že uživatel svolí k jejímu použití.[12]
- Push API Díky této API může aplikace přijímat zprávy zaslané serverem a to i v případě, že pouze běží na pozadí. Podmínkou zpřístupnění této API je přítomnost service workeru.[13]

Ani jedna z těchto API však v současné době není podporována prohlížečem Safari, tudíž je na zařízeních s iOS tato funkcionalita bohužel zatím nedostupná.

3.5.1 Firebase cloud messaging

Aby server mohl posílat klientovi push notifikace, je nutné aby implementoval `Web Push protokol`[1]. Jednou z možností, jak obejít nutnost jeho implementace na vlastním serveru je použití již existující platformy pro zasílání zpráv. Pro potřeby této aplikace byla vybrána služba `Firebase cloud messaging`[2], která slouží jako prostředník mezi vlastním serverem a klientskou částí aplikace - ze serveru přijímá informace o tom, jakou zprávu má odeslat a na základě toho pak pomoci web push protokolu zašle zprávu relevantnímu uživateli.[14]



Obrázek 3.1. Znázornění fungování služby FCM [14] Pro zprovoznění této služby a push notifikací v klientské části aplikace je nutné:

- Vytvořit nový projekt ve službě Firebase Cloud Messaging a zkopírovat konfigurační údaje
- Nainstalovat do projektu knihovnu `firebase`[3]
- Inicializovat Firebase aplikaci a vytvořit novou instanci messaging objektu

```
const messaging = firebase.messaging();
```

- Zajistit uživatelův souhlas se zobrazením notifikací v aplikaci. Notifications API obsahuje metodu *requestPermission*, která uživateli zobrazí UI pro schválení či zamítnutí notifikací.

```
const status = await Notification.requestPermission();
```

- Získat nový FCM token a uložit jej do localStorage. Tento token slouží k identifikaci uživatele ve službě FCM a Firebase díky němu dokáže určit, kterému uživateli poslat konkrétní zprávu.

```
const fcm_token = await messaging.getToken();
localStorage.setItem('fcm_token', fcm_token);
```

- Vytvořit nový service worker, který má na starosti zobrazování notifikací uživateli
- Přidat do aplikace kód, který umožňuje zobrazení zprávy uživateli

[1] `https://datatracker.ietf.org/doc/html/draft-ietf-webpush-protocol`
[2] `https://firebase.google.com/products/cloud-messaging`
[3] `https://www.npmjs.com/package/firebase`

```
messaging.onMessage((message) => {
const { title, body } = JSON.parse(message.data.notification);
const options = {
body,
};
self.registration?.showNotification(title, options);
});
```

Pro serverovou část jsou kroky vedoucí ke zprovoznění následující:

- Instalace knihovny `firebase-admin`/urlnotehttps://www.npmjs.com/package/firebase-admin/
- Inicializování aplikace za použití privátního klíče, který je možné získat ve vytvořeném Firebase projektu
- Poslání zprávy konkrétnímu uživateli, či uživatelům. Pro tyto potřeby je nutné v posílaném objektu specifikovat FCM token uživatele.

```
await admin
.messaging()
.send({
token: <fcm_token>,
notification: {
title: "New notifications",
body: 'Hello from server.',
},
})
```

Kapitola 4
Analýza a návrh aplikace

Tato kapitola se zabývá specifikací funkčních požadavků aplikace a jejím návrhem. Následně popisuje, jak bylo přistupováno k tvorbě prototypu a prvnímu uživatelskému testování.

4.1 Funkční požadavky

- **FR1 Vytvořit nový výlet**
  Systém umožní vytvořit *přihlášenému uživateli* nový výlet zvoleného typu (soukromý nebo veřejný).
  - *soukromý výlet* bude dostupný pouze pozvaným uživatelům
  - *veřejný výlet* bude dostupný všem uživatelům

- **FR2 Zobrazit dostupné výlety**
  Systém umožní *přihlášenému uživateli* zobrazit nadcházející veřejné výlety.

- **FR3 Filtrovat dostupné výlety**
  Systém umožní *přihlášenému uživateli* filtrovat dostupné výlety na základě jejich ceny, lokality a data konání.

- **FR4 Zobrazit údaje o veřejném výletu**
  Systém umožní *přihlášenému uživateli* zobrazit obecné údaje o konkrétním veřejném výletu. Podrobnější informace budou dostupné pouze *účastníkům výletu*.

- **FR5 Zobrazit profil jiného uživatele**
  Systém umožní *přihlášenému uživateli* zobrazit profil jiného uživatele.

- **FR6 Poslat zprávu jinému uživateli**
  Systém umožní *přihlášenému uživateli* poslat zprávu jinému uživateli.

- **FR7 Zobrazit zprávu**
  Systém umožní *přihlášenému uživateli* zobrazit zprávy, které obdržel. Po zobrazení zpráv budou zprávy označené jako přečtené.

- **FR8 Projevit zájem o veřejný výlet**
  Systém umožní *přihlášenému uživateli* přihlásit se na konkrétní veřejný výlet.

- **FR9 Zobrazit přihlášky na výlet**
  Systém umožní *přihlášenému uživateli* zobrazit jeho přihlášky na výlet a jejich detail.

- **FR10 Zrušit přihlášku na výlet**
  Systém umožní *přihlášenému uživateli* zrušit nepotvrzenou přihlášku na výlet.

- **FR11 Zobrazit pozvánky na výlet**

Systém umožní *přihlášenému uživateli* zobrazit jeho pozvánky na výlet a jejich detail.

- **FR12 Odpovědět na pozvánku na výlet**
  Systém umožní pozvanému *přihlášenému uživateli* odpovědět na pozvánku na výlet kladně či záporně.
- **FR13 Zobrazit nadcházející výlety**
  Systém umožní *přihlášenému uživateli* zobrazit jeho nadcházející výlety, včetně právě probíhajícího výletu, a jejich detail.
- **FR14 Zobrazit proběhnuté výlety**
  Systém umožní *přihlášenému uživateli* zobrazit jeho proběhnuté výlety a jejich detail.
- **FR15 Zobrazit vlastní profil**
  Systém umožní *přihlášenému uživateli* zobrazit vlastní profi.
- **FR16 Editovat profil a osobní údaje**
  Systém umožní *přihlášenému uživateli* upravit některé údaje v jeho profilu.
- **FR17 Zobrazit notifikaci**
  Systém umožní *přihlášenému uživateli* zobrazit obdržené notifikace.
- **FR18 Zobrazit vytvořené výlety**
  Systém umožní *přihlášenému uživateli* zobrazit jím vytvořené výlety.
- **FR19 Upravit údaje o výletu**
  Systém umožní *pořadateli výletu* upravit údaje o vytvořeném výletu.
- **FR20 Smazat výlet**
  Systém umožní *pořadateli výletu* smazat výlet. Ostatním účastníkům bude odeslána notifikace s kontaktními údaji pořadatele.
- **FR21 Pozvat na výlet konkrétní uživatele**
  Systém umožní *pořadateli výletu* pozvat na výlet konkrétní registrované uživatele.
- **FR22 Zrušit pozvánku**
  Systém umožní *pořadateli výletu* zrušit pozvánku, kterou vytvořil
- **FR23 Odpovědět na žádost o přidání se k výletu**
  Systém umožní *pořadateli výletu* potvrdit nebo zamítnout přihlášku na výlet.
- **FR24 Přidat dopravu ke konkrétnímu výletu**
  Systém umožní *účastníkovi výletu* přidat do itineráře výletu informace o dopravě.
- **FR25 Přidat ubytování ke konkrétnímu výletu**
  Systém umožní *účastníkovi výletu* přidat do itineráře výletu informace o ubytování. Uživatel bude mít na výběr mezi přidáním vlastního ubytování a vybráním ubytování ze seznamu navrhovaných ubytování v destinaci výletu.
- **FR26 Přidat aktivitu ke konkrétnímu výletu**
  Systém umožní *účastníkovi výletu* přidat do itineráře aktivitu. Uživatel bude mít na výběr mezi přidáním vlastní aktivity a vybráním aktivity ze seznamu navrhovaných aktivit v destinaci výletu.
- **FR27 Upravit dopravu**
  Systém umožní *účastníkovi výletu* upravit v itineráři existující dopravu.

