VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO DISPEČINK SOUKROMÉ SANITNÍ SLUŽBY

SYSTEM FOR A DISPATCHING OF A PRIVATE AMBULANCE SERVICE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **LUKÁŠ PROKOP**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Prokop Lukáš**
Program: Informační technologie
Název: **Systém pro dispečink soukromé sanitní služby**
System for a Dispatching of a Private Ambulance Service
Kategorie: Informační systémy

Zadání:

1. Seznamte se s dostupnými prostředími pro tvorbu webových a mobilních aplikací a se základy práce s GPS daty.
2. Analyzujte požadavky na systém pro dispečink soukromé sanitní služby, který se bude skládat z webové a mobilní části. Mobilní aplikace bude řidiči zobrazovat přidělené trasy a na server bude odesílat informaci o poloze vozidla. Webová aplikace umožní dispečerovi plánovat další jízdy na základě polohy sanitek a dalších parametrů. Prostudujte existující řešení v této oblasti.
3. Navrhněte aplikaci splňující požadavky z předchozího bodu.
4. Po konzultaci s vedoucím navrženou aplikaci implementujte a ověřte její funkčnost s využitím vhodného vzorku dat.
5. Zhodnoťte dosažené výsledky a diskutujte další možné pokračování v tomto projektu.

Literatura:

- CASTLEDINE, Earle. Vytváříme mobilní web a aplikace pro chytré telefony a tablety. 1. vyd. Brno: Computer Press, 2013. 288 s. ISBN 978-80-251-3763-5
- PILGRIM, Mark. Ponořme se do HTML5: CZ.NIC, z.s.p.o, 2015. ISBN 978-80-905802-6-8.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 28. října 2018

Abstrakt

Tato práce se zabývá návrhem a implementací webové aplikace pro dispečink soukromých sanitních služeb a mobilní aplikace pro řidiče sanitních vozů. Webová aplikace umožňuje dispečerovi vytvářet nové objednávky, efektivně plánovat trasy a přiřazovat je sanitním vozům. Mobilní aplikace slouží k odesílání polohy sanitních vozů a zobrazování přidělených tras. Na počátku práce byla provedena analýza způsobů vývoje webových a mobilních aplikací, analýza práce s GPS daty, následovala analýza požadavků na systém a zhodnocení existujících řešení. Na základě získaných informací byla navržena a následně implementována webová i mobilní aplikace. V realizovaném systému je kladen důraz na jednoduchost a intuitivní ovládání.

Abstract

This thesis deals with design and implementation of web application for a dispatching of a private ambulance service and mobile application for ambulance drivers. The web application allows the dispatcher to create new orders, plan routes effectively and assign them to ambulance vehicles. The mobile application is used for sending ambulance vehicle's position and displaying assigned routes. At the beginning of this thesis, the analysis of methods of web and mobile applications development, analysis of work with GPS data, analysis of system requirements and evaluation of existing solutions were performed. Web and mobile applications were designed and implemented on the basis of the information obtained. The implemented system emphasizes simplicity and intuitive control.

Klíčová slova

informační systém, sanitní služba, dispečink, GPS, sledování polohy, webová aplikace, mobilní aplikace, HTML, CSS, JavaScript, PHP, AJAX, Bootstrap, Framework7, jQuery, Cordova

Keywords

information system, ambulance service, dispatching, GPS, location tracking, web application, mobile application, HTML, CSS, JavaScript, PHP, AJAX, Bootstrap, Framework7, jQuery, Cordova

Citace

PROKOP, Lukáš. *Systém pro dispečink soukromé sanitní služby.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Systém pro dispečink soukromé sanitní služby

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Lukáš Prokop
30. dubna 2019

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce, Ing. Vladímíru Bartíkovi, Ph.D., za poskytnuté cenné rady, ochotu a čas věnovaný konzultacím. Dále bych rád poděkoval sanitní službě SANIT PLUS za poskytnutí informací potřebných k vytvoření systému.

Obsah

Kapitola 1

Úvod

Dnešní svět bez informačních systémů si dokáže představit jen málokdo. Využívají je výrobní i obchodní společnosti, veřejná správa, sektor služeb, malé firmy i živnostníci. Ať už se vydáme do nemocnice, knihovny, školy, na letiště, nádraží či na dispečink zdravotnické záchranné služby, každá z těchto institucí má svůj informační systém, bez kterého by se neobešla. Ačkoliv by se mohlo zdát, že již existuje řešení pro všechny oblasti, soukromé sanitní služby jsou jedním z míst, kde dosud vývoj informačních systémů není rozšířený. Není neobvyklé, že zaznamenávání převozů probíhá do papírových deníků, trasy se plánují na základě nástěnné mapy České republiky a svoji polohu a stav realizace trasy hlásí řidiči dispečinku prostřednictvím vysílačky v sanitním voze.

Cílem této bakalářské práce je navrhnout takový systém, který by mohl sloužit všem soukromým převozovým sanitním službám, kterých je v současné době v České republice více než sto. Systém se bude skládat z webové aplikace určené pro dispečink a z mobilní aplikace určené pro řidiče sanitních vozů.

Webová aplikace bude sloužit ke vkládání nových objednávek do systému, efektivnímu plánování tras na základě propojení systému s mapovými podklady, přidělování tras řidičům na základě polohy a stavu sanitních vozů a základní správě zaměstnanců, zdravotních pojišťoven, zdravotnických zařízení a sanitních vozů.

Mobilní aplikace bude využita pro přihlašování řidičů do služby, odesílání aktuální polohy sanitního vozu, zobrazení tras přidělených dispečinkem danému sanitnímu vozu, zaznamenávání začátku a konce trasy i dokončení jednotlivých bodů na aktuální trase.

Tato práce je rozdělena do deseti kapitol. První kapitolou je úvod. Druhá kapitola se věnuje vývoji webových aplikací, seznamuje čtenáře mimo jiné s principem fungování webových aplikací a frameworky, které je možné k jejich tvorbě využít. Třetí kapitola se nese v podobném duchu jako kapitola předchozí, avšak pojednává o vývoji mobilních aplikací a zaměřuje se především na rozdíl mezi nativními a hybridními aplikacemi. Čtvrtá kapitola se věnuje systému GPS, jeho struktuře, principu určování polohy a práci se zeměpisnými souřadnicemi. Tím končí teoretická část a začíná část praktická.

Pátá kapitola se zabývá analýzou požadavků na webovou i mobilní aplikaci a zhodnocením existujících řešení. Šestá kapitola se věnuje návrhu informačního systému, je zde vyobrazen ER diagram a podrobně popsán návrh jednotlivých částí systému. Sedmá kapitola se zabývá implementací webové i mobilní aplikace. Osmá kapitola zmiňuje způsob testování vytvořeného systému. Devátá kapitola zmiňuje rozšíření, která by bylo možné při dalším vývoji do systému integrovat. Poslední kapitolou je kapitola desátá, která obsahuje závěr.

Kapitola 2

Vývoj webových aplikací

Tato kapitola se věnuje vysvětlení pojmu *webová aplikace*, výhodám a nevýhodám, stručné historii a principu fungování webových aplikací. Dále se zabývá technologiemi a moderními frameworky, které je možné k jejich tvorbě využít.

2.1 Webová aplikace

Pojem *webová aplikace* značí takovou aplikaci, která je přístupná za pomoci webového prohlížeče s dostupností připojení k internetu. Webový prohlížeč se také nazývá tenký klient, protože logika aplikace je pro něj neznámá, a je tak plně závislý na serveru, na kterém je samotná aplikace uložená a ke kterému klient ze svého zařízení pouze přistupuje. Jedná se tedy o typ aplikace, kterou uživatel neinstaluje do svého zařízení.

Webová aplikace je seskupením statických a dynamických webových stránek. Může nám proto připomínat běžnou webovou stránku, avšak ve skutečnosti je oproti ní mnohem složitější a umožňuje nám provádět spoustu úkonů, při kterých obvykle probíhá komunikace s databází. Statické webové stránky se nemění, pokud o ně klient požádá, server je bez jakékoliv změny klientovi odešle. Oproti tomu dynamické webové stránky jsou před odesláním zpět klientovi modifikovány podle požadavků klienta. Princip zpracování je tedy pro statické stránky a dynamické stránky odlišný.

V případě statických stránek je zdrojový kód předem napsaný a následně uložený na server. Nejdříve o takovou stránku požádá prohlížeč webový server. Jakmile webový server tento požadavek přijme, požadovanou stránku vyhledá a pošle ji zpět prohlížeči, který o ni žádal.

U dynamických stránek je způsob zpracování poměrně složitější. Na začátku požádá prohlížeč webový server o stránku stejně jako v případě statických stránek. Po přijetí požadavku a vyhledání požadované stránky není stránka odeslána zpět prohlížeči, avšak je předána aplikačnímu serveru. Aplikační server je speciální software, který odpovídá za dokončení stránky – najde na stránce instrukce a podle nich danou stránku dokončí. Tato dokončená stránka je předána zpět webovému serveru, který ji následně pošle zpět prohlížeči [1].

2.2 Výhody a nevýhody webových aplikací

Tvorba webových aplikací je v posledních letech velmi populární a jejich oblíbenost každým rokem roste. S jejich praktickým využitím se setkáváme stále častěji – využívají se

pro tvorbu výukových programů, online editorů grafiky i textu, elektronických kalendářů, nejrůznějších evidencí, systémů pro půjčovny, knihovny a v mnoha dalších odvětvích [20].

Výhody:

- **Přístup kdykoliv a odkudkoliv.** Webové aplikace jsou z pohledu používání velmi flexibilní, k jejich využívání postačí počítač s připojením k internetu.
- **Nezávislost na zařízení i operačním systému.** V případě běžných desktopových aplikací, které je nutné instalovat do zařízení, musíme připravit verze pro všechny druhy zařízení a všechny typy operačních systémů, pokud chceme zajistit bezproblémový chod. To je ale velmi náročné, ať už z časového hlediska při vývoji, nebo také z finančního hlediska. Webové aplikace budou fungovat na všech zařízeních, které mají prohlížeč a připojení k internetu.
- **Snadné přizpůsobení.** Úpravy vzhledu webových aplikací jsou pro vývojáře jednodušší než úpravy desktopových aplikací, je tedy možné v krátkém časovém intervalu zareagovat na požadované změny.
- **Žádné aktualizace.** Velmi pozitivní je, že po úpravách webových aplikací nejsou nutné žádné aktualizace na straně uživatele. Při každém načtení této aplikace je automaticky zobrazena nejnovější dostupná verze uložená na serveru.
- **Nižší výdaje při vývoji.** Hlavní výhodou webových aplikací je nižší cena. Běžné desktopové aplikace musí být vyvíjeny pro různé operační systémy. Následně musejí testeři otestovat funkčnost aplikace na všech dostupných platformách. U webových aplikací není nutné testovat všechny dostupné platformy, jedna webová aplikace bude fungovat všude.

Nevýhody:

- **Pomalejší chod.** Pokud máme velmi pomalé připojení k internetu, práce s webovou aplikací může být časově náročnější. Také v případě, kdy by se jednalo o náročnější aplikaci například z pohledu velkého množství grafiky, která musí být přenášena, nemusí být reakce webové aplikace vždy okamžité.
- **Bezpečnost.** Nelze říci, že by byly webové aplikace nebezpečné, avšak je nutné je dobře zabezpečit. Obzvlášť pokud přenášíme různá citlivá data, musíme je při přenosu šifrovat z důvodu ochrany soukromí a rizika bezpečnostních útoků. Měli bychom také dbát na autentizaci - ověření totožnosti - a autorizaci - ověření práv pro vykonání určité činnosti.

2.3 Stručná historie

Pokud se přesuneme do období 60. let 20. století, dostáváme se k počátkům sálových počítačů. Takové počítače byly velmi drahé, proto si firmy, které je plánovaly začít využívat, nemohly dovolit koupit počítač pro každého zaměstnance. Zakoupil se jeden centrální počítač, na kterém aplikace běžely. K tomuto počítači mohlo být připojeno více stanic – monitorů a klávesnic. Nevýhodou sálových počítačů byla především vysoká cena a nízký výkon. Naopak z pohledu bezpečnosti můžeme mluvit o výhodě, protože veškerá data byla

uložena na jednom počítači, a ne jako později na počítačích jednotlivých zaměstnanců. Jednoduchá byla také správa sálových počítačů, kdy stačilo pouze na daný počítač nainstalovat požadovanou aplikaci nebo na něm provést potřebné aktualizace a nebylo třeba tyto akce provádět u každého zaměstnance na jeho počítači.

S postupem doby a snižováním cen počítačů se počítače začaly dostávat do menších firem i domácností, přičemž firmy si mohly dovolit více počítačů než jen jeden sálový. S tímto obdobím přichází desktopové aplikace, které fungují na principu, že každý uživatel má danou aplikaci nainstalovanou na svém počítači. Výhody sálových počítačů se nám najednou obracejí v nevýhody, je nutné provádět instalace a aktualizace na každém počítači zvlášť a zvyšuje se bezpečnostní riziko úniku dat. Získáváme ale vyšší výkon, protože se o něj nemusíme dělit s ostatními uživateli.

S rozšiřováním internetu začalo vznikat velké množství webových stránek. Ze začátku se jednalo pouze o statické stránky, jejichž způsob zpracování byl popsán v kapitole 2.1. Současná architektura je do jisté míry podobná architektuře sálových počítačů – soubory jsou uložené na serveru a pouze se k nim přes tenkého klienta přistupuje. Postupem času ale statické stránky přestaly stačit, lidé začali mít potřebu do stránek přidávat dynamickou funkčnost. Tyto pokusy byly dotaženy až tak daleko, že jsme v dnešní době schopni desktopovou aplikaci plně nahradit webovou aplikací.

2.4 Princip fungování webových aplikací

Princip zpracování statických i dynamických stránek jsme si již nastínili v kapitole 2.1. Na těchto principech jsou postavené i webové aplikace.

Webové aplikace fungují tak, že poté, co uživatel zadá konkrétní URL adresu stránky, kterou požaduje, se na server odešle požadavek k získání dané stránky. Na serveru běží tzv. CGI skript, který zpracuje skripty obsažené na požadované stránce – připojí se k databázi a načte data, která uživatel vyžaduje – a vrátí statickou webovou stránku, která je odeslána zpět uživateli jako výstup jeho požadavku.

Všechna data jsou uložena v databázi, se kterou uživatel webové aplikace komunikuje prostřednictvím uživatelského rozhraní [1].

2.5 Technologie na straně klienta

Tato podkapitola se věnuje technologiím používaným na straně klienta, kam mimo jiné patří HTML, CSS a JavaScript. Zmíněny jsou také populární CSS frameworky a javascriptové knihovny, které umožňují rychlejší a pohodlnější vývoj frontendu.

2.5.1 HTML

Jazyk HTML je základním prostředkem pro tvorbu webových stránek. Jedná se o značkovací jazyk, což je prostředek, který slouží k obohacení textu o dodatečné informace, které se vkládájí přímo do textu prostřednictvím značek. Každá značka má svůj význam a ovlivňuje základní vzhled stránky [18].

Kořeny tohoto jazyka sahají do jazyka SGML, který je možné si představit jako sadu pravidel pro tvorbu značkovacích jazyků. Na počátku 90. let 20. století vznikl aplikací těchto pravidel jazyk HTML s cílem ujednotit způsob definice struktury hypertextových dokumentů, což jsou dokumenty, jejichž text není lineární – obsahuje odkazy na další hyper-

textové dokumenty. K šíření hypertextových dokumentů po síti Internet napomáhá protokol HTTP [4].

Významnou verzí HTML byla verze 4 publikovaná v roce 1997, která přinesla počátek velkého rozvoje webových stránek. Důležitým nástupcem se stala verze 5, která je nyní aktuální a jejíž vývoj stále probíhá.

Tato nejnovější verze přinesla mnoho novinek, které v různých ohledech zjednodušují vývoj webů. Mezi hlavní přínos patří například:

- nové speciální elementy pro přehlednější strukturování dokumentů
- podpora audia a videa (dosud řešeno pomocí Flash Player)
- tag `<canvas>` pro vykreslování grafických objektů – vliv na vývoj webových her
- geolokační rozhraní
- vestavěná validace formulářů

2.5.2 CSS

CSS je jazyk vytvořený pro stylování značkovacího jazyka HTML. Pomocí CSS definujeme způsob zobrazení elementů na stránce. Styly je možné definovat buď v hlavičce HTML dokumentu, nebo v externím souboru s příponou `.css`, který k HTML dokumentu přiřadíme [33].

V kaskádových stylech se definují pravidla, přičemž každé pravidlo se skládá ze selektoru a bloku deklarací. Každá deklarace se skládá z názvu vlastnosti a hodnoty oddělenými dvojtečkou a je ukončena středníkem [32].

Výrazným posunem v CSS bylo vydání verze CSS3, která přináší mnoho nových vlastností, díky kterým je možné se často obejít bez JavaScriptu v situacích, kde byl dříve potřeba. Mezi hlavní novinky je možné zařadit například [14]:

- selektory, pomocí kterých přistupujeme k elementům (už nemusí mít ID ani třídu)
- zaoblené rohy (dříve se muselo řešit pomocí obrázků)
- stín u textu i u blokových prvků
- transformace prvků – posunutí, otočení a změna měřítka
- barevné modely RGBA, HSL a HSLA (umožňují práci s průhledností)

Bootstrap

Knihovna Bootstrap je sada nástrojů pro tvorbu webu a webových aplikací, která nabízí kodérům a programátorům pohodlný vývoj pomocí předdefinovaných elementů a tříd. Mezi její hlavní přednosti patří [5]:

- **Šetří čas a je snadná na použití.** Díky Bootstrapu není třeba trávit čas psaním kódu, stačí využít předdefinované grafické šablony a třídy a pouze je aplikovat na vhodné místo.
- **Skvělý grid systém.** Vývoj webu pomocí mřížky je výhodný nejen z pohledu rozložení elementů, ale také pro vývoj responzivních variant webu.

- **Přehledná dokumentace.** Bootstrap nabízí přehledně rozčleněnou dokumentaci zahrnující mnoho praktických ukázek použití.
- **Kompatibilita.** Bootstrap je kompatibilní se všemi moderními webovými prohlížeči včetně Internetu Exploreru verze 8 a novějších.

Bulma

Podobně jako Bootstrap je i Bulma sadou nástrojů pro efektivní tvorbu webu a webových aplikací. Bulma není tak populární jako Bootstrap, což se odráží na menší komunitě a střídmější dokumentaci. K výhodám například patří, že oproti Bootstrapu, který pro funkčnost některých elementů potřebuje jQuery, Bulma nepotřebuje ke své funkčnosti JavaScript. Dalším pozitivem je minimální velikost stylového souboru, díky čemuž je načítání stylů velmi rychlé [2].

2.5.3 JavaScript

Javascript je základním jazykem pro vývoj webových aplikací. Původně byl vyvinut Brendanem Eichem ve společnosti NetScape v letech 1995 a 1996. JavaScript není kompilovaným (překládaným) jazykem, proto byl na počátku odmítán v domnění, že se jedná o málo výkonný jazyk. Brzy na to ale byla objevena jeho síla spočívající ve vytváření interaktivity webů – vysouvací navigační nabídky, transformace textu, dynamické přidávání elementů do stránky či validace a zpracování formulářových polí.

Mezi hlavní výhody patří rychlost zpracování, jednoduchost pochopení i implementace, popularita mezi uživateli a neustálý vývoj jazyka. K nevýhodám je možné zařadit to, že uživatel může JavaScript v prohlížeči vypnout, případně jej sám prohlížeč nemusí podporovat. Omezující také může být, že JavaScript pracuje na straně klienta a neumožňuje přístup k souborům ani k databázím [26].

jQuery

jQuery je malá a rychlá javascriptová knihovna, která toho může nabídnout opravdu hodně. Celá knihovna je implementovaná v jediném `.js` souboru a nabízí snadnou manipulaci s HTML prvky – umožňuje prvkům přidávat efekty a animace, měnit jejich obsah, přidávat a rušit třídy i jiné atributy, vytvářet události, získávat či měnit pozici a rozměry elementů a mnoho dalšího.

Vše, co je možné realizovat s knihovnou jQuery, je samozřejmě možné i v JavaScriptu, avšak jQuery nabízí mnoho funkcí, které jsou jednoduché na použití a ušetří vývojářům spoustu času. Tato knihovna se stala velmi oblíbenou a ve svých projektech ji využívají velké společnosti jako Google, WordPress, Yahoo a další [15].

FancyBox

FancyBox je knihovna postavená na jQuery, která se zabývá prezentováním různých typů médií moderním způsobem, a to formou modálních oken. Soustředí se především na obrázky a videa, ale dokáže pracovat například i s Google Maps. Je možné s ní také vytvářet uživatelsky přívětivé potvrzovací dialogy. Mezi její hlavní výhody patří plná optimalizace pro mobilní zařízení, automatické rozpoznání typu obsahu a snadné použití [8].

