Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Java rámec Stripes pro lokalizaci dopravních stanic

Zdeňka Boháčová

Diplomová práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Zdeňka Boháčová**
Osobní číslo:	**I09355**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Java rámec Stripes pro lokalizaci dopravních stanic**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

V úvodní části bude zpracován přehled nejpoužívanějších java frameworků v současné době. U jednotlivých rámců budou nastíněny charakteristické vlatnosti a zvážena vhodnost při použití v praktické části diplomové práce. Bude následovat rozbor algoritmů pro hledání nejkratší cesty v grafu. Dále bude teoreticky popsáno zvolené mapové rozhraní a jeho spojení s webovou aplikací.
Práce si klade za cíl na realizovat webovou aplikaci pro vyhledání dopravních uzlů dle zadaných parametrů. Tyto nalezené uzly pak budou využity pro vyhledání několika variant spojení přes aplikaci třetí strany. Pro realizaci bude použit webový java framework Stripes.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

Documentation [online]. 2011 [cit. 2011-10-27]. Stripes framework. Dostupné z WWW: http://www.stripesframework.org/display/stripes/Documentation
Chaps spol. s r.o. [online]. 2011 [cit. 2011-10-13]. Možnost využití odkazu na www.idos.cz. Dostupné z WWW: http://www.chaps.cz/idos-moznost-vyuziti-odkazu.aspvolba_objektu
HLINĚNÝ, Petr. Základy teorie grafů. Elportál [online]. Brno: Masarykova univerzita, 2010. [cit. 2011-10-27]. Aktualizováno: Březen 2010. Dostupné z WWW:http://is.muni.cz/elportal/id=878389 ISSN 1802-128X.
GOODRICH, Michael T a Roberto TAMASSIA. Data structures and algorithms in Java. 5th ed. New York: J. Wiley, c2010, xxii, 714 p. ISBN 04-703-9880-9.

Vedoucí diplomové práce: **Ing. Zdeněk Šilar**
Katedra informačních technologií

Datum zadání diplomové práce: **31. října 2012**
Termín odevzdání diplomové práce: **17. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2012

Prohlašuji:

Tuto práci jsem vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury.

Byla jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 23.8. 2013

Zdeňka Boháčová

Poděkování

Na tomto místě bych chtěla poděkovat svému vedoucímu Ing, Zdeňku Šilarovi za vstřícnost, rady a podnětné připomínky. Dále bych ráda poděkovala prof. Ing. Antonín Kavičkovi, PhD. za věcné připomínky ve stádiu návrhu aplikace.

Anotace

Práce se zabývá javovým frameworkem Stripes a technologií Java Enterprise Edition, která je využita pro návrh a realizaci webové aplikace vyhledávající dopravní stanice. K lokalizované stanici je vyhledána nejkratší cesta. Vyhledávání probíhá nad vlastními daty na straně databáze.

Součástí práce je seznámení s použitými technologiemi jakou jsou databáze podporující práci s geografickými daty (PostgreSQL s rozšířením PostGIS), AJAX, OpenLayers, GeoJSON. Je provedeno srovnání Stripes s vybranými zástupci frameworků.

Klíčová slova

Javový webový framework Stripes, GIS aplikace, PostGIS, openstreetmap, dikstrův algoritmus

Title

Java framework Stripes in localization of transport services

Annotation

Work is engaged in java framework Stripes and technology Java Enterprise Edition, which is used to designing and creating web application. Application finds nearest traffic stop and shortest path to this traffic stop. Searching is made on database side with own data set.

There is part of work which contains familiarization with used technologies like database systems allowing work with geographic data (PostgreSQL with PostGIS extension), AJAX, OpenLayers, GeoJSON. Stripes is compared with other frameworks too.

Keywords

Java web framework Stripes, GIS application, PostGIS, openstreetmap, dijkstra algorithm

Obsah

1 Úvod

V této kapitole je nastíněna motivace proč se danou problematikou zabývat, dále popsán cíl práce a budou zde předloženy výsledky průzkumu stávajícího řešení.

1.1 Motivace

Níže je popsána motivace z pohledu vývoje geografických informačních systémů a používání rámců (frameworků) pro webové aplikace.

Frameworky

Vývoj webových stránek je stále méně častěji pojímán jako projekt na zelné louce, tj. že by se začínalo programovat úplně od začátku. V základu jsou si všechny webové aplikace podobné – mají stejnou kostru, kterou je nějaká navigace mezi stránkami. Vyžadují také podobnou funkcionalitu jako je přihlašování, ošetření uživatelských vstupů do formulářů nebo vytvoření několika jazykových mutací.

Je jak časově tak finančně neefektivní tuto funkcionalitu vyvíjet stále znovu. Pro tyto opakující se úlohy je vhodné vytvořit sjednocený postup jak je řešit. Své „rámce“ si vytvářely společnosti zabývající se vývojem webových aplikací, právě aby zamezili redundantnímu psaní kódu. Jádro aplikace bylo společné a přidávaly se komponenty na míru zákazníkovi. Technologie pracující na odděleném modelu vrstev (prezentační, kontrolní, logická) tento přístup umožňují. Tato práce se zabývá rámci pro technologii Java Enterprise Edition (JEE někdy také J2EE).

Některé z frameworků měly za cíl pokrýt co největší počet funkcí (a vrstev aplikace), některé se spíše zaměřují na rychlost zvládnutí práce s frameworkem. Tato práce se snaží představit rámec Stripes, který se klace za cíl ulehčit vývojáři práci bez nároku na to, aby pokrýval všechny oblasti vývoje. Nabízí základní knihovny pro nejčastěji se vyskytující úlohy, ale nechává poměrně velký prostor pro integraci s jinými frameworky.

Geografické systémy

Se zvyšujícím se výkonem osobních počítačů dochází i k rozšíření počtu oborů, do nichž informační technologie vstoupily. Nejdříve byly počítače používány převážně pro matematické výpočty nad alfanumerickými daty. Postupem času začalo být možné tyto data uživateli také zobrazovat, pokud to aplikace vyžadovala. Otevřela se tak možnost uchovávat a spravovat data ve formátu, který je pro člověka lépe čitelný a představitelný – ve vizuální

podobě. Možností, jak data vizualizovat existuje několik a záleží na typu dat a také na typu samotné aplikace, zda je vhodnější použít graf, diagram či mapu.

Pro uživatele takovéhoto programu je několikanásobně snazší např. rozpoznat chybu v datech, pokud je zadává vizuálně než, pokud je musí upravovat v textové podobě. Samotná správa dat se tak stává rychlejší s přesnější.

Vznikaly aplikace pro stolní počítače, které umožňovaly práci nad daty, které se daly ať už dvojrozměrně, později i trojrozměrně vizualizovat. Zasahovaly do různých odvětví – od CAD[1] systémů po aplikace pro zdravotnictví (počítačová tomografie), robotika, statistické programy pro práci s grafy nebo programy pracující s geografickými daty (GIS).

Poslední z uvedených oblastí zahrnuje široké pole použití, protože informace o poloze lze přidat k velkému množství ať už obecných nebo specializovaných systémů a zvýšit tak informační hodnotu dat již v systémech používaných. Je možné např. rozšířit seznamy osob s uloženými adresami a vyhledávat dle zadaného rádiu bez ohledu na správní celky (okresy, kraje atp.).

Rozmach vývoje webových aplikací nad mapovými podklady je jednoznačně spjata se zpřístupněním aplikačních rozhraní online mapových služeb pro volné použití. V zahraničí to byla v této oblasti dominantní firma Google. Možnost přidávat vlastních vrstvy otevřela široká pole pro tvorbu specializovaných projektů nad velmi detailními kartografickými daty. Tyto datové podklady je totiž pro soukromé subjekty téměř nemožné v potřebném rozsahu nadefinovat. Použití těchto dat je možné přes předem nadefinované rozhraní a je silně omezeno na funkce, které dané rozhraní má.

Nemožnost volného přístupu k těmto datům pro vlastní aplikace, kde není omezeno jak bude s daty manipulováno, zapřičinilo vnik mezinárodního sdružení (Open Geospatial Consortium), které si vzalo za cíl shromáždil volně dostupnou sadu dat pro bezplatné použití. Projekt nese název Open Street Map a je výstup je v současnosti natolik podrobný, že je v některých oblastech detailnější než konkurenční sady dat.

1.2 Cíle práce

V teoretické části je zpracován přehled webových frameworků, který si neklade za cíl být úplný, ale představuje zástupce hlavních typů rámců pro vývoj webových aplikací v Java Enterprise Edition. Je provedeno srovnání těchto vybraných frameworků se Stripes a zhodnocena vhodnost použití pro cíl této práce.

Další neméně důležitou kapitolou jsou vybrané kapitoly z teorie geografických systémů, který se poskytuje základní informace potřebné pro implementaci GIS systému. Jsou zde také

1 Anglicky Computer-aided design (počítačem podporované projektování).

popsány mechanismy jak vytvořit z dat topologii použitelnou pro hledání cest a popsán algoritmus hledání nejkratší cesty.

V praktické části je cílem realizovat databázovou webovou aplikaci postavenou na frameworku Stripes pro vyhledání dopravních uzlů dle zadaných parametrů (vstupní pozice v souřadnicích GPS). Samotné hledání požadované cesty je řešeno tak, aby bylo využitelné chodci.

Aplikace vyhledá ze zadaného bodu nejbližší dopravní zastávku (autobusové, vlakové nebo zastávky městské hromadné dopravy). Do ní je následně vyhledána nejkratší cesta z pohledu chodce a ta je zobrazena na mapě ve webovém prohlížeči.

Hledání cesty je realizováno nad vlastní sadou dat získanou z projektu Open Street Map. Tyto data budou naimportovány do vhodné databáze, podporující práci s kartografickými daty. Analýza a návrh aplikace bude součástí teoretické části diplomové práce. Výběr použitých technologií je proveden s ohledem na to, aby byly pod otevřenou licencí.

Hledání cest je realizováno na straně databáze za pomoci několika zřetězených funkcí.

1.3 Současný stav řešení

Geografické informační systémy, zkráceně GIS, jsou aplikace zaměřené na pořizování, uchování, správu, analýzu a zobrazování dat týkajících se údajů o poloze na povrchu Země.

V několika posledních letech dochází k vzniku velkého množství specializovaných webových projektů založených na geografických datech. Jsou to ale často aplikace orientované na motoristy.

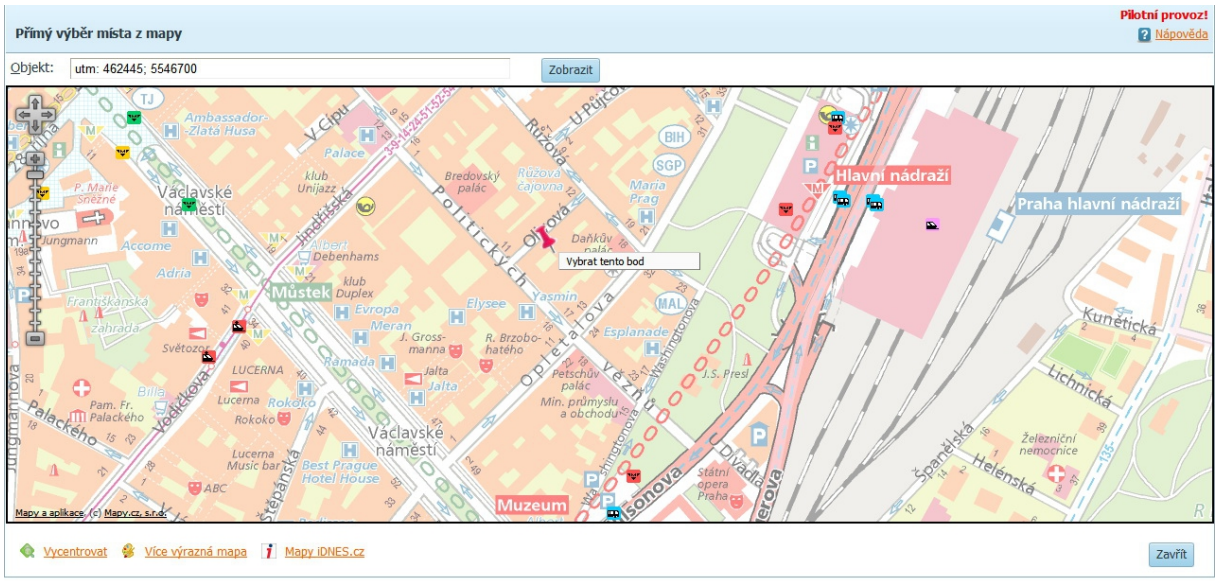
Existují také aplikace pro routování (zobrazení aktuální pozice a směru pohybu), které vyhledávají vhodnou cestu s ohledem na to, jestli je zvolen pohled chodce, cyklisty, vozíčkáře[2] nebo i motoristy. Takovým projektem je např. Open Route Service[3]. Takových projektů je menšina. Zajímavou aplikací je i GPSIES[4] zaměřující se na ukládání výletních tras a to i s ohledem na splavnost vodních toků.

Tomuto boomu také napomohla rostoucí popularita a tím pádem i rozšířenost mobilních zařízení, které jsou schopná zobrazit vizuální kartografická data. Jedná se především o PDA, tablety a mobilní telefony s dostatečně přesným displejem. Často tato zařízení obsahují GPS modul, který dále rozšiřuje možnosti použití – zejména v oblasti routování[5]. Je pak možné vyhledávání trasy v reálném čase.

2 Vyhledáváním cesty pro vozíčkáře se zabývá německá aplikace Rollstuhlrouting, k nalezení na adrese http://rollstuhlrouting.de/routenplaner.html
3 Služba je k dispozici na adrese http://www.openrouteservice.org/
4 Projekt je k vidění na adrese http://www.gpsies.com/
5 Zde ve významu hledání a zobrazení trasy do nějakého bodu.

Zde se stává jednoznačným trendem online uložení map, které skýtá výhodu přístupu k aktuálnímu stavu kartografických dat a dalších doplňujících informací (uzavírky, dopravní situace) a opouštění jednoúčelových zařízení jako jsou navigace s offline mapovými podklady. [1]

Cílem této práce je lokalizace dopravních zastávek ze zadané pozice. Podobnou službu nabízí i aplikace IDOS.cz. Na obrázku č. 1 je zachycen pilotní provoz vyhledávání dle zadané polohy v aplikaci IDOS.cz z března roku 2012. Uživatel byl po zadání bodu, ze kterého chce zahájit hledání dopravního spojení přesměrován na standardní formulář.



***Obrázek 1**: Vyhledávání dopravního spojení za pomoci zadání GPS souřadnic*

Předávaná pozice je ve formátu UTM, což je Univerzální transverzální Mercatorův systém souřadnic. Podrobněji je tento systém popsán v kapitole 4.3 Souřadnicový systém UTM. Ukázka přesměrování se zadanou polohou je vidět na obrázku č. 2. Lze použít i jiné zápisy GPS souřadnic. Viz dokumentace IDOS.

Název stanice, která byla označena jako nejbližší, uživatel zjistil až poté, bylo spuštěno vyhledání spojení, stejně tak je uživateli skryto, jak se k dané zastávce dostat nejkratší cestou. Tuto problematiku si klade za cíl řešit tato diplomová práce.

***Obrázek 2**: Ukázka předvyplněného počátečního bodu zadaného pomocí GPS souřadnic*

Vývoj aplikace v průběhu času pokročil hlavně v integrací vyhledávacího formuláře přímo na stránku s mapou (jak je vidět na obr. č. 3). Uživatelsky je zadávání bodů více příjemné, protože je možné měnit jeho polohu. Velkou výhodou je také možnost vidět aktuální odjezdy vlaků ze stanice. Nicméně informace o tom, jak se do zvolené stanice dostat, stále schází.

***Obrázek 3**: Funkcionaliza aplikace IDOS v roce 2013 v podporovaném prohlížeči.*

2 Architektura Java Enterprise Edition

V této kapitole bude teoreticky popsána technologie Java Enterprise Edition, na které je postaven zkoumaný framework Stripes.

Java Enterprise Edition (Java EE) je platforma pro vývoj distribuovaných podnikových aplikací podporujících transakční zpracování[6]. Nyní existuje ve verzi Java EE 7. *Distribuovanou transakcí* je rozuměno provedení jedné transakce nad několika informačními systémy. Například akce, která vyvolá zápis do několika databází (či jiných spolupracujících systémů).

Java EE vznikla zastřešením původní specifikace od firmy Sun Microsystems firmou Oracle. Původní název J2EE se používal do verze 1.4 a poté se přešlo na označení Java EE 5 až k současné verzi Java EE 7. Základem této platformy je Java Standard Edition (Java SE) se všemi knihovnami, včetně těch pro práci s grafikou (AWT, Swing). A k tomuto základu jsou vytvořeny další rozšiřující specifikace. Nadstavbami se podrobně zabývá kapitola 2.1 Architektura kontejneru Java EE.

Architektura Java Enterprise Edition vychází z *n-vrstvého modelu.* Ten obsahuje stejné části jako známý **třívrstvý model** a to:

- prezentační vrstvu (pro různé typy výstupů),
- vrstvu samotné logiky aplikace (tzv. bussiness logiky),
- vrstvu přístupu k datům (nejčastěji k databázovému serveru). [2]

Třívrstvý model je model, kdy je fyzicky odděleno pouze uložení dat od logiky aplikace. Na straně serveru existuje tedy pouze aplikační a databázový server. Klient slouží pro prezentaci.

Předchůdcem třívrstvého modelu byla **dvouvrstvá architektura** (také nazývaná jako klient – server), kde na straně serveru běžela pouze jedna služba, nejčastěji šlo o databázový server a na straně klienta byla realizována jak prezentace dat tak business logika. Na klienta jsou v tomto modelu kladeny větší nároky, co se týče programového vybavení, je na něm potřeba mít běhové prostředí pro takzvaného *tlustého* klienta.

N-vrstvý model oproti tomu odděluje prezentační vrstvu a přiřazuje jí samostatné službě (nejčastěji webovému serveru), aby tak byly optimálněji využity prostředky aplikačního serveru, který by tak byl neefektivně zaneprázdněn tvorbou vizuálního výstupu pro připojené klienty. Přístup aplikačních serverů a web serverů v n-vrstvé architektuře je realizován nativními protokoly. Komunikace je tedy rychlá a lze ji zabezpečit standardními způsoby.

6 Transakce – nedělitelná operace; musí proběhnout všechny kroky operace nebo žádný z nich.

Distribuované aplikace typicky využívají tzv. *middleware servery* pro využití systémových služeb jako jsou transakce, zabezpečení a dynamické přidělování databázových konektorů. K tomuto účelu je využíváno middleware API, které ovšem znesnadňuje přenositelnost na jiné platformy a škálovatelnost aplikací.

Architektura Java Enterprise Edition přejímá z toho konceptu oddělení jednotlivých služeb. To vede k efektivnějšímu využití systémových zdrojů a i těch lidských. Tato koncepce umožňuje zaměstnávat kvalifikované specialisty na databázové systémy a jiné starající se o vývoj samotné aplikace a GUI. [2]

2.1 Architektura kontejneru Java EE

Architektura Java EE je založena na konceptu soustavy několika kontejnerů poskytujících běhové prostředí pro objektově orientované aplikace psané v Javě. Někdy je tento typ aplikací nazýván Rich Internet Application (RIA). Kontejnery nabízí nízkoúrovňové služby jako zabezpečení, transakční přístup či management životního cyklu aplikace.

Aplikační server obsahuje dva kontejnery [2]:

- *webový kontejner* – hostí komponenty prezentační vrstvy jako Java Server Pages (JSP), Java Server Faces (JSF) a servlety (Java servlets). Tyto komponenty také komunikují s EJB kontejnerem pomocí vzdálených protokolů.
- *kontejner pro bussiness logiku - Entity Java Beans kontejner* – Reálně spouští a ukončuje jednotlivé EJB komponenty. Spravuje stav transakce (obnovení, sdružování, ukončení) a poskytuje pro ni běhové prostředí.

Na straně klienta komunikuje webový prohlížeč s webovým kontejnerem skrze HTTP protokol.

2.1.1 Komponenty Java Bean

Základním kamenem architektury Java EE jsou takzvané komponenty *Java Bean*, zkráceně nazývané *beany*. Což nejsou nic jiného než javové třídy, které rozšiřují nějakou bázovou nadtřídu nebo implementují speciální rozhraní. Na beany je možné se odkazovat z jiných komponent systému.

