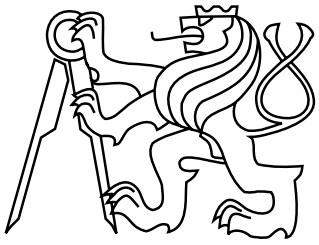
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ

BAKALÁŘSKÁ PRÁCE

PRAHA 2012 Michal MED

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ
OBOR GEODÉZIE A KARTOGRAFIE



BAKALÁŘSKÁ PRÁCE

WEBOVÁ MAPOVÁ APLIKACE PRO SDRUŽENÍ BUDĚJOVICE KULTURNÍ

Vedoucí práce: Ing. Jiří Cajthaml, Ph.D.
Katedra mapování a kartografie

červen 2012

Michal MED

ZDE VLOŽIT LIST ZADÁNÍ

Z důvodu správného číslování stránek

ABSTRAKT

V rámci bakalářské práci byla vyhotovena webová aplikace kulturního informačního systému pro město České Budějovice. S aplikací je propojena databáze obsahující jednotlivé kulturní události, místa konání, typy událostí a informace o uživatelích. Aplikace má uživatelské rozhraní pro vyhledávání událostí podle atributů i v mapě a administrativní rozhraní pro vkládání událostí samotnými organizátory. Technické prostředky použité k tvorbě aplikace jsou podrobně popsány v textové části práce. V rámci práce byl také vytvořen popis aplikace pro laiky. Práce dále obsahuje rozbor aplikačního programového rozhraní Google Maps pro JavaScript.

ABSTRACT

In this bachelor work, web application of culture information system for city of České Budějovice was created. The application is connected to the database, which contains informations about cultural events, places and informations about users. Application has user interface for searching cultural events according to attributes and for seaching from map and administrative interface for inserting new events by organisers themselves. Technical resources used to create application are described in the text part of work. To the work belongs description of application for laics and analysis of application programming interface of Google Maps for JavaScript.

PROHLÁŠENÍ

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jiřího Cajthamla, Ph.D. Použitou literaturu a další zdroje uvádím v seznamu zdrojů.

V Praze dne ....

(podpis autora)

PODĚKOVÁNÍ

Mé poděkování patří především vedoucímu práce a to za připomínky, konstruktivní návrhy a pomoc při zpracování této práce. Kromě toho bych chtěl poděkovat rodině a přátelům za trpělivost a podporu.

Obsah

Seznam obrázků

1 Úvod

Webová aplikace, kterou jsem vytvořil v rámci této bakalářské práce a které jsem dal pro potřeby tohoto textu pracovní název Culture, se zabývá kulturním děním ve městě České Budějovice. Jedná se o kulturní přehled, jehož obsah je vytvářen samotnými organizátory kulturních událostí. Do aplikace je zakomponována interaktivní mapa, která slouží uživatelům k zobrazování výsledků a organizátorům k přidávání nových míst. Aplikace uchovává informace o kulturních událostech, o místech, ve kterých se tyto události odehrávají, a o typech těchto událostí.

Hlavními technickými prostředy při tvorbě aplikace byly skriptovací jazyky `JavaScript` a `PHP` a databáze `PostgreSQL`. V aplikaci je použita mapa **Google** a její aplikační programové rozhraní, které využívá `JavaScript`. K programování funkční části webového dokumentu bych mohl použít i jiné skriptovací jazyky (například `Flash` nebo `Java`), ale zvolil jsem `JavaScript`, protože ho používám pro tvorbu mapy a jejích ovládacích prvků a kromě toho jsem s `JavaScriptem` už v minulosti pracoval a znám jeho syntaxi. Skriptovací jazyk `PHP` byl zvolen díky široké škále předdefinovaných metod pro práci s databází a s relacemi a velkému množství přehledných příkladů a referencí na internetu. S tímto skriptovacím jazykem jsem pracoval poprvé až při tvorbě aplikace Culture. A konečně databázi `PostgreSQL` jsem si vybral proto, že jsem s ní v minulosti už pracoval a jedná se o Open Source projekt.

Tato textová dokumentace je tematicky rozdělena do tří částí. První z nich se zabývá aplikačním programovým rozhraním projektu **Google Maps**. Nejprve jsou nastíněny příklady využití aplikačních programových rozhraní z rodiny **Google APIs**, které souvisí s tvorbou interaktivních mapových aplikací, dále jsou zde rozebrány základní třídy `Google Maps JavaScript API V3` formou jednoduchého tutorialu jak zobrazit mapu, marker a informační okno, a nakonec je nastíněno využití `Google Maps JavaScript API V3` v aplikaci Culture.

Druhá část textové dokumentace se zabývá praktickým používáním aplikace Culture z hlediska uživatelů i organizátorů. Je v ní vysvětleno, jakým způsobem pomocí aplikace

vyhledávat v databázi, jak přidávat do databáze nové záznamy a také jak se zaregistrovat jako nový uživatel a poté přihlásit. Tato část je napsaná takovým způsobem, aby ji porozumněl i laik.

V poslední části textové dokumentace jsou popsány principy fungování aplikace Culture z technického hlediska. Je zde popsána struktura a fungování databáze `PostgreSQL`, využití skriptů napsaných v `PHP` i v `JavaScriptu` a je zde i letmá zmínka o využití stylů.

Snahou bylo vytvořit aplikaci Culture tak, aby byla co nejpřehlednější, ovládaní bylo intuitivní a aby byly požadavky na administraci ze strany provozovatele pokud možno minimální.

2 Google Maps API V3

API znamená `Aplication Programming Interface`, tedy aplikační programové rozhraní. Používá se pro zjednodušení práce programátora při integraci aplikace do svého programu.

Jako mapový podklad jsem použil mapy Google, k práci s obsahem mapy a s mapovými prvky jsem použil JavaScriptové API verze 3. Zvolil jsem je z několika důvodů:

- S **Google Maps API** jsem už v průběhu studia pracoval. Společně se spolužákem Štěpánem Turkem jsme Google mapy použili jako mapový podklad pro naši semestrální práci v předmětu 153INKA – Interaktivní kartografie (pod vedením Ing. Jiřího Cajthamla Ph.D.), ve které jsme zpracovávali formou interaktivní mapy program Budějovického Majálesu 2011. Proto jsem se během práce nemusel znovu seznamovat se základy a mohl jsem rovnou řešit konkrétní problémy týkající se této bakalářské práce.

- Velké množství snadno použitelných předdefinovaných objektů a tříd. Společnost Google je podle mě hodně otevřená programátorům, kteří chtějí využívat jejích služeb, a proto jsou API jejích produktů velice dobře provedené. To se netýká zdaleka jenom Google Maps, ale i Google Calendar, Google Chrome, Google+ nebo Google Earth. Společnost Google podporuje i programátory aplikací do svého operačního systému pro mobilní zařízení Android, pro který vytvořila obrovské množství knihoven zjednodušujících vývoj aplikací v jazyce Java.

- Přehledně zpracované reference k třídám i objektům. Kromě toho, že je dobře zpracované API, jsou velice dobře zpracované i reference a dokumentace. Na stránkách Google Developers je u každé třídy popsán konstruktor, metody, vlastnosti a události a jak s nimi pracovat. Orientace mezi jednotlivými prvky je zde velmi jednoduchá a intuitivní.

- Velké množství příkladů a řešených problémů v internetových diskuzích. Tento bod je dvousečnou zbraní, protože i přes to, že se v diskuzích dá najít téměř vše, ne vždy najdete to, co hledáte. Zrovna v případě Google Map to ale úplně neplatí, protože

problémy většinou bývají dost konkrétní, a tak se dají snadno formulovat i vyhledat v diskuzích a odpovědi bývají dost jednoznačné. Stejně jsem většinou hledal ověřené odpovědi na webu stackoverflow.com nebo na stránkách Google Developers. Navíc tento způsob vyhledávání řešení problémů dost často vyžaduje celkové porozumění jak problému, tak i technickému prostředku, kterým se problém řeší. Navíc se mi často stávalo to, že jsem musel hledat řešení problému po částech, jednotlivé části spojit a výsledné řešení přizpůsobit vlastní aplikaci. Navzdory těmto nevýhodám je možné najít velice rychle elegantní řešení téměř každého problému.

Mapových serverů dobře použitelných pro aplikaci podobného typu jako je tato práce je více, pro přehlednost bych zde uvedl nejznámější:

- UMN MapServer (Open Source)
- Open Street Maps API (otevřená mapa světa)
- ArcGIS server (produkt firmy ESRI, používá například informační server města Stuttgart)
- Mapy.cz API (nejznámější český mapový server)

2.1 Google APIs

Google Maps API je vhodné pro několik druhů aplikací a je propojená s řadou dalších API. V této části se budu věnovat těm, které mají co dočinění s tvorbou mapových aplikací.

2.1.1 Google Places API

Slouží k práci s databází Google Places, ve které jsou zaneseny miliony bodů po celém světě. Vzhledem k tomu, že v databázi Google jsou místa rozčleněna do nejrůznějších typů a sama databáze je dost obsáhlá, může využití této databáze dost ulehčit práci. Na druhou stranu si myslím, že se tato databáze nedá využít vždy. Když jsem procházel místa, která by měla být obsažena v kulturní databázi aplikace, většinu z nich jsem v Google databázi našel. Ale databáze Google Places se více než na kulturu zaměřuje na „businesses and points of interest“ (převzato z [1]). Tedy hlavně na obchody, sídla a pobočky firem, historické a přírodní památky, restaurace, hotely a podobně. Hudební kluby, koncertní

sály, výstavní síně a divadla jsou v databázi samozřejmě také, ale z náhodného výběru pěti mně známých míst pořadajících kulturní akce jsem v databázi Google Places našel jen tři. Jedním z těch, které jsem nenašel, je dokonce největší městský Kulturní dům, což je místo, které by v databázi rozhodně mělo být.

