Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Aplikace pro moderátory Radia Haná konsolidující dopravní informace

2024

Bc. Matěj Cukr

Vedoucí práce:
RNDr. Martin Trnečka, Ph.D.

Studijní program: Aplikovaná informatika,
Specializace: Vývoj software

Bibliografické údaje

Autor:	Bc. Matěj Cukr
Název práce:	Aplikace pro moderátory Radia Haná konsolidující dopravní informace
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2024
Studijní program:	Aplikovaná informatika, Specializace: Vývoj software
Vedoucí práce:	RNDr. Martin Trnečka, Ph.D.
Počet stran:	31
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Bc. Matěj Cukr
Title:	Application for presenters of Radio Haná consolidating traffic information
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2024
Study program:	Applied Computer Science, Specialization: Software Development
Supervisor:	RNDr. Martin Trnečka, Ph.D.
Page count:	31
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Práce popisuje celý průběh vývoje aplikace. Nejprve je uvedeno, jak současné zjišťování dopravních informací probíhá a co by měla výsledná aplikace obsahovat. Následuje popis návrhu a vývoje backend a frontend části aplikace spolu se zpětnou vazbou od moderátorů. V poslední kapitole je popsána finální verze aplikace z pohledu uživatele.

Synopsis

The thesis describes the whole process of application development. At first, it is presented how the current traffic information is done and what the final application should contain. This is followed by a description of the design and development of the backend and frontend parts of the application along with feedback from presenters. The last chapter describes the final version of the application from the user's perspective.

Klíčová slova: doprava; dopravní informace; WPF

Keywords: traffic; traffic information; WPF

Děkuji vedoucímu své diplomové práce panu RNDr. Martinu Trnečkovi, Ph.D. za jeho vedení, podporu a odborné rady. Děkuji také kolegyni Bc. Leoně Vaculkové za ochotnou spolupráci při definování požadavků na aplikaci a její zpětnou vazbu.

Odevzdáním tohoto textu jeho autor/ka místopřísežně prohlašuje, že celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

1 Úvod **7**
1.1 Způsob zjišťování dopravních informací . . . 7
1.2 Ověření realizace aplikace . . . 7
1.3 Bližší specifikace cílové aplikace . . . 8

2 Architektura aplikace **9**
2.1 Backend . . . 9
2.1.1 Waze . . . 9
2.1.2 Policie . . . 10
2.1.3 Konsolidace . . . 11
2.1.4 Radio Haná . . . 13
2.1.5 Další pomocné třídy . . . 13
2.2 Frontend . . . 13
2.2.1 Návrh aplikace v programu Figma . . . 14
2.2.2 Vytvoření WPF aplikace . . . 18
2.3 Zpětná vazba . . . 19
2.3.1 Zpětná vazba na návrh ve Figma . . . 19
2.3.2 Zpětná vazba na aplikaci . . . 20

3 Uživatelská příručka **23**
3.1 Spuštění aplikace . . . 23
3.2 Úvodní stránka . . . 23
3.3 Hlavní stránka . . . 24
3.4 Okno s mapou události . . . 26

Závěr **28**

Conclusions **29**

A Obsah elektronických dat **30**

Literatura **31**

Seznam obrázků

1 Návrh úvodní stránky ve Figma 14
2 Návrh hlavní stránky ve Figma 15
3 Návrh detailu události ve Figma 16
4 Návrh okna s mapou události ve Figma 16
5 Návrh filtru událostí na hlavní stránce ve Figma 17
6 Návrh stránky s označenými událostmi ve Figma 18
7 Úvodní stránka aplikace 24
8 Hlavní stránka při zobrazení všech událostí 25
9 Hlavní stránka při zobrazení pouze označených událostí 26
10 Okno s mapou zobrazující přesné místo události 27

Seznam zdrojových kódů

1 Metoda pro určení dodatečných informací pro událost typu nehoda 11

1 Úvod

K tématu této diplomové práce mě tak trochu přivedla moje kolegyně, která je moderátorkou v Radiu Haná. Bavili jsme se o tom, jak to v rádiu funguje, a protože jsem aktivní řidič, nejvíce mě zajímalo, jak probíhá příprava na vysílání dopravy. Proces byl prostý. Moderátor navštíví několik webových stránek obsahujících dopravní informace, z nich vybere ty důvěryhodné a užitečné, a odvysílá je. Aktuální stav na silnicích jim také hlásí jejich posluchači. A tak jsem dostal nápad. Vytvořit aplikaci, která by data o dopravě získala z webových stránek automaticky.

1.1 Způsob zjišťování dopravních informací

Zjistil jsem si od kolegyně bližší podrobnosti k získávání dat o dopravě. Hlavní dvě webové stránky, odkud moderátoři získávají většinu informací o aktuální dopravě, jsou Waze[1] a Policie[2]. Bližší informace k uzavírkám si pak zpravidla dohledávají na webových stránkách konkrétních obcí, ve kterých nebo u kterých se uzavírka nachází. Zjišťování informací na různých webových stránkách, které by nebyly předem definovány, by bylo v rámci aplikace velmi obtížné. Po krátké diskuzi jsme došli k závěru, že mnohem důležitější je zjištění dat pro aktuální dopravu, kterou je potřeba zpracovat rychle, aby byla data stále platná. Uzavírky obvykle trvají dlouho, je čas si ručně dohledat bližší informace a zpravidla jsou informace stejné po celou dobu uzavírky. Není proto potřeba před každým vysíláním uzavírky dohledávat všechny informace znovu.

Cílem by bylo vytvořit aplikaci, která získá data z webových stránek Waze a Policie, provede jejich konsolidaci, odstraní nedůvěryhodné a nedůležité dopravní události a zbylé přehledně zobrazí. Před definitivním stanovením tématu práce jsem však musel ověřit, zda je taková aplikace vůbec realizovatelná.

1.2 Ověření realizace aplikace

Dopravní informace na Waze jsou znázorněny graficky v rámci mapy. Pro člověka krásně přehledné, ovšem pro počítač nečitelné. Pro získání dat z určitého zdroje na internetu typicky slouží API. Zjistil jsem, že Waze API nabízí, ale pouze v rámci partnerského programu pro velké instituce, například města. Později jsem však našel i neoficiální API [1], které umožňuje získat data ze zadané oblasti, kde se oblast zadává pomocí vymezení levé, horní, pravé a spodní hranice.

V případě Policie je situace podstatně jednodušší. Přímo na jejich webových stránkách je proklik pro stažení souboru se všemi aktuálními dopravními událostmi. Při HTTP dotazu je pak obsah tohoto souboru vrácen jako odpověď. Díky tomu není nutné soubor stáhnout, otevřít a přečíst.

[1] https://www.waze.com/cs/live-map/
[2] https://aplikace.policie.cz/dopravni-informace/

Při svém pátrání jsem nalezl ještě jeden zdroj dopravních informací, Národní dopravní informační centrum (NDIC) [2]. Toto centrum vzniklo v návaznosti na usnesení vlády z roku 2005 o vytvoření Jednotného systému dopravních informací (JSDI) [3]. V systému se shromažďují data o dopravě z celé České republiky. Data do tohoto systému zasílá Policie ČR, Hasičský záchranný sbor, Ředitelství silnic a dálnic a další subjekty. Odběr těchto dat pro veřejnost je možný právě skrze NDIC. Jednotlivé události jsou v době vzniku zasílány na server odběratele. Já však chci, aby výsledná aplikace byla samostatná a nezávislá na dalších prvcích. Proto jsem se rozhodl tento zdroj dat nevyužít.

1.3 Bližší specifikace cílové aplikace

Mám ověřeno, že data z Waze i Policie dokážu získat, aplikace je tak realizovatelná. Následovalo specifikování, co dalšího by aplikace měla umožňovat. Výsledný výpis dopravních událostí by měl být seřazený podle typů událostí, kterými jsou nehody, policejní hlídky, kolony, komplikace, omezení a uzavírky. Každá událost by měla obsahovat informaci o tom, z jakého je zdroje a také kdy byly získané informace naposledy aktualizovány. Moderátoři také rozlišují dopravu aktuální a univerzální.

Aktuální doprava, jak už z názvu vyplývá, se týká aktuálních událostí na silnicích, jako jsou typicky nehody či kolony. Tato doprava se vysílá živě a jsou potřeba co nejaktuálnější data. Do univerzální dopravy se řadí dlouhodobé události jako jsou uzavírky a omezení (především práce na silnici). Tyto události trvají od několika dnů až po měsíce či roky. Právě k těmto událostem jsou k dispozici podrobné informace na webových stránkách příslušných obcí, jak již bylo zmíněno dříve. Moderátor si tyto informace ručně dohledá a výsledné hlášení o události se může vysílat opakovaně v průběhu trvání těchto událostí. Aplikace by tudíž měla obsahovat rozdělení na tyto dva druhy dopravy.