Bean komponenty mají několik společných vlastností, které je charakterizují. Jsou perzistentní, lze trvale uložit a obnovit jejich stav. Kromě standardních *vlastností* a *metod* – které mají i obyčejné POJO objekty – obsahují ještě *události*. Ty slouží pro oznámení jiným komponentám, že v bean komponentě nastala nějaká změna.

Pokud nějakou komponentu zajímá změna události, tak se musí zaregistrovat u zdroje této události a zahájit naslouchání. Pak je při každé změně na všech posluchačích zavolána předem určená metoda.

Na technologii Java Bean je postavena například i knihovna Swing, která se používá pro tvorbu vizuálního rozhraní 'tlustých' aplikací. Pojmenování jednotlivých bean se pak používá pro vizuální vytváření formulářů v různých vývojových prostředích.

Služby, které platforma JEE podporuje jsou standardní HTTP, HTTPS, JDBC, JavaMail. [3] Další služby budou dále popsány podrobněji.

2.1.2 JTA – Java Transaction API

EJB kontejner na rozdíl od toho webového podporuje transakční zpracování[7]. To je realizováno skrze JTA – Java Transaction API. Samotné transakce jsou realizovány na straně serveru, přičemž každá transakce je přidělena jednomu vláknu. Transakční manažer se následně *„poskytuje služby a řídící funkce potřebné k vymezení transakce, správě transakčních zdrojů, k synchronizaci, a k prezentaci obsahu transakce“*. [4]

Rozhraní JTA je realizována skrze *Java Transaction Manager*. Což je kontejner implementující rozhraní JTA, konkrétně:

- javax.transaction.TransactionManager,
- javax.transaction.Transaction,
- javax.transaction.UserTransaction.

V rámci Java Transaction Manageru také možné synchronizovat transakce. Slouží k tomu tzv. *Synchronization callback* objekty, které se registrují u aplikačního serveru vždy před a po ukončení transakce. [4]

2.1.3 RMI – *Remote Method Invocation*

Základním kamenem Java Enterprise Edition je RMI – volání vzdálených metod. Vzdálenou metodou, resp. objektem se rozumí objekt nacházející se v jiném adresním prostoru (angl. *namespace* nebo jiná instance JVM), než odkud je volán.

Toto volání je realizováno podobně jako volání metod mezi klientem a serverem. Jsou při tom použity zástupné objekty (stub – metoda na straně klienta), zástupné reference (remote reference) a skeleton (metoda na straně serveru). Mezi vrstvou, která obstarává komunikaci mezi

7 Transakční zpracování je atomické, konzistentní, izolované a trvalé. (ACID – atomicity, consistency, isolation, durability)

„*Klient používá pro odkazy na vzdálené objekty proměnné typu rozhraní, které vzdálený objekt implementuje.*“ [5]

Samotné zacházení se vzdáleným objektem je následně stejné jako s lokálním objektem.

2.1.4 JPA – *Java Persistence API*

Platforma Java EE nabízí další dostupná rozhraní. Je to JPA – *Java Persistence API* pro přístup k perzistentní vrstvě aplikace. Jedná se o více obecný přístup k připojení k zdrojům, kde jsou uložena data. JPA má na starosti otázku bezpečnosti, správu transakcí a realizaci samotného připojení. Nejčastěji se k připojení k perzistentně uloženým datům využívá *JDBC připojení*[8], které má tu výhodu, že sjednocuje přístup k různým databázovým platformám. Samotný překlad nativních volání dané databáze obstarává JDBC, není tedy potřeba znát konkrétní volání, ale využívají se standardizované metody.

2.1.5 JNDI – Java Naming and Directory Interface

Rozhraní JNDI je součástí balíku J2EE od verze 1.3. Poskytované služby, které rozhraní podporuje, jsou

- LDAP – protokol pro přístup k datům na adresářovém serveru pro jednotlivé uživatele. Poskytuje autentizaci klienta, který přistupuje k serveru a pro komunikaci je možné použít jak synchronní tak asynchronní mód. Tento protokol sjednocuje přístup k datům a umožňuje tak různým aplikacím pracovat se soubory standardním způsobem.
- CORBA – platformně a protokolově nezávislá architektura pro transparentní použití distribuovaných objektů. „*Klient přistupuje k těmto službám zasíláním požadavků. Forma generování požadavků závisí na jazyce, ve kterém je klient napsán. V objektově orientovaném jazyce to může být volání metod zástupného objektu, zatímco v klasických programovacích jazycích volání vygenerovaných funkcí (stubs*[9]*).*“ [6]
- RMI – vzdálené volání metod. Umožňuje jedné instanci javového virtuálního stroje volat metody jiné instance. Na této technologii stojí celá architektura Enterprise JavaBeans.

Volitelné části pak jsou

- NIS – *Network Information System* – platforma klient-server, vyvinutá firmou Sun Microsystems, je vhodná pro administraci menší lokální sítě[10]. Je alternativou k DNS pro izolované LAN.

8 V případě připojení k relačním databázím.
9 Zástupná metoda, generovaná frameworkem.
10 Zdroj: http://searchnetworking.techtarget.com/definition/NIS

- NDS – *Novell Directory Services* – programový balík pro síťovou administraci od firmy Novell. Umožňuje správu uživatelů, účtů pro vzdálený přístup apod. Základní charakteristikou je uložení objektů do stromové struktury. Tyto objekty pak popisují strukturu sítě. [7]

Jako analogii k objasnění toho, co JNDI dělá, lze použít příměr, že jde o API – neboli systém knihoven různých jmenných a adresářových služeb (viz výše). Sjednocuje tak pro vyvíjenou aplikaci přístup k různým implementacím. Zjednodušeně by se dalo říci, že jde o obdobu JDBC pro přístup k souborům.

2.1.6 JMS – Java Message Service

Poslední komponentou Služba, která umožňuje zasílání zpráv mezi klienty či business komponentami. Velkou výhodou toho řešení je jednoduchá konfigurace všech služeb pomocí XML.

2.1.7 Shrnutí Java EE

Architektura Java EE plně podporuje vývoj aplikací založených podle vzoru MVC. „*Vzor MVC má tři ústřední součásti:*

- *Model zodpovídá za správu vnitřní struktury*
- *Pohled zodpovídá za vnější prezentaci vnitřní struktury*
- *Řadič zodpovídá za kontrolu průběhu a stavu uživatelského vstupu.*

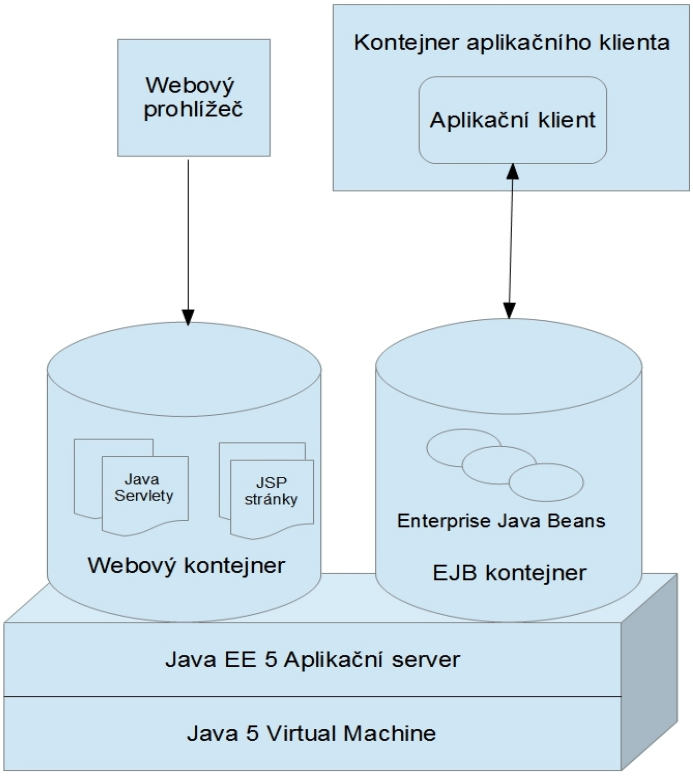
U vzoru MVC se obvykle objeví nějaký druh oznámení události, které ohlásí pohledu, že došlo ke změně v nějaké části modelu.“ [8].

Model v architektuře Java EE zastupují beany, řadič servletem obsluhované ActionBeany a pohled realizují JSP stránky. Svým uspořádáním Java EE nutí vývojáře důsledně oddělovat jednotlivé funkce do příslušných vrstev.

Pro běh servletů je potřeba webový server. Java EE jich podporuje několik. Nejběžněji používanými opensource aplikačními servery jsou GlassFish, Apache Tomcat či JBoss. Za komerční produkty jsou to BEA WebLogic, IBM WebSphere či Oracle AS. Komerční webové servery navíc podporují virtualizaci běhového prostředí.

Pro účely této práce byl zvolen Apache Tomcat pro svoji relativně malou paměťovou náročnost, je tedy vhodný pro vývoj na osobním počítači, kde nespotřebuje příliš velké množství operační paměti.

Na obrázku č. 4 je vidět uspořádání jednotlivých kontejnerů Javy EE.



Obrázek 4: *Architektura Java EE platformy*

Nutnou podmínkou pro korektní fungování aplikačního serveru je vyřešení konkurenčního přístupu ke sdíleným zdrojům. Toto kritérium Java EE architektura splňuje. Využívá k tomu technik zamykání měněných objektů (mutex[11]) a přístup k těmto objektům přes řízený *pool* objektů.

K optimalizaci přístupů k připojení databázových připojení je standardně využito poolingu pro efektivní přidělování připojení klientům žádajícím přístup. Pooling také zajišťuje ochranu před přetížením serveru z důvodu příliš velkého počtu připojení, které již není server schopen obsloužit.

2.2 Enterprise Java Beans

Enterprise Java Bean (dále EJB) je komponentní technologie na straně serveru, která umožňuje vyvíjet distribuované, transakční, zabezpečené a platformně nezávislé aplikace.

K pochopení technologie EJB je třeba probrat distribuované systémy obecně. Jedná se tedy o technologie, které slouží k obsluze většího množství paralelních procesů sdílejících stejná

11 Mutual exclusion – technika vzájemného vyloučení threadů pomocí konkurenčního přístupu ke kritické sekci.

data. Distribuovaný systém musí zajistit jejich synchronizaci v reálném čase a nezávislost na místě uložení dat.

EJB komponenty rozlišujeme dvou typů:

- **Session bean** – vhodné pro řízení běhu. Mohou být stavové a bezstavové.
    - Bezstavové session beany se používají pro jednoduchá volání skládající se z jednoho požadavku. Není tedy třeba uchovávat nějaký konverzační stav.
    - Jsou užitečné pro popis komunikace mezi ostatními beany, vykonává práci pro klienta, pokud o to požádá. Zastávají roli konzumentů entity beanů a jejich životnost bývá zpravidla velmi krátká. Jsou to objekty implementující bussiness logiku a nejsou perzistentní.
- **Entity bean** – používají se, pokud je potřeba persistentní uchování komponent a sdílené mezi uživateli. Mohou být řízeny dvěma způsoby:
    - bean-managed persistence (BPM) a
    - container-managed persistence (CPM). BPM je používána pokud je mechanismus perzistence v beaně a CPM pokud není. [3] Jsou identifikovány dle primárního klíče (ukazatele v paměti do databáze) a reprezentovány jako POJO objekty. Často jsou namapovány na konkrétní tabulku v relační databázi. [9]

Dále v textu bude tedy pod pojmem aplikace rozuměna aplikace podniková transakční a distribuovaná.

2.3 Servlety

Skrytou ale velmi důležitou částí architektury Java EE jsou servlety. Jsou to speciální objekty jazyka Java běžící na webovém serveru. Mají předdefinované rozhraní, pomocí něhož komunikují s okolím. Obsluhují HTTP požadavky, vytvářejí a spravují jednotlivá sezení. Následuje popis standardního životního cyklu servletu:

„*Poprvé, když se vytvoří servlet, vyvolá se jeho metoda* `init`. *Po tomto bude mít každý uživatelský požadavek za následek nové vlákno, které vyvolá metodu* `service` *již vytvořené instance. Několik souběžných požadavků za normálních podmínek mají za následek současné spuštění několika vláken, které zavolají* `service`. *I když váš servlet může implementovat speciální rozhraní, které specifikuje, že je v témže čase povoleno spustit pouze jeden proces. Metoda* `service` *pak vyvolá* `doGet`, `doPost` *nebo nějakou metodu* `doXXX`, *v závislosti na typu požadavku HTTP. Který přijala. Nakonec pak, když server rozhodne uvolnit servlet, zavolá metodu* `destroy`“ [10].

Servlety jsou prostředníky mezi klientem a modelem aplikace. Pracují automaticky a po jejich prvotní konfigurace nebývá nutné jejich funkci nějak ovlivňovat.

2.4 JSP – Java Server Pages

Zobrazování údajů do HTML stránek pomocí servletů je sice možné, ale je velmi krkolomné a zdlouhavé. Srovnej kód vypsání textu 'Ahoj světe' a zobrazení aktuálního data servletem a poté v JSP.

```
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

   public void doGet(HttpServletRequest req, HttpServletResponse
res)
               throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Hello World</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<H1>Hello World</H1>");
    out.println("Today is: " + (new java.util.Date().toString()) );
    out.println("</BODY></HTML>");
   }
}
```

Ta samá funkcionalita nyní v JSP.

```
<HTML>
 <HEAD>
  <TITLE>Hello World</TITLE>
 </HEAD>
```

```
  <BODY>
   <H1>Hello World</H1>
   Today is: <%= new java.util.Date().toString() %>
  </BODY>
</HTML>
```

Jak je vidět výše, servlety jsou pro práci s prezentační vrstvou velmi neobratné, kvůli nutnosti vytisknout každou značku spolu s hodnotou do speciálního výstupu, který je pak odeslán jako odpověď prohlížeči.

Záhy vznikla nadstavba nad servlety zjednodušující tvorbu vizuálního zobrazení dat – Java Server Pages. Ta byla uvolněna v březnu roku 2002. JSP umožňují kombinovat statické prvky webu (HTML) a dynamicky vkládaný obsah (JSP funkce, načítání dat z databáze, …).

Dynamické chování JSP stránek zajišťuje tzv. Expression Language (EL). Pomocí něj lze do JSP importovat jiné knihovny a kompletně zpracovávat data. Má formu značek `<% %>`. Soubor JSP jako takový je rozšířením HTML, obsahuje navíc vlastní sadu značek – Expression Language (EL) – a obstarávají dynamickou práci s daty. Je to mechanismus, který zajišťuje komunikaci mezi prezentační a kontrolní vrstvou.

JSP jako takové servlety vnitřně používá, ale odklání programátora od jejich ručního vytváření. Pro práci na webovém rozhraní tedy není nutná znalost jazyka Java. Možností vkládat funkční kód přímo do HTML stránky dochází k jasnějšímu oddělení vrstev aplikace podle vzoru MVC [10].

Jak název napovídá, stránky JSP jsou spouštěny na straně serveru (v rámci architektury klient – server) a klientu je zasílána vygenerována čistá HTML stránka. Není proto potřeba aby byl klient vybaven jakýmkoliv prostředím pro spouštění skriptů. Postačí běžný webový prohlížeč.

Standardní sada značek EL je dále rozšiřitelná pomocí tzv. uživatelských *tagů*. Tato vlastnost umožnila vzniknout nepřebernému množství nadstaveb (frameworků), nad JSP, jejichž částí se zabývá i tato práce.

Pro úplnost je třeba zmínit existenci standardní knihovny značek JSTL (Java Standard Tag Library), která se stala součástí technologie JEE. Poskytuje základní funkcionalitu pro nejčastěji používané operace v prezentační vrstvě aplikace. Nicméně je její používaní většinou frameworků nahrazováno vlastními značkami, které poskytují další funkce navíc.

2.5 Předdefinované proměnné

Technologie JSP používá několik standardních předvytvořených objektů, které servlety používají pro komunikaci s protokolem HTTP. Jsou to následující objekty:

- `request` – objekt třídy HttpServletRequest, zpřístupňuje parametry požadavku, jako je např. typ (POST, GET), záhlaví (cookies).
- `response` – objekt třídy `HttpServletResponse`
- `session` – objekt třídy `HttpSession`, bývá sdružen s objektem `request`. Má různé druhy platností (`page`, `session`).
- `out` – objekt třídy `JspWriter`, používá se pro odpověď uživateli.
- `application` – objekt třídy `ServletContext`. Slouží pro ukládání perzistencích dat jak servlety tak JSP.

V JSP stránce je k těmto objektům možné přistupovat. Odkaz na ně je velmi snadný [10].

```
<%= request.getRemoteHost() %>
```

Značky EL mohou tedy přistupovat jak k vlastnostem bean, tak k objektům jednotlivých sezení.

2.6 AJAX

Další technologií, která sice není součástí Javy EE, ale je velmi často integrována do aplikací v ní napsaných, je AJAX. Jak název technologie napovídá – Asynchronous JavaScript and XML – jedná se o využívání JavaScriptu, který používá XML. A to v podobě modifikace požadavku, který je odesílaný od klienta na server. Požadavek je ve formátu XML a je zapouzdřen novým objektem `XMLHttpRequest`.

Metody protokolu HTTP, které tento objekt dokáže zasílat jsou následující: GET, POST, PUT, DELETE, HEAD a OPTIONS. Výchozí typ volání je GET.

AJAX není nová technologie jako taková, ale spíše nový způsob jak stávající technologie využívat. [11]

Hlavní výhoda tohoto způsobu komunikace klienta se serverem je ta, že není potřeba obnovovat stránku ve webovém prohlížeči, aby byly uživateli zobrazeny výsledky nějaké akce. Jistou nevýhodou je nemožnost použití tlačítka Zpět v prohlížeči, protože se jedná o navigaci mezi statickými stránkami, o což v tomto případě nejde. Uživatelé jsou na jeho funkcionalitu zvyklí a je třeba na tuto vlastnost AJAXu pamatovat. Většinou postačí přidat

možnost zpětného průchodu webovou aplikací nějakým alternativním odkazem přímo na stránce nebo přidat ke každé akci i vygenerování nové URL adresy, ke které je pak možné se vrátit.

Další věcí, kterou by měl mít programátor při používání AJAXu na paměti je to, že to není technologie vhodná pro čilou komunikaci se serverem. Jedná se pouze o nadstavbu nad velmi omezeným protokolem HTTP, který má tu vlastnost, že pro každý požadavek je potřeba navazovat nové spojení. To je věc, která komunikaci velmi zpomaluje.

Komunikaci vždy zahajuje klient, takže není možné, aby server cokoliv oznámil klientům z vlastní vůle. Bylo by to výhodné např. v případě, když nastane na serveru nějaká událost. Toto lze obejít opakovaným dotazováním klienta, zda již událost nastala, ale pro některé typy aplikací není tento způsob komunikace vhodný a je třeba se poohlédnout po jiných komunikačních protokolech než je HTTP.

V této práci bude AJAX využit pro komunikaci mezi javascriptovým API a webovým serverem.

2.7 GeoJSON

GeoJSON je dialekt formátu pro uložení alfanumerických dat v notaci JSON (JavaScript Objct Notation). Podobně jako dokumenty XML popisuje strukturu a typ dat – jejich sémantiku čitelnou i pro počítačové programy. Na rozdíl od zmiňovaného XML není velikost dat nutná pro popis struktury dokumentu tolik rozsáhlá, protože nepoužívá značky, ale složené závorky.

Níže je ukázka souboru typu GeoJSON, který obsahuje sadu koordinátů *x* a *y* určujících body na čáře.

```
{"type":"LineString",
"coordinates":
    [
          [13.8421808,50.3680868],[13.8420351,50.3683329],
          [13.8421471,50.3689267],[13.8436797,50.369074],
          [13.8451601,50.3693337],[13.8452549,50.3693865]
    ]
}
```

V ukázkové aplikaci je formát dat GeoJSON použit pro předávání dat z databáze. Jeho přenos probíhá za pomocí AJAXu. Nadstavba PostGIS obsahuje standardizované funkce pro vyex-

portování jakéhokoliv datového typu `geometry` (`geography`) do toho formátu. Je to velmi užitečná funkce. V PostGIS verze má tato funkce následující zápis: `ST_AsGeoJSON(var geometry)`.

