VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

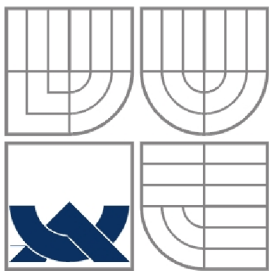
# PLÁNOVÁNÍ CEST - MASHUP

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

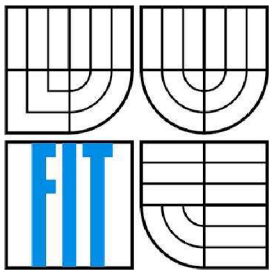
AUTOR PRÁCE
AUTHOR

ŠTĚPÁN PŘICHYSTAL

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# PLÁNOVÁNÍ CEST - MASHUP

TRIP PLANING - MASHUP

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ŠTĚPÁN PŘICHYSTAL

VEDOUCÍ PRÁCE
SUPERVISOR

ING. RUDOLF KAJAN

BRNO 2011

## Abstrakt

Cílem bakalářské práce je vytvoření mashup aplikace pro plánování cest, která kombinuje data z více zdrojů. Práce obsahuje teoretické informace o problematice týkající se tématu. Je tedy zaměřená na vysvětlení pojmu Web 2.0, na seznámení s mashup aplikacemi a jejich typy a na popis charakteristik RIA aplikací a jejich technologií. Není opomenuto ani seznámení s implementační technologií Silverlight, které je věnován poměrně velký prostor. Další část bakalářské práce obsahuje jednotlivé fáze vývoje aplikace. Kapitoly popisují rozbor zadání, návrh, stručný popis implementace a testování. Závěr je věnován zhodnocení dosažených výsledků a zamyšlení nad případným rozšířením aplikace v budoucnu.



## Abstract

The target of this thesis is creating an mashup aplication for planning trip, which combines dates from more sources. The thesis contains theoretic informations of problems concerned about the topic so it is focused on explication of notion Web 2.0, then it is focused on acquaint with mashup applications and their types and on desription of characterizations of RIA applications and their technologies. It is not omitted even an acquaint with implementing technology Silverlight, that is dedicated relatively a lot of place in this thesis. The other part of thesis contains single period of development of application. The chapters describes an analysis of entering, the proposal, a brief description of implementation and testing. The finish of the thesis is dedicated to estimation and thinking of pertinent enlargement of application in the future.



## Klíčová slova

Mashup, Plánování cest, Web 2.0, RIA aplikace, Silverlight

## Keywords

Mashup, Trip planing, Web 2.0, RIA application, Silverlight


## Citace

Přichystal Štěpán: Plánování cest - mashup, bakalářská práce, Brno, FIT VUT v Brně, 2011

# Plánování cest - mashup

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Rudolfa Kajana a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Štěpán Přichystal
9.května 2011

## Poděkování

Rád bych na tomto místě poděkoval vedoucímu bakalářské práce panu Ing. Rudolfu Kajanovi za odborný dohled a cenné rady, díky kterým se mi práci podařilo vypracovat a úspěšně dokončit.

# Obsah

# 1 Úvod

Cílem této bakalářské práce je vytvoření webové aplikace sloužící pro plánování cest. Aplikace by měla umožňovat naplánování vícedenní například pracovní cesty a následné vytisknutí přehledného plánu. V takovém plánu by mělo být zahrnuto například obecné informace o cestě jako vzdálenost, termín cesty a cílová města. Dále pak naplánované dopravní spojení mezi jednotlivými městy a v případě setrvání ve městě také informace o hotelech určených k přespání. Protože plán je vícedenní a na jednotlivé dny většinou plánovaná nějaká schůzka či jiná činnost, tak aplikace by měla umožnit zaznamenat i veškerých aktivit, které uživatel na své cestě plánuje. Pro zaznamenání dat, jako vzdálenosti mezi městy či vyhledání hotelů ve městě, jsou potřeba externí zdroje, na kterých je aplikace postavena a tímto se dostáváme k tématu bakalářské práce, kterým je mashup. Jednoduše řečeno mashup je aplikace kombinující data a funkcionalitu ze dvou a více zdrojů.

Celá práce obsahuje několik kapitol a první polovina z nich se zabývá obecnou problematikou tématu a jednotlivými pojmy s ní spojenými. Mezi tyto pojmy patří Web 2.0, který je s mashup aplikacemi úzce propojen a je mu věnována první kapitola. Následující kapitola je věnována samotným mashup aplikacím, její historii, jednotlivým druhům mashup aplikací a nemalá část také editorům, určeným k vytváření těchto specifických aplikací. Celá aplikace je vyvinuta v technologií Silverlight s poměrně krátkou historií. Této technologie je v práci věnován velký prostor, který obsahuje obecný popis, jednotlivé verze technologie, bezpečnost a srovnání s podobnou technologií WPF. Protože Silverlight je technologie pro tvorbu specifických aplikací označovaných jako RIA, tak této problematice je věnována další samostatná kapitola.

Druhá polovina kapitol je věnována samotné aplikaci a jejím vývojovým fázím. V šesté kapitole se čtenář v textu dočte jaké je zadání práce a požadavky na aplikaci. Následující kapitola se zaměřuje na zhodnocení jednotlivých dostupných externích zdrojů použitelných pro aplikaci, na návrh funkcionality a dále na návrh prezentační vrstvy aplikace. Další kapitola informuje o použitých nástrojích a technologiích při vývoji aplikace. Dále pak se věnuje stručnému popisu implementace, seznámení se strukturou aplikace a na závěr problémům, které se vyskytly během implementace. Důležitou fází vývoje je samozřejmě testování, kterému se věnuje předposlední kapitola.

# 2 Web 2.0

Tato kapitola se zaměřuje na pojem Web 2.0. Protože mashup aplikace do tohoto pojmu spadají, čtenář se dozví co je to Web 2.0 a jaké jsou jeho typické znaky. V kapitole není opomenuto srovnání Webu 1.0 s Webem 2.0 a v poslední části je také zmínka o budoucnosti Webu 2.0.

## 2.1 Vysvětlení pojmu

Web 2.0 je označení pro weby, kde byl pevný statický obsah nahrazený dynamickým obsahem, tvořeným uživateli na internetu a sdílením informací z více zdrojů [3]. Weby, jak jsme je znali dříve, byly statické plné textu, který chtěl uživatele o něčem informovat. Hlavním rozdílem Webu 1.0 oproti Webu 2.0 je, že u Webu 2.0 se na vytváření obsahu webu podílí daleko větší procento uživatelů než u Webu 1.0. Po stránce vzhledové Weby 1.0 nebyly ani tak příliš graficky bohaté, protože technologie, které byly v té době dostupné, nebyly tak vyspělé. Nejčastěji se používal jazyk HTML[1], kaskádové styly CSS[2] a JavaScript[3]. Od roku 2004 a s příchodem nových technologií, například AJAX (podrobnější informace v kapitole 4.3.1), byl pojem Web 2.0 slyšet stále častěji. Přesná definice tohoto pojmu však neexistuje a je vykládána různými způsoby. Obecné povědomí o Webu 2.0 je tedy zhruba takové, že je to web využívající nové technologie s bohatým grafickým ztvárněním, kde informace můžou být zkombinovány z více zdrojů a hlavně kde k tvoření obsahu přispívají velkým procentem sami uživatelé internetu.

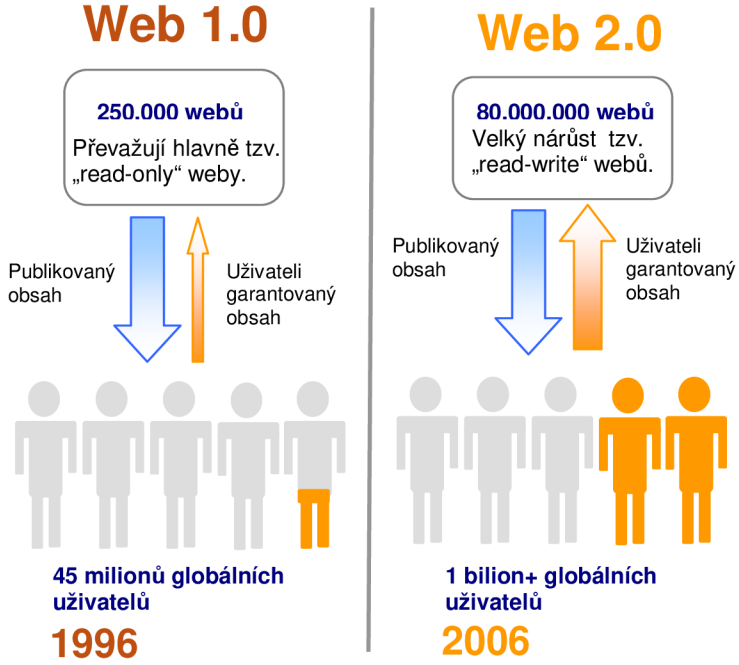
## 2.2 Typické znaky

Pro Web 2.0 existuje více typických znaků, následující souhrn jsou nejtypičtější z nich:

- Uživatelé internetu se podílí na tvorbě obsahu webu a vzniká tak obsah garantovaný uživateli. Ti jsou pak vtaženi více do navštěvování daného webu a roste tak i jeho popularita a podílí se na vytváření podoby internetu celkově. Na obrázku 2.1 je srovnání celosvětového internetu v roce 1996 a 2006, kde je vyobrazeno jak se na tvorbě obsahu webů podílí uživatelé internetu,
- Dalším znakem je sdílení a znovupoužití všech možných informací, od prostého textu až po obrázky a videa.
- Tagy - slouží pro označování obsahu uživateli internetu. Tyto slova či fráze jsou pak někde zvýrazněna v prostoru webu a upozorňují na nejnavštěvovanější či nejvíce populární témata.
- Sociální sítě - dalším velmi významným prvkem je všemožné propojení webu se sociálními sítěmi. Uživatelé pomocí nich mohou snadno reagovat na jakýkoliv článek, video či jiný obsah na webu. Usnadňuje se tímto způsobem sdílení informací a vytváří se komunita webu. Příkladem takové sítě je Facebook[4].

[1] http://www.w3.org/MarkUp/
[2] http://www.w3.org/TR/CSS21/
[3] http://www.w3.org/standards/techs/js/
[4] http://www.facebook.com/

- RSS - je to technologie, která umožňuje uživatelům se přihlásit k odběru novinek z webu, který RSS podporuje. Uživateli pak stačí použít RSS čtečku a najít vhodný RSS zdroj na webu.
- Mashup - je dalším typickým znakem pro Weby 2.0. Je to namíchání jednoho nebo několika větších či menších zdrojů do jednoho webu. Jako zdroje jsou dnes nejoblíbenější portály například YouTube[1] nebo Flicker[2]. Pojem mashup bude dále vysvětlen v kapitole 3.
- Blogy - v dnešní době, kdy je vytvoření vlastního blogu opravdu jednoduché, tak uživatelé tímto způsobem také rozšiřují web a zvyšují konkurenceschopnost jednotlivých webů.
- Lepší struktura webu a snadnější orientace a procházení webu díky propracované hyperlinkové struktuře, která není založena jen na statických odkazech. Moderní hyperlinkové struktury nabízí uživateli odkazy na informace, o kterých se domnívají, že by je mohli zajímat. Je to dáno způsobem jakým uživatel prochází web, z kterého je pak vyhodnoceno co by ho mohlo v dalším kroku zajímat.
- Vzhled webu - posledním nezanedbatelným znakem je vzhled a interaktivita webů. Pro grafický design to znamená používání nejnovějších technologií [1].




Obrázek 2.1: Srovnání obsahu webu, jak se na něm podíleli uživatelé v roce 1996 a roce 2006 [1]

## 2.3 Nástupce webu 2.0

V souvislosti s Webem 2.0 se také někdy mluví o tom, že je to nástupce Webu 1.0. Tento názor podle mě není správný, protože kvalitní a hodnotné informace se dají najít většinou na klasickém webu, tedy Webu 1.0. V dnešní době se již objevuje i pojem Web 3.0, který by měl být nástupcem Webu 2.0. Není však jasné, jak by měl Web 3.0 vypadat, ale zhruba by se

[1] http://www.youtube.com/
[2] http://www.flickr.com/

mělo jednat o to, že web bude ještě více interaktivnější a lépe strukturovaný než doposud. Odhaduje se že znaky Webu 3.0 by tedy mohly být následující.

- Použití prvků sémantického webu - sémantický web si dává za úkol ukládat informace pomocí standardizovaných pravidel, které usnadní vyhledávání a zpracování informací.
- Použití mikroformátů - mikroformát[1] je způsob jak ukládat informace do webů, které by byly následně strojově čitelné. Pak by speciální programy jednoduše mohly zpracovávat informace určené koncovým uživatelům.
- Přístup na web pomocí různých zařízení - s tímto se běžně setkáváme již dnes. V zásadě jde o to, aby weby byly dostupné skrz veškerá zařízení PC, PDA, mobilní telefony a jiná elektronická zařízení.
- Částečná umělá inteligence webů - weby by měly inteligentně spolupracovat s uživatelem, nabízet mu přesně ten obsah, který potřebuje.
- Větší využití videa.
- 3D prostředí webových prohlížečů [2].

# 3 Mashup

Hlavním tématem mé bakalářské práce jsou mashup aplikace a proto v této kapitole bude vysvětleno k čemu mashupy slouží, jaká je jejich historie, s jakými druhy aplikací se můžeme setkat a pojednání o budoucnosti těchto aplikací. Na mashup aplikace existují i speciální editory, kterým je věnována poslední kapitola.

## 3.1 Vysvětlení pojmu

Mashup je webová stránka či aplikace, která kombinuje data, funkčnosti či jiné použitelné prvky ze dvou a více zdrojů [5]. Česky se pojem může přeložit jako míchanice. Mashup můžeme zařadit do oblasti Webu 2.0 a je to určitý současný trend. U nás v České republice není však zdaleka tak populární jako jinde ve světě. Mashupy jsou založené na informacích z cizích zdrojů, ke kterým přistupují skrze tzv. API. Tento pojem bude vysvětlen v podkapitole 3.4.1 .

Výhoda jednoduchých mashup aplikací je v tom, že jsou poměrně snadno vytvořitelné bez složitého programování a zdroje pro mashup aplikaci jsou v mnoha případech zdarma. Největší síla mashup aplikací je ovšem ve využití existujících propracovaných a výkonných službách. Dříve, když nebyly tyto služby k dispozici, tak firmy vytvářely složitá drahá řešení například pro práci s polohou a souřadnicemi GPS[2]. Dnes, například pomocí Google Maps[3], je to mnohem jednodušší a zdarma. Základem je tedy zpřístupnění projektu velkého vývojáře skrze API a malý vývojář poté může využívat výhody velkého a složitého projektu ve svém malém projektu. Malý vývojář tedy dostává k dispozici zdarma nástroj, na jehož vývoj by neměl peníze a využívá tyto zdroje a funkce. Výhoda

[1] http://microformats.cz/
[2] http://www.gps.gov/
[3] http://maps.google.com/

pro velké vývojáře a důvod, proč vůbec své projekty a zdroje vybudované za velké peníze nabízejí, je reklama. Tím, že je vytvořen mashup s jejich zpřístupněným projektem, je automaticky vytvořena reklama pro vývojáře poskytující zdroj. Poskytovatel například v případě portálu Amazon[1], potom má určitý podíl na prodeji z knihy, kterou uživatel webu koupí přes mashup aplikaci, využívající API od Amazon. V konečném důsledku tedy z takových aplikací těží jak vývojář mashup aplikace, tak vývojář poskytující zdroj, neboli API ke svému projektu [6].

## 3.2 Historie

Nástup mashup webů je spojen s rozvojem Webu 2.0. Ve stejnou dobu rozšiřování Webu 2.0 se začaly rozšiřovat i mashup aplikace. První aplikace využívali nejčastěji mapové služby a služby s různými fotoalby a kombinovaly je s daty libovolného druhu. Později začaly mashup aplikace využívat i v podnikové sféře, kde firmy kombinovaly dostupné mashup zdroje s vlastními podnikovými daty a poskytovaly tak nové zajímavé pohledy na svoje data. Dnes je trendem snaha vývojářů zpřístupnit každý jejich větší projekt na internetu skrze své API. Doposud vzniklo obrovské množství takových webů a i nadále je počet mashupů stále vzrůstající [4].

## 3.3 Typy mashup aplikací

V zásadě existují tři druhy mashup aplikací a to datové, uživatelské a podnikové. Nejběžnějším typem je uživatelský mashup zaměřený na širokou veřejnost. Následující text obsahuje popis jednotlivých typů.

- Datový mashup - kombinuje různé zdroje s podobnými daty a ze všech použitých zdrojů vytvoří jednotný celek. Tento zkombinovaný celek dále zpřístupní jako jeden velký nový zdroj. Tento nový zdroj poskytuje poté data, které neposkytoval žádný zdroj, ze kterého je tento nový zdroj vytvořen. Tímto způsobem vzniká zatím neexistující nová služba na internetu. Příkladem je webová stránka http://www.hotelscombined.com, která kombinuje zdroje http://www.expedia.com, http://www.booking.com, http://www.hotels.com a z těchto zdrojů vytváří nový zdroj pro vyhledávání hotelů.
- Uživatelský mashup – kombinuje různé typy dat z více zdrojů, na rozdíl od datového, který kombinuje podobné typy dat. Uživatelská mashup aplikace je určena pro konečné uživatele internetu. Například může kombinovat fotoalba z portálu Flickr s mapovými podklady z Google Maps. Příkladem takové stránky je web http://www.housingmaps.com, který zobrazuje ceny domů a bytů v různých státech USA, kde jsou spojené dva zdroje a to Google Maps a zdroj s informacemi z realitní kanceláře.
- Podnikové mashupy – jsou aplikace, které kombinují externí zdroje a vlastní podniková data. Vznikají tak bezpečné a vizuálně bohaté aplikace, které zobrazují obohacená podniková data o informace z externích zdrojů [7].

