UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Zobrazení polohy kolejového vozidla

Jan Molnár

Bakalářská práce
2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan MOLNÁR**
Osobní číslo:	**I07721**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Zobrazení polohy kolejového vozidla**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Na internetu je dostupné větší množství mapových portálů, interaktivní mapy některých z nich lze využít při tvorbě vlastních WWW aplikací.
Úkolem teoretické části bakalářské práce je popsat
- možnosti umísťování vlastních objektů do interaktivních map různých mapových portálů a
- licenční podmínky pro jejich použití.

V praktické části student s využitím vybraného mapového portálu navrhne WWW aplikaci, která zajistí zobrazení známé polohy kolejového vozidla udané GPS souřadnicemi na interaktivní mapě.
Vytvořená aplikace umožní zobrazit:
- poslední známou polohu vozidla
- ikonu pro konkrétní kolejové vozidlo (ikona by mohla být volena dle typu objektu, například řady kolejového vozidla)
- při najetí myší nad ikonu vozidla se zobrazí vyskakovací okno s detailními informacemi
- šipku, jejíž délka bude odpovídat rychlosti vozidla a směr azimutu jeho pohybu
- trajektorii pohybu kolejového vozidla v definovaném časovém období

Pro zjištění souřadnic vozidla bude využit databázový server Oracle. Řešení musí být odolné vůči SQL injection.
Důležitým aspektem je funkční i grafický návrh stránek WWW aplikace.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

***Castagnetto, J. a kol. Programujeme PHP profesionálně. Computer Press, 2004.**
***Kout, P. Praktický JavaScript. Zoner Press, 2004.**
***Riordan, R.M. Vytváříme relační databázové aplikace. Computer Press, 2001.**
***Ullman, L. PHP a MySQL - Názorný průvodce tvorbou dynamických WWW stránek. Computer Press, 2004.**
***http://maps.live.com/**
***http://maps.google.com**

Vedoucí bakalářské práce: **Ing. Zuzana Kleprlíková**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. ledna 2010**

Termín odevzdání bakalářské práce: **14. května 2010**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2010

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 10. 08. 2010

Poděkování

Na tomto místě chci poděkovat rodině za podporu při vytváření této bakalářské práce a v první řadě vedoucí mé práce slečně Ing. Zuzaně Kleprlíkové za cenné rady a celkové vedení mé bakalářské práce.

Anotace

Práce se zabývá zobrazováním polohy kolejového vozidla. K zobrazování je použita API mapa Google maps API. Je zde popsaná práce s touto mapou a postup pro zobrazování jakýkoli značek na mapě. Stručně je zde popsána i tvorba ikon.

Klíčová slova

Google maps API, JavaScript, PHP, Ajax, MySQL, azimut, rychlost, souřadnice, vlak

Title

Display position of rolling stock.

Annotation

Work deals with the viewed the position of rolling stock. Display is used API map Google maps API. As described here work with this map and procedur efor dysplaying any mark. Briefly described here as the creation of icons.

Keywords

Google maps API, JavaScript, PHP, Ajax, MySQL, azimuth, speed, coordinates, train

Obsah

Seznam zkratek

API	Application Programming Interface
GPS	Global Positioning System echnická
HTML	Hypertext Markup Language
Gmaps	Google Maps Javascript API
PHP	Hypertext Preprocesor
PNG	Portable Network Graphics
GIF	Graphics Interchange Format
JPEG	The Joint Photographics Experts Group
Ajax	Asynchronous JavaScript and XML
XML	Extensible Markup Language
JSon	JavaScript Object Notation
SQL	Structured Query Language
HTTP	Hypertext Transfer Protocol
DOM	Document Object Model

Seznam obrázků

Seznam tabulek

1 Úvod

Dnes se většina lidí dívá s úctou na internet, jako na vše znající, hladce fungující a spolehlivé průběžné monstrum, kde lze dělat takřka vše možné ba i nemožné. Většina lidí však neví, že internet je soustava počítačů, které obsahují informace a sítě, které jim dovolují k těmto informací přistupovat. Internet je tedy zdroj informací dostupných uživateli a nabízí několik standardních služeb, jejichž prostřednictvím fungují jednotlivé uživatelské aplikace. Součástí internetu je velké množství mapových portálů. Některé lze využít při vlastní tvorbě WWW aplikací. Tyto aplikace mohou být využívány pro zobrazování jak samotných map, tak i přesnějších lokalit či míst. Místa je možné označit libovolnými ikonami nebo značkami. Současně se v dnešní době tyto aplikace využívají pro zobrazení základních informací pro uživatele internetu o daném místě, v daném čase (například zobrazení počasí v určitých lokalitách, dopravní uzavírky nebo jiné komplikace na leteckých, silničních či kolejových trasách, nebo pouhé zobrazení polohy firmy či podniku).

Předkládaná práce podává základní informace o vybraných mapových portálech a možnosti umístění vlastních objektů do map těchto portálů. Zahrnuje výhody, nevýhody licenční podmínky a samozřejmě použití map jako webové aplikace. Seznámí s možností umístění základních i vlastních značek do mapy.

1.1 Cíl práce

Cílem této práce je navrhnout webovou aplikaci, která pomocí zvoleného mapového portálu zobrazí polohu kolejového vozidla. Zobrazovaná poloha musí jednoduše určit typ lokomotivy, směr pohybu a rychlost. Přijatelnou formou zobrazit ostatní informace o kolejovém vozidle. Zobrazit kolejové vozidlo na mapě v závislosti na poloze vozidla v daném čase. Aplikace by měla také řešit problematiku překrývání více zobrazovaných poloh kolejových vozidel, v závislosti na místě, čase a přehlednosti.

2 API Aplikace a Mapy

2.1 API aplikace

Mapové portály poskytující své mapy pro tvorbu vlastních webových aplikací, poskytují tyto mapy jako API aplikace, někdy také API mapy. API je zkratka pro *Application Programming Interface* neboli rozhraní pro programování aplikací. Jedná se o funkce a procedury nějaké knihovny, v našem javascriptové, které smí využívat. API tedy definuje rozhraní mezi zdrojovým kódem aplikace a knihovnou, kterou zdrojový kód využívá. Definuje způsob volání funkcí knihovny ze zdrojového kódu.

API se může rozdělit třeba podle způsobu využití. Například grafická API, mezi která patří DirectX a OpenGL. Nebo API operačních systémů POSIX pro unixové operační systémy a Win32 pro Microsoft. Pomocí Posix a Win32 mohou programy komunikovat s operačními systémy.

2.2 Mapy

Zeměpisná mapa může být označována jako model reálného světa, kontinentu, státu či jiného území. To není nic nového. Tato část textu je spíše zaměřená na to, co jsou zeměpisné souřadnice a jak se měří azimut.

K určení polohy na Zemi se dnes používají nejčastěji dva typy souřadnic. Těmi jsou zeměpisné souřadnice a GPS souřadnice. Zeměpisné souřadnice určují dvě hodnoty, jednak je to zeměpisná šířka a zeměpisná délka. Zeměpisná šířka určuje úhlovou vzdálenost od rovníku a zeměpisná délka zase od poledníku. Zeměpisná šířka i délka se měří ve stupních. Zeměpisná šířka *(ϕ)* určuje polohu od rovníku směrem k severnímu nebo jižnímu pólu. Rozsah šířky je od rovníku 0° do 90° buď severní, nebo jižní šířky. Zeměpisnou délkou *(λ)* se rozumí poloha od poledníku k východní, nebo západní polokouli. Rozsah délky je od poledníku 0° do 180° východní, nebo západní délky. Například zeměpisné souřadnice Prahy jsou *50° 05′ s. š.* a *14° 25′ v. d.*

Když se řekne GPS[1] souřadnice, tak si jsou představovány dvě hodnoty, které určují polohu podobně jako zeměpisné souřadnice, ovšem to je omyl. GPS souřadnice sami o sobě neexistují. GPS pracuje převážně s S42 souřadnicovým systémem. S42 má také, jako zeměpisné souřadnice dvě hodnoty Latitude a Longtitude, v překladu šířku a výšku. Latitude a Longtitude není vesměs nic jiného, než zeměpisná šířka a délka převedená do desetinného formátu. Převod ze zeměpisné délky nebo šířky je možný pomocí vzorce:

$$stupně + \left(\frac{vteřiny + minuty \times 60}{3600}\right)$$

Pomocí zeměpisných souřadnic se může tedy určit poloha kolejového vozidla, ale pokud bude potřeba určit směr jízdy, tak tyto souřadnice stačit nebudou. K určení směru

[1] *Global Positioning System* je vojenský polohovací družicový systém

se používá azimut. Azimut tvoří hodnota úhlu směru. Každá světová strana má přidělen svůj azimut viz tabulka 1.

Tabulka 1 - Azimut

Světová strana	Azimut
Sever	0°
Východ	90°
Jih	180°
Západ	270°

Z tabulky 1 je patrné, že úhel azimutu se počítá od severu, ve směru hodinových ručiček.

3 Vybrané mapové portály

API mapy tedy jsou knihovny, které poskytují různé společnosti. Jejich účelem je zobrazení mapy například na webových aplikacích. Společností nabízející tyto mapy je dnes velká škála. Jen na českém trhu jsou tři velké společnosti poskytující knihovny svých map. Tyto společnosti se nazývají Google maps, Mapy.cz a Ampy.cz.

3.1 Mapy.cz

Tyto mapy jsou produkované vyhledávacím portálem *Seznam.cz*. Mapy.cz spolu s Google maps patří mezi nejvyužívanější API mapy na českém trhu.

Pro použití API Mapy.cz se nejdříve musí projít řádnou registrací. Registrace je na domovských stránkách Mapy.cz[2] a skládá se z těchto úkonů:

1. Odsouhlasení podmínek (mapy lze využít pouze pro nekomerční provoz, API mapy mají omezené podklady oproti Mapy.cz, API mapy mají omezený počet zobrazení na den),
2. při registraci je nutné zadat url adresu webové stránky, kde bude mapa použita,
3. vyplnění kontaktního emailu.

Po úspěšné registraci bude zaslán na kontaktní email vygenerovaný klíč, který bude sloužit pro načtení javascriptové knihovny.

Použití map je podobné jako u všech API map. Musí se načíst javascriptová knihovna s mapou. To provede tento kód, umístěný v hlavičce html stránky, na kterých bude mapa zobrazena.

```
<script type="text/javascript"
src="http://api.mapy.cz/main?key=vas_klic&ver=3&encoding=utf-8">
</script>
```

Po té musí být vytvořen na stránce html element, který bude obsahovat mapu, například:

```
<div id="mojeMapa" style="width:640px;height:480px;border:1px solid
silver;"></div>
```

Prvek obsahující mapu nesmí být menší než 400px a větší než 2000px. Vytvořený element *mojeMapa* má velikost 640px na 480px. Poté již stačí pouze zavolat funkci, která vykreslí mapu.

```
var mapa = new SZN.MapEngine(document.getElementById('mojeMapa'));
```

Z tohoto kódu je patrné, že je vytvořena nová proměnná. Do této proměnné je uložena nová mapa, která bude vykreslena do html elementu *mojeMapa*.

