

Aplikace pro měření a vyhodnocování času stráveného na cestách pro Windows Phone 8

The Creation of a Vehicle Tracking System Application for
Windows Phone 8

Bc. Jakub Krampfl

Diplomová práce
2013



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Jakub Kramp**
Osobní číslo: **A11404**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **prezenční**

Téma práce: **Aplikace pro měření a vyhodnocování času stráveného na cestách pro Windows Phone 8**

Zásady pro vypracování:

1. Zpracujte literární rešerši na téma tvorba aplikací pro Windows Phone 8 s důrazem na softwarovou architekturu Model View ViewModel.
2. Analyzujte daný problém a navrhňte řešení.
3. Na základě analýzy vypracujte návrh aplikace.
4. Publikujte aplikaci na Windows store.
5. Demonstrujte výsledky a formulujte závěr.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **WILDERMUTH, Shawn. Essential Windows Phone 7.5: Application Development with Silverlight. Upper Saddle River, N.J.: Addison-Wesley/Pearson Education, 2012, xxxiii, 476 p. ISBN 978-032-1752-130.**
2. **CAMERON, Rob. Pro Windows Phone App Development. 2nd ed. Berkeley, CA: Apress, 2011. ISBN 978-143-0239-376.**
3. **PETZOLD, Charles. Programming Windows Phone 7. Redmond, WA: Microsoft Press, 2010, p. cm. ISBN 978-073-5643-352.**
4. **VAUGHAN, Daniel. Windows Phone 7.5 Unleashed. Indianapolis, Ind.: Sams Pub., 2012, xi, 1095 p. ISBN 06-723-3348-1.**
5. **LECRENSKI, Nick G, Karli WATSON a Robert FONSECA-ENSOR. Beginning Windows Phone 7 Application Development: Building Windows Phone Applications using Silverlight and XNA. Indianapolis, Ind.: Wiley, 2011, xxi, 578 p. ISBN 978-1-118-09628-4.**

Vedoucí diplomové práce:

Ing. Erik Král

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

22. února 2013

Termín odevzdání diplomové práce:

22. května 2013

Ve Zlíně dne 22. února 2013

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Práce popisuje možnosti implementace aplikací pomocí návrhového vzoru MVVM a konkrétně užití MVVM Light Toolkit využitelného nejenom pro Windows Phone. Dále uvádí způsoby využití nástrojů pro manipulaci s mapou a souřadnicovým systémem ať už z pohledu plánování trasy anebo pravidelného sledování polohy. V neposlední řadě je zmíněna práce se souborovým úložištěm SkyDrive prostřednictvím Live Connect API.

Klíčová slova: cestování, plánování, navigace, GPS, Windows Phone, SkyDrive

ABSTRACT

The theses describes ways how to implement applications using the MVVM design pattern and particular use of MVVM Light Toolkit which is usable not only for the Windows Phone. There are also mentioned ways how to use tools for manipulation with map and coordinate system either from the view of route planning or periodical position tracking. Last but not least, there is mentioned how to work with SkyDrive file storage using Live Connect API.

Keywords: travelling, planning, navigation, GPS, Windows Phone, SkyDrive

Rád bych poděkoval panu Ing. Eriku Královi za cenné rady a postřehy při psaní diplomové práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 WINDOWS PHONE	11
1.1 ROZLIŠENÍ.....	11
1.2 DLAŽDICE	12
1.3 LAYOUT.....	15
1.4 TESTOVÁNÍ APLIKACE	16
2 MVVM.....	18
2.1 VIEW	18
2.2 VIEWMODEL.....	18
2.3 MODEL	19
3 MVVM LIGHT TOOLKIT	20
3.1 OBSAH TOOLKITU	20
3.2 VIEWMODELLOCATOR.....	21
3.3 OBSERVABLEOBJECT	22
3.4 MESSENGER	23
3.5 VIEWMODELBASE	25
3.6 RELAYCOMMAND	26
3.7 EVENTTOCOMMAND.....	27
3.8 DISPATCHERHELPER	28
4 APP BAR UTILS.....	29
4.1 BEHAVIORS.....	29
4.2 TRIGGERS	30
5 LIVE SDK	32
5.1 CLIENT ID.....	32
5.2 SCOPES	32
II PRAKTICKÁ ČÁST	33
6 KONVERTORY.....	34
7 ROZŠÍŘENÍ STÁVAJÍCÍCH TŘÍD.....	38
8 LOGOVÁNÍ CHYB	40
8.1 LOGTYPE.....	40
8.2 LOG	40
9 LOKALIZACE.....	42
10 OVLÁDACÍ PRVKY	43
10.1 CUSTOMLISTPICKER	43
10.2 CUSTOMTEXTBLOCK	43
10.3 MAPCONTROLS	44
10.4 STATS	44
10.5 FORMULÁŘE.....	45

11	PŘEVODNÍKY JEDNOTEK	49
11.1	DÉLKA, RYCHLOST A OBJEM	49
11.2	CENA PALIVA	51
11.3	SPOTŘEBA AUTA	51
11.4	OBJEM DAT	52
12	SLUŽBY	53
12.1	NAVIGACE	53
12.2	NASTAVENÍ	53
12.3	SKYDRIVE	55
12.4	DATABÁZE	56
13	MODELY	58
13.1	CESTA	58
13.2	AUTO	59
13.3	MAPA	59
13.4	NAVIGACE	61
13.5	STATISTIKY	62
13.6	SKYDRIVE	63
13.7	XML	64
14	IMPLEMENTACE STRÁNEK	68
14.1	HLAVNÍ ČÁSTI APLIKACE	68
14.2	ÚVODNÍ STRÁNKA	69
14.3	NASTAVENÍ	70
14.4	STATISTIKY	71
14.5	SKYDRIVE	71
14.6	TRASA	73
15	PUBLIKOVÁNÍ APLIKACE	78
	ZÁVĚR	79
	ZÁVĚR V ANGLIČTINĚ	80
	SEZNAM POUŽITÉ LITERATURY	81
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	85
	SEZNAM OBRÁZKŮ	86
	SEZNAM TABULEK	88
	SEZNAM PŘÍLOH	89

ÚVOD

Chytré telefony v dnešní době nabízí pomocí svých nákupních center (Windows Phone Store, App Store, Google Play, ...) spoustu aplikací a služeb, které se přizpůsobují potřebám uživatelů. I s ohledem na to, že lidé více a více cestují, se poohlíží po aplikacích schopných cesty plánovat a nějakým způsobem zaznamenávat.

Společnost Nokia poskytuje uživatelům svých chytrých telefonů prostřednictvím Windows Phone Store v rámci Nokia collection celou řadu aplikací, mezi nimiž je možné nalézt HERE Drive+. Jedná se o turn-by-turn hlasovou navigaci fungující offline. Nevýhodou však je nemožnost plánování trasy skrz více specifických bodů a také není možné si ujetou trasu uložit.

Podobnou aplikací je gMaps využívající Google mapy, ovšem značnou nevýhodou je závislost na internetovém připojení.

Vzhledem k výše popsaným nedostatkům je cílem vytvořit aplikaci pro plánování a zaznamenání jízdy autem s možností ukládání a zpětného načítání dat aplikace určené pro všechny telefony s Windows Phone 8. Pro uchovávání dat bude využito úložiště SkyDrive, které je dostupné na všech platformách.

Při návrhu řešení bude brán zřetel na užití návrhového vzoru Model-View-ViewModel, který slouží pro oddělení aplikační logiky od uživatelského rozhraní. Umožňuje tedy udržovat kód přehlednější, znovupoužitelný a lépe testovatelný.

Pro ulehčení tvorby aplikací pomocí MVVM je k dispozici celá řada volně dostupných frameworků. V rámci této práce bude použit MVVM Light Toolkit.

I. TEORETICKÁ ČÁST

1 WINDOWS PHONE

Windows Phone je mobilní operační systém vyvíjený společností Microsoft a je nástupcem Windows Mobile.

První generace se objevila pod označením Windows Phone 7 a byla uvolněna na podzim roku 2010. Hlavním rysem celého systému je uživatelské rozhraní zvané Metro, ve kterém dominuje dlaždicové uspořádání. Dlaždice odkazují na jednotlivé aplikace či webové stránky, nastavení apod. a také nás mohou informovat např. o počtu nových zpráv (SMS, e-mail...) a událostech z dané aplikace. [1]

Druhou generací je Windows Phone 8 pod kódovým označením Apollo zveřejněná na podzim 2012. Přichází s podporou více jádrových procesorů, NFC technologií, dalších dvou rozlišení včetně HD a také disponuje zpětnou kompatibilitou s aplikacemi vytvořenými pro Windows Phone 7. [2]

Tab. 1. Přehled verzí Windows Phone [1][2]

Verze	Označení	Zveřejnění
Windows Phone 7	-	listopad 2010
	NoDo	březen 2011
Windows Phone 7.5	Mango	květen 2011
	Refresh (Tango)	červen 2012
Windows Phone 7.8	-	leden 2013
Windows Phone 8	Apollo	říjen 2012
	Apollo Plus	?

Postoj Microsoftu vůči hardwarovým požadavkům na zařízení s operačním systémem Windows Phone se zdá být poněkud přísný, což má samozřejmě své klady i zápory. Pro vývojáře to znamená, že nemusí pracně a složitě zjišťovat, zdali konkrétní zařízení obsahuje či neobsahuje specifické funkce a vybavení. Na druhou stranu však uživatel nemá tak široký výběr mezi low-end a high-end zařízeními. [3]

1.1 Rozlišení

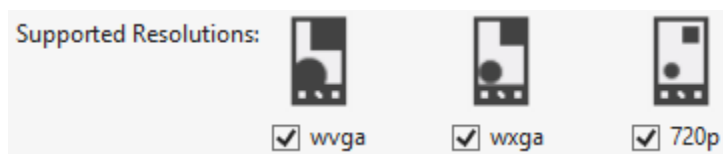
Všechna zařízení s operačním systémem Windows Phone 7.x jsou prodávána pouze s rozlišením – WVGA.

Windows Phone 8 znamenal v jistém smyslu malou revoluci, protože k dosud jedinému rozlišení přibyla další dvě – WXGA a 720p. [4]

Tab. 2. Podporovaná rozlišení [4]

Označení	Rozlišení	Poměr	Poměr [%]	Měřítko	OS
WVGA	480 x 800	15:9	100	480 x 800	7.x/8.0
WXGA	768 x 1280	15:9	160	480 x 800	8.0
720p	720 x 1280	16:9	150	480 x 853	8.0

Výběr podporovaných rozlišení pro danou aplikaci je vždy v plné režii vývojáře prostřednictvím nastavení v souboru WMAppManifest.xml (Obr. 1). Vždy však musí být vybráno minimálně jedno. [4]



Obr. 1. Výběr rozlišení

Vytvoření různých specifik a jejich následné aplikování pro dané rozlišení není nikterak složité, stačí porovnávat vlastnost ScaleFactor (App.Current.Host.Content.ScaleFactor), která vyjadřuje poměr stran v procentech (Tab. 2). [4]

1.2 Dlaždice

Hlavním symbolickým prvkem mobilního operačního systému Windows Phone jsou bezesporu dlaždice. Umožňují zobrazení poutavých informací z dané aplikace na úvodní „Start screen“ obrazovce systému. [5]



Obr. 2. Ukázka dlaždic [5]

Velikostně je lze rozdělit do tří kategorií:

- small (malé),
- medium (střední),
- wide (velké/široké).

Windows Phone 8 aktuálně podporuje tři druhy dlaždic:

- flip,
- iconic,
- cycle.

Tab. 3. Rozměry dlaždic [5]

Velikost	Flip, Cycle [px]	Iconic [px]
Small	159 x 159	110 x 110
Medium	336 x 336	202 x 202
Wide	691 x 336	-

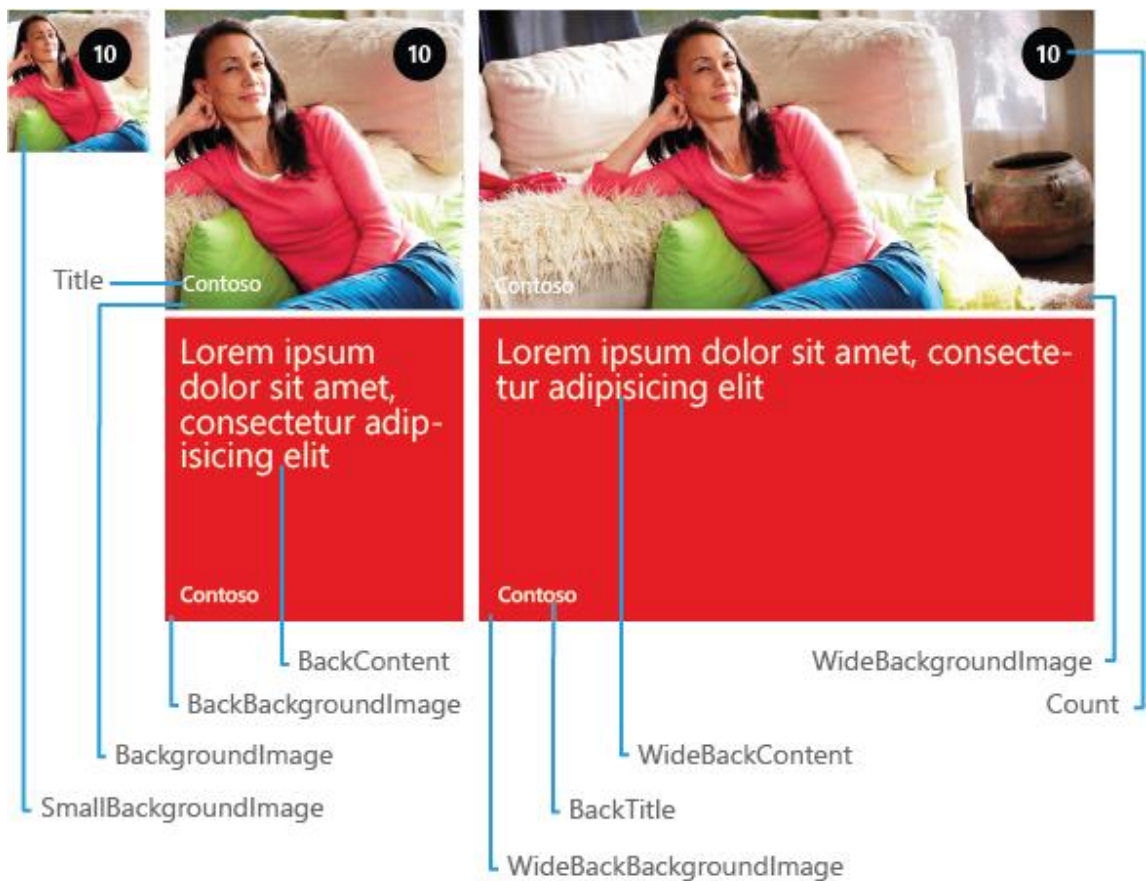
Flip

Flip je základní typ podporovaný již od verze Windows Phone 7¹ a je rozdělen na přední (Front Tile) a zadní část (Back Tile). V rámci přední části je možné nastavit údaj např. o signalizaci počtu nových zpráv, titulek a samozřejmě pozadí. Zadní strana může obsahovat rovněž titulek a navíc krátkou zprávu s jiným podkladovým obrázkem. Celá situace je vyobrazena na obrázku (Obr. 3). [6]

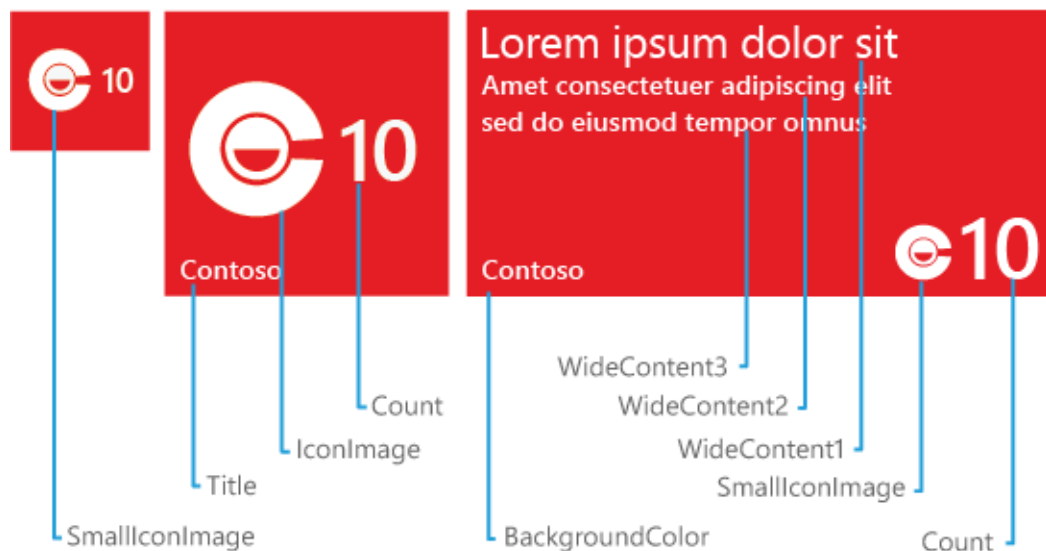
Iconic

Iconic varianta je v jistém smyslu jednotvárnou dlaždicí, protože nedochází ke změně obrázku na pozadí tak jako u předchozího typu. Na druhou stranu však nijak neruší uživatele při čtení informací na dlaždicí (Obr. 4). [6]

¹ Ve verzi Windows Phone 7 je pod označením TileTemplate5 a pouze ve velikosti medium.



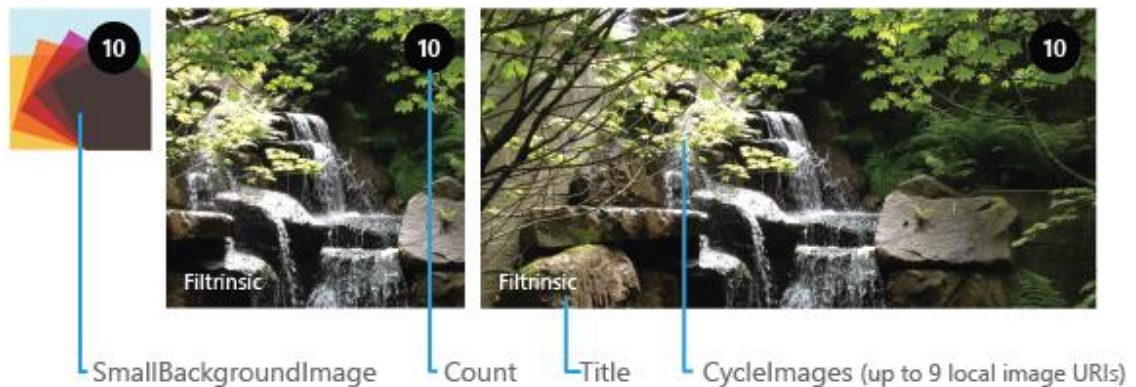
Obr. 3. Flip dlaždice [6]



Obr. 4. Iconic dlaždice [6]

Cycle

Cycle, na rozdíl od předchozích dvou typů, umožňuje rotovat až devět obrázků. Na dlaždici je možné zobrazit pouze titulek a nějaký početní údaj (Obr. 5). Je tedy zaměřena spíše na vizuální prezentaci než na obsahovou. [6]



Obr. 5. Cycle dlaždice [6]

1.3 Layout

Layout jednotlivých stránek je možné vytvořit několika způsoby. Nejjednodušší variantou je použití Grid spolu s užitím předdefinovaných stylů jak pro hlavní nadpisy tak i obsah.