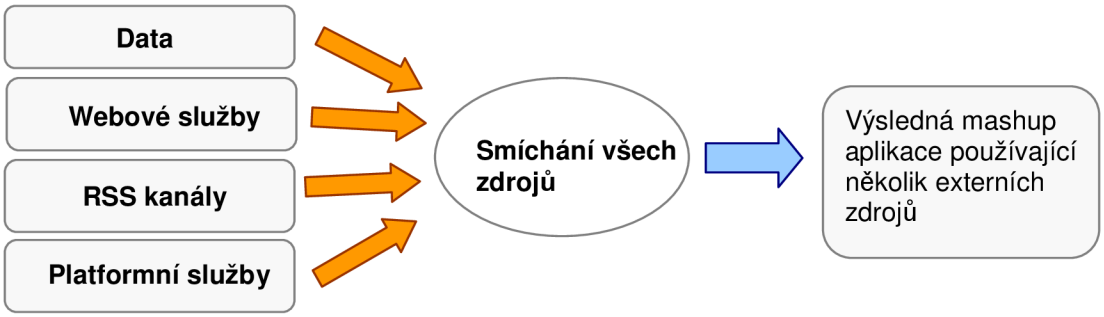
[1] http://www.amazon.com/

## 3.4 Architektura mashup aplikací

Architektura Mashup aplikace je rozdělena do tří vrstev:

- Uživatelské rozhraní - nejčastěji používanými technologiemi jsou HTML/XHTML, Javascript, CSS, AJAX, Flash, Silverlight (podrobnější informace v kapitole 5).
- Webové služby - zdroje jsou zpřístupněny pomocí API. Informace co je to API a jaké využívá technologie jsou v podkapitole 3.4.1.
- Data - na mysli je správa dat, posílání ukládání, přijímání a jiné. Používané technologie jsou XML[1], JSON[2] [7].

Mashupy se dají rozdělit ještě na další kategorie a to podle místa zpracování dat. Dělí se na webové a serverové. Webové mashupy využívají na zpracování dat ze zdrojů a následné zobrazení přímo webové prohlížeče u klienta. Serverové mashupy zpracovávají data už na straně serveru a posílají je ve finální podobě klientovi do prohlížeče. Zdroje typické pro mashup aplikaci jsou vidět na obrázku 3.1.




Obrázek 3.1: Ukázka nejtypičtějších zdrojů pro mashup aplikaci [4]

### 3.4.1 API

Obecně pojem API (Application Programming Interface) znamená rozhraní pro programování aplikací. Jde o sbírku procedur, funkcí nebo tříd nějaké knihovny, které může programátor využívat. API určuje, jakým způsobem se tyto funkce volají ze zdrojového kódu. Zkratka API ve spojení s webem je myšleno jako rozhraní pro webovou službu. Webová služba je softwarový systém, který umožňuje komunikaci dvou aplikací na síti. API webových služeb využívá protokol HTTP[3] a pracuje na principu zaslání dotazu a přijetí odpovědi. Technologie pro tuto komunikaci jsou XMLHTTPRequest[4], XML-RPC[5], JSON-RPC[6], SOAP[7], a REST[8] [8].

Vývojáři kteří poskytují ke svým webovým projektům API, si kladou většinou jistá omezení. Pokud je služba zdarma, většinou je v podmínkách používání API, že je smí vývojář používat jen nekomerčně. Pokud by chtěl komerční využití, musí kontaktovat osobně majitele nebo je služba zpoplatněna. Dalším omezením může být zákaz kombinování dat

[1] http://www.w3.org/XML/
[2] http://www.json.org/
[3] http://www.w3.org/Protocols/
[4] http://www.w3.org/TR/XMLHttpRequest/
[5] http://www.xmlrpc.com/
[6] http://json-rpc.org/wiki/specification/
[7] http://www.w3.org/TR/soap/
[8] http://ajaxpatterns.org/RESTful_Service/

ze zpřístupněného API s jinými externími zdroji a tedy zákaz vytváření mashup aplikací. Často se také můžeme setkat s omezeným počtem povolených dotazů na daný externí zdroj za jeden den. Pro API bývá charakteristické, že při jeho používání se musí aplikace prokázat vývojářským klíčem, který je předem přidělen. Většinou je tahle metoda zavedena i u API poskytovaných zdarma. Například v mém projektu využívám služby Bing maps[1], kde je klíč vyžadován při každém dotazu. Naproti tomu k využívání služeb Underground Weather[2], které používám pro získání informací o počasí, žádný klíč nepotřebuji.

## 3.5 Budoucnost

V počátcích, kdy se začínaly tyto aplikace rozvíjet, byl vývoj takových aplikací časově náročný. V dnešní době je situace opačná a vytváření mashupů je stále jednodušší, díky snadnému dotazování na informace pomocí dostupných API a přehledných odpovědí v XML či JSON formátu. Ke snadnému používání API přispívá i většinou kvalitně zpracovaná dokumentace a použití takového API zvládne i ne příliš zkušený programátor. Díky tomu, že mashup aplikace nabízejí lákavý obsah z více zdrojů, se i do budoucna budou podle mě těšit velké popularitě.

Jako menší problém bych viděl stálost a neustálou obměnu již existujících zdrojů a jejich API. Tuhle skutečnost můžu podepřít vlastní zkušeností. Při vypracovávání praktické části bakalářské práce jsem chtěl využít API portálu Atlas-mapy[3], bohužel už nešlo získat vývojářský klíč, protože majitelé Atlas-mapy přecházely na vyšší verzi, která jak bylo řečeno, bude dostupná do tří měsíců. Dalším příkladem je API Keyak[4], které jsem chtěl použít pro vyhledávání hotelů, ale bylo odstaveno pro časté zneužívání. Nic to nemění na tom, že v dnešní době existují tisíce mashup aplikací, pro které existuje i speciální stránka[5] s jejich seznamem. Podle její statistiky jsou každý den zaregistrovány tři nové mashup aplikace a k datu 20.3.2011 jich bylo celkem zaregistrováno 5714.

## 3.6 Editory

Realizace mashupu je možná dvěma způsoby. Klasický způsob je ten, kde programátor naprogramuje aplikaci v nějakém jazyce, přidá se nějaké uživatelské rozhraní, vybere požadované externí zdroje a použije jejich příslušné API ke zasílání a přijímání odpovědí. Tímto způsobem vzniká většina mashup aplikací. Druhým způsobem je použití speciálního editoru, který usnadní práci, ale ne ve všech případech můžete vytvářet složité aplikace. Nejznámějšími editory jsou dnes Yahoo Pipes, Kapow Katalyst nebo Dapper. Existují i další editory, které umožní vytvářet různé widgety (miniaplikace, sloužící např. pro zobrazování počasí nebo aktualit), ale už bych je nezahrnoval přímo do skupiny mashup editorů. Mám na mysli například Google Gadgets[6], které jsou určeny přímo pro tvorbu widgetů. Editoru Yahoo Pipes a Yahoo Dapper věnuji zvláštní podkapitolu, protože jsem s nimi osobně pracoval.

---

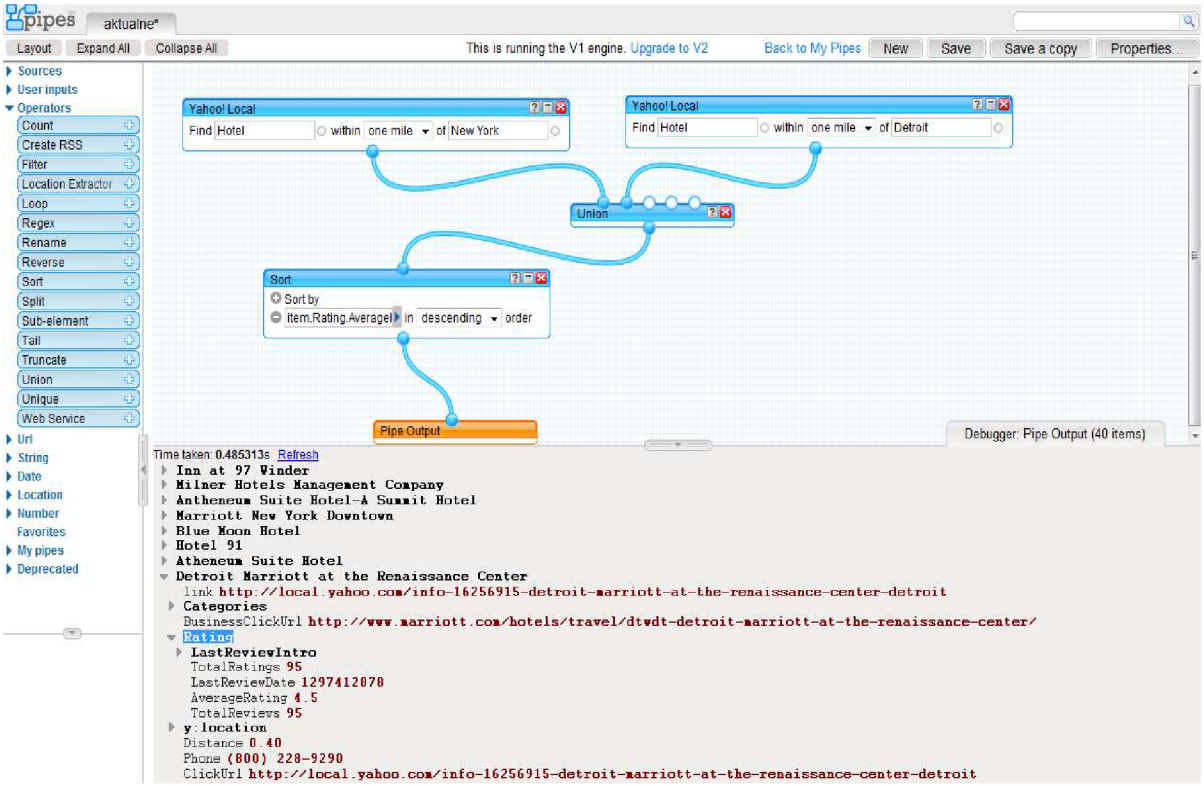
[1] http://msdn.microsoft.com/en-us/library/dd877180.aspx/
[2] http://wiki.wunderground.com/index.php/API_-_XML/
[3] http://amapy.centrum.cz/api-doc/
[4] http://www.kayak.com/labs/api/search/
[5] http://www.programmableweb.com/
[6] http://www.desktop.google.com/

Kapow Katalyst je robustní nástroj, který umožňuje vytvoření jak prezentační vrstvy (JavaScript, HTML), tak i aplikační vrstvy používající nejrůznější API. Tento editor není online a bohužel není zdarma. Lze si akorát vyzkoušet 30 denní trial verzi, která musí být ještě schválena, což se v mém případě nestalo. V minulosti existovaly ještě další velice robustní editory například Microsoft Popfly, který fungoval jako online editor webových stránek a zároveň umožňoval jednoduše používat a mixovat nejrůznější externí zdroje. Jako náhradu za tento již nefunkční editor doporučuje Microsoft program WebMatrix[1], který také usnadňuje tvorbu webových stránek. Dalším již nefunkčním editorem je Google Mashup Editor. Tento online editor také umožňoval spojování nejrůznějších API například s vlastními daty a poté hostování webu na stránkách Google.

### 3.6.1 Yahoo Pipes

Yahoo Pipes, jak bylo uvedeno, je nástroj pomocí kterého můžete vytvářet a mixovat různé zdroje a funkce. Například Yahoo Search, Yahoo Local, nástroje od společnosti Google a hlavně různé RSS zdroje. Pomocí tohoto programu lze tyto zdroje různě filtrovat, řadit, spojovat je dohromady a to pak použít třeba jako podprogram v jiné aplikaci. Například chce-li uživatel odebírat články z RSS z polského webu, ale polsky nerozumí, tak vytvoří aplikaci, která přesměruje tok polské RSS na Google překladač a vrátí RSS třeba v českém jazyce. V rámci bakalářské práce jsem si zkusil s tímto editorem pracovat. Editor je přístupný online na adrese http://yahoo.pipes.com/, je zdarma a stačí být registrovaný u společnosti Yahoo, Google, nebo Facebook. Po otevření editoru se zobrazí vpravo panel s nástroji, které se jednoduše přetahují táhnutím myši do vymezeného prostoru. Po nastavení komponent se propojí všechny části mezi sebou tzv. potrubím. V případě, že je návrh funkční, tak se po načtení dat zobrazí v dolní polovině výsledky z použitých externích zdrojů. Práce s tímto nástrojem je snadná a můžete vytvářet mashupy od jednoduchých až po ty hodně složité, kde se dá mixovat snad úplně vše. Já jsem si zkusil vytvořit aplikaci, která vyhledá pomocí Yahoo Local hotely ve městech New York a Detroit a seřadí je sestupně podle průměrného hodnocení hotelu. Ukázka online prostředí editoru a výsledek vyhledávání hotelu ve městech je na obrázku 3.2.

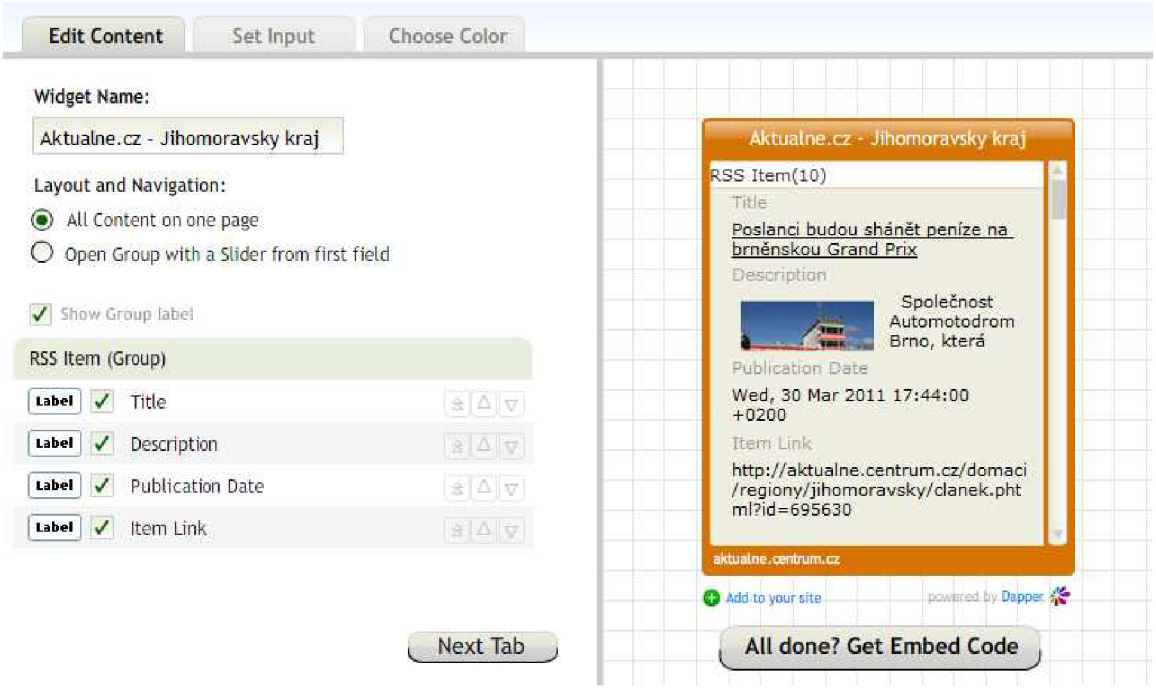
[1] http://www.webmatrix.cz/




Obrázek 3.2: Ukázka prostředí online editoru Yahoo Pipes při vyhledávání hotelů ve městech New York a Detroit

### 3.6.2 Yahoo Dapper

Dapper je další mashup editor, který je ve vlastnictví společnosti Yahoo od roku 2010. Pomocí tohoto nástroje si můžete vytvářet vlastní mashup aplikace, spojovat různé externí zdroje, RSS kanály a jiné. Práci s tímto nástrojem jsem si v rámci bakalářské práce osobně vyzkoušel a je opět velmi snadná a intuitivní bez jakéhokoliv programování podobně jako u Yahoo Pipes. Editor je dostupný online zdarma na adrese http://open.dapper.net/. Při vytvoření vlastní aplikace je uživatel provázen po krocích přehledným průvodcem, který pomůže vyhledat vhodný zdroj dat pomocí vlastního vyhledávače nebo použít již předem vybraný zdroj dat. V dalších krocích se nastavují vlastnosti a úpravy použitého zdroje. V posledním kroku je uživatel vybídnut k uložení popřípadě k dalším operacím s hotovou aplikací. Já jsem si vyzkoušel vytvořit vlastní widget, který mi vypisoval novinky z informačního portálu http://aktualne.centrum.cz/ v Jihomoravském kraji. Vytvořený widget je pak možné vložit do jakékoliv webové schránky ve flash formátu. Na obrázku 3.3 je vidět část editoru s výslednou aplikací s aktuálními zprávami z portálu http://aktualne.centrum.cz/.




Obrázek 3.3: Ukázka online prostředí editoru Dapper s výslednou ukázkovou aplikací zobrazující novinky z portálu http://aktualne.centrum.cz/

# 4 RIA

Praktická část mé bakalářské práce, díky technologii kterou používám, spadá do oblasti RIA. Čtenář se v této kapitole proto dozví, co jsou to RIA aplikace, jak se vyznačují a v neposlední řadě také jaké technologie se používají na jejich vývoj.