S databází se pracuje pomocí **Google Places API**, které je zatím pouze experimentální. Zobrazovaná místa je možné selektivně vybírat podle kategorií, což je výhodné, pokud někdo chce v mapě zobrazovat tématické vrstvy například pro restaurace nebo historické památky. U míst pořádajících kulturní akce ale dochází k problému, že v databázi Google Places mohou být klasifikovány víceméně jakkoliv. Bývá běžnou praxí, že v kavárnách probíhají divadelní představení, na náměstích koncerty a v restauracích vernisáže. Samozřejmě nejde o neřešitelný problém, ale myslím si, že pro účely této práce je jednodušší a rychlejší vytvářet vlastní databázi míst.

Google Places API je nástrojem ke komunikaci mezi uživatelem a databází Google Places, mezi kterými komunikuje pomocí dotazů (requests). Jsou dostupné čtyři základní typy dotazů:

- **Place Search** – vrací seznam blízkých míst na základě polohy uživatele
- **Place Detail request** – vrací detailnější informace o daném místě
- **Place Check-ins** – slouží ke zjišťování popularity míst. Zjišťuje, zda uživatel o dané místo projevil zájem. Místům s větším počtem „checků“ se zvyšuje hodnocení při vyhledávání.
- **Place Reports** – umožňuje přidávat do databáze nová místa a mazat místa, která uživatel přidal.

K místům je možnost přiřazovat události. Toto se řeší pomocí **Events in the Places API**. Výstupy z dotazů jsou buď ve formátu `XML`, nebo `JSON`. Parametry vyhledávacího dotazu jsou souřadnice polohy (`location`) a poloměr (`radius`), ve kterém budou vyhledána místa. Volitelnými parametry jsou pak jazyk (`language`), klíčové slovo pro vyhledávání (`keyword`), jméno (`name`) nebo typ hledaného místa (`types`) a podle čeho budou výsledky seřazeny (`rankby`) – zda podle důležitosti, nebo podle vzdálenosti (`prominence, distance`).

U ostatních typů dotazů jsou parametry podobné. U dotazu na detailní popis místa je povinným parametrem identifikátor (`reference`), volitelným parametrem je jazyk (`language`). Další dva dotazy se odesílají metodou `POST`. Parametrem předáváným touto metodou pro Place Check-in request je identifikátor (`reference`), pro přidávání míst pomocí Place report je parametrů více. Jsou to souřadnice polohy (`location`), přesnost (`accuracy`), jméno (`name`), typ (`types`) a jazyk (`language`). Pro mazání míst je parametrem jen identifikátor (`reference`).

Povinnými parametry u všech dotazů jsou navíc API klíč a sensor (`key, sensor`). API klíč slouží k identifikaci aplikace, takže u všech přidaných míst je jasně definován autor. Sensor definuje, zda dotaz přichází z přístroje, který dokáže určit polohu (například GPS). Může být nastaven na hodnotu `true` nebo `false`.

2.1.2 Google Images API

Do **Google Image API** patří **Google Static Maps API**, které slouží k umístění statické mapy do aplikace, a **Google StreetView Image API**, které slouží k vložení obrázku z Google Street View. Tyto API nepotřebují k práci JavaScript, místo toho využívají html tag <`img`> s parametrem `src`. V závislosti na uvedených parametrech je možné statické mapě nastavit souřadnice středu mapy (`center`), úroveň přiblížení (`zoom`), velikost obrázku (`size`), měřítko (`scale`), formát obrázku (`format`), typ a jazyk mapy (`maptype` a `language`) a region, který se zobrazí (`region`). Do statické mapy mohou být vkládány i markery (body) a cesty, což jsou seřazené sady bodů. U Street View tolik možností není. V parametrech je možnost nastavit jenom souřadnice polohy kamery (`location`), velikost obrázku (`size`) a pootočení kamery ve 3D prostoru (`heading, fov` a `pitch`). Kromě toho jsou povinnými parametry opět API klíč a sensor (`key, sensor`).

2.1.3 Elevation API

Elevation API poskytuje výškopisná data na celém zemském povrchu včetně moří a oceánů, kde vrací záporné hodnoty. V případě dotazu na místo, kde není zanesena výška, je výška vypočítána interpolací z okolních bodů. Je vhodné pro vývoj turistických a cyklistických map nebo pro zeměměřické aplikace vyžadující nízkou přesnost. Do **Elevation**

API se dá přistupovat buď přes **Google Maps JavaScript API V3**, nebo přímo pomocí objektu `ElevationService()`.

Elevation API poskytuje výšková data vztažená k dotazovaným objektům. To mohou být body, nebo cesty. V závislosti na tom, zda jde o body nebo o cesty, se také mění parametry. Pro body je zde parametr `locations`, který může obsahovat jeden bod ve formátu zeměpisná šířka/délka oddělené čárkou, nebo více bodů ve formě pole. Pro cesty existují parametry `path` a `samples`, které určují cestu, na které budou zjištěny výškopisné hodnoty, a specifikují počet bodů, které budou na cestě. Cesta je těmito body rozdělena na stejně dlouhé úseky. Posledním povinným parametrem jak pro body, tak pro cesty je sensor (`sensor`). Výstup je ve formátu `XML`, nebo `JSON`.

2.1.4 Google Maps JavaScript API V3, využití pro mobilní zařízení

Google Maps API je od začátku navržené tak, aby mapové aplikace jím vytvořené a spravované byly podporovány v mobilních zařízeních se systémem Android nebo iOS. Google Maps jsou také jediná mapová platforma, která nabízí SDK (Software Development Kit) jak pro Android, tak pro iOS. Pro správu a přístup do databáze Google Places je k dispozici **Google Places API**, taktéž optimalizované i pro využití v mobilních zařízeních. Obrázky vytvořené pomocí **Google Images API** jsou v mobilních zařízeních samozřejmě také plně podporované. **Google Maps JavaScript API V3** bude popsána samostatně.

2.2 Práce s Google Maps API V3 a jeho použití v této práci

V této části podrobně rozeberu **Google Maps JavaScript API V3** a podrobně popíšu ty části, které jsou v aplikaci použity.

2.2.1 Zobrazení mapy na stránce a třída `Map`

Základním prvkem **Google Maps JavaScript API** je třída `Map`. Základním úkolem je potom zobrazení mapy na stránce. K tomu je potřeba nejprve načíst v `html` stránce **Google Maps API**. Do hlavičky html souboru musí být vložen následující kód:

```
<script type="text/javascript"
    src="http://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&sensor=TRUE_OR_FALSE">
</script>
```

Když je API načtené, je možné načíst do stránky mapu. Vzhledem k tomu, že se JavaScript nespustí sám, ale musí být zavolán, používám k načtení mapy funkci `initializeMap()`, která se spouští při načítání těla stránky.

```
<body onLoad="javascript:initalizeMap();">
```

Funkce pro načtení mapy je velice krátká a jednoduchá. Nejdřív je třeba vytvořit instanci třídy `Map`. Konstruktor vypadá takto:

```
Map(mapDiv:Node, opts?:MapOptions)
```

Parametr `mapDiv` určuje, ve kterém `divu` v html kódu se mapa zobrazí, a v `opts?` je seznam vlastností (`properties`) objektu `MapOptions`, který určuje, jak se mapa zobrazí. Úplný seznam properties zde uvádět nebudu, uvedu jen nejčastěji používané:

- `center` – určuje souřadnice středu zobrazené části mapy. Uvadí se jako instance třídy `LatLng`, obsahující číselné hodnoty zeměpisné šířky a délky (`lat` a `lng`) v souřadnicovém systému WGS-84. Toto je povinný parametr.
- `zoom` – určuje úroveň přiblížení mapy. Uvádí se jako číselná hodnota, přiblížení roste se zvyšující se hodnotou. Nejvyšší možná hodnota přiblížení je 21, nejmenší je 0. Při zadání vyššího nebo nižšího čísla je zoom nastaven na nejbližší možnou hodnotu. Toto je povinný parametr.
- `MapTypeId` – určuje typ mapy. Uvádí se jako instance třídy `MapTypeId`. Možnosti nastavení jsou:
  - `HYBRID`
  - `ROADMAP`
  - `SATELLITE`
  - `TERRAIN`

  Toto je povinný parametr.

- `draggable` – určuje, zda je možné mapu uchopit kursorem. Uvádí se jako hodnota typu `bool`. Implicitně je nastaven na `true`.
- `heading` – určuje pootočení mapy vůči severu. Uvádí se jako číselná hodnota azimutu od severu ve stupních po směru hodinových ručiček.
- `scrollwheel` – určuje, zda je možné mapu přibližovat a oddalovat kolečkem myši. Uvádí se jako hodnota typu `bool`. Implicitně je nastaven na `true`.

