VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

APLIKACE PRO PLÁNOVÁNÍ ITINERÁŘE

ITINERARY PLANNING APPLICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ FILÁK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MARTIN KRČMA**
SUPERVISOR

BRNO 2018

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Filák Tomáš**
Obor: Informační technologie
Téma: **Aplikace pro plánování itineráře**
Itirenary Planning Application
Kategorie: Softwarové inženýrství

Pokyny:

1. Nastudujte metodiky sestavování cestovního itineráře.
2. Prostudujte dostupné aplikace. pro řešení této úlohy.
3. Navrhněte intuitivní aplikaci pro plánování itineráře, která bude podporovat vkládání informací o dopravě, časech, nákladech a dalších vlastnostech ke všem položkám a úsekům itineráře. Aplikace bude podporovat vedení různých verzí itineráře a jejich vzájemné kombinování s automatickým přepočtem vložených údajů.
4. Navrženou aplikaci implementujte.
5. Vyhodnoťte vytvořenou aplikaci, získejte zpětnou odezvu od skupiny testerů a zhodnoťte dosažené výsledky.
6. Navrhněte další možné směry práce.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Krčma Martin, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2
L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

Předmětem této bakalářské práce je návrh a implementace aplikace pro plánování itineráře. Aplikace má za cíl zjednodušit plánování celé cesty a poskytnou uživateli nástroj, díky kterému si bude moci vytvořit více verzí cesty. Tyto verze může dále porovnávat mezi sebou jak z finančního tak i časového hlediska. Aplikace poskytuje uživateli přehledné a atraktivní uživatelské rozhraní spolu s funkcemi, které celý proces plánování usnadňují. Mezi takové funkce patří například našeptávání lokací z databáze Google Map nebo automatický přepočet měn dle aktuálního kurzu České národní banky.

Abstract

The subject of this bachelor thesis is the design and implementation of an application for planning itineraries. The application aims to simplify the planning of the entire travel and provide a tool for the user to create multiple travel versions. These versions can also be compared with each other in both financial and time aspects. The application provides a clear and attractive user's interface along with features which make the entire process of planning easier. Such features, for example, include autocompleting of location names from the Google Map database or automatic currency conversion according to the current Czech national bank rate.

Klíčová slova

Web, webová aplikace, PHP, Laravel, Google Mapy, itinerář, cestování

Keywords

Web, web aplication, PHP, Laravel, Google Maps, itinerary, traveling

Citace

FILÁK, Tomáš. *Aplikace pro plánování itineráře*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Krčma

Aplikace pro plánování itineráře

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Krčmy. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Filák
13. května 2018

Poděkování

Rád bych poděkoval panu Ing. Martinu Krčmovi za vedení této bakalářské práce a za rady a věcné poznámky, které mi poskytoval v průběhu tvorby této práce. Také chci poděkovat rodině a své přítelkyni za podporu během celého studia.

Obsah

Kapitola 1

Úvod

Cestování je velmi oblíbenou činností lidí po celém světě. Lidé rádi poznávají nová místa, kulturu, jídlo nebo si chtějí jen odpočinout někde na pláži. V posledních letech je čím dál více oblíbené cestování tak zvaně na vlastní pěst, tedy bez cestovní kanceláře. Tento styl cestování spolu ale nese spoustu starostí s plánováním a zařizováním, aby vše proběhlo podle očekávání. A právě zjednodušení plánování takové cesty má za cíl tato bakalářská práce.

Jejím cílem je navrhnout a vytvořit aplikaci pro intuitivní plánování itineráře. Tato aplikace by měla být nástrojem pro celkové vytvoření plánu cesty, jak z časového, tak finančního pohledu. Aplikace tedy bude umět pracovat s více měnami, několika verzemi cesty a vzájemným porovnáním těchto verzí. Zaměříme se na uživatelské rozhraní aplikace a pokusíme se jej implementovat přehledně a pro uživatele srozumitelně.

Níže je popsána teoretická příprava spolu s návrhem a samotnou implementací této aplikace. Dokument je rozdělen do kapitol, které se těmto bodům blíže věnují. V první kapitole se nachází tento úvod. Druhá kapitola obsahuje uvedení do problematiky celého zadání a analýzu konkurenčních aplikací. Třetí kapitola se věnuje teoretickému popsání všech použitých technologií jako je programovací jazyk nebo databázové systémy. Ve čtvrté kapitole je popsán framework Laravel, který využijeme pro implementaci této aplikace. Můžeme zde nalézt popis vybraných nástrojů, které Laravel poskytuje a které v této práci využijeme. Pátá kapitola se věnuje návrhu celé aplikace za pomoci diagramu případů užití nebo ER diagramu pro návrh databáze a dále je zde také popsán návrh uživatelského rozhraní a jeho reprezentace v podobně wireframu. V této kapitole se také zaměříme na možnosti, které by nám mohlo nabídnout využití API Google map. Šestá kapitola je věnována samotné implementaci aplikace, kterou jsme v předchozí kapitole navrhli. Zabývá se jak samotnou konfigurací frameworku Laravel, tak i implementací frontendu, backendu a jejich následné propojení. V této kapitole je zahrnuto i testování aplikace a návrhy na její další rozšíření. V kapitole sedm je závěr a zhodnocení této práce.

Kapitola 2

Úvod do problematiky

Cestování je velmi oblíbená činnost lidí po celém světě. Samotný cestovní ruch se dá rozdělit na několik typů například podle místa, délky pobytu, způsobu dopravy a také podle způsobu organizace.Podle způsobu organizace se cestovní ruch dá rozdělit na dva druhy. Prvním je cestování prostřednictvím cestovní kanceláře. A tím druhým je individuální cestování, tedy cestování bez cestovní kanceláře.

Cílem cestování bývá poznávání nových míst a zážitků. Každá cesta ale vyžaduje dlouhé plánování a vybírání míst, které chce cestovatel navštívit. Pro tyto účely lze najít několik dostupných aplikací, které by měly celý proces plánování uživateli zjednodušit a ten se tak mohl soustředit opravdu jen na hledání nových míst a zážitků. S několika takovými aplikacemi provedeme v této kapitole analýzu a zjistíme tak konkurenceschopnost aplikace ještě před započetím samotného vývoje.

2.1 Oragnizované cestování

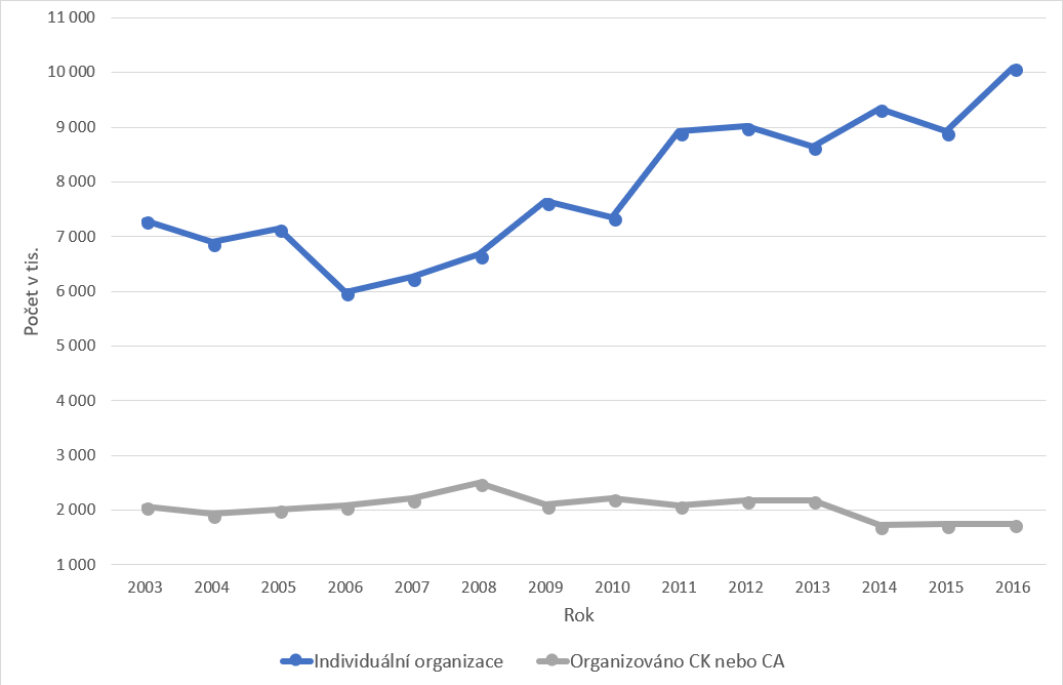
Organizované cestování probíhá prostřednictvím cestovní kanceláře nebo cestovní agentury. Cestovní kancelář i agentura je v České republice definována zákonem č. 159/1999 Sb. o některých podmínkách podnikání v oblasti cestovního ruchu. Cestovní kancelář je živností koncesovanou a vyznačuje se tím, že organizuje a prodává zájezdy, nese plnou zodpovědnost za jejich realizaci. Cestovní agentura je vázanou živností a prodej zájezdů zprostředkovává. V případě organizovaného cestování ponechávají účastníci veškerou organizaci cesty na těchto poskytovatelích cestovních služeb. Cestovní kancelář nebo agentura účastníkům zpravidla zajišťuje dopravu, ubytování, pobyt a služby s tím spojené. [8]

Mezí výhody organizovaného cestování patří zejména ušetřený čas účastníků při hledání vhodného ubytování, dopravy a plánovaní celé cesty. Nevýhodou pak mohou být vyšší náklady nebo nemožnost sestavení plánu cesty přesně podle svých představ. Dále je zde riziko úpadku pořádající cestovní kanceláře, což sebou nese spoustu problémů a celá dovolená se nemusí vůbec uskutečnit.

2.2 Individuální cestování

Individuální cestování neboli také cestování na vlastní pěst je cestování bez použití služeb cestovní kanceláře. Právě při tomto typu cestování je potřeba si celou cestu sám naplánovat. Člověk má tedy volnost a nemusí se ohlížet na harmonogram, který mu připravila cestovní kancelář. I proto je tento typ cestování v posledních letech velmi oblíbený.

Jak můžeme vidět na grafu níže podle Českého statistického úřadu individuální cestování několikanásobně převyšuje cestování organizované cestovními kanceláři, a navíc v posledních letech má rostoucí tendenci právě na úkor zájezdů pořádaných cestovními kancelářemi.[29]



Obrázek 2.1: Graf delších cest (4 a více přenocování) rezidentů v tuzemsku a do zahraničí (v tis.) rozdělený na individuální a organizované cesty

2.3 Itinerář

Pojem itinerář v turistice a cestovním ruchu označuje plán cesty obsahující seznam míst, případně úseků této cesty. Itinerář se může plánovat jak pro jednodenní výlet, tak pro delší dovolenou nebo služební cestu. Obvykle obsahuje místa s datem příjezdu a odjezdu, způsob dopravy na tato místa. Dále může obsahovat obecné informace o cestě, informace o letech, ubytování, cenách a podobně.

2.4 Analýza konkurence

Před vytvářením vlastního návrhu aplikace jsme provedli analýzu konkurence. Našli jsme několik volně dostupných aplikací, které nabízí funkci plánování itineráře. Zaměřili jsme se na funkce, které tyto aplikace poskytují, převážně na tvorbu více verzí itineráře a jejich porovnávání, práci s měnami a přehlednost celé aplikace. Často tvorbu itineráře poskytují mapové portály, na které jsme se při analýze konkurence zaměřili také. Dále si popíšeme jednotlivé aplikace a jejich výhody a nevýhody, na které jsme narazili. Cílem této analýzy je zjistit konkurenceschopnost naší aplikace, získat inspiraci a vyvarovat se chybám a nedostatkům konkurence.

2.4.1 Mapy.cz

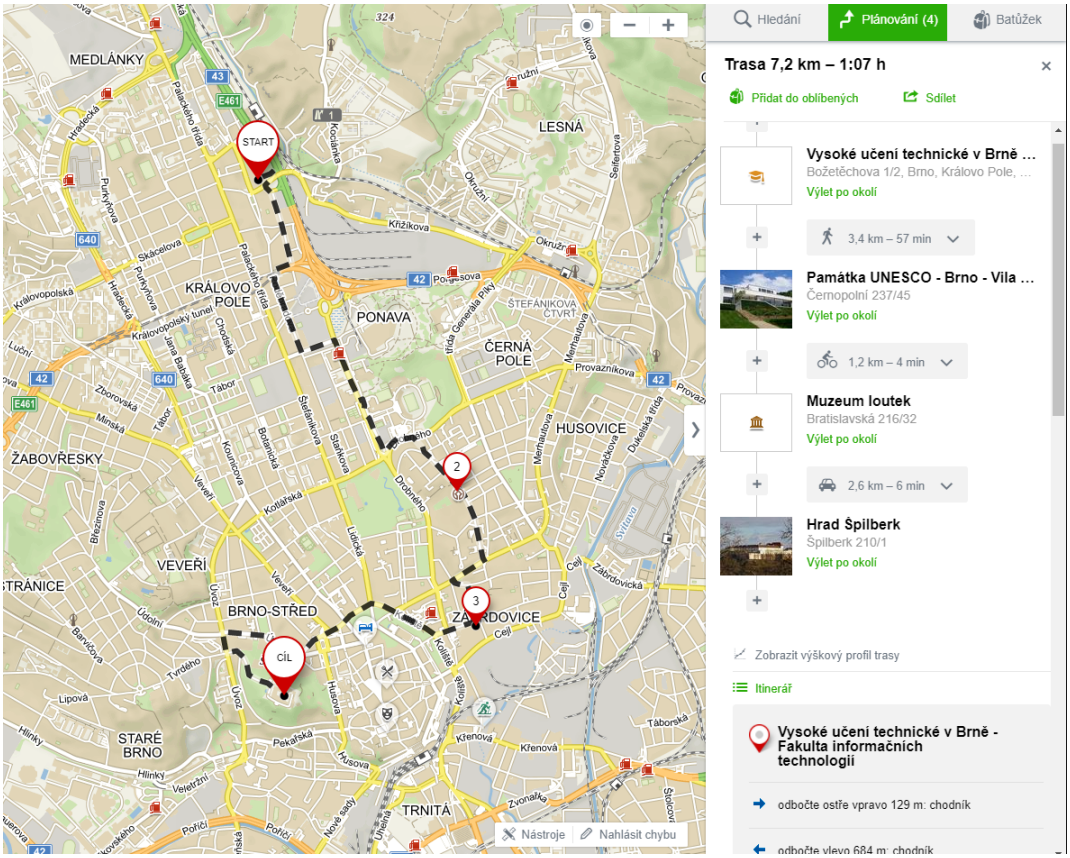
Jako první můžeme narazit na funkci plánování, která se nachází v mapové aplikaci Mapy.cz. Tato aplikace je vyvíjena jednou z největších českých internetových firem Seznam.cz, která s ní na českém trhu přímo konkuruje mapové aplikaci od Googlu. Zaměřuje se především na mapové podklady a funkce plánování tuto aplikaci jen doplňuje.

Funkce plánování umožňuje:

- vyhledávání míst v databázi Seznamu,
- přidávání míst chronologicky za sebe,
- plánování přepravy mezi místy a zobrazení vzdálenosti a času,
- nápovědu míst k navštívení,
- znázornění všech míst a tras na mapě.

Jednou z předností aplikace je, že dokáže sama navrhovat místa, kam by se uživatel mohl chtít podívat. Dále dokáže naplánovat přesnou trasu kudy jít. Chybí zde ale přesnější práce s časem, nelze jednoduše vidět v kolik musí uživatel z určitého místa vyjít aby stíhal vše, co má naplánované. Dále zde úplně chybí plánování financí nebo vytvoření více verzí. Celkově tuto funkci v aplikaci Mapy.cz můžeme hodnotit kladně pouze v případě, že si chceme naplánovat malý výlet po památkách v jednom městě, kde nám nejdelší čas zaberou právě přechody mezi těmito památkami. Důvodem je právě nedostatečný časový plán, jelikož vidíme pouze dobu strávenou přesunem mezi místy. Proto tuto aplikaci nepovažujeme za vhodnou pro plánování itineráře trochu rozsáhlejší cesty.

