FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Systém pro Mistrovství ČR v autostopu
Student:	Bc. Karím Abu Nofal
Vedoucí:	Ing. Oldřich Malec
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce zimního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je kompletní tvorba webové aplikace pro správu obsahu webu charitativního závodu MČR v autostopu.

Postupujte dle následujících kroků:

1. Analyzujte případy užití současného řešení a zjistěte, jaké nové funkce by měl systém podporovat.
2. Na základě analýzy proveďte vhodný návrh.
3. Návrh zrealizujte v podobě funkčního prototypu.
4. Prototyp podrobte vhodnými Vámi navrženými testy.
5. Na základě testování prototyp upravte.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 28. června 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Systém pro Mistrovství ČR v autostopu

Bc. Karím Abu Nofal

Katedra Softwarového inženýrství
Vedoucí práce: Ing. Oldřich Malec

6. ledna 2022

Poděkování

Děkuji vedoucímu mé diplomové práce za jeho čas a podporu. Dále děkuji organizátorům závodu za příjemnou spolupráci a v neposlední řadě děkuji rodině a přátelům za podporu, kterou mě povzbuzovali po celou dobu studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 6. ledna 2022 . . . . . .

Odkaz na tuto práci

Abu Nofal, Karím. *Systém pro Mistrovství ČR v autostopu.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Tato práce se zabývá vývojem nové verze webové aplikace pro závod Mistrovství ČR v autostopu, který pořádá Světem stopem z. s. Text obsahuje seznámení se současnou verzí webových stránek, na jejichž základě stojí návrh nové aplikace. Hlavním cílem bylo analyzovat požadavky organizátorů, vybrat technologie potřebné k implementaci, navrhnout novou verzi a tu následně implementovat. Vývoj probíhala ve frameworku Django v programovacím jazyce Python. Výsledný prototyp byl nasazen do cloudu, aby mohl být podroben uživatelskému testování. To odhalilo drobné nedostatky, které byly posléze opraveny.

Klíčová slova webová aplikace, Django, Python, Mistrovství ČR v autostopu

Abstract

This work deals with the development of a new version of the web application for the Czech Championship race in hitchhiking, organized by Světem stopem z. s. The main goal was to analyze the requirements of the organizers, select the technologies needed for implementation, design a new version and then implement it. The development was carried out in the Django framework in the Python programming language. The resulting prototype was deployed in the cloud to be tested by users. This revealed minor flaws that were later corrected.

Keywords web application, Django, Python, Czech Hitchhiking Championship

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Cestování patří mezi oblíbené volnočasové aktivity mnoha lidí. Existuje široká škála způsobů cestování, od pěší turistiky až po zaoceánské plavby. Jedním z takových způsobů je autostop. Ten se díky svým nízkým nákladům a vidinou nepředvídatelných událostí stává favoritem u lidí vyhledávajících nové zážitky. Tato skutečnost dala vzniku několika úspěšným závodům po Evropě, které mají stopování jako hlavní disciplínu.

Jedním z těchto závodů je Mistrovství ČR v autostopu. Organizátoři však závodu chtěli dát nějaký hlubší smysl, a tak ke klasickému hodnocení v závodě (tj. kdo je nejblíž cíli, ten je první v pořadí) přidali element sponzorů. To znamená, že diváci sledující tento závod mohou své oblíbené posádce posílat peníze, díky kterým posádka obdrží plusové body do hodnocení. Peníze vybrané od všech posádek po ukončení závodu putují na pomoc předem vybrané osobě, na kterou pořádá sbírku Konto Bariéry.

Světem stopem z. s. jsou nespokojeni s aktuální verzí webových stránek závodu, a proto se rozhodli vypsat téma této diplomové práce. Některé úkony ohledně správy závodu by chtěli automatizovat a tím usnadnit práci jak sobě, tak účastníkům závodu. Například odevzdávání úkolů nemusí probíhat tak, že fotky a texty poslané organizátorům přes sociální sítě budou do systému zadávat organizátoři ručně. Cílem práce je tedy vytvořit webovou aplikaci, která bude lépe splňovat jejich požadavky.

Text této práce popisuje softwarový vývoj webové aplikace, včetně administračního prostředí. V první kapitole seznamuje čtenáře se současnou verzí webu. V další kapitole je analyzována architektura, technologie, požadavky a případy užití. Na základě těchto dat je v třetí kapitole navržena nová webová aplikace, jejíž implementace je popsaná v kapitole čtvrté. Následující dvě kapitoly se zabývají nasazením a uživatelským testováním. Poslední kapitola shrnuje výsledky práce a diskutuje o možnosti dalších rozšíření aplikace do budoucna.

Mistrovství ČR v autostopu

V této sekci je představen průběh závodu, jeho pravidla, cíle a vše potřebné pro průběh závodu. Informace v následujících sekcích znázorňují stávající verzi závodu a nejsou zde obsaženy změny, které budou přidány v rámci diplomové práce.

Popis závodu

Mistrovství ČR v autostopu je charitativní závod, ve kterém posádky stopují po Evropě. Každým rokem se vybere za spolupráce s veřejnou sbírkou Konto Bariéry osoba, na kterou se bude pomocí příspěvků od diváků vybírat. Více informací o Konto Bariéry je možné získat na jejich webových stránkách www.kontobariery.cz [3]

Do hlavního závodu je potřeba se kvalifikovat ve kvalifikačním závodě. Ten probíhá pár měsíců před hlavním závodem, trvá dva až tři dny a koná se na území České republiky. Průběh tohoto závodu nemá nic společného s webovými stránkami závodu, posádky nemají žádné sledovací zařízení a ani se nevybírají žádné peníze. Prvních 12 posádek se kvalifikuje do hlavního závodu.

V hlavním závodě musí posádky splnit několik předem stanovených úkolů. Některé tyto úkoly jsou povinné a jejich nesplnění vede k diskvalifikaci posádky. Při plnění úkolů posádky musí dodržovat pravidla, se kterými musí před startem závodu souhlasit. Porušení těchto pravidel také vede k diskvalifikaci ze závodu. Splnění úkolu dokládá posádka příběhem a fotkou či videem, které pošlou administrátorům závodu. Pokud je splnění úkolu administrátory schváleno, je posádce připočítán bod.

Body také mohou posádky získávat za pomoci sledujících, kteří mají možnost zasláním DMS posádce přispět. Veškeré peníze vybrané tímto způsobem pak putují do veřejné sbírky a pomohou tak osobě ve složité životní situaci.

Celkové pořadí tedy určuje počet bodů, které posádka za celý závod dokázala nasbírat. Pokud se na konci závodu stane, že mají dvě posádky stejný počet bodů, lepší umístění získá posádka, která do cíle dorazila jako první.

Fáze závodu

Z hlediska správy závodu je celková akce rozdělena do několika fází, které jsou reflektovány i na webových stránkách závodu.

V první fázi probíhá plánování nového ročníku závodu. Na webových stránkách je stále zobrazeno shrnutí posledního dokončeného závodu. Po naplánování trasy a určení termínu je založen nový závod a na webových stránkách se již zobrazují informace o tomto závodu. Zatím jsou jediné zveřejněné informace pravidla, datum a cílová destinace. Úkoly se zveřejňují přibližně týden před začátkem závodu, aby jejich obsah byl pro účastníky závodu překva-

pením a nemohli se na jejich plnění předem připravit. Během první fáze je také naplánován kvalifikační závod, do kterého se mohou zájemci registrovat. Po skončení kvalifikačního závodu je prvních 12 posádek kvalifikováno do hlavního závodu. V posledním týdnu první fáze se vytvoří a zveřejní úkoly a domluví se s Konto Bariéry, na kterou osobu se bude tento rok v rámci závodu vybírat. Těsně před zahájením závodu administrátoři obdrží sledovací zařízení od společnosti GPS Dozor, které jsou následně přiřazeny posádkám.

Druhá fáze je závod jako takový. Po odstartování závodu zúčastněné posádky stopují po Evropě a plní stanovené úkoly. Aktivita posádek je dokumentována pomocí zážitků, které přidávají na webové stránky. Pro návštěvníky webu je také připravena mapa s polohou posádek, aby si mohli udělat lepší představu, kde se momentálně posádky nacházejí. Jednou z hlavních součástí této fáze je přispívání posádkám. Pomocí přispívání mohou sledující pomoci posádce k lepšímu umístění a také tím přispějí do veřejné sbírky Konta Bariéry.

Po ukončení závodu probíhá vyhlášení výsledků, předávání cen a na webových stránkách je zobrazen článek se shrnutím daného ročníku závodu. Další rok se poté pokračuje opět první fází.

Kapitola 1

Stávající řešení

V této kapitole je rozebráno současné řešení webových stránek Mistrovství ČR v autostopu (www.mcrautostop.cz).[4] Podle organizátorů jsou nynější stránky zastaralé, uživatelsky nepřívětivé, což dalo vzniknout této diplomové práci. Nedostatky současných webových stránek jsou shrnuty v podsekcích popisujících jejich části. Analýza současného řešení je kvalitním zdrojem informací o tom, na co je potřeba se při vývoji nové verze zaměřit, čeho se vyvarovat a jaké nové funkcionality by bylo dobré přidat.

1.1 Klientská část

Nynější verze webových stránek je napsána ve skriptovacím jazyce PHP. Při implementaci nebyl kladen velký důraz na návrh, který by umožňoval budoucí rozšíření aplikace. Absence použití jakéhokoli webového frameworku tohoto jazyka činí údržbu webu značně složitou. Pro ukládání dat aplikace využívá databázový server MariaDB a je nasazená na serverech společnosti WEDOS.

1.1.1 Sledování posádek

Jedním z hlavních požadavků na stránky je sledování poloh posádek a vzdálenosti, kteou od začátku závodu stihly urazit. Služby zajišťující tyto informace zprostředkovává hlavní partner závodu GPS Dozor. Každá posádka s sebou vozí GPS zařízení, které o sobě každých třicet vteřin odesílá informace na servery GPS Dozoru. Z těchto serverů je následně možné pomocí vystaveného API informace získat a zobrazit. API, které využívá současný web, je však staré a společností GPS Dozor již není podporované. Je třeba aktualizovat aplikaci, aby využila jejich nově vystaveného API.

1.1.2 Platby

První ze dvou kritérií při vyhodnocení závodu je množství peněz, které dané posádce poslali fanoušci. Momentálně lze posílat peníze pouze formou DMS. Každá posádka má přidělené jedno telefonní číslo, na kterém se akumulují pro ně poslané peníze. Dárcovské SMS je však možné posílat pouze na území České republiky, což značně omezuje množinu lidí, kteří by potenciálně mohli přispět. Detailnější informace o DMS jsou popsány v sekci 4.6.

1.1.3 Organizace posádek

Druhé kritérium hodnocení je počet úkolů, které posádka stihla do ukončení závodu splnit. Aby úkol mohla dokončit, musí organizátorům, odeslat fotografii nebo videozáznam z jeho konání společně s textem, který popisuje průběh plnění. Správa úkolů však není implementována v klientské části, takže závodníci nemohou přidávat své příspěvky skrze webové stránky. Místo toho komunikují s organizátory přes sociální sítě nebo formou SMS/MMS. Tímto způsobem odešlou splnění úkolu organizátorům, kteří ho následně musí ručně zadat v administrační části aplikace do systému.

1.1.4 Lokalizace

Během závodu účastníci cestují po celé Evropě. Na cestách se setkávají s mnoha lidmi, kterým vysvětlují, čeho se právě účastní. Tito lidé by třeba chtěli sledovat, jak se dané posádce v závodu daří, popřípadě ji i podpořit skrze DMS, ale nemůžou. Současná aplikace nepodporuje vícejazyčnost, která je v dnešní době pro webové aplikace standardem.

1.1.5 Vzhled aplikace

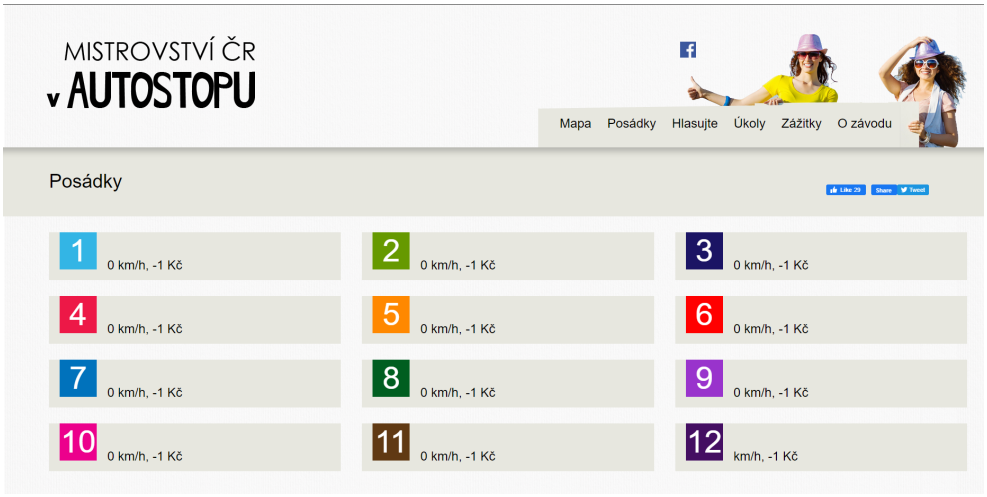
V současném řešení nebyla použita pro úpravu vzhledu základních HTML komponent žádná šablona, ani žádná CSS knihovna, jako např. Bootstrap. Vzhled je řešen sadou vlastních CSS stylů, které v porovnání s moderními webovými stránkami působí zastarale.

Jelikož aplikace neprošla žádným uživatelským testováním, rozvržení stránek a následná orientace v nich není vždy intuitivní a pro návštěvníky webu může být matoucí.

1.2 Administrativní část

Administrační část současné aplikace je CMS[1], který si zakladatel závodu nechal zhotovit pro svou bývalou firmu, a zde je použita jeho upravená verze.

[1] CMS, anglicky Content Management System - rozhraní, které umožňuje uživateli publikovat obsah přímo na web.[5]



Obrázek 1.1: Snímek obrazovky webových stránek současného řešení

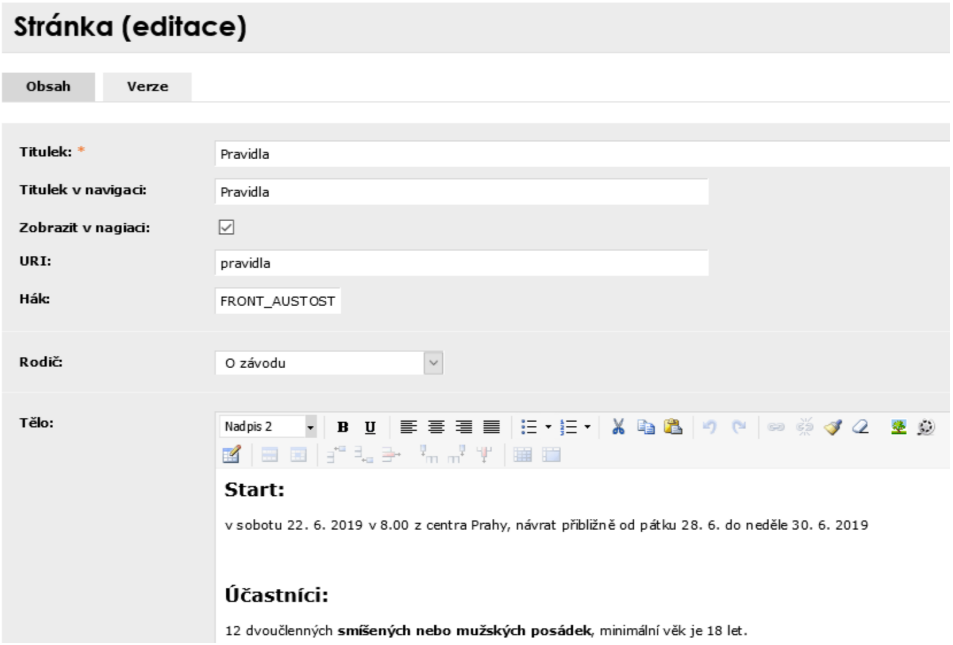
Pomocí tohoto systému se generují šablony, které se následně používají pro zobrazování stránek v klientské části. Toto řešení připadá současným organizátorům velice nepřehledné a zmatečné. Například pro založení nového závodu se musí založit kompletně nová stránka, místo toho, aby se založil jen záznam v databázi a použila se jednotná šablona. Proto jsou webové stránky jednotlivých závodů odlišné a tato nekonzistentnost může uživatele zmást. Organizátoři by raději měli jednotnou šablonu a spravovali by pouze zobrazovaná data, namísto vytváření celých stránek jako takových.

1.3 Shrnutí nedostatků aplikace

V poslední části této kapitoly je popsáno několik základních nedostatků současné aplikace, které bude cílem eliminovat v nově navrhované verzi.

- **Vzhled aplikace** - zastaralý vzhled a nekonzistentní styly HTML komponent,
- **Platby** - posílání příspěvků posádkám je možné pouze na území České republiky,
- **Posádky** - registrace na závod, odevzdávání úkolů a psaní vlastních zážitků je zajištěno pomocí externích komunikačních kanálů, nikoli pomocí webových stránek,
- **Administrativní část** - pro organizátory nepřehledná a složitá,
- **GPS Dozor API** - použití starého, již nepodporovaného API,

- **Lokalizace** - webové stránky jsou pouze v českém jazyce.



Obrázek 1.2: Snímek obrazovky administrační části současného řešení

Kapitola 2

Analýza

2.1 Funkční požadavky

Funkční požadavky jsou vlastnosti produktu nebo funkce, které musí vývojář implementovat, aby uživatelům umožnil splnit jejich úkoly.[6]

FP1: Správa účtu Uživateli bude umožněno registrovat svoji posádku do kvalifikačního závodu. Pokud ta postoupí do hlavního závodu, bude jí vytvořen uživatelský účet, přes který se budou moci oba členové posádky přihlásit do aplikace.

- Registrace do kvalifikačního závodu,
- Přihlásit se,
- Odhlásit se,
- Změnit heslo,
- Obnovit heslo.

FP2: Zobrazení výsledků Návštěvníci webových stránek budou mít možnost zobrazit si aktuální výsledky a statistiky závodu. Kromě celkového pořadí budou moci sledovat i výsledky jednotlivých částí hodnocení. Poloha posádek bude zobrazena na mapě.

- Zobrazit celkové pořadí,
- Zobrazit pořadí podle počtu splněných úkolů,
- Zobrazit pořadí podle vzdálenosti od cíle,
- Zobrazit pořadí podle vybraných peněz,

- Zobrazit posádku na mapě,
- Zobrazit detail posádky se všemi kritérii hodnocení,
- Zobrazit celkové množství vybraných peněz.

FP3: Přehled posádek Aplikace bude umožňovat procházení posádek aktuálního závodu.

- Zobrazit seznam posádek,
- Zobrazit detail posádky.

FP4: Přehled úkolů V aplikaci bude možné procházet úkoly právě probíhajícího závodu. Přihlášený člen posádky bude moci odevzdat vybraný úkol. Odevzdaný úkol se stane viditelným pro veřejnost a bude započítaný do statistik po schválení jedním z administrátorů závodu.

- Zobrazit seznam úkolů,
- Zobrazit detail úkolu,
- Odevzdat úkol.

FP5: Přehled zážitků Posádka může během závodu přidávat své zážitky, které budou k přečtení pro návštěvníky webových stránek. Přidaný zážitek musí být nejdříve schválen jedním z administrátorů závodu. Zážitky se budou moci filtrovat podle posádky.

- Zobrazit seznam zážitků,
- Zobrazit detail zážitku,
- Vytvořit zážitek.

FP6: Informace o závodu Návštěvníkům webových stránek bude umožněno přečíst si veřejné informace o závodu, jeho pravidla a často kladené otázky. Místa, která posádky musejí navštívit, budou pro lepší představu trasy zobrazena jako body na mapě s jejich popisem.

FP7: Články o charitativní činnosti závodu Každý závod bude mít jeden článek před a po dobu konání závodu a jeden článek po ukončení. První s informacemi o tom, na co se daný ročník vybírají peníze, a druhý jako shrnutí daného ročníku.

FP8: Historie závodu Návštěvníkům bude představena historie závodu s možností procházení starších ročníků závodu, kde si mohou přečíst shrnutí daného ročníku a zobrazit zážitky posádek.

FP9: Přispívání posádce Návštěvníci stránek budou moci přispívat jimi vybrané posádce. Budou mít na výběr z několika způsobů placení.

- DMS,
- PayPal,
- Platební karta.

FP10: Administrace webu Aplikace bude umožňovat přihlásit se jako administrátor do administrační části aplikace. V této části aplikace bude možné spravovat závod od jeho založení po jeho ukončení. Administrátoři budou mít pravomoc data upravovat, mazat či přidávat.

FP11: Lokalizace klientské části aplikace Návštěvníci webových stránek budou mít možnost vybrat si jazyk, ve kterém se budou stránky zobrazovat.

FP12: Sociální sítě Aplikace bude umožňovat otevírání nových oken se sociálními sítěmi Mistrovství ČR v autostopu.

2.2 Nefunkční požadavky

Nefunkční požadavky popisují, jak se systém musí chovat, a stanovují omezení jeho funkčnosti.

NP1: Webová aplikace Aplikace bude webová a dostupná přes webový prohlížeč.

NP2: Podpora webových prohlížečů Aplikace bude podporovat níže zmíněné webové prohlížeče, a to jak na počítači, tak na mobilních zařízeních.

- Chrome,
- Firefox,
- Microsoft Edge.

NP3: Responzivita Klientská část aplikace bude responzivní, a tím dobře zobrazitelná i na mobilních zařízeních.

NP4: Vícejazyčnost Klientská část bude implementována v českém a anglickém jazyce s možností přidání dalších jazyků.

NP5: Rozšiřitelnost Aplikace bude napsána tak, aby bylo možné přidávat rozšíření o další funkcionality bez většího zásahu do již implementovaných.

2.3 Případy užití

Případy užití vznikly na základě funkčních požadavků na aplikaci. Diagramy případů užití byly kvůli přehlednosti rozděleny podle jednotlivých aktérů. Některé z nich byly rozepsány do kroků, které je potřeba provézt pro docílení chtěné funkcionality.

