UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Informační tabule pro cestující se zobrazením polohy spoje

Karel Steinmetz

Bakalářská práce
2010

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

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Karel STEINMETZ**
Osobní číslo:	**I07789**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Informační tabule pro cestující se zobrazením polohy spoje**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Teoretická část práce se bude zabývat problematikou zobrazení více objektů v Google Maps.
Praktická část práce spočívá v realizaci WWW aplikace pro zobrazení dat o poloze do stanice přijíždějících vozidel (vlaků, autobusů) v mapě (Google Maps) v místě železniční nebo autobusové zastávky či stanice na širokoúhlé LCD nebo plasmové obrazovce se vstupem HDMI nebo PC.
Údaje na displeji by měly umožnit orientaci cestujícího ohledně doby příjezdu konkrétního spoje (dle jízdního řádu a dle skutečnosti), zobrazeni musí bezobslužně umožnit zobrazení příjezdů všech během určitého času (například 15, 30, 60 minut) očekávaných spojů a spojů opožděných, které ještě nepřijely do vybrané stanice. Péči je třeba věnovat i problematice překryvu informací v mapách a viditelnosti základních informací i pro osoby se sníženou kvalitou zraku.
Zpráva o poloze vozidla bude obsahovat datum a čas, identifikaci spoje a GPS souřadnice. Tyto zprávy jsou generovány vozidly a zasílány do stacionární části v definovaných časových periodách a při zastavení a rozjetí vozidla.
Pro archivaci dat bude použita technologie Oracle 10g, pro WWW stránky technologie PHP, EXTJS a framework ZEND.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

***Ružbársky, J.: Disertační práce: Graficke nastroje pre podporu dopravnej technologie. FRI Žilinskej univerzity, 2008.**
***Lacko, L.: Oracle - Správa, programování a použití databázového systému. Computer Press, 2007.**
***http://www.extjs.com/products/extjs/**

Vedoucí bakalářské práce: **RNDr. David Žák, Ph.D.**
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 12. 8. 2010 Karel Steinmetz

Poděkování

Tímto bych rád poděkoval prof. RNDr. Davidu Žákovi, Ph.D. za poskytnutí cenných rad a připomínek v průběhu vypracování bakalářské práce.

Anotace

Tato práce se zabývá problematikou zobrazení více objektů ve webové aplikaci Google Maps a využitím této aplikace v informační tabuli pro cestující se zobrazením aktuální polohy a základních informací spoje.

V praktické části je vytvořena www aplikace pro zobrazení dat o poloze do stanice přijíždějících vozidel (vlaků, autobusů) v internetové mapě (Google Maps) v místě železniční nebo autobusové zastávky či stanice na širokoúhlé LCD nebo plasmové obrazovce se vstupem HDMI nebo PC.

Klíčová slova

Gogole Maps, HTML, JavaScript, Zend Framework, informační tabule

Title

Passenger information board showing the location of routes.

Annotation

This work deals with the display of multiple objects in a Web application using Google Maps and the application of the information board for passengers to display their current locations and basic services.

The practical part is a web application to display data on the location of vehicles coming into the station (trains, buses) in online map (Google Maps) at the bus stop or railway station on a widescreen LCD or plasma screen with HDMI or PC.

Keywords

Google Maps, HTML, JavaScript, Zend Framework, information board

Obsah

Seznam zkratek

PHP	Hypertext Preprocesor (původně Personal Home Page)
EXTJS	Eee Eks Tee JavaScript
HDMI	High-Definition Multimedia Interface
PC	Personal Computer
LCD	Liquid Crystal Display
WWW	World Wide Web
XML	Extensible Markup Language
HTTP	Hypertext Transfer Protocol
SMTP	Simple Mail Transfer Protocol
Telnet	Telecommunication Network
DNS	Domain Name System
RAC	Oracle Real Application Clusters
ASM	Automatic Storage Management
AWR	Automatic Workload Repository
ADD	Automatic Database Diagnostic Monitor
XHTML	Extension Hypertext Markup Language
CGI	Common Gateway Interface
API	Application Programing Interface
URL	Uniform Resource Locator
MVC	Model View Controller
GPS	Global Positioning System
CSS	Cascading Style Sheets
LED	Light Emiting Diode
RGB	Red Green Blue
ČD	České dráhy
BSD	Barcley Software Distribution
DOM	Document Object Model

Seznam obrázků

Seznam tabulek

1 Úvod

Tato bakalářská práce se zabývá problematikou zobrazení více objektů ve webové aplikaci Google Maps. Cílem práce je vytvoření informační tabule pro cestující se zobrazením aktuální polohy a základních informací spoje v internetové mapě Google Maps.

Informační tabule je vytvořená jako webová aplikace, která využívá moderní technologie. Pro archivaci dat je použita technologie Oracle 10g, pro WWW stránky technologie PHP, EXTJS, Zend Framework, JavaScript a XML. Aplikace je navržena tak, aby jí bylo možné využít v jakékoli hromadné osobní přepravě, např. vlaky, autobusy, atd. Obsahuje dvě základních částí a to administrační a informační tabuli.

Administrační část slouží pro správu dat o vozidlech, stanicích či zastávkách, jízdních řádech a nastavení informační tabule.

Informační tabule prezentuje za pomocí Google Maps získaná data od vozidel, vlastnosti dopravních prostředků a informace o konkrétní stanici či zastávce.

2 Google Maps

Gogole Maps je jednou z nabízených internetových služeb společnosti Gogole, která je známá především díky svému internetovému vyhledávači. Tato služba nabízí internetovou mapu s mnoha možnostmi, které může uživatel využít. Mezi základní a mnoha lidmi běžně využívané patří především mapy ulic, plánovač cest pro cestování pěšky, automobilem nebo veřejnou dopravou a zobrazení polohy mnoha historicky či jinak významných míst. Podle jednoho z tvůrců Larse Rasmussena jsou Google Maps způsob, jak zorganizovat světové geografické informace. Podobným produktem společnosti google je Google Earth. Za použití technologie Google Maps API je také možné vkládat informace od třetích stran [1].

2.1 Zobrazení informací

Google Maps používají pro zobrazení informací satelitní pohled a Mercatorovo zobrazení a tudíž nemohou zobrazit oblast kolem pólu [1].

2.1.1 Satelitní pohled

Satelitní pohled je tvořen satelitními snímky ve vysokém rozlišení dnes, už všech světových oblastí. Google po postupném zveřejňování zmapovaných oblastí čelil mnoha stížnostem, důvodem byl strach některých vlád, které měli obavu z potenciální hrozby použití map pro plánování teroristických útoků. Proto Gogole rozmazal některé oblasti. Rozmazané oblasti se nachází především ve Spojených státech amerických např. Bílý dům. Některé oblasti jsou ovšem zmapovány pouze leteckými snímky a ne všechny jsou ve stejném rozlišení. Nepsaným pravidlem je, že méně osídlené oblasti jsou zmapovány hůře nebo zkresleně [1].

2.1.2 Mercatorovo zobrazení

Mercatorovo zobrazení je druh úhlojevného válcového mapového zobrazení, které navrhl roku 1569 vlámský kartograf Gernard Mercator. Tento typ zobrazení je hlavně používaný u námořních a leteckých navigačních map.

Základem zobrazení je válec v rovnoběžné poloze se zemskou osou a dotýkajícím se na rovníku. Po zobrazení povrchu koule na válec a po rozvinutí pláště do roviny vznikne pravoúhlá síť poledníků a rovnoběžek. Poledníky jsou zobrazeny ve stejných rozestupech, zatímco vzájemná vzdálenost rovnoběžek směrem k pólům narůstá až do nekonečna. Díky tomuto je nepřesnost v oblasti kolem rovnoběžníku celkem zanedbatelná. Naopak v oblastech kolem poledníků je tento ty mapy nepoužitelný [3].

2.2 Google Maps v internetovém prohlížeči

Verzi webové aplikace Google Maps pro Českou Republiku nalezneme na internetové adrese <http://maps.google.cz/>. Stránka na této adrese je složena z horního

panelu, který je určený pro zadání hledaných výrazů v mapě. V levé části stránky se nachází panel, který nabízí rozšířené vyhledávání a další užitečné funkce. Posledním prvkem na stránce je samotná mapa, která obsahuje navigaci, malou orientační mapku a tlačítka pro nastavení typu zobrazení jiných možností.

2.3 Vyhledávání

Vyhledávat je možné podle názvů firem, adres, ulic a křižovatek, míst, souřadnic, významných míst jako jsou parky, hory, jezera atd. Nově také Google zavádí možnost vyhledávání nemovitostí podle daných kriterií jako je rozmezí ceny a plochy, počet ložnic a koupelen.

2.4 Mapy a trasy

Rozšířené vyhledávání nabízí vyhledání trasy z bodu A do bodu B. Google nabízí výběr mezi pěšími a silničními cestami. Vzdálenost obou typů cest je možné zobrazit v mílích či kilometrech a silniční doprava je obohacena o volby vynechat dálnice a vyhnout se zpoplatněným úsekům. Vyhledaná trasa je na mapě zobrazena jako modrá čára. V levém panelu je vyhledaná trasa rozdělena na očíslované úseky. Jednotlivé úseky lze poté zobrazit podrobněji. Trasy je také možné ještě přizpůsobit zadáním dalších cílových míst.

2.5 Tvorba vlastní mapy

Google nabízí v závislosti na své poloze vytváření a sdílení vlastních map, které je možné přizpůsobit vlastní potřebě. Vlastní mapy mohou obsahovat označení míst, čáry a tvary. Po vytvoření mapy můžete přidat popisný text ve formátu RTF nebo HTML, vkládat do mapy fotky a videa, sdílet mapy, otevřít mapu v Google Earth, spolupracovat s ostatními uživateli, importovat do mapy soubory. Vlastní mapy může vytvářet pouze každý, kdo vlastní účet u Google, ale prohlížení mapy je možné bez účtu a přihlášení.

2.6 Rozhraní API

Google Maps mají širokou škálu rozhraní API, které vývojáři umožňují tuto aplikaci vložit do vlastní aplikace, ať už webové, či desktopové a využít tak možnosti, které toto API nabízí. Rozhraní jsou rozdělena podle typu použití mapy.

2.6.1 JavaScript API

Toto API rozhraní umožňuje vložení mapy do vlastní webové stránky pomocí javascriptu a dále s mapou manipulovat či upravovat její obsah podle potřeby konečného uživatele. Aktuální verzí tohoto rozhraní je verze 3. Verze je určena pro mobilní zařízení i desktopové aplikace. API programátorovy poskytuje řadu služeb pomocí, kterých je možné vytvořit i velmi složité aplikace na ve vlastní webové aplikaci [4].

2.6.2 API for Flash

Pomocí API for Flash je možné vkládat mapy do internetové stránky založené na technologii Flash. Podobně jako verze JavaScript poskytuje toto API řadu nástrojů pro manipulaci a úpravy obsahu v mapě [4].

2.6.3 Google Earth API

Google Earth API umožňuje vložit aplikaci Google Earth do webových stránek a zobrazit tak informace ve 3D. Použitím tohoto API můžeme pouze krátkým kódem vytvořit propracovanou 3D mapu s vlastním obsahem [4].

2.6.4 Static Maps API

Toto API slouží pro zobrazení mapy jako statického obrázku na webových stránkách bez použití JavaScriptu nebo dynamického načítaní stránky. Statická mapa je sestavena podle požadavku přes URL odeslaného pomocí standardního HTTP požadavku. Do mapy je možné vkládat značky, či vlastní ikony, kreslit lomené čáry a různé oblasti, atd. [4].

