VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

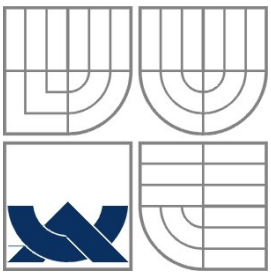
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO PLÁNOVÁNÍ CEST

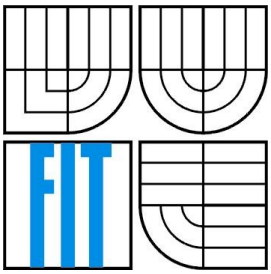
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE BOHUMIL VRBA
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO PLÁNOVÁNÍ CEST

JOURNEY PLANNING SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

BOHUMIL VRBA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. LADISLAV RUTTKAY

BRNO 2009

Abstrakt

Bakalářská práce se zabývá návrhem a vývojem webové aplikace sloužící k plánování cestovních tras s využitím technologie Google Maps. Věnuje se popisu použitých technologií, zejména ASP.NET, databázovému systému MS SQL Server a mapovému systému Google Maps. Dále se věnuje specifikaci požadavků, jejich zápisem za pomoci jazyka UML a návrhem datového modelu. Popisuje implementaci vyvíjené aplikace s důrazem na použitou architekturu.

Abstract

Bachelor's thesis covers design and developing of web application which serves for planning itinerary. I use Google Maps technogy to display and plan the journies. The thesis describes applied technology especially ASP.NET, database system MS SQL Server and Google map system. It is also dedicated to requirements specification, their registration with the help of language UML and design of data model. It is devoted to the implementation of aplication with an emphasis on used architecture.

Klíčová slova

Plánování cest, ASP.NET, Google Maps API, MS SQL, AJAX, ASP.NET AJAX Extensions, UML, datový model, třívrstvá architektura, ASP.NET Resources, XSLT, XHTML, CSS

Keywords

Journey planning, ASP.NET, Google Maps API, MS SQL, AJAX, ASP.NET AJAX Extensions, UML, data model, 3 layered architecture, ASP.NET Resources, XSLT, XHTML, CSS

Citace

Bohumil Vrba: Systém pro plánování cest, bakalářská práce, Brno, FIT VUT v Brně, 2009

Systém pro plánování cest

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Ladislava Ruttkaye.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Bohumil Vrba
20. května 2009

Poděkování

Děkuji Ing. Ruttkayi za odborné vedení, ochotu, cenné rady a připomínky při zpracování a řešení bakalářské práce.

Obsah

1 Úvod

Stejně tak jako v mnoha jiných zemích ani v České Republice neexistuje dokonalý dopravní systém. Dlouhá doba dojezdnosti autobusů způsobená častými zastávkami, nebo zpoždění a přeplněnost vlaků jsou důvody k hledání možné alternativy. Řešení lze najít v dopravě vlastním automobilem nebo využít svezení s lidmi, kteří tuto službu nabízí.

Cílem mé bakalářské práce je vytvořit webový systém, který slouží k plánování cest s využitím mapové technologie. Systém umožní uživateli naplánovat cestu, den odjezdu, typ dopravního prostředku a jiné podrobnosti týkající se cesty. Zadavatel trasy má poté možnost vybrat spoluúčastníky, kterým bude tato informace oznámena a mohou se na cestu přihlásit. Lze se tak podělit o volná místa ve svém automobilu na cestách za prací nebo za studiem s ostatními uživateli systému. Řidič získá předem smluvenou finanční odměnu a tím, že sveze více lidí, zredukuje poměr množství do ovzduší vypuštěného CO_2 na osobu.

Webové rozhraní aplikace bude poskytovat grafické zobrazení trasy a další podpůrné informace tykající se cesty. Pro plánování i zobrazení bude využita volně dostupná mapová aplikace a technologie Google Maps. K uložení potřebných dat pro bezproblémový běh aplikace použiji Microsoft SQL Server 2008.

Práce je rozdělena do 5 kapitol. Nejdříve se zabývám technologiemi, které jsou použity při vývoji webové aplikace. Je zde popsáno prostředí .NET Framework a jeho součást ASP.NET pro tvorbu webových stránek. Dále jsem do použitých technologií zahrnul databazový systém MS SQL Server 2008 a mapový systém Google Maps.

Následuje třetí kapitola, která se zabývá prostředky použitými pro návrh aplikace. Dále se věnuje specifikaci požadavků a návrhu relační databáze. Nakonec je zde uvedenen diagram tříd.

Ve čtvrté kapitole jsem se zaměřil na kokrétní postupy při implementaci stránek a komponent. Popisuji zde použitou třívrstvou architekturu a jednotlivé vrstvy této architektury v souvislosti s manipulací, získáváním a zobrazením dat uložených v databázi. V závěru bakalářské práce je vytvořený systém zhodnocen a jsou diskutovány možnosti dalšího vývoje.

2 Technologie pro vývoj webové aplikace

Tato kapitola obsahuje stručný přehled technologií a programů, které jsem využil při zpracování a implementaci webové aplikace. Mimo jiné obsahuje popis .NET frameworku, služby Google Maps včetně API a uložení geografických dat v SQL Serveru 2008.

2.1 Značkovací jazyky HTML a XHTML

HTML (Hypertext Markup Language) a XHTML (eXtended Hypertext Markup Language) jsou značkovací jazyky pro tvorbu hypertextových dokumentů. Tvoří základ pro publikaci dokumentů na Internetu.

2.1.1 HTML

HTML je typickým zástupcem rodiny SGML (Standard Generalized Markup Language). Jedná se o jazyk nejčastěji používáný pro vytvoření základní struktury dokumentu. Obsahuje také prvky a atributy pro změnu vzhledu nebo přidání další funkcionality dokumentu, ale tyto rozšiřující vlastnosti se již nevyužívájí, protože byly nahrazeny přehlednějšími a k tomuto účelu přímo určenými tabulkami kaskádových stylů (CSS).

2.1.2 XHTML

XHTML je aplikací jazyka XML (eXtensible Markup Language) na rozdíl od svého předchůdce (HTML), který je aplikací staršího a složitějšího jazyka SGML. Syntaktická stránka XHTML je obsažena v DTD (Dokument Type Definition) a význam jednotlivých značek i atributů je popsán v oficiální specifikaci, která je spravována konsorciem W3C. Vetšina značek a atributů zůstala zachována z HTML až na ty pro definici vzhledu, které lze nahradit pomocí CSS. XHTML má striktní syntaxi. Tam kde HTML připouští více způsobů zápisu, XHTML povoluje pouze jediný.

Nejvýznamější změny tykající se zápisu značek v XHTML (viz [1]):

- všechny názvy značek musí být psány malými písmeny (XML je case-sensitive),
- všechny neprázdné znaky musí mít koncovou značku,
- nepárové značky nejsou povoleny (některé značky lze psát zkráceným způsobem, např.: obrázek „”) ,
- tagy se nesmí nikdy křížit,
- atributy jsou case-sensitive a vždy neprázdné,
- hodnoty atributů musí být uvedeny v uvozovkách.

2.2 CSS

Tabulky kaskádových stylů (Cascading Style Sheets) (viz [2]) jsou nadstavbou vyznačovacích jazyků HTML, XHTML či XML a slouží k popisu prezentace dokumentů. Nespornou výhodou je, že popisují vzhled uživatelského rozhraní, aniž by jakkoli ovlivňovaly obsah a strukturu dokumentu. Na druhou stranu značnou nevýhodou tohoto přístupu je různá interpretace a špatná podpora CSS stylů ve starších prohlížečích, které jsou bohužel v dnešní době stále používány. Existují však různé způsoby, jak tyto nedostatky obejít a vytvořit tak téměř totožnou prezentaci dokumentu (aplikace) v nejvíce využívaných prohlížečích.

2.3 Microsoft .NET Framework

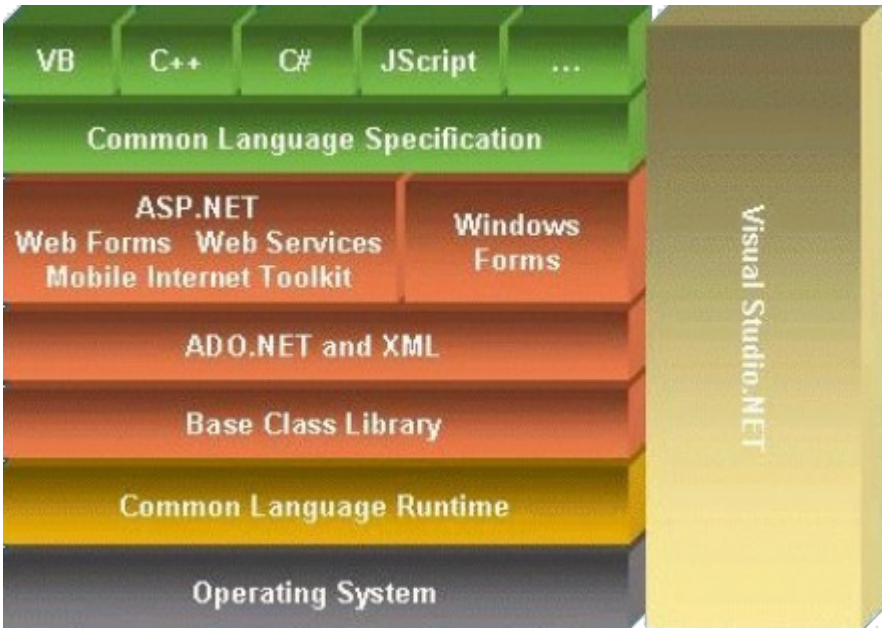
Než začneme vyvíjet webovou aplikaci je třeba vybrat softwarovou platformu. Pro tuto práci jsem zvolil Microsoft .NET Framework, který přináší zajímavé vyčerpávající řešení pro vývoj a provoz webových služeb, desktopových a webových aplikací.

Hlavními komponentami .NET Frameworku (viz [3]) jsou společný jazykový běhový modul CLR (Common Language Runtime) a knihovna tříd rámce .NET FCL (Framework Class Library).

CLR poskytuje abstrakci služeb operačního systému a má kontrolu nad všemi akcemi, které spravovaná aplikace provadí. Tento modul nabízí strukturované prostředí pro vykonávání kódu kompilovaného z C#, Visual Basicu .NET, C++ a jiných jazyků. Řízený (managed) kód, který je posloupností instrukcí společného zprostředkovacího jazyka CIL (Common Intermediate Language), buď běží v CLR nebo mu CLR povolí běžet mimo tento modul. Podrobněji (viz [3]).

FCL je velmi rozsáhlá a komplexní knihovna tříd poskytující objektově orientované rozhraní API využívané aplikacemi běžícími pod CLR. Obsahuje kolem 100 jmenných prostorů. Každý jmenný prostor poskytuje třídy a další typy, které pojí společný učel. BCL (Base Class Library) je součástí FCL a přináší zakládní funkcionalitu, která obsahuje třídy ve jmenném prostoru System. Například jmenný prostor System.IO zahrnuje třídy pro operace vstupu a výstupu se soubory. FCL celkem zahrnuje více než 7000 typů – tříd, struktur, rozhraní, výčtů a delegátů.

Kromě webových aplikací lze vyvíjet webové služby XML, klienty webových služeb, konzolové aplikace, formuláře Windows (WinForms) a dokonce i služby Windows. Komplexnost Microsoft .NET Framework nejlépe vyjadřuje blokové schéma jeho architektury (viz obr. 2.1).



Obr. 2.1: Architektura Microsoft .NET Framework
Dostupný z WWW: <http://merc.tv/img/fig/dotnetfw.gif>

2.4 ASP.NET

ASP.NET (viz [3]) je součástí prostředí .NET framework pro vývoj dynamických webových nebo intranetových aplikací a webových služeb. Jedná se o nástupce technologie aktivních serverových stránek ASP (Active Server Pages), která je snadno použitelným modelem dynamického generování HTML na webových serverech. ASP soubory respektující syntax HTML obsahují příkazy v daném skriptovacím jazyku. ASP.NET se díky výhodám modulů CLR a FCL platformy .NET Framework (viz předchozí podkapitola) znatelně liší od svého předchůdce.