## 4.1 Vysvětlení pojmu

Pojem RIA je zkratka pro Rich Internet Application, což znamená volně přeloženo bohatší uživatelský zážitek z internetu [10]. RIA jsou tedy webové aplikace, které se podobají grafickým desktopovým aplikacím a mají jejich vlastnosti. Tyto aplikace uživateli internetu poskytnou lepší vizuální zážitek a vyšší interaktivitu naproti klasickým webovým stránkám. Je to dáno technologií jakou jsou RIA aplikace vytvářeny. Technologií pro vývoj je hned několik a spojují to nejlepší z prostředí desktopových a internetových aplikací. Jako příklad RIA aplikací může sloužit známý Gmail[1] od Google sloužící jako emailový klient, či Google Docs[2] sloužící k vytvoření online dokumentu pro sdílení po internetu, Zoho writer[3] další nástroj k vytváření dokumentů a mnoho dalších. RIA aplikace se dnes objevují stále častěji, jsou oblíbené a i přes jejich nevýhody, které budou popsány v následující kapitole 4.2, podle mého názoru mají nezastupitelnou roli ve světě internetu a v budoucnu se budou objevovat stále častěji.

[1] http://mail.google.com/
[2] http://docs.googel.com/
[3] http://writer.zoho.com/

## 4.2 Charakteristika

Jak již bylo řečeno RIA aplikace jsou webové aplikace blížící se svým komfortem desktopovým aplikacím. Oproti standardním webovým aplikacím se RIA vyznačuje těmito charakteristikami:

- Interakce s uživatelem - standardní webové aplikace nemají tolik a tak bohaté ovládací prvky jako RIA. Většinou jsou omezeny na formulář, tlačítko, radiobutton a checkbox. To omezuje aplikace v jejich funkčnosti a nedává jim prostor vytvořit jakékoliv uživatelské prostředí. RIA disponují širokou škálou ovládacích prvků včetně pohodlného ovládání drag and drop, které je pro uživatele internetu komfortnější a mnohdy intuitivnější. Tak je dosaženo i graficky bohatého a mnohdy zajímavého vzhledu a funkčnosti.
- Rychlá zpětná vazba - obecně webové aplikace pracují s protokolem HTTP a modelem žádost/odpověď. To znamená, že jakákoliv změna stavu u klienta vyvolá požadavek na server, ten ho zpracuje a vrátí odpověď. Takových požadavků může být celá řada, např. aktualizace části dat, přejití na jinou stránku, kontrola formulářů a mnoho dalších. Tento model chování tedy výrazně zpomaluje rychlost celé aplikace. RIA aplikace jsou kompletně klientské aplikace a veškeré programování se týká klientské strany. Tím pádem je odezva daleko rychlejší, prakticky jak u desktopových aplikací. Samozřejmě u RIA je i serverová část, která může být naprogramována v jakémkoliv jazyce. Z toho důvodu typickým znakem pro RIA aplikaci je, že například pro přístup k databázi potřebuje mezi sebou a serverem prostředníka. Dále přispívají k rychlé zpětné vazbě technologie jako real-time streamování, vysoce výkonné virtuální stroje běžící na straně klienta a cacheovací mechanismy, které dokáží snížit dobu odpovědi.
- Offline provoz - jak již bylo řečeno veškerý běh programu se vykonává u klienta, proto je možné v případě výpadku připojení k internetu pracovat s aplikací i v módu offline. Záleží na tom, jak je aplikace naprogramována a zda ukládá veškerý svůj stav u klienta.
- Instalace pluginu[1] - RIA technologie (kromě technologie AJAX) se vyznačují potřebou instalovat do prohlížeče zásuvné moduly pro svůj provoz. Instalace je většinou rychlá a mají velikost několik MB. Toto je vlastnost, která může bránit masové rozšiřitelnosti některých RIA technologií. Naproti tomu klasické technologie jako HTML/CSS/JavaScript běží v každém prohlížeči, což je jejich obrovská výhoda. Nevýhodou klasických webových aplikací je, že každý webový prohlížeč zobrazuje určité elementy stránky jinak a ne všechny prohlížeče podporují hned nejnovější standardy. Při programování je tedy nutné brát na to ohled a ladit kód tak, aby se zobrazil v co největším počtu prohlížečů. RIA aplikace se po nainstalování pluginu do prohlížeče zobrazují vždy jednotně a v tom mají výhodu [9].

## 4.3 Technologie

Technologií pro vývoj RIA aplikací je celá řada a v této kapitole jsou tři z nich stručně popsány. Mezi dnes nejpoužívanější technologie patří AJAX, Flash/Flex od společnosti

[1] Zásuvný modul, který se nainstaluje do webového prohlížeče za účelem fungování aplikace.

Adobe, Silverlight od společnosti Microsoft a JavaFX od společnosti Sun Microsystems. Další používané technologie jsou Google Gears[1], OpenLaszlo[2] a Oracle WebCenter[3]. Technologií pro vývoj je zkrátka spousta a je těžké je si vybrat tu správnou pro konkrétní projekt. Každá technologie má své pro a proti a musí se zhodnotit kritéria jako výkonová stránka, bezpečnost, rozšířenost mezi uživateli, funkčnost a robustnost technologie.

### 4.3.1 AJAX

AJAX (Asynchronní JavaScript a XML) je jednou z nejznámějších ani ne tak technologií, ale návrhovým vzorem, který se poprvé objevil v dubnu roku 2005. Jedná se o obecný koncept či návrhový vzor, protože AJAX není sám o sobě technologií či softwarovým produktem. Je postaven na dlouhodobě používaných a osvědčených technologiích na HTML, DHTML a JavaScriptu. Obrovským rozdílem proti ostatním RIA technologiím tedy je, že ho podporují všechny prohlížeče bez další instalace pluginu. Základní myšlenkou AJAXU je použití JavaSriptu k aktualizaci stránky bez potřeby jejího načítání. JavaScript tak může měnit podobu stránky či aktualizovat její část bez opakovaného načtení. Základním stavebním kamenem tohoto chování je objekt `XMLHttpRequest`, který umožňuje asynchronní volání serveru. Při změně stavu klienta se nová data pak přenášejí ze serveru ke klientovi na pozadí. Za nevýhodu AJAXU se pokládá nutnost mít zapnutou podporu JavaScriptu v prohlížeči. Bez zapnutého JavaScriptu aplikace nebude fungovat. Další nevýhodou je podpora různých verzí DHTML a JavaScriptu u různých prohlížečů. To pak způsobuje nekonzistentní chování v odlišných prohlížečích. Proto pokud je aplikace vyvíjena pro širokou veřejnost, je ji potřeba testovat na různých verzích prohlížečů a operačních systémů. Jako poslední nevýhodu může někdo považovat otázku bezpečnosti, tím je myšleno, že zdrojové kódy JavasSriptu jsou vždy volně dostupné [10].

### 4.3.2 Flash a Flex

Flash je produkt od společnosti Adobe a velký konkurent AJAXu. Ke svému provozu potřebuje plugin zvaný FlashPlayer. FlashPlayer původně určený k přehrávání videí se postupně vyvíjel a s každou novou verzí přinesl nové možnosti. V současné době je zhruba na 97% zařízení připojených k internetu Flash nainstalovaný a podporovaný. Výhoda pluginu FlashPlayer je, že si zachovává velmi malou velikost. Použitím Flashplayeru se tedy také dosáhlo kompatibility a stejné funkčnosti ve všech prohlížečích a na různých operačních systémech. Asi největší překážkou pro vývoj RIA aplikací bylo prostředí Editoru Flash. Ten byl zpočátku primárně určený k vývoji různých animací a svým neznámým prostředím spoustu vývojářů od vývoje RIA aplikací v tomto prostředí odlákal. Na tom se podílel i fakt, že existovalo málo materiálů pro výuku Flashe co by nástorje pro tvorbu RIA aplikací.

Pro aplikační logiku aplikací je použitelný pouze jeden jazyk a to plnohodnotně objektový ActionScript. Určitým zlepšením a zjednodušením pro vývoj aplikací pro FlashPlayer byl v roce 2004 příchod nového nástroje od Adobe - Flex. Z pohledu technologického je Flex velmi podobný architektuře AJAX a dokáže stejně jako ona aktualizovat uživatelské rozhraní, přijímat a odesílat data na pozadí za běhu aplikace. Dnes nástroj Flex poskytuje řadu vývojářských nástrojů a služeb, které velice usnadňují tvorbu

[1] http://gears.google.com/
[2] http://openLaszlo.org/
[3] http://www.oracle.com/

aplikací RIA[10]. Zahrnuje v sobě ActionScript, Flex Framewor - knihovna tříd, která umožňuje jednoduché využití nejlepších postupů při vývoji RIA aplikací, Flex Builder integrované vývojářské prostředí určené přímo pro vývoj RIA aplikací, Flex Data Services rozšiřují klientský Flex Framework a poskytují výkonné propojení s existující serverovou logikou a daty. Flash/Flax je tedy velice robustní a časem prověřené řešení, které je podle mě oproti ostatním RIA technologiím stále o krok napřed.

### 4.3.3 Silverlight

Silverlight je technologie od společnosti Microsoft určená pro tvorbu nejrůznějších RIA aplikací. Ke svému běhu potřebuje nainstalovat Silverlight plugin do prohlížeče. Pak podobně jako Flash běží na všech systémech a prohlížečích totožně. Na rozdíl od Flashe/Flexu Silverlight podporuje několik jazyků a to všechny které spadají pod .NET Framework. Další informace o technologie Silverlight jsou podrobně popsány v následující kapitole 5.

### 4.3.4 JavaFX

JavaFX je softwarová technologie postavená na bázi platformy Java od společnosti Sun MicroSystems, která byla představena v květnu roku 2007. Primárně je určená pro vývoj RIA aplikací a je navržená tak, aby ti co v ní vyvíjí, se soustředili více na kreativní část práce než na programování a aby vývoj byl snadný a rychlý. Podpora této technologie je velmi rozšířená díky tomu, že aplikace naprogramované v JavaFX běží na všech systémech, kde běží JRE[1] - Java Runtime Environment. JRE je běhové prostředí, které je v dnešní době oficiálně podporováno systéme Windows a MacOS a do budoucna se chystá i podpora pro Linux. Další výhodou je, že při programování aplikací může vývojář použít jakoukoliv třídu napsanou v Javě. Klíčové elementy technologie JavaFX jsou zveřejněny jako opensource, tak aby zákazníci společnosti Sun mohli technologii maximálně využívat. Vývoj pomocí této technologie je tedy velmi snadný a má také velkou podporu používání multimediálních prvků videa, audia a animací. Pro vývoj aplikací se používá vývojové prostředí NetBeans IDE 6.5 for JavaFX[2], skriptovací jazyk JavaFX, množství vývojových nástrojů grafických knihoven a audio/video knihoven [11].

# 5 Silverlight

Jako implementační technologii pro praktickou část mé bakalářské práce jsem si vybral platformu Silverlight. V následující kapitole se o ní čtenář dozví základní informace jako historie platformy, jednotlivé verze a bezpečnost. Není opomenuto ani srovnání s příbuznou technologií WPF a seznámení s editory ve kterých se Silverlight aplikace vyvíjejí.

[1] http://www.java.com/en/download/index.jsp
[2] http://netbeans.org/features/javafx/

## 5.1 Obecně o technologii

Silverlight je moderní technologie od firmy Microsoft pro tvorbu dynamického obsahu a práce s ním. Cílem této technologie je překlenout nevýhody Webu 1.0, kdy se snadno stalo, že designér navrhl krásně zpracovaný grafický design stránek, ale při konzultaci s programátorem stránek musel dost designových a efektivních prvků odstranit, protože to nebylo možné naimplementovat. Silverlight se snaží těmto problémům čelit a umožňuje implementovat jakkoliv efektní a detailní uživatelské prostředí je libo. Přímým a nejvíce srovnávaným konkurentem je například Adobe Flash/Flex probíraný v kapitole 4.3.2.

Jako většina technologií pro tvorbu RIA aplikací i Silverlight potřebuje ke svému běhu nainstalovat plugin do prohlížeče. Ten je dnes podporován většinou prohlížečů a operačních systémů a má velikost cca 4MB. Primárně je určen Silverlight pro operační systémy Windows a MacOS, avšak i pro linuxové systémy existuje varianta jak na něm Silverlight používat. Je to implementace Moonlight[1] od společnosti Novell po jejímž nainstalování běží Silverlight i na operačním systému linux.

V dnešní době, kdy mobilní zařízení disponují vysokým výkonem a funkcionalitou, je neméně důležité dostat nové technologie i do těchto oblastí, takže je vytvořena podpora Silverlightu i pro operační systém Windows Mobile. Základní princip technologie je vytváření uživatelského rozhraní pomocí značkovacího jazyka XAML, který je probrán v kapitole 5.2 a programování aplikační logiky v některém objektově orientovaném jazyce z .NET Frameworku.

## 5.2 XAML

XAML (Extensible Application Markup Language) používá Silverlight pro vytváření prezentačního rozhraní, jak bylo uvedeno v předešlé kapitole. V následujícím textu je jazyk XAML stručně popsán. Je to značkovací jazyk, který je pro definování vzhledu ideálním řešením. Rychle, stručně, přehledně a jednoduše nadefinuje vlastnosti, umístění a spojení jednotlivých elementů. Výhodou použití XAML je oddělení programové části od uživatelského rozhraní, což usnadňuje spolupráci designerům a programátorům. Protože tento jazyk je založený na značkovacím jazyku XML, tak striktně dodržuje syntaktická a sémantická pravidla XML dokumentu. Pro upřesnění, syntaxe XML je definovaná podle doporučení konsorcia W3C[2]. Tato doporučení definují, jak má být dokument formátovaný, definují syntaxi pro zápis jednotlivých elementů, jejich atributy, počáteční a koncové ohraničení a obsah prvků. Výhodou XML je její dobrá, díky pravidlům zápisu, strojová čitelnost. A tím, že je platformě nezávislá, tak dovoluje i technologii Silverlight platformovou nezávislost. Další výhodou je skutečnost, že XAML využívá textový formát založený na XML a tak dokáže snadno projít přes síťové firewally. Jiné technologie to tak snadné nemají, protože jejich soubory jsou v binární formě a mají proto větší problém s bezpečnostními kontrolami při zasílání obsahu klientovi. Nakonec bych dodal, že jazyk XAML není klíčovým pilířem jen pro platformu Silverlight, ale hlavně také pro WPF (pojem WPF je probrán v kapitole 5.4). Ukázka XAML kódu, který definuje vlastnosti elementu `TextBlock`, který je zanořený v elementu `Grid` je vidět v následující ukázce 5.1 [12].

[1] http://www.mono-project.com/Moonlight/

[2] W3C je mezinárodní konsorcium pro vyvíjení webových standardů.

```
<UserControl xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
x:Class="SilverlightApplication1.MainPage"
Height="480"
Width="640">
    <Grid Background="White" Name="Mrizka">
        <TextBlock Width="100" Height="20" Text="Oznam" Name="Pole" />
    </Grid>
</UserControl>
```

Ukázka 5.1: Příklad zápisu XAML který zobrazí textové pole (`TextBlock`) zanořené v mřížce (`Grid`) [12]

## 5.3 Historie

Silverlight je poměrně nová technologie a její historie je poměrně krátká. Tato technologie spatřila světlo světa v září roku 2007, kdy byl představen společností Microsoft ve finální verzi 1.0. Zároveň s touto verzí byla představena také verze 1.1, která byla o rok později, pro své velké pokroky od verze 1.0, přejmenována na verzi 2.0. Ve verzi 1.0 byl jako programovací jazyk pro aplikační logiku určen JavaScript, zatímco ve verzi 2.0 bylo možné v plné míře využít všechny jazky z prostředí .NET Framework[1]. V roce 2009 byla uvedena další verze pod označením 3.0 a v listopadu stejného roku přišla opět nová verze pod označením 4.0. Podrobnější informace o rozdílech mezi jednotlivými verzemi budou probrány v následujících podkapitolách. V roce 2009, tedy za 2 roky své existence, se vyšplhal počet klientských instalací na 350 milionů, přičemž počet instalací vzrůstá exponenciálně. V současné době je nainstalovaný zhruba na více než 45 % různých zařízeních, které jsou připojeny k internetu [12].

### 5.3.1 Verze 1.0

Architektura verze 1.0 byla rozdělena na dvě vrstvy. Prezentační vrstva používala komponenty určené ke generování uživatelského rozhraní a interakci s uživatelem. Uživatelské rozhraní zahrnovalo textový výstup, animace, multimediální formáty WMA,VC1, MP3 a renderování vektorové a bitmapové grafiky. Interakce s uživatelem byla zajištěna pomocí událostí generovaných klávesnicí a myší. Na programování aplikační logiky se používal jazyk JavaScript. Ten pracoval s objektovým modelem pracovní plochy zvaným `Canvas`. `Canvas` reprezentoval viditelnou prezentační vrstvu Silverlight aplikace. Napojení na zdroje údajů se zajišťovalo pomocí technologie ASP.NET[2] a AJAX, které podporují volání webových služeb z JavaScriptu s výměnou údajů ve formátu JSON [12].

### 5.3.2 Verze 2.0

Ve verzi 2.0 přibyly oproti předešlé verzi zásadní novinky. Asi největší změnou bylo zavedení vlastního .NET Frameworku, který je součástí zásuvného modulu Silverlight. To umožňovalo použití všech programovacích jazyků, které spadají pod .NET Framework. Nová verze přinesla také nové ovládací prvky, styly, šablony a DataBinding. DataBinding je jakési svázání vlastností. Jakmile se změní hodnota zdrojové vlastnosti, tak se změna projeví

[1] http://www.microsoft.com/net/
[2] http://www.asp.net/

i v provázaných vlastnostech a naopak. Novinkou jsou také nové přídavné moduly jako Isolate storage, který umožňuje uložit data u klienta v tzv. izolovaném úložišti, dále Asynchronous programming, File management, Serialization, Packaging a XML libraries [12].