2.6.5 Web Services API

Web Services API slouží pro poskytnutí geografických dat. Komunikace probíhá pomocí HTTP požadavku a následné odpovědi na požadavek. Toto API můžeme dále rozdělit na:

- **Geocoding API** - požadavek pro převod adresy na zeměpisné souřadnice pomocí, kterých je možné adresu zobrazit a naopak.
- **Directions API** – požadavek pro vypočtení trasy z bodu A do bodu B, ovšem tato služba není navržena tak, aby reagovala v reálném čase u uživatele.
- **Elevation API** – toto API poskytuje údaje o nadmořské výšce pro všechny lokality na světě včetně oceánu. Přístup k tomuto API je poskytnut i přes JavaScript API formou objektu ElevationService().
- **Places API** – tato služba zprostředkovává geografické umístění a význam daného místa.

Toto API nám tedy usnadňuje a zpřehledňuje prácí s geografickými daty v naší webové aplikaci [5].

2.7 Použití JavaScript API V3

JavaScript API V3, kde V3 značí aktuální verzi 3, nahrazuje předchozí verzi 2, která byla spuštěna v roce 2006. Oproti předchozím verzím se vyznačuje V3 přepracovanou vnitřní architekturou MVC, díky které byla rozšířena podpora pro mobilní zařízení, např. iPad, iPhone atd. Díky této architektuře došlo k přejmenování tříd či

prototypů, které jsou nyní v objektu `google.maps`. Třída pro značky se pak tedy jmenuje `google.maps.Marker`, oproti verzi 2, kde byl třída pojmenovaná `GMarker`. Už také není potřeba vlastnit API Key neboli licenční klíč, který byl u verze 2 dříve potřeba pro webové aplikace, kde se Google Maps vyskytují. Byly přidány vrstvy Traffic Layer (vrstva pro zobrazení dopravní situace), Bicycle Layer (vrstva zobrazující cyklistické stezky) a nově spuštěná služba Elevation API, pomocí které lze snadno zjistit nadmořskou výškou daného místa [6].

2.7.1 MVC model

Díky tomuto modelu můžeme intuitivně proměnným daného objektu. Veškeré vlastnosti se nastavují pomocí konstruktoru nebo pomocí tzv. setrů (např. `setCenter(latlng)`) a jejich hodnoty je možné získat pomocí tzv. getrů (např. `getCenter()`). Toto nám vždy umožňuje konkrétně zvolit název metody pro přístup k dané vlastnosti.

Stavy jednotlivých objektů jsou uloženy v jejich proměnných a lze k nim tedy snadno přistupovat. Změny těchto stavů jsou tedy zobrazeny a řízeny pomocí událostního modelu.

2.7.2 „Ahoj, světě!" v Google Maps

Pro prácí s verzí 3 je doporučeno používat nový značkovací jazyk HTML5, proto použijeme:

```
<! DOCTYPE html>
<html>
 <head>
  <META name="viewport" content="initial-scale=1.0, user-scalable=no" />
  <style type="text/css">
   html (výška: 100%)
   body (výška: 100%; margin: 0px; padding: 0px)
  </ Style>
  <script type="text/javascript"
   src="http://maps.google.com/maps/api/js?sensor=false">
  </ Script>
```

URL <http://maps.google.com/maps/api/js> odkazuje na umístění souboru, pomocí něhož načte prohlížeč všechny potřebné objekty z Google Maps API, které jsou používané ve verzi 3.

Parametrem senosor je potřeba určit zda bude aplikace využívat kolokaci. Jedná se o určení pozice uživatele s využitím GPS senzoru nebo síťové adresy. Nastavení sensoru na hodnotu true má význam pouze pro aplikace, které využívají mobilní zařízení. Po té pomocí technologie AJAX webová aplikace snadno zjistí jeho polohu.

```
<script type="text/javascript">
   function initialize() {
    var myLatlng = new google.maps.LatLng(-34.397, 150.644);
    var myOptions = {
      zoom: 8,
      center: myLatlng,
      mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    var map = new google.maps.Map(document.getElementById("map_canvas"),
      myOptions);

    var marker = new google.maps.Marker({
      position: myLatlng,
      map: map,
      title: "Ahoj, světe!"
    });
   }
  </script>
 </head>
```

Objekt `google.maps.LatLng` nám umožňuje pracovat se souřadnicemi. Tuto třídu lze tedy využít všude, kde je potřeba definovat souřadnice jednotlivých bodů na mapě. Při tvoření jeho instance je potřeba konstruktoru předat hodnotu zeměpisné šířky a délky oddělené čárkou.

Objekt `google.maps.Map` je nejvýznamnějším prvkem celé aplikace, jelikož je to základní objekt, samotná mapa, do které dále vkládáme další objekty.

První parametrem konstruktoru tohoto objektu je přiřazení HTML bloku, ve kterém se má mapa vykreslit. V našem případě je to HTML tag div, kde má vlastnost id hodnotu map_canvas.

Druhým parametrem konstruktoru je pole, ve kterém jsou nastaveny jednotlivé vlastnosti. Ve výše zmiňovaném kódu jsou nastaveny vlastnosti:

- `zoom` na hodnotu 8. Pomocí zoom nastavíme úroveň zvětšení. Úroveň je možné volit v rozsahu 0-21, kde 0 značí nejmenší zvětšení a 21 naopak největší.
- `center` slouží pro nastavení geografické polohy středu mapy za pomocí výše zmiňovaného objektu google.maps.LatLng.
- `mapTypeId`, určuje typ zobrazené mapy. Google Maps API nabízí volbu mezi roadmap, satellite, hybrid a terrain. V našem případě je použit typ roadmap, který zobrazuje standardní 2D segmenty – dlaždice.

Je možné do pole přidat další nastavení jako je vypnutí nebo zapnutí navigace, Street View atd. Kompletní výčet těchto parametrů najdeme na adrese <http://code.google.com/intl/cs/apis/maps/documentation/javascript/reference.html#MapOptions>. Jednotlivé vlastnosti je možné za běhu aplikace dále modifikovat pomocí standardních settrů a gettrů pro daný typu mapového objektu.

`Objekt google.maps.Maker` je dalším významným prvkem. Marker neboli značka umožňuje vyznačit místa na mapě. Tomuto objektu je opět při vytvoření instance předáno pole s vlastnostmi. Mezi důležité vlastnosti patří zejména:

- `possition`, která udává polohu značky v mapě
- `map` přiřazuje značku na danou mapu. Toto je velmi důležitá vlastnost, protože naše webová aplikace může samozřejmě obsahovat i více map.

```
 <body onload="initialize()">
  <div id="map_canvas" style="width:100%; height:100%"></div>
 </body>
</html>
```

V tomto případě je poté funkce initialize() zavolána při načtení stránky. To nemusí ovšem být pravidlem. Je možné mapu vytvořit i při jiných událostech, které poskytuje technologie JavaScript.

Toto byl jen jednoduchý příklad, jak lze Google Maps API V3 použít při tvorbě vlastní webové aplikace. Na internetové adrese <http://code.google.com/intl/cs/apis/maps/documentation/javascript/reference.html> najdeme referenční manuál, kde je práce s touto technologií přehledně popsána a předvedena na vzorových příkladech se zdrojovými kódy.

2.8 Zobrazení více objektů

Při tvorbě složitější aplikace můžeme narazit na nepříjemný problém a to překryvu informací. Máme-li aplikaci, ve které nám tento jev nevadí, můžeme jednoduše použit z HTML vlastnost z-index pro vyskytující se objekty. V opačném případě je potřeba vyřešit problém překryvu informací jinými způsoby např. sdružováním objektů, které odstavení překryv objektů nebo vyhledat volnou pozici pro překrývající se objekt a na dané místo odkazovat pouze odkazem jako je šipka, čára atd.

2.8.1 Vlastnost z-index v HTML

Tato CSS vlastnost určuje tzv. výšku prvku na ose z, to jest vlastně priorita při překryvu prvků ve webové stránce. Tato vlastnost je funkční pouze u pozivovaných prvků, to znamená, že prvky musí mít nastavenu vlastnost possition na absolute, relative nebo fixed [8].

Vlastnost může běžně nabývat hodnot auto, kladného a záporného čísla. U hodnoty auto je vždy na hoře prvek, který je v kódu stránky později zapsán. Při kladném čísle platí, že čím vyšší je číslo, tím blíže je ke čtenáři a překrývá vzdálenější prvky. Pokud je použité záporné číslo, podsouvá se prvek pod ostatní vrstvy [8].

Velmi častá chyba pří vývoji webových aplikací je nastavení nízkého z-indexu prvku, který leží vespod a obsahuje další prvek, u kterého je nastaven vyšší z-index pro

překrytí všech prvků na stránce. Tento způsob nelze aplikovat, důvodem je částečná dědičnost vyššího prvku.

Vlastnost je velmi dobře podporovaná všemi prohlížeči. Ovšem je potřeba dát pozor na select, iframe a flashové prvky, jelikož tyto prvky nejsou vykreslovány prohlížečem, ale operačním systémem [8].

2.8.2 Sdružování objektů

Sdružování objektu, především markerů neboli značek, je potřeba aplikovat zejména v případě, kdy je na mapě zobrazeno velké množství informaci. Mapa se poté stává více přehlednou. Google Maps jsou rozděleny na jednotlivé rámce a bylo by možné sdružování značek aplikovat na jednotlivé rámce zvlášť. To by nám určitě umožnilo úsporu výkonu a tím zrychlení aplikace. Ovšem i tak by mohlo dojít k překryvu značek a proto je potřeba funkci sdružování použít na celou plochu mapy [9].

Při sdružování značek je potřeba si nejdříve ujasnit od jaké vzdáleností a jaké vyjádření vzdálenosti použít. Mohli bychom sdružovat skupinu značek na základě dané vzdálenosti a kontrolovat vzdálenost mezi jednotlivými značkami. Např. Kdybychom sdružili všechny značky v okruhu 10 km daného místa. U tohoto typu se vyskytuje problém s jednotkou km. Při zobrazení mapy na obrazovce u různých úrovní přiblížení je jednotka km nepřesná. U jednoho přiblížení by mohlo 100 pixelu znamenat 1 km, ale při oddálení mapy by 100 pixelů mohlo znamenat 100 km. Proto je tento převod nepoužitelný [9].

Existuje jen jediná jednotka pro vzdálenost, která řeší tento problém. Tou je vzdálenost pixelech při aktuální úrovni zvětšení. Jeden pixel na obrazovce je vždy jeden pixel a nemůže tedy dojít k nepřesnosti. Řekněme, že chceme sdružovat všechny značky v okruhu nějaké vzdálenosti. Vzdálenost nebo-li poloměr je vhodné určit podle velikosti plochy zobrazené značky. Vždy by mělo stačit zvolení delší vzdálenosti u výšky a šířky plochy [9] .

Vzdálenost mezi dvěma body, lze vypočítat několika způsoby, např. podle vzorců Haversine, Great Circle, Vincenty [9].

Haversineuv vzorec je velice důležitý u navigace. Poskytuje nejkratší možnou vzdálenost mezi dvěma body na povrchu koule z její délky a šířky. Vzorec vyhází z Haversinova zákonu, který patří do kategorie sférické trigonometrie. Týká se stran a úhlů kulového trojúhelníku. Tento způsob výpočtu vzdálenosti nám poskytuje přesnost kolem ± 2 až 3 metru na 1 kilometr.

