UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Možné druhy vizualizace multidimenzionálních dat s využitím technologie Oracle Spatial a MapViewer

Jan Mach

Bakalářská práce
2013

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

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Mach**
Osobní číslo:	**I09182**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Možné druhy vizualizace multidimenzionálních dat s využitím technologie Oracle Spatial a MapViewer**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

V teoretické části práce budou popsány možnosti vizualizace prostorových dat a technologie Oracle Spatial a MapViewer.
V praktické části se student zaměří na vybudování aplikace umožňující různé druhy vizualizace multidimenzionálních dat pomocí technologie MapViewer.
Implementované vizualizace budou porovnány a budou navržena doporučení jejich využití.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

1. MURRAY, Chuck, et al. Oracle? Spatial : User?s Guide and Reference 10g Release 2 (10.2) [online]. [s.l.] : Oracle, 2006 Dostupné z WWW: http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14256.pdf.
2. MURRAY, Chuck, et al. Oracle? Spatial : Topology and Network Data Models 10g Release 2 (10.2) [online]. [s.l.] : Oracle, 2005. Dostupné z WWW: http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14256.pdf.
3. ŘEZANINA, P.: Návrh modelu železniční sítě s využitím technologie Oracle Spatial Topology and Network Data Model , Univerzita Pardubice, 2012.

Vedoucí bakalářské práce: **Ing. Jan Fikejz**
Katedra softwarových technologií

Datum zadání bakalářské práce: **21. prosince 2012**
Termín odevzdání bakalářské práce: **10. května 2013**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 29. března 2013

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen 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 Jan Mach

Poděkování

Rád bych zde poděkoval vedoucímu mé bakalářské práce Ing. Janu Fikejzovi za odbornou pomoc, hodnotné rady a důležité připomínky, které mi pomohly při zpracování bakalářské práce. Také bych rád poděkoval své rodině a kamarádům za veškerou podporu během mého studia, jmenovitě pak Jirkovi Pénzešovi a Štěpánu Kartákovi za jejich odbornou pomoc.

Anotace

V teoretické části bakalářské práce je popsána technologie Oracle Spatial a možnosti vizualizace multidimenzionálních dat pomocí této technologie a Oracle MapViewer. V praktické části jsou navrženy aplikace na vizualizaci prostorových dat a simulace pohybujících se kolejových vozidel po modelu železniční sítě. Aplikace jsou otestovány a je navrženo jejich využití.

Klíčová slova

Databáze, Oracle Spatial, GPS, geoprostorová data, poloha, multidimenzionální data, MapViewer, vizualizace, Oracle Maps, Java Server Page, Java applet, desktopová Java aplikace.

Title

Possible types of visualization of multidimensional data using Oracle Spatial and MapViewer.

Annotation

The theoretical part of the thesis describes Oracle Spatial technology and visualization of multidimensional data using this technology and Oracle MapViewer. In the practical part are designed application on the visualization of spatial data and simulation of rolling stock objects in the railway network model. Applications are tested and designed its use.

Keywords

Database, Oracle Spatial, GPS, geospatial data, position, multidimensional data, MapViwer, visualization, Oracle Maps, Java Server Page, Java applet, Desktop Java application.

Obsah

Seznam zkratek

Ajax	Asynchronous JavaScript and XML
API	Aplikační programovací rozhraní (Application Programming Interface)
BP	Bakalářské práce
CSV	Comma-Separated Values
FOI	Features of interest
GIF	Graphics Interchange Format
GUI	Grafické uživatelské rozhraní (Graphical User Interface)
HTTP	Hypertext Transfer Protocol
JPEG	Joint Photographic Experts Group
JS	JavaScript
JSP	Java Server Page
JVM	Java Virtual Machine
KV	Kolejové vozidlo
LRS	Linear Referencing System
OGC	Open Geospatial Consortium
PHP	Personal Home Page
PL/SQL	Procedural Language/Structured Query Language
PNG	Portable Network Graphics
SDO	Spatial Data option
SVG	Scalable Vector Graphics
TUDU	Traťový definiční úsek
URL	Uniform Resource Locator
XML	Extensible Markkup Language

Seznam obrázků

Seznam tabulek

Úvod

Cílem této práce je navrhnout, implementovat a porovnat možné druhy vizualizace multidimenzionálních dat pomocí technologie Oracle Spatial a programovatelného nástroje MapViewer.

První část bakalářské práce popisuje možnosti uchování multidimenzionálních dat v databázi Oracle a základní operace s těmito daty. Pozornost je primárně zaměřena na technologii Oracle Spatial, prostorové datové typy a metadata. Čtenář se zde dozví, jakou edici databáze zvolit, které databázové schéma spravuje vícerozměrná data, nebo které užitečné funkce a operátory použít pro efektivní práci s vícerozměrnými daty. Každá popisovaná problematika je vždy doplněna o praktické příklady.

V druhé teoretické části je popsána technologie MapViewer, která umožňuje vizualizovat prostorová data. Jsou podrobně popsány možnosti využití MapViewer API, princip jejich činnosti a jsou uvedeny jejich výhody a nevýhody v návaznosti na praktickou část bakalářské práce. Na konci teoretické části je popsán i samostatný program Oracle Map Builder, sloužící pro vytvoření a správu mapovacích metadat uložených v databázi.

Praktická část je zaměřena na návrh, implementaci a otestování demonstračních aplikací, které jsou schopny vizualizovat prostorová data. V některých aplikacích je popsána a implementována animace, zobrazující pohyb kolejových vozidel po modelu železniční sítě. Tato část zahrnuje i dokumentaci a porovnání výhod a nevýhod jednotlivých vzorových implementací. Na konci praktické části je uvedeno doporučení pro možné využití dané implementace s přihlédnutím na její charakter.

1 Databáze Oracle

Oracle je databázový server, který strukturovaně spravuje data. Poskytuje uživatelům ukládání a zpracování jejich dat. Jeho rychlost, spolehlivost a bezpečnost se dá využít v mnoha směrech. Jedním z těchto použití může být podniková aplikace, využití datových skladů nebo nejrůznějších datových analýz.

Tato databáze umožňuje uchovávat vícerozměrné data a nadále s nimi efektivně a rychle pracovat. První pokusy začlenění této technologie do databáze Oracle se datuje k verzi Oracle 4. Zlom přišel ale až s verzí Oracle 7 a prostorovým indexováním pomocí SDO „Spatial Data Option“. Od verze Oracle 8 se objevilo rozšíření Oracle Spatial a změna indexování obsahu využívající R-strom index. [17]

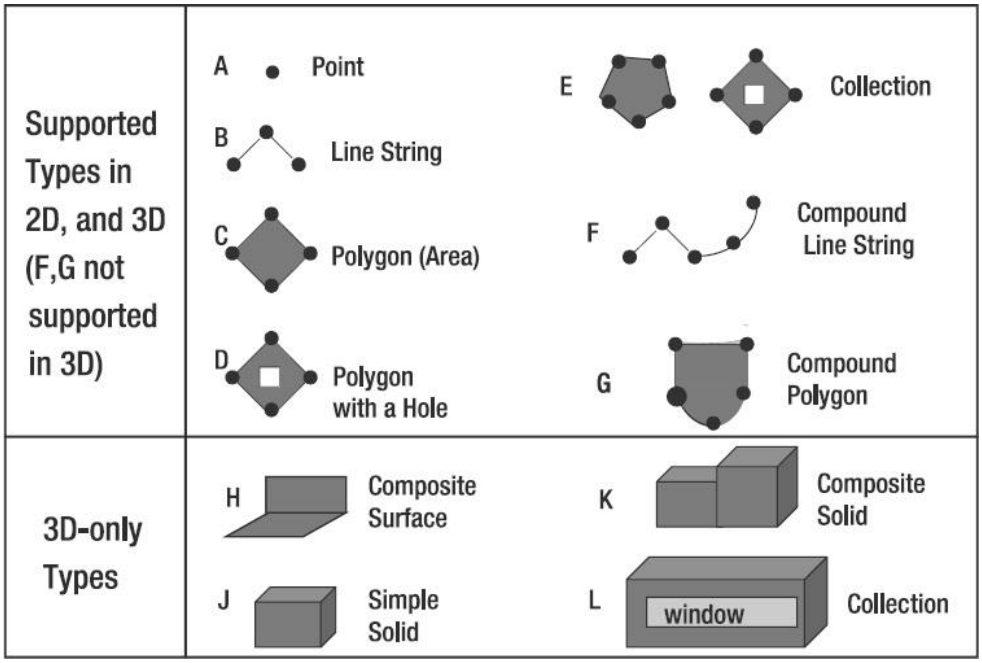
Pro moji potřebu jsem zvolil verzi 11g v edici Enterprise, která poskytuje nadstavbu Spatial a spoustu dalších užitečných funkcí.

1.1 Instalace databáze

Instalace databáze Oracle se skládá z několika kroků. Je dobré si dělat poznámky o tom, kam daný software instalujete, jaký zvolíte globální název databáze a v neposlední řadě klíčové heslo. Tyto údaje budete později potřebovat, abyste se dostali do administrace databáze a mohli ji spravovat se vším všudy. Pokud uděláte někde chybu a heslo zapomenete nebo špatně nainstalujete databázi, není poté lehké uvést ji do správného stavu nebo dokonce kompletně odinstalovat z počítače.

1.2 Oracle Spatial

Oracle Spatial je nadstavba databáze, která podporuje prostorové dvojrozměrné a třírozměrné datové typy, které jsou obohaceny funkcemi pro efektivní práci s nimi. Mezi tyto typy patří např. bod, linie bodů, polygon nebo polygon s dírou, jak zobrazuje Obrázek 1: Typy prostorových dat.



Obrázek 1: Typy prostorových dat.
Zdroj: [12]

Ukládáním dat, jejich syntaxí a podporou se zabývá schéma MDSYS, které zároveň poskytuje funkce, procedury, operátory a indexový mechanismus pro efektivní práci s prostorovými daty. Pro moji bakalářskou práci bylo ale využito pouze operátoru SDO_NN, který využívá prostorového indexu k identifikování nejbližšího souseda podle jeho geometrického popisu. [12]

Výhody Oracle Spatial

- Eliminuje potřebu dvojí architektury, protože všechna data mohou být uložena stejným způsobem. Jednotné úložiště dat znamená, že všechny typy dat (text, mapy a multimédia) jsou uloženy společně místo toho, aby byl každý typ uložen zvlášť.
- Používá SQL, standardní jazyk pro přístup k relační databázi, což odstraňuje potřebu pro konkrétní jazyky, které by zpracovávali prostorová data.
- Definuje datový typ SDO_GEOMETRY, který je v podstatě ekvivalentní k prostorovým typům v OGC[1] a SQL/MM[2] standardům.
- Implementuje SQL / MM formáty pro určení prostorových dat. Z toho vyplývá, že jakékoli řešení, které se hlásí k SQL / MM specifikaci lze snadně ukládat data do Oracle Spatial a naopak, bez potřeby konvertorů třetích stran. (opravit překlad jestli je správně).

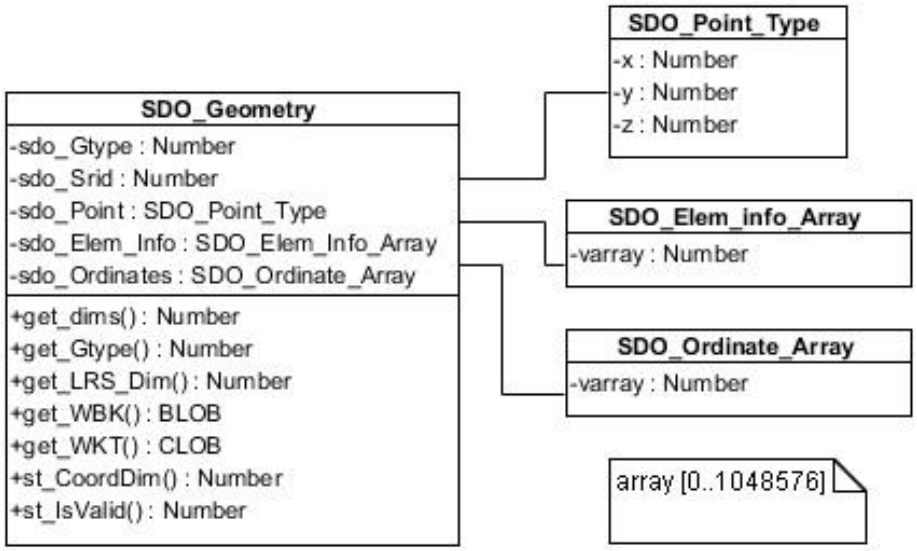
[1] OGC (Open Geospatial Consortium) – dříve také Open GIS Consortium, je mezinárodní organizace, zabývající se implementací standardů pro geoprostorová data a služby.
[2] SQL/MM (Multimedia) – Standard pro práci s prostorovými daty v SQL.