Na snímku obrazovky níže můžeme vidět uživatelské rozhraní aplikace. To samotné se jeví jako dostatečné k funkcím, které aplikace nabízí. Mezi nedostatky můžeme zařadit trochu nepřehledné grafické rozhraní s nedostatečným popisem prvků, které se zde objevují.[22]



Obrázek 2.2: Snímek uživatelského rozhraní aplikace Mapy.cz

2.4.2 Inspirock

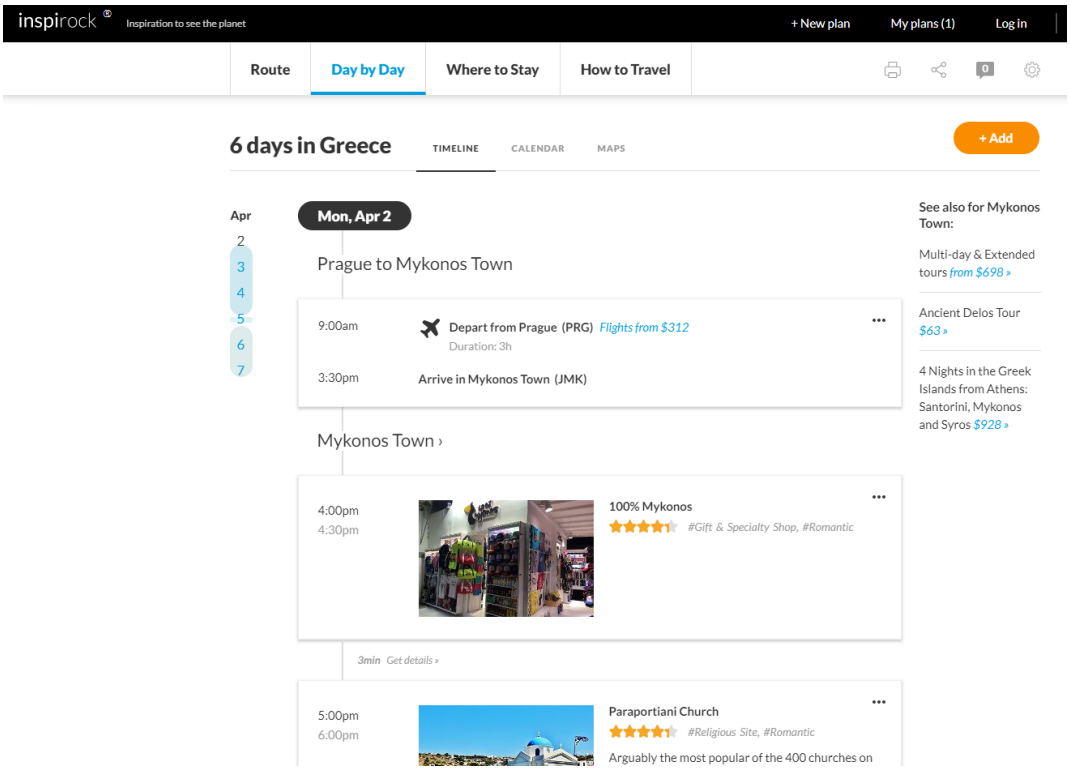
Tato aplikace se již více podobá plnohodnotnému plánování Itineráře. Je zde časová osa, na které jsou veškerá místa s přesnými časy příjezdu a odjezdu. Mezi nimi se nachází délka přesunu na nové místo. Pokud ale uživatel cestuje individuálně, z důvodu, že si chce dovolenou celou naplánovat podle sebe, tak jej tato aplikace nejspíš zklame. Při vytvoření itineráře se celý předvyplní navrhovanými letenkami, hotely a místy. Bohužel se nepodařilo zjistit, jak tuto funkci vypnout. Pokud si tedy chcete v aplikaci vytvořit svůj plán cesty úplně sami, budete nuceni všechny tyto místa ručně odstraňovat.

Je tedy vhodná spíše pro uživatele, kteří chtějí cestovat individuálně z jiného důvodu, než je možnost plánování vlastních míst k navštívení nebo pro ty, kteří hledají inspiraci, čemuž napovídá i název této aplikace. Dále zde chybí možnost vytvoření více verzí cesty, práce s financemi a měnami.

Celá aplikace je pravděpodobně vytvořená za účelem zisku pomocí affiliate marketingu, a tedy nabízení ubytování a letenek za určitou provizi. Na tomto určitě není nic špatného, jelikož aplikace z něčeho být financována musí. Ovšem dostává se pocit, že se autoři zaměřili hlavně na tuto část a dávají ji přednost před volností uživatelů při tvorbě itineráře. Dokonce se zde objevila i reklama formou rušivého vyskakovacího okna přes celou obrazovku.

Uživatelské rozhraní je čisté a moderní ale jako celek je poněkud nepřehledné a uživateli může chvíli trvat, než se zorientuje a zjistí co mu aplikace zrovna zobrazuje. Toto uživatelské

rozhraní můžete vidět níže na obrázku 2.3, na kterém lze vidět vytvořenou část ukázkové cesty. [17]

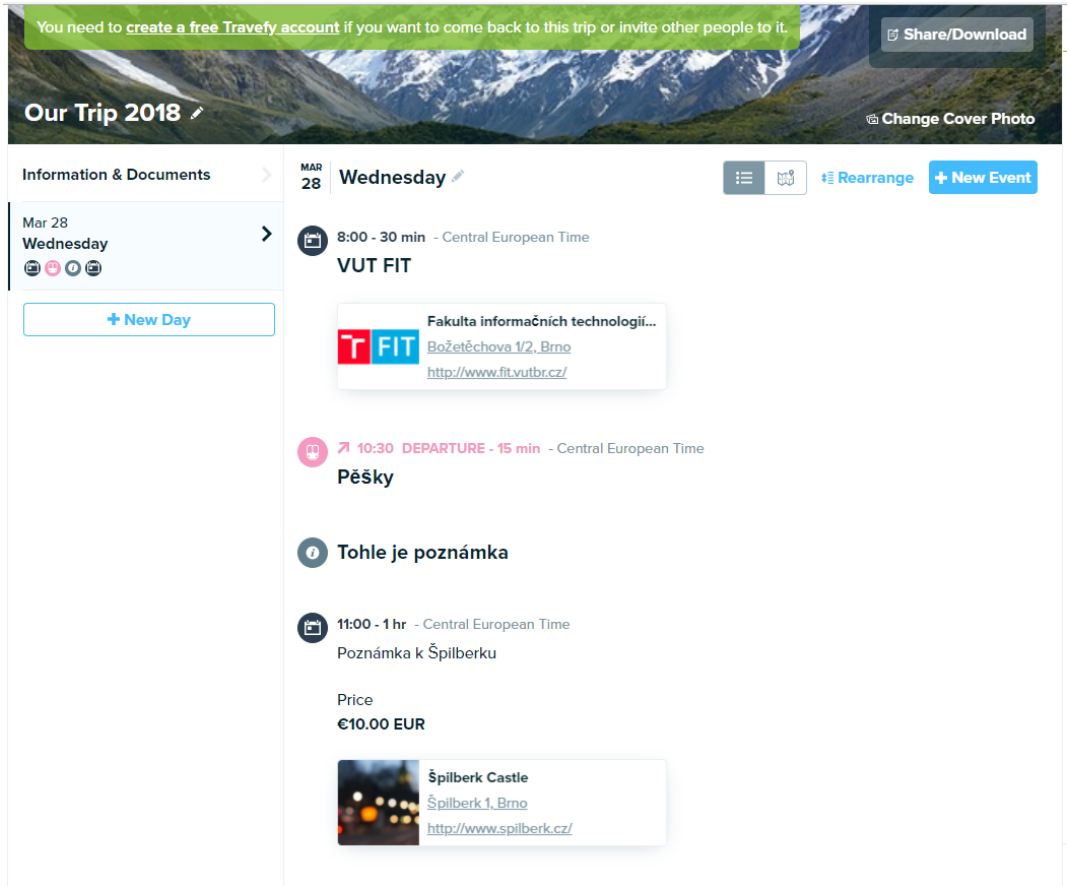


Obrázek 2.3: Snímek uživatelského rozhraní aplikace Inspirock

2.4.3 Travefy

Tato aplikace v analýze konkurence dopadla nejlépe. Poskytuje funkce pro přidávání míst a přepravy. Ke každému místu lze také zadat cena v různých měnách, bohužel pak již ale nelze zobrazit přepočet cen v jednotné měně. Aplikace je napojená na API Google Map a je možné tak k místům přiřadit kartu s odkazem na dané místo v Google Mapách. Dále lze místa i zobrazit všechny pohromadě na mapě. Toto napojení na Google Mapy hodnotíme kladně. Dále aplikace poskytuje funkci exportu itineráře do formátu iCalendar a je tedy možné jej takto vložit do svého elektronického kalendáře.

Graficky ale aplikace nepatří mezi nejpovedenější. Některé grafické prvky jsou nepřehledné a chvíli trvá, než se mezi nimi uživatel zorientuje. Chybí zde časová osa, která by zajistila větší přehlednost a orientaci v čase. Uživatelské rozhraní lze vidět na obrázku 2.4 níže. [25]



Obrázek 2.4: Snímek uživatelského rozhraní aplikace Travefy

2.5 Shrnutí konkurenčních aplikací a stanovení cílů

Mezi konkurenčními aplikacemi, které byly zahrnuty do analýzy, se objevily aplikace zahraniční ale i jedna z dílny české internetové jedničky. Všechny tyto aplikace mají podobný cíl, a to zjednodušit uživateli naplánovaní cesty, výletu nebo celé dovolené. Jak lze z analýzy vypozorovat, každá aplikace na to jde jiným způsobem a každá se zaměřuje na trochu jiné problémy a funkce. Zatímco jedna je zaměřená více na plánovaní přesunu mezi místy, druhá se zase snaží vytvořit uživateli již hotový itinerář za pomoci reklamních partnerů. Společným nedostatkem ale je příliš složité nebo nedostatečně přehledné uživatelské rozhraní aplikace.

Na základě této analýzy byly stanoveny cíle této práce. Cílem této práce bude vytvořit aplikaci, která by skloubila užitečné funkce těchto konkurenčních aplikací jako je plánování dopravy mezi místy, práci s financemi, časem a měnami. Dále by přidala lepší práci právě s měnami, možnost vytvoření více verzí a jejich porovnání jak z hlediska časového, tak z hlediska finančního. Také se zaměříme na možnosti zjednodušení a zpřehlednění celé práce s aplikací.

Kapitola 3

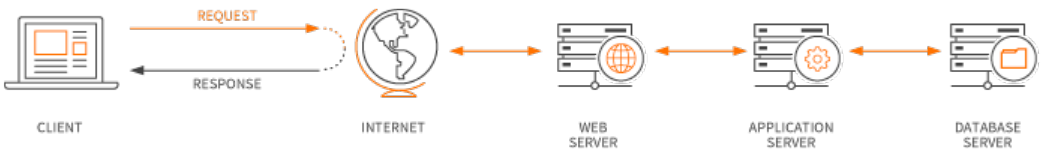
Použité technologie

Cílová skupina této aplikace je poměrně velká a jedná se o aplikaci, u které je pravděpodobné, že bude využívána i přímo na cestách, na cizím počítači například na hotelu. Proto je vhodné, aby byla aplikace dostupná na více platformách. Z těchto důvodů jsme si zvolili tvorbu webové aplikace. Tato volba vyřeší veškeré problémy s kompatibilitou na více platformách a aplikace tak bude dostupná online z jakéhokoliv zařízení, které podporuje webový prohlížeč.

V této kapitole si tak postupně popíšeme všechny technologie, které budeme při vývoji této aplikace využívat. Mezi ně patří například informace o fungování webových aplikací, použitý programovací jazyk a jiné převážně webové technologie.

3.1 Webová aplikace

Webová aplikace je počítačový program, který využívá webové prohlížeče a webové technologie pro provádění úkonů přes internet. Taková aplikace využívá kombinace provádění skriptů na straně serveru například pomocí jazyka PHP nebo ASP, které obstarávají komunikaci s databází a odpovídá na dotazy. Na druhé straně se nachází klient, který za pomoci HTML a Javascriptu zobrazuje požadované informace v okně prohlížeče uživateli.



Obrázek 3.1: Schéma fungování webové aplikace [20]

Webová aplikace může být statická nebo dynamická. V případě statické aplikace není potřeba zpracovávání na straně serveru. Naopak dynamická aplikace na straně serveru při každém dotazu zpracovává informace, případně komunikuje s databází, kde ukládá informace a podle výsledku tohoto zpracování poté odesílá odpověď klientovi, který ji zobrazí uživateli.[20]

Fungování typické webové aplikace:

- Uživatel odešle požadavek na webový server skrz internet pomocí webového prohlížeče nebo uživatelského rozhraní aplikace.
- Webový server předá tento požadavek aplikačnímu serveru.
- Aplikační server provede přijatý dotaz, což znamená získání dat z databáze a jejich zpracování a poté vygenerování výsledku.
- Aplikační server zašle výsledek webovému serveru.
- Webový server poté odešle výsledek klientovi, kterému se tyto výsledné informace zobrazí na displeji.

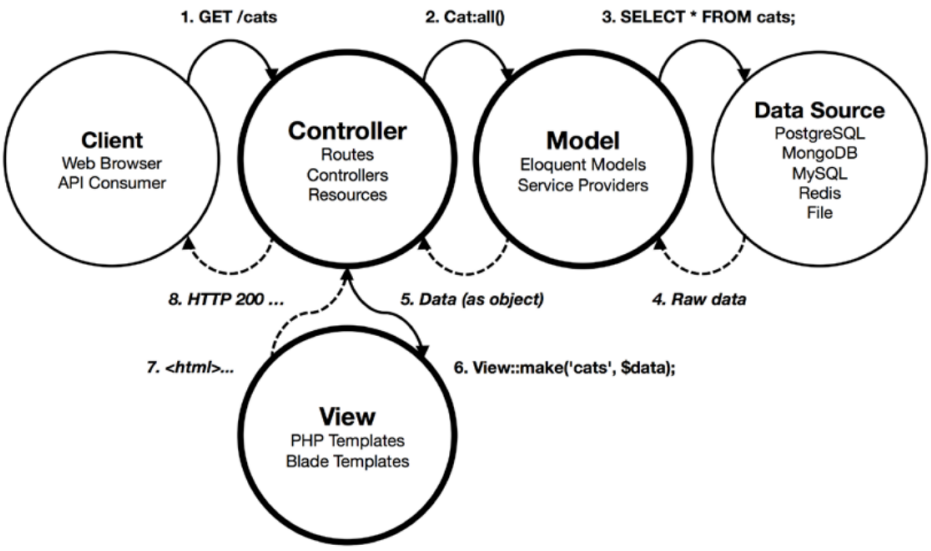
Mezi výhody webové aplikace patří:

- Aplikace běží na všech platformách, které disponují webovým prohlížečem.
- Všichni uživatelé přistupují vždy ke stejné verzi aplikace.
- Uživatel nemusí aplikaci instalovat a ta tak nezabírá místo na jeho pevném disku.

3.2 Model View Controller architektura

Model View Controller neboli MVC je softwarová architektura převážně využívána pro implementaci aplikací s uživatelským rozhraním. Snaží se obecně oddělit logiku od výstupu za pomoci rozdělení aplikace na tři na sobě nezávislé části. Jak již název této architektury napovídá jedná se o Model, View a Controller.[30]

- **Model** – jedná se o část, která se stará o veškerou logiku aplikace. Mohou se zde provádět výpočty, dotazy na databázi a další zpracování dat. Model neví odkud přišel dotaz ani jak budou zpracovaná data zobrazena.
- **View** – neboli pohled má za úkol zobrazit data, která mu byla předána, uživateli. Jedná se tak nejčastěji o šablonu obsahující HTML stránku doplněnou například jazykem PHP, který zde umožňuje vkládat proměnné, provádět iteraci a podobně.
- **Controller** – je prostředník mezi uživatelem, modelem a view. Drží tak celý systém pohromadě a propojuje jednotlivé komponenty mezi sebou.



Obrázek 3.2: Schéma architektury Model View Controller [6]

3.3 PHP

PHP je široce používaný skriptovací jazyk, který je distribuován pod licencí open source. Jedná se o imperativní, objektově orientovaný jazyk, který je nejčastěji využíván právě pro skriptování na straně webového serveru. PHP je rekurzivní zkratkou z anglického názvu *PHP: Hypertext Preprocesor*. Jeho flexibilita a poměrně krátká doba pro jeho osvojení z něj dělá jeden z nejoblíbenějších skriptovacích jazyků současnosti.[19] Síla tohoto jazyka spočívá také v jeho vestavěných funkcích, které jsou kvalitně zdokumentované na webu PHP. [14]

Jazyk PHP byl vydán v roce 1995 a od té doby až do dnešního dne dosáhl verze 7.2.0. Autorem první verze tohoto jazyka je dánsko-kanadský programátor Rasmus Lerdorf.[14] I když jeho obliba v posledních letech lehce klesá, stále se jedná o jazyk, který využívá drtivá většina webů a webových aplikací. V květnu roku 2017 byl tento podíl přes 82%.[21] PHP využívají i velké webové aplikace jako je Wikipedie nebo Facebook.

V praxi se pro webové aplikace často používají PHP frameworky jako je například CodeIgnitor, Symfony nebo také v tuto chvíli nejpoužívanější Laravel framework, který jsme si zvolili i pro tuto aplikaci a více se o něm píše v kapitole 4. Frameworky ulehčují celou práci s tímto jazykem pro účely vytváření webových aplikací a často jsou založeny na architektuře Model View Controller, která je popsána v sekci 3.2.