Výpočet vzdálenosti je možný podle následující funkce:

```
function haversineDistance (lat1, lon1, lat2, lon2) {

      var latd = this.deg2rad(lat2 - lat1);
      var lond = this.deg2rad(lon2 - lon1);
      var a = sin(latd/2) * sin(latd / 2) +
                  cos(this.deg2rad(lat1)) * cos(this.deg2rad(lat2)) *
                  sin(lond / 2) * sin(lond / 2);
      var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
      return 6371.0 * c;
};

function deg2rad (angle) {
      return (angle/180)*Math.PI;
};
```

Nyní, když máme možnost výpočtu vzdálenosti mezi dvěma body, je potřeba ještě implementovat převod mezi kartézským souřadnicovým systémem na zeměpisnou šířku a délku. Převod lze provést pomocí těchto funkcí:

```
var radius = 85445659.4471
var offset = 268435456;

function lonToX (lon){

      return Math.round(offset + radius * lon * Math.PI / 180);
};


function latToY (lat) {
      return Math.round(offset -radius *
                        Math.log((1 + Math.sin(lat * Math.PI / 180)) /
                        (1 - Math.sin(lat * Math.PI / 180))) / 2);
};
```

Poté, už můžeme jednoduše vypočítat vzdálenost mezi jednotlivými zeměpisnými šířkami a délkami podle následující funkce:

```
function pixelDistance (lat1, lon1, lat2, lon2, zoom){
      var x1 = this.lonToX(lon1);
      var y1 = this.latToY(lat1);

      var x2 = this.lonToX(lon2);
      var y2 = this.latToY(lat2);

      return Math.sqrt(Math.pow((x1-x2), 2) + Math.pow((y1 - y2), 2))
                                    >> (21 - zoom);
};
```

Nyní máme k dispozici vše potřebné pro sdružování značek v Google Maps. Pokud máme k dispozici pole všech vyskytujících se značek v mapě, známe minimální přípustnou vzdálenost a aktuální úroveň zvětšení je možné sdružit značky, které mezi sebou mají menší vzdálenost než je přípustná. Pro zlepšení přesnosti zobrazení je možné u sdružených značek vypočítat aritmetický průměr zeměpisné šířky a délky [9].

Funkce pro sdružování objektů může mít například tuto podobu:

```
function cluster(markers, distance, zoom){
      var clustered = new Array();

      while (markers.length) {
            var marker = markers.pop();

            while (marker == null) {
                  marker = markers.pop();
                  if (markers.length == 0) {
                        return clustered;
                  }
            }
            var cluster = new Array();

            for ( var i = 0; i < markers.length; i++) {
                  if (markers[i] != null || markers[i] != undefined) {
                        var pixels = pixelDistance(marker['lat'],
                  marker['lon'],
                                    markers[i]['lat'], markers[i]['lon'],
                                    zoom);
                        if (distance > pixels) {
                              cluster.push(markers[i]);
                              markers[i] = null;
                        }
                  }
            }

            if (cluster.length > 0) {
                  cluster.push(marker);
                  clustered.push(cluster);
            }
            else {
                  clustered.push(marker);
            }
      }
      return clustered;
};
```

2.8.3 Hledání neobsazené pozice

Tato metoda opět umožňuje odstranit překryv objektů v mapě a obecně lze tuto metodu využít i v jiném případě, kde je nežádoucí překryv objektů. Příkladem může být u map informační okno k příslušné značce. Pokud ošetříme aplikaci, aby bylo možné v daný okamžik zobrazit pouze jedno informační okno je tento problém vyřešen. Pokud je ovšem potřeba z nějakého důvodu zobrazit více informačních oken zároveň je dobré využít tuto metodu.

Metoda spočívá v hledání nového a neobsazeného místa na obrazovce. Postupným procházením všech zobrazených objektů provádíme kontrolu pozice jednotlivých objektů.

Kontrolu můžeme provést pomocí následující podmínky:

```
function CheckObjektPossition(first, second) {
	if((second.offsetTop <= (first.offsetTop + first.clientHeight) &&
	((second.offsetTop + second.clientHeight) >= first.offsetTop))
	&&
	(second.offsetLeft <= (first.offsetLeft + first.clientWidth) &&
	((second.offsetLeft + second.clientWidth) >= first.offsetLeft))
	)
	{
		return false;
	}
	else
	{
		return true;
	}
}
```

Kde je: `offsetTop` odsazení od vrchního okraje mapy

`offsetLeft` odsazení od levého okraje mapy

`clientWidth` šířka mapy

`clientHeight` výška mapy

Jestliže této podmínce vyhovují předané objekty, pak se tyto objekty překrývají a je potřeba najít jinou pozici pro objekt. Jednou z možností hledání nové polohy je pohyb po opsané kružnici objektu. Pohybu lze docílit průchodem všech 360° kružnice. Pro každý ° je dána nová pozice v ose x:

```
var x = centerX + radius * Math.cos(i * Math.PI / 180);
```

v ose y:

```
var y = centerY + radius * Math.sin(i * Math.PI / 180);
```

kde je: `centerX` pozice středu v ose x

`centerY` pozice středu v ose y

`radius` poloměr opsané kružnice

Po každém výpočtu souřadnic je třeba kontrolovat vzájemnou polohu objektů podle výše uvedené podmínky. Jestliže, poloha vyhovuje je možné průchod ° ukončit a tyto souřadnice pro objekt určit za konečné.

Jestliže, po průchodu 360° nebyla nalezena vyhovující poloha je třeba zvětšit poloměr prohledávané oblasti o daný přírůstek a výpočet opakovat.

U tohoto algoritmu je možné ovlivnit přesnost pomocí přírůstku ° a přírůstku poloměru, kde platí, že čím menší přírůstek je, tím větší je přesnost výsledné polohy. Nutno dodat, že při aplikování této metody je potřeba brát v úvahu zatížení systému. Platí,

že čím vetší přesnost zvolíme, tím větší je zatížení systému, které má vliv na rychlost naší aplikace. Obzvlášť při tvorbě webových aplikací je třeba dbát na rychlost a zatížení systému.

2.8.4 Shrnutí

Využitím vlastnosti HTML z-index, lze do jisté míry vyřešit přeryv informací v Google Maps. Tuto metodu bych spíše doporučil využít v případě, že zobrazených objektů není mnoho a zároveň není nutné se tímto problémem zabývat.

Sdružováním zobrazených objektů, především značek, lze velmi efektivně zpřehlednit informace v mapě. Je třeba si ale uvědomit, že přehlednosti docílíme na úkor ztráty nebo redukce zobrazených informací.

Výhodou metody hledání nové polohy objetu nedochází ke ztrátě zobrazených informací, ale při větším množství objektů v dané oblasti může dojít k velmi nepřehledné situaci.

Pokud jsme schopni pro každý typ objektu určit nejvhodnější metodu, dosáhneme velmi přehledného zobrazení informací v internetové mapě.

3 INFORMAČNÍ TABULE

Informační tabule obecně slouží k poskytování nejrůznějších informací. Tento typ prezentace informací může být použit u dopravy, reklamy, sportovních výsledkových tabulí, poutačů ve formě světelných informačních tabulí, projekce na velkoplošné obrazovce, billboardů, stojanů, nástěnek atd. Vždy jsou, ale vytvořeny pro stejný cíl, přitáhnout očí diváka na zobrazené informací, především u reklamy.

Nejčastěji jsou využívaný technologie tekutých krystalů (LCD) nebo svítivých diod (LED).

3.1 Technologie

3.1.1 LCD

LCD technologie umožňuje zobrazení informací na tenké a ploché zobrazovací zařízení. Zařízení se skládá z omezeného počtu barevných nebo monochromatických pixelů, které jsou seřazeny před zdrojem světla nebo reflektorem. Přičemž počet pixelů je omezen velikostí monitoru. Tato technologie vyžaduje poměrně velmi malé množství elektrické energie, která se liší v závislosti na jasu [10].

Barevné LCD displeje mají jednotlivé pixely rozděleny do tří subpixelů, tj. červeného, zeleného a modrého. Z toho tedy vyplivá, že technologie využívá aditivní způsob míchání barev (RGB). Díky možné kontrole a úpravy svítivosti jednotlivých pixelů je možné dosáhnout milionů barev. Barevné subpixely neboli složky lze sestavit v různých geometriích v závislostní na zobrazovacím zařízení. V případě, že software zná geometrii monitoru, máme možnost zvýšit viditelné rozlišení. Toto oceníme především při vyhlazování písma [10].

3.1.2 LED

Tato technologie využívá LED diod, které jsou umístěny vedle sebe po celé výšce a délce. Jednotlivé obrazce se potom zobrazují pomocí zapínáním příslušných diod.

LED dioda je elektronická polovodičová součástka obsahující přechod P-N, tj. propouští elektrický proud pouze v jednom směru. Pásmo spektra zářící diody závisí na chemickém složení použitého polovodiče. LED diody mohou být vyráběny s pásmy vyzařovaní do skoro ultrafialových po infračervené pásmo. Z principu funkce LED, že není možné emitovat přímo bíle světlo. K dosažení bílého světla se používá trojce čipů, pomocí kterých pak dochází ke vjemu bílého světla. Regulace vyzařující ho světla je možné dosáhnout pomocí podřadného odporu, který je zapojen v sérii s LED diodou. Obecně totiž platí, že čím více proudu, tím více světla [11].

3.2 Informační tabule v dopravě

V dopravě se můžeme běžně setkat s dvěma druhy informačních tabulí a u časté reklamy a zobrazením informací o spojích. V této podkapitole se budeme především zabývat zobrazením informací o spojích.

3.2.1 Informace o spojích

Využití informačních tabulí při zobrazení informací o spojích se můžeme setkat převáženě u hromadné osobní dopravy, tj. železniční, autobusové, městské atd. Každý se určitě setkal s informačními tabulemi na vlakovém nádraží ČD, v posledních letech se tyto tabule využívají alespoň na autobusových nebo trolejbusových nádražích ve větších městech. Ovšem mohou se vyskytovat i na jednotlivých zastávkách.

3.3 Informační tabule ve stanicích ČD

Tento podnik využívá obou zobrazovacích technologií, tj. LED diod a LCD. U většiny případů se setkáme spíše s LED tabulemi, je to především proto, že tato technologie se běžně využívá několik let zpět, kdežto zobrazení informací pomocí LCD monitorů je záležitostí posledních let.

Obsah zobrazených informací je ovšem téměř identický. Obsahují informace o nejbližších možných a předpokládaných spojích, které ve stanici staví. Informační tabule obsahují:

- označení typu vlaku. V dnešní době české dráhy nabízejí několik typů, např. osobní vlak (OS), rychlík (R), expresní vlak (EX), intercity (IC), procity (EC) a supercity (SC). Jednotlivé typy se vyznačují především komfortem, který nabízí cestujícím a stanicemi, ve kterých vlak staví. Dále by bylo možné typy rozdělit na vnitrostátní a mezinárodní.
- směr vlaku, tj. do jaké stanice vlak míří
- výčet stanic, ve kterých daný vlak staví
- pravidelný příjezd či odjezd
- předpokládané zpoždění

3.4 Informační tabule v městské hromadné dopravě

I zde najdou informační tabule svůj důvod k umístění na jednotlivých zastávkách. Tabule jsou většinou umístěny na zastávkách, kde je projíždí větší počet dopravních prostředků, zpravidla to bývají autobusové a trolejbusové nádraží.

Tyto tabule zobrazují vedle jízdních řádů, také nejbližší očekávané spoje pro příslušnou zastávku. Zde jsou nejčastěji zobrazeny informace směru linky, označení

zastávky a přibližném odjezdu. Zde není příliš nutné počítat se zpožděním, protože zpoždění linek je zanedbatelné. Důvodem jsou kratší vzdálenosti a větší intenzita odjezdů daných spojů.

4 INFORMAČNÍ TABULE PRO CESTUJÍCÍ SE ZOBRAZENÍM AKTUÁLNÍ POLOHY SPOJE

4.1 Požadavky

Na informační tabuli pro cestující se zobrazením polohy jsou zejména kladeny tyto požadavky:

- bezobslužné zobrazení příjezdů a odjezdů všech očekávaných a opožděných spojů během určitého času (například 15, 30, 60, … minut)
- dostupnost služby v místě železniční nebo autobusové zastávky
- přehlednost a jednoduchost zobrazených dat
- administrační rozhraní, které umožní spravování vozidel, zastávek, jízdních řádů, poskytovaných služeb atd.