- Jde de facto o standard pro ukládání a přístup k datům v Oracle a je plně podporován předními světovými prodejci geo-prostorových dat, nástrojů a aplikací, včetně NAVTEQ, Tele Atlas, Digital Globe, Autodesk, ESRI, Skyline a mnoho dalších.
- Poskytuje škálovatelnost, integritu, bezpečnost, obnovitelnost a pokročilé uživatelské funkce pro zpracování prostorových dat, které jsou standardem v databázích Oracle.
- Odstraňuje potřebu pro oddělení organizací zachovat prostorovou datovou infrastrukturu (hardware, software, podporu atd.) a eliminuje potřebu zvláštních nástrojů a dovedností pro práci s prostorovými daty.
- Prostřednictvím aplikačního serveru umožňuje téměř jakékoliv aplikaci využít dostupností prostorových informací a inteligence, snížení nákladů a komplexnosti prostorových aplikací.
- Zavádí výkonné vizualizace prostorových dat, což eliminuje potřebu spoléhat se na samostatné vizualizační nástroje pro mnoho aplikací. [12]

1.2.1 Prostorové datové typy

SDO_GEOMETRY

Prostorová data jsou charakterizována především geometrickým popisem objektu, který je uložen pomocí objektového datového typu SDO_GEOMETRY. Tento datový typ uchovává informace o geometrickém tvaru a umístění pro každý řádek tabulky. Obrázek 2: Struktura objektu SDO_GEOMETRY. zobrazuje UML diagram struktury objektu SDO_GEOMETRY.



Obrázek 2: Struktura objektu SDO_GEOMETRY.
Zdroj: [2], upraveno

SDO_GTYPE

Tento atribut specifikuje typ tvaru geometrie (bod, polygon, kolekce apod.). I když atribut říká, který geometrický tvar reprezentuje, nespecifikuje jeho skutečnou polohu. Číslo se skládá ze 4 číslic ve tvaru DLTT, kde:

- D – identifikuje počet dimenzí a může nabývat hodnot 2-4 podle počtu dimenzí,
- L – označuje lineární referenční měřítko dimenze pro tří-dimenzionální lineární referenční systém (LRS), když dimenze (3 nebo 4) obsahuje hodnotu měřítka,
- TT – specifikuje geometrický typ a nabývá hodnot 0-9. [5] [16]

Dvou-dimenzionální bod by poté měl hodnotu 2001, kde 2 označuje druhou dimenzi, 0 říká, že se nejedná o LRS geometrii, a 01 značí, že se jedná o bod.

Tabulka 1: SDO_GTYPE hodnoty
Zdroj: [2], upraveno

Označení	Význam
0	Neznámý typ geometrie
1	Bod (Point)
2	Linie (Line) nebo křivka (Curve)
3	Polygon (Polygon)
4	Kolekce (Collection)
5	Sada bodů (Multipoint)
6	Sada linií (Multiline)
7	Multipolygon, areál (Multipolygon)
8	Solid (pro 3D)
9	Multisolid (pro 3D)

SDO_SRID

SDO_SRID atribut určuje ID prostorového souřadnicového systému se kterým je geometrie asociována. Může nabývat hodnoty *null*, ale geometrie nebude mít přiřazený souřadnicový systém.

SDO_POINT

Definuje objekt typu SDO_POINT_TYPE, který má atributy X, Y, Z. Všechny tři atributy jsou typu NUMBER. Hodnoty X a Y většinou označuj souřadnice zeměpisné šířky a délky. Pokud jsou zadána dvojdimenzionální data, musí byt hodnoty SDO_ELEM_INFO a SDO_ORDINATES obě prázdné. [16]

SDO_ELEM_INFO

Atribut říká, jak mají být interpretovány souřadnice geometrického popisu prvků, uložených v atributu SDO_ORDINATES.

SDO_ORDINATES

Zde jsou uloženy souřadnice bodů, které vytváří geometrický popis hranice prostorového prvku.

1.2.2 Práce s vícerozměrnými daty

Vytvoření funkčních prostorových dat lze provést v několika krocích. Je důležité si předem rozmyslet, jestli budou data dvojrozměrná nebo třírozměrná a k čemu budou použita. Některé atributy totiž mohou nabývat hodnoty *null* a nemusí být vždy potřeba. Vytvoření prostorových dat se skládá ze 4 operací:

- vytvoření tabulky pro uchování prostorových dat,
- vytvoření metadat v tabulce, v níž jsou uložena prostorová data,
- vytvoření prostorového indexu nad sloupcem s geometrickými údaji,
- doplnění tabulky o data.

Vytvoření tabulky pro prostorová data

Vytvoření tabulky uchovávající prostorová data je snadná záležitost. Nejdůležitější je vložit sloupec, který bude charakterizovat prostorová data pomocí datového typu SDO_GEOMETRY. Níže vidíte jednoduchý SQL příkaz pro vytvoření prostorové tabulky s obyčejnými atributy a jedním atributem typu SDO_GEOMETRY.

```
CREATE TABLE node (
id NUMBER PRIMARY KEY,
route VARCHAR2(11 CHAR),
tudu CHAR(6 CHAR) NOT NULL,
km NUMBER(5,1) NOT NULL,
kv CHAR(1 CHAR) NOT NULL,
gps SDO_GEOMETRY NOT NULL,
);
```

Obrázek 3: Vytvoření tabulky pro prostorová data
Zdroj: Vlastní zpracování

Vytvoření metadat

K vytvoření metadat slouží pohled USER_SDO_GEOM_METADATA do kterého se vloží informace o každém sloupci typu SDO_GEOMETRY. Nastaví se zde také hranice každé dimenze, které reprezentují zeměpisnou šířku a délku, a atribut SDO_SRID. Tento atribut slouží k identifikaci souřadnicového systému, se kterým je asociována geometrie. Pokud je atribut nastaven na NULL, není asociace provedena. Kompletní seznam různých souřadnicových systému obsahuje tabulka CS_SRS ve schématu MDSYS. [12]

Na Obrázek 4: Vytvoření metadat je uveden příklad aktualizování prostorových metadat do tabulky NODE nad sloupcem GPS. Parametry atributu SDO_DIM_ELEMENT jsou dimenze, spodní a horní hranice dimenze a tolerance. Souřadnice X je ohraničena hodnotou 13 až 17, souřadnice Y hodnotou 48 až 52. Obě souřadnice mají tolerancí nastavenou na 0.001. Poslední vloženou hodnotou je SRID souřadnicového systému.

```
INSERT INTO user_sdo_geom_metadata
VALUES('NODE', 'GPS',
  SDO_DIM_ARRAY(
    SDO_DIM_ELEMENT('X', 13, 17, 0.001),
    SDO_DIM_ELEMENT('Y', 48, 52, 0.001)
  ),
8307);
```

Obrázek 4: Vytvoření metadat
Zdroj: Vlastní zpracování

Vytvoření prostorového indexu

Je potřeba vytvořit prostorový index nad sloupcem GPS, kde jsou uchovány informace o geometrických útvarech. Na Obrázek 5: Vytvoření prostorového indexu je uveden příklad vytvoření prostorového indexu na tabulce NODE nad sloupcem GPS.

```
CREATE INDEX node_spatial_idx
ON node(gps)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

Obrázek 5: Vytvoření prostorového indexu
Zdroj: Vlastní zpracování

Doplnění tabulky o data

Tabulka je připravena k plnohodnotnému použití a stačí ji pouze naplnit daty podle předpřipravených pravidel. Při vkládání vícerozměrných dat musíme použít konstruktor SDO_GEOMETRY, který definuje bod. Na Obrázek 6: je zobrazen příklad vložení dat do tabulky NODE, kde je použit konstruktor SDO_GEOMETRY na vložení dvojrozměrného bodu v souřadnicovém systému s SRID číslem 8307.