- **FR28 Upravit ubytování**
  Systém umožní *účastníkovi výletu* upravit v itineráři existující ubytování.

- **FR29 Upravit aktivitu**
  Systém umožní *účastníkovi výletu* upravit v itineráři existující aktivitu.

- **FR30 Smazat dopravu**
  Systém umožní *účastníkovi výletu* smazat v itineráři existující dopravu.

- **FR31 Smazat ubytování**
  Systém umožní *účastníkovi výletu* smazat v itineráři existující ubytování.

- **FR32 Smazat aktivitu**
  Systém umožní *účastníkovi výletu* smazat v itineráři existující aktivitu.

- **FR33 Poslat zprávu spoluúčastníkům výletu**
  Systém umožní *účastníkovi výletu* poslat skupinovou zprávu všem účastníkům výletu.

- **FR34 Založit účet**
  Systém umožní *nepřihlášenému uživateli* založit nový účet.

- **FR35 Přihlásit se**
  Systém umožní *nepřihlášenému uživateli* přihlásit se do svého účtu.

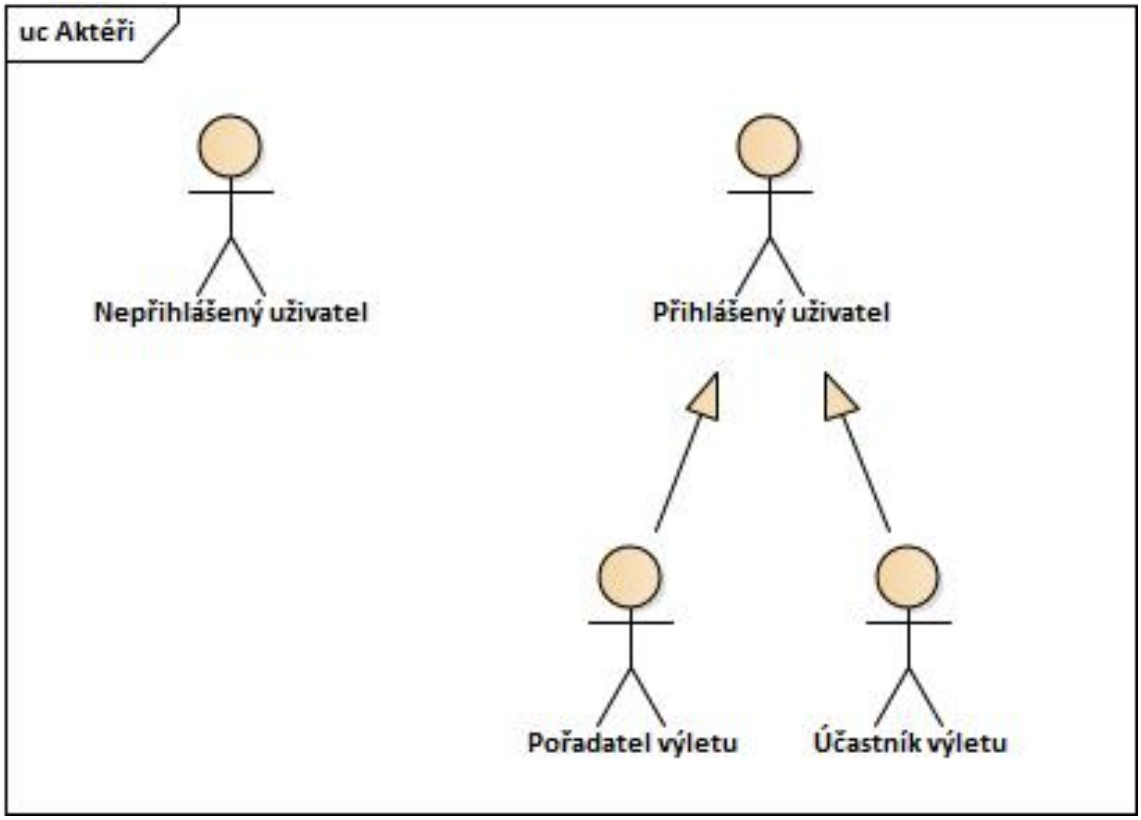
4.2 Případy užití

Funkční požadavky byly zpracovány do diagramů případů užití, které zachycují role uživatelů aplikace a znázorňují ke kterým funkcionalitám aplikace jim bude umožněn přístup.

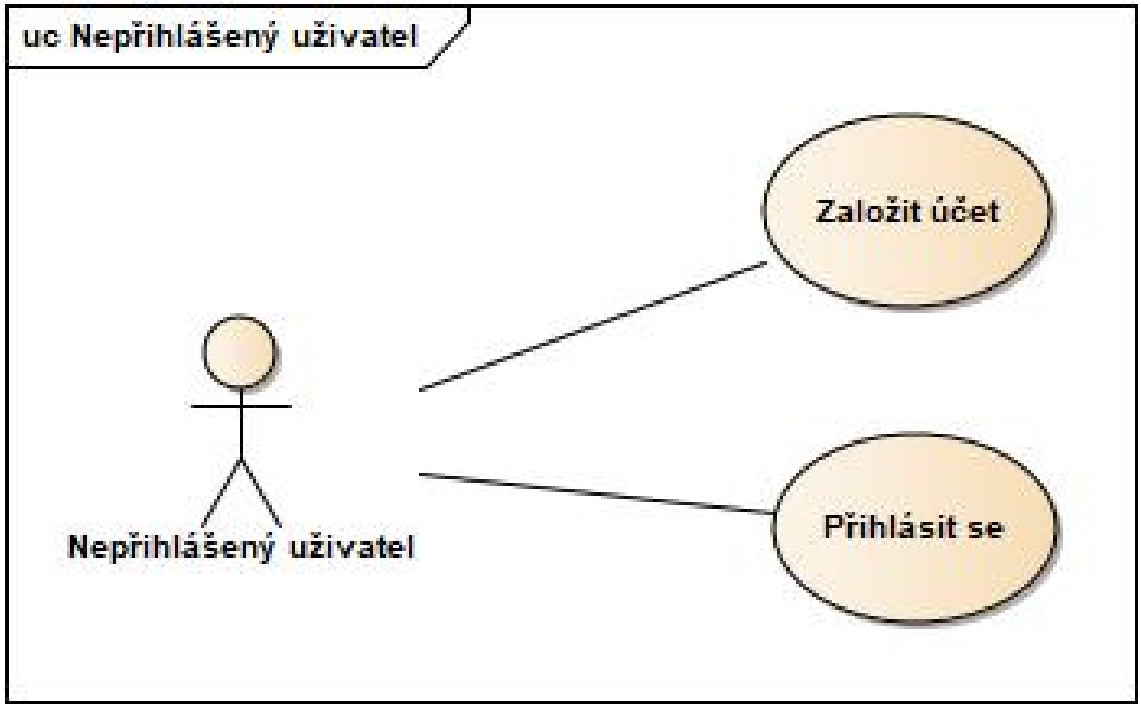
Uživatele systému lze rozčlenit do několika rolí, přičemž jednotlivé role se mohou překrývat (uživatel může být například pořadatelem výletu a zároveň účastníkem jiného výletu).