Poslední užitečnou funkcionalitou je možnost výběru oblasti, ze které mají být data zobrazena. Radio Haná má svůj název odvozen od oblasti, ve které vysílá, Haná. Navíc má však Radio Haná ještě stanici Metropole, která vysílá na území města Olomouce a nejbližšího okolí. Protože moderátoři vysílají dopravní informace pro obě tyto stanice, měla by jim aplikace umožnit si vybrat, pro kterou oblast se mají data zobrazit.

2 Architektura aplikace

Celá aplikace je vytvořena na platformě .NET, konkrétně ve verzi 7, která byla nejnovější verzí v době začátku vývoje aplikace. Celá aplikace je rozdělena na dvě části, backend a frontend. Backend je napsán kompletně v jazyce C#, lze jej samostatně spustit a je možné k němu vytvořit zcela nový frontend (například pro jiného zákazníka). Frontend je vytvořen jako desktopová WPF aplikace.

2.1 Backend

Backend realizuje získávání všech dat z obou zdrojů, Waze i Policie. Následuje zpracování dat nejprve samostatně pro každý zdroj a poté konsolidace dat do jednotného tvaru. Až po tuto část je vše univerzální a lehkou úpravou lze získat data z jakékoli části České republiky. Až v posledním kroku jsou aplikovány filtry dle požadavků moderátorů Radia Haná, které modifikují a mažou vybraná data. Níže podrobněji popíšu jednotlivé třídy, ze kterých se backend skládá.

2.1.1 Waze

Třída `Waze` zajišťuje získání a základní zpracování dat z Waze ze zadané oblasti. Získání dat je realizované skrze neoficiální Waze API na endpointu `https://www.waze.com/row-rtserver/web/TGeoRSS`. Tento endpoint má navíc 4 vstupní parametry, `left`, `right`, `bottom` a `top`. Hodnotou parametrů jsou souřadnice geografického obdélníku, který udává hranice oblasti, ze které jsou data získána. Parametr `left` určuje západní hranici, `right` východní, `bottom` jižní a `top` severní hranici oblasti. Endpoint vrací data ze zadané oblasti ve formátu JSON.

Data získaná z tohoto API obsahují všechny údaje, které moderátoři při vysílání používají. Bohužel má toto API několik nevýhod.

První nevýhoda, na kterou jsem narazil, je ta, že množství dat, které se dá na jedno zavolání získat, je značně omezené. JSON objekt, získaný po zavolání endpointu, obsahuje tři vlastnosti – `alerts`, `jams` a `users`. Každá z těchto vlastností obsahuje pole tomu odpovídajících objektů. Maximální počet objektů pro jednotlivá pole je však omezený. Abych měl jistotu, že pro požadovanou oblast získám všechna data, rozdělil jsem oblast do segmentů. Každému segmentu jsem ručně nastavil jeho velikost tak, aby byl získaný počet objektů v jednotlivých polích zhruba na polovině svých maximálních možností. Počet dopravních událostí je proměnlivý a potřebuji mít jistotu, že i v té největší dopravní špičce vždy získám všechna data. Segmenty jsou obdélníkového tvaru, stejně jako oblasti, které je zadávají jako vstup endpointu Waze API. Pro oblast vysílání Radia Haná jsem takto vytvořil celkem 8 segmentů, takže 8 volání, abych získal všechna požadovaná data. Abych běh programu zbytečně nezpomaloval postupným voláním, pro každé volání jsem vytvořil samostatné vlákno a všechna volání provádím paralelně.

Další nevýhodou jsou informace o uzavírkách. Ačkoli při běžném prohlížení na mapě přímo na stránkách www.waze.com/live-map vypadají uzavírky jako jedna dopravní událost, v pozadí tomu tak není. Ve skutečnosti je jedna uzavírka složena z mnoha objektů. Musel jsem vytvořit algoritmus, který z těchto mnoha objektů týkajících se jedné uzavírky, vytvoří jeden objekt.

Algoritmus projde všechny objekty z pole `jams` a vyfiltruje ty, které mají vlastnost `blockingAlertID`. Takto získané objekty se týkají uzavírek a každý objekt obsahuje vlastnost `line`, jejíž hodnotou je pole bodů. Tyto body pak vytyčují místo uzavírky. Algoritmus postupně projde všechny objekty a pro každý objekt porovná jeho první a poslední bod z pole `line` se všemi prvními a posledními body ostatních objektů. Pokud dojde ke shodě, znamená to, že byl nalezen další objekt týkající se stejné uzavírky. Takto pak mohu vytvořit nový objekt pro uzavírku s vlastností `lines`, která obsahuje pole bodů, jejichž objekty označují stejnou uzavírku. Dále je díky hodnotě vlastnosti `blockingAlertID` dohledán příslušný objekt z pole `alerts` a z tohoto objektu jsou převzaty údaje o počtu palců, data vzniku a popisu uzavírky. Palec u události na Waze udává, kolik dalších uživatelů označilo tuto událost za stále aktuální.

Poslední nevýhodou je to, že je používané API neoficiální. Neexistuje žádná dokumentace, která by vysvětlovala význam jednotlivých vlastností. Na význam některých vlastností lze po čase přijít, u jiných stále netuším, co znamejí a jak byla jejich hodnota určena.

Kromě chybějící dokumentace je tu ještě další problém. Waze může data získaná skrz API kdykoli změnit a nikde o tom není ani zmínka. Půl roku, co jsem s touto API pracoval, bylo vše stejné, v posledním měsíci však Waze provedl několik úprav, na které bylo třeba reagovat patřičnou úpravou kódu. Pokud dojde ke změně dat ze strany Waze, zjistí se to až v praxi. Aplikace je na takovou situaci připavena. Při zpracování chybných či neúplných dat je zapsána výjimka do log souboru a uživatel je upozorněn chybovou hláškou. Z chybového hlášení může uživatel určit, jak závažný problém je a jaká data jsou postihnuta.

2.1.2 Policie

Získání a základní zpracování dat z Policie probíhá ve třídě `PCR`. Data z Policie jsou zasílána ve formátu XML. Proces zpracování dat proto probíhá odlišně oproti Waze, kde byla data reprezentována jako JSON objekt. Zpracování dat je u Policie také mnohem snadnější, neboť se zde nepotýkám s takovými problémy jako u Waze. Přesto jsem však na jeden problém narazil.

Občas se stane, že se některá událost v celkovém výčtu všech událostí vyskytuje vícekrát. Naštěstí každá událost obsahuje unikátní `ID` a v případě opakujícího se výskytu jedné události mají všechny tyto výskyty stejné `ID`. Kontrola duplikátů a jejich případné odstranění je již triviální.

Zatímco v případě dat z Waze mají události (respektive jejich objekty) vlastnost, která říká, o jaký typ události se jedná, v případě dat z Policie tomu tak není. Určení typu pro každou událost je klíčové. Slouží při konsolidaci událostí

k zajištění, že se mohou sloučit pouze události stejného typu. Dále se používají pro řazení výsledných událostí pro uživatele.

Bylo nutné k jednotlivým událostem doplnit jejich typ. K tomuto účelu jsem vytvořil slovník klíčových slov a frází, na základě kterých se pak určí jejich typ. Například události, obsahující ve svém popisu slovo *nehoda*, je nastaven typ na nehoda. Pokud obsahuje frázi *práce na silnici*, je typ nastaven na omezení. V případě události typu nehoda ještě navíc zjišťuji, zda je místo nehody neprůjezdné a zda je daná nehoda se zraněním. Tyto informace později využívám pro upozornění uživatele na tyto skutečnosti. Ve zdrojovém kódu 1 je uvedena ukázka metody `SetSubtypeForAccident`, která slouží ke zjištění dodatečných informací o nehodě[3]. Pro testování uzavření komunikace hledám v popisu místa `input` řetězec „uzavřen“. Může se stát, že je pouze uzavřen jeden jízdní pruh, nikoli celá komunikace. Proto je navíc přidána podmínka, že popis nesmí obsahovat sousloví „jízdní pruh“.

```
1  private void SetSubtypeForAccident(JToken myEvent, string input)
2  {
3     if (input.Contains("uzavřen") && !input.Contains("jízdní pruh"))
4        myEvent["subtype"] = "neprujezdne";
5
6     if (input.Contains("bez zranění"))
7        myEvent["injury"] = "bez";
8     else if (input.Contains("se zraněním") || input.Contains("lehké
          zranění"))
9        myEvent["injury"] = "s";
10    else
11       myEvent["injury"] = "neznamo";
12 }
```

Zdrojový kód 1: Metoda pro určení dodatečných informací pro událost typu nehoda

V rámci zpracovávání dat z Policie si vytvářím nový JSON objekt, abych měl během konsolidace všechna data ve stejném formátu.