[2] http://api.mapy.cz/keygen

Takto bude vypadat právě vytvořená mapa od Mapy.cz.

Obrázek 1 - Mapy.cz

Mapy.cz poskytují plno základních funkcí. Například nastavení středu mapy, zoomování, zobrazení měřítka, či jiných ovládacích prvků a také vložení vlastní ikonky do mapy. Mapy.cz nám také umožňují zobrazení více podkladů pro mapy. K dnešnímu dni je na výběr ze čtyř podkladů. Tyto podklady jsou vyjmenovány v tabulce 2.

Tabulka 2 - Pdoklady pro Mapy.cz

Název podkladu	Vysvětlení
base	Základní mapa
ophoto	Satelitní mapa
turist	Turistická mapa
army2	Historická mapa

Mapy.cz mají pokročilé nastavení, kde do samotné mapy mohou být zakresleny úsečky, vykreslování ikonek na základě aktuálního zoom. To je výhodné pro velké množství zobrazených ikon na mapě, aby mapa nebyla přeplněná ikonami. Ikony se nechají sjednotit v určitém počtu do jedné hlavní, která se při menším měřítku rozdělí na další. Slouží to pro větší přehlednost značek a rychlejší práci s mapou.

Přesnější použití a implementace jednotlivých funkcí jsou k nalezení na domovské stránce Mapy.cz, kde je i dokumentace[3] k této aplikaci.

[3] dokumentace mapy.cz - http://api.mapy.cz/static?doc=index.

3.2 Amapy.cz

Provozovatelem této webové služby je firma *Centrum Holdings s.r.o*. Tato firma mimo jiné vlastní vyhledávací portály *atlas.cz* a *centrum.cz*.

Stejně jako u Mapy.cz i zde je před použitím nejprve nutná registrace[4]. Registrace se skládá:

1. Z vyplnění kontaktní emailové adresy,
2. odsouhlasení smluvního ujednání,
3. zadání doménového jména stránky, na které má být použita API mapa.

Po těchto úkonech bude opět vygenerován klíč, pro použití map. Znění smluvních podmínek je k nalezení na stránkách registrace[4]. Nejdůležitější smluvní podmínky jsou následující:

- Uživatel vyplňující registrační formulář musí být starší 18 let nebo jej musí vyplňovat za přítomnosti svého zákonného zástupce,
- služba je zdarma,
- mapové podklady lze využívat výhradně pro svou osobní potřebu,
- mapy nelze upravovat ani vytvářet odvozená mapová díla.

Amapy nabízejí také velké množství více, či méně potřebných funkcí. Od kreslení vektorových úseček, zobrazování ikon, zoomování, až po další možnosti funkcí. I zde je několik možností podkladů. Jejich celkem sedm, tyto podklady jsou vyjmenovány v tabulce 3.

Tabulka 3 - Podklady pro Amapy

Název podkladu	Vysvětlení
a_normal_map	Základní mapa Česka
a_photo_map	Letecká mapa Česka
a_cycle_map	Mapa cyklotras Česka
a_touristic_map	Turistická mapa Česka
a_normal_word	Základní mapa Světa
a_photo_world	Letecká mapa Světa
a_hippo_map	Koňské trasy

Přidání Amapy na webové stránky se provádí stejně jak u předchozího příkladu a to načtením javascriptové knihovny.

```
<script type="text/javascript"
src="/api/api.php5?guid=VAS_GUID"></script>
```

Dalším postupem se vytvoří html element DIV s určitými rozměry.

```
<div id="mainmap" style="width: 450px; height: 450px">
```

Nyní již stačí vytvořit objekt typu AMap a zavolat načtení mapy.

[4] registrace je na stránkách: http://amapy.centrum.cz/api/registration.html

```
var mainMap = new AMap("mainMap");
            // nacteni mapy - pokud neni urcen pocatecni bod nacteni
mapy, pouzije se implicitni stred
            mainMap.loadMaps();
```

Nově vytvořená mapa bude vypadat takto:

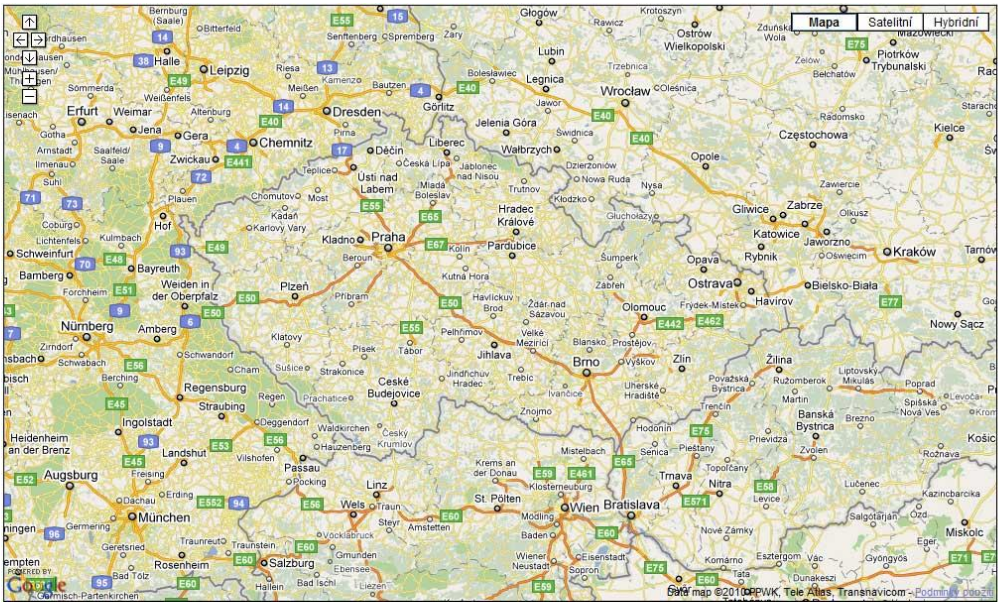
Obrázek 2 - Amapy

Amapy také poskytují dokumentaci[5] k API mapám v češtině.

3.3 Google maps API

Google Maps API jsou nejpoužívanějšími API mapami na trhu. Poskytuje je společnost Google. Tyto služby jsou poskytovány zcela zdarma, neplatí zde žádná omezení v počtu načtení za den a není potřebná ani žádná registrace. Nutností je ovšem mít účet na Google. Tím se rozumí, mít založenou emailovou schránku u této společnosti. Další výhodou je, že služba nezahrnuje žádné reklamy. V porovnání s předchozími API mapami, je tato mapa graficky nejlépe zpracována viz obrázek 3.

[5] Dokumentace k Amapy: http://amapy.centrum.cz/api-doc/docs/files/AMap-js.html



Obrázek 3 - Google Maps API

Podmínky[6] k užívání služby API od společnosti Google, se berou jako odsouhlaseny, samotným používáním mapy. Zde jsou uvedeny ty nejdůležitější:

- Společnost Google si vyhrazuje právo provádět změny těchto podmínek čas od času, tyto změny budou uloženy a zobrazeny na adrese podmínek[6],
- Pokud používáte tyto služby, prohlašujete, že máte plné oprávnění zavázat svého zaměstnavatele s těmito podmínkami,
- Google má právo svázat používanou aplikaci s limity, nejčastěji se jedná o odesílání a příjímání služby,
- souhlasíte, že jste výhradně odpovědní za používání služeb Google, pokud zjistíte zneužití pomocí vašeho hesla například, informujete o tom Google,
- Google API nesmí být použity na stránkách obsahující obsah pro dospělé, nelegální aktivity, prodej alkoholu a tabáku osobám mladší 21 let nebo na stránkách které porušují zákon,
- nesmí se odstranit, narušit jakýkoli prvek google značky,
- Google API mapy musí být přístupné ostatním uživatelům bez poplatků.

Těchto podmínek a nařízení je celá škála a před používáním by si je měl každý uživatel důkladně přečíst. Nevýhodou Google maps API je dostupnost pouze v anglické verzi. K prostudování dokumentací, podmínek, je nutností znalost anglického jazyka.

[6] Podmínky k Google maps API jsou: http://code.google.com/intl/cs/apis/maps/terms.html

3.3.1 Typy Google maps API

Google maps API mají tři základní části. Všechny tři patří do tak zvané *Google maps API family*. Tyto tři části jsou:

- Maps JavaScript API,
- Maps API for Flash,
- Google Earth API.

V této podkapitole podrobněji rozeberu pouze poslední dvě části. Maps JavaScript API budu věnovat celou následující kapitolu a použiju ji pro příklad přidání objektů do mapy.

Maps API for Flash

Google Maps API pro Flash[7] nabízí nový způsob, jak přidat interaktivní aplikace mapy Google na webové stránky, pomocí Adobe Flash ® plugin pro zobrazení dynamického obsahu mapy. Tato API existuje jako zcela nezávislá alternativa ke stávajícím JavaScript Maps API. K využití této služby je nutná znalost Flash, ActionScript® programování a objektově orientované programování.

Google Earth API

Google Earth je 3D pohled na svět. Umožňuje procházet celý svět v 3D formátu. Pomocí javascript API se může tato novinka vložit na webové stránky.

Pro využití služby 3D Google Earth API, je nutná znalost KML. KML je programovací jazyk s gramatikou XML. KML jazyk se používá pro programování Google Earth API, pro přidávání objektů, ikon a podobných modifikací.

Obrázek 4 - Google Earth API

[7] Flash je grafický vektorový program

4 Google Maps JavaScript API

Google Maps JavaScript API je nejpoužívanější službou z Google Maps API Family. Google Maps JavaScript API, dále jen Gmaps, je možné umístit na webové stránky pomocí javascriptové knihovny.

Dnešní Gmaps obsahuje tři dostupné verze. Od nejstarší první po nejnovější třetí. Já se budu zabývat druhou dostupnou verzí, značenou Google Maps JavaScript API V2. Tato verze je podle google od 19. května zastaralá, nicméně stále fungující. Vybral jsem si ji, protože se mi líbila forma zpracování dokumentace[8] k této verzi. Gmaps také poskytují plno možností a funkcí. Vybrané funkce zde podrobněji popíšu.

Pro práci s Gmaps je nutná znalost objektivního programování a samozřejmě JavaScript, pro složitější aplikace i PHP. Gmaps je podporován těmito prohlížeči:

- Microsoft Internet Explorer (IE) 7.0 a novější (pro systém Windows),
- Firefox 3.6 a novější (pro systémy Windows, Mac a Linux),
- Safari 3.1 a novější (pro systém Mac),
- Google Chrome (pro systémy Windows a Mac).

Gmaps také poskytují funkci *GBrowserIsCompatible()*, která prověří komptabilitu vašeho prohlížeče. Gmaps poskytuje výchozí kódování znaků ve formátu UTF-8[9]. Je vhodné, aby webová aplikace, která používá Gmaps, používala stejné kódování jako Gmaps.