Tímto je přehled použitých technologií úplný.

3 Java frameworky

V první řadě je vhodné definovat, co je myšleno pod pojmem *framework*. Jedná se o množinu znovupoužitelného software, který slouží jako základ pro nějakou aplikaci. Pomáhá vývojářům programovat rychleji tím, že často opakované úlohy a běžně používaná rozhraní, jsou již předdefinovány a připraveny pro použití.

Nespornou výhodou je, že frameworky zapouzdřují osvědčené *návrhové vzory.* Návrhový vzor[12] je obecný způsob jak řešit nějaký problém. Popisuje interakci mezi objekty systému. Frameworky nutí programátory tyto vzory využívat tím, že mají připravená rozhraní způsobem, jakým jednotlivé návrhové vzory vyžadují.

Další motivací pro vytvoření jakéhokoli frameworku je oddělení jednotlivých vrstev aplikace, které je pak možné odděleně optimalizovat. Případně lze tyto vrstvy i odděleně vyvíjet, pokud to rozsah projektu vyžaduje.

Rámce pro webové aplikace lze rozdělit dle několika kritérií. Nejzákladnější dělení je dle oblasti vývoje, kterou mají rámce pokrývat. Jsou to dále uvedené.

3.1 Integrační frameworky

Pro vývoj rozsáhlejší webové aplikace je potřeba udržovat ve správných verzích několik knihoven a dalších komponent – např. komponenty pro testování, sestavování (*deploying*), objektově relační mapování apod... Aby spolu jednotlivé komponenty spolupracovaly, tak je nutné, aby byly korektně nakonfigurovány.

Tuto nezbytnou část vývoje se programátorům snaží usnadnit takzvané integrační frameworky. Ty poskytují jednotné rozhraní pro připojení a konfiguraci různých knihoven a komponent. Nejčastěji bývá toto jednotné rozhraní realizováno pomocí javových anotací.

3.1.1 JBoss Seam

Nejznámějším zástupcem integračních frameworků je Seam. Jedná se o integrační rámec pro webové aplikace vyvíjený pod hlavičkou firmy JBoss divizí RedHat. Je integrován do vývojového prostředí Eclipse.

Oproti ostatním frameworkům nabízí Seam několik již hotových projektů, které mají sloužit jako základ pro vlastní přizpůsobení. Vývojář si nejprve vybere nejpodobnější projekt a pokračuje v jeho modifikaci. Framework Stripes nic podobného nenabízí a ponechává návrh aplikace plně v rukou programátora.

12 Anglicky design pattern.

3.1.2 Spring MVC

Vznik tohoto rámce je spojen s knihou Roda Johnsona Expert One-on-One J2EE Design and Development[13], ke které byl vytvořen ukázkový kód demonstrující jak pracovat s architekturou J2EE verze 1.3. Cílem bylo minimalizovat nutnost opakovaně psát kontejnery pro jednotlivé enterprise beany. Postupně se okolo projektu sdružila komunita vývojářů a následně vznikl framework s otevřeným kódem jako alternativa k J2EE.

V dalších verzích J2EE došlo k revizi konceptu webového kontejneru a některé principy použité ve Spring byly do něj následně zapracovány.

Hlavní principy tohoto frameworku:

- aplikace je *modulární* – výhodou je tedy snadná rozšiřitelnost. Tato funkčnost je realizována přes nadefinované rozhraní (`java.Lang.Interface`), přes které je přistupováno k jednotlivým bean komponentám. Nejčastěji jsou oddělené pro manipulaci s daty a pro jejich zobrazení.
- *Inversion of Control (IoC) Container* – implementace návrhového vzoru IoC. Mezi třídami jsou realizovány vazby za pomoci předávání instance závislé třídy parametrem.

Framework Spring MVC není klasický rámec ve smyslu pouhého asistenta pro pohodlnou tvorbu samotné prezentační vrstvy aplikace. Jedná se o integrační rámec, který je možné použít i pro 'desktopové' aplikace [12]. Spring pokrývá svým rozsahem všechny vrstvy aplikace.

Doba potřebná na zvládnuti tohoto frameworku je přímo úměrná jeho komplexnosti. Je tedy zřejmé, že se nehodí pro malé projekty, kde by nebyl čas nutný pro vývoj, vyvážen nějakým přínosem tohoto frameworku.

Architektura aplikací ve Spring MVC frameworku vychází z návrhového vzoru model – view – controler. Počínaje uživatelovým prohlížečem, zpracováním aplikačním a databázovým serverem, požadavek traverzuje napříč různými vrstvami systému.

Prezentační vrstva externího klienta obsahuje vše, co spolupracuje s aplikací po síti. Je to webový prohlížeč, čtečka PDF souborů či agregátor RSS zdrojů.

3.2 Aplikační frameworky

Rámce, které se zaměřují na převážně na usnadnění samotného vývoje, nikoliv na možnost rozsáhlé konfigurace a variabilnosti při vývoji projektu se nazývají aplikační frameworky.

13 ISBN: 1861007841

3.2.1 Platforma NetBeans

Zástupcem této skupiny frameworků je rámec, který jako svoji hlavní výhodu představuje pokročilou integraci do vývojového prostředí NetBeans. Je to platforma, která nabízí pro programátora sadu průvodců pro nejčastěji vyvíjené komponenty (beany). Samozřejmostí je možnost vyvíjet a ukládat si svoje komponenty.

Velkou výhodou této platformy je velké množství průvodců, které usnadňují prvotní konfiguraci jednotlivých komponent. Programátor se poté soustředí pouze na návrh business logiky.

Obsahuje předdefinované komponenty, ze kterých je možné aplikaci skládat. Pro prezentační vrstvu využívá technologii JSF a pro perzistenci JPA. JSF jako takové je nadstavbou nad technologií JSP, kdy obchází nutnost importovat jednotlivé knihovny značek. [13]

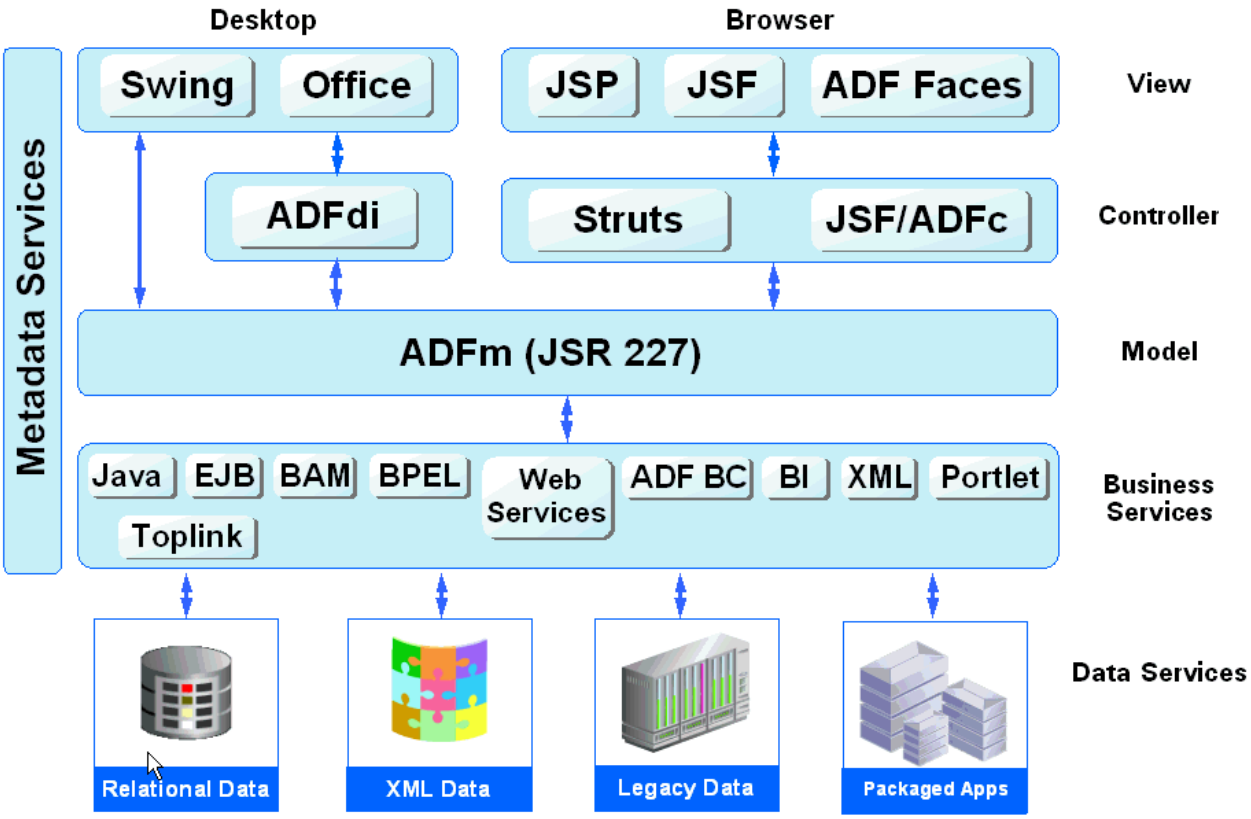
Srovnání se Stripes je na v tomto případě velmi obtížné. Oba frameworky přistupují k vývoji diametrálně odlišným způsobem.

3.2.2 Oracle Application Development Framework

Také jinak nazývaný Oracle ADF Fusion je framework postavený na open-source technologiích. Není určen pouze pro webové aplikace, ale poskytuje možnost stejná data prezentovat i v desktopovém programu. K tomu využívá knihovny Swing.

Vývoj fusion (smíšené) aplikace je možný pouze ve vývojovém prostředí JDeveloper podobně jako je NetBeans platforma omezena pouze na toto IDE. Toto omezení ale vyvažuje opravdu velká míra integrace jednotlivých kroků do IDE, které pak funguje jako jakýsi průvodce a konfigurátor vznikající aplikace. Samozřejmostí je vizuální návrh formulářů a to jak pro webovou tak i desktopovou variantu.

Na obrázku č. 5 je popsána architektura tohoto řešení. Je vidět oddělení jednotlivých vrstev a služby se kterými framework dokáže spolupracovat.



Obrázek 5: *Architektura Oracle ADF Fusion*[14]

3.3 Webové frameworky

Rámce, které svoji činnost zaměřují pouze na podporu vývoje webové části aplikace spojuje ponechávání volné ruky při práci na ostatních částech projektu. Není výjimkou kombinace různých integračních (i aplikačních) a webových frameworků.

3.3.1 Framework Stripes

> Motto: „Používali jste někdy nějaký framework a cítili, že toho musíte udělat tolik pro framework ve srovnáním s tím co vám framework dá nazpět?“
>
> *Freddy Daoud, Stripes Book*

Rámců zaměřených na usnadnění tvorby hlavně webové části aplikace existuje nepřeberné množství. Liší se rozsahem, který se snaží při vývoji webové aplikace pokrýt a s tím úzce souvisí i krátká doba nutná pro jeho zvládnutí.

14 Převzato z http://docs.oracle.com/cd/E23549_01/web.1111/b31974/intro.htm

Framework Stripes je projekt s otevřeným kódem, koncepčně se podobá Struts a WebWorku. Byl vytvořen americkým programátorem Timem Fenellem, který se primárně snažil kompenzovat nedostatek výše zmíněných rámců a tím je poměrně složitá konfigurace pomocí několika XML souborů (povinně `web.xml`, `struts.xml`, volitelně `struts.properties` atd.).

Oblast, která je plně v rukou programátora je propojení s databázovou vrstvou. Existuje množství velmi kvalitních frameworků pro tuto oblast vývoje a vývojáři Stripes nevidí jako potřebné duplikovat tuto funkcionalitu. Je možné použít jak přímé spojení javové třídy a JDBC nebo robustní framework dynamicky přidělující databázové konektory (Hibernate, JPA).

Stripes je postaven na návrhovém vzoru Model – View – Controler (MVC) a jeho hlavní zaměření je vrstvu prezentační a kontrolní. Podoba toho, jak bude vypadat model je plně v rukou vývojáře.

Kontrolér je založený na principu takzvaných `ActionBean`. Je to koncept, převzatý z frameworku Struts, kde existují speciální javové třídy (beany), které nemají pouze funkci uchovávat nějaké vlastnosti tříd, ale fungují jako zprostředkovatelé komunikace mezi jinými částmi systému. Samotné ActionBeany vycházejí z konceptu komunikace v rámci HTTP protokolu. Uživatel poklikem na tlačítko v JSP stránce zašle požadavek (request) webovému serveru. Objekt `ActionBean` (na serveru) tento požadavek zaznamená a spustí příslušnou metodu. Metoda po provedení akce vrací objekt typu `Resolution`, který framework Stripes překládá na odpověď (`response`) v protokolu HTTP. Ten JSP stránce předává ActionBean, která je následně překreslena s novými výsledky. Cyklus požadavek – odpověď je tedy uzavřen. Programátor nepotřebuje explicitně pracovat s objekty `HttpRequest` a `HttpResponse`, framework překládá komunikaci metod s těmito objekty sám.

Hlavni oblasti, které framework Stripes pokrývá jsou tyto:

- *Automatické mapování*: Framework automaticky mapuje URL adresy, parametry z HTTP protokolu do javových tříd.
- *Automatické načtení nových komponent:* Stripes automaticky zjišťuje a načítá nové třídy. Samozřejmostí je možnost přejmenovávat a odstraňovat třídy bez nutnosti měnit konfiguraci v XML souboru.
- *Validace:* Je k dispozici výkonný mechanismus pro validaci vstupních formulářových polí založený na anotacích.

- *Konverze a formátování: A*utomatická konverze mezi datovými typy Stripes frameworku a běžnými javovými typy. Lze snadno přidat metody pro vlastní uživatelské konverze.
- *Jednotné generování layoutu*: Za pomocí tří tagů je možné vytvořit jednotnou šablonu (layout) pro vzhled JSP stránek, která se bude používat v celém projektu. Není zde potřeba žádná dodatečná konfigurace.
- *Lokalizace*: Všechny tagy frameworku Stripes mají připravené jednotné rozhraní pro vyhledávání jazykových mutací. Jedná se o jednoduchou možnost, jak do stávající aplikace přidat jiný jazyk bez nutnosti zasahovat do již hotového kódu. Klíčové popisy v jednotlivých jazycích jsou načítány z externích souborů.
- *Zachytávání výjimek*: Framework Stripes odchytává výjimky, které nezachytil a neošetřil kód napsaný vývojářem a zobrazuje uživateli webových stránek předdefinované chybové stránky. Lze přidávat chybové stránky pro specifické typy chyb.
- *Hezké URL adresy*: Není nutné se jakkoliv starat o podobu URL adres, protože je framework spravuje sám. Existuje aale echanismus, který URL adresy generuje polde předem zvolených šablon.
- *Integrace AJAX*: Díky transparentní práci s objekty typu `HttpRequest` a `HttpResponse` je snadné integrovat nějaký dostupný AJAX framework.
- *Testování*: Jsou k dispozici připravené třídy pro jednotkové testování webové aplikace. Chybový hlášení jsou ve Stripes velmi detailní a někdy obsahují i tip na to co by mohlo chybu způsobit.
- *Snadná rozšiřitelnost a přizpůsobitelnost*: Stripes je navržen jako modulární systém, takže je možné vkládat vlastní funkcionalitu do jakékoliv části frameworku či jakoukoliv část modifikovat.

Dalším cílem bylo vytvoření knihoven pro často řešené úlohy, jako je např. přihlašování, validace vstupních formulářů, lokalizace, upload souborů, jednotkové testování, apod. Dochází tak ke standardizaci návrhu aplikace a lze jednoduše zvýšit znovupoužitelnost jednotlivých komponent. Důsledkem toho je vývoj dalších projektů výrazně rychlejší. Dále existuje řada uživatelských rozšíření, které je možné k přidat, nejčastěji jde o propojení s jinými balíky či integraci do různých IDE.

Využívá pro svojí funkci vlastnosti jazyka Java verze 1.5 – a to anotace a generícitu. I když je možné s jistými obtížemi zajistit provoz i na Javě verze 1.4. Vyjma prvotního nastavení aplikace je konfigurace frameworku je řešena pomocí javových anotací. Je tak obejit mechanizmus běžný v jiných rámcích, kdy je nutné udržovat v konzistentním stavu často několik

XML souborů. Jsou tak omezeny na minimum situace, kdy je při rozšiřování projektu nutné upravovat konfiguraci na několika místech.

Stripes je také možné použít ve spojení s jinými frameworky. Pro propojení do databáze je doporučováno použít nějaký specializovaný ORM framework jako např. Hibernate[15] či JPA[16]. Častá je také kombinace aplikačního rámce Spring a využití Stripes pouze pro tvorbu webového rozhraní. Nebo použití Apache Ant[17] pro výslednou kompilaci a sestavení aplikace.

Řídící servlet ve Stripes je nastavován pouze konfiguračním souborem `/WEB-INF/web.xml`. Nastavení je třeba provést pouze při založení projektu a obvykle není nutné do něj dále zasahovat, pokud se výrazněji nemění adresářová struktura. Doplňování nových `ActionBean` probíhá automaticky.

Z praktického hlediska je framework knihovna tříd, která se přiloží k danému projektu a jsou volány její funkce. V prostředí Java je to JAR soubor, který do projektu vkládá jako knihovna do adresáře `WEB-INF/lib`. Jsou to tyto dvě:

- `stripes.jar`: knihovny frameworku
- `commons-logging.jar`: vyžadovaná knihovna, kterou Stripes používá pro logování

Tato práce se zabývá frameworkem ve verzi 1.5.7, která byla vydána 17. května 2012. Spolupracuje s webovými servery Jetty 5.0, Resin 3.0 a Tomcat 5.5 (čísla popisují minimální verzi).

Mechanismus mapování ActionBean

Framework Stripes usnadňuje samotné mapování bean (a jejích vlastností) na události JSP stánky automatickým mechanismem. Ten přiřazuje odkaz volání bean komponenty. Na základě názvu třídy, která implementuje rozhraní `ActionBeanContext`.

Jeden `ActionBean` objekt může obsluhovat více než jeden formulář.

Konfigurace pomocí anotací

Jak bylo zmíněno výše, tak frameworku Stripes je sdělováno, jak se má chovat k určitým metodám, pomocí speciáln konstrukce jazyka Java, zvaná anotace. Anotace se v jazyce Java objevují už delší dobu (viz anotace @override[18], `@deprecated`[19], `@transient`[20]), ale od verze Javy 1.5 je možné přidávat i anotace uživatelsky nadefinované.

15 Hibernate je framework s otevřeným kódem pro objektově relační mapování.
16 Java Persistence API pro objektově relační mapování, rozhraní je dostupné v balíku javax.persistence.
17 Apache Ant je nástroj pro sestavování projektů napsaný v jazyce Java, umí řídit kompilaci a testy.
18 Anotace `@override` říká, že metoda je přetěžuje metodu předka.
19 Anotae `@deprecated` oznamuje překladači, že metoda je nedoporučovaná.
20 Anotace `@transient` určuje, že objekt nebude uložen při serializaci.

Anotace je speciální druh modifikátoru (podobně jako modifikátory `static`, `public`, `private`, `final`), který říká překladači, jak má naložit s části kódu, ke které se anotace vztahuje.

Anotace pro odbavení objektů typu `httpRequest` a `httpResponse`:

@UrlBinding – přiřazuje třídu ActionBean příslušnému formuláři v JSP

@HandlesEvent

@DefaultHandler – nastavuje, která metoda beany bude výchozí pro odbavování požadavků

@SessionScope – nastaví platnost sezení

@Wizard

Anotace pro validaci vstupních polí formulářů:

@DontValidate – validace uživatelských vstupů je vypnuta

@Validate – validace uživatelských vstupů je zapnuta

@ValidateNestedProperties

@ValidationMethod

Ostatní anotace:

@Before

@After

@SpringBean – spojení s beanou Spring frameworku

Validace vstupních polí a lokalizace chybových hlášení

Soubor `StripesResources.properties` obsahuje obecná chybová hlášení v anglickém jazyce. Hlášení je možné měnit a jejich množinu libovolně rozšiřovat. Pokud je potřeba lokalizovat chybová hlášení v češtině, tak postačí vytvořit soubor `StripesResources_cs.properties` s následující strukturou.