2.3.1 Aktéři

Aplikaci mohou využívat tři typy uživatelů:

- **Nepřihlášený uživatel**
  Nepřihlášený uživatel může využívat většinu funkcí webových stránek (viz obrázek 2.1).
- **Přihlášený uživatel**
  Přihlášený uživatel má kromě možností nepřihlášeného uživatele k dispozici funkce pro správu vlastní posádky.
  (viz obrázek 2.2).
- **Administrátor**
  Administrátor má jako jediný přístup do administrační části aplikace, kde má zpřístupněné veškeré funkce pro správu závodu.
  (viz obrázek 2.3).

2.3.2 Případy užití nepřihlášeného uživatele

2.3.2.1 PU1: Zobrazit výsledky

1. Případ užití začíná na úvodní obrazovce, kde je zobrazena mapa Evropy a sloupeček s celkovými statistikami.
2. Uživatel si zvolí, podle kterého kritéria chce zobrazit výsledky.
3. Aplikace zobrazí tabulku posádek s jejich pořadím.
4. Uživatel klikne na řádek s posádkou.
5. Aplikace zobrazí detailní informace o výsledcích zvolené posádky a zvýrazní její polohu na mapě.

2.3.2.2 PU10: Přispět posádce

1. Případ užití začíná, když se uživatel rozhodne podpořit posádku.
2. Uživatel spustí akci chci přispět.
3. Aplikace zobrazí seznam posádek, na které může přispět.
4. Uživatel vybere posádku.
5. Aplikace zobrazí možnosti platebních metod s popisem jejich použití.
6. Uživatel vybere metodu a, je-li to v rámci platební metody potřeba, uživateli se zobrazí kolonka pro zadání částky.
7. Uživatel dokončí platbu na základě typu platby, kterou vybral.
8. Pokud to platební metoda umožňuje, aplikace zobrazí informaci o úspěšně provedené platbě.

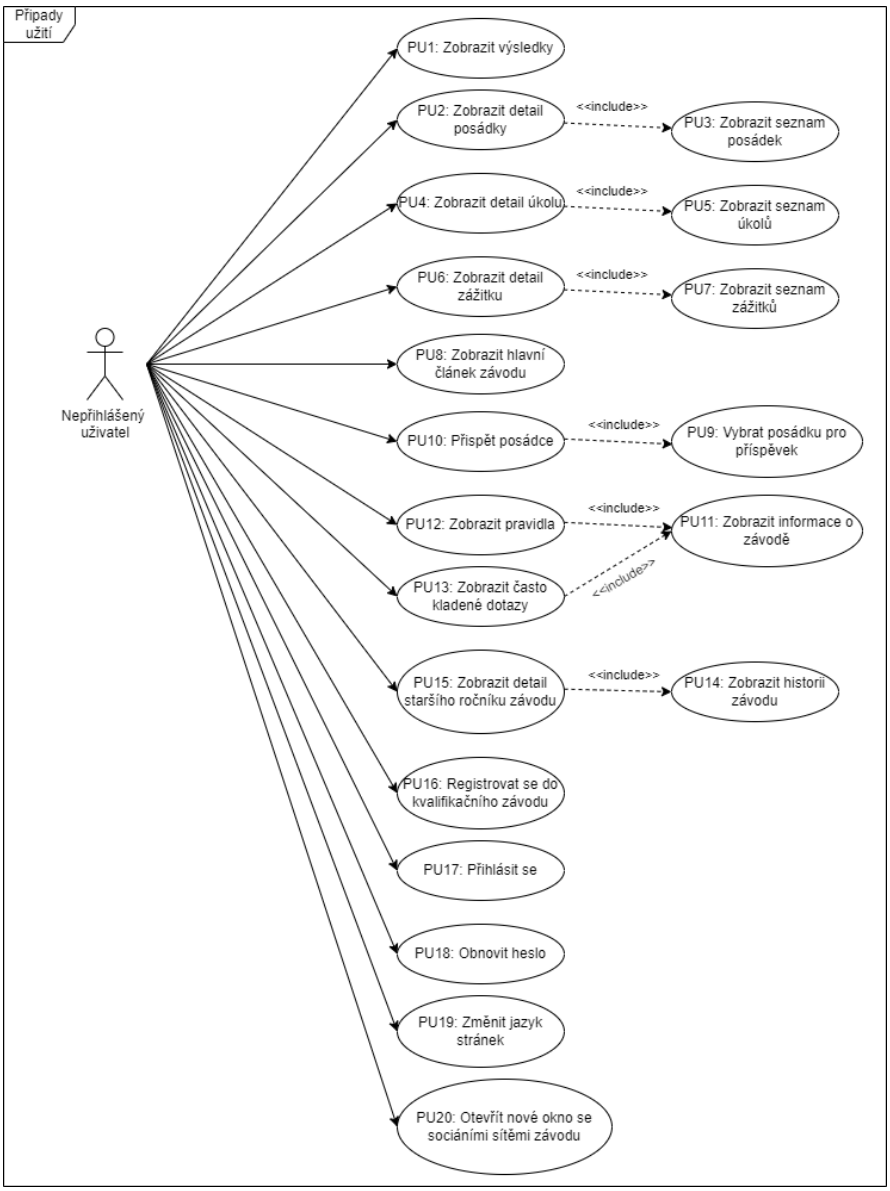
2.3.2.3 PU15: Registrace do kvalifikačního závodu

1. Případ užití začíná za předpokladu, že je aktivní kvalifikační závod a uživatel se chce registrovat.
2. Uživatel zvolí akci registrovat se.
3. Aplikace postupně zobrazí tři formuláře: První člen posádky, Druhý člen posádky a Informace o posádce.
4. Pokud byly zadané informace validní, aplikace informuje uživatele o dokončení registrace a automaticky odešle e–maily s dalšími informacemi.

2.3.2.4 PU16: Obnovit heslo

1. Případ užití začíná, když uživatel potřebuje změnit své přihlašovací heslo, ale zároveň není přihlášen do aplikace.
2. Na stránce s přihlašovacím formulářem uživatel spustí akci obnovit heslo.
3. Aplikace zobrazí formulář pro obnovení hesla.
4. Uživatel vyplní formulář.
5. Pokud byl formulář vyplněný správně, odešlou se na e–maily obou členů posádky URL adresy pro obnovení hesla. V opačném případě aplikace informuje uživatele o chybně zadaných hodnotách.
6. Po otevření zaslaného URL je uživatel přesměrován na formulář pro změnu hesla.

7. Pokud uživatel vyplní formulář správně, tak je přesměrován na přihlašovací obrazovku a je informován o dokončení změny hesla. V opačném případě je aplikací informován o chybně zadaných hodnotách.



Obrázek 2.1: Diagram případů užití pro nepřihlášeného uživatele

2.3.3 Případy užití přihlášeného uživatele

2.3.3.1 PU20: Změnit heslo

1. Případ užití začíná, když je uživatel přihlášen a chce si změnit heslo.
2. Uživatel zvolí akci pro změnu hesla.
3. Aplikace zobrazí formulář pro změnu hesla. V tomto případě je vyžadováno stávající heslo.
4. Pokud uživatel vyplní formulář správně, tak je přesměrován na stránku nastavení posádky a je informován o úspěšném dokončení změny hesla. V opačném případě je informován o chybně zadaných informacích.

2.3.3.2 PU23: Odevzdat úkol

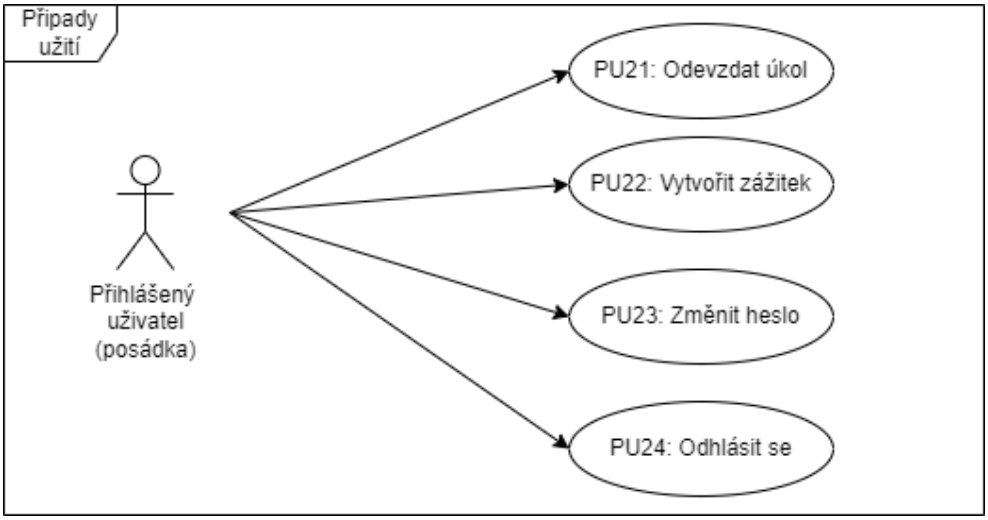
1. Případ užití začíná, když posádka splnila úkol a chce ho odevzdat. Přitom se musí nacházet na stránce s detailem daného úkolu.
2. Uživatel spustí akci pro odevzdání úkolu.
3. Pokud uživatel daný úkol může odevzdat, aplikace zobrazí formulář pro odevzdání úkolu. V opačném případě aplikace informuje uživatele, z jakého důvodu momentálně nemůže daný úkol odevzdat.
4. Uživatel vyplní formulář, vybere fotografii a ořízne ji na fixní poměr stran. Následně formulář odešle ke schválení.
5. Pokud byl formulář vyplněn správně, aplikace odešle organizátorům informaci o tom, že vznikl nový požadavek na schválení úkolu, a informuje uživatele, že jeho žádost byla odeslána. V opačném případě aplikace informuje uživatele o chybně vyplněném formuláři.

2.3.4 Případy užití administrátora

V administrativní sekci má administrátor k dispozici veškeré funkcionality ke správě závodu. Veškeré datové modely, které jsou k tomu potřeba, může libovolně upravovat, vytvářet a mazat.

2.3.4.1 PU25 - PU34

Tyto případy užití znázorňují správu jednotlivých entit pro správu závodu. U každé z nich se zobrazí administrátorovi všechny dostupné záznamy, které může mazat, vytvářet nové, nebo si zobrazit detail každého záznamu. Kromě těchto základních operací, které mají všechny entity společné má administrátor



Obrázek 2.2: Diagram případů užití přihlášeného uživatele

možnost u některých entit používat dodatečné akce, které jsou v diagramu znázorněny zvlášť, např. PU:39.

Pro ilustraci je v krocích popsán případ změny informací u posádky.

1. Případ užití začíná v administrativní části aplikace, když administrátor chce změnit informace o posádce.

2. Administrátor zobrazí sekci s posádkami.

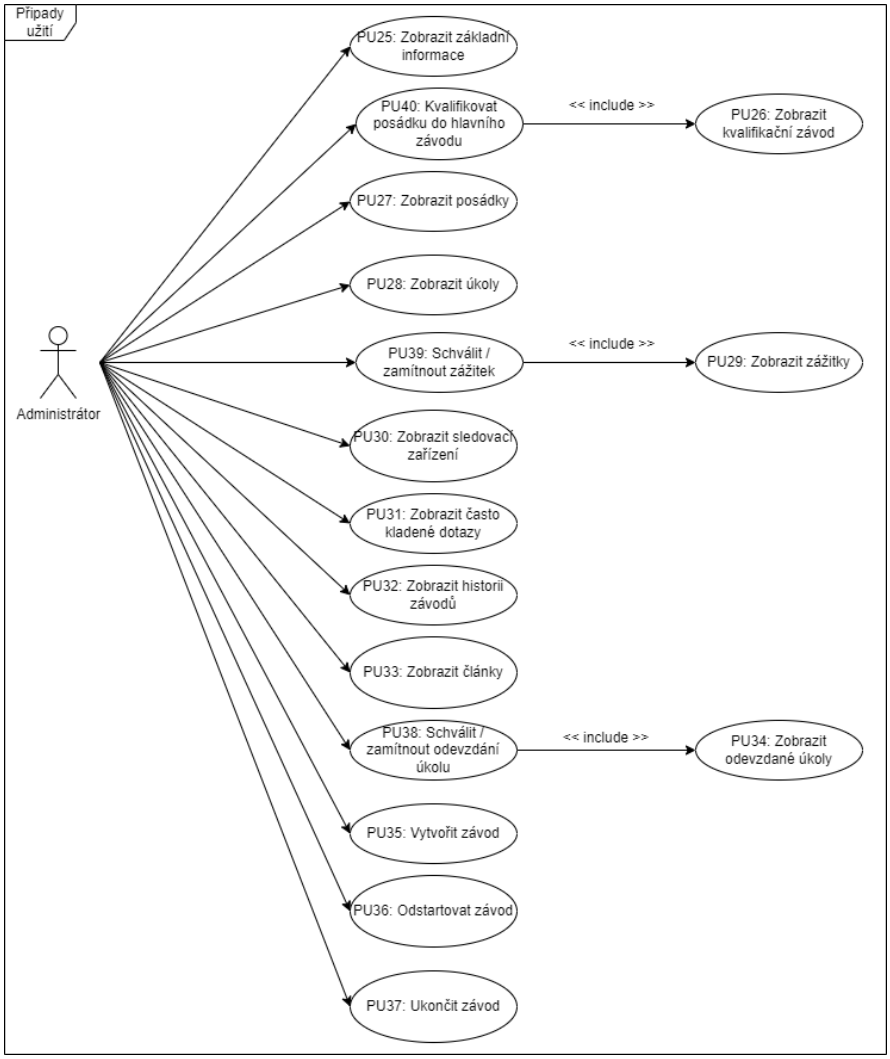
3. Aplikace zobrazí seznam všech posádek.

4. Administrátor najde požadovanou posádku a zvolí akci upravit.

5. Aplikace zobrazí detail posádky jako formulář.

6. Po upravení dat ve formuláři administrátor odešle žádost o uložení změn.

7. Pokud byla zadaná data v pořádku, aplikace informuje administrátora o dokončení změn a přesměruje ho na seznam posádek. V opačném případě aplikace administrátora upozorní na špatně vyplněná pole formuláře.



Obrázek 2.3: Diagram případu užití administrátora

2.4 Kontrola pokrytí požadavků

Níže zobrazená tabulka 2.1 znázorňuje naplnění funkčních požadavků případy užití. Vzhledem k tomu, že se v každém sloupci vyskytuje alespoň jedna značka splnění, je evidentní, že všechny funkční požadavky byly namodelovanými případy užití pokryty. Tato skutečnost ukazuje, že aplikace bude splňovat veškeré funkcionality, které se po aplikaci požadují.

	FP1	FP2	FP3	FP4	FP5	FP6	FP7	FP8	FP9	FP10	FP11	FP12
PU1		✓										
PU2			✓									
PU3			✓									
PU4				✓								
PU5				✓								
PU6					✓							
PU7					✓							
PU8							✓					
PU9									✓			
PU10									✓			
PU11						✓						
PU12						✓						
PU13						✓						
PU14								✓				
PU15								✓				
PU16	✓											
PU17	✓											
PU18	✓											
PU19											✓	
PU20												✓
PU21				✓								
PU22					✓							
PU23	✓											
PU24	✓											
PU25-40										✓		

Tabulka 2.1: Tabulka splnění funkčních požadavků

2.5 Výběr technologií

Před samotným návrhem aplikace je nejprve potřeba vybrat technologie, pomocí kterých bude aplikace implementována, nebo kterých bude využívat. V této sekci je popsáno, jaké technologie byly brány v potaz a které se nakonec použijí pro samotnou implementaci.

Výběr technologií je rozdělen do pěti částí:

- Serverová část aplikace,
- Klientská část aplikace,
- Databáze,
- Služby zprostředkovávající mapy a navigaci,
- Nasazení.

2.5.1 Serverová část

V této části bylo nutné nejdříve vybrat, v jakém programovacím jazyce bude aplikace napsána. Pro rychlejší vývoj webové aplikace je vhodné vybrat některý z webových frameworků, které celý proces značně zjednodušují. Proto byl jazyk vybírán zprvu podle toho, jaký pro něj existuje webový framework. Mezi dva hlavní kandidáty patřily programovací jazyky *C#* a *Python*, a to na základě *2020 Developers Survey*[7] společně s autorovou znalostí těchto jazyků. Po diskuzi s organizátory závodu a jejich spolupracovníky, kteří aplikaci po produkčním vydání budou spravovat, byl vybrán jazyk Python, neboť s ním mají větší zkušenosti.

Python je interpretovaný, objektově orientovaný programovací jazyk s dynamickou sémantikou. Je vhodný jak pro rychlý vývoj aplikací, tak pro skriptování. Má jednoduchou a snadno naučitelnou syntaxi, díky které se kód snadno čte a tím i lépe udržuje. Jelikož je to interpretovaný jazyk, odpadá zde krok kompilace, a proto je cyklus upravit-otestovat-vyladit velice rychlý.[8]

Po zvolení programovacího jazyka bylo na řadě vybrat webový framework, ve kterém bude aplikace implementována. Na základě *Python Developers Survey*[9] jsou zdaleka nejpopulárnějšími frameworky pro vývoj webových aplikací v Pythonu *Flask a Django.*

2.5.1.1 Flask

Flask je nenáročný webový framework WSGI aplikací[2]. Vznikl jako nadstavba nad Jinja2 (šablonový framework) a Werkzeug (serverový framework). Ze základu ho tvoří pouze funkce pro práci s URL, HTTP a šablonami.[11] Vše ostatní programátor musí dodat ve formě rozšíření, které si sám vybere. Například, pokud chce programátor do projektu zahrnout práci s databází, musí sám vybrat nějaký ORM framework, např. FlaskSQLAlchemy. To poskytuje širokou škálu možností, ale také velký prostor pro chyby a nekompatibilitu mezi potřebnými rozšířeními. Vzhledem k jeho jednoduchosti v něm lze velice

[2]WSGI - Web Server Gateway Interface. WSGI aplikace implementuje serverovou část WSGI rozhraní pro nasazení webových aplikací napsaných v Pythonu[10]

rychle vyvinout webovou aplikaci. Nicméně díky absenci jakékoli dané struktury může být pro nově příchozí vývojáře obtížné se v aplikaci zorientovat.

Výhody

- Rychlý vývoj aplikace,
- Absence robustní, předem dané struktury,
- Kompletní kontrola nad použitými rozšířeními.

Nevýhody

- Potencionální nekompatibilita potřebných rozšíření,
- U větších projektů horší přehlednost aplikace,
- Je nutná dobrá znalost velkého množství rozšíření pro docílení běžně potřebných funkcionalit webových aplikací.

2.5.1.2 Django

Na rozdíl od Flasku je Django velice robustní webový framework. Součástí základního balíčku jsou bohaté databázové API, dobře strukturované adresy URL, propracované administrační prostředí pro správu dat, vícejazyčnost a mnoho dalšího. Díky tomu je možné v Djangu vyvíjet aplikace velice rychlým tempem, a jelikož je programátor nucen držet se stanovených struktur, výsledné aplikace dodržují principy dobrého návrhu. Django se snaží co nejvíce funkcionalit automatizovat a dodržovat princip DRY[3]. Django je mocný nástroj, který programátorovi umožňuje plně se soustředit na funkcionality vyvíjené aplikace a starosti s technickými záležitostmi přenechat na frameworku samotném.[12]

Výhody

- Velká část funkcionalit pro tvorbu webových aplikací je již ve frameworku implementována,
- Kvalitní administrační prostředí,
- Skvělá dokumentace a velice rozsáhlá aktivní komunita,

[3]DRY - Don't Repeat Yourself - neopakuj se

- Vestavěné middleware knihovny, které se starají o běžné bezpečnostní hrozby, např. SQL injection[4].

Nevýhody

- Funkcionality, které nejsou plně podle principů frameworku Django, se implementují s velkými obtížemi,
- Potencionálně příliš mnoho nevyužitých souborů jako součást základního projektu.

2.5.1.3 Výběr frameworku

Po dlouhém zvažování byl vybrán framework Django. Jelikož autor práce neměl zkušenosti ani s jedním, dokumentace a rozsáhlá aktivní komunita byly hlavní faktory při rozhodování. Dalším důvodem byl fakt, že implementovaná aplikace využije velkou část základních funkcí frameworku, což značně usnadní a urychlí vývoj. V neposlední řadě díky struktuře a přehlednosti projektu nebude v budoucnu problém s ním seznámit správce závodu, kteří výslednou webovou aplikaci budou udržovat.

2.5.2 Klientská část

Pro tvorbu webových stránek budou použity technologie HTML, CSS a JavaScript. Administrační prostředí bude využívat CSS knihovnu Bootstrap. Zbytek webových stránek bude obsahovat vlastní styly, aby bylo docíleno unikátního vzhledu stránek. Pro pohodlnější a efektivnější práci s CSS soubory se používají CSS preprocesory, mezi které patří např. LESS a SASS. V této práci byl vybrán preprocesor SASS z důvodu zkušeností autora práce s danou technologií.

2.5.3 Databáze

Při výběru databáze byl brán v potaz seznam databází podporovaných webovým frameworkem Django. Mezi ty patří[14]:

- PostgreSQL,
- MariaDB,
- MySQL,
- Oracle,

[4]SQL injection - technika napadeni databázové vrstvy systému vložením vlastního SQL dotazu přes neošetřený vstup, pomocí kterého útočník může získat informace z databáze, které nebyly určeny ke zveřejnění [13]

- SQLite.

Jelikož aplikace komunikuje s databázovou vrstvou skrze ORM, tak pro implementaci není výběr databáze podstatný, pokud se jedná o jednu z výše uvedených. Django umožňuje vyměnit tyto databáze bez většího zásahu do zdrojových kódů. Nakonec tedy byla z autorovy osobní preference vybrána databáze PostgreSQL.

PostgreSQL je objektově relační databázový systém, který využívá a rozšiřuje jazyk SQL. Tento systém je v aktivním vývoji již přes 30 let a je plně kompatibilní s vlastnostmi ACID od roku 2001. Běží nativně na všech rozšířených operačních systémech, jako je Linux, systémy UNIX a Windows. Kromě standardních SQL datových typů obsahuje i ty moderní, jako např. JSON nebo XML [15].