4.1 Načtení Gmaps na webové stránky

Bude-li chtít uživatel načíst Gmaps na webové stránky, musí se nejprve načíst javascriptovou knihovnu. Kód pro načtení javascriptové knihovny poskytuje dokumentace[8]. Tento kód se vkládá do hlavičky html souboru obsahující mapu.

```
<script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=abcdefg&sensor=
true_or_false"
type="text/javascript"></script>
```

Pro načtení je nutné mít klíč. Tento klíč je možné vygenerovat na stránkách Google. Klíč bude vložen do kódu na místo *key=abcdefg*, kde nahradí *abcdefg*.

Gmaps poskytují možnost zjištění souřadnic místa výskytu počítače, který načítá novou mapu. Po zjištění těchto souřadnic, načte mapu, taky aby ukazovala toto místo. Tuto možnost poskytuje tato část kódu *senzor=true_or_false*. Pokud by byl *senzor* nastaven na hodnotu *true*, Gmaps zjistí souřadnice a zobrazí upravenou mapu.

[8] Dokumentace k Gmaps: http://code.google.com/intl/cs/apis/maps/documentation/javascript/v2/basics.html
[9] Je zkratka pro *UCS Transformation Format*. Je to způsob kódování znaků Unicode/UCS do sekvencí bajtů

Po načtení této knihovny je nutné vytvořit html tag id. Název může být libovolný, například *map*. Dále je nutné nastavit tomuto tagu velikost, která bude určovat i velikost mapy.

```
<div id="map" style="width: 500px; height: 300px"></div>
```

Tento kód vytvořil html tag id, s názvem *map*, se šířkou 500px a výškou 300px. Po načtení knihovny a vytvoření html tagu, se musí vytvořit objekt typu GMap se základními parametry pro vytvoření a načtení mapy. Pro vytvoření objektu je nutné vytvořit funkci s libovolným názvem, například *vytvor_mapu*. JavaScript s funkcí *vytvor_mapu,* může být vložen opět do hlavičky html souboru nebo do těla souboru.

```
<script type="text/javascript">
      function vytvor_mapu() {
      if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map"));
        map.setCenter(new GLatLng(50.009075, 15.396396), 7);
      }
    }
</script>
```

V tomto kódu je použita funkce *GBrowserIsCompatible(),* která zkontroluje komptabilitu prohlížeče. Pokud prohlížeč Gmaps podporuje, vytvoří se proměnná *map,* do které se uloží nový objekt GMap. Přesněji GMap2, dvojka na konci určuje verzi, která je použita. Parametr při vytváření objektu je html tag, do kterého se má nová mapa vykreslit.

Samotné vytvoření objektu GMap, je pro vykreslení mapy nedostačující. Mapa neví, kde se má určit střed mapy. Proto je nutné je nastavit funkcí *setCenter*. Tato funkce má dva parametry, prvním je GLatLng a druhým je Zoom level.

GLatLng je bod v zeměpisných souřadnicích. Vytváří přímo při volání funkce *setCenter*, ale je možné jej vytvořit jako proměnnou i před voláním. Při vytváření jsou zadávány dva parametry, ty jsou Latitude a Longitude.

Zoom level je ve stručnosti velikost přiblížení pohledu na určitých souřadnicích mapy udávaná číslem. S největším možným přiblížením se rovná Zoom level nule, naopak s nejmenším devatenácti. Důležité je, kde se nachází střed mapy, protože ne všechny oblasti mapy mají tak velký Zoom level.

Nyní je tedy nastaven střed mapy na souřadnice Prahy a přiblížení na hodnotu sedm. Stačí tedy jen funkci *vytvor_mapu* zavolat k vykonání. Funkce se zavolá při načítání těla funkce pomocí příkazu *onload="vytvor_mapu()"*. HTML kód vytvořené webové stránky vypadá takto:

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Google Maps JavaScript API Example</title>

    <!-- Zde načítáme javascriptovou knihovnu //-->
    <script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=abcdefg&sensor=
true_or_false"
            type="text/javascript"></script>

    <script type="text/javascript">
    //Zde vytváříme funkci pro vytvoření mapy
    function vytvor_mapu() {
      if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map"));
        map.setCenter(new GLatLng(50.009075, 15.396396), 7);
        map.setUIToDefault();
      }
    }
    </script>
</head>

  <!— Při vytváření těla rovnou voláme funkci initialize//-->
  <body onload="vytvor_mapu()">
    <div id="map" style="width: 500px; height: 300px"></div>
  </body>
</html>
```

Vytvořená mapa bude bez ovládacích prvků, čistě jen obrázek s mapou.

Obrázek 5 - první mapa

4.2 Nastavení vlastností mapy

Nyní ukážu, jak se dají do map přidat ovládací tlačítka, měřítka a nastavování jiných vlastností. Uvedu zde nejzákladnější vlastnosti, které se nejčastěji používají.

4.2.1 Typ mapy

Tak jako Mapy.cz nebo Amapy mají různé podklady pro své mapy, tak i Gmap používají různé podklady. Ty jsou vyjmenovány v tabulce 4.

Tabulka 4 - Podklady pro Gmap

Název podkladu	Vysvětlení
g_normal_map	Zobrazení standardní silniční mapy
g_satellite_map	Zobrazení satelitních snímků
g_hybrid_map	Zobrazení hybridní mapy, zobrazené cesty na satelitní mapě
g_default_map_types	Mapa všech tří typů
g_physical_map	Mapy založené na terénu

Nastavení jednoho z typů těchto map se provádí pomocí příkazu:

```
map.setMapType(G_SATELLITE_MAP);
```

Tento kód zobrazí satelitní mapu.

4.2.2 Zobrazení ovládacích prvků

Pro ovládání mapy, přiblížení, pohybování, změny podkladu jsou nutná ovládací tlačítka. Gmaps. Gmaps poskytuje tyto ovládací prvky:

- GLargeMapControl3D – velký panel s ovládáním zoom a pohybu, zobrazí se v levém horním rohu,
- GLargeMapControl – jednoduší velký panel s podporou zoomu a pohybu na mapě, zobrazí se v levém horním rohu,
- GSmallMapControl – malý panel s podporou zoomu a pohybu na mapě, zobrazí se v levém horním rohu,
- GSmallZoomControl3D – malý panel s podporou zoomu, ale bez podpory pohybu na mapě, zobrazí se v levém horním rohu,
- GSmallZoomControl – malý jednodušší panel s podporou zoomu, ale bez podpory pohybu na mapě, zobrazí se v levém horním rohu,
- GScaleControl – zobrazení měřítka mapy, zobrazí se vedle ikonky google blízko dolního levého rohu,
- GMapTypeControl – tlačítka na přepínání podkladů (typů) mapy, zobrazí se v pravém horním rohu.

Pro přidání některého z ovládacích prvků se používá funkce *addControl*. Tato funkce je používána podobně jako funkce pro nastavení středu mapy. Přidání ovládacích prvků může vypadat takto:

```
map.addControl(new GLargeMapControl());
```

Tento příkaz přidá k mapě velký panel na ovládání pohybu a zoomu. Typ ovládacího prvku předáváme parametrem jako objekt. Proto jej musíme inicializovat pomocí *new*.

4.2.3 Výchozí nastavení

Pokud se uživatel nebude chtít zdržovat nastavováním jednotlivých prvků, Gmaps poskytuje defaultní nastavení. Pro defaultní nastavení slouží funkce *setUIToDefault()*. Tato metoda přidá všechny potřebné prvky bez nutnosti postupného nastavování. Příklad použití kódu:

```
map.setUIToDefault();
```

K mapě budou nyní přidána tlačítka pro pohyb, zoomování a přepínání podkladů. V dolním rohu bude zobrazeno také měřítko. Mapa bude vypadat takto:

Obrázek 6 - Mapa s ovládacími prvky

5 Zobrazení objektu na Google maps

Dalším krokem ke splnění zadání práce je zobrazení značky na mapě. Jak zobrazit značku nebo ikonu vysvětlím v této kapitole. Každý objekt, který je zobrazován na mapě se nazývá GOverlays. Těchto objektů je několik, polygony, ikony atd. Pro zobrazení kolejového vozidla stačí znát zobrazení ikony neboli značky.

5.1 Marker

Zobrazení jakékoli značky na Gmaps, nejde udělat jen načtením ikonky a zadáním souřadnic místa zobrazení. Pro přidání značky do mapy se nejdříve musí vytvořit Marker. Marker má vlastnosti, do kterých spadá zmiňovaná značka, souřadnice a mnohé další. Marker se tedy používá k zobrazení bodů na mapě. Je to objekt typu GMarker, který spadá pod GOverlays.

Marker poskytuje tyto události, na které může aplikace reagovat zavoláním některých funkcí:

- click – jednou kliknutí,
- dbclick – dvojité kliknutí,
- mousedown – při stisknutí tlačítka myši,
- mouseup – uvolnění tlačítka myši, nemá vliv na mousedown,
- mouseover – při najetí myši na marker,
- mouseout – při vyjetí myši z marker.

Tyto události jsou nejpodstatnější, Marker poskytuje ještě další, které ale už nejsou tolik podstatné.

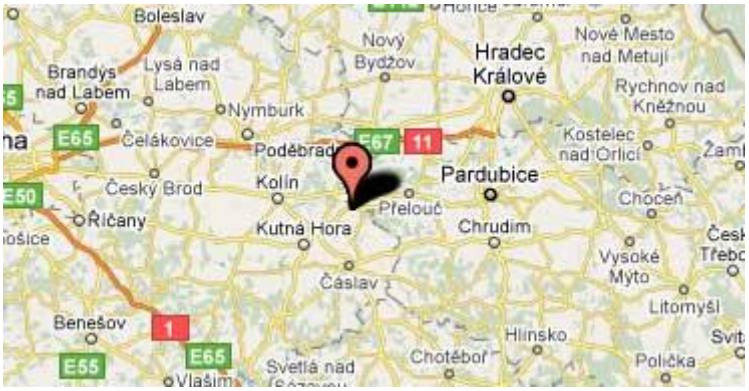
5.2 Vytvoření Marker s defaultní ikonou

GMarker má při vytváření dva základní parametry, přičemž povinný je pouze ten první. Jedná se o souřadnice, na kterých má být marker zobrazen. Souřadnice jsou typu GlatLng. Druhým parametrem je ikona. Ta není povinná. Pokud není zadaná, je nastavena defaultní ikona.

Při postupu se tedy nejdříve musí vytvořit souřadnice, tedy GlatLng. Poté je nutné vytvořit marker pomocí objektu GMarker. Na závěr je přidán do mapy jako objekt GOverlays. Ukázka kódu pro vytvoření markera s defaultní ikonou:

```
var point=new GLatLng(50.009075, 15.396396);
var Marker= new GMarker(point);
map.addOverlay(Marker);
```

Z této části javascriptu je patrné, že nejdříve byl vytvořen bod. Pokračuje se vytvořením marker, který má jako parametr zadaný pouze již vytvořený bod. Ostatní nastavení bude tedy defaultní. Nakonec bude marker přidaný do objektů mapy a zobrazen. Následující mapa s marker bude vypadat takto:



Obrázek 7 - Marker default

5.3 Vytvoření Marker s vlastní ikonou a GIcon

Na rozdíl od předchozího vytvoření marker zde bude použit při vytváření nejen bod zobrazení, ale i parametr s ikonou. Tato ikona se však nejdříve musí vytvořit.

5.3.1 Vytvoření ikony – GIcon

GIcon je vlastností marker. Určuje, jak bude marker vypadat. Do GIcon stačí načíst pouze obrázek a použít defaultní nastavení *G_DEFAULT_ICON*. To ovšem nemusí ladit s použitým obrázkem. Proto je lepší používat vlastní nastavení. Vlastnosti, které se dají nastavit u GIcon jsou vyjmenovány v tabulce 5.

Tabulka 5 - Vlastnosti GIcon

Vlastnost	Vstupní parametr	Vysvětlení
image	String	Adresa načtené ikony
shadow	String	Adresa obrázku pro stín
iconSize	GSize	Velikost ikony
shadowSize	GSize	Velikost stínu
iconAnchor	GPoint	Pixel k ukotvení obrázku
infoWindowAnchor	GPoint	Pixel pro ukotvení info okna

Tyto vlastnosti se nastavují po vytvoření ikony. Zde je ukázka kódu pro vytvoření vlastní ikony:

```
var ikona_vlak = new GIcon();
      ikona_vlak.image = "../images/vlak.png";
      ikona_vlak.iconSize = new GSize(70, 70);
      ikona_vlak.iconAnchor = new GPoint(35, 35);
      ikona_vlak.infoWindowAnchor = new GPoint(35, 35);
```

Je vytvořena GIcon o velikosti 70px na výšku a 70px na šířku, s kotvištěm uprostřed. Do ikony je načten obrázek *vlak.png*. Ne všechny vlastnosti musí být použity. U této GIcon není například nastaven stín ikony.

5.3.2 Vytvoření Marker s vlastí ikonou

Pro vytvoření marker vlastní ikonou se tedy používá parametr GlatLng a přidá se k němu parametr GIcon. Následující kód ukazuje vytvoření marker s GIcon z předchozího kódu.

```
var point=new GLatLng(50.009075, 15.396396);
var Marker= new GMarker(point,ikona_vlak);
map.addOverlay(Marker);
```

Nový marker bude vypadat takto:

Obrázek 8 - Marker s vlastní GIcon

5.4 Grafické formáty pro vytvořené ikony

Pro vytvoření ikon se používají nejčastěji tyto tři formáty.

5.4.1 PNG

PNG je zkratka z anglického *Portable Network Graphics*, česky to znamená přenosová síťová grafika. Je to grafický formát, vyvinutý jako náhrada za GIF. PNG oproti GIF nabízí podporu 24 bitové barevné hloubky. Obsahuje alfa kanál neboli průhlednost, proto je také preferován Gmaps jako grafický formát pro stíny ikon. Další výhodou je bezeztrátová komprese nebo například automatická detekce poškození. PNG je označován jako jediným formátem pro bitmapovou grafiku na internetu. Ze všech tří formátů je nejmladší, vznikl roku 1996.

5.4.2 GIF

Graphics Interchange Format, to je zkratka pro GIF. GIF je grafický formát pro rastrovou grafiku. Stejně jako PNG používá bezztrátovou kompresi. Výhodou oproti PNG je podpora jednoduché animace. Velkou nevýhodou je omezení v maximálním počtu barev, které činí 8 bitů, tedy 256 barev. Podle počtu barev, je i velikost souboru. GIF byl vyvinut společností CompuServe již v roce 1987. Dnes je jeden z nejpoužívanějších formátů webové grafiky, jak již ale bylo zmíněno, je postupně vytlačován PNG.

5.4.3 JPEG

JPEG je označováno *The Joint Photographics Experts Group*. Tento grafický formát je nejvíce využíván pro fotografie a podobné obrázky, kde je velké množství barev. Oproti předchozím formátům, používá ztrátovou kompresi, a proto není vhodný pro text, nebo ikony. Obsahuje až 24 bitou grafiku, tedy 16 777 216 barev. Všechny informace o barvách ukládá v RGB složkách a každá barva je kombinací tří základních červené, zelené a modré. JPEG také patří do skupiny nejpoužívanějších formátů. JPEG byl vytvořen okolo roku 1990.

6 Webová aplikace

V této kapitole zkusím stručně podat mojí tvorbu praktické části. Kde pomocí Gmaps zobrazuji polohu kolejového vozidla.

6.1 Použité technologie

Nyní představím programovací jazyky a jiné technologie, které je možné použít pro vypracování aplikace.

6.1.1 Ajax

Ajax neboli *Asynchronous JavaScript and XML,* znamená interaktivní webové aplikace. Ajax se nejčastěji používá pro změnu obsahu stránek, bez nutnosti znovu načítání (obnovení) stránky. Ajax se nejčastěji spojuje s *XMLHttpRequest*, což je rozhraní umožňující komunikaci mezi serverem a klientem prostřednictvím protokolu http.

Hlavní výhodou, jak již jsem řekl, je změna obsahu stránek bez nutnosti znovu načítání stránky celé. Výhoda je to jednak proto, že práce je plynulejší a také se tolik nezatěžují servery, protože odesílají jen potřebná data.

Nevýhodou Ajaxu je nefunkčnost procházení historiemi stránek, na kterých již byl Ajax použit. Ajax navíc nefunguje na starších prohlížečích, či mobilních telefonech a PDA. Ajax je poměrně mladý, jeho start se datuje k roku 2005.

Ajax jsem použil, protože se mi líbila jeho výhoda změny obsahu stránek bez znovu načítání. Tu jsem využil při pohybu vlaků.

6.1.2 XML a JSon

XML

XML je zkratkou pro *Extensible Markup Language,* v překladu rozšiřitelný značkovací jazyk[10]. Využívá se nejčastěji pro výměnu dat mezi aplikacemi nebo publikaci dokumentů. Podporuje i potřeby jiných jazyků než je angličtina. Pomocí XML tagů vyznačujeme význam dat v XML. Aktuální verze XML je 1.1.

Já jej využívám jako převod dat z databáze do ajax aplikace. Využil jsem jej kvůli přehlednosti výpisu dat a jednoduchosti použití.

JSon

JSon jsem sice nepoužil, ale rád bych ho zmínil, protože *Google* jeho podporu při práci s jeho mapy doporučuje. JSon je zkratkou pro *JavaScript Object Notation*. Jedná se o způsob zápisu dat nezávislých na platformě počítače, určených pro přenos dat. Vstupem je libovolná datová struktura, ale výstupem je vždy řetězec. JSon se nejvíce používá s aplikací Ajax a je jedním z největších konkurentů XML.

[10] Jazyk, jehož zdrojový kód obsahuje jak vlastní text, tak instrukce pro jeho zpracování

6.1.3 PHP

PHP je rekurzivní zkratkou *Hypertext Preprocessor* dříve *Personal Home Page.* Jedná se o skriptovací programovací jazyk, který je pro programování dynamického webu. PHP se provádí na straně serveru, klientovi se vrací pouze výsledek. Syntaxe je podobná programovacím jazykům *C, Pascal, Java* a je nezávislí na platformě. Jeho poslední verze je 5.3.3.

Má práce je s PHP spojená především vyžitím PHP GD knihovny, kdy při výběru správné ikony pro kolejové vozidlo používám funkce právě z této knihovny. Dále jej využívám pro stáhnutí dat z MySQL databáze a převodu na XML.

6.1.4 MySQL

MySQL je databázový systém, vytvořený firmou MySQL AB, kterou nyní vlastní společnost Sun Microsystems. MySQL je multiplatformní a jak už název napovídá, komunikuje se s ní pomocí jazyka SQL. MySQL se nejvíce používá ve spojení s PHP a Apache, kde tvoří základ software serveru. Jeho masivní používání podporuje to, že je volně šiřitelný. Nyní je nejnovější verze 5.1.

6.1.5 JavaScript

Je multiplatformní programovací jazyk, který se nejvíce používá pro tvorbu webových stránek, kde se často vkládá přímo do html stránky. Zapisuje se mezi tagy *<script>*.

