Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Mobilní aplikace pro včasné varování řidičů při průjezdu vozidel záchranných složek

2021

Jiří Šesták

Vedoucí práce:
Mgr. Jiří Balun

Studijní obor: Aplikovaná informatika,
prezenční forma

Bibliografické údaje

Autor:	Jiří Šesták
Název práce:	Mobilní aplikace pro včasné varování řidičů při průjezdu vozidel záchranných složek
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2021
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Jiří Balun
Počet stran:	34
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Jiří Šesták
Title:	Mobile application for a swift warning of drivers when an ambulance or a fire truck is about to pass by
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2021
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Jiří Balun
Page count:	34
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Závěrečná práce se zabývá včasným varováním řidičů v provozu, jehož cílem je umožnit rychlou reakci řidičů a vytvoření prostoru na silnicích pro vozidla záchranných a bezpečnostních složek na výjezdu. Základem systému je mobilní aplikace klienta (řidiče) a webový server záchranných složek. Aplikace komunikuje s webovým serverem, který reaguje na polohu zařízení řidiče a polohy záchranných vozidel na výjezdu.

Synopsis

The thesis focuses on a swift warning of drivers to speed up their reaction and to help them clear the road faster for emergency vehicles. Thesis stands on a client's (driver's) mobile application and emergency services' web server. Client's application communicates with the web server, which reacts to the device location of the client and locations of any emergency vehicles in action.

Klíčová slova: varování řidičů; záchranné složky; mobilní aplikace; webový server; poloha vozidel

Keywords: warning of drivers; emergency services; mobile application; web server; vehicle's location

Rád bych poděkoval Mgr. Jiřímu Balunovi za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

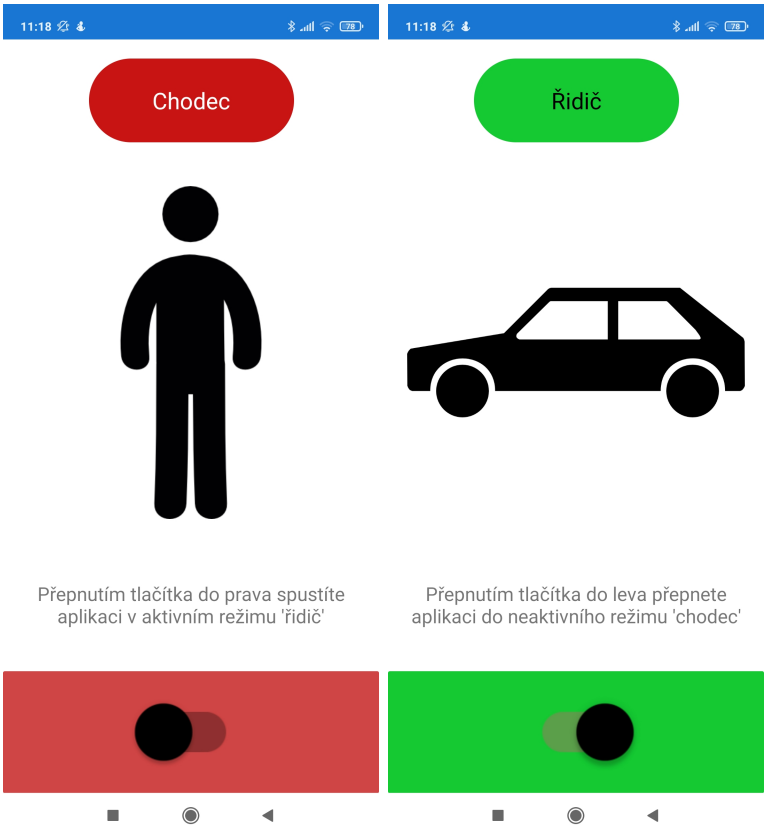
Seznam obrázků

Seznam zdrojových kódů

1 Úvod

Doba příjezdu vozidel záchranných a bezpečnostních složek je často rozhodujícím faktorem při záchraně lidských životů. Na silnicích lze často vidět, jak se doba příjezdu vozidel záchranných složek prodlužuje kvůli pomalým reakcím jednotlivých účastníků silničního provozu. V dnešní době je reakce ostatních účastníků provozu závislá na zvuku sirény záchranného vozidla. Bohužel záleží na množství faktorů jako je hlasitost rádia v automobilu nebo okolní ruch provozu, jestli řidič sirénu uslyší včas a s předstihem vytvoří záchranářskou uličku pro vozidlo na výjezdu. V ideálním světe, kde by vozidlo záchranných složek mělo včas připravený volný průjezd po celé délce trasy k nehodě nebo pacientovi, by se doba příjezdu velmi zkrátila, a tím by se zvýšil počet zachráněných lidských životů. Proto se nabízí otázka, jestli neexistuje jiné efektivnější řešení. Standartním vybavením téměř každého člověka je v moderní době chytrý telefon, který se často používá i při jízdě automobilem například kvůli navigaci. Použití mobilních telefonů pro detekci poloh automobilů a jejich kolizí s trasami a polohami vozidel záchranných složek, je jedno z řešení, které by mohlo při dostatečném počtu uživatelů zkrátit dobu příjezdu vozidel na výjezdu.

V budoucnu by se podobná funkcionalita dala využít v autonomních vozidlech nebo chytrém řízení provozu pomocí semaforů. Komunikací mezi jednotlivými účastníky silničního provozu se zabývá projekt C-Roads, jehož cílem je propojení silničního provozu do komunikující sítě tak, aby se snížil počet dopravních nehodu a zefektivnil přesun vozidel záchranných a bezpečnostních složek. Mým hlavní cílem bylo vymyslet systém, který bude založený na komunikaci mobilní aplikace řidiče (klienta) se serverem záchranných složek s využitím *Google Maps API*, jehož cílem bude notifikovat řidiče o blížících se vozidlech na výjezdu tak, aby vytvořili záchranářskou uličku s dostatečným předstihem a vozidla na výjezdu měla volný průjezd.



Obrázek 1: Výchozí obrazovka v neaktivním/aktivním režimu

2 Uživatelská dokumentace

2.1 Mobilní aplikace

Mobilní aplikace je určená pro uživatele (řidiče). Při spuštění se aplikace nachází v neaktivním režimu (chodec), ve kterém čeká na přepnutí do aktivního režimu (řidič) uživatelem, ve kterém zasílá polohu zařízení webovému serveru a přijímá jeho odezvu. Pro správný běh aplikace je potřeba mít své zařízení připojené k Internetu (zapnutá mobilní data) a mít zapnuté a povolené sdílení polohy zařízení. Pokud není nějaká z těchto podmínek splněna, aplikace se přepne do neaktivního režimu. Aplikace je zaměřena na operační systém Android. Minimální podporovaná verze systému je 5.0 (Lollipop).