Níže je vidět ukázka, jak funguje zobrazení chybového hlášení, pokud uživatel zadá nevalidní vstup do formulářového pole.

```
stripes.errors.header=<div style="color:#b72222; font-
weight:200;">Opravte prosím následující chyby:</div><ol>
stripes.errors.beforeError=<li style="color: #b72222;">
```

```
stripes.errors.afterError=</li>
stripes.errors.footer=</ol>

stripes.fieldErrors.header=
stripes.fieldErrors.beforeError=<span style="color: #b72222; font-
weight:200";>
stripes.fieldErrors.afterError=</span><br />
stripes.fieldErrors.footer=

# Resource strings used by the stripes:messages tag
stripes.messages.header=<ul class="messages">
stripes.messages.beforeMessage=<li>
stripes.messages.afterMessage=</li>
stripes.messages.footer=</ul>

# Validation error messages produced by converter classes
converter.number.invalidNumber=Hodnota ({1}) zadaná v poli {0} musí
být platné číslo
```

Nejprve je vytvořeno formátování pro chybně zadaného pole – v tomto případě bude podbarveno žlutě a zvýší se tučnost fontu. Následuje stejné nastavení pro řádkové[21] objekty .

3.3.2 Apache Struts

Tento framework je dílem Apache Software Foundation a je součástí seskupení Jakarta, do kterého patří i další projekty vyvíjené v jazyce Java.

„*Prezentační vrstvu tvoří JSP stránky se speciálními Struts značkami, případně servlety, logickou část tvoří tzv. akční (Action) třídy a modely jsou komponenty JavaBeans. Všechny části dohromady spojí konfigurační soubor struts-config.xml, který dále přidává některé drobnosti navíc, jako je připojení do databáze s JDBC2 connection poolingem a podobně.*“

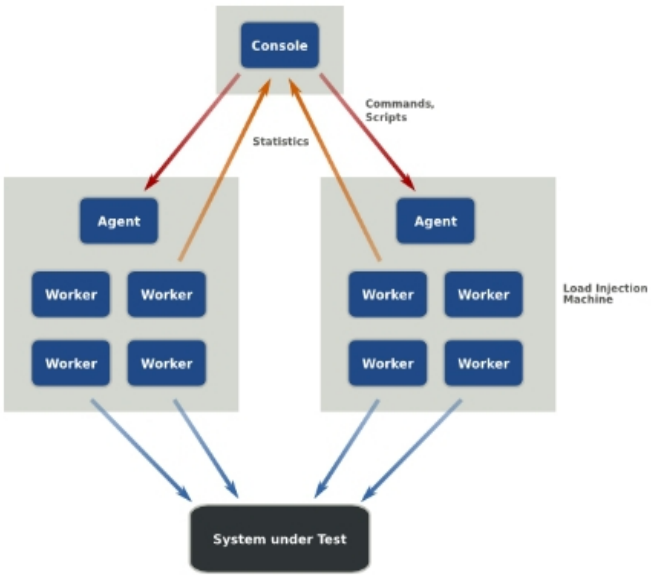
21 Tzv. inline tagy.

Porovnání se Stripes

Silnou inspirací pro vytvoření rámce Stripes byla autorova dlouholetá práce s frameworkem Struts. Stripes vychází z jeho logiky práce s formuláři, ale snaží se zjednodušit toto řešení. Pro obsluhu událostí na formuláři se používají také objekty typu `ActionBean`. Nicméně je ve frameworku Struts potřeba pro každý formulář (tedy JSP stránku s tagem `form`) vytvořit vlastní třídu typu `ActionBean`, která by jej obsluhovala, dále objekt typu `Form` a nakonec přidat konfiguraci do souboru struts-config.xml.. [14] Je zřejmé, že může být obtížné, při velkém rozsahu webové aplikace, udržet všechny tyto soubory v konzistentním stavu a samotné rozšiřování je poměrně zdlouhavé.

3.4 Ostatní frameworky

Do této skupiny by bylo možné zařadit rámce pro speciální užití, jako například framework pro distribuované testování aplikací, jímž je například Grinder.[22] Ten umožňuje skrze konzoli nadefinovat volitelné množství tzv. *agentů*, kteří řídí paralelní procesy pod rolemi *worker*. Tyto procesy potom vystupují jako klienti při komunikaci s testovanou aplikací. Vše je znázorněno na obrázku č. 6.



***Obrázek 6**: Systém testování ve frameworku Grinder*

Konzole poté sbírá statistiky, které zobrazuje. Lze nastavit frekvenci odesílání reportů, které jednotlivé role *worker* zasílají na konzoli. Tento framework je vhodný pro samo-testování vývojářem. Umožňuje nadefinovat si testovací úlohy na jednotlivé funkce testované aplikace.

22 Framework napsaný v Javě pro testování webových aplikací. Viz http://grinder.sourceforge.net/

4 Klasifikace databázových systémů pro uchování geografických dat

V této kapitule bude představen přehled databázových systémů, které jsou schopny pracovat s geografickými daty.

Databázové systémy (DBMS) jako takové, byly primárně vyvinuty pro efektivní uchování **alfa-numerických dat**. Jednalo se zejména o podporu standardních aplikací např. pro správu obchodních dat. Nejčastěji byly tyto data uchovávány v datových typech jako jsou `char, double, integer.` V posledním desetiletí se ale vývoj databázových systémů, stejně tak jako aplikací samotných, ubírá také směrem k optimalizaci ukládání dat pro nestandardní aplikace. Jedná se o systémy pro lékařský výzkum – např. počítačová tomografie, která používá velké objemy rastrovách dat nebo systémy pracující s daty ve formátu polygonů (reprezentace mapových dat).

Druhými jmenovanými systémy se zabývá tato práce. Vyvstávají zde nové požadavky na samotné dotazování do databáze. Častým krokem je vyhledání, zda je nějaký bod součástí plochy nebo linie, zda dvě linie mají průsečík apod. Výpočet takovýchto kroků musí být rychlý a efektivní, což uložení v klasickém relačním databázovém systému neumožňuje. Nejsou zde k dispozici podpůrné funkce pro zjištění vztahu mezi geografickými daty.

Jako hlavní důvody proč je uložení v konvenčních databázových systémech neefektivní uvádí Ben Chin Ooi v knize *Efficent Query Processing in Geographic Systems* tyto důvody:

1. *Objem dat je obvykle velmi rozsáhlý (např. informace o hranách) a samotné získání geografických dat z databáze je nákladné.*
2. *Datové typy se sestávají z komplexních objektů jako jsou linie a regiony.*
3. *Prostorové operátory jako často mnohem komplexnější než ty numerické.*
4. *Nadefinování řazení prostorových objektů je mnohem složitější.*
1. *Entity a vztahy jsou mnohem více složitější jak ve smyslu pochopení tak samotného výpočtu nad nimi.* [15].

Relační databázové systémy jako takové většinou neobsahují funkce pro optimální uložení geografických dat (PostgreSQL obsahuje podporu datového typu Geometry, ale neobsahuje efektivní funkce pro práci s ním). Pro některé databázové systémy (DBMS) existují rozšíření, která přidávají vhodné datové typy a příslušné funkce. Požadované datové typy jsou schopny uchovat hodnotu multidimenzionální (koordinát x a y), což žádný primitivní datový typ neumožňuje. Požadavkem na daný DBMS je, aby byl efektivní jak v ukládání polohových tak konvenčních typů dat a nabízel dotazovací jazyk společný pro oba typy dat.

Pro malé objemy geografických dat je použití běžného databázového systému možné, ale s narůstající velikostí sady dat se stává takovéto řešení nepoužitelným. Pokud by nad daty byly prováděny pouze jednoduché dotazy a nebyly by požadovány výpočty související se změnou projekce, tak je také použití klasických databázových systémů bez rozšíření možné, nicméně se v praxi nevyužívá.

Základní funkce, které jsou pro nutné pro práci s daty v GIS:

- zjištění, zda existuje průnik dvou geografických objektů
- výpočet vzdálenosti dvou geografických objektů
- zjištění nejbližších sousedních geografických objektů
- porovnání dvou objektů, zda jsou si rovny

Matematickým základ pro popis polohy na povrchu Země vychází z geometrie v eukleidovském prostoru – používá se souřadný systém, kde platí, že rovnoběžky se neprotínají a součet úhlů v trojúhelníku je 180°. Stejný prostor se využívá i u klasických map. Pro určení přesné polohy bylo nutné vytvořit jednotný souřadnicový systém odvozený z proporcí Země.

Rozmanitost aplikací vede k několika návrhům, jak definovat prostorové datové typy. Pro nadefinování pozice v dvojrozměrném prostoru je použit vázaný pár reálných čísel – *souřadnic*. To přináší několik obtíží, se kterými je nutné počítat. Prakticky nelze v počítačové reprezentaci uchovat reálná čísla beze ztrát, vždy je uložena jejich konečná aproximace. Je proto nutné zvolit dostatečnou přesnost, aby byly data odlišitelná.

Výsledné promítání do konkrétního mapového zobrazení je vypočteno až ve finální části zobrazovacího řetězce a není tedy uloženo přímo v databázi. Výhodou tohoto přístupu je nezávislost dat na typu zobrazení. [16]

4.1 Reprezentace prostorových dat

V této kapitole jsou nastíněny postupy jak uchovávat geografická data v počítači, aby s nimi bylo možné pohodlně pracovat. Na problematiku je pohlíženo i historického hlediska.

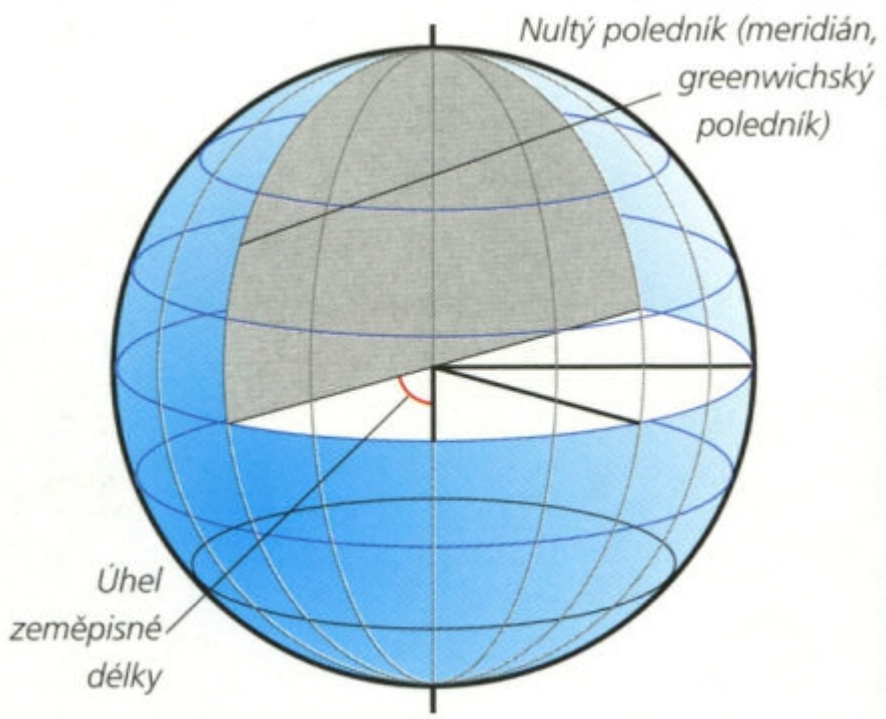
Pro práci s prostorovými daty bylo nutné vytvořit jazyk, pro jejich popis. V minulosti to byly mapy, které podobně jako notový zápis, pomáhaly popsat skutečnost velmi obtížně zachytitelnou. V rozměrech překračujících lidské měřítko je poměrně nesnadné udělat si představu např. o podobě kontinentů. Z map je také možné vyčíst to, zda je terén přístupný, podobně jako jestli je nějaká linka zazpívatelná nebo nikoliv.

Bylo proto třeba vytvořit schéma v němž by bylo možné popsat povrch Země. Vznikl globální systém poledníků a rovnoběžek jako základní souřadný systém, od kterého se dá odvodit

poloha na povrchu Země. Pro tvorbu souřadného systému se tvar planety aproximuje na elipsoid, neberou se tedy v úvahu geologické nerovnosti zemské kůry, byť rozdíly jsou v řádu kilometrů.

Poledníky, také označované jako zeměpisné délky či meridiány (anglicky *longitudes*, zkráceně ***lon***), jsou určeny řezem roviny jdoucí osou Země. Jsou definovány takto: „*Měříme úhel mezi středem země a bodem na povrchu země, po rovníku od hlavního poledníku na východ (východní délka) a na západ (západní délka) v rozmezí od 0° do 180°* [17] Nultý poledník prochází britskou observatoří v Greenwiche.

Na obrázku č. 7 je zobrazeno odvození poledníků.

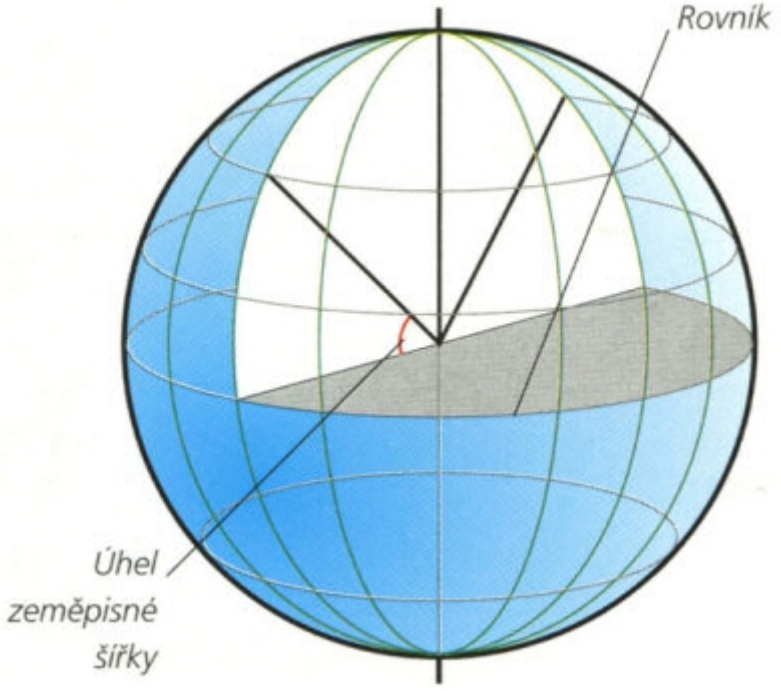


Obrázek 7: *Systém poledníků, převzato z [17]*

Rovnoběžky, také označované jako zeměpisné šířky (anglicky *latitudes*, zkráceně ***lat***) jsou tvořeny řezy rovinou kolmou k zemské ose. Nultou rovnoběžkou je rovník, který dělí zemi na dvě polokoule severní a jižní.

Rovnoběžky jsou definovány takto: „*Měříme úhel mezi středem země a bodem na povrchu země, po hlavním poledníku z rovníku (0°) na sever (severní šířka) a na jih (jižní šířka) a je v rozmezí od 0° do 90°*.“ [17]

Na obrázku č. 8 je zobrazeno odvození rovnoběžek od referenčního elipsoidu.



Obrázek 8: *Systém rovnoběžek, převzato z [17]*

V této práci budu dále používat anglické zkrácené označení **lon** pro poledníky a **lat** pro rovnoběžky, protože se jedná o ustálené pojmy používané při vývoji geografických informačních systémů.

4.2 Reprezentace dat pro počítačové zpracování

Výše popsaný souřadnicový systém popisuje polohu na zemském povrchu v tomto tvaru: 40° 51′ 13″ N, 78° 02′ 30″ W. Protože je ale šedesátková soustava nevhodná pro výpočty, převádí se tento popis souřadnic do desetinného formátu.

Převod je prováděn podle následujícího vzorce:

$$DD = dec + \frac{min}{60} + \frac{sec}{3600}$$

Legenda:

- DD – decimal degrees – desetinné stupně
- dec – decimal – stupně
- min – minutes – minuty

- sec – seconds – sekundy

Výsledkem je pak desetinný formát: 40.853611, 78.041667. Pro převod opačným směrem se používají tyto vzorce:

$$D = trunc(DD) \qquad M = trunc(|DD| * 60) \, mod \, 60 \qquad S = (|DD| * 3600) \, mod \, 60$$

S daty v tomto formátu již lze provádět výpočty, nicméně jak bylo zmíněno výše, tato reprezentace má tu nevýhodu, že dochází k zaokrouhlování. Pro uložení do databáze se tento formát dál zakóduje, podle toho jak jsou jednotlivé databázové systémy navrženy. Při dotazování na data jsou opět uživateli zobrazeny v této čitelné podobě.

Přehled prostorových rozšíření pro jednotlivé databázové servery

Níže je uveden seznam nejvýznamnějších nadstaveb pro databázové systémy, které podporují práci s geografickými daty:

- *PostGIS* pro PostgreSQL (volně šiřitelný balík)
- *Spatial Data* pro Microsoft SQL Server (komerční produkt)
- *SpatiaLite* pro SQLite (nepracuje na architektuře kleint-server, samotná databáze je uložena v souboru na disku. Dalším[23] specifikem je, že se jedná o netypovou databázi, ovladač SQLite rozpozná datový typ uložený ve sloupci podle hodnoty uložené v prvním řádku.)
- *Oracle Spatial* (komerční produkt)
- *Spatial extender* pro IBM DB2

Existují i samostatné databázové systémy speciálně navržené pro uchovávání geografických dat. Je to např. GeoDB, podobně jako SQLite je to souborová databáze.

Všechny výše uvedené rozšíření přistupují k datům podle specifikace, která vznikla pod hlavičkou mezinárodního konsorcia Open Geospatial Consortium (dále OCG). Byly načrtnuty základní vlastnosti, které by měly geografická data obsahovat, aby bylo možné s nimi pracovat standardizovaným způsobem a nadále se pracuje na nových verzích norem.

Pro potřeby této práce jsem zvolila databázový systém PostgreSQL s rozšířením PostGIS. Je to výchozí databázový systém v němž jsou spravována data z projektu Open Street Map a jsou tedy k dispozici nástroje pro import dat do databáze. Další výhodou je, že tento volně vyvíjený a šiřitelný projekt má velmi propracované možnosti řešící základní potřeby aplikací nad geografickými daty. Nespornou výhodou je aktivní komunita, která vytváří cenné informační zdroje, ze kterých lze čerpat při vývoji aplikace. V kapitole Implementace systému bude rozšíření PostGIS rozebráno podrobněji.

23 Více v referenční příručce pro SQLite: http://www.gdal.org/ogr/drv_sqlite.html

4.3 Uspořádání dat do topologie

Vektorová data (body a linie) lze uspořádat několika způsoby do složitějších struktur. Tyto modely se liší mírou využitelnosti pro popis topologických vztahů.

Špagetový model

„Tento model patří mezi nejjednodušší. Princip vychází z digitalizace map, kde se každý objekt na mapě reprezentuje jedním logickým záznamem v souboru a je definovaný jako řetězec x,y souřadnic. (…) Jeho nevýhoda spočívá v tom, že ačkoli jsou všechny objekty v prostoru definovány, struktura neposkytuje informace o vztazích mezi objekty, odtud také pochází název špagetový, je to soubor řetězců souřadnic nemající žádnou logickou strukturu. Další nevýhodou je způsob uložení sousedících polygonů. Společná linie je totiž ukládána dvakrát, pro každý polygon zvlášť.

Základní topologický model

V tomto modelu každá linie začíná a končí v bodě nazývaném uzel – node. Dvě linie se mohou protínat opět jenom v uzlu. Každá část linie je uložena s odkazem na uzly a ty jsou uloženy jako soubor souřadnic x,y. Ve struktuře jsou ještě uloženy identifikátory označující pravý a levý polygon vzhledem k linii. Tímto způsobem jsou zachovány základní prostorové vztahy použitelné pro analýzy. Navíc tato topologická informace umožňuje aby body, linie a polygony byly uloženy v neredundantní podobě. Jak špagetový, tak topologický formát mají velikou nevýhodu v naprosté neuspořádanosti jednotlivých záznamů. K vyhledání určitého liniového segmentu je třeba sekvenčně projít celý soubor.