3.4 MySQL

MySQL je systém řízení báze dat distribuovaný pod licencí open source. Byl vytvořen švédskou firmou MySQL AB a později byl odkoupen firmou Oracle Corporation, která jej vlastní do dnes. Jedná se o databázový server, který umožňuje ukládání a správu velkého množství informací. MySQL server zpracovává dotazy pomocí dotazovacího jazyka SQL(Structured Query Language). Tento jazyk je navržený pro zpracování velkého množství složitých dotazů. Jelikož se jedná o relační databázový systém, umožňuje velmi vysokou rychlost a

efektivitu, díky možnosti spojování několika tabulek.[19]

Hlavní výhody a funkce MySQL serveru:

- Využití více procesorů prostřednictvím vláken jádra.
- Možnost běhu nezávisle na platformě.
- Dostatek datových typů.
- Dostupnost funkcí pro matematické výpočty.
- Možnost definice až 32 indexů pro každou tabulku.

Příklad jednoduchého dotazu v jazyce SQL:

```
SELECT UserName FROM User WHERE UserEmail='user@email.com'
```

3.5 HTML

HyperText Markup Language neboli HTML je značkovací jazyk využíván pro tvorbu webových stránek. Vznikl v roce 1990 spolu s protokolem HTTP, který slouží pro přenos HTML dokumentů na internetu. Spolu s vývojem jazyka HTML probíhá také vývoj webových prohlížečů, který uživateli HTML soubory zobrazuje. Proto je nezbytné, aby nové funkce jazyka HTML byly podporovány i ve webových prohlížečích. Poslední verzí jazyka HTML je verze 5.2 a již teď je ve vývoji verze 5.3. [27]

Jazyk HTML se skládá z množiny značek a jejich atributů. Text stránky se uzavírá mezi dvojice značek a tím se definují vlastnosti textu uvnitř. Každý dokument v jazyku HTML by měl obsahovat:

- Deklaraci typu dokumentu
- Kořenový element *<html>*
- Hlavičku dokumentu *<head>*
- Tělo dokumentu *<body>*

3.6 CSS

CSS neboli *Cascading Style Sheets* vznikl v roce 1997 za účelem úprav vzhledu webové stránky. CSS je jazyk pro popis vzhledu dokumentu napsaného v jazyce HTML nebo XML. CSS udává, jak má být celá stránka zobrazena uživateli. CSS tak odděluje vzhled dokumentu od jeho struktury napsané v jazyce HTML. Poslední verzi je CSS 3, které svými funkcemi navazuje a doplňuje HTML verze 5.

CSS využívá selektorů pro výběr elementu, kterému chceme definovat pravidla. Výběr elementu může být například podle HTML značky, třídy nebo identifikátoru elementu. Za pomoci těchto selektorů se definuje vzhled jednotlivých elementů. Díky CSS tak můžeme definovat vzhled více elementů se stejnou HTML značkou nebo třídou.

Funkčnost nástrojů v CSS je vždy závislá na webových prohlížečích. Ne každá funkce, převážně v nejnovějších verzích CSS, bude funkční ve všech prohlížečích. Podporu jednotlivých funkcí CSS v nejpoužívanějších prohlížečích si můžeme snadno ověřit na některých webových stránkách.[10]

3.7 JavaScript

Javascript je objektově orientovaný programovací jazyk, který byl vytvořen v roce 1995. V dnešní době se nejčastěji využívá právě na webových stránkách, kde je interpretován u uživatele ve webovém prohlížeči. Jeho syntaxe vychází z jazyků C++ a Java. Celkově ale nemá s těmito jazyky až tolik společného a slovo java v jeho názvu na jazyk Java nijak neodkazuje, jedná se pouze o marketingový tah.

Tento programovací jazyk, se na rozdíl například od jazyka PHP spouští na straně klienta po stažení webové stránky. Z toho plynou bezpečnostní rizika, jelikož je Javascript ve všech prohlížečích, je zde možnost zneužití a spouštění škodlivého kódu. V posledních měsících se například objevují weby, které na pozadí pomocí Javascriptu těží kryptoměny a tím zatěžují procesor uživatele.

Javascript přidává webovým stránkám nové možnosti přechodů, různých animací a vylepšené interakce s uživatelem. Zajišťuje asynchronní zpracování webových stránek a může být využit i pro zvýšení náročnosti automatizovaného vykrádání webu. [9]

3.7.1 JQuery

JQuery je Javascriptová knihovna, která je na svém webu prezentována jako malá, rychlá a bohatá na funkce. První stabilní verze této knihovny se objevila v roce 2006. Jednalo se o rozšíření pluginu pro zjednodušení práce se selektory, který vytvořil v roce 2005 John Resig. Poslední verzí této knihovny je verze 3.3.1 představena na začátku roku 2018.

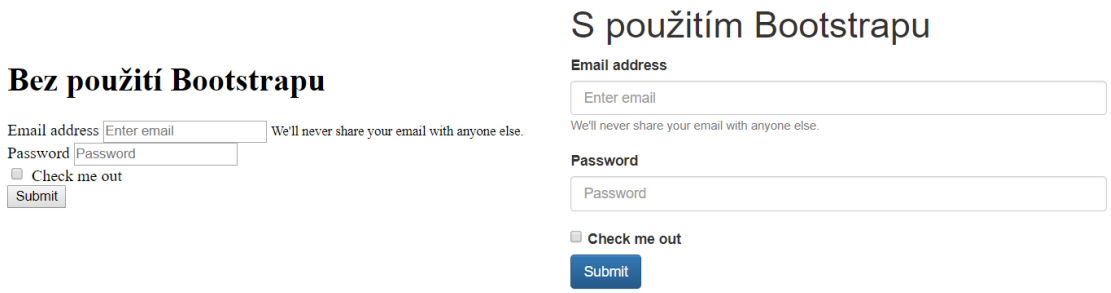
Jedná se o velmi oblíbenou a často využívanou knihovnu, která celou práci s ne úplně jednoduchým Javascriptem velmi usnadňuje. Usnadňuje práci s HTML dokumentem, zpracováním vyvolaných událostí, animacemi, ajaxem a mnoha dalšími funkcemi. [24]

3.8 Bootstrap

Bootstrap je frond-end framework, který rozšiřuje možnosti CSS a Javascriptu. Zajišťuje snadnou implementaci responzivity aplikace na různých zařízeních. Za jeho vznikem stojí Mark Otto a Jacob Thornton, kteří měli za úkol ve společnosti Twitter vytvořit interní nástroj pro vývoj frontendu. Později se tento nástroj začal rozrůstat a v roce 2011 byla vydána veřejná open-source verze pod názvem Bootstrap.

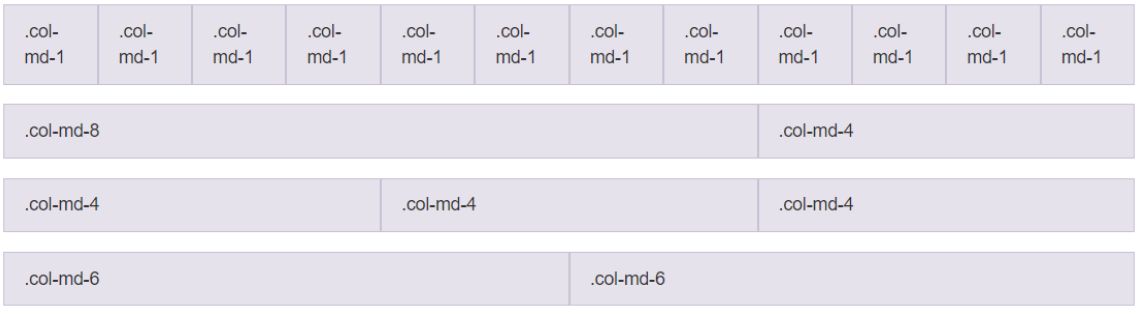
Tento nástroj usnadňuje vývoj frontendu založeného na HTML a CSS. Bootstrap nabízí hotové šablony pro formuláře, tlačítka, tabulky a spoustu jiných komponentů. Na verzovacím systému GitHub získal 194 tisíc kladných hodnocení, což jej dostalo na druhou příčku.

Pro používání Bootstrapu stačí zahrnout do HTML soubor CSS a Javascript, který lze stáhnout na webu getbootstrap.com. Na stejném webu se nachází i dokumentace, ve které lze najít všechny dostupné komponenty. Většina komponent se používá za pomoci předdefinovaných tříd. Níže je na obrázku 3.3 příklad rozdílu mezi jednoduchým formulářem bez a s použitím Bootstrapu. [1]



Obrázek 3.3: Porovnání formuláře v HTML bez a s použitím knihovny Bootstrap

Další obrovskou funkcí knihovny Bootstrap je mřížkový systém, v angličtině *grid system*. Tento nástroj umožňuje snadné definování rozložení stránky, u kterého je zajištěna responzivita na různé šířce displeje. Bootstrap rozděluje stránku na 12 sloupců, což je maximální šířka stránky. Poté se vytvářejí řádky široké jeden nebo více sloupců a do nich se zasazují komponenty. Pro tuto funkci jsou zde předdefinované třídy, které zajištují toto rozložení. Příklad rozložení do mřížky díky příslušným třídám je na obrázku 3.4. [1]



Obrázek 3.4: Příklad rozložení pomocí mřížkového systému knihovny Bootstrap [1]

3.9 Git

Je distribuovaný systém pro správu verzí. Byl vyvinut Linusem Torvaldsem, který tento nástroj vytvořil při vývoji jádra Linuxu. Tento systém sleduje a zaznamenává změny v souborech a zaručuje jejich zálohu a možnost obnovení zpět na vybranou verzi. [3]

Pro správu této práce byla použita platforma GitHub, která zajišťuje správu vývoje právě pomocí nástroje Git. GitHub je velmi rozšířený a spojuje komunitu programátorů, kteří zde mohou zveřejnovat své zdrojové kódy. Proto je vhodný pro publikaci open-source projektů. GitHub nabízí Studentský vývojářský balík, který studentů umožňuje zdarma vytvářet soukromé repositáře pro školní projekty. [11]

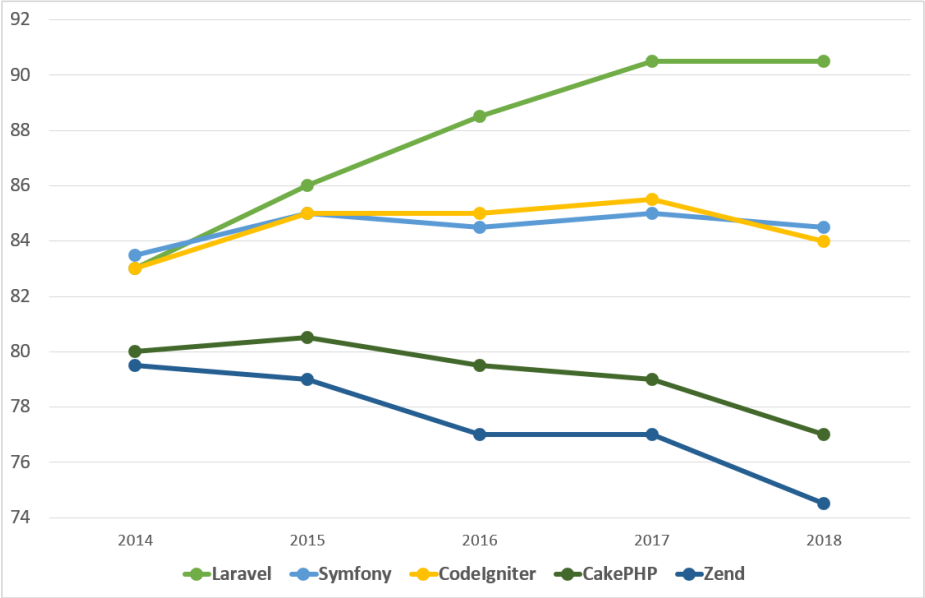
Kapitola 4

Laravel framework

Pokud budeme uvažovat o programovacích jazycích používaných na straně serveru, PHP je naprosto nejvyužívanějším a ve většině případů je dostupný i na tom nejlevnějším hostingu. Každý, kdo vytvořil několik webových aplikací pomocí čistého PHP zjistil, že spoustu funkci a tříd, které si při jedné webové aplikaci vytvořil využije i při další. Přesně z tohoto důvodu se využívají nejrůznější PHP frameworky, které programátorovi dodají balík funkcí, které se dají využít na většinu aplikací. Jedná se o správu databáze, frontendu, autentizace uživatelů, směrování url adres a podobně. Jedním z takových frameworků, které umožnují více strukturovaný a pragmatický vývoj webových aplikací, je i Laravel. [6]

Laravel je PHP framework distribuovaný jako open-source. Byl vytvořen Taylorem Otwellem a jeho první verze vyšla v roce 2011. Laravel byl vytvořen za účelem sjednotit všechny užitečné funkce z jiných frameworků, a to nejen pro PHP. Mezi frameworky, ze kterých Laravel přebral část funkcí patří CodeIgniter, Yii, ASP.NET MVC, Ruby nebo Symfony.

Většina těchto frameworků využívá architekturu Model View Controller, který je blíže popsaný v sekci 3.2, a Laravel není výjimkou. Laravel se na rozdíl od ostatních frameworků snaží držet dobu s PHP a využívá tak spoustu nových funkcí a možností, které PHP nabízí. [6] I z těchto důvodů získal Laravel na webu hotframeworks.com [2] hodnocení 90 bodů, což ho řadí na první příčku mezi všemi PHP frameworky. Toto hodnocení frameworků vychází z oblíbenosti na GitHubu a podle počtu otázek položených na webu Stack Overflow, což dohromady dává objektivní představu o oblíbenosti a používání jednotlivých frameworků. [2]



Obrázek 4.1: Graf oblíbenosti pěti nejlépe hodnocených PHP frameworků podle bodového hodnocení webu hotframeworks.com [2]

Laravel obsahuje velké množství různých modulů a funkcí, jelikož byl postaven za pomoci více než 20 různých knihoven. Zaměřuje se také na testování nebo směrování. Nabízí pokročilou konfiguraci a není tak problém nastavit například e-mailový server rozdílně pro různá prostředí, na kterých aplikace běží. Dále obsahuje dotazování na databázi pomocí PHP syntaxe, migraci, práci s emaily nebo šablonami a mnoho dalšího. Níže jsou popsány vybrané části laravelu, které jsou v této práci dále použity. [6]

4.1 Směrování

Laravel poskytuje velmi velkou flexibilitu při vytváření cest v URL adrese aplikace. Většina cest je definována v souboru *routes/web.php*. V tomto souboru můžete jednoduše vytvářet cesty a pro tyto cesty dále provádět funkce podle toho, jaký typ HTTP dotazu byl volán. Základní vytvoření cesty může vypadat například takto:

```
Route::get('/user', 'UserController@index');
```

Tímto se nám vytvoří směrování na url *"www.naseaplikace.cz/user"* a po odeslaní požadavku GET na tuto URL server vrátí výsledek funkce index v kontroléru UserController. V případě, že vytvoříme směrování pro požadavek typu POST, PUT nebo DELETE spolu s daty z formuláře se na tuto URL musí zaslat i CSRF token, který byl vygenerován při načtení formuláře. Tento token slouží jako ochrana proti stejnojmennému útoku. V směrovacích cestách můžeme předávat i parametry, které poté zpracujeme a na základě jich vrátíme uživateli požadované informace.

```
Route::get('user/{id}', function (\$id) {
return 'User '.\$id;
});
```

V tomto případě je definována například cesta *"www.naseaplikace.cz/user/42"*, kde číslo 42 je předávaný parametr ID, který značí identifikátor uživatele, kterého má aplikace zobrazit. Pokud se v definici cesty přidá za parametr znak "?", bude to parametr volitelný a ve funkcí, která obsluhuje dotaz na tuto cestu musí být definována výchozí hodnota této proměnné. Přijímané hodnoty je možné specifikovat regulárním výrazem pomocí metody where, volané nad instancí objektu *Route*. Definované cesty můžeme dále pojmenovávat prostřednictvím metody *name*. Díky tomu můžeme pomocí jména cesty generovat URL, nebo na ně přímo přesměrovávat. [4]

4.2 Blade šablony

Blade je jednoduchý ale výkonný nástroj pro tvorbu šablon dostupný právě v Laravelu. Na rozdíl od ostatních nástrojů pro tvorbu šablon v PHP, Blade neomezuje od používání čistého PHP kódu v šabloně. Naopak, Blade šablona se překládá do čistého PHP kódu a ukládá do mezipaměti dokud není šablona upravena. Tím Blade šablona nezatěžuje žádným způsobem výslednou aplikaci. Soubory s touto šablonou mají koncovku *".blade.php"* a typicky jsou uloženy ve složce *resources/views*.