Výchozí obrazovka je rozdělená na ukazatel režimu, ikonu režimu, popis a tlačítko přepnutí do/z aktivního/neaktivního režimu.

2.1.1 Notifikace

Mobilní aplikace přijímá 3 typy notifikací:

1. Varování

Obrázek 2: Varování a upozornění

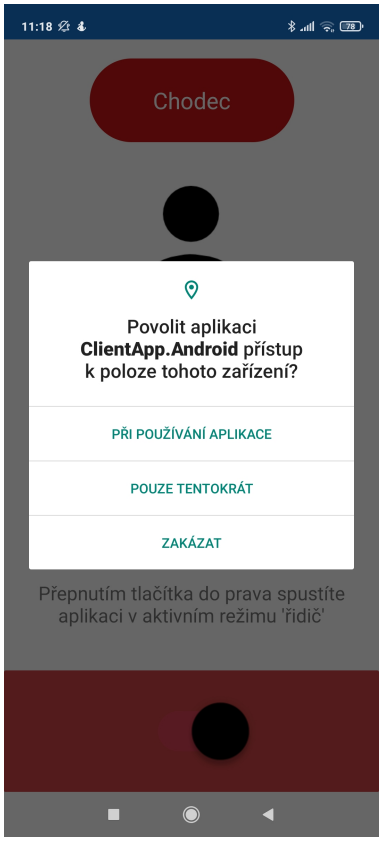
2. Upozornění

3. Chybová hlášení

Varování je notifikace s vyšší prioritou a znamená, že zařízení je méně než 200 metrů od aktivního záchranného vozidla na výjezdu a zároveň je přímo na jeho trase. Reakcí řidiče na příjem varování by mělo být okamžité vytvoření záchranářské uličky podle pravidel silničního provozu. Po přijetí varování jej lze potvrdit tlačítkem „rozumím“. Pokud to uživatel neudělá, je varování uzavřeno automaticky po 30 vteřinách.

Upozornění je notifikace s nižší prioritou a znamená, že zařízení je méně než 200 metrů od aktivního záchranářského vozidla, ale není přímo na jeho trase. Při přijetí upozornění by měl řidič zvýšit svoji pozornost, aby mohl reagovat na změnu aktuální situace, pokud by záchranářské vozidlo změnilo svoji trasu. Stejně jako u varování lze upozornění zavřít pomocí tlačítka „rozumím“, nebo jej zavře aplikace automaticky po 30 vteřinách.

Chybová hlášení jsou notifikace týkající se chodu aplikace, která jsou reakcí na neaktivní přístup k Internetu a vypnuté sdílení polohy zařízení.



Obrázek 3: Chybové hlášení

2.2 Webový server

Webový server je určen k simulaci systému, se kterým pracuje operátor dané záchranné nebo bezpečnostní složky státu. Operátor vidí v reálném čase pohyb aktivních záchranných vozidel na výjezdu, ke které události patří a jestli se vrací z výjezdu s pacientem, nebo jsou teprve na cestě k pacientovi.

Hlavní stránka webového serveru obsahuje:

1. Mapa zobrazující záchranná vozidla na výjezdu
2. Filtry
3. Zadávací formulář pro událost
4. Zadávací formulář pro záchranné vozidlo
5. Tabulka aktivních událostí
6. Tabulka záchranných vozidel v systému

2.2.1 Mapa

Mapa zobrazuje záchranná vozidla a pro potřeby testování také polohy řidičů s mobilní aplikací v aktivním režimu. Podržením klávesy Ctrl a posouváním kolečka myši, lze mapu přiblížit/oddálit. Na mapě se zobrazují ikony a vypočtená trasa záchranného vozidla na výjezdu.

1. Ikona záchranného vozidla 
2. Trasa záchranného vozidla je vykreslena na mapě modrou čarou.
3. Ikona vozidla s mobilní aplikací 
4. Vozidlo po přijetí varování 
5. Vozidlo po přijetí upozornění 

2.2.2 Filtry

Filtry slouží operátorovi k zobrazení pouze těch informací na mapě, které potřebuje. Pro potřeby testování je přidáno tlačítko pro spuštění/zastavení obnovování mapy.

Typy filtrů:

1. Podle *Id* záchranného vozidla – zobrazí pouze vozidlo s daným *Id*.

2. Podle stavu záchranného vozidla – zobrazí pouze aktivní/ neaktivní vozidla.

3. Podle stavu výjezdu záchranného vozidla – zobrazí vozidla, která jedou k pacientovi nebo se vrací do nemocnice.

4. Podle *Id* aktivní události – zobrazí záchranná vozidla připojená k dané události.

2.2.3 Zadávací formulář pro událost

Při nahlášení události operátor zadá událost do systému pomocí zadávacího formuláře.

Operátor zadává:

1. Adresa

2. Priorita

3. Popis události

2.2.4 Zadávací formulář připojení záchranného vozidla

Po vzniku události připojí operátor k dané události záchranné vozidlo.

Operátor zadává:

1. Adresa

2. Popis vozidla (posádka, vybavení...)

3. *Id* události

2.2.5 Mazací formuláře pro událost a záchranné vozidlo

Operátor může vymazat záchranné vozidlo z databáze nebo událost, pokud na ni není navázané žádné záchranné vozidlo.