A dále uvedu několik dalších, jejichž funkce je zřejmá:

- `maxZoom` – číslo
- `minZoom` – číslo
- `disableDoubleClickZoom` – bool
- `overviewMapControl` – bool
- `scaleControl` – bool
- `streetViewControl` – bool

Pro zobrazení mapy na stránce je zde uvedeno již vše potřebné, uvedu tedy praktický příklad. Toto je html kód stránky, ve které bude mapa zobrazena:

```
 <!DOCTYPE>
<html>
<head>
    <script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?sensor=false">
    </script>
    <script type="text/javascript" src="map.js"></script>
</head>
<body onload="initializeMap();">
    <div id="mapCanvas"></div>
</body>
</html>
```

JavaScriptový soubor `map.js` vypadá následovně:

```
var map;

function initializeMap() {
```

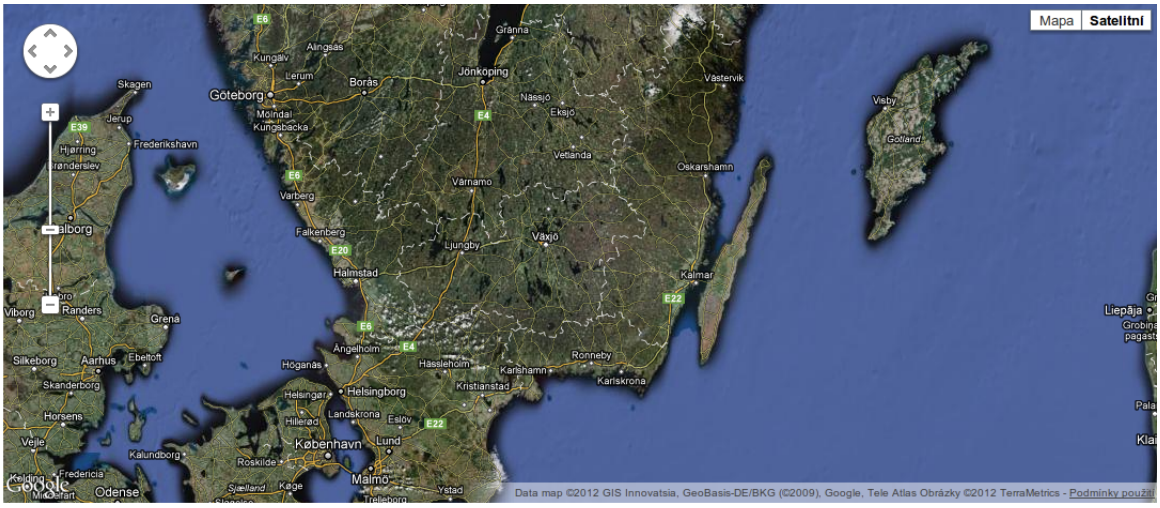
```
var myOptions = {
    center: new google.maps.LatLng(56.83, 15.16),
    zoom: 7,
    mapTypeId: google.maps.MapTypeId.HYBRID,
    streetViewControl: false,
    draggable: false
};
map = new google.maps.Map(document.getElementById("mapCanvas"), myOptions);
}
```

Při otevření stránky se načtou dva JavaScriptové soubory. Jedním z nich je mnou vytvořený soubor `map.js`, druhý zajišťuje přístup ke **Google Maps API**. V souboru `map.js` je definována proměnná `map` a dále je v něm funkce `initializeMap()`. Při spuštění skriptu se tedy pouze vytvoří prázdná proměnná `map`. Je to z toho důvodu, aby proměnná existovala po celou dobu běhu stránky (kdyby byla definována uvnitř funkce, její platnost by skončila s průběhem funkce).

V okamžiku, kdy se začne načítat tělo html souboru, je spuštěna funkce `initializeMap`. Ta nejprve vytvoří proměnnou `myOptions` se zvolenými vlastnostmi a poté naplní proměnnou `map`. První parametr konstruktoru říká, že mapa bude načtena v prvku definovaném `id=mapCanvas`, druhý parametr načítá objekt `myOptions`, podle kterého se mapa bude řídit. Při načtení stránky v prohlížeči se objeví hybridní mapa s úrovní přiblížení 7, se středem v bodě o souřadnicích 56.83° s.š. a 15.16° v.d., bez možnosti posunování mapy uchopením myší a bez možnosti používat Street View.

Ke všem vlastnostem mapy se přistupuje členskými metodami přes instanci třídy `Map`. Metody jsou čtyř různých typů:

- `get` – pomocí metod typu `get` se získávají informace o vlastnostech mapy. Vlastnosti mapy jsou většinou nastaveny v objektu `MapOptions` a metody, které získávají jejich hodnotu, jsou pojmenovány podle nich. Mezi metody typu `get` patří například:
  - `getCenter()` – vrátí souřadnice středu mapy jako instanci třídy `LatLng`
  - `getBounds()` – vrátí souřadnice jihozápadního a severovýchodního rohu mapy jako instanci třídy `LatLngBounds`
  - `getDiv()` – vrátí uzel DOM, ve kterém se mapa nachází



Obr. 2.1: Mapa vytvořená na základě funkce `initializeMap()`

  - `getMapTypeId()` – vrátí typ mapy (`HYBRID`, `ROADMAP`, `SATELLITE` nebo `TERRAIN`)
  - `getZoom()` – vrátí hodnoty přiblížení mapy
  - a další (pojmenované analogicky k názvům vlastností)

- `set` – slouží k nastavení nebo přenastavení vlastností mapy. Mají hodně společného s metodami typu `get`. Patří mezi ně například:
  - `setCenter(latlng:LatLng)` – nastaví souřadnice středu mapy na hodnotu instance třídy `LatLng`
  - `setMapTypeId(mapTypeId:MapTypeId|string)` – nastaví typ mapy na hodnotu `HYBRID`, `ROADMAP`, `SATELLITE` nebo `TERRAIN`
  - `setZoom(zoom:number)` – nastaví přiblížení mapy na hodnotu `zoom:number`
  - `setOptions(options:MapOptions)` – přenastaví vlastnosti mapy podle objektu `MapOptions`
  - a další (pojmenované analogicky k názvům vlastností)

- `pan` – slouží k posunu středu mapy. V případě, že pozice bodu, do kterého se střed posunuje, je viditelná už před posunem, proběhne posun plynulým pohybem. V opačném případě je mapa posunuta skokově. Metody tohoto typu jsou tři:
  - `panBy(x:number, y:number)` – posune střed mapy o hodnoty `x` a `y`
  - `panTo(latlng:LatLng)` – posune střed mapy do bodu `latlng`

  - `panToBounds(bounds:LatLngBounds)` – posune mapu tak, aby v ní byly viditelně meze nastavené instancí třídy `LatLngBounds`, která definuje meze souřadnicemi jihozápadního a severovýchodního rohu

- `fit` – nastavuje zobrazovanou část mapy. Patří sem jediná metoda:
  - `fitBounds(bounds:LatLngBounds)` – nastavuje meze mapy dle zadané instance třídy `LatLngBounds`.

Dále třída map obsahuje některé vlastnosti (`properties`), které umožňují například přidávání ovládacích prvků mapy nebo spravují registr mapových typů. Tyto vlastnosti jsem během práce nepoužíval, proto se zde o nich nebudu podrobněji rozepisovat.

Poslední důležitou součástí třídy `Map` jsou události (`events`). Události jsou podněty, na základě kterých je provedena nějaká akce. V praxi jsem události používal jako parametr pro `Listener`, který kontroluje, zda se provede událost, a poté vykoná příslušnou akci (spustí funkci). Mezi nejpoužívanější události patří:

- `clicked` – událost se spustí kliknutím myší do mapy
- `dblclicked` – událost se spustí dvojitým kliknutím myší do mapy
- `rightclick` – událost se spustí kliknutím do mapy pravým tlačítkem myši
- `drag` – tato událost se opakuje, dokud uživatel drží kursorem mapu
- `dragstart` – tato událost se provede pouze jednou v okamžiku, kdy uživatel chytí kursorem mapu
- `mousemove` – událost se spustí, kdykoliv uživatel pohne myší v mapě
- `mouseover` – událost se spustí, když uživatel přejede myší nad mapu
- `bounds_changed` – událost se spustí, když se změní meze mapy
- `center_changed` – událost se spustí, když se změní souřadnice středu mapy
- `maptypeid_changed` – událost se spustí, když se změní typ mapy
- `zoom_changed` – událost se spustí, když se změní úroveň přiblížení mapy

- `resize` – událost se spustí, když se změní velikost `div`u, ve kterém se mapa nachází

Použití událostí demonstruji v následujícím příkladu:

```
google.maps.event.addListener(map, 'click', function(event) {
    map.setCenter(event.latLng);
  });
```

V tomto příkladu vytvářím `Listener`, který při kliknutí (`click`) do proměnné `map`, která je instancí třídy `Map`, spustí funkci, která nastaví souřadnice středu mapy `map` na místo, kam uživatel klikl.

Třída `Map` je tedy naprostým základem při práci s **Google Maps JavaScript API V3** a všechny další prvky, které k mapě patří, musí být identifikovány instancí třídy `Map`, aby bylo jasné, ke které mapě patří.

2.2.2 Zobrazení markeru na mapě a třída `Marker`

V aplikaci, kterou jsem vytvořil v rámci této bakalářské práce, jsou velmi důležitou součástí značky, které na mapě označují určité místo, tedy `markery`. K vytváření, používání a editování `markerů` slouží v **Google Maps JavaScript API V3** třída `Marker`.