Hlavní výhody používání Blade šablony jsou možnost dědictví z jiné šablony a rozdělování částí šablony do sekcí. Sekce se vytvoří pomocí příkazu *@section("název sekce")*. Daná sekce se může vložit a zobrazit na určitém místě pomocí příkazu *@yeld("název sekce")*. Pokud se vytváří soubor šablony, který rozšiřuje jiný soubor šablony, definuje se tato dědičnost pomocí příkazu *@extends("layouts.název rodičovského souboru)*. Díky tomu je možno rozdělit často se opakující části šablon do jednotlivých sekcí a sdílet je do více částí aplikace. Obsah PHP proměnné vložíme do HTML kódu pomocí dvojice dvojitých závorek *{{$var}}*. Je možnost také vytváření podmínek typu *if*. Ty se vytváří za pomoci příkazů *@if("podmínka")*, *@else*, *@elseif* a *@endif*.

Mezi další příkazy patří například *@isset("proměnná")*, *@empty("proměnná")* nebo příkazy pro iteraci jako je *@for(...)*, *@while(...)* a *@foreach(...)*. Díky tomu je možné podmiňovat a iterovat HTML kód na vstup. Pokud je potřeba přepnout se v této šabloně do čistého PHP, lze to pomocí *<?php*. Tuto sekvenci čistého PHP kódu lze ukončit znaky *?>*. [4]

4.3 Lokalizace

Laravel nabízí i funkce pro lokalizaci webové aplikace. Tento nástroj zajišťuje možnost mít aplikaci ve více jazycích. Nástroj lokalizace v Laravelu zajišťuje jednoduchý způsob jakým získávat textové řetězce v různých jazycích. Jazykové řetězce jsou všechny uloženy v adresáři *resources/lang*, který obsahuje podadresáře pro každý jazyk. Všechny lokalizační soubory navrací pole klíčů a k nim přiřazené jazykové řetězce. Například:

```
<?php

// resources/lang/en/messages.php

return [
'welcome' => 'Welcome to our application'
];
```

Pokud je tvořena velká aplikace, složitá na jazykový překlad, definování každého textového řetězce pomocí klíče může být dost nepřehledné a matoucí. Proto Laravel nabízí další

možnost jazykových souborů a tou je soubor ve formátu JSON. V tomto souboru jsou jako klíče použity celé originální textové řetězce.

```
{
"I love programming.": "Me encanta programar."
}
```

Přeložený textový řetězec je možné získat více způsoby. Jedním takovým je použití pomocné funkce ___. Tato funkce přijímá jméno souboru a klíč překládaného řetězce jako první argument. Vypsání přeloženého řetězce pak může vypadat takto:

```
echo __('messages.welcome');
```

Nebo v případě používání Blade šablony můžeme využít dvě dvojité závorky, případně volání funkce *@lang* takto:

```
{{__('messages.welcome') }}
@lang('messages.welcome')
```

[4]

4.4 Autentizace

Autentizace u webové aplikace znamená hlavně možnost registrace a přihlašování uživatelů a tím tak zabezpečení jejich osobních dat, které si ve webové aplikaci uloží. Dosud samotné PHP neposkytuje žádné konvence, jak by tento problém měl být řešen a nenabízí k tomu ani pomocné funkce. Proto si každý programátor implementoval autentizaci do své aplikace sám a někdy se tak mohlo stát, že bezpečnost aplikace nebyla na dostatečně vysoké úrovni. Proto Laravel připravil vlastní řešení pro autentizaci uživatelů, které je jednoduché pro integraci do jakékoliv aplikace a zároveň poskytuje dostatečnou bezpečnost pro uživatele.

Konfigurační soubor autentizace se v laravelu nachází v *config/auth.php* a obsahuje nastavení několika možnosti pro tuto funkci. V základním nastavení Laravel nabízí v balíku autentizace přihlašování uživatelů do aplikace, jejich registraci, pamatování přihlášení pomocí *remember_token* a samozřejmostí je také obnova zapomenutého hesla.

Pro všechny tyto funkce je připraven model *App\User*, který zabezpečuje přihlašovací a jiné údaje o uživateli a zajišťuje jejich ukládání do databáze a čtení z databáze. Dále se zde nachází několik kontrolérů v adresáři *App\Http\Controllers\Auth*. Mezi ně patří *RegisterController*, *LoginController*, *ForgotPasswordController* a *ResetPasswordController*. Všechny tyto kontroléry poskytují základní funkce, které jsou ve většině aplikací potřebné a ve většině případů nejsou potřeba již nijak modifikovat.

Laravel nabízí i automatické vytvoření směrování pro autentizaci a to za pomocí příkazu:

```
php artisan make:auth
```

Tento příkaz by měl být používán pouze na začátku vývoje aplikace a nainstaluje do aplikace přihlašovací a registrační stránky. K datům přihlášeného uživatele se lze dostat pomocí třídy *Auth*.

```
use Illuminate\Support\Facades\Auth;

// Get the currently authenticated user...
$user = Auth::user();
```

```
// Get the currently authenticated user's ID...
$id = Auth::id();
```

K zajištění přístupu na určitou stránku pouze pro přihlášené uživatele lze pomocí metody *middleware* nad instancí třídy *Route*.

```
Route::get('profile', function () {
// Only authenticated users may enter...
})->middleware('auth');
```

[4]

4.5 Plánování úkolů

Funkce *Plánování úkolů* (v angličtině *Task scheduling*) umožňuje pravidelné provádění určité funkce. Tato funkcionalita se provádí pomocí nástroje Cron na serveru a pro každý úkon je třeba nastavovat Cron zvlášť a nejsou tak všechny úkoly pohromadě. Tento problém řeší Laravel a nabízí možnost jednoduchého vytváření úkolů, které se mají provádět pravidelně v zadané periodě a jsou dostupné přímo ze zdrojového kódu Laravelu.

Celé Plánování úkolů se v Laravelu spustí pomocí jednoho úkolu v Cronu, který se spouští každou minutu a obstarává spouštění všech úkolů definovaných přímo ve zdrojovém kódu. Úkoly, které se mají provádět se nastavují v souboru *app/Console/Kernel.php* v metodě *schedule*. Poté stačí nastavit v Cronu pouze jeden úkol, který obstará všechny úkoly v této metodě. Vstup pro Cron je:

```
* * * * * php /path-to-your-project/artisan schedule:run >> /dev/null 2>&1
```

Tento úkon v Cronu spustí každou minutu metodu *schedule:run*, která již vyhodnotí úkoly, které se mají provádět. V Laravelu se pak úkoly nastavují pomocí PHP syntaxe podle příkladu níže.

```
$schedule->call(function () {
DB::table('recent_users')->delete();
})->daily();
```

Délka periody lze nastavit buďto pomocí syntaxe Cronu *->cron('* * * * *')* nebo syntaxí PHP například následovně *->everyFiveMinutes()*, *->hourly()*, *->dailyAt('13:00')* nebo *->weekly()*. [4]

4.6 Databáze

Práce s databází je v Laravelu velmi zjednodušena a dokáže pracovat s několika různými druhy databáze. Momentálně Laravel podporuje databáze:

- MySQL
- PostgreSQL
- SQLite
- SQL Server

Databáze lze konfigurovat v souboru *config/database.php*.V tomto souboru se nastavuje veškeré připojení k databázím a jedno připojení by mělo být nastaveno jako výchozí. Pro připojení k databázi je potřeba nastavit její název a přihlašovací údaje a také vybrat jeden z podporovaných druhů databáze.

Laravel poskytuje nástroj pro stavbu dotazů na databázi. Tento nástroj umožňuje jednoduché volání SQL dotazů za pomoci PHP syntaxe. Veškeré dotazy se vytvářejí prostřednictvím instance třídy DB, která drží spojení s databází. Tato třída obsahuje metody zajišťující dotazy na databázi. Mezi tyto metody patří například *table*, *where*, *get* nebo *update*. Pokud pro potřebný dotaz nestačí dostupné metody, je možné pomocí metody *raw* zapsat dotaz přímo v jazyce SQL. Níže jsou příklady dotazů, které je možno provádět.

```
$users = DB::table('users')->get();
$user_john = DB::table('users')->where('name', 'John')->first();
$max_price = DB::table('orders')->max('price');
DB::table('users')->delete();
```

[4]

Kapitola 5

Návrh aplikace

Nyní když máme nastudované veškeré technologie a problémy, které budeme v této bakalářské práci využívat a řešit, můžeme se posunout k samotnému návrhu aplikace. Samotný návrh si rozložíme do několika částí, které si podrobněji níže popíšeme. Součástí návrhu bude rekapitulace zadání, vytvoření ER diagramu, relační databáze, návrh uživatelského prostředí, analýza možností využití Google API a rozdělení implementace do několika dílčích částí.

Návrh aplikace se velmi často podceňuje ale ve skutečnosti je opravdu velmi důležitý, jelikož se na něm později staví celá aplikace. Pokud se tedy udělá chyba v návrhu, která se objeví až po implementaci, je možné že nepůjde odstranit vůbec nebo jen velmi obtížně.

Z těchto důvodů jsme se rozhodli návrh nepodcenit a vytvořit si nejprve jednoduchý diagram případů užití a poté ER diagram. Tyto dva diagramy nám dají přesnou představu o tom jaké informace budeme potřebovat uchovávat v databázi a jaké funkce by měla aplikace umět. Na základě těchto výsledků se může dále vytvořit prvotní návrh uživatelského prostředí, které by mělo byt navrženo tak, aby v něm byly dostupné veškeré možnosti vyplývající z diagramu případu užití.

5.1 Rekapitulace zadání práce

V této části se pokusíme shrnout zadání této práce a veškeré informace a nápady na funkcionalitu, které jsme doposud získali. Díky tomu získáme představu o celé aplikaci a budeme moci z těchto informací vycházet při dalším návrhu.

Ze zadání této práce vyplývá, že aplikace má zvládat plánování itineráře. Tedy vkládat lokace a dopravu na tyto lokace spolu s časovými a finančními informacemi. Aplikace by měla umět vytvářet více verzí itineráře a jejich vzájemné porovnávání. Dále jsme se rozhodli implementovat přepočet veškerých měn zadaných do aplikace podle aktuálního kurzu a zpřístupnit tak možnost uživateli vidět vždy aktuální přepočet cen na jeho domovskou měnu, ve které se lépe orientuje. Součástí aplikace by mohlo být i napojení na API Google Map, které by mohlo sloužit pro jednodušší vyhledávání lokací a získávání dalších informací o místech, které jsou v Google Mapách dostupné. Jelikož je vhodné aby měl uživatel aplikace všechny své naplánované itineráře uložené a dostupné, implementujeme i správu uživatelů spolu s registrací nebo přihlašováním a zohledníme i možné rozšíření do budoucna a to sdílení itineráře mezi uživateli. Proto navrhneme i práva uživatelů pro jednotlivé itineráře.

Součástí zadání pro tuto aplikaci je i přehledné uživatelské rozhraní. Rozhodli jsme se omezit maximální počet verzí itineráře na tři. Vyhodnotili jsme toto rozhodnutí jako

vhodné, jelikož počet uživatelů, kteří by verzí itineráře využili více je minimum a naopak toto omezení přinese větší přehlednost pro uživatele.

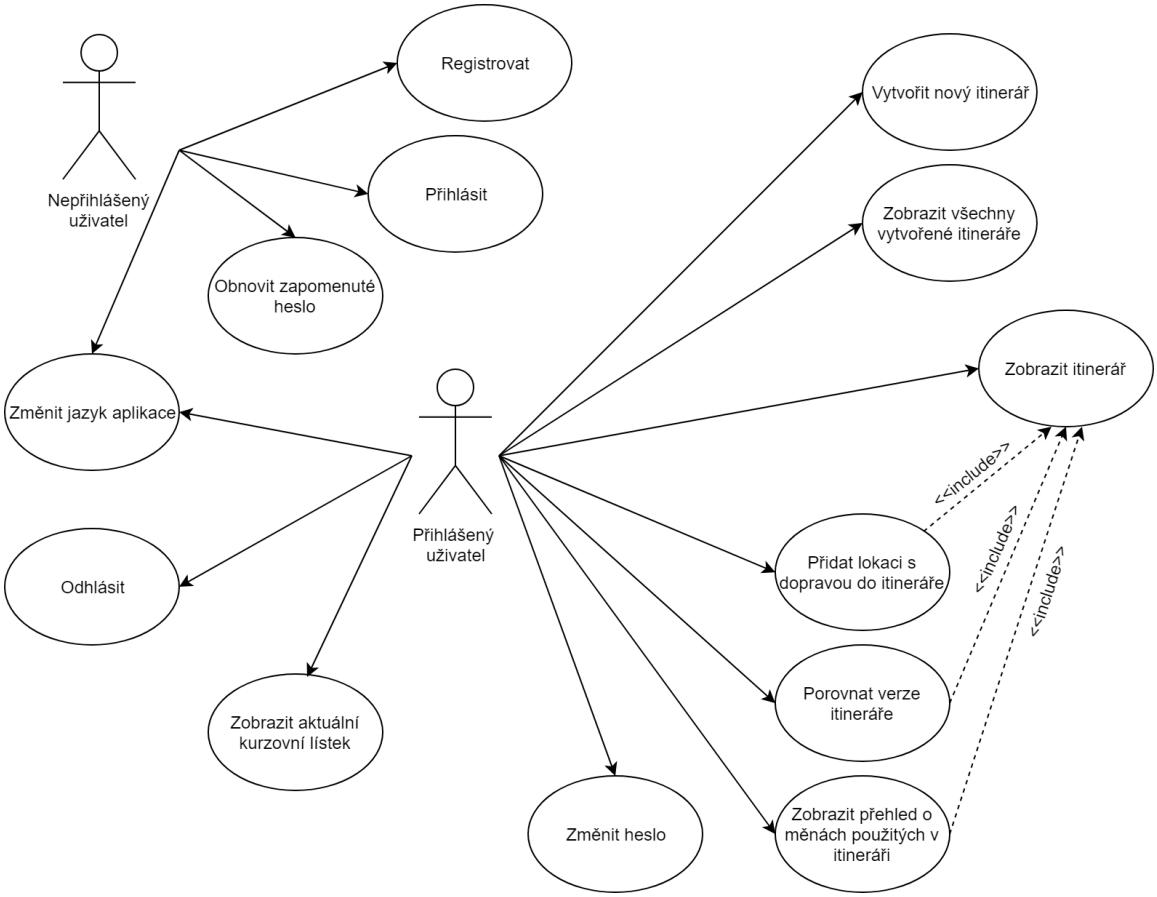
Další funkcí bude zobrazení finančního a časového souhrnu o cestě, ve kterém aplikace sama vyhodnotí nejvýhodnější verzi itineráře nebo upozorní při přečerpání rozpočtu, který jsme si pro tuto cestu stanovili. Součástí přehledu bude i přehled o měnách, který nám zobrazí výpis jednotlivých měn, které jsou v itineráři použity spolu s jejich množstvím. Tím uživatel získá představu o potřebném množství jednotlivých měn na tuto cestu a možnost směny finančních prostředků ve směnárně tak aby mu finance v dané měně nepochyběly a naopak zbytečně nepřebývaly.

Pro budoucí uplatnění aplikace na trhu, jsme se rozhodli implementovat i vícejazyčnost, tak aby byla aplikace připravena pro překlad do jiných jazyků. Výchozím jazykem, v jakém budeme aplikaci psát bude angličtina a v rámci této práce provedeme překlad aplikace do českého jazyka.

5.2 Diagram případů užití

Diagram případů užití, neboli *Use Case Diagram* zobrazuje chování systému z pohledu uživatele. Je vhodný pro návrh informačních systémů, ve kterých se objevuje řada typů uživatelů s různými možnostmi v rámci systému. Díky tomu získáme abstraktní pohled na uživatele a jejich interakce s aplikací. Dokážeme si tak lépe představit a poté navrhnout celou aplikaci tak, aby dokázala obsluhovat všechny požadavky, které by uživatel mohl v různých částech aplikace provádět.

V naší aplikaci se budou momentálně nacházet jen dva typy uživatelů, ale i přesto jsme se rozhodli pro vytvoření diagramu případů užití. Ten není nijak velký a obsáhlý, ovšem dokáže nám shrnout veškeré funkce, které bychom uživateli chtěli poskytnout. Výsledný diagram pak využijeme při dalším návrhu této aplikace.



Obrázek 5.1: Diagram případů užití

Diagram případů užití z obrázku 5.1 není nijak složitý, ale ukazuje nám, jaké funkce a možnosti má mít uživatel v této aplikaci k dispozici. V diagramu je uživatel vyobrazen jako přihlášený a nepřihlášený. Oba tito uživatelé mají možnost změny jazyka celé aplikace.