```
INSERT INTO node (id, route, tudu, km, kv, gps)
VALUES (1, 'B421021  1', '1061C1', 17, 'V',
  SDO_GEOMETRY(2001,8307,
    SDO_POINT_TYPE(15.3623454168108,50.4262515279982,null),
  null,null)
);
```

Obrázek 6: Doplnění tabulky o data
Zdroj: Vlastní zpracování

2 Možnost vizualizace multidimenzionálních dat

Vše kolem nás se nachází v prostoru. Samotný prostor se může skládat z několika dimenzí. Polohu určitého objektu můžeme určit ve smyslu geometrickém. V takovém případě se poloha bodu určí za pomocí souřadnic X, Y, Z. Pokud ale budeme potřebovat určit polohu bodu v určitém čase například z důvodu pozorování chování tohoto bodu, přibude nám do prostoru další dimenze v podobě času. Takových parametrů může být ale mnoho. Tyto parametry vstupují do celého systému jako další dimenze, které nám pomáhají určit polohu objektu nebo jevu v prostoru. V této kapitole se však budu zabývat pouze základním určením polohy bodu ve smyslu geometrických souřadnic X, Y.

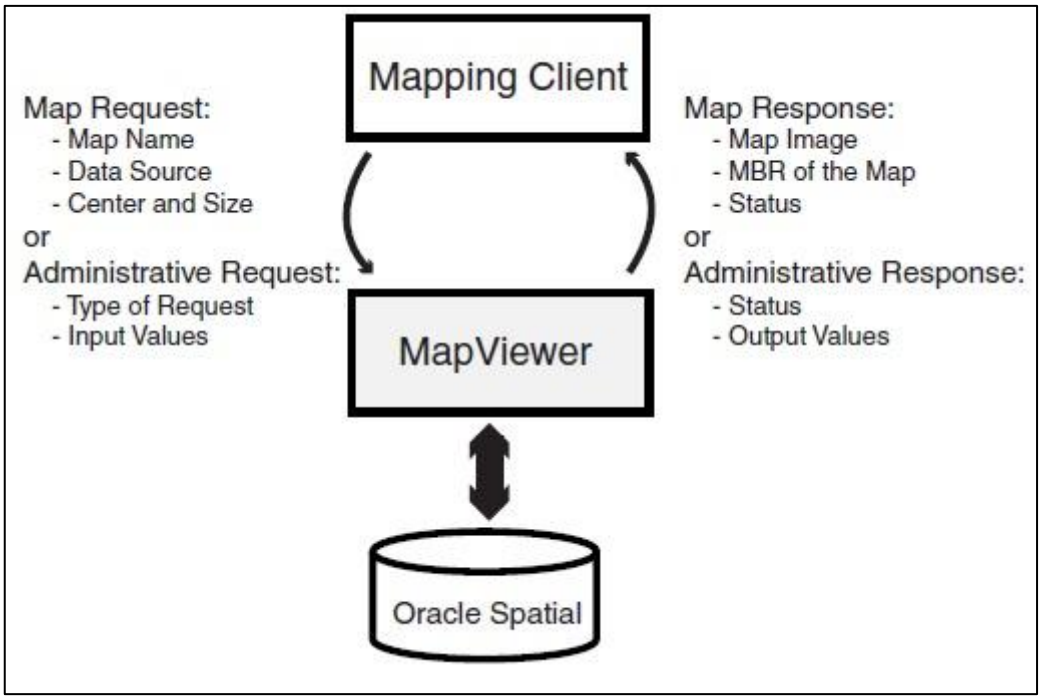
Klasickým příkladem vizualizace prostorových dat ve světě kartografie a geografie je papírová mapa. Objekty na mapě jsou znázorněny pouze dvojrozměrně, ale je možné využít různých technik k vytvoření dojmu o více rozměrech. Mezi tyto techniky patří například lineární perspektiva, stereoskopy, anaglyfy nebo stínování. [4]

V předchozí části (1.2.2) jsem popsal práci s prostorovými daty. Pro ty, kteří neprogramují nebo se tímto tématem nikdy nezbývali, jsou tyto data pouze jedničky a nuly, a proto je potřeba zobrazit data v takové podobě, aby každý přesně viděl a pochopil, co reprezentují. Pokud jde o zobrazení dat dvojrozměrného prostoru ve světě kartografie lze tyto data zobrazit pomocí mapy. K tomu slouží programovatelný nástroj od firmy Oracle, MapViewer.

2.1 Oracle MapViewer

Oracle MapViewer je komponenta na straně serveru, která vytváří mapy čtením pohledů a tabulek z databáze a vrací je klientské aplikaci v odpovídajícím formátu. Každá mapa je tvořena specifickými vrstvami nebo tématy, které reprezentují logické seskupení geografických prostorových prvků, jako jsou například silnice, železnice, řeky atd. Tyto grafické objekty jsou vykreslovány speciálními styly. [12]

Oracle MapViewer funguje na modelu žádost/odpověď, kde klient žádá server o informace o mapě jako je datový zdroj, střed mapy, název mapy apod. a server vrací obrázek nebo URL obrázku a hranice vytvořené mapy. [16]



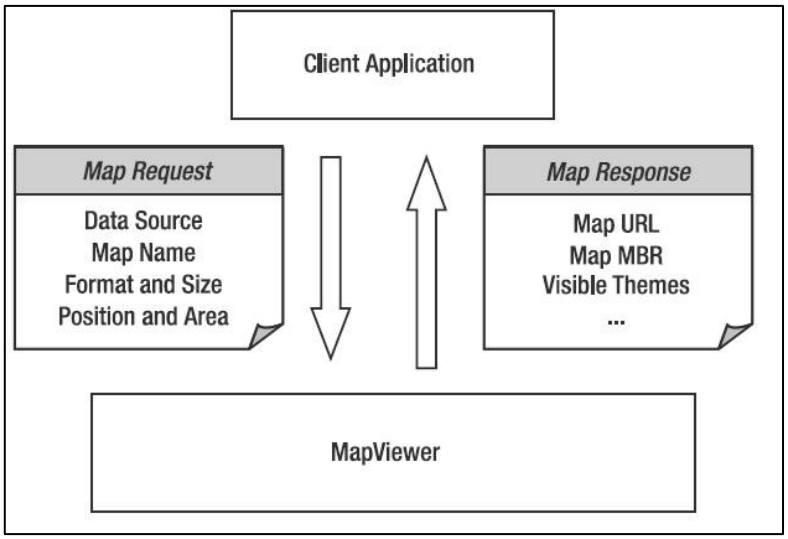
Obrázek 7: Princip činnosti Oracle MapViewer
Zdroj: [16]

Oracle MapViewer se skládá z:

- Mapově vykreslovacího enginu nazývaného SDOVIS, který běží v Oracle aplikačním serveru. Vykreslovací engine je vystaven jako servlet, který zpracovává žádosti zaslané klientskými aplikacemi, načte správné informace z prostorových tabulek a vytvoří mapy v různých grafických formátech (GIF, PNG, JPEG nebo SVG), které pak vrátí klientovi, jež o ně žádal. Kromě jádra mapovacího servletu, MapViewer server poskytuje mapový server pro vyrovnávací paměť a FOI[3] server.
- Série rozhraní pro API, které umožňují přistupovat k vlastnostem MapVieweru z různých vývojových prostředí aplikací. Tato rozhraní API obsahují XML, Java, PL/SQL a JavaScript (Ajax) rozhraní. Java API obsahuje také JSP značky pro usnadnění začlenění map v JSP.
- Grafický nástroj Map Builder – program, který pomáhá se správou mapových definic uložených v databázi.
- Mapové definice – jsou uloženy v databázi. Na tomto místě je popsáno, které tabulky používat, jak by měla být mapa zobrazena (barvy, tloušťka čáry a písmo) atd. [16] [18]

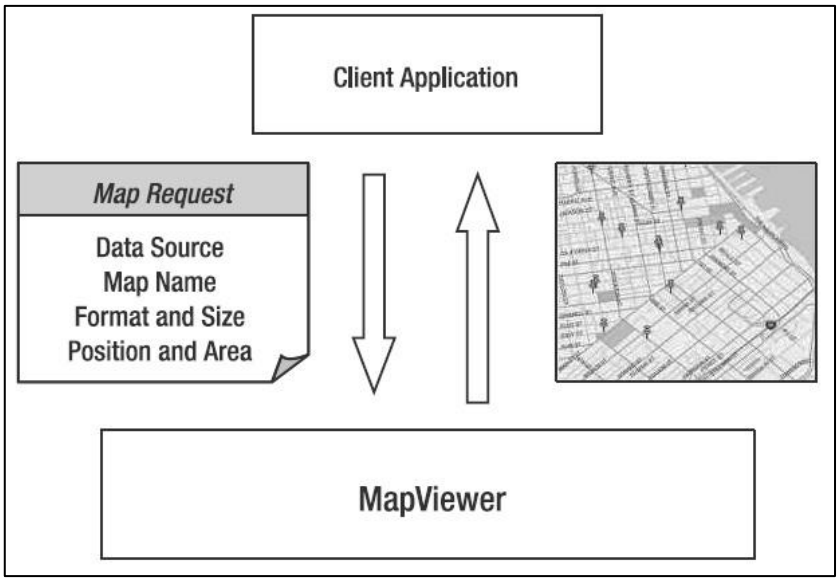
[3] FOI jsou prvky na mapě, které představují prostorová data. Tyto data obsahují obrázek mapy a příslušná data o objektu, které jsou poté poslána klientovi, a uživatel s nimi může nadále pracovat. FOI jsou dynamické a představují obsah databáze nebo aplikace v reálném čase. Společně se statickými mapami umožňují vytvářet webové mapovací aplikace.

Klientské aplikace komunikuji s MapViewer servletem přes HTTP pomocí modelu žádost/odpověď, jak je zobrazeno na Obrázek 8: MapViewer žádost/odpověď tok. Tyto požadavky a odpovědi jsou kódovány v jazyce XML. Java klienti mohou použít Java API, která se stará o konstrukci a zasílání XML žádostí, stejně jako čtení a parsování XML odpovědí. [12]



Obrázek 8: MapViewer žádost/odpověď tok
Zdroj: [12]

Možnou alternativou pro MapViewer je přímé zaslání obrázku mapy klientským aplikacím namísto URL generované mapy, viz Obrázek 9: MapViewer tok se zasláním obrázku.



Obrázek 9: MapViewer tok se zasláním obrázku
Zdroj: [12]

2.2 Použití API

V úvodu předchozí části (2.1) bylo vysvětleno, že MapViewer je komponenta na straně serveru. Aplikace pak komunikuje se serverem pomocí modelu požadavek/odpověď. MapViewer poskytuje několik API k provedení těchto výměn. Výběr API má pak významný dopad na způsob, jak aplikaci nadále vyvíjet ale i na způsob interakce aplikace s uživatelem. Tento výběr bychom mohli rozdělit do dvou hlavních kategorií.

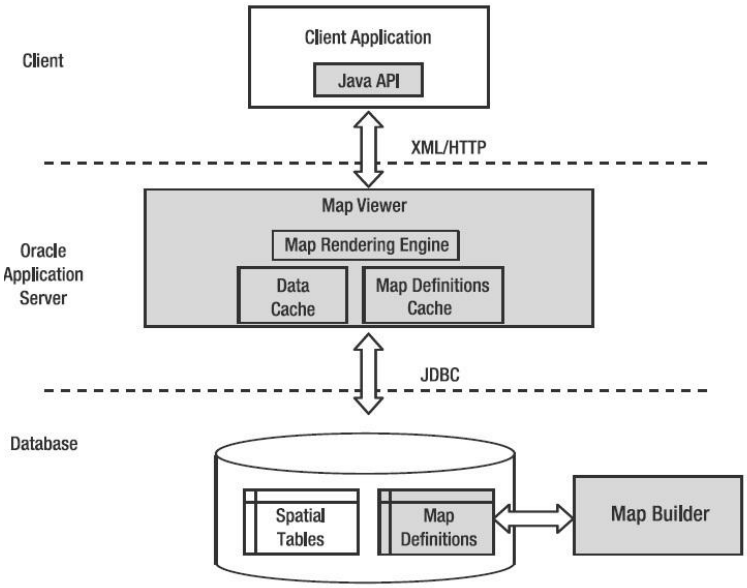
První možností je použít jeden z klasických rozhraní API, která MapViewer nabídl už od první verze. Tato rozhraní umožňují používat MapViewer z téměř libovolného vývojového prostředí a použít libovolný programovací jazyk buď pomocí přímé výměny XML nebo prostřednictvím klientských knihoven pro Javu, JSP nebo PL/SQL.

Druhá možnost je použít relativně nový JavaScript/Ajax rozhraní s názvem Oracle Maps. Tato možnost umožňuje vytvářet vysoce dynamické rozhraní pomocí jazyka JavaScript z kteréhokoliv prohlížeče. [12]

2.2.1 XML, Java, JSP a PL/SQL

Nejnižší úroveň interakce je přímá výměna XML zpráv. Aplikaci stačí pouze sestavit XML kód, odeslat ho na server přes HTTP protokol a poté dekomponovat odpověď ze serveru. Tato technika, díky své jednoduchosti, je dostupná ze všech vývojových prostředí, které jsou schopné odeslat HTTP požadavek a manipulovat s XML (Java, NET, C #, ale také Perl, PHP a Python). Obrázek 10: Architektura jádra MapVieweru zobrazuje proces požadavku:

1. Klientská aplikace vytvoří žádost pro webovou službu k získání mapy. Žádost obsahuje název zdroje dat ke čtení (data v databázi), informace o mapě, formát mapy (GIF, PNG nebo JPEG), její velikost v pixelech a oblast mapy, která má být vykreslena.
2. Klient zavolá vykreslovací servlet MapVieweru přes protokol HTTP a pošle mu XML požadavek.
3. Servlet analyzuje žádost, přečte mapový definice z databáze, z tabulek s prostorovými daty, a na výstupu vygeneruje mapu v příslušném formátu. Mapa je uložena jako soubor. Mapová definice je uložena v paměti cache na aplikačním serveru a individuálně zde může být uložena část nebo všechny prostorové data, které servlet čte.
4. Server vytvoří XML odpověď, která obsahuje URL adresu vygenerovaného souboru a vrátí ji klientovi.
5. Klient pak rozebere XML a vytáhne URL obrázku, který pak předá klientskému prohlížeči.



Obrázek 10: Architektura jádra MapVieweru
Zdroj: [12]

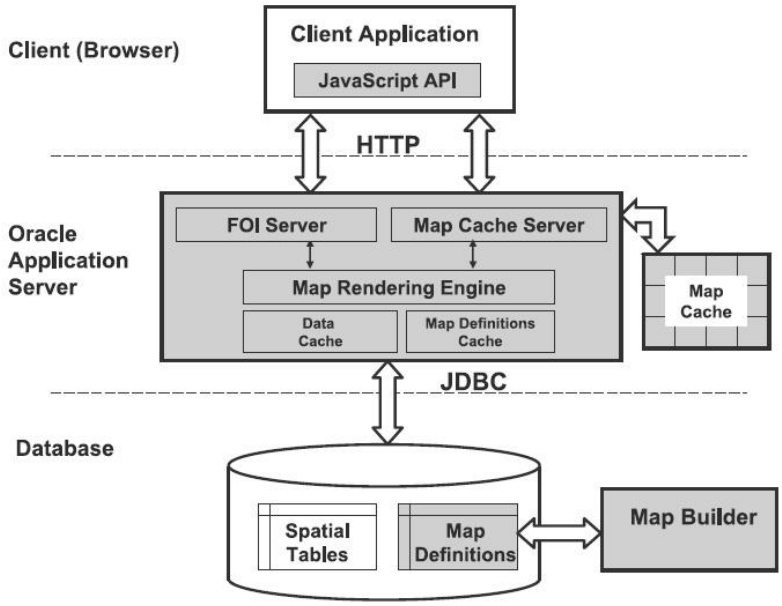
Bez ohledu na to, které API je použito (XML, Java, JSP nebo PL/SQL), bude MapViewer server provádět všechny výše zmíněné operace. Přečte data, které potřebuje, přečte mapové definice, použije vaše definované styly a vygeneruje kompletně mapu od nuly, dokonce i kdyby se žádný parametr nezměnil, a vy jste jenom požádali o aktualizaci mapy. [12]

2.2.2 JavaScript a Ajax: Oracle Maps

Techologie Oracle Maps nabízí odlišný přístup a jiné zpracování. Aplikace již neřeší hlavní MapViewer server (mapově vykeslovací engine). Raději spolupracuje se dvěma jinými:

- Map Cache server a
- FOI server.

Tyto dva servery společně představují mapově vykreslovací engine, který je zobrazen na Obrázek 11: Architektura Oracle Maps.



Obrázek 11: Architektura Oracle Maps
Zdroj: [12]

Interakce mezi klientem (webový prohlížeč) a MapViewerem existuje pomocí paradigmatu Ajaxu, který využívá schopnost webového prohlížeče poslat HTTP požadavek na server a získat odpověď plně asynchronním způsobem, transparentním pro uživatele. Tento jednoduchý, ale účinný postup umožňuje vývoj velmi dynamické aplikace, která nemusí spoléhat na klasický cyklus webového prohlížeče, jež vyžaduje úplné obnovení stránky, i kdyby se změnila hodnota nějaké malé ikony. Pomocí tohoto paradigmatu mohou aplikace měnit obsah svých webových stránek bez nutnosti jejich znovunačítání. Nebudete mít možnost interakce s mapovým serverem nebo ukládáním na FOI server ale budete komunikovat s MapViewerem pomocí Ajaxu. Mnoho z těchto interakcí nastane automaticky v důsledku nějaké akce od uživatele, jako je například zvětšování, posouvání, kliknutí na dynamické objekty zobrazující určité informace aj. [12]

Klientská aplikace komunikuje s MapViewerem následovně:

- Komunikace s Map Cache serverem:
    - Knihovna JS klienta určuje, které části mapy potřebuje ke splnění požadavků aktuální mapy v požadovaném měřítku. Pošle více požadavků na Map Cache servlet, který vrátí požadované části.