Alternativními a lepšími variantami jsou Panorama a Pivot, které nás oprostují od zbytečného manuálního stylování a navíc umožňují jednoduché přepínání mezi záložkami.

Panorama

Panorama nabízí jedinečnou příležitost jak efektně zobrazit data na dlouhém horizontálním plátně, které sahá až za hranice displeje telefonu (Obr. 6). Za specifikum lze označit způsob zobrazení jednotlivých panelů. Kromě aktuálně vybraného je na pravé straně zobrazen částečný náhled následující záložky.

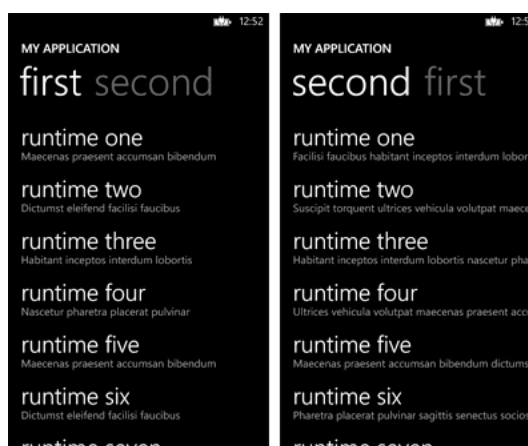
Při návrhu Panorama stránky bychom se měli vyvarovat užití většího počtu panelů. Doporučuje se užití maximálně pěti položek. [7][8]

Pivot

Pivot je založen na principu zobrazení vždy jediné záložky bez částečného náhledu obsahu následujícího prvku, tak jako je tomu v případě Panorama – viditelné jsou pouze nadpisy dalších položek v závislosti na jejich délce (Obr. 7). Pivot je dobré použít hlavně v případě zobrazení stejných nebo podobných dat odlišným způsobem. [7][9]



Obr. 6. Panorama



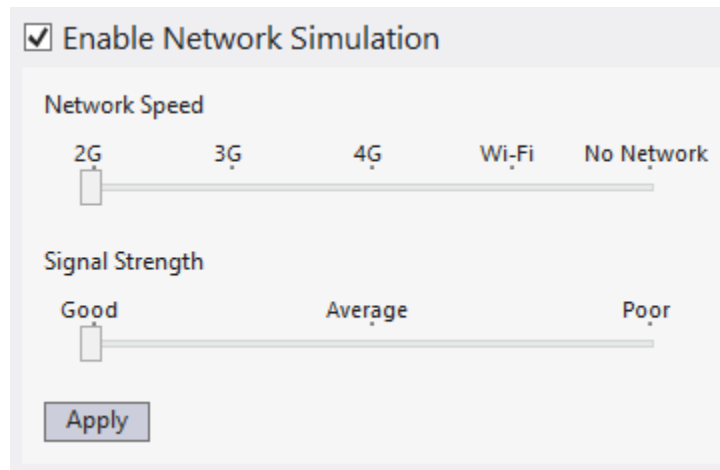
Obr. 7. Pivot

1.4 Testování aplikace

Před samotným publikováním hotové aplikace je vhodné provést jednoduché testy, které nám mohou poodhalit případné problémy. Pro testování lze využít nástrojů integrovaných přímo ve vývojovém prostředí Visual Studio.

Simulation Dashboard

U aplikací, kde využíváme připojení k internetu, můžeme využít nástroje Simulation Dashboard, který dokáže simulovat různé druhy a kvalitu internetového připojení. Nabízí nám tak jedinečnou možnost otestovat chování aplikace v případě méně kvalitního připojení.



Obr. 8. Simulation Dashboard

Windows Phone Application Analysis

Jedná se o monitorovací a profilovací nástroj pro ověření výkonnosti celé aplikace.

Monitorovací nástroj pomáhá identifikovat a řešit následující problémy:

- pomalý start aplikace (za akceptující se považuje 0-5 s),
- pomalá odezva na podněty od uživatele,
- velká spotřeba baterie,
- odezva síťového připojení apod. [10]

Profilovací nástroj se zaměřuje především na nadměrné využívání zdrojů telefonu, tedy:

- využití CPU,
- spotřeba baterie v mAh,
- využití paměti,
- snímkování apod. [10]

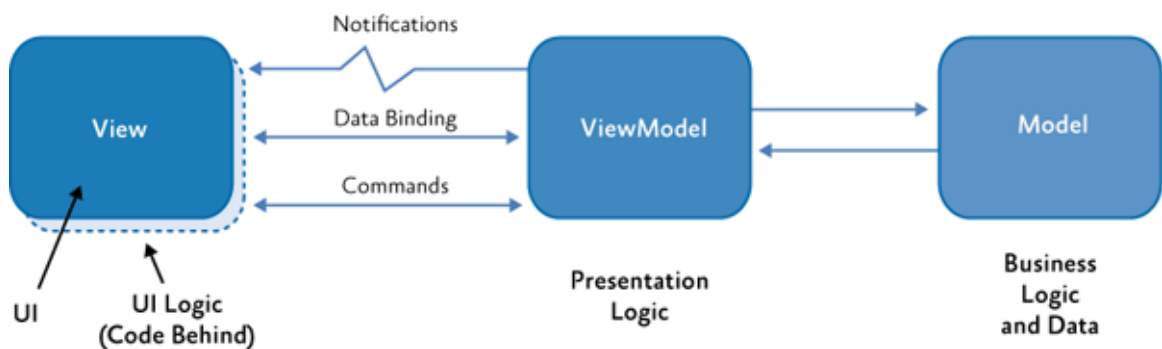
Analýzu není radno podceňovat, protože uživatelé očekávají především plynulou a rychlou odezvu aplikace a nechtějí na nic zbytečně čekat. Veškeré nedostatky pak kriticky hodnotí prostřednictvím recenzí, na základě kterých si další lidé stahují či nestahují danou aplikaci.

2 MVVM

Model-View-ViewModel (MVVM) je návrhový vzor pomáhající udržovat logiku aplikace a uživatelské rozhraní vzájemně oddělené a nezávislé na sobě, což vede především k čistšímu kódu. Toto řešení je přínosné jak pro samotné vývojáře a designéry, tak napomáhá i lepšímu testování, údržbě a dalšímu rozvoji aplikace. [11]

Jak už název napovídá, je aplikace rozdělena do tří oblastí:

- View (pohled) – uživatelské rozhraní,
- ViewModel – zapouzdřuje prezentační logiku a uchovává si stavy,
- Model – struktura dat a jejich získání.



Obr. 9. Logika MVVM vzoru [11]

2.1 View

Pohled má za úkol definovat strukturu a vzhled toho, co uživatel uvidí na své obrazovce – tvorba v jazyce XAML. Kód v pozadí by měl obsahovat pouze inicializaci jednotlivých komponent a také je zde možné realizovat různé vizuální efekty, animace apod., které by bylo složité implementovat přímo v XAML.

Propojení pohledu na ViewModel je vytvořeno prostřednictvím vlastnosti DataContext. [11]

2.2 ViewModel

ViewModel slouží jako prostředník mezi pohledem a modelem. Implementuje vlastnosti a příkazy, které může pohled využívat pomocí provázání (data binding). Dále připravuje a upravuje data do požadovaného tvaru. Rovněž informuje pohled o případných změnách hodnot vlastností skrz notifikační rozhraní `INotifyPropertyChanged`.

Pokud pracujeme s kolekcí dat, je nutné implementovat rozhraní `INotifyCollectionChanged`, resp. je možné využít již existující typ generické kolekce `ObservableCollection<T>` implementující toto rozhraní. [11]

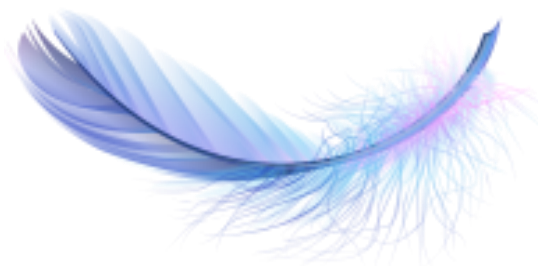
2.3 Model

Model popisuje samotná data a jejich strukturu, případně slouží k získání dat z databáze či souborů. Co však model nesmí obsahovat je aplikační logika.

Z pohledu MVVM je standardní implementovat `INotifyPropertyChanged` v rámci `ViewModel`, ale je možné jej aplikovat i na model, což se hodí především při aktualizaci dat v kolekcích, kde si tím ušetříme spoustu nadbytečného kódu, který by náš model musel obalovat kvůli informování pohledu o případných změnách. [11][12]

3 MVVM LIGHT TOOLKIT

MVVM Light Toolkit je sada komponent dostupná pod MIT licenci umožňující a především ulehčující práci při tvorbě aplikací pomocí návrhového vzoru Model-View-ViewModel. Autorem toolkitu je Laurent Bugnion. [13]



Obr. 10. Logo MVVM Light Toolkit [13]

Toolkit je možné získat přímo ze stránek Codeplex [13] anebo pomocí rozšíření NuGet.

NuGet je nástroj pro instalaci a aktualizaci knihoven třetích stran a je součástí vývojového prostředí Visual Studio. V případě užití NuGet jsou žádané knihovny staženo do složky packages v adresáři našeho solution. V rámci každého projektu je veden konfigurační soubor packages.config, jenž obsahuje informace o stažených knihovnách a jejich verzích. [14]

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <packages>
3   <package id="AppBarUtils" version="2.2" targetFramework="wp80" />
4   <package id="LiveSDK" version="5.3" targetFramework="wp80" />
5   <package id="MvvmLight" version="4.1.26.1" targetFramework="wp80" />
6 </packages>
```

Obr. 11. Ukázka souboru packages.config

Zvýrazněný řádek (Obr. 11) dokumentuje nainstalovaný MVVM Light Toolkit ve verzi 4.1.26.1 pro cílovou platformu Windows Phone 8.

3.1 Obsah toolkitu

Samotný toolkit je rozdělen do dvou knihoven obsahujících jednotlivé komponenty.

GalaSoft.MvvmLight.WP8

- ViewModelBase
- ObservableObject
- RelayCommand
- Messenger (DialogMessage, NotificationMessage...)
- DispatcherHelper

GalaSoft.MvvmLight.Extras.WP8

- EventToCommand
- SimpleIoc

3.2 ViewModelLocator

Třída ViewModelLocator se dá považovat za centrální prvek toolkitu a tedy zároveň celé aplikace. V konstruktoru se provádí registrace služeb a ViewModel pomocí SimpleIoc.

```
1 SimpleIoc.Default.Register<ISkyDriveService, SkyDriveService>();  
2 SimpleIoc.Default.Register<MainViewModel>();
```

SimpleIoc

SimpleIoc je v tomto případě jednoduchý IoC kontejner při jehož realizaci se autor inspiroval řešením Kevina Jonese [16]. Umožňuje již zmíněnou registraci instancí tříd (obdoba cache paměti), získání vytvořených instancí prostřednictvím GetInstance a samozřejmě uvolnění zdrojů pomocí Unregister. [13]

```
3 SimpleIoc.Default.GetInstance<MainViewModel>();  
4 SimpleIoc.Default.Unregister<MainViewModel>();
```

Kontejner implementuje rozhraní:

- ISimpleIoc,
- IServiceLocator,
- IServiceProvider.

Vlastnosti

Dále jsou v rámci ViewModelLocator realizovány vlastnosti, které umožňují přístup k jednotlivým instancím ViewModel. Samotné vlastnosti pak slouží pro propojení pohledů s ViewModel.

```
5 public MainViewModel Main  
6 {  
7     get { return ServiceLocator.Current.GetInstance<MainViewModel>(); }  
8 }
```

Propojení pohledů s ViewModel

Abychom mohli propojit pohledy s příslušnými ViewModel, zaregistrujeme náš ViewModelLocator v App.xaml – bude tak přístupný ve všech pohledech pod označením Locator.

```
1 <Application.Resources>
2     <ResourceDictionary>
3         <vm:ViewModelLocator
4             d:IsDataSource="true"
5             x:Key="Locator" />
6     </ResourceDictionary>
7 </Application.Resources>
```

V pohledu pak stačí nastavit kořenovému prvku stránky (PhoneApplicationPage) DataContext, kde zdrojem (Source) je náš Locator a cesta (Path) odkazuje na vlastnost Main (vlastnost ViewModelLocator), jež vrací instanci MainViewModel.

```
8 DataContext="{Binding Path=Main, Source={StaticResource Locator}}"
```

3.3 ObservableObject

ObservableObject slouží pro obalení jiných tříd (typicky Model, ViewModel aj.), u kterých potřebujeme realizovat notifikaci při změně hodnoty vlastnosti. Z toho důvodu musí samozřejmě implementovat rozhraní:

- INotifyPropertyChanged,
- INotifyPropertyChanging. [13]

Použití je pak možné pomocí dvou alternativ. První delší varianta využívá ruční kontrolu, zdali se hodnota vlastnosti změnila či nikoliv.

```
1 private string _title;
2 public string Title
3 {
4     get { return this._title; }
5     set
6     {
7         if (value == this._title) { return; }
8         this._title = value;
9         this.RaisePropertyChanged(() => this.Title);
10    }
11 }
```

Druhá varianta automatizuje výše popsany proces v rámci metody Set.

```
1 protected bool Set<T>(Expression<Func<T>> propertyExpression, ref T
  field, T newValue);
2 protected bool Set<T>(string propertyName, ref T field, T newValue);
```

V případě, že dojde ke změně hodnoty, je vyvolána událost PropertyChanged a metoda Set vrací true, v opačném případě false.

```
1 private string _title;
2 public string Title
3 {
4     get { return this._title; }
5     set { this.Set(() => this.Title, ref this._title, value); }
6 }
```

3.4 Messenger

Messenger je určen pro komunikaci uvnitř aplikace prostřednictvím zasílání zpráv. Zpráva může být libovolného typu ať už základní typy string, int, nebo i libovolná třída.

Odeslání zprávy se provádí generickou metodou Send<T>. [13]

```
1 Messenger.Default.Send<string>("text");
```

Abychom vůbec mohli zprávu někde zachytit a zpracovat, musíme ji zaregistrovat. Registraci je vhodné provést např. v konstruktoru cílové třídy pomocí metody Register, jež je přetížená a je tedy možné využít několik variant. Základní nejpoužívanější implementace obsahuje dvojici parametrů – příjemce zprávy a akci, která se má vykonat. [13]

```
2 Messenger.Default.Register<string>(this, action =>
  this.DoSomething(action));
```

MVVM Light Toolkit však zatím postrádá mechanismus pro kontrolu již zaregistrovaných zpráv. Aby tedy nedocházelo k opětovnému zaregistrování při znovu načtení stránky (při registraci zpráv v pohledech), je vhodné registraci provádět v metodě OnNavigatedTo a rušení v OnNavigatedFrom. Alternativní variantou pak může být registrace v konstruktoru pohledu a rušení v metodě OnRemovedFromJournal. [15]

DialogMessage

Zobrazení dialogových zpráv je sice možné realizovat pomocí standardního zápisu MessageBox.Show, ovšem s ohledem na MVVM návrhový vzor to však není příliš správné řešení. Proto je zde k dispozici třída pro zasílání dialogových zpráv DialogMessage.

V parametrech konstruktoru je možné předávat samotný obsah zprávy a metodu, která se má vykonat v případě, že uživatel kliknul na jedno ze zobrazených tlačítek. [13]

```
1 DialogMessage message = new DialogMessage(content, result =>
2 {
3 if (result == DialogResult.OK) { /* do something */ }
4 });
5 message.Button = MessageBoxButton.OKCancel;
6 message.Caption = caption;
7 Messenger.Default.Send<DialogMessage>(message);
```

Aby se nám zpráva vůbec zobrazila, je nutné ji také zaregistrovat na některém z pohledů.

```
8 Messenger.Default.Register<DialogMessage>(this, msg =>
9 {
10 DialogResult result = MessageBox.Show(msg.Content, msg.Caption,
    msg.Button);
11 msg.ProcessCallback(result);
12 });
```

NotificationMessage

Notifikační zprávy lze s úspěchem využít v situaci, kdy pracujeme se stejným objektem, ale potřebujeme nad ním vykonávat rozdílné činnosti, jako jsou: vytvoření, editace a smazání.

```
1 Messenger.Default.Send<NotificationMessage<T>>(new
    NotificationMessage<T>(T, string));
```

Do zprávy předáváme samotný objekt a řetězec popisující odpovídající činnost. U příjemce je pak možné danou činnost rozlišit prostřednictvím vlastnosti Notification. [13]

NotificationMessageAction

Obdoba předchozího typu s tím rozdílem, že dědí NotificationMessageCallback a může tedy zpětně informovat o případných dokončených změnách. [13]

PropertyChangedMessage

Tento druh zprávy je možné využít pouze pro zachytávání tzv. broadcastů (viz dále).

3.5 ViewModelBase

Při vytváření aplikace s mnoha pohledy a tedy i ViewModel by bylo velmi obtížné a pracné udržovat některé společné aspekty našich ViewModel, proto je zde k dispozici ViewModelBase.

Jedná se o abstraktní třídu implementující již zmíněný ObservableObject a rozhraní ICleanup. Z třídy ObservableObject přetěžuje metody RaisePropertyChanged a Set, u nichž jsou přidány parametry pro zasílání tzv. broadcastů.

```
1 protected bool Set<T>(string propertyName, ref T field, T newValue,
   bool broadcast);
2 protected virtual void RaisePropertyChanged<T>(string propertyName,
   T oldValue, T newValue, bool broadcast);
```

Broadcast slouží pro obeznámení jiných objektů o změně hodnoty určité vlastnosti prostřednictvím PropertyChangedMessage. Jelikož je informace zasílána jako zpráva pomocí Messenger, je nutné ji v cílovém objektu zaregistrovat. [13]

```
3 Messenger.Default.Register<PropertyChangedMessage<T>>(this, action
   => { /* do something */ });
```

Parametr action obsahuje:

- původní hodnotu,
- novou hodnotu,
- název vlastnosti, u níž byla hodnota změněna,
- odesílatele,
- cíl.

V rámci „do something” pak můžeme navázat řadu dalších potřebných úkonů spojených s touto notifikací.

ViewModelBase dále nabízí prostředky pro využití zasílání zpráv MessengerInstance a vlastnost IsInDesignMode a jeho statickou variantu IsInDesignModeStatic, které můžeme využít při návrhu pohledů v aplikacích Visual Studio či Blend.

V neposlední řadě je také nutné zmínit možnost předávání parametrů v konstruktoru, jimiž mohou být některé typy služeb (navigace, SkyDrive, nastavení, apod.).