- nepřihlášený uživatel
- přihlášený uživatel
- pořadatel výletu
- účastník výletu

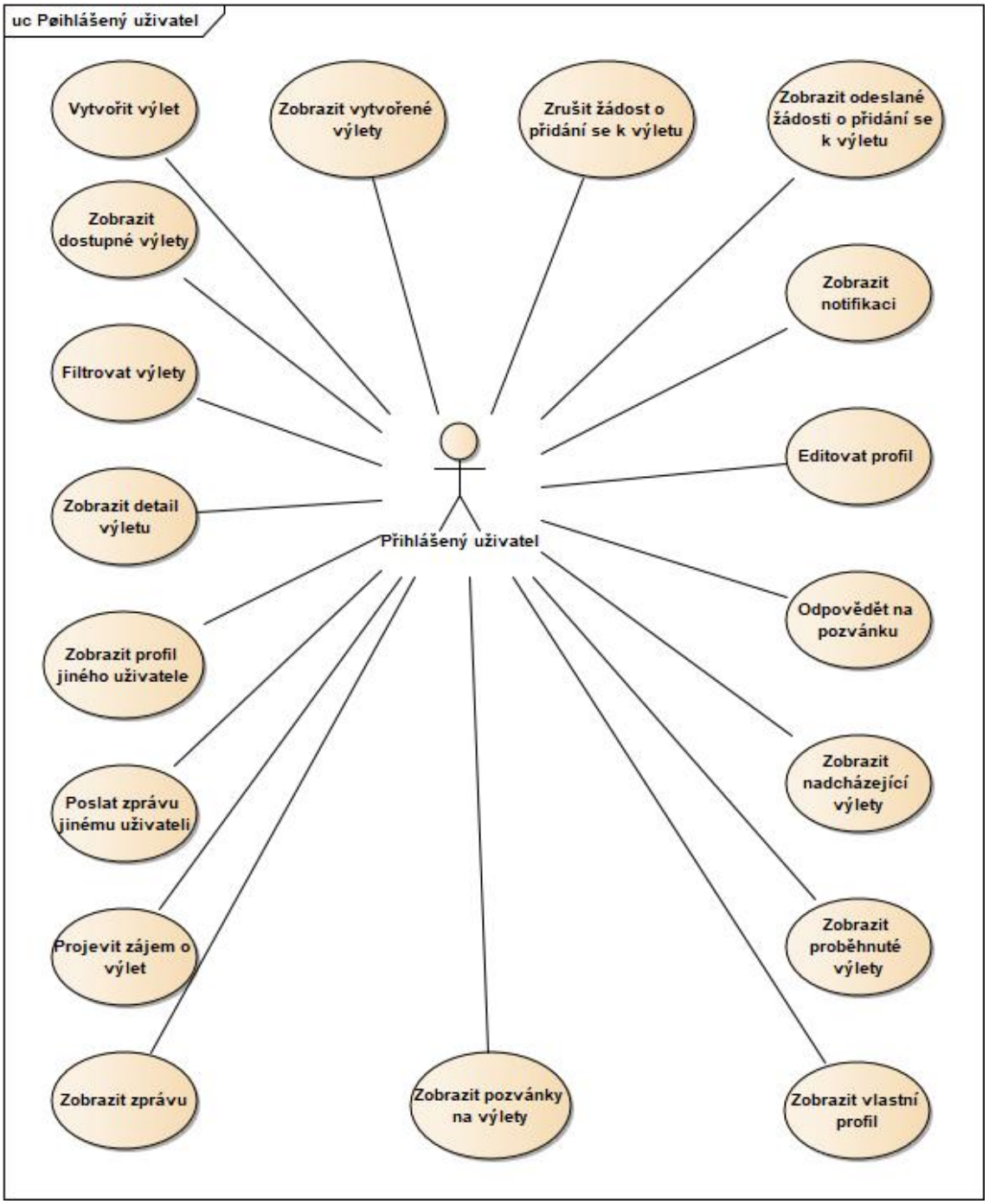
Vztahy mezi jednotlivými rolemi jsou zobrazeny na obrázku 5.1.



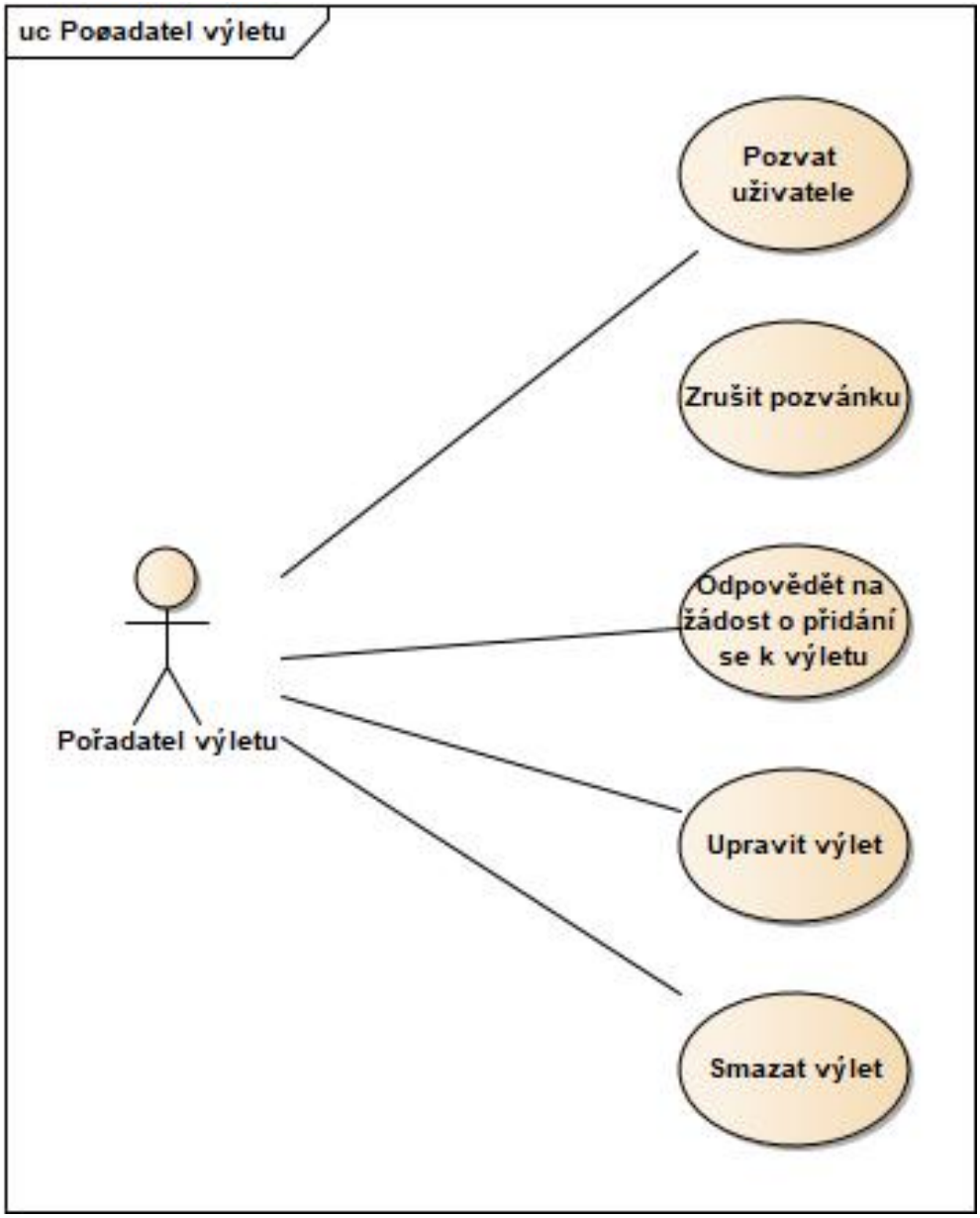
Obrázek 4.1. Hierarchie aktérů v aplikaci