```
<body>
Toto je normální text stránky.<br>
<script>
document.write("A toto napsal JavaScript");
</script>
</body>
```

Možností vložení je však víc. Skript může být vkládán do hlavičky *html* souboru nebo může být načten v hlavičce *html* souboru jako externí javascriptový soubor s koncovkou *js.*

JavaScript vytvořil *Brendan Eich* z tehdejší společnosti *NetScape*. Ačkoli název obsahuje slovo *Java*, nemá sám o sobě s programovacím jazykem Java nic společného. JavaScript se převážně spouští, až na straně klienta, to znamená, že to znesnadňuje práci se soubory a jinými daty na serveru. JavaScript je závislý na prohlížeči, tedy ne všude může náš skript fungovat stejně nebo nemusí fungovat vůbec.

6.1.6 Apache

„Apache HTTP Server je softwarový webový server s otevřeným kódem pro GNU/Linux, BSD, Solaris, Mac OS X, Microsoft Windows a další platformy. V současné době dodává prohlížečům na celém světě většinu internetových stránek.“[1]

6.2 Vytvoření ikony znázorňující polohu kolejového vozidla

Před vytvořením ikony je důležité určit si, co má ikona zobrazovat, o čem má informovat. Cílem práce je, aby z ikony byly patrné tyto věci:

- typ lokomotivy,
- směr vlaku,
- rychlost vlaku.

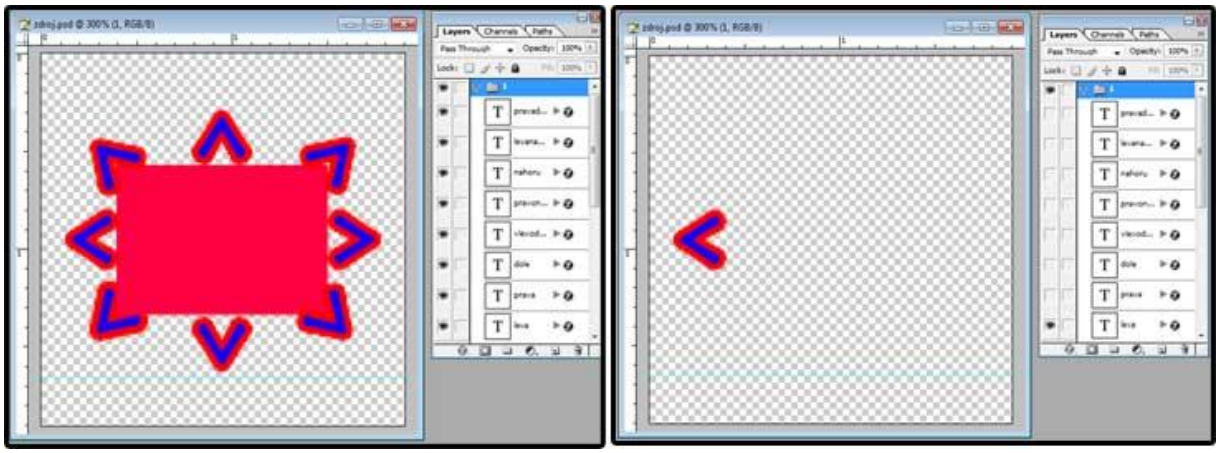
Určil jsem si jako nejlepší variantu ikonu, která bude obsahovat obrázek lokomotivy a obrázek šipky. Obrázek lokomotivy určí typ lokomotivy, bude to miniatura lokomotivy. Obrázek šipky bude určovat rychlost vlaku a směr vlaku, toto řešení je podle mne nejjednodušším.

Šipky určující směr vlaku budou rozděleny podle místa, kam ukazují. Bude celkem osm šipek znázorňujících možný směr vlaku. Každá šipka bude mít tři kategorie. Každá kategorie bude svým tvarem určovat rychlost vlaku viz obrázek 13. Celkem tedy bude 24 šipek, určujících směr a rychlost. Počet směrů a kategorie pro rozlišení rychlosti jsem si volil sám. Tento počet se mi zdá být optimálním k řešení mého úkolu.

Je možné vytvořit obrázek lokomotivy společně s již zmíněnými šipkami. To by ale muselo být 24 obrázků pro jeden vlak. Při větším počtu vlaků by byla aplikace velmi paměťově náročná. Při přidání nového vlaku pro zobrazení, by byla jeho tvorba ikon velmi zdlouhavá. Proto jsem zvolil způsob vytvoření ikon šipek zvlášť a ikonky lokomotivy zvlášť. Ke každému vlaku se tak vybere pouze jedna šipka v daný okamžik. To zmenší počet potřebných ikon. Při tvorbě nového vlaku bude stačit vytvořit pouze samotnou lokomotivu. Ikony lokomotiv budou rozměrově všechny stejné, šipky budou také rovněž stejné. To bude výhodné pro pozdější sjednocení těchto dvou ikon.

Pro vytvoření ikon je možné použít program Photoshop. Pro tvorbu ikonky kolejového vozidla se z fotografie vyřízne daná lokomotiva a upraví se její velikost. Upravené foto lokomotivy se uloží v GIF formátu.

Pro vytvoření ikon šipek pomocí programu Photoshop, se vytvoří zdrojový soubor. Tento soubor obsahuje několik vrstev. Každá vrstva obsahuje různou šipku. Aby šipky byly rozmístěny v požadovaných směrech, ve stejné vzdálenosti, vytvoří se u prostřed souboru čtverec. Tento čtverec má parametry ikony lokomotivy a simuluje ji. Po rozmístění šipek se každá šipka uloží zvlášť v GIF formátu. Obrázek 9 ukazuje tvorbu těchto šipek.



Obrázek 9 - Tvorba ikon

V obou dvou případech je použit formát GIF. Tento formát je použit, protože při spojování ve formátu PNG se pozadí ikony zbarvilo nevhodnou barvou. Na obrázku 9 je patrné vytvoření všech šipek rovnoměrně rozložených kolem červeného čtverce simulující ikonu vlaku. Druhá část obrázku je již v okamžiku, kdy je zobrazena pouze vrstva se šipkou směřující na západ. V tuto chvíli se šipka na západ dá uložit.

6.2.1 Spojení dvou ikon pomocí PHP skriptu

Vytvořené ikony šipek a lokomotiv, je nutné sjednotit dříve, než budou vloženy do mapy. To je výhodnější kvůli menšímu počtu marker a zahlcení mapy.

Sjednocení dvou ikon poskytuje knihovna PHP GD[11]. Pro práci s obrázky je v této knihovně část *PHP GD and Image functions*. Pro potřebu spojení dvou GIF obrázků se použijí pouze tyto funkce:

- *Imagecreatefromgif()* – funkce načte a vytvoří obrázek GIF,
- *Imagecopy()* – kopíruje část obrázku, mezi parametry patří kopírující obrázek, obrázek pro uložení kopie, začínající souřadnice části obrázku, výška a šířka výřezu a na závěr body vložení kopie výřezu,
- *Imagesx()* a *Imagesy()* – vrátí šířku nebo výšku obrázku v px,
- *header()* – použijeme pro zobrazení obrázku,
- *Imagegif()* – vytvoří obrázek gif,
- *Imagedestroy()* – uvolnění paměti.

Tyto funkce by bylo vhodné zkompilovat do php souboru a mít tento soubor jako knihovnu připravenou k použití. Tento kód ukazuje příklad, jak by mohl takový soubor vypadat:

[11] PHP GD – je knihovna obsahující funkce napsané v PHP

```php
<?php
// načtení parametrů (názvů obrázků) z url adresy
    $imgVlakFile = trim($_GET["vlak"]) . ".gif"; //parametr sipka
    $imgSipkaFile = trim($_GET["sipka"]) . ".gif"; //parametr vlak

// vytvorení obrázků, připravených k úpravě
    $imgSipka = imagecreatefromgif($imgSipkaFile);
    $imgVlak = imagecreatefromgif($imgVlakFile);

//kopírování části obrázku do druhého
imagecopy($imgSipka, $imgVlak, 33, 76, 0, 0, imagesx($imgVlak),
imagesy($imgVlak));
//zobrazení obrázku a uvolnění paměti
    header('Content-type: image/gif');
    imagegif($imgSipka);
    imagedestroy($imgSipka);
    $imgSipka = null;
    $imgVlak = null;
?>
```

Tento php soubor se volá url adresou, která obsahuje adresu php souboru a adresy ikon šipky a lokomotivy. Skript funguje pouze pro obrázky formátu GIF. Výsledek obrázku bude spojení šipky a lokomotivy.

Příklad volání souboru: *http://soubor.php?vlak=obrVlaku&sipka=obrSipky*. *ObrVlaku* a *obrSipky*, jsou adresy umístění obrázků. Takto vypadající adresa se pak smí zadat jako url adresa ikony pro GIcon.

6.3 GInfoWindow

Jedním z úkolů je zobrazení důležité informace o vlacích, které budou zobrazeny na mapě a to přijatelnou formou. GInfoWindow tento úkol velice jednoduše vyřeší. GInfoWindow je okno nebo také textové pole ve tvaru popisky či bubliny, které dokáže zobrazit potřebné informace o marker, mapě nebo o nějakém objektu. Jedná se o modifikaci marker.

GInfoWindow se objeví reakcí na nějakou událost. Jedna z možností je například svázat ho s událostí mouseover. Při najetí na marker se zobrazí toto okno, ve kterém je možné nechat vypsat potřebné informace. Pokud se nastaví otevření okna při události mouseover, tak většinou při události mouseout se nastaví zavření tohoto okna. Na mapě může být v jednu chvíli otevřeno pouze jedno okno.

GInfoWindow se otevře přes funkci *openInfoWindowHtml()*. Ta se může spustit jako funkce spojená s marker nebo přímo objektu GMap. U marker obsahuje pouze jeden parametr, a to text, který má zobrazit. Při spuštění funkce u GMap, se musí zadat dva parametry. Jednak bod ukotvení okna a jednak text.

Tento kód vytvoří GInfoWindow:

```
GEvent.addListener(marker, "mouseout", function() {
   map.closeInfoWindow();
    });
GEvent.addListener(marker, "mouseover", function() {
    marker.openInfoWindowHtml("Ahoj Světe!!");
//druhá varianta vytvoření
//   map.openInfoWindowHtml(point, "Ahoj Světe!!");
  });
```

Tento skript je pokračováním skriptu, kde byl vytvořen první marker s vlastní ikonou. První funkce zachytává událost, kdy myš opustí marker. Po této události se zavře GInfoWindow. Druhá funkce zachytává událost najetí kurzoru myší na marker. Tehdy se okno otevře a zobrazí nápis *„Ahoj Světe!!“*. Tento text se v práci nahradí informacemi o daném vozidle.

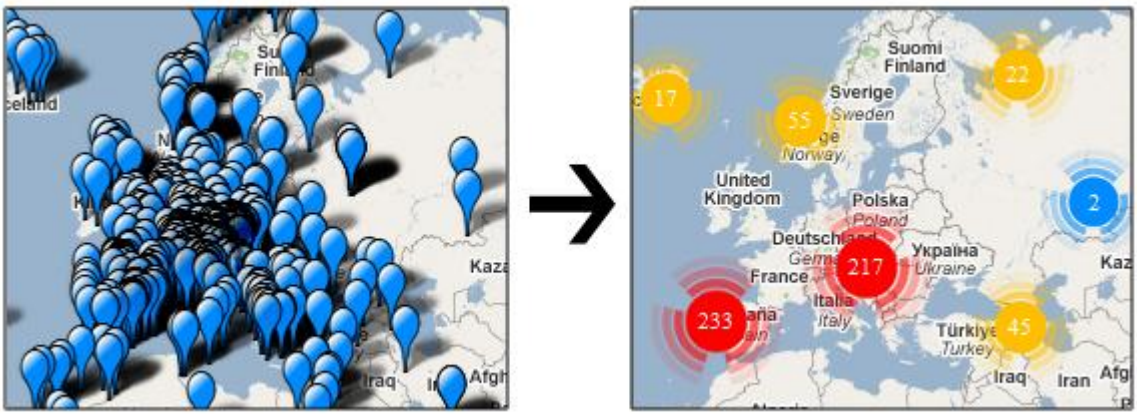
Obrázek 10 - GInfoWindow

Do vlastností GInfoWindow patří nastavování písma, velikost bubliny a podobných věcí, které jsou pro splnění úkolu nepotřebné.

6.4 MarkerCluster

MarkerCluster je javascriptová knihovna, kterou poskytuje google. Jak již název napovídá, pracuje hlavně s markers. Představme si mapu, na které bude zobrazeno velké množství značek, markers. To způsobí dva problémy. Prvním je velká nepřehlednost těchto značek, kdy jedna značka překrývá druhou. Druhý problém je velké zpomalení aplikace, která bude zahlcená vykreslováním těchto značek. MarkerCluster si umí s těmito problémy pohodlně poradit. Jednoduše řečeno, MarkerCluster překrývající se markers, sjednotí a zobrazí jako jeden. Po změně měřítka se překrývání přepočítá a sjednocené markers se upraví. Sjednocené markers se nazývají podle druhého slova v názvu této knihovny a to Clustery.

Clustery mají vlastní ikonu a jsou zobrazené na mapě jako marker. Obsahují pole, které obsahuje všechny markers v tomto Clusteru. Příklad použití MarkerCluster najdeme na obrázku 11.



Obrázek 11 - MarkerCluster

První část obrázku dává příklad zmíněného problému. Počet modrých značek je na tolik velký, že jejich zobrazení je nepřehledné. V druhé části je použití MarkerCluster. Číslo uprostřed těchto clusterů, udává počet markers v tomto clusteru. Mezi další výhodu MarkerCluster patří i barevná rozlišnost clusterů podle počtu markers. Slouží to k větší přehlednosti na mapě.

Pokud tedy bude potřeba vyřešit překrývání kolejových vozidel MarkerCluster bude vhodné řešení.

6.4.1 Implementace

Použití MarkerCluster není nikterak těžké. Vytvořené markers místo umístění do objektů na mapě, jsou umístěny do pole. Pak již stačí inicializovat MarkerCluster s parametrem tohoto pole. Následující kód vytvoří markers, přidá je do pole a inicializuje MarkerCluster.