2.1.3 Konsolidace

Po získání a zpracování dat z Waze a Policie nastává velmi důležitá část konsolidace těchto dat (konsolidace dat = sloučení dat z více zdrojů do jednoho cílového formátu). Mnoho stejných událostí je nahlášeno jak na Waze, tak na Policii. Navíc se často stává, že je i v rámci Waze nahlášena jedna událost vícekrát kousek od sebe. Všechny tyto případy jsou ošetřeny v rámci třídy `DataConsolidation`, která projde všechna data z Waze a Policie a v případě potřeby je sloučí tak, aby se ve výsledném výpisu neobjevily pokud možno žádné duplikáty. Výstupem této třídy jsou seznamy objektů, kde každý objekt obsahuje informace o jedné

[3] z ukázky byl pro lepší čitelnost odstraněn kód pro ošetření výjimek

konkrétní události. Objekty jsou instancemi tříd, kde každá třída reprezentuje jeden typ dopravní události. Výsledné třídy jsou:

- `Accident` (nehoda),
- `Police` (policejní hlídka),
- `Jam` (kolona),
- `Hazard` (komplikace),
- `RoadConstruction` (omezení),
- `RoadClosed` (uzavírka),
- `Unclassified` (nezařazené).

Jako první se musí zpracovat kolony, které obdržím pouze z Waze. Zde nemohou být duplikáty. Důležité je, že kolona může být způsobena nějakou jinou událostí, typicky nehodou. V takovém případě dokážu z dat z Waze určit, jaká událost tuto kolonu způsobuje. Proto kolonami začínám, aby se všechny další typy událostí mohly odkazovat na již finální objekty kolon.

Po kolonách je možné provést konsolidaci všech zbývajících typů událostí. V případě událostí typu nehoda, policejní hlídka, komplikace, omezení a uzavírka je způsob konsolidace stejný. Vytvořím nový objekt odpovídajícího typu na základě dat z Policie nebo Waze, projdu všechny zbývající události stejného typu a porovnávám je.

Porovnání událostí se provádí podle jejich vzájemné vzdálenosti. Zde popíšu, jak probíhalo porovnání v první verzi aplikace. Později na základě zpětné vazby došlo k úpravě porovnávání, více v podkapitole 2.3.2. Každá událost má vlastnost, která udává, zda se daná událost nachází či nenachází na dálnici. Při porovnávání jsem vypočítal vzdálenost mezi těmito událostmi. Výpočet vzdálenosti mezi dvěma body jsem musel počítat podle souřadnicového systému, ve kterém jsou body zadány. Tím je Světový geodetický systém 1984 (WGS 84) [4]. Pokud byla vypočítaná vzdálenost v toleranci, události jsem sloučil. Hodnotu tolerance jsem stanovil na 500 metrů pro události na dálnici a 200 metrů pro události mimo dálnici. Při sloučení dvou událostí jsem prošel data z druhé události a vybrané údaje jsem přidal do první události. Sečetl jsem jejich počty palců a sloučil slovní popisy. Nakonec jsem porovnal časy aktualizace obou událostí. Čas aktualizace první události jsem nastavil na hodnotu z druhé události, pokud byl tento čas vyšší (novější). Po těchto krocích došlo k odstranění druhé události.

Nyní již zbývají pouze události typu nezařazené. Tyto události mohou být pouze z Policie, které ve svém popisu neobsahují žádné z klíčových slov a frází, a proto jim nebyl nastaven žádný konkrétní typ. Protože u dat z Policie již byly odstraněny duplikáty, zde již žádné porovnání neprobíhá.

Navíc jak již bylo zmíněno u kolon, u každé události z Waze je k dispozici informace, zda daná událost způsobuje kolonu. Pokud to mu tak je, tak k dané

události přidám `ID` kolony, kterou způsobuje. To budu využívat při výpisu událostí pro uživatele v rámci frontend části aplikace.

2.1.4 Radio Haná

Až dosud byl veškerý průběh zpracování dat zcela univerzální a nezávislý na koncovém uživateli. Poslední důležitou třídou v rámci části backend, je třída `RadioHana`, která získaná data upraví pro potřeby moderátorů Radia Haná.

Jako konkrétní příklad mohu uvést třeba odstranění kolon s délkou zdržení pod 4 minuty, kdy takto krátkou kolonu není třeba hlásit. Odstraňují se také všechny události, které jsou pouze ze zdroje Waze a mají 0 palců. Tyto události nemají dostatečnou důvěryhodnost na to, aby mohly být odvysílány.

Poslední velmi důležitou operací této třídy je uspořádání událostí v rámci svých typů. Obecné uspořádání je podle aktuálnosti, tedy podle času, kdy naposledy byla událost aktualizována a tím bylo potvrzeno, že stále platí. Výjimku při uspořádání tvoří nehody a kolony.

V případě nehod mají největší prioritu nehody, které zneprůjezdní danou pozemní komunikaci. To má totiž největší dopad na dopravu, protože je nutné neplánovaně jezdit objízdnou trasou. Zvýšenou prioritu mají také nehody, kde došlo ke zranění osob. V takovém případě se jedná o vážnější nehodu a na místě zasahují složky integrovaného záchranného systému. Proto je nutné o tom řidiče (posluchače) důrazně informovat, aby dbali o to zvýšené pozornosti.

Data o kolonách jsou k dispozici pouze z Waze. Kolony generuje Waze na základě dat získaných z informací o pohybu svých uživatelů. Čas poslední aktualizace je u všech kolon téměř identický, a proto zde uspořádání podle času aktualizace nedává smysl. Kolony jsou uspořádány nejprve do skupin podle jejich lokace a v rámci těchto skupin jsou navíc uspořádány podle časového zdržení, od nejdelšího po nejkratší.

2.1.5 Další pomocné třídy

Kromě již zmíněných hlavních tříd je v backend části ještě několik pomocných tříd. Ty slouží například pro nastavení oblasti, ze které se data získávají. Dále pro rozdělení oblasti do jednotlivých segmentů, jak již bylo zmíněno u třídy Waze. Jsou zde také třídy pro jednotlivé typy událostí, statická třída pro filtrování dat podle nastavené oblasti a v neposlední řadě také statická třída pro logování výjimek do souboru.

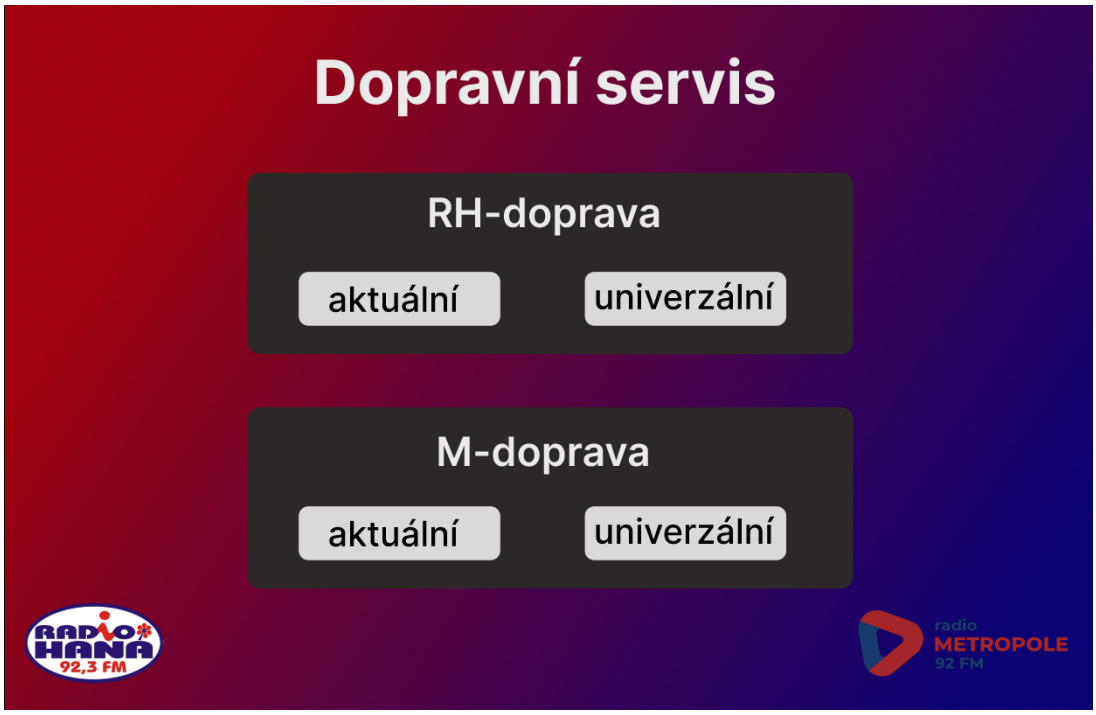
2.2 Frontend

Vývoj frontend části aplikace jsem si rozdělil do dvou kroků. V prvním kroku jsem navrhl jednotlivé stránky pomocí nástroje Figma. Ve druhém kroku jsem již vytvořil samotnou WPF aplikaci právě na základě vytvořeného návrhu.