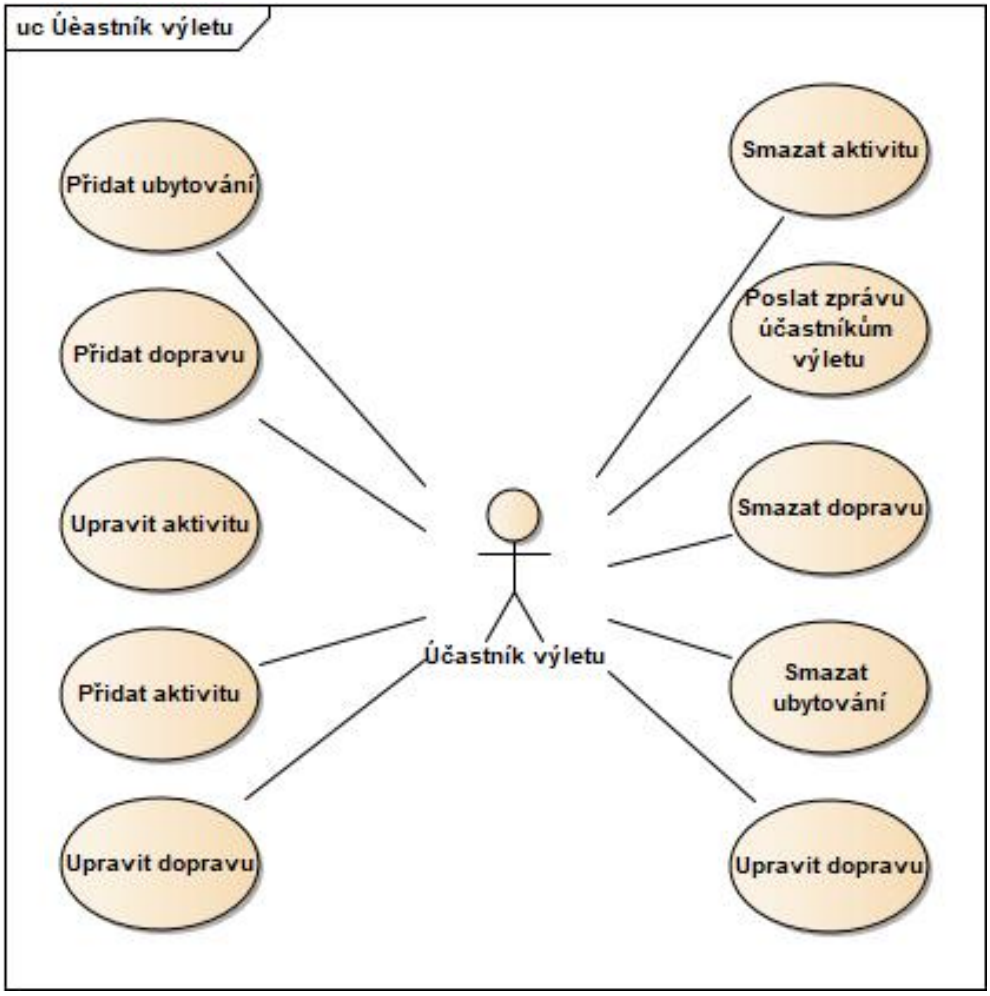
Obrázek 4.2. Případy užití nepřihlášeného uživatele



Obrázek 4.3. Případy užití přihlášeného uživatele



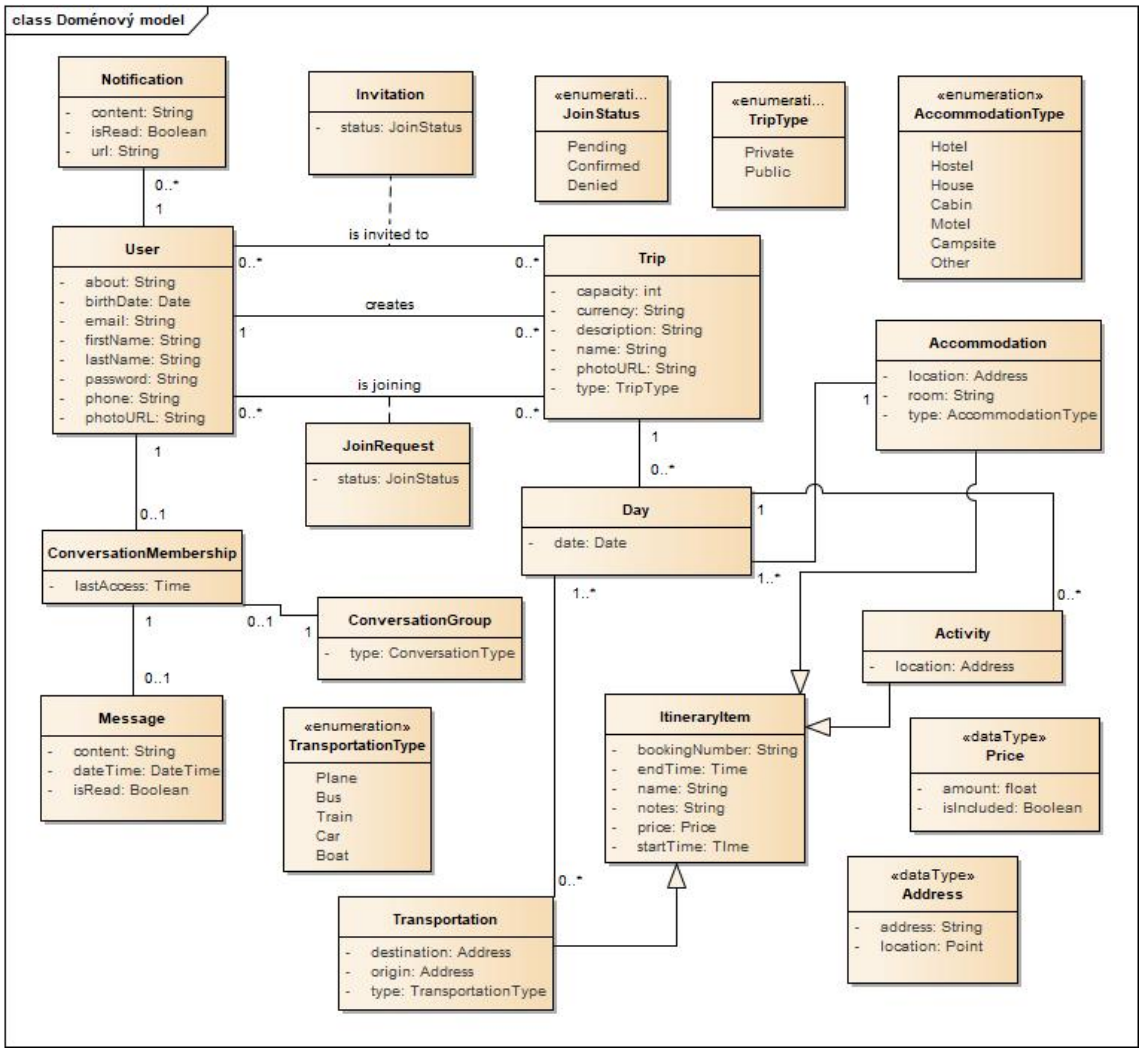
Obrázek 4.4. Případy užití pořadatele výletu



Obrázek 4.5. Případy užití účastníka výletu

4.3 Doménový model

Doménový model zachycuje atributy jednotlivých entit v aplikaci a vztahy mezi nimi. Model slouží jako podklad pro vytvoření databázového schématu a definici datových typů v aplikaci.