### 5.3.3 Verze 3.0

Verze 3.0 vychází z předchozí verze 2.0 a je tedy zpětně kompatibilní což je důležité. Nová verze přinesla spoustu dalších novinek. Jednou z hlavních je režim běhu aplikace mimo prohlížeč tzv. OOB (Out Of Browser). Silverlight aplikace se nainstaluje do lokálního operačního systému, kde se chová jako klasická desktopová aplikace a je reprezentována zástupcem na ploše a v nabídce Start. Protože verze 3.0 je plnohodnotná technologie pro vývoj RIA, tak přinesla rozšířenou datovou podporu a podporu pro bussines objekty, které je pak možné jednoduše řadit filtrovat a stránkovat. Na straně klienta přibyl také nový objekt `CollectionView` a množina operací pro práci s údaji na serveru. Pro propojení údajů byl vylepšen databinding tzv. ElementNameBinding, který umožňuje propojení více prvků navzájem přímo v XAML kódu. Další nový prvek je `DataGrid`, který podporuje ověření aktualizaci a stránkování údajů. S příchodem verze 3.0 byly také zveřejněny zdrojové kódy všech ovládacích prvků a začal vznikat projekt Silverlight Toolkit, který v dnešní době obsahuje obrovské množství nových ovládacích prvků a šablon. Důležitým faktorem pro webové aplikace je jejich dostupnost. Klíčovým faktorem úspěchu v prakticky každé oblasti podnikání je, aby odkazy na webové stránky firmy byly na předních pozicích vyhledávačů. Silverlight verze 3.0 řeší tuto SEO[1] problematiku pomocí Deep Linking. Tato vlastnost dokáže převést obsah který se generuje z databáze na lehce indexovatelný HTML kód a tím umožní vyhledávačům zaindexovat obsah webových stránek. Pro dobrý dojem zobrazení obrázků s vysokým rozlišením přibyla Funkce DeepZoom a pro animace reálného pohybu podle fyzikálních zákonů přibyla funkce Animation Easing. Poslední novinkou, kterou verze 3.0 přináší je streamování videa v úplném rozlišení HD (720+)[2] kombinované s technologií Smooth Streaming, která přispívá k plynulosti přehrávání obrazu [12].

### 5.3.4 Verze 4.0

S novou verzí 4.0 přišlo opět spoustu novinek. Přibyla podpora tisku, která nebyla do této verze vůbec dostupná. Přibyly nové možnosti prvku `DataGrid` jako kopírování informací do schránky. Je vylepšená grafická stránka, která zajišťuje plynulý přechod prvků mezi jednotlivými stavy prvků, byla přidána podpora pro pravé tlačítko myši, byla přidána funkce drag and drop pro přemisťování prvků na ploše aplikace a byla přidána podpora pro webové kamery a mikrofony [12].

### 5.3.5 Verze 5.0

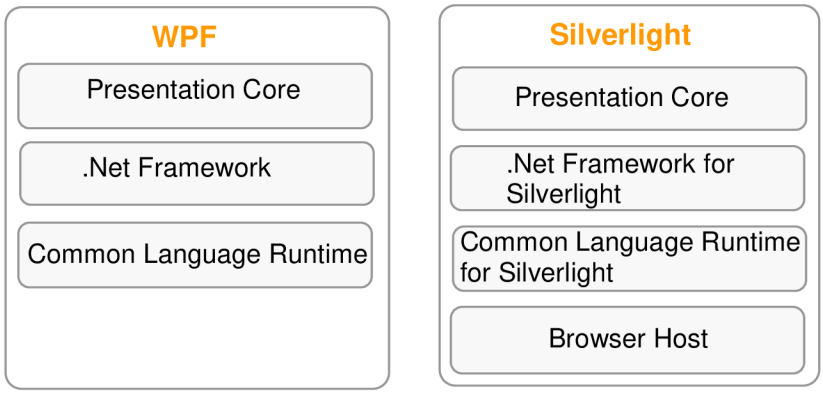
Jako zatím poslední verzi kterou Microsoft plánuje vydat je verze 5.0. Ta má přinést novinky jako vylepšení podpory 64-bitových systémů, lepší využití GPU pro renderování 2D a 3D obsahu a funkci TrickPlay, která umožní přehrávání videa v různých rychlostech. Nová verze

[1] SEO – optimalizace webových stránek pro vyhledávače.
[2] HD (720+) Standard pro video s rozlišením minimálně 1280x720 pixelů.

by měla mít také lepší možnosti pro dálkové ovládání a podporu tisku ve formátu jazyka PostSkript. Tato plánovaná verze by se měla objevit v průběhu roku 2011 [13].

## 5.4 Silverlight versus WPF

WPF - zkratka pro Windows Presentation Fundation je technologie od společnosti Microsoft pro vytváření prezentačního rozhraní. WPF je podmnožinou .NET Frameworku a používá značkovací jazyk XAML pro vytváření bohatého uživatelského prostředí. Díky použití jazyku XAML jsou od sebe odděleny funkčnost a vzhled aplikace. Technologie WPF je vestavěná v operačních systémech Windows Vista a Windows 7. WPF si klade za cíl sjednotit poutavé uživatelské rozhraní, rastrovou/vektorovou grafiku, 2D/3D grafiku, audio/video, animace a vázání dat. Silverlight je technologie, která se může zdát téměř stejná (používá XAML pro uživatelské rozhraní a libovolný .NET jazyk pro aplikační logiku), ale není tomu tak. Dalo by se říct, že do verze Silverlight 2.0 byl Silverlight určen jen pro webové aplikace a WPF pro klasické desktopové aplikace. Od verze Silverlight 3.0 tomu však tak není, protože vznikla podpora instalovat a spouštět aplikace i lokálně, stejně jako WPF. Rozdíl je tedy v tom, že zatímco WPF aplikace jdou spustit jen pod operačními systémy Windows s nainstalovanou podporou .NET Framework 3.0 a vyšší, tak Silverlight aplikace jdou spustit i na operačních systémech MacOs, případně Linux. Obecně může být řečeno, že Silverlight je jakási multiplatformní podmnožina WPF, z čehož vyplívají také jistá omezení. Další pohled na porovnání technologií ukazuje obrázek 5.1, kde můžeme vidět, že Silverlight má vlastní jádro .NET i vlastní běhové prostředí CLR [12].




Obrázek 5.1: Porovnání blokových schémat principu fungování WPF a Silverlight. Silverlight má vlastní jádro .NET i CLR [12]

## 5.5 Bezpečnost

U platformy Silverlight se předpokládá vykonávání plnohodnotného objektového kódu na straně klienta a v pozadí prohlížeče. Aby mohla být technologie používána v běžné praxi je nutné zajistit bezpečnost vykonávání klientského kódu, aby nebyla narušena u klientského počítače. Bezpečnost je zajištěna pomocí tzv. Sandboxu. Sandbox je obálka v které běží Silverlight aplikace a vývojáři nemají možnost se nijak dostat se svým kódem přes tuto obálku a ohrozit tak bezpečnost klientského počítače. I když Silverlight aplikace vyžaduje služby .Net Frameworku, běží ve speciálním odděleném frameworku obsaženém

zásuvném modulu Silverlight. Přestože daný zásuvný model má velikost cca 4MB, obsahuje veškerou potřebnou funkcionalitu. Další zabezpečení je, že v jádře platformy Silverlight je zabudováno speciální běhové prostředí, které se nazývá coreCLR. Každý blok kódu, který se zavádí přes toto coreCLR, získává jen částečná práva a nemá oprávnění volat metody vyžadující vyšší oprávnění. Silverlight využívá formu bezpečnosti založenou na třech atributech:

- `SecurityTransparent` - kód s částečnou důvěryhodností,
- `SecuritySafeCritical`- kód který je označený tímto atributem je důvěryhodný a může být volán jakýmkoliv transparentním kódem,
- `SecurityCritical` - takto označený kód má plnou důvěru jádra.

Každý z těchto atributů definuje určitou úroveň zabezpečení kódu. Rozdíl oproti .Net Frameworku je ten, že Silverlight coreCLR považuje za takový kód všechny zdrojové soubory, které nejsou označeny jiným atributem. To znamená i soubory bez označení bezpečnostní úrovně. Taková vlastnost znamená, že jakýkoliv kód Silverlight aplikace nemůže vykonávat jakékoliv operace nebo přistupovat k operačnímu systému [12].

## 5.6 Vývojová prostředí

Pro vývoj prezentačního a aplikačního rozhraní existují v zásadě dva nástroje: návrhové prostředí Microsoft Expression Blend a Visual Studio. Jejich funkcionalita se překrývá, protože jak v nejnovější verzi Expression Blendu 4, tak i v nejnovější verzi Visual studia 2010, se může navrhovat prezentační vrstva i aplikační logika. Tyhle dva nástroje jsou si tedy podobné a to z následujícího důvodu. V praxi designér navrhne v Expression Blend, které je pro navrhování prezentační vrstvy určené, design a prezentační vrstvu aplikace. Takto vytvořenou prezentační vrstvu odevzdá programátorovi pracujícímu ve Visual Studiu, které je vhodnější pro programování aplikační logiky, a ten naprogramuje funkčnost aplikace. Protože jak Expression Blend tak Visual Studio umí pracovat s návrhem prezentační vrstvy, tak bude finální podoba aplikace vypadat stejně, jak ji navrhl designér a nebude se lišit z důvodu, že programátor byl limitován možnostmi prezentační vrstvy v jeho návrhovém prostředí.

A nyní pár informací k samotným vývojovým prostředím. Visual studio jako nástroj pro vývoj Silverlight aplikací existuje ve dvou verzích. Pro práci v první verzi 2008 je do ní nutné doinstalovat Service Pack 1 for Visual Studio 2008 a Miscrosof Silverlight Tools 3.0. Po instalaci těchto částí je možné využívat Visual Studio 2008 pro vývoj Silverlight aplikací. Novější verze 2010 již v sobě obsahuje plnou podporu pro vývoj Silverlight aplikací a to i návrháře prezentačního rozhraní podobně jako Expression Blend. Expression Blend existuje dnes ve verzi 3 a nejnovější verzi 4. Je to prostředí určené pro návrh designu prezentačního rozhraní pomocí značkovacího jazyka XAML. Nástroj je určený pro technologie WPF a Silverlight. Oproti Visual Studiu má výhodu v dobře propracované správě šablon stylů a v lepších možnostech návrhu prezentační vrstvy. Součástí tohoto prostředí je také aplikace s názvem SketechFlow, která slouží k prototypování aplikace. To znamená, že než se začne aplikace vyvíjet, vytvoří se pomocí SketchFlow návrh prezentačního rozhraní a přechodů mezi jednotlivými obrazovkami pomocí dostupných prvků.

# 6 Rozbor zadání

V této kapitole se čtenář seznámí se zadáním praktické části mé bakalářské práce. Kapitola popisuje podrobně zadání a požadavky kladené na funkčnost aplikace. V textu je i zmínka o důvodu vybrání tohoto tématu.

## 6.1 Zadání práce

Cílem mé bakalářské práce bylo navrhnout mashup aplikaci pro plánování cest, ve které by se dala naplánovat vícedenní cesta, například pracovní. V aplikaci by mělo být možné navrhnout důležité události týkající se cesty a to možnost zaznamenat si dopravu či spojení mezi jednotlivými městy, vyhledat si ubytování v daném městě a naplánovat konkrétní činnosti na každý den. Takto kompletně naplánovaný výlet by měl mít možnost vytisknutí na papír v podobě přehledného plánu.

Toto téma bakalářské práce mě velmi zaujalo, proto jsem si je zvolil. Jednou z hlavních motivací bylo zaměření aplikace jako internetové aplikace a to je v oboru mého zájmu. Další důvod pro výběr byla skutečnost, že v prostředí českého internetu nevím o žádné aplikaci, která by umožňovala navrhnout takový kompletní plán vícedenní cesty.

Když vezmu v úvahu například zaměstnance, který má jet na pracovní cestu, s tím že má v daném cílovém městě několik úkolů v různé dny a musí si sám obstarat spojení a ubytování, tak zkompletování všech informací a dohledání je poměrně zdlouhavé. Nejdříve si pravděpodobně na nějakém portálu zjistí, kde dané město vůbec leží na mapě a jak je vzdálené. Potom začne dohledávat spojení do města a v průběhu si informace bude nějakým způsobem někam zaznamenávat. Pak pravděpodobně začne v daném městě vyhledávat pomocí různých webových portálů hotely a v neposlední řadě si udělá poznámky na jakou činnost nebo věc nesmí v průběhu výlet zapomenout. Zároveň se nezapomene podívat na další informační portál jaké bude v daný den počasí, aby věděl jaké oblečení si má na cestu sbalit. Takové plánování je podle mě tedy dosti zdlouhavé a vyžaduje navštívení několika webových portálů, kdežto moje aplikace by měla umožnit většinu těchto služeb a hlavně jejich přehledné zkompletování na jedné webové stránce. Z tohoto důvodu se domnívám, že aplikace by neměla být zbytečná a najde svoje využití.

## 6.2 Požadavky

Požadavek v rámci praktické části bakalářské práce byl tedy vytvoření aplikace pro plánování cest. Aplikaci bylo možné implementovat pomocí libovolné technologie. Já jsem si vybral pro implementaci platformu Silverlight a jazyk C#, s nimiž jsem sice neměl žádné zkušenosti, ale byly mi doporučeny a velmi mě zaujaly. Požadavky na funkčnost byly:

- Použití mapových podkladů pro vyhledávání měst - města, která byla plánovaná během cesty navštívit, by se měla zobrazovat na mapě popřípadě počítat jejich vzdálenosti. Pro tyto potřeby se musí použít vhodný zdroj s mapovými podklady.
- Schopnost aplikace vyhledávat hotely - aplikace by měla umět vyhledávat v různých městech hotely za použití vhodného externího zdroje.

- Možnost zaznamenávání spojů a dopravy - v aplikaci by mělo být možné zaznamenat si vyhledané spojení (spojení si uživatel vyhledá sám pomocí externí aplikace) mezi jednotlivými městy.
- Možnost zaznamenání plánovaných aktivit - aplikace by měla umožňovat zaznamenání aktivit či činností, které jsou plánované v průběhu vícedenní cesty.
- Vytisknutí plánu - všechny zaznamenané informace o cestě musejí mít možnost vytisknutí na papír v podobě přehledného a kompletního plánu vícedenní cesty.

V průběhu vývoje aplikace se požadavky na aplikaci také trochu měnily podle časových rezerv, které byly k dispozici. Hlavní požadavky, které jsou uvedeny v horním odstavci, byly však splněny. Více informací o dalších možných funkcích aplikace v kapitole 10.

# 7 Návrh

Tato kapitola popisuje návrh praktické části bakalářské práce. V rámci návrhu bylo potřeba zmapovat dostupné externí zdroje a vybrat ty nejvhodnější, promyslet kompletně funkcionalitu aplikace a navrhnout grafický vzhled prezentační vrstvy.

## 7.1 Výběr zdrojů

V této kapitole jsou uvedeny možné použitelné zdroje s krátkým popisem, proč jsou vhodné nebo naopak nevhodné pro mou aplikaci. Dostupných zdrojů existuje pro můj projekt celá řada. Jak jsem se jimi však probíral, postupně jsem zjišťoval, že není vůbec lehké najít zdroj, který bude odpovídat požadovaným podmínkám. To znamená, že nebude mít různá omezení, bude stabilní a hlavně dostupný zdarma. Dalším kritériem, které měly mé zdroje splňovat, bylo aby poskytovali data nejen z Evropy, ale z celého světa. Původně jsem zamýšlel, že aplikace bude schopná plánovat cesty jen po Evropě, později jsem toto stanovisko přehodnotil a chtěl rozšířit možnosti aplikace, aby podporovala plánování cest po celém světě. Následující část obsahuje zdroje, ze kterých jsem vybíral a které jsou rozdělené do tří kategorií podle dat, které jsem potřeboval získávat.

**Mapové podklady:**

- **Mapy.cz** - jsou volně dostupný mapový zdroj od české společnosti Seznam. Pro tuto možnost hovořila česká, pěkně zpracovaná dokumentace. Bohužel slabá stránka těchto map je funkčnost. Například zde není možnost vykreslení trasy z města A do města B. Další nevýhodou je, že tento mapový podklad je dostupný pouze pro Evropu a také by bylo komplikovanější zprovoznění na platformě Silverlight.
  Adresa API: http://api.mapy.cz/.
- **Amapy.cz** - jsou volně dostupný mapový zdroj od další české společnosti Atlas. Tento zdroj nabízí více funkcí než Mapy.cz, bohužel převažují nevýhody. První z nich je poskytování dat jen pro Evropu. Druhá je opět problematické začlenění zdroje do Silverlight aplikace. Největší překážkou však bylo přecházení API na novou verzi v době vypracovávání mého projektu, a tedy nebylo možné zvolit tuto možnost.
  Adresa API: http://amapy.centrum.cz/api-doc/.

- **Google Maps** - je mapový zdroj od společnosti Google, který je volně dostupný. Tento zdroj nabízí nepřeberné množství funkcí a nabízí v dnešní době nejvíce možností práce s mapou. Má rozsáhlou přehlednou dokumentaci, poskytuje celosvětová data a jako implementační jazyk se používá JavaScript, který není ideálním řešením pro implementaci zdroje do Silverlight aplikace.
  Adresa API: http://code.google.com/intl/cs/apis/maps/signup.html.
- **Bing Maps** - jsou posledním zdrojem a to od společnosti Microsoft. Bing Maps poskytují své API také zdarma a podobně jako Google Maps jsou vybavené velkým množstvím funkcí a poskytují celosvětová data. Bing Maps jsou dnes největším konkurentem pro Google Maps. Obrovskou výhodou pro Silverlight vývojáře je, že poskytují speciální knihovnu, díky které se Bing Maps můžou do aplikace implementovat v jakémkoliv jazyku podporovaném .NET Frameworkem.
  Adresa API: http://msdn.microsoft.com/en-us/library/dd877180.aspx/.