```
1 public MainViewModel(INavigationService navigationService,
   ISkyDriveService skyDriveService) { ... }
```

3.6 RelayCommand

V rámci MVVM vzoru jsou akce nebo operace, které může uživatel provádět prostřednictvím uživatelského rozhraní, obvykle definovány jako příkazy. Příkazy poskytují pohodlný způsob jak propojit např. tlačítka s následnou akcí a přitom samotný kód obsluhy je oddělen od vizuální podoby tlačítka.

MVVM Light Toolkit realizuje příkazy pomocí RelayCommand, jenž implementuje rozhraní ICommand. Obsahuje jak klasickou variantu tak i generickou RelayCommand<T> a umožňuje přijímat příkazy i s parametry libovolného typu což se hodí např. v případě reakcí na událost Tap.

V konstruktoru třídy se předává metoda, která se vyvolá při aktivaci příkazu a volitelně parametr CanExecute, kterým lze ovlivnit, zdali příkaz půjde aktivovat (true) či nikoliv (false). [13]

```
1 public RelayCommand(Action<T> execute, Func<T, bool> canExecute);
```

Parametr CanExecute je výhodné užít u příkazů, po jejichž aktivaci dochází k uložení záznamu, obsahujících např. titulek, popis apod., do kolekce či databáze. V tomto případě je totiž nežádoucí, aby se uložil záznam s prázdným titulkem. Pro ošetření tohoto stavu si deklaruujeme vlastnost typu boolean vracející hodnotu true v případě, že délka titulku není nulová.

Deklarace a vytvoření instance příkazu pro obsluhu tlačítka „Save“:

```
1 public RelayCommand SaveCommand { get; private set; }
2 this.SaveCommand = new RelayCommand(this.SaveRecord, () =>
  this.SaveCommandCanExecute);
```

Ošetření nulové délky titulku pomocí CanExecute:

```
3 public bool SaveCommandCanExecute
4 {
5     get { return !String.IsNullOrEmpty(this.Title); }
6 }
```

V rámci vlastnosti Title pak při každé změně provedeme aktualizaci stavu zavoláním metody RaiseCanExecuteChanged nad našim příkazem.

```
7 private string _title;
8 public string Title
9 {
10     get { return this._title; }
11     set
12     {
13         if (this.Set(() => this.Title, ref this._title, value))
14         {
15             this.SaveCommand.RaiseCanExecuteChanged();
16         }
17     }
18 }
```

3.7 EventToCommand

S předchozí kapitolou bezesporu souvisí již zmiňovaný převod událostí na příkazy. EventToCommand se používá doslova k nahrazení libovolné události příkazem přímo v XAML.

Pro použití v XAML je nutné nadefinovat příslušný jmenný prostor, v tomto případě bude k dispozici pod označením „cmd“:

```
1 xmlns:cmd="clr-namespace:GalaSoft.MvvmLight.Command;assembly=
   GalaSoft.MvvmLight.Extras.WP8"
```

Ukázka převodu události Tap na příkaz TapCommand (vlastnost typu RelayCommand):

```
2 <i:Interaction.Triggers>
3     <i:EventTrigger
4         EventName="Tap">
5         <cmd:EventToCommand
6             Command="{Binding Path=TapCommand}" />
7     </i:EventTrigger>
8 </i:Interaction.Triggers>
```

V rámci EventToCommand je možné předávat i parametry ať už s využitím data binding nebo tzv. hard coded, případně si nechat předat automaticky EventArgs ze zdrojové události nastavením atributu PassEventArgsToCommand na hodnotu true. RelayCommand pak deklarujeme jako generický typ s odpovídajícím typem proměnné. [17]

Parametry, na které je možné aplikovat data binding:

- CommandParameter,
- MustToggleIsEnabled.

Hard coded parametry, využitelné v objektech, které nejsou součástí FrameworkElement a tudíž není možné využít data binding:

- CommandParameterValue,
- MustToggleIsEnabledValue.

MustToggleIsEnabled slouží pro povolení, nebo zakázání provázanosti atributu IsEnabled s aktivačním parametrem příkazu CanExecute. Pokud je hodnota atributu nastavena na true a CanExecute vrací false, bude ovládací prvek zakázán. Při nastavení na false nebude bráno CanExecute v potaz. [17]

3.8 DispatcherHelper

DispatcherHelper je pomocná třída pro řízení operací na UI vlákne. Inicializace je navázána na událost Startup elementu Application (App.xaml.cs). [13]

```
1 private void Application_Startup(object sender, StartupEventArgs e)
2 {
3     DispatcherHelper.Initialize();
4 }
```

Provedení akcí na UI vlákne je uskutečněno pomocí metody CheckBeginInvokeOnUI:

```
5 DispatcherHelper.CheckBeginInvokeOnUI(() =>
6 {
7     /* do something */
8 }
```

4 APP BAR UTILS

V základní verzi `AppBar` není možné přímo využít `data binding` ať už k napojení příkazů nebo k provedení lokalizace.

V jistých případech je sice možné napojit příkaz na událost `Click` a provést lokalizaci při vytváření tlačítek v pozadí XAML (code behind). Toto řešení má ovšem značnou nevýhodu v tom, že není možné jednoduše realizovat vizuální podobu zablokovaného tlačítka pomocí `IsEnabled`.

Z výše popsaných důvodů je nutné sáhnout po knihovnách, jež rozšiřují základní verzi o možnosti provázání pomocí `data binding`.

Jednou z takových knihoven je `AppBarUtils` dostupná pod MIT licenci ze stránek Codeplex, případně prostřednictvím NuGet.

`AppBarUtils` rozšiřuje základní `AppBar` o chování a spouštěče (behaviors, triggers).

4.1 Behaviors

Pomocí behaviors můžeme na tlačítko resp. položku menu navázat příkaz nebo akci navigace. Svázání je realizováno pomocí atributu `Text` u tlačítka resp. položky menu a `Id` vlastnosti u `AppBarItemCommand` resp. `AppBarItemNavigation`. V případě, že se jedná o položku menu, musíme navíc specifikovat atribut `Type` jako `MenuItem`. [18]

`AppBarItemCommand`

```
1 <appBar:AppBarItemCommand
2     Command="{Binding Path=ButtonCommand}"
3     Id="button"
4     Text="{Binding Path=LocalizedResources.Button,
5     Source={StaticResource LocalizedStrings}}" />
```

`AppBarItemNavigation`

```
1 <appBar:AppBarItemNavigation
2     Id="button"
3     TargetPage="/View/NewPageView.xaml"
4     Text="{Binding Path=LocalizedResources.Button,
5     Source={StaticResource LocalizedStrings}}" />
```

4.2 Triggers

Triggers je možné využít pro vytvoření dynamického řešení, kde se bude více AppBar přepínat v závislosti na vybraném PivotItem resp. PanoramaItem.

Vzájemné provázání je vytvořené pomocí indexů záložek a atributu Id příslušného AppBar. [18]

```
1 <appBar:SelectedItemChangedTrigger>
2     <appBar:SwitchAppBarAction>
3         <appBar:AppBar Id="0">...</appBar:AppBar>
4         <appBar:AppBar Id="1">...</appBar:AppBar>
5     </appBar:SwitchAppBarAction>
6 </appBar:SelectedItemChangedTrigger>
```

Mapování

Pokud bychom chtěli využít jeden AppBar pro více záložek, je možné využít mapování. SourceIndex odpovídá indexu záložky, kterému chceme přiřadit AppBar, jehož Id odpovídá TargetIndex. [18]

```
1 <appBar:SelectedItemChangedTrigger.SelectionMappings>
2     <appBar:SelectionMapping
3         SourceIndex="1"
4         TargetIndex="0" />
5 </appBar:SelectedItemChangedTrigger.SelectionMappings>
```

Různé AppBar pro odlišné stavy stránky

Uvažujme stránku s ovládacím prvkem LongListMultiSelector zobrazující seznam položek s možností výběru. Ve standardním režimu se bude stránka nacházet v případě, že nebude vybrána žádná položka a naopak v editačním režimu za předpokladu minimálně jedné vybrané položky. Pro každý režim budeme chtít zobrazit rozdílné AppBar, k čemuž lze využít StateChangedTrigger, který prostřednictvím atributu State umožňuje přepínání definovaných AppBar. [18]

```
1 <appBar:StateChangedTrigger State="{Binding Path=IsSelecting}">
2     <appBar:SwitchAppBarAction>
3         <appBar:AppBar Id="0">...</AppBar>
4         <appBar:AppBar Id="1">...</AppBar>
5     </AppBar:SwitchAppBarAction>
6 </AppBar:StateChangedTrigger>
```

5 LIVE SDK

Live SDK nabízí prostředky pro integrování služeb spojených s Microsoft účtem, tedy Outlook.com a SkyDrive. SDK je dostupné pro všechny tři nejrozšířenější mobilní platformy, tedy Windows Phone, iOS a Android. [19]

5.1 Client ID

Každá aplikace, která chce využívat Live Connect Api, musí být zaregistrována (manage.dev.live.com) a mít přiřazen jedinečný identifikátor. Pro správnou funkčnost je navíc potřeba v nastavení zvolit, že se jedná o mobilní klientskou aplikaci. [19]

5.2 Scopes

Scopes jsou množina oprávnění, které vývojář přidělí své aplikaci, a tedy požaduje po uživateli, aby mu k nim umožnil přístup.

Základní

- wl.basic – základní informace o profilu uživatele
- wl.offline_access – přístup k profilu kdykoliv
- wl.signin – pokud se uživatel jednou přihlásí, nemusí při dalším přihlášení zadávat přihlašovací údaje

Některé další

- wl.skydrive – čtení souborů uložených na SkyDrive
- wl.skydrive_update – k předchozímu oprávnění přidává navíc i zápis
- wl.calendars – přístup ke kalendáři a událostem

Kompletní přehled oprávnění i s popisem je k dispozici v dokumentaci [20].

II. PRAKTICKÁ ČÁST

6 KONVERTORY

Konvertory slouží pro úpravu dat ať už z jednoho typu na druhý – nejčastěji z hodnoty boolean na Visibility, nebo pro prosté doplnění stávající hodnoty o další údaje.

Každý konvertor musí implementovat rozhraní IValueConverter a jeho dvě metody Convert a ConvertBack.

Convert modifikuje zdrojová data před odesláním k cíli, tedy před tím, než se zobrazí v uživatelském rozhraní.

ConvertBack zpětně upravuje data z uživatelského rozhraní do původních zdrojových dat a je volána pouze při použití obousměrného provázání.

Obě metody přebírají stejné parametry:

- object value – zdrojová hodnota určená k modifikaci,
- Type targetType – cílový typ proměnné,
- object parameter – nepovinný libovolný parametr,
- CultureInfo culture – informace o regionu.

Abychom mohli konvertory využívat, musíme je zaregistrovat ve zdrojích pohledu, případně aplikace (App.xaml). [21]

Registrace konvertoru:

```
1 xmlns:c="clr-namespace:CarTracking.Converters"  
2 <c:BooleanToVisibilityConverter x:Key="BooleanToVisibility" />
```

UpperCase, LowerCase

Jak už název napovídá, tyto konvertory se starají o převod textu na velká resp. malá písmena. Využívány jsou především u ovládacího prvku Pivot a jeho potomků PivotItem, kde je dobrým zvykem a nepsanou konvencí uvádět Title velkými písmeny a Header malými písmeny.

Realizace těchto konvertorů není nikterak složitá, pro převod velikostí písmen využívají metod z třídy String.

```
1 return value.ToString().ToUpper();  
2 return value.ToString().ToLower();
```

BooleanToVisibility

Převádí boolean hodnoty na výčtový typ `Visibility` a umožňuje tak ovládat viditelnost vizuálních prvků na stránce pomocí navázané vlastnosti typu `boolean`.

Hodnota `True` je tedy převedena na `Visible` a `False` na `Collapsed`.

```
1 return (bool)value ? Visibility.Visible : Visibility.Collapsed;
```

InvertVisibility

Invertuje hodnotu viditelnosti ovládacího prvku. Z neviditelného udělá viditelný a naopak, což se hodí v případě, že potřebujeme přepínat viditelnost mezi dvěma objekty v závislosti na sobě.

```
1 return (Visibility)value == Visibility.Visible ?  
Visibility.Collapsed : Visibility.Visible;
```

CountToVisibility

Jako vstupní parametr se předpokládá počet záznamů např. z `ListBox` resp. podobného prvku umožňujícího zobrazovat kolekci dat. Konvertor převede počet záznamů na odpovídající viditelnost, tedy v případě, že kolekce nemá žádný záznam, je `ListBox` zneviditelněn.

```
1 return (int)value > 0 ? Visibility.Visible : Visibility.Collapsed;
```

ToLocalTime

U lokalizovaných aplikací je vhodné ukládat datum a čas v univerzálním formátu `UTC` a teprve na základě zvolené oblasti (regionu) na straně uživatele převést na místní čas (zohlednění časového posunu).

Pro zohlednění časového pásma je v konvertoru využita metoda `ToLocalTime` třídy `DateTime`.

```
1 return ((DateTime)value).ToLocalTime();
```

EnumToResource

Umožňuje převést libovolný výčtový typ na odpovídající lokalizovaný řetězec. Kromě vstupní hodnoty je možné využít i parametr, který slouží jako prefix pro danou hodnotu.

Lokalizovaný řetězec je pak získán prostřednictvím ResourceManager.

```
1 string prefix = parameter != null ? (string)parameter : "";
2 return AppResources.ResourceManager.GetString(prefix +
   value.ToString());
```

StringToDouble

Využití nalézá u prvku TextBox, kde přijímáme vstupní data od uživatele ve tvaru řetězce a je tedy nutné je převést na double.

Oproti doposud zmíněným konvertorům, které využívají pouze metodu Convert, je zde využita i druhá metoda ConvertBack, jež se stará právě o konverzi formátů mezi string a double.

```
1 double newValue = 0.0;
2 Double.TryParse(value.ToString(), out newValue);
3 return newValue;
```

ColorToBrush

Na vstupu přijímá výčtový typ AccentColor deklarující název barvy a na výstupu je vrácena odpovídající barva typu SolidColorBrush získaná ze slovníku AccentColors.Colors, ve kterém jsou definovány všechny dostupné barvy.

```
1 return AccentColors.Colors[(AccentColor)value];
```

TypeToStyle

Slouží pro převod výčtového typu Icon a tříd File a Folder (SkyDrive objekty) na odpovídající vizuální styl. Jako parametr se uvádí cesta k souboru, ve kterém se příslušný styl nachází.

```
1 string styleName;
2 string uri = String.Format("/CarTracking;component/Styles/{0}",
   parameter);
```

Načtení zdrojového souboru:

```
3 ResourceDictionary resourceDictionary = new ResourceDictionary();
4 Uri resourceLocator = new Uri(uri, UriKind.Relative);
5 Application.LoadComponent(resourceDictionary, resourceLocator);
```

Rozlišení ikony trasy resp. souboru a složky na SkyDrive:

```
6 if (value.GetType().Name == "Icon") { styleName = value.ToString();}
7 else { styleName = value.GetType().Name; }
```

Ověření existence daného stylu:

```
8 if (resourceDictionary.Contains(styleName)) { return
  resourceDictionary[styleName]; }
```

7 ROZŠÍŘENÍ STÁVAJÍCÍCH TŘÍD

V rámci rozšiřujících metod (Extension Methods) je možné k již existujícím třídám přidat nové metody, aniž bychom museli vytvářet nové odvozené objekty.

Mezi nejznámější rozšiřující vrstvu patří bezesporu LINQ, což je nástroj určený pro manipulaci s různými daty.

Pokud chceme vytvořit rozšiřující metody, musíme je deklarovat jako statické uvnitř statické třídy. Každá taková metoda navíc obsahuje před typem prvního parametru klíčové slovo „this“. [22]

AccentColorsExtensions

Vytváří slovník základních barev telefonu přístupný přes výčtový typ AccentColor.

```
1 public static Dictionary<AccentColor, SolidColorBrush> Colors = new
  Dictionary<AccentColor, SolidColorBrush>()
2 {
3     { AccentColor.Black,    0xFF000000.ToSolidColorBrush() },
4     { AccentColor.Lime,    0xFFA4C400.ToSolidColorBrush() },
5     ...
6 }
```

Kódy barev jsou získány ze souborů instalovaných spolu se SDK (C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v8.0\Design\AccentColors). Jedná se o základní barvy. V závislosti na volbě mobilního operátora mohou být přidány další.

Implementace rozšiřujících metod ToSolidColorBrush a ToColor je převzata z [23].

ColorExtensions

Rozšiřuje třídy SolidColorBrush a Brush o metodu schopnou invertovat původní barvu. Invertování je provedeno na úrovni jednotlivých RGB složek, tedy odečtením originálních hodnot od maximální.

```
1 public static SolidColorBrush InvertColor(this SolidColorBrush c)
2 {
3     byte r = (byte) (255 - c.Color.R);
4     byte g = (byte) (255 - c.Color.G);
5     byte b = (byte) (255 - c.Color.B);
6     return new SolidColorBrush(Color.FromArgb(255, r, g, b));
7 }
```

```
8 public static SolidColorBrush InvertColor(this Brush b)
9 {
10     return InvertColor((SolidColorBrush)b);
11 }
```

DateTimeExtensions

Implementuje rozšíření třídy `DateTime` o metodu schopnou vypočítat datum začátku týdne na základě libovolného data i s ohledem na rozdílný počáteční den (neděle, pondělí...) v některých zemích.

Samotná implementace je s malou modifikací převzata z [24].