4.1.1 Bezobslužné zobrazení příjezdů a odjezdů spojů

Poskytnutí bezobslužné obsluhy je velice důležité, protože by bylo neúnosné mít např. v každé zastávce administrátora, který by jednotlivé spoje řadil a třídil podle jízdních řádů. Proto se nabízí využití moderní technologie, která je ve výsledku levnější na náklady a v provozu do jisté míry spolehlivější.

4.1.2 Dostupnost služby

Je potřeba navrhnout řešení, které umožní jednoduché nainstalování této technologie a poté také jednoduchá správa systému. Je třeba dbát na viditelnost základních informací i pro osoby se sníženou kvalitou zraku.

4.1.3 Přehlednost a jednoduchost zobrazených dat

Jednotlivá data spojů musí být přehledně a přesně zobrazena. Umožnění zobrazení aktuální polohy vozidla.

4.1.4 Administrační rozhraní

Aplikace musí umožnit jednoduchou a přehlednou správu o jednotlivých prvcích v dopravě. V administračním rozhraní by mělo být umožněno vytvoření, upravení a odebíraní linek, stanic, vozidel, jízdních řádů, služeb pro jednotlivé stanice. Dále by mělo být umožněno variabilní nastavení, tj. doba očekávaných spojů atd.

4.2 Použité technologie

4.2.1 Webová aplikace

Webová aplikace je forma aplikace, která je webovým serverem poskytována přes počítačovou síť Internet či Intranet. Intranet je obdoba Internetu pro vnitropodnikovou síť. Tyto aplikace jsou v dnešní době stale více a více používané díky jejich flexibilním vlastnostem. Jednou z nejvýznamnějších vlastností je při dodržení celosvětově uznávaných norem a standardů možnost jejich spuštění na libovolném operačním systému prostřednictvím webového prohlížeče. Díky této vlastnosti se nabízí možnost spuštění aplikace na jakémkoliv místě ve světě, kde je přístupné připojení k internetu[12].

Webová aplikace je založená na typu síťové architektury klient-setver. Tato architektura je rozdělena na stranu klienta a serveru. Klientská strana je často grafickou aplikací s uživatelským rozhraním, která předává, žádá a poté prezentuje, data ze serverové strany. Komunikace probíhá pomocí internetových protokolů, např. HTTP, SMTP, Telnet, DNS, atd. Komunikace klienta může probíhat s více připojenými servery a naopak. Serverové strany je možné rozdělit na webové, databázové a e-mailové servry[12].

Webové aplikace je možné rozdělit na statické a dynamické. Statické webové aplikace se většiny případu vyznačují svojí jednoduchostí a menšími možnostmi modifikace. Tyto aplikace bývají často použity pro webové prezentace firem, nadací, organizací bez větších nároků na měnící se obsah. U tohoto typu web server předá klientovy statický obsah. Naopak dynamické webové aplikace vyžadují mnohem větší znalost o tvorbě webových prezentací. Zpracování obsahu stránky je provedeno na web serveru společně s databázovým serverem a výsledek je odeslán na klientskou stranu. Tento typ aplikace např. reprezentují internetové obchody, rezervační a redakční systémy [12].

4.2.2 Oracle 10g

Oracle je systémem pro řízení baze dat, tj. softwarové vybavení, které zajišťuje práci s databází, neboli vytváří rozhraní mezi aplikačními programy a uloženými daty. Oficiálním názvem databázové platformy je Oracle Databáze. Tento produkt je od společnosti Oracle Corporation. Aktuální verzí této platformy je Oracle 11g [13].

Historie

V roce 1977 založil Lawrence J. Ellison, Robert N. Miner a Edward Oates firmu SDL. Firma tehdy pracovala na vývoji relačních databází podle teorie doktora E.F Codda [13].

Oracle byl kódový nazev projektu, který společnost SDL vytvářela na zakázku pro CIA. Tato organizace tehdy potřebovala shromažďovat velké množství dat a rychle v nich podle požadavků vyhledávat [13].

V roce 1978 se společnost SDL přejmenovala na Relational Software Inc. V tomto roce také vznikla první verze databázové platformy Oracle. Tento software byl vytvořen v asembleru pro počítač PDP=11 s operačním systémem RSX [13]..

V roce 1980 došlo k přejmenování společnosti, tentokrát na Oracle Corporation. Vznikla také nová verze s označením Oracle 2.0. U této verze byl problém portability na jiné hardwarové platformy a proto bylo strategicky rozhodnuto, že další verze produktu budou již tvořeny v jazyce C [13]..

V roce 1983 byla vydána verze Oracle 3.0. Tato verze již byla napsána v jazyce C a dala se portovat na úrovni zdrojového kódu. Zdrojový kód tedy stačilo přeložit překladačem pro příslušnou hardwarovou platformu [13]..

Do verze Oracle 10g tento systém prošel značným vývojem kupředu. Byla vylepšena komunikace mezi více servery při zachování konzistentního čtení dat a podpora pro všechny významnější hardwarové platformy. Bylo také umožněno budování velkých transakčních systémů, kde více počítačů sdílelo společný diskový prostor. To umožnilo budovat velmi rozsáhlé databáze a datové sklady. V roce 1997 byl integrován jazyk Java pro aplikace klient-server. Významnou verzí tohoto systému Oracle 8i a 9i, kde písmeno i symbolizuje kompletní orientaci této databázové platformy na prostředí internetu [13].

V roce 2003 byla vydána první verze Oracle 10g. Tato verze byla rozšířena o funkci RAC. Tato funkce umožnila připojení a otevření spojení s databází více než jedné instanci. Došlo také vylepšení ovladatelnosti, výkonu, škálovatelnosti, pohybu dat pro export a import. Verze také byla obohacena o ASM, AWR, ADD, plánovač – DBMS_SCHEDULER, který je více propracovaný pro plánování úloh, DBMS_FILE_TRANSFER pro přenos databázových souborů a ASM. V roce 2005 došlo k vylepšení této verze o transparentní šifrování, asynchronní potvrzení změny dat, šifrování db odkazů a vytvořena obslužná aplikace pro ASM [13].

V roce 2009 byla vydána aktuální verze Oracle 11g. Tato verze nabízí ještě lepší kompresní poměr, možnost aktualizace, zatím co uživatele jsou stále připojeni. Došlo také ke zlepšení automatizace pro řízení systému.[4]

Relační databáze

Relační databáze či relační systém řízení baze dat lze jednoduše definovat jako sadu nástrojů pro efektivní a spolehlivé ukládání dat a pro manipulaci s nimi [13].

Databáze

Pod databází si může představit skladiště informací, ve kterém se nachází vlastnosti osob, produktů, služeb a čehokoliv jiného. V dnešní době většina lidí nevědomě s databází pracuje pomocí vyhledávače, kde si čtou internetové zpravodajství, hledají různé informace, provádí platební transakce přes Internet banking atd. Databáze může vzniknout jako seznam v textovém editoru nebo tabulkovém preprocesoru. Data ve formátu seznamu se s přibývajícími daty stávají více a více nepřehlednými a je složité se v nich orientovat

při tvorbě podmnožin. Relační databáze je z toho ohledu více flexibilní. Pojem databáze zahrnuje data a nástroje zajišťující jejich ukládání a manipulaci s těmito daty. Tímto pojmem je spojen pojem databázový server, který představuje soubor programových prostředků určených pro s daty, včetně organizace a realizace přístupu klientů k těmto datům. V relačních databázích jsou data uložena ve formě klasických tabulek, kde řádek představuje kombinaci hodnot sloupců v tabulce a je označen jako jeden záznam. Každý řádek by měl být definovaný klíčem. V tabulce se vyskytují dva typy klíčů a to primární a cizí. Pomocí primárního klíče jsme schopni jednoznačně určit jeden záznam v tabulce a definovat relace mezi jednotlivými tabulkami. Cizí klíč je sloupec či skupina sloupců, které odkazují na primární klíč v jiné tabulce [13].

Relace

Relace mezi tabulkami popisují vztahy mezi objekty reálného světa, které jsou reprezentovány tabulkami. Při návrhu databáze můžeme využít relace typu jedna ku jedné, jedna ku více, více ku více a unární relaci [13].

Relaci typu jedna ku jedné lez popsat touto definicí: každý řádek primární tabulky lze svázat s právě jedním řádkem sekundární tabulky. V reálném světě lze tuto relaci vysvětlit jako vztah mezi řidičem a automobilem. Pouze jeden řidič může v daném okamžiku řídit jeden automobil a právě jeden automobil může být v daném okamžiku řízen pouze jedním řidičem [13].

U relace jedna ku více musí být svázán jeden řádek primární tabulky s jedním či více řádky sekundární tabulky. Příkladem může být vztah mezi autobusem a cestujícím. V autobuse může být v jednom okamžiku jeden a více cestujících, kdežto každý cestující může být v daném okamžiku pouze v jednom autobuse [13].

Relace více ku více lze definovat jako, že více řádků primární tabulky je svázáno s více řádky sekundární tabulky. Praxi se tento vztah realizuje pomocí spojovací tabulky, což znamená, že vztah rozdělíme na dva vztahy jedna ku více [13].

Unární relace je vztah jedné tabulky se sama sebou. Pomocí tohoto vztahu často vyjadřujeme vztah typu nadřazený – podřízený. Sloupec tabulky může obsahovat vazbu na primární klíč jeho nejbližšího nadřazeného. Jeli jedno pole v tomto sloupci prázdné, jedná se o nejvyššího nadřazeného. Příkladem může být vytvoření rodokmenu, který získáme pomocí relací mezi jednotlivými lidmi v tabulce [13].

4.2.3 PHP

PHP je skriptovací programovací jazyk, který je určený pro programování dynamických webových aplikací. Tento programovací jazyk lze také využít i k tvorbě konzolových a desktopových aplikací. Kód je začleněn přímo do struktury kódu HTML či XHTML jazyka. PHP skripty jsou prováděny na straně serveru a na klientskou stranu je přenášen pouze výsledek jejich činnosti. PHP jazyk je nezávislí na platformě a proto je možné skripty přeložit na mnoha operačních systémech [14].

Historie

PHP bylo původně označením pro Personál Home Page, v překladu osobní domácí stránky [14].

Zakladatelem PHP je dánsko/grónský programátor Rasmus Lerdorf, který napsal v roce 1994 binární část CGI v programovacím jazyku C. Zpočátku vytvořil tento nástroj pro vytvoření vlastních osobních domácích stránek. Důvodem tehdy bylo zaznamenávání návštěvnosti jeho stránek. Nejprve byl tento nástroj napsán v perlu, ale ještě tentýž rok byla tato technologie spojena s Form Interpretr a tak vznikla kombinace PHP/FI. PHP/FI obsahovala širokou implementaci pro programovací jazyk C a komunikaci s databázemi, což umožnilo první tvoření webových aplikací [14].

Aby Rasmus Lerdorf odhalil co nejvíce chyb a poté došlo ke zdokonalení kódu, bylo PHP 8.června 1995 veřejně vydáno. Po zdokonalení kódu byla vydána nová verze PHP 2, která zahrnovala proměnné, zpracování formulářů a začlenění HTML kódu [14].

Ukázka kódu „Ahoj, světe!"

```
<? php
      echo "ahoj, světe!";
?>
```

PHP se stalo velmi oblíbeným především díky svému jednoduchému použití. Přidáme-li k tomuto jazyku databázový a webový server je možné velmi snadno vytvářet dynamické webové aplikace [14].