    - Map Cache servlet poté ověří svoji cache pamět. Pokud najde části mapy, pošle je zpět klientovi. Pokud jeden nebo více částí chybí, vyvolá mapově vykreslovací servlet a pošle mu instrukce ke generování chybějících částí mapy, které potom uloží v paměti cache, než je vrátí klientovi.

- Komunikace s FOI serverem:
    - JS klient pošle požadavek FOI servletu, požádá ho o načtení FOI dynamických objektů a zobrazení na aktuální mapě. Klient může zadat název tématu nebo poskytnout SQL příkaz typu select.
    - FOI servlet poté provede prostorový dotaz k získání informací z databázových tabulek. Také generuje grafické vykreslování každého prostorového objektu, který najde a vrátí grafiku společně s veškerými atributy požadujícímu klientovi.
- JS klient nakonec sestaví všechny části mapy a překreslí je s FOI objekty předtím, než je zobrazí koncovému uživateli. [12]

2.3 Výběr API

Mapovací knihovna Ajaxu umožňuje automaticky veškeré výměny mezi aplikací a MapViewer servey (Map Cache and FOI server). Rovněž umožňuje řídit veškerou interakci s uživatelem, jako je přibližování, posouvání, identifikace atd. To vše pomůže vytvořit aplikaci snazší na psaní a výsledkem je i kratší kód. Aplikace napsané v JavaScriptu, poběží uvnitř prohlížeče uživatele. [12]

Další API (Java a XML) se obvykle používají ve vývojových prostředích na straně serveru, jako jsou servlety nebo JSP stránky, ale můžete využít i Java API v appletu. XML API lze použít kdekoliv, kde je vývojové prostředí schopno vytvořit a získat XML a použít HTTP požadavek.

PL/SQL AIP lze použít pouze pomocí databáze Oracle. Ve všech těchto prostředích bude vaše aplikace zodpovědná za zpracování všech uživatelských interakcí.

Dalším důležitým rozdílem mezi Java/XML/PLSQL API a JavaScript API spočívá v přítomnosti Map Cache serveru. Vzhledem k tomu, že server generuje části mapy na úrovni přiblížení, které se zadávají při definování mapy, je JavaScript aplikace omezena na zobrazení mapy pouze v těchto definovaných úrovních přiblížení. Také nemůžete měnit vykreslení základní mapy, která je uložena v paměti. [12]

Naopak jiná API mohou nabídnout plnou flexibilitu pro generování mapy s jakýmkoliv obsahem v libovolném měřítku. To samozřejmě proběhne na úkor generování celé mapy pro každý dotaz.

Pro rekapitulaci možností jednotlivých API je níže zobrazena tabulka srovnání.

Tabulka 2: Možnosti MapViewer API
Zdroj: [12], vlastní zpracování

API	Vývojové prostředí	Měřítko	Znovupoužití mapy	Interakce s uživatelem
Java	Každé Java prostředí: Servlety, JSP stránky, applety, tlustý klient	Jakýkoli	Nikdy	Závislé na programu
XML	Každé prostředí: Java, .NET atd.	Jakýkoli	Nikdy	Závislé na programu
PL/SQL	Databáze	Jakýkoli	Nikdy	Závislé na programu
JavaScript	Každý prohlížeč	Pevné/fixní	Vždy	Automatická

2.4 Oracle Map Builder

Oracle Map Builder je samostatná aplikace, která umožňuje vytvořit spravovat mapovací metadata (informace o stylech, tématech a základních mapách), které jsou uloženy v databázi. Například použití tohoto nástroje k vytvoření nebo úpravě definice stylu. Kromě zpracování metadat, nástroj poskytuje rozhraní pro náhled metadat (například vidět, jak se styl čáry zobrazí na mapě) a také prostorové informace. [16]

Kdykoli je to možné, měl by se použít Oracle Map Builder k vytváření, upravování a mazání informací o tématech, stylech a mapách místo přímé změny metadat pohledů MapViewer. Pro jakékoliv úpravy provedené mimo Oracle Map Builder, jako například SQL příkazy, by se mělo aktualizovat připojení k databázi v Oracle Map Builderu pro zobrazení aktuálních položek.

Oracle Map Builder obsahuje nejen vlastní kód, ale také základní renderovací třídy MapVieweru, JDBC ovladače a podporu knihoven pro XML. [12]

3 Návrh a implementace aplikace

Hlavním cílem praktické části BP bylo navrhnout a implementovat aplikaci, která by porovnala různé přístupy vizualizace prostorových dat. Použil jsem dvojrozměrné body, reprezentující GPS souřadnice hektometrovníků/staničení, které ve své zjednodušené podobě, odrážejí infrastrukturu železniční sítě. Desetitisíce bodů bylo uloženo v databázi Oracle pomocí technologie Oracle Spatial. Aplikace by měla umožňovat základní operace s mapou, jako je přiblížení, posouvání, identifikaci objektů aj. Aby byl využit alespoň částečně potenciál aplikace, bylo potřeba implementovat jednoduché jádro diskrétní simulace pro emulaci provozu kolejových vozidel na železniční síti. Pro tyto účely bylo využito MapViewer komponenty, která by měla těmto požadavkům plně dostačovat.

V předchozí kapitole bylo vysvětleno, že MapViewer je komponenta na straně serveru, zahrnující Oracle aplikační server, a že Oracle Maps je souhrn technologií, poskytované MapViewerem, ale s odlišnou architekturou. Pro každý z těchto způsobů existuje jiné použití API. Mým úkolem bylo použít klasické MapViwer API, které používají technologie jako XML, Java, JSP a PL/SQL. Vybral jsem si 3 následující aplikace:

- desktopová Java aplikace,
- Java applet a
- Java Server Page.

Požadavky na aplikaci

Aplikace by měla umožňovat následující akce:

- vizualizace železniční sítě,
- manipulace s mapou (posun, přiblížení aj.),
- emulování pohybu vlaků po železniční trati,
- načtení průjezdu vlaku ze souboru,
- zobrazení ujeté trasy vlaku a všech potřebných informací.

Použité technologie

Všechny tři aplikace vychází z programovacího jazyka Java. Práce s nimi se ale liší a to nejen z pohledu jejich zobrazení ale i z pohledu samostatného programování a implementace jednotlivých metod. Mimo jiné byly použity technologie:

- vývojové prostředí prostředí NetBeans 7.2.1,
- databáze Oracle 11g,
- doplňková technologie Oracle Spatial,
- SQL Developer,
- Microsoft Excel,
- webový server Apache,

- aplikační server Apache Tomcat,
- Oracle MapViewer,
- Oracle Map Builder.

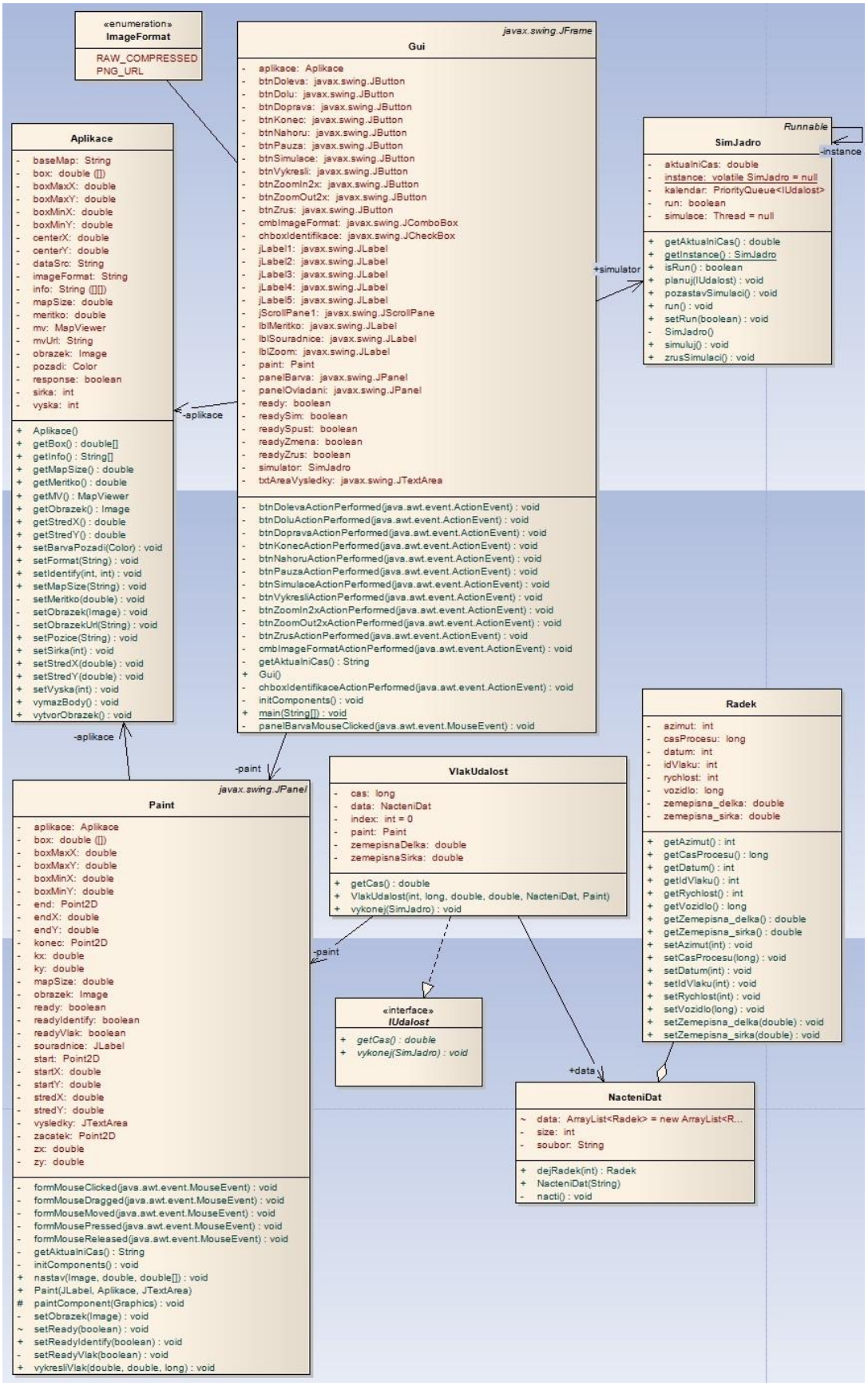
3.1 Desktopová Java aplikace

Výhoda klasické desktopové aplikace je především ve chvíli, kdy nejste připojeni k internetu. K této výhodě se přidává ještě jazyk Java, se kterým se stane aplikace snadněji přenositelná na mnoho platforem.

Aplikace se skládá ze sedmi balíčků:

- uživatelské rozhraní a vykreslovací panel (Gui),
- samostatná aplikace pro práci s komponentou MapViewer (Aplikace),
- načtení dat (Vstup),
- simulace (Simulace),
- csv soubor (Data),
- výčtového typu Enum, který představuje formát obrázku (Enum),
- pomocný balíček s ikonami (Ikony).

Balíček *Gui* obsahuje hlavní okno aplikace společně s ovládacím panelem. V balíčku se také nachází třída *Paint*, která znázorňuje plátno pro vykreslení mapy a interakci s uživatelem. Balíček *Aplikace* využívá stejnojmennou třídu k vytvoření, správě a další práci s komponentou MapViewer. V sekci *Vstup* se nachází třídy, zabývající se načítáním dat ze souboru. Pro výběr formátu výsledného obrázku je použita třída *ImageFormat* v balíčku *Enum* a grafické prostředí zkrášlují ikony z balíčku *Ikony*. V poslední části, *Simulace*, se nachází především simulační jádro.



Obrázek 12: UML diagram tříd
Zdroj: Vlastní zpracování

3.1.1 Využité návrhové vzory

Návrhové vzory představují způsob postupu, jak provádět určité problémy efektivně. Aplikace využívá ke své implementaci následující vzory:

- výčtový typ,
- jednoduchá tovární metoda,
- Jedináček (Singleton).

Výčtový typ

Výčtový typ *enum* umožňuje vytvářet vlastní datové typy, které mohou nabývat pouze určitého omezeného množství hodnot. Každé z těchto hodnot poté odpovídá právě jedna instance výčtu. V bakalářské práci je pro tyto účely použita třída *ImageFormat*, která definuje 2 formáty pro práci s obrázkem a to *PNG_URL* a *RAW_COMPRESSED*.

Jednoduchá tovární metoda

Je to návrhový vzor, který je jinou variantou volání konstruktoru, kterému se za určitých podmínek chceme vyhnout. Tento návrhový vzor je v bakalářské práci použit jako součást návrhového vzoru *Jedináček* ve třídě *SimJadro*, kde obstarává vytvoření právě jedné instance simulačního jádra.

Jedináček (Singleton)

Podstatou návrhového vzoru *Jedináček* je vytvoření pouze jedné instance, která se používá kdekoliv v aplikaci. Tímto se dá ušetřit značné množství paměti. Instance je globálně dostupná přes jméno třídy a tovární metodu *getInstance()*. Nevýhodou tohoto vzoru je, že pokud používáme vícevláknových aplikací, může nastat situace, kdy první vlákno požádá o vytvoření *Jedináčka*. Procesor poté přepne na druhé vlákno, kde ještě není *Jedináček* vytvořen a je spuštěn proces tvorby *Jedináčka*. Poté je přepnuto na první vlákno, kde byl *Jedináček* započat a je dokončen. V tu chvíli jsou „jedináčci“ dva. Tomu zabráníme nastavením atributu instance za *volatile* a synchronizováním bloku kódu, starajícího se o tvorbu *Jedináčka*, kde se znova ověří, jestli již mezitím nebyl vytvořen. Díky klíčovému slovu *volatile* je zajištěno, že všechna vlákna vidí jeden a tentýž stav. Tento vzor používá třída *SimJadro*. [14]

3.1.2 Simulace pohybu železničního vozidla

Jeden z požadavků na aplikaci bylo vytvoření jádra diskrétní simulace a následně vizualizace zkoumaných objektů v podobě animace, která by zobrazovala pohyb vlaku po trati. Bylo využito jádra diskrétní simulace, kde události nastávají v nespojitém čase, nastala-li pouze určitá událost. Následující část vysvětluje základní pojmy a rysy simulace.