```
1 public static DateTime StartOfWeek(this DateTime dt)
2 {
3     int diff = (int)dt.DayOfWeek -
4     (int)CultureInfo.CurrentCulture.DateTimeFormat.FirstDayOfWeek;
5     if (diff < 0) { diff += 7; }
6     return dt.AddDays(-1 * diff).Date;
7 }
```

8 LOGOVÁNÍ CHYB

V každé aplikaci je vždy potřeba jistým způsobem zaznamenávat chybové či ladící informace. Někdy a někomu možná postačí výpis pomocí třídy Debug do debugovacího okna Output ve vývojovém prostředí Visual Studio. Ne vždy je to však dostačující.

Jelikož se jedná o aplikaci využívající GPS a pro plnohodnotné testování je nutný pohyb v terénu (jízda autem či hromadnou dopravou), a není tedy moc praktické mít s sebou např. notebook pro sledování zpráv v Output okně, je vhodné se vytvořit vlastní logovací třídu, u které bude možné nastavit, kde se ladící informace mají zobrazit.

Logování do souboru se v případě mobilních aplikací nejeví jako příliš vhodné.

8.1 LogType

Jedná se výčtový typ určující, kam se budou chybové a ladící hlášky vypisovat:

- None – žádný výpis,
- Output – výpis do debugovacího okna Output,
- MessageBox – zobrazení klasického dialogového okna přímo v aplikaci,
- Both – současné užití obou předchozích variant.

8.2 Log

Logovací třída je pro rychlé a účelné použití deklarována jako statická. Obsahuje jedinou vlastnost typu LogType, která je veřejně přístupná a je tedy možné ji nastavit kdekoliv v aplikaci dle potřeby.

Pro výpis je určena metoda Write, v níž se na základě vlastnosti Type určí, kam se zpráva odešle. Aby nedocházelo k zobrazení chybových hlášek v Release verzi aplikace, je obsah metody obalem direktivou #if DEBUG.

Metoda pro výpis do Output okna:

```
1 private static void WriteOutput(Exception e)
2 {
3     Debug.WriteLine(String.Format("{0}: {1} {2}",
4     AppResources.Error, e.Message, e.HResult));
4 }
```

Zobrazení zprávy přímo v aplikaci:

```
5 private static void WriteMessageBox(Exception e)
6 {
7     MessageBox.Show(e.Message, AppResources.Error,
8     MessageBoxButtons.OK);
9 }
```

Veřejná metoda pro logování chybových událostí:

```
9 public static void Write(Exception e)
10 {
11     #if DEBUG
12
13     // none
14     if (Type == LogType.None) { return; }
15
16     // output
17     if (Type == LogType.Console || Type == LogType.Both)
18     {
19         WriteConsole(e);
20     }
21
22     // message box
23     if (Type == LogType.MessageBox || Type == LogType.Both)
24     {
25         WriteMessageBox(e);
26     }
27
28     #endif
29 }
```

9 LOKALIZACE

Aplikace jsou vytvářeny pro uživatele, a čím více uživatelů si naši aplikace stáhne, tím větší úspěch potažmo zisk budeme mít. Musíme se tedy snažit prorazit do celého světa. Ovšem není to pouze o umístění aplikaci na daný trh, nýbrž i o lokalizaci textů do příslušných jazyků.

V aplikaci musíme mít vždy nastaven výchozí tzv. neutrální jazyk a volitelně další jazyky, pro které budeme vytvářet lokalizaci. V tomto případě je jako neutrální jazyk zvolena angličtina a druhým jazykem je čeština.

Ke každému jazyku existuje příslušný soubor (Resources File), jehož název obsahuje kód daného jazyka. U neutrálního jazyka se kód neuvádí. [25]

- angličtina – AppResources.resx
- čeština – AppResources.cs.resx

V rámci těchto souborů jsou definovány jedinečné identifikátory spolu s přeloženým textem a volitelným komentářem.

	Name	Value	Comment
	Account	Účet	
	ActiveTrip	aktivní	
	Add	přidat	
	AllTrips	všechny	
	Altitude	převýšení	

Obr. 12. Soubor s českou lokalizací

LocalizedStrings

Třída LocalizedStrings, která je mj. automaticky generována při vytvoření nového projektu ve Visual Studiu, umožňuje přístup k lokalizovaným řetězcům jak v kódu, tak i v XAML prostřednictvím navázání přes vlastnost LocalizedResources. [26]

```

1 private static AppResources _localizedResources = new
  AppResources();
2 public AppResources LocalizedResources
3 {
4     get { return _localizedResources; }
5 }
```

10 OVLÁDACÍ PRVKY

Windows Phone SDK spolu s rozšířením [23] nabízí celou řadu dobře předpřipravených ovládacích prvků, avšak mnohokrát se dostaneme do situace, kdy potřebujeme vytvořit specifický objekt kombinací již existujících prvků pro vícenásobné použití.

Pro vytvoření nových ovládacích prvků se využívá objekt `UserControl`, do kterého vkládáme další elementy. V případě vytváření specifických formulářů pro zadávání nových a úpravu stávajících záznamů je to ideální způsob, protože kopírování většího množství stejného kódu a jeho následné udržování by v případě změny bylo zbytečně pracné a časově náročné. Navíc u MVVM koncepce je nutné dodržet propojení jednoho pohledu s jedním `ViewModel`, tedy využít stejný pohled jak pro vytváření a zároveň editaci záznamů nepřipadá v úvahu.

10.1 CustomListPicker

Slouží jako imitace originálního `ListPicker` [23] a je vytvořen pomocí obyčejného vzhledově upraveného tlačítka `Button`. Visuální chování tlačítka je upraveno přes `VisualStateManager`. Konkrétněji je přizpůsoben stav stisknuto (`Pressed`), v němž se mění barva ohraničení tlačítka a jeho pozadí resp. popředí tak, aby uživatel nabyl dojmu, že bude přesměrován na pohled s výběrem dalších objektů, stejně tak, jako je tomu u originálního `ListPicker`.

U `ListPicker` je sice možné vytvořit a nastavit pomocí atributu `PickerPageUri` alternativní stránku s výběrem jednotlivých položek, ale pro určité specifické užití to příliš vhodné není.

V rámci `CustomListPicker` jsou implementovány dvě `DependencyProperty` pro možnost využití data bindingu:

- `ICommand Command` – příkaz vykonaný při stisknutí tlačítka,
- `string Text` – zobrazený text.

10.2 CustomTextBlock

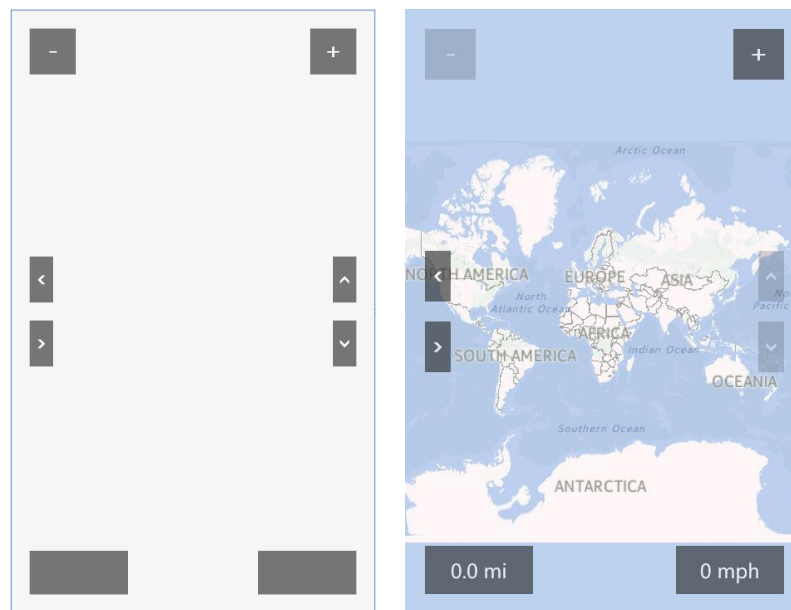
Ke klasickému `TextBlock` prvku pro zobrazení textu a popisků přidává navíc barevné pozadí v závislosti na nastavení telefonu. Jelikož `TextBlock` nedisponuje vlastností `Background`, je obalen prvkem `Grid`, který má tento atribut nastaven na `PhoneChromeBrush`.

Pomocí `DependencyProperty` jsou vytvořeny dvě vlastnosti. Jedna pro nastavení textového výstupu (`string`) a druhá pro možnost zarovnání textu (`TextAlignment`).

10.3 MapControls

Jedná se o speciální vrstvu ovládacích tlačítek (Obr. 13) určenou pro manipulaci s mapou. Tlačítka pro přiblížení, oddálení a natočení mapy jsou vytvořena pomocí RepeatButton, stačí tedy držet prst na tlačítku a navázaný příkaz se vyvolává opakovaně. Veškeré příkazy jsou vytvořeny jako DependencyProperty a stejně tak jsou vytvořeny i vlastnosti pro ovládání viditelnosti jednotlivých tlačítek.

Efekt překrytí je docílen obalením všech tlačítek elementem Grid.



Obr. 13. Vrstva ovládacích tlačítek mapy

10.4 Stats

Stats je prvek určený pro jednoduché zobrazení statistických údajů. V horní části (Obr. 14) vyniká název události (TextBlock), v tomto případě název měsíce. Dále následuje horizontální ProgressBar pro procentní zobrazení vybraného údaje a pod ním je oblast Grid, rozdělená přes ColumnDefinitions na dva sloupce, pro zobrazení dalších informací.



Obr. 14. Ovládací prvek Stats

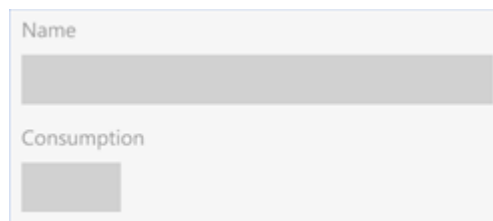
10.5 Formuláře

Jak už bylo zmíněno, v pohledech pro vytváření a úpravu záznamů je třeba využít stejného seskupení ovládacích prvků, a proto jsou vytvořeny zvlášť, jako samostatné fungující jednotky.

CarForm

Formulář pro manipulaci s vozidlem (Obr. 15), v němž je možné zadat název a spotřebu auta. TextBox pro nastavení spotřeby má definován `InputScope` [27] na `Number` o délce 5 znaků a je na něj aplikován konvertor `StringToDouble`. Dále je u něj využito formátování `StringFormat (F1)` v rámci `data binding`.

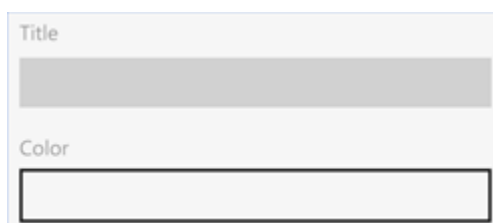
```
1 <TextBox
2     InputScope="Number"
3     MaxLength="5"
4     Text="{Binding Path=Consumption, ElementName=control,
Mode=TwoWay, StringFormat=F1, Converter={StaticResource
StringToDouble}}" />
```

The image shows a screenshot of a user interface control. It contains two text input fields. The first field is labeled 'Name' and is empty. The second field is labeled 'Consumption' and is also empty. The control has a light gray background and a thin blue border.

Obr. 15. Formulář *CarForm*

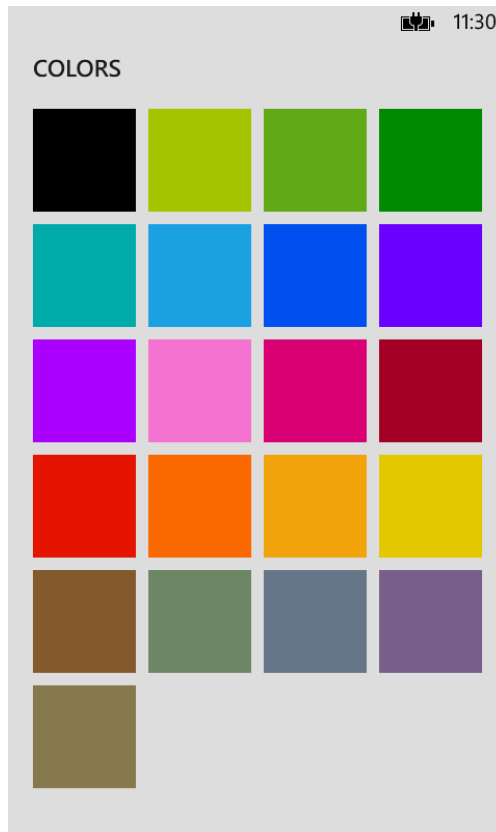
PinForm

Editací panel pro práci se špendlíky. Název je omezen pomocí `MaxLength` na 40 znaků. Výběr barvy je udělán přes `ListPicker` a jelikož barev je více, tak je zapnut `ExpansionMode` na `FullScreenOnly`, tedy barva je vybírána na nové stránce (Obr. 17).

The image shows a screenshot of a user interface control. It contains two text input fields. The first field is labeled 'Title' and is empty. The second field is labeled 'Color' and is empty. The control has a light gray background and a thin blue border.

Obr. 16. Formulář *PinForm*

Aby bylo možné zobrazit barvy za sebou (Obr. 17), bylo nutné v původním zdrojovém kódu modifikovat `ItemsPanelTemplate` za pomoci `WrapPanel`, který toto chování umožňuje.



Obr. 17. Výběr barvy špendlíku

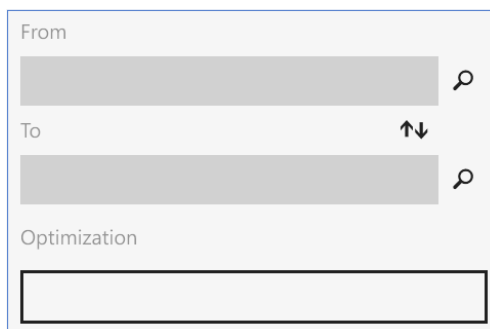
NavigationForm

Navigační formulář (Obr. 18) pro tvorbu a plánování cest obsahuje TextBox prvky pro zadání odkud kam trasa povede a ListPicker pro výběr optimalizace cesty.

Tlačítka pro vyhledávání a prohození obou lokací jsou vytvořena pomocí klasického Button s tím, že mají implementován ContentTemplate jako TextBlock, v němž je místo textu využito symbolů z písma Segoe UI Symbol.

Všechny použitelné symboly si lze prohlédnout pomocí aplikace Mapa znaků přímo ve Windows. U každého symbolu je vždy uveden příslušný unicode znak, který je pro něj charakteristický.

```
1 <TextBlock
2     Style="{StaticResource Symbol}"
3     Text="&#xE11A;" />
```

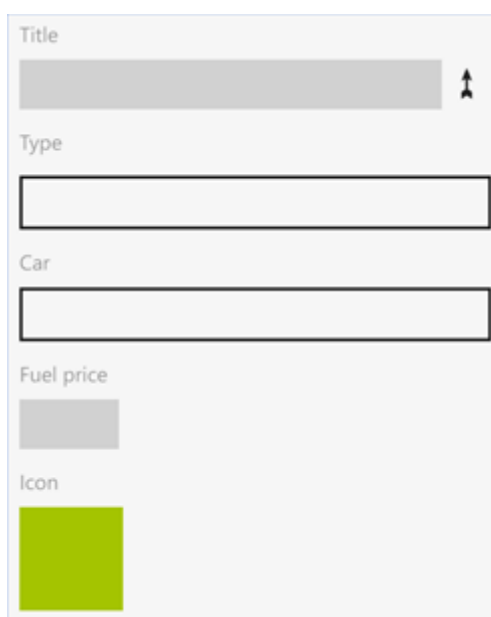


Obr. 18. Formulář *NavigationForm*

TripForm

Jde o poslední typ z rodiny formulářů (Obr. 19) určený pro tvorbu cesty.

Pro zadání názvu cesty je připraven `TextBox`, vedle něhož se nachází tlačítko odkazující na pohled s výběrem navigací. Visuální stránka tlačítka je upravena přes `ContentTemplate` stejným způsobem, jako tomu je u tlačítek v `NavigationForm`.



Obr. 19. Formulář *TripForm*

Typ cesty se vybírá přes `ListPicker` s povoleným `ExpansionMode`. Naproti tomu pro výběr auta je využit výše popsáný `CustomListPicker`, protože odkazuje na jiný specifický pohled pro výběr vozidel.

`TextBox` pro nastavení ceny paliva má, stejně jako v případě spotřeby auta u `CarForm`, nastaven `InputScope` [27] jako `Number` a je aplikováno omezení na 6 znaků. I zde je navíc navázán konvertor `StringToDouble`.

Oblast ikony tvoří TextBlock s nabídnutým atributem Style, na základě kterého se pomocí konvertoru TypeToStyle zobrazí příslušná ikona. V parametru konvertoru se zde předává cesta k souboru s definicí všech ikon.

```
1 <TextBlock
2     Style="{Binding Path=Icon, ElementName=control,
3     Converter={StaticResource TypeToStyle},
4     ConverterParameter=IconPatterns.xaml}" />
```

11 PŘEVODNÍKY JEDNOTEK

Vzhledem k tomu, že je aplikace lokalizována a ne všude se využívají metrické jednotky tak jako u nás, bylo zapotřebí nějakým způsobem vyřešit konverzi měrných systémů.

Výčtový typ `Unit` definuje základní rozdělení jednotek na metrické a imperiální. Imperiální jsou dvojího druhu, jednak britské, užívané ve Velké Británii a přilehlých zemích a také americké. Rozdílů je hned několik, ovšem v rámci této aplikace je nutné zmínit pouze odlišnou velikost objemové jednotky galon (viz dále).

```
1 enum Unit { Metric, ImperialUK, ImperialUS };
```

Všechny třídy pro převod jednotek implementují generické rozhraní `IUnitConverter<T>` definující převodní tabulku jako dvourozměrné pole a tyto metody:

- `double Convert(double value, T from, T to),`
- `double ConvertFromMetric(double value, Unit to),`
- `double ConvertToMetric(double value, Unit from),`
- `T GetUnit(Unit unit),`
- `string GetUnitName(Unit unit).`

Pro lepší a jednodušší práci s převodníky jednotek jsou vytvořeny následující výčtové typy definující použitelné jednotky.

```
1 enum LengthUnit { M, Km, Mi };  
2 enum SpeedUnit { Mps, Kph, Mph };  
3 enum VolumeUnit { L, ImpGal, UsGal };
```

11.1 Délka, rychlost a objem

Převod jednotek délky zastřešuje třída `LengthConverter` v rámci níž jsou deklarovány konstanty `Feet` (stopa) a `Mile` (míle) v metrech pro použití v převodní tabulce (Tab. 4).

```
1 const double Feet = 0.3048;  
2 const double Mile = 5280 * Feet;
```

Pro konverzi jednotek slouží metoda `Convert` v níž je využito přetypování výčtových typů na jejich odpovídající celočíselné indexy, a tak je možné se odkazovat do převodní tabulky.

```
1 double Convert(double value, LengthUnit from, LengthUnit to)  
2 {  
3     return value * this.ConvertTable[(int)from][(int)to];  
4 }
```

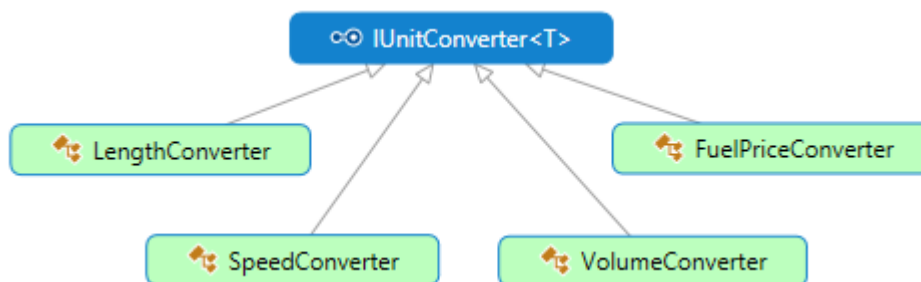
Metody ConvertFromMetric a ConvertToMetric mají zafixovanou zdrojovou resp. cílovou jednotku na metrický systém.

Tab. 4. Převodní tabulka délky

	m	km	mi
m	1	0,001	$\frac{1}{\text{Mile}}$
km	1000	1	$\frac{1000}{\text{Mile}}$
mi	Mile	$\frac{\text{Mile}}{1000}$	1

Naprosto stejným způsobem jsou implementovány i třídy SpeedConverter a VolumeConverter, jenž má navíc jako konstanty definovány hodnoty galonů v litrech.