2.2.1 Návrh aplikace v programu Figma

Na začátek je dobré zmínit, že návrh jsem prováděl v době, kdy jsem ještě neměl žádné zkušenosti s vývojem WPF aplikací. Cíleně jsem nastudování vývoje WPF aplikací odkládal, abych při návrhu aplikace ve Figma nebyl ovlivněn technickými omezeními. Díky tomu jsem mohl navrhovat tak, jak bych si přál, aby výsledná aplikace vypadala a fungovala, místo abych navrhoval pouze to, co je reálně možné provést a co dokážu vytvořit.

Bylo potřeba vymyslet, jak získaná data vhodně zobrazit uživatelům. Důležité bylo rychlé a snadné pochopení významu všech údajů, stejně tak logické uspořádání. Abych mohl svůj nápad vizualizovat, vytvořil jsem mockup pro každou stránku v aplikaci. Mockup je grafický návrh stránky aplikace, důraz je kladen především na rozložení prvků a jejich význam. Návrh neobsahuje funkční prvky, jako například načítání reálných dat a práce s nimi. Mockup pomáhá při konzultaci se zákazníkem, aby bylo možné získat zpětnou vazbu ještě před samotným vývojem aplikace. To je velmi výhodné, protože vytvoření mockup je rychlé a snadné, zatímco vývoj samotné aplikace je mnohem delší a náročnější.

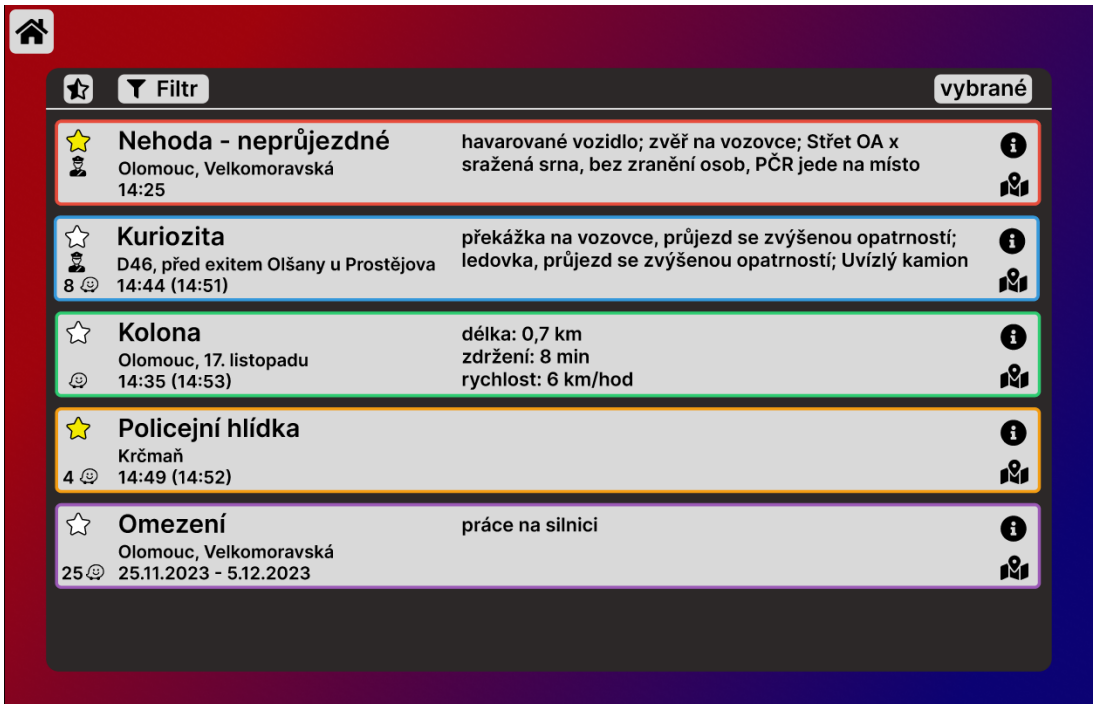


Obrázek 1: Návrh úvodní stránky ve Figma

Když uživatel spustí aplikaci, měl by si jako první vybrat, jaký druh dopravy a z jaké oblasti si chce nechat zobrazit. První stránka by tak měla být nějaký druh rozcestníku. Nejprve mě napadlo udělat rozcestník na dva kroky. V prvním kroku by si uživatel vybral oblast, zda chce dopravu pro Radio Haná nebo Radio Metropole. Ve druhém kroku by si pak zvolil, zda chce dopravu aktuální nebo

univerzální. Tento návrh jsem však rychle přehodnotil, protože by uživatel musel zbytečně provést dva kroky, dvě kliknutí, což je naprosto zbytečné. Proto jsem nakonec zvolil rozcestník s kombinací těchto možností, jak jde vidět na obrázku 1.

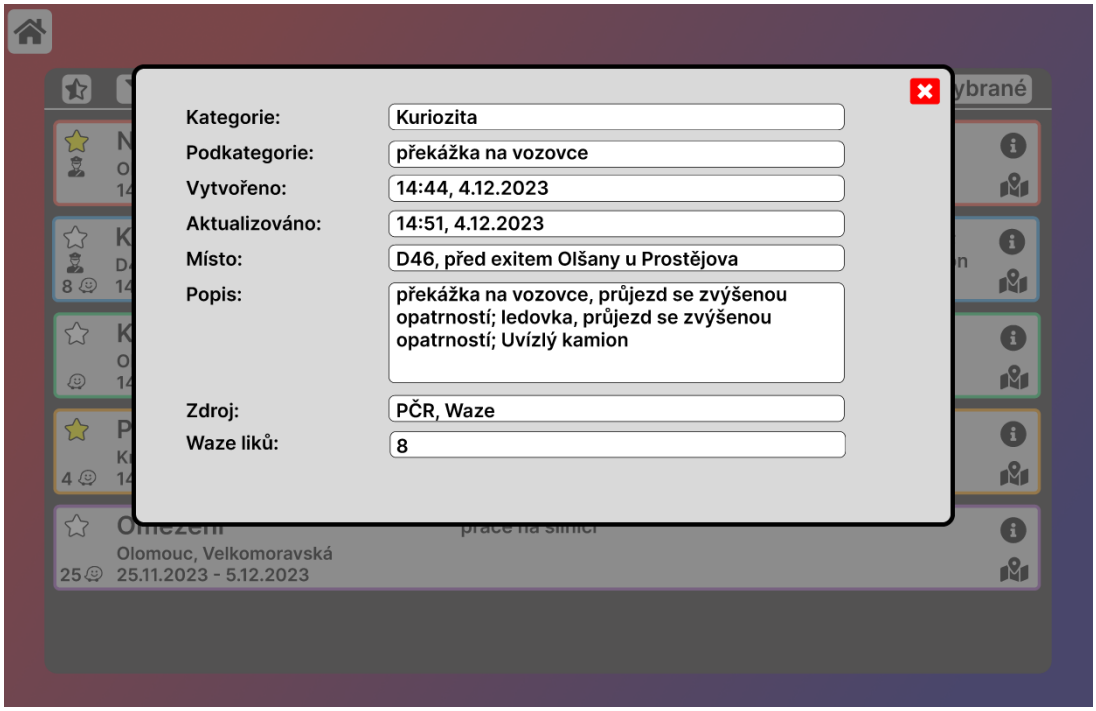
Po výběru oblasti a druhu dopravy se uživateli zobrazí požadovaná data podle zvolených kritérií. Pro strukturovaný výpis dat jsem se rozhodl použít seznam, kde každá událost má svoji vlastní graficky odlišitelnou komponentu. Každá komponenta zobrazuje informace o události, jako je zdroj dat, typ, místo, čas a popis události. Jak je vidět na obrázku 2, každá komponenta má navíc také barevný rámeček. Události stejného typu mají stejnou barvu rámečku. To má pomoci jasně vytyčit, kde končí události jednoho typu a kde začínají události dalšího typu.