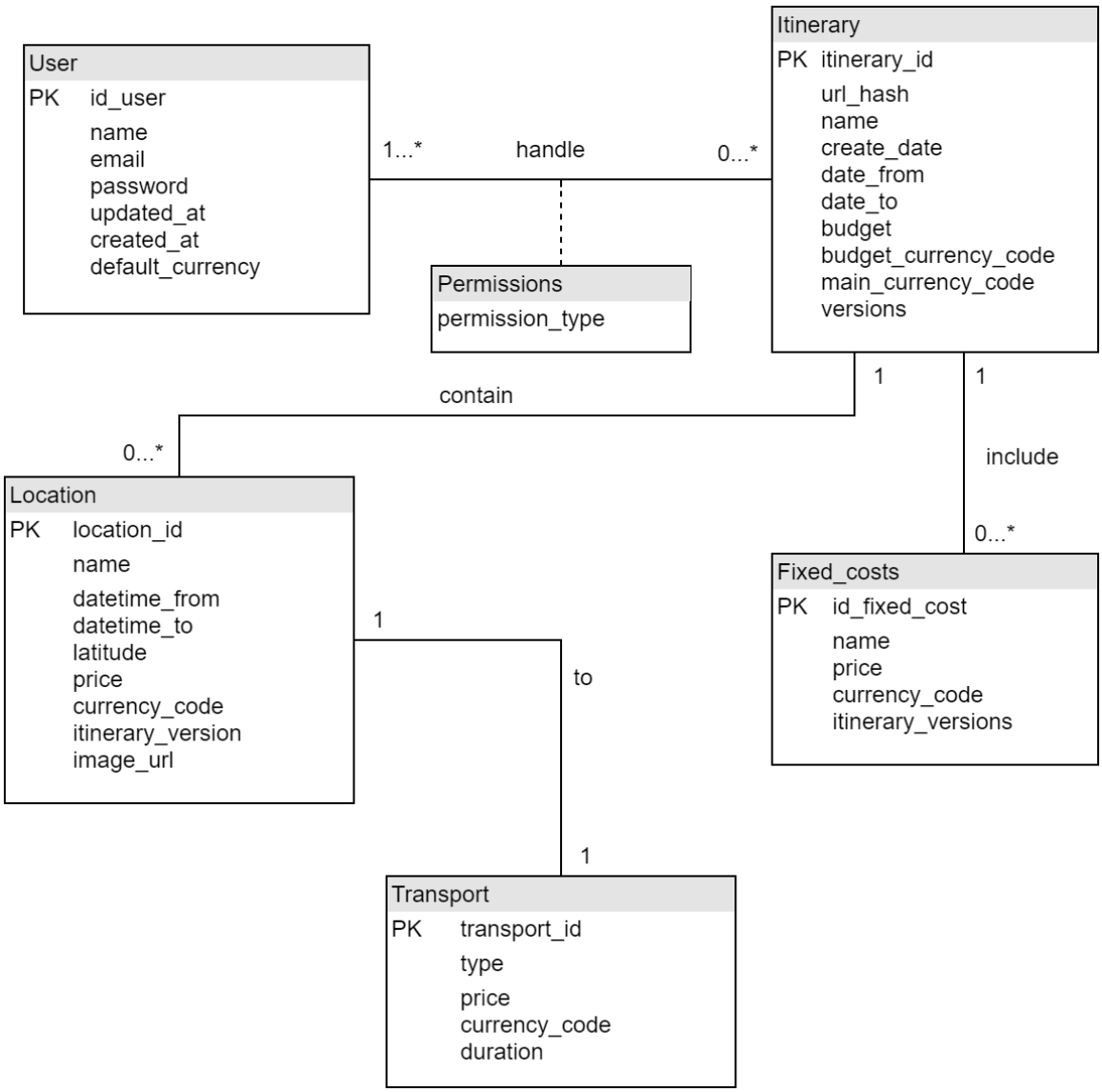
Nepřihlášený uživatel má k dispozici případ užití přihlášení v případě, že již má vytvořen uživatelský účet. Pokud uživatelský účet zatím nemá, nebo chce nový, má možnost vytvoření účtu nového prostřednictvím registrace. Dále pro tohoto uživatele je k dispozici případ užití obnovení zapomenutého hesla.

Přihlášený uživatel má dostupné případy užití vytvoření nového itineráře, zobrazení všech vytvořených itinerářů a zobrazení detailu jednoho itineráře. Případ užití zobrazení itineráře, je součásti i případů přidání lokace a dopravy do itineráře, porovnání verzí itineráře a zobrazení přehledu o měnách použitých v itineráři. Dále je zde pro přihlášeného uživatele možnost změny hesla, zobrazení aktuálního kurzovního lístku, podle kterého se všechny měny v aplikaci přepočítávají a možnost odhlášení ze systému.

Z tohoto diagramu budeme dále vycházet při tvorbě ER diagramu a relační databáze, stejně tak při návrhu uživatelského prostředí.

5.3 ER diagram

ER diagram neboli *Entity-relationship diagram* je vztahový model systému, který se vytváří pomocí *Entity-relationship modelu*, což je metoda pro abstraktní a konceptuální vyjádření dat. Je vhodný při návrhu informačních systémů a jiných aplikací. Do ER diagramu se zahrnují veškeré entity, které systém bude obsahovat a dále jsou zde znázorněny vztahy mezi nimi. Tento diagram neznázorňuje výslednou relační databázi ale pouze abstraktní pohled na strukturu výsledné relační databáze, kterou lze z tohoto diagramu za pomocí několika pravidel transformovat.



Obrázek 5.2: Entity-relationship diagram

Na obrázku 5.2 lze vidět výsledný ER diagram pro tuto aplikaci. Informace, které se v něm nachází jsou velmi důležité a při návrhu bylo myšleno i na budoucnost aplikace. Proto je zde připraven vztah mezi uživatelem a itinerářem takový, aby v budoucnu umožňoval

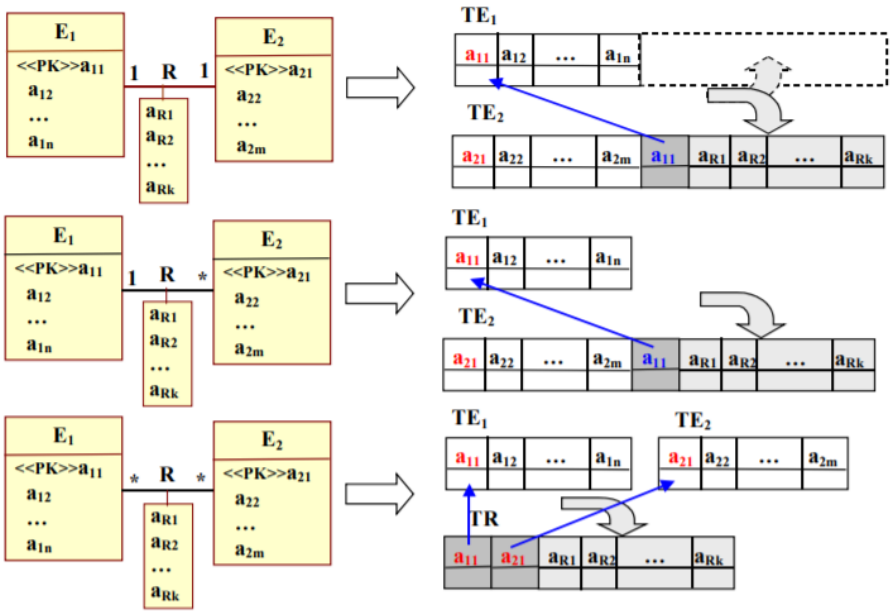
implementaci práv uživatelů k danému itineráře a tím tedy možnost sdílet itinerář mezi více uživateli.

Dále se zde nachází entita pro fixní náklady cesty, která může být obsažena v itineráři. Každý itinerář může obsahovat žádnou nebo více lokací, přičemž každá lokace náleží pouze do jednoho itineráře. Dalším vztahem mezi dvěma entitami je vztah jedna ku jedné mezi entitou lokace a entitou dopravy. Tento vztah není v ER diagramu úplně standartní, ovšem rozhodli jsme se tyto dvě entity od sebe logicky oddělit, jelikož se přece jen jedná o dvě různé věci, i když zde platí, že doprava do lokace je vždy jen jedna a naopak doprava je vždy jen pro jednu lokaci. Tento ER diagram je níže použit pro potřeby vytvoření relační databáze.

5.4 Návrh relační databáze

Relační databáze nám poskytuje pohled na výslednou strukturu celé databáze. Lze ji vytvořit transformací ER diagramu. Celý proces transformace se provádí pomocí stanovených pravidel, které se určují podle vztahu mezi entitami.

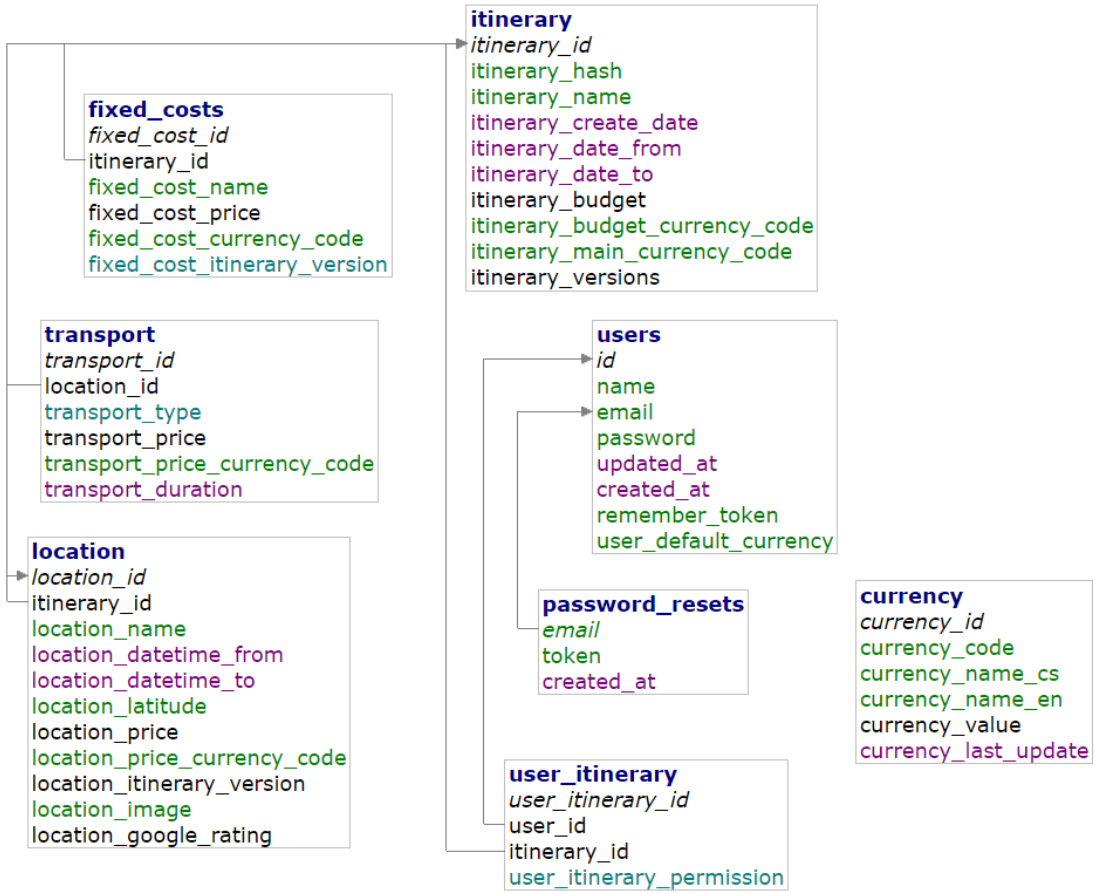
Při vztahu s kardinalitou 1:1 se pouze rozšíří jedna z tabulek entit o cizí klíč druhé entity a atributy jejich vztahu. Vztah 1:N se transformuje tak, že se tabulka entit, která má na svém konci vztah s kardinalitou N, rozšíří o cizí klíč druhé tabulky a o atributy vztahu mezi nimi. Nejsložitější transformace nastane při vztahu N:N. V tomto případě je nutné vytvořit třetí tabulku, která bude obsahovat jako cizí klíče primární klíče obou entitních tabulek a atributy jejich vztahu. Všechna tato pravidla lze přehledně vidět na obrázku 5.3.



Obrázek 5.3: Pravidla pro převod ER diagramu na relační databázi [16]

Za pomoci těchto pravidel jsme převedli ER diagram ze sekce 5.3 na relační databázi. Výsledné schéma relační databáze lze vidět na obrázku 5.4. Pro entity uživatele a itineráře jsme použili třetí pravidlo a vytvořili tak další tabulku, která tyto dvě entity spojuje. U ostatních vztahů poté stačilo použít první nebo druhé pravidlo, tedy jen jednu z tabulek rozšířit o cizí

klíče primárních klíčů tabulky druhé. Přidali jsme navíc tabulku *password_resets*, která je potřebná pro funkci autentizace z kapitoly 4.4. Dále byla přidána tabulka *currency*, která slouží pro uchovávání aktuálního kurzu, který bude každý den aktualizován pomocí nástroje pro plánování úkolů, popsaném v kapitole 4.5.



Obrázek 5.4: Výsledná relační databáze

5.5 Návrh uživatelského rozhraní

Uživatelské rozhraní je jedna z nejdůležitějších částí aplikace, jelikož je to ta část, kterou běžný uživatel uvidí a se kterou bude pracovat. Proto je důležité před samotnou implementací dobře si rozmyslet veškeré rozložení ovládacích prvků a jiných částí grafického uživatelského rozhraní. Uživatelské rozhraní totiž není jen vzhled aplikace nebo její částí. Jde zde především o zajištění komfortního ovládání pro uživatele, přehlednost a jednoduchost pro dosažení cílů uživatele v rámci dané aplikace.

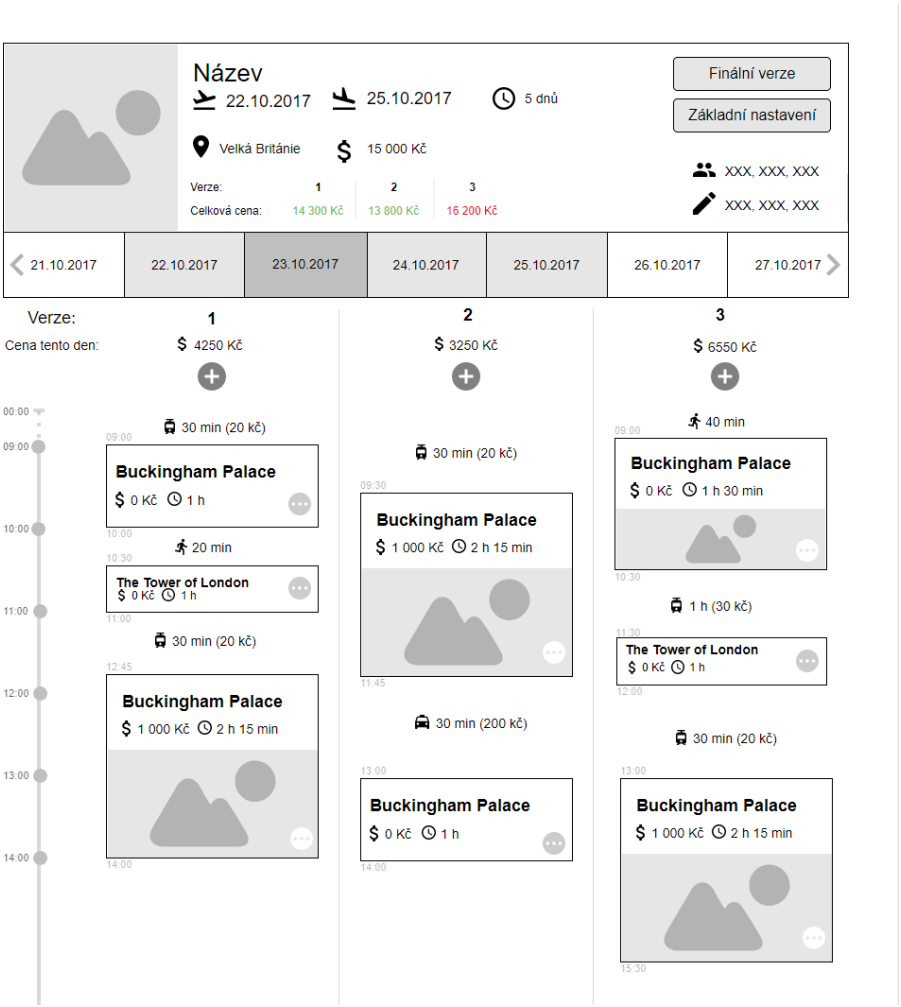
Mezi základní pravidla pro vytvoření úspěšného uživatelského rozhraní můžeme zahrnout:

- přehlednost,

- efektivita,
- intuitivní ovládání,
- přiměřený počet věcí, které si uživatel pro používání musí pamatovat.

Tyto pravidla jsme si stanovili i při návrhu uživatelského rozhraní této aplikace a na základě nich spolu s výsledkem analýzy konkurenčních aplikací v kapitole 2.5 jsme vytvořili návrh uživatelského rozhraní, který si níže popíšeme.

Pro tento návrh jsme vytvořili drátěný model uživatelského rozhraní tzv. *wireframe*. Jedná se o nákres rozhraní, které nám utvoří představu o výsledném vzhledu a rozložení aplikace. Pro tvorbu wireframů je možno použít obyčejný papír a tužku nebo některý z online nástrojů, které jsou na internetu dostupné. My jsme si pro tvorbu wireframe vybrali nástroj *Moqups* [23] a vytvořili wireframe pro stránku s detailem itineráře, která bude v této aplikaci pro uživatele klíčová.