```
1 const double ImpGallon = 4.54609;
2 const double UsGallon = 3.785411784;
```



Obr. 20. Třídy implementující rozhraní IUnitConverter

Tab. 5. Převodní tabulka rychlosti

	m/s	km/h	mph
m/s	1	3,6	$\frac{3600}{\text{Mile}}$
km/h	$\frac{1}{3,6}$	1	$\frac{1000}{\text{Mile}}$
mph	$\frac{\text{Mile}}{3600}$	$\frac{\text{Mile}}{1000}$	1

Tab. 6. Převodní tabulka objemu

	litr	Imp. galon	US galon
litr	1	$\frac{1}{\text{ImpGallon}}$	$\frac{1}{\text{UsGallon}}$
Imp. galon	<i>ImpGallon</i>	1	$\frac{\text{ImpGallon}}{\text{UsGallon}}$
US galon	<i>UsGallon</i>	$\frac{\text{UsGallon}}{\text{ImpGallon}}$	1

11.2 Cena paliva

Cena paliva se vždy uvádí v určité měně za jednotku objemu ať už je to Kč/l, nebo \$/gal.

V rámci FuelPriceConverter je v převodu zohledněna pouze objemová část, jelikož měny jako takové jsou variabilní. Sice by bylo možné stáhnout aktuální měnové kurzy z internetu, ale pro potřeby této aplikace to však není nutné.

```

1 public double Convert(double value, VolumeUnit from, VolumeUnit to)
2 {
3     return value * this.volume.Convert(1, from, to);
4 }

```

Symbol měny je získán v závislosti na nastavení telefonu:

```

5 RegionInfo.CurrentRegion.CurrencySymbol

```

11.3 Spotřeba auta

Spotřeba auta se běžně uvádí v l/100 km resp. mpg, z čehož vyplývá jakási neregularita obou údajů a je tedy potřeba provést v rámci výpočtu příslušnou úpravu.

```

1 public double Calculate(double distance, double carConsumption, Unit
  unit)
2 {
3     double c;
4     LengthUnit lengthTo;
5
6     if (unit == Unit.Metric)
7     {
8         c = carConsumption / 100;
9         lengthTo = LengthUnit.Km;
10    }

```

```
11     else
12     {
13         c = 1 / this.ConvertFromMetric(carConsumption, unit);
14         lengthTo = LengthUnit.Mi;
15     }
16
17     return c * this.length.Convert(distance, LengthUnit.M, lengthTo);
18 }
```

11.4 Objem dat

StorageUnit je statická třída, která skrz metodu Format dokáže převést hodnotu udávající např. velikost souboru na její odpovídající násobek včetně příslušné jednotky. Rozsah výstupních hodnot se může pohybovat v řádech B až TiB.

```
1 public static string Format(long sizeByte)
2 {
3     ...
4     // KiB
5     else if (sizeByte < Math.Pow(2, 20))
6     {
7         unit = AppResources.KiByte;
8         size = sizeByte / Math.Pow(2, 10);
9     }
10    ...
11    return String.Format("{0:N1} {1}", size, unit);
12 }
```

12 SLUŽBY

12.1 Navigace

Navigační služba `NavigationService` je vytvořena pro potřeby přecházení mezi jednotlivými pohledy v rámci aplikace. Implementuje rozhraní `INavigationService`.

```
1 event NavigatingCancelEventHandler navigating;
2 void GoBack();
3 void NavigateTo(Uri pageUri, bool removeFromJournal);
4 void NavigateTo(string pageUrl, bool removeFromJournal = false);
```

Řešení je převzato z [29] s malou modifikací. Modifikace spočívá v úpravě metod `NavigateTo`, které kromě cílové stránky přebírají navíc parametr, jenž umožňuje odstranit ze zásobníku stránek předchozí navštívenou.

```
1 public void NavigateTo(Uri pageUri, bool removeFromJournal = false)
2 {
3     if (this.EnsureMainFrame())
4     {
5         this.mainFrame.Navigate(pageUri);
6
7         // remove from journal
8         if (removeFromJournal) { this.mainFrame.RemoveBackEntry(); }
9     }
10 }
```

Metoda `NavigateTo` je přetížená a můžeme jí předávat prosté URL jako řetězec, nebo přímo instanci `Uri`.

12.2 Nastavení

Pro permanentní uchování specifických nastavení dle voleb uživatele i po ukončení aplikace je nutné využít `IsolatedStorageSettings` [28][30]. Uživatelské volby jsou uchovávány ve slovníku `Dictionary<TKey, TValue>` a pro komfortnější přístup a manipulaci s těmito volbami je vytvořena třída `SettingsService`. Každé individuální nastavení má ve třídě vytvořenu vlastnost a výchozí hodnotu definovanou jako konstantu.

```
1 public const MapLink MapLinkDefault = MapLink.Nokia;
2 public MapLink MapLink
3 {
4     get { return this.GetValue("MapLink", MapLinkDefault); }
5     set { this.UpdateValue("MapLink", value); }
6 }
```

Do `IsolatedStorageSettings` se přistupuje přes metodu `GetValue` vracející uživatelem zvolenou hodnotu. Pokud uživatel dosud nastavení neměnil, je vrácena výchozí hodnota.

```
7 public T GetValue<T>(string key, T defaultValue)
8 {
9     if (this.storage.Contains(key)) { return (T)this.storage[key]; }
10    else { return defaultValue; }
11 }
```

Ukládání nových resp. změna stávajících nastavení probíhá přes metodu `UpdateValue`, v níž se nejdříve kontroluje existence příslušné hodnoty a pokud existuje, je aktualizována. V opačném případě je do slovníku vložena nová dvojice klíč + hodnota.

Nastavení se ukládá automaticky ihned při změně zavoláním metody `Save` nad instancí `IsolatedStorageSettings`. Není tedy potřeba vytvářet zvlášť tlačítko pro potvrzení uložení.

```
1 public void UpdateValue(string key, object value)
2 {
3     if (this.storage.Contains(key))
4     {
5         if (this.storage[key] != value)
6         {
7             this.storage[key] = value;
8             this.storage.Save();
9         }
10    }
11    else
12    {
13        this.storage.Add(key, value);
14        this.storage.Save();
15    }
16 }
```

12.3 SkyDrive

SkyDriveService nabízí nástroje pro jednodušší přístup k vlastnostem objektu LiveConnectClient. Služba mj. implementuje ISkyDriveService a definuje dvě důležité vlastnosti:

- string ClientId – obsahuje unikátní řetězec identifikující tuto aplikaci,
- bool IsConnected – indikuje, zdali je uživatel přihlášen ke službě otestováním členské proměnné typu LiveConnectClient.

Přihlášení resp. odhlášení ze služby se děje v rámci metod SignIn a SignOut, což ovšem není tak úplně pravda, protože tyto akce jsou implementovány automaticky v rámci přihlašovacího „live“ tlačítka a metody jsou vyvolány v reakci na něj.

```
1 public void SignIn(LiveConnectSession session)
2 {
3     this.client = new LiveConnectClient(session);
4 }
5 public void SignOut()
6 {
7     this.client = null;
8 }
```

Pro získání informací o profilu uživatele a stavu úložiště jsou určeny metody GetMeAsync a GetQuotaAsync. Obě jsou vytvořeny asynchronně a vrací údaje ve formátu slovníku.

Práci s adresáři z pohledu vytváření a mazání zajišťují následující metody:

- CreateFolderAsync(string parentId, string name),
- DeleteObjectAsync(string id).

Pokud chceme vytvořit novou složku, je třeba zadat identifikátor nadřazeného adresáře a název nové složky. Metodu pro mazání je možné využít jak pro adresáře, tak i pro soubory, stačí uvést příslušný identifikátor daného objektu.

Zacházení se soubory upravují metody pro stahování resp. nahrávání na úložiště:

- DownloadFileAsync – skrz parametr „content“ stahuje obsah souboru,
- UploadFileAsync – je pouze prostředníkem pro zavolání UploadAsync nad LiveConnectClient.

12.4 Databáze

Pro ukládání relačních dat jsou určeny databáze a ne jinak je tomu v případě Windows Phone. V rámci aplikace je možné využít lokální databáze uložené v místním adresáři. Pro manipulaci s databází se využívá vrstva LINQ to SQL. [28][31]

Databázi prezentuje třída DataContext, od které je odvozena DatabaseService, v rámci níž je definován připojovací řetězec obsahující umístění a název databáze. [32]

```
1 public const string ConnectionString = "Data Source=isostore://" +
    DatabaseName;
2 public const string DatabaseName = "CarTracking.sdf";
```

Data jsou ukládána do tabulek deklarovaných následovně:

```
3 Table<Car> Cars;
4 Table<Navigation> Navigation; // od verze 1.1.0.0
5 Table<NavigationWaypoint> NavigationWaypoints; // od verze 1.1.0.0
6 Table<Pin> Pins;
7 Table<Trip> Trips;
8 Table<Waypoint> Waypoints;
```

Jelikož každá aplikace prochází postupným vývojem a ne jinak tomu bylo v tomto případě, byla prvně publikována aplikace bez možnosti plánování cest (verze 1.0.0.0), tedy bez tabulek Navigation a NavigationWaypoints. Tato funkce přibyla až ve verzi 1.1.0.0.

Z tohoto důvodu je třeba kontrolovat verzi databáze při spuštění aplikace a starou verzi aktualizovat na novou prostřednictvím DatabaseSchemaUpdater.

```
9 if (version == 0)
10 {
11     schemaUpdater.AddTable<Navigation>();
12     schemaUpdater.AddTable<NavigationWaypoint>();
13
14     schemaUpdater.DatabaseSchemaVersion = 1;
15     schemaUpdater.Execute();
16 }
```

Při tvorbě databáze jsou rovněž vytvořeny výchozí špendlíky, které jsou automaticky využívány aplikací (Žádný, Start a Cíl). Jejich pojmenování je závislé na nastaveném jazyku telefonu.

Dále jsou definovány specifické metody pro přístup k datům. Za zmínku stojí především GetTripsAsync, jež vrací jednotlivé cesty seskupené dle roku a měsíce vytvoření.

```
17 public Task<IList<TripsGroup<Trip>>> GetTripsAsync()  
18 {  
19     ...  
20     IList<TripsGroup<Trip>> tripsAll = this.Trips  
21         .GroupBy(t =>  
22             new DateTime(t.Created.Date.Year, t.Created.Date.Month, 1))  
23         .OrderByDescending(t => t.Key)  
24         .Select(t =>  
25             new TripsGroup<Model.Trip>(t.Key, t.OrderByDescending(o =>  
26                 o.Created)))  
27         .ToList();  
28     ...  
29 }
```

V poslední řadě je třeba ještě zmínit metodu `GetStatsAsync` určenou pro výběr statistických údajů jednotlivých tras.

```
30 public Task<IQueryable<Stats>> GetStatsAsync(DateTime dtFrom,  
31     DateTime dtTo)  
32 {  
33     ...  
34     IQueryable<Stats> stats = this.Waypoints  
35         .Where(w => w.AbsoluteTime.Date >= dtFrom.ToUniversalTime() &&  
36             w.AbsoluteTime.Date <= dtTo.ToUniversalTime())  
37         .GroupBy(w => w.tripId)  
38         .Select(t => new Stats  
39             {  
40                 Distance = t.Max(d => d.Distance) - t.Min(d => d.Distance),  
41                 ElapsedTime = t.Max(d => d.ElapsedSeconds) - t.Min(d =>  
42                     d.ElapsedSeconds),  
43                 FuelPrice = t.Select(d => d.Trip.FuelPrice).First(),  
44                 CarConsumption = t.Select(d =>  
45                     d.Trip.Car.Consumption).First()  
46             });  
47     ...  
48 }
```

13 MODELY

Modely slouží pro popis jednotlivých objektů použitých v aplikaci. Třídy a jejich vlastnosti jsou doplněny o atributy Table a Column pro definici databázových tabulek.

13.1 Cesta

Trip

Popisuje cestu vytvářenou uživatelem včetně všech vlastností a nastavení. Z pohledu databáze je zde propojení přes cizí klíč na auto skrz vlastnost Car. V rámci třídy jsou dále definovány konstanty pro vymezení přípustného rozsahu ceny paliva (1,0 – 100,0).

Kromě základních objektů obsahuje i dvě kolekce pro uchování waypointů a špendlíků:

- `IList<Waypoint> Waypoints`,
- `IList<Waypoint> Pins`.

Kompletní přehled vlastností je zobrazen v diagramu tříd (Obr. 21).

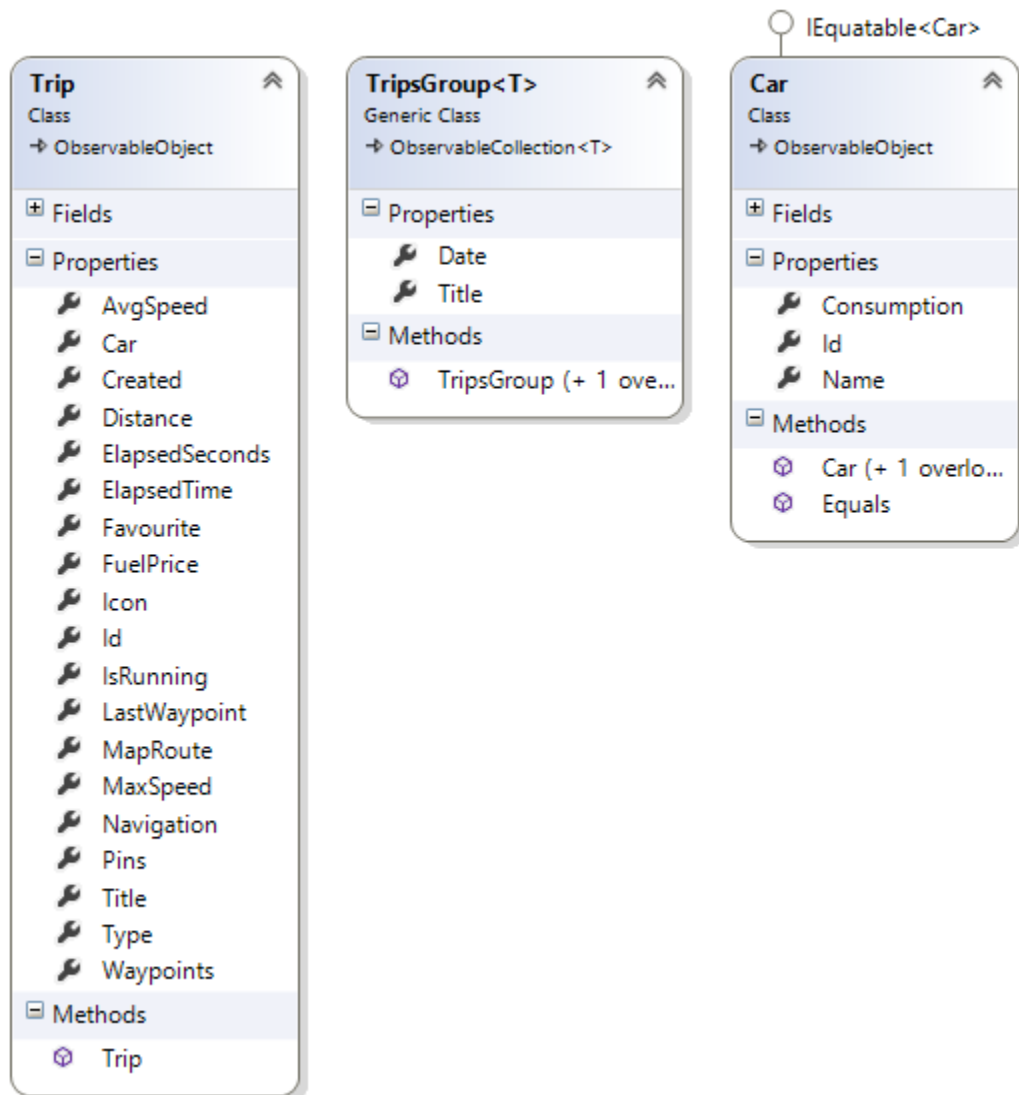
`TripsGroup<T>`

Jedná se o generickou třídu odvozenou od `ObservableCollection<T>` seskupující jednotlivé cesty do skupin dle roku a měsíce. Přetížený konstruktor slouží pro nastavení data a jednotlivých položek.

```
1 public TripsGroup(DateTime date, IEnumerable<T> items) : base(items)
2 {
3     this.Date = date;
4 }
```

Název skupiny tvoří specificky naformátované datum ve vlastnosti Title.