**Vyhledávání hotelů:**

- **Kayak.com** - je webový portál poskytující funkce pro vyhledávání hotelů, vyhledávání leteckých spojů a různých cestovních destinací. API s vyhledáváním hotelů bylo zdarma, poskytovalo celosvětové údaje a dávalo dobré výsledky vyhledávání hotelů i pro menší města. Bohužel těsně před zahájením implementace bylo API zablokováno kvůli častému zneužívání.
  Adresa API: http://www.kayak.com/labs/api/search/.
- **Tripadwisor.com** - je další webový portál, který poskytuje nejen funkce pro vyhledávání hotelů ale i restaurací, leteckých spojů, dovolených a další služby. Tento zdroj poskytuje opět celosvětová data. Co se týče vyhledávání hotelů, je to velice kvalitní zdroj a obrovskou výhodou je nepřeberné množství uživatelských recenzí hotelů z celého světa. Naproti tomu nevýhodou je, že API není zdarma, ale musí být vývojáři nejdříve schváleno. Protože se mně tento zdroj velice zamlouval, zažádal jsem o povolení přístupu k API v rámci bakalářského projektu, bohužel mi však nebylo vyhověno.
  Adresa API: http://www.tripadvisor.com/help/what_is_an_api/.
- **Hotwire.com** - je webový portál poskytující vyhledávání hotelů a dopravních spojů různého typu. Tento zdroj je volně dostupný, ale neposkytuje již tak kvalitní API s vyhledáváním hotelů jako uvedené zdroje. Vrací sice výsledky vyhledávání ve městech z celého světa, ale s menšími městy se nemůže počítat a výsledů vyhledávání je obecně málo. Další obrovskou nevýhodou je absence fotek u jednotlivých hotelů a absence alespoň stručného popisu hotelu.
  Adresa API: http://developer.hotwire.com/docs/.
- **Cleartrip.com** – je posledním uvažovaným zdrojem. Tento webový portál poskytuje celosvětové vyhledávání hotelů, dále pak vyhledávání leteckých a vlakových spojů. API je volně dostupné a vrací velké množství informací o hotelu, včetně detailů jako popis cesty k hotelu. Po stránce vyhledávání bohužel to tak slavné není. I přesto, že je zveřejněný seznam podporovaných měst po celém světě, tak API mnohdy na některá města nevrátí žádnou odpověď nebo celé spadne. Například v České republice podporuje API asi 15 měst a to není úplně nejhorší výsledek. Pro představu v Brně API najde 5 hotelů a v Praze cca 180.
  Adresa API: http://www.cleartrip.com/api/docs/hotel-api/.

**Předpověď počasí:**

- **Underground Weather** - pro předpověď jsem nevyhledával žádné speciální zdroje. Stačilo najít zdroj, který bude poskytovat zdarma celosvětově zhruba týdenní předpověď, kde bude informace o teplotě a jednoduchý popis počasí. Pro tento účel je tento zdroj ideální.
  Adresa API: http://wiki.wunderground.com/index.php/API_-_XML/.

Ze seznamu dostupných zdrojů pro každou část aplikace jsem vybral vždy jeden použitelný zdroj. V podstatě zvolení daného zdroje bylo ulehčeno tím, že ostatní popisované nebyly z nějakého důvodu příliš vhodné nebo nešli použít. Vybrané zdroje tedy jsou:

- mapové podklady: Bing Maps,
- vyhledávání hotelů: Cleartrip,
- předpověď počasí: Underground Weather.

## 7.2 Funkce aplikace

V této kapitole budou probrány všechny funkce a možnosti, které moje aplikace bude obsahovat. Program je navržen tak, aby jeho používání bylo co nejvíce intuitivní a snadné. To byl především úkol dobrého návrhu prezentační vrstvy, jejíž návrh je popsán v následující kapitole Prezentační vrstva. Původně měla aplikace zvládat podle mých představ více možností než jen vytvořit vícedenní cestu. Další možnosti měli být vytvoření jednodenního výletu nebo jen samostatné vyhledání hotelů. Nakonec po zvážení časových dispozic jsem se rozhodl, že bude obsahovat jen jedinou možnost z původně plánovaných a to naplánování vícedenní cesty.

Vytvoření takového plánu se bude dít v pěti po sobě jdoucích krocích, které na sebe budou logicky navazovat. V každém kroku bude pro případ pomoci podrobná nápověda. Uživatel se také bude moci vracet k jednotlivým krokům a zadané údaje upravovat, pokud se například spletl v zadávání údajů o cestě. Hlavním menu bude obsahovat ovládací prvky pro zobrazení údajů o aplikaci a možnost přepínání mezi českým a anglickým jazykem. Posloupnost úkonů a toku programu při vytvoření vícedenní cesty je zobrazena na obrázku 7.1. Následující text popisuje funkce a posloupnost vytváření plánu v jednotlivých krocích.

**První krok - termín výletu**

V prvním kroku uživatel vyplní název výletu a vybere se termín, na kdy je výlet plánován. Minimální délka výletu bude dva dny a maximální délka neomezená. Cestu bude vždy možné naplánovat od následujícího dne. Poté se přejde na druhý krok.

**Druhý krok - cíle cesty**

V tomto kroku uživatel vybere město, ze kterého bude začínat cestu a vybere cílové popřípadě cílová města, kam plánuje cestovat. Cílové město musí být vždy alespoň jedno a jejich konečný počet není omezen. V tomto kroku je obsažena mapa a při zadání každého města se město vyhledá a označí na mapě. Pokud se dané město se na mapě neoznačí, bude to znamenat, že dané město neexistuje a uživatel bude vyzván k opětovnému zadání města.

K takovým chybám by však nemělo docházet, protože zdroj Bing Maps je velice kvalitně zpracovaný a vyhledá snad všechna města a jiné zeměpisné body po celém světě.

Když bude již přidáno v seznamu více jak jedno město, tak se automaticky, je-li to možné, vyhledá a vyznačí trasa mezi městy po silnici a zároveň se spočítá vzdálenost mezi městy a celková vzdálenost výletu. Pokud se bude město nacházet například na jiném kontinentě, kam se nelze dostat po silnici, vykreslí se vzdušná spojnice a vypočítá vzdušná vzdálenost mezi městy. Uživatel dále bude mít možnost měnit pořadí měst v seznamu, popřípadě je vymazat. Po dokončení výběru měst se přejde na krok třetí.

**Třetí krok - ubytování**

V třetím kroku uživatel určí termíny návštěvy jednotlivých měst. V případě, že bude označen pobyt ve městě delší jak dva dny, automaticky se nabídne (je-li město podporované pro vyhledání hotelů) seznam volných hotelů s podrobnostmi na daný termín. Zobrazený seznam bude pak možno třídit podle různých kritérií. Uživatel bude mít také možnost změnit kritérium vyhledávání a to počet osob na pokoj. Ten bude omezen v rozmezí 1 až 4 osoby. Pokud se uživatelovi zalíbí nějaký hotel, bude mít možnost si o něm zobrazit detaily. Okno s detailem hotelu bude obsahovat následující informace:

- název hotelu,
- počet hvězdiček,
- popis hotelu,
- dostupné služby v hotelu,
- ceny za jednu noc a za osobu v Eurech,
- celkový počet podlaží a počet pokojů v hotelu,
- adresu hotelu,
- foto-galerii hotelu,
- popis cesty k hotelu,
- zobrazení hotelu na mapě.

Výběr konkrétního hotelu pak uživatel potvrdí speciálním tlačítkem. Po dokončení výběru termínu a hotelu ve městech se přejde na následující čtvrtý krok.

**Čtvrtý krok – spojení doprava**

Ve čtvrtém kroku si uživatel zaznamená způsob dopravy mezi jednotlivými městy. Pomocí nějakého externího vyhledávače spojů (např. Idos[1] jízdní řády, který vyhledává vlakové autobusové a letecké spoje pro Českou republiku) si vyhledá informace o spoji a poté vyplní připravený formulář o spojení obsahující pole pro zadání informací odkud a kam spoj jede, čas odjezdu a příjezdu spoje, druh dopravy a další podrobnosti. Pokud neexistuje přímý spoj do města ale jen s přestupem, tak bude možné přidat další spojení. Při uložení každého spoje se na mapě, která bude součástí čtvrtého kroku, zobrazí startovní a cílové město spoje, vykreslí se mezi nimi spojnice a zobrazí přibližná vzdálenost. Uživatel bude mít také možnost zvolit si, zda chce nebo nechce plánovat i zpáteční cestu. Po vyplnění všech potřebných informací o dopravě se přejde na pátý krok.

[1] http://jizdnirady.idnes.cz/

**Pátý krok – denní plán**

Tento krok bude určený pro zaznamenání činností nebo aktivit, které bude chtít uživatel na cestě v daný den vykonávat. Pomocí formuláře uživatel vyplní název aktivity, časové údaje a další podrobnosti. Uživatel si takových aktivit bude moci naplánovat na jeden den libovolné množství. Zároveň s vyplňováním aktivit na konkrétní den se vyhledá předpověď počasí ve městě, kde se bude uživatel v ten den nacházet. Důvodem zobrazování předpovědi počasí v tomto kroku je, aby si uživatel mohl například přizpůsobit podle ní svoje aktivity či promyslet jaké si sbalí oblečení. Informace o předpovědi počasí se bude zobrazovat pokud návštěva konkrétního města je v den, který je maximálně šestým dnem od dne plánování výletu. Předpověď je tedy omezena na šest dní dopředu a bude zobrazovat informace o minimální a maximální teplotě v konkrétní den a několik různých stavů počasí. Po dokončení tohoto kroku se přejde na poslední výsledný krok.

**Výsledný plán**

Po dokončení pátého kroku se ze všech vyhledaných a vyplněných údajů vytvoří přehledný plán. Uživatel si ho překontroluje a v případě nalezení chyby se bude moci vrátit na jakýkoliv předchozí krok a učinit úpravy zadaných údajů. Pak-li že je s plánem spokojen, bude moci udělat poslední krok a to vytisknutí plánu na papír pomocí speciálního tlačítka. Kompletní výsledný plán bude obsahovat následující informace.

Obecné informace:

- název vícedenní cesty,
- termín vícedenní cesty,
- trasu v podobě seznamu měst, jak je uživatel bude postupně navštěvovat,
- celkovou orientační vzdálenost cesty v kilometrech.

Informace o spojení:

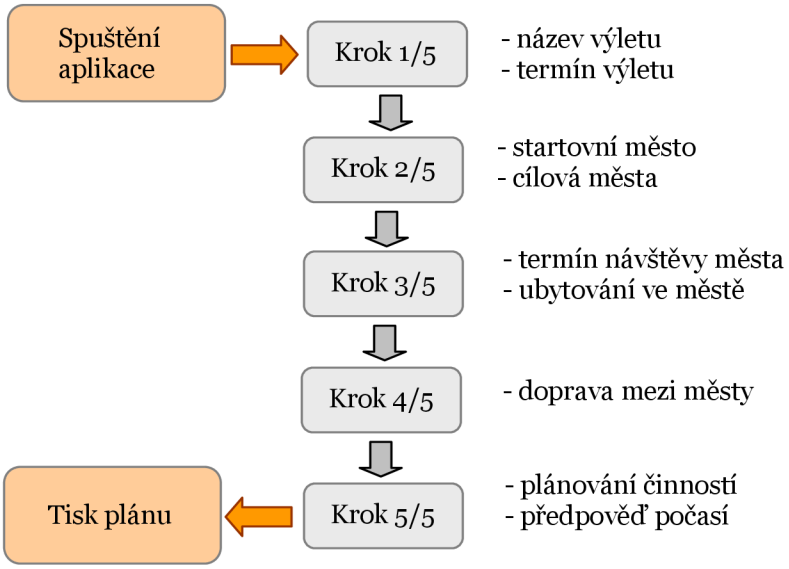
- z kterého města do kterého města se bude cestovat,
- datum na které je cesta naplánovaná,
- jednotlivé spoje, které budou obsahovat informace o typu dopravy, o startovním a cílovém městě, o času příjezdu a odjezdu spoje, číslo nástupiště, číslo spoje a poznámku o spoji.

Informace o ubytování:

- seznam hotelů, kde bude uživatel ubytovaný,
- informace o každém hotelu budou obsahovat jméno hotelu, počet hvězdiček, výpis všech služeb poskytovaných v hotelu, cenu za noc a za osobu, adresu hotelu.

Denní plán:

- bude obsahovat informace o každém dnu,
- tyto informace budou pro každý den popisovat jmenovitě co je za den v týdnu, kde se bude uživatel v daný den nacházet, počasí ve městě (minimální a maximální teplotu a obrázek ze stavem počasí), jaké aktivity jsou naplánovány, přičemž každá aktivita bude mít o sobě informaci jako typ aktivity, její název, čas vymezený pro aktivitu a poznámku.



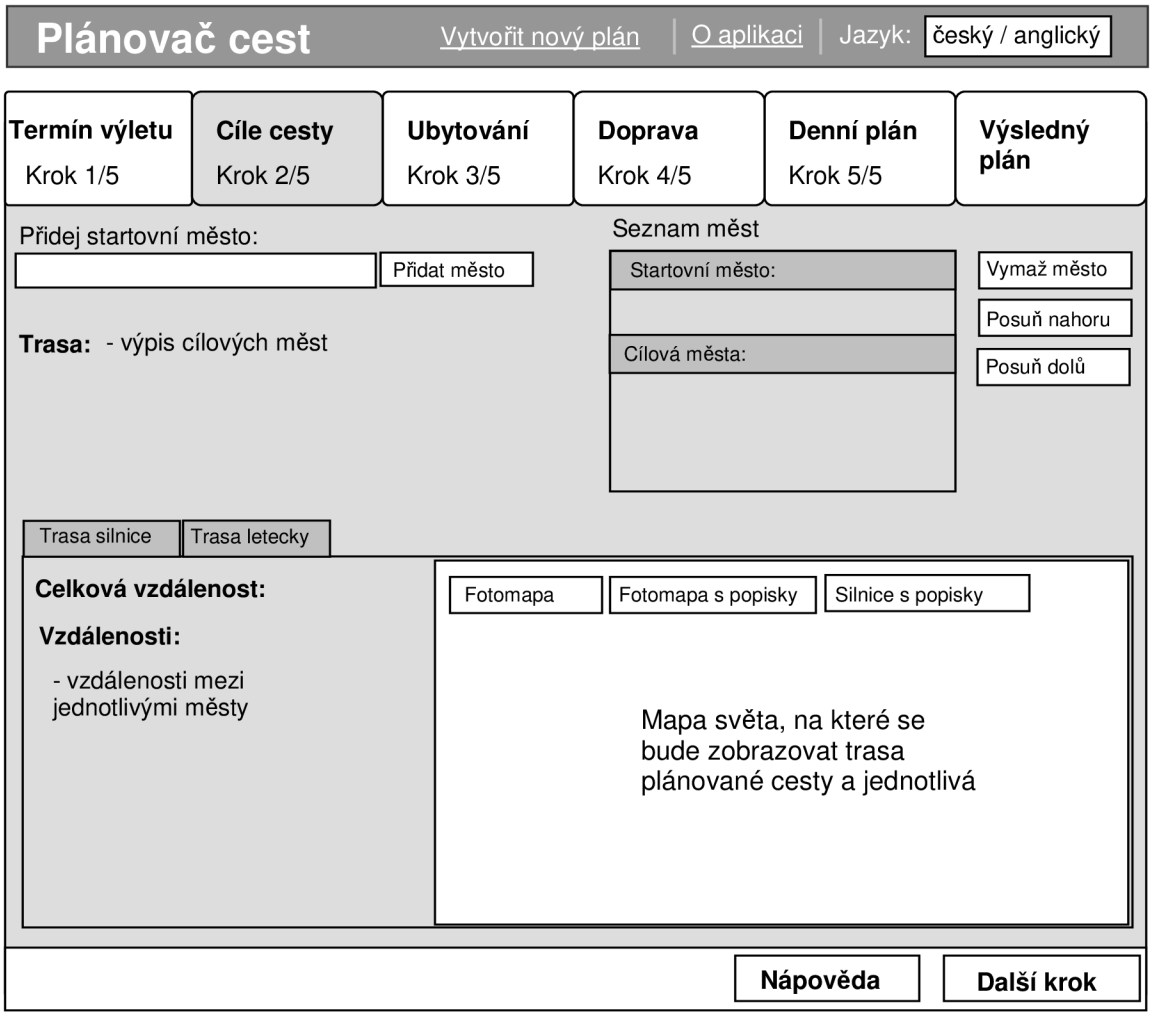

Obrázek 7.1: Posloupnost toku aplikace při vytváření plánu vícedenní cesty uživatelem

## 7.3 Prezentační vrstva

Prezentační vrstvu jsem se snažil navrhnout s ohledem na to, aby používání aplikace bylo co nejvíce intuitivní a pohodlné. Jak bylo popsáno v předchozí kapitole, vytvoření plánu bude probíhat v pěti krocích a každý krok bude mít svoje specifické grafické ztvárnění a použití vhodných ovládacích prvků. Pro každý takový krok bude použita jedna záložka a tento systém záložek bude tvořit hlavní vzhled aplikace. Aby byla zajištěna jakási konzistence designu mezi jednotlivými kroky aplikace, bude každý krok obsahovat ve své spodní části lištu s tlačítkem nápovědy a tlačítko na přechod na další krok. Bude mít také hlavní menu obsahující tlačítko pro zobrazení informací o aplikaci, tlačítko pro vytvoření nového plánu a ovládací prvek pro přepínání jazyků mezi češtinou a angličtinou. Toto menu bude dostupné po celou dobu spuštění aplikace.