Simulace

Simulaci lze definovat několika způsoby: [3]

1. Simulace je proces tvorby modelu reálného systému a provádění experimentů s tímto modelem za účelem dosažení lepšího pochopení chování studovaného systému či za účelem posouzení různých variant činnosti systému (Shannon).
2. Numerická metoda, která spočívá v experimentování s matematickými modely dynamických reálných systémů na číslicových počítačích (Naylor).
3. Simulace je technika, která nahrazuje zkoumaný dynamický systém jeho modelem s cílem získat informace o systému pomocí experimentu s modelem (Dahl).

Existují dva systémy:

- statický,
- dynamický.

Statický systém je popsán pouze statickou charakteristikou vyjadřující závislost výstupu na hodnotách vstupu.

Oproti tomu dynamický systém je systém, který není jednoznačně určen pouze jeho vstupy, ale závisí také na čase. Systém existuje v okamžicích, kdy se vykonává nějaká akce a mimo ní systém vlastně neexistuje. Aktivita představuje jednotku simulace, která zobrazuje konkrétní činnost v simulovaném systému a platí pro ni, že má časové trvání a mění stav systému. [13]

Simulaci pak můžeme obecně rozdělit na:

- diskrétní,
- spojitou.

Spojitá simulace obsahuje výhradně spojité aktivity, které mohou měnit stav systému v průběhu celé doby svého trvání. Pro diskrétní simulaci platí, že obsahuje pouze diskrétní aktivity, které jsou charakterizovány tím, že mohou měnit stav systému pouze v okamžiku svého ukončení. Událostí pak označujeme okamžik ukončení diskrétní aktivity a následnou změnu stavu systému. [10]

Metoda plánování události

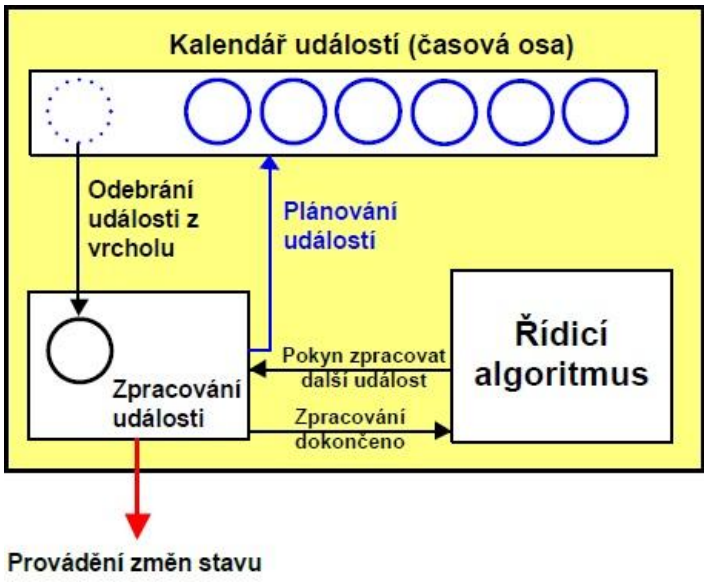
Metoda plánování událostí se uplatňuje především při realizaci diskrétní simulace a je založena na plánování výskytů událostí do budoucnosti. Diskrétní simulace je charakteristická tím, že její doba trvání a ukončení je předem známa. Doba trvání pak odpovídá časovému zpoždění výskytu koncové události aktivity. Informace o naplánované události se musí uchovávat až do okamžiku jejího výskytu. Informace je uložena kalendáři událostí, který je postaven na datové struktuře prioritní fronta.

Kalendář událostí (někdy též časová osa) obsahuje události zapouzdřující atribut *časové razítko*, které reprezentuje okamžik konce událostí v budoucnu, a operace *Vykonej* a *Plánuj*. Metoda *Vykonej* provede příslušné změny stavu v simulovaném systému v okamžiku výskytu události a metoda *Plánuj* vloží události do kalendáře, čímž naplánuje další události na jistý čas výskytu v budoucnosti.

Běh simulačního programu, který využívá diskrétní simulaci, je řízen algoritmem, který naplánuje události, jejich vybírání z kalendáře a provádí příslušné změny stavu systému.

KROK	Činnost	Vykonána za podmínek
0	Inicializace simulačního času t_S ($t_S = 0$)	
1	Ukončení běhu simulačního programu	Kalendář neobsahuje události nebo je vyčerpán čas pro běh simulace
2	Výběr události z „vrcholu“ kalendáře (s nejmenší hodnotou /t_U / plánovaného času výskytu)	
3	Aktualizace simulačního času ($t_S = t_U$)	
4	Výkon akce spojené s výskytem události (akce provádí stavové změny a případné plánování dalších událostí)	
5	Návrat na **KROK 1**	

Obrázek 13: Jednotlivé kroky plánování událostí
Zdroj: [10]



Obrázek 14: Algoritmus metody plánování událostí
Zdroj: [10]

Všechny výše zmíněné postupy jsou obsaženy v samostatném modulu, nazývaném *simulační jádro*. Tento modul umožňuje monitorovat simulaci a aktivně do ní zasahovat.

3.1.3 Jednotlivé třídy desktopové aplikace

Třída Gui

Tato třída se především stará o zobrazení všech komponent a reaguje na události vytvořené interakcí uživatele a ovládacího panelu. Třída také zabezpečuje chyby, které by mohly vzniknout neopatrnou manipulací s ovládacím panelem, pomocí proměnných typu *boolean*.

Třída Paint

Třída slouží pro vykreslování mapy generované komponentou MapViewer. Vykresluje se zde nejen mapa železniční sítě ale i případný simulující pohyb vlaku po dráze. Na kliknutí, s pomocí zaškrtnutí políčka identifikace v ovládacím panelu, se zavolá metoda *setIdentify()* s parametry GPS souřadnic X a Y, a vykreslí na mapě bod, kde uživatel kliknul společně s nejbližším železničním bodem. Zobrazí také informace o tomto bodu, jako je název traťového úseku, na kterém bod leží, identifikační číslo, GPS souřadnice aj. Třída poskytuje také zobrazení aktuálních souřadnic, kam ukazuje myš a posun mapy pomocí táhnutí myši na mapě.

Třída Aplikace

Cílem této třídy je vygenerovat obrázek poskytovaný MapViewerem při jeho běhu. MapViewer poskytuje jako výchozí formát buď samotný obrázek, nebo URL adresu obrázku, a proto je třída opatřena dvěma metodami pro nastavení správného výstupního formátu. Ve většině případů se používá spíše URL adresa, se kterou se pak pracuje uvnitř HTML kódu. MapViewer poskytuje 3 formáty tohoto typu, které jsou GIF, PNG a JPEG. Pro tenké klienty je pak využito komprimovaného formátu RAW.

Třída využívá nejrůznějších metod komponenty MapViewer pro nastavení a manipulaci výsledného obrázku. Některé metody lze řešit dvojím způsobem a to například přibližování nebo posun mapy do světových stran. Pro výpočet nové pozice mapy lze použít buď vlastních proměnných, se kterými se nadále v aplikaci pracuje, ale není potřeba aby byli drženy v relaci. Nebo se může použít metod *pan(), zoomIn()* a *zoomOut(),* pro které je potřeba, aby byli drženi v relaci a při každém načtení stránky obnoveny.

Hlavní metoda vytvorObrazek() se pak stará o následující operace:

- inicializace mapy pokud tak již nebylo učiněno,
- nastavení formátu obrázku mapy (PNG_URL, RAW_COMPRESSED),
- nastavení pozadí mapy,
- nastavení rozměrů mapy,
- nastavení středu mapy a úrovně přiblížení.

Pokud budou všechna nastavení správná, pošle se pomocí metody *run()* požadavek na nastavení parametrů. Poté metoda uloží do proměnných hranice mapy, zobrazí na ovládacím panelu aktuální měřítko a získá správný formát obrázku mapy.

Třída poskytuje ještě metodu *setIdentify(X, Y)*, která pomocí metody *identify()* zjistí údaje o nejbližším bodu k místu, kde se na mapě kliknulo, a přidá oba body do vykreslení mapy s patřičným stylem. Následně zobrazí i informace o nejbližším bodu, které jsou uloženy v databázi.

Třída NacteniDat

Zde se realizuje načtení souboru do bufferu a jeho následné rozebrání řádků do příslušných proměnných. Na závěr načtení všech dat z řádku je řádek přidán do proměnné *data*, která je polem s proměnlivou délkou (*ArrayList*) a obsahuje každý řádek souboru.

Třída VlakUdalost

Abstraktní třída, která implementuje metody z rozhraní *IUdalost*. Od této třídy rovněž se rovněž nedá vytvořit instance. Obsahuje metodu pro přístup k privátnímu atributu *čas* a metodu *vykonej()*, kterou využívá simulační jádro k provedení akcí pro simulaci vlaku na železniční trati a k naplánování nové události.

Třída SimJadro

Simulační jádro obsahuje především privátní atribut *aktualniCas* a kalendář událostí do kterého se může přidat kterýkoliv proces implementující abstraktní třídu *VlakUdalost*. Tato třída používá návrhový vzor *Jedináček* a implementuje rozhraní *Runnable* s předepsanou metodou *run(), ve* které se volá důležitá metoda *vykonej()*.

Simulace v metodě *run ()* odebírá procesy s nejnižším časovým razítkem z kalendáře událostí a provádí akce definované aktuálním procesem. Poté se vlákno uspí metodou *sleep()* na několik milisekund. Mezi další metody této třídy patří metody *zrusSimulaci()* a *pozastavSimulaci()*, které umožňují simulaci pozastavit nebo úplně zrušit.

3.2 Java applet

Java applet je aplikace napsaná v jazyce Java, která je spuštěna uživatelem při návštěvě konkrétní webové stránky. Applet je spuštěn pomocí prostředí Java Virtual Machine, které je instalováno společně s moderními prohlížeči. Internet Explorer mezi tyto prohlížeče nepatří a virtuální stroj se zde musí instalovat zvlášť. Struktura appletu je tvořena třídou *java.applet.Applet*, která je potomkem třídy *java.awt.Panel*. Umožňuje appletu provádět grafický výstup a zachycovat události z klávesnice a myši. Applet má

životní cyklus, který závisí na prohlížeči, který během své činnosti může volat jednotlivé metody appletu: [6] [7] [11] [15]

- public void init() – volá se při inicializaci,
- public void start() – při spuštění,
- public void paint(java.qwt.Graphics g) – při překreslován,
- public void stop() – při zastavení,
- public void destroy() – při ukončení.

Java applet se skládá z identických balíčků a tříd jako desktopová Java aplikace, ale některé metody jsou zde upraveny, aby mohl běžet na serveru. Použil jsem Apache server se základním nastavením a projekt vložil do složky htdocs, což je kořenový adresář, v jehož podadresářích pak server hledá soubory a složky.