```
5 public string Title
6 {
7     get { return this.Date.ToLocalTime().ToString("MMMM, yyyy"); }
8 }
```



Obr. 21. Implementace tříd `Trip`, `TripsGroup<T>` a `Car`

13.2 Auto

Každé auto je definováno unikátním názvem a spotřebou, která se může pohybovat jen v určených mezích daných konstantami (0,1 – 500,0). Pro zachování konzistentnosti dat v databázi, je nad sloupcem udávajícím název vytvořen unikátní index. Třída rovněž implementuje rozhraní `IEquatable<Car>` a jeho metodu `Equals(Car)` z důvodu vyhledání možných duplicit v rámci kolekce všech aut (Obr. 21).

13.3 Mapa

Pro zobrazení určitých souřadnic na mapě v rámci trasy slouží třída `Waypoint`, která může být doplněna uživatelem či automaticky o špendlík nesoucí specifickou informaci o daném místě.

Waypoint

Vzhledem k tomu, že Waypoint již dědí třídu GeoCoordinate nutnou pro popsání souřadnic v mapě, není tedy možné pro notifikaci změn vlastností využít ObservableObject, a proto bylo nutné zvlášť implementovat rozhraní INotifyPropertyChanged.

I díky tomu, že jsou souřadnice waypointů ukládány do databáze, bylo nutné zděděné vlastnosti z GeoCoordinate přepsat pomocí klíčového slova „new“ a uvést atribut Column.

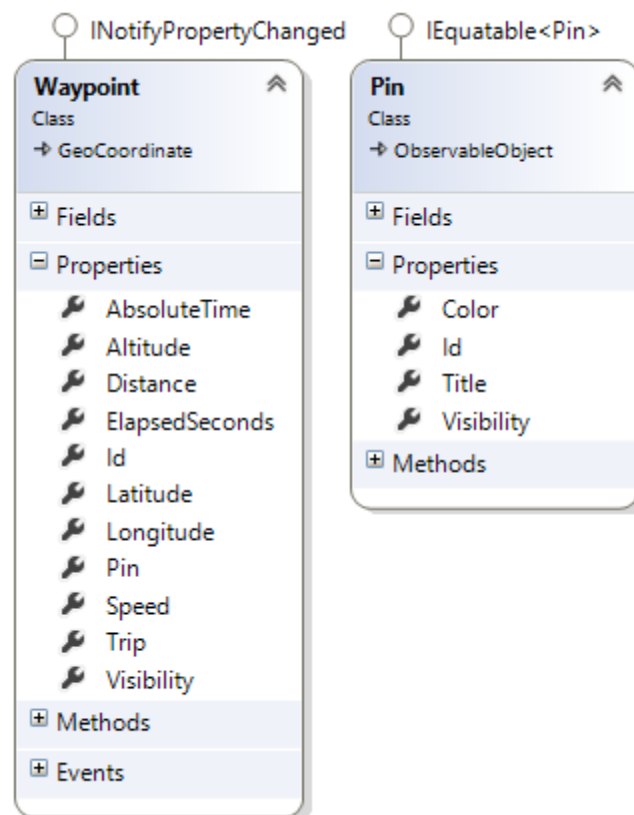
```

1 [Column]
2 public new double Latitude
3 {
4     get { return base.Latitude; }
5     set { base.Latitude = value; }
6 }

```

Všechny Waypointy jsou navíc v rámci databáze spojeny pomocí cizích klíčů s příslušným záznamem v tabulce tras a špendlíků.

Souřadnicový systém GeoCoordinate využívá standard WGS 84.



Obr. 22. Implementace tříd Waypoint a Pin

Pin

Jak už bylo zmíněno, waypointy mohou být doplněny špendlíky, které jsou jednoznačně definovány jedinečným názvem. Unikátnost je zajištěna unikátním klíčem připojeným ke sloupci v tabulce špendlíků. Navíc pro kontrolu duplicit je implementována metoda `Equals(Pin pin)` z rozhraní `IEquatable<Pin>`.

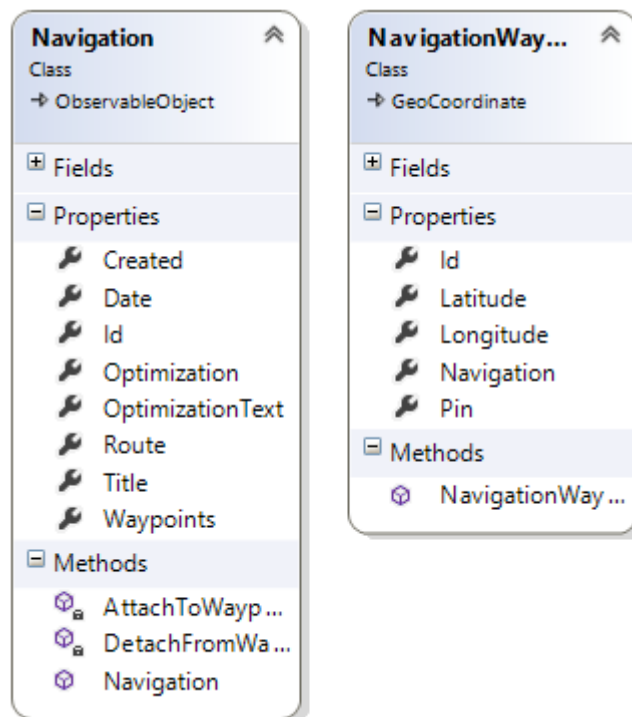
Kromě pojmenování lze špendlík navíc doplnit o určitou barvu z výčtového typu `AccentColor`.

13.4 Navigace

Navigation

Mezi základní vlastnosti navigace patří název (`Title`), jenž nemusí být unikátní, protože se vytváří na základě zdrojové a cílové destinace. Dále je třeba uvést optimalizaci cesty (`Optimization`), v rámci níž je možné volit mezi alternativami rychlejší či kratší cesty.

Ke každé naplánované trase jsou navíc přiřazeny příslušné navigační body pomocí `EntitySet<NavigationWaypoint>`, jenž má implementovány metody volané při přidání (`AttachToWaypoint`) a odebrání (`DetachToWaypoint`) waypointů pro přiřazení navigace ke konkrétnímu bodu.



Obr. 23. Implementace navigace

NavigationWaypoint

Oproti předchozí popsané třídě Waypoint obsahuje podstatně méně informací, protože skutečně stačí evidovat jen souřadnice a případně textový popis pro zobrazení dočasného špendlíku na mapě. Stejně jako Waypoint dědí od třídy GeoCoordinate a přepisuje její vlastnosti udávající souřadnice. Pomocí cizího klíče je navíc v rámci databáze propojena s příslušnou navigací (Navigation).

13.5 Statistiky

Statistické údaje jsou popsány třídami CurrentStats, MonthStats a YearStats (Obr. 24), které implementují abstraktní třídu Stats.

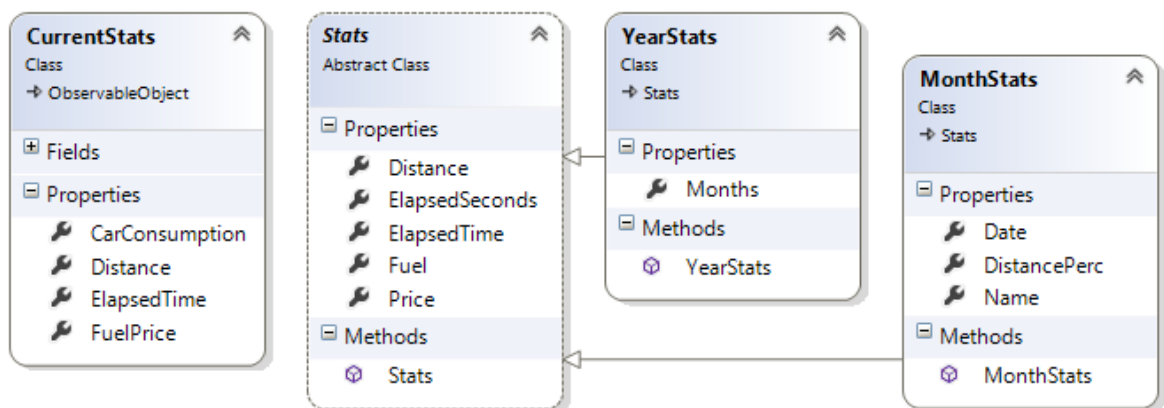
CurrentStats je využita pro popis aktuálních statistik daného měsíce (dnes, týden, měsíc).

YearStats je dále rozšířena o kolekci `IList<MonthStats>` určenou pro měsíční statistiky, u nichž je třeba zmínit vlastnost `Name` vracející naformátované datum na název daného měsíce.

```

1 public string Name
2 {
3     get { return this.Date.ToString("MMMM"); }
4 }

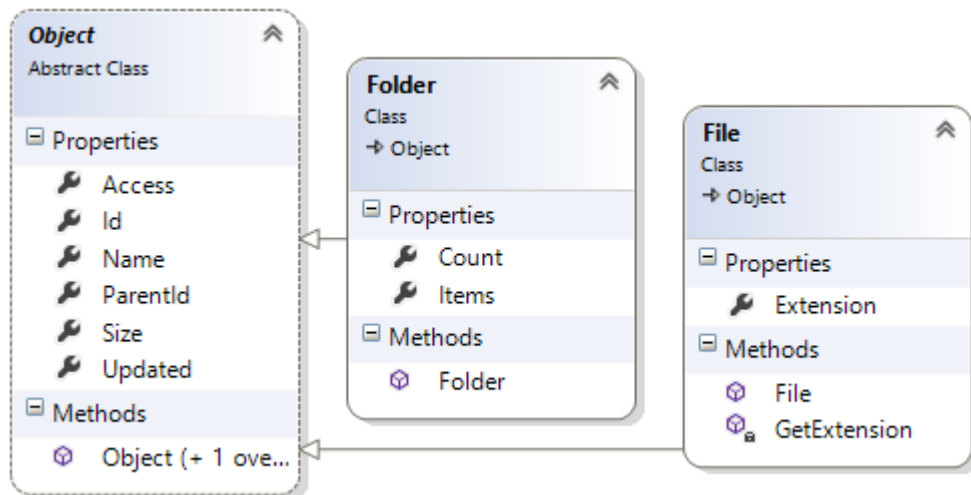
```



Obr. 24. Implementace statistik

13.6 SkyDrive

Společné vlastnosti složek a souborů jsou popsány v rámci abstraktní třídy Object (Obr. 25), od které jsou odvozeny již specifické objekty Folder a File.



Obr. 25. Hierarchie SkyDrive objektů

Folder

Složka má navíc definovanou kolekci objektů `ICollection<Object>` a může tedy obsahovat jak další adresáře, tak i soubory. Vlastnost `Count` udává počet všech položek uvnitř složky, na rozdíl od `Items.Count`, protože v této kolekci se mohou nacházet jen určité typy souborů vyfiltrované např. dle přípony.

File

Soubor je rozšířen o vlastnost `Extension` a metodu `GetExtension`, v rámci níž se přípona odfiltruje z názvu souboru.

```

1 private void GetExtension()
2 {
3     int index = this.Name.LastIndexOf(".") + 1;
4     this.Extension = this.Name.Substring(index);
5 }
  
```

13.7 XML

Writer

Základem všech tříd pro vytváření XML souborů je třída `Writer` (Obr. 26). Kromě konvertorů pro převod jednotek dle nastavení uživatele implementuje dvě speciální formátovací metody.

`UtcFormat` převádí datum na univerzální UTC formát:

```

1 protected string UtcFormat(DateTime dateTime)
2 {
3     return dateTime.ToString("yyyy'-'MM'-'dd'T'HH':'mm':'ss'Z'");
4 }

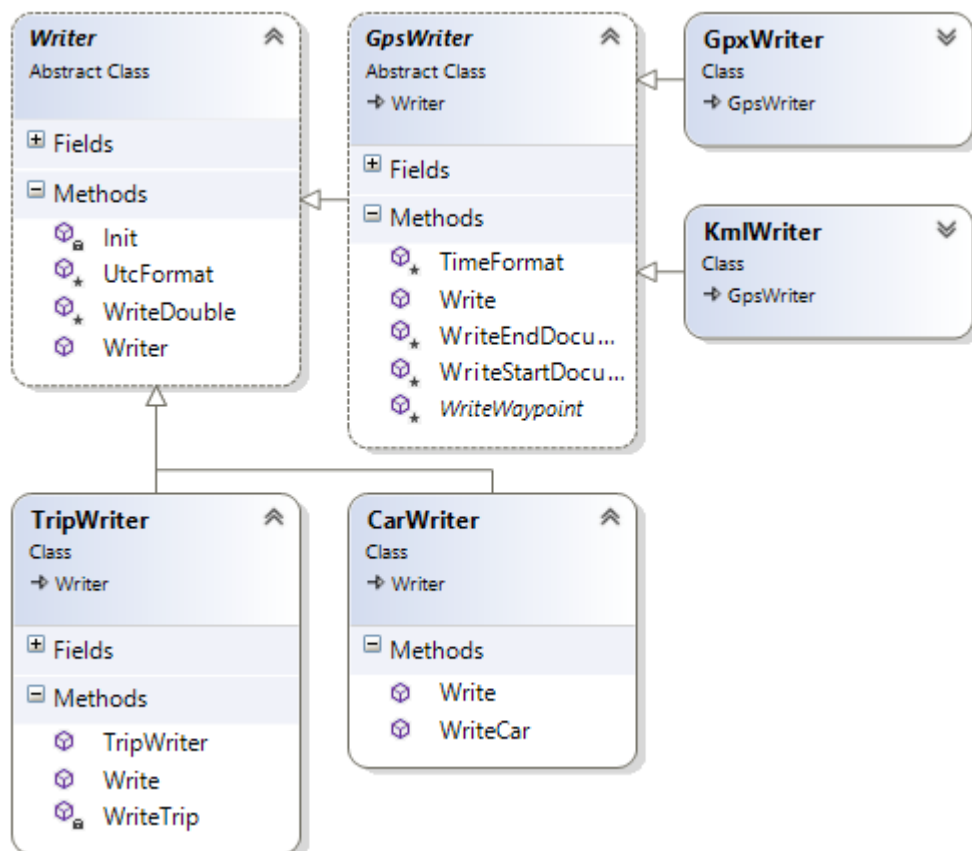
```

`WriteDouble` upravuje desetinná čísla na zvolený formát bez ohledu na region uživatele:

```

5 protected string WriteDouble(double value, string format = "F2")
6 {
7     return value.ToString(format, CultureInfo.InvariantCulture);
8 }

```



Obr. 26. Hierarchie tříd pro vytváření XML souborů

TripWriter

Vytváří XML strukturu včetně všech důležitých údajů popisujících jednotlivé cesty. Užitý jednotkový systém je zohledněn atributem „unit“ (Metric, ImperialUk, ImperialUs) v kořenovém elementu. Pro vytvoření struktury auta je využit CarWriter.

```
1 <trip unit="Metric"> // metrický formát
2   <title>Třebětice - Zlín</title>
3   <created>2013-04-05T07:08:38Z</created> // UTC formát
4   <type>Private</type>
5   <icon>People</icon>
6   <fuelPrice>35.90</fuelPrice> // cena za litr
7   <elapsedTime>00:28:06</elapsedTime>
8   <distance>23.3</distance> // km
9   <maxSpeed>90</maxSpeed> // km/h
10  <avgSpeed>51</avgSpeed> // km/h
11  <car>
12    <name>Honda Civic</name>
13    <consumption>5.9</consumption> // l/100 km
14  </car>
15 </trip>
```

GpsWriter

Abstraktní třída obalující GpxWriter a KmlWriter (Obr. 26) obsahující abstraktní metodu WriteWaypoint a virtuální metody Write, WriteStartDocument a WriteEndDocument.

Formáty GPX a KML jsou široce podporovány mnoha aplikacemi včetně Google Earth, v níž si můžeme průběh cesty bez problémů prohlédnout včetně doplňujících informací.

GpxWriter

Generuje uložené waypointy ve formátu GPX popsaném standardem GPX 1.1. [33]

Pro uložení specifických informací je využit rozšiřující element <extensions>, do kterého jsou umístěny údaje o vzdálenosti, uplynulém čase, rychlosti a volitelně barva špendlíku, jehož název je uveden v tagu <desc>.

```
1 <gpx version="1.1" creator="Car Tracking">
2   <wpt lat="49.23132031" lon="17.65909706">
3     <time>2013-04-18T13:13:45Z</time>
4     <geoidheight>253</geoidheight>
5     <desc>Start</desc>
6     <extensions>
7       <distance>0</distance>
8       <elapsedTime>00:00:00</elapsedTime>
9       <speed>0</speed>
10      <pinColor>Black</pinColor>
11    </extensions>
12  </wpt>
13  ...
14 </gpx>
```

KmlWriter

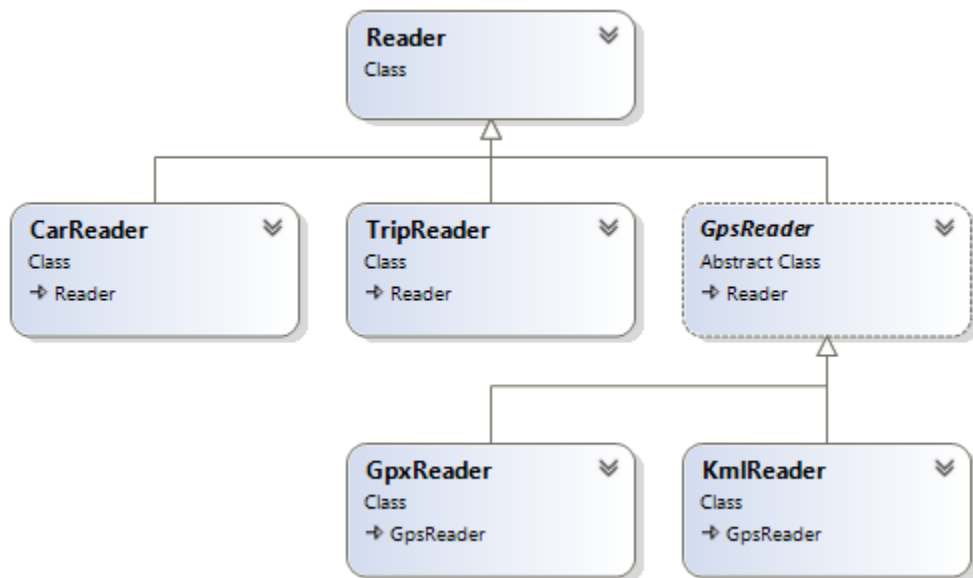
Vytváří soubor waypointů v novějším formátu KML 2.2. [34]

Stejně jako v případě GPX, jsou doplňující informace o waypointu přidány do speciálního elementu, v tomto případě <ExtendedData>.

```
1 <kml xmlns="http://www.opengis.net/kml/2.2">
2   <Folder>
3     <Placemark>
4       <name>Start</name>
5       <Point>
6         <coordinates>17.65909706,49.23132031,253</coordinates>
7       </Point>
8       <TimeStamp>
9         <when>2013-04-18T13:13:45Z</when>
10      </TimeStamp>
11      <description>Vzdálenost = 0 km
12        Uplynulý čas = ${elapsedTime}
13        Rychlost = 0 km/h</description>
14      <ExtendedData>
15        <Data name="distance">
16          <value>0</value>
17        </Data>
18      ...
19 </kml>
```

Reader

Pro parsování souborů se využívá XmlReader u něhož je třeba zmínit přímý necachovaný přístup k XML, na rozdíl od XmlDocument resp. XPathDocument, které nejdříve celé XML natáhnou do paměti. Nevýhodou XmlReader je však trochu pracnější postup při zpracování souboru. [35]



Obr. 27. Hierarchie tříd pro parsování XML souborů

Všechny parsery využívají zděděnou metodu ReadElement, které předávají své vlastní implementované metody pro konkrétní případy a struktury XML. Zpravidla se předávají obě metody (ReadText, ReadStartElement). ReadStartElement slouží pro kontrolu správnosti názvů jednotlivých elementů a ReadText čte jejich obsah.

```

1 ...
2 switch (reader.NodeType)
3 {
4     case XmlNodeType.Element:
5         elementName = reader.Name;
6         if (readStartElement != null) { readStartElement(reader,
7             elementName); }
8         break;
9     case XmlNodeType.Text:
10        if (readText != null) { readText(reader, elementName); }
11        break;
12 }

```

14 IMPLEMENTACE STRÁNEK

14.1 Hlavní části aplikace

App.xaml

Ve zdrojích (Resources) jsou zaregistrovány:

- ViewModelLocator – pro přístup k vlastnostem odkazujících na jednotlivé ViewModel,
- LocalizedStrings – využití lokalizace v XAML,
- všechny konvertory ze jmenného prostoru Converters.

ViewModelLocator

Jedná se o hlavní organizační část aplikace. Seskupuje URL řetězce odkazující na jednotlivé pohledy, což ulehčuje jejich správu v případě změn.

```
1 public const string CarsUrl = "/View/Car/CarsView.xaml";
2 public const string ColorsUrl = "/View/ColorsPinView.xaml";
```

V konstruktoru se registrují pomocí IoC kontejneru všechny ViewModel, k nimž jsou pro spojení s pohledy vytvořeny vlastnosti, vracející odpovídající instanci daného ViewModel.