2.5.4 Služby zprostředkovávající mapy a navigaci

Z funkčních požadavků plyne, že musí být na mapě znázorněné polohy posádek a statistiky o nich, které se týkají ujetých vzdáleností a vzdáleností posádky od cíle. Pro splnění tohoto požadavku bylo nutné využít nějaké aplikace třetí strany, protože implementace takovéto služby je velice složitá, časově náročná a nad rámec této diplomové práce. Vzhledem k tomu, že závod pořádá nezisková organizace, byly při vybírání služby hlavním kritériem co nejmenší náklady při splnění požadavků.

2.5.4.1 Komerční řešení

Existuje mnoho komerčních společností, které potřebné služby poskytují. Jednou z největších takových společností je Google, která nabízí jak statické mapy, které se zobrazují jako obrázky, tak dynamické mapy, které jsou interaktivní a přizpůsobitelné. Z jejich nabídky směrování našim požadavkům vyhovuje pouze matice vzdáleností, která pro zadanou sekvenci světových souřadnic vrátí vzdálenosti a časy tras kombinací souřadnic v reálném provozu.

Společnost Google poskytuje uživatelům každý měsíc 200 $, za které mohou jejich služby využívat. Ceny za tyto služby v době psaní tohoto textu jsou zobrazeny v tabulce 2.2 [16].

Služba	**Částka za 1000 požadavků**
Statické mapy	5 $
Dynamické mapy	7 $
Matice vzdáleností	5 $

Tabulka 2.2: Tabulka cen služeb společnosti Google

Dalším zkoumaným řešením byly mapy od společnosti Seznam. Tyto mapy jsou zdarma i pro komerční použití, avšak mají striktní pravidla užívání. Po te-

lekomunikaci s jejich technickým oddělením byla tato možnost zavržena z důvodu možného porušení jejich smluvních podmínek pro používání produktu.

2.5.4.2 Open source řešení

Open source řešení existuje ve dvou variantách. Buď je možné využít některého z veřejně dostupných API, nebo použít jejich zdrojové kódy pro vlastní nasazení dané služby. Výhodou tohoto řešení jsou nulové náklady a kromě limitovaného počtu požadavků, který je možné posílat na veřejně dostupné API, není toto řešení ničím omezeno.

2.5.4.3 Výběr

Kvůli absenci statistik ohledně návštěvnosti současných webových stránek nebylo možné odhadnout, jaké náklady by obnášelo použití služeb společnosti Google, jelikož celková cena závisí i na počtu zobrazení map na stránkách. Proto byla tato možnost zavržena.

Pro tuto práci bylo nakonec vybráno nasazení vlastního směrovacího stroje, konkrétně *Open Source Routing Machine*. Jelikož autorem nasazený směrovací stroj nebude veřejně dostupný, bude obsluhovat pouze požadavky z autorem nasazené webové aplikace. Díky tomu nebude provoz této služby natolik náročný, aby nemohla být nasazena na stejný server jako samotná webová aplikace. To má za důsledek nulové náklady na provoz požadovaných služeb. Také díky tomuto řešení nebude nijak omezeno rozšiřování nebo upravování závodu, jako by tomu mohlo být v případě komerčních řešení a veřejně dostupných API.

2.5.5 Nasazení

Aby výsledná aplikace fungovala, musí být nasazeny 3 části: Databáze, Webová aplikace a OSRM směrovací stroj. V této práci byl využit nástroj Docker, který každou část obalí do jednoho kontejneru. Ty poté budou nasazeny do cloudu a kontejner obsahující webovou aplikaci bude zpřístupněn veřejnosti.

Pro nasazení výsledné aplikace do cloudu byla zvolena platforma AWS[5] od společnosti Amazon. Tato platforma poskytuje bezplatnou infrastrukturu v míře, která je dostačující pro tuto práci. Více informací o službách, které byly v této práci využity, je uvedeno v sekci 5.3.

Django obsahuje jednoduchý webový server, který slouží čistě pro lokální vývoj, a tak bylo potřeba vybrat některý z WSGI webových serverů. Pro tuto práci byl vybrán server Gunicorn. Poskytování statických souborů je ve frameworku Django pomalé. Aby se tomuto omezení předešlo, používá se pro poskytování statických souborů samostatně běžící webový server. Na základě doporučení v Django dokumentaci byl vybrán webový server Nginx [17].

[5] Amazon Web Services

Kapitola 3

Návrh

3.1 Architektura aplikace

3.1.1 Vícestránková aplikace (MPA)

Architektura vícestránkové aplikace je považována za tradiční způsob, jak vyvíjet webové aplikace. Ty jsou složeny z více stránek, které je při každém požadavku od uživatele potřeba celé obnovit. Uživatelův požadavek je zpracován v serverové části, kde se následně vyrenderuje celá HTML stránka a odešle se na klientskou část. Některé stránky mohou potencionálně obsahovat velké množství dat, což může vézt k pomalému načítání. Opakované načítání takové stránky způsobuje špatný uživatelský zážitek. Proto je někdy potřeba načítat data na stránku postupně, nebo obnovit pouze její část. Pro tento účel existují AJAX požadavky. Ty ze serveru získají data, která se následně vloží do stránek pomocí jazyku Javascrip [18].

Vícestránková aplikace má velice úzce spjatou klientskou a serverovou část. Většina logiky aplikace je napsána v serverové části a klientská část slouží především pro prezentaci dat a zachycování vstupů od uživatele.

Výhody

- Dobrá navigace v rámci aplikace,
- Nižší riziko bezpečnostních hrozeb,
- Rozšiřitelnost - snadné přidání nového obsahu,
- Jednoduchá implementace SEO[6],

[6]SEO, anglicky Search Engine Optimization - optimalizace pro vyhledávače zajišťují, aby se stránka zobrazovala na předních místech ve výsledcích vyhledávání.

- Možnost využití analyzátorů dat, které poskytují data o fungování systému, chování uživatelů a další užitečné informace.

Nevýhody
- Úzce spjatá klientská a serverová část,
- Je nutné pro každou stránku psát vlastní HTML kód.

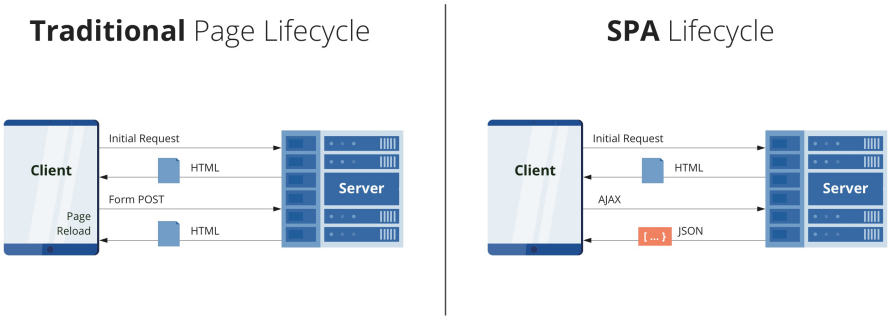
3.1.2 Jednostránková aplikace (SPA)

Jak už název napovídá, jedná se o samostatnou stránku, která při prvním zobrazení načte do lokálního úložiště veškerý obsah na stranu klienta. Na rozdíl od vícestránkových aplikací není potřeba načítat celou stránku znovu při změně zobrazovaných dat. Informace jsou na stránku vykreslovány pomocí jazyka JavaScript. Serverovou část používají pouze jako zdroj a úložiště dat, se kterou komunikují přes serverem vystavené API. Tuto architekturu se vyplatí použít, pokud je, nebo v budoucnu bude, požadovaná aplikace se stejnou funkcionalitou jako webové stránky na jiné platformě, např. mobilní aplikace. Tu je poté snazší implementovat, protože může využít stejné serverové části jako webová aplikace, kdežto u vícestránkové aplikace je nutné buďto vystavit API v serverové části, nebo napsat pro novou aplikaci vlastní server [18].

Pro implementaci jednostránkové aplikace je tedy nutné vystavit API a implementovat její klientskou část, která se nejčastěji implementuje pomocí některého z JavaScript frameworku, jako např. Angular, React.js nebo Vue.js [19].

Výhody
- Po prvním načtení stránky rychlejší načítání obsahu - posílá se pouze část dat, nikoli celá stránka,
- Jsou multiplatformní - vzhledem k vystavenému API může být serverová část použita pro více platforem, např. mobilní zařízení,
- Použití bez přístupu k internetu - jelikož je vše uloženo lokálně v prohlížeči, je možné některé funkcionality používat i bez připojení k internetu.

Nevýhody
- Složitá implementace SEO,
- Potencionálně dlouhé první načtení - musí se stáhnout veškeré JavaScript soubory do lokálního úložiště klienta.



Obrázek 3.1: Životní cyklus MPA vs. SPA [1]

3.1.3 Výběr rozdělení

Po důkladném zvážení obou možností bylo rozhodnuto implementovat aplikaci jako vícestránkovou. Hlavním důvodem pro tento výběr byl fakt, že výsledná aplikace nebude poskytovat rozsáhlé uživatelské rozhraní s mnoha funkcemi. Na stránkách pro uživatele bude obsah převážně pro čtení. V době psaní práce organizátoři neuvažovali ani o rozšíření aplikace na jinou platformu.

3.1.4 MVT (Model-View-Template) architektura

MVC[7] je jedním z architektonických vzorů pro vývoj webových aplikací. Odděluje řídící logiku od datového modelu a uživatelského rozhraní pomocí tří základních vrstev: Model, Pohled a Řadič. Výměna jedné vrstvy by ve výsledku měla co nejméně ovlivnit ty ostatní a tím docílit lepší škálovatelnosti a rozšiřitelnosti.

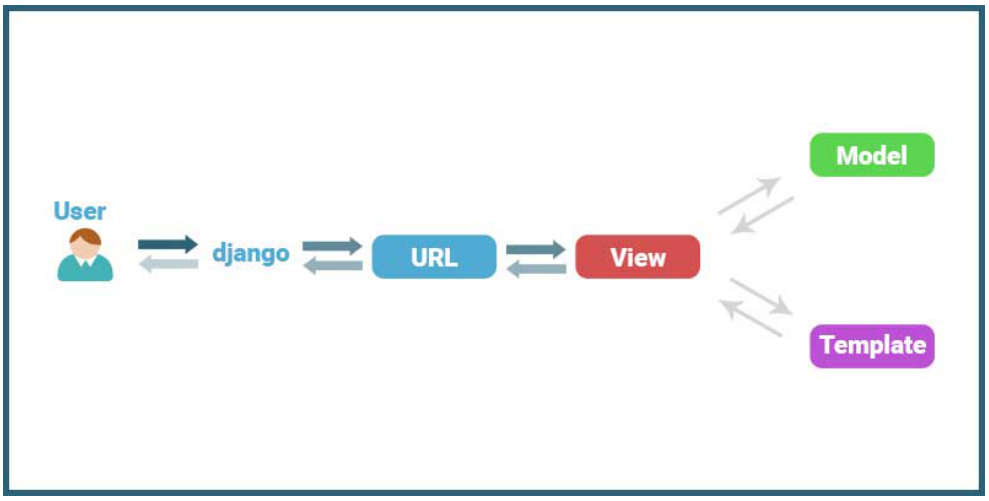
Webový framework Django implementuje MVT architekturu, která je od MVC odvozena. V architektuře MVT je vynechána vrstva řadič, o jejíž funkcionalitu se stará framework sám. Nicméně také rozděluje aplikaci do tří vrstev:

Model Model zajišťuje veškerou logiku ohledně dat a komunikaci s databází.

Pohled Ve frameworku Django stojí vrstva pohled mezi modelem a šablonou. Tato vrstva shromažďuje data od uživatele nebo z modelu a následně je pomocí šablony zobrazí.

[7] MVC, anglicky Model View Controller

Šablona Šablony se využívají ke specifikaci struktury výstupu. Šablona je mix HTML a DTL[8].



Obrázek 3.2: Znázornění architektury MVT

3.2 Uživatelské rozhraní

Tato sekce se zabývá celkovým vzhledem a popisem jednotlivých částí aplikace. Návrh uživatelského rozhraní vznikl na základě spolupráce s organizátory závodu. Výsledek této spolupráce vyústil v Lo-Fi a Hi-Fi design prototyp klientské části, které dodali organizátoři závodu. Ukázka těchto prototypů je k nalezení v příloze D. Celé jsou dostupné na přiloženém médiu. Uživatelské rozhraní administrační části vytvořil autor práce sám. Vznikl pouze Lo-Fi prototyp, jelikož pro administrační prostředí bude použita CSS knihovna Bootstrap, a tím pádem nebylo potřeba navrhovat design komponent. Tento prototyp je také k dispozici na přiloženém médiu.

3.2.1 Záhlaví

Záhlaví je rozděleno do tří částí. První část tvoří logo závodu, které současně slouží jako odkaz na úvodní stránku. Druhá část jsou odkazy na jednotlivé sekce stránek. Třetí část se dynamicky mění na základě stavu, ve kterém se závod právě nachází. Společné prvky pro všechny stavy jsou: přihlášení, změna jazyka a profil přihlášeného uživatele. Proměnlivé jsou pak: přispění posádce a registrace do závodu.

[8]DTL, anglicky Django Template Language - Django šablonový jazyk

3.2.2 Zápatí

Zápatí obsahuje základní informace o webových stránkách (např. rok vzniku), kontaktní informace na pořadatele závodu, číslo bankovního účtu, na který mohou lidé posílat příspěvky pro podporu závodu, a také odkazy na sociální sítě závodu.

3.2.3 Hlavní část

3.2.3.1 Úvodní stránka

Obsah této stránky se mění v závislosti na stavu závodu. Napříč všemi stavy závodu tato stránka slouží jako rozcestník na ostatní stránky a jsou zde zobrazena loga partnerů závodu, která slouží jako odkazy na jejich webové stránky.

Ve fázi přípravy závodu není na této stránce kromě společných prvků nic navíc.

Ve fázi odstartovaného závodu je nejprve zobrazen sloupeček s výsledky, který má vedle sebe mapu. Na mapě jsou zobrazené body, kde se v danou chvíli posádky nachází. Celou dobu je zobrazen počet doposud vybraných peněz. Uživatel poté může vybrat jednu z kategorií hodnocení, což mu zobrazí výsledky a pořadí posádek v dané kategorii. Po kliknutí na řádek v tabulce, nebo na ikonku posádky na mapě se zobrazí detailní výsledky pro danou posádku a ikonka na mapě bude zvýrazněna a vycentrována.

Po ukončení závodu se zde zobrazí článek, ve kterém bude daný ročník závodu shrnut.

3.2.3.2 Posádky

Na této stránce je seznam posádek zobrazený formou dlaždic. Ty obsahují fotografii a základní informace o členech posádky. Uživatel má možnost otevřít detail každé posádky. Každá posádka má přiřazené číslo a barvu, která danou posádku reprezentuje skrze celé webové stránky. Na každé dlaždici je tedy ve čtverci s barvou posádky zobrazeno i její číslo.

V detailu posádky je zobrazena stejná fotka jako na stránce se seznamem posádek. Uživateli je zde poskytnut k přečtení příběh o tom, jak se posádka dala dohromady. Tento příběh se vyplňuje při registraci do závodu.

Pokud je závod odstartován, je na pravé straně od popisu posádky viditelný panel se stručným popisem metod, kterými lze posádce přispět, a také odkaz na stránku s přispíváním. Dále je zde zobrazeno aktuální umístění posádky a počet peněz, které jí lidé přispěli.

Na konci stránky jsou dlaždice se zážitky dané posádky, které obsahují fotografii, titulek a prvních pár vět z příběhu zážitku. Dlaždice zároveň slouží jako odkaz na detail daného zážitku.

3.2.3.3 Úkoly

Stejně jako u posádek se úkoly zobrazují na dvou stránkách: seznam všech úkolů a detail úkolu. Stránka se seznamem má stejné rozvržení jako stránka se seznamem posádek.

Na stránce detailu úkolu jsou informace o úkolu, jako je zadání, odhadovaný čas plnění a kolik posádek daný úkol splnilo. Vedle toho jsou vyobrazené miniatury profilových obrázků posádek, které úkol splnily. Ty slouží jako odkazy na stránku detailu dané posádky.

Na konci stránky je list zážitků, které popisují plnění zobrazeného úkolu. Rozložení těchto dlaždic je stejné jako na stránce detailu posádky.

Pokud je přihlášená posádka a daný úkol ještě nemá splněný, je zde tlačítko na odevzdání úkolu. Pro odevzdání úkolu je posádka přesměrována na stránku s formulářem. Po úspěšném vyplnění formuláře je přesměrována opět na zadání úkolu a je informována o úspěšném odeslání ke schválení.

3.2.3.4 Zážitky

Na stránce zážitků je zobrazený seznam všech zážitků všech posádek, který je řešen stejně jako seznam zážitků na stránce *Detail posádky*. Uživatel si může filtrovat zážitky podle posádek.

V detailu zážitku je zobrazena fotografie, která dokumentuje zážitek, jeho nadpis a příběh, který se posádce stal. Pod článkem je zobrazena miniatura profilové fotky posádky s jejich jmény. Tato miniatura slouží jako odkaz na detail posádky.

Na konci stránky jsou zobrazeny pomocí dlaždic ostatní zážitky posádky, které je možné si přečíst.

3.2.3.5 O závodu

Na této stránce jsou veškeré informace o závodu s odkazem na stránku s pravidly závodu, často kladené dotazy a mapa se znázorněním trasy závodu.

Pokud je možné se ještě registrovat do kvalifikačního závodu, je zde popis kvalifikačního závodu s odkazem na registraci.

Po odstartování závodu je zde také informace o tom, kolik se zatím vybralo peněz.

3.2.3.6 Pomáháme

Tato stránka bude přístupná až pár týdnů před startem závodu, jelikož výběr osoby s Konto Bariéry neprobíhá při zakládání závodu. Poté, co se domluví subjekt pro výběr peněz, bude na stránce zobrazen aktuální článek s informacemi o dané osobě. Dále zde bude zobrazena informace o tom, kolik se vybírá a kolik bylo doposud vybráno.

Po odstartování závodu zde přibude tlačítko s odkazem na stránku s přispíváním.

3.2.3.7 Přispíváme

Na této stránce budou zobrazeny dlaždice s posádkami. Ty budou sloužit pro výběr posádky, na kterou chce návštěvník přispět. Po kliknutí na posádku bude uživatel přesměrován na stránku s výběrem platební metody. Na té bude možnost vybrat jednu ze tří platebních metod. Po vybrání bude uživatel přesměrován na stránky dané metody pro provedení platby a po jejím úspěšném dokončení bude přesměrován na stránku, která ho o úspěšné platbě bude informovat.

Dále bude na stránce zobrazeno číslo transparentního účtu pro ty, kteří by chtěli podpořit přímo závod a jeho organizaci.

3.2.3.8 Historie

Na této stránce se bude moci uživatel dozvědět o minulých ročnících závodu. Budou zde odkazy na zprávy o závodu v médiích.

Dále zde v podobě dlaždic budou odkazy na minulé ročníky, kde si návštěvníci budou moci přečíst shrnutí závodu, jaké úkoly daný rok posádky musely plnit a jaké zážitky při cestování zažily.

3.2.4 Autentizace

Stránky, které zajišťují autentizaci, mají jiné celkové rozvržení stránek. Záhlaví je změněno pouze na logo závodu, které je současně odkaz zpět na úvodní stránku závodu.

3.2.4.1 Přihlášení

Pro přihlášení je uživateli zobrazen klasický formulář pro zadání uživatelského jména a hesla. Dále je zde poskytnuta možnost obnovení hesla, pokud ho uživatel zapomněl.

3.2.4.2 Registrace

Registrace do závodu probíhá najednou pro celou posádku a je rozdělena do tří částí. První dvě části jsou totožné, u obou je zobrazen formulář pro vyplnění údajů o jednom ze spolujezdců. Poslední část je věnována informacím o posádce, kde je potřeba nahrát fotografii a napsat příběh, který popisuje posádku.

Mezi jednotlivými částmi registrace se dá navigovat pomocí zobrazených navigačních tlačítek.

Po úspěšné registraci je zobrazena stránka s informací o dokončení registrace.

3.2.5 Administrační prostředí

Administrační prostředí má jiné rozložení než zbytek webových stránek. V záhlaví je pouze logo, odkaz na úvodní stránku závodu a tlačítko na odhlášení.

Pro navigaci je zde postranní menu, ve kterém jsou odkazy na jednotlivé stránky administračního prostředí. Tato lišta je rozdělena do dvou sekcí: současný závod a ostatní. Pokud byl závod ukončen, přibude zde tlačítko na založení nového závodu.

Stránky jsou dvojího typu: tabulky a formuláře. Pokud daná sekce obsahuje více záznamů, je nejprve zobrazena tabulka se všemi záznamy, nad kterými jde provádět hromadné akce, např. mazání. Poté je možné si každý záznam zobrazit jako detail, který se otevře ve formě formuláře. Pokud se jedná o unikátní záznam, je zobrazen rovnou formulář.

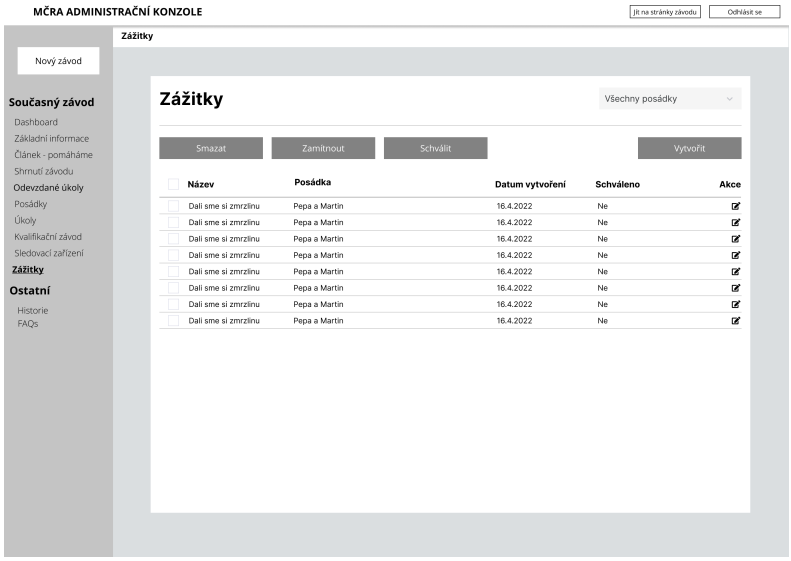
Příkladem stránky s více záznamy jsou zážitky. Na stránce se zážitky se zobrazí tabulka se všemi záznamy současného závodu. Zde je možné je hromadně smazat, schválit nebo zamítnout. Po zobrazení detailu zážitku se zobrazí formulář, pomocí kterého může administrátor záznam libovolně upravit. Také jsou zde tlačítka pro akce, které bylo možné použít hromadně v případě tabulky.