Základem technologie ASP.NET jsou dynamické stránky na serveru, jejichž kód se před prvním spuštěním kompiluje do jazyka MSIL (Microsoft Intermediate Language) stejně jako každá aplikace v .NET Frameworku. Tím se odstranila nutnost analýzy a interpretace jednotlivých řádků při každém přístupu klienta a zrychlil se tak běh aplikace. Programovací přístup se v ASP.NET snaží co nejvíce přiblížit programování desktopových aplikací pro windows (formuláře windows). Používají se serverové ovládací prvky (Controls), které spouštějí události. Tyto události lze zpracovávat skripty na straně serveru. Je tedy možné používat ovládací prvky jako je tlačítko (Button), popisek (Label) a spoustu dalších. ASP.NET od verze 2.0 generuje validní XHTML.

2.4.1 Problém bezstavového protokolu HTTP

Webový protokol HTTP je bezstavový, ale ve webových aplikacích je často nutné stav relace nějakým způsobem ukládat. ASP.NET tento problém řeší bezpečně pomocí dvou technik, jimiž jsou:

- **ViewState** – uchovává informace o stavu relací jednotlivých uživatelů ve skrytých formulářových polích kruhově odesílaných klientovi a zpět. Výhodou je, že nevyžaduje žádné speciální prostředky na straně klienta ani serveru, ale bohužel na úkor většího množství přenášených dat.

- **SessionState** – uchovává tyto informace v relaci na straně serveru a vrací prvek Cookie jednoznačně identifikující tuto relaci. Cookie prvek se poté používá k ustavení vztahu mezi relací a novým požadavkem. ASP.NET podporuje různé modely ukládání (do samostatného procesu nebo do databáze) a funguje i když má klient webový prohlížeč bez podpory Cookie.

- **V poli adresy –** informace je obsažena přímo v poli adresy. Část adresy, která nese dané informace se nazývá QueryString. Je to poslopnost párů „klič=hodnota“ oddělených znakem „*&*”. K hodnotě klíče lze v ASP.NET přistupovat pomocí vlastnosti `QueryString` třídy Request.

- **Cookie –** malé množství dat lze uložit na straně klienta (pokud je to povoleno). Většinou tyto informace slouží k rozlišení jednotlivých uživatelů a uložení uživatelských předvoleb, například jazykové lokalizace stránek.

2.5 AJAX (Asynchronous Javascript And XML)

AJAX (viz [5]) je technologie pro vývoj moderních webových aplikací. Využívá jazyk Javascript pro zobrazovaní daných dynamických změn stránek na klientovi (webovém prohlížeči) a formát XML pro přenos informací na server a zpět. Samozřejmostí je také zpracování požadavků na straně serveru. Důležitým přínosem této technologie je asynchronní komunikace klienta se serverem pomocí objektu XMLHttpRequest a následná změna fragmentu stránky bez nutnosti opětovného načtení celé stránky. Tím může webová aplikace hodně získat a stát se uživatelsky přívětivější a rychlejší. Aplikace “oživené“ touto technologií se svých chováním blíží klasickému desktopovému programu.

2.5.1 AJAX Extensions

Pro vývoj interaktivních a efektivních webových aplikací ASP.NET založených na technologii AJAX lze použít rozšíření Microsoft AJAX Extensions. Díky tomuto rozšíření se práce programátora ASP.NET s technologií AJAX rovná standardní práci se serverovými ovládacími prvky. Není nutné aby programátor znal a tvořil složité JavaScriptové funkce a staral se o kompatibilitu v různých webových prohlížečích. Extensions mají vestavěnou knihovnu Microsoft AJAX Library, která je platformově nezávislou sbírkou JavaScriptových klientských skriptů. Dále lze zahrnout sadu rozšiřujících komponent Control Toolkit, které využívají AJAX Extensions a jsou vyvýjeny i lidmi mimo Microsoft. Na serveru CodePlex se nachází volně dostupné zdrojové kódy.

2.6 Microsoft SQL Server 2008

Při návrhu webové aplikace je nutno vybrat prostředky, pomocí kterých budou získané informace uchovávány. Já jsem k tomuto účelu zvolil MS SQL Server 2008, který nově podporuje datové typy pro uložení prostorových dat. V této podkapitole se budu zabývat uložením geografických dat.

Microsoft SQL Server 2008 je relační databázový systém poskytující komplexní datovou platformu pro aplikace. Je schopný nejen efektivně pracovat s velkým množstvím dat, ale také kvalitně řídit a definovat strukturu těchto dat. Tento systém se v české odborné literatuře nazývá systém řízení baze dat (SŘBD). Primárními jazyky pro manipulaci a definici dat MS SQL Serveru jsou MS-SQL a T-SQL.

2.6.1 Uložení geografických dat

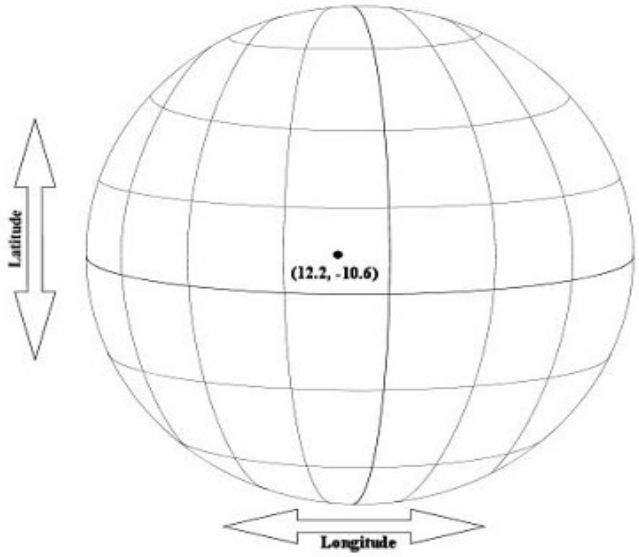
Uživatel pří plánování cesty zadá do systému adresy měst, skrze které chce projet. Aplikace následně využije služeb mapového systému Google Maps. V případě úspěchu poskytne systém Google Maps podrobné strukturované informace o dané trase, a to včetně adres v dokonalejším formátu a geografických souřadnic. Vybrané informace budou uloženy v databázi aplikace tak, aby byly dostupné v příštích přístupech uživatelů do systému.

SQL Server 2008 nabízí nové datové typy pro ukládání, dotazování a manipulaci s prostorovými daty v databázi. Geometrické a geografické datové typy byly navrženy pro reprezentaci a práci s lokalizovanými daty v souladu s geografickým značkovacím jazykem GML (Geography Markup Language), který je specifikován mezinárodní standardizační organizací pro geoprostorová data a služby - OGC (Open Geospatial Consortium). Podporují vektorové (body, čáry, polygony), rastrové prostorové data a geo-prostorové data (s umístěním na povrchu Země). Uložení geoprostorových dat a jejich funkce usnadňují propojení SQL Serveru s mapovými systémy jako jsou Google Maps nebo Microsoft Virual Earth a umožňuji provádět dotazy tykající se geografické polohy. Lze s nimi pracovat pomocí ADO.NET.

Rozlišujeme souřadnice planární nebo geodetické:

- Planární souřadnice (geometrický datový typ)
  - reprezentace dvou-dimenzionálních planárních dat (bodu na mapě)
  - výpočty s planárními daty jsou jednodušší

- Geodetické souřadnice (geografický datový typ)
  - reprezentace troj-dimenzionálních geodetických dat stejně jako je využívají GPS aplikace
  - příkladem jsou body na elipsoidním povrchu (geodetickém modelu Země viz. Obr. 2.3), umístění restaurací, hotelů, kin, atp.
  - výpočty s geodetickými daty jsou složitější



Obr. č. 2.3: Bod reprezentovaný párem zeměpisné šířky a výšky na povrchu země. Zdroj (viz[6]).

Geografické i geometrické typy jsou sestaveny z vektorových objektů specifikovaných ve formátech Well-Known Text (WKT), Well-Known Binary (WKB) nebo GML. Textový značkovací jazyk (WKT) a jeho binární ekvivalent pro přenos a uložení dat v databázovém systému (WKB) jsou transportními formáty pro prostorová data reprezentující vektorové objekty na mapě, kartografické projekce objektů a transformace mezi nimi. WKT i WKB jsou formáty taktéž spravované konsorciem OGC.

Geografická data uložená v SQL Serveru používají světový geodetický systém WGS 84 stejně jako jej využívá GPS (Global Positioning System). Prostorový referenční identifikátor (Spatial Reference Identifier - SRID) tohoto geo-systému je 4326. Určuje například, že výsledek vestavěné statické metody `STArea` (spočítání plochy) nad nějakým prostorovým útvarem bude v m^2. Nebo výsledek metody `STDiference` (vzdálenost dvou objektu) v metrech. Pro geometrická data, která geodetický systém nepoužívají, je třeba vložit na toto místo 0. Na příkladu níže demonstruji vytvoření bodů z formátu WKT pomocí statické funkce `STGeomFromText`, kde jsou tučně zvýrazněny identifikátory SRID.

```
DECLARE @g geometry;
SET @g = geometry::STGeomFromText('POINT (3 4)', 0);
DECLARE @g geography;
SET @g = geography::STGeomFromText('POINT (49.838 14.7656)', 4326);
```

Ukázka č. 4.2: deklarace a nastaveni prostorových proměnných

Prostorové typy v MS SQL Serveru 2008 poskytují spoustu vestavěných metod k manipulaci s instancemi těchto typů a k určení vztahu mezi nimi.

V následující tabulce je uveden výčet všech sedmi typů podporovaných v assembly Microsoft.SqlServer.Types.

Objekt	**Popis**
Point	Bod
MultiPoint	Kolekce bodů
LineString	Sekvence spojedných bodů (lomená úsečka, která se neprotíná)
MultilineString	Kolekce LineString objektů
Polygon	Region popsaný množinou spojených LineString objektů
MultiPolygon	Kolekce Polygon objektů
GeometryCollection	Kolekce Geometrických typů

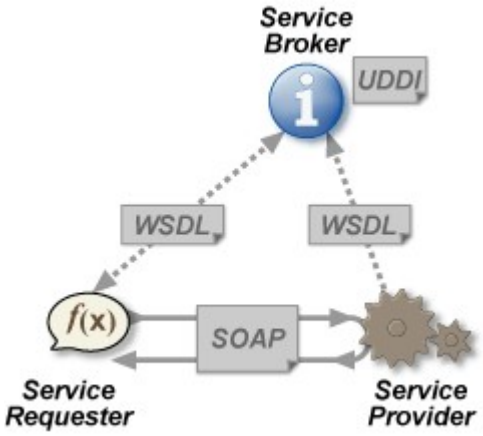
Tabulka č. 2.1: Vektorové objekty podporované SQL Serverem 2008

2.7 Webové služby

Webová služba (viz [3]) je jiným druhem webové aplikace, která nabízí funkce rozhraní API (webové metody) prostřednictvím internetu. Je to standardizované řešení, díky němuž si mohou různorodé systémy vyměňovat informace přes internet a to i v případech, kdy jsou vyvíjeny na různých platformách. Na rozdíl od webové aplikace nemá uživatelské rozhraní pro koncové uživatele, ale poskytuje služby a informace jiným aplikacím a systémům (např.: aktuální informace o počasí nebo stav kurzů měn).

Webová služba je průmyslový standard vystavěný na otevřených protokolech, jimiž jsou HTTP a protokol jednoduchého přístupu k objektům SOAP (Simple Object Access Protocol). A to tak, že požadavky HTTP na služby obsahují zprávy SOAP.

SOAP je protokolem pro vykonávání vzdálených volání založených na formátu značkovacího jazyka XML (eXtensible Markup Language) přes počítačovou síť. WSDL (Web Services Description Language) je jazyk taktéž na bázi XML. Popisuje rozhraní webových služeb a proto klientská aplikace musí rozumět WSDL aby zjistila, jaké funkce jsou dostupné na serveru. Poslední zkratkou o které se zde zmíním je UDDI (Universal Description, Discovery, and Integration). Je to technika pro popis, objevení a integraci webových služeb. Slouží k vytváření distribuovaných databází. Jedná se v podstatě globální registr, který obsahuje strukturované informace o firmách a webových službách, které jsou jimi nabízeny. Tyto vztahy jsou zpřehledněny na následujícím obrázku.