```
3 SimpleIoc.Default.Register<MainViewModel>();
4 public MainViewModel Main
5 {
6     get { return ServiceLocator.Current.GetInstance<MainViewModel>(); }
7 }
```

ViewModelBaseEx

Rozšiřuje základní ViewModelBase nacházející se v MVVM Light o specifické vlastnosti a metody.

Prostřednictvím metod SetProgressBar a ResetProgressBar umožňuje aktivovat či deaktivovat progress bar aplikace včetně volitelného informačního textu.

Přetížená metoda DialogMessage umožňuje odeslání zprávy pohledu pro zobrazení MessageBox z definovanými údaji.

Všechny následující popisované stránky obsahují ViewModel odvozený právě od této třídy.

14.2 Úvodní stránka

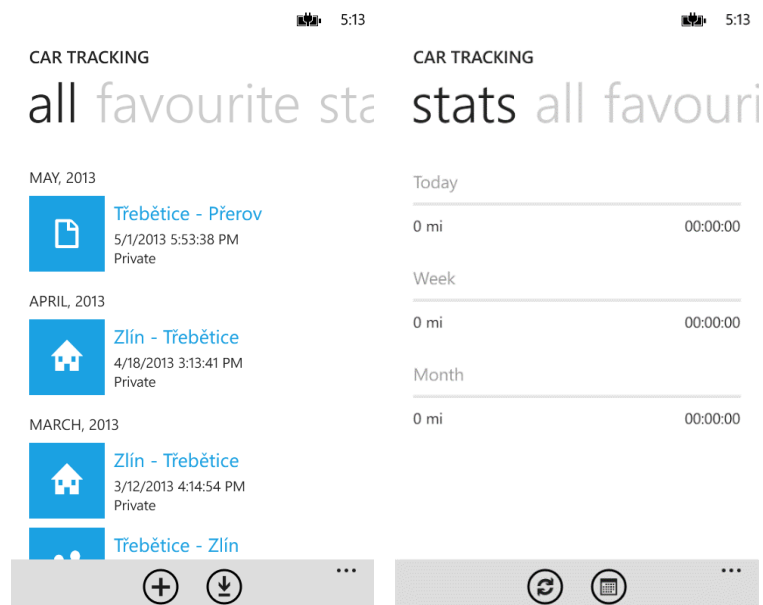
Při načítání úvodní stránky se v události NavigatedTo provádí kontrola, jestli uživatel odsouhlasil informace týkající se ochrany soukromý, pokud ne, je na ně přesměrován (PrivacyPolicyViewModel). Uživatelův souhlas je uložen do IsolatedStorageSettings.

Z pohledu uživatele je nejdůležitější kolekce Groups obsahující jednotlivé trasy seskupené dle roku a měsíce zobrazené v LongListSelector, u něhož bylo třeba aktivovat IsGroupingEnabled a implementovat šablonu GroupHeaderTemplate.

```
1 ObservableCollection<TripsGroup<Model.Trip>> Groups;
```

Metoda AddTripToGroup je určena pro přidání nové trasy do příslušné skupiny. Nejdříve proběhne ověření, jestli odpovídající skupina existuje a pokud ne, je vytvořena.

```
1 ...
2 TripsGroup<Model.Trip> group = this.GetGroup(trip);
3 if (group == null)
4 {
5     group = new TripsGroup<Model.Trip>(new
6         DateTime(trip.Created.Date.Year, trip.Created.Date.Month, 1));
7     this.Groups.Insert(0, group);
8 }
9 group.Insert(0, trip);
10 ...
```



Obr. 28. Úvodní stránka

Pro mazání tras slouží metoda Delete jejíž implementace je obdobná. Prázdná skupina je z kolekce odstraněna.

Seznam oblíbených cest není potřeba nijak seskupovat, a tak jsou jednotlivé trasy uloženy přímo v kolekci.

```
1 ObservableCollection<Model.Trip> FavouriteTrips
```

Zobrazení statistik je vytvořeno přes vlastní ovládací prvek Stats. Zobrazeny jsou denní, týdenní a měsíční statistiky. Hodnoty progress barů jsou nastaveny jako procenta z nadřazených údajů, tedy denní z týdenních a týdenní z měsíčních hodnot.

Ochrana soukromí

Aplikace využívající prostředky pro sledování polohy musí splňovat určitá pravidla, v rámci nichž je požadováno informování uživatele o způsobu využití sledování polohy. Navíc musí být uživateli umožněno deaktivovat volbu zjišťování polohy pro danou aplikaci.

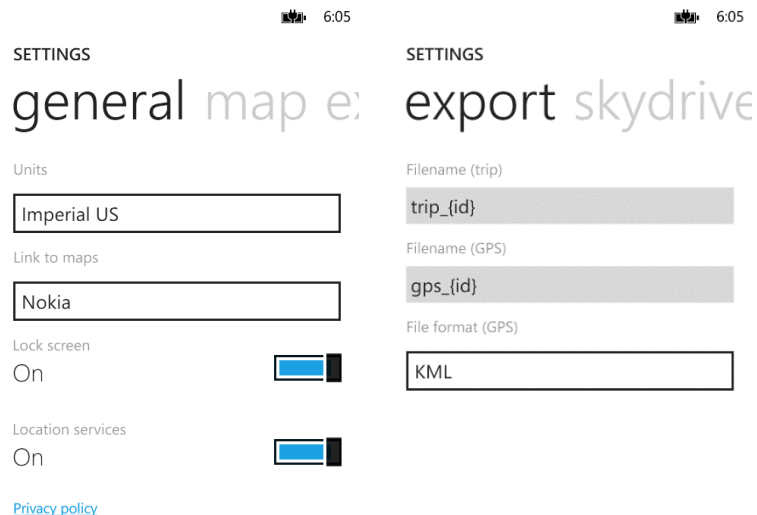
14.3 Nastavení

V nastavení si uživatel může zvolit jednotkový systém podle toho na co je zvyklý. Výchozí hodnota je nastavena v závislosti na regionu (`RegionInfo.CurrentRegion.IsMetric`).

Dále je možné zakázat vypnutí displeje v rámci aplikace přes `UserIdleDetectionMode`. Z důvodu správného fungování se hodnota nastavuje vždy při spuštění aplikace.

Pro přihlášení uživatele k SkyDrive účtu je využito přihlašovacího „live“ tlačítka, u něhož je odchycena událost `SessionChanged`. Odchycení je provedeno v „code behind“, protože v případě převedení na příkaz, není vykonán při automatickém přihlášení [36].

```
1 <live:SignInButton
2     Branding="Skydrive"
3     ClientId="{Binding Path=SkyDrive.ClientId}"
4     Scopes="wl.signin wl.skydrive_update wl.offline_access"
5     SessionChanged="SignInButtonSessionChanged" />
```



Obr. 29. Nastavení

14.4 Statistiky

Statistiky jsou uchovávány ve slovníku `IDictionary<int, YearStats>`, kde klíčem je odpovídající rok a hodnotou instance třídy `YearStats` obsahující kolekci všech měsíců s daty. K dispozici jsou dva režimy. První zobrazuje spotřebu v litrech a v penězích, druhý vzdálenost a uplynulý čas. Přepínání mezi nimi je realizováno přes `DataTrigger` reagující na hodnotu vlastnosti `IsDistance`.

```

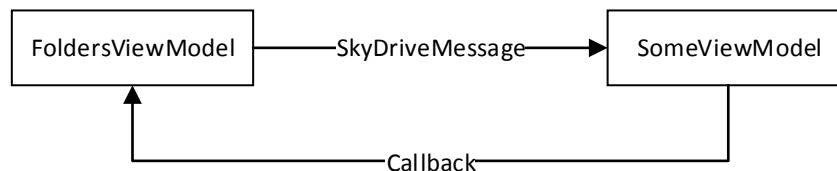
1 <ei:DataTrigger
2     Binding="{Binding Path=IsDistance}"
3     Value="True">
4     <ei:ChangePropertyAction
5         PropertyName="ContentTemplate"
6         TargetName="totalStats"
7         Value="{StaticResource TotalDistanceTemplate}" />
8 </ei:DataTrigger>
9     ...
10 </ei:DataTrigger>

```

14.5 SkyDrive

Jedná se o jednoduchý SkyDrive klient pro export a import tras ve formátu XML. Klient se může nacházet v režimu `Load`, nebo `Save` (typ `Procedure`).

Režim se nastavuje prostřednictvím zprávy SkyDriveMessage. Zde bylo třeba ošetřit stav, kdy instance ViewModel obsluhujícího SkyDrive ještě nebyla vytvořena (zpráva není nikde přijata). Při vytváření instance se tedy sám dotazuje, v jakém režimu má pracovat. Jedná se o zprávu s „callback“ funkcí [37].



Obr. 30. Předání typu pracovního režimu

Složky a soubory jsou uloženy v kolekci ObservableCollection<Model.SkyDrive.Object>. V události Tap nad jednotlivými objekty se rozlišuje typ daného objektu dle instance a podle toho se provede další činnost.

```

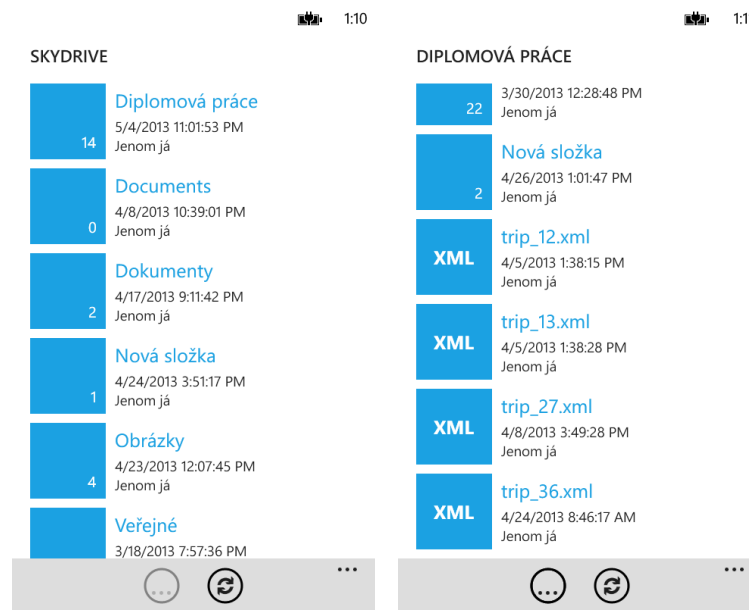
1 private void ObjectTap(Model.SkyDrive.Object obj)
2 {
3     if (obj is File)
4     {
5         Messenger.Default.Send(new NotificationMessage<File>(this,
6             this.targetType, (File)obj, ""));
7         this.navigation.GoBack();
8     }
9     else
10    {
11        this.BrowseFolder((Folder)obj);
12    }
  
```

Protože není možné si vyžádat přímo soubory s určitou příponou, jsou odfiltrovány až po načtení pomocí regulárního výrazu.

```

1 this.regPattern = String.Format(@"\.\.({0})", this.GetExtensions());
2 bool reg = Regex.IsMatch(obj["name"].ToString(), this.regPattern,
3     RegexOptions.IgnoreCase);
  
```

Pro lepší orientaci mezi složkami a soubory je pro odlišení souborů využit TypeToStyle konvertor, který zobrazí příponu souboru (XML, GPX, KML) v ikoně jako text.



Obr. 31. SkyDrive klient

14.6 Trasa

Pokud bychom měli určit, který ViewModel je z pohledu uživatele nejdůležitější a nejvíce používaný, tak je to TripViewModel. Důležitý je především kvůli tomu, že pracuje s instancí třídy Geolocator umožňující sledovat polohu.

Přesnost zjišťování polohy je nastavena na High, tedy vysokou, protože výchozí hodnota Default není pro potřeby této aplikace dostačující. Samotná poloha je sledována každých 100 metrů.

```
1 App.Geolocator.DesiredAccuracy = PositionAccuracy.High;
2 App.Geolocator.MovementThreshold = 100;
```

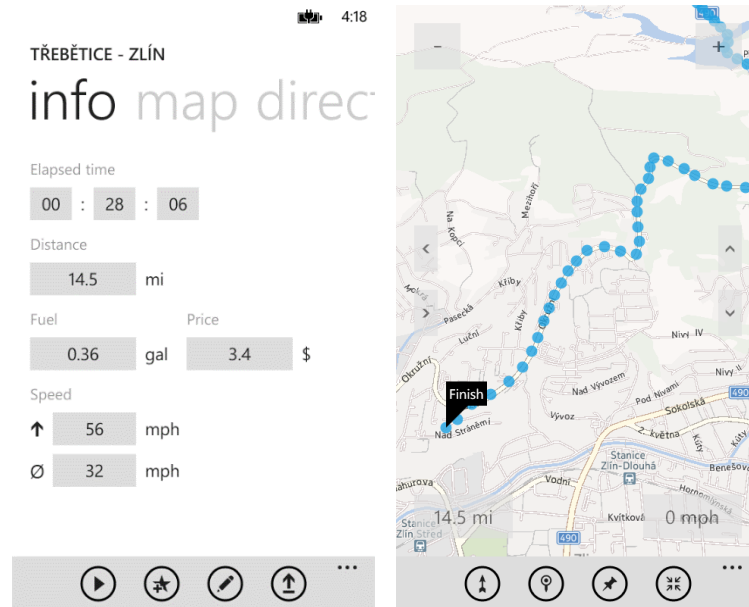
Změna polohy je sledována od okamžiku, kdy napojíme na událost PositionChanged odpovídající event handler. Napojení je provedeno v metodě Start.

Metoda Start nejdříve ověří, zdali má uživatel povoleno, ať už v rámci aplikace či telefonu, zjišťování polohy, pokud nemá, není možné trasu spustit. Před zahájením sledování změn polohy se však nejdříve zjistí aktuální umístění a teprve poté se začne pravidelně snímat poloha.

```
1 Geoposition position = await App.Geolocator.GetGeopositionAsync(...)
```

Obdobně to funguje při ukončení sledování. Nejdříve se zjistí aktuální poloha a po té se deaktivuje sledování změn polohy.

Za běhu je vzdálenost mezi dvěma místy počítána metodou `GetDistanceTo` volanou nad jedním z bodů `GeoCoordinate`, protože přesné určení přes `RouteQuery` je časově náročnější. Přesný výpočet přes `RouteQuery` je však dostupný přes položku menu „vypočítat cestu“, která všechny vzdálenosti mezi uloženými waypointy přesně přepočítá.



Obr. 32. Informace o trase

Mapa

Waypointy i špendlíky jsou do mapy přidány přes `MapItemsControl` se specifickou šablonou jednotlivých položek. Waypointy jsou vykresleny pomocí naformátovaného útvaru `Ellipse` a špendlíky využívají `Pushpin`.

Aktuální poloha uživatele je vyobrazena pomocí `UserLocationMarker`.

```

1 <maps:Map
2     x:Name="map"
3     CartographicMode="Road"
4     Center="{Binding Path=MapCenter, Mode=TwoWay}"
5     ColorMode="{Binding Path=ColorMode}"
6     Heading="{Binding Path=Heading, Mode=TwoWay}"
7     LandmarksEnabled="{Binding Path=Settings.MapLandmarks}"
8     Pitch="{Binding Path=Pitch, Mode=TwoWay}"
9     ZoomLevel="{Binding Path=ZoomLevel, Mode=TwoWay}">

```

```

10     <mapsToolkit:MapExtensions.Children>
11         <mapsToolkit:MapItemsControl
12             x:Name="waypoints"
13             ItemsSource="{Binding Path=Trip.Waypoints,
14                 Source={StaticResource Locator}}"
15             ItemTemplate="{StaticResource WaypointTemplate}" />
16     ...
17 </maps:Map>

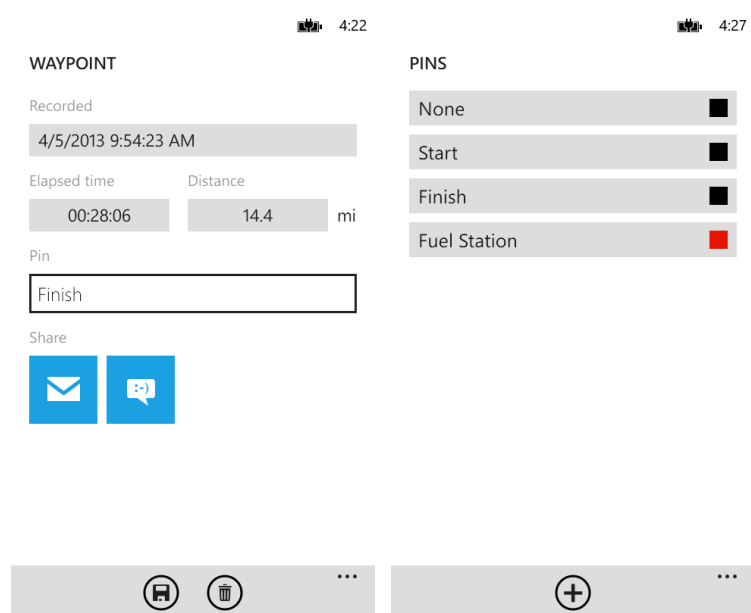
```

Na waypointy i špendlíky jsou navázány příkazy reagující na události Tap a Hold. Hold přeměrovává na stránku s informacemi o vybrané souřadnici a Tap v případě špendlíku vyvolává metodu MapSetView, která efektně přiblíží mapu k odpovídajícímu místu. U waypointů slouží pro jejich označení a následný výpočet času a vzdálenosti mezi dvěma waypointy.

```

1 private void MapSetView(GeoCoordinate waypoint, double zoom = 14,
2     double heading = 0, MapAnimationKind animation =
3     MapAnimationKind.None)
4 {
5     MapCenter mapCenter;
6     mapCenter.course = heading;
7     mapCenter.coordinate = waypoint;
8     mapCenter.zoom = zoom;
9     mapCenter.animation = animation;
10    Messenger.Default.Send<MapCenter>(mapCenter, "MapCenter");
11 }

```



Obr. 33. Detaily waypointu a přehled špendlíků

Waypointy je možné sdílet přes SMS (SmsComposeTask) resp. e-mail (EmailComposeTask). Tělo zprávy obsahuje odkaz na mapu (dle preferencí uživatele Nokia, nebo Google mapy) obsahující souřadnice vybraného waypointu.

Navigace

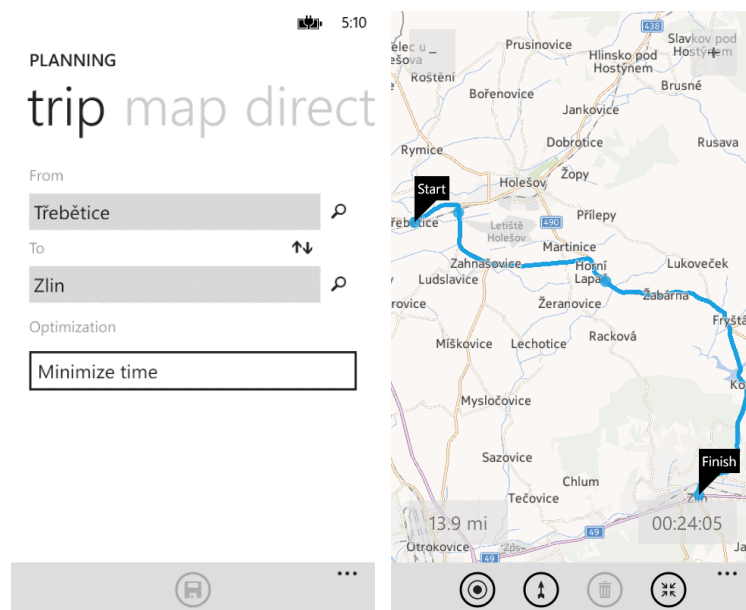
Plánování trasy nabízí dvě možnosti pro vytvoření navigace.

První možností je využití formuláře pro zadání odkud – kam trasa povede. Vyhledávání zadané lokace je realizováno přes GeocodeQuery, které zadaný text porovnává nejen s názvy měst, ale i třeba s okresy, kraji, názvy ulic apod. a může se stát, že lokace nebude obsahovat název města. Z toho důvodu jsou tyto oblasti odfiltrovány.