Příkladem unikátní stránky je hlavní článek závodu. Na stránce je rovnou formulář, ve kterém lze pouze upravovat data, nikoli přidávat nová, nebo mazat.

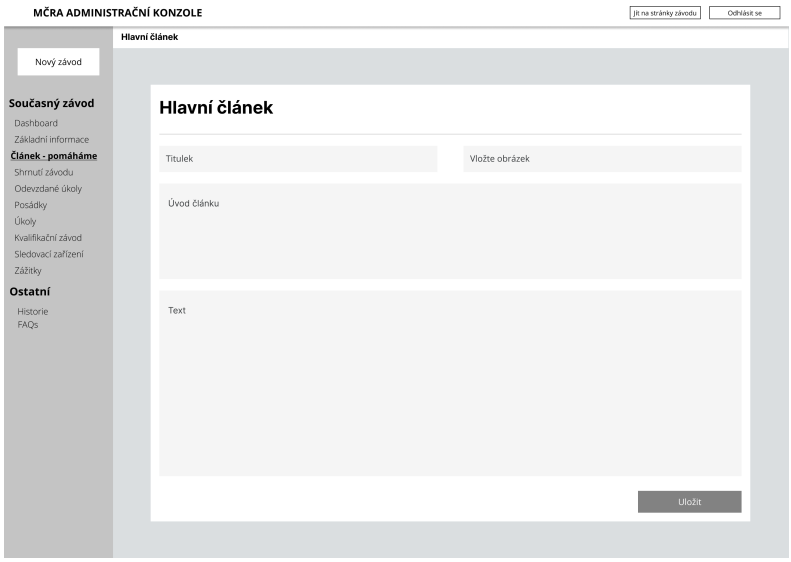
Jedinou stránkou, která nepatří ani do jedné skupiny, je řídící panel v rámci současného závodu. Na této stránce jsou zobrazeny statistiky probíhajícího závodu, tabulky s nově vytvořenými zážitky a nově odevzdanými úkoly a tlačítka na změnu stavu závodu.

3.2.6 Mobilní verze

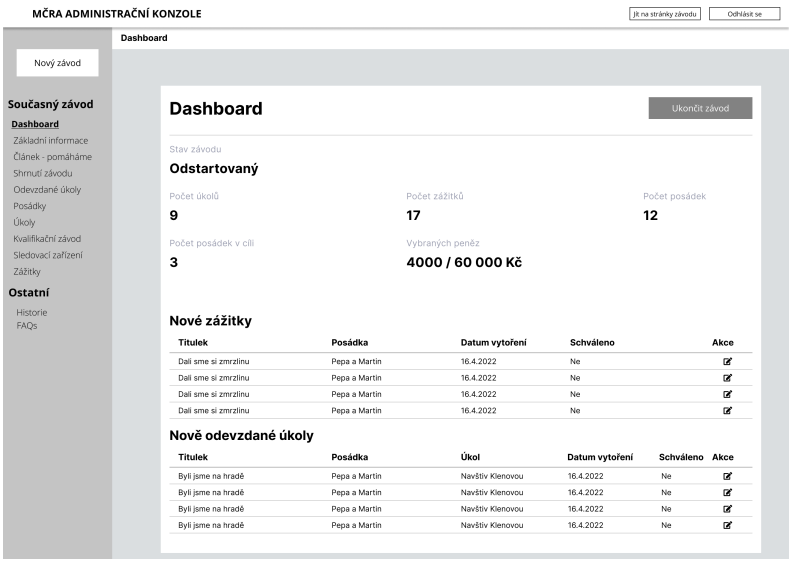
Z nefunkčních požadavků plyne, že je nutné, aby webové stránky byly optimalizované i pro mobilní zařízení. Uživatelské rozhraní se mění v závislosti na rozlišení obrazovky, na kterém je zobrazeno. Při zobrazení na mobilním zařízení se některé prvky, které jsou ve výchozím rozložení vedle sebe, zobrazí pod sebou, např. mapa na hlavní obrazovce se na mobilním zařízení zobrazuje až pod tabulku výsledků. Stejně tak se mění i záhlaví stránky. Na mobilním zařízení je vyobrazeno pouze logo závodu a tlačítko, které po stisknutí zobrazí vyjíždějící menu s odkazy.



Obrázek 3.3: Návrh obrazovky Zážitky v administrační části



Obrázek 3.4: Návrh obrazovky Hlavní článek v administrační části



Obrázek 3.5: Návrh obrazovky Dashboard v administrační části

Kapitola 4

Implementace

4.1 Struktura projektu

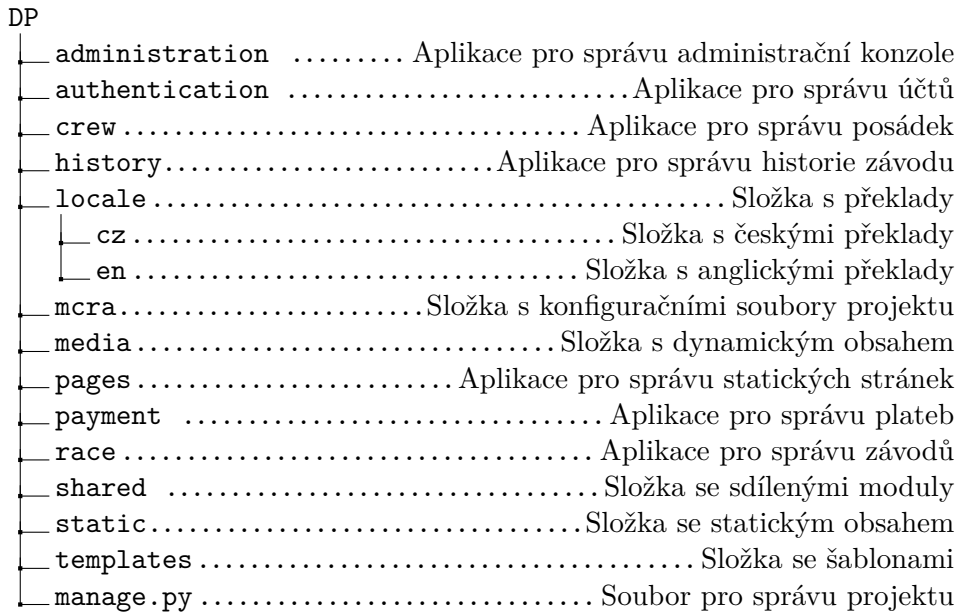
Webový framework Django vyžaduje pevně danou základní strukturu projektu. Hlavním prvkem projektu této práce je složka *mcra*, která obsahuje konfigurační soubory projektu. Pro větší přehlednost je projekt rozdělen do několika tzv. aplikací, které definují funkcionality ohledně logických celků projektu. Složka *locale* obsahuje překlady statického obsahu aplikace, v případě této práce anglický a český. Složka *static* obsahuje veškerá statická data aplikace. Naopak složka *media* slouží k ukládání dynamicky nahrávaných dat, jako například obrázků. Ve složce *templates* jsou veškeré šablony aplikace. Celý projekt je spravován souborem *manage.py*, který obsahuje například funkce pro vytvoření migrací, uživatelů atp.

Struktura hlavních prvků projektu je zobrazena na obrázku 4.1. Některé méně důležité soubory byly pro větší přehlednost vynechány.

4.2 Struktura jednotlivých aplikací

Django také přichází se strukturou jednotlivých aplikací, jejíž nejdůležitější části jsou popsány v této sekci. Mezi ty patří:

- Modely,
- Pohledy,
- Šablony,
- Směrování.



Obrázek 4.1: Struktura projektu

4.2.1 Modely

Pomocí modelů Django vytváří tabulky, jejich atributy a omezení, které jsou v modelu uvedeny. Do tabulek relační databáze jsou modelové třídy nahrány pomocí databázových migrací.

Každý model musí dědit od třídy *django.db.Model*. Atributy jsou definovány jako třídní proměnné, kde každá z nich má svůj datový typ. Základní datové typy jsou přímými potomky třídy *django.db.models.Field*. Mezi takové patří například *django.db.models.CharField* nebo *django.db.models.IntegerField*. Django však poskytuje i datové typy, které rozšiřují funkcionalitu těchto základních typů. Příkladem je *django.db.models.EmailField*, který automaticky nastaví validátor e–mailu na tento atribut. Pokud ani tyto rozšířené možnosti nestačí požadavkům, existuje spousta rozšíření třetích stran, které poskytují více datových typů. V této práci byl použit datový typ *ckeditor.fields.RichTextField*, který se stará o formátovaný text. Uživateli je na stránce zobrazen textový editor, kde uživatel může psaný text formátovat a následně uložit do databáze.

Vztahy mezi modely jsou definovány pomocí tří atributů [20]:

- *django.db.models.OneToOneField* - 1:1,
- *django.db.models.ForeignKey* - 1:N,
- *django.db.models.ManyToManyField* - M:N.

4.2.1.1 Databázové migrace

Databázové migrace jsou ve frameworku Django způsob, jak propagovat změny modelových tříd do databázového schématu. Jsou připraveny tak, aby byly co nejvíce automatické. Nová migrace vytvoří soubor, ve kterém jsou uloženy veškeré informace o tom, jaké změny se v jakém modelu staly. Tyto soubory Django ukládá do databáze, aby se dalo mezi migracemi přepínat a aby byl někde seznam všech změn modelů. Každá aplikace má svůj vlastní soubor s migracemi, aby byly stále co nejvíce odděleny pro případné samostatné využití v jiném projektu.

Migrace se vytvářejí pomocí příkazů, které jsou definované v souboru *manage.py*. Nejprve je třeba spustit příkaz `python manage.py makemigrations`, na základě kterého se ve všech aplikacích vytvoří nový migrační soubor, pokud v modelech dané aplikace proběhla nějaká změna. Následně příkazem `python manage.py migrate` se aplikují migrace do databázového schématu. Pokud nějaká změna vyžaduje další informace, programátor je může zadat do příkazové řádky, kde migraci spustil. Takovou informací může být například výchozí hodnota u atributu [21].

4.2.1.2 Django databázové API

Po vytvoření modelových tříd je programátorovi automaticky k dispozici databázové API, které umožňuje CRUD operace nad databází [22].

Pro vytváření záznamu v databázi stačí vytvořit instanci modelové třídy. To samo o sobě neodešle žádné informace do databáze. Pro vyvolání požadavku na vytvoření záznamu v databázi je nutné použít metodu *save*, která na pozadí provede SQL příkaz `INSERT`. Pro vytvoření a uložení objektu najednou lze použít metodu *create*. Metoda *save* se také používá pro uložení změn instance, která již do databáze vložena byla.

Pro získávání objektů z databáze používá Django databázové API třídy *QuerySet* a *Manager*.

QuerySet reprezentuje množinu objektů v databázi a může obsahovat filtry, které danou množinu zužují, na základě jejich parametrů. Ekvivalent QuerySetu v SQL jazyce je `SELECT` s filtry jako `WHERE` a `LIMIT`. Programátor má přístup ke QuerySetu skrze třídu *Manager*.

Manager je hlavním zdrojem QuerySetů pro model. Ten ze základu obsahuje jednoho Managera se jménem *objects*. Pomocí toho je programátor schopen dostat QuerySet se všemi záznamy daného modelu v databázi.

Filtrování a vyžádání QuerySetu samo o sobě nevyvolá žádný SQL dotaz, který by se aplikoval na databázi. Dotaz je do databáze odeslán až po samotném optání na obsah dat (např. vypsání některého z atributů modelu). Při načítání dat z databáze se s modelem nenačítají související entity přes

cizí klíče. Pokud entita obsahuje pole souvisejících entit o velikosti N, které by chtěl programátor vypsat, do databáze se odešle N dotazů. Tento problém se nazývá N + 1 [23], který se ve frameworku Django řeší použitím metod *select_related a prefetch_related*.

select_related následuje cizí klíče a načte související entity v rámci původního dotazu. Díky tomu není potřeba nadbytečných N dotazů do databáze. Tato metoda však nelze použít pro vztah M:N.

prefetch_related narozdíl od *select_related* je možné použít pro vztah M:N, jelikož pro každý vztah modelu zavolá vlastní dotaz do databáze a výsledné spojování probíhá na straně serveru v Pythonu.

4.2.1.3 Signály

Django implementuje tzv. signály, které pomáhají odděleným aplikacím dostávat upozornění, že se stala nějaká akce kdekoli jinde v aplikaci. Pro to, aby byl signál přijat, se musí zaregistrovat přijímací funkce použitím funkce *Signal.connect*. Tato funkce se zavolá pokaždé, kdy je odeslán signál, ke kterému se zaregistrovala [24].

Django poskytuje několik předdefinovaných signálů, které jsou programátorovi k dispozici. Jedním z nich je signál *django.db.models.signals.pre_save*. Tento signál je odeslán těsně před uložením entity do databáze. V této práci je použit například pro inicializaci hodnot posádky, která se kvalifikovala do hlavního závodu. Po obdržení signálu se inicializují některé potřebné hodnoty, přiřadí se sledovací zařízení, je-li nějaké volné a vytvoří se pro danou posádku účet pro přihlášení.

```
1  def init_crew_creation(sender, instance, **kwargs):
2      if instance.pk is None:
3          race = Race.objects.filter(is_active=True).first()
4          crew_number = _get_crew_numbers()[0]
5          instance.race_id = race.pk
6          instance.crew_number = crew_number
7          instance.account =  Account.objects.create_crew_user(
       instance)
8          tracker = Tracker.objects.filter(crew__isnull=True,
       race__is_active=True).first()
9          if tracker:
10             instance.tracker = tracker
11
12 pre_save.connect(crew_account, sender=Crew)
```

Ukázka kódu 4.1: Signál pro inicializaci posádky

4.2.2 Pohledy

Pohledy jsou v rámci frameworku Django funkce, které jako vstupní parametr přijímají webový požadavek společně s dalšími možnými parametry z URL dotazu a vrací webovou odpověď. Tou může být HTML kód, přesměrování, obrázek, JSON dokument nebo cokoli jiného. V rámci této funkce je definovaná veškerá logika, která je potřeba k připravení a odeslání odpovědi.

Django poskytuje několik předpřipravených odpovědí. V této práci jsou nejčastěji použity tyto:

- *django.shortcuts.render* - Tato funkce přijímá parametry: požadavek, název šablony, která se má vykreslit, a kontext. Kontextem je objekt, který obsahuje data potřebná k vykreslení šablony.
- *django.shortcuts.redirect* - Tato funkce slouží k přesměrování na jiný pohled aplikace, jehož jméno je do funkce předáno jako parametr.
- *django.http.HttpResponseBadRequest* - Klasická HTTP odpověď se status kódem 400 .
- *django.http.HttpResponseNotFound* - Klasická HTTP odpověď se status kódem 404.
- *django.http.JsonResponse* - Tato funkce přijímá data, která mají být serializována do JSON objektu, který následně odešle jako odpověď. Tato funkce se v práci používá pro posílání dat v rámci AJAX dotazů.

Pokud je potřeba zobrazovat na stránce pouze list instancí modelu nebo stránku detailu modelu, je možné vytvořit třídu, která dědí od těchto tříd:

django.views.generic.ListView V dané třídě je nutné definovat model, jehož instance mají být zobrazeny v listu, a název šablony, do které se mají data zobrazit. Množinu zobrazovaných objektů je možné specifikovat zadáním QuerySetu, ze kterého se mají data nahrát z databáze.

django.views.generic.DetailView Stejně jako u ListView je potřeba definovat model, šablonu a je také možné zadat QuerySet. V požadavku na použití této třídy je potřeba zadat identifikátor objektu, jehož detail má být zobrazen. Pokud chce programátor upravit objekt, který je do šablony poslán, může toho docílit přepsáním funkce *get_object*. Pro doplnění kontextu o jiná data je možné přepsat funkci *get_context_data*, ve které se přidají do aktuálního JSON objektu reprezentující kontext.

4.2.3 Šablony

Šablony jsou způsob, jak framework Django dynamicky generuje HTML obsah. Šablona se skládá ze statické části tvořené klasickým HTML kódem a speciální

synataxí, která definuje, jak se má vkládat dynamický obsah. Ta se nazývá Django Template Language (DTL)[9]. Tento obsah se skládá ze čtyř základních komponent [25]:

Proměnné Pomocí této komponenty se zobrazují hodnoty proměnných nebo kontextu. Syntaxe proměnných je `{{ proměnná }}`.

Tagy Tagy mají v šablonách mnoho využití. Mohou generovat text, zajišťují možnost používání cyklů a podmínek nebo nahrávají do šablony externí informace. Syntaxe tagů je následující: `{% tag %}`. Některé tagy potřebují i ukončovací tag, např. for-cyklus: `{% for %} logika cyklu {% endfor %}`.

Filtry Pro upravování zobrazení proměnných DTL využívá tzv. filtrů. V práci se využívá filtr *safe*, který vypne escapování[10] znaků a tím je možné zobrazit formátovaný text, který je do šablony poslán jako řetězec obsahující HTML kód. Syntaxe filtru je následující: `{{ proměnná|safe }}`.

4.2.3.1 Statické soubory

Veškeré statické soubory jsou uloženy ve složce *mcra/static*. Pro produkční prostředí je potřeba pomocí příkazu `python manage.py collectstatic` přesunout statické soubory do složky *static* v kořenovém adresáři projektu, ze které je bude příslušná služba poskytovat. V této práci statické soubory poskytuje webový server Nginx.

Aby bylo možné statické soubory používat v šablonách, je nutné je pomocí tagu `{% load static %}` do šablony nahrát. Poté už je možné ke statickým souborům přistupovat pomocí tagu
`{% static 'název statického souboru' %}`.

4.2.3.2 Dědičnost

Jednou z užitečných vlastností Django šablon je dědičnost. Pomocí té jde napsat kostru stránky, na které se definují bloky. Šablona, která od dané kostry dědí, může bloky přepsat a tím vytvořit specifický obsah stránky. Pokud některý z bloků v šabloně nebude přepsán, použije se blok z rodičovské šablony. Syntaxe dědění např. od šablony *base.html* je: `{% extends "base.html"%}` [25] V této práci sou vytvořeny tři kostry: pro stránky zabývající se autentizací, pro veřejné webové stránky a pro administrační konzoly. V těchto kostrách je nahrán všechen obsah, který mají stránky dané části aplikace společný, včetně společných referencí na styly a soubory JavaScript.

[9] DTL, anglicky Django Template Language - Django šablonovací jazyk

[10] Escaping je metoda, která nám umožňuje říci počítači, aby ignoroval speciální funkci znaku [26].

4.2.4 Adresy URL

Každá aplikace projektu obsahuje soubor *urls.py*, ve kterém jsou definované veškeré URL adresy aplikace. URL adresa zde přímo odkazuje na jeden pohled. Tento soubor je potom nahrán do souboru *mcra/urls.py*, kde musí být nahrány veškeré URL adresy projektu, aby framework Django mohl obstarávat směrování.

4.3 Autentizace

Framework Django poskytuje autentizaci uživatelů jako výchozí funkcionalitu. Avšak pro účely této práce musela být upravena. Posádku tvoří dva závodníci a přihlašují se za účelem odevzdání úkolů nebo psaní zážitků posádky. Z tohoto důvodu bylo rozhodnuto, že bude účet pro posádku, nikoli pro jednotlivé závodníky. Ve výchozím nastavení autentizace ve frameworku Django používá pro přihlašování e–mail. Toto nastavení bylo potřeba změnit na přihlašování pomocí uživatelského jména, čehož bylo docíleno napsáním vlastního přihlašovacího backendu, který dědí od výchozího *django.contrib.auth.backends.ModelBackend*. Zde bylo nutné přepsat funkci *authenticate*, kde bylo pro nalezení účtu použito uživatelské jméno. Dále bylo potřeba vytvořit vlastní model pro přihlášené uživatele. Ti jsou dvojího typu: posádka a admin. Výchozí třída již obsahuje proměnné *is_admin* a *is_superuser*, podle kterých framework Django pozná, zda se jedná o admina, či nikoli. Bylo tedy potřeba vyřešit, jak do účtu zahrnout informace o posádce, aniž by tyto informace vyžadoval i účet typu *admin* nebo *superuser*. To bylo vyřešeno pomocí samostatného modelu *Crew*, který slouží jako profil daného účtu. Ten v případě *admina* a *superusera* neexistuje. Aby framework Django používal vlastně definované třídy, bylo nutné je explicitně zadat v souboru *mcra/settings.py*.

4.4 Překlady

Pro překlady webových stránek do více jazyků bylo nutné provést tyto kroky:

1. V souboru *mcra/settings.py* nastavit, kam se mají vygenerované soubory pro překlad ukládat, jaké jazyky bude aplikace podporovat a jaký je výchozí jazyk.

2. Všechny řetězce, které mají být přeloženy předat do funkce pro přeložení. V Python a Javascript kódu je v této práci použita funkce *gettext*. V šablonách je místo funkce použit tag `{% trans %}`

3. Pomocí příkazu `django-admin makemessages` vygenerovat soubory s příponou *.po*. Pro každý podporovaný jazyk se vygeneruje jeden takový soubor, ve kterém je potřeba doplnit překlady řetězců označených v bodě 2.

4. Po přeložení všech řetězců je potřeba soubory s koncovkou *.po* zkompilovat do binárních souborů s koncovkou *.mo*, a to příkazem `djano-admin compilemessages`.