Obr. č. 2.4: Architektura webových služeb
Dostupný z WWW: <http://en.wikipedia.org/wiki/File:Webservices.png>

2.8 Mapový systém Google Maps

Tento projekt je cílen hlavně českým a slovenským uživatelům. Při výběru mapového systému jsem proto trval na tom, aby byly mapy lokalizovány do češtiny. Dále jsem bral v potaz kvalitu mapových podkladů a schopnost plánování cesty. Google Mapy jsou ve všech ohledech vhodné a splňují mé požadavky.

Webová mapová technologie Google Maps pohání spoustu služeb souvisejících s mapami. Podporuje vestavěné mapy do soukromých stránek. Nabízí mapy celého světa, plánovač cest. Google mapy jsou moderním mapovým portálem fungujícím na principu složitého komplexu vzájemně propojených databází (databáze restaurací, hotelů, firem a jiných služeb). Do Google Maps lze náhrát vrstvy tzv. Maplety, které jsou mini aplikacemi a přidávají mapám novou funkčnost. Webové mapy tak lze obohatit například zobrazením aktuálního počasí nebo videí z Youtube. Vytváření Mapletů funguje na podobném principu jako práce s rozhraním Google Maps API s tím rozdílem, že maplety jsou určeny pro běh na stránkách mapové služby Google Maps, kdežto Google Maps API slouží k vytváření map od základů na jiných stránkách.

2.8.1 Ovládání

Google Maps se snadno ovládájí pomocí klávesových zkratek a myši.

- Klávesové zkratky - Posouvání doleva, doprava, nahoru a dolů pomocí kláves se šipkami. Posouvání do šířky pomocí kláves Page Up, Page Down, Home a End. Přiblížení a oddálení pomocí kláves plus (+) a mínus (-).
- Funkce lupy aktivovaná dvojitým kliknutím - Dvojitým kliknutím levým tlačítkem mapu lze přiblížit a dvojitým kliknutím pravým tlačítkem oddálit (pro uživatele počítačů Mac: Ctrl + dvojité kliknutí).

- Přiblížení pomocí posouvacího kolečka - Pomocí posouvacího kolečka myši lze mapy přiblížit nebo oddálit.

2.8.2 Google Maps API

Google Maps API umožňuje vývojářům integravat Google Maps do vlastních webových stránek. Aby byly takto vestavěné Google Maps funkční, je nutné nechat si vygenerovat API klíč. Tento klíč se váže k URL adresáře stránky, do které mají být mapy vestavěny. Práce s Google Maps API vyžaduje znalost programování v Javascriptu. Pomocí konstrukcí v tomto jazyce lze vestavit mapy a neomezeně využívat funkcí API stejně jako fungují na webu „http://maps.google.cz“. Na webové mapě můžeme vyobrazit prakticky jakýkoliv tvar a propojit jej se svou vlastní databází. Google poskytuje podrobnou online dokumentaci (viz [9]). Následující podkapitoly obsahují vybrané třídy.

2.8.2.1 Třída GDirections

Použitím objektu této třídy lze mapám přidat schopnost plánování cesty. Výsledek této služby je vyžádán pomocí metody `load`, jejímž prvním parametrem je buď řetězec adres (měst) popisující trasu (příklad: „From: Praha, Jiráskova to: Brno to: Olomouc“) a nebo posloupnost párů zeměpisné šířky a délky ("40.712882, -73.967257 to 41.943181,-87.770677"). Pokud chceme přetížit chování této metody, lze tak učinit druhým parametrem, jímž je objekt třídy GDirectionsOptions (viz další podkapitola). Získaná trasa může být buď vykreslena jako lomená čára na mapu, nebo může být vypsána jako série navigačních kroků do určité oblasti (elementu) na stránce a nebo obojí.

Tabulka č. 2.2 obsahuje některé vybrané metody třídy GDirection. Podrobnější informace lze nalézt v online referenční dokumetaci (viz [9]).

Metoda	Popis
load(query:String, queryOpts?:GDirectionsOptions)	vydává nový dotaz na směrování na základě dvou parametrů, které byly popsány v úvodu podkapitoly, výsledek může obsahovat více cest (objektů třídy GRoute), jednu cestu pro každý pár míst specifikovaných v dotazu
loadFromWaypoints(waypoints: Array, queryOpts?:GDirectionsOptions)	funguje stejně jako load, s rozdílem prvního vstupního parametru jímž je pole textových vstupních adres nebo textových párů zeměpisné a délky (lat/lon point)
getBounds()	vrátí objekt typu GLatLngBounds, který reprezentuje hranici(obdélník) ve kterém je zobrazen výsledek, pomocí souřadnic jeho jihozápadního a severovýchodního rohu
getRoute(i:Number)	vrátí i-tou část cesty (objekt třídy GRoute)
getNumGeocodes()	vrátí počet geokódovaných vstupů, pokud byl náš dotaz úspěšný tak se tato hodnota rovná počtu vstupních míst(adres)
getGeocode(i:Number)	vrátí strukturované informace o i-té lokaci ve formátu pro výměnu dat JSON (JavaScript Object Notation).

Tabulka č. 2.2: Metody třídy GDirection

Následuje popis části struktury informací o daném místě na trase získané pomocí metody `getGeocode(i:Number)`:

- **address** - správný formát adresy včetně správně psaných kapitálek.
- **AddressDetails** - adresa ve formátu mezinárodního standardu pro formátování adres xAL (eXtensible Address Language).
  - **Accuracy** - atribut indikující přesnost těchto informací (viz GGeoAddressAccuracy v online referenční příručce [9])
- **Point** - bod ve 3D prostoru.
  - **coordinates** - trojice zeměpisné délky, šířky a nadmořské výšky, v tomto případě bude nadmořská výška vždy 0.

2.8.2.2 Třída GDirectionsOption

Objekt této třídy obsahuje vlastnosti popsané v tabulce č. 2.3.

Vlastnost	Typ	Popis
locale	String	specifikuje jazyk výsledků hledání
travelMode	GTravelModes	specifikuje způsob přepravy, buď jdeme pěšky (G_TRAVEL_MODE_WALKING) a nebo nějakým motorovým vozidlem (G_TRAVEL_MODE_DRIVING)
avoidHighways	Boolean	pokud bude tento parametr nastaven (true), bude se Google Maps při směrování vyhýbat dálnicím, ale pouze v případě kdy to bude možné
getPolyline	Boolean	pokud bude tento parametr nastaven (true), metoda *load* třídy *GDirection* získá objekt třídy *GPolyline* reprezentující lomenou čáru pro vykreslení cesty na mapu, ikdyž není připojena mapa
getSteps	Boolean	pokud bude tento parametr nastaven (true), metoda *load* třídy *GDirection* získá kroky (*GStep*) podrobné navigace, ikdyž není připojena oblast na stránce ( ), kde mají být navigační kroky vypsány
preserveViewPort	Boolean	zajistí ponechání stávající pozice mapy, při výchozím nastavení se mapa zaměří na oblast obklopující vyhledanou trasu

Tabulka č. 2.3: Vlastnosti GDirectionsOption

2.8.2.3 Třídy GRoute a GStep

Objekty třídy GRoute jsou tvořeny za účelem uložení a získání informací o cestě mezi dvěmi zadanými adresami. Objekt nelze přímo vytvořit, ale lze jej získat jako výsledek metody `load` třídy GDirections (vyhledání trasy). Metoda `getStep(i:Number)` vrací i-tý krok (GStep) dané části cesty, který obsahuje souřadnice, slovní navigační popis, vzdálenost a časovou délku tohoto kroku.

2.8.3 Google Maps .NET

Ikdyž Google Maps poskytuje přehledné API s hierarchií mnoha tříd, o značné zjednodušení a zvýšenou efektivitu pro programátory ASP.NET se může postarat Reimers GoogleMaps .NET pro ASP.NET 2.0. Je to serverový ovládací prvek s plnou funkcionalitou oficiálního Google Maps API. Překonává bariéry nepřehledného a obvykle těžce laditelného Javascriptu. GoogleMaps.NET je pro nekomerční použití zdarma.

Instalace této komponenty je jednoduchá, stačí zkopírovat soubor „GoogleMap.dll“ do složky `Bin` webové aplikace. Pokud používáme vývojové prostředí Visual Studio lze přidat tuto control komponentu do nástrojového panelu. Dále je třeba deklarovat jmenný prostor `Reimers.Map` a zaregistrovat komponentu do stránky pomocí direktivy `<%@ Register Assembly="GoogleMap" Namespace="Reimers.Map" TagPrefix="Reimers" %>`. Poté s ní lze pracovat jako s jakoukoliv jinou ASP.NET komponentou. V předchozí kapitole jsem se zmínil o klíči, který musíme mít vygenerovaný pro konkrétní doménu, abychom mohli používat Google Maps. To samé platí pro využívání této komponenty. Klíč je vhodné zadat do souboru web.config a přistupovat k němu pomocí třídy ConfigurationManager ze jmenného prostoru `System.Configuration`.

```
...
  <appSettings>
   <add key="GoogleMapsKey"
value="ABQIAAAASj2H51qMXA81okWPlLla8hRoDnP5o5lzhFFR8vNtOuCyXNYL3hSGmuuVS
kNBj0rb1Vlk2Ly3AZu0hw" />
     <!-- klíč vygenerovaný pro  http://journeyplanning.aspone.cz/ -->
  </appSettings>
...
```

Ukázka č. 2.1: Doplnění souboru Web.config

Další informace lze nalézt na domovských stránkách komponenty (viz [10]).

3 Návrh aplikace

V této kapitole jsou nejdříve charakterizovány prostředky použité k návrhu systému. Dále jsou specifikovány požadavky na systém. A nakonec zde uvedu návrh databáze ve formě E-R diagramu (Entity-Relationship Diagram).

3.1 Prostředky použité pro návrh

3.1.1 UML

Unified Modeling Language (viz [7]) je v softwarovém inženýrství grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů. Zahrnuje množinu grafických notačních technik k vytváření abstraktních modelů a k detailnímu návrhu systémů pro práci v reálném čase. Standardní sémantiku UML je možno doplnit omezeními, které umožňují změnu interpretace elementů. UML je standardem přijatým kromě jiných také skupinou OMG (Object-Management Group).

3.1.2 Visual Paradigm for UML

K návrhu systému jsem použil program Visual Paradigm for UML Community Edition, který se dá označit jako CASE nástroj (viz [11]) podporující jazyk UML verze 2.1. Zkratka CASE (Computer-aided software engineering) v této oblasti znamená počítačem podporované softwarové inženýrství. CASE nástroje jsou nástroje pro podporu analýzy a návrh aplikací. Kromě podpory UML lze modelovat databáze a firemní procesy k vizualizaci a porozumění. Tato verze je pro nekomerční použití zdarma. Nespornou výhodou je to, že návrhy lze exportovat do mnoha grafických formátů.

Visual Paradigm for UML poskytuje celkem 13 druhů UML diagramů, z nichž v návrhu využiji diagram tříd (Class diagram) a digramy případů užití (Use case diagram). Tento nástroj využijí také ke konstrukci modelu databáze (E-R diagramu).