Hierarchický model

Tento model odstraňuje neefektivnost při vyhledávání v jednodušším topologickém modelu pomocí ukládání v logicky hierarchické podobě. Vzhledem k tomu, že polygony se skládají z linií, které odpovídají jejich hranicím, a linie se skládají ze souboru bodů, jsou do modelu zahrnuty odkazy mezi jednotlivými druhy objektů (polygony, liniemi a body). Tyto odkazy pak umožňují mnohem snadnější vyhledávání jednotlivých objektů než v případě topologického modelu.“ [18]

Pro potřeby této práce je potřeba převést data ze špagetového modelu, ve kterém jsou nadefinovány mapové podklady, do topologického. Podrobněji se tímto bude zabývat kapitola. Tvorba topologie z dat.

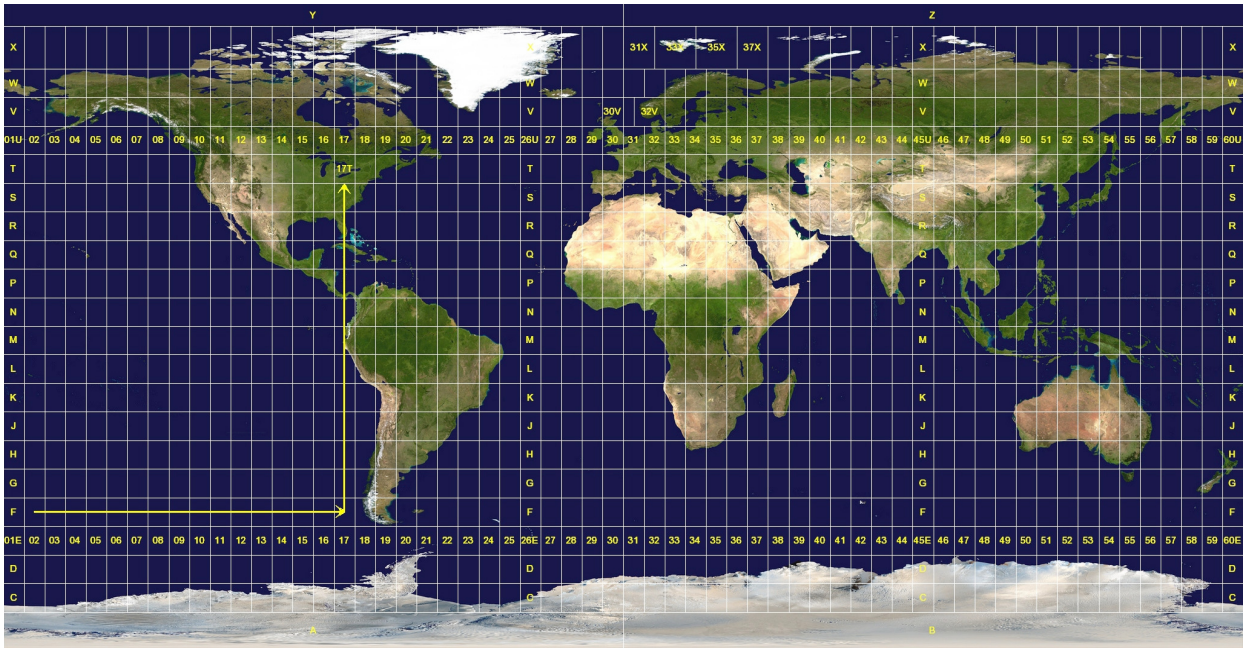
Obecně lze uspořádání dat do topologie využit nejen pro geografická data. Stále populárnější virtuální sociální sítě pracují na podobném principu. Osoby jsou jednotlivými uzly grafu a relace mají význam vztahu v reálném světě, který může mít přiřazenou různou váhu (ohodno-

cení). Příbuzenský vztah může mít vyšší význam než pracovní vztah nebo společná studia. Lze pak vybírat skupiny lidí na základě významu vztahu k dané osobě apod.

4.4 Souřadnicový systém UTM

Problematika určení polohy na elipsoidu (Zemi), který je promítán na rovinnou plochu, má dlouhou historii. V polovině dvacátého století byl americkou armádou vytvořen souřadnicový systém UTM (Univerzální transverzální Mercatorův souřadnicový systém), který používá síť mřížek. Zásadně se liší od systému zemských délek a šířek, protože se nejedná o jedno zobrazení, ale o soustavu šedesáti zón s různým mapovým zobrazením.

Tyto zóny jsou uspořádány do čtvercové sítě, s několika výjimkami, okolo osamělých ostrovů. Viz obrázek č. 9.



***Obrázek 9**: Souřadnicový systém UTM, Zdroj [29]*

Vzhledem k tomu, že geografická data nemají stejné zobrazení, tak je možné měření vzdálenosti podle Pythagorovy věty pouze v jedné zóně. Pokud je potřeba měřit vzdálenost přes několik zón, tak je potřeba použít prostorové funkce, které berou ohled na zakřivení Země.

To že geografická data patří do nějaké zóny určuje speciálně vytvořený identifikátor – SRID. V databázovém systému PostGIS lze informaci o tom, jaký mají data SRID lze nalézt v pohledu spatial_ref_sys systémového katalogu[24].

24 V předchozích verzích balíku PostGIS se jednalo o tabulku, jejíž hodnoty byly automaticky aktualizovány voláním funkce AddGeometry().

SRID je implementací OGC standardu, takže se s tímto termínem lze potkat ve všech systémech z OCG vycházejících. Existuje několik geodetických modelů elipsoidu (WGS), ne které je souřadnicový systém UTM použit. Jak se zpřesňovaly možnosti změřit tvar Země, tak se měnily i elipsoidy, podle nichž se popisovaly data. V minulosti se používaly modely z roku1960, 1966 a 1972.

V současné době se jich vyskytuje několik, ale nejpoužívanější je WGS 84 World Mercator.

- 4326 – WGS 84 Long Lat – Souřadnicový systém délek a šířek.
- 4269 – NAD 83 Long Lat – Souřadnicový systém délek a šířek.
- 3395 – WGS 84 World Mercator – Nejpoužívanější projekce pro webové aplikace. Výpočty jsou velmi rychlé, protože tento model pracuje s modelem, kdy by měla země tvar koule.
- 2163 – US National Atlas Equal Area

4.5 Databázový systém PostgreSQL

V této kapitole budou probrány základní specifika tohoto relačního databázového systému – s ohledem na téma práce. Popis si proto neklade za cíl být úplný a vyčerpávající.

Databázový systém PostgreSQL obsahuje bohatou podporu pro procedurální nadstavby jazyka SQL. Je to z toho důvodu, že dříve nebyla ve standardu SQL zahrnuta možnost jak větvit výběry dat (IF, CASE klauzule) a tak se používaly externí nadstavby nad SQL jazykem.

Pro vývoj uložených procedur a funkcí je připraven vlastní jazyk PL/pgSQL. Syntakticky vychází ze starších verzí jazyka PL/SQL od firmy Oracle.

Je možné také použití uložených procedur napsaných i v jiných jazycích, pro jejich používání je nutné přidání příslušného rozšiřujícího balíku. Některé jsou obsaženy v distribuci PostgreSQL, jiné je třeba doplnit ručně. Jazyky, které DBMS PostgreSQL podporuje jsou tyto:

PL/Tcl	C	http://www.idiap.ch/~formaz/doc/postgreSQL/xplang17074.htm
PL/Perl	Perl	http://www.postgresql.org/docs/9.1/static/plperl.html
PL/SQL	PL/SQL	http://www.techonthenet.com/oracle/
PL/Java	Java	http://pljava.projects.postgresql.org/
PL/PHP	PHP	http://www.commandprompt.com/community/plphp/
PL/Py	Python	http://python.projects.postgresql.org/

PL/R	R	http://www.joeconway.com/plr/
PL/Ruby	Ruby	http://raa.ruby-lang.org/project/pl-ruby/
PL/Scheme	Scheme	http://plscheme.projects.postgresql.org/
PL/sh	Unix shell	http://plsh.projects.postgresql.org/
PL/Lua	Lua	http://pllua.projects.pgfoundry.org/

Pro zjištění všech aktuálně dostupných balíků je možné použít pohled do systémového katalogu pomocí následujícího příkazu.

```
select * from pg_pltemplate;
```

Standardně to bývá PL/SQL, PL/TCL, PL/Perl a PL/Python. Pokud je podpora jazyka k dispozici, lze jeho podporu přidat do databáze příkazem `create langugage`.

```
CREATE LANGUAGE plpgsql;
```

Pokud požadovaný balík není v základní distribuci (jeho vývoj je oddělený), tak je třeba doplněk stáhnout, nainstalovat a až poté do databáze přidat požadovaný jazyk.

PostgreSQL je projekt s otevřeným kódem, nicméně existují i komerční varianty, které mají garantovanou podporu pro podnikové aplikace. Toto zajišťují produkty Enterprise DB a SRA.

4.6 PostGIS – nadstavba pro práci s geografickými daty

PostgreSQL jako takové obsahuje podporu datového typu `geometry`. Práce s ním je ale neefektivní, protože nejsou k dispozici vhodné funkce, které by usnadňovaly dotazy nad daty tohoto typu.

PostGIS je nadstavba pro databázi PostgreSQL, které je stejně jako tato databáze vyvíjena jako volně šiřitelný projekt. Základní funkcionalita je stejná jako u podobných databázových nadstaveb jako jsou *ArcSDE (Spatial Database Engine)*[25] od firmy ESRI nebo *Spatial extension*[26] od Oracle. Na rozdíl od výše uvedených je k dispozici zdarma.

Je napsáno převážně v jazyce C a používá několik externích knihoven jako je GEOS[27] či Proj4[28]. PL/pgSQL PostgreSQL umožňuje volání uložených procedur napsaných v jazycích Java, Perl, Ruby, Python, C, C++. Seznam dostupných jazyků je možné zjistit pohledem do systémového katalogu do tabulky pg_pltemplate.

```
select * from pg_pltemplate;
```

25 http://www.esri.com/software/arcgis/arcsde/
26 http://www.oracle.com/cz/products/database/options/spatial/index.html
27 Volný programový balík GEOS přidává funkce pro práci s geometrickými datovými typy v jazyce C.
28 Knihovna Proj4 slouží ke změně projekce v kartografickém zobrazení.

Datové typy, které PostGIS zavádí jsou tyto:

- *geometry* – dvourozměrný prostorový datový typ
- *geography* – elipsoidní prostorový datový typ
- *geometry_dump* – prostorový datový typ skládající se ze dvou polí – *geom* (typu geometry) a jedno až n-dimenzionální pole *path[]* pro uchování informace o uložení v objektu
- *box2d* – datový typ používající se pro ohraničení 2D objektu. Je reprezentován hodnotami xmin, ymin, xmax, ymax.
- *box3d* – datový typ používající se pro ohraničení 3D objektu. Je reprezentován hodnotami xmin, ymin, zmin, xmax, ymax, zmax. [19]

Rozšíření zavádí abstraktní datový typ geometry, který může být několika typů:

- POINT – jeden koordinát, je obvykle dvoudimenzionální i když může být popsán i třemi dimenzemi.
- LINESTRING
- POLYGON
- MULTIPOINT
- MULTILINESTRING
- MULTIPOLYGON
- GEOMETRYCOLLECTION

Databáze PostgreSQL obsahuje v základní verzi datové typy pro práci s daty o územní pozici – geometry; nadstavba PostGIS přidává *funkce* pro práci s těmito geografickými daty. Jejich počet přesahuje ve verzi 2.0 osm set. Přičemž funkce mohou být dvou druhů. Ty, které jsou vytvořeny dle SQL/MM standardu[29] mají předponu 'ST_'. Funkce bez předpony patří k jádru databázového systému PostGIS. Tato norma vychází ze specifikace OpenGis konsorcia – *Simple Features Specication for SQL* z roku 1999 [20].

Další zásadní vlastností, která bez rozšíření PostGIS nefunguje je indexování multidimenzionálních dat. Indexování je realizováno několika způsoby, v závislosti na tom, jaký typ dat má být indexován.

29 Norma ISO/IEC 13249 SQL/MM upravující rozšíření jazyka pro SQL pro geografické a multimediální aplikace.

- *„Indexy využívající B-stromy jsou používány pro data, která mohou být seřazena podle jedné osy. Jsou to např. číslo, text, datum. Data pro GIS nemohou být jednoduše seřazena podle jedné osy (co je větší (0,0) nebo (0,1) nebo (1,0)?) takže indexy založené na B-stromech nemají pro nás využití.*
- *Indexy postavené na R-stromech dělí data do obdelníků, pod-obdelníků, pod-pod-obdelníků, atd... Tyto indexy jsou použity v některých databázových systémech pro indexaci GIS dat, ale implementace těchto indexů není v PostgreSQL databázi tolik robustní jako GiST indexy.*
- *Indexy využívající obecný vyhledávací strom GiST (Generalized Search Trees) dělí data do skupin „věci na jedné straně“, „věci, které přesahují“,“věci, které jsou uvnitř“ a může být použit pro širokou škálu datových typů včetně dat pro GIS aplikace.*“ [21]

Další nespornou nevýhodou R-Strom indexů je, že nejsou tzv. „null safe“. To znamená, že nelze vytvořit index nad polem, které obsahuje i prázdné hodnoty (null[30]).

Funkce pro práci s geografickými daty jsou tohoto typu:

- `ST_Distance(geometry,geometry)` Vrací spočtenou vzdálenost mezi dvěma body v mapových jednotkách.
- `ST_AsText(geometry)` Vrací textovou reprezentaci datového typu geometry.
- `ST_AsGeoJSON(geometry)` Vrací geometrii ve formátu GeoJSON.

4.7 XML v databázi PostgreSQL

Databázový systém PostgreSQL nabízí stejně jako vyspělejší komerční databázové systémy podporu pro práci s daty ve formátu XML. Je zavedena podpora datového typu XML, jehož použití umožňuje dotazování na hodnoty jednotlivých elementů XML souboru. Což je funkcionalita, která by byla při použití datového typu VARCHAR velmi obtížně realizovatelná.

Pro export dat z tabulek do XML je připravena řada funkcí, s nimiž lze nadefinovat požadovanou strukturu XML dokumentu. Není ale vhodné tímto způsobem exportovat velké sady dat. Běžně se uvádí, že cca 40% datové velikosti XML dokumentu je použito pro popis struktury dat a nadefinování metadat. [22]

30 Hodnotou `null` je v programovacích jazycích reprezentován referenční odkaz, který nikam neodkazuje. V databázovém pojetí vyjadřuje hodnotu, která není známa nebo je nedefinovaná. Platí pro ní speciální pravidla pro množinové operace.

V této práci byl tento balík funkcí použit pro volitelné vygenerování KML souboru s výslednou trasou. Skript je přiložen v části Přílohy.

4.8 Realizace funkce routování v databázi

Kruciálním předpokladem pro realizaci funkce vyhledání trasy v databázi je uspořádání cest do síťové topologie. Geografická data, která se používají pro zobrazení v mapách nerespektují topologická pravidla. Není to jejich primárním účelem. Pro vizualizaci mapový dat je vhodnější, pokud jsou *plochy* definovány polygony (odpovídá to realitě), kdy jsou jednotlivá katastrální území zaměřovány co nejpřesněji s ohledem na hranice pozemku a není již kladen takový důraz na korektní napojení a vyznačení cest a stezek.

Jednoznačné vymezení cest a následné vytvoření topologie také komplikuje skutečnost, že se jedná v reálu o objekt s jistou plochou a při tvorbě topologického modelu je cílem vytvořit jeho zjednodušený popis, kdy cestu reprezentuje jedna *čára*. Je zřejmé, že tak vzniká velké množství možností v jakém bodě vlastně cesty (čáry) spojit.

Tato ne-jednoznačnost, ve spojení s orientací definovat útvary v geografických datech jako polygony a příliš se nezaměřovat na to zda je popis cest korektní a kompletní, značně komplikuje tvorbu samotné topologie.

Otevírá se tak prostor pro poměrně rozsáhlou kapitolou kterou je oprava chyb v datech. Existuje řada pomocných reportů, které v OSM databázi dokáží označit místa, kde není provedeno standardní napojení – například ke křižovatce vedou pouze jednosměrky, mezi napojeními existují mezery apod. Jednou z takovýchto aplikací, pomáhající prověřit a případně opravit mapová data je např. MapDust[31].

Další neméně důležitou věcí je korektní označení objektů v databázi OpenStreetMap. Pokud u nějaké cesty chybí označení, že se jedná o komunikaci, tak ji potom nelze vybrat pro tvorbu topologie. Není možné tuto komunikaci odlišit od jiných objektů na mapě, jako jsou např. hrany domů, hranice pozemků atd.

Při úpravách již existujících geografických dat se obvykle postupuje tak, že je nová cesta zanesena jako celek a pokud je cílem pouze vizualizace mapy, tak je napojena na již existující čáry pouze na svém začátku a konci. Vizuálně se jeví protnutí s jinými cestami jako reálné, ale v databázi se žádná taková informace nenachází. Pro pojmenování tohoto způsobu uložení dat se vžil název *špagetový model.* Vyznačuje se tím, že neřeší redundanci dat. Sousední polygony nemají společné hranice a jsou v databázi uloženy několikrát.

31 Oficiální stránky projektu jsou k vidění na http://www.mapdust.com/

4.9 Hledání nejkratší cesty za pomoci Dijkstrova algoritmu

Jedná se o grafový algoritmus, který principem vychází z prohledávání do šířky (breadth-first search, někdy uváděný jako BFS). Autorem je nizozemský matematik Dr. Edsger Wybe Dijkstra. Algoritmus pracuje nad grafem, který se skládá z vrcholů, které jsou spojeny ohodnocenými hranami.

„Nechť v0 je vrchol grafu, ze kterého chceme určit délky nejkratších cest. Budeme si udržovat pole délek zatím nalezených cest z vrcholu v0 do všech ostatních vrcholů grafu. Navíc u některých vrcholů budeme mít poznamenáno, že cesta nalezená do nich je už ta nejkratší možná. Takovým vrcholům budeme říkat trvale ohodnocené. Na začátku inicializujeme v poli všechny hodnoty na ∞ kromě hodnoty odpovídající vrcholu v0, kterou inicializujeme na 0 (délka nejkratší cesty z v0 do v0 je 0). V každém kroku algoritmu pak provedeme následující: Vybereme vrchol w, který není trvale ohodnocený, a mezi všemi takovými vrcholy je délka zatím nalezené cesty do něj nejkratší možná. Vrchol w prohlásíme za trvale ohodnocený. Dále otestujeme, zda pro nějaký vrchol v cesta z vrcholu v0 do w a pak po hraně z w do v není kratší, než zatím nalezená cesta z v0 do v a je-li tomu tak, pak změníme délku zatím nalezené cesty do v. Toto provedeme pro všechny takové vrcholy v.

Celý algoritmus skončí, pokud jsou už všechny vrcholy trvale ohodnocené nebo všechny vrcholy, co nejsou trvale ohodnocené, mají délku cesty do nich rovnou ∞ (v takovém případě se graf skládá z více nesouvislých částí).“ [23]

Podmínky pro realizaci algoritmu:

- **nezáporně ohodnocené hrany**: jako ohodnocení hrany se jeví jako nejvhodnější prostá délka cesty. Chodce neovlivňují jiné proměnné než je délka trasy. Přiřazení příslušného ohodnocení cestě je v databázi realizováno následujícím příkazem UPDATE.

```
alter table praha.ma_chodniky add column ohodnoceni REAL;
UPDATE praha.ma_chodniky SET ohodnoceni =
ST_Length(linestring);
commit;
```

- **graf je orientovaný:** Z pohledu chodce je zbytečné uvažovat nad obousměrnou nebo jednosměrnou průchodností cesty. V reálném světě se nevyskytují stezky u nichž by byla omezena průchodnost pouze jedním směrem. Pokud lze cestu projít jedním směrem, tak je možný i průchod opačně. Jistou výjimkou by mohly být eskalátory, i ty se

však vyskytují v páru. Proto pro zjednodušení práce s daty bude v této práci uvažováno pouze s obousměrnými cestami pro chodce.

Existuje mnoho modifikací tohoto algoritmu. Efektivním zrychlením je použití prioritní fronty pro uložení odkládaných vrcholů, potom je vždy první vybíraný prvek z fronty ten nejbližší.

5 Analýza a návrh aplikace

Hlavním požadavkem je umožnit uživateli vyhledat cestu k vhodným zastávkám po stezkách po nichž se může pohybovat i chodec. Ze zadaného bodu – místa, kde se nachází uživatel, je třeba vyhledat zastávku městské hromadné dopravy.