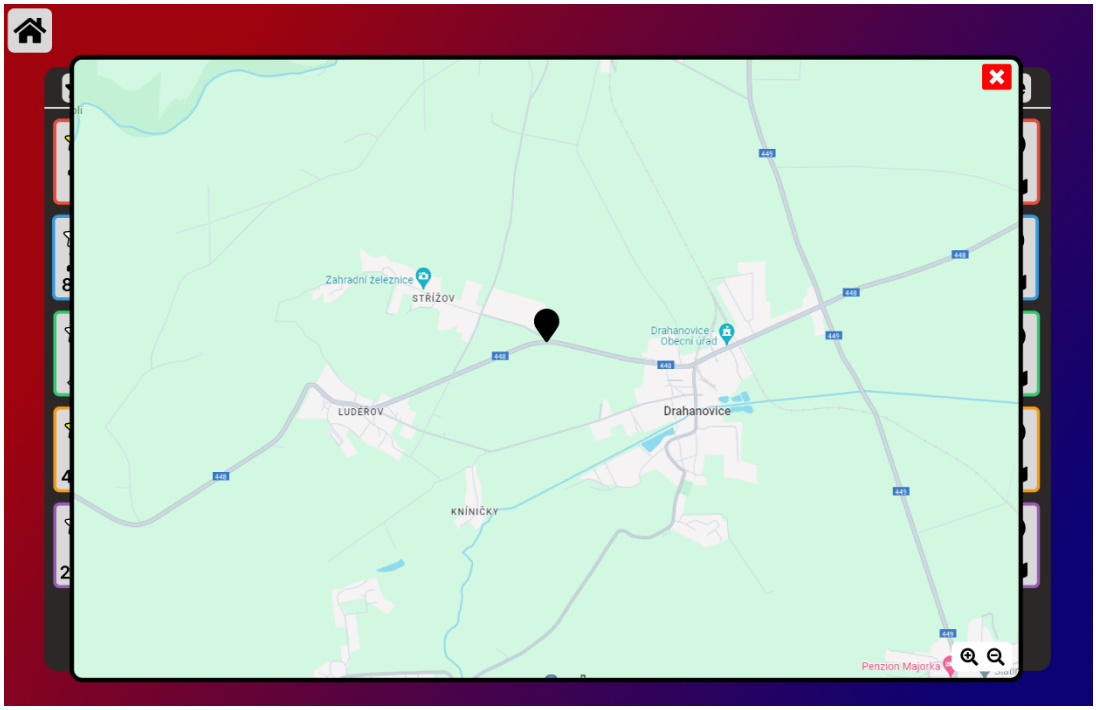
Obrázek 2: Návrh hlavní stránky ve Figma

Pokud by si uživatel chtěl nechat zobrazit data o konkrétní události s vysvětlením všech zobrazených dat, může k tomu využít příslušné tlačítko se symbolem **i**. To provede zobrazení dat ve formuláři, jak je zobrazeno na obrázku 3.

Protože ne vždy stačí slovní popis místa události k tomu, aby bylo jasné, kde se nachází, je možné si otevřít mapu s vyznačeným místem události, jak je vidět na obrázku 4. K tomu slouží ikona mapy v pravém spodním rohu v rámci každé komponenty události.

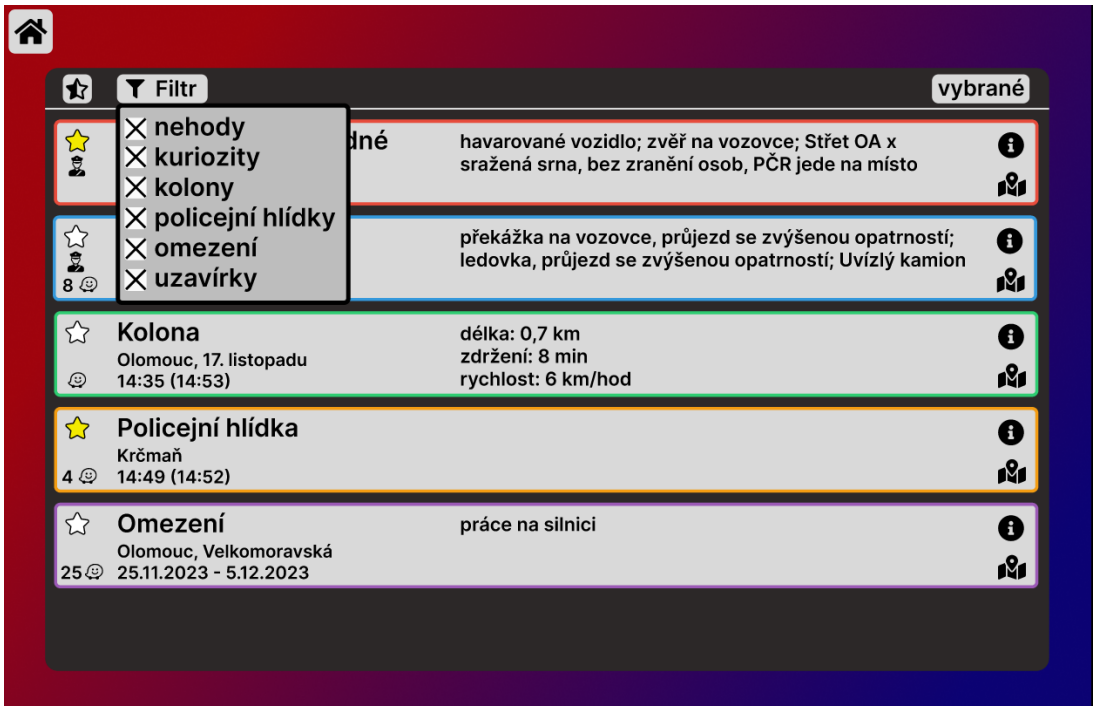


Obrázek 3: Návrh detailu události ve Figma



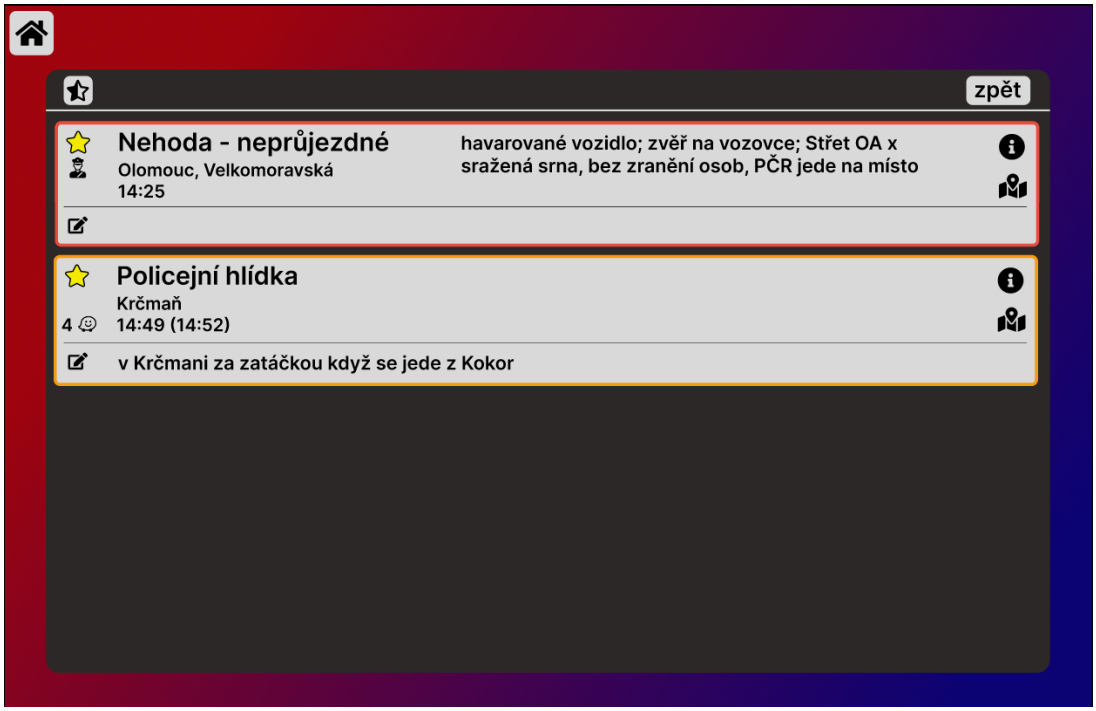
Obrázek 4: Návrh okna s mapou události ve Figma

Zobrazených událostí je poměrně mnoho, proto má uživatel možnost si jimi filtrovat. K tomu slouží okno filtru, kde si jednoduchým označením a odznačením jednotlivých typů událostí uživatel nastaví, které události mají zůstat viditelné a které se mají naopak skrýt. Obrázek 5 ukazuje nastavení filtru, kdy jsou viditelné události všech typů.



Obrázek 5: Návrh filtru událostí na hlavní stránce ve Figma

Ze získaných událostí z backend části již sice byly odstraněny ty, které uživatelé určitě nevyužijí, přesto však musí uživatel zobrazené události ještě sám projít a rozhodnout, které nakonec odvysílá. K tomu mu má sloužit možnost označení události hvězdičkou. Po průchodu všemi událostmi a označení si těch zajímavých, si může uživatel nechat zobrazit pouze ty označené. Takto zobrazené události obsahují navíc novou funkcionalitu. Na obrázku 6 je vidět, že lze k jednotlivým událostem doplnit vlastní poznámku. Hlavní myšlenkou bylo, že si do poznámky může uživatel dopsat přesnější popis místa události poté, co si událost zobrazí na mapě.



Obrázek 6: Návrh stránky s označenými událostmi ve Figma

2.2.2 Vytvoření WPF aplikace

Na základě návrhu ve Figma a zpětné vazby od uživatelů, která je uvedena v kapitole 2.3, jsem mohl přistoupit k vytvoření samotné WPF aplikace. Je důležité zmínit, že je aplikace určena pro operační systém Windows a dále pro větší monitory s úhlopříčkou 15 a více palců. V rámci této podkapitoly se věnuji problematice spojené s vývojem WPF aplikace. Jak vypadá finální verze aplikace, co vše zobrazuje a nabízí, je popsáno v kapitole 3.