3.2 Specifikace požadavků

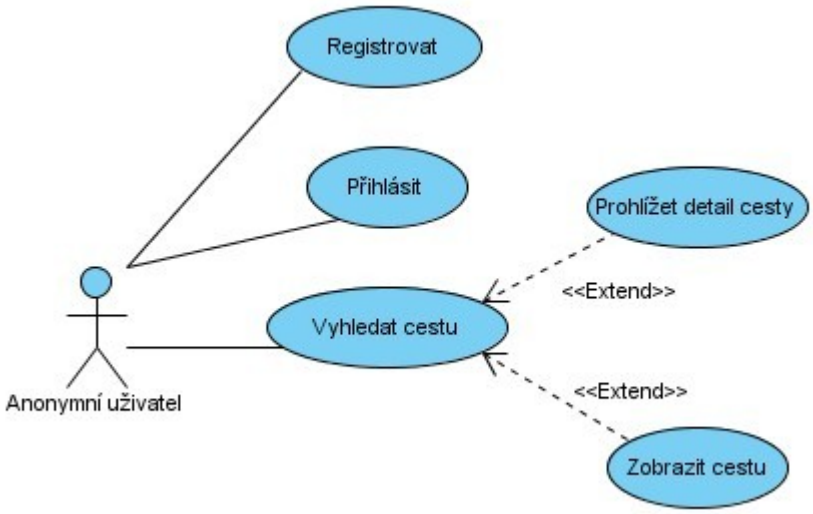
V této podkapitole se budu věnovat specifikaci funkcí a vlastností, které by měly být ve vyvíjeném systému implementovány. V podstatě jde o charakterizaci toho, co bude systém nabízet a nezajímá nás zde, jak to bude zařízeno. V jazyce UML jsou požadavky uživatele podporovány v případech užití (Use case).

3.2.1 Režimy a uživatelé systému

Aplikace může běžet ve třech režimech, které jsou rozepsány v následujících podkapitolách. Nachází se zde také Use case diagramy, které znázorňují případy užití. Hlavní funkcionalitu systému zastává režim běžného uživatele.

3.2.1.1 Režim anonymního uživatele

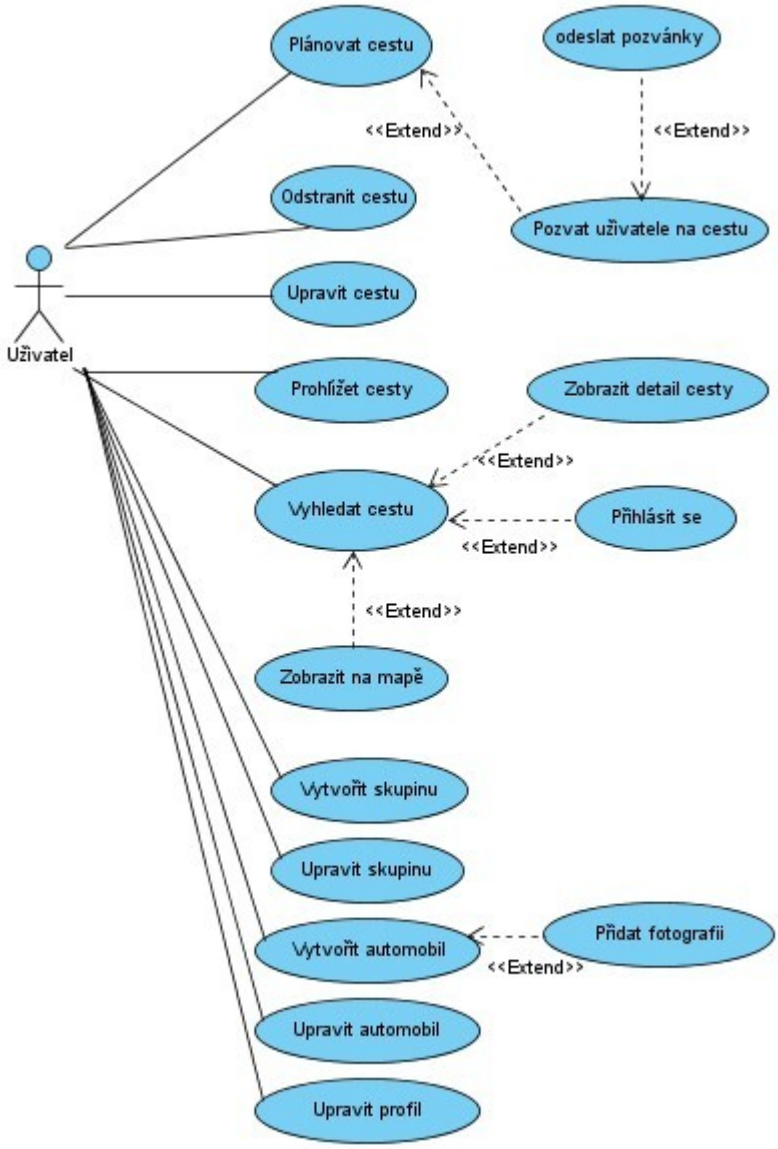
Anonymní uživatel smí vyhledávat v databázi naplánovaných cest. Dále je mu umožněno grafické zobrazení vyhledané cesty pomocí Google map, včetně detailních informací o cestě, dopravním prostředku a řidiči. V zobrazeném detailu cesty nechybí ani tabulka pozvaných a přihlášených uživatelů. Režim anonymního uživatele znázorňuje Use case diagram zobrazený na obrázku č. 3.1.



Obr. č. 3.1: anonymní uživatel

3.2.1.2 Režim běžného uživatele

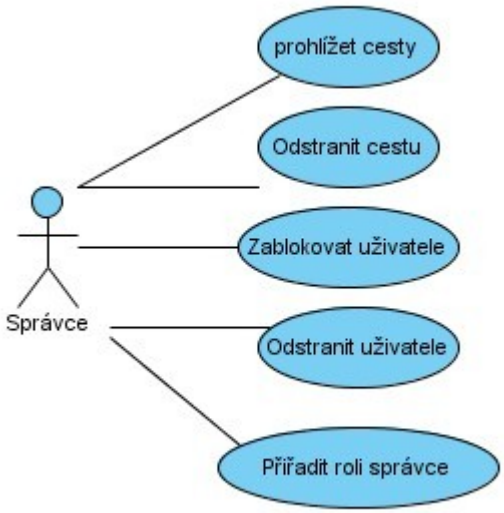
Anonymní uživatelé se musí nejprve zaregistrovat do systému. Zadat uživatelské jméno, email a heslo. Poté mohou poskytnout informace o své osobě a vytvořit si tak svůj profil. Do systému se budou přihlašovat pomocí uživatelského jména a hesla zadaného při registraci. Po přihlášení mají možnost plánování cesty, vyhledání vhodné cesty a přihlášení se na ni. Mohou vytvářet skupiny lidí, kterým budou zasílat pozvánky na cestu, tak aby nemuseli v příští relaci znovu vyhledávat často zvané účastníky. Grafické zobrazení vyhledané cesty pomocí map včetně detailních informací je pro přihlášeného uživatele samozřejmostí. Režim běžného uživatele znázorňuje Use case diagram zobrazený na obrázku č. 3.2.



Obr. č. 3.2: běžný uživatel

3.2.1.3 Režim správce systému

Správce systému má přístup ke všem informacím, může mazat naplánované cesty, přiřadit jiného uživatele do role správce, zablokavat nebo odstranit uživatelský účet. Dále má k dispozici všechny funkce, které jsou systémem nabízené běžnému uživateli.



Obr. č. 3.3: správce systému

3.2.2 Specifikace případu užití

Pro specifikaci případů užití bohužel neexistuje žádný standard UML. Obvykle se však používá šablona (tabulka) znázorněná v ukázce č. 3.1. Vedlejší scénář (NeexistenceAutomobilu) v tomto případě znázorňuje alternativní cestu tokem událostí a nastává v případě, kdy uživatel plánující cestu nemá předem definovaný automobil. Proto musí v průběhu plánovaní cesty vložit informace o automobilu, který chce pro cestu využít.

Případ užití: PlánovatCestu
ID: UC1
Účastníci: Uživatel
Vstupní podmínky: 1. Uživatel je přihlášen do systému.
Hlavní scénář: 1. Případ užití začíná v okamžiku, kdy uživatel zadá příkaz „plánovat cestu“. 2. Systém zobrazí formulář obsahující dvě vstupní pole pro počáteční a cílovou stanici (obec). 3. Dokud nemá trasa cesty takový průběh, jaký si uživatel přeje.

3.1 Uživatel zvolí počet míst, kterými chce cestu vést. Zadá tyto místa a potvrdí. 3.2 Systém pošle požadavek službě Google Maps. 3.3 Systém zobrazí cestu na mapách 4. Uživatel vyplní zbývající vstupní pole formuláře zahrnující: ceny do jednotlivých destinací, datum cesty, výběr automobilu atp. 5. Systém uloží cestu a pokračuje ve scénáři, který je specifikován případem užití UC3 (Pozvat uživatele na cestu).
Vedlejší scénář: NeexistenceAutomobilu
Následné podmínky: 1. Cesta je uložena a k dispozici pro prohlížení.

Ukázka č. 3.1: Detail případu užití

Případ užití: PlánovatCestu **Vedlejší scénář: Neexistence Automobilu**
ID: UC2
Účastníci: Uživatel
Vstupní podmínky: 1. Uživatel je přihlášen do systému
Vedlejší scénář: 1. Případ užití začíná ve 4. kroku případu užití PlánovatCestu. 2. Systém zobrazí formulář pro zadaní informací o novém automobilu. 3. Uživatel vyplní formulář a potvrdí. 4. Systém zobrazí stránku s možností nahrání fotky a zobrazení detailu automobilu. 5. Uživatel zadá „pokračovat“. 6. Případ užití pokračuje v 5. kroku případu užití PlánovatCestu.
Následné podmínky: 1. Automobil je uložen.

Ukázka č. 3.2: Detail případu užití (vedlejší scénář)

3.2.3 Nastavení dostupnosti cesty pro ostatní uživatele

Uživatel plánující cestu bude mít možnost nastavit přístupnost cesty:

- **Pro všechny** - cesta se stane veřejně dostupnou s určitým počtem volných míst. Poté může být cesta vyhledána a obsazena i nepozvanými uživateli.
- **Jen pro pozvané** - uživatel musí vytvořit pozvánky a odeslat je, nebo může rovnou přihlásit vybrané lidi na cestu. Na cestu se mohou přihlásit jen pozvaní.

3.2.4 Grafické zobrazení cesty

Požadavek bude splněn pomocí vestavěného mapového systému Google Maps. Systém zobrazí naplánovanou trasu pomocí lomené čáry.

3.2.5 Vytvoření skupiny

Skupina slouží ke sdružení lidí, kteří spolu obvykle sdílí cestu. Správcem skupiny bude automaticky ten, kdo ji vytvořil. Při pozvání skupiny na cestu jsou vytvořeny pozvánky pro všechny členy skupiny. Uživatel může upravovat a mazat svoje skupiny.