5.1 Specifikace požadavků

Výběr výchozího bodu na mapě bude proveden poklikem na zobrazenou mapu. Nebude nutné umístit výchozí bod přímo na stezku, aplikace automaticky aproximuje bod na nejbližší stezku.

Interakce uživatele se systémem bude spočívat v zadání bodu, ze kterého mají být vyhledány nejbližší dopravní stanice. Po zhodnocení, zda je výběr správný uživatel zadá požadavek na vyhledání zastávky.

Po zobrazení cesty nalezené zastávce může uživatel zvolit přesměrování na rozhraní externí aplikace IDOS s předvyplněným vyhledávacím polem ODKUD.

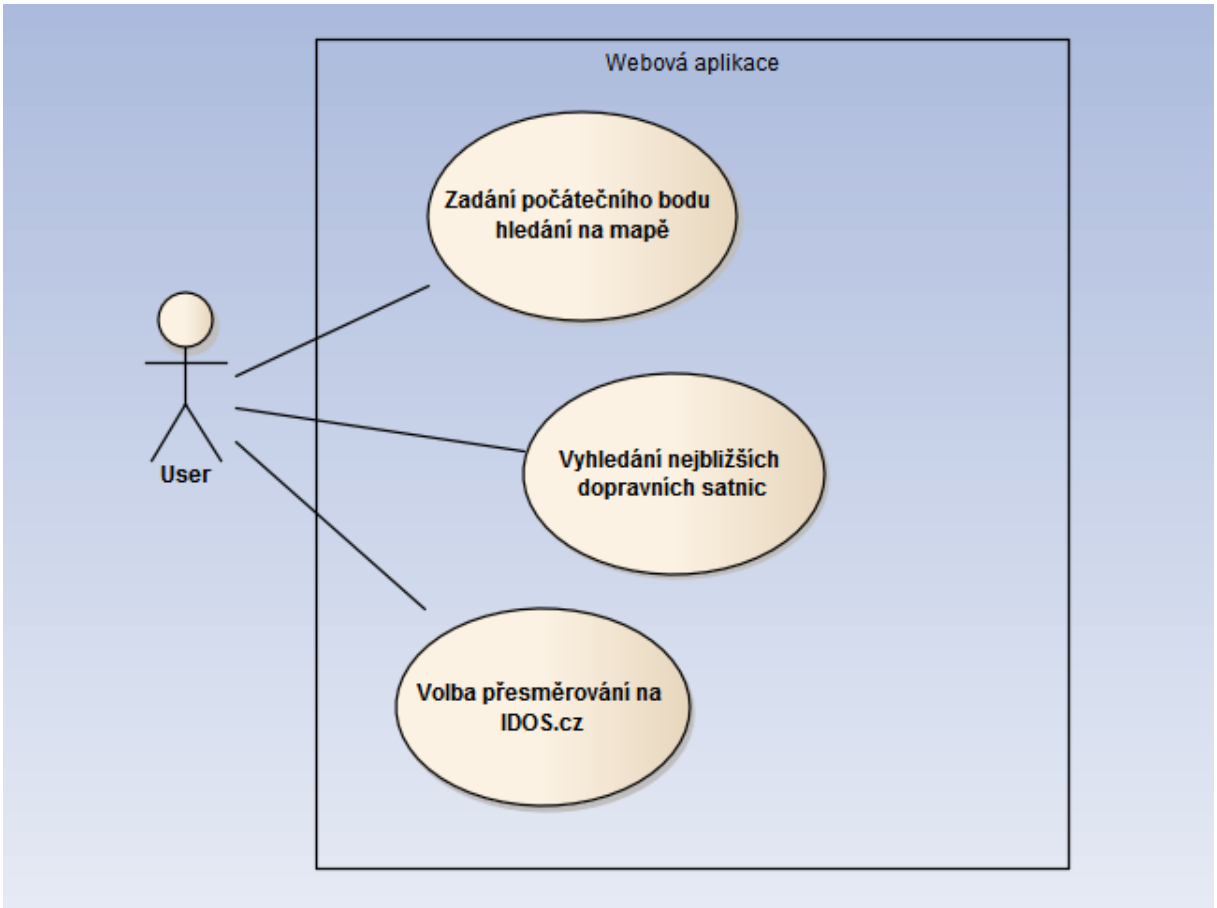
Funkční požadavky na systém:

- Vyhledání nejbližší dopravní stanice.
- Zobrazení trasy k této stanici uživateli.
- Přesměrování na webovou službu IDOS.

Non-funkční požadavky na systém:

- Rychlost odezvy systému bude v řádu sekund.

Na obrázku č. 10 je digram případů užití (*use case* diagram), který ukazuje interakci uživatele se systémem.



Obrázek 10: *Diagram případů užití*

5.2 Výběr vhodných mapových podkladů

V následující části budou představeny webové služby poskytující mapové podklady a provedeno zhodnocení vhodnosti jejich použití v této práci.

5.2.1 Google Maps

Celosvětově nejrozšířenější webovým rozhraním poskytující různé služby nad kartografickými daty jsou bezesporu Google Maps. V jistém smyslu udává technologický směr, jakým se budou webové mapové aplikace v budoucnu ubírat. Komerčně úspěšná společnost stojící v pozadí tohoto projektu umožňuje realizovat poměrně odvážné plány, jako je například fotografická dokumentace ulic z pohledu projíždějícího auta (projekt Street View).

Jako první bylo spuštěno vyhledávání tras do zadaných bodů, později i s možnost zvolit pro jestli má být vyhledávání provedeno pro automobil, cyklistu nebo chodce.

Aplikační rozhraní (dále API) této služby je k dipozici zdarma do určitého počtu požadavků, které je možné po serveru žádat. Pro velmi vytížené aplikace je třeba použít paušálně placenou verzi API.

5.2.2 Yahoo!

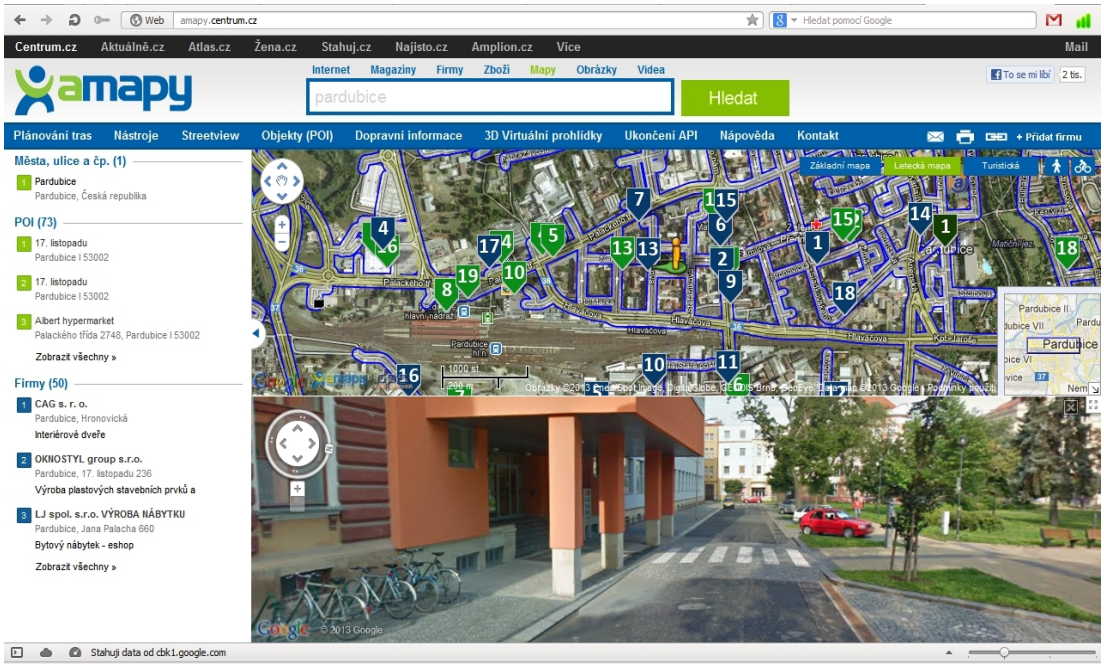
Dalším významným zahraničím hráčem na trhu jsou mapy Yahoo![32] mající ale řádově menší návštěvnost. Disponují vlastními daty a co se týče funkcionality, tak poskytují standardní služby, které ovšem v porovnání s ostatními poskytovateli nijak nevybočují.

5.2.3 Mapy.cz

U nás se těší uživatelské oblibě především aplikace mapy.cz od společnosti Seznam.cz. Před několika lety se mohla služba chlubit nejpodrobnějšími daty v odlehlejších oblastech České republiky. V současnosti je kvalita zmapování u většiny služeb srovnatelná.

5.2.4 Atlas.cz

Další často využívanou alternativou je provozovatel atlas.cz, který své API výrazně pozměnil na konci roku 2012[33]. Nyní spojuje vlastní mapové podklady a některé služby z Google Maps jako je StreetView, jak je vidět na obr. 11.



***Obrázek 11**: Atlas mapy*

32 Dostuné na adrese http://maps.yahoo.com/
33 Konkrétně k datu 29. 2. 2012, kdy přešel na novou technologickou platformu.

Aplikace http://mapy.idnes.cz má vzhled přizpůsobený primárně pro motoristy, tomu odpovídá velmi výrazné značení dopravních tahů.

Všichni výše vyjmenovaných poskytovatelé dávají k dispozici API skrze jež je možné vkládat objekty s mapovou službou do vlastních webových stránek. Možnosti, které jednotlivá aplikační rozhraní nabízejí jsou různá. Nejvíce služeb nabízí rozhraní firmy Google, ale jistým omezením je, že od 25 000 načtení denně je služba placená a je pro plnou šíři služeb zapotřebí přejít na *Maps API for Business.*

5.2.5 OpenStreetMaps

Jelikož všechny výše uvedené kartografické služby neumožňují přímý přístup k datům a práci s nimi vlastními silami, vzniklo sdružení *OpenStreetMap Foundation*[34], které si klade za cíl vytvořit volně dostupnou, volně použitelnou, sadu kartografických dat.

Sdružení provozuje několik serverů, které slouží jako datové úložiště. Existují také webové servery pro provoz mapové služby s volně přístupným API. Tu lze nalézt na adrese[23].

Sběr dat pro projekt Open Street Maps je velmi podobný tomu jak funguje otevřená encyklopedie Wikipedia. Data jsou volně upravitelná zaregistrovanými uživateli a je k dispozici logování těchto úprav.

Vzhledem k dostupnosti zdrojových kartografických dat a možnosti využít i webovou mapovou službu se jeví tato služba jako nejvhodnější.

5.3 UML diagramy

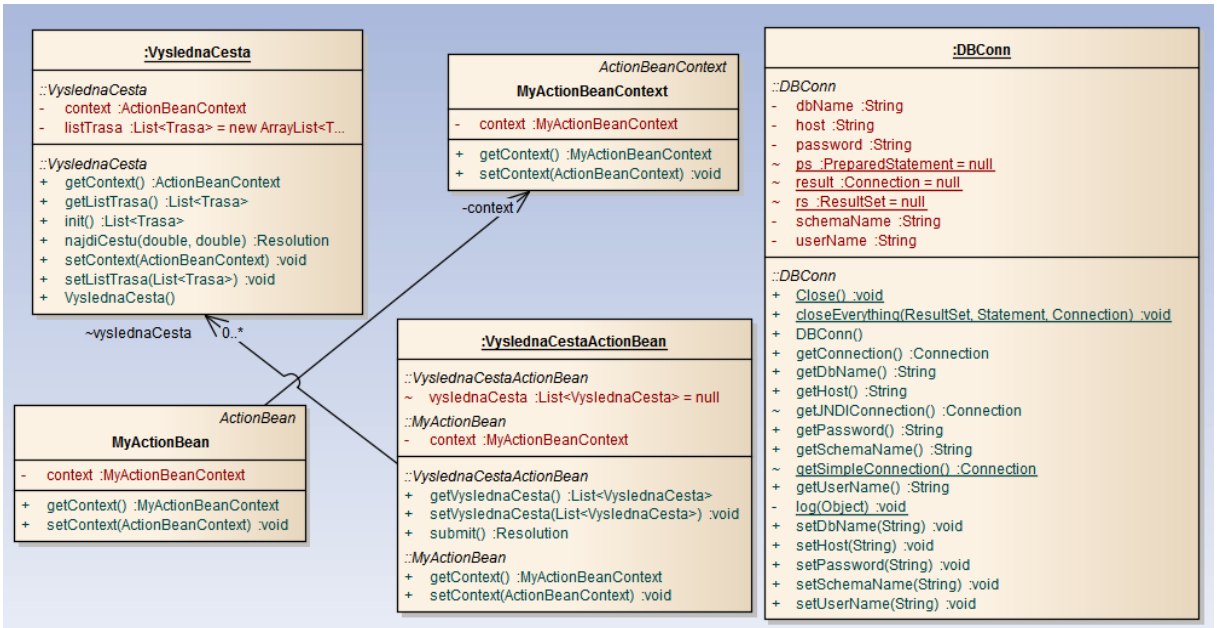
V této kapitole bude navržen systém pro lokalizaci dopravních stanic.

5.4 Diagram tříd

Vzhledem k tomu, že aplikace obsluhuje jednu JSP stránku s vloženou mapou a samotný výpočet nejkratší trasy zajišťuje několik databázových funkcí, tak není počet tříd nijak rozsáhlý.

Pro obsluhu formuláře pomocí frameworku Stripes je třeba mít nadefinovanou třídu dědící od rozhraní ActionBean. Vztahy jsou zobrazeny na obr. č. 12.

34 Oficiální web sdružení: http://www.osmfoundation.org/



***Obrázek 12**: Diagram tříd kontroleru*

Logika aplikace, která bude mít na starosti vyhledávání nejkratší cesty bude na straně databázového serveru. K tomuto rozhodnutí vedlo několik skutečností. Javová aplikace se k databázi připojuj pomocí JDBC připojení, které má poměrně velkou režii a je tedy pomalé. Je proto vhodné omezit počet přenosů na minimum.

Objem dat, který musí algoritmus prohledat je někdy opravdu velký a přenášet je do aplikačního serveru by nebylo účelné. I z pohledu víceuživatelského přístupu je databázový server lépe připraven na zpracování většího množství dat. Operace vyhledání nad kruhovým segmentem je nad sloupci, kde je uložen datový typ geometry. PostGIS je dobře vybaven indexovacími mechanismy pro urychlení těchto operací.

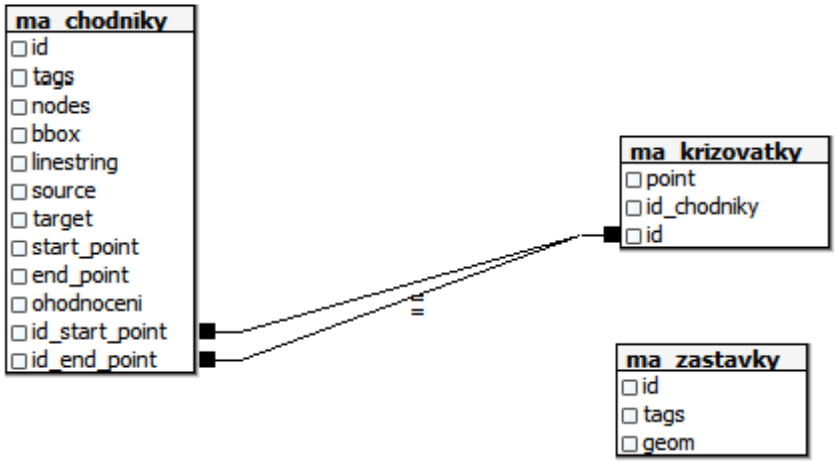
5.5 Návrh databázového systému

Z pohledu referenční integrity je databázový model systému velmi prostý. Obsahuje dvě tabulky – CHODNIKY a KRIZOVATKY. Přičemž koncový a počáteční bod chodníku jsou reference na bod v tabulce KRIZOVATKY. Viz ER diagram na obrázku č. 13.

V datech neexistuje žádná vazba mezi tabulkou ZASTAVKY, protože není zaručeno, že bude zastávka ležet na chodníku nebo dokonce na křižovatce. Vyhledání nejbližší křižovatky k zastávce probíhá vždy na vyžádání. Není ani účelné tuto informaci v databázi uchovávat, protože to je informace zajímavá pouze pro jeden konkrétní bod.

V tabulce CHODNIKY je podoba trasy uložena v poli linestring. Nad tímto sloupcem je volána funkce pro vygenerování GeoJSON souboru a je tak zajištěno, že vyhledaná trasa bude po zobrazení na webu lícovat s chodníky na mapě.

V tabulce KRIZOVATKY je GPS souřadnice uložena v poli point a v tabulce ZASTAVKY je to pole geom. Nad těmito sloupci byly vytvořeny databázové indexy pro zrychlení vyhledávání dat v těchto sloupcích.



***Obrázek 13**: Entitně relační diagram*

6 Implementace informačního systému

V této části bude popsáno praktické vytvoření webové aplikace nad geografickými daty. Hlavní část se věnuje práci s frameworkem Stripes.

Dále zde budou popsána jednotlivá API, se kterými webová aplikace bude spolupracovat.

6.1 Konfigurace Stripes frameworku

Před prvním sestavením projektu je třeba provést iniciální konfiguraci frameworku. Provádí se úpravou souboru `web.xml` umístěním ve složce `home\web\WEB-INF\`. Je zde potřeba nastavit JSP stránku, která bude výchozí a dobu platnosti session. Běžně se používá doba 30 minut.

```
<?xml version="1.0" encoding="UTF-8"?><web-app version="2.5"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
  <session-config>
       <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

Nyní je již možné aplikaci sestavit, ale není nastavena hlavní funkcionalita a to automatické vyhledávání událostí v kontextu aplikace. Toto nastavení patří do obslužné ActionBean třídy, kde je potřeba nadefinovat jednu metodu, která bude nastavena jako výchozí pro zpracování požadavků. Nastavení se provede za pomoci anotace `@DefaultHandler`.

```
@DefaultHandler
@HttpCache(allow = true)
public Resolution submit() {
…
```

Spojení JSP stránky a příslušné ActionBeany je provedeno tímto kódem:

```
<jsp:useBean id="manager" class="MapyApp.VyslednaCesta"
scope="page"/>
```

6.2 OpenStreetMap API

Aplikační rozhraní webové služby OpenStreetMap je k dipozici, stejně jako u jiných, v jazyce JavaScript. Ten slouží pro vkládání interaktivní mapy a konfiguraci spojení s mapovou službou na serveru.

JavaScript se používá pro práci s vkládanou mapou. „*Příběh JavaScriptu začal roku 1995 vývojář Brendan Eich pracující pro Netscape Communications Corporation. Vytvořil rannou verzi JavaScriptu, původně nazývaného LiveScript, později však byl přejmenován na JavaScript. Programovací jazyk Java byl v té době velmi populární, takže lze pouze hádat, že právě to vedlo k názvu JavaScript, protože JavaScript jako takový toho s programovacím jazykem Java nemá příliš společného. Společnost Sun Microsystems, tvůrce Javy, má autorská práva na název Java, takže je název JavaScript ve skutečnosti ochrannou známkou společnosti Sun Microsystems. JavaScript byl uveden na společné konferenci společností Netscape a Sun v prosinci roku 1995.*“[11]

Je to jazyk interpretovaný. [24]

POI v systému OpenStreetMap (dále OSM) nemají jednoznačný význam, tak jako tomu bývá zvykem u jiných mapových informačních systémů. Úzus je takový, že POI vyjadřuje nějaký významný bod na mapě. Jmenovitě např. čerpací stanici, kostel, turistickou památku atp. Oproti tomu OSM využívají POI i pro definici úseků komunikací.

Volně přístupné API umožňuje volit mezi různými typy zobrazení. Výchozí je možnost použití mercatorova zobrazení, což je cylindrické (válcové) zobrazení od vlámského kartografa Gerharda Mercatora. Používá se se pro námořní a letecké navigace. Poledníky jsou zobrazeny v reálném poměru, ale rovnoběžky jsou 'narovnány' a směrem k pólům tak dochází k dálkojevnému efektu a velkému zkreslení délky a plochy, jak je vidět na obr. č. 14. [25].

Obrázek 14: *Mercatorovo zobrazení, [23]*

I v této práci bude použito toto zobrazení, jelikož se jedná o standard se kterým se dnes běžně pracuje.

6.2.1 Získání dat z OpenStreetMap

Existuje několik způsobů, jak získat geografická data ze serveru OpenStreetMap.com. Prvotní rozhodnutí, které je třeba udělat, je zvolit v jakém formátu data vyexportovat. Na výběr je binární soubor PDB nebo XML soubor typu OSM.

Dalším krokem je volba, zda je třeba pracovat se sadou dat pro celou Zemi (soubor planet.osm) nebo pro projekt postačí použit její podmnožinu. Pro naplnění tématu této práce je vhodné zvolit část dat nad nimiž budou výpočty prováděny, protože následné dohledání dopravního spojení je možné pouze pro území České republiky.

Samotný výběr podmnožiny je možné provést několika způsoby. Webové rozhraní umožňuje stažení dat do OSM souboru podle hranice vybírané oblasti, které si uživatel navolí poklikem. Výsledná množina je ale limitována počtem vyexportovaných vrcholů – může jich být maximálně 15000, takže výsledné vyexportované území není nijak velké.

Další možností je použití nástroje Osmosis, který je určen k exportu a importu dat do různých typů souborů a databází. Tento programový balík umožňuje vyexportovat množinu dat, která není nijak omezena co do velikosti.

6.3 PostGIS

Užitečnou funkcionalitou je možnost zjištění seznamu tabulek, které obsahují datový typ GEOMETRY. Tento seznam je k dispozici pomocí databázového pohledu GEOMETRY_COLUMNS, které získává informace ze systémového katalogu PostgreSQL. Tento pohled lze využít jako pomůcku pro rychlejší orientaci ve schématu.[35]

35 V předchozích verzích balíku PostGIS (verze 1.9 a nižší) je tato informace k nalezení ve stejnojmenné tabulce, která byla aktualizována při spuštění funkce *AddGeometryColumn()*.

Instalace nadstavby PostGIS se sestává z několika kroků. Před samotným importem dat je třeba je nutné vytvořit uživatele, který bude moci přistupovat databázi a bude mít k dispozici slovník. Jsou k dispozici vytvořené spustitelné soubory v adresáři instalované databáze PostgreSQL.