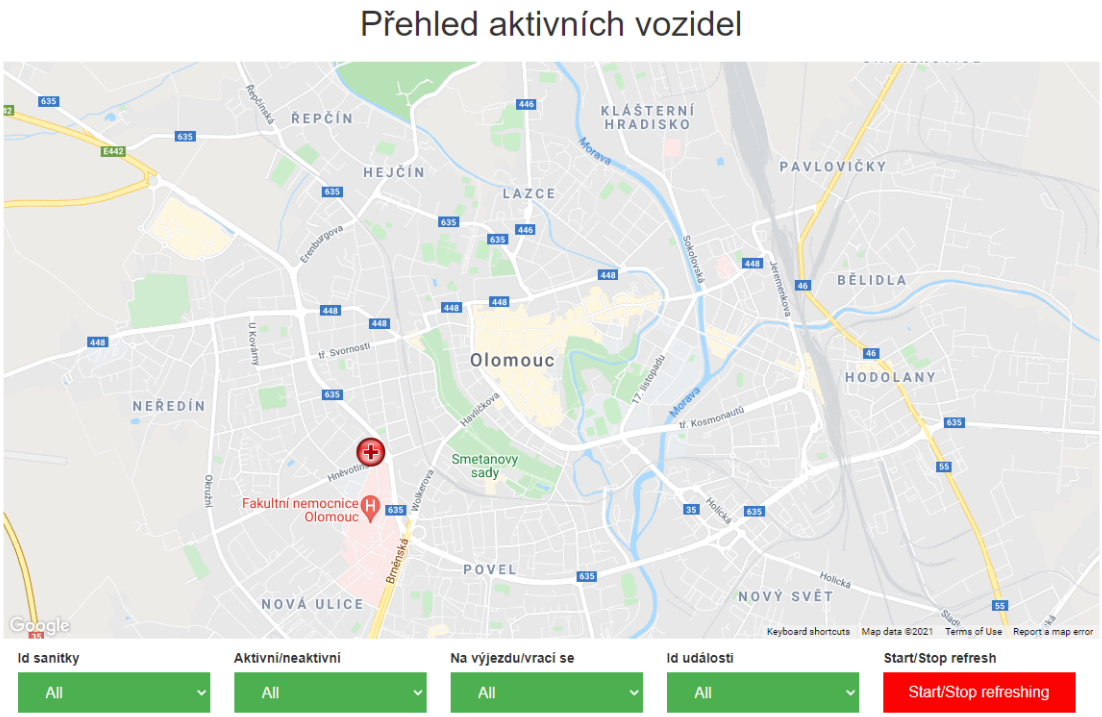
2.2.6 Tabulka aktivních událostí

V tabulce operátor vidí aktivní události a jejich zeměpisné souřadnice, čas vzniku události, prioritu a popis události. Priorita u událostí je v současných systémech záchranných složek definovaná do tří kategorií. Z důvodu nedostatku informací o přesné definici jednotlivých kategorií si je definujme jako:

1. Vážná zranění s nutností akutního zákroku.

2. Zranění s nízkým rizikem smrtelného ohrožení.

3. Událost bez ohrožení na zdraví. (např. převoz z bydliště do nemocnice)



Obrázek 4: Mapa zobrazující záchranná vozidla na výjezdu

2.2.7 Tabulka záchranných vozidel

V tabulce operátor vidí vozidla spojená s aktivní událostí a jejich zeměpisné souřadnice, popis vozidla (posádka, vybavení...) a *Id* události, ke které je vozidlo připojeno.

2.3 Desktopová aplikace

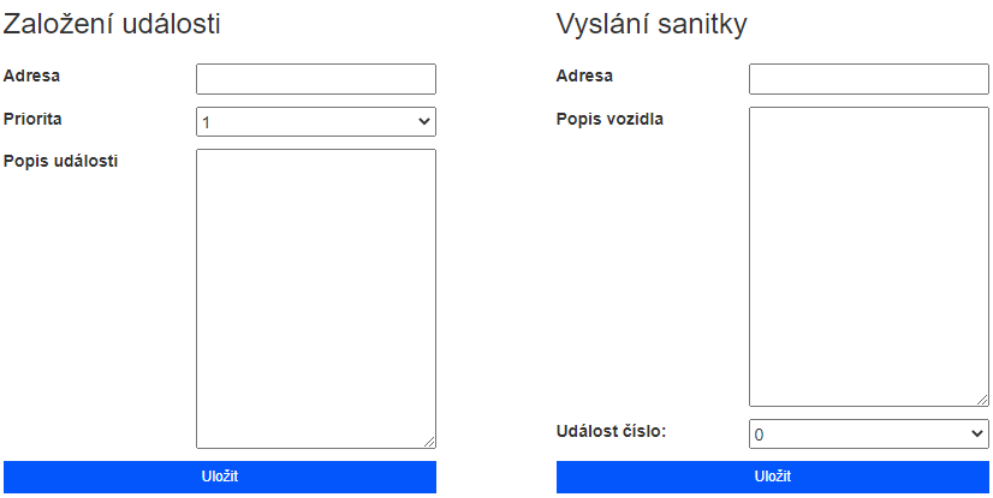
Desktopová aplikace slouží výhradně k simulaci a testování pohybu záchranných vozidel a aktivních uživatelských zařízení. Aplikace je tvořena sloupcem tlačítek a výpisovým prostorem. Pokud dané vlákno klientského vozidla přijme varování nebo upozornění, je do výpisu zapsáno *Id* vozidla a typ notifikace. Tlačítka jsou rozdělena na dva typy:

1. *Run rescue car* pro simulaci záchranného vozidla
2. *Run client car* pro simulaci aktivního zařízení uživatele

Po stisknutí *Run rescue car* nebo *Run client car* se objeví okno pro vybrání textového souboru, který obsahuje posloupnost dat pro simulaci pohybu.

Formát dat v textovém souboru:

1. *Id* (pouze pro simulaci záchranného vozidla)



Obrázek 5: Zadávací formuláře

Obrázek 6: Uzavírací formuláře

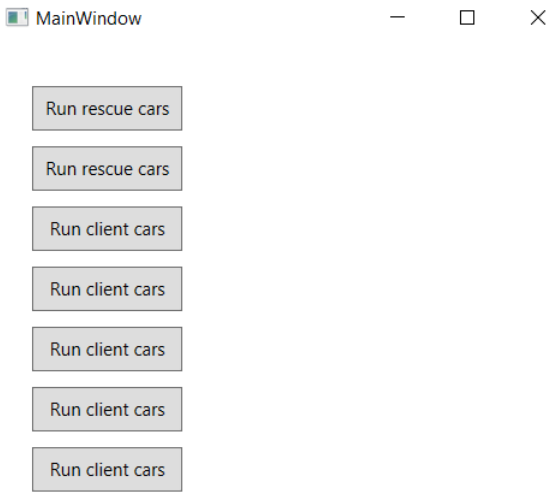
Aktivní události

Id	Zem. šířka cílové lokace	Zem. délka cílové lokace	Čas vytvoření	Priorita	Popis
0	90	90	2021-08-16T00:00:00	1	popis
1	90	80	2021-08-16T00:00:00	2	popis
2	90	100	2021-08-10T23:46:00	1	popis