5. Do navigační lišty umístit výběr požadovaného jazyka.

Po splnění všech kroků byla aplikace schopná zobrazovat obsah ve více jazycích, konkrétně v angličtině a češtině. Pro přidání dalšího podporovaného jazyka je nutné pouze zopakovat kroky 3 a 4 s tím, že v kroku 3 je nutné doplnit překlady jen pro nový jazyk. Při prvním spuštění webových stránek se aplikace pokusí zjistit, jaký jazyk má nastavený prohlížeč, ze kterého požadavek přišel. Pokud tuto informaci získá a daný jazyk je mezi podporovanými, pak bude nastaven jako výchozí. V opačném případě bude použit výchozí jazyk ze souboru *mcra/settings.py*.

4.5 Odesílání e–mailů

Aplikace musí být schopná odesílat informace organizátorům a závodníkům pomocí e–mailů. Proto bylo nutné vybrat nějakou službu, která odesílání umožňuje. Vzhledem k tomu, že bude aplikace nasazena na serverech AWS, nabízela se možnost využít jejich službu Amazon Simple Email Service (SES). Jelikož služba, která se v aplikaci použije, je snadno zaměnitelná, bylo rozhodnuto SES pro implementaci použít. Hlavním důvodem byla autorova možnost prozkoumat a naučit se používat další z AWS služeb. SES je cenově efektivní, flexibilní a škálovatelná e–mailová služba, která umožňuje vývojářům odesílat e–maily z jakékoli aplikace. Pomocí Amazon SES můžete e–maily posílat bezpečně a globálně[27]. Po založení má účet status *SANDBOX*. To znamená, že není možné posílat e–maily na jakoukoli adresu, ani z jakékoli e–mailové schránky. V tomto stavu je možné používat pro přijímání i odesílání pouze ověřené účty, zadané ve vývojářské konzoli. V době psaní této práce ještě účet vyvíjené aplikace neprošel schvalovacím procesem a má tedy status *SANDBOX*. Proto byly pro účely testování zřízeny speciální e–mailové schránky, které testované osoby mohou využívat. Vzhledem k tomu, že existuje pouze jedno GPS zařízení, není potřeba více než dvou posádek. Jednu s GPS zařízením a jednu bez ní, aby se otestovala funkčnost aplikace. Proto byly pro účely testování aplikace vytvořeny pouze čtyři e–mailové schránky pro účty posádek a jedna pro samotnou aplikaci.

Po získání ověření účtu aplikace je možné odesílat e–maily z jakékoli schránky na jakoukoli adresu. Pokud je aplikace nasazená na Amazon EC2 nebo pomocí AWS Lambda, což je případ této práce, je možné odeslat zdarma 62000 e–mailů za měsíc [28]. Vzhledem k tomu, že aplikace odesílá e–maily primárně při registraci, obnově hesla nebo odevzdání úkolu, je nepravděpodobné, že by danou hranici přesáhla.

4.6 Platby

Z funkčních požadavků vyplývá, že má aplikace uživateli umožňovat platit třemi způsoby: DMS, PayPal, platební karta. V následujících podsekcích je popsána integrace těchto platebních metod.

4.6.1 DMS platby

Placení pomocí DMS nevyžaduje žádnou interakci mezi uživatelem a webovými stránkami. Jde o odeslání SMS ve specifickém tvaru na předem dané číslo. Tyto údaje uživatel může najít na webových stránkách. Několik dní před spuštěním závodu jsou zřízena počítadla pro posádky závodu. Ta mají za úkol uschovávat informace o příchozích SMS, podle kterých je možné zjistit, kolik přidružené posádce bylo posláno peněz. Informace z počítadel jsou dostupné přes vystavené API, které odesílá dva různé formáty. Buďto odešle pouze číslo reprezentující celkovou vybranou sumu, nebo detailní výpis, ve kterém jsou vybrané sumy rozděleny podle typu poslané SMS.

V aplikaci je vybírání peněz pomocí DMS simulováno automatickým přičítáním peněz posádce v přesně definovaném intervalu, protože v době psaní práce nebyl aktivní žádný závod a autorovi nebylo poskytnuto testovací API. V reálném provozu by ve stejném intervalu byly odesílány dotazy pro získání celkové vybrané sumy, která by se posádce přidělila. Proto je přičítání peněz adekvátní simulace.

4.6.2 PayPal platby

Přijímání plateb přes službu PayPal vyžaduje založení firemního účtu na jejich stránkách. Pro účely vývoje aplikace je zpřístupněno testovací prostředí, pomocí kterého jde aplikaci testovat bez posílání reálných peněz, tzv. *PayPal sandbox* [29]. Toto prostředí je využito i v této práci.

Aplikace využívá knihovnu *paypalrestsdk*, která poskytuje Python API pro vytváření, zprostředkování a správu plateb. Po zadání částky uživatelem je odeslán požadavek na vytvoření platby v dané výši. Pokud je tato platba úspěšně vytvořena, je uživatel přesměrován na stránky PayPal pro provedení samotné platby. Při placení si po přihlášení do PayPal účtu může uživatel vybrat, zda chce platit svým zůstatkem na tomto účtě, nebo kartou. Po dokončení platby odešle PayPal informaci o stavu platby a přesměruje uživatele zpět na stránky závodu.

Ukládání dat o platbě při přesměrování zpět na stránky závodu není doporučeno, jelikož při aktualizaci stránky po přesměrování dojde k odeslání stejných dat do aplikace, a proto by docházelo k duplikaci dat. Proto je v aplikaci využit webhook, který po každé dokončené platbě odešle do aplikace informaci o stavu dané platby. Až v tento moment se uloží záznam o platbě do databáze a peníze jsou přičteny posádce.

4.6.3 Platby platební kartou

Pro přispívání pomocí platební karty byly brány v potaz služby Stripe a PayPal. Hlavní rozdíl mezi těmito službami je v ceně za transakci. Stripe strhává 1.4% + 6.5Kč za transakci pro evropské karty.[30] a PayPal 1.9% + 10Kč [31] za transakci. Po dohodě s organizátory závodu bylo rozhodnuto pro přijímání plateb platebními kartami využít levnější variantu.

Po založení Stripe účtu je účet ve stavu pro vývojáře. Je možné ho integrovat a vyzkoušet si tak funkcionality této platformy před aktivací účtu na produkční verzi.

Aplikace využívá knihovnu *stripe*, která zprostředkovává komunikaci se Stripe API. Po zadání částky uživatelem je zažádáno o vytvoření relace pro provedení platby. Pokud byla relace vytvořena úspěšně, je uživatel přesměrován na stránky Stripe. Po dokončení platby je uživatel přesměrován zpět na stránky závodu společně s informacemi o platbě. Stejně jako v případě platby pomocí PayPal jsou tyto informace ukládány na základě dat získaných z webhooku, nikoli z dat po přesměrování uživatele.

4.7 Odevzdávání úkolů typu Místo

Během závodu je potřeba splnit několik úkolů a některé z nich jsou typu *místo*. Takové úkoly mají přesné souřadnice a rádius, ve kterém je možné úkol splnit. Aby mohla posádka pokračovat v cestě a zážitek o splnění úkolu odeslat později, existuje pro ně možnost *Navštívit místo*. Tato akce určí, zda se posádka nachází v povolené vzdálenosti od místa plnění, a pokud ano, započítá se jim tento bod jako navštívený.

Jelikož směrovací stroj OSRM je nastaven na hledání cest pro automobily, není vhodné ho použít pro tento typ určování vzdálenosti. Je možné, že dané místo bude ležet mimo pozemní komunikace, a pokud by byl rádius pro odevzdání úkolu dost malý, nebylo by možné dané místo podle směrovacího stroje OSRM nikdy navštívit.

Proto byl v práci použit tzv. *Haversine vzorec* [32], který slouží pro výpočet přímé vzdálenosti mezi dvěma body na mapě. Ty jsou určeny zeměpisnou šířkou a délkou. Jedno omezení tohoto vzorce je, že je v něm Země brána jako dokonalá koule. Nicméně pro účely této práce, kde budou vypočítávané vzdálenosti maximálně do několika jednotek kilometrů, je jeho přesnost dostačující. Funkce pro výpočet vzdálenosti je zobrazen v ukázce kódu 9.

```
def haversine(lat1, long1, lat2, long2):
    lat1, long1, lat2, long2 = map(radians, [lat1, long1, lat2,
    long2])

    dlong = long2 - long1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlong/2)**2
    c = 2 * asin(sqrt(a))
    return c * 6371
```

Ukázka kódu 4.2: Výpočet vzdálenosti dvou bodů pomocí Haversine vzorce

4.8 OSRM směrovací stroj

Jedná se o výkonný směrovací stroj napsaný v C++14 a navržený pro používání dat z *OpenStreetMap* [33]. Tento stroj je dostupný ve dvou formách: Zdrojové kódy v C++ a Docker image. Jelikož nebylo v plánu zdrojové kódy nijak upravovat a výsledná aplikace bude nasazena do cloudu pomocí Dockeru, byla zvolena forma Docker image. Pro zprovoznění OSRM směrovacího stroje je potřeba provézt následující kroky:

1. Stáhnout Docker image *osrm/osrm-backend*.
2. Stáhnout výpisy dat *OpenStreetMap*.
3. Předzpracovat stažené výpisy pro jeden ze tří profilů: auto, kolo a chodec. V případě této diplomové práce byl využit profil auto.
4. Spustit Docker image za přítomnosti předzpracovaných dat.

```
# Krok 2
wget http://download.geofabrik.de/europe/czech-republic-latest.osm.
    pbf

# Krok 3
docker run -t -v "$PWD:/data"osrm/osrm-backend osrm-extract
docker run -t -v "$PWD:/data"osrm/osrm-backend osrm-partition /data/
    czech-republic-latest.osrm
docker run -t -v "$PWD:/data"osrm/osrm-backend osrm-customize /data/
    czech-republic.osrm
```

Ukázka kódu 4.3: Předzpracování dat pro OSRM směrovací stroj

Výše uvedené řešení funguje pouze pokud pouštíte příkaz ve složce, která data obsahuje. Tento postup byl použit pro lokální vývoj, kdy nebyly služby spouštěny přes Docker kontejnery. V této práci jsou v rámci úspory času použita data pouze pro Českou republiku. První krok předzpracování dat pro celou Evropu na autorově pracovním stroji trval přes dva dny čistého času

a vyžadoval přes 70GB paměti RAM. To bylo vyřešeno pomocí tzv. *Swap memory*.[11]. V rámci úspory času bylo rozhodnuto použít jen omezenou množinu dat, která je pro účely testování prototypu aplikace dostačující. Nicméně při produkčním nasazení budou pouze změnou parametru příkazů použita data pro celou Evropu.

[11]Swap memory - odkládací prostor, jehož primárním úkolem je nahradit místo na disku pamětí RAM, ve chvíli, kdy se skutečná RAM zaplní, ale stále je potřeba více místa [34]

Kapitola 5

Nasazení

5.1 Architektura

Výsledná aplikace se skládá ze tří komponent. Hlavní komponentou je samotná webová aplikace, která je napsaná ve frameworku Django. Další komponenty jsou Směrovací stroj OSRM a databáze PostgresSQL. Aplikace dále využívá dalších externích komponent. První takovou komponentou je Google, která slouží k odesílání e–mailů. Další komponenty jsou KontoBariery, PayPal a Stripe, které se starají o funkcionality ohledně příspěvků. Poslední komponentou je API GPS Dozor, pomocí kterého aplikace získává informace o poloze posádek.

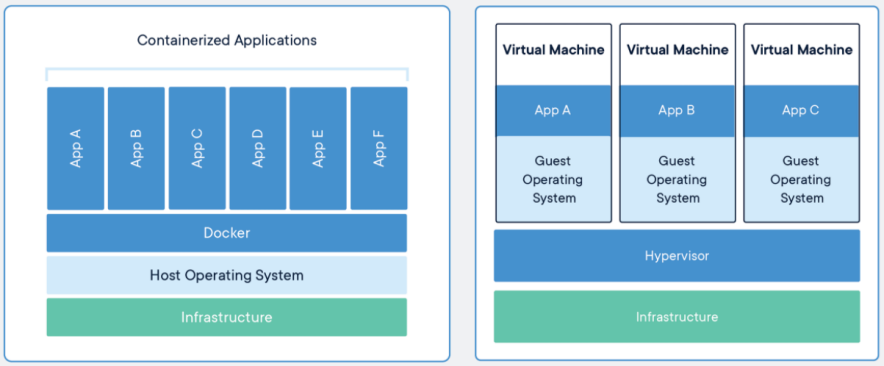
Komponenty, které nejsou externí, jsou nasazeny na serveru v samostatných Docker kontejnerech. Na hostovacím operačním systému jsou tedy nainstalovány pouze služby Docker a Docker Compose. Navíc se zde nachází kontejner obsahující Nginx server, který slouží pro poskytování statických souborů, ostatní požadavky přeposílá do kontejneru s webovou aplikací.

5.2 Docker

Docker je projekt s otevřeným kódem, který zajišťuje virtualizaci nad operačním systémem, takzvanou kontejnerizaci. U klasické virtualizace obsahuje každý virtuální stroj vlastní operační systém. Ten pak komunikuje s hostitelským operačním systémem přes hypervizora, kdežto Docker kontejnery přímo používají funkce hostitelského operačního systému, viz obrázek 5.1.

Výhody použití Dockeru jsou[35]:

- Kontejnery jsou izolované prostředí, a tím jsou do jisté míry nezávislé na hostitelském operačním systému.
- Hostitelský operační systém neobsahuje žádné závislosti aplikace nasazené v kontejneru.



Obrázek 5.1: Porovnání klasické virtualizace a kontejnerizace [2]

- Důsledkem toho, že kontejnery nevidí navzájem běžící procesy, je zvýšená bezpečnost. Pokud je aplikace v jednom kontejneru napadena, neovlivní to aplikace v jiných kontejnerech.

5.2.1 Konfigurace

Pro vytvoření Docker image je potřeba vytvořit soubor Dockerfile, ve kterém je popsán postup sestavení image. V této práci existuje Dockerfile pouze pro webovou aplikaci a pro Nginx server. Jelikož databáze PostgreSQL a směrovací stroj OSRM přichází s již dostupným Docker imagem, není potřeba psát vlastní Dockerfile i pro ně.

5.2.1.1 Dockerfile

Docker umožňuje automatizované sestavování imagů pomocí souboru Dockerfile. To je textový dokument, který obsahuje příkazy, které by uživatel musel zadat do příkazové řádky, aby image vytvořil [36].

Pro sestavení image výsledné webové aplikace bylo využito vícestupňového sestavení, které se používá pro zmenšení výsledné velikosti image. Každý stupeň sestavení vytváří vlastní image, který, pokud to není ten poslední, je na konci sestavení smazán. Během celého procesu se dají mezi těmito stupni sdílet data [37]. V této práci byly použity dva stupně sestavení. Pomocí prvního se vytvoří tzv. *Python Wheels*, které se použijí v druhém stupni. Pro jejich vytvoření je zapotřebí instalace velkého počtu závislostí. Použitím vícestupňového sestavení nejsou tyto závislosti nainstalovány ve výsledném imagi a tím je výrazně zredukována jeho velikost. V druhém stupni pak již probíhá sestavováni image pro výslednou aplikaci, jehož kroky jsou popsány níže.

1. Specifikace základního image pro kontejner. V práci je využit image *python:3.9.6-apline*.

2. Vytvoření složky pro projekt, uživatele a skupiny.
3. Nastavení proměnných prostředí a vytvoření složek pro statické a dynamické soubory.
4. Instalace knihoven, které jsou potřeba pro běh aplikace.
5. Nakopírování souborů, které byly vytvořeny v prvním stupni sestavení a souboru *requirements.txt*, ve kterém jsou vypsány veškeré balíčky použité v aplikaci. Tento soubor je vytvořen příkazem `pip freeze`.
6. Instalace všech balíčků za použití souborů z prvního kroku sestavení.
7. Nakopírování dat aplikace a nastavení práv spouštěcích skriptů.
8. Změna vlastníka hlavní složky a přepnutí uživatele na výše vytvořeného.
9. Nastavení spouštěcího skriptu.

Skript, který se spouští, zajišťuje, aby bylo připraveno prostředí pro běh aplikace. Jsou v něm obsaženy tyto příkazy:

1. `python manage.py collectstatic --no-input` pro shromáždění statických souborů ze všech aplikací do určené složky.
2. `python manage.py migrate --no-input` pro aplikování migrací do databáze.

5.2.1.2 Docker Compose

Docker Compose je nástroj, pomocí kterého se dají spouštět aplikace, obsahující více než jeden kontejner. Pro konfiguraci se používá soubor typu *YAML*, ve kterém jsou popsané jednotlivé služby. Poté stačí spustit příkaz `docker-compose up`, který spustí veškeré služby, které jsou v souboru *docker-compose.yml* nadefinované. Aby nebylo nutné pokaždé spouštět všechny nadefinované služby, Docker Compose umožňuje specifikovat jméno služby, která se má znovu spustit. V případě této práce jsou služby OSRM a PostgreSQL neměnné (Docker image jsou stahovány, nikoli vytvářeny vlastní), tudíž opakované spouštění by znamenalo opakované stahování a spouštění toho samého. Proto je dobré specifikovat jméno nasazované webové aplikace při opakovaném spouštění, a to `docker-compose up web` [38].

Pro přehlednost bude popsána pouze služba webové aplikace s názvem *web* (viz ukázka kódu 27). Níže jsou popsány prvky, které byly použity pro nastavení této aplikace.

build Možnosti konfigurace, které jsou aplikovány při sestavování image. V tomto případě je určena složka s kódem aplikace a Dockerfile, podle kterého se má sestavit Docker image.

image Docker image, který se má použít pro běh kontejneru. V případě této práce se používají image nahrané na AWS službu ECR.

command Příkaz, který se spustí po spuštění kontejneru.

expose Tato konfigurace nastavuje, na kterém portu bude služba dostupná. Obsahuje dvojici *HOST_PORT:CONTAINER_PORT*. Z hostitelského prostředí je služba dostuplná na portu *HOST_PORT*. Docker Compose vytvoří výchozí vlastní síť při každém spuštění. Každá služba se k této síti připojí a může komunikovat s ostatními službami připojenými k této síti pomocí portu *CONTAINER_PORT*.

env_file Jeden nebo více souborů obsahující proměnné, které se mají v kontejneru nastavit jako proměnné prostředí. Očekávaný formát je `VAR=VAL`. Každý takovýto pár je na řádce samostatně.

depends_on Nastavení závislostí mezi službami. Tyto závislosti zajišťují mimo jiné pořadí, ve kterém se budou služby sestavovat, zastavovat nebo mazat.

volumes Specifikují připojení úložiště v kontejneru a hostitelského souborového systému. V případě služby *web* je použito *Named Volume*, u kterých není potřeba specifikovat, na kterou složku hostitelského systému se úložiště v kontejneru připojí.

```
  web:
    build:
      context: ./mcra
      dockerfile: Dockerfile.stage
    image: 650420972883.dkr.ecr.eu-central-1.amazonaws.com/django-
    ec2:web
    command: gunicorn mcra.wsgi:application --bind 0.0.0.0:8000
    volumes:
      - static_volume:/home/mcra/web/static
      - media_volume:/home/mcra/web/media
    expose:
      - 8000
    env_file:
      - ./.env.stage
    depends_on:
      - db
      - osrm


volumes:
  postgres_data:
  media_volume:
  static_volume:
```

```
23   certs:
24   html:
25   vhost:
26   acme:
```

Ukázka kódu 5.1: nastavení služby mcra pro Docker Compose

5.3 Amazon web services

Amazon Web Services poskytuje širokou škálu globálních cloudových produktů, např. výpočetní, úložné, databázové, analytické, síťové, mobilní nebo vývojářské nástroje. Celkem je služeb poskytovaných AWS přes 175. Služby jsou dostupné na vyžádání během několika vteřin a jsou účtovány podle strategie *pay-as-you-go*[12]. AWS také přichází s volně dostupnými prostředky, které kdokoli může volně využívat. Tyto prostředky jsou výkonnostně a velikostně značně omezeny, ale pro účely této práce dostačující [40].

V následujících podsekcích jsou popsány služby, které byly při nasazení využity.

5.3.1 Elastic Compute Cloud

Pro nasazení aplikace byla využita služba EC2 (Elastic Compute Cloud), což je webová služba, která poskytuje výpočetní kapacitu v cloudu. Ta vývojářům umožňuje úplnou kontrolu nad poskytnutými výpočetními zdroji a skrze její webové rozhraní jde snadno konfigurovat [41]. Pro spuštění EC2 instance je potřeba provézt tyto kroky:

1. **Výběr Amazon Machine Image (AMI)** Ami je šablona obsahující operační systém, aplikační server a další aplikace, které jsou potřeba pro nastartování instance. Jelikož nasazovaná aplikace bude spuštěna jako Docker image, není výběr operačního systému podstatný. Byl tedy vybrán *Ubuntu server 20.04 LTS*, protože na stejném operačním systému byla aplikace vyvíjena.

2. **Výběr instance virtuálního serveru** Instance jsou různými kombinacemi parametrů: CPU, paměť, velikost úložiště a síťová kapacita. Výběr kombinací je rozsáhlý, avšak pouze jedna instance patří mezi volně dostupné prostředky. Jelikož v této práci není nasazován finální produkt, ale pouze prototyp pro testování, omezené prostředky jsou dostačující. Pokud bude rozhodnuto nasadit produkční verzi na AWS, změna instance je snadno proveditelná změna. Specifikace vybrané instance je následující:

[12] pay-as-you-go je platební metoda pro cloudové služby, kde se účtují pouze využité služby [39]

- **vCPU** - 1×2.5GHz,
- **Paměť** - 1GB,
- **Úložiště** - pouze Elastic Block Store (EBS),
- **Síťová kapacita** - Nízká až střední.

3. **Výběr úložiště** Úložiště je v práci řešeno pomocí externího EBS. To lze přiřazovat i ostatním instancím, vznikne tedy úložiště nezávislé na serveru a tím bude přechod na jinou instanci snazší. Byla tedy vybrána výchozí varianta poskytovaného úložiště s maximální možnou velikostí 30GB.