Obrázek 4.6. Doménový model

4.4 Prototyp a testování

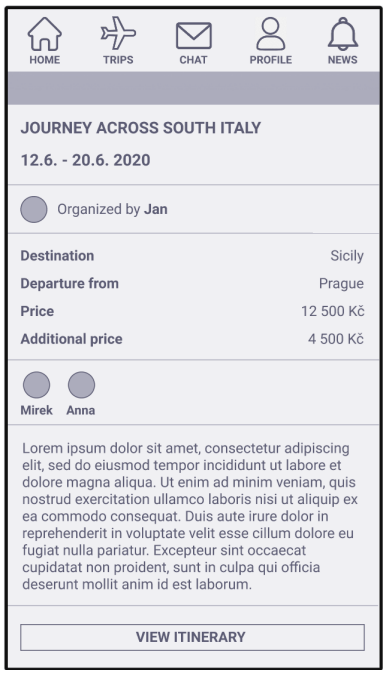
Dle výše uvedené specifikace byl zhotoven `prototyp`[1] v aplikaci Figma, která mimo jiné umožňuje vytváření interaktivních prototypů bez nutnosti použití kódu.

Prototyp byl koncipován jako low-fidelity protoyp, slouží tedy primárně k vizualizaci rozmístění prvků na obrazovkách a ke znázornění nejdůležitějších průchodů aplikací.[15] Grafické prvky v něm jsou velmi jednoduché a jejich účel je pouze poskytnout představu, jak budou informace uživateli zobrazeny. Prototyp umožňuje interakci s vybranými komponentami, jeho cílem však není poskytnout uživatelem plně interaktivní zážitek.

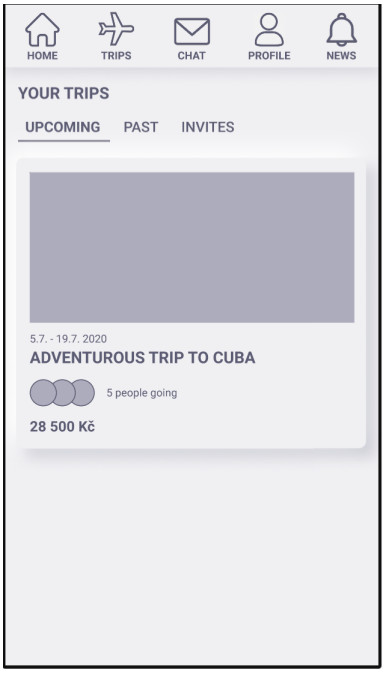
Prototyp byl před samotnou implementací aplikace otestován osmi uživateli. Ti byli seznámeni s tím, co očekávat od low-fidelity prototypu a jaká jsou jeho omezení, a následně byli požádáni o provedení několika různých průchodů aplikací. Se třemi z těchto uživatelů bylo provedeno skupinové testování. Při tomto typu testování účastníci procházejí prototyp společně, v menších skupinách./cite[chauncey]

[1] `https://www.figma.com/proto/FXAeTIaVaHh7SG6wCJjOAy/Trip-application`

Při testování byly identifikovány problémy při vytváření nového výletu. Původní proces byl příliš dlouhý, na základě doporučení uživatelů byl tedy ve výsledné aplikaci přepracován a některé jeho části(konkrétně vytváření itineráře a pozvání uživatelů) byly přesunuty do samostatných procesů. Dále nebylo několika uživatelům jasné, co znamenají jednotlivé *price* atributy v detailu výletu. Tento problém byl v aplikaci vyřešen přidáním tooltipů s doplňujícími informacemi.



Obrázek 4.7. Ukázka detailu výletu v prototypu



Obrázek 4.8. Ukázka zobrazení nadcházejících výletů v prototypu

Kapitola 4
Implementace aplikace

Tato kapitola se věnuje výčtu a popisu nejdůležitějších technologií, které byly použity při implementaci aplikace.

5.5 TypeScript

Veškerý kód aplikace je psán v jazyce TypeScript, který je nadstavbou jazyka JavaScript. Narozdíl od něj však nabízí podporu statického typování, což znamená, že proměnné mohou mít jednoznačně definované datové typy a kontrola jejich správnosti probíhá před kompilací a spuštěním kódu. Díky tomu dokáže TypeScript snadněji identifikovat chyby a překlepy v kódu. TypeScript navíc poskytuje řadu funkcionalit pro IDE (např. našeptávání nebo anotace proměnných), které výrazně přispívají k lepšímu „developer experience“.[16]

5.6 Klientská část aplikace

5.6.1 React

Pro tvorbu klientské části byla využita knihovna React, která byla vyvinuta Facebookem v roce 2011. Ta je mezi vývojáři velmi oblíbená – v současné době ji na `githubu`[1] používá přes 3,5 milionů repozitářů. Důvodů pro popularitu Reactu je hned několik:

- automaticky překreslí stránku, bez nutnosti jejího znovunačtení, pokud na ní došlo ke změnám
- je efektivní a rychlý – implementuje algoritmy, díky kterým jsou většinou překresleny pouze komponenty, ve kterých změna proběhla
- podporuje komponentový přístup k psaní kódu, díky čemuž je kód možné snadno znovu použít a jednoduše škálovat[17]
- disponuje rozsáhlou komunitou vývojářů, díky čemuž je k dispozici mnoho knihoven pro práci s Reactem a také mnoho tutoriálů

Při psaní kódu budou využity nejnovější funkcionality Reactu; místo starších `metod životního cyklu`[2] bude použit novější standard `hooks`[3], který mimo jiné umožňuje lepší znovupoužitelnost kódu.

5.6.2 Stylování

Při tvorbě aplikace byla použita knihovna Material-UI. Hlavní motivací pro použití této konkrétní knihovně byl fakt, že knihovna poskytuje široký výběr komponent navržených

[1] `https://github.com/facebook/react`
[2] `https://reactjs.org/docs/state-and-lifecycle.html`
[3] `https://reactjs.org/docs/hooks-intro.html`

dle standardů Material Designu. Material Design je design systém vytvořený Googlem, který je navržen primárně pro mobilní zařízení.

Samotný Material-UI pak poskytuje několik funkcionalit, které umožňují jednoduchou implementaci responzivity. V aplikaci byly konkrétně použity:

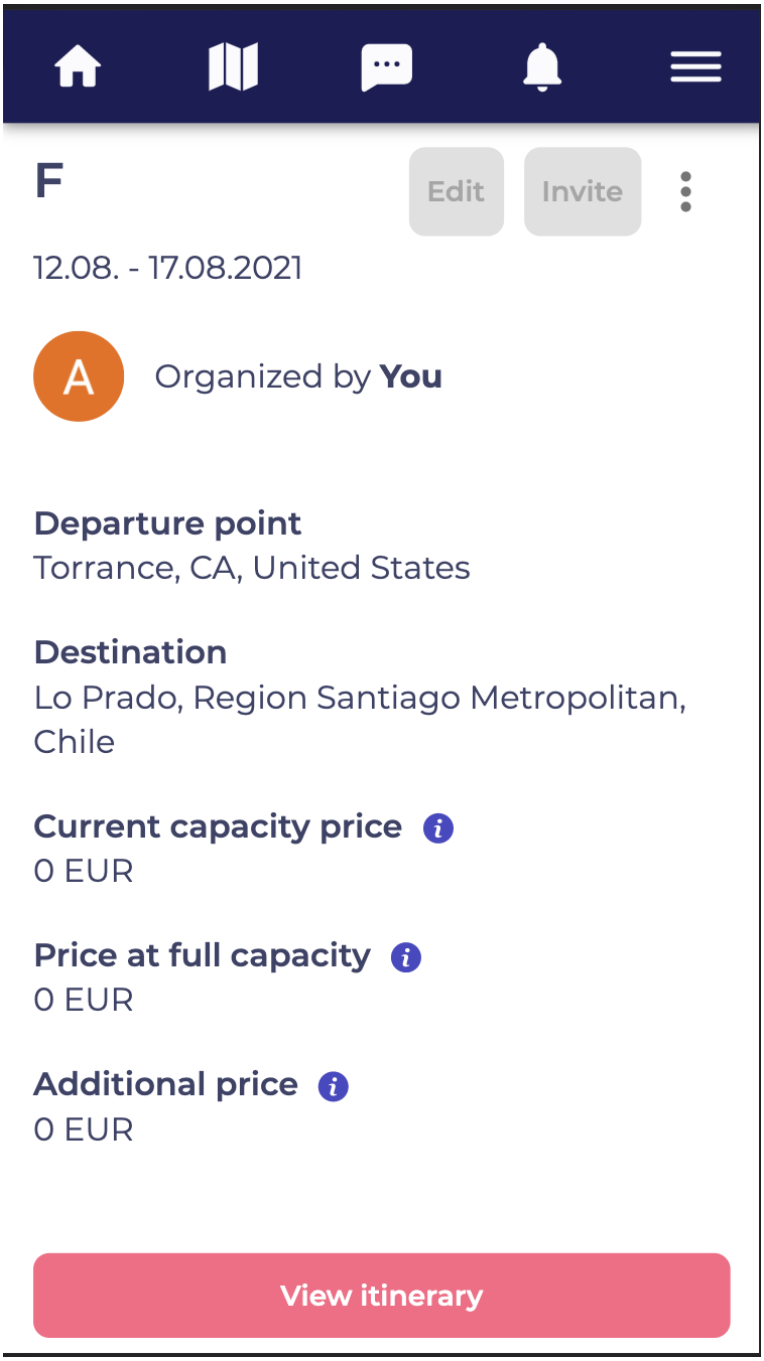
- useMediaQuery - React hook, který vrací hodnotu Boolean v závislosti na tom, zda daná velikost obrazovky odpovídá specifikovaným parametrům

```
const isLargeDevice =
useMediaQuery((min-width:${breakpoints['md-up']}px));
```