Aktivní sanitky

Id	Aktuální zem. šířka	Aktuální zem. délka	Zem. šířka cílové lokace	Zem. délka cílové lokace	Aktivní	Popis	Id události
0	49.589126138273926	17.24393981848953	49.58767863465699	17.238197684983405	true	Aktivní sanitka	0
1	49.58787361608891	17.2385917918272	49.58767863465699	17.238197684983405	true	Aktivní sanitka	0
2	49.58789229106458	17.238718917791967	49.58767863465699	17.238197684983405	true	Aktivní sanitka	0

Obrázek 7: Tabulky



Obrázek 8: Desktopová aplikace

2. *Id* události (pouze pro simulaci záchranného vozidla)
3. Cílová zeměpisná šířka, cílová zeměpisná délka (pouze pro simulaci záchranného vozidla)
4. Aktuální zeměpisná šířka, aktuální zeměpisná délka
5. Pokračuje sekvencí bodů zeměpisné šířky a délky pro celou trasu

3 Programátorská dokumentace

3.1 Mobilní aplikace

Mobilní aplikace byla vyvíjena pomocí .NET Framework a Xamarin s využitím jazyka C#. .NET Framework je prostředí potřebné pro běh aplikací nabízející rozhraní a potřebné knihovny.[1] Xamarin je open source platforma pro vytváření aplikací pro iOS, Android a Windows s využitím C#, která rozšiřuje .NET Framework pro práci s mobilními zařízeními.[2] Projektová složka mobilní aplikace se skládá z projektů *ClientApp*, *ClientApp.Android* a *ClientApp.iOS*. Projekt *ClientApp.iOS* je určen pro vývoj aplikací s operačním systémem iOS, na který se práce nezaměřuje.

3.1.1 ClientApp

Projekt *ClientApp* obsahuje definici rozložení aplikace *MainPage.xaml*, která je sdílená pro oba operační systémy a její třídu *MainPage.xaml.cs*. Dále obsahuje třídu *ClientCarsModelClass*, která se využívá pro vytváření objektů a jejich odesílání pomocí HTTP dotazů serveru.

3.1.1.1 MainPage.xaml

Rozložení mobilní aplikace je definováno v XAML a obsahuje definice jednotlivých prvků, ze kterých se rozložení skládá. XAML je značkovací jazyk založený na XML, který byl vyvinut společností Microsoft pro .NET Framework a Windows Presentation Foundation. Základem použitého rozložení je *Grid layout* rozdělený do 5 řádků a 3 sloupců. Prvky rozložení aplikace:

1. *mode_frame*: Rámec obalující popisek, který zobrazuje, v jakém režimu aktuálně aplikace běží.
2. *image_frame*: Rámec obalující obrázek (chodec, automobil).
3. *message_frame*: Rámec obalující popisek, který zobrazuje dodatečné informace k funkčnosti aplikace.
4. *switch_frame*: Rámec obalující přepínač, kterým se přepíná aplikace do/z aktivního/neaktivního režimu.
5. *popupWarning* a *popupCaution*: *Contentview* (třída rozložení), které zobrazují varování nebo upozornění při aktivním běhu aplikace.

3.1.1.2 MainPage.xaml.cs

Po spuštění aplikace se nastaví uživatelské rozhraní na neaktivní režim (chodec), ve kterém čeká na přepnutí uživatelem do aktivního režimu (řidič). Po přepnutí přepínače do aktivní polohy kontroluje metoda *OnToggled* oprávnění přístupu k

poloze zařízení a připojení zařízení k Internetu. Poté spouští metodu *StartSending*.

3.1.1.3 StartSending

Metoda *StartSending* provádí velkou část funkcionality aplikace. Hlavní částí je cyklické odesílání HTTP dotazů webovému serveru. Po prvním spuštění posílá HTTP POST s objektem třídy *ClientCarViewModel*, ve které jsou uloženy souřadnice zařízení. Po odezvě z webového serveru si ukládá přidělené *Id* a případně reaguje na odpověď serveru, pokud je v odpovědi varování nebo upozornění. Jakmile má metoda přidělené *Id* serverem, tak posílá své souřadnice serveru v HTTP PUT s daným *Id*. Aplikace si nastavuje interval odesílání souřadnic serveru podle vzdálenosti k nejbližšímu záchrannému vozidlu. V reakci na upozornění nebo varování jej zobrazuje v uživatelském rozhraní aplikace. Pokud aplikace pošle více než 10 stejných souřadnic (zařízení se nehýbe), nebo je aplikace přepnuta do neaktivního režimu, je odeslán HTTP DELETE serveru.

Délka intervalu odesílání souřadnic podle vzdálenosti od nejbližšího aktivního záchranářského vozidla:

1. Vzdálenost je větší než 5 km – 30 vteřin
2. Vzdálenost je menší než 5 km a větší než 3 km – 15 vteřin
3. Vzdálenost je menší než 3 km a větší než 1 km – 5 vteřin
4. Jinak - 2 vteřiny

3.2 Databáze

Databáze *CarLocationsDB* je vytvořena v Microsoft SQL Server Management Studio. Obsahuje tabulky *RescueCarsTable* pro záchranná vozidla, *CarsTable* pro zařízení uživatelů a *EventsTable* pro události.

3.2.1 RescueCarsTable

Obsahuje:

1. *Id* – Primární klíč
2. *Latitude* – Aktuální zeměpisná šířka
3. *Longitude* – Aktuální zeměpisná délka
4. *FinishLatitude* – Cílová zeměpisná šířka
5. *FinishLongitude* – Cílová zeměpisná délka
6. *InAction* – Aktivní záchranné vozidlo (true/false)

7. *Description* – Popis
8. *IdEvent* – *Id* události na kterou je vozidlo navázané

3.2.2 CarsTable

Obsahuje:

1. *Id* – Primární klíč
2. *Latitude* – Aktuální zeměpisná šířka
3. *Longitude* – Aktuální zeměpisná délka
4. *OnRoute* – Zařízení se nachází na trase nějakého záchranného vozidla (true/false)
5. *IdRescueCarByRoute* – *Id* záchranného vozidla, na jehož trase se zařízení nachází
6. *DistanceFromNearestRescueCar* – Vzdálenost od nejbližšího aktivního záchranného vozidla
7. *IdNearestRescueCarByDistance* – *Id* nejbližšího aktivního záchranného vozidla
8. *UpdateTime* – Čas poslední aktualizace souřadnic zaslané serveru