2.6 Technologie na straně serveru

Tato podkapitola se věnuje technologiím používaným na straně serveru, kam mimo jiné patří jazyk PHP a relační databáze sloužící pro uchovávání dat. Podkapitola se věnuje nejen PHP frameworkům, ale zahrnuje také srovnání nejznámějších systémů řízení báze dat.

2.6.1 PHP

PHP je programovací jazyk, který se využívá pro programování dynamických webových stránek a webových aplikací, ale také pro desktopové i konzolové aplikace.

Jedná se o interpretovaný jazyk – na webovém serveru běží PHP interpret, což je software, který skript zpracuje a vytvoří výstup, který je předán klientovi. Oproti kompilovaným jazykům, kterými jsou například Java nebo C++ a které je nutné překládat při každé změně, je vývoj v interpretovaných jazycích výrazně rychlejší.

Oproti HTML, CSS a JavaScriptu má PHP jeden zásadní rozdíl. Všechny tři zmíněné jazyky běží kompletně v prohlížeči, tedy na straně klienta, avšak PHP běží na straně serveru. K uživateli je přenášen až výsledek. Proto není například možné prozkoumat v prohlížeči PHP kód webové stránky [19].

PHP frameworky

V posledních letech se rozmohl nový trend, kterým jsou frameworky. Ty se snaží programátorovi poskytnout efektivnější způsob řešení a ušetřit množství kódu. V souvislosti s PHP frameworky je nezbytné zmínit pojem MVC neboli *model-view-controller*, což je návrhový vzor, který odděluje data a jejich zpracování od jejich zobrazení. Stručně řečeno, *model* představuje úložiště dat a obstarává jejich získání a práci s nimi, *controller* si o data žádá a následně je předává *view*, který je zobrazí.

Ačkoliv by se mohlo zdát, že využití PHP frameworku přináší pouze výhody, není tomu tak. V následujících řádcích zmíním hlavní výhody a nevýhody, které použití PHP frameworku přináší [16].

Výhody:

- **Rychlý vývoj.** Dobře vytvořený PHP framework nabízí již předpřipravené implementace běžných problémů, což může urychlit vývoj.
- **Bezpečnější aplikace.** Při rychlém vývoji se často přechází na rychlá a ne tak dobrá řešení bezpečnosti. Řešení zahrnutá ve frameworku jsou oproti tomu pravidelně testovaná a vylepšovaná. To ale neznamená, že při použití frameworku se o bezpečnost starat nemusíme, naopak si musíme být jistí, že framework správně používáme, aby fungoval tak, jak má.
- **Lepší týmová práce.** Sám vývojář zná své třídy a funkce velmi dobře, avšak při příchodu nového vývojáře do týmu se situace mění. Pokud si během vytváření projektu netvoříme podrobnou dokumentaci, bude pro nového člena týmu velmi složité se zorientovat. Oproti tomu dokumentace k frameworkům jsou obvykle podrobné a díky nim se orientace v projektu značně zjednoduší.

Nevýhody:

- **Pomalejší běh aplikace.** PHP framework je komplexní řešení, proto před začátkem vykonávání kódu musí být veškeré třídy a knihovny načteny. Především u menších aplikací může být rozdíl znatelný.
- **Příliš obecná řešení specifických problémů.** Hlavní myšlenkou PHP frameworků je přinést obecná řešení problémů, které bude většina vývojářů během vývoje aplikace řešit. Málokdo ale řeší obecný problém, většina má specifický problém. A i když framework tento specifický problém dokáže vyřešit, vyřeší jej tak, že dostaneme mnoho dalších věcí, které pro náš problém nepotřebujeme. Proto pokud se rozhodneme PHP framework použít, je důležité si vybrat takový, který pokrývá potřeby naší aplikace.
- **Omezená transparentnost a řízení.** Využití frameworku způsobí, že aplikace bude vyvíjena na vyšší úrovni abstrakce, což může způsobit problémy. Mnoho částí může být přizpůsobeno, avšak nad jádrem frameworku a jeho knihovnami nemáme přílišnou kontrolu.

Mezi nejpopulárnější frameworky pro rok 2019 bezpochyby patří **Laravel**, který je dle statistik Google Trends[1] nejvyhledávanějším a nejdiskutovanějším PHP frameworkem. Dále jsou to například **Code Igniter**, **Symfony**, **Zend** nebo **CakePHP** [21].

2.6.2 Relační databáze

Databáze je seskupení informací, které jsou uspořádány tak, aby mohly být jednoduše spravovány, aktualizovány a aby k nim mohlo být jednoduše přistupováno. Předchůdcem databází byly papírové kartotéky. Stejně jako zmíněné kartotéky umožňují databáze třídit data podle různých kritérií a vkládat data nová.

Relační databáze je jeden z nejrozšířenějších typů databází po celém světě. Je založena na relačním modelu, což znamená, že data jsou uložena v tabulkách – řádky tabulky označují jednotlivé záznamy a sloupce tabulky atributy. Mezi tabulkami můžeme specifikovat vztahy, což slouží k tomu, abychom vzájemně svázali data, která spolu souvisejí, i když jsou uložena ve více různých tabulkách [22].

Mezi nejznámější systémy řízení báze dat, které pracují s relačním databázovým modelem, patří SQLite, PostgreSQL a MySQL.

SQLite

SQLite je souborově založený systém řízení báze dat, který nevyžaduje žádnou instalaci. To znamená, že aplikace neběží pod samostatným serverovým procesem, který by musel být spouštěn, vypínán či konfigurován. Tato architektura umožňuje, aby byla databáze kompatibilní se všemi platformami. Někteří z klíčových zákazníků, kteří ve svých produktech SQLite používají, jsou Facebook, Google či Apple. [30].

Hlavní **výhodou** je, že celá databáze je obsažena v jediném souboru na disku, díky čemuž je databáze lehce přenositelná. Pokud hledáme co nejjednodušší databázi pro vývoj či testování, SQLite je vhodným adeptem.

Mezi **nevýhody** patří především nemožnost správy uživatelů. Pokročilé systémy poskytují možnost práce více uživatelů včetně řízení oprávnění k databázi i tabulkám. Tuto funkci SQLite nenabízí [27].

[1] https://trends.google.com/

MySQL

MySQL je jeden z nejpopulárnějších relačních databázových systémů. Oproti SQLite využívá MySQL model klient-server, který se skládá z vícevláknového SQL serveru. To umožňuje vyšší výkon, jelikož vlákna mohou snadno využívat více procesorů. Databáze je napsána v jazycích C a C++ a podporuje různé platformy jako Windows Server či linuxové distribuce. Díky škálovatelnosti, zabezpečení a replikaci je MySQL jednou z nejoblíbenějších možností v podnikových aplikacích. MySQL je vlastněn a spravován firmou Oracle a využívají jej firmy jako Facebook, GitHub nebo YouTube [30].

Výhodou oproti SQLite je především podpora přístupu více uživatelů. Má také jednoduchou instalaci a rozsáhlou komunitu uživatelů. Snadná použitelnost z něj dělá skvělý nástroj mimo jiné pro webové stránky a webové aplikace.

Ačkoliv je MySQL stavěno na zvládnutí prakticky neomezeného počtu dat, **nevýhodu** můžeme zpozorovat v tom, že může mít problém vypořádat se s příliš velkým množstvím operací v určitý časový okamžik [27].

PostgreSQL

PostgreSQL je objektově-relační databázový systém, na jehož vývoji se podílí rozsáhlá komunita uživatelů a firem. PostgreSQL, tak jako MySQL, má pokročilé funkce týkající se bezpečnosti a replikace, a využívá model klient-server. Na serveru běží proces, který se stará o komunikaci s klienty, správu databázových souborů a operace vykonávané nad databází. V porovnání s MySQL má PostgreSQL menší podíl na trhu, přesto jej ale využívají populární klienti jako Instagram a Skype [30].

K **výhodám** tohoto databázového systému mimo jiné patří rozrostlá komunita uživatelů a dostupnost mnoha nástrojů pro práci s databázovým systémem podporujících PostgreSQL.

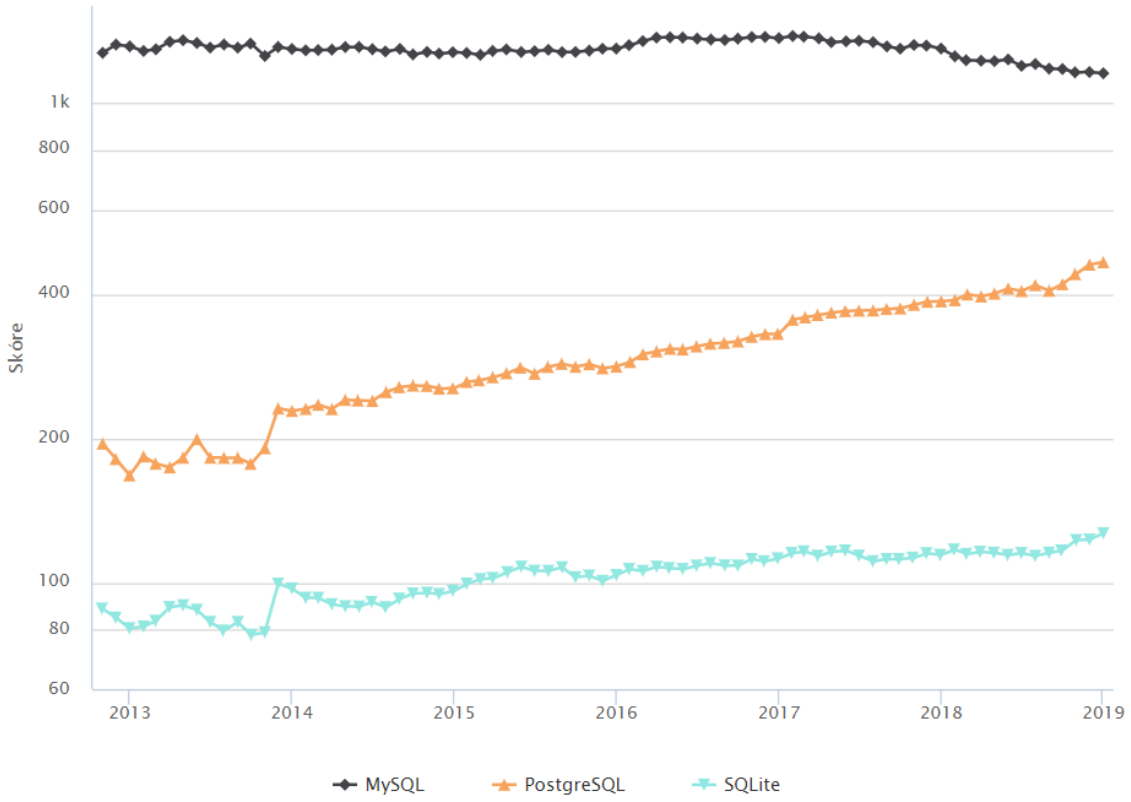
Mezi **nevýhody** patří ve srovnání s MySQL větší nároky na výkon i paměť. PostgreSQL také není vhodné používat při vysokých požadavcích na rychlost – pokud aplikace potřebuje číst velké objemy dat z databáze a provádět časté aktualizace dat, tento databázový systém nepatří mezi nejlepší možnou volbu.

Popularita databázových systémů

Služba DB-Engines[2] nabízí k porovnání jednotlivé databázové systémy. Každý databázový systém má přidělené skóre založené na mnoha parametrech, mezi které patří zastoupení ve vyhledávačích, všobecný zájem o systém, počet nabídek práce týkajících se systému či množství příspěvků na sociálních sítích na téma konkrétního databázového systému.

V grafu na obrázku 2.1 je možné vidět, že z výše zmíněných systémů se řadí MySQL mezi nejpopulárnější již mnoho let. Zatímco SQLite zažívá pouze nepatrný růst v oblíbenosti, popularita PostgreSQL v posledních letech výrazně roste.

[2] https://db-engines.com/en/



Obrázek 2.1: Graf popularity databázových systémů

Kapitola 3

Vývoj mobilních aplikací

Tato kapitola se věnuje vysvětlení pojmu *mobilní aplikace*, jednotlivým typům mobilních aplikací a jejich výhodám a nevýhodám. Další část je věnována vývojovým prostředím pro vývoj nativních a hybridních aplikací. Závěr kapitoly je věnován frameworkům pro hybridní aplikace.

3.1 Mobilní aplikace

Pojmem *mobilní aplikace* se rozumí softwarová aplikace, která je vytvořená speciálně pro mobilní zařízení, mezi které řadíme mimo jiné mobilní telefony a tablety. Aby bylo možné danou aplikaci v mobilním zařízení používat, je nezbytné si ji stáhnout a nainstalovat. Do mobilních zařízení se aplikace nejčastěji stahují prostřednictvím online obchodů s aplikacemi – Google Play pro operační systém Android, App Store pro operační systém iOS a Microsoft Store pro operační systém Windows Phone.

Počátky mobilních aplikací můžeme hledat již v době před rozšířením prvních chytrých telefonů. Typickým příkladem takové aplikace je známá hra had, která se nacházela v proslulém tlačítkovém telefonu Nokia 3310. S nástupem doby chytrých telefonů začaly postupně vznikat aplikace z nejrůznějších oblastí – cestování, doprava, fotografie, mapy a navigace, nakupování a mnoho dalších [13].

Důležitou otázkou spojenou s mobilními aplikacemi je soukromí uživatelů. Zatímco některé aplikace žádají o přístup pouze k datům, která nezbytně potřebují pro svoji funkčnost, jiné aplikace se snaží od uživatelů získat maximum dat. Může se jednat o přístup k emailovým kontaktům, záznamům hovorů, kalendáři, GPS poloze zařízení či k fotoaparátu.

Vývoj mobilních aplikací je závislý na typu aplikace, pro který se rozhodneme. Jednotlivým typům mobilních aplikací a jejich odlišnostem se budu podrobně věnovat v následujících podkapitolách.

3.1.1 Nativní mobilní aplikace

Prvním typem mobilních aplikací jsou nativní mobilní aplikace. To jsou takové aplikace, které se vyvíjí v programovacích jazycích typických pro danou platformu – Java nebo Kotlin pro Android, Objective-C nebo Swift pro iOS a C pro Windows Phone. Vyvinutá nativní aplikace je tedy určená pouze pro jednu platformu, což je hlavní problém při vývoji nativních aplikací. Pokud chce vývojář, aby byla aplikace dostupná pro všechny 3 základní platformy, musí stejnou aplikaci implementovat pro každou platformu zvlášť [24].

Výhody

- **Výkon.** Při nativním vývoji vytváříme aplikaci v jazyce dané platformy, aplikace jsou tak překládány přímo pro danou platformu.
- **Rychlost.** Vzhledem k tomu, že jsou aplikace optimalizovány pro konkrétní operační systém, v rychlostních a výkonnostních testech dosáhneme vyššího hodnocení.
- **Uživatelská přívětivost.** Snadná orientace v aplikaci je pro uživatele zásadní při rozhodování, zda aplikaci bude využívat. Při nativních aplikacích využíváme přímo prvky v podobě závislé na uživatelském rozhraní mobilního zařízení.
- **Flexibilita.** Nativní aplikace nabízejí snadný a rychlý přístup k vestavěným funkcím mobilního zařízení, jako je fotoaparát, GPS, mikrofon či kalendář.

Nevýhody

- **Dlouhá doba vývoje.** Ve srovnání s hybridními aplikacemi je potřeba na vývoj nativní aplikace výrazně více času.
- **Vysoká cena vývoje.** Pokud potřebujeme aplikaci pro všechny platformy, znamená to, že budeme muset několikrát implementovat tu stejnou aplikaci, což také znamená několikanásobnou cenu.
- **Zdlouhavé a náročné aktualizace.** Při každém nalezení nějaké chyby či vytvoření nové verze aplikace musíme upravit aplikaci pro všechny platformy, což výrazně zpomaluje proces vydávání nových aktualizací.

Použití

Nativní aplikaci je vhodné zvolit při vývoji 2D a 3D her náročných na výkon, v aplikacích s náročnými animacemi či v případě, kdy vyvíjíme aplikaci pouze pro jednu platformu. Nevhodné je nativní aplikace používat při jednoduchých a multiplatformních aplikacích.

Příklady aplikací

Mezi známé nativní aplikace patří například aplikace pro online hovory Skype, populární sociální síť Instagram, hra Pokémon GO nebo profesní sociální síť LinkedIn.

3.1.2 Hybridní mobilní aplikace

Dalším typem mobilních aplikací jsou hybridní mobilní aplikace, které jsou založené na jazycích HTML, CSS a JavaScript, tedy na jazycích, které se používají k tvorbě webových stránek a webových aplikací. Při vytváření hybridní aplikace vytváříme pouze jednu aplikaci, kterou je následně možné použít na více platformách. Hybridní aplikace jsou v podstatě kombinací nativních aplikací s webovými aplikacemi [24].

Výhody

- **Multiplatformnost.** Vývoj pro více platform probíhá souběžně.
- **Nízká cena vývoje.** Vzhledem k tomu, že pro více platform vyvíjíme aplikaci souběžně, omezíme tím náklady spojené s více programátory i množstvím času.

- **Snadná údržba.** Aktualizace hybridních aplikací jsou snadné a rychlé, proto nečiní žádný problém jejich aktualizování při každé změně.
- **Brzy k dispozici.** Pokud máme nápad a nechceme, aby jej o nás připravila konkurence, hybrídní aplikace je tou správnou cestou. Rychlost vývoje tak bude výrazně vyšší.

Nevýhody

- **Nižší výkon.** Hybridní aplikace vkládá mezi zdrojový kód a cílovou platformu další vrstvu v podobě frameworku, která může mít za následek ztrátu výkonu aplikace.
- **Odladění aplikace.** Vývojáři se spolehají na to, že framework je správně vytvořený a bude se chovat na všech cílových platformách co nejpodobněji. To tak ale nemusí vždy být.
- **Vyšší nároky na hardware.** Ve srovnání s nativními aplikacemi jsou nároky na hardware pro plynulý běh aplikace vyšší, i když v posledních letech se tento rozdíl se zvyšujícím se výkonem mobilních zařízení značně minimalizuje.

Použití

Hybridní aplikaci je vhodné zvolit při vývoji jednodušších 2D her, v aplikacích využívajících API, v aplikacích, které chceme mít co nejrychleji dokončené a především v aplikacích určených pro více platforem. Nevhodné je hybridní aplikace používat v aplikacích náročných na výkon a 3D hrách.

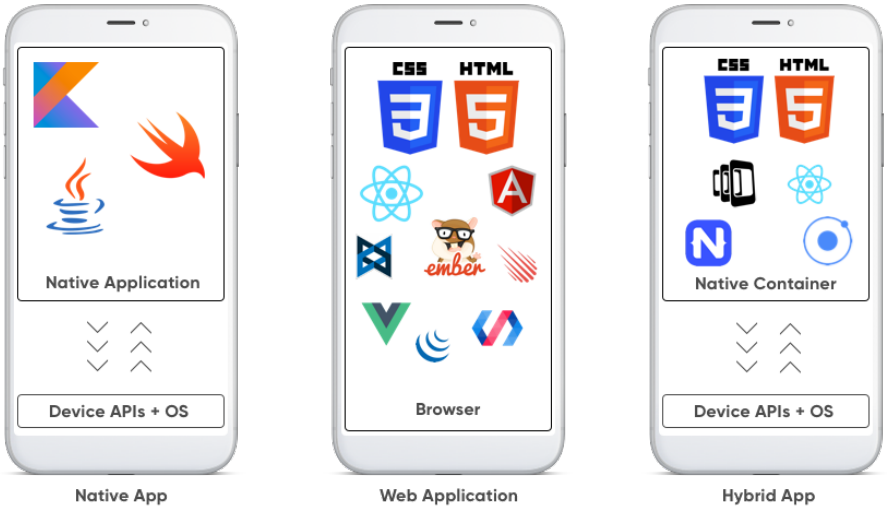
Příklady aplikací

Mezi známé hybridní aplikace patří například aplikace pro pořizování poznámek Evernote, aplikace s cvičebními a fitness plány Sworkit nebo aplikace pro relaxaci a boj proti stresu Pacifica.

3.1.3 Mobilní weby

Posledním typem mobilních aplikací jsou mobilní weby. Nejedná se však o typickou mobilní aplikaci, ale o důkladně optimalizovaný web pro mobilní zařízení. Ze všech tří zmíněných typů je z hlediska rychlosti vývoje tento typ nejrychlejší, protože k vytvoření mobilní verze webu stačí pouze upravit současné stylové soubory. Druhou zásadní výhodou je, že uživatel nemusí do svého telefonu instalovat žádnou aplikaci. Nevýhody ale u tohoto typu převládají, mobilní web nemá přístup k funkcím telefonu, pro přístup na něj musí uživatel zadat URL adresu do prohlížeče, nemá tedy žádnou ikonu ve svém mobilním zařízení. Zároveň také mobilní web nebude funkční, pokud nebude uživatel připojený k internetu.