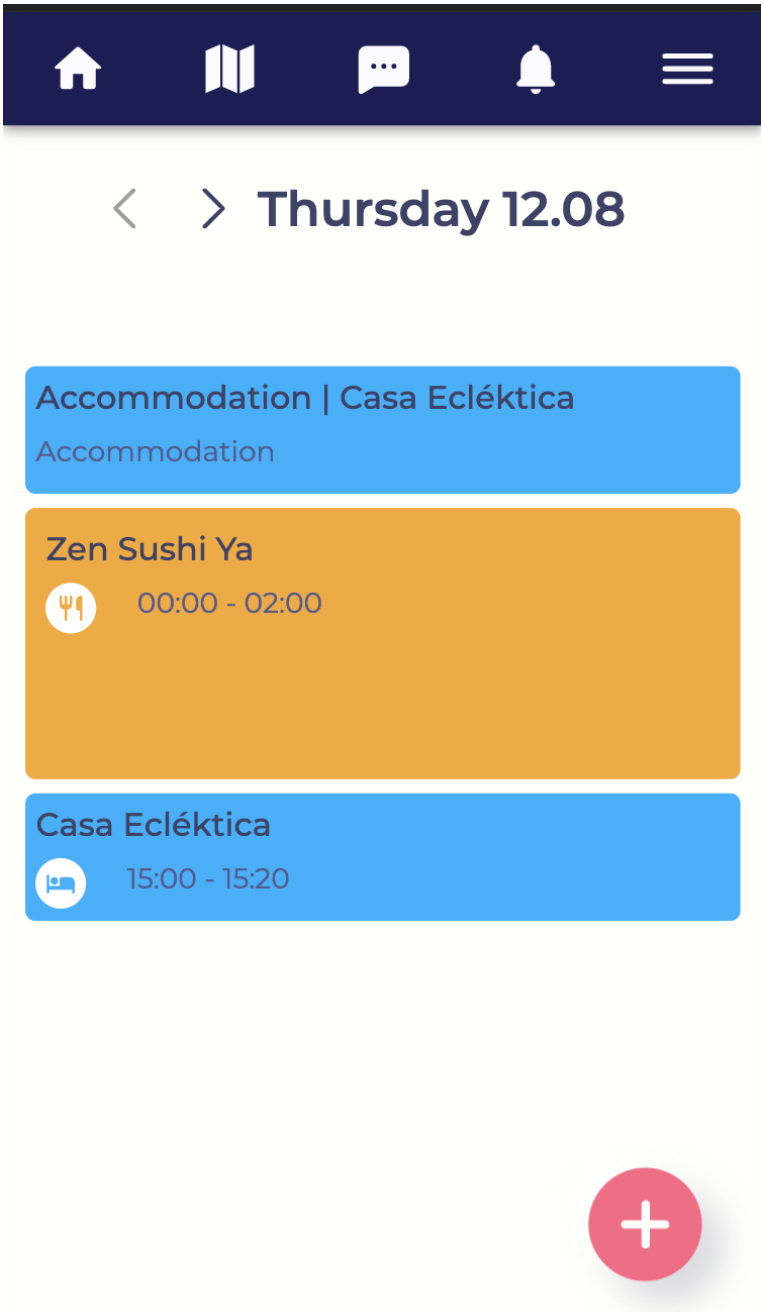
- Hidden - komponenta, která umožňuje skrytí jejího obsahu na specifikovaných velikostech obrazovky [18]

Spolu s Material-UI byla použita knihovna styled-components, která umožňuje psaní CSS prostřednictvím JavaScriptu. Mezi důvody, proč byla vybrána právě tato knihovna patří:

- podpora komponentového přístupu psaní aplikace
- automatický vendor-prefixing (přidání proprietárních prefixů k CSS atributům, které se liší svou implementací v jednotlivých prohlížečích)
- jednoduchá změna CSS atributu na základě vnitřního stavu komponenty[19]
- snadná integrace s Material-UI



Obrázek 4.9. Ukázka detailu výletu v aplikaci



Obrázek 4.10. Ukázka itineráře výletu v aplikaci

5.6 Serverová část aplikace

5.7.3 Node.js

Serverová část aplikace využívá technologii Node.js. Jedná se o prostředí pro spuštění JavaScriptu postavené nad javascriptovým enginem V8, jehož účelem je umožnění spuštění javascriptového kódu mimo prohlížeč. Díky tomu je možné vytvářet backendové aplikace v JavaScriptu.[20] Jelikož je TypeScript před spuštěním aplikace kompilován do JavaScriptu, je Node.js rovněž kompatibilní i s TypeScriptem,. Podpora TypeScriptu

a tudíž jednotný jazyk napříč celou aplikací byla hlavní motivací pro výběr této technologie.

5.7.4 Nest.js

Spolu s Node.js byl použit framework Nest.js, který je postaven nad frameworkem Express.js. Narozdíl od něj ale uživatele nutí k použití určitých architekturálních vzorů:

- Module - Slouží k organizaci komponent v aplikaci. Jednotlivé moduly by měly obsahovat komponenty, které se věnují pouze určité oblasti aplikace.
- Controller(nebo Resolver v GraphQL) - Zpracovává požadavky klienta a reaguje na ně
- Service - Obsahuje business logiku, provádí složitější operace s daty v databázi za použití vrstvy Repository
- Repository - Umožňuje přímý přístup k datům v databázi a manipulaci s nimi [21]

Pro objektově relační mapování mezi entitami v kódu a entitami v databázi byla použita knihovna `TypeORM`[1].

5.8 Komunikace mezi klientem a serverem

Pro výměnu dat mezi serverem a klientskou části byly zváženy dva různé přístupy – REST a GraphQL. REST je softwarová architektura pro implementaci webových služeb, která pro dotazování využívá HTTP metody (GET, PUT, POST, DELETE). Při tvorbě API, které využívá REST, jsou pro přístup k datům definovány přístupové body a pro každý přístupový bod jsou určeny metody, které jsou pro něj povoleny.[22] Klientská část aplikace při použití RESTu nemá kontrolu nad tím, jak budou vypadat data, která získává ze serveru.