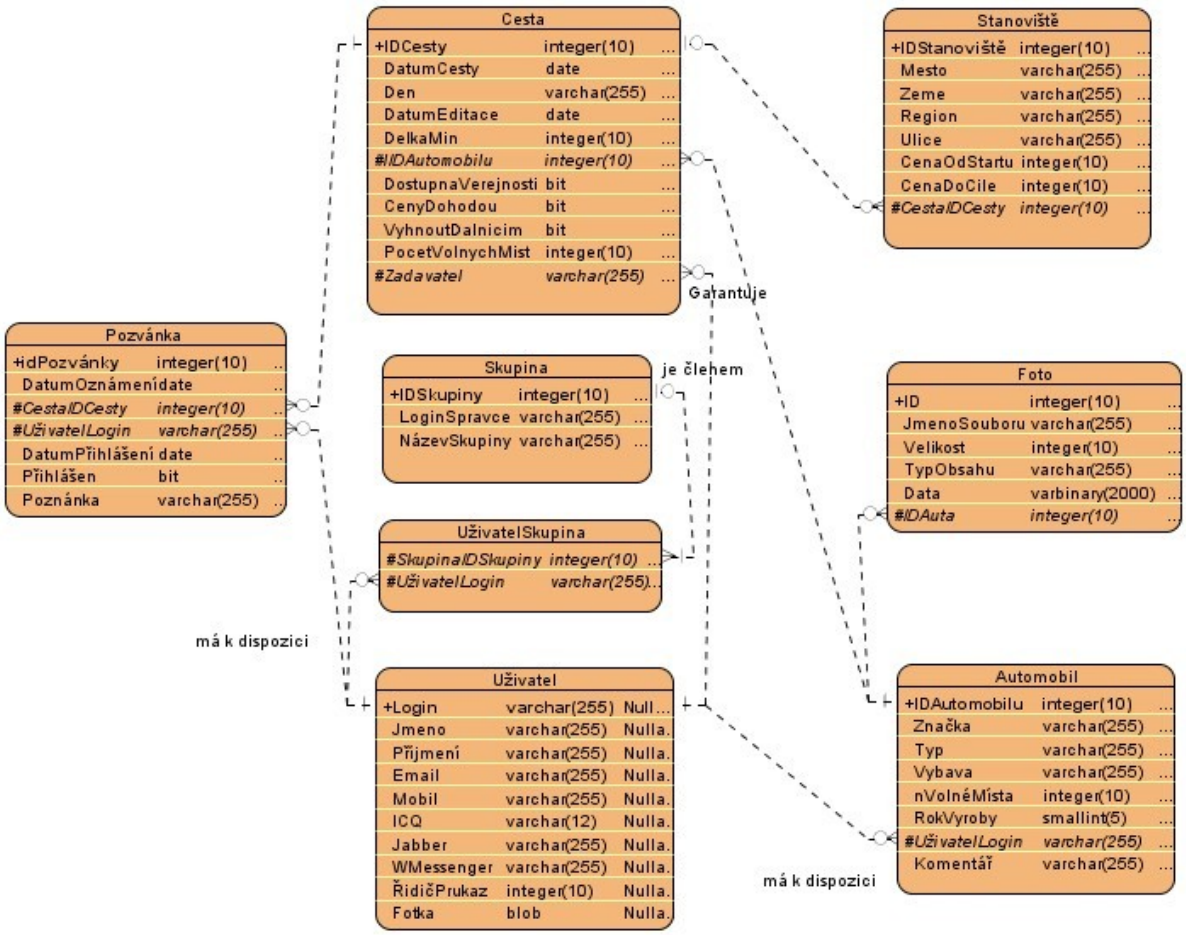
3.2.6 Vytvoření automobilu

Systém bude umožňovat předem nebo i během plánování zadat informace o automobilu a nahrát fotky automobilu. Uživatel může mít v systému zaevidovaných více automobilů. Detail automobilu potom bude použit jako součást detailu cestu, tak aby učástník cesty měl „představu“ o automobilu, kterým bude cestovat.

3.3 Návrh databáze

Schéma databáze bylo navrženo na základě funkčních požadavků na systém. Je zde zobrazeno pomocí E-R Diagramu (Entity Relationship Diagram) vytvořeného nástrojem Visual Paradigm. V další podkapitole se věnuji popisu jednotlivých entit (tabulek) relační databáze.

3.3.1 E-R Diagram



Obr. č. 3.3: Schéma databáze (E-R Diagram)

3.3.1.1 Popis entitních množin E-R diagramu

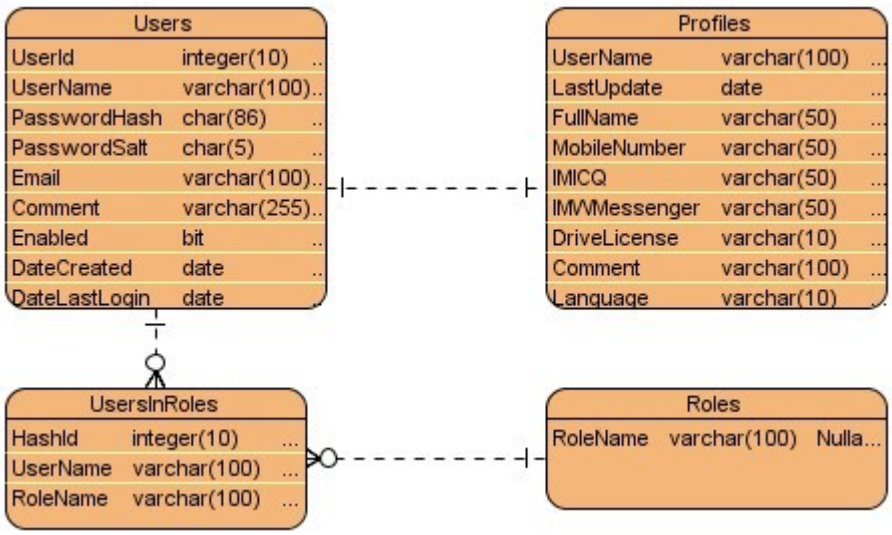
Hlavní entitní množina `Cesta` specifikuje atributy pro uchovávání informací o jednotlivých naplánovaných cestách. Obsahuje cizí klíč `IdAutomobilu`, který jednoznačně identifikuje automobil použitý na cestě. Dále obsahuje cizí klíč `Zadavatel` znázorňující přihlašovací jméno uživatele, který cestu naplánoval.

Entitní množina `Stanoviště` specifikuje atributy pro uchovávání informací o zastávkách na cestě. Kromě atributů zobrazených v diagramu obsahuje `GeoPozici` pro uchování geografických souřadnic zastávky. V programu Visual Paradigm nelze vytvořit atribut typu Geography, proto jsem `GeoPozici` v diagramu vynechal.

Entitní množina `Pozvánka` specifikuje atributy pro uchovávání informací o účásti uživatele na cestě. Atribut `Přihlášen` určuje, zdali je uživatel na cestu přihlášen. Pokud je nastaven na bitovou nulu (false), potom je cestující pouze pozván, ale doposud nepřihlášen.

Existence entitní množiny `Skupina` a vazební entitní množiny `UživatelSkupina` umožňuje sdružovat uživatele do skupin, které obsahují atributy `LoginSprávce` a `NázevSkupiny`.

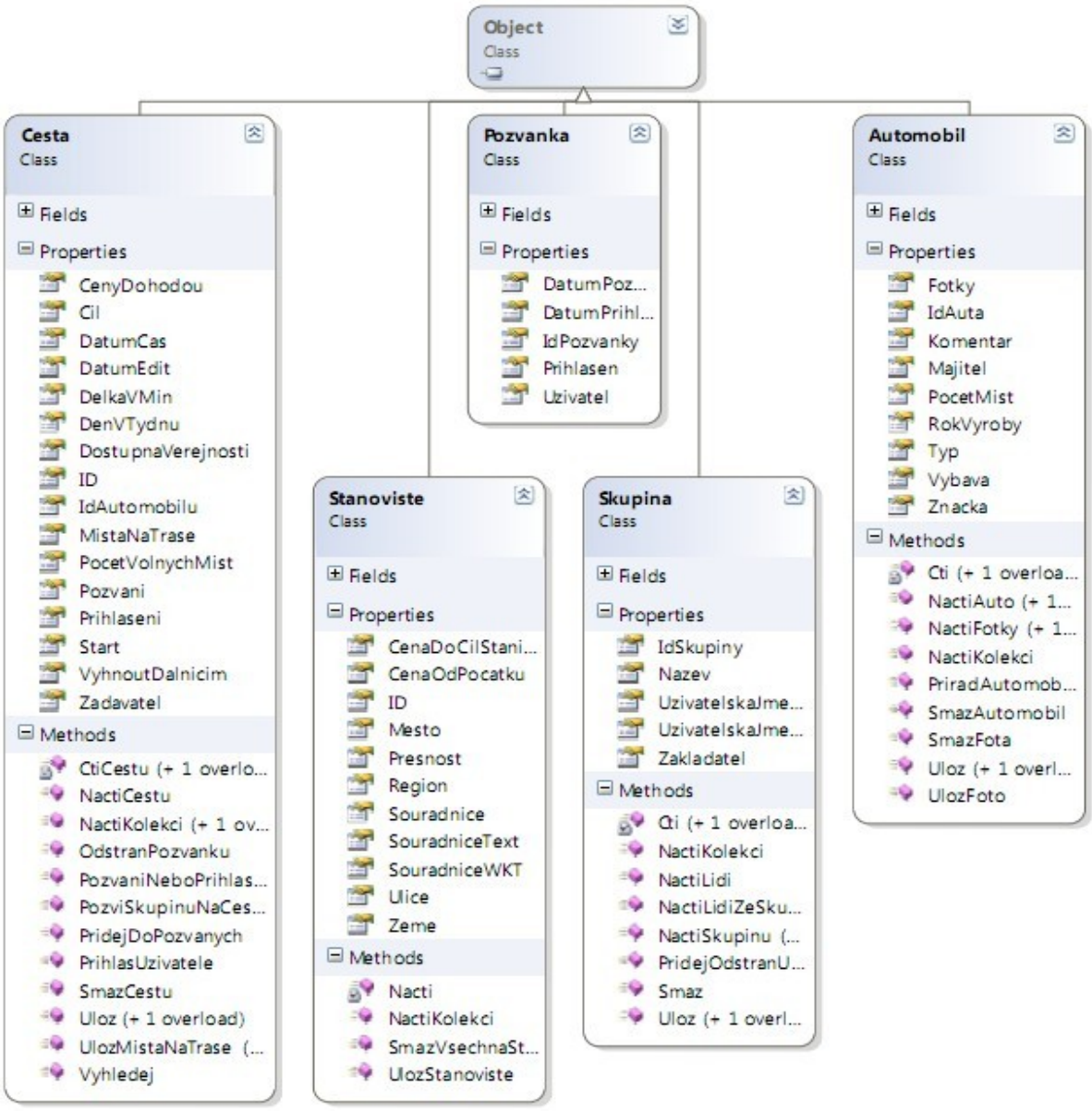
Entitní množina `Uživatel` v uvedeném E-R diagramu zastupuje serii tří entitních množin (`Users, Profiles, Roles`) a vztahové entitní množiny `UsersInRoles`. Při implementaci (viz podkapitola 4.2) jsem tyto množiny, které jsou součástí sady poskytovatelů Altairis(viz[14]) pro správu uživatelských účtů, začlenil do mé databazové struktury. Schéma těchto tabulek je na obrázku č. 3.4.



Obr. č. 3.4: Schéma začleněných tabulek (E-R Diagram)

3.4 Diagram tříd

Diagram zobrazuje 5 hlavních tříd aplikační vrstvy třívrstvé architektury. Aplikační vrstva tvoří základní logiku a funkčnost celého systému. Transformuje data pro zobrazení prezentační vrstvou aplikace. V aplikační vrstvě se kromě tříd z tohoto diagramu vyskytují třídy typu kolekce (`KolekceAutomobilu`, `KolekceCest`, `Kolekce Stanovist`, `KolekceSkupin`), které můžou sloužit například jako zdroj dat ASP.NET prvku DataGrid v prezentační vrstvě. Popis třívrstvé architektury lze nalézt v kapitole 4.



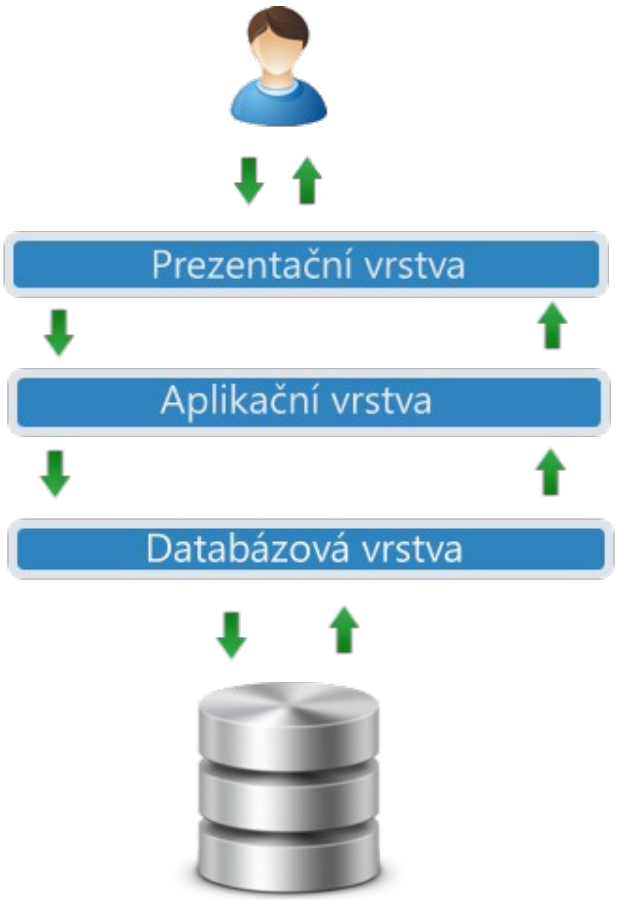
Obr. č. 4.1 Diagram tříd aplikační vrstvy

4 Implementace

Tato kapitola se zabývá konstrukcí aplikace a konkrétními postupy při implementaci jednotlivých vlastností aplikace. Jsou zde popsány vytvořené stránky, uživatelské serverové komponenty, použitá architektura vrstev, jazyková lokalizace systému a odesílání emailů. Systém jsem implementoval v ASP.NET. Zvolil jsem programovací jazyk C# a vývojové prostředí Microsoft Visual Studio 2008.