4. **Specifikace bezpečnostní skupiny** Tyto skupiny definují pravidla pro přístup na instanci po síti. V této práci je nastaveno, že má k instanci na portu 80 přístup jakákoli IP adresa.

5.3.2 Elastic Block Storage

Elastic Block Storage svazek je úložné zařízení na úrovni bloku, který je možné připojit k instancím. Po připojení svazku k instanci funguje, jako kdyby byl připojen fyzický pevný disk. Je možné dynamicky měnit jejich velikost, jejich typ i kapacitu IOPS [42]. V práci je použit svazek typu *General Purpose SSD (gp2)*. Tento svazek poskytuje rovnováhu mezi cenou a výkonem a je společností Amazon doporučován pro pokrytí většiny potřeb.[43]. Použitý svazek má následující specifikace:

- **Velikost** - 30GB,
- **IOPS** - 100/3000 (minimum 100IOPS, při potřebě možné rozšířit až na 3000IOPS).

5.3.3 Doplňkové AWS služby

Aby nebylo nutné využívat server pro vytváření Docker imagů, bylo využito služby Elastic Container Registry (ECR). Lokálně vytvořený Docker image se nahraje na ECR, ze kterého je následně možné ho stáhnout a používat na EC2 serveru.

Dále byla použita služba Elastic IP addresses, pomocí které se pro AWS účet alokuje statická IP adresa. Ta byla následně použita v DNS záznamu vlastní domény, pod kterou je nyní aplikace dostupná.

5.4 Výsledek nasazení

Výsledná webová aplikace je nasazena v cloudu pomocí služeb AWS a je dostupná na adrese www.dev-mcrautostop.cz. Jelikož výsledkem této práce je

funkční prototyp, nikoli produkční verze aplikace, není nasazena pod doménou závodu. Ta je v době psaní práce stále užívána starou webovou aplikací. Aplikace byla nasazena pro testovací účely.

Kapitola 6

Testování

Po dokončení implementace bylo potřeba výsledný prototyp otestovat. Nejprve byl podroben Nielsenově heuristické analýze a po opravení nalezených problémů byl otestován uživateli. Podrobněji jsou tato testování popsána v následujících sekcích.

6.1 Nielsenova heuristická analýza

Před otestováním webové aplikace uživateli bylo uživatelské rozhraní otestováno pomocí heuristické analýzy podle Jacoba Nielsena. Heuristika se skládá z 10 hlavních bodů, které by mělo uživatelské rozhraní splňovat. Pokud tyto body splňuje, pak by mělo být uživatelsky přívětivé a snadno použitelné [44].

Tuto heuristickou analýzu prováděli čtyři lidé, kterým byla poskytnuta nasazená aplikace se všemi potřebnými informacemi, jako jsou přihlašovací údaje, testovací platební karta, množina souřadnic určitých míst, které je potřeba zadávat, atp. Každý testující prošel celou aplikaci a informoval autora o nalezených chybách a porušení výše zmíněných deseti pravidel. Díky tomuto testování bylo možné značnou část chyb a problémů s uživatelským rozhraním eliminovat před uživatelským testováním.

V následujících podsekcích bude nejprve rozebráno všech deset bodů Nielsenovi heuristické analýzy. U každého bodu bude vysvětleno, co se od systému tímto bodem očekává, následně bude popsáno, jaké problémy byly během testování odhaleny a na konci bude popis prototypu po opravení nalezených chyb.

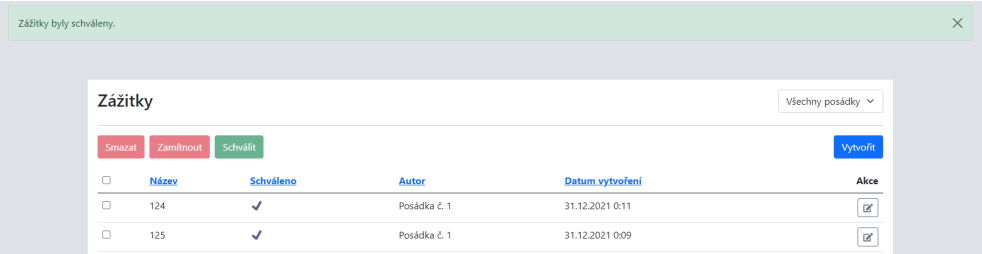
6.1.1 Viditelnost stavu systému

Zadání Uživatel při používání systému musí být obeznámen s tím, kde se právě nachází a co systém v danou chvíli dělá.

Testování Během testování bylo odhaleno, že u některých stránek je chybně zvýrazněná sekce v záhlaví, ve které se uživatel právě nachází. Dále po ode-

vzdání úkolu nebyla uživateli předána informace o tom, co se s odesláním stalo. Názvy stránek v liště prohlížeče nekorespondovaly s tím, kde se uživatel v danou chvíli nachází.

Výsledná aplikace Stránky v aplikaci obsahují přehledné nadpisy, které uživateli poskytují informaci o tom, kde se právě nachází. V navigační liště je dále zvýrazněný odkaz na danou část aplikace. Po odeslání jakéhokoli formuláře je uživatel informován o výsledcích svého požadavku nebo je mu poskytnuta možnost přesměrování na nově vytvořenou položku. Nadpisy v liště webového prohlížeče se shodují s aktuálně zobrazovanou stránkou.



Obrázek 6.1: Notifikace ohledně provedené akce

6.1.2 Shoda mezi systémem a realitou

Zadání Systém by měl používat jazyk uživatele a měl by zachovávat konvence reálného světa, např. ikona koše by měla symbolizovat smazání.

Testování U tohoto bodu bylo odhaleno několik tlačítek a nadpisů, které nebyly přeloženy. Největším problémem byla ikona používaná pro úpravu jednotlivých záznamů v tabulkách v administrační části aplikace. Byla zde použita ikonka oka, která se běžně používá pro zobrazení něčeho. Později byla vyměněna za ikonu tužky, která obecně značí úpravu.

Výsledná aplikace Aplikace je lokalizovaná do dvou jazyků a v obou případech je využívá v jednoduché formě. Použité ikony se dají považovat za standardní a v souladu s jinými aplikacemi. Navíc po najetí kurzorem na tlačítko s ikonou je zobrazen text, který dané tlačítko popisuje slovně.

6.1.3 Uživatelská kontrola a svoboda

Zadání Uživatelé občas spustí akci v systému omylem. Systém jim tedy musí umožnit jednoduchou cestu zpět.

Výsledná aplikace Z povahy webové aplikace je možnost vracet se na předchozí navštívenou stránku splněna. V aplikaci jsou všechny důležité akce, např. odesílání formuláře se splněným úkolem, doplněny dialogem, zda uživatel chce tuto akci opravdu provést, nebo je uživateli před finálním odesláním formuláře zobrazeno shrnutí, ve kterém vidí veškeré informace, které budou odeslány na server.

6.1.4 Konzistence a standardizace

Zadání Systém by měl dodržovat platformní a průmyslové konvence, aby uživatel nemusel přemýšlet, zda různá slova, situace nebo akce znamenají totéž.

Testování Bylo zjištěno, že se napříč aplikací objevují dva rozdílné termíny pro posádku. Někde bylo uvedeno *Tým*, někde *Posádka*.

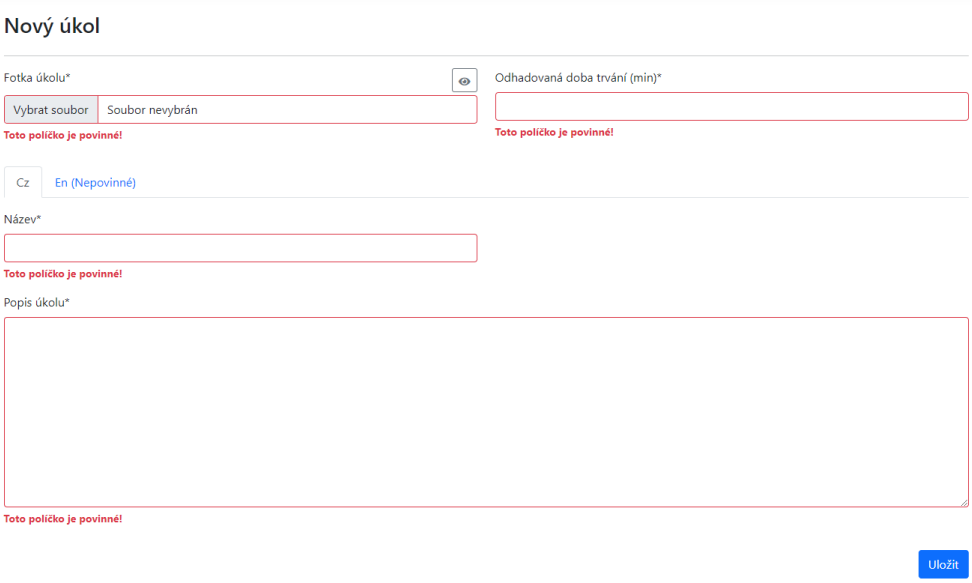
Výsledná aplikace Napříč celou aplikací mají komponenty se stejnou, či podobnou funkcionalitou konzistentní tvar i barvu a rozložení webových stránek podobného typu je vždy stejné. Terminologie napříč aplikací je konzistentní a koresponduje s realitou.

6.1.5 Prevence chyb

Zadání Systém by měl předcházet chybovým stavům, upozornit uživatele na to, pokud zapomene vyplnit povinné políčko formuláře. To by mělo být jasně zvýrazněno, aby se v první řadě uživatel vůbec nepokusil odeslat formulář bez jeho vyplnění. Dále jsou užitečné dialogy, ve kterých se systém uživatele zeptá, zda danou akci opravdu chce provést, např. při mazání obsahu.

Testování U některých políček ve formulářích chyběly hvězdičky pro identifikaci povinného údaje. Některá políčka měla špatně nastavené validátory, např. u telefonního čísla při registraci bylo vyžadováno mezi každými třemi číslicemi napsat mezeru, nebo při zadávání místa startu byla povinná minimální délka tři, přičemž existují i města, která v názvu obsahují pouze dvě písmena.

Výsledná aplikace Ve formulářích jsou povinné údaje znázorněny hvězdičkou vedle nadpisu daného políčka. Ihned po zadání údaje jsou nevalidní vstupy ukázány uživateli. Kontroluje se, zda je pole povinné, jestli je zadaná hodnota ve správném formátu, nebo zda je validní jeho logická hodnota.



Obrázek 6.2: Validace formulářů

6.1.6 Rozpoznání místo vzpomínání

Zadání Uživatel by neměl být nucen pamatovat si informace z jedné části systému pro použití druhé části. Informace potřebné pro používání určité části systému by měly být viditelné, nebo snadno vyhledatelné.

Testování Největším prohřeškem vůči tomuto bodu je nezobrazování splněných úkolů posádky. Posádka si v době testování musela pamatovat, který úkol již splnila, nebo si to musela dohledávat v detailech jednotlivých úkolů. Místo toho by mělo být v přehledu úkolů pro posádku viditelné, které úkoly již odevzdala a které ne.

Výsledná aplikace Uživateli k ovládání systému stačí navigační tlačítka a nadpisy obrazovek. Akce, které momentálně uživatel nemůže provézt, mu buďto nejsou vůbec zobrazeny, nebo jsou zobrazeny tak, aby bylo patrné, že danou akci není možné použít.

6.1.7 Flexibilní a efektivní použití

Zadání Pokročilí uživatelé systému by měli mít možnost využívat zkratek, které urychlují proces častých akcí.

Výsledná aplikace Tato aplikace neobsahuje žádné složité akce, které by bylo možné zjednodušit zkratkou pro pokročilé uživatele.

6.1.8 Estetický a minimalistický design

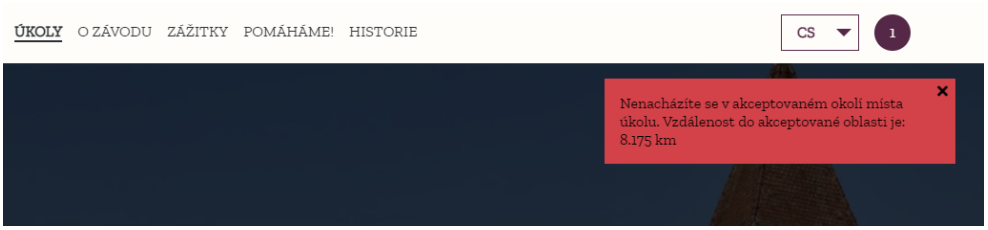
Zadání Obsahem stránek by neměli obsahovat informace, které jsou irelevantní nebo zřídka potřebné. Takové informace snižují viditelnost těch relevantních.

Výsledná aplikace V aplikaci je použito zobrazování různé úrovně detailu informací. Např. v seznamu posádek jsou zobrazena pouze jména, věk, povolání členů posádek a množství vybraných peněz. Na stránce detailu je navíc dostupný jejich popis a zážitky.

6.1.9 Pomoc uživatelům pochopit a vzpamatovat se z chyb

Zadání Chybové hlášky by měly být vyjádřeny v prostém jazyce (žádné chybové kódy), přesně definovat vzniklý problém a konstruktivně navrhnout jeho řešení.

Výsledná aplikace Při odeslání požadavku, který skončí chybou, je uživatel buď přesměrován na odpovídající stránku, např. stránka nenalezena, nebo je informován zprávou přímo na stránce, kde se momentálně nachází ve tvaru vyskakujícího okna v pravém horním rohu s textem, který uživateli poskytne informace o chybě, popřípadě s návrhem, jak postupovat dál.



Obrázek 6.3: Notifikace ohledně nevalidního pokusu o navštívení místa

6.1.10 Nápověda a dokumentace

Zadání Nejlepší je, pokud systém nevyžaduje žádné další vysvětlení. Může však být nutné poskytnout uživatelům dokumentaci, která jim pomůže dokončit své úkoly.

Výsledná aplikace Nápověda a dokumentace systému není momentálně vytvořena.

6.1.11 Shrnutí analýzy

Aplikace splňovala až na několik výjimek těchto deset bodů obstojně. Během testování bylo odhaleno značné množství chyb. Většina z nich však byla snadno opravitelná. Kromě prohřešků proti deseti bodům Nielsenovi heuristické analýzy bylo odhaleno několik chyb, které se týkaly fungování systému jako takového. Všechny objevené chyby byly opraveny před tím, než se přistoupilo k uživatelskému testování. Výsledkem tohoto testování je tedy aplikace, která svým uživatelským rozhraním splňuje z velké části všechny výše uvedené body a je připravena na uživatelské testování.

6.2 Uživatelské testování

Uživatelské testování je hlavní část testování webové aplikace, která vznikla v rámci této diplomové práce. Aplikace musí během uživatelského testování obstát v předem připravených situacích, které mají co nejpřesněji simulovat využití aplikace v reálném provozu. Dále také ověří, zda případy užití, které vznikly při návrhu aplikace, odpovídají reálnému používání aplikace uživateli. Výsledkem testování mohou být i návrhy na změny, které aplikaci pro uživatele zjednoduší či zpřehlední.

Při testování je také možné odhalit způsoby použití, které v rámci návrhu a implementace nebyly brány v potaz, a tudíž je zvýšená šance výskytu implementačních chyb.

V této sekci bude nejprve shrnuto, jakým způsobem byly uživatelské testy vytvořeny a co obsahovaly. Poté budou rozebrány průběhy testování s jednotlivými účastníky testování. Konec této sekce bude obsahovat shrnutí testování.

6.2.1 Struktura

Příprava na uživatelské testování začala vytvořením testovacího scénáře, který provede uživatele celým cyklem závodu. Testovací scénář byl navržen tak, aby obsáhl co nejvíce funkcionalit, které v reálném provozu budou jak běžní uživatelé, tak posádky a administrátoři využívat. Na druhou stranu nesměl být příliš obsáhlý, aby samotné testování aplikace nezabralo příliš dlouhou dobu. V takovém případě by testovaná osoba mohla začít ztrácet pozornost a zbytek testování by přestával být relevantní. Příliš krátký scénář by naopak neměl dostatečnou vypovídající hodnotu. Výsledný scénář se nachází v příloze B.

Jelikož výslednou aplikaci budou používat tři typy uživatelů, byl scénář rozdělen do tří částí: běžný uživatel, člen posádky a administrátor. Aby byl otestován celý cyklus závodu, nebylo možné testovat každou část odděleně. Testující tedy byly instruováni, aby každou část měli otevřenou v separátním anonymním okně v prohlížeči a při postupování scénářem mezi těmito okny přepínali.

Při plnění úkolů byla účastníkům testování poskytnuta nutná data pro vyplňování formulářů jako například souřadnice úkolů, názvy zařízení a některé delší texty, aby při pozdějším průchodu aplikací dávala data zobrazená na webových stránkách smysl. Dále jim byly poskytnuty přihlašovací údaje do systému a do platební brány PayPal a také číslo testovací kreditní karty pro platby přes platformu Stripe.

6.2.2 Průběh

Testování probíhalo celkem se čtyřmi uživateli. Se dvěma uživateli probíhalo testování osobně v jejich domácím prostředí. S ostatními probíhalo po internetu přes aplikaci Google Meets. Všichni zúčastnění využili pro testování vlastní pracovní stanice, aby se potencionálně přišlo na chyby spojené s infrastrukturou, např. rozlišení obrazovky nebo jiný internetový prohlížeč.

Každý uživatel byl před testováním do jisté míry seznámen se závodem, jeho cyklem, průběhem a pravidly, což bylo nutné zejména pro orientaci v administrační části, kde je znalost závodu očekávána. Dále už každý účastník testování postupoval sám podle připraveného scénáře.

V následujících odstavcích je popsáno testování s jednotlivými uživateli. Všechny osoby testovaly na stejné verzi aplikace, tudíž se při testování na některé chyby přišlo u více uživatelů. Pro větší přehlednost budou tyto chyby popsány pouze u jednoho z nich.

Ing. Markéta Hejná - 26let, UX designer

Paní Markéta Hejná jako UX designer neměla s orientací v rámci aplikace žádné větší problémy. Většina jejích zaváhání se projevila v rámci administrační konzole, což bylo způsobeno tím, že neznala souvislosti v závodu. Po krátkém vysvětlení, které nijak přímo neposkytovalo informace k tomu, jak postupovat v rámci scénáře, již komplikace pominuly.

Během testování bylo odhaleno, že při velkém oddálení mapy se body znázorňující polohu posádek objevují jinde, než by měly, a to o značné vzdálenosti. To indikovalo stejnou chybu i při zobrazování míst plnění úkolů. Další velkou objevenou chybou byla kvalifikace posádek. Uživatelské jméno posádky se tvořilo na základě roku, ve kterém se konal závod. Tím pádem pokud byl v jednom roce založen další závod, kvalifikace posádky do tohoto závodu skončila chybou, jelikož se aplikace pokusila přidat účet s duplikací uživatelského jména.

Menší chyby a postřehy testované osoby:

- Ve formuláři na založení závodu by bylo dobré při postupu na další krok posunout obrazovku na začátek formuláře

- V administrační části aplikace u hodnot v tabulkách přidat do závorky jednotky
- Chybějící značení povinného políčka u fotky během přidávání článku
- Ve formulářích, které jsou na více stran se při navigaci zpět se objeví indikátor validního vstupu, i když daná stránka ještě nebyla vyplněna
- V klientské části byl font formulářů malý, skoro nečitelný

Bc. Jan Míšek 29let, kontrolor nedestruktivní metodou

Pro pana Míška nebyla orientace v klientské části aplikace žádným problémem. Všechny s tím související úkoly byl schopen bez větších obtíží splnit. Kde se však začaly objevovat problémy, byla administrační část aplikace. Jelikož v ničem takovém nikdy nedělal, několik minut mu zabralo se zorientovat. Poté, co si v rychlosti proklikal všechny sekce, aby si je spojil s informacemi o závodu, používal již administrační prostředí s větší sebejistotou.

Největší problém se vyskytl u kvalifikace posádek do hlavního závodu, kdy pro pana Míška bylo zmateční, že kvalifikační posádky se nacházejí v sekci kvalifikační závod pod formulářem s informacemi o závodu a ne v sekci posádky. Uvedl, že by bylo lepší v postranním menu nějak lépe oddělit hlavní a kvalifikační závod, aby bylo zřetelnější, kam má uživatel upřít pozornost. Dále se vyskytla chyba během přidávání úkolu, kdy po vložení fotografie ve vysokém rozlišení velkých rozměrů se oblast pro ořezávání roztáhne mimo modální okno a není možné tak kliknout na navigační tlačítka.

Menší chyby a postřehy testované osoby:

- Tabulky v administrační sekci obsahují sloupec s anglickým nadpisem *Action*
- Ve formulářích, které jsou na více stránek, by bylo dobré, kdyby mezi stránkami šlo přepínat pomocí navigační lišty a ne pouze pomocí navigačních tlačítek
- U detailu odevzdaného úkolu chybí tlačítko na zamítnutí, je pouze na schválení
- Při pokusu odeslat platbu přes PayPal s nevyplněným formulářem se nezobrazí chybová hláška

Ing. Jan Štuiber - 27let, geodet

Pan Štuiber stejně jako předchozí uživatelé neměl problémy orientovat se v klientské části. První problém nastal při úkolu spustit administrační prostředí.

Jelikož pan Štuiber nikdy nic takového nepoužil, tak chvíli tápal, ale poté se přes ikonku profilu do administračního prostředí bez pomoci dostal. Průběh testování v administrační části byl velice podobný jako u pana Míška. Několik minut trvalo, než si spojil vědomosti o závodu s jednotlivými sekcemi aplikace, poté již procházel scénářem bez větších potíží. Stejně jako u pana Míška byl problém s nejasností ohledně kvalifikačního závodu a rozdělení hlavního a kvalifikačního závodu v postranním menu považoval za dobré řešení problému.

Během testování bylo odhaleno, že bonusové body přičtené posádce se nepřipočítávají do celkového počtu bodů posádky. Dále bylo zjištěno, že pokud je závod ukončen, posádka se stále může přihlásit a přidávat zážitky.