```
var markers = [];
  for (var i = 0; i < 100; ++i) {
    var latlng = new GLatLng(latitude, longitude);
    var marker = new GMarker(latlng);
    markers.push(marker);
  }
var markerCluster = new MarkerClusterer(map, markers);
```

MarkerCluster má dva hlavní parametry, první typu GMap. GMap je mapa, do které se mají markers vykreslit. Druhý parametr je pole markers.

6.4.2 Nastavení

MarkerCluster se může nastavit dvěma způsoby. Buď to přímo v knihovně MarkerCluster, tudíž pak nové nastavení bude jako defaultní. Druhá možnost je nastavení pomocí konstruktoru.

U MarkerCluster se smí nastavit vlastnosti převážně pro funkčnost, viz další tabulka 6.

Tabulka 6 - Vlastnosti MarkerCluster

Vlastnost	Vstupní parametr	Vysvětlení
gridSize	Number	Velikost mřížky clusteru, která pohltí markery v ní obsažené, defaultní velikost je 60
maxZoom	Number	Maximální zoom, při kterém může být cluster zobrazen
styles	pole MarkerStyleOptions	Styly pro zobrazení clusterů

Poslední vlastnost *styles*, určuje vzhled Clusterů a do tohoto pole se vkládají další vlastnosti *MarkerStyleOptions*. V další tabulce jsou vlastnosti *MarkerStyleOptions* pro určování vzhledu clusterů.

Tabulka 7 - Vlastnosti MarkerStyleOptions

Vlastnost	Vstupní parametr	Vysvětlení
height	**Number**	**Výška obrázku**
width	**Number**	**Šířka obrázku**
opt_anchor	**pole čísel**	**Kotva pro text v clusteru, jedná se o pole dvou souřadnic, pokud nejsou nastaveny, text bude zarovnán do středu**
opt_textcolor	**String**	**Barva písma, defaultní je černá**
url	**String**	**Url obrázku pro načtení**

6.4.3 Modifikace

Mnozí uživatelé by mohli chtít informace o markers, které jsou obsaženy v Clusteru. MarkerCluster neposkytuje tuto možnost, nicméně knihovna MarkerCluster se dá upravit, tak aby tyto informace zobrazila.

Například pro zobrazení informací o Clusteru, se může použít GInfoWindow. Skript pro vytvoření GInfoWindow se musí začlenit do funkce pro vytvoření MarkerCluster. GInfoWindow se vytvoří s parametry *GLatLng* a textem. *GLatLng* jsou souřadnice, na kterých se na mapě otevře informační okno. Text obsahuje informace, které by se měli zobrazit po otevření tohoto okna.

Text, který bude obsahovat názvy maker, se přidá jako proměnná do Clusteru. V případě přidání marker do Clusteru, se přidá marker název do textu. Zdůrazňuji, přidá, nikoli nahradí. Tento text se poté předá parametrem funkci inicializující MarkerCluster. Ta si jej uloží a při vytváření MarkerCluster ho použije.

6.5 Databáze vlaků

Zobrazení polohy vlaků se dá nazvat simulací železniční dopravy. Aby ovšem šlo něco simulovat, musí existovat data potřebná k této akci. Data, která mi poskytla univerzita, se dají rozdělit na dvě odlišné tabulky dat. Nazývám je statická data a dynamická data. Obě tyto tabulky mají stejnou strukturu.

Sloupec	Typ	Nulový	Výchozí	Komentáře	MIME
MSG_ID	int(10)	Ne		id_zpravy	
UIC	bigint(20)	Ne		cislo_lokomotivy	
TRAIN_NUMBER	int(10)	Ano	*NULL*	cislo_vlaku	
LATITUDE	int(10)	Ne			
LONGITUDE	int(10)	Ne			
DATUM	timestamp	Ne	CURRENT_TIMESTAMP		
SPEED	int(3)	Ne			
AZIMUT	int(3)	Ne			
STATUS_MSG	int(10)	Ne			

Obrázek 12 - struktura tabulky databáze

Data tedy tvoří:

Tabulka 8 - Data aplikace

Název	Typ	Vysvětlení
MSG_ID	**8 místné číslo**	**Id zprávy, jedinečné**
UIC	**11 místné číslo**	**Číslo značící lokomotivu**
TRAIN_NUMBER	**5 místné číslo**	**Číslo vlaku, někdy není nezadáno**
LATITUDE	**7 místné číslo**	**Značí zeměpisnou šířku, bez desetinné čárky**
LONGITUDE	**7 místné číslo**	**Značí zeměpisnou délku, bez desetinné čárky**
DATUM	**rok-měsíc-den hodina:minuta:sec.**	**Datum a čas záznamu o vlaku**
SPEED	**1-3 místné číslo**	**Rychlost vlaku**
AZIMUT	**3 místné číslo**	**Udává směr vlaku**
STATUS_MSG	**1-2 místné číslo**	

6.5.1 Práce s databází

Databáze je předem daná a neupravuje se. Jde tedy jen o výpis dat. Do databáze se přistupuje pouze při spouštění aplikace. Poté se převedou veškerá data. Tím práce s databází končí.

```
$data = new mysqli("adresa_serveru","login",'heslo','databáze');
$sql = "databázový objekt";
$res = $data->query($sql);
```

Nejprve je nutno se k databázi připojit. Poté vytvořit databázový objekt, třeba dotaz. Data poté uložit do proměnné. Tento příklad je napsán pomocí PHP.

6.5.2 Převod dat z databáze do XML formátu

Po stažení dat je nutný převod do XML formátu, s kterým pracuje Ajax. V tomto případě se smí navázat k PHP skriptu, který je použit k připojení k databázi.

Nejprve se musí vytvořit XML dokument. Do tohoto dokumentu budeme přidávat postupně elementy a podelementy obsahující databázová data. Zde je ukázka převodu pouze jednoho elementu MSG_ID.

```
// pro výstup tohoto souboru bude xml dokument
header("Content-Type: text/xml");
//nový xml dokument
$xml = new DOMDocument('1.0','utf-8');
//vytvořím nový xml element vlaky
 $vlaky = $xml->createElement('vlaky');
 $xml->appendChild($vlaky);
//zde vytvořím element vlak, do kterého pomocí- cyklu načtu data z
databáze
while($zaznam = $res->fetch_object()){
 $vlak = $xml->createElement("vlak");
    $MSG_ID = $xml->createElement("MSG_ID");
    $MSG_ID_data = $xml->createTextNode($zaznam ->MSG_ID);
    $MSG_ID->appendChild($MSG_ID_data);
// uložím jako podelementy
 $vlak->appendChild($MSG_ID);
 $vlaky->appendChild($vlak); }
//uloží výstup
 $vystup = $xml->saveXML();
//vypíše výstup
 echo $vystup;
```

6.6 Statická data:

Přesný název tabulky je *poloha*_kv. Statická data obsahují celkem 83 záznamů. Z toho ani jeden vlak se stejným *TRAIN_NUMBER*, je-li zadané, se neopakuje. V této databázi je tedy 83 různých vlaků.

Pro každý vlak jsou dány pouze jedny souřadnice jeho umístění. To znamená, že v simulaci nebude zaznamenán pohyb vlaku, ale pouze jeho poloha v určitém čase. To je také důvod názvu statická data. Po zobrazení této polohy a uplynutí časového intervalu, vlak již nebude znovu zobrazen.

Výpis této tabulky by měl být seřazen podle času záznamu jednotlivých vlaků. Pro výpis se může použít tento databázový objekt:

```
SELECT * FROM poloha_kv ORDER BY DATUM;
```

Vlaky budou vypsány v pořadí podle času záznamu. Vlak s nejnižším časem záznamu bude první, naopak vlak s nejvyšším poslední. Seřazení vlaků je důležité, kvůli rychlejšímu vyhledávání dat, co mají být vykreslena na mapu. Tyto data budou seřazena do pole právě podle pořadí výpisu.

6.6.1 Funkce pro výběr dat na vykreslení

Simulace bude probíhat v intervalech po 30 sekundách. Po každém intervalu se vymažou vlaky, které byly zobrazeny a vyberou nové. Nové vlaky se můžou vybírat následujícím javascriptem.

```
function vykresli_vlaky(){
  var sirka,delka,dat,spe,c_vlak,point,cesta;
  var markers=[];
  hodiny();
    markerCluster.clearMarkers();
    map.clearOverlays();
    if(prvek!=null){
       while(prvek<TRAIN_NUMBER.length && porovnej(prvek)){
            sirka=LATITUDE.item(prvek).firstChild.data;
            delka=LONGITUDE.item(prvek).firstChild.data;
            sirka/=100000;
            delka/=100000;
            dat=DATUM.item(prvek).firstChild.data;
            spe=SPEED.item(prvek).firstChild.data;
            c_vlak=TRAIN_NUMBER.item(prvek).firstChild.data;
            point = new GLatLng(sirka,delka);
            cesta=vyberikonu(prvek);
            markers.push(createMarker(point,prvek,dat,spe,c_vlak,cesta));
            prvek++;
    }
  if(markers.length>0){
  markerCluster=new MarkerClusterer(map,markers);
   }
  }
 }
```

Na začátku funkce budou vytvořeny potřebné proměnné. Vymažou se veškeré objekty na mapě a aktualizuje se simulovaný čas. Globální proměnná *prvek* je na začátku aplikace inicializovaná na hodnotu null. Až po správném načtení XML dat je hodnota změněna na hodnotu 0 a značí index aktuálního prvku. Po změně aktuálního prvku se již nebudeme moct vrátit k použitým vlakům.

Funkce *porovnej* má jako parametr index aktuálního prvku. Jejím úkolem je porovnat čas prvku se simulovaným časem. Pokud je čas prvku menší než čas simulovaný a index aktuálního pole je menší než celkový počet prvků, nastanou tři věci.

- Jsou vybrána potřebná data na indexu aktuálního prvku pro vytvoření marker.
- Marker je následně zařazen do pole markers.
- Proměnná *prvek*, se zvýší o jednu jednotku a cyklus pokračuje s novým aktuálním prvkem.

Pokud některá ze dvou podmínek nebude splněna, cyklus se ukončí. Po ukončení cyklu testujeme, zda je naplněno pole markers, pokud ano inicializujeme MarkerCluster.

6.7 Dynamická data:

Přesným názvem tabulky je *table_export_pohyby_kv*. Obsahuje celkem 990 záznamů. Data obsahují šest stejných vlaků. Tedy pouze šest rozdílných hodnot *TRAIN_NUMBER*. Každý vlak má jiný počet záznamů s různými zeměpisnými souřadnicemi viz tabulka 9. Dynamická data tedy simulují pohyb těchto šesti vlaků.

Tabulka 9 - Počet výskytů jednotlivých vlaků v dyn. datech

Číslo vlaku	Počet výskytů
9375	81
9320	293
9326	223
9329	145
9304	158
2103	90

Výpis této tabulky by měl být seřazen podle vlaku, tedy *TRAIN_NUMBER*. Jednotlivé vlaky pak budou seřazeny podle času. Tedy budeme mít šest kategorií, seřazené podle času. Pro tento výpis můžeme použít tento dotaz:

```
SELECT * FROM table_export_pohyby_kv ORDER BY TRAIN_NUMBER, DATUM
```

Naše pole bude rozděleno do šesti pomyslných dílů. Prvních 81 prvků pole, budou mít stejné číslo vlaku *9375*. Budou ovšem seřazeny podle času záznamu. Od 81. prvku bude zase 293 prvků pole obsahovat číslo vlaku *9320*, také seřazeny podle času. Takto bude seřazeno celé pole. V takto seřazených datech se dá mnohem lépe orientovat a vyhledávat správné vlaky, které mají být zobrazeny.

6.7.1 Funkce pro výběr dat na vykreslení

Nyní jsou data v poli seřazená podle vlaků a až následně podle času. To znamená, že se nedá použít skript pro statická data. Pro rychlejší vyhledávání potřebných dat se budou používat tři globální proměnné typu pole.

- První pole počtem prvků v poli udává počet různých vlaků. Každá hodnota v poli pak udává počet záznamů s různými souřadnicemi aktuálního vlaku. Jedná se vlastně o tabulku 9,
- druhé pole také počtem prvků v poli určuje počet různých vlaků, ovšem hodnota v tomto poli, určuje index aktuálního prvku daného vlaku,
- třetí pole má opět počet prvků rovno k počtu různých vlaků, ovšem jejich hodnota je inicializována na hodnotu false.

Pole naplníme tímto skriptem:

```
function naplnPole(TRAIN_NUMBER){
    var PTN=TRAIN_NUMBER.item(0).firstChild.data;
    var pocet=0;
    var i=0;
// naplneni prvniho pole
    while(i<TRAIN_NUMBER.length){
         if(PTN==TRAIN_NUMBER.item(i).firstChild.data){
            pocet+=1;
        }else{
            if(pole[0]==0){
                pole[0]=pocet;
            }else{
                pole.push(pocet);
            }
            pocet=1;
            PTN=TRAIN_NUMBER.item(i).firstChild.data;
        }
        i++;
    }
    if(pocet>1){
        pole.push(pocet);
    }

//naplneni druheho pole, pro aktualni zaobrazovanej prvek
    for(var k=0; k<pole.length;k++){
        if(k==0){
            dpole[0]=0;
        }else{
            dpole.push((pole[k-1]+dpole[k-1]));
        }
    }
//Naplneni tretiho pole
    for(var k=0; k<pole.length;k++){
        bpole[k]=false;
    }
}
```

První cyklus projde celé pole s daty obsahující číslo vlaku. Dokud je předchozí číslo vlaku rovno následujícímu, prvek aktuálního pole se zvyšuje o jednu jednotku. Pokud není, do prvku pole se uloží aktuální hodnota a vytvoříme nový prvek. Pro tento nový prvek se opět počítá počet výskytů. Tento cyklus se opakuje, dokud se neprojde celé pole s daty.

Druhý cyklus je s pevným počtem opakováním, protože už je znám počet různých vlaků z předchozího pole. Do prvků druhého pole se vkládá součet aktuálního prvku předchozího vlaku a celkový počet výskytů předchozího vlaku. To zajistí, že v tomto poli budou uložené aktuální indexy prvků, od kterých začínají data jednotlivých vlaků.

Třetí pole bude obsahovat pravdivostní hodnotu false. Ta se změní na true v případě, že první vlak vyjel. První vlak vyjede, když simulovaný čas bude větší, než čas prvního záznamu vlaku. Samotná funkce pro výběr dat.

```
function vykresliVlaky(){
    var markers=[];
    hodiny();
    markerCluster.clearMarkers();
    map.clearOverlays();
    var sirka,delka,dat,spe,c_vlak,point,cesta,pom;

    for(var j=0;j<=pole.length; j++){
        if(dpole[j]<TRAIN_NUMBER.length){
           pom=dpole[j];
            while(pole[j]>0){
                if(porovnej(dpole[j])){
                    pole[j]--;
                    dpole[j]++;
                    bpole[j]=true;
                    if(pole[j]==0){
                        bpole[j]=false;
                    }
                }else{
                    if(pom<dpole[j]){
                        dpole[j]--;
                        pole[j]++;
                    }
                    break;
                }
            }

            if(bpole[j]){
                sirka=LATITUDE.item(dpole[j]).firstChild.data;
                delka=LONGITUDE.item(dpole[j]).firstChild.data;
                sirka/=100000;
                delka/=100000;
                dat=DATUM.item(dpole[j]).firstChild.data;
                spe=SPEED.item(dpole[j]).firstChild.data;
                c_vlak=TRAIN_NUMBER.item(dpole[j]).firstChild.data;
                point = new GLatLng(sirka,delka);
                cesta=vyberikonu(dpole[j]);
GMarker(point,createMarker(point,dpole[j],dat,spe,c_vlak,cesta));
markers.push(createMarker(point,dpole[j],dat,spe,c_vlak,cesta));
          }
        }
      }
    }
    if(markers.length>0){
        markerCluster=new MarkerClusterer(map,markers);
    }
}
```

I zde se nejdříve vytvoří potřebné proměnné, aktualizuje se simulovaný čas a vymažou se objekty na mapě. Tady je ovšem cyklus s pevným počtem opakováním. Cyklus má za úkol projet každý vlak. Počet vlaků je dán počtem prvků jednoho ze tří polí.

Druhý cyklus prochází prvky vlaků. Vždy začne od aktuálního prvku, tedy od hodnoty uložené v druhém poli. Tento cyklus je s podmínkou na začátku. Podmínkou je, že čas aktuálního vlaku je menší než simulovaný čas. Pokud se tak stane, hodnoty v polích se přepočítají, respektive posunou. Po nesplněné podmínce se cyklus ukončí.

Po skončení druhého cyklu může nastat tato situace. Pokud se index aktuálního prvku změnil a neprošel podmínkou na začátku, musí se vrátit indexy prvního a druhého pole. Musí se odečíst o jednu jednotku v případě druhého pole a přičíst u prvního.

Pokud již vlak vyjel, vybereme potřebná data a vytvoříme marker. Ten pak vložíme do pole markers. Po ukončení prvního cyklu inicializujeme MarkerCluster.

6.8 Funkce pro výběr ikon

Jak mají ikony vypadat, je patrné již z předchozích kapitol. Jedná se o šipku, která bude podklad pro ikonku vlaku. Nejprve si určíme parametry, podle čeho se bude ikona vybírat:

- UIC – a to konkrétně 6. 7. a 8. číslice, toto trojčíslí určí typ lokomotivy
- AZIMUT – směr šipky je dán azimutem, je vytvořeno 8 druhů šipek viz obrázek 9
- SPEED – typ šipky určí rychlost, jsou vytvořeny 3 druhy šipek

Obrázek 13 - Typ šipek

Funkce pro volání má jako parametr index aktuálního prvku dat. Podle toho jsou z pole dat vyjmuta potřebná data jako azimut, rychlost a číslo lokomotivy.

```
function vyberikonu(pr) {
   var azimut=AZIMUT.item(pr).firstChild.data;
   var rych=SPEED.item(pr).firstChild.data;
   var lok=UIC.item(pr).firstChild.data;
   var lok=lok.substring(5,8);
   var lokomotiva=parseInt(lok, 10);
   var sipka;
    if(rych==0){sipka="0";} else{
                if((azimut>=22)&&(azimut<67)){sipka="sv";}else{
                if((azimut>=67)&&(azimut<112)){sipka="v";}else{
                if((azimut>=112)&&(azimut<157)){sipka="jv";}else{
                if((azimut>=157)&&(azimut<202)){sipka="j";}else{
                if((azimut>=202)&&(azimut<247)){sipka="jz";}else{
                if((azimut>=247)&&(azimut<292)){sipka="z";}else{
                if((azimut>=292)&&(azimut<337)){sipka="sz";}else{
                sipka="s";
             }}}}}}}}
   if(rych>0 && rych<50){sipka=sipka+"1";}else{
    if(rych>=50 && rych<100){sipka=sipka+"2";}else{
     if(rych>=100){sipka=sipka+"3";}
   }
 }
var ik;
ik="http://localhost/spojeni/spoj.php?vlak=../images/lokomotivy/"+lokomot
iva+"&sipka=../images/sipky/"+sipka;
   return ik;
 }
```

Funkce vybere správnou ikonu pro vlak, i pro šipku. Url adresy šipky a obrázku lokomotivy se vloží do url adresy *spoj.php*, která je vrácena jako text. Při vytvoření marker se tato url adresa vloží místo adresy ikony.

6.9 Použití Ajaxu

V prvním případě se musí vytvořit XMLHttpRequest pro stažení dat v XML. Proto existuje funkce:

```
function vytvorXHR(){
    var xhr;
    try{
        xhr = new XMLHttpRequest();
    }catch(e){
        var MSXmlVerze = new
Array('MSXML2.XMLHttp.6.0','MSXML2.XMLHttp.5.0','MSXML2.XMLHttp.4.0','MSX
ML2.XMLHttp.3.0','MSXML2.XMLHttp.2.0','Microsoft.XMLHttp');
        for(var i = 0; i <= MSXmlVerze.length; i ++){
            try{
                xhr = new ActiveXObject(MSXmlVerze[i]);
                alert(MSXmlVerze[i]);
                break;
            }catch(e){}
       }
     }
    if(!xhr)
        alert("Došlo k chybě při vytváření- objektu XMLHttpRequest!");
    else
        return xhr;
}
```

Funkce je odolná vůči různým prohlížečům a obsahuje i starší verze XMLHttpRequest, v případě, že by prohlížeč nepodporoval nejnovější verze.

Další funkce slouží pro načtení XML z databáze za použití *vlaky_dyn.php* souboru. Skript v tomto souboru byl popsán v kapitole Databáze vlaků.

```
function prectiSoubor(){
    try{
        xhr.open("POST","../vlaky_dyn.php",true);
        xhr.onreadystatechange = ctiOdpoved;
        xhr.send(null);
    }
    catch(e){
        alert("Nelze se připojit k serveru:\n" + e.toString());
    }
}
```

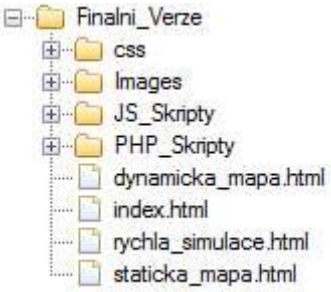
Poté je zavolána funkce *ctiOdpoved*, pro zpracování těchto dat. *CtiOdpoved* je funkce, která zpracuje data XML a zařadí je do pole. Dále volá ostatní funkce pro vytvoření mapy a pravidelné vykreslování marker.