3.2.3 EventsTable

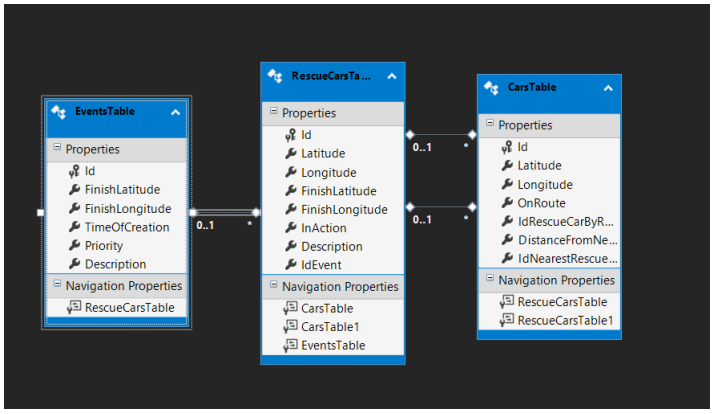
Obsahuje:

1. *Id* – Primární klíč
2. *FinishLatitude* – Cílová zeměpisná šířka
3. *FinishLongitude* – Cílová zeměpisná délka
4. *TimeOfCreation* – Čas vytvoření události
5. *Priority* – Priorita
6. *Description* – Popis

3.3 Webový server

Webový server je implementován pomocí ASP.NET Web API 2, což je framework umožňující implementaci webových služeb založených na HTTP dotazech, které jsou přístupné z různých platform a aplikací. Server je spojený s databází*CarLocationsDB* pomocí webové služby *RescueCarsDataAccess*, která umožňuje webovému serveru komunikaci s databází.

Hlavními částmi webového serveru jsou Controllers, Models a Views.



Obrázek 9: ADO.NET entity data model

3.3.1 Models

Models obsahují ViewModels třídy pro každou tabulku databáze a používají se v kontrolerech pro přístup k databázi.

1. *RescueCarsViewModel* – pro záchranná vozidla
2. *ClientCarViewModel* – pro uživatelská zařízení
3. *EventsViewModel* – pro události

```
1  using System;
2
3  namespace RescueCarsServer.Models
4  {
5     //třída pro vytváření objektů záchranných vozidel
6     public class RescueCarsViewModel
7     {
8        public int Id { get; set; }
9        public double Latitude { get; set; }
10       public double Longitude { get; set; }
11       public double? FinishLatitude { get; set; }
12       public double? FinishLongitude { get; set; }
13       public bool InAction { get; set; }
14       public String Description { get; set; }
15
16       public int? IdEvent { get; set; }
17    }
18 }
```

Zdrojový kód 1: C#

3.3.2 Controllers

Kontrolery definují jakým způsobem bude server odpovídat na HTTP dotazy, které mu přijdou z uživatelských aplikací. V projektu *RescueCarsServer* existuje 5 kontrolerů:

1. *HomeController*
2. *RescueCarsController*
3. *ClientCarsController*
4. *EventsController*
5. *StandartController*

HomeController implementuje GET metody pro každou z tabulek databáze. Vrací pro danou tabulku databáze výpis všech záznamů tabulky.

RescueCarsController implementuje HTTP metody pro tabulku *RescueCarsTable*.

- Metoda *GetAll()* je HTTP GET metoda, která vrací všechny záznamy z tabulky *RescueCarsTable*.
- Metoda *GetAllRescueCars()* je HTTP GET metoda, která vrací všechny záznamy z tabulky *RescueCarsTable*, které odpovídají zadaným filtrům na hlavní stránce webového serveru.
- Metoda *PutRescueCar(RescueCarsViewModel car)* je HTTP PUT metoda, která aktualizuje data v databázi pro instanci *RescueCarsViewModel* daného *Id*.
- Metoda *PostRescueCar(RescueCarsViewModel car)* je HTTP POST metoda, která ukládá instanci *RescueCarsViewModel* a vrací vygenerované *Id*, pokud se uložení zdařilo.
- Metoda *DeleteRescueCar (int id)* je HTTP DELETE metoda, která z databáze maže záznam s daným *Id*.

ClientCarsController implementuje HTTP metody pro tabulku *CarsTable*.

- Metoda *GetOldCars()* je HTTP GET metoda, která vrací všechny záznamy z tabulky *CarsTable*, které mají v atributu *UpdateTime* čas starší než 30 minut.
- Metoda *PutCar(ClientCarViewModel car)* je HTTP PUT metoda, která aktualizuje data v databázi pro instanci *ClientCarViewModel* daného *Id*. Pokud je vzdálenost mezi zařízením a aktivním záchranným vozidlem menší než 200 metrů a zařízení je na trase tohoto záchranného vozidla, tak metoda vrací varování, pokud zařízení není na trase daného vozidla, vrací upozornění.

- Metoda *PostClientCar(ClientCarViewModel car)* je HTTP POST metoda, která ukládá objekt *ClientCarViewModel* a vrací vygenerované Id, pokud se uložení zdařilo.
- Metoda *DeleteClientCar (int id)* je HTTP DELETE metoda, která z databáze maže záznam s daným *Id*.

EventsController implementuje HTTP metody pro tabulku *EventsTable*.

- Metoda *GetAll()* je HTTP GET metoda, která vrací všechny záznamy z tabulky *EventsTable*.
- Metoda *PostEvent(EventsViewModel newEvent)* je HTTP POST metoda, která ukládá objekt *EventsViewModel*.
- Metoda *DeleteEvent (int id)* je HTTP DELETE metoda, která z databáze maže záznam s daným *Id*.

StandartController je pomocný kontroler, který implementuje HTTP GET metodu *GetAllCars()*, která vrací všechny záznamy z tabulky *CarsTable*.

3.3.3 Views

Views obsahují soubory formátu cshtml, které utváří design a logiku hlavní stránky webového serveru.

- *_Layout.cshtml* – obsahuje html definici stránky
- *Index.cshtml* – obsahuje logiku zapsanou v Javascriptu

3.3.4 Fungování webového serveru

Soubor *Index.cshtml* obsahuje skripty, které se vykonávají po spuštění webového serveru.

- Skripty pro načtení dat z databáze a naplnění tabulek pro události a Záchranářská vozidla.
- Skripty pro naplnění filtrů tak, aby se dala filtrovat aktuální data na mapě.
- Skripty pro zadávací formuláře události záchranného vozidla.
- Hlavní skript webového serveru obsluhující mapu a vnitřní logiku serveru.