Na obrázku 3.1 je možné vidět jednoduché srovnání mezi nativními aplikacemi (vlevo), mobilními weby (uprostřed) a hybridními aplikacemi (vpravo) z pohledu využívaných technologií a principu fungování.



Obrázek 3.1: Srovnání typů aplikací z pohledu technologií a principu fungování. „Převzato z https://codeburst.io/hybrid-apps-villains-or-good-guys-59a9fa2066d5."

3.2 Vývojová prostředí

Vývojové prostředí je software, který usnadňuje práci vývojáři. Zahrnuje editor zdrojového kódu, kompilátor, případně i interpret a debugger. V této podkapitole se budu věnovat vývojovým prostředím vhodným pro vývoj nativních a hybridních mobilních aplikací.

3.2.1 Vývojová prostředí pro vývoj nativních aplikací

Android Studio

Android Studio[1] je vývojové prostředí založené na vývojovém prostředí IntelliJ IDEA[2]. Za tímto vývojovým prostředím stojí Google, který vypustil první verzi v roce 2013. Od té doby je prostředí pravidelně aktualizováno a je dostupné na platformách Windows, Linux a Mac OS X.

Instalace Android Studia je jednoduchá, stačí pouze stáhnout instalační balík z oficiálních stránek a Java Developement Kit, tedy soubory základních nástrojů pro vývoj aplikací pod platformou Java. Kromě samotného vývojového prostředí obsahuje instalační balík kompilátor pro Android a emulátory s plnohodnotným systémem Android pro účely testování.

[1] https://developer.android.com/studio
[2] https://www.jetbrains.com/idea/

Xcode

Xcode[3] je vývojové prostředí patřící společnosti Apple, které obsahuje vývojářské nástroje pro vývoj aplikací na platformy iOS, macOS, watchOS a tvOS. První verze byla spuštěna již v roce 2003 a vývoj stále pokračuje. Xcode podporuje programovací jazyky C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, Rez a Swift.

Xcode v sobě zahrnuje nástroje jako LLVM kompilátor, LLDB debugger, iOS Simulator pro testování vyvíjených aplikací, Interface Builder pro tvorbu uživatelského rozhraní bez nutnosti psát kód a Instruments pro testování a analýzu výkonu.

3.2.2 Vývojová prostředí pro vývoj hybridních aplikací

Apache Cordova

Apache Cordova je vývojový framework určený pro vytváření multiplatformních mobilních aplikací za použití standardních webových technologií, mezi které řadíme HTML, CSS a JavaScript. V současnosti Cordova podporuje nejen tři hlavní platformy, kterými jsou Android, iOS a Windows Phone, ale také řadu menších platforem. Stejně jako ve webových aplikacích je možné v Cordově používat javascriptové knihovny jako například jQuery. Oproti webovým aplikacím má vývojář přístup k hardwaru mobilního zařízení – může tedy využívat fotoaparát, senzory, geolokaci, úložiště, stav připojení k síti a mnoho dalšího.

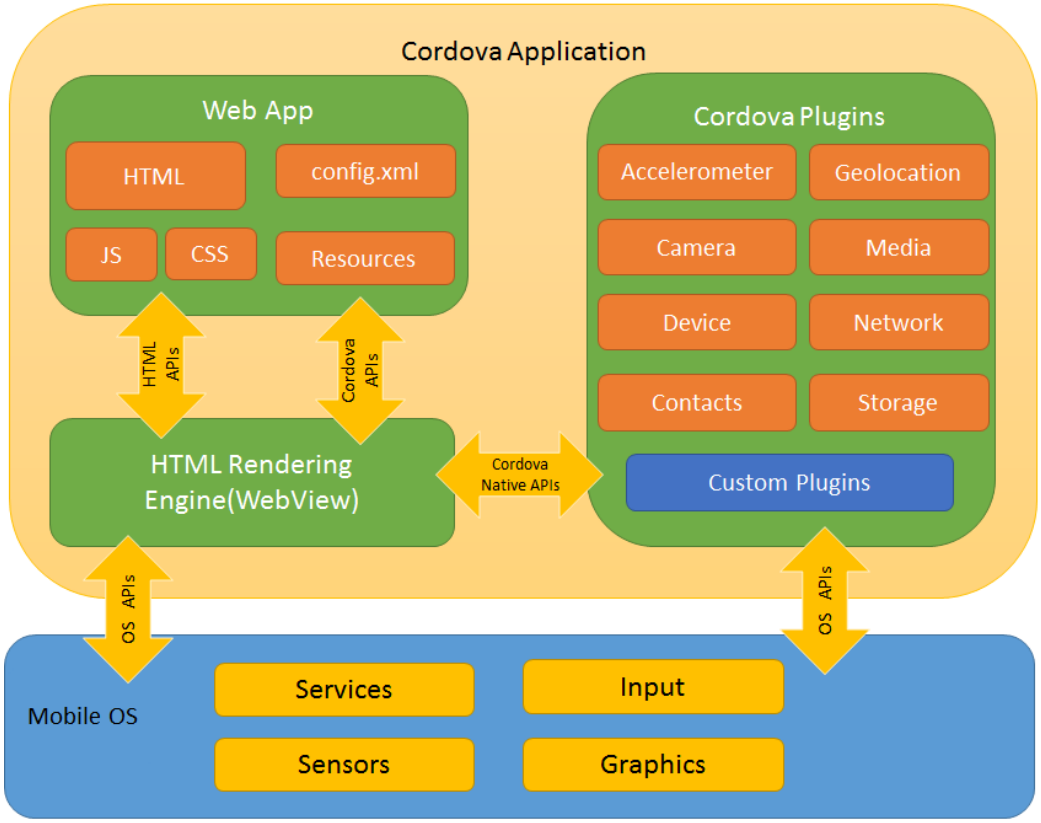
Aplikace vytvořená ve frameworku Cordova se skládá z několika částí. Na obrázku 3.2 je možné vidět architekturu takové aplikace.

Při sestavování aplikace pro konkrétní platformu vytvoří Cordova nativní kostru aplikace, která obsahuje pouze prvek nazývaný WebView. WebView je webová vykreslovací komponenta, kterou je možné integrovat jako součást aplikace. V podstatě se jedná o webový prohlížeč, který v aplikaci naše stránky zobrazuje.

Další částí architektury je webová aplikace. To je část, kde se nachází kód dané aplikace. Samotná aplikace je implementovaná jako webová stránka. Obvykle se jedná o soubor s názvem `index.html`, který odkazuje na CSS, JavaScript, obrázky a další soubory potřebné pro správný běh aplikace. Velmi důležitý je také soubor `config.xml`, který obsahuje informace o aplikaci a lze v něm nastavit její chování.

Nedílnou součástí jsou také pluginy. Z pohledu Cordovy je plugin doplňující kód, který poskytuje JavaScriptu rozhraní pro přístup k nativním komponentám. Díky tomu můžeme v mobilní aplikaci využívat hardware mobilního zařízení. Projekt Apache Cordova udržuje omezenou sadu oficiálních pluginů, které poskytují přístup k základním funkcím zařízení – stav baterie, fotoaparát, geolokace, notifikace, vibrace, úložiště a další. Kromě toho na svých oficiálních webových stránkách[4] nabízí pro vývojáře více než 4000 pluginů třetích stran.

[3] https://developer.apple.com/xcode/
[4] https://cordova.apache.org/plugins/



Obrázek 3.2: Architektura aplikace vytvořené v Cordově.
„Převzato z
https://cordova.apache.org/docs/en/latest/guide/overview/index.html.“

PhoneGap

PhoneGap je moderní framework pro vývoj mobilních aplikací, který je předchůdcem již zmíněného frameworku Apache Cordova. V roce 2011 byl PhoneGap, vyvíjený společností Adobe, darován společnosti Apache, která se na jeho vývoji v současnosti podílí. Produkt PhoneGap ale zůstal zachován. Tak jako je například WebKit pohonem pro mnoho moderních webových prohlížečů, framework Apache Cordova se stal pohonem pro PhoneGap.

Právě robustní nástroje jsou tím, co odlišuje PhoneGap od Apache Cordova. PhoneGap nabízí nejen vlastní příkazovou řádku a desktopovou aplikaci pro vytváření aplikací, ale také mobilní aplikaci, která umožňuje okamžité sledování změn ve vyvíjené aplikaci přímo na připojeném mobilním zařízení. Není tak nutné při každé změně zkompilovat a přeinstalovat aplikaci. Vývojáři také ocení nástroj PhoneGap Build, který sestaví aplikaci v cloudu, díky čemuž bude aplikace vždy vytvořena na nejaktuálnější sadě SDK pro cílovou platformu [17].

React Native

React Native je framework určený pro vývoj mobilních aplikací, za jehož vývojem stojí Facebook. Hlavní výhodou technologie React Native je možnost vývoje pouze jednoho sdíleného kódu pro všechny platformy. React Native tak přináší příležitost jednoduššího vývoje aplikace pro telefony či tablety, ale také pro televize a chytré hodinky.

React Native blízce souvisí se samotným Reactem, což je javascriptová knihovna sloužící pro vytváření uživatelských rozhraní. React je postaven na používání tzv. komponent, které se zapisují jako HTML tagy. Tak jako React Native je i React dílem společnosti Facebook.

Tento framework umožňuje vytvářet mobilní aplikace pouze za použití JavaScriptu. Aplikace vytvořené v React Native využívají stejné základní stavební bloky uživatelského rozhraní jako běžné aplikace pro Android a iOS. Vše je ale zjednodušené tím, že namísto používání jazyků jako Java, Swift nebo Kotlin tyto stavební bloky skládáme dohromady pomocí JavaScriptu a Reactu [7].

Hlavní odlišností frameworku React Native od dříve zmíněných frameworků Apache Cordova a PhoneGap je fakt, že výsledným produktem je plnohodnotná nativní aplikace složená z nativních komponent dané platformy. Nenachází se zde tedy žádný integrovaný webový prohlížeč, který by vykresloval stránky.

Při využívání nativních komponent nastává problém v tom, že nativní komponenty nejsou na všech platformách totožné. To může vést k situaci, kdy bude nezbytné použít pro jednu platformu jinou nativní komponentu než pro platformu druhou. S tím naštěstí React Native počítá a je tak možné pomocí jednoduché podmínky vytvořit různé části kódu specifické pro každou platformu. V případě většího množství odlišného kódu lze vytvořit soubory specifické pro danou platformu ve formátu `*.android.js` a `*.ios.js` [11].

3.3 Frameworky pro hybridní aplikace

Vývoj hybridních mobilních aplikací je v posledních letech na výrazném vzestupu, a to především díky rychlému vývoji, ke kterému přispívá možnost vytvořit pouze jeden kód, který bude spustitelný na všech platformách. V souvislosti s tímto vzestupem vzniká mnoho užitečných frameworků, které pomáhají vytvářet skvělé hybridní mobilní aplikace. V této podkapitole zmíním ty neoblíbenější a dle mého názoru nejzdařilejší z nich.

3.3.1 Ionic

Framework s názvem Ionic lze bezpochyby zařadit mezi nejpoužívanější frameworky pro vývoj hybridních mobilních aplikací. Dle slov Maxe Lynche[5], spoluzakladatele Ionicu, byly k začátku roku 2017 vytvořeny téměř 4 miliony aplikací využívajících Ionic, a to za pouhé tři roky existence.

Ionic je vytvořen na bázi AngularJS, což je javascriptový webový framework zaměřující se na tvorbu single-page aplikací. Pro sestavení aplikací na cílových platformách využívá Cordovu. Ionic se může pyšnit především propracovanou dokumentací a rozsáhlou komunitou vývojářů. Součástí frameworku je také sada několika set nejpoužívanějších ikon [12].

3.3.2 Framework7

Framework7 je framework sloužící k vývoji hybridních mobilních aplikací i webových aplikací, které budou mít vzhled připomínající nativní aplikace v Androidu nebo iOSu. Jedná se také o nepostradatelný nástroj pro vytváření prototypů aplikací, obzvlášť v situacích, kdy potřebujeme prototyp co nejrychleji. Hlavním cílem je poskytnout možnost rychlého vývoje aplikací pro Android a iOS za pomocí HTML, CSS a JavaScriptu [12].

[5] https://medium.com/ionic-and-the-mobile-web/ionics-2016-by-the-numbers-3a8e2c65177b

Kapitola 4

GPS

Tato kapitola se zabývá vysvětlením pojmu GPS, stručnou historií vzniku, strukturou systému a principem určování polohy. Detailně jsou zde vysvětleny zeměpisné souřadnice, jejich souvislost se zeměpisnou šířkou a zeměpisnou délkou a různé způsoby jejich zápisu a čtení. Závěr kapitoly je věnován GPS čipům v mobilních zařízeních – principu fungování a jejich přesnosti.

4.1 Globální polohový systém

Zkratka GPS značí globální polohový systém, což je nepřetržitě pracující systém pro určování polohy a času na Zemi i nad Zemí. Jedná se o radionavigační systém, který je provozován Ministerstvem obrany Spojených států amerických a který slouží nejen pro vojenské, ale i pro civilní užití [23].

4.1.1 Historie vzniku

Počátky satelitních navigačních systémů lze hledat ve 2. polovině 20. století, kdy v roce 1960 začalo Námořnictvo Spojených států amerických umisťovat na oběžnou dráhu družice historicky prvního družicového polohového systému zvaného Transit, který měl sloužit především pro určování polohy plavidel, a to s přesností stovek metrů. Postupně byla jeho přesnost zvyšována a byl uvolněn pro civilní účely. Tento projekt byl s postupem let následován řadou dalších systémů, přičemž tím nejrozšířenějším se stal NAVSTAR GPS, dnes již známý pouze jako GPS.

První zmínky o systému GPS spadají do roku 1973, kdy v první fázi byly vypuštěny 4 družice a začal probíhat vývoj uživatelských zařízení. O 6 let později bylo vypuštěno dalších 11 družic. Postupně bylo dosaženo celkového počtu 24 družic. Důležitým milníkem se stal konec roku 1993, kdy bylo zprovozněno trojrozměrné zaměřování. V roce 1995 byla oficiálně vyhlášena plná operační způsobilost systému [3].

Od té doby až do současnosti je systém GPS stále využívanější ve vědeckých i komerčních oblastech. Pro dosažení kompletního pokrytí celé planety Země je potřeba minimálně 24 družic. K lednu 2019 má globální polohový systém 31 funkčních obíhajících družic. Nejnovější družice byla vypuštěna 23. prosince 2018, která je nyní ve fázi testování [28].

4.2 Struktura systému GPS

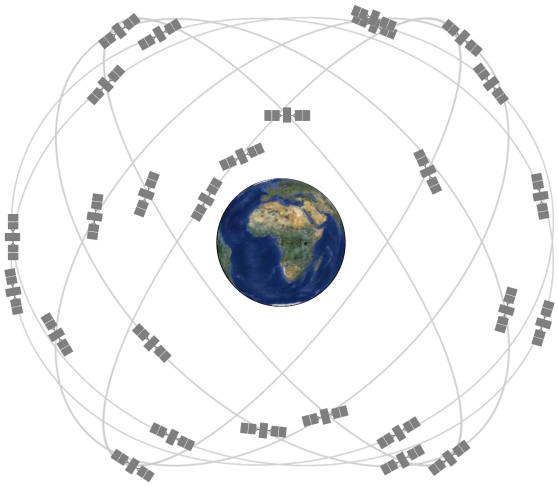
Globální polohový systém je tvořen třemi segmenty – kosmický, kontrolní a uživatelský. Kosmický segment zahrnuje družice umístěné na oběžné dráze, kontrolní segment tvoří pozemní řídící, monitorovací a vysílací stanice a uživatelský segment se skládá z přijímačů GPS.

4.2.1 Kosmický segment

Kosmický segment představují družice, které jsou umístěny na 6 téměř kruhových drahách, jak je ukázáno na obrázku 4.1. Nacházejí se ve vzdálenosti přes 20 tisíc kilometrů od povrchu Země a pohybují se rychlostí přes 11 tisíc kilometrů v hodině. Celkově jich musí být minimálně 24, skutečný počet družic je ale proměnlivý, protože staré družice jsou rušeny v závislosti na jejich technickém stavu a nové družice jsou vypouštěny.

Každá družice během jednoho dne oběhne kolem planety Země dvakrát, jeden oběh trvá přesně 11 hodin a 58 minut. Další den je na stejném místě oběžné dráhy vždy o 4 minuty dříve.

Součástí každé družice je přijímač, vysílač, atomové hodiny, procesory a mnoho dalších důležitých přístrojů, které slouží například pro detekci jaderných výbuchů. Pozemní antény dokáží družici předat informace, které si družice uchovává a zpracovává [31].



Obrázek 4.1: Rozmístění oběžných drah a družic.
„Převzato z https://www.gps.gov/multimedia/images/constellation.jpg.“

4.2.2 Řídící segment

Řídící segment tvoří soustava 5 monitorovacích stanic, 3 pozemních vysílačů povelů a hlavního řídícího střediska.

Monitorovací stanice jsou rovnoměrně rozmístěny podél rovníku, jak je vidět na obrázku 4.2. Nachází se na Havaji, v Colorado Springs v USA a na ostrovech Ascension, Diego Garcia a Kwajalein. Monitorovací stanice neustále přijímají signály z družic, uchovávají je a předávají do hlavní stanice v Colorado Springs.

V hlavním řídícím středisku se zpracovávají telemetrické údaje a výsledky sledování pohybu družic ze všech monitorovacích stanic. Také se zde uchovává časový systém GPS.

Řídící segment se mimo jiné stará o správu a údržbu stávajících družic – změny pozic družic, změny oběžných drah, stahování vysloužilých družic z oběžné dráhy a podobně. Zabývá se také vypouštěním nových družic na oběžnou dráhu [31].

Obrázek 4.2: Rozmístění monitorovacích stanic.
„Inspirace na https://www.aldebaran.cz/bulletin/2005_02/svet.gif.“

4.2.3 Uživatelský segment

Pro příjem a zpracování GPS signálů slouží GPS přijímače. Existují přijímače jednokanálové, vícekanálové a hybridní. Jednokanálové přijímače dokáží zachytit a zpracovat signál jen z jedné družice, vícekanálové mají pro každou družici vyčleněn samostatný kanál. Mohou tedy přijímat a zpracovávat signály z většího počtu družic najednou. Hybridní přijímače obsahují více kanálů, avšak ne tolik, aby byly schopny současně komunikovat se všemi družicemi. Proto musí hybridní přijímače mezi příjmem signálů z jednotlivých družic přepínat.

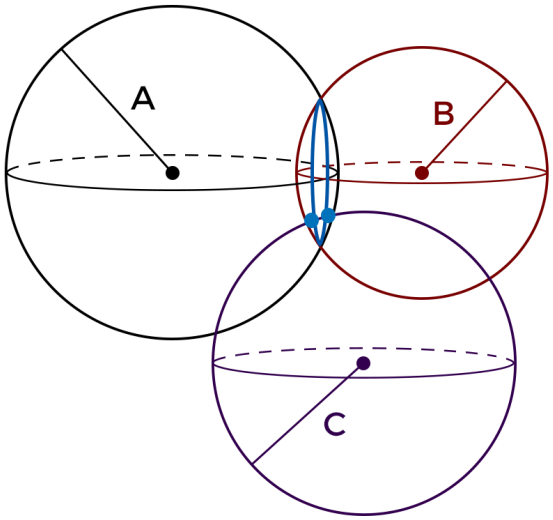
GPS přijímač je tvořen anténou, mikroprocesorem, radiofrekvenční jednotkou, komunikační jednotkou, pamětí a zdrojem napětí. Mikroprocesor řídí celý přijímací systém, radiofrekvenční jednotka zpracovává přijaté signály a komunikační jednotka slouží pro práci uživatele s přijímačem.

Podle použití je možné GPS přijímače dělit na geodetické, navigační a přijímače pro časovou synchronizaci. Geodetické přijímače slouží k velmi přesnému měření ve všech odvětvích geodézie, což je vědní obor zabývající se zkoumáním zemského tělesa nebo části zemského povrchu. Navigační přijímače se využívají v cykloturistice, turistice, motorismu, námořnictví, letectví a dalších oblastech. V každé oblasti jsou na přijímač kladeny jiné nároky, proto se jejich konstrukce i funkce liší. Mezi základní funkce patří zaměření polohy a nadmořské výšky přijímače, zaměření rychlosti či směru pohybu [31].