4.1 Architektura aplikace

Při implementaci jsem se snažil dodržovat zásady třívrstvé architektury, které zajišťují lepší přehlednost a rozšířitelnost aplikace. Tento přístup logicky rozděluje aplikaci na několik částí (vrstev). V podkapitolách popisuji jednotlivé vrstvy aplikace, které jsou ve Visual Studiu repzentovány samostatnými projekty (`ClassLibrary`) a sdružovány v `Solution`. Schéma třívrstvé aplikace je zobrazeno na obrázku č. 4.1.



4.1.1 Prezentační vrstva

Prezentační vrstva tvoří rozhraní, pomocí kterého uživatelé komunikují s aplikaci. Není přímo zavislá na datech nebo databázi, ale využívá rozhraní veřejných tříd a metod aplikační vrstvy. Základním požadavkem na tuto vrstvu je bezpečné zpracování vstupních dat a uživatelsky přívětivé grafické prostředí. Prezentační vrstva může být tvořena webovými formuláři (stránkami), formuláři windows (WinForms) nebo jinými prostředky. V systému, kterým se zabývá tato práce, je realizována jako ASP.NET webové stránky (`Web site` ve Visual Studiu 2008). Stránky mají základ grafického rozhraní definovaný v MasterPage a využívají uživateské serverové komponenty. Vzhled stránek je dotvářen pomocí CSS (kaskádových stylů) a systému témat (Themes), které jsou podporovány v ASP.NET a propojují CSS styl s konkrétním typem serverové komponenty. Výhoda tohoto přístupu je v tom, že není třeba při každém použití stejného prvku definovat CSS styl pomocí vlastnosti `CssClass`. Stačí to udělat jednou pro každý typ prvku za pomoci Themes.

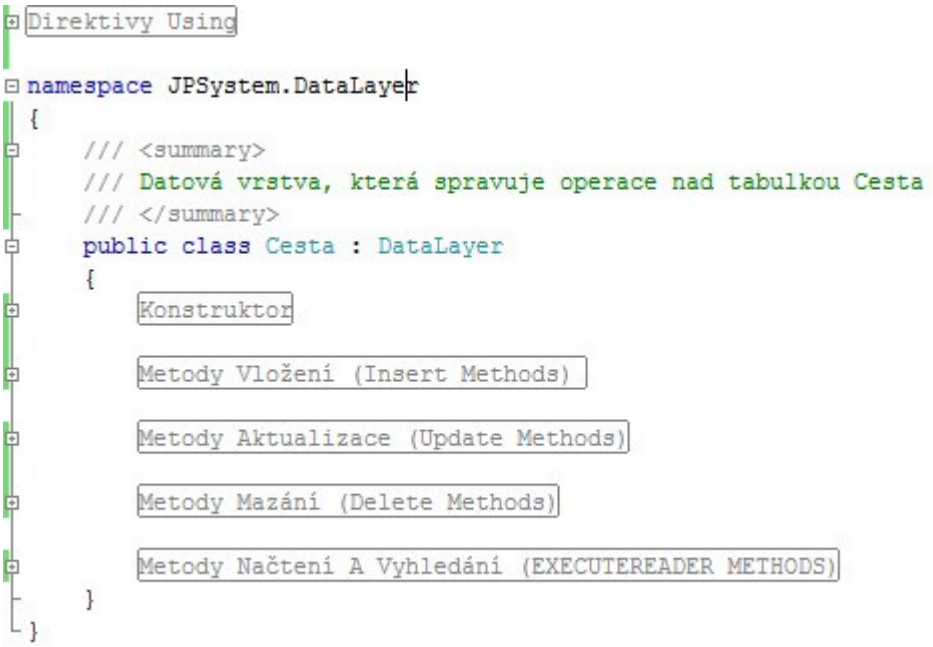
V některých částech jsou webové stránky doplněny volně dostupnými prvky ze sady AJAXControlToolkit. Bližší informace k implementaci prezentační vrstvy aplikace se nachází v podkapitole 4.4 (Stránky a komponenty aplikace).

4.1.2 Aplikační vrstva

V projektu aplikační (Business) vrstvy jsou sdruženy třídy poskytující hlavní funkcionalitu systému. Aplikační vrstva definuje rozhraní veřejných tříd a jejich metod, které slouží převážně k získávání a zadávání dat do databáze. Patří do ní třídy z digramu tříd zobrazeném v předchozí kapitole a kolekce těchto tříd. Kolekce aplikační vrstvy (`KolekceCest`, `KolekceSkupin`, atp.) slouží převážně jako zdroj dat pro zobrazení ASP.NET prvkem DataGrid v prezentační vrstvě. Podrobné informace ke třídám této vrstvy lze nalézt v programové dokumentaci ve jmenném prostoru `JPSystem.BusinessLayer`.

4.1.3 Databázová vrstva

Databázová vrstva poskytuje rozhraní mezi aplikační vrstvou a databází. Řídí základní operace nad databází. Metody tříd, které se nachází v této vrstvě, volají uložené procedury databáze a vytváří parametry využívané těmito procedurami. K tomu využívají pomocnou třídu `DBHelper` (viz[13]). Tato třída otevírá a uzavírá spojení s databází, nastavuje `ConnectionString` umístěný v souboru "Web.config", spravuje parametry uložených procedur a poskytuje metody pro čtení dat. Struktura třídy z databázové vrstvy je zobrazena v ukázce č. 4.2. Podrobné informace ke třídám této vrstvy lze nalézt v programové dokumentaci ve jmenném prostoru `JPSystem.DataLayer`.



Ukázka č. 4.2: Struktura třídy v databázové vrstvě

4.2 Databáze

Jak jsem již zmínil, uložení dat je realizováno Microsoft SQL Serverem 2008. Struktura tabulek databáze byla vytvořena na základě E-R diagramu uvedeného v předchozí kapitole. Aplikace nevyužívá samostatné SQL dotazy, ale veškerá manipulace s daty a dotazování je řešeno v celkem 33 uložených procedurách. Například pro operace s daty nad tabulkou `Cesta` databázová vrstva využívá zde uvedené uložené procedury:

- `JPS_JOURNEY_DELETE` – odstranění řádku podle identifikátoru cesty.
- `JPS_JOURNEY_LOAD` – načtení řádku podle indentifikátoru cesty.
- `JPS_JOURNEY_SAVE` – uložení řádku.
- `JPS_JOURNEY_UPDATE` – uprava řádku.
- `JPS_JOURNEYS_LOAD` – načtení kolekce řádků (cest). Tato procedura umožňuje stránkování na straně databáze, proto je nutné vstupním parametrem dodat požadované číslo stránky a počet řádků na stránku. Procedura tak vráti pouze určený počet řádků a toho lze využít při prezentaci dat uživateli, kde je možné zobrazit jen malé množství řádků kvůli přehlednosti a rychlosti aplikace.
- `JPS_JOURNEYS_SEARCH_BY_DISTANCE` – vyhledání cesty podle počáteční a cílové stanice. Hledá i „podtrasy“ (části cest) ve všech cestách. Při hledání využívá statickou vestavěnou metodu `STDistance` SQL Serveru pro určení vzdálenosti dvou geografických objektů, která je použita nad sloupci `GeoPozice` tabulky `Cesta`. Díky tomu lze

vyhledávat trasy se zadanou tolerancí vzdálenosti. Například vyhledání trasy s počáteční stanicí vzdálenou maximálně 20 km od města Fryšták. Stejně jako předchozí procedura umožňuje stránkování dat v databázi.

4.3 Common projekt

Common projekt je balíček tříd, které jsou umístěny v samostatném projektu (ClassLibrary) stejně jako každá z předchozích vrstev. Jsou v něm obsaženy obecné třídy včetně tříd, které je třeba sdílet mezi více vrstvami. Vložil jsem zde třídu `Email`, která slouží k vytvoření HTML e-mailů a odeslání (viz podkapitola 4.4). Dále se zde nachází třída `ZakladKolekce`, od které jsou odvozené všechny kolekce aplikační vrstvy (`KolekceCest`, `KolekceStanovist`, `KolekceAutomobilu`, `KolekceSkupin`). Tato třída definuje základní vlastnosti potřebné pro stránkování (číslo aktuální stránky, počet položek na stránku, počet všech položek a počet stránek).

4.4 Stránky a komponenty aplikace

Prezentační vrstva aplikace je realizována ASP.NET stránkami, které ve většině případů používají uživatelské serverové komponenty nacházející se v adresáři UserControls. Výhodou uživatelských komponent je hlavně jejich znovupoužitelnost. Uživatelskou komponentu lze vkládat do stránky stejně jako kteroukoliv jinou ASP.NET komponentu, je však nutné ji zaregistrovat pomocí direktivy `<%@ Registr %>` na začátku stránky. Pro použití ve stránkách aplikace jsem vytvořil tyto komponenty:

- **CarDetail** – slouží k zobrazení informací o automobilu, včetně fotografie.
- **UserDetail** – slouží k zobrazení profilových informací uživatele.
- **JourneyDetail** – slouží k zobrazení informací o cestě, zahrnuje mimo jiné komponenty CarDetail, UserDetail a InvitationList.
- **JourneyList** – stránkovatelný seznam cest s tlačítky pro smazaní, přihlášení na cestu, editaci a detail cesty.
- **SelectUser** – stránkovatelný seznam uživatelů s tlačítky pro pozvání uživatele na cestu.
- **InvitationList** – slouží k zobrazení pozvaných a přihlášených uživatelů na cestu.

4.4.1 Master page

K zachování jednotného uživatelského rozhraní na všech implementovaných stránkách využívám tzv. MasterPage. MasterPage, která je definována v souboru s názvem MasterPage.master, je rozdělena na hlavičku, obsah a zápatí. Hlavička obsahuje menu, panel pro přepínání jazyků a dva prvky, z nichž jeden (LoginName) zobrazuje jméno přihlášeného uživatele a druhý (LoginStatus) tlačítko pro přihlášení respektive odhlášení ze systému. V druhé sekci MasterPage se vyskytuje komponenta ContentPlaceHolder, na jejíž místo se po začlenění MasterPage vloží obsah komponenty Content konkrétní stránky.