Menší chyby a postřehy testované osoby:

- V základních informacích o závodu se souřadnice zadávají v opačném pořadí než v přidání úkolu. Mělo by to být jednotné a nejdříve zeměpisná šířka a poté zeměpisná délka

Emil Patta - 35let, učitel

Posledním uživatelem, který vytvořenou aplikaci testoval, byl Emil Patta. Toto testování probíhalo jinak než u ostatních uživatelů, protože pan Patta je jedním z organizátorů závodu. Jako organizátor byl u sestavování požadavků na aplikaci a probíhaly s ním konzultace ohledně vývoje. Vzhledem k těmto okolnostem mu připravený scénář nedělal žádné potíže.

Více než na chyby bylo toto testování zaměřeno na to, zda byly splněny veškeré požadavky, které při zadávání práce očekával a zda všechny procesy fungují tak, jak mají. V rámci testování bylo zjištěno, že některé kroky, které byly přítomny u vytváření závodu, by tam být nemusely. Jedná se o vytváření úkolů a psaní článku ohledně Konta Bariéry. Úkoly jsou sice nepovinné, ale neměly by tam být vůbec, protože žádný úkol se nezveřejňuje při vytváření závodu, ale až týden před závodem. Stejně je tomu i u článku ohledně Konta Bariéry. Závod se vytváří před tím, než se vůbec začne s Konto Bariéry jednat. Tento výstup byl velice užitečný, neboť tato informace při návrhu aplikace chyběla.

Pan Patta měl z aplikace dobrý pocit a splňuje jeho očekávání. Administrační prostředí je srozumitelné, přehledné. Dále se mnou konzultoval některé okrajové situace, jako například kvalifikace více než dvanácti posádek, zda je na to aplikace připravená.

6.2.3 Výsledky uživatelského testování

V rámci testování byla aplikace otestována nejprve čtyřmi experty. Díky tomuto testování byla odhalena většina nejasností v uživatelském rozhraní po technické stránce. Chyby nalezené v testováni experty byly opraveny před uživatelským testováním.

Uživatelského testování se zúčastnili další čtyři lidé. Všichni testovaní patří do mladší věkové skupiny, jelikož to více vystihuje cílovou skupinu, na kterou se aplikace zaměřuje. Zúčastnění byli různého vzdělání, pohlaví i míry dovedností a zkušeností s používáním podobných webových aplikací. Díky tomuto testování bylo odhaleno několik závažných chyb, několik drobných chyb a byly navrženy změny, které aplikaci do produkčního nasazení značně vylepší.

Největší problém, který se opakoval u většiny testujících, byla zmatečně umístěná sekce s kvalifikačním závodem v administrační sekci. Sice bylo řečeno, že po prvním úspěšném kvalifikování už by s tím dále problém neměli, avšak bude lepší sekci hlavního a kvalifikačního závodu přehledněji rozdělit.

Všechny chyby, které byly při testování odhaleny byly opraveny, a většina návrhů na zlepšení ze strany testujících byla implementována.

Kapitola 7

Nápady pro další rozvoj

Výsledkem této práce je funkční prototyp, který obsahuje všechny náležitosti potřebné k pořádání závodu. Díky návrhu aplikace je v budoucnu možné jednoduše přidávat nové funkcionality a tím aplikaci nadále rozšiřovat. V této kapitole je popsáno několik rozšíření aplikace, která by ji mohla vylepšit.

7.1 Hledání spolujezdce

Pro účast v závodu je zapotřebí dvou účastníků v posádce. Ne každý má však ve svém okolí někoho, kdo by s ním tento závod podstoupil. Proto by bylo dobré pro takové lidi vytvořit místo, kde by o sobě mohli napsat nějaké detailnější informace a na základě těch si najít spolujezdce do závodu.

7.2 Komunikace mezi posádkou a organizátory

Momentálně posádky komunikují s organizátory přes sociální sítě nebo přes SMS. Dobrým rozšířením by byl komunikační kanál ve formě okna pro chat, kde by komunikace probíhala. Organizátoři by tedy měli veškeré komunikační kanály na jednom místě a organizace závodu by pro ně byla pohodlnější.

7.3 Komunikace mezi posádkou a návštěvníky stránek

Bylo by dobré návštěvníkům stránek poskytnout možnost vyjádřit podporu posádkám i jiným způsobem, než anonymním příspěvkem. Toho by se dalo docílit pomocí přímých zpráv posádkám, nebo formou komentářů, např. pod zážitky posádky nebo přímo pod jejich profil.

7.4 Zaznamenání trasy jednotlivých posádek

Každá posádka pro svojí cestu voli jinou trasu. Proto by mohlo být zajímavé, ukládat trasu posádky, která by byla zobrazena na mapě v detailu posádky. Návštěvníci webových stránek by tak měli lepší představu, kudy posádka jela a která místa po své cestě navštívila.

Závěr

Cílem této práce bylo vytvořit a otestovat prototyp webové aplikace pro neziskovou organizaci Světem stopem, která by v budoucnu nahradila jejich stávající, pro organizátory nevyhovující webové stránky. Jednalo se o aplikaci spravující závod Mistrovství ČR v autostopu.

Nejprve byla provedena analýza stávajícího řešení. Při této analýze byla odhalena spousta nedostatků, které bylo potřeba eliminovat v nově navrhované aplikaci. Na základě této analýzy a rozhovorech s organizátory závodu byly definovány funkční a nefunkční požadavky, které výsledná aplikace musí splňovat. Dále byly v kapitole Analýza vybrány technologie, pomocí kterých se aplikace implementovala.

Návrhu aplikace a jejího uživatelského rozhraní se věnuje kapitola 3. V této kapitole bylo navrženo uživatelské rozhraní. Wireframes společne s Hi-Fi prototypem klientské části byly dodány organizátory závodu. Wireframes administrační části byly zhotovené autorem této diplomové práce. Na základě těchto prostředků bylo s organizátory diskutováno splnění požadavků na aplikaci před její implementací.

Důležité a zajímavé části implementace, která nakonec probíhala ve frameworku Django programovacího jazyka Python, jsou popsány v kapitole 4. Po dokončení implementace bylo potřeba prototyp uživatelsky otestovat. Proto byla aplikace nasazena do cloudu pomocí služby AWS a je dostupná na adrese www.dev-mcrautostop.cz. Nasazení produkční verze bude probíhat ve stejném prostředí, proto nasazení sloužilo také pro vyzkoušení potřebných AWS služeb. Podrobnostem nasazení je věnována kapitola 5 a jejímu testování kapitola 6. Na konci práce bylo diskutováno, jaká rozšíření by se v budoucnu mohla do aplikace dodat pro její zkvalitnění.

Všechny cíle, které vycházely jak ze zadání této práce, tak z funkčních a nefunkčních požadavků na výslednou aplikaci, byly splněny. Aplikace poskytuje uživatelům přehledné webové stránky s informacemi o závodu Mistrovství ČR v autostopu a administrátorům závodu kvalitní administrační prostředí, ve kterém mohou závod spravovat. Aplikace nebyla nasazena do produkčního

prostředí, protože organizátoři chtějí aplikaci nejprve vyzkoušet na cvičném závodě, který nebylo možné kvůli vládním opatřením v době psaní práce zorganizovat.

Literatura

[1] Valuy, S.: A Comparison of Single-Page and Multi-Page Applications. červen 2020, [Online]. Dostupné z: https://dzone.com/articles/the-comparison-of-single-page-and-multi-page-appli

[2] Docker, Inc.: *What is a Container? [online].* [cit. 2021-4-7]. Dostupné z: https://www.docker.com/resources/what-container

[3] Nadace Charty 77: *Konto Bariéry [online].* [cit. 2021-4-6]. Dostupné z: https://www.kontobariery.cz/O-nas/Konto-Bariery

[4] Světem stopem z. s.: *Mistrovství ČR v autostopu [online].* [cit. 2022-1-1]. Dostupné z: https://www.mcrautostop.cz

[5] Techopedia Inc.: *Content Management System (CMS) [online].* [cit. 2021-4-7]. Dostupné z: https://www.techopedia.com/definition/24075/content-management-system-cms

[6] AltexSoft: *Functional and Nonfunctional Requirements: Specification and Types [online].* [cit. 2021-4-9]. Dostupné z: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/

[7] Stack Exchange Inc: *Most Popular Technologies [online].* [cit. 2021-4-10]. Dostupné z: https://insights.stackoverflow.com/survey/2020#most-popular-technologies

[8] Python Software Foundation: *What is Python? Executive Summary [online].* [cit. 2021-4-10]. Dostupné z: https://www.python.org/doc/essays/blurb/

[9] JetBrains s.r.o. and Python Software Foundation: *Python Developers Survey 2020 Results [online].* [cit. 2021-4-10]. Dostupné z: https://www.jetbrains.com/lp/python-developers-survey-2019/

[10] Makai, M.: WSGI Servers. *Full Stack Python [online]*, květen 2014, [cit. 2021-4-12]. Dostupné z: https://www.fullstackpython.com/wsgi-servers.html

[11] Tutorials Point India Limited: *Flask – Overview [online]*. [cit. 2021-4-7]. Dostupné z: https://www.tutorialspoint.com/flask/flask_overview.htm

[12] Tutorials Point India Limited: *Django - Basics [online]*. [cit. 2021-4-7]. Dostupné z: https://www.tutorialspoint.com/django/django_basics.htm

[13] Imperva: *SQL (Structured query language) Injection [online]*. [cit. 2022-1-3]. Dostupné z: https://www.imperva.com/learn/application-security/sql-injection-sqli/

[14] Django Software Foundation and individual contributors: *Databases [online]*. [cit. 2021-4-7]. Dostupné z: https://docs.djangoproject.com/en/3.2/ref/databases/

[15] The PostgreSQL Global Development Group: *About [online]*. [cit. 2021-4-7]. Dostupné z: https://www.postgresql.org/about/

[16] Google: *Google Maps Platform [online]*. [cit. 2021-4-7]. Dostupné z: https://cloud.google.com/maps-platform

[17] Django Software Foundation and individual contributors: *Deploying Django [online]*. [cit. 2021-4-7]. Dostupné z: https://docs.djangoproject.com/en/3.2/howto/deployment/

[18] Skólski, P.: Single-page application vs. multiple-page application. *Neoteric [online]*, prosinec 2016, [cit. 2021-4-12]. Dostupné z: https://neoteric.eu/blog/single-page-application-vs-multiple-page-application/?utm_source=medium.com&utm_medium=social&utm_content=neo&utm_campaign=blog

[19] Monocubed: *5 Best Single Page Application Frameworks To Consider For Web Apps [online]*. [cit. 2022-1-1]. Dostupné z: https://www.monocubed.com/top-single-page-application-frameworks/

[20] Django Software Foundation and individual contributors: *Model [online]*. [cit. 2021-4-7]. Dostupné z: https://docs.djangoproject.com/en/3.2/topics/db/models/

[21] Real Python: *Django Migrations: A Primer [online]*. [cit. 2021-4-7]. Dostupné z: https://realpython.com/django-migrations-a-primer/

[22] Django Software Foundation and individual contributors: *Making queries [online]*. [cit. 2021-4-7]. Dostupné z: `https://docs.djangoproject.com/en/3.2/topics/db/queries/`

[23] Johnson, A.: Django and the N+1 Queries Problem. *scoutapm [online]*, září 2020, [cit. 2021-4-12]. Dostupné z: `https://scoutapm.com/blog/django-and-the-n1-queries-problem`

[24] Thagana, K.: Introduction to Django Signals. *pluralsight [online]*, říjen 2020, [cit. 2021-4-12]. Dostupné z: `https://www.pluralsight.com/guides/introduction-to-django-signals`

[25] Django Software Foundation and individual contributors: *The Django template language [online]*. [cit. 2021-4-7]. Dostupné z: `https://docs.djangoproject.com/en/3.2/ref/templates/language/`

[26] Grudl, D.: Escapování – definitivní příručka. *phpfashion [online]*, květen 2009, [cit. 2021-4-12]. Dostupné z: `https://phpfashion.com/escapovani-definitivni-prirucka#comment-18832`

[27] Amazon Web Services, Inc. or its affiliates: *Amazon Simple Email Servicee [online]*. [cit. 2021-4-7]. Dostupné z: `https://aws.amazon.com/ses/`

[28] Amazon Web Services, Inc. or its affiliates: *Amazon SES pricing [online]*. [cit. 2021-4-7]. Dostupné z: `https://aws.amazon.com/ses/pricing/`

[29] PayPal: *PayPal sandbox testing guide [online]*. [cit. 2021-4-7]. Dostupné z: `https://developer.paypal.com/docs/api-basics/sandbox/`

[30] Stripe: *Pricing [online]*. [cit. 2022-1-1]. Dostupné z: `https://stripe.com/en-cz/pricing`

[31] PayPal: *Poplatky za používání služby PayPal pro obchodníky [online]*. [cit. 2022-1-1]. Dostupné z: `https://www.paypal.com/cz/webapps/mpp/merchant-fees#fixed-fees-commercialtrans`

[32] ggspatial: *Distance on a sphere: The Haversine Formula [online]*. [cit. 2021-4-7]. Dostupné z: `http://www.ggspatial.co.uk/distance-on-a-sphere-the-haversine-formula/`

[33] PROJECT OSRM: *Open Source Routing Machine [online]*. [cit. 2021-4-7]. Dostupné z: `https://github.com/Project-OSRM/osrm-backend`

[34] Both, D.: An introduction to swap space on Linux systems. *opensource [online]*, březen 2020, [cit. 2021-4-12]. Dostupné z: `https://opensource.com/article/18/9/swap-space-linux-systems`

[35] Covassi, G.: The 6 Benefits of Docker Container. *kiratech [online]*, prosinec 2018, [cit. 2021-4-12]. Dostupné z: https://www.kiratech.it/en/blog/the-6-benefits-of-docker-container

[36] Docker, Inc.: *Dockerfile reference [online]*. [cit. 2021-4-7]. Dostupné z: https://docs.docker.com/engine/reference/builder/

[37] Docker, Inc.: *Use multi-stage builds [online]*. [cit. 2022-1-5]. Dostupné z: https://docs.docker.com/develop/develop-images/multistage-build/

[38] Docker, Inc.: *Docker compose [online]*. [cit. 2021-4-7]. Dostupné z: https://docs.docker.com/compose/

[39] Sullivan, E.: pay-as-you-go cloud computing (PAYG cloud computing). *TechTarget [online]*, březen 2015, [cit. 2021-4-12]. Dostupné z: https://searchstorage.techtarget.com/definition/pay-as-you-go-cloud-computing-PAYG-cloud-computing

[40] Amazon Web Services, Inc. or its affiliates: *About AWS [online]*. [cit. 2021-4-7]. Dostupné z: https://aws.amazon.com/about-aws/

[41] Amazon Web Services, Inc. or its affiliates: *Amazon EC2 [online]*. [cit. 2021-4-7]. Dostupné z: https://aws.amazon.com/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc

[42] Amazon Web Services, Inc. or its affiliates: *Amazon Elastic Block Store [online]*. [cit. 2021-4-7]. Dostupné z: https://aws.amazon.com/ebs/?ebs-whats-new.sort-by=item.additionalFields.postDateTime&ebs-whats-new.sort-order=desc

[43] Amazon Web Services, Inc. or its affiliates: *Amazon EBS volume types [online]*. [cit. 2022-1-1]. Dostupné z: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html?icmpid=docs_ec2_console57

[44] Nielsen, J.: 10 Usability Heuristics for User Interface Design. *Nielsen Norman Group [online]*, listopad 2020, [cit. 2021-4-12]. Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/

Příloha **A**

Seznam použitých zkratek

GPS Global Positioning System

API Application Programming Interface

DMS Dárcovská SMS

CSS Cascading Style Sheets

CMS Content Management System

URL Uniform Resource Locator

HTTP Hypertext Transfer Protocol

ORM Object Relational Mapping

DRY Don't Repeat Yourself

LESS Leaner Style Sheets

SASS Syntactically Awesome Style Sheets

ACID Atomicity Consistency Isolation Durability

XML eXtensible Markup Language

OSRM Open Source Routing Machine

AWS Amazon Web Services

EC2 Elastic Compute Cloud

EBS Elastic Block Storage

SES Simple Email Service

MPA Multi Page Application

SPA Single Page Application

AJAX Asynchronous JavaScript and XML

SEO Search Engine Optimalization

MVT Model View Template

MVC Model View Controller

DTL Django Template Language

JSON JavaScript Object Notation

IPN Instant Payment Notification

IOPS Input / Output Per Second

Příloha B

Scénář uživatelského testování

1. **Založení závodu (administrátor)**

   Nastal čas na další ročník závodu Mistrovství ČR v autostopu. Jako administrátor je Vaším úkolem tento závod založit.

   - Zapněte administrační prostředí
   - Založte nový závod

2. **Registrace do kvalifikačního závodu (nepřihlášený uživatel)**

   Na sociálních sítích jste se dozvěděl/a, že existuje závod v autostopu, který je zároveň charitativní akce. To ve Vás vzbudilo zájem a chcete se dozvědět více.

   - Zjistěte více informací o závodu Mistrovství ČR v autostopu
   - Podívejte se, jak probíhal závod v minulých letech
   - Registrujte se do kvalifikačního závodu

3. **Kvalifikace posádek (administrátor)**

   Několik měsíců po založení závodu se konala kvalifikace. Nyní je na čase kvalifikovat posádky, které se v kvalifikačním závodě umístily na předních příčkách

   - Najděte posádku, která se kvalifikovala (jedná se o posádku, která byla založena v druhém kroku scénáře) a zkontrolujte, zda jsou informace zadané při registraci v pořádku
   - Změňte emailovou adresu prvního člena na *mcra.test.user@gmail.com*
   - Kvalifikujte posádku do hlavního závodu

4. **Přidání úkolu a článku závodu (administrátor)**

   Začátek závodu se pomalu blíží a nastal čas zveřejnit úkoly a trasu závodu. Také jste se dohodli s Konto Bariéry na osobě, které se bude v rámci závodu pomáhat.

   - Přidejte úkol
   - Přidejte místa, která posádky musejí během závodu navštívit
   - Napište článek o tom, na koho se budou tento roční vybírat peníze

5. **Start závodu (administrátor)**

   Nastal čas odstartovat závod a od společnosti GPS Dozor přišla sledovací zařízení pro posádky.

   - Přidejte do systému sledovací zařízení, která Vám přišla
   - Zkontrolujte, zda každá posádka má přidělené sledovací zařízení. Pokud ne, nějaké mu přidělte
   - Odstartujte závod

6. **Odevzdání úkolu (posádka)**

   Závod byl odstartován a vy musíte vymyslet, jakým směrem se vydat.

   - Prohlédněte si, jaké úkoly musíte během své cesty splnit a na jaká místa se musíte vydat
   - Podělte se s návštěvníky webových stránek o svém zážitku z prvního dne závodu
   - Během prvního dne se Vám podařilo splnit i první úkol. Dejte administrátorům vědět, že jste ho splnili

7. **Schválení příspěvků (administrátor)**

   První den se chýlí ke konci. Je na čase se podívat, jak si posádky v závodě vedou a zda už se některým z nich přihodila nějaká zajímavá situace a zda se již někomu podařilo splnit nějaký z úkolů.

   - Zkontrolujte, zda přibyly nějaké nové zážitky nebo zda nějaká posádka nesplnila některé z úkolů
   - U žádosti o schválení nového zážitku zkontrolujte, zda je jeho obsah zveřejnitelný
   - Přeložte obsah příběhu do angličtiny, aby si ho mohli přečíst i zahraniční návštěvníci webových stránek
   - Schvalte zážitek

- Stejný proces udělejte i s novou žádostí o splnění úkolu

8. **Přispění posádce (nepřihlášený uživatel)**

   Dozvěděli jste se, že se Váš kamarád účastní závodu Mistrovství ČR v autostopu. Zajímá Vás tedy, jak si v závodě vede a také byste ho chtěli podpořit.

   - Prohlídněte si, jaké posádky se účastní závodu
   - Po zjištění, že Váš kamarád je v posádce č. 1 se podívejte, kde se momentálně tato posádka nachází
   - Na sociálních sítích jste si všimli, že se Vašemu kamarádovi stala zajímavá věc. Podívejte se, jaké situace během jeho cestování zažil
   - Přispějte mu nějaký finanční obnos

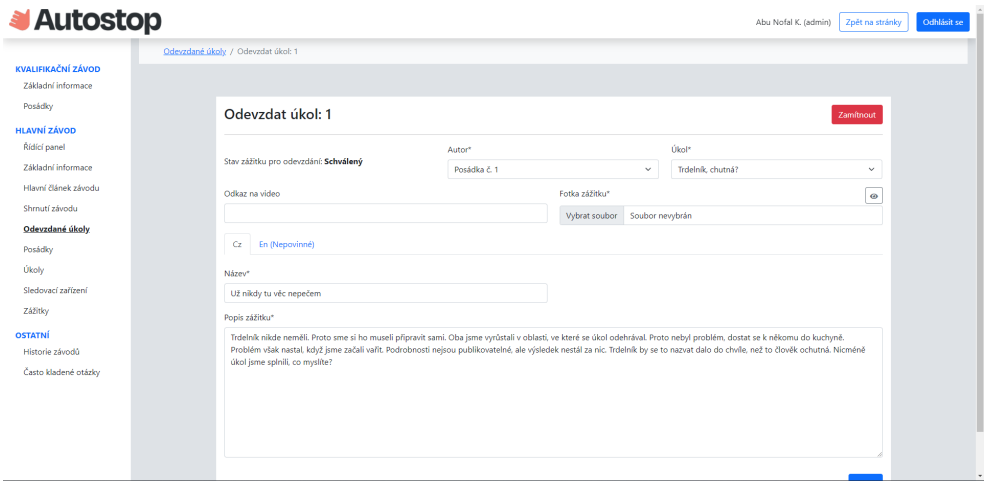
9. **Ukončení závodu (administrátor)**

   Závod je již u konce. Poslední posádky dojíždějí do cíle a přichází vyhlašování vítězů a předávání cen.