Při vytvoření instance třídy `Marker` se nejprve spustí konstruktor. Konstruktor třídy `Marker` vypadá takto:

```
Marker(opts?:MarkerOptions)
```

Parametr `opts?` zde obsahuje seznam vlastností (`properties`) objektu `MarkerOptions`. Mezi tyto vlastnosti patří:

- `position` – obsahuje souřadnice bodu, na kterém se marker zobrazí. Tento atribut je povinný
- `map` – instance třídy `Map`, ve které se marker zobrazí
- `flat` – pokud je nastavena na hodnotu `true`, stín markeru se nezobrazuje
- `icon` – určuje obrázek, kterým bude marker v mapě reprezentován

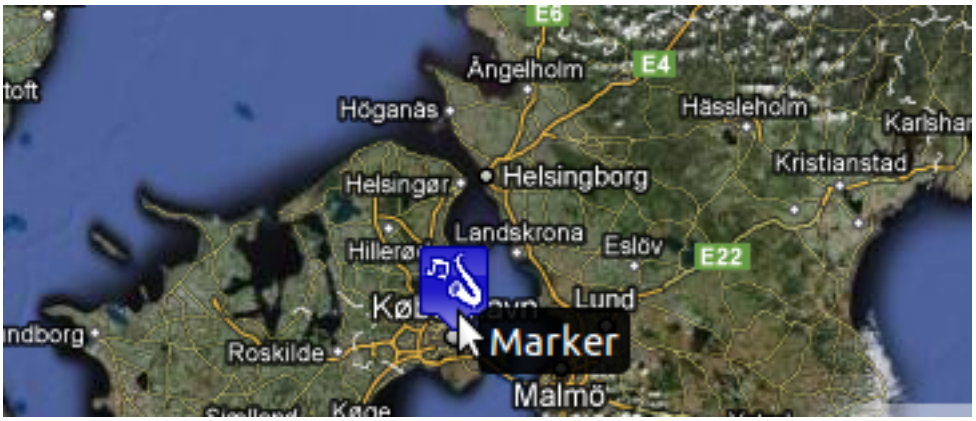
- `shadow` – určuje obrázek, kterým bude v mapě zobrazen stín markeru
- `title` – text, který se zobrazí při podržení myši nad markerem
- `visible` – pokud je nastaven na hodnotu `true`, marker je viditelný
- `cursor` – určuje, jak se změní kursor myši při podržení nad markerem
- `optimized` – pokud je nastavena na hodnotu `true`, všechny markery se vykreslují najednou jako samostatný prvek. Nastavuje se na `false` pouze v případech, kdy je jako ikona nastaven animovaný `.gif`, nebo v případě, kdy je potřeba každý marker rendrovat jako samostatný prvek DOM
- `raiseOnDrag` – pokud je nastavena na hodnotu `true`, po uchopení markeru kursorem myši se ikona markeru zvětší

Pro zobrazení markeru na mapě je zde už uvedeno vše potřebné, uvedu tedy příklad. Do mapy, kterou jsem vytvořil dříve, vložím marker na fixní bod pomocí funkce `addMarker()`, která vypadá takto:

```
function addMarker() {
  var marker = new google.maps.Marker({
    position: new google.maps.LatLng(55.68, 12.57),
    map: map,
    title: "Marker",
    icon: "icons/jazzclub.png"
  });
}
```

K tomu, aby se funkce spustila, ji musím zavolat. To udělám na posledním řádku funkce `initializeMap()`. Při načtení mapy se tedy v mapě `map` na souřadnicích 55,68° s.š. a 12,57° v.d. zobrazí marker s obrázkem `jazzclub.png`, který je ve složce `icons` a s titulkem `Marker`.

K ovládání markeru slouží členské metody. Ve třídě `Marker` jsou pouze dva typy členských metod. Jsou to metody typů `get` a `set` a slouží k získávání nebo nastavování vlastností objektu `MarkerOptions`. Existuje 13 metod `get`, kterými se dají nastavit všechny vlastnosti markeru s výjimkou `optimized` a `raiseOnDrag`. Metod `set` je o jednu víc, protože je možné nastavit všechny vlastnosti objektu `MarkerOptions` najednou (slouží k tomu metoda `setOptions( options:MarkerOptions)`).



Obr. 2.2: Marker vytvořený funkcí `addMarker()`

Třída `Marker` také obsahuje jednu konstantu (`Constant`). Je jí `MAX_ZINDEX`, který určuje maximální hodotu `z-index`u, kterou je možné markeru nastavit. Marker s vyšší hodnotou bude zobrazen více v popředí.

Kromě událostí typu `clicked`, `mouseover` a podobných, které se dají vztáhnout k téměř libovolnému prvku, má i třída `Marker` několik událostí, které souvisí přímo s jejími vlastnostmi. Podobně jako vlastnosti třídy `Map` většinou souvisí se změnou vlastností. Jsou to tyto události (na co událost reaguje je zřejmé z názvů):

- `animation_changed`
- `clickable_changed`
- `cursor_changed`
- `draggable_changed`
- `flat_changed`
- `icon_changed`
- `position_changed`
- `shadow_changed`
- `shape_changed`
- `title_changed`

- `visible_changed`
- `zindex_changed`

Třída `Marker` je rozhodně jednou z nejpoužívanějších tříd v **Google Maps JavaScript API V3** vůbec. S její pomocí se dají vytvářet a spravovat markery, které jsou ale vždy vztažené k nějaké mapě (tedy instanci třídy `Map`). S třídou `Marker` dále velice úzce souvisí třídy `MarkerImage` a `Animation`, dříve zmiňovaný objekt `MarkerOptions` a doposud nezmiňovaný objekt `MarkerShape`. Objekt`MarkerOptions` jsem probral podrobněji, k ostatním zde napíšu jenom málo. Ani jeden z nich jsem nikdy nepoužíval.

Třída `MarkerImage` reprezentuje ikonu markeru. Vlastnosti této třídy jsou například ukotvení obrázku vzhledem k souřadnicím, ke kterým se marker vztahuje, velikost a měřítko obrázku a jeho umístění. Třída `Animation` reprezentuje animaci, která může být přiřazena k markeru. Animace se přiřazuje zavoláním metody `setAnimation` ze třídy `Marker`. Jedná se o animace `BOUNCE` (marker se třese, dokud není animace zastavena) a `DROP` (marker spadne na své místo a po dopadu se zatřese). Objekt `MarkerShape` definuje tvar markeru ve smyslu oblasti, do které je možné kliknout. Tvar je definován vlastnostmi `coords` a `shape`, kde `shape` může být kružnice, polygon, nebo obdélník a `coords` definuje relativní polohu prvků tvaru objektu v pixelech od levého horního rohu markeru (například kružnice je definována souřadnicemi středu a poloměrem).

2.2.3 Zobrazení informačního okna na markeru a třída `InfoWindow`

`InfoWindow` je informační pole v mapě, které často obsahuje text nebo `html` kód, a zpravidla bývá připojeno k markeru, ale může se vyskytnout i samostatně. Pro vytvoření instance třídy `InfoWindow` je použit následující konstruktor:

```
InfoWindow(opts?:InfoWindowOptions)
```

Podobně jako `MapOptions` a `MarkerOptions` i objekt `InfoWindowOptions` obsahuje vlastnosti, které definují konkrétní `InfoWindow`. Mezi tyto vlastnosti patří:

- `content` – definuje obsah okna. Obsahem může být string, `html` kód, nebo jejich kombinace. Obsahem může být i uzel `DOM`u, jehož obsah je definován někde jinde
- `position` – obsahuje souřadnice, ke kterým je informační okno vztaženo

- `zIndex` – definuje `z-index` informačního okna. Větší hodnota `z-indexu` znamená, že okno bude na mapě zobrazeno nad okny s menší hodnotou `z-indexu`
- `maxWidth` – maximální šířka okna, která nezáleží na šířce obsahu. Projeví se pouze tehdy, je-li nastavena před otevřením okna (zavoláním metody `open`). Pokud je nastavena později, je třeba okno zavřít a poté znovu otevřít
- `pixelOffset` – slouží k nastavení rozsahu okna v pixelech
- `disableAutoPan` – bez nastavení hodnoty `false` této vlastnosti je při otevření okna mapa vždy posunuta tak, aby bylo otevřené okno celé viditelné v mapě

V popisu vlastností objektu `InfoWindowOptions` jsem zmínil metodu `open`. Společně s metodou `close` jsou to jediné metody třídy `InfoWindow`, které nejsou typu `get` nebo `set`.

- `open(map?:Map|StreetViewPanorama , anchor?:MVCObject)` – tato metoda slouží k otevření informačního okna. Kromě toho, že stejně jako každá členská metoda musí být zavolána instancí třídy, má metoda `open` ještě dva parametry. Mezi vlastnostmi objektu `InfoWindowOptions` není definována mapa, ve které se bude informační okno zobrazovat, ta se totiž nastavuje až při jeho otevírání. Kromě mapy může být informační okno otevřeno i v instanci třídy `StreetViewPanorama`, která slouží k zobrazení **Google Street View** na základě polohy kamery, úhlu pohledu, adresy a podobně. Druhým parametrem metody `open` je ukotvení (`anchor`). V jádru API je jedinou možností ukotvení informačního okna marker, ale prakticky je možné ukotvit informační okno k libovolné instanci třídy `MVCObject`.
- `close()` – slouží k zavření informačního okna odstraněním ze struktury `DOM`
- `get` – metody typu `get` zde slouží k získání hodnoty vlastností `content`, `position`, nebo `z-index`. Jsou to tyto metody:
  - `getContent()`
  - `getPosition()`
  - `getZIndex()`
- `set` – metody tohoto typu slouží k nastavování vlastností.