4.4.2 Plánování cesty

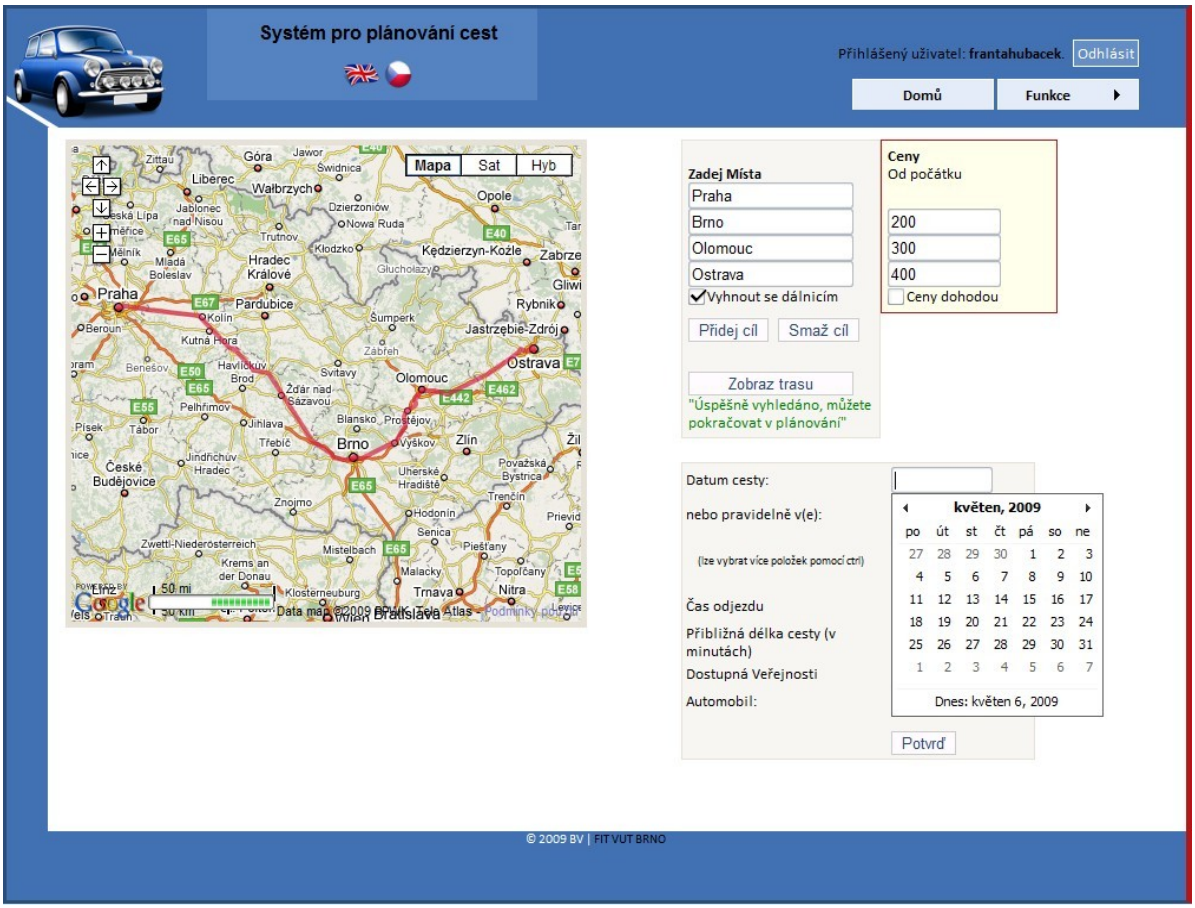
Na stránce pro plánování a editaci cesty (PlanJourney.aspx) se nachází panel, který slouží k určení „průběhu trasy“. Panel umožňuje dynamicky přidávat vstupní pole (Textboxy) pro zadání měst nebo obcí. Používá komponentu UpdatePanel z balíčku Microsoft AJAX Extensions, která zajišťuje obnovení pouze části stránky umístěné uvnitř tagu (značky) `<ContentTemplate>`, viz ukázka č. 4.1.

Vlastnost `UpdateMode` této komponenty je nastavena na hodnotu `Conditional`, proto je UpdatePanel aktualizován pouze v případě, kdy je postback vyvolán některým z tlačítek uvnitř tagu `<ContentTemplate>`. Prvek PlaceHolder slouží jako kontejner, do kterého jsou dynamicky přidávany vstupní textová pole při zpracování stránky na serveru (`postbacku`). Aby UpdatePanel fungoval tak jak má, musí být na stránce přítomen prvek ScriptManager, který je nezbytný pro běh všech AJAX komponent. ScripManager je nevizuální serverový ovládací prvek zastoupený třídou stejného jména. Stará se o připojení potřebných skriptů, jenž zajišťují obsluhu AJAX objektů na straně klienta.

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
  <ContentTemplate>
    <asp:PlaceHolder ID="myPlaceHolder" runat="server"></asp:PlaceHolder>
    <asp:CheckBox ID="CheckBoxAvoidHighways"
     Text="<%$ Resources: AvoidHighways %>" runat="server" /><br />
    <asp:Button ID="btnAddTextBox" runat="server"
     Text="<%$ Resources: Glossary, btnAddTarget %>" OnClick="btnAddTextBox_Click" />
    <asp:Button ID="btnDelete" runat="server"
     Text="<%$ Resources: DeleteTarget %>" OnClick="btnDelete_Click" /><br /><br />
    <asp:Button ID="btnFindRoute" runat="server"
     Text="<%$ Resources: ViewJourney %>" OnClick="btnFindRoute_Click" Width="140px"/>
    <br />
    <asp:Label ID="lblSuccess" runat="server" ForeColor="Red" ></asp:Label>
  </ContentTemplate>
</asp:UpdatePanel>
```

Další součástí stránky „PlanJourney.aspx“ je ASP.NET komponenta pro práci s Google mapami, která umožňuje grafické zobrazení plánované trasy. Tato komponenta má taktéž „ajaxové“ chování. Po kliknutí na tlačítko pro zobrazení trasy, je zaregistrován klientský skript (JavaScript) pro vykreslení trasy na mapě pomocí metody `RegisterStartupScript` třídy `ScriptManager`. Nedochází tak k přepisu celé stránky a chování aplikace se jeví uživatelsky přívětivější.

Služba Google Maps bohužel neposkytuje objekt typu Polyline (křivku) pro grafické znázornění cesty, která by „hladce kopírovala“ záhyby cest na mapách. Tato funkce je zatím dostupná jen v některých zemích, například v USA. Proto lomenou čáru sestavuji z geografických údajů navigačních informací. Tyto geografické souřadnice jsou přítomny u každého navigačního kroku, který je vrácen službou GoogleMaps. Po vykreslení je mapa zaostřena na zobrazovanou cestu. Výsledné rozpoložení grafického uživatelského rozhraní této stránky je zobrazeno na obrázku č. 4.2.



Obr. č. 4.2 : Screenshot aplikace (plánování cesty)

4.4.3 Vytvoření pozvánek

Ve stránce „PlanJourney2.aspx“ jsou použité tři předem vytvořené uživatelské serverové prvky. Prvek definovaný v souboru „SelectUser.ascx“ slouží k zobrazení seznamu uživatelů, ze kterého lze zvát na cestu. K stejnému učelu se na stránce nachází panel, který obsahuje vstupní textové pole propojené s prvkem AutoCompleteExtender ze sady ASP.NET AJAX Control Toolkit. AutoCompleteExtender průběžně zobrazuje „vyskakující“ panel s uživatelskými jmény z databáze vybranými podle prefixu zapsaného v textovém poli. Uživatelská jména odpovídající tomuto prefixu vrací metoda `GetLogins` definovaná v rámci webové služby „UserSearch.asmx“. Použití AutoCompleteExtenderu je demonstrováno na ukázce č. 4.2.

```
<asp:TextBox ID="tbUserSelect" runat="server"></asp:TextBox>
<cc1:AutoCompleteExtender
   ID="AutoCompleteExtenderUserSelect" runat="server"
   TargetControlID="tbUserSelect"
   ServiceMethod="GetLogins"
   ServicePath="~/UserSearch.asmx"
   MinimumPrefixLength="1"
   CompletionInterval="1000"
   EnableCaching="true">
</cc1:AutoCompleteExtender>
```

K zobrazení detailních informací o vybraném uživateli je na této stránce použit prvek „UserDetail.ascx“. Poslední uživatelskou serverovou komponentou na stránce je „InvitationList.ascx“, který zobrazuje pozvané a přihlášené uživatele a tlačítka pro zrušení pozvánek.

4.4.4 Správa automobilů

Správa automobilů je implementována stránkami „Cars.aspx“ a „Upload.aspx“. Na stránce „Cars.aspx“ je prvek DataGrid pro zobrazení uživatelem zadaných automobilů. Při načtení stránky je volána metoda `NaplnDataGrid`, která vytvoří objekt třídy `KolekceAutomobilu` a naplní tuto kolekci pomocí metody `NactiKolekci` třídy Automobil. Tato kolekce je nakonec přiřazena vlastnosti `DataSource` prvku DataGrid a data jsou „svázána“ s prvky DataGridu metodou `DataBind`. Obdobným způsobem je řešeno zobrazení kolekce cest, skupin a stanovišť na jiných stránkách. Pro vytvoření, respektive editaci informací o automobilu, se na stránce vyskytuje formulář. Stránka “Upload.aspx“ slouží k nahrání fotky automobilu a zobrazení detailu automobilu s využitím komponenty `CarDetail`.

4.5 Uživatelské účty

Ke správě uživatelských účtů jsem využil aplikační rozhraní `ASP.NET Membership`. Funkce tohoto API volají konkrétní implementaci Membership providerů (poskytovatelů) nastavených v konfiguračním souboru `web.config`. Membership provider je třída, která obsahuje implementace funkcí z `Membership` a rozhoduje o tom, kde budou data uložena. Obdobně pro vytváření rolí a přidělování rolí uživatelům je v ASP.NET k dispozici `Role Manager (Roles)`, který vyžaduje Role providera. Třetí částí tohoto modelu jsou `Profiles` pro správu profilových informací uživatele.

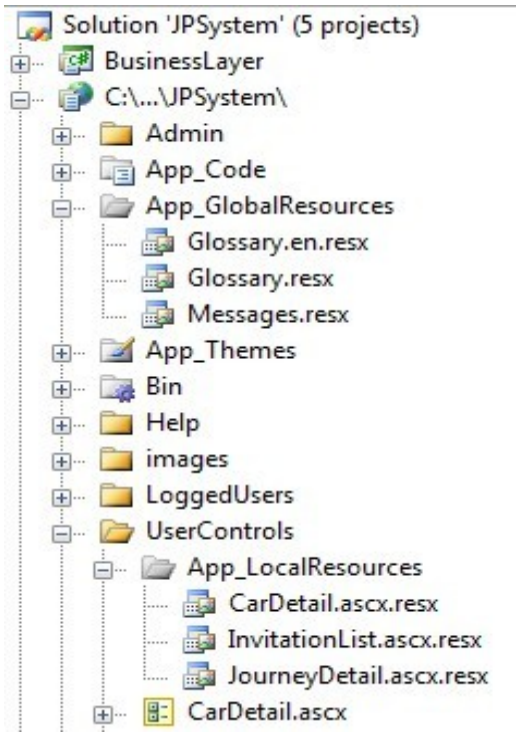
Vestavěná sada SQL providerů v ASP.NET 2.0 byla navržena tak, aby byla maximálně univerzální. Je ale velmi rozsáhlá. Vyžaduje komplikovanou databázovou strukturu a spoustu (cca 50) uložených procedur v databázi. Pokud z důvodu nevhodnosti nebo rozsáhlosti nechceme použít tyto výchozí providery, můžeme si buď napsat vlastní a nebo využít některé alternativní.

Pro práci s uživatelskými účty jsem využil Altairis Simple ASP.NET SQL Providers (viz[14]). Tato sada providerů vyžaduje 4 tabulky v databázi se strukturou, která je zobrazena na obrázku č. 3.3 v předchozí kapitole. Tyto 4 tabulky jsem propojil se zbytkem struktury databáze.

4.6 Jazyková lokalizace aplikace

Jazyková lokalizace aplikace v ASP.NET spočívá ve vytvoření a použití speciálních zdrojových souborů (tzv. Resources). Jsou to soubory, které obsahují lokalizované texty nebo i jiné objekty (obrázky, zvuky, ikony, soubory). Třídy pro správu a přístup k těmto zdrojovým souborům se nachází ve jmenném prostoru `System.Resources`. Při tvorbě stránek je možné se na tyto soubory odkazovat.

Rozlišujeme dva typy Resources. Prvním typem jsou globální Resources, které se umísťují do složky `App_GlobalResources`. Druhým typem jsou lokální Resources, které jsou přístupné jen stránkám nebo serverovým prvkům, kterých se týkají. Umísťují se do adresáře `App_LocalResources`, který se nachází v adresáři stránky respektive uživateské komponenty. Při implementaci mého systému jsem kombinoval oba typy. Na obrázku č. 4.3 lze vidět umístění Resources v rámci `Visual Studio Solution`.



Obr. č. 4.3: Globální a lokální Resources

4.6.1 Použití lokalizace

Na Resources se lze odkázat v jakémkoliv ASP.NET serverovém ovládacím prvku, který má nastavitelnou vlastnost typu `String` (většinou „Text“). Například zde uvedu odkaz na položku v globálním Resources se jménem `Slovnik` v ovládacím prvku Label. Druhý řádek ukázky č. 4.1 přistupuje k položce `DetailAuta` lokalního Resources. Je odkazován z prvku Localize, který je určen přímo k tomuto účelu.