Hlavní skript s funkcí *initMap()*

Funkce *initMap()* zajišťuje načítání dat z databáze pomocí kontrolerů, výpočet tras záchranných vozidel, porovnávání poloh záchranných vozidel a uživatelských zařízení a jejich vykreslování do mapy. K zobrazení mapy, vykreslení ikonek, tras a výpočtu jejich umístění využívá logika webového serveru funkcionalit *Google Maps API*.

Proměnné

- *rescueCarsMarkers* – pole ikonek záchranných vozidel k vykreslení
- *rescueCarsDescriptions* – pole definic popisů záchranných vozidel
- *rescueCarsInfos* – pole hodnot popisů záchranných vozidel
- *carsMarkers* – pole ikonek uživatelských zařízení
- *messageArray* – pole pro uchovávání zaslaných zpráv zařízením
- *arrayOfRescueCars* – pomocné pole pro uchování záchranných vozidel

Průběh *initMap()*

Funkce *initMap()* inicializuje mapu webového serveru. Funkce dále volá funkci *setRescueCars()* a po jejím dokončení funkci *setCars*. Jelikož obě metody používají asynchronní HTTP dotazy, je potřeba je volat pomocí *Promises*.

Funkce *setRescueCars()* si uloží hodnoty zadané ve filtrech mapy a pošle HTTP dotaz kontroleru *RescueCars*. Pokud je request úspěšný a vrácena data nejsou prázdná, začíná v cyklu tato data procházet. Každé záchranné vozidlo vkládá do pomocného pole *arrayOfRescueCars*, které bude později použito ve funkci *setCars()*. Poté kontroluje jestli má daný vůz nastavenou cílovou polohu a jestli je přidělen k aktivní události. Pokud se vůz dostane do vzdálenosti menší než 10 metrů od cílové destinace, posílá HTTP POST webovému serveru s nastavenou cílovou lokací na výchozí nemocnici, protože dorazil k místu dané události a bude se vracet zpět. Následně vykreslí do mapy svoji ikonku na základě aktuální lokace a aktuální trasu do cílové lokace.

Funkce *setCars()* se spouští po dokončení funkce *setRescueCars()*. Pošle HTTP GET request kontroleru *ClientCars*, který mu vrátí všechna aktivní uživatelská zařízení v provozu. Pokud není *arrayOfRescueCars* (které se plní ve funkci *setRescueCars()*) prázdné a zároveň data vrácená requestem také nejsou prázdná, pokračuje funkce cyklem, který prochází *arrayOfRescueCars*. Pokud ale data vrácená requestem obsahují aktivní zařízení a zároveň nejsou žádné aktivní záchranné vozy, je přeskočeno veškeré porovnávání se záchranářskými vozy a jsou pouze vykresleny ikonky zařízení.

Pro každé záchranářské vozidlo si funkce vypočítá jeho aktuální trasu a ve vnořeném cyklu prochází data (pole aktivních uživatelských zařízení), kde jako první nastaví zařízení *DistanceFromNearestRescueCar* a *IdNearestRescueCarByDistance*, což jsou vzdálenost od nejbližšího aktivního záchranářského auta a jeho Id. Poté funkce pokračuje jednou ze čtyřech možností:

1. Zařízení je na trase záchranného vozu a zároveň už má nastaveno, že je na trase jiného aktivního záchranného vozu. Vypočítá se tedy, daný záchranný vůz blíže než ten, na jehož trase už zařízení je. Potom se uloží *Id* bližšího záchranného vozu do *IdRescueCarByRoute* zařízení a pošle se HTTP POST webovému serveru s aktualizovanými daty zařízení. Na základě vrácených dat z webového serveru se potom zařízení nastaví ikonka pro varování, upozornění, nebo se nechá původní ikonka značící pouze aktivní zařízení.

2. Zařízení je na trase záchranného vozu, ale není na trase žádného jiného aktivního záchranného vozu. Pošle se tedy HTTP POST webovému serveru s aktualizovanými daty zařízení, kde se přidá aktuální záchranný vůz do *IdRescueCarByRoute*. Na základě vrácených dat z webového serveru se potom zařízení nastaví ikonka pro varování, upozornění, nebo se nechá původní ikonka značící pouze aktivní zařízení.

3. Zařízení není na trase záchranného vozu a zároveň už má nastaveno, že je na trase jiného záchranného vozu. Pošle se tedy HTTP POST webovému serveru s aktualizovanými daty.

4. Zařízení není na trase záchranného vozu a není na trase žádného jiného aktivního záchranného vozu. Pošle se tedy HTTP POST webovému serveru s aktualizovanými daty.

Funkce *setCars()* tedy ve vnějším cyklu prochází aktivní záchranná vozidla a ve vnitřním cyklu aktivní uživatelská zařízení, kterým nastavuje nejbližší aktivní záchranné vozidlo a popřípadě vozidlo, na jehož trase se zařízení nachází. Z důvodu omezení volání *Google Maps API* se prochází záchranná vozidla, kvůli kterým se volá funkce *Google Maps API* na vypočtení a vykreslení trasy. Tím se minimalizuje počet dotazů *Google Maps API*.

Funkce *refreshRescueCars()* volá použitím *Promises setRescueCars* a *setCars* a je cyklicky volána po stisknutí tlačítka *Start refresh* a její volání je ukončeno stiskem tlačítka *Stop refresh*.

Vykreslování ikonek uživatelských zařízení, stejně jako tlačítka pro spuštění a zastavení obnovování mapy jsou součástí webového serveru čistě z testovacích důvodů a lepší vizualizace.