První problém, na který jsem narazil, bylo správné zobrazení appletu ve webovém prohlížeči. Applet se vůbec nechtěl zobrazit a přitom se stránka jevila na první pohled v pořádku. Za pomoci kontrolního panelu Javy, který jsem si musel ručně nastavit, aby se zobrazoval při načítání Java aplikací na webu, jsem zjistil, že byla chyba v načtení ikon. Kompilátor přeložil pouze Java třídy a na zbytek zapomněl. Musel jsem ručně vložit všechny zbývající složky a soubory do adresáře *classes*, kde jsou uloženy již zmíněné zkompilované Java třídy, ze kterých applet čerpá.

V tu chvíli se applet již správně zobrazil, ale neposkytoval vykreslení mapy. Kompilátor opět „zapomněl“ přidat do výsledného HTML souboru archiv s cestou k MapViewer klientovi. Archiv a cestu ke klientovi jsem tudíž musel nastavit ručně.

Další nevýhodou appletu jsou jeho rozměry a zobrazení v prohlížeči. Šířka a výška appletu se musí ručně nastavit v HTML souboru na požadované rozměry, které jsou po dobu zobrazení neměnné. Možné řešení je použít JavaScript, ale i tehdy není 100% zaručeno, že se bude vše zobrazovat správně. Stejně tak pokud applet přesahuje velikost okna v prohlížeči, nemusí se vždy zobrazit správně. Pokud budete chtít udělat jakékoliv změny programu, musíte ho opět zkompilovat, aby vše fungovalo, a s tím se přepíše i HTML soubor.

Aplikace si načítá data z csv souboru na serveru pomocí metody *nacti()* ve třídě *NacteniDat*, kde je v parametru metody předána URL adresa souboru namísto adresy souboru pomocí řetězce typu *String*. Metoda následně vytvoří objekt na URL spojení a získá z něj vstupní proud. Potom jsem v metodě vytvořil lokální proměnnou *celySoubor* typu *BufferedReader* na vstupním proudu a čtu z ní. Applet totiž nemá přístup nikam než do serverových adresářů. Nemůže nahrávat knihovny, navazovat síťové spojení na jiný než domovský server, spouštět programy na domovském serveru nebo třeba číst některé systémové proměnné.

3.3 Java Server Page

JSP (Java Server Page) jsou nadstavbou nad Java servlety. Servlety slouží k nízkoúrovňové obsluze HTTP protokolu. Umí zpracovávat HTTP dotazy a generovat odpovědi. JSP je poté webově scriptovací jazyk, který generuje dynamický obsah. Jeho použití je vhodné hlavně tam, kde většinu stránky tvoří statický obsah. Oproti klasickému servletu je snazší v něm napsat zdrojový kód naopak je ale pomalejší. [1] [8] [9]

Pro další práci s JSP je nutné pochopit, jak vůbec fungují. Popis činnosti JSP stránek je následující:

1. Webový prohlížeč pošle HTTP požadavek webovému serveru.
2. Server rozezná HTTP požadavek, který je určen pro JSP stránku a předá ho JSP enginu. To se provádí pomocí URL nebo JSP stránky, které mají koncovku *.jsp* namísto *.html*.
3. JSP engine načte tuto stránku z disku a převede ji na obsah servletu. Tento postup je velmi jednoduchý a statický obsah stránky se uloží v *out.println()* příkazech. JSP elementy jsou poté přeloženy do Java kódu, které implementuje dynamické chování stránky.
4. JSP engine zkompiluje servlet na spustitelnou třídu a předá původní požadavek servlet enginu.
5. Servlet engine poté načte zkompilované třídy a vykoná je. Během tohoto procesu servlet vytváří výstup v HTML formátu, který následně předá webovému serveru jako HTTP odpověď.
6. Pokud se jedná o statický HTML obsah, webový server ho pošle jako HTTP odpověď prohlížeči.
7. Nakonec prohlížeč udržuje dynamicky generovanou HTML stránku uvnitř HTTP odpovědi přesně, jako kdyby šlo o statickou stránku.

V rámci MapViewer API můžeme rozdělit JSP na dva možné způsoby, jak aplikaci vytvořit. První způsob je použití klasického Java API, které je použito i v desktopové aplikaci a Java appletu, nebo vytvořit dynamický obsah stránky pomocí JSP tagů (značek). Aby bylo možné pracovat s JSP stránkami je nutné přitom použít aplikační server.

Aplikační server je přitom mnohdy označován jako webový server, což není pravda. Webové servery jsou jedny ze základních kamenů aplikačních serverů, mohou se odkazovat na určitý hardware nebo software, který pomáhá dodávat obsah přístupný z internetu. Aplikační server je softwarový framework[horni index a vysvetlit], který poskytuje prostředí, ve kterém lze spustit aplikace nehledě na to, co aplikace jsou nebo dělají. Pro moje použití mi posloužil aplikační server Apache Tomcat verze 7.0.39.

Aplikace s využitím Java API

Architektura JSP se liší od appletu a desktopové aplikace minimálně už tím, že využívá struktury HTML souboru, obsahujícího statický a dynamický obsah. Aplikace je složena ze třech hlavních souborů a jednoho archivu.

Index.jsp

Hlavní JSP soubor, který se stará o inicializaci MapViewer klienta. Uživatel zde může sám určit velikost mapy, její pozici nebo úroveň přiblížení. Stránka poté pošle informace další JSP stránce, která je rozluští a nadále s nimi pracuje.

Operace.jsp

Tato stránka se stará o chod aplikace. Na úvodu přijme data ze stránky *Index.jsp* pomocí metody getParameter a uloží data, zadaná uživatelem, do proměnných se kterými nadále pracuje. Pokud v relaci neexistuje instance MapVieweru, tak ji stránka vytvoří s parametry, které zadal uživatel nebo je získá z výchozích proměnných, nastaví relaci na *true* a předá jí instanci MapVieweru. Jako formát nastaví *PNG_URL* a přidá na mapu i měřítko se dvěma různými ukazateli (metrické a imperiální jednotky). Pokud je to první zobrazení stránka se nadále o nic jiného nestará. V opačném případě zjišťuje, jestli v relaci existuje parametr *userClick*, který indikuje interakci uživatele se stránkou. MapViewer umožňuje pouze 4 způsoby této interakce, a to vycentrování mapy (jinou možností je pak přímý posun na určitou světovou stranu), přiblížení, oddálení nebo identifikaci objektu. Podle toho, kterou akci uživatel zvolil, aplikace vykoná příslušné změny. Pokud se jedná o identifikaci objektu, aplikace zjistí z bodu, kam uživatel kliknul, nejbližšího souseda a vykreslí oba body na mapu. Červený bod představuje nejbližšího souseda a modrý bod místo kliknutí uživatele. O nejbližším sousedu poté vypíše i příslušné údaje z databáze, jako je id bodu, traťový úsek, na kterém se nachází nebo GPS souřadnice bodu. Aplikace následně vymaže všechny body z mapy, aby se příště nezobrazovali. Na konci souboru importuje stránku *Paint.jsp*, která se stará o vykreslení mapy.

Paint.jsp

Jak již bylo řečeno, tato stránka se stará o vykreslení mapy a příslušných bodů a informací o mapě. Vše je realizováno přes formulář, který vyhodnocuje stránka *Operace.jsp* a metodu *POST*, která přenáší data skrytě na serveru namísto parametrů v URL adrese, jak to dělá metoda *GET*. Obrázek je vykreslen pomocí tagu <*input*>, který je typu *IMAGE*. URL obrázku získá z instance MapVieweru společně s rozměry obrázku. Tento tag má název *userClick*, podle kterého pak stránka Operace.jsp zjistí, co se s mapou dělo během interakce s uživatelem. Dále jsou zde vykresleny zaškrtávací políčka s názvem *action*, které představují pro uživatele výběr prováděné akce s mapou. Nedílnou součástí je pak vykreslení pomocných informací, jako je střed mapy, měřítko mapy, úroveň přiblížení nebo informace o objektech na mapě.

Emulace provozu vlaků na železniční trati

Každá aplikace by měla obsahovat simulační jádro pro emulaci provozu kolejových vozidel na železniční trati. V předchozích aplikacích se tento úkol zdál jako snadný, zde tomu tak není. V appletu a desktopové aplikaci byl průjezd KV (kolejových vozidel) vykreslen na plátno. Simulace probíhala cyklicky, dokud jí uživatel nepozastavil nebo nezrušil. V JSP stránkách žádné plátno na vykreslení není a navíc se tato metoda musí volat uvnitř Java kódu. Což není nejlehčí na naprogramování a stále je to nešikovné řešení.

V teoretické části bylo vysvětleno, že MapViewer se skládá ze dvou různých API. Klasické Java API, které používají desktopové aplikace, applety, JSP stránky, XML, PLS/SQL nebo JS Ajax pomocí Oracle Maps. Nevýhodou u JSP stránek je, že pokud chcete provést sebemenší změnu obsahu, musí se stránka znova načíst (provést refresh). A to v případě simulace není nejlepší řešení. Simulace zatěžuje aplikační server zbytečným vykreslením vlaku (protože se musí generovat celá stránka znovu) a vlak by se nemusel zobrazit vždy ve specifickém intervalu. Možným řešením je použít zmiňovanou technologii Oracle Maps a JS Ajax, kde vykreslení mapy probíhá automaticky, aniž se musela znova načíst celá stránka. Provedení tohoto řešení by ale zabralo spoustu času, který byl vynaložen na vytvoření 3 aplikací používajících Java API. Musely by se změnit tabulky v databázi, vytvořit nový základ mapy prostřednictvím webového rozhraní MapVieweru a napsat JS kód, který by vykreslení mapy prováděl automaticky. Proto zde simulace nebyla implementována.

Aplikace s využitím JSP tagů

Aplikace s využitím JSP tagů vypadá podobně jako s využitím Java API. Rozdíl je v tom, že nastavení parametrů obrázku, generovaného MapViewerem, není psáno pomocí jazyka Java ale pomocí tagů (značek). Vypadají skoro jako klasické HTML tagy. Abyste mohli používat tyto tagy, je potřeba je definovat. Definice tagů je uložena v souboru *mvtaglib.tld*, který se nachází v instalační složce MapVieweru v adresáři *web/WEB-INF/*. Stačí zahrnout do JSP stránky ukazatel na knihovnu a zvolit si prefix, se kterým budete knihovnu používat, jak můžete vidět na Obrázek 15: Vložení knihovny tagů a prefixu.