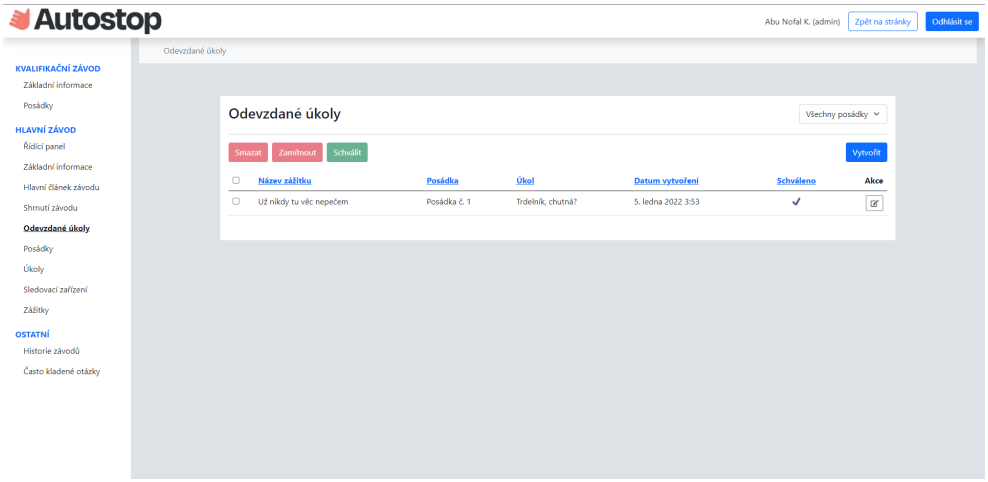
   - Ukončete závod
   - Napište článek o tom, jak se tento ročník vydařil, kdo vyhrál a jaké ceny si účastníci závodu odnesli

Příloha C

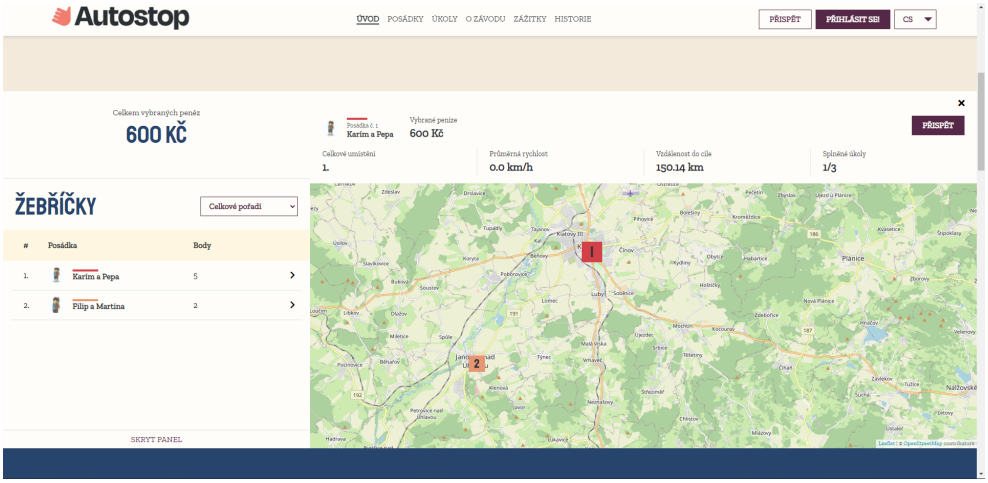
Ukázky výsledné aplikace



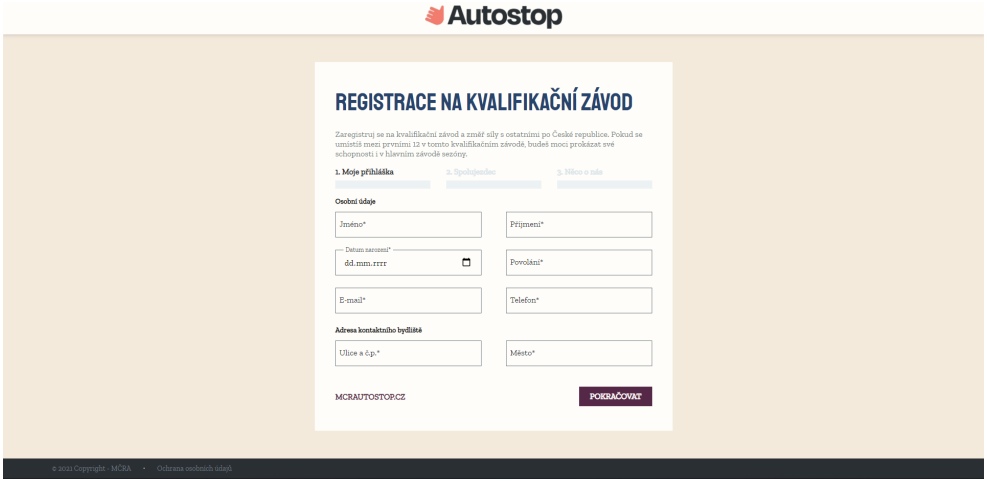
Obrázek C.1: Výsledná aplikace: Detail odevzdání úkolu v administrační části



Obrázek C.2: Výsledná aplikace: Seznam odevzdaných úkolů v administrační části



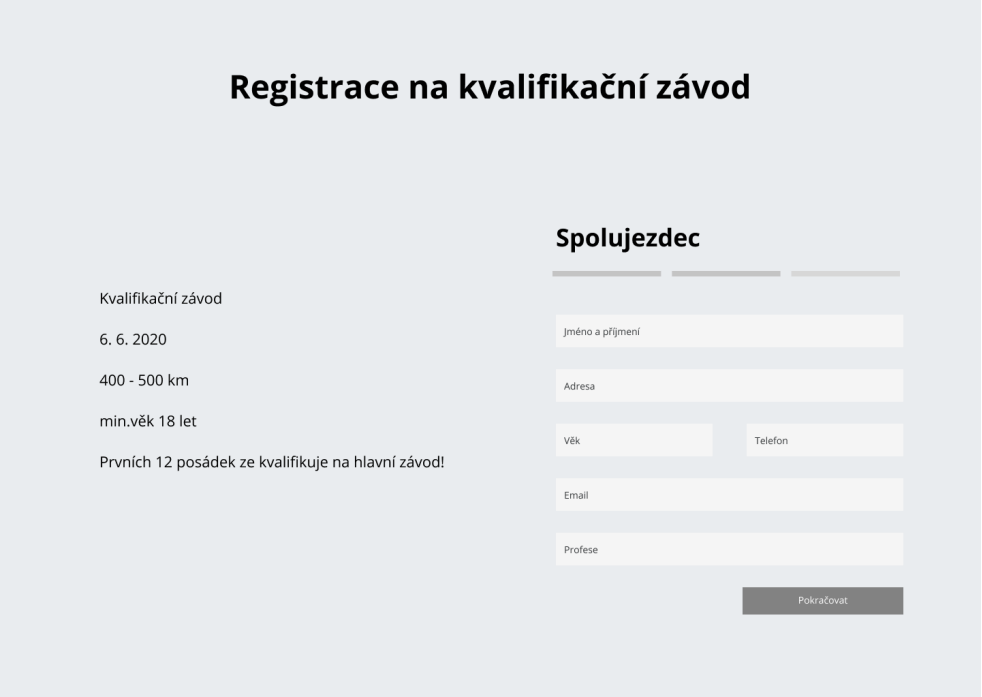
Obrázek C.3: Výsledná aplikace: mapa s žebříčky posádek na úvodní stránce



Obrázek C.4: Výsledná aplikace: registrační formulář

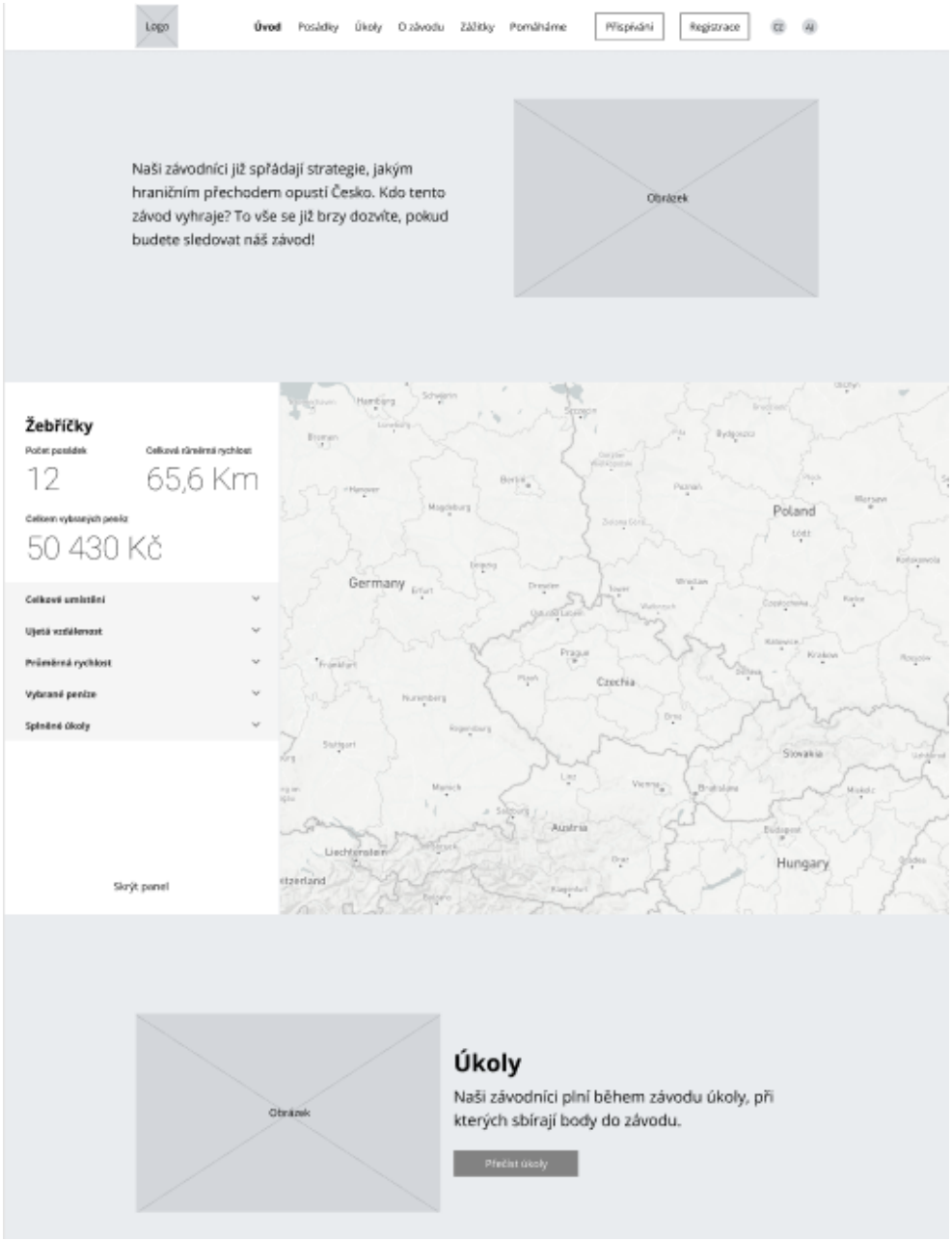
Příloha D

Ukázky Lo-fi a Hi-Fi prototypu dodaných organizátory

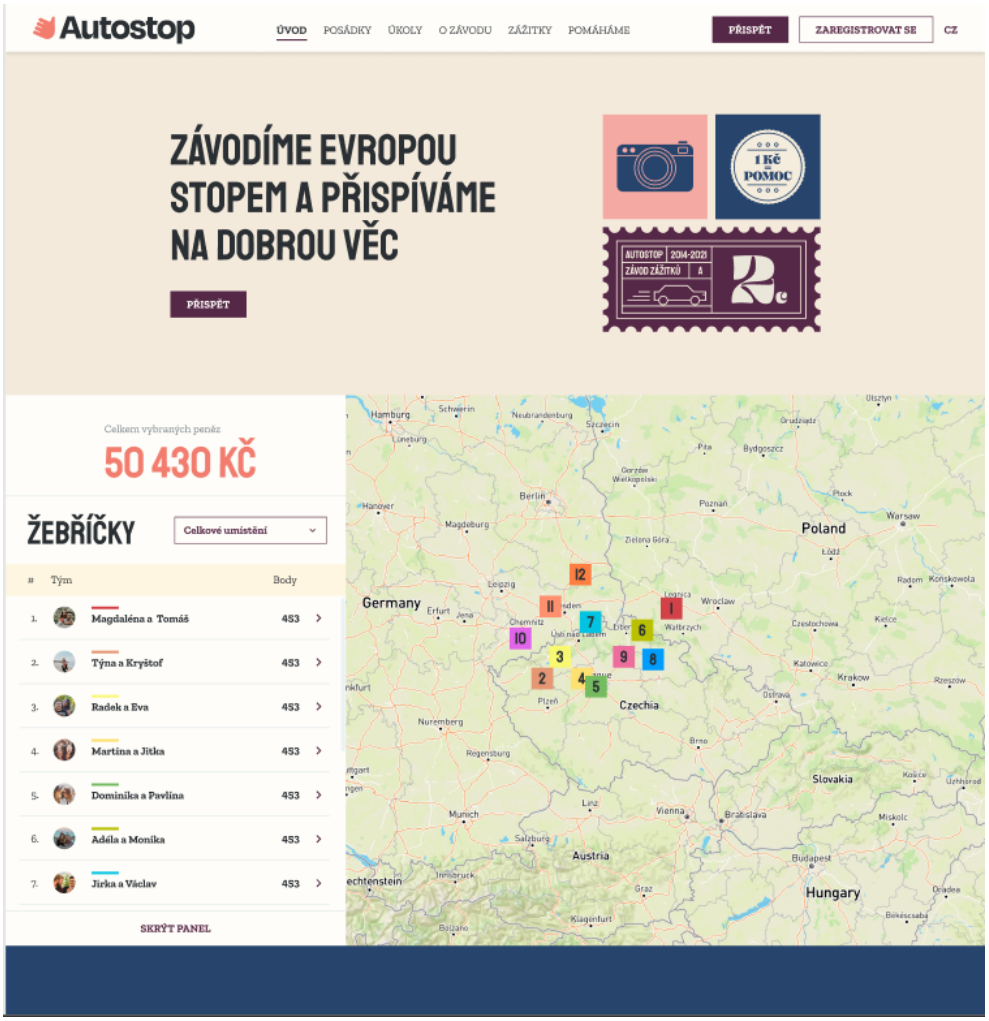


Obrázek D.1: Lo-Fi prototyp: Registrace

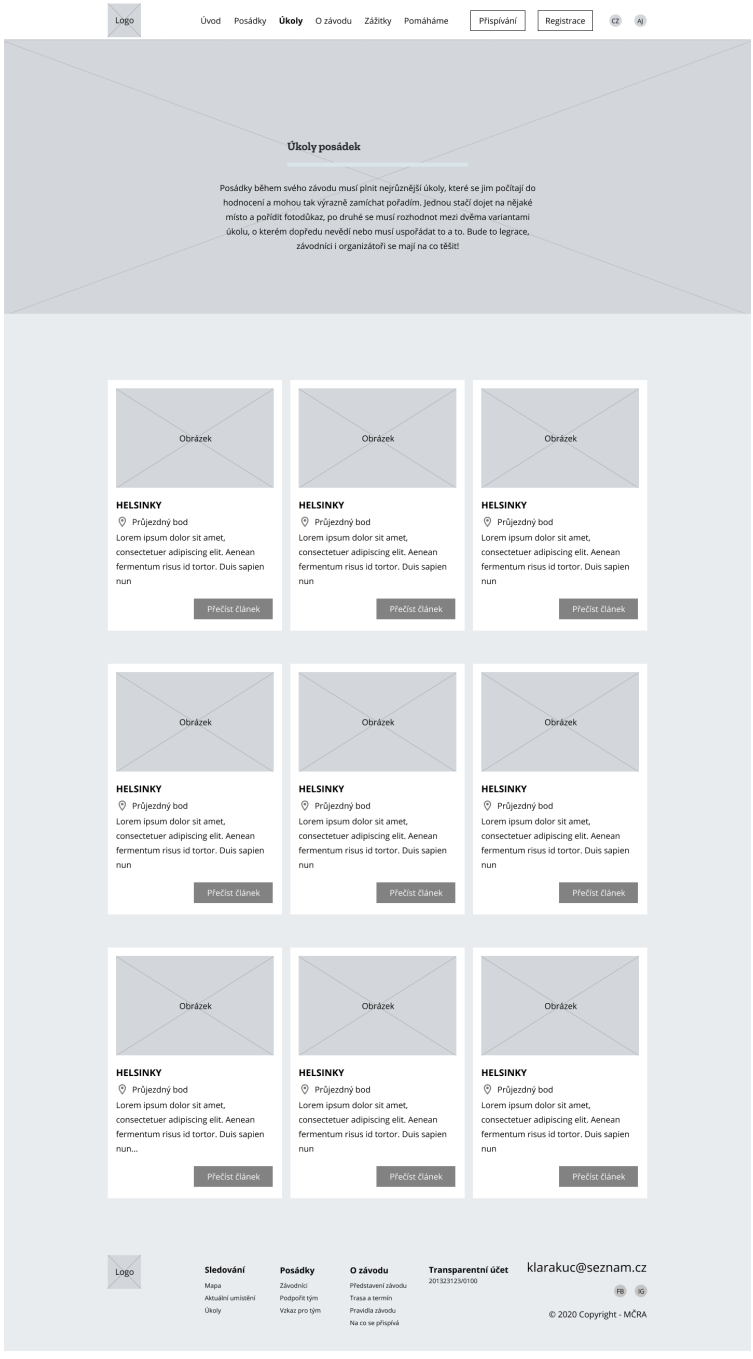
Obrázek D.2: Hi-Fi prototyp: Registrace



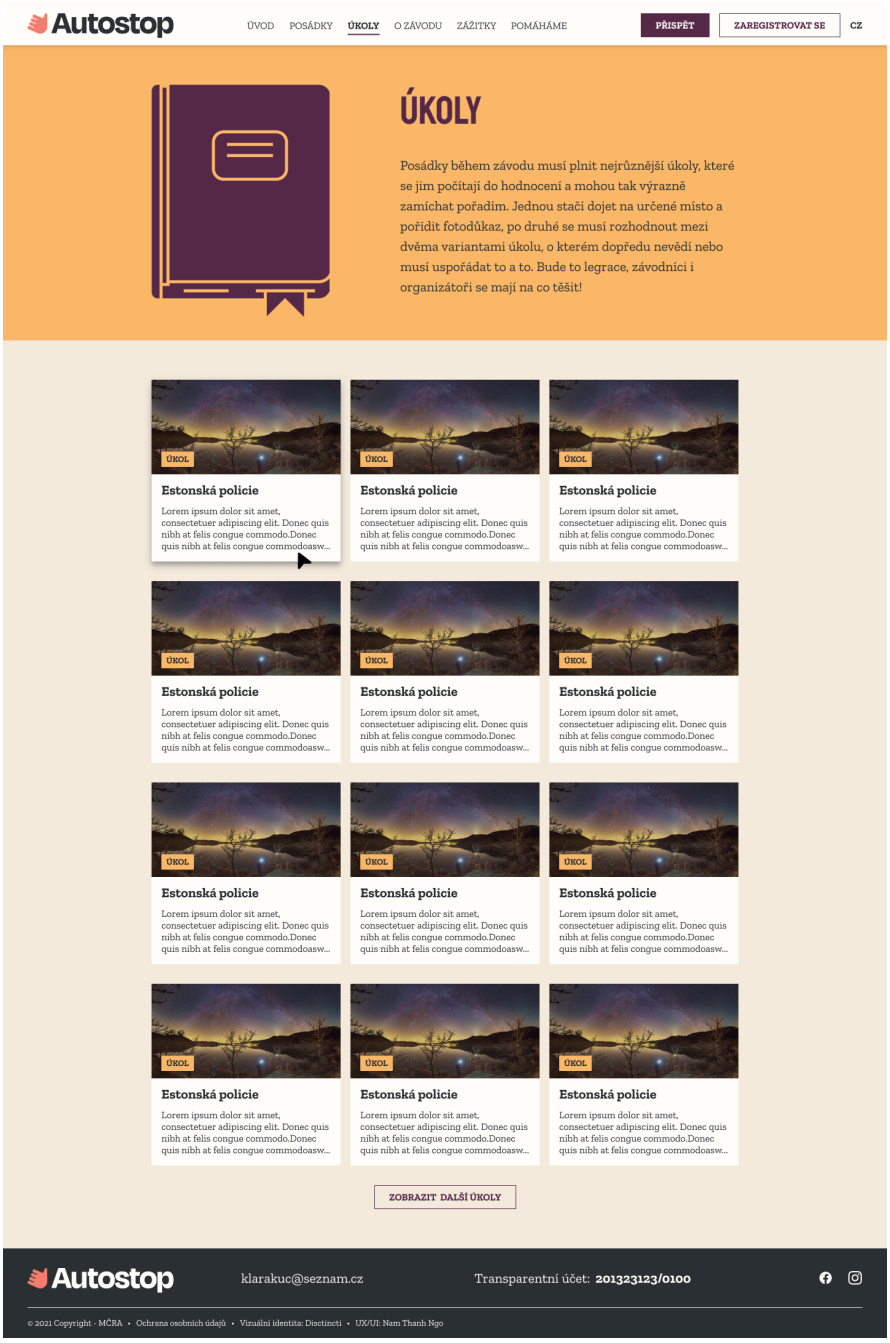
Obrázek D.3: Lo-Fi prototyp: Mapa



Obrázek D.4: Hi-Fi prototyp: Mapa



Obrázek D.5: Lo-Fi prototyp: Úkoly



Obrázek D.6: Hi-Fi prototyp: Úkoly

Příloha E

Obsah přiloženého média

- `README.md` ..... stručný popis obsahu CD
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `mcra` ..... zdrojové kódy implementace
  - `osrm` ..... předzpracovaná data pro OSRM směrovací stroj pro ČR
  - `hifi.pdf` ..... Hi-Fi prototyp aplikace dodaný organizátory
  - `lofi.pdf` ..... Lo-Fi prototyp aplikace dodaný organizátory
  - `lofi_admin.pdf` ..... Lo-Fi prototyp administračního prostředí
- `text`
  - `thesis.pdf` ..... text práce ve formátu PDF