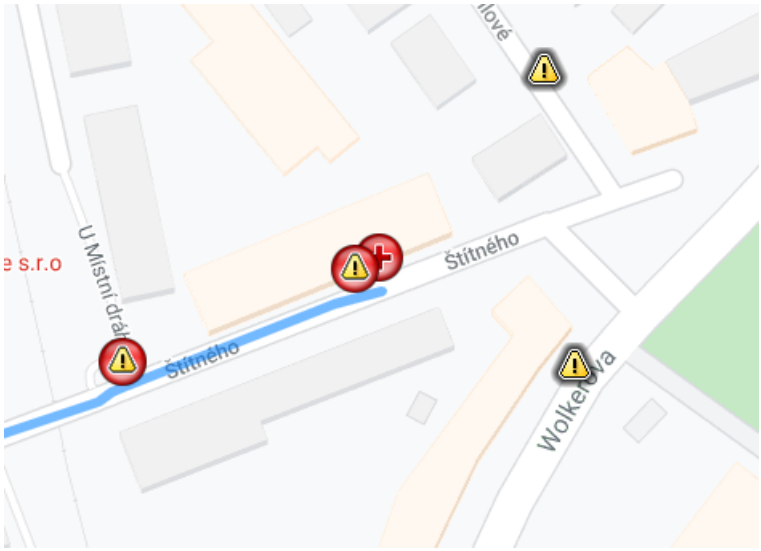
3.4 Desktopová aplikace

Desktopová aplikace byla implementována ve *Windows Presentation Foundation*[3] použitím .NET. Slouží k simulaci více uživatelských zařízení a záchranných vozů zároveň. V aplikaci jsou implementovány třídy *ClientCarsModelClass*, *EventsModelClass* a *RescueCarsModelClass*, které jsou použity pro zasílání objektů záchranných vozidel a uživatelských zařízení webovému serveru.

3.4.1 Metody

Metoda *RescueCarsSimulation()* předpokládá existenci záchranářského vozu v databázi a pouze simuluje pohyb aktivního záchranářského vozidla na výjezdu. Po kliknutí na *Run rescue car* a nahrání souboru se zpracuje vstupní textový soubor. Uloží si *Id* záchranného vozu, v události ke které je připojený a sekvenci souřadnic tvořící trasu do dvou polí. Poté cyklicky zasílá HTTP POST dotazy webovému serveru tak, jak prochází pole uložených souřadnic, čímž simuluje pohyb záchranného vozu.

Metoda *ClientCarSimulation()* je spuštěna po kliknutí na tlačítko *Run client car* a funguje obdobně jak metoda *RescueCarsSimulation()* s tím rozdílem, že nepředpokládá existenci uživatelského zařízení v databázi. Při prvním spuštění posílá webovému serveru HTTP POST request se svými souřadnicemi. Server odpoví při úspěšném odeslání zasláním Id pro dané uživatelské zařízení, které si dané *Id* uloží a dále ho posílá při simulaci pohybu v HTTP PUT requestu. Po celou dobu simulace vlákno kontroluje odezvy webového serveru a pokud dostane varování nebo upozornění, vypíše danou notifikaci do výpisu se svým *Id* a uspí se na 10 vteřin. Po průchodu celého pole souřadnic, vlákno pošle webovému serveru HTTP DELETE dotaz, čímž simuluje ukončení aplikace a je serverem vymazáno z databáze jako neaktivní zařízení.



Obrázek 10: Záchranné vozidlo při průjezdu s varováním a upozorněním

4 Testování

Aktivní uživatelské zařízení přijímá dva typy notifikací, což jsou varování a upozornění. Server se rozhoduje jakou notifikaci poslat na základě hodnot uložených v atributech uživatelského zařízení, které si načítá z databáze.

Jsou to:

- *OnRoute* – nachází/nenachází se na trase nějakého záchranného vozidla
- *IdRescueCarByRoute* – pokud se nachází na trase vozidla, jaké je jeho *Id*
- *DistanceFromNearestRescueCar* – vzdálenost od nejbližšího vozidla
- *IdNearestRescueCarByDistance* – *Id* nejbližšího vozidla

4.1 Varování

Server zasílá varování aktivnímu uživatelskému zařízení, pokud se *IdRescueCarByRoute* a *IdNearestRescueCarByDistance* zařízení rovnají. To znamená, že je zařízení na trase záchranného vozidla na výjezdu, které je zároveň nejbližším vozidlem na výjezdu a vzdálenost mezi uživatelským zařízením a záchranným vozidlem je kratší než 200 metrů. Uživatel (řidič) tedy dostává varování s dostatečným předstihem a vytvoří záchranářskou uličku včas.

Při provozu můžou nastat dva typy situací:

1. Uživatelské zařízení je v dosahu a na trase pouze jednoho vozidla na výjezdu – Server uživatelskému zařízení pošle varování.

Obrázek 11: Záchranné vozidlo při průjezdu s upozorněním

2. Uživatelské zařízení je v dosahu a na trase více než jednoho vozidla na výjezdu – Server uživatelskému zařízení pošle varování a uloží bližší vozidlo a vzdálenost od něj do *DistanceFromNearestRescueCar* a IdNearestRescueCarByDistance.

Při druhé variantě situace se může stát, že v jednom okamžiku bude řidič v dosahu a na trasách dvou (nebo více) záchranných vozů. Dostane varování a bližší vozidlo projede a vzdálí se. V této situaci dostane řidič druhé varování od druhého vozidla na výjezdu, pokud bude stále v jeho dosahu. Nemůže se tedy stát, že by řidič vytvořil prostor pro bližší vozidlo na výjezdu a následně způsobil dopravní nehodu kvůli druhému vozidlu na výjezdu, které kolem něj ještě neprojelo.

4.2 Upozornění

Server zasílá upozornění aktivnímu uživatelskému zařízení, pokud je vzdálenost mezi uživatelským zařízením a nejbližším záchranným vozidlem kratší než 200 metrů. Upozornění nepožaduje po řidiči vytvoření záchranářské uličky, protože se uživatel nenachází přímo na trase vozidla na výjezdu, ale nabádá ho ke zvýšení pozornosti, pokud by došlo ke změně trasy záchranného vozidla. V tomto případě je řidič s dostatečném předstihu informován o pohybu vozidla na výjezdu v jeho blízkém okolí a je tím pádem schopný pohotově reagovat a případně vytvořit prostor pro průjezd záchranářského vozidla.

4.3 Rozhodování

Poslední situací, kterou může nastat, je aktivita dvou nebo více vozidel na výjezdu ve vzdálenosti kratší než 200 metrů, a zároveň kolize uživatelského zařízení s trasou pouze jednoho z nich. V tomto případě dostává uživatelské zařízení notifikaci bližšího vozidla na výjezdu, protože se předpokládá, že to projede dříve. Uživatel tedy dostane upozornění, pokud je blíže vozidlo na výjezdu, na jehož trase uživatelské zařízení není. Varování dostane, pokud je blíže vozidlo na výjezdu, na jehož trase se uživatel nachází.

5 Google Maps API

V závěrečné práci je využita neplacená licence Google Maps API určená pro nekomerční využití a vývoj, což se pojí s omezeními na výkon a množství dotazů.