```
<asp:Label  Text="<%$ Resources: Slovnik, Hlavicka %>"  runat="server" />
<asp:Localize Text="<%$ Resources: DetailAuta %>"  runat="server" />
```

Ukázka č. 4.1: Odkazy na resources

K položkám globálních Resources lze navíc přistupovat pomocí typovaných vlastností tříd ve jmenném prostoru `System.Resources`. Například „Resources.Slovnik.Hlavicka“ vrátí položku `Hlavicka` z globálního Resources (`Slovnik`). K lokálním nelze takto přistupovat. Lze využít metodu `GetLocalResourceObject` třídy HttpContext. Více informací (viz [15]).

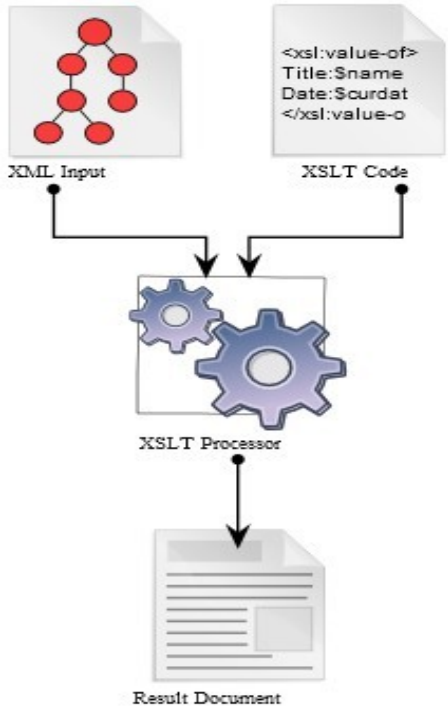
4.6.2 Uživatelská volba jazyka

Uživatelská volba jazyka se ukládá do Cookies na straně klienta, poté co uživatel klikne na ikonu (vlaječku) znázorňující kulturu dané oblasti. Při každém požadavku klienta na stránku proběhne funkce `Application_BeginRequest` definováná v souboru „global.asax“, která z Cookies přečte uživatelskou volbu a přepne lokalizaci aplikace.

4.7 Odesílání pozvánek

Systém umožňuje automatické generování a odesílání emailů lidem pozvaným na cestu. Na žádost uživatele se z pozvánek, které jsou uložené v databázi, vytvoří XML dokument s potřebnými daty. Tento dokument je využit při generování HTML emailu. Následuje XSL transformace dokumentu podle předem vytvořené XSLT šablony.

Extensible Stylesheet Language Transformations (XSLT) je standard pro převod dat ze zdrojových dokumentů XML do jiných formatů podle určitých pravidel. Výstupní dokument nemusí striktně dodržovat požadavky XML standardu, ale může jím být holý text nebo HTML dokument. Stěžejní třídou pro provádění XSL transformací je v knihovně tříd .NET frameworku třída XslTransform ve jmenném prostoru `System.Xml.Xsl`. Zdrojová XML data lze doplnit předáním informací pomocí parametrů. K tomu slouží metoda `AddParam` třídy XsltArgumentList jmenneho prostoru `System.Xml.Xsl`. Na diagramu znázorněném na obrázku č. 4.4 je zobrazeno, jak XSL transformace probíhá.



Obr. č. 4.4: Diagram základních prvků a toku procesů XSLT (viz[16])

Před odesláním e-mailu je vytvořen XML dokument s informacemi, které se po odeslání vyskytnou v tomto e-mailu. Příklad XML zdroje lze vidět v ukázce č. 4.2. Důležitým prvkem je ID (identifikátor cesty), jehož hodnota je při XSLT převodu „vytažena“ do výsledného odkazu v e-mailu. Tento odkaz slouží k přihlášení uživatele na cestu.

```
<?xml version="1.0" encoding="utf-8"?>
<Pozvanka>
    <Pozvany>
        <Login>janstrejcek</Login>
    </Pozvany>
    <Odesilatel>
        <Login>frantahubacek</Login>
        <Email>frantahubacek@tiscali.cz</Email>
    </Odesilatel>
    <Cesta>
        <Start> Praha </Start>
        <Cil> Hodonín </Cil>
        <Datum>4/28/2009 1:07:15 PM</Datum>
        <ID>2147483647</ID>
    </Cesta>
</Pozvanka>
```

Ukázka č. 4.2: Zdrojový XML soubor

Příklad XSLT šablony lze vidět na ukázce č. 4.3. V této šabloně se vyskytuje předpis zapsaný tagem `<xsl:template/>` pro každý hlavní uzel z XML (`Pozvaný, Odesílatel, Cesta`). Jakmile XSLT procesor začne zpracovávat příkaz zapsaný tagem `<xsl:apply-templates/>`, vyhledá a použije vhodný předpis podle atributu `match` tohoto tagu. V případě, že bude vyžadováno odesilání e-mailů v jiném jazyce, stačí vytvořit XSLT šablonu v tomto jazyce a použít ji.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt" exclude-result-prefixes="msxsl">
    <xsl:output method="xml" indent="yes" encoding="utf-8"/>
    <xsl:template match="/">
        <html><head/>
            <body>
                <xsl:apply-templates/>
            </body>
        </html>
    </xsl:template>
    <xsl:template match="Pozvanka/Pozvany">
        <h1>
            Dobrý den <xsl:value-of select="Login" />
        </h1>
    </xsl:template>
    <xsl:template match="Pozvanka/Odesilatel">
        <p>Uživatel <strong><xsl:value-of select="Login"/></strong>(<xsl:value-of select="Email"/>)
        Vás pozval na cestu: </p><br></br>
    </xsl:template>
    <xsl:template match="Cesta">
        <p><strong><xsl:value-of select="Start" /> >>>>> <xsl:value-of select="Cil" /></strong>
        </p>
        která se uskuteční dne: <xsl:value-of select="Datum"/> <BR /><BR />
        Přihlásit na cestu se můžete kliknutím na
        <a>
            <xsl:attribute name="href">http://journeyplaning.aspone.cz/default.aspx?
            idcesty=<xsl:value-of select="ID"/></xsl:attribute>tento odkaz.
        </a>
    </xsl:template>
</xsl:stylesheet>
```

Ukázka č. 4.3: XSLT šablona emailu.

5 Závěr

Cílem této práce byl návrh a vytvoření ASP.NET webové aplikace, která slouží k plánování cestovních tras s využitím technologie Google Maps. Během zpracování jsem prošel hlavními etapami vývoje webové aplikace. Věnoval jsem se popisu použitých technologií, zformuloval jsem požadavky na vývoj tak, aby výsledná aplikace splňovala zadání práce a byla použitelná pro reálné nasazení v praxi.

Ačkoliv již na Internetu existují systémy určené ke stejnému účelu, můj systém by mohl zaujmout díky tomu, že nabízí více možností pro budoucí uživatele. K těmto možnostem patří implementovaný systém pozvánek, zasílání emailů pozvaným lidem a grafické zobrazení cestovní trasy na mapě. Přinosem je i samotná technická zpráva popisující vývoj webové aplikace. Výsledný systém splňuje požadavky, které na něho byly kladeny v zadání. Věřím, že může být prospěšný nejen mnoha lidem, ale i životnímu prostředí. K jeho větší atraktivitě a lepší použitelnosti bych chtěl v budoucnu přispět případným rozšířením.

Jako možné rozšíření do budoucna se nabízí vylepšení použitelnosti panelu pro úpravu plánované trasy tak, aby bylo možné přidávat a odebírat zastávky v každém úseku, připadně pozměnit trasu přetažením zastávek přímo na mapách. O toto rozšíření jsem se již zajímal a zjistil jsem, že je realizovatelné za pomoci využívaného mapového systému Google Maps a technologie AJAX. Obecně by další vývoj aplikace směřoval k dokonalejšímu propojení s mapovým systémem a lepšímu využití uložených geografických dat, které jsou doposud využívány pouze pro vyhledávání naplánovaných cest a k zobrazení uživateli. Při označení většího dopravního uzlu by mohl systém zobrazit všechny dostupné cesty, které vedou daným uzlem.

Při budoucím vývoji systému je možné vydat se některou z možných cest. Jednou variantou je rozvoj stávajícího grafického uživatelského prostředí včetně rozšiřování aplikace tak, jak jsem zde popisoval. Další možností je naopak zjednodušení a úprava grafického uživatelského rozhraní aplikace pro webové prohlížeče moderních mobilních telefonů a PDA.

Díky této práci jsem si prohloubil znalosti týkající se vývoje webových aplikací za pomoci technologie ASP.NET včetně AJAX Extensions. Prostudoval jsem spoustu anglické i české literatury k danému tématu. Rád bych se v této technologii dále vzdělával a získával zkušenosti s využitím jazyků C# a SQL.

Literatura

[1] Burget, R., Zeman, D.: *Tvorba webových stránek*. 1.11.2008 [cit. 27.01.2009]. Dostupný z WWW: <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ITW-IT/texts>

[2] Staníček, P.: *CSS Kaskádové styly*. Brno, Computer Press, 2003, ISBN 80-7226-872-4

[3] Prosise, J.: *Programování v Microsoft .NET*. Brno, Computer Press, 2003, ISBN 80-7226-879-1

[4] MSDN: *Overview of the .NET Framework* [online]. 2009 [cit. 8.01.2009]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/zw4w595w.aspx>

[5] Lacko, L.: *Ajax Hotová řešení.* Brno, Computer Press, 2008, ISBN 978-80-251-2108-5

[6] Coles M.: *Pro SQL Server 2008 XML.* Apress, 2008, ISBN 978-1-59059-983-9

[7] Schmuller, J.: *Myslíme v jazyce UML*. Grada, 2001, ISBN 80-86815-38-2.

[8] Wikipedia: The Free Encyclopedia: *Google Maps* [online]. 2009 [cit. 12.01.2009]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Google_Maps>

[9] Google Code: *Google Maps API Reference* Dostupný z WWW: <http://code.google.com/intl/cs/apis/maps/documentation/reference.html>

[10] Reimers: *Google Map .NET Documentation* [online]. Dostupný z WWW: <http://www.reimers.dk/documentation/default.aspx>

[11] Wikipedia: The Free Encyclopedia: *Visual Paradigm for UML* [online]. 2009 [cit. 8.01.2009]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Visual_Paradigm_for_UML>

[12] Mitchell S.: *Creating a Business Logic Layer* [online]. 2006 [cit. 12.03.2009] Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/aa581779.aspx>

[13] Shyam, S.: *DbHelper for DotNet 2.0 using DbProviderFactory* [online]. 2008 [cit. 13.03.2009] Dostupný z WWW: <http://www.codeproject.com/KB/aspnet/DbHelper.aspx>

[14] Valášek, M.: *Altairis Simple ASP.NET SQL Providers* [online]. 2000-2009 [cit. 13.03.2009] Dostupný z WWW: <http://www.aspnet.cz/Articles/115-simple-providers.aspx>

[15] Pattison T., MSDN: *Resources and Localization in ASP.NET 2.0* [online]. 2009 [cit. 8.04.2009]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/magazine/cc163566.aspx>

[16] Wikipedia: The Free Encyclopedia: *XSL Transformations*[online]. 2009 [cit. 12.03.2009]. Dostupný z WWW: <http://en.wikipedia.org/wiki/XSL_Transformations>

Seznam příloh

Příloha 1

Obsah přiloženého CD

- Technická zpráva – složka obsahuje písemnou zprávu ve formátu PDF a ODT (OpenOffice).
- Dokumentace – složka obsahuje programovou dokumentaci aplikace.
- JPSystem_source – složka obsahuje zdrojové texty aplikace.
- JPSystem_website – složka obsahuje webové stránky, včetně všech potřebných souborů pro běh aplikace. Zkompilované binární soubory se nachází v podsložce Bin.
- Návod – složka obsahuje návod k instalaci na server a SQL skript pro vytvoření databáze.