4.3 Princip určení polohy

Princip určení polohy je poměrně jednoduchý. Nejprve si přijímač vypočte vzdálenost, která je mezi ním a několika okolními družicemi, a to z doby cesty signálu a z rychlosti světla včetně vlivů atmosféry. Jakmile zná vzdálenost k jedné z družic, předpokládá, že leží někde na plášti koule s poloměrem rovným dané vzdálenosti, jejíž střed tvoří daná družice. To je na obrázku 4.3 označeno písmenem A. Jakmile přijímač zjistí vzdálenost k druhému satelitu (na obrázku označeno B), dokáže vypočítat průnik povrchů koule, čímž dostane kružnici.

Po zjištění třetí vzdálenosti se poloha z rozsahu kružnice zúží pouze na 2 body – jeden z nich leží vysoko v prostoru, druhý z nich leží hluboko v Zemi (ten je očividně špatný). V tomto okamžiku je znám přesný bod, kde se přijímač nachází.



Obrázek 4.3: Ilustrace principu trilaterace.
„Inspirace na http://www.baharis.cz/images/clanky/moudra/jak-pracuje-navigacni-system-gps/07.jpg.“

Tomuto postupu určení polohy se říká trilaterace. V realitě mohou vzniknout chyby způsobené odchýlením skutečné hodnoty rychlosti šíření signálu atmosférou i samotnou družicí, která může poslat nesprávné či nepřesné údaje. Vzhledem k těmto nepřesnostem se k určení polohy používají vždy nejméně 4 družice [25].

4.4 GPS souřadnice

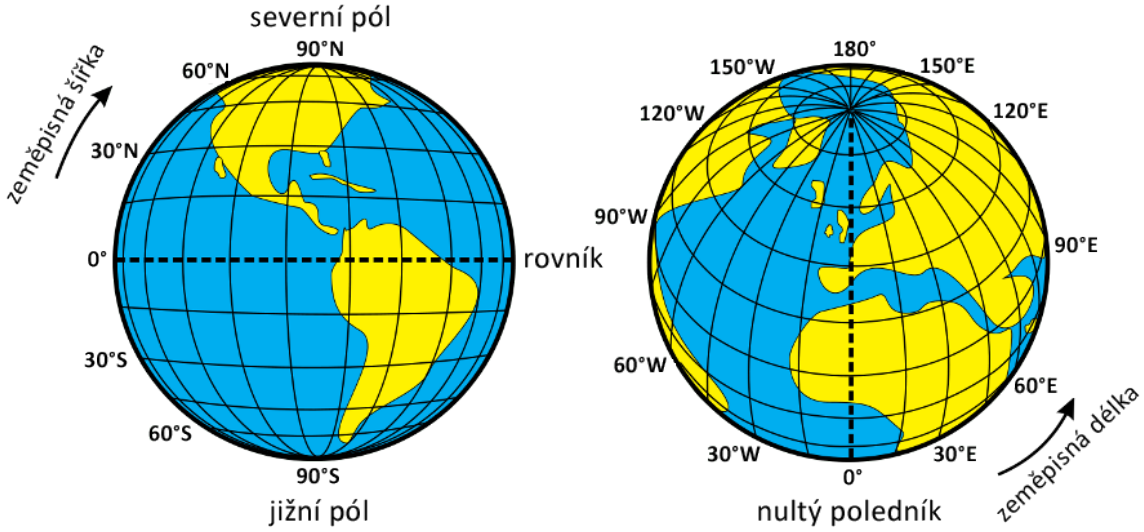
Souřadnice neboli koordináty jsou číselné hodnoty, které umožňují jednoznačně popsat polohu bodu v dané soustavě. Do obecné kategorie souřadnic řadíme mimo jiné zeměpisné souřadnice, které slouží k určení polohy na povrchu Země. Jednotlivými souřadnicemi jsou zeměpisná šířka, zeměpisná délka a nadmořská výška, přičemž pro nalezení konkrétního místa na mapě postačí pouze první dvě zmíněné.

4.4.1 Zeměpisná šířka a zeměpisná délka

Systém GPS využívá zeměpisnou šířku a zeměpisnou délku proto, aby byl schopen poskytnout přesné souřadnice polohy osoby či místa. Kombinací těchto dvou číselných hodnot jsme schopni určit polohu jakéhokoliv místa na Zemi.

Zeměpisná šířka je znázorněna na obrázku 4.4 pomocí vodorovných čar a udává, na jaké rovnoběžce se dané místo nachází. Nejdelší rovnoběžka se nazývá rovník a zeměpisná šířka zde nabývá hodnoty 0°. Její hodnota dosahuje až 90°, což je hodnota v místě severního a jižního pólu. Aby se rozlišilo, zda se jedná o severní či jižní polokouli, následuje za počtem stupňů písmeno – v případě severní polokoule se jedná o písmeno N jako *north*, v případě jižní polokoule se jedná o písmeno S jako *south*.

Zeměpisná délka je na obrázku 4.4 znázorněna pomocí svislých čar a udává, na jakém poledníku se dané místo nachází. Nejdelší poledník se nazývá nultý poledník, prochází Královskou observatoří v Greenwichi v Anglii a nabývá hodnoty 0°. Maximální hodnota zeměpisné délky je 180°, což je místo, kde vede datová hranice, tedy hranice, při jejímž překročení je třeba změnit datum. Tak jako u zeměpisné šířky je třeba rozlišit severní a jižní polokouli, u zeměpisné délky rozlišujeme východní a západní polokouli. Pokud se jedná o východní polokouli, za počtem stupňů následuje písmeno E jako *east*, v případě západní polokoule se jedná o písmeno W jako *west* [10].



Obrázek 4.4: Ilustrace zeměpisné šířky a zeměpisné délky na planetě Zemi „Inspirace na https://www.ubergizmo.com/how-to/read-gps-coordinates/."

4.4.2 Zápis a čtení zeměpisných souřadnic

Při zápisu zeměpisných souřadnic je nejprve uvedena zeměpisná šířka, která je následována zeměpisnou délkou. Souřadnice nějakého bodu tedy mohou být například `49°N, 16°E`, což by se správně četlo jako „čtyřicet devět stupňů severní zeměpisné šířky a šestnáct stupňů východní zeměpisné délky".

U většiny objektů na Zemi ale nelze určit přesnou polohu pouze na základě stupňů, proto je nutné číselný údaj zpřesnit. Existuje několik formátů, ve kterých je možné zeměpisné souřadnice určit [10].

Stupně, minuty a vteřiny (DMS)

Prostor mezi každou pomyslnou čárou zeměpisné šířky nebo zeměpisné délky představuje jeden stupeň, který je možné dále rozdělit na šedesát minut. Každou minutu je možné ještě rozdělit na šedesát vteřin. Příkladem takového formátu je:

`49°12′0.796″N, 16°36′28.228″E`

Stupně a minuty v desetinném formátu (DMM)

Prostor mezi každou pomyslnou čárou zeměpisné šířky nebo zeměpisné délky představuje jeden stupeň, který je možné dále rozdělit na šedesát minut. Každá minuta je dále dělena a vyjádřena formou desetinného čísla. Příkladem takového formátu je:

```
49 12.01327, 16 36.47047
```

Stupně v desetinném formátu (DD)

Prostor mezi každou pomyslnou čárou zeměpisné šířky nebo zeměpisné délky představuje jeden stupeň, který je dále dělen a vyjádřen formou desetinného čísla. Příkladem takového formátu je:

```
49.2002211, 16.6078411
```

Důležité je zmínit, že v případě formátů DMM a DD se již nepoužívá písmenné označení pro rozlišení severní a jižní polokoule a východní a západní polokoule, a to kvůli tomu, že v těchto dvou formátech mohou čísla nabývat i záporných hodnot. To znamená, že pokud by se v příkladu uvedeném ve formátu DD mělo jednat o jižní zeměpisnou šířku místo severní zeměpisné šířky, pouze by se před číslo `49.2002211` uvedlo znaménko minus.

4.5 GPS čipy v mobilních zařízeních

Lokalizační čipy v mobilních zařízeních získávají v posledních letech na popularitě. Vzhledem k jejich stále se zmenšujícím rozměrům je můžeme najít nejen v chytrých telefonech a tabletech, ale i chytrých hodinkách či fotoaparátech. Důležitou roli pro nás mohou hrát v mobilních navigacích a mapách, při zaznamenávání polohy vyfocené fotografie, ale také při krádeži či ztrátě zařízení.

4.5.1 Princip fungování

Princip fungování GPS čipů je stejný jako princip fungování běžných GPS zařízení. GPS čip přijímá signál ze satelitů a získaná data předává svému řídícímu zařízení. Dále už záleží jen na dané aplikaci, jak s daty naloží, zda je bude zpracovávat přímo v telefonu nebo odesílat do vzdálené databáze.

4.5.2 Přesnost

Přesnost GPS závisí na mnoha faktorech. Mezi ty hlavní patří samotné zařízení a kvalita přijatého signálu. Signál přicházející ze satelitů je čistší, pokud přichází do zařízení s co nejmenším rušením. Rušení mohou způsobit například budovy, hory či stromy.

Při zpracování přijatého signálu záleží i na samotném zařízení. Právě při výpočtu pozice založené na konfliktních signálech se mohou zařízení, co se týká přenosti, lišit. Zařízení není samo o sobě schopné jednoduše zjistit, který z přijatých signálů je čistý a který ne. Zařízení se také liší v tom, zda jsou schopna svoji přesnost zlepšovat, pokud jsou ponechána po určitou dobu v klidu. Některá zařízení se tak při zjištění, že je zařízení v klidu, snaží zpřesnit aktuální polohu. Naopak některá se s výsledkem spokojí a pouze čekají, až bude zařízení uvedeno do pohybu.

Geografickou polohu přijímače je možné v případě čistého signálu určit nejčastěji v rozmezí jednotek metrů. V případě rušení signálu například okolními stromy se může přesnost zvýšit na nižší desítky metrů.

V roce 2018 došlo v mobilních zařízeních k výraznému zpřesnění díky novému čipu BCM47755 od společnosti Broadcom, která vyvinula čip pro určení GPS polohy s přesností 30 cm [6]. Jako první tento čip použila společnost Xiaomi ve svém telefonu Xiaomi Mi 8 [9].

Kapitola 5

Analýza požadavků

Tato kapitola se věnuje požadavkům pro webovou aplikaci určenou pro dispečery a pro mobilní aplikaci určenou pro řidiče. Dále jsou zmíněna již existující řešení a jejich zhodnocení vůči požadavkům. V závěru kapitoly je uveden diagram případů užití.

5.1 Požadavky na systém

Cílem této práce je vytvořit systém pro dispečink soukromých převozových sanitních služeb. Hlavním cílem je zjednodušit práci dispečera a poskytnout mu všechny potřebné nástroje pohromadě, dále také zrychlit a zjednodušit komunikaci mezi dispečinkem a jednotlivými sanitními vozy.

Pro správné porozumění systému je třeba odlišovat dva pojmy. **Záchranná sanitní služba** se využívá v případě akutních zdravotních potíží a náhlých situací. Vždy je možné ji kontaktovat pod číslem 155. **Soukromá převozová sanitní služba** se využívá v neakutních případech, tedy pro přepravu pacientů do nemocnic, lázní, rehablitačních ústavů a zpět domů. Tuto službu zajišťují soukromé firmy, případně některé nemocnice.

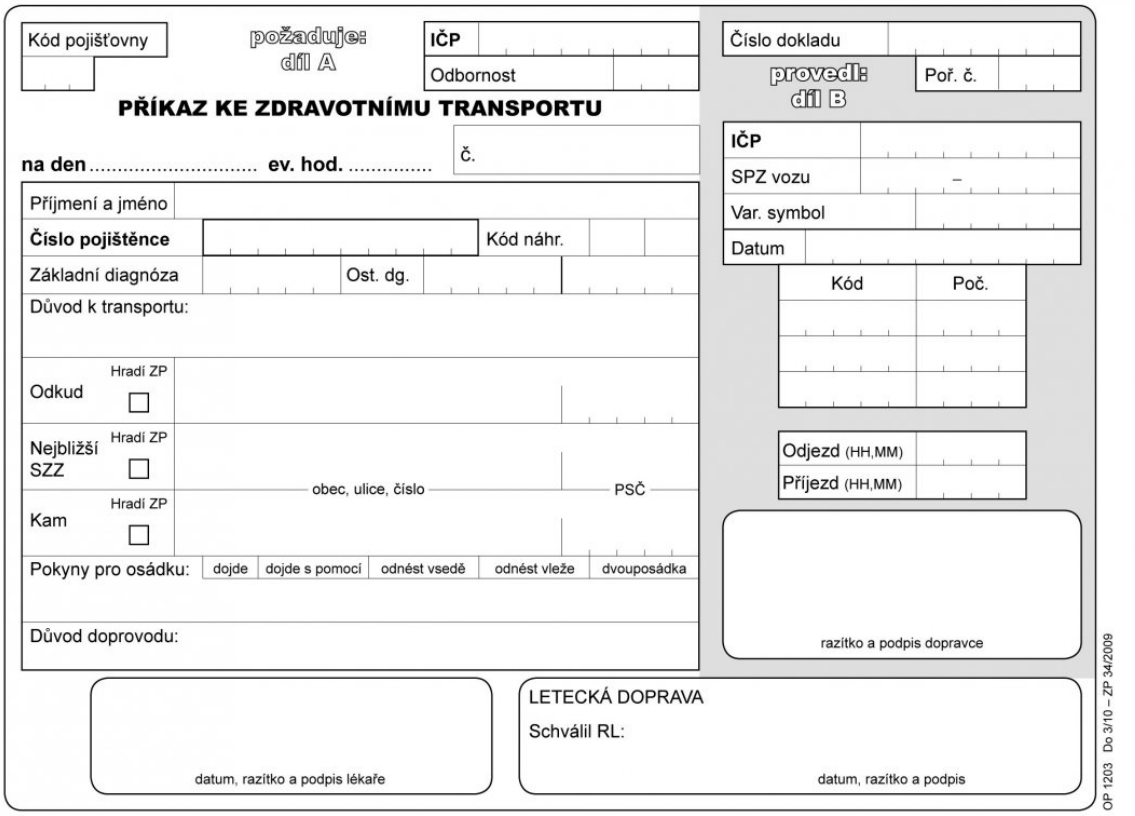
5.1.1 Webová aplikace pro dispečery

Práce dispečera spočívá v první řadě v přijímání objednávek k převozu pacienta, a to buď přímo od samotných pacientů, nebo z nemocničních zařízení. Pacient dostane od svého lékaře žádanku (tzv. příkaz ke zdravotnímu transportu), kterou je možné vidět na obrázku 5.1. Na základě žádanky má pacient převoz uhrazený zdravotní pojišťovnou, se kterou má sanitní služba uzavřenou smlouvu. Údaje z této žádanky nahlásí pacient (resp. zdravotnický personál) dispečerovi, který je uloží do systému.

Kromě přijímání objednávek je dispečer zodpovědný za plánování tras sanitních vozů, tedy seskupení několika pacientů, kteří pojedou jedním sanitním vozem nejlépe v podobném směru, aby sanitní vůz stihl odvést co nejvíce pacientů v co nejkratším čase a na co nejkratší trase. Dispečer musí umět poskládat jednotlivé převozy takovým způsobem, aby byl tento cíl naplněn.

Dispečer musí mít také přehled o sanitních vozech, které má daný den k dispozici, a nakládat s nimi tak, aby nenastala situace, že všechny vozy budou obsazené a zároveň bude nutné realizovat nějakou trasu.

Cílem je tedy vyvinout systém, který bude schopen ukládat jednotlivé převozy, následně z nich skládat trasy a tyto trasy přidělovat jednotlivým sanitním vozům s cílem zjednodušit, zrychlit a zefektivnit práci dispečera.



Obrázek 5.1: Příkaz ke zdravotnímu transportu
„Převzato z https://www.optys.cz/zbozi/prikaz-ke-zdravotnimu-transportu-a5-cislovany-samopropisovaci-2-x-25-listu_1203/.“

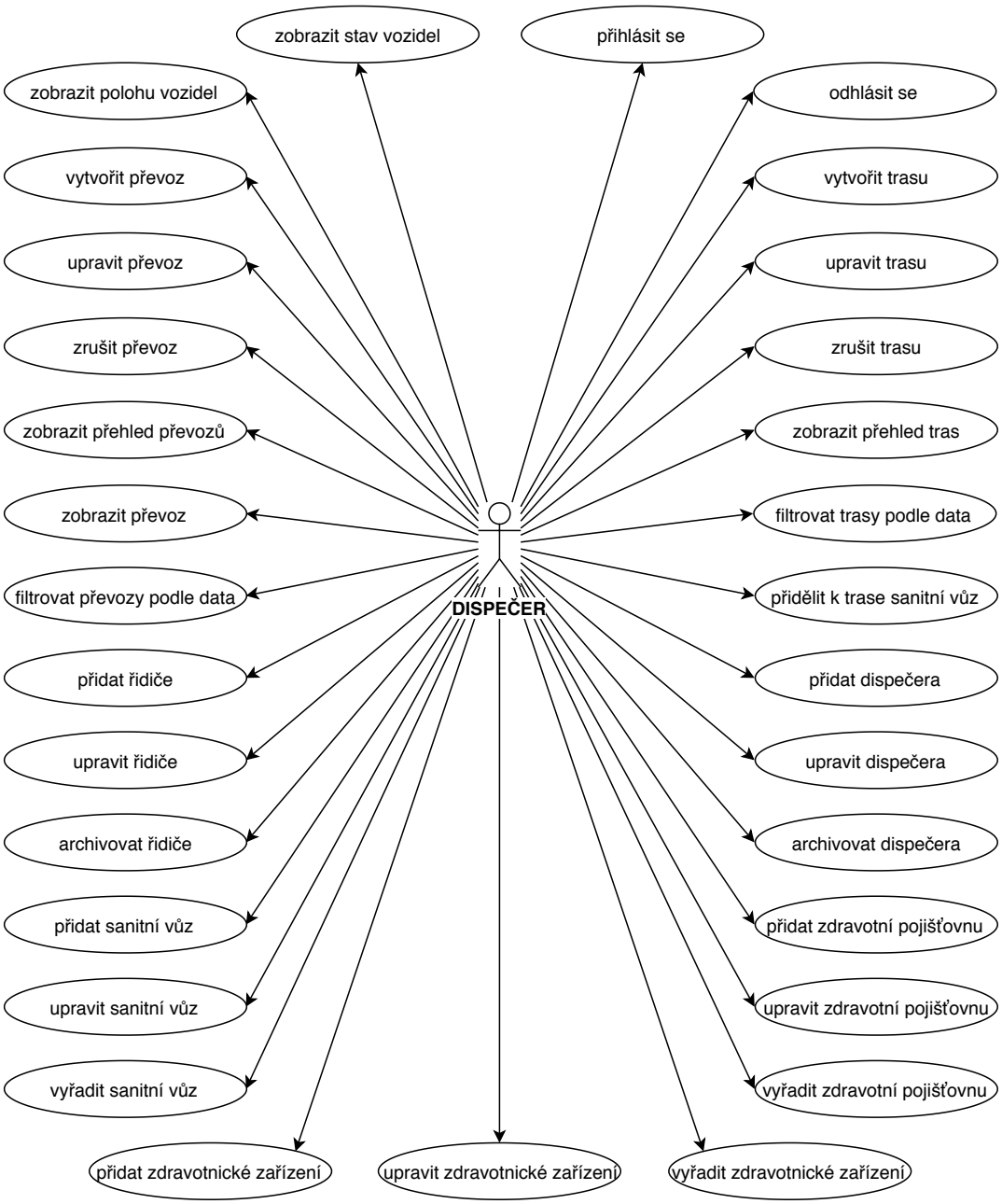
5.1.2 Mobilní aplikace pro řidiče

Pomocí mobilní aplikace se bude řidič přihlašovat do směny a odhlašovat ze směny, aby dispečer věděl, zda je řidič připravený realizovat trasy.

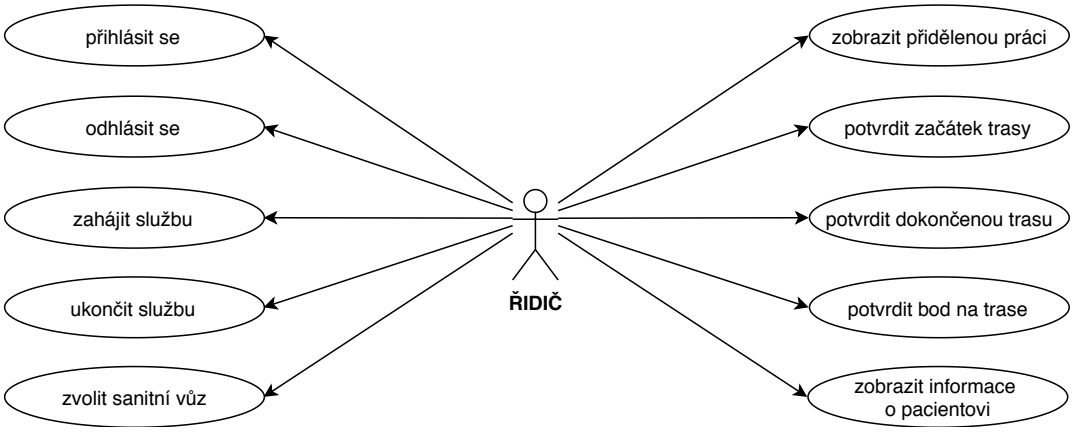
Pro dispečera je také důležité vědět, kde se v daný okamžik vůz nachází, aby mohl řidičům práci efektivně přidělit. Proto bude aplikace sloužit k pravidelnému odesílání GPS souřadnic polohy vozidla, které budou následně vykresleny do interaktivní mapy dispečera.