Při návrhu bylo potřeba také promyslet, jaké navrhnout rozměry aplikace pro její pohodlné zobrazení na většině monitorů. Dnes jsou třemi nejpoužívanějšími rozlišeními podle statistiky portálu Navrcholu.cz[1] rozlišení 1280x1024px s poměrem stran 5:4, 1024x768px s poměrem stran 4:3 a 1600x1200px s poměrem stran 4:3. Podle těchto dostupných údajů jsem navrhl šířku mé aplikace 1024px a výška aplikace se bude měnit v závislosti na kroku, se kterým se bude právě pracovat v rozmezí 800px - 1024px . Takovéto navržené rozlišení se bude, co se šířky týče, na většině monitorů pohodlně zobrazovat. Pokud horizontální rozlišení monitoru nebude dosahovat alespoň šířky 1024px, tak se zobrazí horizontální posuvníky. Vertikální posuvníky budou automatické podle aktuální velikosti aplikace na výšku a použitého rozlišení monitoru. V rámci vylepšení designu a ovládání aplikace bude vytvořeno několik grafických prvků. Pro ovládání aplikace bude sloužit myš a klávesnice. Na obrázku 7.2 je zobrazena ukázka návrhu prezentační vrstvy pro krok 2. Pro další kroky se bude měnit obsah tabů pro jednotlivé kroky, jinak design zůstane stejný.

[1] http://www.iinfo.cz/tiskove-centrum/tiskove-zpravy/navrcholu-obrazovky-2009/




Obrázek 7.2: Ukázka návrhu prezentační vrstvy pro krok 2/5

# 8 Implementace

Tato kapitola popisuje nejdůležitější fázi vývoje aplikace a tou je implementace. Následující text je rozdělen do několika podkapitol. První popisuje stručně strukturu aplikace a popis nejdůležitějších tříd, druhá se věnuje popisu implementace, zejména jak aplikace funguje. Další podkapitola nabízí ukázku grafického designu. V poslední kapitole se čtenář dozví, jaké nastaly problémy při implementaci.

## 8.1 Technologie a nástroje

Jak bylo uvedeno v dřívějších kapitolách, pro vývoj aplikace jsem si zvolil platformu Silverlight. Tato platforma podporuje všechny .NET jazyky a já jsem si zvolil konkrétně jazyk C#. Jak Silverlight, tak i jazyk C# byly pro mě v době výběru tohoto tématu novinkou a výsledná aplikace je první větší projekt, který jsem v nich vyvinul. V době vývoje aplikace jsem používal nejnovější dostupnou verzi Silverlight 4 a .NET Framework 4. Jako nástroj

pro vývoj jsem používal Microsoft Visual Studio 2010 s plnou podporou vývoje Silverlight. Používal jsem verzi Visual Studio staženou ze školních stránek[1].

## 8.2 Struktura programu

Aby aplikaci bylo možné dobře vyvíjet a popřípadě v budoucnu rozšiřovat, bylo potřeba navhrnout dobrou strukturu programu. Pro tento účel dobře posloužil objektově orientovaný jazyk C#. Program je rozdělěn přehledně do zhruba třiceti souborů, které tvoří jednotlivé třídy, prezentační vrstvu a v neposlední řadě například vlastní WCF službu, či lokalizační soubory. Aplikaci jsem rozdělil na tzv. hlavní a tzv. vedlejší třídy. Celkem obsahuje 7 hlavních a 7 vedlejších tříd. Hlavní třídy tvoří kostru programu a jsou určené k vytvoření objektů, které poté slouží pro uložení všech údajů, během vytváření vícedenního plánu uživatelem. Seznam hlavních tříd a všech jejich atributů je na obrázku 8.1.
Popis hlavních tříd:

- **`Trida_vylet`** - objekt této třídy se vytvoří při inicializaci aplikace a je to objekt, do kterého se ukládají kompletně všechny informace zadané uživatelem během vytváření vícedenního plánu. Tento objekt je tvořen několika vlastnostmi, do kterých se ukládá název výletu, termín výletu, celková vzdálenost cesty (autem a letecky). Pak je tvořen čtyřmi seznamy jejichž typem jsou třídy, které jsou popsány v následujících čtyřech odrážkách. Jsou to seznamy s informacemi o cílových městech, o dopravě mezi městy, o hotelech a o jednotlivých dnech. Implementace této třídy je v ukázce 8.1.
    - **`Trida_cilovaMesta`** - objekt této třídy slouží pro uložení informací jednoho konkrétního města zahrnutého v plánu cesty. Vlastnosti této třídy jsou: název města, stát, poloha města (latitude, longitude[2]), zda-li je město v databázi měst pro vyhledávání hotelů a další potřebné údaje o městě.
    - **`Trida_hotel`** - slouží pro uchování informací o hotelu. Objekt této třídy má vlastnosti uchovávající: název hotelu, počet hvězdiček, adresu hotelu, adresu fotek hotelu a další informace.
    - **`Trida_doprava`** - slouží pro uchování informací o spojích mezi jednotlivými městy. Vlastnosti této třídy jsou: název měst odkud a kam spoj jede, čas odjezdu a příjezdu spoje, nástupiště, číslo spoje a další informace.
    - **`Trida_den`** - objekt této třídy v sobě uchovává informace o konkrétním dnu výletu. Vlastnosti této třídy jsou: datum dne, informace, zda-li uživatel bude v ten den spát v hotelu, v jakém hotelu bude spát a v jakých městech se bude nacházet. Dalšími vlastnostmi jsou dva seznamy jejichž typy jsou třídy, které jsou popsány v následujících dvou odrážkách. Jsou to seznamy uchovávající informace o předpovědi počasí v ten den

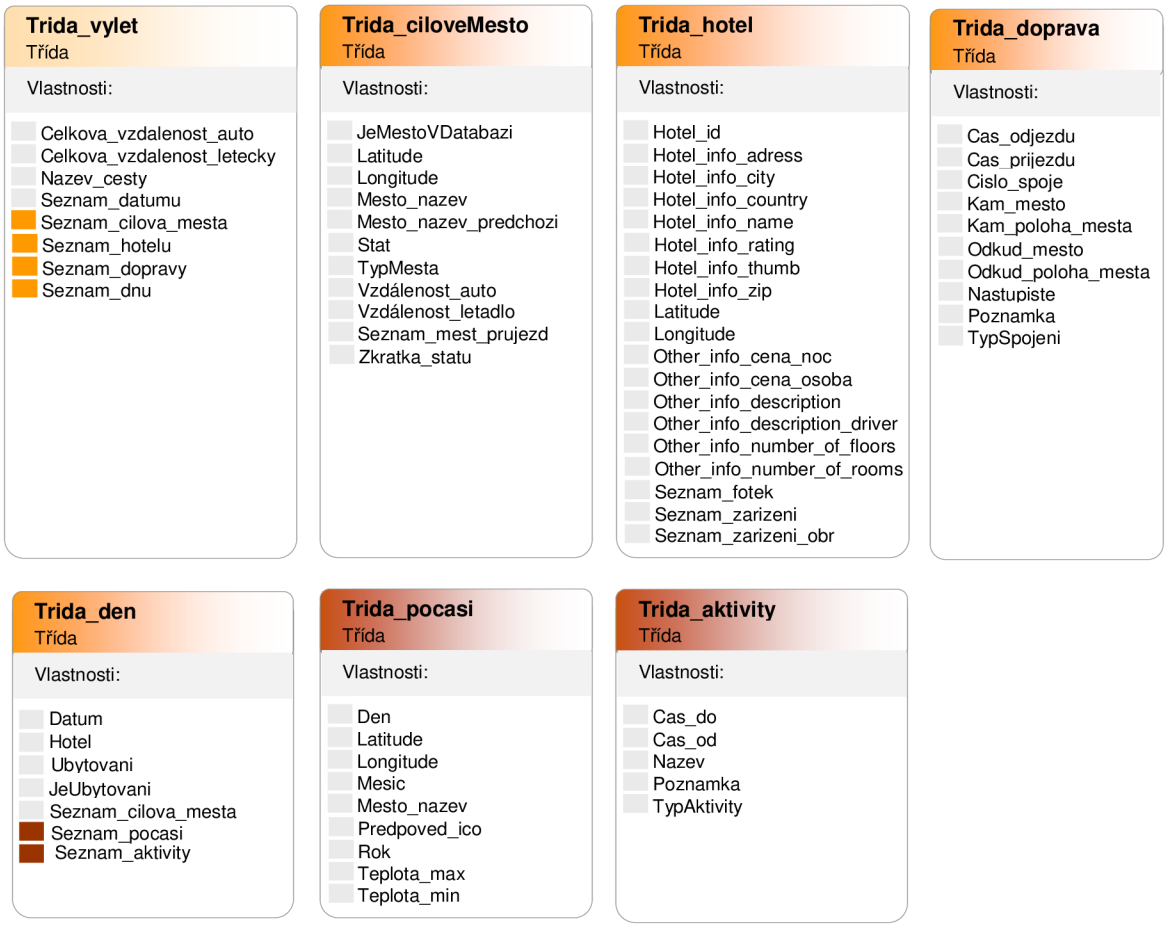
[1] http://msdn61.e-academy.com/elms/Storefront/Home.aspx?campus=vut_fit
[2] http://en.wikipedia.org/wiki/Geographic_coordinate_system

a o aktivitách plánovaných na konkrétní den. Implementace této třídy je v ukázce 8.1.

- **`Trida_pocasi`** - slouží pro uložení informací o počasí na daný den v konkrétním městě. Vlastnosti třídy reprezentují maximální a minimální teplotu, stav počasí a další podrobnosti.
- **`Trida_aktivity`** - je poslední z hlavních tříd aplikace a reprezentuje konkrétní aktivitu v daný den. Objekt třídy má vlastnosti reprezentující název, typ, časové rozmezí aktivity a případnou poznámku.

Všechny vedlejší třídy jsou určené jako pomocné a vykonávají různé činnosti nutné pro běh aplikace. Následující odrážky obsahují seznam a stručný popis vedlejších tříd:

- **`Trida_mesta`** - zpracovává a ukládá soubor ze seznamem měst, které jsou určené pro vyhledávání hotelů,
- **`Trida_hotely`** - tato třída se stará o zpracování a uložení odpovědi ze seznamem hotelů, popřípadě detailem hotelu,
- **`Trida_ciloveMesta`** - provádí operace s cílovými městy,
- **`Trida_pocasiPars`** - třída se stará o zpracování a uložení odpovědi s předpovědí počasí,
- **`Trida_pomocna`** - obsahuje pomocné metody, které vyžaduje aplikace pro svůj chod,
- **`Trida_konvertory`** - obsahuje metody konvertující data před zobrazením v prezentační vrstvě,
- **`Trida_lokalizace`** - stará se o změnu jazyka v aplikaci.




Obrázek 8.1: Seznam hlavních tříd tvořící kostru programu a určených pro uložení veškerých údajů zadaných uživatelem během vytváření plánu vícedenní cesty

```
public class Trida_vylet
{
 public string Nazev_cesty { get; set; }                //nazev vyletu
 public string Celkova_vzdalenost_auto { get; set; }     //celkova vzdalenost autem
 public string Celkova_vzdalenost_letecky { get; set; }  //celkova vzdalenost letecky
 public ObservableCollection<DateTime> Seznam_datumu { get; set; }               //termin vyletu
 public ObservableCollection<Trida_ciloveMesto> Seznam_cilova_mesta { get; set; } //cilova mesta
 public ObservableCollection<Trida_hotel> Seznam_hotelu { get; set; }            //ulozené hotely
 public ObservableCollection<Trida_pomDoprava> Seznam_dopravy { get; set; } //naplanovana spojeni
 public ObservableCollection<Trida_den> Seznam_dnu { get; set; }           //jednotlive dny cesty
}

public class Trida_den
{
  public DateTime Datum {get; set;}           //datum konkretniho dne
  public bool JeUbytovani { get; set; }       //indikace, zdali ve meste bude uzivatel spat
  public Trida_hotel Hotel { get; set; }      //informace o hotelu kde bude uzivatel spat
  public ObservableCollection<Trida_ciloveMesto> Seznam_cilova_mesta { get; set; } //cilova mesta
  public ObservableCollection<Trida_pocasi> Seznam_pocasi { get; set; }        //pocasí tento den
  public ObservableCollection<Trida_aktivity> Seznam_aktivit { get; set; }   //aktivity tento den
}
```

Ukázka 8.1: Implementace dvou tříd `Trida_vylet` a v ní použité třídy `Trida_den`, která je typ kolekce `ObservableCollection`, v které jsou uloženy informace o konkrétním dnu

## 8.3 Popis implementace

Následující text popisuje, jak jsou implementovány některé důležité části aplikace a jak aplikace pracuje. První část popisuje implementaci a využití zdrojů, druhá popisuje stručnou implementaci jednotlivých kroků aplikace. V rámci implementace jsou v aplikaci ověřeny všechny možné výjimky a kontroly údajů.

### 8.3.1 Práce se zdroji

Hlavním pilířem aplikace je použití externích zdrojů. Na tyto zdroje je nutné se nějakým způsobem napojit, posílat na ně dotazy a přijímat odpovědi. V mé aplikaci využívám konkrétně 4 služby a v následujícím textu bude popsáno jaká technika je použita pro jejich využívání.

#### Mapové podklady

Tento zdroj má specifickou techniku pro používání. Jak bylo popsáno v kapitole 7.1 pro mapové podklady jsem si vybral Bing Maps. Hlavním důvodem bylo jejich snadné propojení s platformou Silverlight. Microsoft pro tento účel vytvořil sadu knihoven Bing Maps Silverlight Contorl v1.0, která se použije na zaimplementování mapových podkladů do aplikace. Postup je takový, že nejprve se stáhnou příslušné knihovny[1]:

- Microsoft.Maps.MapControl.dll,
- Microsoft.Maps.MapControl.Common.dll.

Poté se na ně v projektu přidají reference. Posledním krokem pro zobrazení map v projektu je vložení prvku s mapou do kódu XAML. Po vložení kódu `<m:Map CredentialsProvider="<API_KEY>"></m:Map>`, kde místo hesla `API_KEY` se vloží přidělený vývojářský klíč, se zobrazí základní mapa. Díky knihovnám, na které je v projektu odkaz, je pak možné provádět další operace s mapou například přibližování, vykreslování bodů či spojnic mezi body a mnoho dalších operací. Vše se pak programuje již v jazyce C#.

#### Vyhledávání měst

Pro tento účel jsem použil další službu od Bing a to Geocode Service. Tato služba dokáže vyhledat například města, ulice či jiné zeměpisné objekty na celém světě a vrátit jejich polohu v podobě zeměpisné šířky a délky. Tato služba je založena na technologii SOAP [14]. Aby bylo možné tuto službu využívat, je na ní třeba přidat referenci do projektu. Adresa služby Geocode Service je:
http://dev.virtualearth.net/webservices/v1/geocodeservice/geocodeservice.svc?wsdl.

Po přidání reference je v tomto stavu možné začít službu využívat. Pro zaslání dotazu je nutné vytvořit speciální dotaz `GeocodeRequest`, kterému nastavíme atributy podle toho, co požadujeme nalézt. Jedním z atributů je i `Credential.ApplicationId`, kam je nutné uložit přidělený vývojářský klíč. Když je vytvořená odpověď, tak ji zašleme na odkazovanou službu klientem vytvořeným pomocí služby Geocode Service. Tento klient zavolá asynchronně službu Geocode Service a ta po nějaké době vrátí odpověď ve speciálním formátu `GeocodeResponse` s polohou bodu zadáného v dotazu. Samozřejmě ne vždy je

[1] http://www.microsoft.com/downloads/en/details.aspx?displaylang=en&FamilyID=beb29d27-6f0c-494f-b028-1e0e3187e830/

objekt nalezen, a proto je nutné testovat přijatou odpověď pomocí speciálních parametrů v odpovědi.

### Vyhledávání cesty

V projektu jsem potřeboval službu pro vyhledání trasy po silničních komunikacích, je-li to možné. K tomu mi posloužila další služba od Bing a to Route Service. Tato služba je založena na technologii SOAP [14] a pracuje na stejném principu jako předchozí služba Geocode Service. Adresa této služby je:
http://dev.virtualearth.net/webservices/v1/routeservice/routeservice.svc?wsdl.

Po přidání reference je možné službu začít využívat. Podobně jako v předchozím případě existuje speciální formát dotazu a to `RouteRequset`. Do tohoto dotazu je nutné zadat minimálně dva body v podobě zeměpisné šířky a délky, které reprezentují startovní a cílový bod. Jedním z parametrů je opět přidělený vývojářský klíč. Kompletní dotaz se zašle pomocí speciálního klienta vytvořeného pomocí služby Route Service. Klient zavolá asynchronně službu Route Service a ta vrátí v určitém čase odpověď. Odpověď má speciální formát `RouteResponse` obsahující seznam silničních bodů mezi startovním a cílovým bodem s jejich zeměpisnou polohou. U této odpovědi je naprosto nutné kontrolovat příznaky odpovědi, protože cesta po silnici často nebývá nalezena. Stává se tak v případech když jsou body od sebe moc vzdálené například více jak 5000Km nebo jsou odděleny vodní plochou.

### Vyhledávání hotelů

Pro vyhledávání hotelů využívám službu portálu Cleartrip. Služba je určena pro použití pomocí technologie RESTful, která vrací odpovědi ve formátu XML. Standardní způsob použití je vytvoření klienta pomocí speciální třídy `WebClient` volajícího asynchronně službu reprezentovanou jedinečným identifikátorem `URI` [14]. Příkladem `URI` pro dotaz na hotely v Praze s jedním jednolůžkovým pokojem na termín 25.4. - 26.4.2011 může mít například podobu:
http://api.staging.cleartrip.com/hotels/1.0/search?check-in=2011-04-25&check-out=2011-04-27&no-of-rooms=1&adults-per-room=1&city=Prague&country=CZ.