GraphQL je poměrně nová technologie, která byla poprvé vydána v roce 2015. Jedná se o dotazovací jazyk pro aplikační rozhraní. Na rozdíl od RESTu umožňuje klientovi definovat, jaká data má server poslat.[23]. Díky tomu je možné zabránit odesílání nadbytečných dat, která klient nevyužije. Podle experimentu provedeném mezi 22 studenty na Federal University of Minas Gerais v Brazílii je navíc implementace dotazů v GraphQL snadnější a intuitivnější než implementace stejných dotazů v RESTu. Účastníkům průzkumu průměrně zabrala implementace v GraphQL o 3 minuty méně než v RESTu.[24]

GraphQL je nezávislé na protokolu a je tedy možné používat jej i s jinými protokoly než HTTP. Kvůli této vlastnosti GraphQL nemůže využívat HTTP metody a stavové kódy HTTP, ze kterých je možné zjistit, zda došlo na serveru k nějaké chybě. GraphQL nicméně tuto skutečnost řeší tak, že posílá podrobné informace o těchto chybách ve speciálním poli s názvem errors a chyby je tak možné v aplikaci řádně ošetřit.[23]

Po zvážení všech pro a proti byla ke komunikaci vybrána technologie GraphQL a to zejména za účelem seznámení se s novou technologií.

5.8.1 ApolloGraphQL

V klientské části aplikace byla pro implementaci GraphQL dotazů použita knihovna ApolloGraphQL. Tato knihovna slouží nejen ke komunikaci s GraphQL serverem, ale

[1] `https://typeorm.io`

poskytuje také funkcionality, které umožňují jednodušší správu stavu aplikace a zároveň zajišťuje automatické cachovaní načtených dat.

V knihovně je možné nastavit, jaký přístup bude při zavolání dotazu použit. Na výběr je mezi několika druhy, výchozím přístupem je *cache-first*. Ten v případě, že požadovaná data existují v cachi, vrací cachovaná data a eliminuje komunikaci se serverem. Pokud chce vývojář zcela předejít použití cachovaných dat, může použít přístup *network-only*. ,[]

GraphQL používá dva druhy operací.Tou první, která je používána pro získání dat ze serveru, je query. Apollo umožňuje provedení *query* díky hooku *useQuery*. Ten kromě dat ze serveru(či z cache), vrací také atributy loading a error, díky nimž je možné zobrazit uživateli relevantní obsah.

```
const { data, loading, error } = useQuery<getUser>(GET_CURRENT_USER)
```

Druhou operací, sloužící k manipulaci dat na serveru je mutation. Tuto operaci Apollo implementuje v hooku useMutation, který vrací funkci, která po zavolání provede vybranou mutaci.

```
const [createMessage] = useMutation<sendMessage, sendMessageVariables>(
SEND_MESSAGE
);
```

5.9 Databáze

Při výběru databáze byly posuzovány dva různé způsoby ukládání dat – relační databáze a databáze NoSQL.

Relační databáze jsou známy již mnoho desítek let. Data v nich jsou uchovávána v tabulkách s pevně daným schématem, kde jsou prostřednictvím cizích klíčů realizovány vztahy mezi jednotlivými entitami.[25] Díky tomu jsou tyto databáze vhodné pro uchovávání dat, která jsou mezi sebou značně provázána. Výhodou relačních databází je zajištění takzvaných ACID principů v transakcích:

- atomicita
- konzistence
- izolovanost
- trvalost [26]

NoSQL (nejen SQL) databáze jsou novějším přístupem k uchovávání dat. To je realizováno většinou buď ve formě párů klíč-hodnota, nebo v souborech podobných JSONu. Tyto databáze by měly být rychlejší a škálovatelnější. Dle průzkumu provedeném na University of Auckland na Novém Zélandu se však rychlost různých NoSQL databází značně liší a některé jsou dokonce pomalejší než databáze relační. [27]

Pro aplikaci této velikosti není výběr databáze tolik důležitý. Pokud by ale měla být aplikace skutečně uvedena na trh, bylo by nejspíše vhodnější použít relační databázi, vzhledem k povaze dat v aplikaci a také nutnosti konzistentních a spolehlivých dotazů. S ohledem na familiaritu s jejím ekosystémem bude tedy při tvorbě aplikace použita open source databáze PostgreSQL. Zvažována byla také NoSQL databáze MongoDB, která se v roce 2018 stala první NoSQL databází zajišťující ACID principy[28] a zároveň byla jednou z nejrychlejších databází ve výše uvedeném průzkumu.

5.10 PostGIS

Aby bylo možné filtrovat výlety na základě jejich destinace nebo výchozího bodu, bylo nutné nainstalovat do databáze rozšíření `PostGIS`[1], které zprostředkovává práci s prostorovými objekty. PostGIS poskytuje datový typ geography, který umožňuje uložit do databáze zeměpisnou šířku a výšku. Zároveň nabízí několik funkcí vhodných pro práci s tímto datovým typem. V aplikaci byly konkrétně použity tyto funkce:

- STDWithin(location1, location2, radius): určuje, zda jsou od sebe dvě místa vzdálená nanejvýš [radius] metrů.

```
query.where(
STDWithin(departure.location::geography, STMakePoint(:lat,:lng)::geography,
:radius),
{
lat: departureTown[0],
lng: departureTown[1],
radius: departureRadius * 1000, //Převedení kilometrů na metry
}
```

- STMakePoint(latitude, longitude): vytvoří nový datový typ geometry odpovídající specifikované zeměpisné šířce a výšce

5.11 Použité API

Pro optimalizaci UX byly při implementaci funkcionalit souvisejících s vytvářením výletu a itineráře použity dvě API třetích stran.

5.11.1 HERE Geocoding Search API

`HERE API`[2] byla použita k usnadnění práce s adresami v aplikaci. Díky endpointu Autocomplete

```
GET https://autocomplete.search.hereapi.com/v1/autocomplete
```

nemusí uživatel ručně zadávat celou adresu. Namísto toho si může vybrat z návrhů, které vrací tento endpoint. Dále byly využity endpointy Geocode

```
GET https://geocode.search.hereapi.com/v1/geocode
```

a Reverse geocode

```
GET https://geocode.search.hereapi.com/v1/revgeocode
```

pro překlad adresy do koordinant a naopak. Značnou nevýhodou této API, která byla ale zjištěna až ve velmi pozdní fázi vývoje, je absence podpory endpointu Geocode pro Japonsko ve Freemium verzi.

[1] `https://postgis.net/`
[2] `https://developer.here.com/documentation/geocoding-search-api/dev_guide/index.html`

5.11.2 Foursquare Places API

Z `této API`[1] byly využity dva endpointy a to konkrétně:

- Venue recommendations - vrací seznam míst v blízkosti dané lokality, umožňuje filtrování těchto míst dle kategorií

```
GET https://api.foursquare.com/v2/venues/explore
```

- Venue detail - vrací detail konkrétního místa

```
GET https://api.foursquare.com/v2/venues/VENUEID
\enditems
Oba tyto endpointy byly použity pro účely zjednodušení vytváření iti-
```

neráře. Uživatel má díky informacím z těchto endpointů přímo v aplikaci možnost procházet doporučené podniky a ubytování v destinaci výletu a jednoduše je přidat do itineráře.

[1] https://foursquare.com/products/places/

Kapitola 6
Uživatelské testování aplikace

Dokončená verze aplikace byla otestována několika koncovými uživateli, za účelem odhalení nedostatků a zjištění použitelnosti aplikace.

6.1 Výběr účastníků a průběh testování

Cílovou skupinou aplikace jsou primárně aktivní lidé ve věku 20-35 let, kteří rádi tráví čas s novými lidmi. Při výběru účastníků testování byl kladen důraz na to, aby byli přítomni jak zástupci z této skupiny, tak i jedinci, kteří nemají tolik zkušeností s používáním webových a mobilních aplikací.