Zároveň bude mobilní aplikace sloužit k zobrazení přidělené práce přesně v takovém pořadí, v jakém ji má daný řidič vykonat.

5.2 Diagram případů užití



Obrázek 5.2: Diagram případů užití pro webovou aplikaci



Obrázek 5.3: Diagram případů užití pro mobilní aplikaci

5.3 Existující řešení

V současnosti existuje několik systémů pro dispečink, například pro dispečink taxi služeb. Ve velmi omezené míře je možné najít systémy týkající se soukromých sanitních služeb, které ale zahrnují pouze malou část požadavků. Často se ale zabývají zcela jinou činností jako například zpracováním dokladů pro účely vykázání jízd zdravotním pojišťovnám. V této podkapitole se pokusím zhodnotit tato existující řešení a srovnat je s vyvíjeným systémem.

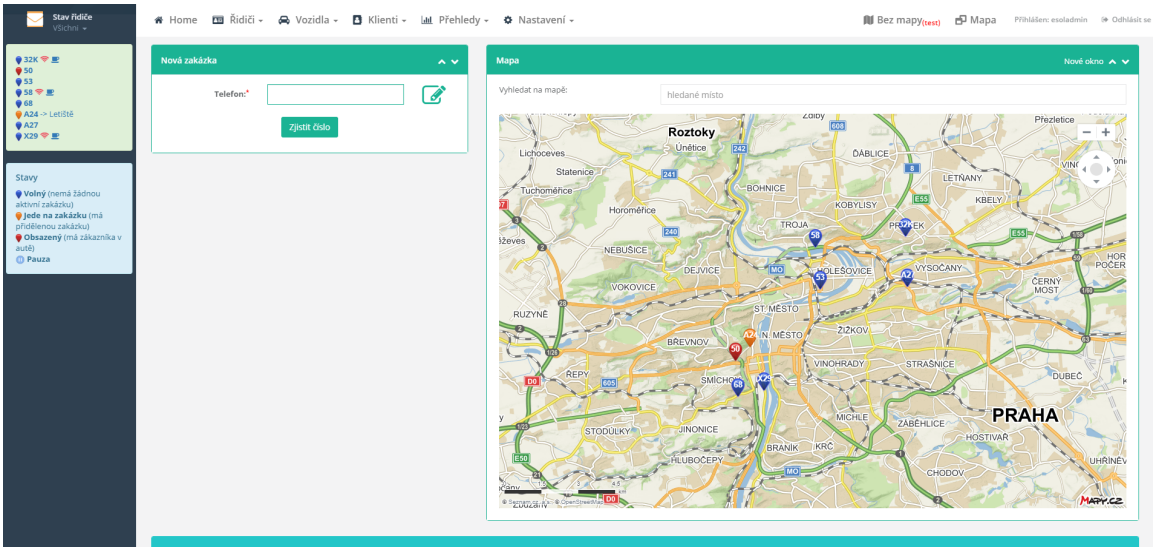
5.3.1 eDispečink

Jedná se o moderní systém pro řízení taxislužby, který vznikl v roce 2008 a v únoru 2018 byl vyvinut znovu od základu a postaven na nových technologiích.

eDispečink[1] zahrnuje dispečerské rozhraní, klientské rozhraní a aplikaci pro řidiče. Dispečerské rozhraní je možné vidět na obrázku 5.4. Je to webová aplikace, ve které může dispečer sledovat vozidla na mapě a zadávat, editovat a přidělovat zakázky, zároveň může také celý systém nastavit a zobrazit si přehledy uskutečněných zakázek. Klientské rozhraní slouží pro objednávání taxi bez nutnosti volat na dispečink. Řidičská aplikace sleduje GPS polohu vozu, umožňuje komunikovat s dispečinkem, přijímat a odbavovat zakázky.

Kdyby nebylo bráno v potaz, že se jedná o systém pro taxi službu, pro účely sanitní služby systém zcela postrádá možnost plánovat jednotlivé trasy skládající se z více převozů, což je pro převozovou sanitní službu nezbytná součást.

[1] https://www.edispecink.cz/



Obrázek 5.4: Dispečerské rozhraní v systému eDispečink
„Převzato z https://www.edispecink.cz/Images/dispecinkScreen1.png.“

5.3.2 Sanitka.info

Program Sanitka.info[2] slouží pro zpracování dokladů týkajících se zdravotní přepravy. S vyvíjeným systémem se shoduje pouze z pohledu automatického počítání vzdálenosti zadaných tras a možností zobrazit trasy na mapovém podkladu. Podle informací z dubna 2019 je připravován kompletní systém pro dispečink s přenosem dat z provozu přímo do programu Sanitka.info. Vzhledem k aktuálnímu nedostatku informací není možné zhodnotit, zda bude nový systém sloužit pouze k evidenci převozů nebo bude umožňovat i pokročilejší funkce.

5.3.3 Meridian

Meridian[3] je sada aplikací zajišťujících efektivní provoz v neurgentních převozových službách. Snaží se snižovat náklady a poskytovat pracovníkům dispečinku informace, které jsou důležité k tomu, aby mohli poskytovat služby na vysoké úrovni. Meridian se soustředí na maximalizaci využití vozového parku, což zajišťuje zobrazením vozidel na mapě s možností zjištění jejich stavu, informovanost personálu dispečinku a snadné a rychlé vytváření výkazů.

Tato sada zahrnuje tři aplikace – mobilní aplikaci pro řidiče vozidel, online rezervační systém a plánovací aplikaci. Mobilní aplikace slouží ke sběru informací o času příjezdu a odjezdu, získávání dat o poloze vozidla a monitorování spotřeby paliva. Rezervační systém slouží k eliminování telefonních, faxových a emailových rezervací a minimalizaci neuskutečněných jízd, přístup do něj mají po vytvoření profilu i samotní klienti. Plánovací aplikace obsahuje nástroje pro včasné plánování jízd, sledování realizace jízd oproti plánovaným časům přepravy a snadnou správu žádostí o přepravu na poslední chvíli.

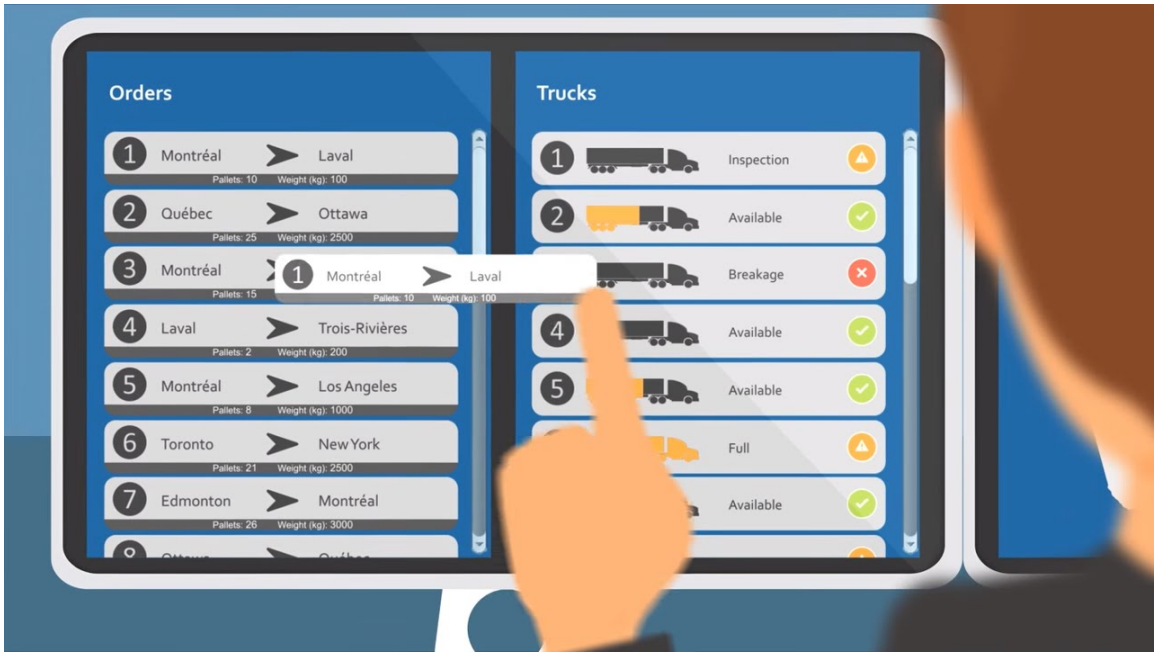
Ačkoliv svými funkcemi se jedná o systém, který je nejblíže vyvíjenému systému, je vytvářen pro požadavky Spojeného království, kde je využíván. Každý stát má odlišné principy fungování převozových sanitních služeb. Kromě absence české lokalizace je dle

[2] http://www.sanitka.info/?page=program
[3] http://www.3tcsoftware.com/software/non-emergency-transport-services/

mého názoru rezervační systém pro klienty v této oblasti nevhodný. Velká část pacientů je v pokročilém věku, často bez chytrého telefonu, případně bez datového připojení.

5.3.4 JFleet

JFleet[4] je systém pro řízení dopravy, který zahrnuje činnosti od přijímání objednávek přes řízení dispečinku až po fakturaci. Primárně je určen pro řízení dopravy u přepravních a doručovacích společností, celý systém je tedy přizpůsoben pro přepravu nákladu v kamionech a dodávkách. Mimo jiné umožňuje vytvářet trasy a sledovat vozidla.



Obrázek 5.5: Systém jFleet
„Převzato z https://i.ytimg.com/vi/TbHxegOcOzk/maxresdefault.jpg.“

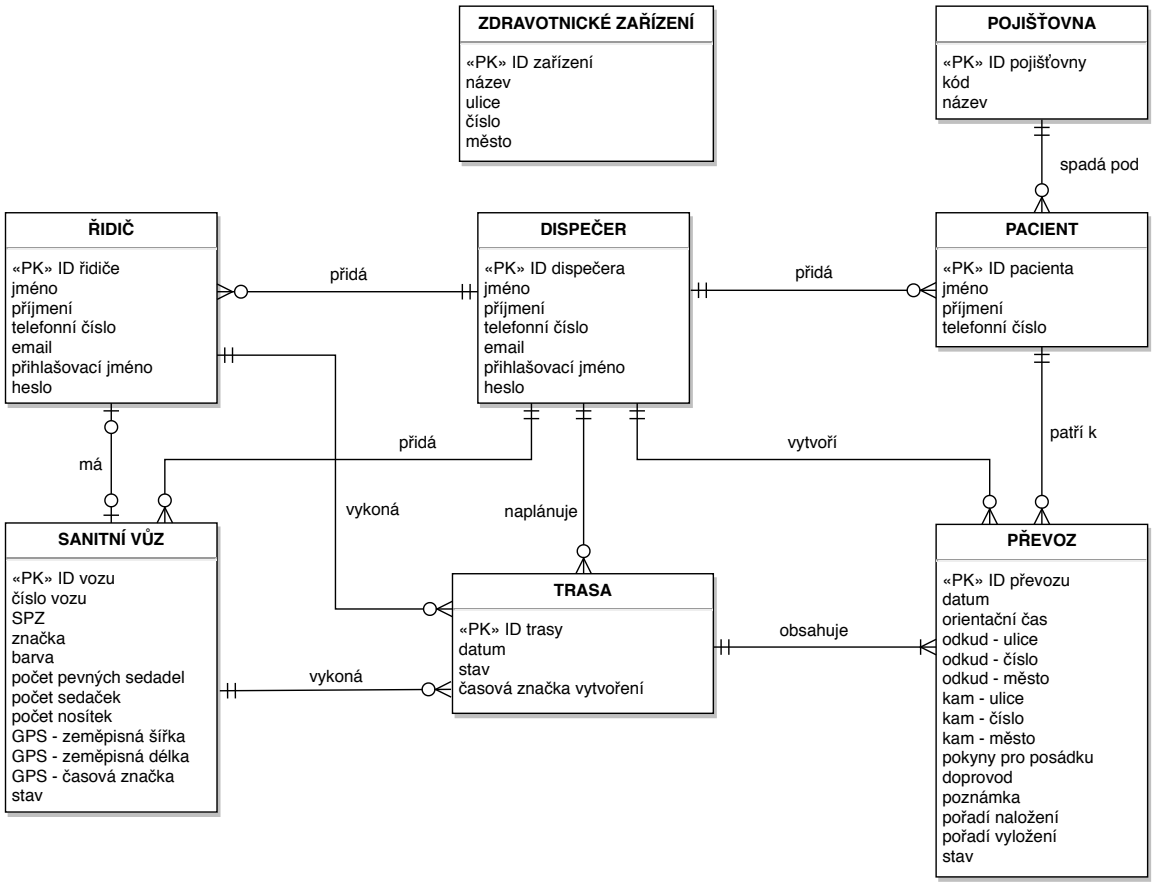
[4]https://www.jfleet.net/

Kapitola 6

Návrh informačního systému

V této kapitole je vyobrazen ER diagram navrženého systému a popsán návrh jednotlivých částí systému pro webovou i mobilní aplikaci.

6.1 ER diagram



Obrázek 6.1: ER diagram

6.2 Návrh webové aplikace

Tato podkapitola obsahuje detailní specifikaci jednotlivých částí, ze kterých bude webová aplikace pro dispečery sestávat. Tyto části vznikly na základě analýzy požadavků na systém.

6.2.1 Interaktivní mapa

Po přihlášení do systému bude mít dispečer k dispozici interaktivní mapu, díky které bude mít přehled o aktuální pozici všech vozidel. Vozidla budou na mapě vyobrazena formou ikony sanitního vozu s číslem daného vozu a v pravidelném intervalu bude jejich pozice obnovována. Pro snadnou orientaci dispečera bude vzhled ikon odlišen – v případě, že bude vozidlo aktivní, tedy přes aplikaci pro řidiče se řidič přihlásí, vybere si dané vozidlo a zahájí na něm službu, ikona sanitního vozu bude barevná, v opačném případě bude ikona černobílá.

Aby nemusel dispečer vozidla na mapě hledat, vedle mapy bude mít seznam všech vozidel formou velkých ikon, které se budou, stejně jako ty na mapě, barevně lišit na základě aktivity daného vozu. U každé ikony sanitního vozu bude k dispozici tlačítko se symbolem lupy, po kliknutí na něj se uskuteční přiblížení požadovaného vozu na mapě. Druhým tlačítkem u každého vozu bude tlačítko pro zobrazení informací o vozu. Po stisku tohoto tlačítka se zobrazí všechny základní informace, které dispečer o daném vozu potřebuje vědět – zda je vůz volný či obsazený, jakou aktuálně vykonává trasu či kdo jej řídí.

6.2.2 Vytvoření převozu

Tato část systému bude sloužit řidiči k uložení nového převozu do databáze. Informace, které dispečer zjistí od volajícího, zapíše do formuláře, který se bude dělit na čtyři části – informace o pacientovi, detaily převozu, adresa počátku převozu a adresa konce převozu.

V části „informace o pacientovi“ vyplní dispečer jméno, příjmení a telefonní číslo pacienta a vybere ze seznamu zdravotní pojišťovnu, pod kterou pacient spadá.

V detailech o převozu vybere datum převozu z kalendáře a orientační čas ze seznamu zahrnujícího části dne – ráno, dopoledne, odpoledne a večer. Bude mít také možnost vybrat, že čas je nespecifikován, případně možnost „ihned“ pro okamžitý převoz. Dále dispečer zvolí z nabídky pokyny pro posádku, případně k převozu přidělí doprovod. V rámci této části bude moci dispečer připsat k převozu poznámku zahrnující například informaci o poschodí či čísle bytu, kde pacient bydlí, nebo o oddělení nemocnice, na kterém bude pacient na sanitní vůz čekat.

V posledních dvou částech zadá dispečer do formuláře adresu počátku převozu a adresu konce převozu. Pro urychlení bude mít u obou částí k dispozici seznam zdravotnických zařízení. Z tohoto seznamu si bude moci zdravotnické zařízení vybrat napsáním několika prvních písmen názvu zařízení nebo listováním v seznamu a adresa se do odpovídajících polí vyplní automaticky.

6.2.3 Přehled převozů

Přehled převozů bude sloužit dispečerovi k zobrazení všech převozů vložených do systému. Převozy bude možné filtrovat podle data a automaticky budou řazeny podle orientačního času. Kromě jména pacienta, počáteční a koncové adresy a orientačního času se bude u každého převozu zobrazovat stav, v jakém se převoz nachází.

Stav může nabývat následujících hodnot:

- čeká na zpracování
- zařazen do trasy
- převoz zahájen
- převoz dokončen
- zrušeno

V rámci přehledu převozů si bude moci dispečer zobrazit detailní informace o každém převozu, případně všechny informace zadané prostřednictvím formuláře pro vytvoření převozu následně upravit. Pokud převoz ještě nebude ve stavu „převoz zahájen“, dispečer jej bude moci zrušit.

6.2.4 Vytvoření trasy

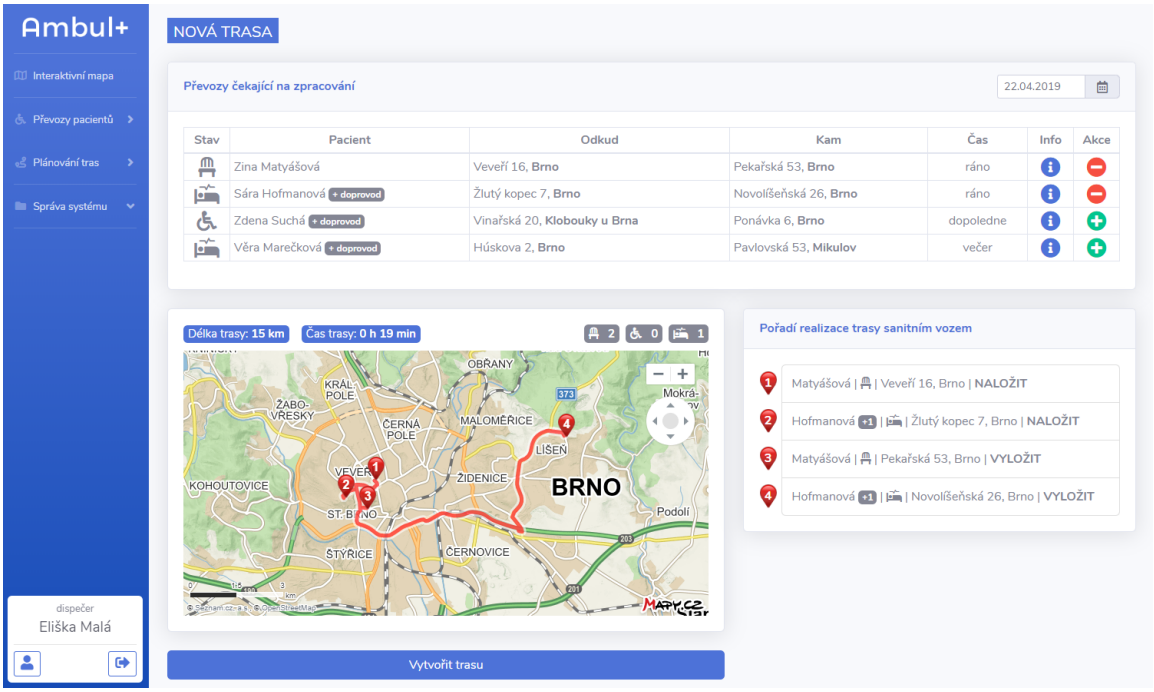
V této části systému bude dispečer plánovat trasy. Jedná se o jednu z nejdůležitějších částí vyvíjeného systému, která by měla především zefektivnit a zjednodušit plánování tras, které v současnosti vyžaduje jisté znalosti území České republiky i schopnosti efektivně naplánovat trasu s ohledem na její délku a čas potřebný k realizaci.

Návrh této části systému je možné vidět na obrázku 6.2. Dispečer bude mít k dispozici tabulku s převozy, které byly pouze vytvořené a nadále s nimi nebylo pracováno – nacházejí se tedy ve stavu „čeká na zpracování“. Tyto převozy v rámci plánování trasy bude dispečer seskupovat a určovat pořadí jejich realizace.