- `setContent(content:string|Node)`
- `setPosition(position:LatLng)`
- `setZIndex(zIndex:number)`
- `setOptions(options:InfoWindowOptions)` – slouží k nastavení více vlastností najednou načtením objektu `InfoWindowOptions`

Stejně, jako jsem uváděl příklad k vytvoření mapy a markeru, uvedu zde i příklad na vytvoření a otevření informačního okna. Informační okno přiřadím k markeru, takže budu muset vytvořit `Listener`, který bude reagovat na událost `clicked` na markeru. V souboru `map.js` jsem tedy vytvořil novou funkci `createInfoWindow(marker)`, kterou jsem zavolal na posledním řádku funkce `addMarker()`.

```
function createInfoWindow(marker){
  var infowindow = new google.maps.InfoWindow({
    content: "New InfoWindow"
  });
  google.maps.event.addListener(marker, 'click', function() {
    infowindow.open(map, marker);
  });
}
```

Funkce `createInfoWindow(marker)` vytvoří nové informační okno, které bude obsahovat text „New InfoWindow“, a poté zavolá statickou metodu `addListener` z jmenného prostoru `google.maps.event`, která přiřadí uvedenou funkci k události `clicked` a k instanci marker. Jinými slovy – při kliknutí na marker se spustí funkce `open(map, marker)`, která otevře informační okno pro instanci `map` třídy `Map` a instanci `marker` třídy `Marker`. Pro úplnost zde ještě uvádím celou definici metody `addListener` a několika dalších užitečných funkcí z jmenného prostoru `google.maps.event`:

- `addListener(instance:Object, eventName:string, handler:Function)` – přiřadí funkci zadané události a zadané instanci a vrací identifikátor tohoto `Listeneru`, který může být použit funkcí `removeListener`
- `addListenerOnce(instance:Object, eventName:string, handler:Function)` – stejná jako `addListener`, ale po prvním použití odstraní handler
- `clearInstanceListeners(instance:Object)` – odstraní všechny `Listenery` zadané instance

- `clearListeners(instance:Object, eventName:string)` – odstraní všechny `Listenery` pro zadanou událost zadané instance
- `removeListener(listener:MapsEventListener)` – odstraní `Listener` podle identifikátoru, který vrátila funkce `addListener`

Obr. 2.3: Informační okno vytvořené funkcí `createInfoWindow(marker)`

Vzhledem k tomu, že akcí, které se dají provádět s informačním oknem, není mnoho, je seznam událostí třídy `InfoWindow` oproti událostem tříd `Map` a `Marker` dost omezený. Nejsou zde žádné události typu `clicked` či `mouseover`, protože ty se nevztahují na samotné okno, ale přímo na jeho obsah. Seznam událostí třídy `InfoWindow` je zde:

- `closeclick` – událost je spuštěna se stisknutím křížku v informačním okně
- `content_changed` – událost je spuštěna při změně vlastnosti `content`
- `domready` – událost je spuštěna v okamžiku, kdy je <`div`> obsahující `InfoWindow` přidán do `DOM`. Tato událost je velice užitečná zejména v případech, kdy je obsah oken sestavován dynamicky
- `position_changed` – událost je spuštěna při změně vlastnosti `position`
- `zindex_changed` – událost je spuštěna při změně vlastnosti `zIndex`

2.2.4 Využití Google Maps Javascript API V3 v aplikaci Culture

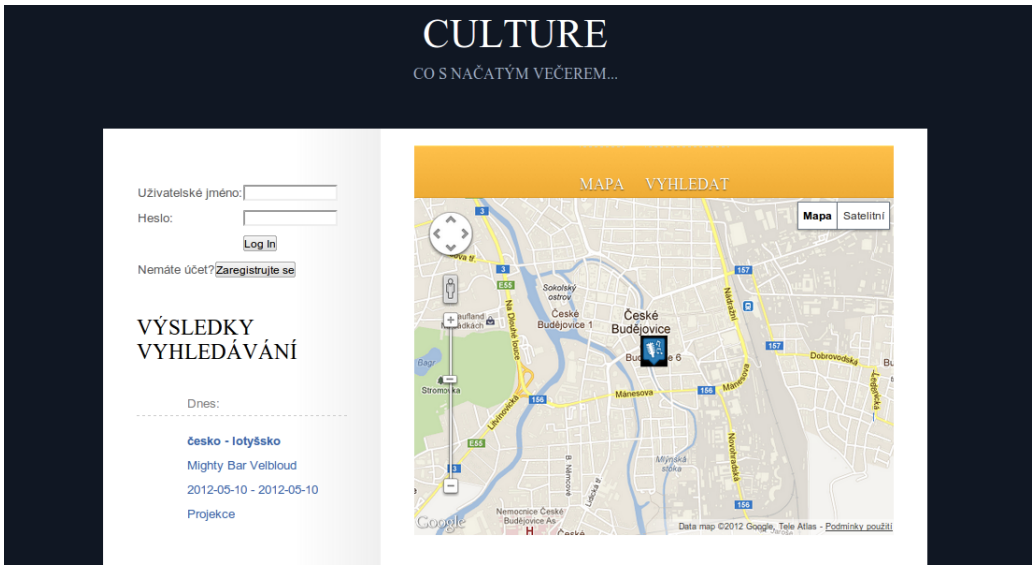
Úloha **Google Maps JavaScript API V3** v této bakalářské práci je dost zásadní. Jsou v ní použity všechny tři třídy, které byly popsány dříve, tedy třídy `Map`, `Marker` a `InfoWindow`. Všechny tři třídy jsou využívány pro vyhledávání kulturních událostí a pro přidávání míst jsou použity instance tříd `Map` a `Marker`.

Třída `Map` je poprvé použita hned v hlavním okně aplikace, kde je od začátku zobrazena hlavní mapa sloužící pro vyhledávání kulturních událostí. Při inicializaci této mapy jsou zobrazeny markery označující místa, ve kterých je pro aktuální datum naplánována nějaká kulturní akce. Pokud je v zadaném rozmezí na jednom místě akcí víc, marker se zobrazí pouze jeden. Pro každý marker je vytvořen `Listener`, který při kliknutí na marker otevře informační okno. Okno obsahuje seznam kulturních událostí na daném místě v zadaném časovém rozmezí. Obsah (`content`) okna je vkládán jako `html` kód, který obsahuje seznam (<`ul`>) a při kliknutí na položku seznamu (<`li`>) jsou zobrazeny podrobnosti o události.

Druhá mapa, která je v aplikaci vytvořena, slouží k vkládání nových míst. Je k dispozici pouze pro některé uživatele. K vkládání nových míst musí mít uživatel oprávnění. Mapa se nachází ve formuláři, kterým se místa přidávají. Při kliknutí do této mapy je na místě vytvořen marker označující vybranou polohu. Při opětovném kliknutí je původní marker smazán a nahrazen novým. Při vkládání místa do databáze je poté z markeru metodou `getPosition` převzata poloha. Souřadnice markeru jsou uloženy do databáze.

3 Aplikace Culture – uživatelský pohled

Aplikace Culture by měla sloužit k snadnému a rychlému vyhledávání kulturních akcí podle názvu, data, typu a místa konání. Zároveň by měla sloužit organizátorům ke snadnému přidávání kulturních událostí do databáze a k jejich propagaci. Celá aplikace by se tak dala rozdělit na dvě části. Jednou z nich je vyhledávání, druhou pak přidávání kulturních událostí. V této kapitole rozeberu obě dvě sekce z uživatelského hlediska, včetně jednoduchých návodů pro vyhledávání i vkládání.

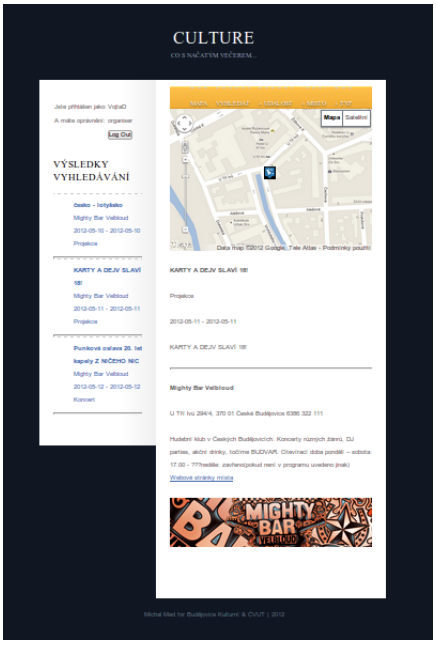


Obr. 3.1: Ukázka úvodní obrazovky uživatele s oprávněním `guest`

3.1 První pohled na aplikaci

Při prvním otevření aplikace návštěvníka uvítá logo se sloganem, při jehož stisknutí dojde k obnovení stránky. Pod logem už se nachází přihlašovací formulář a jednoduchá nabídka, která ovládá formuláře pod ní. Při načtení stránky je ve formuláři zobrazena mapa Českých Budějovic. Aplikace je rozdělena do dvou sloupců. V levém, užším, se nachází přihlašovací formulář a seznam vyhledaných kulturních událostí. V tom druhém, širším sloupci, je v záhlaví menu a pod ním prostor pro mapu nebo vyhledávací a vkládací formuláře. Ještě níže se nachází hlavní blok stránky, do kterého se vypisují podrobnosti o vybraných událostech. Už při spuštění aplikace je provedeno první vyhledávání, při kterém se ve

spodní části levého panelu zobrazí kulturní akce, které probíhají v aktuální den, a zároveň jsou v mapě zobrazeny markery míst, ve kterých se tyto události odehrávají.