API Cleartrip vyžaduje použití vývojářského klíče, který je nutné umístit jako hodnota do HTTP hlavičky každého dotazu ve formátu `"X-CT-API-KEY:<vyvojarsky_klic>"`. Poté se může zavolat služba vracející po určitém čase XML odpověď. Bohužel na platformě Silverlight není žádná možnost jak vytvořit vlastní HTTP hlavičku v dotazu ve tvaru `X-CT-API-KEY`. Jediným řešením, jak tuto vlastnost obejít, bylo vytvoření vlastní WCF webové služby obsahující ono volání služby za pomocí klienta vytvořeného třídou `WebClient`. Protože WCF služba už nemá žádné dočinění s platformou Silverlight, tak v ní jde pohodlně vytvořit požadovaná HTTP hlavička s vývojářským klíčem pomocí kódu:

```
WebClient_1.Headers.Add("X-CT-API-KEY", "<vyvojarsky_klic>");.
```

Tuto vlastní WCF službu používám obdobně jako předchozí popsané. Vytvořím na ni referenci v projektu a pomocí klienta asynchronně volám službu s parametrem, který zastupuje identifikátor `URI`. Moje webová služba tento parametr převezme a zašle asynchroně dotaz na Cleartrip API. API vrátí za určitý čas XML odpověď mojí webové službě a ta přepošle tuto XML odpověď ve formátu `string` do mé aplikace. Úplně stejným způsobem se vyhledávají případné detaily o jednotlivých konkrétních hotelech.

#### Předpověď počasí

Posledním externím zdrojem, se kterým bylo potřeba pracovat, byl Underground Weather pro zjištění předpovědi počasí. Použití API je velice jednoduché a je určeno pro komunikaci pomocí technologie RESTful [14]. Stejně jako v předchozím případě se použije klient vytvořený třídou `WebClient` volající službu reprezentovanou jedinečným identifikátorem `URI`. Jediný rozdíl je, že API nevyžaduje žádný vývojářský klíč. Příklad `URI` dotazujícího se na předpověď počasí pro město Praha je:
http://api.wunderground.com/auto/wui/geo/ForecastXML/index.xml?query=50.07,14.4.

Prahu v tomto případě reprezentuje poloha zastoupená zeměpisnou šířkou(50,07°) a délkou (14,4°). API nabízí i jiné možnosti jak město reprezentovat, například adresou. Já jsem si však v mém projektu vybral uvedený způsob. Po zaslání dotazu služba vrací odpověď ve formátu XML s předpovědí počasí na 6 dní dopředu.

### 8.3.2 Stručný popis implementace

V předchozí podkapitole byla probrána práce s externími zdroji, v následujícím textu je stručně popsána implementace aplikace po jednotlivých fázích jak se s programem pracuje.

#### Spuštění aplikace

Po spuštění aplikace se zavolá hlavní třída `MainPage` a nainicializují se komponenty prezentační vrstvy. Zároveň proběhne deklarace proměnných a objektů potřebných pro běh aplikace. Při startu se nastaví česká lokalizace. Pro českou lokalizaci slouží soubor s přeloženými texty Strings.cs.resX a pro anglickou verzi Strings.en.rexX. Při prvním spuštění se vytvoří a nainicializuje globální objekt hlavní třídy `Trida_vylet` sloužící pro následné ukládání veškerých informací o vícedenní cestě

#### První krok - termín výletu

Po těchto úkonech se v prvním kroku aplikace provede inicializace objektu třídy `Trida_mesta` starajícího se o načtení měst podporovaných ve vyhledávání hotelů. Objekt rozparsuje textový soubor obsahující cca 13000 položek a uloží položky do seznamu, sloužícího jako datový zdroj pro `AutocompletBox` (textové pole našeptávající názvy měst při postupném zadávání názvu města) ve druhém kroku aplikace. V prvním kroku se dále uloží název výletu a setříděný seznam dnů, které bude uživatel trávit na výletě, do objektu třídy `Trida_vylet`.

#### Druhý krok – cíle cesty

V tomto kroku se nainicializuje objekt třídy `Trida_cilovaMesta` starající se o uložení přidaných měst a všech jejich vlastností do kolekce `Seznam_cilovaMesta`, který je vlastností objektu třídy `Trida_vylet`. Při každém dalším přidaném městě se volá služba Geocode Service pro vyhledání pozice. Pokud je město nalezeno, uloží se, jinak ne. Po vyhledání pozice se spustí metoda zasílající dotaz na vyhledání trasy pomocí služby Route Service mezi dvěma naposledy přidanými městy. Když je trasa po silnici nalezena, tak se uloží. Poté se volá metoda pro vykreslení této trasy mezi městy, a protože druhý krok obsahuje dvě mapy, vykreslí se jak trasa po silnici, tak trasa vzdušnou čarou. Jakmile je trasa

nelezena, volá se metoda `VratVzdalenost` z třídy `Trida_pomocna` pro výpočet vzdálenosti (vzdušné, popřípadě i silniční) mezi dvěma body, určenými zeměpisnou polohou.

### Třetí krok - ubytování

Na začátku třetího kroku se provede inicializace objektu třídy `Trida_hotely`, sloužícího pro zpracovávání XML odpovědí s vyhledanými hotely pomocí technologie LINQ to XML[1] a pro zpracování XML odpovědí s detaily hotelů. Jakmile uživatel ve třetím kroku označí délku pobytu v jednotlivých městech, tak se tyto termíny pobytu v konkrétních městech uloží do objektu třídy `Trida_vylet`. Pakliže je délka pobytu alespoň dva dny a město je vybráno ze seznamu měst podporující vyhledávání hotelů, zavolá se metoda zasílající dotaz na webovou službu Sluzba_hotely.svc pro vyhledání hotelů v daném městě. Program je naimplementován takovým způsobem, že dotazů na vyhledávání hotelů může být spuštěno více ve stejný čas pro různá města. Po obdržení odpovědi od služby, vytvořený objekt třídy `Trida_hotely` rozparuje odpověď a přiřadí seznam hotelů ke správnému městu. Pokud si uživatel nechá zobrazit detail hotelu, je volán další požadavek na webovou službu Sluzba_hotely.svc a ta vrátí odpověď ve formátu XML s detaily o hotelu a objekt třídy `Trida_hotely` ji zpracuje. Po zpracování se zobrazí nové okno tzv. `ChildWindow`, s detailem hotelu reprezentováno samostatným XAML souborem a aplikační logikou. V případě uložení konkrétního hotelu se hotel uloží do kolekce `Seznam_hotel`, který je vlastností objektu třídy `Trida_vylet`.

### Čtvrtý krok – spojení doprava

Po přidání nového spojení uživatelem, je toto spojení uloženo do kolekce `Seznam_dopravy`, která je vlastností objektu třídy `Trida_vylet`. Jakmile je spoj uložen volá se metoda zasílající dotaz na službu Geocode Service s názvy měst, které jsou startovním a cílovým městem daného spoje. V případě vyhledání polohy měst služba postupně vrátí zeměpisné souřadnice. Poté je volána další metoda pro vykreslení spojnice mezi městy a zároveň metoda `VratVzdalenost`, určená pro výpočet vzdálenosti vzdušnou čarou, nacházející se v pomocné třídě `Trida_pomocna`.

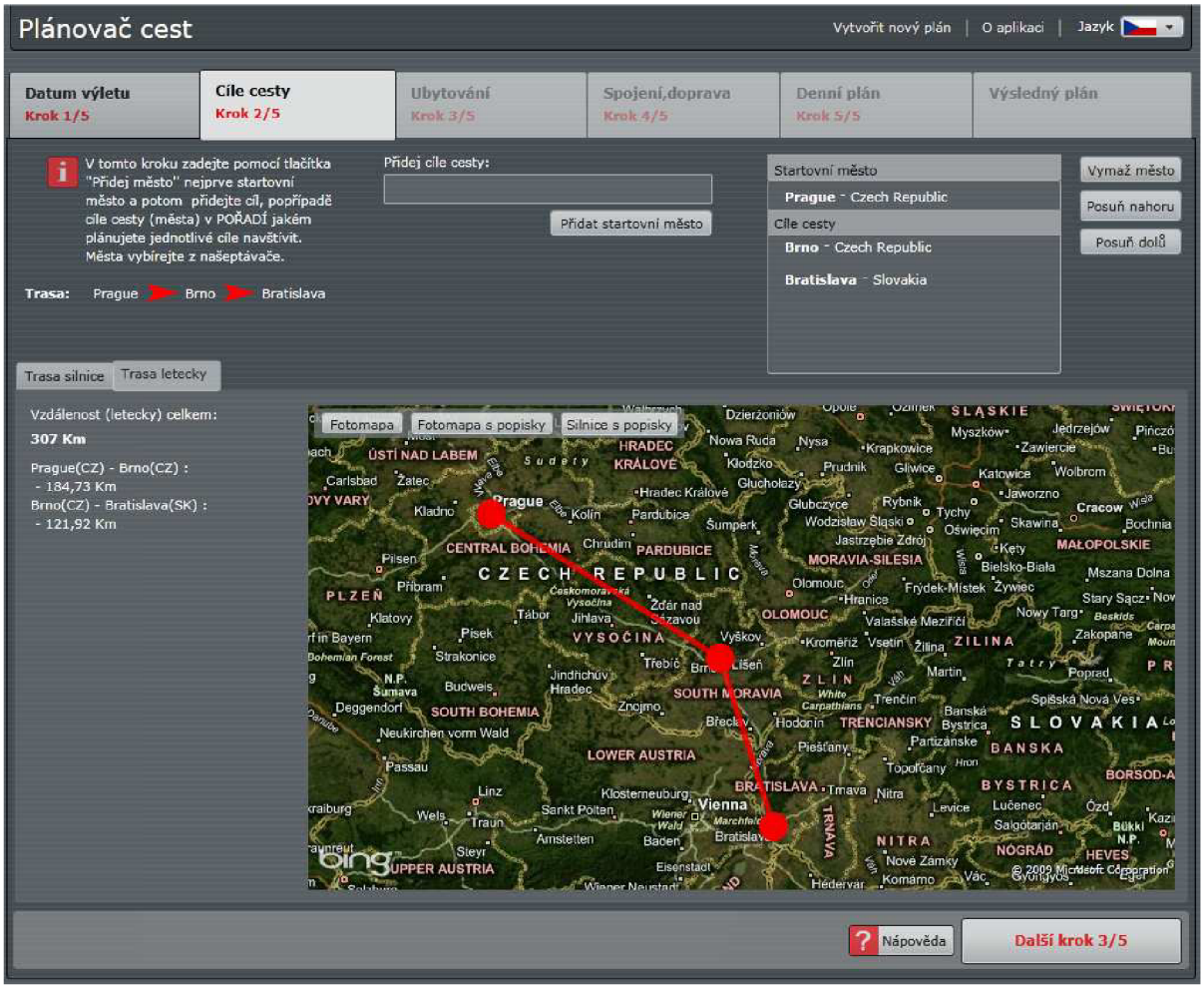
### Pátý krok – denní plán

Na začátku pátého kroku se provede inicializace objektu třídy `Trida_pocasiPars` sloužící pro zpracovávání a uložení XML odpovědí s předpovědí počasí pomocí technologie LINQ to XML. Postupně jak uživatel vyplňuje aktivity v jednotlivé dny, tak se postupně ukládají do kolekce `Seznam_aktivity` nacházející se v kolekci `Seznam_dnu`, který je vlastností objektu třídy `Trida_vylet`. Zároveň s výběrem konkrétního dne se vždy zavolá metoda zasílající dotaz na službu Weather Underground, který obsahuje pozici města pro níž chceme předpověď vyhledat. Po přijetí odpovědi pro jednotlivá města ve kterém se uživatel bude nacházet se předpověď uloží do kolekce `Seznam_pocasi`, nacházející se v kolekci Seznam_dnu, která je vlastnosti objektu třídy `Trida_vylet`.

[1] http://msdn.microsoft.com/en-us/library/bb387098.aspx

## 8.4 Prezentační vrstva

Prezentační vrstva je reprezentována souborem MainPage.xaml obsahující zhruba 950 řádků kódu jazyka XAML. První řádky souboru obsahují deklaraci `<UserContorl>` obsahující všechny potřebné jmenné prostory. Za ní následuje část `<UserContorl.Resources>` obsahující definici všech použitých konvertorů. Konvertor je speciální třída určená pro převod dat z různých datových zdrojů na požadovaný formát před cílovým zobrazením v prezentační vrstvě. Po definici konvertorů následuje blok kódu se styly určenými pro některé ovládací prvky. Zbytek souboru obsahuje již samotný kód s prezentační vrstvou aplikace. Do designu aplikace byly přidány některé prvky, vytvořené pomocí programu Adobe Photoshop CS2, pro obohacení vzhledu. Na obrázku 8.1 je ukázka grafického zpracování aplikace v kroku 2, které bylo zpracováno podle návrhu z kapitoly 7.3.




Obrázek 8.1: Ukázka grafického designu prezentační vrstvy kroku 2 zpracovaného podle návrhu

## 8.5 Problémy při implementaci

### Zprovoznění API Cleartrip

Největším problémem bylo využití API pro vyhledávání hotelů jak bylo již popsáno v kapitole 8.3.1. Následující text obsahuje stručné připomenutí problému. Služba je určena pro využívání pomocí technologie RESTful vracející odpovědi ve formátu XML. Standardní způsob použití je vytvoření klienta pomocí speciální třídy `WebClient` volajícího asynchronně službu reprezentovanou jedinečným identifikátorem `URI`. API Cleartrip vyžaduje použití vývojářského klíče, které je nutné umístit jako hodnota do HTTP hlavičky každého dotazu ve formátu `"X-CT-API-KEY:<vyvojarsky_klic>"`. Poté se může zavolat služba vracející po určitém čase XML odpověď. Bohužel na platformě Silverlight není žádná možnost jak vytvořit vlastní HTTP hlavičku v dotazu ve tvaru `X-CT-API-KEY`. Toto byla největší překážka. Naštěstí mě pak napadlo celý kód s klientem vytvořený třídou `Webclient` přesunout do vlastní webové služby WCF. Protože WCF služba už nemá žádné dočinění s platformou Silverlight, tak v ní jde požadovaná HTTP hlavička `"X-CT-API-KEY:<vyvojarsky_klic>"` pohodlně vytvořit.

Další problém ovšem nastal se samotným zprovozněním webové služby. Jako příčina nefunkčnosti služby bylo odhaleno špatné nastavení v konfiguračním souboru web.config, které poté bylo nastaveno do funkční podoby.

### Vyhledávání hotelů ve velkých městech

Po výše popsaných problémech se objevil další a to takový, že při vyhledávání hotelů ve větších městech program vždy po určité době spadl. Nejprve jsem to dával za vinu špatně naprogramované vlastní webové službě WCF, ale pak jsem zjistil, že chyba je v aplikaci. Pomocí nástroje Fiedler 2[1] jsem zjistil, že odpověď s vyhledanými hotely pro větší města se stáhne i přestože aplikace spadne. Taková odpověď může mít například až 3MB (což odpovídá zhruba seznamu 300 hotelů) a doba čekání na ni se může vyšplhat např. na 1,5 minuty a v tom byl kámen úrazu. Můj klient měl nastavený časovač, čekající na odpověď od webové služby, na krátký časový interval. Vypršení intervalu tohoto časovače způsobilo nepřijetí odpovědi a následný pád aplikace. Vše vyřešilo nastavení časovače na 3 minuty pomocí kódu:

```
klient.InnerChannel.OperationTimeout = TimeSpan.FromMinutes(3);.
```

### Nestabilita použitých zdrojů

Všeobecný problém, který mě provázel po celou dobu implementace byla nestabilita použitých zdrojů. Ta se projevovala téměř ve všech krocích programu a bylo se s ním potřeba nějakým způsobem vypořádat. Nestabilitou mám na mysli, že konkrétní zdroj nevrátí vždy odpověď, kterou měl vrátit, či nevrátí žádnou odpověď nebo se na odpověď musí dlouho čekat a to by mohlo potencionálního uživatele odradit. Tento problém se vyskytuje při vyhledávání měst. Ne vždy se podaří vyhledat poloha města a s tím souvisí poté nemožnost vykreslení spojnice mezi městy, spočítání vzdálenosti a zobrazení počasí ve městě. Tím pádem by ztratil program půl své funkcionality. Tento stav je ošetřen tedy tím, že když se nenajde město na mapě, tak se považuje za neexistující a nelze je přidat do seznamu.

[1] http://www.fiddler2.com/fiddler2/

U vyhledávání hotelů existuje podobný problém a to ten, že ne u každého hotelu jsou dostupné všechny informace a například u některých hotelů 80% informací chybí. S tím je spojena například nedostupnost foto-galerie hotelu, nemožnost zobrazení hotelu na mapě a jiné. Tady toto chování si tedy vyžadovalo při parsování XML odpovědi vše dokonale ošetřit pro případy, kdy detaily nejsou kompletní, popřípadě odpověď je prázdná. Stejný problém se samozřejmě vyskytuje i při parsování XML odpovědi s předpovědí počasí.

V případě delšího čekání na odpověď byly přidány do všech potřebných míst preloadery[1], aby uživatel věděl že se vykonává nějaká akce.

# 9 Testování

Velice důležitou fází každého vývoje aplikace je testování. V této kapitole jsou ukázky testů, které v rámci této fáze proběhly. Byly testovány jednotlivé části aplikace a nakonec i jakási průchodnost aplikace v podobě plánování různých vícedenních cest po celém světě.

Aplikace při testování byla umístěna na internetu u webhostingové společnosti ASPone[2] nabízející různé webhostingové služby jako ASP.NET hosting, freehosting a jiné. Můj projekt jsem umístil v rámci služby freehosting na adresu: http://planovac-cest.aspone.cz/. Projekt byl testován na několika operačních systémech a pomocí různých prohlížečů uvedených v následujícím přehledu.

**Windows XP SP3, Windows 7:**

- Mozilla Firefox 3.6,
- Internet Explorer 8,
- Google Chrome 10.0,
- Opera 11.

**Ubuntu 10.04:**

- Mozilla Firefox 3.6,
- Google Chrome 7.0.

Ve všech uvedených prohlížečích na operačním systému Windows XP a Windows 7 se po nainstalování zásuvného modelu Silverlight aplikace chovala bezproblémově, korektně a jednotně. Problém nastal při testování na systému Ubuntu. Po stažení projektu Moonlight[3] a nainstalování do systému se nepodařila aplikace korektně rozjet ani v jednom ze dvou prohlížečů. Při spuštění v Mozille Firefox sice naběhne úvodní preloader do sta procent, ale aplikace se už nespustí. Google Chrom na tom je o něco lépe. Aplikace se spustí, ale je neúměrně pomalá a práce s ní je prakticky vyloučena. Po těchto zjištěních testy probíhaly již jen v prohlížeči Mozzila Firefox na operačním systému Windows 7.