V tabulce převozů bude mít k dispozici stav pacienta vyjádřený symbolickou ikonou, jeho jméno, počátek a konec cesty, orientační čas a případně informaci o doprovodu. To jsou dostatečné údaje k tomu, aby na jejich základě mohl dispečer sestavit trasy pro sanitní vozy. V případě potřeby si bude moci podrobnější informace o pacientovi zobrazit.

Jednotlivé převozy ze zmíněné tabulky bude moci dispečer přesouvat do trasy a z trasy odebírat pomocí tlačítek se symboly plus a minus. Po přesunutí do trasy se převoz rozdělí na dva body, kterými jsou naložení pacienta a vyložení pacienta. Zároveň proběhne vykreslení trasy do mapy a výpočet délky trasy a minimálního času nutného pro její realizaci. Dispečer také bude informován o minimální nutné velikosti sanitního vozu pro realizaci takové trasy.

Převozy v rámci dané trasy bude možné libovolně přesouvat a na základě každého přesunutí bude znovu vykreslena trasa včetně nové délky trasy a minimálního času, aktualizován bude také údaj o minimální nutné velikosti sanitního vozu. Díky této funkci bude moci dispečer i bez znalosti území České republiky plánovat optimální trasy a ušetřit čas řidičům případnými zajížďkami.

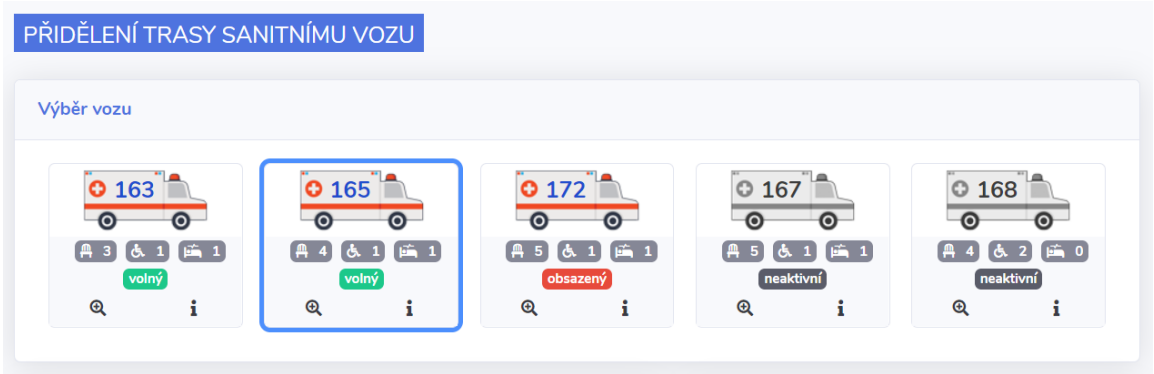


Obrázek 6.2: Vytvoření trasy v aplikaci pro dispečery

6.2.5 Přidělení trasy

Poté, co řidič trasu naplánuje, bude trasa uložena. K této trase je následně nutné přidělit vůz, který ji bude vykonávat. Dispečerovi se zobrazí podobné prostředí jako pro vytváření trasy, avšak místo tabulky pro výběr převozů se mu zobrazí přehled sanitních vozů včetně jejich aktuálního stavu – zda jsou aktivní či neaktivní, volné či obsazené, jakou mají kapacitu, kde se aktuálně nacházejí a případně podrobnosti o nich, které má dispečer k dipozici i v interaktivní mapě.

Označením karty vozu dispečer sanitní vůz vybere a po stisknutí přiřazovacího tlačítka bude vůz k trase přidělen. Na obrázku 6.4 je možné vidět detail výběru vozu pro realizaci dané trasy.



Obrázek 6.3: Detail výběru vozu při přidělení trasy v aplikaci pro dispečery

6.2.6 Přehled tras

Přehled tras bude sloužit dispečerovi k zobrazení všech naplánovaných tras. Trasy bude možné filtrovat podle data a automaticky budou řazeny podle jejich identifikačního čísla, tedy podle jejich pořadí vytvoření.

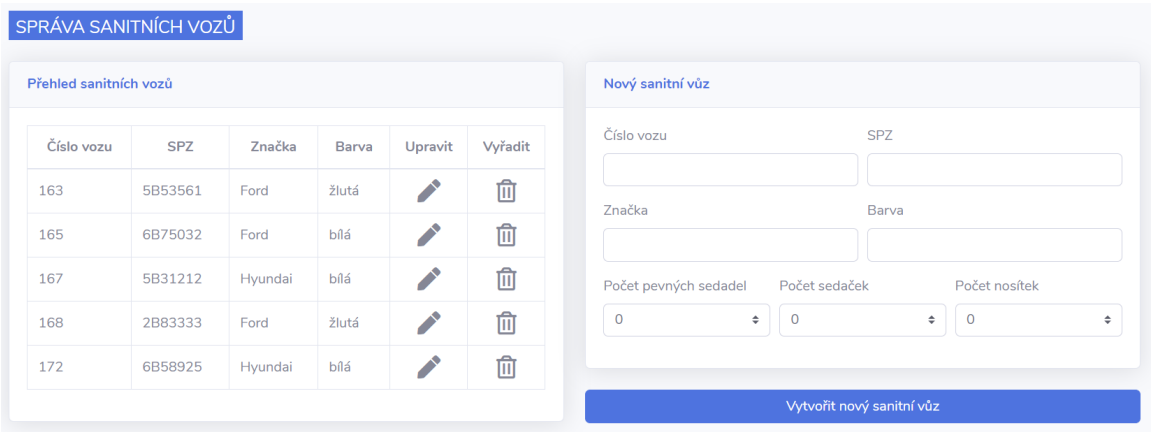
Každá trasa má svůj stav, který slouží k tomu, aby dispečer věděl, v jakém stádiu trasa je. Celkem může nabývat čtyř různých stavů:

- čeká na přidělení
- přidělena
- probíhá
- dokončena

V rámci přehledu tras bude moci dispečer vytvořené trasy upravovat, a to jak z hlediska konkrétních pacientů na dané trase, tak z hlediska pořadí realizace jednotlivých převozů. V případě potřeby bude moci celé trasy i rušit. U každé trasy bude mít také tlačítko pro přidělení sanitního vozu k dané trase.

6.2.7 Správa systému

Poslední částí webové aplikace je správa celého systému, která také patří k práci dispečera. Správa systému bude zahrnovat správu řidičů, sanitních vozů, dispečerů, zdravotních pojišťoven a zdravotnických zařízení.



Obrázek 6.4: Detail správy sanitních vozů v aplikaci pro dispečery

6.3 Návrh mobilní aplikace

Zatímco webová aplikace pro dispečery má za cíl nabízet co nejvíce nástrojů ulehčujících práci dispečera, mobilní aplikace by měla být co nejjednodušší a obsahovat pouze funkce, které jsou pro řidiče i funkčnost celého systému nezbytné. Tato podkapitola obsahuje specifikaci jednotlivých částí, ze kterých bude mobilní aplikace pro řidiče sestávat. Tyto části vznikly na základě analýzy požadavků na systém.

6.3.1 Profil

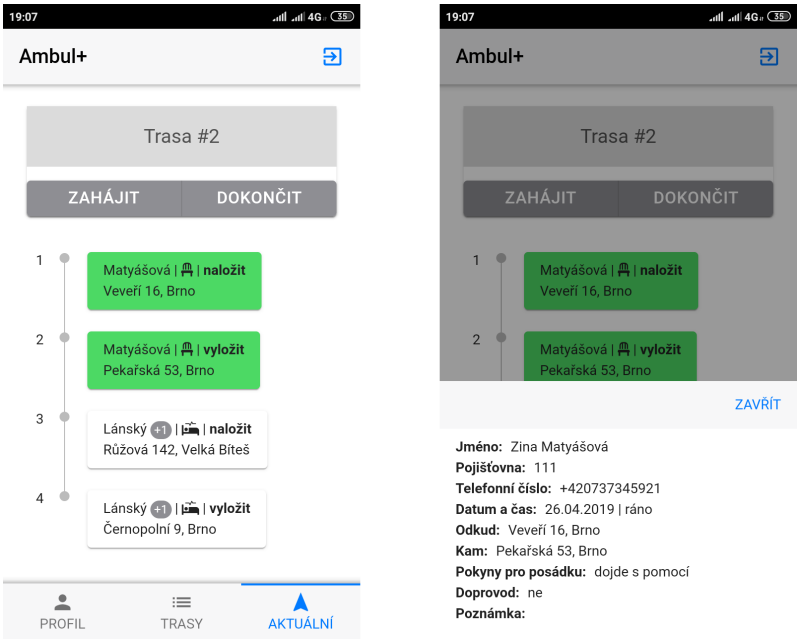
Profil je částí mobilní aplikace, na které se řidič po přihlášení do systému ocitne. Kromě jeho jména a příjmení se zde bude nacházet výběr sanitního vozu, ve kterém bude na dané směně sloužit. Po výběru vozu prostřednictvím tlačítka „zahájit službu“ bude moci zahájit službu, od tohoto okamžiku budou odesílány GPS souřadnice v pravidelném intervalu na server. Stejným tlačítkem, tentokrát s popisem „ukončit službu“, bude moci službu ukončit a souřadnice přestanou být odesílány.

6.3.2 Přehled tras

Přehled tras bude sloužit řidiči k zobrazení tras, které byly dispečerem přiděleny k jeho sanitnímu vozu. Trasy bude možné filtrovat podle data, řidič si tedy bude moci zobrazit i historii uskutečněných tras na daném vozidle, případně zjistit přidělené trasy na další dny.

6.3.3 Aktuální trasa

Při výběru trasy z přehledu tras se zobrazí detail dané trasy a trasa se bude považovat za aktuální trasu. Řidič bude mít možnost tuto trasu zahájit a dokončit a v průběhu uskutečňování trasy jednotlivé body na trase potvrzovat jako dokončené. Díky tomu bude mít dispečer naprostý přehled o realizaci dané trasy.



Obrázek 6.5: Zobrazení aktuální trasy a informací o pacientovi v aplikaci pro řidiče

Kapitola 7

Implementace

Tato kapitola se zabývá implementací webové i mobilní aplikace. Na jejím začátku jsou zmíněny technologie zvolené pro implementaci systému. Další část je věnovaná uživatelskému rozhraní. Následuje popis komunikace s databází a způsob implementace. Zmíněny jsou také komponenty, které byly při implementaci využity. Další podkapitola je věnována výběru mapového API. Na závěr jsou popsány implementační detaily hlavních a zajímavých částí systému.

7.1 Zvolené technologie

Tato podkapitola zmiňuje technologie, které byly vybrány pro implementaci webové aplikace pro dispečery a mobilní aplikace pro řidiče. Detaily jednotlivých technologií byly popsány v kapitolách o vývoji webových a mobilních aplikací, zde je pouze zmíněno, které technologie byly vybrány a případně z jakého důvodu.

7.1.1 Technologie na straně serveru

Pro funkčnost aplikace je nutné mít databázi, do které budou ukládána všechna data a k níž se bude při získávání dat přistupovat. Pro realizaci systému byl zvolen jeden z nejpopulárnějších a nejpoužívanějších relačních databázových systémů, kterým je MySQL. Jedním z hlavních důvodů je, že tento databázový systém patří k nejrozšířenějším a většina webhostingů jej nabízí, což například do budoucna umožňuje nasadit systém na webhosting, na kterém má již firma své webové stránky.

Pro zjednodušení správy obsahu MySQL byl vybrán nástroj phpMyAdmin, který je napsán v jazyce PHP a umožňuje správu obsahu MySQL prostřednictvím webového rozhraní. Tento nástroj umožňuje vytvářet a rušit databáze, provádět SQL příkazy, spravovat klíče a vytvářet, upravovat a rušit databázové tabulky.

Veškerá logika umožňující komunikaci s databází bude naprogramována ve skriptovacím programovacím jazyce PHP. Vzhledem k tomu, že se jedná o interpretovaný jazyk, tak na webovém serveru běží PHP interpret, který skript zpracuje a vytvoří výstup, který je následně předán klientovi.

7.1.2 Technologie na straně klienta

Pro uživatelské rozhraní bude využit značkovací jazyk HTML a kaskádové styly CSS. Aby bylo uživatelské rozhraní ve webové aplikaci co nejpřizpůsobivější různým velikostem zob-

razovacího zařízení, bude využit framework Bootstrap pro vývoj responzivních webových stránek.

Pro vývoj hybridní mobilní aplikace bude využit framework Framework7, který docílí, že aplikace bude působit dojmem nativní aplikace. Zároveň také zajistí, aby byly rozměry elementů dostatečně velké pro ovládání na dotykovém displeji.

Dynamičnost webové i mobilní aplikace zajistí JavaScript. Bude využita převedším javascriptová knihovna jQuery pro snadnou manipulaci s HTML prvky.

Hojně bude také využita technologie AJAX, která slouží pro vývoj interaktivních webových aplikací a umožňuje měnit obsah stránek bez nutnosti jejich kompletního znovunačtení.

7.2 Uživatelské rozhraní

Při plánování uživatelského rozhraní bylo stanoveno několik bodů, které by výsledné uživatelské rozhraní mělo splňovat, aby bylo co nejvíce uživatelsky přívětivé. V případě webové aplikace se jedná především o následující požadavky:

- **Maximální využitelná plocha.** Aplikace pro dispečery by měla být plně využitelná na jednom monitoru, proto je v některých případech nutné na jednu stránku umístit větší množství informací. Uživatelské rozhraní by proto mělo být co nejčistší a mělo by poskytovat maximální možnou plochu využitelnou pro samotný obsah dané stránky.
- **Snadná orientace pro začátečníky.** Zaměstnanci na pozici dispečera se mohou často střídat, proto je nutné, aby porozumění systému bylo co nejjednodušší.
- **Rychle a jednoduše proveditelné akce.** Tento požadavek značí, že všechny úkony, které má dispečer provádět, by měly být co nejvíce „na povrchu“, tedy aby dispečer nemusel k provedení nějaké akce zacházet příliš hluboko do systému. Jednotlivé úkony musí být logicky rozřazeny do kategorií a v rámci těchto kategorií snadno dostupné.

Aby byly výše zmíněné požadavky splněny, bylo zvoleno vertikální menu umístěné na levé straně obrazovky. Původním plánem bylo vytvořit menu sestavené pouze z logických ikon, které by mohlo být nejméně o polovinu užší a poskytnout tak širší využitelnou plochu, avšak právě kvůli jednoduchému pochopení systému pro nové zaměstnance bylo z tohoto plánu ustoupeno ve prospěch menu zkombinovaného z ikon a popisků.

Pro základní kostru systému byla nakonec vybrána šablona *SB Admin 2*[1], která je postavená na nejnovější verzi frameworku Bootstrap. Jedná se o bezplatnou open-source šablonu, která využívá moderní designový styl v kombinaci s jemnými stíny. Šablona byla upravena dle potřeb vyvíjené aplikace, ponecháno bylo z původní šablony především velmi zdařilé vertikální menu doplněné o potřebné elementy.

Také pro mobilní aplikaci bylo potřeba specifikovat požadavky na uživatelské rozhraní, zde hrály roli především následující body:

- **Velká tlačítka.** Mobilní telefon s aplikací bude mít řidič pravděpodobně připevněný na držáku v sanitním vozu, proto je důležité, aby se bez problémů trefil do jednotlivých tlačítek.
- **Vše viditelné na obrazovce.** Žádná skrytá menu dostupná po přejetí prstem po displeji, ale naopak vše přehledně dostupné a viditelné.

[1] https://startbootstrap.com/themes/sb-admin-2/

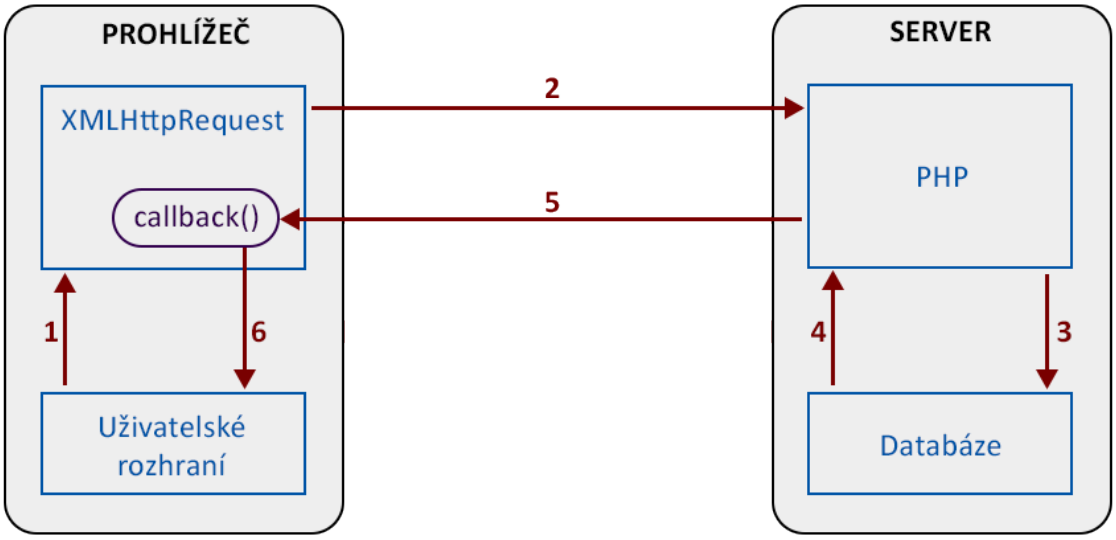
Vzhledem k tomu, že se jedná o méně rozsáhlou aplikaci s menším počtem funkcí, avšak z pohledu rozhraní poměrně specifickou, nebyla pro její vývoj vybrána žádná šablona. Aby ale aplikace splňovala alespoň základní parametry pro vzhled mobilních aplikací, pro její sestavení byly využity komponenty, které nabízí Framework7.

7.3 Komunikace s databází

S cílem optimalizovat počet načítání stránek aplikace jsem se rozhodl pro technologii AJAX, která umožňuje měnit obsah stránek bez nutnosti jejich kompletního znovunačtení, a to za pomoci javascriptové knihovny pro asynchronní zpracování webových stránek. Díky technologii AJAX má uživatel pocit, že je práce mnohem plynulejší jako například u desktopových aplikací. Problémem technologie AJAX může být síťová latence, tedy doba uplynulá mezi akcí a reakcí. V takovém případě je dobré uživatele o zpracovávání informovat [29].

7.3.1 Proces výměny dat prostřednictvím technologie AJAX

Jak je možné vidět na obrázku 7.1, proces výměny dat prostřednictvím technologie AJAX lze rozdělit do 6 kroků. Jednotlivé kroky jsou popsány pod obrázkem.



Obrázek 7.1: Proces výměny dat prostřednictvím technologie AJAX

1. Uživatel odešle požadavek z uživatelského rozhraní, událost je obsloužena javascriptovým handlerem.
2. Je vytvořen objekt `XMLHttpRequest`, přes tento objekt je na server odeslán HTTP požadavek.
3. Server komunikuje s databází za pomoci jazyka PHP.
4. Data jsou z databáze získána.
5. Server odesílá data ve formátu XML nebo JSON funkci zpětného volání.
6. Data jsou zpracována a pomocí HTML a CSS zobrazena na stránce.

7.3.2 Implementace technologie AJAX prostřednictvím jQuery

jQuery nabízí řadu funkcí, které zjednodušují práci s technologií AJAX. V čistém JavaScriptu bychom identický kód, který v jQuery zabere několik řádků, museli napsat na několik desítek řádků.

Základem všech asynchronních požadavků v jQuery je funkce `$.ajax()`, kterou je možné dále konfigurovat pomocí dodatečných nastavení. Funkce sama o sobě pouze načte aktuální obsah stránky, avšak se získaným výsledkem nic neprovede.

Odesílání dat ke zpracování

Prvním případem, kdy je technologie AJAX v aplikaci nápomocna, je odesílání dat ke zpracování PHP skriptem. Toho je využíváno v případě, kdy máme PHP skript, pomocí něhož budou do databáze vkládána nová data nebo budou existující data upravována.

Jednotlivé parametry funkce `$.ajax()` se nastavují prostřednictvím dvojice klíč–hodnota. Mezi základní parametry patří:

- `type` – HTTP metoda, která bude použita
- `data` – data, která budou odeslána na server
- `url` – URL adresa, na kterou bude požadavek zaslán
- `beforeSend` – funkce, která se vykoná před odesláním požadavku
- `success` – funkce, která se vykoná po úspěšném dokončení požadavku; proměnná `data` obsahuje data přijatá ze serveru

Přijetí dat ze serveru

Druhým případem, kdy je technologie AJAX v systému využívána, je přijímání dat ze serveru. Taková situace nastává v případě, kdy máme PHP skript, pomocí nehož jsou získávána data z databáze.