Mezi výhody PHP patří:

- velmi podobná syntaxe s jazykem C
- multiplatformost
- velká podpora webhostingových poskytovatelů
- otevřenost a jednoduchost projektu s rozsáhlou podporou komunity
- snadná komunikace s databázemi jako je Oracle, MySQL, PostgreSQL atd.

4.2.4 Zend Frameworrk

Zend Framework je opensource PHP MVC Framework od společnosti Zend Technologies Ltd. Společnost nabízí i jiné produkty, např. Zend Srver, Zend Studio, Zend Guard atd. Knihovny toho frameworku výrazně usnadňují práci tvůrcům webových aplikací implementovaných v PHP. Z důvodu zabezpečení a výkonu je doporučeno používat verzi PHP 5.2.3 nebo novější. V této době je aktuální verzí Zend Framework 1.10.7 Released, která je volně stažitelná na oficiálních stránkách <http://devzone.zend.com/article/12367-Zend-Framework-1.10.7-Released>. Tato verze, už posedmé upravuje sérii 1.10. V příštích týdnech by ovšem měly být zveřejněny další

informace o vydání verze 2, kde aktuálně probíhají poslední úpravy. Při každém vývoji je kladen důraz především na:

- vytvoření standardu při tvorbě pokročilejších PHP aplikací
- poskytnout vývojářům vysoce kvalitní nástroje
- vytvořit stabilní a bezchybný framework
- vytvořit štábní kulturu
- zajisti zpětnou kompatibilitu a přehlednou dokumentaci

Licence

Zend framework patří pod novou BSD licenci, což znamená, že použití toho frameworku nemusíte platit žádné poplatky i v případě komerčního využití. Framework je možné dále rozšiřovat a však vaše nové třídy nesmějí začínat předponou Zend [15].

Návrhový vzor MVC

MVC návrhový vzor je doporučený vrstvení při tvorbě aplikace. Říká nám jak co nejjednodušeji a nejefektivněji vytvářet aplikace, aby jí bylo možné jednoduše rozšiřovat, testovat a neobsahoval chyby [15].

Model MVC je rozdělen na:

1. M – Model je část aplikace, která vytváří hlavní logiku a ovládá databázi prostřednictvím adaptéru, v podstatě je to jádro aplikace.
2. V – View přebírá data od modelu a prezentuje je uživateli
3. C – Controller zpracovává vstup uživatele a předává ho modelu

Bootstrap

Každá aplikace vytvořená v Zend Frameworku má jeden vstupní bod a tím je právě Bootstrap. Bootstrap je PHP soubor, nejčastěji jde o index.php, který obsluhuje všechny požadavky prohlížeče. Měl by to být také jediný soubor, který je přístupný přímo přes webové rozhraní. Boostrap vytváří například spojení s databází, připojuje soubory, definuje funkce pro autoload atd [15].

Přehled základních knihoven:

- Zend_Controller – nejdůležitější knihovna pro vytváření MVC aplikace, která následně umožňuje používat SEO url, jedná se o aplikaci návrhového vzoru `ForntController`.
- Zend_Db – jedna z nejdůležitějších knihoven. Umožňuje pracovat se všemi podporovanými databázemi pomocí jediného rozhraní. Třída obsahuje adaptery pro jednotlivé databáze, napr. MySQL, Oracle atd.

- Zend_Form – pomocí této třídy můžeme jednoduše nadefinovat formuláře, např. určit povinné položky, validátory a jejich chybové hlášky, filtry pro jednotlivé položky. Po té jedním příkazem formulář zobrazit nebo zpracovat.
- Zend_Gdata – tato třída je jakým si rozhraním pro prací se službami, které nabízí Gogole.
- Zend_Layout – tato knihovna dalším významným prvkem frameworku. S touto knihovnou lze velmi snadno vytvářet a měnit layout webové aplikace.
- Zend_Acl – třída umožňuje spravovat přístup jednotlivých uživatelů k jednotlivým sekcím podle nastavení uživatelských práv. Pomocí pluginů je možné celou autorizaci zautomatizovat.
- Zend_Auth – tato třída umožňuje přihlašovaní a odhlašování uživatelů a zároveň udržení informací pomocí sessions. Pro přihlašování obsahuje několik vestavěných adaptérů.
- Zend_Config – tato třída umožňuje spravovat nastavení v souborech .ini či .xml.
- Zend_Currency – třída pro jednoduché formátování měny.
- Zend_Date – jednoduché API pro práci s datem a časem v nejrůznějších formátech.
- Zend_Exception – Základní třída, od která jsou odvozené ostatní výjimky frameworku.
- Zend_Pdf – třída pro jednoduché pracování s PDF dokumenty.

4.2.5 JavaScript

JavaScript je objektově orientovaný programovací jazyk vyvinuty společností Netscape a Sun Microsystems. Tento to interpretovaný programovací jazyk se používá při tvorbě webových aplikací. Napsané skripty pracují na straně klienta a proto je tedy možné bezprostředně reagovat na uživatelovy akce v prohlížeči. Napsané skripty jsou staženy ze serverové strany společně s obsahem stránky do prohlížeče a poté spuštěn, ovšem je také možné spustit skripty na straně serveru [17].

4.2.6 AJAX

AJAX je obecné ozančení pro technologii vývoje interaktivních aplikací, které mění obsah stránek bez nutnosti jejich znovu načítaní. Tyto aplikace jsou v poslední době velmi oblíbené a vyžadují podporu moderních prohlížečů [18].

Tato technologie využívá jiných technologií, jako jsou prezentace HTML nebo XHTML, DOM, JavaScriptm, XMLHttpRequest pro asynchronní výměnu dat s webovým serverem [18].

4.3 Role uživatelů

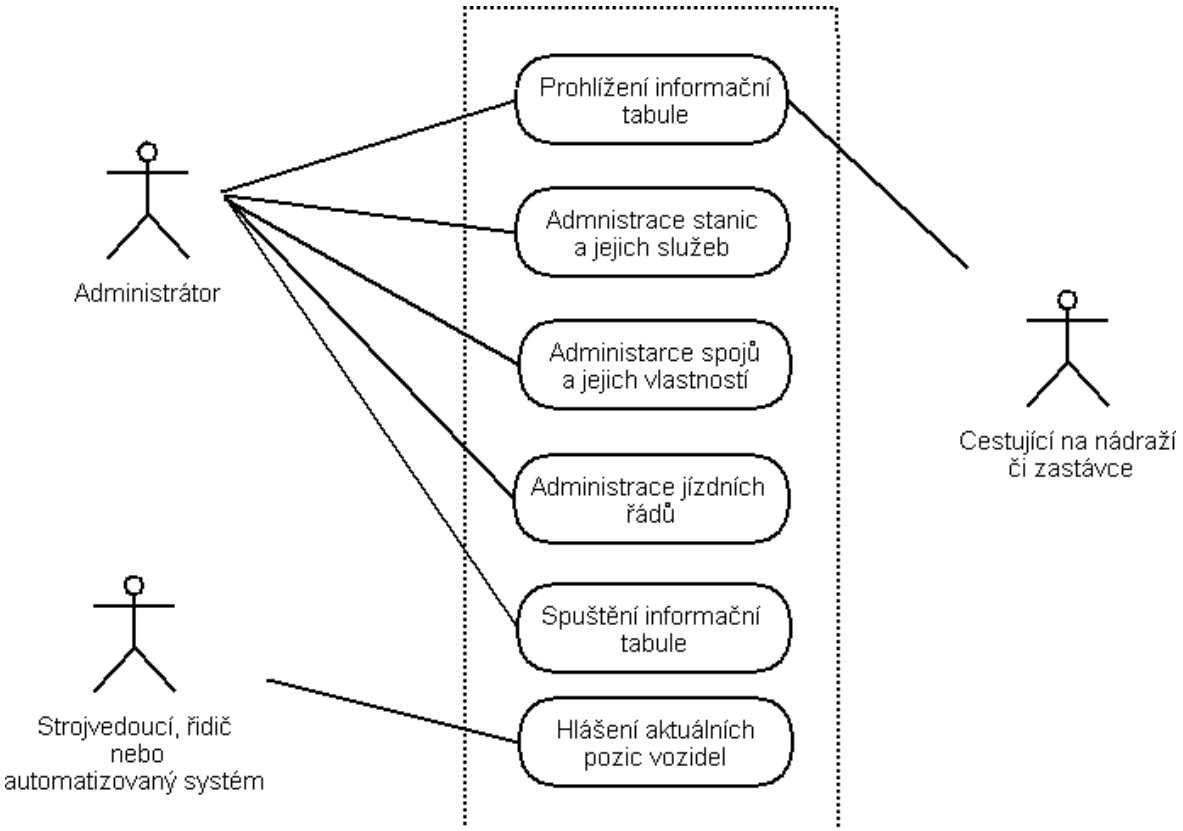
V informační tabuli pro cestující jsou požadovány celkem tři uživatelské role.

První rolí jsou vozidla či jejich strojvedoucí nebo řidiči. Tato role spočívá k hlášení aktuálních poloh pohybujících se vozidel. Zpráva o poloze musí obsahovat datum a čas, identifikaci spoje a GPS souřadnice.

Druhou rolí je administrátor, který pomocí administračního rozhraní zpravuje data o stanicích, spojích a jízdních řádech. Dále má možnost vygenerování informačních tabulí dané stanice.

Třetí rolí jsou cestující na nádražích či zastávkách, kterým jsou data o poloze vozidla prezentována.

Možnosti jednotlivých rolí uživatelů jsou zobrazeny v následujícím obrázku, který prezentuje Use case diagram.

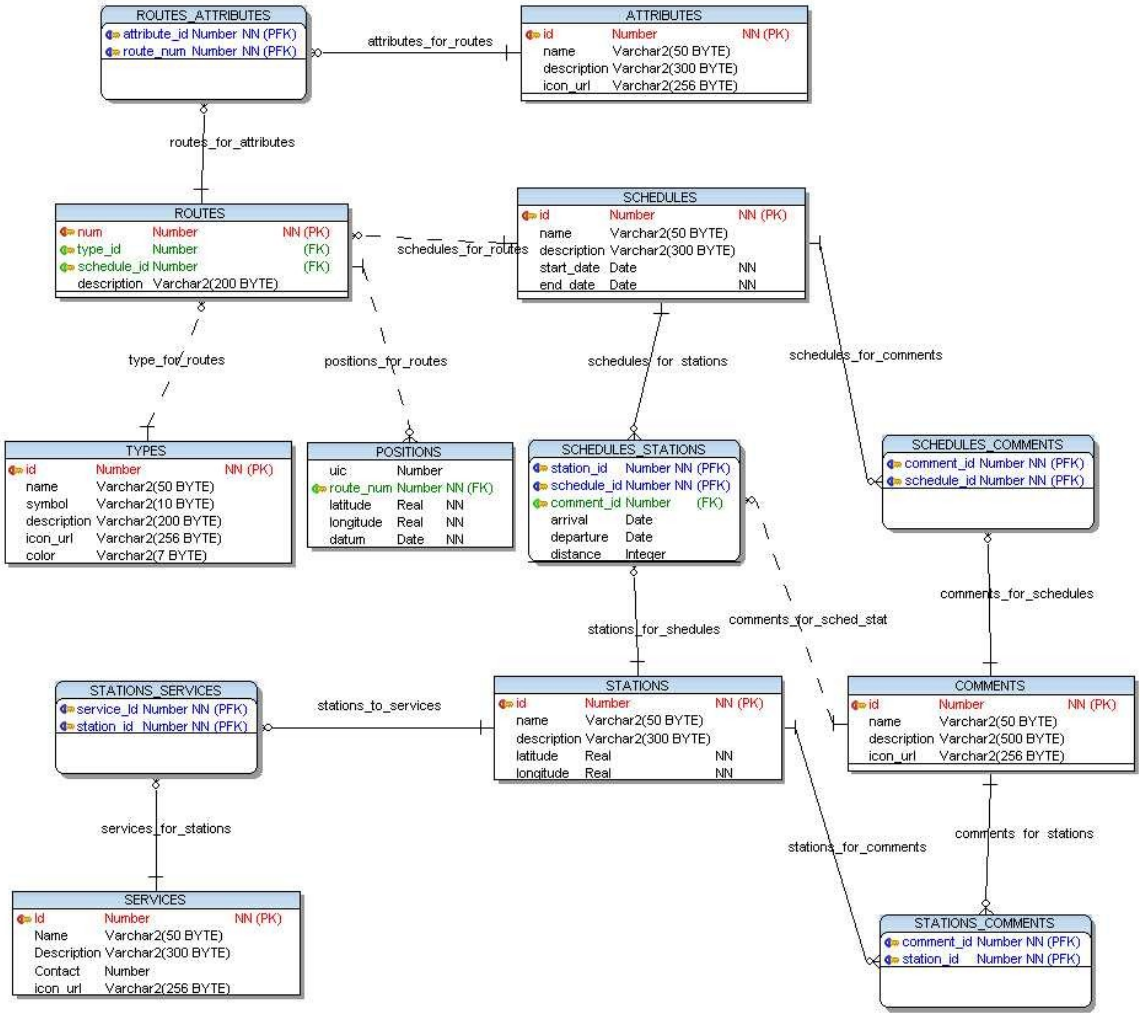


Obrázek 4.1 - Use case diagram

4.4 Návrh databáze

Při navržení databáze byl kladen důraz, aby splňovala třetí normální formy a zároveň byla zachována integrita a bezpečnost uložených dat.