```

1 ...
2 this.geocode.SearchTerm = term;
3 var data = await this.geocode.GetMapLocationsAsync();
4 for (int i = 0; i < data.Count; i++)
5 {
6     if (!String.IsNullOrEmpty(data[i].Information.Address.City))
7     {
8         locations.Add(data[i]);
9     }
10 }
11 ...

```



Obr. 34. Plánování trasy

Druhou možností jak naplánovat trasu je možnost využití přímo mapy, která odchyťává událost Hold. Uživatel si sám vloží do mapy body, které má trasa protínat. Může tedy vytvořit libovolné alternativy cest.

```
1 <i:EventTrigger
2     EventName="Hold">
3     <ctMaps:MapGestureToCommand
4         Command="{Binding Path=MapHoldCommand}"
5         PassEventArgsToCommand="True" />
6 </i:EventTrigger>
```

Pro správné určení souřadnic na mapě bylo třeba vytvořit vlastní příkaz odvozený od `EventToCommand` převádějící bod z mapy na odpovídající `GeoCoordinate`.

```
1 protected override void Invoke(object parameter)
2 {
3     object param = null;
4     Map map = this.AssociatedObject as Map;
5     if (map != null)
6     {
7         GestureEventArgs e = parameter as GestureEventArgs;
8         if (e != null)
9         {
10            param = map.ConvertViewportPointToGeoCoordinate(e.GetPosition(map));
11        }
12    }
13    if (param == null) { param = parameter; }
14    base.Invoke(param);
15 }
```

15 PUBLIKOVÁNÍ APLIKACE

Vždy když vyvíjíme aplikace je našim primárním cílem je po dokončení dát k dispozici dalším lidem ať už zdarma nebo za odpovídající poplatek. Pro potřeby aplikací a her v rámci Windows Phone je k dispozici Windows Phone Store (dříve Marketplace). Nabízí veškeré dostupné aplikace a hry od vývojářů z celého světa.

V rámci první části procesu publikování je nutné uvést název aplikace, kategorii, cenu a vybrat cílový trh.

Tato aplikace je publikována pod názvem Car Tracking a je zařazena do kategorie „cestování + navigace“, podkategorie „navigace“ [40]. Distribuována je zdarma na všechny dostupné trhy.

V druhé části se provádí nahrání XAP souboru, tedy přeložené aplikaci v release verzi. Dále je třeba zadat popis aplikace a screenshoty (max 8) v podporovaných jazycích a rozlišeních.

Car Tracking je lokalizována v českém a anglickém jazyce, obsahuje tedy jak český tak i anglický popis včetně lokalizovaných screenshotů pro všechna rozlišení (WXGA, WVGA, 720p).

Po splnění předcházejících náležitostí aplikace přechází do schvalovacího procesu, který je zpravidla dokončen do 5 pracovních dnů.

Zde je však nutné zmínit jednu ne zrovna příjemnou skutečnost. Pokud se aplikace nachází v ověřovacím cyklu, je v jistém smyslu zakonzervována a nelze ji nijak upravovat (ani popisek či screenshoty). Když tedy sami nalezneme chybu v aplikaci nebo nesrovnalost v popisku, musíme nejdříve vyčkat na kladné či záporné vyjádření testera a teprve poté můžeme provést úpravy. V konečném důsledku pak dochází ke zbytečným časovým ztrátám.

ZÁVĚR

Vytvořená aplikace umožňuje plánování, sledování a zaznamenání jízdy autem s využitím nativních mapových podkladů Nokia vytvořených pro vývoj aplikací na Windows Phone 8, což v konečném důsledku znamená, že aplikace nevyžaduje pro své fungování připojení k internetu.

V rámci plánování si uživatel může vytvořit libovolné varianty jakékoliv cesty pomocí bodů, které si sám přidává do mapy. Sledování a zaznamenání jízdy pomocí GPS je prováděno vždy při spuštění resp. ukončení a při změně každých 100 metrů.

Data z aplikace je možné ukládat a v případě potřeby i zpět načítat z úložiště SkyDrive pomocí vlastního integrovaného klienta využívajícího Live Connect API.

Při testování byl zjištěn docela zajímavý problém. Prostředek pro výpočet vzdáleností (RouteQuery) mezi jednotlivými waypointy si nedokáže poradit s body umístěnými v místech s mimo úrovnovým křížením cest (vždy se bere výše položená) a dochází tak ke zkreslení naměřených údajů. V této situaci je pak vhodné daný bod zcela odstranit.

V případě dalšího rozvoje aplikace by se dalo uvažovat o rozšíření prostřednictvím dopravních informací získaných ze serveru dopravniinfo.cz. Data jsou distribuována zdarma na základě jednotné typové smlouvy pomocí standardních webových služeb v datovém formátu XML.

Od 28. března do 9. května 2013 si aplikaci i díky lokalizaci do anglického jazyka nainstalovalo přes 500 uživatelů z celého světa. Největší podíl zaujímá Thajsko, Nigérie a Saudská Arábie (Tab. 7).

Tab. 7. Počet stažení aplikace ve vybraných zemích

	Země	Počet stažení
1. – 2.	Thajsko, Nigérie	82
3.	Saudská Arábie	36
...
10.	Česká republika	15

ZÁVĚR V ANGLIČTINĚ

Created application enables planning, tracking and logging of car movement with use of native Nokia map sources which were created for Windows Phone 8 application development, which in final consequence means that there is no need for internet connection for the application to work.

In the planning part, the user could create many arbitrary variants of journey using points, which he/she adds to map by himself/herself. Car position logging withing route using GPS is done when application starts/closes and periodically when changes occurs each 100 meters.

Application data could be stored and loaded in the case of need to/from SkyDrive storage using its own integrated client which is based on Live Connect API.

In the testing phase, there has been discovered very interesting problem. The agent for distant computation (RouteQuery) between particular waypoints is unable to work with points located in places with grade-separated crossing of routes (it just takes route which is the highest) and this causes distortion of measured entries. The best solution in this situation is to remove given point completely.

In the future application enhancements, there is a speculation of using traffic information loaded from dopravniinfo.cz server. Data are distributed at no costs with format which is based on unified type contract using standard web services and XML data format.

From 28th March to 9th May 2013, owing to English localization, the application has been installed by more than 500 users which came from all over the world. The biggest share has Thailand, Nigeria and Saudi Arabia (Tab. 8).

Tab. 8. App downloads in selected countries

	Country	Downloads
1. – 2.	Thailand, Nigeria	82
3.	Saudia Arabia	36
...
10.	Czech Republic	15

SEZNAM POUŽITÉ LITERATURY

- [1] Historie aktualizací Windows Phone 7. In: *Windows Phone* [online]. 2013 [cit. 2013-03-28]. Dostupné z: <http://www.windowsphone.com/cs-cz/how-to/wp7/basics/update-history>
- [2] Historie aktualizací Windows Phone 8. In: *Windows Phone* [online]. 2013 [cit. 2013-03-28]. Dostupné z: <http://www.windowsphone.com/cs-CZ/how-to/wp8/basics/windows-phone-8-update-history>
- [3] Windows Phone 8: Multiple Screen Resolutions. In: *Silverlight Show* [online]. 2012 [cit. 2013-03-27]. Dostupné z: <http://www.silverlightshow.net/items/Windows-Phone-8-Multiple-Screen-Resolutions.aspx>
- [4] Multi-resolution apps for Windows Phone 8. In: *MSDN* [online]. 2013 [cit. 2013-03-27]. Dostupné z: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj206974\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj206974(v=vs.105).aspx)
- [5] Tile design guidelines for Windows Phone. In: *Windows Phone* [online]. 2013 [cit. 2013-03-28]. Dostupné z: [http://msdn.microsoft.com/en-us/library/windowsphone/design/jj662929\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/design/jj662929(v=vs.105).aspx)
- [6] Tiles for Windows Phone. In: *Windows Phone* [online]. 2013 [cit. 2013-03-28]. Dostupné z: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202948\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202948(v=vs.105).aspx)
- [7] PETZOLD, Charles. *Programming Windows Phone 7*. Redmond, WA: Microsoft Press, 2010, p. cm. ISBN 978-073-5643-352.
- [8] Panorama control for Windows Phone. In: *Windows Phone* [online]. 2013 [cit. 2013-03-29]. Dostupné z: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff941104\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff941104(v=vs.105).aspx)
- [9] Pivot control for Windows Phone. In: *Windows Phone* [online]. 2013 [cit. 2013-03-29]. Dostupné z: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff941098\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff941098(v=vs.105).aspx)
- [10] App monitoring for Windows Phone. In: *Windows Phone* [online]. 2013 [cit. 2013-03-30]. Dostupné z: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj215907\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj215907(v=vs.105).aspx)

- [11] Implementing the MVVM Pattern. *Microsoft Developer Network* [online]. 2012 [cit. 2013-01-24]. Dostupné z: [http://msdn.microsoft.com/en-us/library/gg405484\(PandP.40\).aspx](http://msdn.microsoft.com/en-us/library/gg405484(PandP.40).aspx)
- [12] In MVVM model should the model implement INotifyPropertyChanged interface. In: *StackOverflow* [online]. 2011 [cit. 2013-03-26]. Dostupné z: <http://stackoverflow.com/a/6923833>
- [13] BUGNION, Laurent. *MVVM Light Toolkit* [online]. 2013 [cit. 2013-01-23]. Dostupné z: <http://mvvmlight.codeplex.com/>
- [14] *NuGet Gallery* [online]. c 2013 [cit. 2013-01-24]. Dostupné z: <http://www.nuget.org/>
- [15] MVVM Light messenger check if a class is already registered. In: *Stack Overflow* [online]. 2011 [cit. 2013-01-26]. Dostupné z: <http://stackoverflow.com/a/5479828>
- [16] JONES, Kevin. *WP7 IoC* [online]. 2011 [cit. 2013-01-23]. Dostupné z: <https://gist.github.com/716137>
- [17] MVVM Light Toolkit V3 Alpha 2: EventToCommand behavior. *Laurent Bugnion (GalaSoft)* [online]. 2009 [cit. 2013-01-24]. Dostupné z: <http://geekswithblogs.net/lbugnion/archive/2009/11/05/mvvm-light-toolkit-v3-alpha-2-eventtocommand-behavior.aspx>
- [18] AppBarUtils. LEE, Allen. *AppBarUtils* [online]. 2012 [cit. 2013-03-25]. Dostupné z: <http://appbarutils.codeplex.com/>
- [19] Windows Phone apps. *Live Connect* [online]. 2013 [cit. 2013-05-03]. Dostupné z: <http://msdn.microsoft.com/en-us/library/live/hh826550.aspx>
- [20] Scopes and permissions. *Live Connect* [online]. 2013 [cit. 2013-05-03]. Dostupné z: <http://msdn.microsoft.com/en-us/library/live/hh243646.aspx>
- [21] IValueConverter Interface. In: *Windows Phone* [online]. 2013 [cit. 2013-04-25]. Dostupné z: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/system.windows.data.ivalueconverter\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/system.windows.data.ivalueconverter(v=vs.105).aspx)
- [22] Extension Methods. In: *MSDN* [online]. 2012 [cit. 2013-04-25]. Dostupné z: [http://msdn.microsoft.com/en-us/library/bb383977\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/bb383977(v=vs.110).aspx)
- [23] The Windows Phone Toolkit. *CodePlex* [online]. 2012 [cit. 2013-04-27]. Dostupné z: <http://phone.codeplex.com/>

- [24] How can I get the DateTime for the start of the week. In: *Stack Overflow* [online]. 2008 [cit. 2013-04-25]. Dostupné z: <http://stackoverflow.com/a/38064/1136101>
- [25] Globalization and localization for Windows Phone. In: *Windows Phone* [online]. 2013 [cit. 2013-04-26]. Dostupné z: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff637522\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff637522(v=vs.105).aspx)
- [26] How to build a localized app for Windows Phone. In: *Windows Phone* [online]. 2013 [cit. 2013-04-26]. Dostupné z: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff637520\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff637520(v=vs.105).aspx)
- [27] CAMERON, Rob. *Pro Windows Phone App Development*. 2nd ed. Berkeley, CA: Apress, 2011. ISBN 978-143-0239-376.
- [28] WILDERMUTH, Shawn. *Essential Windows Phone 7.5: Application Development with Silverlight*. Upper Saddle River, N.J.: Addison-Wesley/Pearson Education, 2012, xxxiii, 476 p. ISBN 978-032-1752-130.
- [29] BUGNION, Laurent. Navigation in a #WP7 application with MVVM Light. In: *Laurent Bugnion (GalaSoft)* [online]. 2011 [cit. 2013-04-30]. Dostupné z: <http://geekswithblogs.net/lbugnion/archive/2011/01/06/navigation-in-a-wp7-application-with-mvvm-light.aspx>
- [30] Quickstart: Working with settings in Windows Phone. In: *Windows Phone* [online]. 2013 [cit. 2013-04-30]. Dostupné z: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj714090\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj714090(v=vs.105).aspx)
- [31] Local database for Windows Phone. In: *Windows Phone* [online]. 2013 [cit. 2013-04-30]. Dostupné z: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202860\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202860(v=vs.105).aspx)
- [32] How to create a basic local database app for Windows Phone. In: *Windows Phone* [online]. 2013 [cit. 2013-04-30]. Dostupné z: [http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202876\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202876(v=vs.105).aspx)
- [33] GPX 1.1 Schema Documentation. *TopoGrafix* [online]. 2007 [cit. 2013-05-03]. Dostupné z: <http://www.topografix.com/gpx/1/1/>
- [34] KML Documentation Introduction. *Google Developers* [online]. 2012 [cit. 2013-05-03]. Dostupné z: <https://developers.google.com/kml/documentation/>
- [35] Deciding on when to use XmlDocument vs XmlReader. In: *StackOverflow* [online]. 2009 [cit. 2013-05-01]. Dostupné z: <http://stackoverflow.com/a/1506316/1136101>

- [36] OnSessionChanged - Only called after first login?. In: *Live Connect* [online]. 2011 [cit. 2013-05-04]. Dostupné z: <http://social.msdn.microsoft.com/Forums/en-US/messengerconnect/thread/e729fd2e-0993-4a15-85e2-6895efd440a9>
- [37] MVVM Light - Passing Params to Target ViewModel Before Navigating. In: *Geoff's Blog* [online]. 2012 [cit. 2013-05-06]. Dostupné z: <http://geoffwebbercross.blogspot.cz/2012/04/mvvm-light-passing-params-to-target.html>
- [38] VAUGHAN, Daniel. *Windows Phone 7.5 Unleashed*. Indianapolis, Ind.: Sams Pub., c2012, xi, 1095 p. ISBN 06-723-3348-1.
- [39] LECRENSKI, Nick G, Karli WATSON a Robert FONSECA-ENSOR. *Beginning Windows Phone 7 Application Development: Building Windows Phone Applications using Silverlight and XNA*. Indianapolis, Ind.: Wiley, 2011, xxi, 578 p. ISBN 978-1-118-09628-4.
- [40] KRAMPL, Jakub. Car Tracking. *Windows Phone Store* [online]. 2013 [cit. 2013-05-03]. Dostupné z: <http://www.windowsphone.com/cs-cz/store/app/car-tracking/4c555056-5376-4a82-a0e0-72a3ccaccee>
- [41] Servis médiím a odběratelům. *Dopravní info* [online]. 2010 [cit. 2013-05-08]. Dostupné z: <http://www.dopravniinfo.cz/servis-mediim-a-odberatelum>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

IoC	Inversion of Control
LINQ	Language Integrated Query
mpg	Miles per gallon
mph	Miles per hour
MVVM	Model-View-ViewModel
SDK	Software Development Kit
SQL	Structured Query Language
UTC	Coordinated Universal Time
WGS 84	World Geodetic System 1984
WP8	Windows Phone 8
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
XML	Extensible Markup Language

SEZNAM OBRÁZKŮ

Obr. 1. Výběr rozlišení	12
Obr. 2. Ukázka dlaždic [5].....	12
Obr. 3. Flip dlaždice [6].....	14
Obr. 4. Iconic dlaždice [6]	14
Obr. 5. Cycle dlaždice [6].....	15
Obr. 6. Panorama.....	16
Obr. 7. Pivot.....	16
Obr. 8. Simulation Dashboard	17
Obr. 9. Logika MVVM vzoru [11].....	18
Obr. 10. Logo MVVM Light Toolkit [13].....	20
Obr. 11. Ukázka souboru packages.config	20
Obr. 12. Soubor s českou lokalizací.....	42
Obr. 13. Vrstva ovládacích tlačítek mapy	44
Obr. 14. Ovládací prvek Stats.....	44
Obr. 15. Formulář CarForm.....	45
Obr. 16. Formulář PinForm	45
Obr. 17. Výběr barvy špendlíku.....	46
Obr. 18. Formulář NavigationForm	47
Obr. 19. Formulář TripForm.....	47
Obr. 20. Třídy implementující rozhraní IUnitConverter	50
Obr. 21. Implementace tříd Trip, TripsGroup<T> a Car.....	59
Obr. 22. Implementace tříd Waypoint a Pin	60
Obr. 23. Implementace navigace.....	61
Obr. 24. Implementace statistik	62
Obr. 25. Hierarchie SkyDrive objektů.....	63
Obr. 26. Hierarchie tříd pro vytváření XML souborů	64
Obr. 27. Hierarchie tříd pro parsování XML souborů.....	67
Obr. 28. Úvodní stránka	69
Obr. 29. Nastavení.....	71
Obr. 30. Předání typu pracovního režimu.....	72
Obr. 31. SkyDrive klient.....	73
Obr. 32. Informace o trase	74

Obr. 33. Detaily waypointu a přehled špendlíků	75
Obr. 34. Plánování trasy	76

SEZNAM TABULEK

Tab. 1. Přehled verzí Windows Phone [1][2]	11
Tab. 2. Podporovaná rozlišení [4]	12
Tab. 3. Rozměry dlaždic [5]	13
Tab. 4. Převodní tabulka délky	50
Tab. 5. Převodní tabulka rychlosti	50
Tab. 6. Převodní tabulka objemu	51
Tab. 7. Počet stažení aplikace ve vybraných zemích	79
Tab. 8. App downloads in selected countries	80

SEZNAM PŘÍLOH

P I CD-ROM