Při vývoji jsem se snažil co nejvíce držet vytvořeného návrhu. Pro zobrazení dat jedné události jsem si vytvořil vlastní `UserControl` obsahující všechny potřebné prvky, především `SvgViewBox` pro zobrazení svg obrázků a `TextBox` pro zobrazení formátovaného textu. Každý prvek je umístěn v gridu v příslušném řádku a sloupci tak, aby byly informace přehledné a dobře čitelné. Pro každou událost je vytvořena nová instance tohoto `UserControl`, která obsahuje data odpovídající dané události.

Menší problém nastal při nastavování ikonek. Při použití obrázků ve formátu `png` byl výsledný obraz neostrý. Bylo proto nutné najít a použít obrázky ve formátu `svg`. K tomu bylo zapotřebí doinstalovat vhodný NuGet balíček [5] pro práci s svg obrázky. Když jsem během vývoje spouštěl aplikaci v debug režimu, bylo načítání svg obrázků velmi pomalé a výrazně to zpomalovalo běh celé aplikace. Při běžném spuštění bez debug režimu se však tento problém neprojevoval, nebylo tak zapotřebí dalších úprav.

Mnohem větší oříšek bylo zobrazení mapy v rámci WPF aplikace. Dlouhou dobu mi trvalo najít vhodné řešení, které by splňovalo má očekávání. Chtěl jsem, aby se po kliknutí na ikonu mapy otevřelo nové okno s mapou. Jako hlavní zdroj map jsem chtěl Google mapy, jednak jsou nejrozšířenější a také je moderátoři do teď používali a byli s nimi spokojeni. Nakonec se mi podařilo najít vhodný NuGet balíček `GMap.NET.WinPresentation` [6], který umožňuje implementaci map do WPF aplikace. Dokonce podporuje řadu dalších poskytovatelů map. Od Google, přes Bing, OpetStreetMaps až po Mapy.cz. Uživatelům jsem tak do okna s mapou přidal i možnost výběru typu mapy. Celkem jsou k dispozici 4 možnosti: základní Google mapa, letecká Google mapa (bez popisků), hybridní Google mapa (letecká včetně popisků) a základní Mapy.cz. Okno s mapou obsahuje také označení místa události, pro kterou byla mapa otevřena. Chtěl jsem však uživatelům nabídnout něco navíc.

Může se někde stát nehoda, které znemožní průjezd danou pozemní komunikací. V takovém případě moderátoři doporučují řidičům vhodnou objízdnou trasu. Tu si však moderátoři musí sami ručně najít. Chtěl jsem zjistit, zda by aplikace dokázala navrhnout vhodnou objízdnou trasu. Pro nalezení trasy mezi dvěma body je možné využít Google Maps API [7]. Pro nalezení běžné trasy to funguje perfektně. Pokud je na Google mapách nahlášená uzavírka, tak stále vše funguje dle očekávání a navržená trasa vede mimo uzavírku (takže dostanu objízdnou trasu). Problém je ovšem v momentě, kdy se chci vyhnout určitému místu, například právě z důvodu nehody. O této komplikaci Google mapy neví a tak navržená trasa vede skrze místo nehody, i když je v danou chvíli místo neprůjezdné. I přes dlouhé hledání a zkoušení jsem nenašel žádnou možnost, jak si nechat navrhnout trasu z bodu A do bodu B a přitom se vyhnout bodu C. Takovou možnost bohužel Google mapy nenabízejí, a to i přesto, že by mnoho jejich uživatelů tuto funkcionalitu velmi ocenilo. Snad se tomuto problému bude někdo v budoucnu věnovat, třeba v rámci závěrečné práce.

2.3 Zpětná vazba

S většinou moderátorů (budoucích uživatelů) jsem se sešel a prodiskutoval jejich připomínky a návrhy. V prvním kroku jsme se bavili o návrhu aplikace, ve druhém kroku už o hotové aplikaci.

2.3.1 Zpětná vazba na návrh ve Figma

První zpětnou vazbu jsem získal ještě před zahájením vývoje frontend části aplikace. Cílem bylo ujasnit si hlavní rozložení prvků v aplikaci, výpis událostí, vzhled, ale také zpřesnění některých funkcionalit. V rámci zpětné vazby jsem dostal následující připomínky:

- zrušit barevné ohraničení jednotlivých událostí,
- možnost přidat vlastní poznámku k události se nevyužije,

- přidat možnost zkopírovat událost ve formě textu,
- stanovení řazení událostí dle typu a následně dle aktualizace,
- u časů událostí uvádět i jejich předpokládaný konec,
- kolony způsobené nějakou událostí nevypisovat jako samostatné události, ale přidat je do popisu původní události a označit je jako zdržení,
- přidat vysvětlivky k jednotlivých tlačítkům a údajům po najetí kurzorem myši,
- monitor, na kterém aplikace poběží, je poměrně daleko od moderátora, zajistit dostatečnou velikost písma a tlačítek.

Všechny tyto připomínky jsem zapracoval do aplikace. Odstranil jsem tlačítko pro přidání vlastní poznámky a naopak přidal tlačítko pro zkopírování události. Po kliknutí na něj se zkopíruje typ, místo, časy a popis události. Navíc jsem do horní lišty nad seznamem událostí přidal tlačítko, které takto zkopíruje všechny události. Možnost kopírování je pouze v režimu zobrazení událostí označených hvězdičkou.

2.3.2 Zpětná vazba na aplikaci

Po dokončení jsem aplikaci umístil na počítač ve studiu, aby ji mohli moderátoři začít používat. Po měsíci používání jsem se s nimi opět sešel, abych dostal zpětnou vazbu z reálného provozu. Připomínky z provozu jsou následující:

- přidat tlačítko pro možnost aktualizace dat, aby nebylo nutné se vždy vracet na úvodní stránku,
- zvětšit písmo ve vysvětlivkách,
- upravit řazení uzavírek podle aktuálnosti a počtu palců,
- možnost označit text přímo v rámci události pro kopírování pouze části textu,
- zvětšit rozsah pro slučování událostí,
- přidat mapu s názvy exitů na dálnicích,
- upravit výchozí přiblížení mapy události při otevření okna,
- zobrazit všechen text a možnosti kopírování rovnou, ne až při zobrazení událostí označených hvězdičkou,
- načítání nových dat je pomalé.

Podle získaných připomínek jsem aplikaci náležitě upravil. Některá řešení více rozvedu ve zbytku této podkapitoly.

Slučování událostí nejprve probíhalo čistě na základě vzdálenosti. Pokud byly dvě události stejného typu dostatečně blízko sebe, tak se sloučily do jedné. Maximální možná vzdálenost pro sloučení byla stanovena na 200 metrů, pokud byly události mimo dálnici a 500 metrů, pokud byly na dálnici. Vzdálenost se počítá vzdušnou čarou. Přesto se stávalo, že dvě různá hlášení stejné události nebyla sloučena, protože byla moc daleko od sebe. Proto vznikl požadavek na navýšení těchto limitů. Já jsem se však obával, že navýšením maximální možné vzdálenosti pro sloučení může nastat opačný problém, že by se mohly sloučit i hlášení týkající se různých událostí.

Upravil jsem kód tak, aby rozlišoval události ve městě (s maximální vzdáleností pro sloučení 300 metrů), mimo město (400 metrů) a na dálnici (600 metrů). Navíc jsem přidal novou podmínku. Pokud jsou události stejného typu blíže, než je maximální vzdálenost pro sloučení, ale více než $\frac{1}{2}$ maximální vzdálenosti pro sloučení, musí se navíc nacházet na stejné ulici/silnici/dálnici.

Google mapy i Mapy.cz mají exity na dálnicích označeny pouze číslem, to však řidičům moc neřekne. Proto moderátoři říkají i názvy těchto exitů. Dle požadavku jsem do výběru map přidal možnost použít mapu od OpenStreetMap. Tato mapa má exity označena nejen číslem, ale také jejich názvem.

Moje původní myšlenka byla, že si moderátor nejprve projde všechny události, označí si hvězdičkou ty, které bude chtít vysílat, následně se přepne do režimu zobrazení takto označených událostí a ty pak znovu projde a odvysílá. Realita však byla jiná. Moderátoři si již při prvním průchodu napíší do svého scénáře ty události, které chtějí odvysílat a do žádného dalšího režimu chodit nepotřebují. Problém byl, že v prvním výpisu všech událostí jsem měl omezenou výšku pro každou událost, takže ne vždy byl všechen text viditelný. Navíc zde nebyla možnost kopírovat text události. To jsem podle jejich zpětné vazby změnil, aby vše bylo viditelné a kopírovatelné hned. Systém označování hvězdičkou jsem ponechal, slouží však již pouze jako filtr.