V tomto případě již není použita obecná funkce `$.ajax()`, ale speciální funkce pro čtení dat zakódovaných do formátu JSON ze serveru přes GET HTTP požadavek. První parametr `url` specifikuje URL adresu, na kterou bude požadavek zaslán. Druhým parametrem je funkce, jejíž parametr `result` obsahuje získaná data ve formátu JSON. Pro zpracování těchto dat je využita funkce `$.each()`, která slouží jako iterátor nad objekty i poli. Ukázka dat ve formátu JSON, která jsou zpracovávána, je zobrazena níže.

```
[{"id_pojistovny":"1","kod":"111","nazev":"V\u0161eobecn\u00e1
 zdravotn\u00ed poji\u0161\u0165ovna"},
 {"id_pojistovny":"2","kod":"201","nazev":"Vojensk\u00e1
 zdravotn\u00ed poji\u0161\u0165ovna"},...]
```

V ukázce je možné vidět, že znaky s diakritikou jsou zakódovány, a to z důvodu správného přenosu dat z databáze až na stránku. O konvertování objektů a zakódování znaků s diakritikou do formátu JSON se stará PHP funkce `json_encode()`.

7.4 Využité komponenty

V této podkapitole jsou zmíněny komponenty, které jsou v systému využity. U každé komponenty je zmíněn důvod a způsob jejího využití.

7.4.1 SortableJS

SortableJS je knihovna napsaná v jazyce JavaScript, která slouží k vytváření přeskupitelných seznamů za pomocí techniky *drag-and-drop* neboli „táhni a pusť“. Své využití najde nejen na dotykových displejích, ale i na běžných obrazovkách, protože není jednodušší způsob pro vytváření pořadí než tento.

A právě s pořadím souvisí využití této knihovny ve vyvíjeném systému. Knihovna byla využita v části systému týkající se plánování trasy, kde dispečer určuje pořadí realizace trasy sanitním vozem. Je to elegantní způsob, jak dispečerovi umožnit sestavit trasu, aniž by musel pořadí určovat například psaním číslovek k jednotlivým bodům trasy či postupně přidávat jednotlivé body v pořadí, v jakém mají být realizovány.

7.4.2 Select2

Select2 by bylo možné nazvat „chytrým výběrovým okénkem“. Jedná se o náhradu běžných výběrových okének, která jsou doplněna o možnost vyhledávání, značkování, nekonečné rolování a mnoho dalších možností.

Jak je možné vidět na obrázku 7.2, tento nástroj byl využit pro výběr zdravotnického zařízení při vytváření nového převozu. Důvodem pro jeho využití oproti běžnému výběrovému okénku, které je možné v jazyce HTML vytvořit pomocí elementu `<select>`, bylo uvědomění, že při větším počtu zdravotnických zařízení přidaných do systému by bylo rolování v seznamu zbytečně časově náročné. V tomto případě dispečer pouze zadá první písmeno či písmena zdravotnického zařízení, které hledá, a výsledky se mu okamžitě filtrují.

Obrázek 7.2: Chytré výběrové okénko vytvořené pomocí nástroje Select2

7.4.3 FancyBox

FancyBox je nástroj pro zobrazování multimediálního obsahu moderním způsobem, a to formou modálních oken. Tento nástroj byl vytvořen s využitím knihovny jQuery.

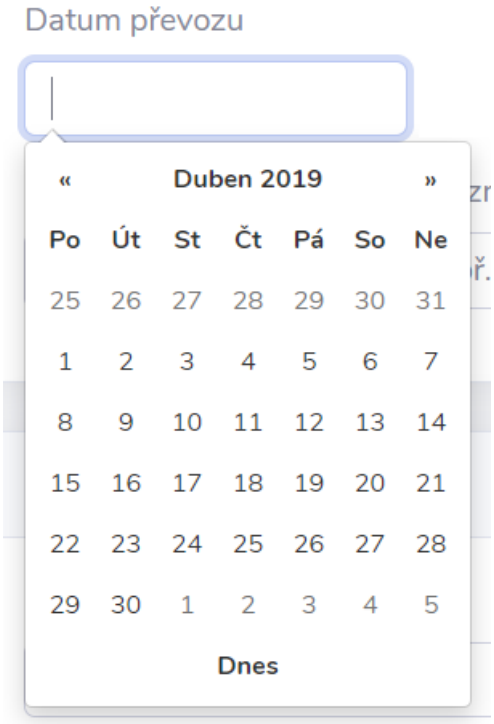
Ve webové aplikaci je využit v několika situacích. První z nich jsou dialogová okna, která se uživateli zobrazují při vkládání nových převozů do systému, rušení převozů a tras, úkonech spojených se správou systému a na mnoha dalších důležitých místech. V této situaci byl využit především z estetického důvodu, jelikož standardní potvrzovací dialog není příliš vzhledný, nepůsobí dojmem provázanosti s uživatelským rozhraním systému a v každém prohlížeči se svým designem liší.

Dalším místem, kde je využit, jsou modální okna vyobrazující podrobné informace o daném pacientovi či sanitním voze. Své využití nachází také při výběru zdravotnického zařízení v rámci vytváření nového převozu. V obou případech je hlavním důvodem využití rychlost zobrazení informací bez nutnosti přechodu na jinou stránku systému.

7.4.4 Bootstrap Datepicker

Bootstrap Datepicker je doplněk, který nabízí pohodlný výběr data prostřednictvím kalendáře. Kalendář nabízí velké množství nastavení – od nastavení formátu data přes určení počátečního dne týdne či omezení přechodů mezi úrovněmi kalendáře (dny, měsíce, roky, staletí) až po speciální tlačítko pro vložení dnešního data a výběr více dat. Velkou výhodou ve srovnání s ostatními podobnými řešeními je přítomnost české lokalizace.

Tento doplněk byl v aplikaci využit pro zadávání data převozu, jak je možné vidět na obrázku 7.3, kde byl doplněn o rychlé tlačítko pro výběr dnešního data, a pro filtrování podle data v přehledu převozů a přehledu tras.



Obrázek 7.3: Využití Bootstrap Datepickeru pro zadávání data převozu

7.5 Mapová API

Výběr vhodného mapového API byl základní cíl pro veškerou práci s mapou, kterou aplikace nabízí. Mezi nejznámější mapová API patří Google Maps, Mapy.cz, TomTom, Mapbox a OpenLayers. Do užšího výběru postoupily první dvě zmíněné, kterým se budu dále věnovat.

7.5.1 Google Maps

Google Maps patří bezpochyby mezi celosvětově nejvyužívanější a nejznámější API, a to zaslouženě, jelikož spektrum funkcí, které nabízí, je opravdu široké. Každý měsíc k němu přistupuje přes miliardu uživatelů a nabízí 99% pokrytí světa. Celé API rozděluje Google do tří částí – mapy, trasy a místa. Mapy nabízejí možnost vytvářet plně přizpůsobitelné statické i dynamické mapy s různým přiblížením, Street View snímky a 360° snímky. Trasy umožňují nacházet nejlepší způsob, jak se dostat z místa A do místa B, a to také díky aktuálním datům z reálného provozu. Místa shromažďují více než 150 milionů míst po celém světě, která mohou být uživatelům užitečná. U každého z těchto míst jsou k dispozici údaje jako adresa, telefonní číslo či grafy návštěvnosti v jednotlivých dnech a hodinách.

V posledních letech se ale podoba Google Maps API výrazně proměnila. Již v polovině roku 2016 zrušil Google možnost implementace map bez API klíče. Důvodem k tomuto kroku bylo především obrovské množství požadavků. Google se rozhodl, že chce mít o uživatelích přehled a v případě překročení 25 tisíc požadavků (počtu volání) denně bude uživatel informován o možnosti placeného programu. Pro mnoho menších či začínajících projektů to nečinilo žádný problém a velké ziskové projekty byly přesunuty do placeného programu. V polovině roku 2018 ale ze strany Googlu došlo ke kompletnímu zpoplatnění API a výraznému navýšení cen. Aby Google vyhověl kritice ze strany uživatelů, v současném modelu nabízí vývojářům každý měsíc kredit ve výši 200 dolarů, jehož výše se snižuje s každým uskutečněným požadavkem na API. Existují 2 možné scénáře – buď vývojář uvede údaje své platební karty a při vyčerpání kreditu mu z ní budou odečítány peníze za nadlimitní uskutečněné požadavky, nebo si nastaví limit, čímž zamezí nechtěnému odečítání z platební karty, avšak po zbytek měsíce mu v jeho aplikaci místo mapy zůstané prázdný bílý obdélník.

Pro zajímavost jsou v tabulce 7.1 uvedeny zaokrouhlené ceny některých zpoplatněných služeb[2] platné k dubnu 2019:

Název služby	**Cena za tisíc požadavků**	
	do 100 tisíc požadavků měsíčně	od 100 tisíc požadavků měsíčně
Dynamická mapa vytvářená přes JavaScript	160 Kč	130 Kč
Street View panorama	320 Kč	250 Kč
Trasa s maximálně 10 průjezdními body	110 Kč	90 Kč
Trasa s více průjezdními body + info z provozu	230 Kč	180 Kč
Nalezení místa na mapě + adresa	390 Kč	310 Kč
Nalezení místa na mapě + adresa + kontakt	450 Kč	360 Kč

Tabulka 7.1: Orientační ceny některých zpoplatněných služeb Google Maps API

[2] https://cloud.google.com/maps-platform/pricing/sheet/

Původním záměrem bylo v aplikaci využít bez rozmýšlení Google Maps API, které je svou propracovaností opravdu bezkonkurenční, avšak jak z tabulky 7.1 vyplývá, nejedná se o zanedbatelné částky. I když bychom vzali v potaz, že máme k dispozici kredit ve výši 200 dolarů, nikde již není dáno, že tento kredit nebude v budoucnu zrušen, případně výrazně snížen. Proto jsem se rozhodl využít Mapy.cz, které, jak můžete záhy zjistit, mohou být důstojnou alternativou.

7.5.2 Mapy.cz

Za projektem Mapy.cz stojí česká společnost Seznam a jedná se tak o mapy určené především pro české uživatele, což může být velkou výhodou, jelikož je tak na detailní zpracování map pro Českou republiku kladen mnohem větší důraz. Kromě mnoha základních funkcí, které nabízí většina podobných služeb, se mohou Mapy.cz chlubit například vyznačením parkovacích zón, přehledem cen pohonných hmot, návrhy výletů a zajímavých míst v okolí, hodnocením institucí či zobrazením panoramatických snímků a 3D pohledů.

Nejdůležitější je zmínit, že API je zcela zdarma, a to i pro komerční účely, což je oproti Google Maps zásadní rozdíl. Nejsou zde dokonce žádná omezení týkající se počtu uskutečněných požadavků, takže API je možné využít zdarma i pro rozsáhlé a hojně navštěvované projekty. Co se týká samotného API, je rozděleno do většího počtu částí – vrstvy, ovládací prvky, značky, geometrické prvky, vizitka, panorama, utility, našeptávač a ostatní. Po delším zkoumání dokumentace je možné zjistit, že co se funkcí týká, není jich výrazně méně než u Google Maps. Zásadní problém ale tkví v dokumentaci, která je velmi zanedbaná a obsahuje pouze nezbytné informace. Oproti tomu dokumentace ke Google Maps API nabízí velké množství ukázkových kódů a velmi podrobně popsané všechny funkce, které nabízí. Není se ale čemu divit, že u služby, která nabízí své API zcela zdarma, nebude na dokumentaci brán takový zřetel.

7.6 Implementační detaily hlavních částí systému

Tato podkapitola se věnuje dosud nezmíněným detailům fungování některých částí systému, které považuji za důležité či zajímavé z pohledu implementace.

7.6.1 Přihlášování do webové i mobilní aplikace

Přihlašování dispečerů i řidičů do systému probíhá na základě autentizace uživatelského jména a hesla. Podle uživatelského jména je v databázi vyhledáno odpovídající heslo a porovnáno s heslem zadaným. Pokud je porovnání úspěšné, uživatel získává přístup do systému.

Za účelem zvýšení bezpečnosti je heslo v databázi uloženo zakódované pomocí hašovacího algoritmu. Pro zakódování hesla byl vybrán hašovací algoritmus SHA-512, který byl navržen pro použití v DNSSEC. Jedná se o algoritmus, který vstupní řetězec zakóduje tak, že na výstupu je řetězec o délce 128 znaků. Srovnání nejpoužívanějších hašovacích algoritmů je možné vidět v tabulce 7.2.

Hašovací algoritmus	Délka výstupního řetězce	Bitové operace
MD5	32 znaků	AND, OR, XOR, ROT, ADD
SHA-1	40 znaků	AND, OR, XOR, ROT, ADD
SHA-224	56 znaků	AND, OR, XOR, ROT, ADD, SHR
SHA-256	64 znaků	AND, OR, XOR, ROT, ADD, SHR
SHA-384	96 znaků	AND, OR, XOR, ROT, ADD, SHR
SHA-512	128 znaků	AND, OR, XOR, ROT, ADD, SHR

Tabulka 7.2: Srovnání hašovacích algoritmů

Po úspěšném přihlášení je vytvořena relace, která uchovává údaje o uživateli, který je přihlášen. Při odhlášení ze systému jsou smazána veškerá data uložená v relaci.

Webová aplikace uplatňuje způsob vytváření relace na straně serveru. Pomocí PHP funkce `session_start()` je zahájena relace. Následně je možné přiřazovat hodnoty do relace prostřednictvím superglobálního pole `$_SESSION[]`, se kterým je možné pracovat jako s běžným polem. Zavoláním funkce `session_destroy()` je možné smazat všechna data uložená v relaci.

V mobilní aplikaci je pro stejný účel využito úložiště localStorage, které slouží k ukládání dat, která není potřeba přenášet na server. Uložení hodnoty do úložiště je možné prostřednictvím funkce `localStorage.setItem()`, které předáme název položky jako první parametr a hodnotu jako druhý parametr. Získat hodnotu z úložiště lze pomocí funkce `localStorage.getItem()`, které předáme název položky. Vymazat úložiště je možné prostřednictvím funkce `localStorage.clear()`, čímž docílíme odhlášení ze systému.

7.6.2 Vytváření uživatelů a jejich správa

Správa uživatelských účtů dispečerů i řidičů probíhá prostřednictvím webové aplikace, kde je možné vytvářet nové uživatele, editovat existující uživatele a případně uživatele archivovat.

Na počátku zakládání nového účtu je vytvořena dočasná kopie všech přihlašovacích jmen řidičů či dispečerů. Následně během vyplňování příjmení uživatele se průběžně generuje přihlašovací jméno, které se porovnává se současnými přihlašovacími jmény, k čemuž byla vytvořena rekurzivní funkce `uniqueLogin()`. Pokud by došlo ke shodě, za přihlašovací jméno je doplněna číslovka a porovnání proběhne znovu. V případě opětovné shody by byla v každém kroku číslovka inkrementována až do doby, než by bylo nalezeno unikátní přihlašovací jméno. Vzhledem k tomu, že se jedná o systém, který bude mít maximálně desítky zaměstnanců, doba běhu funkce je téměř zanedbatelná. Heslo si uživatel zvolí sám.

Při úpravě uživatelského účtu je možné heslo změnit. Zde byla řešena situace, kdy dispečer bude chtít uživateli změnit například telefonní číslo, čímž by ale resetoval heslo. Heslo by bylo možné přenášet spolu s ostatními údaji, ale bylo by to zbytečné bezpečnostní riziko. Proto byl zvolen způsob řešení, kdy pole pro heslo zůstane prázdné a pokud jej dispečer nevyplní, v databázi nebude původní heslo změněno, v opačném případě se u účtu nastaví heslo nové. Dispečer je na toto chování upozorněn prostřednictvím informační hlášky.

Archivace uživatelů je dalším bezpečnostním krokem, kterým je možné především zamezit bývalým zaměstnancům přístup do systému. Při přihlášení se stav účtu kontroluje, v případě archivovaného zaměstnance je autentizace neúspěšná a uživatel je o situaci informován.

7.6.3 Plánování trasy sanitního vozu

Jednou z nejsložitějších částí na implementaci z pohledu počtu operací a dodržení pořadí navazujících operací bylo plánování trasy sanitního vozu. Implementaci lze rozdělit do dvou částí – vytváření nové trasy a úprava existující trasy.

Při vytváření nové trasy se nejdříve zavolá funkce `getTransports()`, která prostřednictvím technologie AJAX získá nezpracované převozy pro zvolené datum a sestaví z nich tabulku, a konstruktor mapy, kterému se mimo jiné předá název prvku, do kterého se mapa vytvoří. Po stisku tlačítka pro přidání do trasy, které se nachází u každého převozu, se zavolá funkce `addToRoute()`, která vytvoří 2 nové položky v seznamu – jednu pro naložení pacienta a druhou pro vyložení pacienta. Průběžně jsou sledovány změny v přeskupitelném seznamu pořadí realizace trasy a při každé změně se provedou následující operace:

- Zkontroluje se, zda se změnil index položky, která byla přesunuta. Pokud by byla vrácena na původní místo, nic dalšího se neprovede.
- Provede se kontrola jednotlivých položek seznamu a vzájemných návazností. Zde může být například odhaleno, že dispečer zařadil vyložení pacienta do trasy dříve než jeho naložení. V takovém případě je pořadí automaticky vráceno do předchozího stavu a dispečer je vizuálním efektem informován, kde v seznamu vznikla chyba.
- Pokud jsou všechny kontroly z předchozího kroku úspěšné, zavolá se funkce `correctMoveDetected()`, která:
  - Odstraní z mapy vrstvy se současnou podobou trasy.
  - Zavolá funkci `checkOccupancy()`, která vypočítá maximální obsazenost sanitního vozu během dané trasy, což bude nezbytně nutné vědět při přiřazování sanitního vozu k dané trase, aby byl vybrán sanitní vůz s vhodnou kapacitou.
  - Uloží adresy jednotlivých bodů na trase v odpovídajícím pořadí do pole.
  - Zavolá funkci `redrawMap()`, které předá pole adres.
- Funkce `redrawMap()` provede následující:
  - Pro každou adresu provede dopředné geokódování, které má za cíl získat souřadnice dané adresy.
  - Po získání všech souřadnic a jejich uložení do dvourozměrného pole je zavolána funkce `drawRoute()`, které je pole souřadnic předáno.
- Funkce `drawRoute()` se postará o následující úkony:
  - Vytvoří dvě nové vrstvy – pro vykreslení trasy a pro vykreslení značek.
  - Realizuje požadavek na vyhledání trasy zavoláním konstruktoru `SMap.Route()` a předáním souřadnic ve správném formátu.
  - Proběhne vykreslení trasy do odpovídající vrstvy.
  - Pro každou dvojici souřadnic se volá konstruktor `SMap.Marker()`, který vytvoří značku na odpovídajících souřadnicích a přidá ji do vrstvy značek.
  - Proběhne vykreslení vrstvy značek.

Jak je vidět, operace na sebe navazují a často potřebují kooperovat s daty získanými z předchozí operace, proto při implementaci bylo nutné řešit mnoho problémů týkajících se příliš brzkého vykonání následující operace. Řešením, které tomu zabránilo, bylo využití konstrukcí zvaných *promise*, což je způsob, jakým je možné se vypořádat s asynchronními operacemi. *Promise* znamená „příslib“ a označuje hodnotu, která ještě není známá. Princip je tedy takový, že pomocí `new Promise()` tyto přísliby vytvoříme a následně kód, který je všechny potřebuje mít dokončené, obalíme do konstrukce `Promise.all().then()`. Tento úsek kódu se vykoná až v okamžiku, kdy všechny přísliby budou označené jako dokončené.

Podobnou situaci bylo nutné řešit i při upravování již existující trasy. Trasa se musí sestavit podle pořadí uvedeného u jednotlivých převozů v rámci trasy, což při nesprávné implementaci způsobovalo libovolné poskládání převozů v rámci trasy, ačkoliv v databázi bylo jejich pořadí uloženo správně. Zde bylo opět využito konstrukce *promise*, která zajistila načtení přesně v takovém pořadí, v jakém byla trasa uložena.

7.6.4 Odesílání GPS souřadnic

Tato část se týká mobilní aplikace pro řidiče, která odesílá v pravidelném intervalu získané souřadnice na server. Pro zahájení odesílání souřadnic slouží tlačítko „Zahájit službu“, které zavolá funkci `startGeoLoc()`. Tato funkce zkontroluje, zda byl vybrán vůz, a pokud ano, v pravidelném intervalu volá funkci `runGeolocation()`.

Tato funkce zavolá metodu `navigator.geolocation.getCurrentPosition()`, která pokud proběhně úspěšně, zavolá funkci `onSuccess()`, té předá získané souřadnice a tato funkce je pomocí technologie AJAX odešle na server. Pokud získání souřadnic ze zařízení neproběhne úspěšně, což se může stát například při vyčerpání časového limitu pro získání souřadnic, zavolá se funkce `onError()`.