```
C:\Program Files\PostgreSQL\8.3\bin\createuser -U postgres
<username>
C:\Program Files\PostgreSQL\8.3\bin\createlang -U postgres plpgsql
gis
```

Dále je třeba nainstalovat rozšíření PostGIS do databáze PostgreSQL. Aby byly k dispozici potřebné tabulky, do nichž proběhne standardní import dat z OSM souboru, je výhodné vytvořit novou databázi z tzv. Šablony. Poslouží k tomu následující SQL příkaz:

```
CREATE DATABASE osmpostgis TEMPLATE template_postgis;
```

Popřípadě lze použít výše zmíněné spustitelné soubory, kdy se přidá volitelný parametr T:

```
createdb -T template_postgis osmpostgis
```

Poté je třeba naimportovat stažený soubor s daty mapových podkladů. Je uložen v textovém formátu XML. Existuje řada projektů předpřipravených souborů s vyexportovanými daty pro různé oblasti. Zvolila jsem projekt *Metro Extracts*, dostupné na adrese http://metro.teczno.com/. Zde jsou ke k dispozici data světových měst v různých formátech – jako jsou OSM nebo PBF soubory. Druhé jmenované neobsahují kompletní sadu dat, ale pouze její denní export.

Zdrojem pro export jednotlivých měst, je soubor celé planety, který má v současnosti (březen 2013) v velikost 27 GB. Přičemž týdenní přírůstek dat je cca 5OO MB.

K dispozici je mnoho cest, jak tyto data do databáze importovat. Lze použít balíky Imposm[36], Osmosis, Osmium[37], Osm2pgsql či jiné. Liší se strukturou tabulek a tím i typem dat, které se ze souboru přenesou. Jedná se o specializované nástroje pro různé typy použití a je tedy důležité vybrat ten správný.

Balíky podobné Osmosis slouží pro aplikace generující z dat vizuální reprezentaci mapy. Není zde kladen důraz na korektní uložení toplogických informací, data jsou načítána duplicitně. Tzn, hrany polygonů se nejsou společné, ale překrývají se. Fyzické uložení v databázi také není normalizované, všechny data týkající se geometrie jsou často uloženy v jednom sloupci (struktura generovaná nástrojem osmosis).

36 Balík Imposm pro import dat z OSM je možné http://imposm.org/docs/imposm/latest/
37 http://osmiumapi.openstreetmap.de/

Pro snazší výběr vhodného nástroje pro import (a způsobu uložení dat) může posloužit následující tabulka č.1, která ukazuje jaké vlastnosti podporují různé balíky pro import dat.

Schema name	Created with	Used by	Primary use case	Updatable?	Geometries (PostGIS)?	Lossless?	Uses hstore columns?	Database
apidb	osmosis	API	Mirroring	Yes	No	Yes	No	PostgreSQL, MySQL
imposm	Imposm		Rendering	No	Yes	No	?	PostgreSQL
mongosm	MongOSM		Analysis	maybe	?	?	?	MongoDB
nominatim	osm2pgsql	Nominatim	Search, Geocoding	Yes	Yes	Yes	?	PostgreSQL
osm2pgsql	osm2pgsql	Mapnik, Kothic JS	Rendering	Yes	Yes	No	optional	PostgreSQL
osmsharp	OsmSharp		Routing	Yes	?	?	?	Oracle
overpass	Overpass API		Analysis	Yes	?	Yes	?	custom
pgsnapshot	osmosis	jXAPI	Analysis	Yes	Yes	No	Yes	PostgreSQL

Tab. č. 1: Ustálené způsoby uložení geografických dat do databáze. Zdroj: [26]

Já jsem zvolila poslední variantu. Funguje jako kopie vzorové databáze, přičemž je vytvořena sada objektů existujících v šabloně. Této vlastnosti lze využít i při vytváření záloh dat. Pro samotný import se používá příkaz s následujícími parametry

```
osm2pgsql -c -d osmpostgis -U postgres -S default.style -W -H
localhost -P 5432 prague.osm.bz2 -s
```

Přepínač -s zapříčiní, že data budou nahrávána méně paměťově náročnějším způsobem. Pokud je při importu k dispozici 64bitový systém s velkou operační pamětí, lze jej vynechat a docílit tak rychlejšího importu. Je zde také možnost importovat přímo komprimovaný soubor (bz2), nicméně se jedná o operaci náročnou na výkon procesoru. [19]

Import dat do databáze pomocí nástroje Osmosis. Je to aplikace umožňující různé typy importu a exportu dat pocházejících z projektu OpenStreetMap. Je možné data načítat jak do

databáze, tak do shapefile souborů. Stejně tak lze data exportovat, třídit, ořezávat pomocí *bounding boxu* (definice hranic) a generovat přírůstek[38].

Před samotným importem je třeba vytvořit schéma a tabulkami, do kterého budou data nahrávána. Ve zdrojovém balíku osmosis je k dispozici sada předpřipravených skriptů pro tento účel. Pro účely této diplomové práce nejvíce vyhovuje základní skript *pgsnapshot_schema_0.6.sql* a následně jej rozšířit o přidání nepovinného sloupce bbox do tabulky ways (*pgsnapshot_schema_0.6_bbox.sql)*. Tento sloupec omezuje cestu o hranice, které jsou pro výpočty zajímavé. Datový typ nového pole je geometry.

Dalším volitelným sloupcem do tabulky way je linestring, obsahující plnou cestu uloženou v datovém typu geometry. Nad tabulkou je dodatečně vytvořen index.

Význam většiny parametrů importního skriptu je zřejmý z názvu parametru. Osmosis umožňuje import a export mezi různými soubory a databázemi.

```
osmosis_latest\bin>osmosis --read-xml file="prague.osm" --write-
pgsql host="localhost" database="osm" user="zdnk" password="zdnk"
validateSchemaVersion=false
```

Import dat z osm souboru velkého 1,2 GB trval cca 20 minut na PC s konfigurací 6 GB RAM, 32 GB SSD disk pro zrychlení běhu systému, pevný disk s rychlostí 5400 t/min, 64-bitová platforma.

Po úspěšném importu je třeba zanalyzovat tabulky, aby databázový systém měl k dispozici statistiky o obsažených datech v tabulkách, pro tvorbu optimálních dotazů. Ty se týkají průměrného počtu řádků v tabulce, průměrné délky vět atp. Na základě těchto informací o datech je databázový systém schopen postavit efektivní exekuční plán dotazu.

To v praxi znamená to, že při spojení dvou tabulek (typ inner join) se volí jedna ze tří možných strategií:

- spojení pomocí vnořených smyček (*nested loop join*)
- spojení pomocí řazení sloučením (*merge sort join*)
- spojení pomocí hash klíče (*hash join*)

Samotný sběr statistik se v PostgreSQL provádí příkazem analyze <jméno tabulky>.

```
analyze users;
```

38 Jedná se o tvorbu tzv. diff souborů – porovnání změn mezi dvěma verzemi.

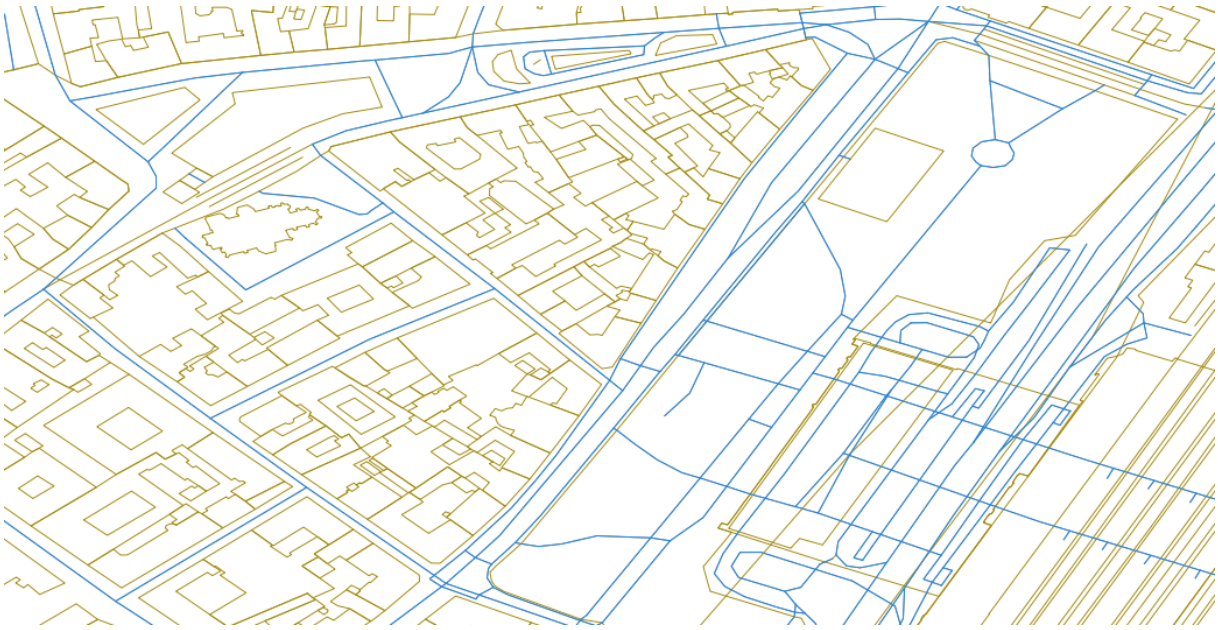
6.3.1 Tvorba topologie z dat

Po úspěšném importu dat do databáze je potřeba vybrat ty data, která jsou potřeba pro tvorbu topologie. Výběr dat byl proveden následujícím dotazem.

Standardizované označení pro cesty v databázi OpenStreetMap je pokrývá většinu případů dopravy. Cílem je vybrat pouze chodníky a stezky schůdné pro chodce. Ostatní data jako půdorysy budov a pozemků je nežádoucí načíst.

```
SELECT * FROM praha.ways WHERE (tags->'highway') in ('tertiary',
'bridleway','bus_guideway', 'cycleway', 'secondary', 'residential',
'footway', 'service', 'living_street', 'steps', 'trunk_link',
'unclassified', 'road', 'motorway', 'motorway_link', 'trunk',
'primary', 'primary_link', 'secondary_link', 'pedestrian', 'path');
```

Na obrázku č. 15 je vidět modrou barvou vyznačeny chodníky pro chodce. Světle hnědou jsou zobrazena data, která nejsou pro tuto aplikaci zajímavá.



***Obrázek 15**: Výběr chodníků z databázových dat, lokalita hlavní nádraží Praha. Vizualizováno programem QuantumGIS.*

Pro použití Dijkstrova algoritmu je potřeba vyznačit počáteční a koncové vrcholy hran. V aplikaci byla zavedena tato terminologie:

- hrana = chodník
- vrchol = křižovatka.

Rozšíření struktury tabulky a naplnění hodnot se provede následujícím skriptem.

```
alter table praha_chodniky add column start_point geometry;
UPDATE praha_chodniky SET start_point = ST_StartPoint(linestring);
commit;
alter table praha_chodniky add column end_point geometry;
UPDATE praha_chodniky SET end_point = ST_EndPoint(linestring);
commit;
```

Přičemž je potřeba myslet na to že pro algoritmus se cesta definovaná počátečním a koncovým vrcholem jeví jako jednosměrná. Jak bylo uvedeno výše, tak jsou chodníky brány vždy jako obousměrné, protože jsou průchozí z obou stran. Co se týče realizace této vlastnosti, tak je nejjednodušší přidání do tabulky chodníků stejný záznam s prohozenými počátečními a koncovými vrcholy. Při této variantě není potřeba nijak upravovat algoritmus pro hledání nejkratší cesty. Nevýhodou tohoto řešení je nárůst objemu dat.

Dále je třeba určit ohodnocení hran aby se mohl algoritmus rozhodovat, kterou cestu použije . Z důvodu, že vyvíjená aplikace je určena pro chodce, nemá význam sledovat maximální povolenou rychlost na komunikaci, což bývá primární parametr pro výpočet ohodnocení hrany v programech pro automobilovou dopravu. Proto pro výpočet ohodnocení postačí pouze délka hrany.

Rozšíření PostgreSQL nabízí několik funkcí pro výpočet vzdálenosti dvou bodů. Liší se v několika bodech.

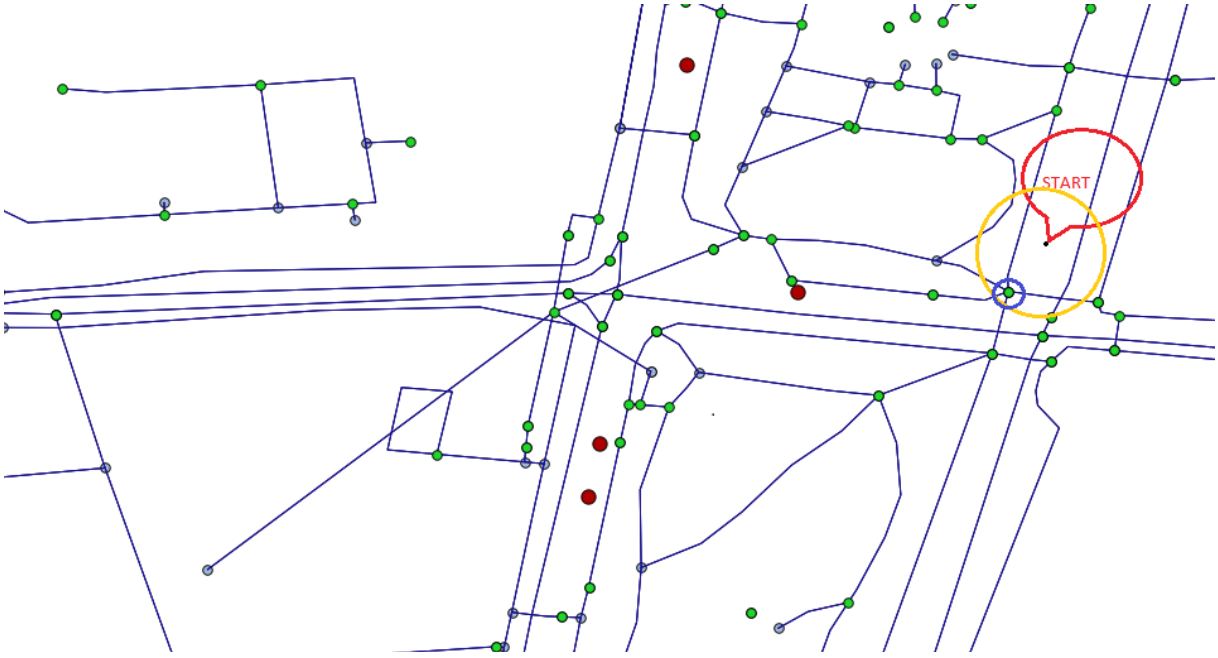
```
alter table praha_chodniky add column ohodnoceni REAL;
UPDATE praha_chodniky SET ohodnoceni = ST_Length(linestring);
commit;
SELECT assign_vertex_id( 'public', 'ways', 0.00001, 'linestring',
'id');
```

Umístění zastávek obvykle neleží na cestě (chodníku) a ani v jednotlivých křižovatkách. Vzhledem k tomu, že algoritmus hledání nejkratší cesty zpracovává pouze vrcholy (kři-

žovatky), tak je nutné dohledat pro příslušnou zastávku nejbližší křižovatku. Ta bude použita jako cíl pro algoritmus vyhledání cesty.

Samotné nalezení trasy v pgsql funkcí má několik kroků:

1. Uživatel může jako vstupní bod označit jakékoliv místo na mapě. Jelikož ale dijkstrův algoritmus umí vyhledávat pouze k vrcholu do vrcholu (z křižovatky do křižovatky), takže je potřeba mu dodat tyto body dodatečně. Pro vyhledání nejbližší křižovatky vzdušnou vzdáleností ze zadaného bodu na mapě byla použita funkce `pgis_fn_nn`(), která hledá nejbližší křižovatky v kruhovém segmentu zadané délky. [27] Vyhledávání probíhá v tabulce KRIZOVATKY. Nyní je nastaven vyhledávací okruh na 10 km. Pokud se v tomto okruhu nenajde žádná křižovatka, tak se nevyhledá žádná cesta. Na obr. č. 16 je zobrazeno kliknutí do mapy a vyhledání nejbližší křižovatky, která je následně označena jako startovací bod.



***Obrázek 16**: Znázornění zastávek (červené body), chodníků a křižovatek v programu QuantumGIS, výběr startovací křižovatky pro hledání nejkratší cesty*

2. Nyní je potřeba zvolit zastávku (na obrázku vyznačena červeným bodem), do které bude následně vyhledána nejkratší cesta. To se provede také funkcí `pgis_fn_nn`(), která tentokrát hledá v tabulce ZASTAVKY.
3. Jelikož ale vyhledaná zastávka s největší pravděpodobností neleží na křižovatce, tak je k této zastávce také potřeba dohledat nejbližší křižovatku. Ta bude pak označena jako cílový bod pro dijkstrův algoritmus. Takže je funkce `pgis_fn_nn`() volána potřetí, tentokrát z cílové zastávky a vyhledává se opět v tabulce KRIZOVATKY.

4. Nyní jsou již k dipozici obě křižovatky, takže je možné pustit algoritmus hledání nejkratší cesty. Výsledkem algoritmu je seznam vrcholů, kterými je nutné projít.
5. Pro získání geometrie chodníků je k těmto vrcholům dohledána geometrie chodníků.
6. Z této geometrie je vygenerován GeoJSON soubor a odeslán za pomocí AJAXu do webové stránky.

Informace o zastávkách jsou k dispozici v tomto detailu:

```
select id, tags, geom from praha.ma_zastavky where (tags->'name')
in ('Jindřišská');


25655264;""name"=>"Jindřišská", "tram"=>"yes",
"source"=>"Bing,cuzk", "railway"=>"tram_stop",
"public_transport"=>"stop_position"";"0101000020E61000006EF5413B4CD
C2C40F052454CE40A4940"
```

Ve sloupci tags je složenina různých informací o zastávce.

6.3.2 Zobrazení vektorové vrstvy z databázových dat

Pro vizualizaci kartografických dat je možné použít různé programové balíky. Nejčastěji jsou v současné době používány balíky OpenLayers nebo CloudMate[39]. Pro potřeby této práce jsem zvolila první jmenovaný, protože je k němu dispozici opravdu široká paleta tutoriálů a kvalitní dokumentace. [30]

OpenLayers je knihovna, která se používá na straně klienta. Spolupracuje s API OpenStreetMap a rozšiřuje je o další funkce jako je vkládání nových vektorových vrstev, přidávání ovládacích prvků apod.

Je to knihovna s otevřeným kódem. Využívá jazyka JavaScript. Vytvoření vrstev je možné z různých zdrojů (WMS,Yahoo! Maps, Nasa WorldWind, Google Maps, atp.). Vrstvy které tento balík dokáže vytvářet mohou být jak vektorové tak rastrové.

V této práci jsou OpenLayers použity pro načtení GeoJSON dat, které JSP stránce předává ActionBean z databáze.