5.1 Maps Javascript API

Javascript API umožnuje vykreslovat mapy na webových stránkách, serverech a mobilních zařízeních. Webový server využívá Javascript API ke zobrazení mapy na hlavní stránce serveru a vykreslování ikonek (markerů) na mapě.[4]

5.2 The Directions API

The Direction API je webová služba *Google Maps API*, která je založena na HTTP dotazování a vrací vygenerované trasy ve formátu JSON nebo XML. Webový server využívá Direction API při generování a vykreslení tras jednotlivých záchranných vozů do mapy.[5]

- *DirectionsService* – vrací vygenerovanou trasu
- *DirectionsRenderer* – vykresluje vygenerovanou trasu do mapy

5.3 Geocoding API

Geocoding API umožňuje vyhledávat lokace podle zadané adresy a pracovat s danými souřadnicemi. Webový server využívá Geocoding API pro získání souřadnic ze zadané adresy operátorem při zadávání nové události.[6]

5.4 Problémy využití Google Maps API

Využití bezplatné omezené licence Google Maps Api pro vývoj a nekomerční užití, se ukázalo jako problémové při větším počtu záchranných vozů a uživatelských zařízení k vykreslení do mapy. Licence pro vývoj je omezená počtem dotazů poslaných Google Maps API, což může vést k chybovosti vykreslování na mapě. Druhým problémem je obsáhlé použití asynchronních volání, která nelze ošetřit jinak, než čekáním na dokončení všech volání na úkor rychlosti reakcí serveru. Jelikož je vykreslování poloh uživatelských zařízení účelově pouze pro vizualizaci a testovaní, tak je dána přednost plynulosti práce webového serveru a rychlosti odezvy na varování a upozornění.

```
var directionsService = new google.maps.DirectionsService();
var directionsRenderer = [];

//počátek a cíl trasy
var request = {
   origin: start,
   destination: end,
   travelMode: google.maps.TravelMode.DRIVING
}
directionsService.route(request, function (result, status) {
   if (status == google.maps.DirectionsStatus.OK) {
      var dirRend = new google.maps.DirectionsRenderer({
         map: map,
         directions: result,
         suppressMarkers: true,
         preserveViewport: true
      });
      dirRend.setMap(map); //vykreslení trasy
      dirRend.setDirections(result);
      directionsRenderer.push(dirRend); //uložení trasy do pole tras
   }
   else {
      reject(status);
   }
});
```

Zdrojový kód 2: JS

Závěr

Funkční řešení problému včasného varování řidičů při průjezdu vozidel záchranných a bezpečnostních složek, které by bylo v provozu, zatím neexistuje.Cílem této práce proto bylo zaměření se na možná řešení tohoto problému. Jako logické se jevilo řešení pomocí mobilní aplikace jakožto klientského zařízení a zjednodušeného webového serveru, imitujícího server záchranných složek. Bylo očividné, že toto řešení nebude možné použít ve velkém měřítku na skutečný provoz. Z praktického hlediska by záležel dopad tohoto systému na průjezd vozidel záchranných složek dopravou na počtu aktivních uživatelů aplikace a jejich ochotě před každou jízdou aplikaci zapnout. Z právního hlediska by bylo obtížné napojení aplikace na servery záchranných a bezpečnostních složek. Proto jsem se při implementaci tohoto řešení zaměřil hlavně na základní koncept řešení problému s včasným varováním řidičů, který by mohl sloužit jako základní myšlenka pro budoucí použití. Samotná logika, kterou jsou porovnávány polohy uživatelů a vozidel na výjezdu, spolu s logikou posílání notifikací, by se v budoucí době dalo využít například pro nasazení v autonomních vozidlech za použití 5G sítě nebo chytrých křižovatkách.

Conclusions

The working solution for a swift warning of drivers when an emergency car is about to pass by does not exist at this time. The goal of this thesis was to focus on possible solutions of this problem. As a logical approach seemed a solution in a form of mobile application as a client device and a simplified web server imitating emergency services' server. It was obvious, that this solution could not be used on real traffic at such a large scale. From a practical standpoint, the efficiency of the system to assist the passage of emergency vehicles through traffic would depend on the number of active users and their willingness to turn on the application before driving. From a legal standpoint, it would be difficult to connect the application to the emergency services' servers. That is why I have focused my efforts during implementation of the solution on the basic concept of the swift warning of drivers, which could serve as a basis for future use. The logic itself by which the user locations and locations of emergency vehicles are compared, as well as notification sending logic, could be used in the future for usage in autonomous vehicles using 5G network or at smart intersections.

A Obsah přiloženého CD/DVD

`DesktopApp/`
: Složka se spustitelným souborem testovací desktopové aplikace DesktopApp.

`Source/`
: Kompletní zdrojové soubory mobilní aplikace ClientApp a projektové složky RescueCarsServer obsahující zdrojové soubory RescueCarsServer, RescueCarsDataAccess a DesktopApp.

`ClientApp.apk`
: Instalační soubor mobilní aplikace ClientApp.

`Testovací scénáře`
: Složka obsahující předvytvořené trasy pro testování.

`readme.txt`
: Instrukce ke spuštění všech programů a další informace.

`Bakalářská práce.zip`
: Text práce v zip archivu.

Literatura

[1] *Desktopová příručka pro .NET, .NET Core a .NET Framework. : Dokumentace k Windows Presentation Foundation*. 2021. Dostupný z: ⟨https://docs.microsoft.com/cs-cz/dotnet/desktop/wpf/?view=netdesktop-5.0⟩.

[2] *Dokumentace pro Xamarin. : Dokumentace ke Xamarin.Forms*. 2021. Dostupný z: ⟨https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/⟩.

[3] *Desktopová příručka pro .NET, .NET Core a .NET Framework. : Dokumentace k Windows Presentation Foundation*. 2021. Dostupný z: ⟨https://docs.microsoft.com/cs-cz/dotnet/desktop/wpf/?view=netdesktop-5.0⟩.

[4] *Google Maps Platform Documentation. : Maps Javascript API*. 2021. Dostupný z: ⟨https://developers.google.com/maps/documentation/javascript/overview⟩.

[5] *Google Maps Platform Documentation. : The Directions API*. 2021. Dostupný z: ⟨https://developers.google.com/maps/documentation/directions/overview⟩.

[6] *Google Maps Platform Documentation. : Geolocation API*. 2021. Dostupný z: ⟨https://developers.google.com/maps/documentation/geolocation/overview⟩.