Obrázek 5.5: Wireframe pro stránku detailu itineráře

Na obrázku 5.5 můžeme vidět wireframe stránky s plánováním itineráře. V horní části se nachází základní přehled o cestě. Pro přehlednost je zde obrázek související s danou cestou,

název itineráře, datum zahájení cesty a datum ukončení spolu s celkovou délkou cesty ve dnech. Mezi další informace, které zde lze vidět patří cílová destinace, maximální rozpočet na tuto cestu a poté přehled jednotlivých verzí z finančního hlediska.

Pod tímto přehledem se nachází posuvná lišta s jednotlivými dny itineráře, mezi kterými lze přepínat aktivitu. Níže se již nachází část, která zobrazuje vybraný den cesty. Nejprve se zde nachází označení verzí spolu s celkovou cenou dané verze ve vybraném dni a tlačítkem pro přidání nové lokace do tohoto dne.

Jednotlivé lokace jsou zde zobrazeny na svislé časové ose, která znázorňuje časový průběh dne. Po pravé straně časové osy se pak nachází jednotlivé lokace, které jsou vertikálně umístěny podle času v kterém začínají. Stejně tak výška těchto lokací je určena podle časové délky lokace. Díky tomu jsou všechny lokace rozloženy po časové ose a uživatel tak bude mít dokonalý přehled v jaké části dne se jednotlivá místa nachází. Nad lokacemi je zobrazen typ dopravy na toto místo, spolu s údaji o trvání a ceně dopravy. Samotná lokace má několik rozložení podle její výšky. V lokaci se zobrazují základní informace jako je název, cena a doba trvání. Dále zde je tlačítko pro zobrazení detailu, úpravu nebo mazání lokace. Nad a pod každou lokací je malým písmem zobrazen přesný čas začátku a konce lokace. V případě lokací s větší výškou je součástí zobrazení i obrázek z dané lokace, což zvýší přehlednost a orientaci mezi všemi lokacemi.

5.6 Analýza možnosti využití Google Map API

V rámci této aplikace jsme se rozhodli pokusit využít API, které poskytuje Google ke své mapové aplikaci. Google pro své mapy nabízí několik druhů API, mezi kterými je i verze pro Javascript, kterou bychom mohli využít v naší aplikaci. Toto API nabízí přístup do veškeré databáze míst, která je dostupná přes webové rozhraní Google Map. Díky tomu by bylo možné implementovat našeptávač lokací a zároveň sbírat informace o dané lokaci jako jsou například souřadnice daného místa nebo fotka z daného místa, která by celé aplikaci dodala na přehlednosti a uživatel by podle ní mohl lehce poznat o jaké místo se jedná.

Podle dokumentace, která je pro toto API dostupná jsme zjistili, že implementace těchto požadovaných funkcí je reálná a API tyto informace o jednotlivých místech poskytuje. V případě, že toto API chceme využít pro volně dostupnou aplikaci poskytovanou zdarma, je možné využít tarif Standard, který nabízí zdarma až 25 000 dotazů denně, což nám pro potřeby této práce vystačí. [12]

5.7 Fáze implementace

Předtím než započneme samotnou implementaci, je potřeba si ji nejprve rozvrhnout do několika částí, podle kterých budeme postupovat.

1. **Příprava** - v této části si přichystáme všechny potřebné nástroje a technologie. Je zde zahrnuta konfigurace Laravelu, vytvoření a inicializace databáze podle vytvořené relační databáze ze sekce 5.4.

2. **Vytvoření šablony** - zde se zaměříme na tvorbu frontendu a vytvoříme si potřebné šablony pro tuto aplikaci. Vycházet budeme jak z požadované funkcionality tak z návrhu uživatelského rozhraní ze sekce 5.5. Dále v této fázi navrhneme i paletu barev, kterou dále budeme využívat a také logo aplikace.

3. **Implementace backendu** - implementujeme všechny potřebné modely a pomocné třídy, které budou zajišťovat veškerou logiku aplikace a komunikaci s databází. Vycházet budeme z požadované funkcionality, tedy z vytvořeného diagramu případu užití v sekci 5.2.

4. **Propojení backendu a frontendu** - vytvoříme kontroléry, které nám propojí vytvořený backend s frontendem.

5. **Finální úpravy a testování** - v poslední části implementace odladíme chyby, doplníme napojení na API Google Map, automatickou aktualizaci měny a dále provedeme poslední úpravy celé aplikace. V této fázi provedeme také osobní testování na vybraném vzorku uživatelů spolu s vyhodnocením.

Kapitola 6

Implementace

V této části práce si popíšeme tu nejdůležitější a nejsložitější část vývoje celé aplikace a tou je samotná implementace. Implementace vychází z návrhu aplikace a jedná se o proces, který realizuje všechny požadované funkce a rozhraní.

Aplikaci jsme se rozhodli implementovat pomocí webových technologií s využitím frameworku Laravel popsaném v kapitole 4. Webovou aplikaci jsme vyvíjeli na webovém serveru, na kterém jsou dostupné všechny potřebné technologie, včetně nejnovější verze PHP. Díky vývoji přímo na webovém serveru dostupném z internetu, jsme zajistili přenositelnost a možnost testování aplikace oslovenými uživateli již v samotném průběhu implementace. Aplikace je dostupná na webové adrese `http://itinerary.tomasfilak.cz`.

Celý proces implementace si zde rozdělíme do několika částí. První bude instalace všech potřebných nástrojů na server jako je Laravel, lokalizace a jiné a dále jejich konfigurace. Poté se přesuneme k vytvoření schématu databáze a její inicializaci. Další v pořadí bude vytvoření a implementace šablony aplikace a poté se zaměříme na implementaci samotného backendu a jeho propojení s frontendem. Nakonec celou aplikaci podrobíme testování, díky kterému získáme zpětnou vazbu od uživatelů a odhalíme chyby.

6.1 Konfigurace Laravelu a databáze

Jako první bylo za potřebí samotný Laravel na server nainstalovat. Instalace Laravelu se provede pomocí nástroje *composer* následovně:

```
composer global require "laravel/installer"
laravel new "nazev adresare"
```

Díky tomu jsme získali veškeré potřebné soubory laravelu a můžeme pokračovat s konfigurací. Jako první je potřeba nastavit soubor *public/index.php*, jako domovský soubor, který zpracovává veškeré HTTP požadavky zaslané na server. Další část konfigurace samotného laravelu se nachází v adresáři *config*. V této složce se nachází soubory s údaji, které se nastavují dle příslušných proměnných prostředí. Tyto proměnné můžeme konfigurovat v souboru *.env*, který se nachází v kořenovém adresáři Laravelu. Zde lze nastavit jméno aplikace, URL adresu. Dále zde můžeme přepínat mezi produkční a ladící verzí a samozřejmostí je i nastavení samotného připojení k databázi a emailovému serveru.

Nastavení lokalizace pro více jazyků je popsáno v kapitole 4.3. My jsme je doplnili ještě balíkem rozšiřujících funkcí *mcamara/laravel-localization* [7]. Ten nám zajistí veškeré nastavení ve směrování a celou lokalizaci rozšíří o užitečné funkce.

Následně jsme provedli vytvoření struktury databáze. Databázi jsme vytvořili pomocí nástroje Adminer [26]. Ten poskytuje přehledné uživatelské prostředí, díky kterému máme možnost přehledné konfigurace databázové struktury. Databázi jsme vytvořili na základě diagramu relační databáze z kapitoly 5.4. Vytvořili jsme příslušné tabulky a jejich atributy a tím získali inicializovanou databázi, se kterou jsme mohli dále pracovat na straně aplikace.

6.2 Tvorba šablony uživatelského rozhraní

Při tvorbě šablony jsme vycházeli z diagramu případů užití z kapitoly 5.2 a návrhu uživatelského prostředí z kapitoly 5.5. Nejprve jsme si určili barevnou paletu této aplikace, kterou budeme využívat. Díky tomu bude barevnost aplikace jednotná, přehledná a nerušivá pro uživatele. Již v této části nesmíme zapomenout, že aplikace má být vícejazyčná. Proto je potřeba veškeré texty, které budou zobrazeny uživateli psát pomocí pomocných funkcí pro lokalizaci popsaných v kapitole 4.3.

Obrázek 6.1: Vybraná barevná paleta

Jelikož bude funkce plánování itineráře obsahovat i vytváření několika verzí, které jsou v návrhu vedle sebe ve sloupcích, bude pro aplikaci žádoucí aby měla dostatek místa do šířky. Proto jsme se rozhodli pro umístění menu do horní části jako jednu lištu a samotný obsah stránek umístit až pod něj. Díky tomu budeme mít pro obsah celou šířku okna. Pro tento navigační panel jsme zvolili světlejší červenou barvu z naší palety. Menu jsme připravili i pro rolovací nabídku, které jsme dali moderní poloprůhledný vzhled, kdy lze vidět rozmazaně i obsah pod touto vysunutou nabídkou.

Dále jsme vytvořili šablonu pro tvorbu nového itineráře. Jedná se o formulář, ve kterém uživatel zadá jeho jméno, rozpočet a počet verzí. Dále pak časový úsek, který specifikuje datum začátku a konce cesty. Pro pohodlné a intuitivní zadávání tohoto časového úseku jsme využili javascriptový modul Daterangepicker [13]. Ten rozšiřuje knihovnu jQuery a modul moments [18] a zajistí tak pohodlné zadávání kalendářních úseků uživatelům. Pro popisky textových polí, jsme se rozhodli využít ikony z fontu Material Design Icons [15], které reprezentují daný popisek a při aktivaci pseudotřídy hover, tedy při přejetí nad ikonou myší, se zobrazí detailní textový popisek.

Nejdůležitější šablonu aplikace a to samotné plánování itineráře, jsme rozvrhli do několika souborů tak aby bylo možné pohodlné načítání jen částí stránky a mohli jsme tak zajistit plynulé aktualizování obsahu bez znovu načítání celého webu. Hlavní soubor šablony se jmenuje *planner.blade.php*.

Horní část stránky s detailem itineráře obsahuje základní informace o itineráři spolu s náhledovou fotografií místa z itineráře. Dále zde jsou informace o datumu začátku a konce cesty a jejich popisky zvolíme opět formou ikon. Pod těmito informacemi implementujeme tabulku, která bude obsahovat základní porovnání verzí. Toto porovnání bude nutné při přidaní nebo úpravě nové lokace vždy aktualizovat, proto jsme ji vložili do samostat-

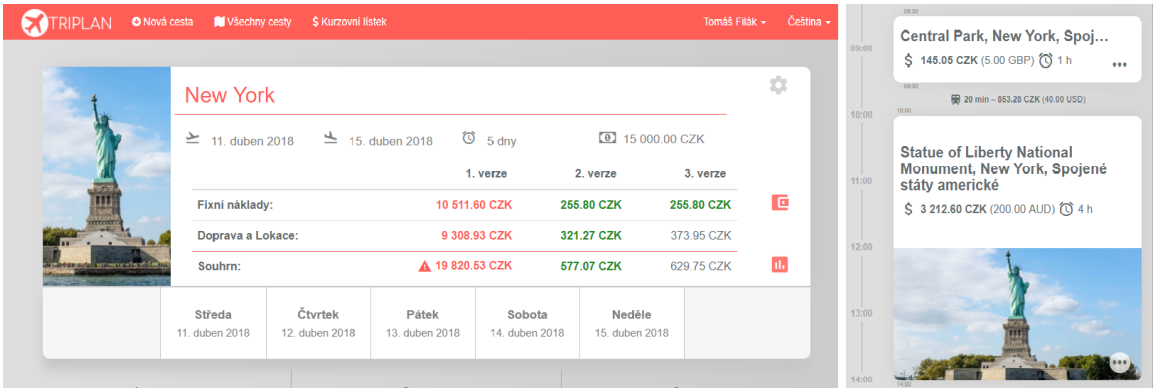
ného souboru *summary-table.blade.php*. Vedle této tabulky jsou tlačítka pro otevření správy fixních nákladů a pro zobrazení podrobných statistik.

Další částí této strany je lišta s přepínáním dnů itineráře. Pro zajištění dobře ovladatelné posuvné lišty, jsme využili modul Slick [28], který nabízí Javascriptovou třídu pro tvorbu carouselů. V něm budou buňky s jednotlivými dny itineráře mezi kterými půjde přepínat.

Pod touto lištou zobrazujeme již jednotlivé lokace podle verzí ve vybraném dni. Verze jsou rozvrhnuty do sloupců pomocí mřížkového systému dostupného v Boostrapu a popsaného v kapitole 3.8. Po levé straně těchto třech sloupců jsme vytvořili časovou osu, která bude zajišťovat lepší orientaci v čase. V jednotlivých sloupcích se pak nachází tlačítko pro přidání nové lokace a poté již samotné lokace v bílých kartách. Ty jsme všechny vytvořili jednotně v HTML a až pomocí CSS jsme je rozdělili na čtyři kategorie podle velikosti. Tak bude stačit jednoduše generovat všechny lokace stejně a jen přepínat jejich třídy, podle kterých se určí výsledné CSS. Tvorbu rozdílných vzhledů lokací, jsme zvolili z důvodu proměnné výšky jednotlivých lokací v závislosti na jejich době trvání. Díky tomu jsme zajistili, že se u delších lokací neplýtvá místem a naopak u těch malých je stále vše přehledně zobrazeno. V kartě lokace zobrazujeme její nazev, cenu, dobu trvání a v případě dostatečně vysoké lokace i fotografii. V prostoru mezi dvěma lokacemi se nachází informace o dopravě na dané místo. Po kliknutí kdekoliv na kartě lokace se otevře detail lokace, který bude umožňovat čtení více informací o lokaci a hlavně jejich upravení. Dále se na kartě nachází tlačítko pro kontextové menu, které obsahuje možnosti zobrazení detailu lokace, přesunutí do navigace na dané místo a smazání lokace.

Šablona využívá komponentu *modal* z frameworku Bootstrap, což je komponenta pro zobrazení vyskakovacího okna uprostřed stránky aplikace bez načítání další stránky nebo otevírání nového okna prohlížeče. V tomto okně implementujeme zobrazení nastavení itineráře, statistik, fixních nákladů nebo přidávání nové lokace. V případě přidávání nové lokace je toto okno rozděleno do dvou částí, kdy první obsahuje pole pro zadání informací o dopravě a v druhé části jsou pole pro zadávání informací o samotné lokaci. V této části jsme řešili například zadávání času, kdy jsme hledali nejvhodnější způsob. Jedním způsobem byl tzv. kruhový styl zadávání, kdy se uživateli nejprve zobrazí kruh s hodinovými hodnotami a po vybrání se zobrazí kruh s minutovými hodnotami. Po konzultaci s několika testovacími uživateli jsme nakonec zvolili jednodušší způsob a to použití dvou rozbalovacích nabídek pro hodiny a minuty. Uživatelé ji hodnotili kladněji z důvodu, že když chtějí například změnit jen minutovou část času, stačí jim manipulace pouze s jednou rozbalovací nabídkou. U varianty s kruhovým výběrem museli vždy znovu zadávat i hodiny.

Na obrázku 6.2 lze vidět vybrané části šablony, které jsme v této kapitole naimplementovali a popisovali. V další fázi implementace se tentokrát zaměříme na backend, který budeme později napojovat na tuto šablonu a díky tomu již pro šablonu získáme potřebná data, která budeme zobrazovat a budeme tak moci šablonu případně rozšířit zejména o Javascriptové prvky.



Obrázek 6.2: Výsledek implementace šablony pro stránku plánování itineráře

6.3 Backend - logika aplikace

Jelikož se ve frameworku Laravel využívá architektura MVC z kapitoly 3.2, většina logiky aplikace probíhá převážně v modelech. V našem případě se v aplikaci vyskytují entity měna, itinerář, lokace, doprava a uživatel. Pro všechny tyto entity jsme tedy vytvořili příslušné modelové třídy, které obsahují metody pro práci s těmito entitami. Jako první model zde máme model *User*, který nám vytvořil samotný Laravel při nastavení autentizace, což je popsáno v kapitole 4.4.

Model pro měnu s názvem *Currency* obsahuje tři metody mezi které patří metoda pro získání dat o měně. Metoda automaticky vybere název měny podle vybrané lokalizace aplikace a označí měnu, kterou má uživatel nastavenou jako výchozí. Další je metoda pro aktualizaci měny. Po zavolání této metody se z webu České Národní Banky [5] stáhne textový soubor obsahující denní kurz vybraných měn. Tento soubor zpracuje a hodnotu kurzu uloží do tabulky k příslušným měnám. Tato metoda je volána vytvořeným příkazem *UpdateCurrencyValue*. ČNB vydává denní kurz každý všední den vždy v 14:30, proto jsme nastavili provádění tohoto příkazu každý den v 14:40 pomocí funkce plánování úkolu z kapitoly 4.5. Díky tomu máme v databázi vždy aktuální informace o kurzech měn. V modelu pro měny nechybí ani funkce pro převod peněžní částky v jedné měně do měny jiné. Této metodě se předá částka, aktuální měna a požadovaná měna. Metoda na základě aktuálního kurzu daných měn v databázi přepočítá částku a tu vrátí zpět.