```
var vector_layer = new OpenLayers.Layer.Vector("Trasa");
    vector_layer.style = {
```

39 Společnost Coudmate poskytuje API pro komerční použití s garancí dostupnosti a jinými službami. Zakladateli jsou členové pracující i na prvotním projektu OpenStreetMap.

```
            stroke: true,
            strokeWidth: 10,
            strokeColor: "#fff000"
        };
        map.addLayer(vector_layer);

        var geojson_format = new OpenLayers.Format.GeoJSON({
        'internalProjection': map.baseLayer.projection,
        'externalProjection': new OpenLayers.Projection("EPSG:4326")
    });
```

Je potřeba nadefinovat styl, který bude nová vrstva mít, poté přidat vrstvu do mapy a nastavit korektní projekci.

Je tedy třeba získat geometrii z databáze a následně ji načíst parserem OpenLayers. Data je poté nutné vložit do vektorové vrstvy mapy, která stopu umožní zobrazit na mapě.

6.4 IDOS API

Aplikace IDOS vyvíjená firmou CHAPS je komerční a její zdrojové kódy ani podkladová data o dopravních spojeních nejsou volně k dispozici. Nicméně je možné aplikaci předávat parametry přes URL adresu.

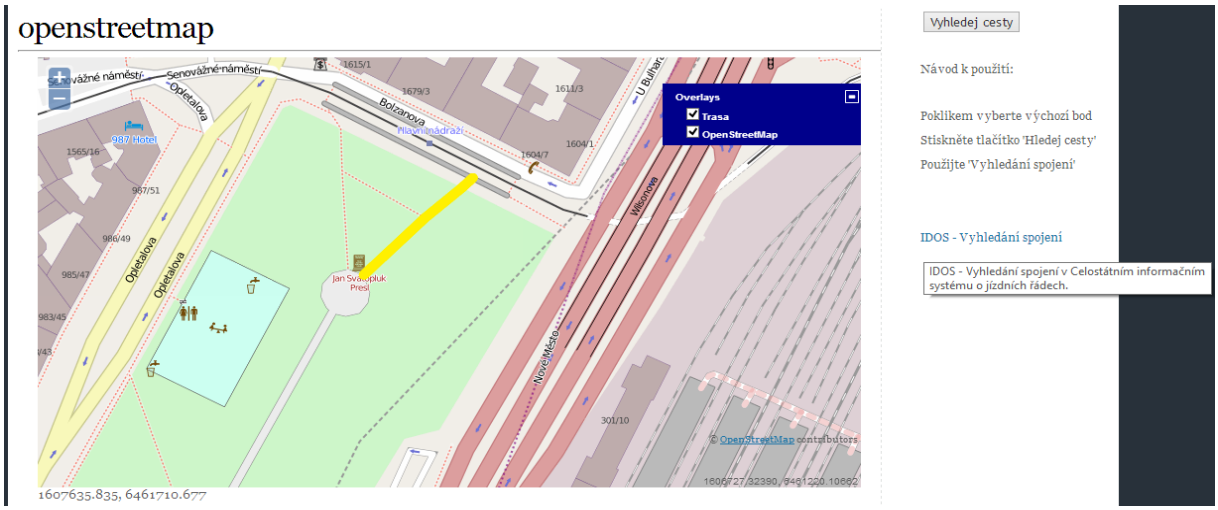
Po zobrazení trasy do stanice je možné přesměrování na formulář s vyhledáním spojení. Výsledkem je poté přesměrování na stránku aplikace, kde jsou předvyplněny vstupní pole formuláře. Povolenými parametry jsou počáteční a cílová stanice, typ spojení (vlak, autobus, městská hromadná doprava).

Kompletní specifikace API je dostupná na adrese [28].

6.5 Interakce uživatele s aplikací

Nejdříve uživatel poklikem zvolí bod na mapě. Jeho GPS souřadnice je vypsána pod oknem s mapou. Pokud je s výběrem spokojen, tak stiskem tlačítka 'Vyhledej cesty' zahájena straně databáze hledání trasy.

Ta je následně zobrazena na mapě žlutou trasou, jak je vidět na obr. č. 17.

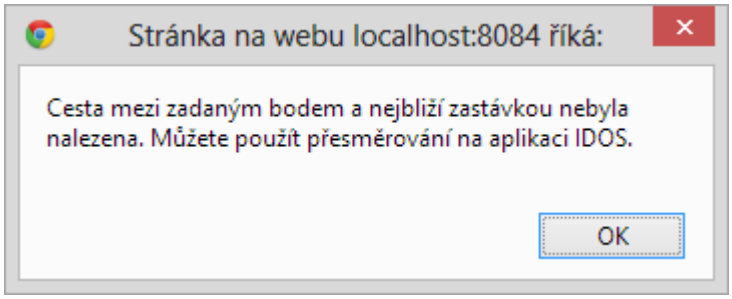


***Obrázek 17**: Interakce uživatele s webovým rozhraním aplikace*

Po vyhledání trasy se může uživatel přesměrovat na formulář pro vyhledané spojení, kde je předvyplněno pole Odkud. Viz. Obrázek č. 18.

***Obrázek 18:** Přesměrování na vyhledání spojení*

Může se stát, že při vyhledávání trasy, kdy je v kruhovém segmentu nalezena nejbližší startovací a konečná křižovatka, nebudou nalezeny takové, které mají mezi sebou spojnici. Potom nebude vrácena žádná trasa a zobrazí se hlášení jako je na obrázku č. 1.



***Obrázek 19**: Hlášení, pokud nedojde k nalezení trasy*

Přesměrování na aplikaci IDOS je možné i v tomto případě, za startovací podmínku se vezme bod, který si zvolil uživatel poklikem do mapy.

7 Závěr

Na závěr je vhodné shrnout výsledky práce a nastínit možné rozšíření řešení a modifikace. Cílem práce byla integrace frameworku Stripes do zajímavého webového projektu, kde slouží k obsluze prezentační a kontrolní vrstvy. Spolupracuje za pomoci AJAXu s rozhraním mapové služby a získává od ní data, která předává ke zpracování databázovému serveru.

Bylo provedeno shrnutí zástupců webových frameworků různých typů z pohledu pro využití pro cíle této práce. Dále byla rozebrána architektura Java EE, na které rámec Stripes stojí.

Dále byl předložen návrh databázové aplikace pro lokalizaci dopravních uzlů s možností dohledání dopravního spojení, která byla realizována na mapových podkladech OpenStreetMap. Výpočet nejkratší cesty byl realizován na straně databáze nad vlastní sadou dat hlavního města Prahy.

Jako další možnosti rozšíření by bylo určitě vhodné směrem k lepší uživatelské přívětivosti. Takže by bylo dobré zobrazovat mezivýsledky v podobě vyhledaného počátečního a koncového vrcholu, ze kterého bude vyhledávána nejkratší trasa. Případně využít AJAX, aby dynamicky vyhledával na pozadí nejbližší křižovatky na základě polohy kurzoru.

V algoritmu pro hledání tras by bylo vhodné eliminovat neúplnost v datech například tak, že by se po neúspěšném hledání vyzkoušelo hledání pro druhou a třetí nejbližší křižovatku. Dále by se daly rozšířit uživatelské vstupy, kdy by si mohl uživatel zvolit v jakém okruhu chce zastávky vyhledávat a případně by jich mohl vyhledat několik.

Téma práce bylo velmi zajímavé a obsáhlé, oblast geografických systémů se v současnosti velmi rychle vyvíjí a bylo velmi přínosné se s touto oblastí seznámit. Zejména, když jsou k dispozici data tohoto typu. Otevírá se tak velké množství možností jak tyto data využít v kombinaci s webovými službami.

Seznam zkratek

GUI – *Graphic User Interface* – Grafické uživatelské rozhraní aplikace.

GIS – *Geographic Information systém* – Geografický informační systém.

JVM – *Java Virtual Machine* – Javový virtuální stroj starající se o management spouštěných java tříd.

HTTP – *Hypertext Transfer Protocol* – Aplikační protokol pro zasílání zpráv mezi klientem a serverm.

MVC – *Model View Controller* – Architektura oddělující vrstvu zobrazení dat, logiku aplikace a komunikaci mezi těmito vrstvami.

JCM – *Java Community Process* – Standardizovaný proces pro vývoj JEE.

API – *Application Programming Interface* – Rozhraní pro programování aplikací.

JCP – *Java Community Process* – Standardizovaný proces pro vývoj součástí platformy Java.

JSF – *Java Server Faces* – Komponentně orientovaný webový rámec stojící na servletech a JSP, součást Java EE.

JDBC – *Java Database Connectivity* – Rozhraní pro jednotný přístup k relačním databázím.

LDAP – *Lightweight Directory Access Protocol* – Protokol pro přístup k souborům na adresářovém serveru.

CORBA – *Common Object Request Broker Architecture* – Standard pro spojení aplikací běžících na různých platformách.

RMI – *Remote Method Invocation* – Technologie umožňující vzdáleně volat metody jiné instance javového virtuálního stroje.

LAN – *Local Area Network* – Síťové prostředí s malým počtem klientů.

NDS – *Novell Directory Services* – Platforma pro správu sítí od firmy Novell.

POJO – *Plain old java object* – Java objekt, který není nijak rozšířen ani nic nezdědil.

IDE – *Integrated Development Environment* – Vývojové prostředí.

GPS – *Global Position System* – Systém umožňující detekovat polohu přijímače na zemském povrchu.

POI – *Point of Interest* – Významné místo na mapě.

GSON (GeoJSON) – Modifikovaná verze JSON pro práci s geografickými daty. Obsahuje prvky jako je bod, čára, polygon nebo kolekce geometrických objektů.

OGC – *Open Geospatial Consortium* – Mezinárodní konsorcium pro otevřenou standardizaci geoprostorových dat a služeb.

DBMS – Data Base Management System – Databázový systém, někdy též označovaný jako Systém řízení báze dat (SŘBD).

SQL – *Structured Query Language* – Strukturovaný dotazovací jazyk pro výběr dat z databázového systému.

KML – *Keyhole Markup Language* – Jedná se o XML schéma pro popis geografických informací, nejčastěji se používá na webových mapách. Byl nadefinován firmou Keyhole, která byla akvírována firmou Google a byl prvně použit pro produkt Google Earth.

UTM – *Universal Transverse Mercator* – Univerzální transverzální Mercatorův souřadnicový systém.

WGS – *World Geodetic System* – Mezinárodní geodetický systém. Definuje tvar geoidu, od kterého jsou dále odvozovány mapová zobrazení.

HTTP – *Hypertext Transfer Protocol* – Protokol pro výměnu HTML souborů v rámci sítě internet, funguje v režimu klient-server.

JSON – *JavaScript Objct Notation* – Notace pro práci s alfanumerickými daty. Vznikla pro práci s daty v JavaScriptu, ale rozšířila se mezi jiné technologie.

ORM – *Object-relational mapping* – Technologie pro mapování objektů z vyšších programovacích jazyků na tabulky v relační databázi.

EL – *Expression Language* – Standardní mechanismus, kterým lze v JSP stránkách dynamicky pracovat s daty. Slouží pro spojení prezentační vrstvy s logickým modelem aplikace.

Seznam obrázků

Seznam tabulek

8 Zdroje

[1] KORTE, George. *The GIS book*. 3rd ed., expanded for education and training. Santa Fe, NM: OnWord Press, c1994, xiv, 220 p. ISBN 15-669-0047-6.

[2] KAYAL, Dhrubojyoti. *Pro Java EE Spring Patterns: best practices and design strategies implementing Java EE patterns with spring framework*. Berkeley, CA: Apress, 2008, xx, 323 s. Expert's voice in open source. ISBN 978-1-4302-1009-2.

[3] GOODRICH, Michael T a Roberto TAMASSIA. *Data structures and algorithms in Java*. 5th ed. New York: J. Wiley, c2010, xxii, 714 p. ISBN 04-703-9880-9.

[4] Java Transaction Service (JTS 1.1). In: SLÍVA, Jiří. K*atedra informatiky: fakulta elektrotechniky a informatiky VŠB-TUO* [online]. Ostrava, 2000, Ne dub 2 16:18:08 [cit. 2012-10-21]. Dostupné z: http://www.cs.vsb.cz/grygarek/dosys/present/transact/jts-intro.html

[5] Remote Method Invocation: Model vzdálených objektů RMI. In: *VŠB | Katedra informatiky FEI VŠB-TUO* [online]. [2012] [cit. 2012-10-28]. Dostupné z: http://www.cs.vsb.cz/grygarek/dosys/inprise/rmi/rmi.html

[6] CORBA a IIOP. In: LAMPA, Petr. *Fakulta informačních technologií VUT v Brně* [online]. [2008] [cit. 2012-10-24]. Dostupné z: http://www.fit.vutbr.cz/~lampa/papers/corba.html

[7] JNDI Overview. In: *Oracle Documentation* [online]. [2012] [cit. 2012-10-28]. Dostupné z: http://docs.oracle.com/javase/jndi/tutorial/getStarted/overview/index.html

[8] CAVANESS, Chuck. *Programujeme Jakarta Struts: Tvorba webových aplikací se servlety a stránkami J*SP. Slavoj Písek. Praha: Grada Publishing, a. s., 2003, 468 s. O'Reilly & Associates, Inc. ISBN 80-274-0667-9.

[9] BURKE, Bill. *Enterprise JavaBeans 3.0*. 5th ed. Sebastopol: O´Reilly, 2006, 732 s. ISBN 05-960-0978-X.

[10] HALL, Marty. *Java: servlety a stránky JSP*. Praha: Neocortex, 2001, xviii, 585 s. ISBN 80-86330-06-0., s. 242.

[11] HOLZNER, Steven. *Mistrovství v Ajaxu: Naučte se programovat moderní aplikace*. Vyd. 1. Překlad Jakub Zemánek. Brno: Computer Press, 2007, 591 s. ISBN 978-80-251-1850-4. s. 45.

[12] MATULÍK, *Petr. Moderní JEE™ technologie a nástroje*: Kapitola 1. Aplikační rámec Spring - teorie. [online]. 2007 [cit. 2012-09-06]. Dostupné z: http://morosystems.cz/java/spring/ch01.php

[13] HEFFELFINGER, David R. *Java EE 5 development with NetBeans 6: develop professional enterprise Java EE 5 applications quickly and easily with this popular IDE*. Birmingham, U.K.: Packt Publishing Ltd., c2008, iv, 384 p. ISBN 978-1-847195-46-3.

[14] Documentation [online]. 2011 [cit. 2011-10-27]. *Stripes framework*. Dostupné z WWW: <http://www.stripesframework.org/display/stripes/Documentation>.

[15] OOI, Beng Chin. *Efficient query processing in geographic information systems*. New York: Springer-Verlag, c1990, viii, 208 s. Lecture Notes in Computer Science. ISBN 03-875-3474-1. s. 5.

[16] RIGAUX, Philippe, Michel O SCHOLL a Agnes VOISARD. *Spatial databases: with application to GIS.* San Francisco: Morgan Kaufmann Publishers, c2002, xxix, 410 s. ISBN 15-586-0588-6.

[17] Začínáme navigovat. *Občanské sdružení Jamesonovy děti moře* [online]. [2013] [cit. 2013-04-21]. Dostupné z: http://www.detimore.cz/zaciname-navigovat-cln15.php

[18] Studijní článek: Základní typy vektorových datových modelů. ZČU. *Úvod do geografických informačních systémů (GIS)* [online]. 2013. vyd. 2013 [cit. 2013-08-23]. Dostupné z: http://www.gis.zcu.cz/studium/ugi/elearning/msgisu02s04cz/default.htm

[19] Getting Started With PostGIS: An almost Idiot's Guide (PostGIS 2.0). *Boston Geographic Information Systems* [online]. 2012 [cit. 2012-12-17]. Dostupné z: http://www.bostongis.com/PrinterFriendly.aspx?content_name=postgis_tut01

[20] OGC® Standards. *OGC* [online]. 2010-08-04 [cit. 2012-12-29]. Dostupné z: http://www.opengeospatial.org/standards/sfs

[21] PostGIS documentation: Chapter 4. Using PostGIS. *PostGIS* [online]. 2013 [cit. 2013-05-06]. Dostupné z: http://postgis.refractions.net/documentation/manual 1.3SVN/ch04.html#id2728121

[22] XML Support in Postgis. SAMOKHVALOV, Nikolay. *CEUR Workshop Proceedings* [online]. 2006 [cit. 2013-07-16]. Dostupné z: http://ceur-ws.org/Vol-256/submission_15.pdf

[23] Recepty z programátorské kuchařky Halda, Dijkstrův algoritmus. *Korespondenční seminář z programování* [online]. [2013] [cit. 2013-08-23]. Dostupné z: http://ksp.mff.cuni.cz/tasks/16/cook1.html

[24] PÍSEK, Slavoj. J*avaScript: efektní nástroj oživení www stránek.* 1. vyd. Praha: ; Grada, 2001, 231 s. ISBN 80-247-0014-X.

[25] Map Projections. ROSENBERG, Matt. *About Geography* [online]. 2013 [cit. 2013-04-19]. Dostupné z: http://geography.about.com/library/weekly/aa031599.htm

[26] HLINĚNÝ, Petr. *Základy teorie grafů*. Elportál [online]. Brno: Masarykova univerzita, 2010. [cit. 2011-10-27]. Aktualizováno: Březen 2010. Dostupné z WWW: <http://is.muni.cz/elportal/?id=878389>. ISSN 1802-128X.

[27] PostGIS Nearest Neighbor: A Generic Solution. *Boston Geographic Information Systems [*online]. 2013 [cit. 2013-08-22]. Dostupné z: http://www.bostongis.com/PrinterFriendly.aspx?content_name=postgis_nearest_neighbor_generic

[28] Chaps spol. s r.o. [online]. 2011 [cit. 2011-10-13]. *Možnost využití odkazu na www.idos.cz.* Dostupné z WWW: <http://www.chaps.cz/idos-moznost-vyuziti-odkazu.asp#volba_objektu>

[29] *OpenStreetMap*: Otevřená wiki-mapa světa [online]. 2013 [cit. 2013-04-20]. Dostupné z: http://www.openstreetmap.org/

[30] ALUR, Deepak, John CRUPI a Dan MALKS. *Core J2EE patterns: best practices and design strategies.* 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, c2001, xxvi, 459 s. ISBN 01-306-4884-1.

9 Přílohy

Skript pro generování KML souboru z databáze. Použit jazyk PGSQL a balík pro práci s XML.

```
CREATE OR REPLACE FUNCTION public.get_kml_from_dijkstra(pocatek
integer, konec integer)
  RETURNS xml AS
$BODY$
select
    xmlroot(
          xmlconcat(
                xmlelement(
                      name "kml",

    xmlattributes('http://earth.google.com/kml/2.0' as "xmlns"),

                      xmlelement(NAME "Document",
                            xmlelement(NAME "Placemark",
                                  xmlelement(NAME "LineString",
                                        xmlelement(NAME "coordinates",
                                              string_agg(
                                                    st_x(point) || ', '
||  st_y(point) || ', 0.','   '))))))
                )), VERSION '1.0' )
from
praha.ma_krizovatky where ARRAY[id] <@
(select path from praha.dijkstra5( $1, $2));
$BODY$
  LANGUAGE sql VOLATILE
```

Vygenerovaný soubor KML.

```
<kml xmlns="http://earth.google.com/kml/2.0">
  <Document>
    <Placemark>
      <LineString>
        <coordinates>
              13.8421808, 50.3680868, 0.
              13.8452549, 50.3693865, 0.
              13.8507712, 50.3735321, 0.
        </coordinates>
      </LineString>
    </Placemark>
  </Document>
</kml>
```