4.4.1 E-R diagram



Obrázek 4.2 - E-R diagram

4.4.2 Popis tabulek a jejich atributů

Routes - linky

Tabulka routes obsahuje data jednotlivých linek. Každé lince může být přidělen jeden typ linky. Jedna linka může mít několik pozic. Každé lince může být přiřazen jeden jízdní řád. Každá linka muže mít několik vlastností. V tabulce 4.1 jsou popsány jednotlivé atributy této tabulky.

Tabulka 4.1 - Popis tabulky routes

Routes		
atribut	**datový typ**	**popis**
num	Number NOT NULL	PK - primární klíč
type_id	Number	FK - cizí klíč tabulky types
chedule_id	Number	FK - cizí klíč tabulky schedules
description	Varchar2(200 BYTE)	popis linky

Types – typy

Do tabulky types jsou ukládány data o jednotlivých typech linek. Každý typ může být přidělen několika linkám. V tabulce 4.2 jsou popsány jednotlivé atributy této tabulky.

Tabulka 4.2 - Popis tabulky types

Types		
atribut	**datový typ**	**popis**
id	Number NOT NULL	PK - primární klíč
name	Varchar2(50 BYTE)	název typu
symbol	Varchar2(10 BYTE)	symbol (zkratka) typu
description	Varchar2(300 BYTE)	popis typu
icon_url	Varchar2(256 BYTE)	url uložené ikony pro zobrazení
color	Varchar2(7 BYTE)	reprezentující barva v hex tvaru

Attributes – vlastnosti

Do tabulky attributes jsou ukládány data o jednotlivých vlastnostech linek. Každá vlastnost může být přiřazená několika linkám. V tabulce 4.3 jsou popsány jednotlivé atributy této tabulky.

Tabulka 4.3 - Popis tabulky attributes

Attributes		
atribut	**datový typ**	**popis**
id	Number NOT NULL	PK - primární klíč
name	Varchar2(50 BYTE)	název vlastnosti
description	Varchar2(300 BYTE)	popis vlastnosti
icon_url	Varchar2(256 BYTE)	url uložené ikony pro zobrazení

Routes_Attributes – vlastnosti linek

Do tabulky routes_attributes jsou ukládány data o tom, která vlastnosti jsou přiřazeny k jednotlivým linkám. V tabulce 4.4 jsou popsány jednotlivé atributy této tabulky.

Tabulka 4.4 - Popis tabulky routes_attributes

Routes_attributes		
atribut	**datový typ**	**popis**
attribute_id	Number NOT NULL	PFK - Primární cizí klíč z tabulky attributes
route_num	Number NOT NULL	PFK - Primární cizí klíč z tabulky routes

Possitions – pozice vozidel

Do tabulky possitions jsou ukládány data o pozicích, které do systémů hlásí v pravidelných intervalech vozidla. Každá pozice musí patřit jediné lince. V tabulce 4.5 jsou popsány jednotlivé atributy této tabulky.

Tabulka 4.5 - Popis tabulky possitions

Possitions		
atribut	**datový typ**	**popis**
uic	Number	identifikační číslo vozidla
route_num	Number NOT NULL	číslo linky
latitude	Real NOT NULL	zeměpisná šířka polohy vozidla
longitude	Real NOT NULL	zeměpisná délka polohy vozidla
datum	Date NOT NULL	datum nahlášení polohy

Schedules – jízdní řády

Do tabulky schedules jsou ukládány data o jízdních řádech. Každý jízdní řád může být přiřazen několika linkám. Každý jízdní řád může mít nerozvrhováno několik stanic. Každý jízdní řád může mít přiděleno několik komentářů. V tabulce 4.6 jsou popsány jednotlivé atributy této tabulky.

Tabulka 4.6 - Popis tabulky schedules

Schedules		
atribut	**datový typ**	**popis**
id	Number NOT NULL	PK - primární klíč
name	Varchar2(50 BYTE)	název jízdního řádu
description	Varchar2(300 BYTE)	popis jízdního řádu
start_date	Date NOT NULL	datum od kdy je jízdní řád platný
end_date	Date	datum ukončení platnosti jízdního řádu

Comments – komentáře

Do tabulky comments jsou ukládány data o komentářích. Každý komentář může být přidělen několika jízdním řádům, nerozvrhované položce v jízdním řádu nebo stanicím. V tabulce 4.7 jsou popsány jednotlivé atributy této tabulky.

Tabulka 4.7 - Popis tabulky comments

Comments		
atribut	**datový typ**	**popis**
id	Number NOT NULL	PK - primární klíč
name	Varchar(50 BYTE)	nazev komentáře
description	Varchar(300 BYTE)	popis komentáře
icon_url	Varchar2(50 BYTE)	url uložené ikony pro zobrazení

Schedules_comments – komentáře jízdních řádů

Do tabulky schedule_comments jsou ukládány data o tom, které komentáře mají jízdní řády přiřazeny. V tabulce 4.8 jsou popsány jednotlivé atributy této tabulky.

Tabulka 4.8 - Popis tabulky schedules_comments

Schedules_comments		
atribut	**datový typ**	**popis**
comment_id	Number NOT NULL	PFK - Primární cizí klíč z tabulky comments
schedules_id	Number NOT NULL	PFK - Primární cizí klíč z tabulky schedules

Stations – stanice, zastávky

Do tabulky stations jsou ukládány data o stanicích. Každá stanice může mít několik komentářů. Každá stanice může být nerozvrhována v několika jízdních řádech. Každá stanice může mít přiřazeno několik služeb. V tabulce 4.9 jsou popsány jednotlivé atributy tabulky.

Tabulka 4.9 - Popis tabulky Stations

Stations		
atribut	**datový typ**	**popis**
id	Number NOT NULL	PK - primární klíč
name	Varchar(50 BYTE)	nazev stanice
description	Varchar(300 BYTE)	popis stanice
latitude	Real NOT NULL	zeměpisná šířka polohy stanice
longitude	Real NOT NULL	zeměpisná délka polohy stanice

Stations_comments – komentáře stanice

Do tabulky stations_comments jsou ukládány data o tom, které komentáře mají stanice přiřazeny. V tabulce 4.10 jsou popsány jednotlivé atributy této tabulky.

Tabulka 4.10 - Popis tabulky stations_comments

Stations_comments		
atribut	**datový typ**	**popis**
comment_id	Number NOT NULL	PFK - Primární cizí klíč z tabulky comments
station_id	Number NOT NULL	PFK - Primární cizí klíč z tabulky stations

Services – služby

Do tabulky services jsou ukládány data o službách. Každá služba může být přiřazena několika stanicím. V tabulce 4.11 jsou popsány jednotlivé atributy tabulky.

Tabulka 4.11 - Popis tabulky services

Services		
atribut	**datový typ**	**popis**
id	Number NOT NULL	PK - primární klíč
name	Varchar2(50 BYTE)	název služby
description	Varchar2(300 BYTE)	popis služby
contact	Number	kontakt na službu
icon_url	Varchar2(256 BYTE)	url uložené ikony pro zobrazení

Stations_Services – služby stanic

Do tabulky stations_services jsou ukládány data o tom, které služby mají stanice přiřazeny. V tabulce 4.12 jsou popsány jednotlivé atributy této tabulky.

Tabulka 4.12 - Popis tabulky stations_services

Stations_services		
atribut	**datový typ**	**popis**
service_id	Number NOT NULL	PFK - Primární cizí klíč z tabulky services
station_id	Number NOT NULL	PFK - Primární cizí klíč z tabulky stations

4.4.3 Indexy

Indexy jsou v databázovém systému pro zrychlení při vyhledávaní požadovaných dat. Použitím indexů dojde k výraznému zrychlení v tabulkách, kde se často vyhledává a je v nich umístěno velké množství záznamů. Zároveň použití indexů zvyšuje režii při vkládání dat. Důvodem je vytváření stromové struktury pomocí, které je pak zrychlen výběr dat při kladených dotazech. Procházení vytvořeného stromu je pak v průměru rychlejší než prohledávání neuspořádaného pole hodnot sloupce. Indexy jsou automaticky vytvářeny nad sloupci primárních klíčů.

V informační tabuli je indexován každý sloupec, ve kterém se je uložen primární klíč dané tabulky. Dále jsou indexovány sloupce, pomocí kterých je v tomto systému vyhledávána poloha spoje, jsou to sloupce route_num a datum v tabulce possitions.

Příklad vytvoření indexu nad sloupcem route_num v tabulce positions:

```
CREATE INDEX index_poss_routeNum ON positions(route_num);
```

Přehled použitých indexů (nejsou uvedeny sloupce PK) je popsán v tabulce 4.13

Tabulka 4.13 - Přehled použitých indexů

Přehled použitých indexů	
Název	**Popis**
index_poss_routeNum	Indexuje sloupec route_num v tabulce positions. Je použit při vyhledávání polohy spoje.
index_poss_datum	Indexuje sloupec datum v tabulce positions. Je použit při vyhledávání polohy spoje.

4.4.4 Sekvence

V databázovém systému Oracle database jsou sekvence použity pro generování jedinečných identifikátorů. V informační tabuli pro cestující jsou sekvence použity ke generování hodnoty primárních klíčů. Názvy sekvencí jsou voleny, tak aby bylo možné jednoduše určit, k jaké tabulce a primárnímu klíči patří.

Příklad vytvoření sekvence seq_attributes_id pro primární klíč id v tabulce attributes:

```
CREATE SEQUENCE seq_attributes_id
      START WITH 1
      INCREMENT BY 1;
```

Přehled použitých sekvencí je popsán v tabulce 4.14.

Tabulka 4.14 - Přehled použitých sekvencí

Přehled použitých sekvencí	
Název	**Popis**
seq_attributes_id	Generování hodnot primárního klíče v tabulce attributes.
seq_schedules_id	Generování hodnot primárního klíče v tabulce shedules.
seq_types_id	Generování hodnot primárního klíče v tabulce types.
seq_stations_id	Generování hodnot primárního klíče v tabulce stations.
seq_comments_id	Generování hodnot primárního klíče v tabulce comments.
seq_services_id	Generování hodnot primárního klíče v tabulce services.