Obr. 3.2: Náhled na celou stránku

3.2 Vyhledávání kulturních akcí

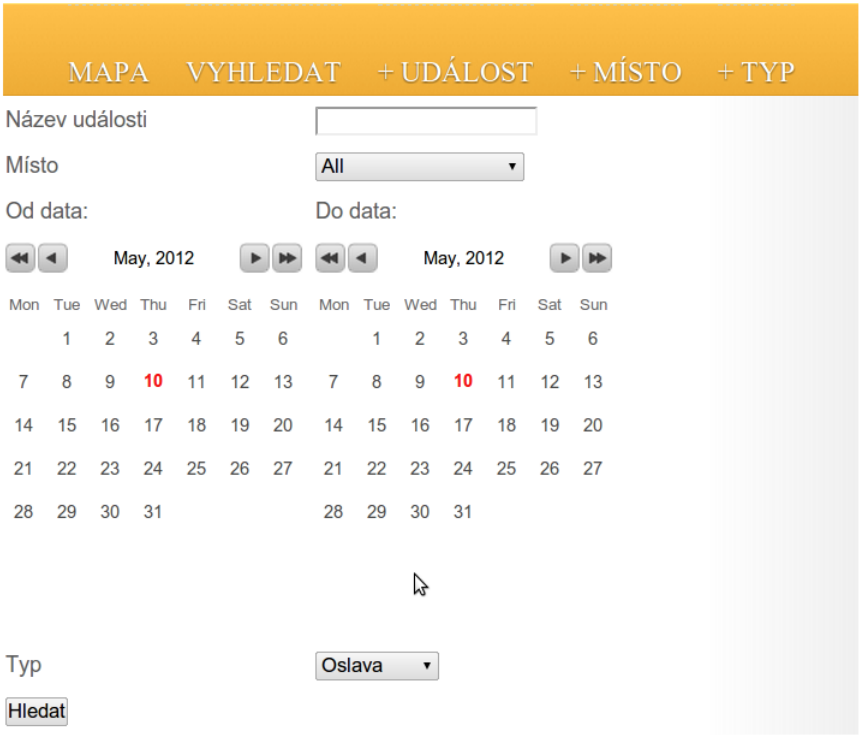
Vyhledávání kulturních událostí by mělo být nejpoužívanější a nejdůležitější součástí aplikace. Vyhledávat se dá nekolika různými způsoby. Při načtení stránky, při obnovení stránky, nebo při volbě „Mapa“ v hlavní nabídce se provede automatické vyhledávání. Možností pro podrobnější vyhledávání je potom vyhledávací formulář pod položkou „Vyhledat“. Obě tyto položky jsou přístupné i pro nepřihlášené uživatele.

3.2.1 Automatické vyhledávání

Automatické vyhledávání je provedeno vždy, když je znovu načtena úvodní stránka aplikace. Kromě prvního načtení s příchodem na stránky se automatické vyhledávání provede i při opětovném načtení stránky, při kliknutí myší na logo aplikace a při volbě možnosti „Mapa“ v hlavní nabídce. Při spuštění automatického vyhledávání jsou v databázi vyhledány všechny kulturní události, které se odehrávají v aktuální den nezávisle na místě

konání či typu události. Parametry výběru tedy splňují i události, které trvají déle než jeden den, ale probíhají v aktuální den.

3.2.2 Vyhledávání pomocí formuláře



Obr. 3.3: Vyhledávací formulář

K vyhledávání kulturních událostí je možmé využít vyhledávací formulář, který je možné otevřít stisknutím tlačítka „Vyhledat“ v menu nad zobrazenou mapou. Ve formuláři není žádná povinná položka. Pokud uživatel nevyplní žádný údaj, výsledkem vyhledávání bude seznam všech akcí probíhajících v následujícím týdnu. Vyhledávání kulturních událostí je možné podle následujících parametrů:

- Název události – upravuje dotaz přímo podle jména události. Slouží spíše pro vyhledávání podrobností pro již známou událost než pro zjišťování nových událostí.
- Místo – vybere pouze události, které se konají přímo na konkrétním místě. Místo je vybráno z roletového menu, kde je i možnost „Vše“.
- Od data/do data – určuje časové rozmezí vyhledávání událostí. Při nevyplnění data začátku je automaticky doplněno aktuální datum, při nevyplnění data ukončení

časového rozmezí je automaticky doplněno datum sedm dní po aktuálním datu. Datum je vybráno z kalendáře, vybrané datum je vypsáno pod ním.

- Typ – umožňuje vyhledávat události podle specifického typu. Typem zde může být například koncert, divadlo, vernisáž, promítání a podobně. Typ je vybrán z roletového menu, kde je i možnost „Vše“.

Samotné vyhledání události se provede až po potvrzení formuláře tlačítkem „Hledat“, které se nachází ve formuláři úplně dole.

3.2.3 Zobrazení výsledků

Obr. 3.4: Ukázka seznamu výsledků

Výsledky jsou vypsány v levém sloupci aplikace. Výsledky jsou seřazeny automaticky podle data začátku, což znamená, že na prvních místech budou většinou zobrazeny dlouhodobější události. V tomto sloupci jsou zobrazeny jen některé atributy. Jsou to název kulturní události, místo, kde se odehrává, datum konání a typ události. Více podrobností je možné zjistit při poklepání myší na položku v seznamu. Poklepání spustí několik dalších akcí. Za prvé je mapa posunuta a přiblížena k místu, kde se akce koná. To je vhodné pro snadnou orientaci. Kromě toho jsou v hlavním bloku aplikace zobrazeny podrobnosti jak o pořádané akci, tak o místě konání. U vybrané události jsou kromě informací, které byly zobrazeny už v seznamu událostí, navíc uvedeny podrobnosti o události,

odkaz na internetové stránky události a plakát. Dále jsou zde zobrazeny obecné informace o místě konání události. Je zde zobrazen název, adresa, popis místa (který by měl obsahovat charakteristiku místa, otvírací dobu, případně další informace, které by mohly potenciálního návštěvníka zajímat), odkaz na internetové stránky a logo nebo fotografie místa. Odkaz na internetové stránky události nebo místa je při poklepání myší automaticky

Obr. 3.5: Zobrazení podrobností o události

otevřen v novém okně, aby mohl uživatel dále nerušeně zobrazovat další události. Plakát se v hlavním bloku zobrazí pouze zmenšený, ale je možné prohlédnout si i obrázek v plné velikosti. Při poklepání na zmenšeninu plakátu je přes celou výšku obrazovky otevřena originální verze obrázku. Pokud je obrázek vyšší, než výška prohlížeče, objeví se vedle obrázku posuvná lišta, kterou je možné celý obrázek prohlédnout. Obrázek je zavřen, pokud uživatel poklepe myší na tlačítko „Zavřít“, nebo na samotný obrázek.

Vyhledávání i zobrazování akcí jsem se pokoušel udělat tak, aby bylo jednoduché a uživatelsky přívětivé. Co se týče intuitivnosti, věřím, že i bez nápovědy je snadno pochopitelné, jak vyhledat kulturní události v určitém časovém období. Propojení textového vyhledávání s mapou ulehčuje orientaci.

3.3 Vkládání kulturních událostí, míst a typů

Obr. 3.6: Formulář pro přidávání kulturních událostí

K vkládáním prvků do databáze musí být uživatel přihlášen s odpovídajícím oprávněním. V této aplikaci existují tři druhy oprávnění:

- `guest` – základní oprávnění. Všichni nepřihlášení uživatelé mají toto oprávnění. Uživatelé s oprávněním `guest` smí v databázi pouze vyhledávat a nemají žádnou možnost přidávat kulturní události, místa, ani nové typy.
- `organiser` – většina uživatelů, kteří budou pravidelně přispívat do databáze, bude mít toto oprávnění. Uživatel s oprávněním `organiser` může přidávat kulturní události přímo do databáze. S místy a typy je to odlišné. Organizátor může přidávat místa i typy do vedlejší tabulky, ve které není možnost vyhledávat. Z této vedlejší tabulky je nové místo nebo typ přesunuto do přístupné tabulky míst nebo typů až po potvrzení správce. Přidáním nového místa nebo typu uživatelem s oprávněním `organiser` je vygenerován e-mail na adresu kulturni.cross@gmail.com, který obsahuje odkaz, jehož otevření přesune nový prvek do přístupné tabulky.
- `admin` – uživatelé s tímto oprávněním můžou přidávat události, místa i typy přímo do přístupných tabulek a změny které provedly se ihned projeví. Toto oprávnění by neměl dostat nikdo jiný než provozovatel aplikace.

Pro přidávání kulturních událostí, míst a typů jsou v hlavní nabídce možnosti „+ Událost“, „+ Místo“ a „+ Typ“. Při jejich výběru jsou zobrazeny přidávací formuláře.