Po opětovném stisku tlačítka se zavolá funkce `stopGeoLoc()`, která ukončí volání funkce `runGeolocation()` a odešle na server změnu stavu vozu na „neaktivní“.

Kapitola 8

Testování

Testování patří mezi nezbytnou součást vývoje jakéhokoliv softwaru. Jeho cílem je odhalit chyby, které je následně nutné opravit. Testování proto obvykle probíhá v několika iteracích. Během implementace byla webová i mobilní aplikace průběžně důkladně testována. Jednotlivé způsoby testování jsou popsány v následujících podkapitolách.

Veškeré testování probíhalo bez využití automatických testů, a to především z důvodu vysoké náročnosti pro jejich vytvoření. Jako mnohem efektivnější řešení se jevilo manuální testování systému po částech a až nakonec jako celku. Díky tomu mohly být chyby včas zachyceny a opraveny, což eliminovalo zdlouhavé hledání chyb při závěrečných fázích testování.

8.1 Testování programátorem

První fází testování bylo testování samotným programátorem. Vždy po vytvoření části systému, kterou bylo možné důkladně otestovat, bylo zahájeno testování, během kterého byly odhalené chyby průběžně opravovány. Vzhledem k tomu, že na sebe do jisté míry navazují některé části webové aplikace i části mezi webovou a mobilní aplikací, obě aplikace byly vyvíjeny souběžně. Před začátkem implementace bylo promyšleno přesné pořadí implementace, které bylo následující: skript pro vytvoření databáze a naplnění vzorovými daty → odesílání GPS souřadnic → vytvoření převozu → přehled převozů → interaktivní mapa → vytvoření trasy → přehled tras → přidělení sanitního vozu → zobrazení tras řidiči → správa systému → přihlašování. V uvedeném pořadí mohly být jednotlivé části otestovány, aniž by k otestování dané části chyběla nějaká funkčnost systému.

8.1.1 Testování webové aplikace

Jak již bylo zmíněno v kapitole 2.1, webová aplikace je aplikace, která je přístupná za pomoci webového prohlížeče. A ačkoliv Google Chrome jakožto webový prohlížeč zaujímá první místo v oblíbenosti a používá jej přes 63 % uživatelů, existují i další prohlížeče, na kterých je třeba aplikaci otestovat.

Testování bylo provedeno nejdříve manuálně v základních prohlížečích. Ověřována byla nejen funkčnost jednotlivých částí systému, ale také grafická podoba uživatelského rozhraní.

Pro testování byly použity následující webové prohlížeče:

- Google Chrome – verze 73
- Mozilla Firefox – verze 63
- Opera – verze 60
- Microsoft Edge – verze 42
- Internet Explorer – verze 11

V případě prohlížečů Mozilla Firefox, Opera a Microsoft Edge se jednalo pouze o drobné problémy týkající se vzhledu uživatelského rozhraní, protože některé CSS vlastnosti mají specifické hodnoty pro různé typy prohlížečů. Tyto specifické hodnoty byly do kódu doplněny. Internet Explorer zklamal i z pohledu funkčnosti, jelikož nepodporuje mnoho nových vlastností JavaScriptu, mezi které patří například *template literals*, což by bylo možné volně přeložit jako „šablonové řetězce“ – díky nim je možné přímo do řetězců vkládat proměnné a využívat víceřádkové řetězce. Vzhledem k tomu, že ve Windows 10 byl oficiálně nahrazen prohlížečem Microsoft Edge a dle statistik je využíván pouze 2 % uživatelů, upřednostnil jsem ponechání moderních vlastností JavaScriptu na úkor nefunkčnosti v prohlížeči Internet Explorer.

Cross-browser testování

Po důkladném manuálním testování bylo ještě několikrát provedeno cross-browser testování na několika desítkách webových prohlížečů za účelem odhalit poslední drobné chyby v uživatelském rozhraní. K realizaci byl využit nástroj **Browsershots**[1], který vytváří screenshoty podoby webu v různých prohlížečích a operačních systémech. Jedná se o bezplatný užitečný nástroj, který může odhalit chyby v grafické podobě webu například ve starších verzích prohlížečů.

8.1.2 Testování mobilní aplikace

Již od začátku implementace byla mobilní aplikace testována přímo na reálném mobilním zařízení, testování na emulátorech bylo vynecháno. Tento krok byl zvolen především z důvodu vývoje hybridní mobilní aplikace, kterou je možné díky Cordově pomocí jediného příkazu nainstalovat do připojeného mobilního zařízení a spustit. Vzhledem k tak jednoduché možnosti testování bylo považováno testování na reálném zařízení za více vypovídající.

8.2 Uživatelské testování

Druhou fází testování bylo testování uživateli. Toto testování probíhalo až v okamžiku, kdy byl systém dokončen jako celek, odhalené chyby z první fáze byly odstraněny a uživatelé tak mohli mít pocit práce ve skutečném funkčním systému.

[1] http://browsershots.org/

8.2.1 Testování webové aplikace

Uživatelské testování webové aplikace probíhalo prostřednictvím připraveného testovacího scénáře, ke kterému byl předpřipraven SQL skript se vzorovými daty. Na základě testovacího scénáře uživatelé se systémem plnili úkoly, které by byly běžné při práci dispečera.

Aby bylo možné jednotlivé akce uživatelů během testování důkladně analyzovat, byl k testování využit nástroj **Smartlook**[2], který umožňuje zaznamenávat chování uživatelů pomocí vizuálních dat. Nahrávání uživatele se spustí automaticky s jeho příchodem do webové aplikace a ukončí se s jeho odchodem. Díky tomuto nástroji je možné si přehrát zaznamenanou obrazovku uživatele a analyzovat tak veškeré jeho kroky.

Analýza testování ukázala, že z pohledu funkčnosti systému nebyl žádný problém, avšak z pohledu orientace v uživatelském rozhraní nastaly u některých uživatelů drobné problémy. Na základě toho byly některé položky jinak pojmenovány, proběhlo přeuspořádání elementů na stránce pro vytváření nové trasy a byla doplněna tlačítka pro přesun na předchozí stránku v některých částech systému. Analýza také potvrdila, že například validace formulářů nebyla zbytečná a díky její vizuální formě se uživatelé ihned zorientovali, kde nastal problém, a dokázali jej vyřešit.

8.2.2 Testování mobilní aplikace

Na všech zařízeních bylo testováno nejen úspěšné nainstalování a spuštění aplikace, ale také všechny základní činnosti, které je možné s mobilní aplikací provádět – přihlášení a odhlášení, odesílání GPS dat, zobrazení trasy a práce s aktuální trasou. Testování probíhalo na následujících zařízeních:

- Xiaomi Redmi 3S Prime (Android 6.0.1)
- Samsung Galaxy A3 (Android 7.0.0)
- Samsung Galaxy J5 (Android 7.1.1)
- Lenovo Moto G5S Plus (Android 8.1.0)

Testování na výše zmíněných zařízeních proběhlo úspěšně. Instalace, spuštění i funkcionalita aplikace, vše bylo zcela bezproblémové. Pouze v případě mobilního telefonu Samsung Galaxy J5 nastala situace, kdy výpočet souřadnic trval z neznámého důvodu v řádu minut místo maximálně několika sekund. Následně ale bylo zjištěno, že telefon měl ve výchozím nastavení zvolen úsporný režim zjišťování polohy, který bylo možné změnit na vysokou přenost. Po přepnutí již bylo vše v pořádku a telefon dokázal získat souřadnice téměř okamžitě.

[2] https://www.smartlook.com/cs/

Kapitola 9

Možnosti rozšíření

Vzhledem k tomu, že se jedná o systém, jehož cílem je zefektivnit a zrychlit chod soukromých převozových sanitních služeb, nabízí se široká škála rozšíření, která by se v případě dalšího vývoje systému mohla do systému integrovat. V následujících podkapitolách se pokusím shrnout ta nejzajímavější z nich.

9.1 Notifikace pacientů o plánovaném příjezdu vozu

Občas se stává, že pacienti svůj převoz nahlásí několik dnů dopředu a následně na něj zapomenou, nebo očekávají, že sanitní vůz pro ně přijede později, než je tomu ve skutečnosti. Dochází tak ke zbytečnému prodlení, kdy řidič i ostatní pacienti musí na daného pacienta čekat i několik desítek minut a mohou kvůli tomu přijet pozdě například na plánované vyšetření.

Nabízí se tedy možnost v podobě notifikace pacientů pomocí automatické SMS, a to ideálně ve dvou fázích – pokud je převoz objednaný delší dobu dopředu, pacient by byl informován den předem, že má na zítřejší den objednaný sanitní vůz na ráno/dopoledne/odpoledne/večer. Druhá automatická SMS by byla odeslána v okamžiku, kdy by řidič ve své mobilní aplikaci potvrdil začátek realizace přidělené trasy. Následně od času začátku trasy by se vypočítaly jednotlivé časy mezi úseky a byly by rozeslány SMS všem pacientům na dané trase, v jaký orientační čas pro ně řidič přijede.

9.2 Automatizované plánování tras

Při plánování trasy v současné podobě systému přichází na mysl *problém obchodního cestujícího*, což je optimalizační problém zabývající se nalezením nejkratší možné cesty procházející všemi zadanými body na mapě.

Automatizované plánování trasy by mohlo probíhat tak, že dispečer by pouze zvolil převozy, které by byly součástí jedné trasy. V okamžiku, kdy by k trase přiděloval vůz, by se mohl rozhodnout, zda chce vypočítat nejkratší trasu z aktuální polohy sanitního vozu nebo z pevného bodu, kterým by bylo naložení nějakého pacienta.

Taková funkce by na první pohled mohla dispečerovi ušetřit spoustu práce s plánováním nejlepšího možného pořadí realizace jednotlivých převozů v rámci trasy. Avšak aby byl takový algoritmus skutečně využitelný v oblasti sanitních služeb, musel by se při plánování nejkratší cesty neřídit pouze vzdálenostmi mezi jednotlivými body, ale také například stavem pacienta či nutností pacienta být na nějakém místě v určitý čas. Není tedy přípustné,

aby například pacient, který je po závažné operaci propuštěn do domácí péče strávil v sanitním voze 3 hodiny, zatímco jiný pacient s méně závažnou diagnózou by byl za 30 minut doma.

9.3 Převod textu na řeč v aplikaci pro řidiče

Další možnost navrhovaného rozšíření je zaměřena na bezpečnost řidiče i pacientů v sanitním voze. Úkony, které řidič sanitního vozu s aplikací provádí, jsou v současném systému vytvořeny tak, aby řidič s aplikací musel pracovat pouze v čase, kdy sanitní vůz stojí a řidič nakládá nebo vykládá pacienty. Avšak může nastat situace, kdy si řidič během jízdy potřebuje připomenout přesnou adresu, kam nyní jede. V takovém případě by buď musel zastavit, což může být v některých místech problematické, nebo by adresu v aplikaci hledal během jízdy a tím by mohl ohrozit zdraví pacientů.

Implementace funkce *text-to-speech* neboli převodu textu do řeči by mohla výrazně zvýšit bezpečnost i komfort řidiče. Pomocí několika základních velkých tlačítek, které by pokrývaly nejpotřebnější požadavky řidiče a které by se zobrazovaly na displeji automaticky během pohybu vozidla, by mohl řidič po jednom stisku i během jízdy získat veškeré informace, které potřebuje.

9.4 Notifikace pro řidiče

Poslední zmiňované rozšíření souvisí s předchozím rozšířením týkajícím se převodu textu na řeč. Právě z důvodu zmiňované bezpečnosti by notifikace pro řidiče v současném stavu systému ztrácely význam, protože jediné, čeho by tím bylo docíleno, je ztráta pozornosti řidiče, který by se snažil zjistit, jakou notifikaci obdržel.

Při spojení s převodem textu na řeč by ale notifikace svůj význam získaly. Během jízdy by mohl být řidič informován o nově přidělené trase, změnách v současné trase či jakýchkoli dalších zprávách, které by potřeboval dispečink řidiči sdělit.

Kapitola 10

Závěr

Cílem této práce bylo vytvořit systém pro soukromé převozové sanitní služby složený z webové aplikace určené pro dispečink a z mobilní aplikace určené pro řidiče sanitních vozů. Po nastudování informací ohledně vývoje webových a mobilních aplikací a práce s GPS daty bylo nutné provést analýzu požadavků a tyto požadavky rozvrhnout do konkrétních částí systému. S analýzou požadavků byla provedena také analýza existujících řešení. Obě navržené aplikace byly implementovány a důkladně otestovány na vzorových datech a situacích. Na základě zpětné vazby z testování byly připomínky do finální podoby obou aplikací zapracovány.

Výsledkem této práce je webová aplikace pro dispečink soukromých sanitních služeb umožňující vkládání objednávek do systému, efektivní plánování tras s využitím mapových podkladů, přidělování tras řidičům na základě polohy a parametrů sanitních vozů a základní správu zaměstnanců, zdravotních pojišťoven, zdravotnických zařízení a sanitních vozů, a multiplatformní mobilní aplikace sloužící k přihlašování řidičů do služby, pravidelnému odesílání polohy sanitního vozu, zobrazení tras přidělených dispečinkem a zaznamenávání postupu realizace aktuální trasy.

Vzhledem k tomu, že v současnosti není na trhu žádný produkt poskytující podobné funkce pohromadě, systém má velký potenciál z hlediska dalšího rozvoje. Již během vývoje bylo vymyšleno mnoho dalších možných rozšíření současné podoby systému, která byla v práci také zmíněna. Tato rozšíření by mohla mít vliv nejen na lepší chod sanitních služeb z hlediska organizace a řízení, ale také na pohodlí a spokojenost pacientů, kteří jejich služby využívají.

Literatura

[1] Adobe.com: *Co jsou to webové aplikace a dynamické webové stránky?* Únor 2017, [Online; navštíveno 18.4.2019].
URL https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html

[2] Bennour, A.: *CSS Framework // Which Is Better: Bulma vs Bootstrap vs Foundation.* Květen 2018, [Online; navštíveno 28.12.2018].
URL https://www.techtalko.com/2018/05/29/css-framework-which-is-better-bulma-vs-bootstrap-vs-foundation/

[3] Bergmann: *Co to je GPS? Historie a úvod do problematiky.* Prosinec 2005, [Online; navštíveno 14.1.2019].
URL https://www.svetmobilne.cz/co-to-je-gps-historie-a-uvod-do-problematiky/244

[4] Brown, T. B.; Butters, K.; Panda, S.: *HTML5 okamžitě.* Computer Press, 2014, ISBN 978-80-251-4296-7.

[5] Caliman, D.: *To use or not to use Bootstrap Framework?* Srpen 2015, [Online; navštíveno 28.12.2018].
URL http://blog.creative-tim.com/web-design/use-not-use-bootstrap-framework/

[6] Chroust, M.: *Superpřesné GPS čipy přijdou do mobilů příští rok. Mají mít přesnost do 30 cm.* Září 2017, [Online; navštíveno 14.1.2019].
URL https://navigovat.mobilmania.cz/bleskovky/superpresne-gps-cipy-prijdou-do-mobilu-pristi-rok-maji-mit-presnost-do-30-cm/sc-266-a-1339723

[7] Facebook.github.io: *React Native · A framework for building native apps using React.* [Online; navštíveno 5.4.2019].
URL https://facebook.github.io/react-native/

[8] FancyApps.com: *fancybox - Touch enabled, responsive and fully customizable jQuery lightbox script.* [Online; navštíveno 4.1.2019].
URL http://fancyapps.com/fancybox/3/

[9] Gsa.europa.eu: *World's first dual-frequency GNSS smartphone hits the market.* Červen 2018, [Online; navštíveno 14.1.2019].
URL https://www.gsa.europa.eu/newsroom/news/world-s-first-dual-frequency-gnss-smartphone-hits-market

[10] James, J.: *How To Read GPS Coordinates*. Únor 2016, [Online; navštíveno 8.4.2019].
URL https://www.ubergizmo.com/how-to/read-gps-coordinates/

[11] Javůrek, A.: *Seznamte se, React Native*. Říjen 2018, [Online; navštíveno 5.4.2019].
URL https://www.zdrojak.cz/clanky/seznamte-se-react-native/

[12] Jscrambler: *12 Frameworks for Mobile Hybrid Apps*. Duben 2017, [Online; navštíveno 5.4.2019].
URL https://blog.jscrambler.com/10-frameworks-for-mobile-hybrid-apps/

[13] Karabec, L.: *Mobilní aplikace: Co je to, jak vzniká a kolik stojí*. Říjen 2016, [Online; navštíveno 2.4.2019].
URL https://blog.aira.cz/mobilni-aplikace-co-je-jak-vznika-kolik-stoji

[14] Khosla, S.: *Top 10 New Features in CSS 3*. [Online; navštíveno 4.1.2019].
URL http://webreference.com/authoring/css3/index-2.html

[15] Kumar, P.: *What is jQuery?* Prosinec 2012, [Online; navštíveno 4.1.2019].
URL https://www.journaldev.com/884/what-is-jquery

[16] Laine, J.: *Should You Use a PHP Framework? Five Pros and Cons*. Červenec 2017, [Online; navštíveno 1.4.2019].
URL https://code.tutsplus.com/tutorials/should-you-use-a-php-framework-five-pros-and-cons--cms-28905

[17] LeRoux, B.: *PhoneGap, Cordova, and what's in a name?* Březen 2012, [Online; navštíveno 5.4.2019].
URL https://phonegap.com/blog/2012/03/19/phonegap-cordova-and-whate28099s-in-a-name/

[18] ManagementMania.com: *Značkovací jazyky (Markup languages)*. [Online; navštíveno 28.12.2018].
URL https://managementmania.com/cs/znackovaci-jazyky-markup-languages

[19] McLaughlin, B.: *PHP & MySQL: The Missing Manual*. Sebastopol: O'Reilly Media, 2013, ISBN 978-1-449-32557-2.

[20] Picurelli, L.: *Advantages and disadvantages of web app development*. Květen 2017, [Online; navštíveno 18.4.2019].
URL https://www.linkedin.com/pulse/advantages-disadvantages-web-app-development-luis-picurelli

[21] Reigns, S.: *11 Best PHP Frameworks for Modern Web Developers in 2019*. 2019, [Online; navštíveno 1.4.2019].
URL https://coderseye.com/best-php-frameworks-for-web-developers/

[22] Rouse, M.: *What is database (DB)? – Definition from WhatIs.com*. [Online; navštíveno 8.1.2019].
URL https://searchsqlserver.techtarget.com/definition/database

[23] Sideropulos, I.: *Navigační systémy GPS*. [Online; navštíveno 14.1.2019].
URL http://www.gpsnavigace.cz/prispevky/co_je_gps.htm

[24] Singh, S. V.: *Native App Development vs. Hybrid App Development* . Září 2018, [Online; navštíveno 2.4.2019].
URL https://hackernoon.com/native-app-development-vs-hybrid-app-development-dd83122a738c

[25] Snášel, J.: *Už vím, jak pracuje navigační systém GPS*. Říjen 2005, [Online; navštíveno 15.1.2019].
URL https://www.mobilmania.cz/clanky/uz-vim-jak-pracuje-navigacni-system-gps/jak-gps-urci-polohu/sc-3-a-1111127-ch-1030454/default.aspx

[26] Suehring, S.: *JavaScript: krok za krokem*. Brno: Computer Press, 2008, ISBN 978-80-251-2241-9.

[27] Tezer, O.: *SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems*. Únor 2014, [Online; navštíveno 8.1.2019].
URL https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

[28] Wikipedia.org: *List of GPS satellites*. [Online; navštíveno 15.1.2019].
URL https://en.wikipedia.org/wiki/List_of_GPS_satellites

[29] Wodehouse, C.: *How AJAX (Asynchronous JavaScript + XML) Works*. Květen 2015, [Online; navštíveno 23.4.2019].
URL https://www.upwork.com/hiring/development/how-ajax-works/

[30] Yigal, A.: *Sqlite vs. MySQL vs. PostgreSQL: A Comparison of Relational Databases*. Listopad 2018, [Online; navštíveno 8.1.2019].
URL https://logz.io/blog/relational-database-comparison/

[31] Čábelka, M.: *Úvod do GPS*. Duben 2008, [Online; navštíveno 15.1.2019].
URL https://www.natur.cuni.cz/geografie/geoinformatika-kartografie/ke-stazeni/vyuka/gps/skriptum-uvod-do-gps/

[32] Čápka, D.: *Základní CSS selektory a vlastnosti*. [Online; navštíveno 28.12.2018].
URL https://www.itnetwork.cz/html-css/webove-stranky/jak-psat-moderni-web-html-tutorial-zakladni-css-selektory-atributy

[33] Čápka, D.: *Úvod do CSS (kaskádových stylů)*. [Online; navštíveno 28.12.2018].
URL https://www.itnetwork.cz/html-css/webove-stranky/jak-psat-moderni-web-html-tutorial-uvod-do-css