4.4.1 Funkce

Funkci je možné definovat jako posloupnost příkazů, které jsou vykonány při spuštění a pomocí návratové hodnoty je vrácen výsledek. Funkci je také možné předávat vstupní parametry. V databázovém systému Oracle database je použit jazyk PL/SQL.

V příloze A je příklad pro vytvoření funkce GetNearestStationId, tato funkce je využita při výpočtu zpoždění daného spoje. Následující tabulka 4.15 zobrazuje přehled vytvořených funkcí v informační tabuli pro cestující. Názvy jsou voleny tak, aby byl zřejmý účel dané funkce.

Tabulka 4.15 - Přehled použitých funkcí

Přehled použitých funkcí	
Název	**Popis**
GetNearestStationId	Funkce na základě předané zeměpisné šířky a délky vrátí nejbližší stanici.
GetStationFromDate	Funkce na základě předaného datumu a id jízdního řádu vrátí id stanice, která je v daném jízdním řádu a nejbližším čase narozvrhována.
GetDelay	Funkce na základě dvou předaných stanic a čísla spoje vrátí rozdíl příjezdu či odjezdu v minutách.
GetLatPerKm	Funkce vravcí zeměpisnou šíři v kilometrech
GetLongPerKmAtLat	Funkce vraví zeměpisnou délku v kilometrech na základě předané zeměpisné šířky

4.4.2 Triggery

Trigger je možné definovat jako, že je to procedura, která není spouštěna příkazem, ale po vykonání určitého dotazu nad danou tabulkou. Pomáhá hlídat validaci vstupních dat, odstraňovat nepotřebná data atd. Typickým případem může být vytvoření auto inkrementace primárního klíče nebo odstranění přebytečných dat v okolních tabulkách při smazání záznamu. V informační tabuli pro cestující jsou vytvořeny právě tyto dva typy trrigerů. V tabulce 4.16 jsou popsány použité trrigery.

Tabulka 4.16 - Přehled použitých trrigerů

Přehled použitých triggerů	
Název	**Popis**
before_insert_attribute	Trriger zajistí inkrementaci primárního klíče při vložení nového řádku do tabulky attributes
before_insert_schedule	Trriger zajistí inkrementaci primárního klíče při vložení nového řádku do tabulky schedules
before_insert_type	Trriger zajistí inkrementaci primárního klíče při vložení nového řádku do tabulky types
before_insert_station	Trriger zajistí inkrementaci primárního klíče při vložení nového řádku do tabulky stations
before_insert_comment	Trriger zajistí inkrementaci primárního klíče při vložení nového řádku do tabulky comments
before_insert_service	Trriger zajistí inkrementaci primárního klíče při vložení nového řádku do tabulky services
before_delete_route	Trriger zajistí odstranění zbylých dat v okolních tabulkách po smazaném spoji
before_delete_attribute	Trriger zajistí odstranění zbylých dat v okolních tabulkách po smazané vlastnosti spoje
before_delete_schedule	Trriger zajistí odstranění zbylých dat v okolních tabulkách po smazaném jízdním řádu
before_delete_station	Trriger zajistí odstranění zbylých dat v okolních tabulkách po smazané stanici
before_delete_comment	Trriger zajistí odstranění zbylých dat v okolních tabulkách po smazaném komentáři
before_delete_service	Trriger zajistí odstranění zbylých dat v okolních tabulkách po smazané službě stanice

4.5 Návrh webové aplikace

Webová aplikace je napsána v jazyce PHP a využívá návrhový vzor MVC. Vstupním bodem do aplikace je soubor index.php, který najdeme v adresáři public. Pomocí

tohoto souboru dochází ke zpracování požadavků klienta. Webová aplikace využívá funkcí knihovny Zend Framework verze 1.10.3 od společnosti Zend Technologies Ltd.

Aplikace byla rozdělena na dvě části.

4.5.1 Administrační rozhraní

Administrační rozhraní umožňuje:

- přidávat, odebírat a upravovat železniční stanice, u stanice je možné volit název a popis stanice, polohu v GPS souřadnicích. U každé stanice je možné vyvolat nové okno prohlížeče pro zobrazení příslušné informační tabule.
- přidávat, odebírat a upravovat jednotlivé služby. Po té je možné jednotlivé služby přiřazovat železničním stanicím. U služeb zaznamenáváme název a popis služby, případný kontakt a symbol pro zobrazení na informační tabuli
- přidávat, odebírat a upravovat různé poznámky, které mohou obsahovat název, popis a symbol pro zobrazení na informační tabuli. Tyto poznámky je následně možné přiřazovat stanicím, jízdním řádům a jednotlivým položkám jízdního řádu.
- přidávat, odebírat a upravovat spoje. Následně jim přiřadit nebo zrušit jízdní řády.
- Přidávat, odebírat a upravovat jednotlivé typy vlaků. U typu vlaku je možné nastavit název, symbol či zkratku, popis, ikonu a barvu pro zobrazení v informační tabuli.
- Přidávat, odebírat a upravovat jízdní řády. U jízdního řádu je možné zvolit datum, od kdy je jízdní řád platný a kdy jeho platnost končí.

Rozvržení

V rámci dodržení návrhového vzoru MVC byla tvorba zdrojových kódu rozdělena na jednotlivé části.

`Models` – třídy, které komunikují s databází. Zpravidla platí, že je pro každou tabulku v databázi vytvořena jedná třída, která je potomkem abstraktní třídy Zend_Db_Table_Abstarct. Díky tomu je možné dosáhnout abstraktní práce s databází. V následující tabulce 4.16 je přehled navržených tříd tohoto typu.

Tabulka 4.17 - Přehled vytvořených tříd typu model

Přehled vytvořených tříd typu model	
Název třídy	**Název tabulky**
Application_Model_DbTable_Attributes	Attributes
Application_Model_DbTable_RoutesAttributes	Routes_Attributes
Application_Model_DbTable_Comments	Comments
Application_Model_DbTable_Positions	Possitons
Application_Model_DbTable_Routes	Routes
Application_Model_DbTable_Services	Services
Application_Model_DbTable_Schedules	Schedules
Application_Model_DbTable_SchedulesStations	Schedules_Stations
Application_Model_DbTable_SchedulesComments	Schedules_Comments
Application_Model_DbTable_Stations	Stations
Application_Model_DbTable_StationsComments	Stations_Comments
Application_Model_DbTable_StationsServices	Stations_Services
Application_Model_DbTable_Types	Types

`Views` – soubory typu .phtml, které prezentují data tříd models. Ke každému View patří příslušný Controller, kde dochází k načtení a úpravě dat. V následující tabulce 4.17 je přehled vytvořených tříd typu Controllers.

Tabulka 4.18 - Přehled vytvořených tříd typu controller

Přehled vytvořených tříd typu Controller
Název třídy
AttributeController
CommentController
ErrorController
IndexController
RouteController
ServiceController
ScheduleController
ScheduleStationController
SidebarController
StationController
TypeController

`Forms` – třídy, které ulehčují práci při zpracování a kontrolou zadaných dat uživatelem do formuláře. Každá třída typu Form je potomkem abstraktní třídy Zend_Form, která je abstraktní třídou pro formuláře. V následující tabulce 4.18 je přehled vytvořených tříd typu Forms.

Tabulka 4.19 - Přehled vytvořených tříd typu form

Přehled vytvořených tříd typu form	
Název třídy	**Použití**
Application_Form_Attribute	formulář pro zadání parametrů linky
Application_Form_Comment	formulář pro zadání parametrů komentáře ke stanici, jízdnímu řádu a položce jízdního řádu
Application_Form_Route	formulář pro zadání parametrů linky
Application_Form_Service	formulář pro zadání parametrů služeb ve stanici
Application_Form_Schedule	formulář pro zadání parametrů jízdního řádu
Application_Form_ScheduleStation	formulář pro zadání parametrů položky jízdního řádu
Application_Form_Station	formulář pro zadání parametrů stanice
Application_Form_Type	formulář pro zadání parametrů typu linky

Vzhled

Vzhled jednotlivých částí aplikace je definován prostřednictvím připojeného souboru default.css. Syntaxe připojení externího souboru se stylopisem prostřednictvím Zend Framework je v následující ukázce.

```
<? php echo $this->headLink()->appendStylesheet(
                                    $this->baseUrl().'/css/default.css'
                                    );

?>
```

Webová stránka je rozdělena na pomocí HTML tagu div na několik částí. Celý obsah webové stránky je vložen do kontejneru wrap, který tvoří obal celé stránky.

Kontejner wrap je rozdělen na tři základní části, tj. hedader, content, footer.

Kontejner header obsahuje úvodní tytulek a logo.

Kontejner content je rozdělen na dvě části, tj. sidecar, pro zobrazení menu a innerpage, pro zobrazení hlavního obsahu stránek. Innerpage je navíc obalen kontejnerem border, pro zobrazení grafických prvků při zalomení.

V kontejneru footer s nachází odkaz na stránky, ze kterých je možné stáhnout původní tempalte, který byl použit a částečně upraven pro tuto webovou aplikaci.

Vzhled administračního rozhraní je prezentován v příloze A.

4.5.2 Informační tabule

Rozvržení

Po odeslání požadavku se základními vlastnostmi pomocí adresy url na server obdrží klient základní obsah a vzhled stránky a poté veškeré nastavení probíhá na straně klienta. Z důvodu přehlednosti a lepší orientace byly vytvořeny některé objekty funkce v programovacím jazyku JavaScript. V následujícím seznamu je přehled základního rozvržení:

- **main.js** – soubor obsahuje globální proměnné `myMap` a `timer a` jedinou funkci `initialize()`, která je zároveň vstupním bodem při načítaní webové stránky. V této funkci dochází k inicializaci třídy `MyGoogleMap`, ve které dochází k vytvoření Google Maps a následnému zobrazení polohy vozidel. Dále je zde také inicializovaný časovač a hodiny zobrazené na layoutu.

- **myGoogleMap.js** – v souboru se nachází hlavní třída `MyGoogleMap`, která reprezentuje standardní třídu `google.maps.Map` a jsou zde také uloženy důležité informace pro chod aplikace, např. id, název a poloha stanice, pro kterou je konkrétní informační tabule určena. Dochází zde k vykreslení samotné mapy a polohy stanice.

- **myMarker.js** – soubor obsahuje třídu `MyMarker` pomocí, které jsou je vykreslena značka daného objektu v mapě.

- **myInfobox.js** – obsahuje třídu, která byla navržena pro zobrazení informací k danému spoji nebo po sjednocení spojům.

- **myClustering.jpg** – obsahuje třídu, která byla navrhnuta pro práci s jednotlivými značkami. Třída nabízí řadu užitečných funkcí pro sjednocení objektů, např. pro převod zeměpisné délky a šířky na souřadnice obrazovky typu x a y, výpočet vzdáleností, sjednocení značek atd.

- **myMoving.jpg** – obsahuje třídu, která je zodpovědná za přesun jednotlivých objektů na mapě. Je využita především pro informační objekty typu `MyInfoBox` pomocí, které jsou zobrazeny informace o konkrétním spoji.

- **fucntions.js** – soubor obsahuje pro vytvoření, odeslání a následnému zpracování požadavku ze serveru pomocí technologie AJAX.

- **clock.js** – soubor obsahuje funkce pro změnu dynamického obsahu hodin.

- **timer.js** – soubor obsahuje funkce pro řízení časovače.

Vzhled

Vzhled jednotlivých částí aplikace je definován prostřednictvím připojeného souboru stationBoard.css. Informační tabule je zobrazená v samostatném okně prohlížeče a je rozvržena do tří základních panelů.

V horním panelu na levé straně se nachází název stanice, ve které je informační tabule umístěna. Na pravé straně najdeme hodiny, které zobrazují aktuální datum. V prostřední části jsou umístěny symboly dostupných služeb konkrétní stanice.