Další entitou pro kterou jsme vytvořili model s názvem *Transport* je doprava. Ta obsahuje metody pro získání dat z databázové tabulky *transport*, jejich vytváření a úpravu. Pro entitu lokace je vytvořen model *Location*. Ten obsahuje metody pro zpracování dat z formuláře o nové lokaci a přidání této nové lokace do databáze. Zároveň volá metodu pro vytvoření dopravy na tuto lokaci. Dále zde je metoda pro úpravu a mazání lokace a také metody pro získání dat o vybrané lokaci ve formátu json. Tato metoda nám slouží při načítání dat o lokaci pomocí ajaxu, při najetí do detailu lokace.

Posledním a nejdůležitějším modelem je model *Itinerary*. Tento model mimo vytváření nového itineráře nebo jeho úpravě a mazání nabízí i metodu pro získání veškerých potřebných dat o celém itineráři, včetně všech lokací, fixních nákladů a jiných informací. Dále zajišťuje výpočet statistik o cestě, tak aby byly vždy aktuální a výpočty týkající se velikosti časové osy, výšky jednotlivých lokací a prostory mezi nimi. Veškeré výpočty se provádí právě v této metodě a to zejména kvůli optimalizaci aplikace, jelikož se veškeré lokace pro-

chází pouze v jednom cyklu, ve kterém se k nim dopočítá vše potřebné. Také se zde nachází metoda pro získání všech itinerářů, ke kterým má daný uživatel práva. Posledními třemi metodami, které jsou v tomto modelu obsaženy jsou metody pro přidání, úpravu a mazání fixních nákladů itineráře.

6.4 Propojení šablon s backendem a kompletace celé aplikace

Propojení frontendu a backendu aplikace se v architektuře MVC provádí v kontrolérech. Zde se získaná data z modelů předají šabloně, která je interpretuje uživateli. Dále zde probíhá zpracování požadavků od uživatele z formulářů.

Vytvořili jsme kontrolér *UserController.php*, ve kterém se nachází metoda *profile*, která zajišťuje zobrazení uživatelského profilu s nastavením. Zajišťuje tedy i zpracování formuláře, pro změnu nastavení nebo uživatelského hesla. Dalším kontrolérem je *PageController.php*, který obsahuje metody pro zobrazení úvodní strany a také kurzovního lístku.

Nejdůležitějším kontrolérem je *ItineraryController.php*. Ten obsahuje všechny metody obsluhující stránky, které pracují právě s itinerářem. První metodou tohoto kontroléru je metoda *list_itineraries*, která je volána při zobrazování stránky se seznamem vytvořených itinerářů. Tato metoda nejprve zavolá metodu pro získání itinerářů, které přihlášený uživatel vlastní. Tyto data jsou poté předány šabloně *itinerary.list*, která tato data zobrazí uživateli. Další metodou je metoda *planner*. Této metodě se skrz url adresu předává parametr hash itineráře. Tento hash má každý itinerář unikátní a zajišťuje pravděpodobnost uhodnutí url adresy určitého itineráře. Pro hash jsme se rozhodli kvůli možnému budoucímu rozšíření o sdílení itineráře s neregistrovanými uživateli pouze pomocí speciálního odkazu obsahující právě tento hash itineráře. Tato metoda volá model itineráře, který vrátí veškeré informace o itineráři, včetně všech lokací a statistik. Tyto data kontrolér poté předává šabloně *itinerary.planner*, kterou jsme v této fázi rozšířili o mnoho Javascriptových funkcí. Tyto změny si blíže popíšeme níže. Metoda *planner* také obstarává data z formulářů pro úpravu informací o itineráři nebo přidávání a úpravu fixních nákladů. Mezi další metody v tomto kontroléru patří metody pro vytváření nebo mazání lokací, které jsou volány pomocí ajaxu. Dále pak metody pro získání částí stránky plánování, tedy metody pro získání lokací v jednom dni nebo tabulky se statistikami. Tyto metody jsou také volány pomocí ajaxu ze stránky plánování a zajišťují tak plynulou aktualizaci zobrazených dat bez nutnosti aktualizace celé stránky.

Rozšíření šablony stránky s plánováním itineráře

V této fázi vývoje proběhlo několik úprav šablony pro stránku plánování itineráře. Jedná se zejména o rozšíření šablony o Javascriptové funkce, které zajišťují plynulou aktualizaci stránky nebo například validaci uživatelem zadaných dat.

Jako první jsme pomocí Javascriptu zajistili otevření správného dne itineráře podle URL adresy. K tomu jsme využili část URL používané pro odkazování na identifikátory elementů zvané hash. Do ní pomocí Javascriptu vkládáme vždy identifikátor aktuálně zobrazeného dne. Díky tomu si může uživatel adresu zkopírovat a uložit. Po jejím otevření se dostane na dříve vybraný den. Tato funkce se bude hodit zejména v rozšíření, kdy bude moci uživatel sdílet itinerář pomocí odkazu.

Pro tvorbu nové lokace i pro zobrazení detailu již vytvořené lokace máme v šabloně pouze jedno okno se stejným formulářem. Odkazy, které otevírají okno detailu mají navíc atribut s informací o identifikátoru lokace. Díky tomu se za pomoci ajaxu zavolá daná funkce

kontroléru, která vrátí veškeré informace o lokaci a tyto informace vložíme do příslušných polí. Po vytvoření nové lokace nebo při aktualizaci již existující se pomocí ajaxu aktualizuje den cesty, ve kterém proběhla změna a také tabulky se statistikami o cestě. Při jakékoliv změně času u lokace nebo dopravy se volá funkce *verify_time*, která zajistí kontrolu zadaného času a času ostatních lokací v dané verzi a dni itineráře. Detekuje tak případné kolize, na které uživatele upozorní a zabrání mu uložení této kolizní lokace vypnutím tlačítka pro odeslání.

Dále se zde nachází funkce využívající ajax pro aktualizaci dne itineráře, tabulek s porovnáním verzí nebo také pro tvorbu, úpravu nebo mazání lokací a fixních nákladů. Níže můžete na obrázku 6.3 vidět snímek konečného uživatelského rozhraní této aplikace na stránce s plánováním itineráře.

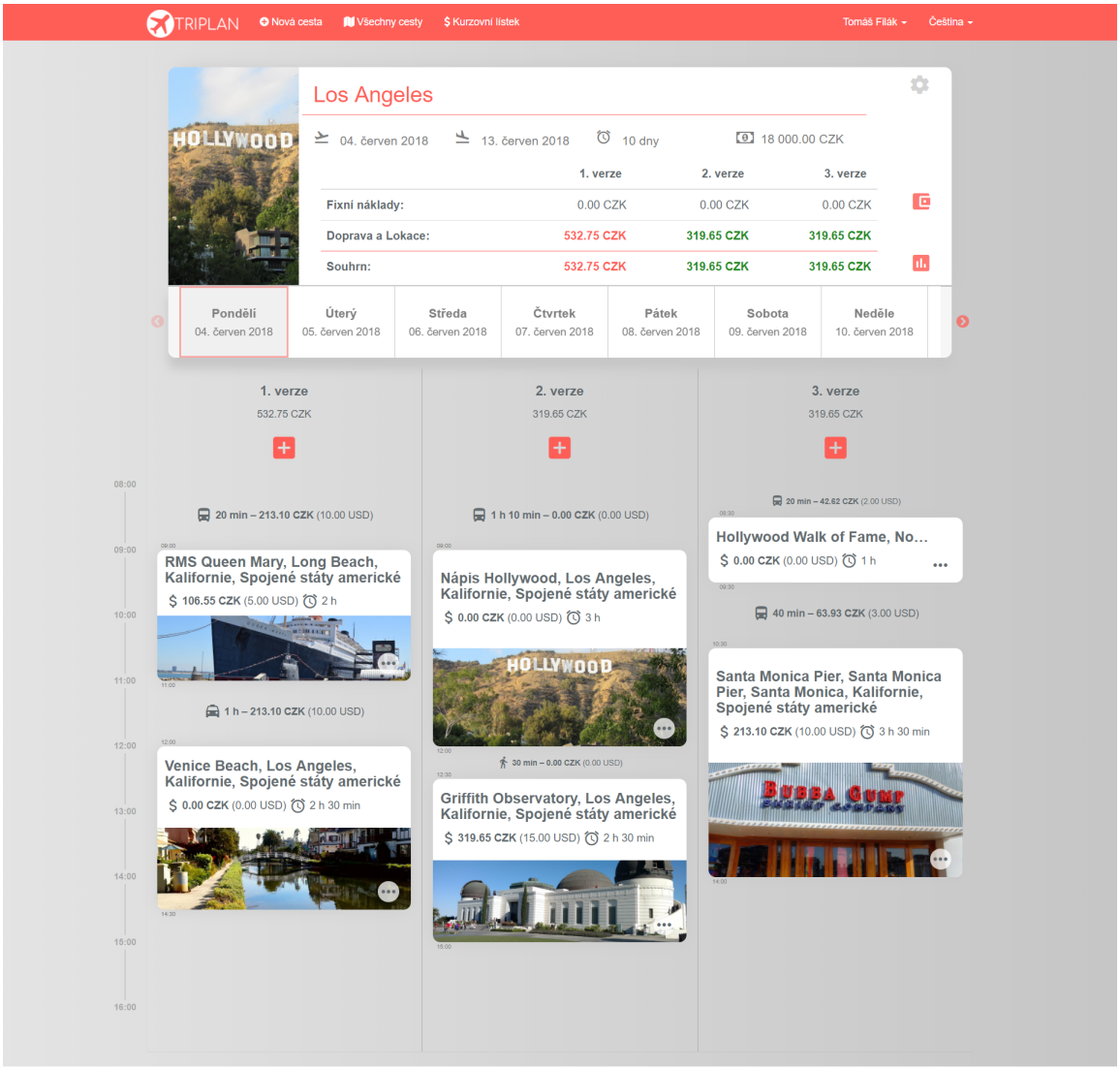
Napojení na API Google Map

Dále jsme se rozhodli pro napojení aplikace na API Google map [12]. Toto API využijeme k našeptávání lokací a získání dalších informací o lokaci. Využili jsme API určené pro Javascript a podle dokumentace vše nastavili tak aby se při zadávání lokace do určeného pole spustilo našeptávání všech míst, které obsahuje databáze Google Map. Pokud uživatel přes tento našeptávač vybere nějaké umístění, získáme objekt se základními informacemi o tomto místě.

Ze získaného objektu si vybereme souřadnice, odkaz na fotografii místa a hodnocení. Tyto informace vložíme do skrytých polí formuláře, tak abychom je mohli po odeslání formuláře pro přidání nebo úpravu lokace uložit do databáze. Fotografii lokace tak můžeme zobrazit uživateli u každé lokace a navíc vybereme ze všech lokací v jednom itineráři tu nejoblíbenější podle hodnocení v Google Mapách a fotografii této nejoblíbenější lokace zobrazíme uživateli jako hlavní fotku celého itineráře.

Tímto napojením na Google Mapy jsme aplikaci rozšířili především o zjednodušení zadávání lokací pro uživatele, kdy nemusí psát celý název a aplikace mu místa sama našeptává. Další výhodou jsou získané informace, se kterými můžeme dál pracovat jako je například fotografie vybraného místa. Díky Google Mapám můžeme i generovat odkaz, který směřuje na stránku navigace k určitému místu. V případě že uživatel aplikaci využívá přímo na cestě, má možnost snadného zapnutí navigace přímo na vybranou lokaci.

Napojení aplikace na Google Mapy má velký potenciál a v budoucnu by šlo toto napojení rozšířit o další funkce.



Obrázek 6.3: Snímek výsledné podoby stránky s plánováním itineráře

6.5 Testování

Testování je důležitou součástí vývoje jakékoliv aplikace a zajišťuje včasné odhalení chyb a nedostatků. Někdy může být testování poněkud podceňované, což se nemusí vyplatit. Aplikace by se nikdy neměla dostat do produkční verze dokud není dostatečně otestována.

V průběhu vývoje jsme aplikaci testovali sami a snažili se tak odladit veškeré chyby, které se objevily a zároveň jsme konzultovali části aplikace s potencionálními uživateli. Od nich jsme tak mohli získat zpětnou vazbu, která přinesla odhalení chyb, kterých jsme si sami nevšimli, nebo nové funkce, které by mohli aplikaci rozšířit.

Po dokončení prototypu aplikace jsme provedli rozsáhlý test, který zahrnoval vytváření několika různých itinerářů a hledání co nejvíce možných scénářů cesty, které jsme do aplikace zadávali. Výsledkem bylo odhalení několika chyb, které byly převážně v uživatelském rozhraní a našli závažnější problém a to nemožnost zadávání lokace, která začíná v jednom

dni a končí v druhém. Proto bylo zadávání času lokace rozšířeno o přepínač, který značí, že lokace končí až další den v zadaný čas. Po odlazení posledních chyb, které jsme v této části testování nalezli jsme se přesunuli k testování na uživatelích.

6.5.1 Uživatelské testování

Další částí testování bylo testování na samotných uživatelích. Jako formu testování jsme zvolili osobní dotazník, kdy jsem se sešli s deseti uživateli. Mezi tyto uživatele patří ti, kteří jsou potencionálními uživateli této aplikace a rádi cestují. Díky tomu jsme od nich mohli získat relevantní poznatky a návrhy. Uživatelům jsme nejprve představili stručně naši aplikaci aby se dozvěděli k čemu slouží. Dále jsme jim předali scénář cesty do Londýna, který obsahoval několik lokací pro dvě verze cesty a fixní náklady. Uživatel si tak prošel celý proces registrace a následnou tvorbu itineráře. Ve scénáři nebyly popisovány prvky aplikace ani jejich umístění. Uživatel tak dostal jen informaci co a kdy navštíví a v samotné aplikaci se již musel zorientovat sám. Při tvorbě itineráře jsme uživatele pozorovali a zapisovali si poznámky s jakou části měli problém. Kterou funkci hledali příliš dlouho, nebo která tlačítka se jim pletla. Po každé jednotlivé části scénáře, jsme se uživatele dotázali na celkové hodnocení této části na stupnici 1 až 10, kdy 10 je nejlepší a zapsali si jeho komentář k této části, ve kterém nám mohl popsat jeho názor.

Výsledkem tohoto testování bylo objevení několika chyb v uživatelském rozhraní aplikace a hlavně jsme získali zpětnou vazbu a návrhy na rozšíření. První částí testování byla samotná registrace. Tu všichni uživatelé zvládli bez problému a neměli k ní žádné výhrady. V další části je čekalo vytvoření samotného itineráře. V této části upozorňovali na nemožnost zadání počtu verzí již při vytváření itineráře a nutnost tak po vytvoření hledat nastavení, kde lze až dodatečně nastavit počet verzí. Další částí bylo vytváření samotných lokací spolu s dopravou. Uživatelé hodnotili kladně zadávání lokací doplněné o našeptávání a také hlášení kolizí s jinými lokacemi. Někteří uživatelé si u hlášení kolizí ale stěžovali na nejasné kolize, kdy nechápali, že aplikace bere v úvahu i zadaný čas dopravy na dané místo. Dále jeden z uživatelů upozornil, že by nabídka při zadávání měny měla být seřazena podle abecedy. V další části scénáře měli za úkol upravit cenu dopravy u již vytvořené lokace. Většina automaticky klikla rovnou na samotnou kartu lokace, což je přesunulo správně na detail lokace, kde mohli potřebný údaj upravit. Vytváření lokací pro jiné verze bylo uživateli celkově hodnoceno kladně. Uživatelům chvíli trvalo hledání tlačítka pro přidávání fixních nákladů. Často jej nedokázali rozlišit od tlačítka pro zobrazení statistik. U zobrazení statistik hodnotili kladně zvýraznění hodnot, které vyobrazují výhodnost dané verze jak z hlediska časového tak i finančního. Někteří ale navrhovali doplnění statistik o procentuální vyjádření ušetřených peněz s jednou verzí oproti jiným nebo oproti rozpočtu na cestu. Posledním úkolem bylo otevřít si navigaci na vybranou lokaci. Zde hodnotili záporně nedostatečně výrazné tlačítko pro vyvolání kontextové nabídky u každé lokace.

Níže lze v tabulce 6.1 vidět výsledná průměrná hodnocení uživatelů k jednotlivým částí scénáře. Celý scénář, tak jak byl uživatelům zadán, lze pak nalézt v příloze C na konci této práce. Spolu s celým scénářem cesty se zde nacházejí i nejčastější připomínky, které uživatelé k jednotlivým částem měli nebo naše poznámky, které jsme vypozorovali.