3.3.1 Vkládání nových kulturních událostí

Jak už jsem psal výše, k přidání události rovnou do přístupné tabulky stačí mít oprávnění `organiser`. Do budoucna by bylo výhodnější, aby každý organizátor měl možnost přidávat události jen na určitá místa. U každého místa by měl být seznam uživatelů, kteří k němu mohou přiřazovat události. Zatím může každý, kdo má opravnění `organiser`, přiřazovat událost k libovolnému místu. Jediným zaím fungujícím ošetřením tohoto problému je přidávání jména autora ke každé vytvořené události. Při případné sabotáži je tak možné vyhledat viníka a omezit mu oprávnění.

Samotné přidávání událostí probíhá pomocí formuláře. Ve formuláři je několik povinných a několik volitelných položek. Povinné položky jsou ty, které událost jednoznačně identifikují. Ve formuláři jsou označeny hvězdičkou.

- Název události – Určuje název, pod kterým bude událost uložena do databáze, pod kterým ji bude možné vyhledat a který bude zobrazen ve výsledcích.

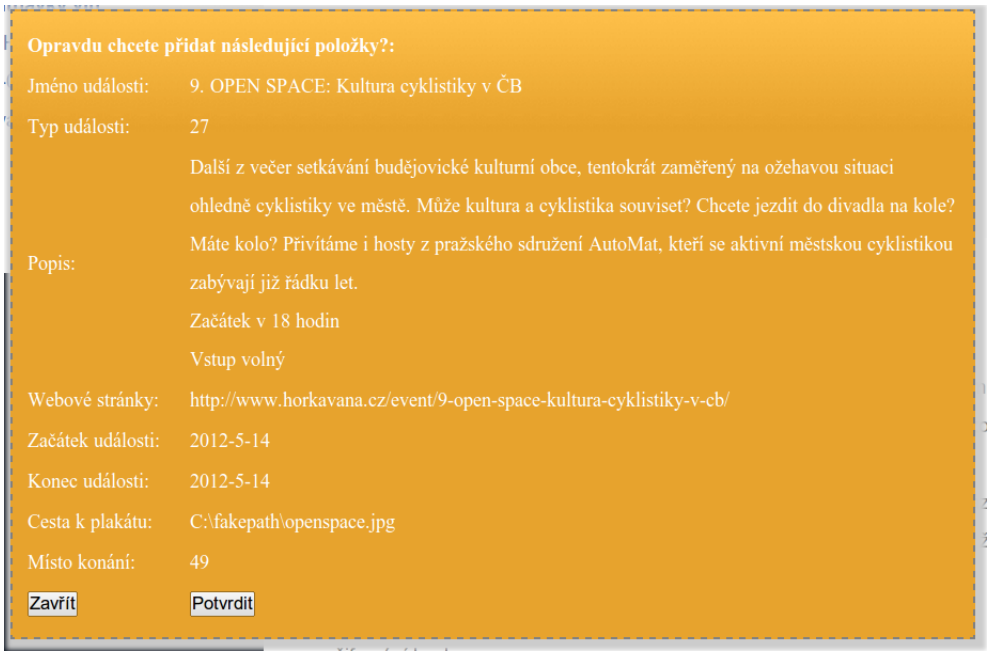
- Typ akce – Typ akce je vybírán z roletového menu. Typem události je například koncert, divadlo, vernisáž a podobně.
- Popis – V popisu události jsou uvedeny všechny informace, které jsou pro případné návštěvníky užitečné, jako popis události, čas začátku, cena vstupného a podobně. Pro zalomení odstavce v popisu události je třeba použít HTML tag `<br/>`.
- Datum začátku/datum konce – Údaje o začátku a ukončení akce jsou vybírány z kalendáře. U jednodenních akcí končících po půlnoci, což je časté třeba u koncertů, by mělo být datum ukončení uvedeno stejné jako datum začátku.
- Místo konání – Místo je vybíráno z roletového menu. Určuje, na kterém místě se bude událost odehrávat. Každá akce může mít přiřazeno pouze jedno místo.

Nepovinné položky jsou takové, které už jen prohlubují informace o události, ale nejsou nezbytné.

- Webové stránky události – Pokud má událost nějaké oficiální webové stránky, je vhodné je sem uvést. Při zobrazování podrobností je zobrazen hypertextový odkaz, který při kliknutí otevře novou kartu prohlížeče a otevře v ní oficiální stránku. Při zadávání adresy musí být na začátku uvedeno `http://`, jinak se prohlížeč pokusí dosadit adresu za aktuální pozici na serveru.
- Plakát – Ke každé kulturní události může být přiložen plakát ve formátu `.jpeg`, `.png` nebo `.gif`. Při vložení obrázku dojde k přizpůsobení velikosti do formátu 800×800 pixelů pro velký plakát a 300×300 pixelů na zmenšeninu.

Formulář se potvrzuje tlačítkem „Přidat“, které je umístěno pod formulářem. Po potvrzení formuláře se na stránce objeví ještě potvrzovací okno se souhrnem všech přidávaných položek a s tlačítky „Zavřít“ a „Potvrdit“.

Při kliknutí na tlačítko „Zavřít“ je formulář skryt. Uživatel může v přidávacím formuláři pozměnit parametry, které byly vyplněny nesprávně, nebo odejít. V potvrzovacím formuláři jsou dobře vidět úpravy vzhledu vytvořené HTML tagy. Po stisknutí tlačítka „Potvrdit“ je zkontrolováno, zda je formulář vyplněn správně, nechybí žádné povinné položky a vkládaný obrázek je v podporovaném formátu. Pokud nejsou všechny položky vyplněné, uživatel je na to upozorněn varovným oknem a záznam není přidán. U kulturních



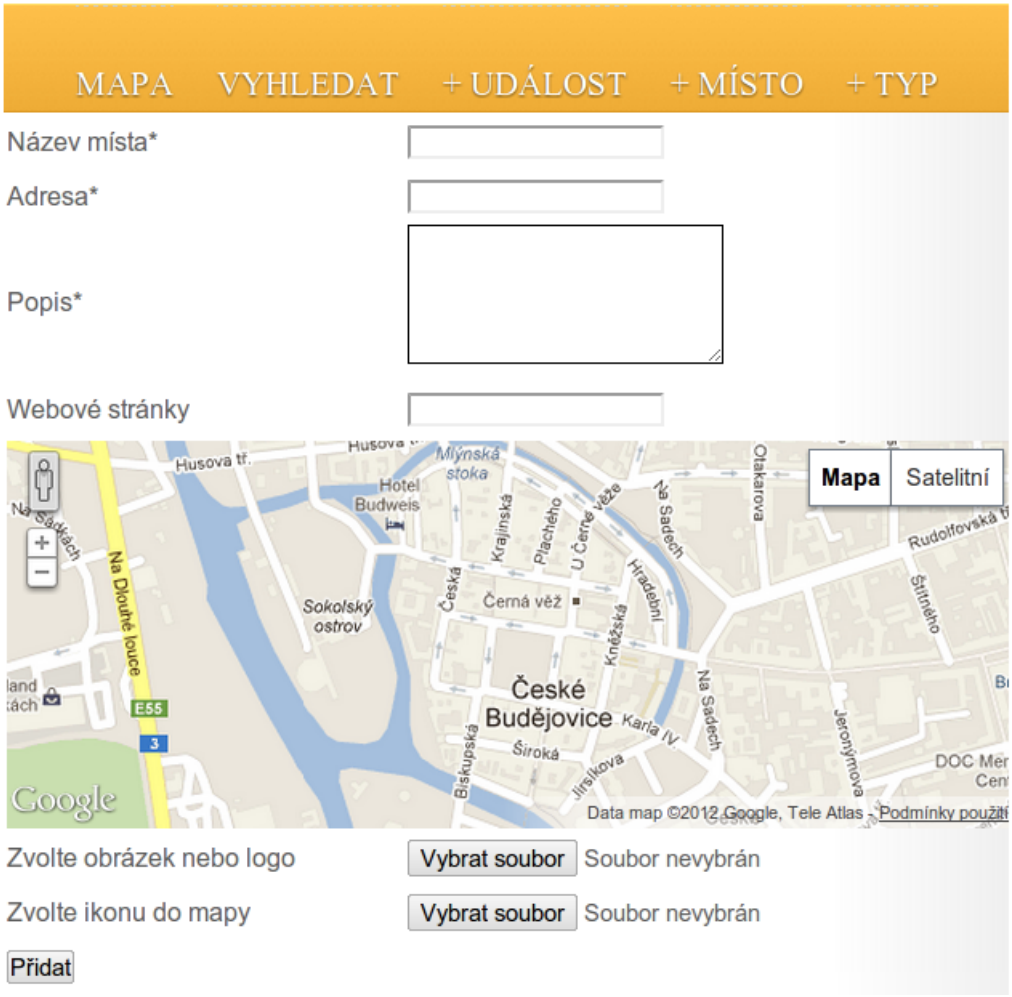
Obr. 3.7: Potvrzovací formulář přidávání akcí

událostí je možné, že se dvě budou jmenovat stejně (narozdíl od typů a míst), takže to není ošetřeno. Pokud jsou všechny položky vyplněny správně, záznam je přidán do databáze, a uživatel je na to upozorněn vyskakovacím oknem s textem „`Nová událost byla úspěšně přidána!`“. Po potvrzení potvrzovacího formuláře jsou změny okamžitě provedeny a je provedeno automatické vyhledávání. Tím je proces přidání kulturní události hotov.