```
<%@ taglib uri="/WEB-INF/mvtaglib.tld" prefix="mv" %>
<%@ page session="true" %>
<%@ page import="oracle.lbs.mapclient.MapViewer" %>
```

Obrázek 15: Vložení knihovny tagů a prefixu
Zdroj: Vlastní zpracování

Na prvním řádku je zobrazeno natažení knihovny s definicí tagů a prefixem *mv*. Druhý řádek pak nastavuje relaci na hodnotu *true* a třetí řádek vloží do aplikace MapViewer klienta.

Velikou nevýhodou použití tagů je počet metod, které se dají vyvolat. Je definováno pouze 10 metod pro nastavení klienta, které dostačují pouze na základní operace s mapou, jako je její zobrazení, posun mapy, identifikace objektu aj. Chybí například nastavení stylů, identifikace a vykreslení více jak jednoho bodu, práce s tématy aj. Zbytek se musí dopsat pomocí Java API. Protože základ aplikace tvoří JSP stránka, simulace zde nebyla implementována ze stejného důvodu.

3.4 Porovnání vzorových implementací

Ve své BP práci jsem použil čtyři různé možnosti vizualizace prostorových dat. Po celou dobu, co jsem s nimi pracoval, jsem u každé našel určité výhody a nevýhody, které popisují následující čtyři tabulky.

Desktopová Java aplikace

Tabulka 3: Výhody a nevýhody desktpové Java aplikace
Zdroj: Vlastní zpracování

Výhody	Nevýhody
Použití v offline režimu	Nemožnost prohlédnout si aplikaci, pokud ji nemáte na svém disku.
Snadná přenositelnost na jiné systémy	Rychlost, resp. její pomalost
Tváří jako plnohodnotná samostatná aplikace	Horší režie paměti než jiné programovací jazyky

Java applet

Tabulka 4: Výhody a nevýhody Java appletu
Zdroj: Vlastní zpracování

Výhody	Nevýhody
Použití na webu	Pevné rozměry na webu
Klasické výhody jazyku Java	Lze pustit pouze v prohlížeči, obsahující JVM
	Nízké práva na serveru

Java Server Page s využitím Java API

Tabulka 5: Výhody a nevýhody JSP s využitím Java API
Zdroj: Vlastní zpracování

Výhody	Nevýhody
Oddělení statického a dynamického obsahu	Je potřeba aplikační server
Použití na webu	Generování celé stránky při sebemenší úpravě
Jednodušší psaní kódu než klasický servlet a předešlé 2 aplikace	
Napsaný kód je kratší	

Využití JSP tagů v rámci JSP

Tabulka 6: Výhody a nevýhody JSP s využitím JSP tagů
Zdroj: Vlastní zpracování

Výhody	Nevýhody
Kód je ještě kratší a snadnější	Omezená počet metod
Vhodné pro velmi jednoduché aplikace	Všude, kde je potřeba, se musí psát tag s příslušným prefixem

Shrnutí

Jak je vidět z výše uvedených tabulek, každá implementace má svoje silné i slabé stránky. Pokud bych měl všechny čtyři aplikace shrnout a provést doporučení využitelnosti, nebude moje rozhodnutí snadné.

Desktopová aplikace je nejsnadnější na naprogramování a pro použití aplikace pouze na jednom počítači je tato varianta nejvíce vyhovující. Výhodou je určitě i to, že aplikace je napsána v jednom programovacím jazyce, se kterým si plně vystačí. Na druhou stranu, pro sdílení s dalšími uživateli je aplikace nevhodná. Tato implementace se tedy hodí pro použití malého počtu uživatelů.

Java applet je program, který pro svůj běh potřebuje prohlížeč využívající JVM a je spuštěn pomocí HTML stránky. Použití je určitě vhodné pro zobrazení aplikace pomocí webového rozhraní, ale applet má značně omezené práva na serveru a pro některé operace, jako například zápis souborů na straně klienta (prohlížeče) nebo spuštění programů na domovském serveru, je nevhodný.

Výhoda JSP technologie je viditelná na první pohled. Umožňuje zobrazit aplikaci skrze webové rozhraní a přitom ještě pomáhá generovat dynamický obsah stránky. JSP technologie spojuje HTML statický kód s dynamickým obsahem stránky, napsaném pomocí jazyka Java. Implementace je vhodná pro jednoduché webové aplikace bez možnosti použití jakýchkoliv animací, při kterých by se měnil byť sebemenší obsah stránky.

Závěr

Všechny cíle bakalářské práce byly splněny v plném rozsahu. V teoretické části byla popsána technologie Oracle Spatial pro uchovávání multidimenzionálních dat včetně možností jak tyto data uložit, práce s nimi a jejich vizualizace. Dále byla popsána technologie Oracle MapViewer společně s použitím různých druhů API a jejich následným výběrem. V poslední řadě byl popsán program Oracle Map Builder, sloužící pro správu metadat v databázi Oracle.

Z hlediska použití různých druhů MapViewer API byla práce popsána podrobně z důvodu návaznosti na praktickou část a následném doporučení jejich využití. Tato část nadále rozděluje použití API do dvou způsobů a vysvětluje použití každého z nich.

V praktické části byly navrženy a popsány aplikace pro vizualizaci prostorových dat. Byly použity čtyři způsoby vizualizace a ve všech aplikacích byly implementovány základní požadavky. Na aplikace byly aplikovány teoretické poznatky ze studia multidimenzionálních dat a byly použity pokročilé programovací techniky, návrhové vzory aj. Bylo také využito poměrně mnoho technologií pracujících na odlišných principech. Každá aplikace je schopna vykreslit infrastrukturu zjednodušeného modelu železniční sítě, dokáže komunikovat s uživatelem pomocí akcí jako je posun mapy nebo identifikace objektu a dokáže zobrazit příslušné informace o objektu.

Nad rámec zadání bylo v desktopové aplikaci a appletu implementováno jádro diskrétní simulace a animace zobrazující pohyb KV v rámci modelu železniční sítě. Existují zde navíc další tlačítka, poskytující změnu barvy pozadí nebo formátu obrázku mapy, která je v dalších aplikacích bezpředmětná.

Na závěr praktické části byly provedeny porovnání jednotlivých aplikací. Ukázalo se, že JSP aplikace nejsou nejlepším řešením pro rozsáhlé aplikace a stránky s častým měnícím se obsahem. Naopak desktopová aplikace se ukázala jako dobré řešení z důvodu jednoduchosti implementace a funkčnosti, ale pouze na samostatném počítači bez možnosti dostupnosti přes webové rozhraní. Java applet byl rozšířen o možnost dostupnosti přes webové rozhraní, ale je omezen právy na serveru.

Možným rozšířením nebo alternativou aplikací je použití Oracle Maps a JS Ajaxu pro tvorbu dynamických a rozsáhlých map, implementace rozsáhlejších dat k tvorbě statistik nebo zahrnutí do aplikace trojrozměrných dat pro vizualizaci některých objektů.

Literatura