Tabulka 6.1: Shrnutí výsledků uživatelského testování

Část scénáře	**Průměrné hodnocení**
1. - Registrace do aplikace	10,0
2. - Vytvoření nového itineráře pro výlet do Londýna	8,8
3. - Přidání lokace *"Madame Tussaunds"*	8,2
4. - Přesun pěšky k *"Hyde Park"*	8,6
5. - Změna ceny dopravy do *"Madame Tussaunds"*	9,0
6. - Nové lokace *"Buckinghamský palác"* a *"Big Ben"* pro druhou verzi itineráře	9,2
7. - Přesun na další den a přidání dvou nových lokací pro první verzi	9,1
8. - Přídaní dvou fixních nákladů	8,6
9. - Porovnávní verzí itineráře	8,3
10. - Spuštění navigace na vybranou lokaci	7,2

6.5.2 Vyhodnocení uživatelského testování

Na základě uživatelského testování jsme ze získaných hodnocení a komentářů provedli jeho vyhodnocení. Díky tomuto testování jsme zjistili několik chyb v aplikaci, které jsme opravili a navíc nás uživatelé upozornili na několik nedostatků aplikace. Jako první jsme tedy na základě připomínek uživatelů přidali možnost výběru počtu verzí přímo do formuláře pro vytvoření itineráře. Dále jsme doplnili hlášku o kolizi při tvorbě lokace o upozornění že vzniklá kolize může být způsobená i překrýváním času dopravy na tuto lokaci. Dále byla doplněna tlačítka pro zobrazení fixních nákladů a statistik o popisek po přejetí myší, který je využíván i v jiných částech aplikace. Nabídku pro výběr měn jsme seřadili podle abecedy.

Uživatelé aplikaci hodnotili celkově velmi kladně a všichni na otázku zda by ji reálně využívali při vytváření cesty odpověděli ano. Někteří dokonce poznamenali, že jim podobná aplikace chyběla. Kladně hodnotili i uživatelské rozhraní, které se jim zdálo velmi přehledné a atraktivní. Navrhnuli nám další vylepšení a funkce, o které by se aplikace mohla dále rozšířit. Mezi ně patřila možnost vytváření jedné lokace, která by byla součástí více verzí itineráře. Dále pak hloubější integrace s Google Mapami, kdy by aplikace mohla automaticky navrhovat místa podle zadaného typu, například restaurace nebo bar. Zeptali jsme se také uživatelů, zda by využili mobilní aplikaci, ve které by mohli mít uložené své itineráře a prohlížet si je off-line na cestách. Tato mobilní aplikace by byla pro uživatelé vítaná a určitě by ji využili. Všechny tyto doporučení na další rozšíření zahrneme do kapitoly o možném rozšířením aplikace v kapitole 6.6.

6.6 Možná rozšíření

Hotová aplikace má spoustu funkcí a možností, které uživatelům usnadňují tvorbu a plánování itineráře. Z uživatelského testování vyplývá, že se aplikace může stát oblíbenou pomůckou pro cestovatele. Z našeho návrhu a implementace a také z uživatelského testování vyplývá několik možných rozšíření aplikace do budoucna. Na některá rozšíření se myslelo i při návrhu celé aplikace a ta je tak na tato rozšíření částečně připravena. Níže je tedy výčet možných rozšíření, které by mohli aplikaci do budoucna přidat nové funkce.

- rozšíření integrace s Google mapami (navrhování lokací, výpočet informací o dopravě, více informací o lokacích),
- možnost vytvořit jednu lokaci pro více verzí itineráře,
- mobilní aplikace pro prohlížení itineráře off-line na cestách,
- export lokací ve formátu iCalendar pro následný import do elektronických kalendářů,
- větší optimalizace aplikace pro mobilní telefony.

Kapitola 7

Závěr

Cílem této práce bylo vytvořit aplikaci, která by pomohla cestovatelům s plánováním itineráře. Aplikace měla zvládat tvorbu itineráře ve více verzích a jejich porovnávání mezi sebou s finančního i časového hlediska. Před samotným vývojem jsme provedli analýzu konkurenčních aplikací. Díky tomu jsme získali možnost rozmyslet si celou funkcionalitu aplikace tak aby byla konkurenceschopná ještě před jejím samotným návrhem. Dále bylo potřeba před započetím vývoje nastudovat postupy při tvorbě webové aplikace a všechny potřebné technologie. Nejdůležitější části bylo seznámit se s problematikou frameworku Laravel, ve kterém jsme se rozhodli tuto aplikaci vyvíjet. Provedli jsme také analýzu využitelnosti integrace API Google map do naší aplikace a rozhodli se pro její implementaci. Dále jsme započali návrh a samotnou implementaci aplikace spolu s napojením na kurzovní lístek České národní banky a Google Map. Provedli jsme také testování aplikace včetně testování uživatelského, ve kterém jsme získali zpětnou vazbu a podařilo se nám odladit několik chyb aplikace. Také jsme tímto testováním získali další možnosti k rozšíření, které jsme doplnili o vlastní nápady a zahrnuli do navržených rozšíření aplikace do budoucna.

Výsledkem této práce je aplikace, která má přehledné a atraktivní uživatelské rozhraní a usnadňuje uživateli práci při plánování itineráře. Při tvorbě této práce bylo postupováno podle bodů zadání a dále byla aplikace rozšířená o další funkcionality. Při vývoji byly funkce aplikace konzultovány s potencionálními uživateli. Ti ji při konečném testování hodnotili velmi kladně a rádi by ji v budoucnu využívali pro své plánování itinerářů. Aplikace je plně funkční a připravena pro používání v praxi. Také je připravena na implementaci dalších rozšíření, které jsou výše navrhovány a má tak velký potenciál do budoucna. Aplikace je veřejně dostupná na adrese http://itinerary.tomasfilak.cz.

Literatura

[1] *Bootstrap*. 2018, [Online; navštíveno 20.03.2018].
URL https://getbootstrap.com/

[2] *Find your new favorite web framework*. 2018, [Online; navštíveno 20.03.2018].
URL https://hotframeworks.com/

[3] *Git*. 2018, [Online; navštíveno 20.03.2018].
URL https://git-scm.com/

[4] *The PHP Framework For Web Artisans*. 2018, [Online; navštíveno 20.03.2018].
URL https://laravel.com/

[5] Česká národní banka: *ČNB je ústřední bankou České republiky a orgánem vykonávajícím dohled nad finančním trhem*. 2018, [Online; navštíveno 20.03.2018].
URL http://www.cnb.cz

[6] Bean, M.: *Laravel 5 Essentials*. Packt Publishing Ltd, 2015, ISBN 978-1-78528-301-7.

[7] Cámara, M.: *Laravel Localization*. 2018, [Online; navštíveno 20.03.2018].
URL https://github.com/mcamara/laravel-localization

[8] EPRAVO.CZ: *Cestovní kancelář a cestovní agentura*. 2018, [Online; navštíveno 20.03.2018].
URL https://www.epravo.cz/top/clanky/cestovni-kancelar-a-cestovni-agentura-17287.html

[9] Foundation, M.: *JavaScript*. 2018, [Online; navštíveno 20.03.2018].
URL https://developer.mozilla.org/en-US/docs/Web/JavaScript

[10] Fyrd: *Browser support tables for modern web technologies*. Can I use, 2018, [Online; navštíveno 20.03.2018].
URL https://caniuse.com

[11] GitHub, I.: *GitHub*. 2018, [Online; navštíveno 20.03.2018].
URL https://github.com/

[12] Google, I.: *Build the next generation of location experiences*. 2018, [Online; navštíveno 20.03.2018].
URL https://developers.google.com/maps/

[13] Grossman, D.: *Date Range Picker - A JavaScript component for choosing date ranges, dates and times*. 2018, [Online; navštíveno 20.03.2018].
URL http://www.daterangepicker.com/

[14] Group, T. P.: *PHP*. 2018, [Online; navštíveno 20.03.2018].
URL http://php.net/

[15] Inc., G.: *MATERIAL DESIGN ICONS*. 2018, [Online; navštíveno 20.03.2018].
URL https://material.io/icons/

[16] doc. Ing. Jaroslav Zendulka a Ing. Ivana Rudolfová: *Databázové systémy - Studijní opora*. 2006.
URL https://www.fit.vutbr.cz/study/courses/index.php.cs?id=11434

[17] Inspirock: *Inspirock*. 2018, [Online; navštíveno 20.11.2017].
URL https://www.inspirock.com

[18] Moment.js: *Parse, validate, manipulate, and display dates and times in JavaScript.* 2018, [Online; navštíveno 20.03.2018].
URL https://momentjs.com/

[19] Naramore, E.; aj.: *PHP5, MySQL, Apache Vytváříme webové aplikace*. Computer Press, a.s., 2006, ISBN 80-251-1073-7.

[20] Ndegwa, A.: *What is a Web Application?* MAXCDN One, Květen 2016, [Online; navštíveno 11.01.2018].
URL https://www.maxcdn.com/one/visual-glossary/web-application/

[21] Q-Success: *Historical trends in the usage of server-side programming languages for websites*. W3Techs, 2018, [Online; navštíveno 20.03.2018].
URL
https://w3techs.com/technologies/history_overview/programming_language

[22] Seznam.cz, a.: *Mapy*. Mapy.cz, 2018, [Online; navštíveno 20.11.2017].
URL http://www.mapy.cz

[23] S.R.L., S. E. S.: *Moqups - Shape Your Ideas. Prove Your Concept.* 2018, [Online; navštíveno 20.03.2018].
URL https://moqups.com/

[24] jQuery Team, T.: *jQuery*. 2018, [Online; navštíveno 20.03.2018].
URL http://jquery.com/

[25] Travefy, i.: *Travefy*. 2018, [Online; navštíveno 20.11.2017].
URL https://travefy.com/

[26] Vrána, J.: *Adminer - Database management in a single PHP file*. 2018, [Online; navštíveno 20.03.2018].
URL https://www.adminer.org/

[27] W3C: *HTML*. 2018, [Online; navštíveno 20.03.2018].
URL https://www.w3.org/html/

[28] Wheeler, K.: *Slick - the last carousel you'll ever need*. 2013, [Online; navštíveno 20.03.2018].
URL http://kenwheeler.github.io/slick/

[29] Český statistický úřad: *Cestovní ruch - časové řady*. [Online; navštíveno 05.03.2018]. URL https://www.czso.cz/csu/czso/cru_cr

[30] Čápka, D.: *MVC architektura*. IT Network, 2018, [Online; navštíveno 14.01.2018]. URL https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor

Příloha A

Obsah CD

- **doc/** - Technická zpráva
  - **source/** - Zdrojové soubory technické zprávy pro LaTeX
  - **xfilak03_report.pdf** - Technická zpráva ve formátu PDF
- **src/** - Zdrojové soubory aplikace
- **manual.txt** - Návod na instalaci aplikace

Příloha B

Návod na instalaci aplikace

1. Zkopírujte soubory z CD umístěné v adresáři **src**.
2. Vytvořte soubor **.env** v kořenovém adresáři zkopírováním a přejmenováním souboru **.env.example**. Ve vytvořeném souboru vyplňte údaje potřebné pro komunikaci s databází a pro správné fungování obnovy zapomenutého hesla zadejte i údaje k emailovému serveru, který zajišťuje zasílání emailů pro obnovu hesla.
3. Nainstalujte všechny závislosti pomocí příkazu **composer install** v příkazovém řádku. Je nutné tento příkaz spouštět v kořenovém adresáři projektu.
4. Vygenerujte unikátní klíč aplikace pomocí příkazu **php artisan key:generate**.
5. Inicializujte databázi pomocí příkazu **php artisan db:seed**.
6. Aktualizujte kurzy měn pomocí příkazu **php artisan UpdateCurrencyValue:updatecurrency**.
7. Pro fungování *Plánovače úkolů* z Laravelu a tedy i automatickou aktualizaci měny z ČNB spusťte Cron job: *** * * * * php /path-to-your-project/artisan schedule:run >> /dev/null 2>&1**

Příloha C

Scénář cesty pro uživatelské testování

Níže se nachází scénář cesty, který byl předán uživatelům pro potřeby testování. U každé části scénaře jsou nejčastější připomínky uživatelů k dané části, nebo poznámky vycházející z našeho pozorování.

1. Přejdi na webovou stránku aplikace(http://itinerary.tomasfilak.cz/) a proveď registraci.
    - Všichni uživatelé zvládli registraci bez problémů
2. Máš v plánu výlet do Londýna od 5. června 2018 do 9. června 2018. Na tuto cestu máš vyhrazeno 10 000 Kč. Budeš si plánovat celkem dvě verze itineráře, proto si po vytvoření itineráře otevři jeho nastavení a vyber možnost pro dvě verze itineráře.
    - Chybí výběr počtu verzí rovnou ve formuláři pro vytvoření itineráře(nutnost po vytvoření otvírat nastavení itineráře)
3. V úterý přidej do 1. verze itineráře návštěvu výstavy "Madame Tussaunds". Na toto místo se dopravíš pomocí autobusu(30 minut), který tě bude stát 1,5 GBP a vstup do muzea stojí 25 GBP. Prohlídku máš objednanou od 9:00 do 11:30.
    - Nutnost výběru času myší (nelze napsat)
    - Chybí seřazení měn podle abecedy
    - Při vybrání lokace z našeptávaných pomocí enteru se odešle formulář
    - Líbilo se našeptávání lokací
4. Po návštěvě musea "Madame Tussaunds"se přesuneš pěšky do "Hyde Parku". Cesta pěšky sem potrvá jednu hodinu. V Hyde Parku budeš od 13:00 do 15:30. Vstup do tohoto parku je zdarma.
    - Přidal by tlačítko pro vytvoření nové lokace i pod timeline
    - Je matoucí, že tlačítko pro přidání nové lokace je nad předchozí lokací, když chce přidávat lokaci pod ni
5. Zjistil jsi, že cesta do muzea "Madame Tussaunds"bude o 1GBP dražší, než jsi zadal, protože pojedete přes dvě pásma. Oprav ji.

- Nejprve klikal na cestu, až poté klikl na kartu s lokací
- Chybí možnost přidání komentáře k lokaci

6. Nyní si pro Úterý vytvoř druhou verzi itineráře. Vlož do druhé verze návštěvu "Buckinghamského paláce". K paláci se dostaneš taxíkem za 1 hodinu a cena je 15 GBP. Prohlídka paláce stojí 20 GBP a bude trvat od 10:00 do 13:00. Poté se pěšky přesuneš (20minut) k Big Benu, kde strávíš hodinu. Naplánuj si ho ale raději až na 15:00, abys měl rezervu pro vyfocení nové fotky z ulic Londýna. Na Big Ben se podíváš jen z ulice, takže tě to nebude nic stát.
    - Automaticky nastavovat čas do po zadání času od, aby nevznikala hned kolize
7. Ve středu, už přidej jen dvě lokace do 1. verze itineráře. První je "London Bridge", dopravíš se sem autobusem(1 hodina, 1,5 GBP). Na tomto mostě budeš od 11:00 do 12:00. Poté se autobusem(20 minut, 1.5 GBP) přesuneš do "Sky Garden"kde máš rezervovaný vstup na 13:00. Cena je 5GBP a můžeš zde být dvě hodiny.
    - Upozornění na špatné zobrazení nadpisu při zobrazeni detailu lokace
8. Nyní se přesuň zpět do části s přehledem o celém itineráři a přidej fixní náklady. Jeden fixní náklad bude ubytování, které tě pro obě verze bude stát celkem 50 GBP. Druhým fixním nákladem bude příplatek za snídani celkem 20 GBP, který si ale zvolíš pouze pro první verzi itineráře.
    - Verze přesahují určené pole
    - Lze vložit i pro 3. verzi i když tvořím jen dvě verze
    - Tlačítka pro statistiku a fixní náklady by potřebovali popisek
9. Teď se podívej na porovnání těchto dvou verzí itineráře. Do komentáře zapiš nějaké zjištěné informace jako například celková cena za jednotlivé verze nebo kolik času strávíš přepravou mezi místy ve verzi 1. Dále zjisti kolik liber si budeš muset na tuto cestu vyměnit ve směnárně.
    - Chybí zobrazení o kolik je jedna verze levnější než ostatní
    - Zobrazení použitého kurzu měny pro přepočty
    - Požadované údaje dokázali z tabulek vyčíst bez problémů
10. V aplikaci zkus zapnout navigaci na nějakou z lokací.
    - Dlouho hledali tlačítko pro kontextovou nabídku, kde se navigace nachází

Další poznámky k celé aplikaci:

- Chybí možnost vytvoření jedné lokace pro více verzí
- Když itinerář neobsahuje žádnou cestu, mohla vy se vypsat místo prázdné plochy hláška
- Přidat možnost řazení v kurzovním lístku
- Přidat možnost propojení s Google Kalendářem
- Přidání zobrazení informace, kolik zbývá do vyčerpání rozpočtu
- Celkové hodnocení aplikace bylo ale jinak pozitivní a uživatelům se aplikace líbila