Do prostření části obrazovky je vložena mapa, která bezobslužně zobrazuje pro daný interval očekávané spoje. Pro mapu byla využita technologie Google Maps API V3, která umožňuje zobrazit aktuální polohu jednotlivých spojů. Po každím načtení nových dat je přizpůsobena úroveň přiblížení. Data jsou načítána v pravidelných intervalech pomocí technologie AJAX. Po každém načtení dat jsou zde využity tři metody úpravy při překryvu zobrazených objektů, tj. využití nastavení HTML vlastnosti z-index, sdružování a přesun objektů. Nastavení vlastnosti z-index a sdružování objektů je zde implementováno na ikony, reprezentující daný spoj. Při sdružení spojů dochází i ke sdružení informačních oken a následně je překontrolován jejich překryv. Při překryvu informačních oken dochází k hledání nové nejbližší možné pozice.

Spodní část informační tabule je určena pro případné vysvětlivky zobrazeného spoje.

Výpočet zpoždění

Zpoždění je počítáno jako rozdíl mezi předpokládaným příjezdem či odjezdem do stanice, kde má dané vozidlo v aktuálním čase přijet a předpokládaným příjezdem či odjezdem do stanice, která je brána jako nejbližší možná k aktuální poloze. Z tohoto důvodu je potřeba o jednotlivém vozidle vědět nejen příjezdy a odjezdy stanic, ve kterých vozidlo zastavuje, ale i časy průjezdu zastávek, kde vozidlo jen projíždí.

4.6 Zdrojová data

Při tvorbě této aplikace byly použiti data ze stránek Českých drah dostupných na internetové adrese <http://www.cd.cz>. Jedná se o data jednotlivých stanic a jejich služeb, spojích a jejich jízdních řádech. Dále v tomto projektu byl použit výčet reálných dat ve formě aktuálních poloh vlaků Elephant City, které poté byly použity a kopírovány pro simulaci informační tabule.

4.7 Použité nástroje

Při realizaci této webové aplikace byla použita řada softwarových nástrojů. Převážná většina je šířena pod freeware nebo trial licence. V následující tabulce 4.20 je přehled použitých softwarových nástrojů.

Tabulka 4.20 - Nástroje použité při tvorbě webové aplikace

Použité nástroje při tvorbě webové aplikace		
Název software	**Typ licence**	**Způsob použití**
Zend Server 5.0.0	trial	server pro vývoj aplikace
Zend Studion 7.1.1	trial	tvorba zdrojových kódů
Zend Framework 1.10.3	open source	využití knihovny při tvorbě zdrojových kódů
Oracle database 10g	studentská licence	databázový systém
SQL developer	freeware	správa databázového systému
Toad data modeler	studentská licence	návrh a tvorba databáze
Microsoft Office 2007	studentská licence	tvorba dokumentace
DiaCze	freeware	tvorba diagramů

5 Závěr

Hlavním cílem této práce bylo vytvořit informační tabuli se zobrazením aktuální polohy spoje, které jsou zpracovány z databázového systému. Informační tabule měla být přizpůsobena jednotlivému nádraží či stanice, kde mělo být zabezpečeno zobrazení všech očekávaných spojů během volitelného časového intervalu.

Byl tedy vytvořen požadovaný systém ve formě webové aplikace, který splňuje žádané požadavky. Pro náročné vytvoření simulačních dat byla tato aplikace rozšířena o administrační rozhraní, která se postupem času ukázala jako dobré rozšíření aplikace. Po vhodném upravení aplikace pro konkrétní hromadnou osobní dopravu je možné prakticky tento systém využít. V současné době většina dopravních podniků běžně využívá informační tabule pro cestující bez grafického zobrazení aktuální polohy, proto by se o tomto systému dalo říci, že je částečně vylepšením dosavadních informačních tabulí pro cestující.

Před praktickým nasazením by tento systém měl být ještě rozšířený zejména v administrační části.

Práce na tomto projektu pro mě byla velkým přínosem, protože při tvorbě praktické části jsem si rozšířil znalosti tvorby webových aplikací pomocí frameworku a znalosti o využití velmi propracovaného Googe Maps API V3. Dále tento projekt vyzkoušel mé dovednosti v databázových systémech a přinesl mi mnoho dalších zkušeností s touto technologií. Také při tvorbě teoretické části jsem si ujasnil některé důležité pojmy z tohoto oboru.

Literatura

[1] *Google Maps In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2010 [cit. 4.8.2010]. Dostupné na: <http://en.wikipedia.org/wiki/Google_Maps>.

[2] *Mercator projection In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2010 [cit. 4.8.2010]. Dostupné na: <http://en.wikipedia.org/wiki/Google_Maps>.

[3] *Google Maps In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2010 [cit. 4.8.2010]. Dostupné na: <http://en.wikipedia.org/wiki/Google_Maps>.

[4] *Google Maps API Family* [online]. 2010 [cit. 5.8.2010] Dostupné na: <http://code.google.com/intl/cs/apis/maps/index.html>.

[5] *Google Maps API Web Services* [online]. 2010 [cit. 5.8.2010] Dostupné na: <http://code.google.com/intl/cs/apis/maps/documentation/webservices/index.html>.

[6] *The Google Maps Javascript API V3* [online]. 2010 [cit. 6.8.2010] Dostupné na: <http://code.google.com/intl/cs/apis/maps/documentation/javascript/basics.html>.

[7] *Google Map Javascript API V3 Tutorial* [online]. 2010 [cit. 6.8.2010] Dostupné na: <http://code.google.com/intl/cs/apis/maps/documentation/javascript/tutorial.html>.

[8] *Z-index* [online]. 2010 [cit. 7.8.2010] Dostupné na: <http://www.jakpsatweb.cz/css/z-index.html>.

[9] *Introduction to Marker Clustering With Google Maps* [online]. 2008[cit. 7.8.2010] Dostupné na: <http://www.appelsiini. net/2008/11/introduction-to-marker-clustering-with-google-maps>.

[10] *Displej z tekutých krystalů: the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2010 [cit. 7.8.2010]. Dostupné na: <http://cs.wikipedia.org/wiki/LCD>.

[11] *LED: the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2010 [cit. 7.8.2010]. Dostupné na: <http://cs.wikipedia.org/wiki/LED>.

[12] *Webová aplikace In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2010 [cit. 7.8.2010]. Dostupné na: <http://cs.wikipedia.org/wiki/Webová_aplikace>.

[13] LACKO, L. *Oracle, správa, programování a použití databázového systému.* Praha : Computer Press, 2002. 464 s. ISBN 80-7226-699-3.

[14] *PHP In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2010 [cit. 8.8.2010]. Dostupné na: <http://cs.wikipedia.org/wiki/Php>.

[15] *Zend Framework – otázky a odpovědi* [online]. 2008[cit. 8.8.2010] Dostupné na: <http://php.interval.cz/clanky/zend-framework-prehled-knihoven>.

[16] *Zend Framework – přehled knihoven – otázky a odpovědi* [online]. 2008 [cit. 9.8.2010] Dostupné na: <http://php.interval.cz/clanky/zend-framework-prehled-knihoven/>.

[17] *JavaScript In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2010 [cit. 9.8.2010]. Dostupné na: <http://cs.wikipedia.org/wiki/Php>.

[18] *AJAX In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2010 [cit. 9.8.2010]. Dostupné na: <http://cs.wikipedia.org/wiki/AJAX>.

Příloha A – Ukázka zdrojového kódu pro vytvoření funkce GetNearestStationId v Oracle database

Funkci je prostřednictvím vstupních hodnot předána zeměpisná šířka a délka. Na základě vstupních dat funkce vrátí nejbližší možnou stanici této polohy. Funkce využívá výsledků funkcí GetLatPerKm a GetLongPerKmAtLat.

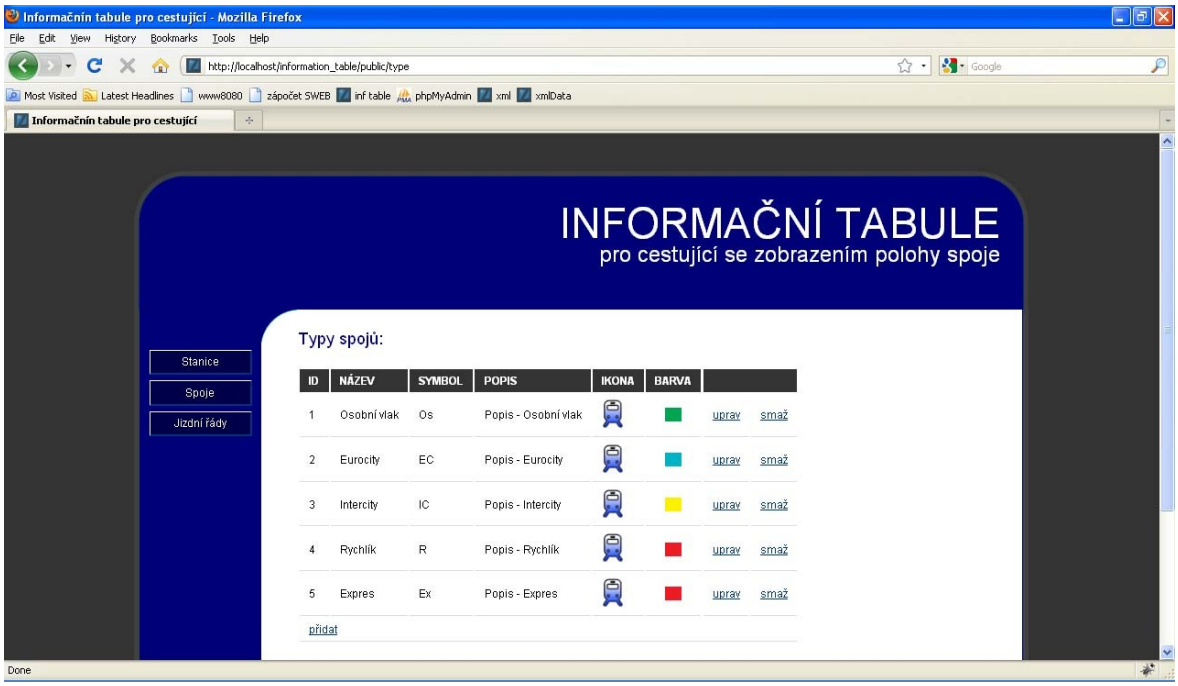
```
create or replace
FUNCTION GetNearestStationId(latitude double precision, longitude double
precision)
RETURN NUMBER
AS
  radiusKm Real := 0.1;
  northBounds Real;
  southBounds Real;
  westBounds Real;
  eastBounds Real;
  resultStationId NUMBER := -1;
BEGIN
  WHILE resultStationId = -1
  LOOP
    northBounds := latitude + GetLatPerKm() * radiusKm;
      southBounds := latitude - GetLatPerKm() * radiusKm;
      westBounds := longitude - GetLongPerKmAtLat(latitude) * radiusKm;
      eastBounds := longitude + GetLongPerKmAtLat(latitude) * radiusKm;

    SELECT s.id INTO resultStationId
            FROM stations s
            WHERE s.latitude > southBounds
        AND s.latitude < northBounds
        AND s.longitude > westBounds
        AND s.longitude < eastBounds
        AND rownum <=1;

    radiusKm := radiusKm * 2;
  END LOOP;

  RETURN resultStationId;
END;
/
```

Příloha B – Ukázka rozvržení webové aplikace – administrační rozhraní – přidání typu spojů



Příloha C – Ukázka rozvržení webové aplikace – informační tabule pro cestující

Informační tabule, jejichž zobrazení je nastaveno pro stanici Úvaly, v čase 14:32:00 a jsou zobrazeny očekávané spoje za následující 1 hodinu.