```
function ctiOdpoved(){
    if(xhr.readyState == 4){
        if(xhr.status == 200){
            try{
                var XMLRes = xhr.responseXML;
                    if(!XMLRes || !XMLRes.documentElement){
                    throw("Chybná struktura XML:\n"+xhr.responseText);
                }
                var rootNodeName = XMLRes.documentElement.nodeName;
                if(rootNodeName == "parsereerror"){
                    throw("Chybná struktura XML:\n"+xhr.responseText);
                }
            }
            catch(e){
                alert("Chyba pri cteni odpovedi:"+e.toString());
            }
            vykresliMapu();
            var xmlRoot = XMLRes.documentElement;
            MSG_ID = xmlRoot.getElementsByTagName("MSG_ID");
            UIC = xmlRoot.getElementsByTagName("UIC");
            TRAIN_NUMBER = xmlRoot.getElementsByTagName("TRAIN_NUMBER");
            LATITUDE = xmlRoot.getElementsByTagName("LATITUDE");
            LONGITUDE = xmlRoot.getElementsByTagName("LONGITUDE");
            DATUM = xmlRoot.getElementsByTagName("DATUM");
            SPEED = xmlRoot.getElementsByTagName("SPEED");
            AZIMUT = xmlRoot.getElementsByTagName("AZIMUT");
            STATUS_MSG = xmlRoot.getElementsByTagName("STATUS_MSG");

            naplnPole(TRAIN_NUMBER);
            vykresliVlaky();
            window.setInterval("vykresliVlaky()",30000);
        }else{}
    }
}
```

Nejprve se projedou data XML a ověří se jejich funkčnost. Poté se převede XML na dokument DOM a naplní se pole, která ponesou data pro jednotlivé elementy. Zavolá se funkce pro vykreslení mapy, naplní se pole pro dynamická data, jsou-li použita. Na závěr je nastaven interval 30 sekund pro pravidelný výběr dat a následné překreslení mapy.

6.10 Architektura aplikace



Obrázek 14 - Architektura aplikace

Všechny potřebné skripty, obrázky a jiné soubory jsou uloženy ve složce *Finální_Verze*. Tato složka se rozděluje na pod složky a pod soubory.

Soubor *index.html* je hlavní spouštěcí soubor aplikace. Při spuštění obsahuje odkazy na další soubory obsahující mapy. Těmito soubory jsou *dynamicka_mapa.html, rychla_simulace.html, staticka_mapa.html.*

Soubor *dynamicka_mapa.html* zobrazuje simulaci pohybu vlaků. Používá k tomu data z databázové tabulky *table_export_pohyby_kv*. Soubor *staticka_mapa.html* zobrazuje simulaci polohy vlaků. Používá k tomu data z databázové tabulky *poloha_kv*. Soubor *rychla_simulace.html* zobrazuje odlehčenou simulaci dynamických dat. Odlehčená je tím, že pohyb kolejových vozidel není závislí na simulovaném čase. Tato simulace zrychleně simuluje pohyb vlaků.

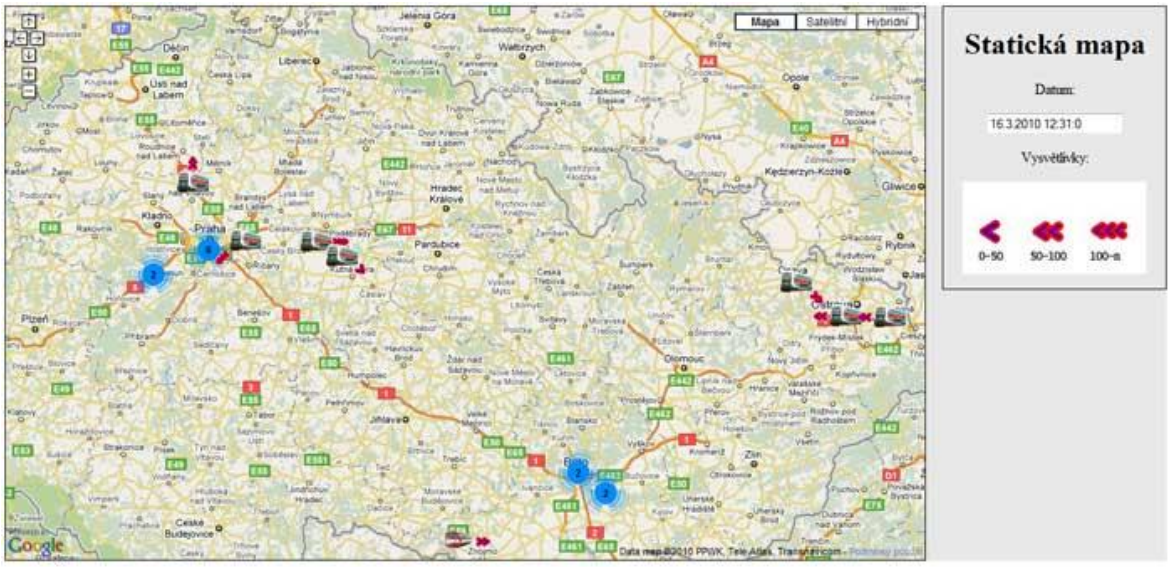
Složka *css* obsahuje soubor kaskádových stylů, který používají soubory s příponou *html.*

Složka *Images* obsahuje obrázky používané pro zobrazení polohy kolejového vozidla. Obsahuje šipky a ikonky lokomotiv- všechny v GIF formátu. Obsahuje i zdrojové soubory obrázků ve formátu *psd*[12], pro případnou úpravu.

Složka *JS_Skripty* obsahuje javascriptové soubory s příponou *js*. V těchto souborech popisované funkce na vytvoření map, MarkerCluster, výběr dat a podobné funkce. Každá mapa má svůj jedinečný javascriptový soubor s odlišnými funkcemi.

Složka *PHP_Skripty* obsahuje soubory s koncovkou *php*. Dva soubory slouží pro převod dat z databázových tabulek pro dynamická a statická data. Poslední soubor *spoj.php* je skript pro spojení dvou ikon.

[12] psd – Adobe Photoshop Image



Obrázek 15 – Ukázka aplikace

7 Závěr

V teoretické části práce jsem popsal využití mapových portálů, jejich možnosti a vlastní práci s mapovými portály.

V praktické části práce jsem vytvořil aplikaci, která využívá mapový portál k zobrazení poloh kolejových vozidel. V aplikaci se podařilo vytvořit všechny funkcionality stanovené v zadání. Aplikace umožňuje zobrazit v daném čase polohu kolejových vozidel a stanovené informace k těmto vozidlům.

Samotná aplikace je rozdělená na tři podaplikace. Poskytuje tři různé simulace zobrazení polohy kolejového vozidla. K těmto simulacím byly využity dva typy dat. Data určující pohyb kolejového vozidla po jeho trase a data určující polohy různých kolejových vozidel.

První podaplikace simuluje pohyb kolejových vozidel, bez závislosti na daném čase. Využívá data určující pohyb kolejového vozidla.

Druhá podaplikace simuluje také tento pohyb, ovšem pohyb tohoto vozidla je závislý na daném čase.

Třetí podaplikace simuluje polohy více kolejových vozidel. Využívá data určující polohy různých kolejových vozidel.

Samotné zobrazení polohy kolejového vozidla je u všech tří aplikací stejný. Liší se pouze daty používané pro simulaci a funkcemi pro práci s těmito daty. Tato aplikace pracuje pouze s určitými daty, které jsou neměnné. To pro mě bylo výhodné z hlediska jednodušší práce s databází, kde jsem nemusel řešit ukládání nových dat ani SQL injection[13]. Ovšem pro větší využití aplikace, nejen jako ukázky, by byla práce s databází klíčová. Například by aplikace mohla mít vytvořenou databázi vlaků, která by obsahovala adresy obrázků. To by bylo vhodné pro přidávání, nových lokomotiv. Aplikace by mohla neustále pracovat s databází, kde by pak mohla sloužit jako informační portál pro běžné uživatele internetu.

Toto je moje první práce s API mapami. Při samotné tvorbě aplikace jsem se naučil mnoho nového a získal cenné zkušenosti, které bych si rád dále rozšiřoval dalším studiem v této oblasti. Myslím si, že práce na této aplikaci pro mě byla velkým přínosem hlavně z hlediska JavaScript a PHP a získané zkušenosti, budu moci využít v dalším průběhu studia i v profesním životě.

[13] Útoky na databázy.

Literatura

[1]Apache HTTP Server. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 12. 6. 2006, last modified on 17. 5. 2010 [cit. 2010-08-09]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Apache_HTTP_Server>.

[2]API. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2. 12. 2004, last modified on 8. 8. 2010 [cit. 2010-08-09]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/API>.

[3]*Google Geo Developers Blog* [online]. 23.4. 2009. 23.4. 2009 [cit. 2010-08-09]. MarkerClusterer: A Solution to the Too Many Markers Problem. Dostupné z WWW: <http://googlegeodevelopers.blogspot.com/2009/04/markerclusterer-solution-to-too-many.html>.

[4]SKŘIVAN, Jaromír. *Interval.cz : Webová grafika* [online]. 16. 05. 2002 [cit. 2010-08-09]. GIF, JPEG a PNG - jak a kdy je použít?. Dostupné z WWW: <http://interval.cz/clanky/gif-jpeg-a-png-jak-a-kdy-je-pouzit/>.

[5]GIF. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 15. 9. 2005, last modified on 12. 7. 2010 [cit. 2010-08-09]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/GIF>.

[6]PNG. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 31. 3. 2007, last modified on 28. 4. 2010 [cit. 2010-08-09]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/PNG>.

[7]JPEG. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 5. 9. 2005, last modified on 20. 6. 2010 [cit. 2010-08-09]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/JPEG>.

[8]ZRALÝ, Jiří. *Digitální citron* [online]. 6. 11. 2005 [cit. 2010-08-09]. AJAX - návod pro začátečníky. Dostupné z WWW: <http://citron.blueboard.cz/clanek-239-ajax-navod-pro-zacatecniky.html>.

[9]GRIMMICH, Šimon. *Tvorba webu : JavaScript* [online]. 2003 [cit. 2010-08-09]. Dostupné z WWW: <http://www.tvorba-webu.cz/>.

[10]LÁSLO, Petr. *Programujte* [online]. 26. 06. 2008 [cit. 2010-08-09]. Ajax - Úvod. Dostupné z WWW: <http://programujte.com/?akce=clanek&cl=2008062101-ajax-%96-uvod>. ISSN 1801-1586.

[11] *PHP* [online]. Naposledy editována 12. 4. 2010 [cit. 2010-08-09]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/PHP>.

[12] *HyperText Markup Language* [online]. Naposledy editována 10. 5. 2010 [cit. 2010-08-09]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Html>.

[13] MySQL In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 21. 11. 2004, 7. 4. 2010 [cit. 2010-08-09]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/MySQL>.

[14]*Google code : Google Maps API Family* [online]. 2008, 21.6.2010 [cit. 2010-08-09]. Dostupné z WWW: <http://code.google.com/intl/cs/apis/maps/index.html>.

[15]*Mapy.cz : API Mapy.cz* [online]. 2006 [cit. 2010-08-09]. Dostupné z WWW: <http://api.mapy.cz/>.

[16]*AMapy : Atlas AMapy API - cesta k vlastním mapovým aplikacím* [online]. 2007 [cit. 2010-08-09]. Dostupné z WWW: <http://amapy.centrum.cz/api-doc/index.html>.

[17] *JavaScript* [online]. Naposledy editována 14. 4. 2010 [cit. 2010-08-09]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/JavaScript>.

[18]*PHP: GD - Manual : Image Processing and GD* [online]. 2001 [cit. 2010-08-09]. Dostupné z WWW: <http://php.net/manual/en/book.image.php>.