[1] **BRANICKÝ, Marek**. JavaServer Pages pro všechny. *Interval.cz* [online]. 6. 8. 2002 [cit. 2013-08-07]. Dostupné z: http://interval.cz/clanky/javaserver-pages-pro-vsechny/.

[2] **Datové typy SDO Geometry**. [online]. Vysoká škola báňská: Ostrava [cit. 2013-08-07]. Dostupné z: http://gis.vsb.cz/wikivyuka/index.php/Datov%C3%A9_typy_SDO_Geometry.

[3] **DORDA, Michal**. Úvod do modelování a simulace systémů [online]. Ostrava [cit. 2013-08-07]. Skriptum. Dostupné z: http://homel.vsb.cz/~dor028/Aplikace_2.pdf. Vysoká škola báňská.

[4] **DUŠEK, Radek a Jakub MIŘIJOVSKÝ**. Vizualizace prostorových dat: Chaos v dimenzích [online]. Česká kartografická společnost, 2003 [cit. 2013-08-07]. 2009: 3, 114. Dostupné z: http://geography.cz/sbornik/wp-content/uploads/2010/01/g09-3-2dusek.pdf.

[5] **JANEČKA, Karel**. Oracle Spatial [online]. Plzeň [cit. 2013-08-07]. Dostupné z: https://portal.zcu.cz/wps/PA_Courseware/DownloadDokumentu?id=59810. Přednáška. Západočeská univerzita v Plzni.

[6] **Java applet**. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 29. 7. 2013 [cit. 2013-08-07]. Dostupné z: http://en.wikipedia.org/wiki/Java_applet.

[7] **Java Applets**. The Java Tutorials [online]. [cit. 2013-08-07]. Dostupné z: http://docs.oracle.com/javase/tutorial/deployment/applet/.

[8] **Java Server Pages**. [online]. Brno, 23. 4. 2013 [cit. 2013-08-07]. Dostupné z: https://kore.fi.muni.cz/wiki/index.php/Java_Server_Pages. Masarykova Univerzita.

[9] **Java Server Pages.** [online]. Praha [cit. 2013-08-07]. Dostupné z: http://atrey.karlin.mff.cuni.cz/~bim/pub/jsp/referat/referat.html. Univerzita Karlova.

[10] **KAVIČKA, Antonín**. Simulace: Pardubice, 2010 [cit. 2013-08-07]. Elektronické sylaby přednášek k předmětu Modelování a simulace. Univerzita Pardubice.

[11] **KOTALA, Zdeněk a Petr TOMAN**. Dioné: Applet [online]. [cit. 2013-08-07]. Dostupné z: http://v1.dione.zcu.cz/java/sbornik/17.html.

[12] **KOTHURI, Ravi, Albert GODFRIND a Euro BEINAT**. Pro Oracle Spatial for Oracle database 11g. New York, NY: Distributed to the book trade worldwideby Springer-Verlag New York, c2007, 787 p. ISBN 1-59059-899-7.

[13] **KŘÍŽIVÝ, Ivan a Evžen KINDLER**. Simulace a modelování [online]. Ostrava, 2001 [cit. 2013-08-07]. UČEBNÍ TEXTY OSTRAVSKÉ UNIVERZITY. Dostupné z: http://prf.osu.cz/kip/dokumenty/Msm.pdf. Ostravská univerzita.

[14] **MERTA, Jan**. Optimalizace vybraných vyhledávacích operací v rámci multidimenzionálních dat nad databází Oracle Spatial [online]. Pardubice, 2013 [cit. 2013-08-07]. Dostupné z: http://hdl.handle.net/10195/52241. Bakalářská práce. Univerzita Pardubice. Vedoucí práce Jan Fikejz.

[15] **MORKES, David**. Java Applet krok za krokem. *Interval.cz* [online]. 27. 9. 2002 [cit. 2013-08-07]. Dostupné z: http://interval.cz/clanky/java-applet-krok-za-krokem/

[16] **MURRAY, Chuck, et al**. Oracle® Fusion Middleware: User's Guide for Oracle MapViewer 11g Release 1 (11.1.1) [online]. 2010 [cit. 2013-08-07]. Dostupné z: http://docs.oracle.com/cd/E14571_01/web.1111/e10145.pdf.

[17] **Oracle Spatial and Graph**. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 24. 5. 2013 [cit. 2013-08-07]. Dostupné z: http://en.wikipedia.org/wiki/Oracle_Spatial_and_Graph.

[18] **ŘEZANINA, Emil**. Návrh modelu železniční sítě s využitím technologie Oracle Spatial Topology and Network Data Models [online]. Pardubice, 2012 [cit. 2013-08-07]. Dostupné z: http://hdl.handle.net/10195/48661. Diplomová práce. Univerzita Pardubice. Vedoucí práce Jan Fikejz.

Příloha A – Uživatelská příručka k aplikaci

Desktopová Java aplikace a Java applet

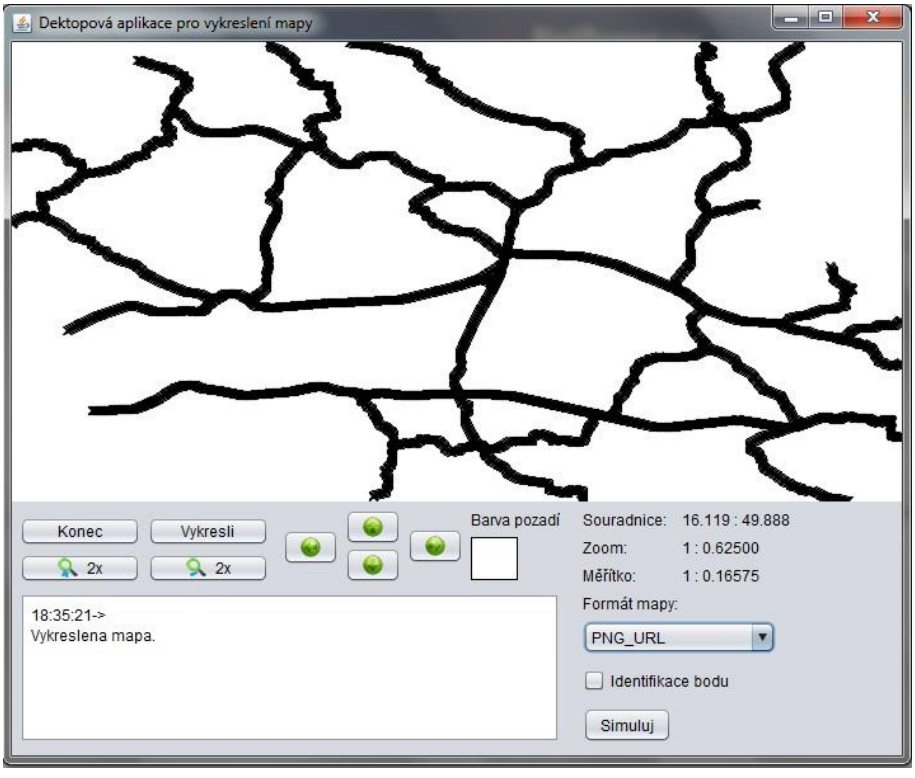
Příloha A popisuje práci s desktopovou aplikací a Java appletem. Obě aplikace obsahují téměř identický ovládací panel až na jedno tlačítko, které v destopové aplikaci umožňuje ukončení programu. Obě aplikace umí vykreslit mapu bodů, představující model železniční sítě, uloženou v databázi Oracle verze 11g. Ovládací panel umožňuje také posun mapy do světových stran buď pomocí tlačítek na panelu nebo přímo pomocí stlačení levého tlačítka myši a držení při posunu po mapě, přiblížení či oddálení nebo změnu barvy pozadí mapy.

Požadavky

Pro spuštění desktopové aplikace je nutné mít na počítači nainstalovánu Javu nejméně verze 6 a pro spuštění appletu je třeba mít v prohlížeči obsažen JVM (Internet Explorer ho nemá v základní instalaci). Applet pak potřebuje od uživatele povolit svoje spuštění, na které je dotázán.

Uživatelské rozhraní

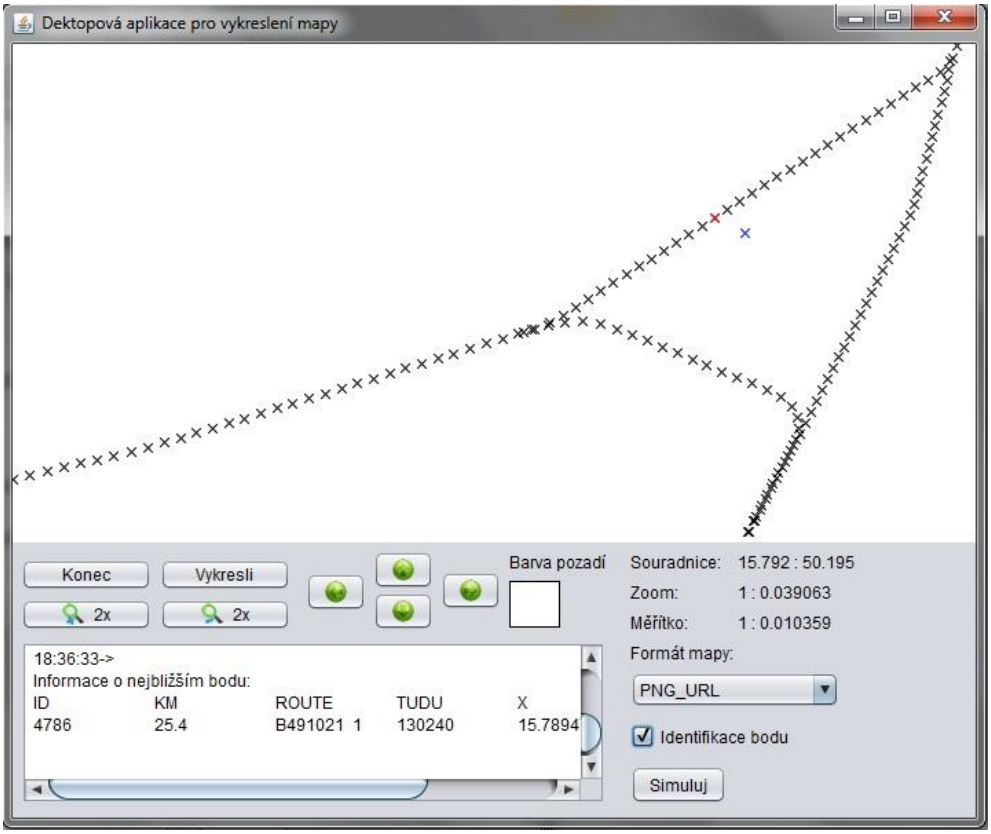
Grafické uživatelské rozhraní je rozděleno na 2 části. Horní část obsahuje kreslící plátno pro vykreslení mapy a reaguje na změny provedené myší. Spodní část obsahuje ovládací panel, který zahrnuje prvky pro ovládání vykreslení, změnu formátu mapy, oblast s výpisem informací nebo tlačítka pro animaci průjezdu vlaku po železniční trati.



Obrázek 16: Vykreslení mapy v desktopové aplikaci

Zdroj: Vlastní zpracování

Obrázek 16 zobrazuje akci identifikace bodu, kdy jsou vykresleny 2 body, z nichž jeden představuje bod, kam klikl uživatel (modrý bod) a druhý představuje nejbližšího souseda (červený bod). V informačním okně jsou pak vypsány informace o nejbližším sousedovi.



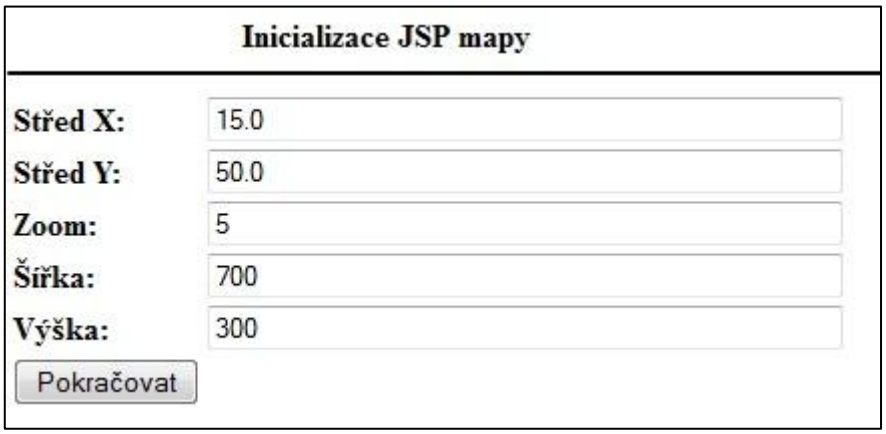
Obrázek 17: Identifikace objektu
Zdroj: Vlastní zpracování

Při průběhu simulace, která je spuštěna tlačítkem *Simuluj* je vypsán údaj o aktuální pozici a čas průjezdu. Navíc se zobrazí vedle tohoto tlačítka dvě další a to *Pauza* a *Stop*.

Obrázek 18: Výpis informací o průjezdu kolejového vozidla
Zdroj: Vlastní zpracování

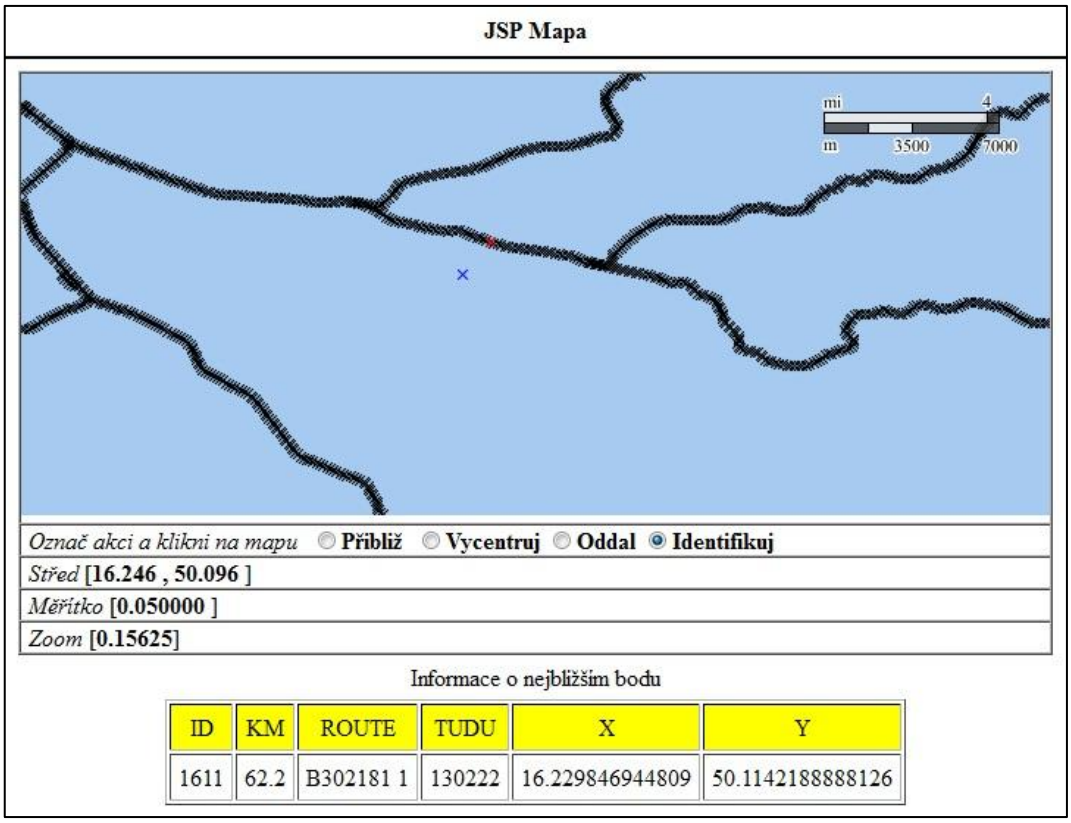
JSP aplikace s využitím Java API a JSP tagů

Obě aplikace jsou identické až na zobrazení více bodů při identifikaci objektů, kde to v aplikaci s využitím JSP tagů není možné. Aplikace se skládá ze 2 stránek, kde na první stránce si může uživatel nastavit polohu mapy, rozměry apod.



Obrázek 19: Inicializační JSP stránka
Zdroj: Vlastní zpracování

V JSP aplikaci je také přidáno měřítko, jako bývá na klasických papírových mapách. Má dvě jednotky a to imperiální (míle, stopy, aj.) a metrické (km, m, aj.).



Obrázek 20: Hlavní JSP stránka
Zdroj: Vlastní zpracování