[1] Objekt, který vizuálně informuje uživatele, že se aktuálně načítají data, či vykonává jiná činnost.
[2] http://www.aspone.cz/
[3] http://www.go-mono.com/moonlight/download.aspx/

### Test č. 1 - Vyhledávání měst

Tento test prověřil schopnost aplikace vyhledávat různá města po celém světě, zobrazovat na mapě a počítat vzdálenosti. Vzdálenost se počítala vždy z města Brna a to jak vzdušnou čarou, tak v případě nalezení cesty i vzdálenost po silnici. V tomto testu aplikace obstála a byla nalezena všechna určená města a hodnoty vzdáleností byly vypočítány podle očekávání. Následující tabulka 9.1 shrnuje výsledky testů.

Tabulka 9.1: Shrnutí výsledků testu, který testoval schopnost aplikace vyhledat město a vypočítat vzdálenost z města Brna

| Město | Stát | Nalezeno | Vzdálenost vzdušnou čarou [Km] | Vzdálenost po silnici [Km] |
|---|---|---|---|---|
| Los Angeles | California | ano | 7 484,4 | - |
| Chicago | Illinois | ano | 9 739,8 | - |
| Valencia | Spain | ano | 1 723,4 | 2 257,2 |
| Milan | Italy | ano | 695,5 | 994,9 |
| Cape town | South Africa | ano | 9 244,2 | - |
| Tokyo | Japan | ano | 9 041,6 | - |
| Moscow | Russia | ano | 1 590,5 | - |
| Olomouc | Czech Republic | ano | 64,3 | 77,7 |

### Test č. 2 - Vyhledávání hotelů

Druhý test prověřoval vyhledávání hotelů v různých městech. Města byla zvolena totožná jako v prvním kroku. Ubytování se v testu vyhledávalo v každém městě vždy s požadavkem jednoho pokoje pro jednu osobu a na jednu noc. Aplikace obstála i v tomto testu a ve všech městech se podařilo nalézt alespoň jeden hotel. Rozdíly v jednotlivých městech byly samozřejmě v počtu vyhledaných hotelů a dobou vyhledávání. Nejkratší doba vyhledávání byla u města Olomouc, kde činila 3 sekundy a nejdelší u města Tokyo, kde trvala 43 sekund. Následující tabulka 9.2 shrnuje výsledky testu.

Tabulka 9.2: Shrnutí výsledků testu, který testoval zda dokáže v určených městech aplikace vyhledat seznam hotelů a jaký počet hotelů

| Město (stát) | Termín ubytování | Vyhledávání úspěšné | Počet nalezených hotelů |
|---|---|---|---|
| Los Angeles (California) | 8.4. – 9.4.2011 | ano | 78 |
| Chicago (Illinois) | 9.4. – 10.4.2011 | ano | 72 |
| Valencia (Spain) | 10.4. – 11.4.2011 | ano | 39 |
| Milan (Italy) | 11.4. – 12.4. 2011 | ano | 138 |
| Cape Town (South Africa) | 12.4. – 13.4. 2011 | ano | 79 |
| Tokyo (Japan) | 13.4. – 14.4. 2011 | ano | 169 |
| Moscow (Russia) | 14.4. – 15.4. 2011 | ano | 39 |
| Olomouc (Czech Republic) | 15.4. – 16.4. 2011 | ano | 1 |

### Test č.3 – Předpověď počasí

Pro test na zjišťování předpovědi počasí byly použity stejné města jako v předchozích dvou testech. Test zjišťoval schopnost aplikace předpovědět počasí v jednotlivých městech po celém světě. Pro zjišťování předpovědi bylo určeno datum 9.4.2011 a test byl proveden právě dva dny před tímto datem. Shrnutí výsledku testů je v tabulce 9.3.

Tabulka 9.3: Shrnutí výsledků testu, který testoval schopnost vyhledat předpověď počasí ve městech po celém světě pro datum 9.4.2011

| Město (stát) | Předpověď nalezena | Min. teplota, max. teplota [℃] | Předpověď počasí |
|---|---|---|---|
| Los Angeles (California) | ano | +9, +16 | déšť |
| Chicago (Illinois) | ano | +10, +16 | polojasno |
| Valencia (Spain) | ano | +17, +28 | jasno |
| Milan (Italy) | ano | +11, +25 | jasno |
| Cape Town (South Africa) | ano | +19, +28 | jasno |
| Tokyo (Japan) | ano | +10, +17 | přeháňky |
| Moscow (Russia) | ano | +0, +4 | déšť |
| Olomouc (Czech Republic) | ano | +0, +12 | polojasno |

### Test č. 4 - Kompletní test

Cílem tohoto testu bylo otestovat kompletní funkčnost aplikace, tak jak ji budou používat uživatelé. Tzn. otestovat veškeré chybové stavy, různý pohyb uživatele po aplikaci a zadávání většího počtu informací do plánu pro otestování stability programu. V rámci tohoto testu bylo naplánováno několik cest na různých kontinentech s různým počtem cílových měst a v několika odlišných termínech. Jako ukázka plánu jednoho z testů slouží výsledný plán který je v příloze A. Aplikace i v tomto testu obstála a byla po celou dobu stabilní a nevykazující žádné chyby.

# 10 Závěr

Bakalářská práce se zabývala tématem mashup aplikací a teorií s tímto tématem spojenou. Hlavním cílem bylo vytvoření aplikace pro plánování vícedenních cest, která umožňuje zaznamenávání nejrůznějších informací s cestou spojených. Výstupem aplikace je pak vytisknutý přehledný plán vícedenní cesty. Aplikace využívá několik externích zdrojů a je implementována pomocí technologie Silverlight. Tato technologie byla pro mě novinkou a z vlastního pohledu musím konstatovat, že mě velice mile překvapila svou robustností a možnostmi, kterými disponuje. Studium této technologie a práce s ní byly pro mě přínosem a velkou zkušeností.

Výsledná aplikace se mi podařila dokončit v dobrém čase a s výsledky své práce jsem spokojen. Aplikace splňuje veškeré podmínky dle zadání a podmínky, které byly kladeny postupně při vypracovávání. Myslím, že výsledný plánovač má pohodlné a přehledné uživatelské rozhraní, které přispívá ke snadné orientaci. Jako malý bonus beru kompletní podporu programu nejen v českém jazyce, ale i v anglickém, což by určitě mohlo přispět ve velké míře k použitelnosti aplikace i za hranicemi České republiky. Pro tento fakt hraje velkou roli i to, že se mi podařilo použít externí zdroje vracející celosvětové údaje a aplikace není nijak zeměpisně omezená. Při implementaci se vyskytly menší problémy týkající se především práce s externími zdroji, které však byly úspěšně vyřešeny.

Jako malé omezení aplikace beru fakt, že zdroj pro vyhledávání hotelů vyhledává jen ve větších městech a ne vždy velké množství hotelů. Ale i u ostatních zdrojů může nastat občas výpadek, či zaslání nekompletní odpovědi. Další omezení se týká použitelnosti aplikace a plyne z použité technologie Silverlight a nutnosti instalování pluginu do prohlížeče při používání a téměř nulové funkčnosti na operačních systémech linux.

V aplikaci nebyly splněny některé požadavky, které jsem si sám určil. Tyto požadavky mohou být předmětem rozšíření do budoucna. Jedná se o možnost naplánovat nejen vícedenní cestu, ale i jednodenní výlet nebo možnost samostatného vyhledávání hotelů. Dalším rozšíření, které připadá v úvahu je vytvoření uživatelských účtů a s tím spojené ukládání plánů online a jejich případnou editací. To by vyžadovalo i následné kompletní zabezpečení aplikace. Co se týče samotné funkcionality aplikace, bylo by vhodné ji doplnit například o možnost vyhledání zajímavých turistický atrakcí, památek, restaurací či jiných služeb ve městě pomocí dalších externích zdrojů. To by se týkalo spíše plánování rodinných výletů, které moje aplikace samozřejmě nevylučuje.

# Literatura


[1] AMBROŽ J.: *Web 2.0: bublina, nebo nový směr webu?* [online]. 2007-04-27, [cit. 2011-03-27]. URL: <http://www.lupa.cz/clanky/web-2-0-bublina-nebo-novy-smer-webu/>.

[2] *Wikipedie*: *Web 3.0* [online]. 2011-01-08, [cit. 2011-03-27]. URL: <http://cs.wikipedia.org/wiki/Web_3.0/>.

[3] O'REILLY T.: *What is Web 2.0* [online]. 2005-09-30, [cit. 2011-03-30]. URL: <http://oreilly.com/web2/archive/what-is-web-20.html/>.

[4] CLARKIN L., Holme J.: *Enterprise Mashups* [online]. [cit. 2011-04-2]. URL: <http://msdn.microsoft.com/en-us/architecture/bb906060/>.

[5] CHOW S.: *Programujeme Mashup aplikace pro Web 2.0 v PHP*. Computer Press, a.s., Brno, první vydání, 2008, ISBN 978-80-251-2057-6.

[6] ZANDL P.: Mashup *aneb Míchanice pro Web 2.0* [online]. 2007-05-21, [cit. 2011-04-04]. URL: <http://www.certodej.cz/view/mashup-aneb-m/>.

[7] *Wikipedie*: *Mashup (web application hybrid)* [online]. 2011-04-01, [cit. 2011-04-04]. URL: < http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)/>.

[8] MALÝ M.: *API webových služeb* [online]. 2007-09-01, [cit. 2011-04-04]. URL: < http://www.apimania.eu/api-webovych-sluzeb/>.

[9] MORAVEC Z.: *RIA – Rich Internet Application* [online]. 2009-04-14, [cit. 2011-04-06]. URL: <http://programujte.com/?akce=clanek&cl=2009041200-ria-rich-internet-applications/>.

[10] VESELKA A.: *Obohaťte své uživatele pomocí RIA* [online]. 2007-02-05, [cit. 2011-04-06]. URL: <http://www.symbio.cz/clanky/obohatte-sve-uzivatele-pomoci-ria.html/>

[11] *Wikipedie*: JavaFX [online]. 2010-12-20, [cit. 2011-04-06]. URL: <http://cs.wikipedia.org/wiki/JavaFX/>.

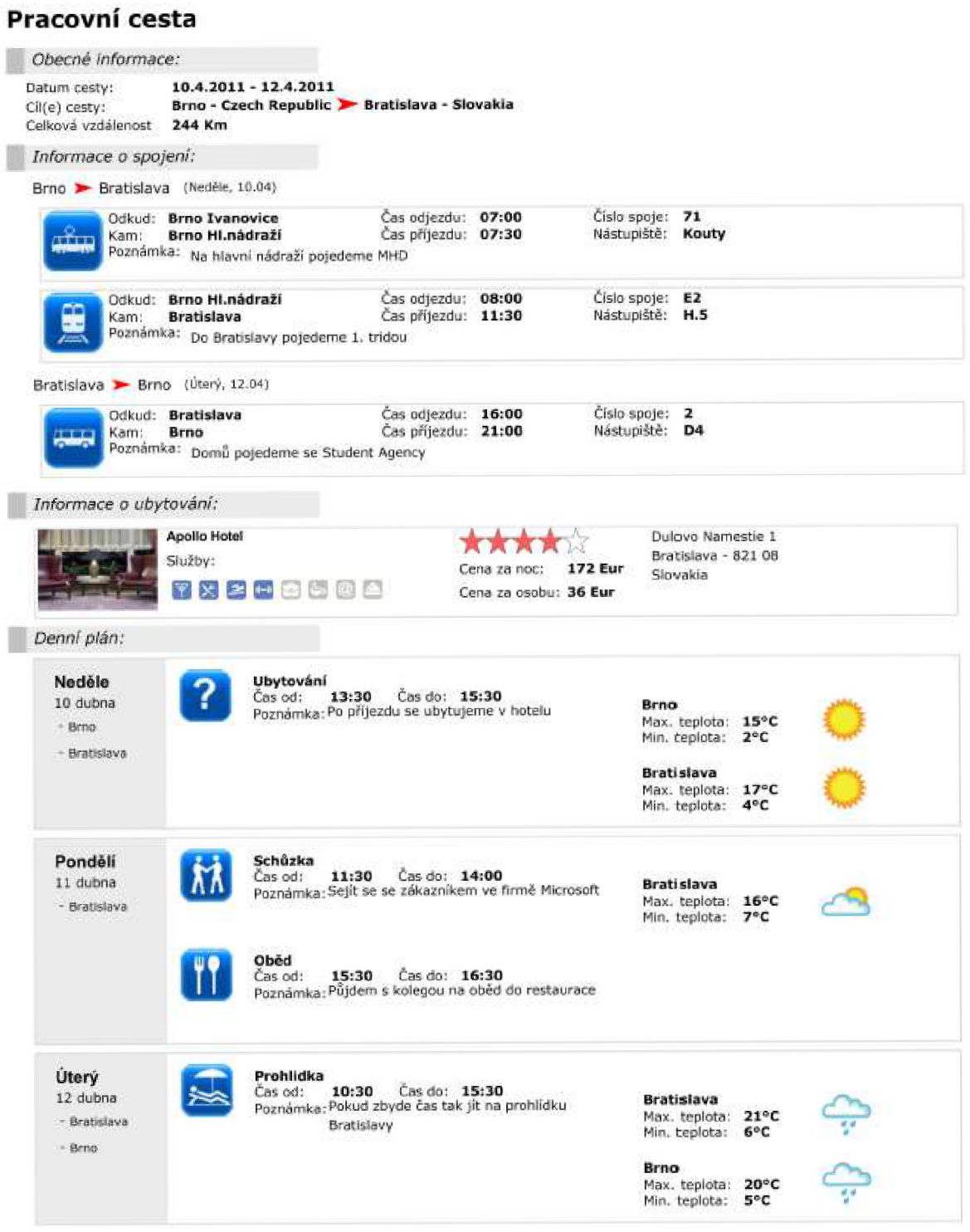
[12] LACKO L.: *Silverlight, Výukový průvodce tvorbou interaktivních aplikací*. Computer Press, a.s., Brno, první vydání, 2010, ISBN 978-80-251-2716-2.

[13] *Microsoft*: The Future of Microsoft Silverlight [online]. [cit. 2011-04-06]. URL: <http://www.microsoft.com/silverlight/future/>.

[14] PAPA J.: *Datové služby*. Zonner Press, a.s., Brno, první vydání, 2009, ISBN 978-80-7413-041-0.

# Příloha A

Ukázka výsledného plánu aplikace v termínu 10.4.2011 - 12.4.2011 se startovním městem Brno a cílovým Bratislava.



**Pracovní cesta**

*Obecné informace:*

Datum cesty: **10.4.2011 - 12.4.2011**
Cíl(e) cesty: **Brno - Czech Republic ➤ Bratislava - Slovakia**
Celková vzdálenost **244 Km**

*Informace o spojení:*

Brno ➤ Bratislava (Neděle, 10.04)

Odkud: **Brno Ivanovice** Čas odjezdu: **07:00** Číslo spoje: **71**
Kam: **Brno Hl.nádraží** Čas příjezdu: **07:30** Nástupiště: **Kouty**
Poznámka: Na hlavní nádraží pojedeme MHD

Odkud: **Brno Hl.nádraží** Čas odjezdu: **08:00** Číslo spoje: **E2**
Kam: **Bratislava** Čas příjezdu: **11:30** Nástupiště: **H.5**
Poznámka: Do Bratislavy pojedeme 1. tridou

Bratislava ➤ Brno (Úterý, 12.04)

Odkud: **Bratislava** Čas odjezdu: **16:00** Číslo spoje: **2**
Kam: **Brno** Čas příjezdu: **21:00** Nástupiště: **D4**
Poznámka: Domů pojedeme se Student Agency

*Informace o ubytování:*

**Apollo Hotel**
Služby:
Cena za noc: **172 Eur**
Cena za osobu: **36 Eur**
Dulovo Namestie 1
Bratislava - 821 08
Slovakia

*Denní plán:*

**Neděle**
10 dubna
- Brno
- Bratislava

**Ubytování**
Čas od: **13:30** Čas do: **15:30**
Poznámka: Po příjezdu se ubytujeme v hotelu

**Brno**
Max. teplota: **15°C**
Min. teplota: **2°C**

**Bratislava**
Max. teplota: **17°C**
Min. teplota: **4°C**

**Pondělí**
11 dubna
- Bratislava

**Schůzka**
Čas od: **11:30** Čas do: **14:00**
Poznámka: Sejít se se zákazníkem ve firmě Microsoft

**Bratislava**
Max. teplota: **16°C**
Min. teplota: **7°C**

**Oběd**
Čas od: **15:30** Čas do: **16:30**
Poznámka: Půjdem s kolegou na oběd do restaurace

**Úterý**
12 dubna
- Bratislava
- Brno

**Prohlídka**
Čas od: **10:30** Čas do: **15:30**
Poznámka: Pokud zbyde čas tak jit na prohlídku Bratislavy

**Bratislava**
Max. teplota: **21°C**
Min. teplota: **6°C**

**Brno**
Max. teplota: **20°C**
Min. teplota: **5°C**

# Příloha B

Obsah CD přiložené k bakalářské práci má následující adresářovou strukturu:

- **`Textova_zprava`** - písemná část bakalářské práce ve formátu PDF a DOC
- **`Aplikace_soubory`** - zdrojové soubory aplikace, přeložená a spustitelná aplikace
- **`Aplikace_navod`** - textový soubor s postupem zprovoznění aplikace
- **`Dokumentace`** – kompletní vygenerovaná programová dokumentace
- **`Aplikace_plakat`** - plakát o aplikaci ve formátu PNG a JPEG