Co mě zaskočilo nejvíce, byla rychlost aplikace při načítání nových dat. Během vývoje na svém počítači byla rychlost přijatelná, do jedné sekundy. Na počítači ve studiu však aplikace běžela výrazně pomaleji až 5 sekund. Řešil jsem, co v aplikaci běží nejdéle a zda by to bylo možné zrychlit. Tři nejnáročnější a nejdelší operace byly načtení dat z Waze a Policie, dále vytvoření uzavírek z dat z Waze a nakonec vytvoření všech grafických prvků ve frontend části.

Zjistil jsem, že načítám a zpracovávám data nejprve z Waze a až po jejich dokončení dělám to samé u dat z Policie. Protože jsou tyto dvě činnosti na sobě nezávislé, umístil jsem jejich vykonávání do samostatných vláken a provádím je nyní paralelně.

Vytvoření uzavírek z dat z Waze trvá velmi dlouhou dobu, protože je potřeba u všech objektů týkajících se uzavírek, projít všechny koncové body a porovnat je se všemi koncovými body všech ostatních objektů. Po konzultaci s moderátory jsme dospěli k závěru, že není nutné zobrazovat uzavírky z Waze v rámci aktuální

dopravy. Aktuální dopravu využívají moderátoři výrazně častěji a uzavírky zde vysílají jen zřídka. Pokud by si chtěl moderátor nechat uzavírky zobrazit, může tak učinit v rámci univerzální dopravy, která obsahuje všechny uzavírky. V rámci aktuální dopravy jsem ponechal uzavírky z Policie, protože nijak neprodlužují čas načtení a zobrazení dat. Navíc se často jedná o neplánované a nečekané uzavírky trvající jen pár hodin nebo dní, takže se i více týkají aktuální dopravy.

Vytvoření grafických prvků trvá dlouho. Bohužel zde nelze provést zrychlení pomocí rozdělení do více vláken, protože práce s UI elementy v rámci WPF aplikace běží vždy v jednom vlákně. Jediné, co zde trochu pomohlo, bylo nastavení výchozích filtrů na získaná data. Ne všechna data získaná z Waze a Policie moderátoři nutně použijí. Proto je možné podle určitých kritérií část těchto dat odstranit a vůbec je nezobrazovat. Zobrazení menšího počtu událostí je jednak rychlejší, protože se musí vytvořit a vykreslit méně prvků, a zároveň moderátor nemusí zbytečně procházet události, které by stejně určitě nevyužil.

3 Uživatelská příručka

V této kapitole je popsána finální verze aplikace z pohledu uživatele. V první podkapitole je popsáno spuštění aplikace. V dalších podkapitolách jsou popsány všechny stránky, jejich význam i obsah.

3.1 Spuštění aplikace

Finální aplikace je publikována jako samostatně spustitelná aplikace. To byl požadavek ze strany Radia Haná, aby aplikace nevyžadovala instalaci, neboť cílový počítač se nachází uvnitř firemní sítě. Aplikace se nachází v adresáři `bin`, který je součástí elektronických dat odevzdaných v systému katedry informatiky. V adresáři se spolu s aplikací nachází několik dalších souborů, které aplikace pro svůj běh používá, není však potřeba žádná instalace. Aplikace se spustí dvojklikem na spustitelný soubor `Dopravni_servis.exe`.

3.2 Úvodní stránka

Po spuštění aplikace se zobrazí úvodní stránka s rozcestníkem, jak je zachyceno na obrázku 7. Hlavní rozdělení je podle oblasti, ze které se mají události zobrazit. Další rozdělení je pak podle druhu dopravy. Aktuální doprava obsahuje všechny typy událostí, jen uzavírky se získávají pouze z Policie. Při výběru univerzální dopravy se zobrazují pouze události typu omezení a uzavírka. Zde se již zobrazují všechny uzavírky z obou zdrojů.

Obrázek 7: Úvodní stránka aplikace

3.3 Hlavní stránka

Po výběru oblasti a druhu dopravy z úvodní stránky, se zobrazí hlavní stránka aplikace. Ta obsahuje seznam všech událostí, odpovídajících zvolenému výběru. Na obrázku 8 je vidět hlavní stránka s výběrem aktuální dopravy pro oblast Radia Haná. Následující popis se týká právě tohoto výběru, pro ostatní výběry to platí analogicky.

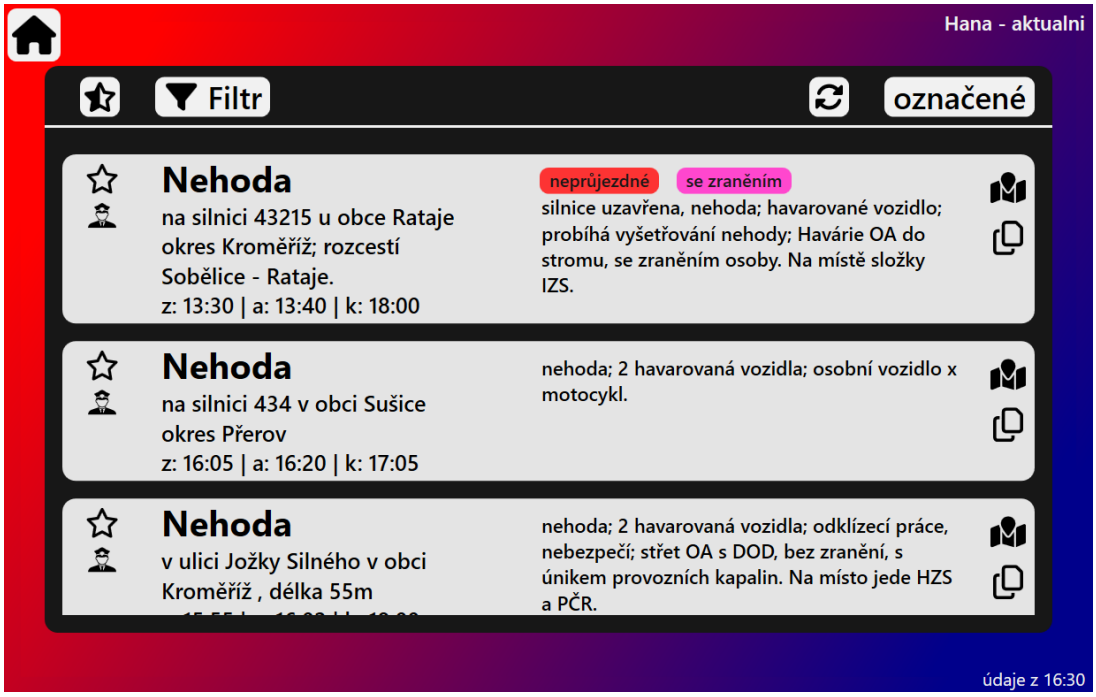
V levém horním rohu se nachází tlačítko s ikonou domečku. Tímto tlačítkem je možné vrátit se zpět na úvodní stránku s rozcestníkem. V pravém horním rohu je uvedeno, jaké oblasti a jakého druhu dopravy se zobrazené události týkají. V pravém spodním rohu se nachází údaj, v jakém čase byla naposledy data o událostech aktualizována.

V samém středu stránky se nachází to nejdůležitější, seznam dopravních událostí. Každá událost se skládá ze tří částí.

První část se nachází úplně vlevo. Nahoře je klikatelná ikona hvězdičky, pomocí které si může uživatel danou událost označit. Uživatel má možnost si nechat zobrazit pouze takto označené události. Dále se v této části nachází informace o zdroji dat. Pokud je zde viditelná ikona policisty, zdrojem dat je Policie. Pokud je viditelná ikona Waze, zdrojem dat je Waze. V případě, kdy je zdrojem Waze, nachází se vedle příslušné ikony i číslo. Toto číslo uvádí počet palců, které daná událost získala od uživatelů. Výjimkou jsou kolony, které jsou na Waze automaticky generovány, a proto neobsahují údaj o počtu palců.

Druhá část, nacházející se uprostřed, obsahuje typ, místo, časy a popis události. Časy jsou zobrazeny dva, někdy i tři. První čas s prefixem *z:* označuje začátek události, tedy kdy byla událost poprvé vytvořena (nahlášena). Druhý čas s prefixem *a:* indikuje, kdy naposledy byla událost aktualizována. V případě dat z Waze je to čas posledního uděleného palce. Poslední čas s prefixem *k:* je odhadovaný konec události. Tento čas je k dispozici pouze u událostí se zdrojem Policie. Výjimku tvoří události typu uzavírka, kde jsou časy nebo pouze data značící začátek a konec uzavírky. U některých uzavírek nemusí být znám konec uzavírky. V takovém případě se doplní symbol pro neznámý údaj, tři otazníky (*???*). Nad popisem události se navíc mohou objevit speciální značky. Ty explicitně upozorňují uživatele na důležité údaje o události. Například u nehod se může objevit značka *neprůjezdné* nebo *se zraněním*, jak je vidět na obrázku 8.