Nakonec bylo vybráno šest následujících účastníků

- Účastník 1 - žena, 32 let, občasná cestovatelka
- Účastník 2 - žena, 27 let, organizátorka výletů a zaměstnankyně cestovní kanceláře
- Účastník 3 - muž, 65 let, méně pokročilý uživatel
- Účastník 4 - muž, 33 let, cestovatel využívající low-cost varianty cestování
- Účastník 5 - muž, 24 let, velmi pokročilý uživatel
- Účastník 6 - muž, 22 let, student v zahraničí

Vzhledem k současné pandemické situaci probíhalo testování se čtyřmi účastníky na dálku prostřednictvím aplikace Google Meet. Účastníci byli požádáni, aby v průběhu testování sdíleli svou obrazovku a upozorněni na to, že část testu může být nahrávána. Tato testování tedy proběhla na počítačích.

Se zbývajícími dvěma uživateli bylo testování provedeno osobně a aplikace byla tentokrát testována na mobilních zařízeních.

Pro potřeby testování byla aplikace nasazena na web a naplněna testovacími daty. Backendová část běžela na platformě `Heroku`[1], klientská část aplikace byla nasazena pomocí platformy `Vercel`[2].

6.2 Testovací úkoly

Každý z účastníků byl požádán, aby v aplikaci provedl následující úkony: addTo/begitems

- Zaregistrujte se do aplikace a vytvořte si nový profil.
- Vyfiltrujte z dostupných výletů pouze výlety, jejichž cena nepřesahuje 700 EUR.
- Zvolte jeden z dostupných výletů a zažádejte o připojení se k výletu.
- Odešlete zprávu pořadateli vámi zvoleného výletu.
- Vytvořte vlastní výlet.
- Přidejte k výletu jednu z aktivit, která je dostupná v návrzích.

[1] `https://www.heroku.com/about`
[2] `https://vercel.com/dashboard`

- Manuálně přidejte k výletu ubytování.
- Pozvěte na výlet uživatele se jménem Jan Novák.
- Upravte zvolený výlet tak, aby neobsahoval původní datum.

Jednotlivé úkony účastníci prováděli sami, bez vnějšího zásahu moderátora. Ten mohl zasáhnout až poté, kdy uživatel na některém z úkolů strávil nepřiměřené množství času. K takové situaci však v průběhu testování nedošlo.

Po skončení testování byl s každým z účastníků proveden krátký pohovor, jehož účelem bylo zejména identifikovat, jaké problémy v aplikaci účastník objevil.

6.3 Výsledky testování aplikace

Většina uživatelů zvládla průchod aplikace bez větších problémů, i tak byly v aplikaci identifikovány některé nedostatky. Ty byly ohodnoceny dle `Nielsonovy škály použitelnosti`[1].

- Na stránce itineráře chybí tlačítko Zpět k výletu - 2
- Při přidávání aktivit do itineráře není pole adresy limitováno pouze na lokality v okolí destinace výletu - 3
- Chybová hláška při špatně zadaném datu není vždy ve formátu, který je jednoduše čitelný pro uživatele - 2
- Při otevření formuláře pro přidání aktivity nevede tlačítko zpět zpátky na stránku s doporučenými aktivitami, ale na stránku itineráře - 2
- V mobilní verzi je na stránce itineráře občas příliš velká mezera mezi názvem dne a první položkou - 1

Výsledky testování byly tedy vcelku uspokojivé, účastníci oceňovali vizuálně hezké UI, nicméně identifikované problémy by bylo vhodné v další iteraci odstranit.

[1] `https://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/`

Kapitola 7
Závěr

Tato práce se měla zaměřovat primárně na implementaci a uživatelské testování progresivní webové aplikace pro plánování výletů. Zadání se povedlo s menšími výhradami splnit, do budoucna by však aplikaci prospělo přidání nových funkcionalit a vylepšení některých stávajících.

7.1 Prostor pro vylepšení

V další fázi vývoje aplikace by bylo optimální zaměřit se především na vylepšení tvorby itineráře. Současná verze aplikace umožňuje přidání pouze jednoho ubytování pro celý výlet, za ideálního stavu by však mělo být možné nastavit pro každý den jiné ubytování. Podobný nedostatek lze pozorovat i u dopravy, kterou lze nyní přidat pouze k prvnímu a poslednímu dni výletu. Chybí tedy možnost přidání dopravy v průběhu výletu. Dále by bylo optimální upravit vytváření výletu tak, aby jako destinace a výchozí adresa mohly být použity kromě měst i země, či kraje.

7.2 Nové funkcionality

V budoucnu by aplikace mohla být rozšířena o následující funkce:

- Možnost personalizace itineráře pro každého účastníka výletu
- Řazení výletů na hlavní stránce
- Podpora offline mutací - uživatel by měl mít možnost provádět některé akce(např. zaslání zprávy) i pokud je offline, data by byla zaslána na server po opětovném získání připojení

Literatura

[1] TripIt. *We Love Our Users! All 13 MILLION of You! https://T.co/8sVkqvatsM.* 2016.
twitter.com/TripIt/status/779408174478524416?s=20.

[2] Rob Cameron. The man behind one of the world's most successful apps. *BBC News*. 2014,

[3] *Progressive web apps (PWAs)*. 2020.
https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps.

[4] Gaurav Kaushik. Progressive Web App - The future of Web Development. *International Journal for Research in Applied Science and Engineering Technology*. 2019, 7 495-498. DOI 10.22214/ijraset.2019.7077.

[5] *What makes a good Progressive Web App?* 2020.
https://web.dev/pwa-checklist/.

[6] *Web app manifests*. 2020.
https://developer.mozilla.org/en-US/docs/Web/Manifest.

[7] *Using Web Workers - Web APIs: MDN*.
https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers.

[8] *Service Worker API*. 2020.
https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API.

[9] *HTTPS*. 2020.
https://developer.mozilla.org/en-US/docs/Glossary/HTTPS.

[10] *What does it take to be installable?*
https://web.dev/install-criteria/.

[11] *Installing and uninstalling web apps - Progressive web apps (PWAs): MDN*.
https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Developer_guide/Installing.

[12] *Notifications API - Web APIs: MDN*.
https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API.

[13] *Push API*. 2020,.
https://developer.mozilla.org/en-US/docs/Web/API/Push_API.

[14] *FCM Architectural Overview — Firebase*.
https://firebase.google.com/docs/cloud-messaging/fcm-architecture.

[15] Morten Hertzum. *Usability testing: a practitioners guide to evaluating the user experience*. Morgan Claypool, 2020.

[16] *The starting point for learning TypeScript*.
https://www.typescriptlang.org/docs/home.

[17] *Getting started*.
https://reactjs.org/docs/getting-started.html.

[18] *UI: A popular React UI framework.*
https://material-ui.com/.

[19] *Basics.*
https://styled-components.com/docs/basics.

[20] *Introduction to Node.js*. 2020.
https://nodejs.dev/learn/introduction-to-nodejs.

[21] *Documentation: NestJS - A progressive Node.js framework.*
https://docs.nestjs.com/.

[22] Fernando Doglio. *Rest API development with Node.JS: manage and understand the full capabilities of successful REST development*. Apress, 2018.

[23] *GraphQL*. 2018.
http://spec.graphql.org/June2018.

[24] Gleison Brito a Marco Tulio Valente. REST vs GraphQL: A Controlled Experiment. *2020 IEEE International Conference on Software Architecture (ICSA)*. 2020,

[25] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*. 1970, 13 (6), 377–387. DOI 10.1145/362384.362685.

[26] S. Ghule a R. Vadali. Transformation of SQL system to NoSQL system and performing data analytics using SVM. 2017, 883-887.

[27] Y. Li a S. Manoharan. A performance comparison of SQL and NoSQL databases. 2013, 15-19.

[28] Eliot Horowitz. *MongoDB Drops ACID: MongoDB Blog*. 2018.
https://www.mongodb.com/blog/post/multi-document-transactions-in-mongodb.