Poslední třetí část události je úplně vpravo a obsahuje dvě klikatelné ikony. První ikona mapy umožňuje otevření nového okna s mapou a vyznačeným místem události. Druhá ikona slouží pro zkopírování události ve formě textu.



Obrázek 8: Hlavní stránka při zobrazení všech událostí

Nad seznamem událostí se nachází lišta s dalšími funkcionalitami, které pracují se všemi událostmi. V pořadí zleva doprava je zde nejprve tlačítko s ikonou hvězdičky. Pokud nejsou označeny všechny události, tak toto tlačítko označí všechny události hvězdičkou. Naopak, pokud jsou všechny události hvězdičkou označené, u všech událostí toto označení zruší. Dalším tlačítkem je filtr, po jehož otevření je možné skrýt či zobrazit určité typy událostí. Seznam typů dostupných ve filtru závisí na výběru druhu dopravy (aktuální, univerzální). Další tlačítko,

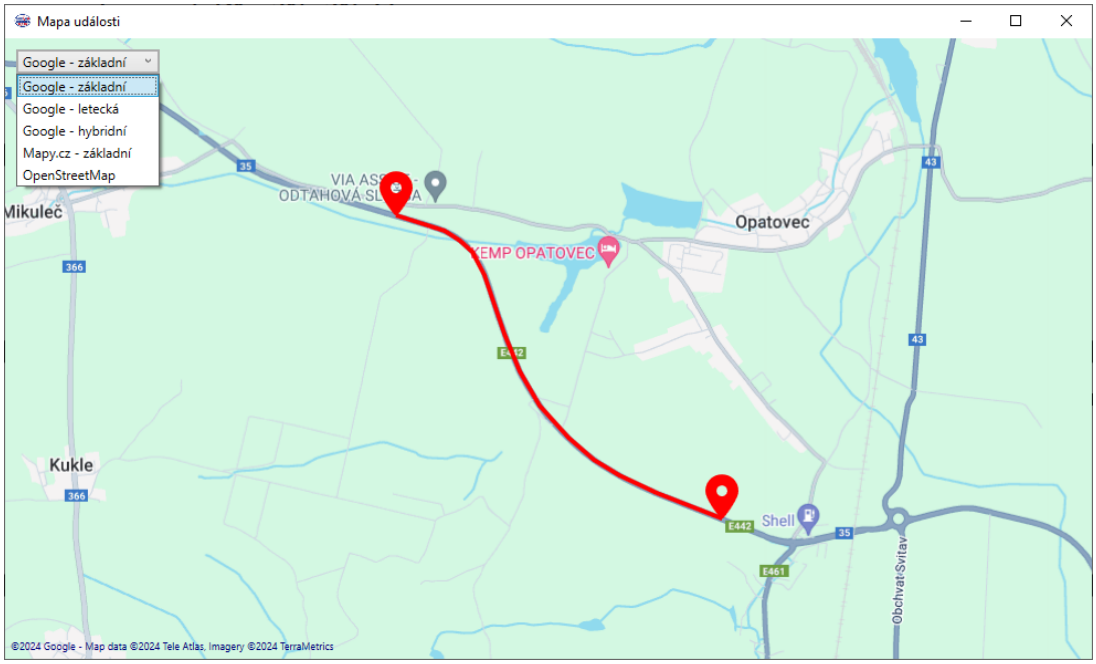
tentokrát se symbolem šipek do kruhu, umožňuje aktualizovat události. Po kliknutí na toto tlačítko se znovu načtou a vykreslí všechny události. Poslední tlačítko s nápisem *označené* slouží pro zobrazení pouze těch událostí, které byly označeny hvězdičkou.

Při zobrazení pouze událostí označených hvězdičkou se upraví nabídka horní lišty, jak je vidět na obrázku 9. Tlačítko pro filtraci událostí je nahrazeno novým tlačítkem pro zkopírování všech událostí v textové formě. Dále zmizelo tlačítko pro aktualizaci dat. Poslední změnou je význam tlačítka vpravo, které nyní odkazuje na zobrazení všech událostí, nejen těch s hvězdičkou. Tomu odpovídá i změna popisu tlačítka na *všechny*.

Obrázek 9: Hlavní stránka při zobrazení pouze označených událostí

3.4 Okno s mapou události

Jak již bylo zmíněno, u každé události je možné si kliknutím na ikonu mapy nechat otevřít a zobrazit mapu s místem události. Díky tomu je možné přesněji zjistit, kde se daná událost nachází. Výchozí mapou je základní mapa od Google. Vlevo nahoře je navíc možnost si změnit mapu na leteckou nebo hybridní (letecká s popisky) od Google, dále základní mapu od Mapy.cz a poslední možností je OpenStreetMap, která zobrazuje nejen čísla exitů na dálnicích, ale i jejich názvy.



Obrázek 10: Okno s mapou zobrazující přesné místo události

Na obrázku 10 je vidět zobrazení události v mapě, když je událost zadána dvěma body. Na začátku a konci se vykreslí značka a také se vyznačí silnice mezi značkami. Událost může být zadána také jedním bodem, v tom případě se vykreslí jen jedna značka v tomto bodě. U kolon a uzavírek jsou vyznačeny pouze dotčené silnice, značky se u těchto typů událostí nevykreslují.

Závěr

Výsledná aplikace dokáže na jedno kliknutí získat, zpracovat a přehledně zobrazit data o aktuálních dopravních informacích z předem definované oblasti. Zobrazené události lze dále označovat, filtrovat, kopírovat a jednotlivě zobrazit na mapě. Aplikace splňuje má očekávání a, dle zpětné vazby od samotných moderátorů Radia Haná, také poskytuje zrychlení a zefektivnění jejich práce při vysílání dopravy.

Možným vylepšením do budoucna by mohlo být zobrazení jedné mapy vedle seznamu událostí. V této mapě by mohly být vykresleny všechny události současně a uživatel by se klikáním na konkrétní události pouze přesouval po mapě. Tím by odpadlo neustálé otevírání nových oken při zobrazování jednotlivých událostí.

Conclusions

The final application can retrieve, process and clearly display data of current traffic information from a predefined area in one click. The displayed events can be further tagged, filtered, copied and individually displayed on a map. The application meets my expectations and, according to feedback from the Radio Haná presenters themselves, it also makes their work faster and more efficient when broadcasting traffic information.

A possible future improvement could be to display one map next to the list of events. This map could then display all events simultaneously and the user could just click on specific events to move around the map. This would eliminate the constant opening of new windows when displaying individual events.

A Obsah elektronických dat

Na samotném konci textu práce je uveden stručný popis obsahu elektronických dat odevzdaných v systému katedry informatiky spolu s textem.

`text/`
: Adresář s textem práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech (textových) příloh, a všechny soubory potřebné pro bezproblémové vytvoření PDF dokumentu textu (případně v ZIP archivu), tj. zdrojový text textu a příloh, vložené obrázky, apod.

`src/`
: Obsahem adresáře jsou všechny zdrojové kódy vytvořené při vývoji aplikace.

`data/`
: Tento adresář obsahuje všechny obrázky použité v aplikaci.

`bin/`
: V adresáři se nachází výsledná aplikace s názvem *Dopravni_servis.exe* spolu s dalšími soubory, které využívá pro svůj běh. Aplikaci lze rovnou spustit bez nutnosti instalace.

`README.txt`
: Textový soubor s informacemi o jednotlivých adresářích. Také obsahuje návod na zprovoznění aplikace pomocí přiložených zdrojových kódů a obrázků.

Literatura

[1] LAHAV, Nimrod. Waze-api [online]. [cit. 2024-04-24]. Dostupné z: https://github.com/Nimrod007/waze-api

[2] NÁRODNÍ DOPRAVNÍ INFORMAčNÍ CENTRUM. Dopravní informace [online]. [cit. 2024-04-23]. Dostupné z: https://dopravniinfo.cz/information

[3] JEDNOTNÝ SYSTÉM DOPRAVNÍCH INFORMACÍ. Ministerstvo vnitra České republiky [online]. [cit. 2024-04-21]. Dostupné z: https://www.mvcr.cz/clanek/jednotny-system-dopravnich-informaci.aspx

[4] WORLD GEODETIC SYSTEM 1984 (WGS 84). NGA - Office of Geomatics [online]. [cit. 2024-04-21]. Dostupné z: https://earth-info.nga.mil/index.php?dir=wgs84&action=wgs84

[5] ELINAMLLC. SharpVectors.Wpf [online]. [cit. 2024-04-23]. Dostupné z: https://www.nuget.org/packages/SharpVectors.Wpf/

[6] JUDERO01. GMap.NET.WinPresentation [online]. [cit. 2024-04-23]. Dostupné z: https://www.nuget.org/packages/GMap.NET.WinPresentation

[7] GOOGLE INC. Google Maps Platform [online]. [cit. 2024-04-24]. Dostupné z: https://developers.google.com/maps