3.3.2 Vkládání nových míst

Vkládat nová místa můžou uživatelé s oprávněním `organiser` nebo `admin`. Uživatelé s oprávněním `admin` vkládají místa přímo do přístupné tabulky, uživatelé s oprávněním `organiser` místa vkládají do vedlejší tabulky, kde jsou data uchovávaná, ale nepřístupná pro vyhledávání. Z této tabulky mohou být přesunuta do přístupné tabulky správcem aplikace. Způsob přidávání míst je však pro vlastníky obou oprávnění stejný. Slouží k tomu formulář pod tlačítkem „+ Místo“. Formulář obsahuje několik povinných a několik volitelných položek. Povinné položky jsou takové, které jednoznačně definují místo a jsou označeny hvězdičkou.

- Název místa – Určuje název, pod kterým bude místo uloženo do databáze, pod kterým ho bude možne přiřadit události a pod kterým ho bude možné vyhledávat.



Obr. 3.8: Formulář pro přidávání nových míst

Název místa musí být unikátní, to znamená, že dvě různá místa se nesmí jmenovat stejně.

- Adresa – Tato položka obsahuje adresu místa. Slouží jen k orientaci.
- Popis – Popis místa by měl obsahovat bližší informace o místě, jaká je jeho otevírací doba, v případě klubů a kaváren jaké točí pivo a podobně.
- Poloha – Poloha je vybrána na mapě. Kliknutím do mapy je vytvořen marker, ze kterého je přebírána poloha. Při novém kliknutí do mapy původní marker zmizí a je nahrazen novým. V mapě je možnost přepínat mezi typem `ROADMAP` a `SATELLITE`, aby bylo snažší přesně lokalizovat budovu. Mapa umožňuje přepnutí do Street View, ale v tomto módu není možné přidat marker.

Kromě povinných položek obsahuje formulář i nepovinné položky. Ty blíže specifikují charakter místa a přibližují ho potenciálním návštěvníkům.

- Webové stránky – Tato položka sice není povinná, ale rozhodně je doporučená. V dnešní době má většina míst, které pořádají pravidelně kulturní akce, své webové stránky. Adresa těchto stránek by měla být obsahem této položky. Při zobrazování podrobností je zobrazen hypertextový odkaz, který při kliknutí otevře novou kartu prohlížeče a v ní je otevřena webová stránka místa. Při zadávání internetové adresy musí být na začátku uvedeno `http://`, jinak se prohlížeč pokusí dosadit adresu za aktuální pozici na serveru.

- Obrázek nebo logo – Pokud má místo nějaké oficiální logo, mělo by být vloženo sem. Během vložení je velikost obrázku změněna do formátu 400×400 pixelů. Vkládaný obrázek může být ve formátu `.jpeg`, `.png` nebo `.gif`. Logo je poté zobrazováno při zobrazování výsledků v podrobnostech o místě. V případě, že místo žádné logo nemá, je možné vložit místo něj fotografii místa.

- Ikona v mapě – Obrázek, kterým bude místo reprezentováno v mapě. Mezi povolené formáty patří `.jpeg`, `.png` a `.gif`. Ikona je automaticky změněna velikost na 40×40 pixelů.

Formulář je potvrzen stisknutím tlačítka „Přidat“ ve spodní části formuláře. Po potvrzení odeslání je ještě zobrazen přehled všech přidávaných údajů. Ten slouží ke kontrole, zda je všechno vkládáno správně. V této fázi je ještě možné stisknout tlačítko „Zavřít“, vrátit se do přidávacího formuláře a upravit údaje. Pokud jsou všechny údaje vyplněny podle představ uživatele, vložení je potvrzeno stisknutím tlačítka „Přidat“. Aplikace zkontroluje, zda jsou správně vyplněny všechny údaje a zda jsou vkládané obrázky v podporovaném formátu. Pokud ne, uživatel je na to upozorněn varovným oknem a údaje nejsou přidány. Pokud nebyla při kontrole nalezena žádná chyba, je záznam přidán do odpovídající tabulky (závisí na oprávnění). Uživatel na to bude upozorněn vyskakovacím oknem s textem „`Nové místo bylo úspěšně přidáno!`“ (v případě oprávnění `admin`) nebo „`Nové místo bude přidáno po schválení administrátorem!`“ (v případě oprávnění `organiser`). Při úspěšném vložení místa je do tabulky navíc vloženo jméno uživatele, který místo přidal. Po potvrzení přidávacího formuláře je stránka znovu načtena a je provedeno automatické vyhledávaní událostí. Tím je přidání místa kompletní.

3.3.3 Vkládání nových typů

Pravidla pro přidávání nových typů jsou stejná jako pravidla pro přidávání míst. Uživatelé s oprávněním typu `organiser` vkládají typy do vedlejší tabulky, která není přístupná pro vyhledávání. Uživatelé s oprávněním typu `admin` vkládají nové typy přímo a nové typy jsou rovnou přístupné pro další akce. K vkládání nových typů kulturních událostí slouží formulář, který je přístupný z hlavní nabídky pod tlačítkem `+ Typ`. Formulář pro přidávání nových typů má pouze jednu položku, která je povinná.

- Název nového typu – Určuje název nového typu, který je zobrazován ve formuláři pro vyhledávání, je přiřazován jednotlivým kulturním událostem a je zobrazován v podrobnostech události. Název typu musí být původní, což znamená, že dva typy se nesmí jmenovat stejně. Vzhledem k tomu, že název je jediná vlastnost typu, bylo by nesmyslné, aby se dva typy jmenovaly stejně. Vzhledem k charakteru typu se zde neuvádí autor.

Obr. 3.9: Formulář pro přidávání nových typů

Nový typ je přidán tlačítkem „Přidat“ a následným potvrzením kontrolního formuláře. V případě nevyplnění jediné položky je uživatel upozorněn varovným oknem a nový typ není přidán. V opačném případě je nový typ přidán a ve varovném okně je nápis „`Nový typ byl úspěšně přidán!`“. Po potvrzení přidávacího formuláře je stránka znovu načtena a je provedeno automatické vyhledávaní událostí. Tím je přidání nového typu kompletní.

3.4 Registrace a přihlášení

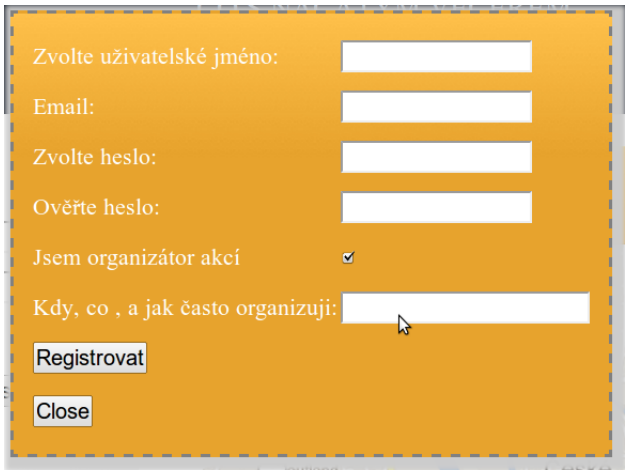
V předchozím textu jsem se už zmínil o přihlašování a o uživatelských právech. Uživatel, který otevře aplikaci poprvé, žádné uživatelské jméno, heslo ani oprávnění nemá. K získání těchto údajů se musí zaregistrovat. Při prvním spuštění aplikace se v levém horním rohu

nachází přihlašovací formulář. Ve formuláři se nachází pole pro uživatelské jméno, pole pro heslo, do kterého je samozřejmě heslo vpisováno skrytě, a tlačítka pro potvrzení hesla a pro registraci.

Obr. 3.10: Formulář pro přihlášení

3.4.1 Registrace

K registraci slouží registrační formulář, který se objeví po stisknutí tlačítka pro registraci. Všechny položky ve formuláři jsou povinné, kromě položky „`Kdy, co a jak často organizuji`“. Ta se objeví pouze po zatrhnutí položky „`Jsem organizátor akcí`“, po jejím zatrhnutí je ale také povinná. Položky ve formuláři jsou tyto:



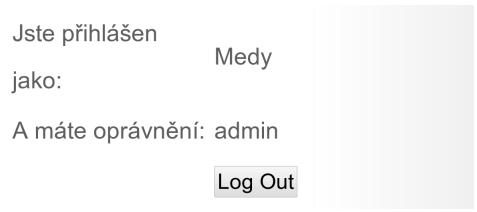
Obr. 3.11: Formulář pro přihlášení

- Uživatelské jméno – Uživatelské jméno musí být původní. Každému přihlášenému uživateli se v aplikaci zobrazuje jeho přihlašovací jméno a jeho oprávnění. Navíc je uživatelské jméno přidáno ke každé události i místu, které uživatel přidá.

- Email – Do tohoto pole uživatel zadá svou emailovou adresu. Adresa se nijak neověřuje, ale pokud uživatel požádá o oprávnění typu `organiser`, nebo bude přidávat místa a typy, v potvrzovacím emailu bude tato adresa uvedena jako odesílatel a provozovatel aplikace na tuto adresu může posílat případné připomínky.
- Heslo – Heslo, kterým uživatel při přihlašování potvrzuje svoji identitu. Do databáze se heslo ukládá zašifrované.
- Potvrzení hesla – Potvrzení předchozí položky, aby se uživatel kvůli překlepu nezaregistroval s heslem, které si nepamatuje.
- Kdy, co, a jak často organizuji – Tato položka je volitelná. K jejímu zobrazení je třeba zaškrtnout možnost „`Jsem organizátor akcí`“. To, co uživatel napíše do této položky, je poté odesláno emailem administrátorovi aplikace pro posouzení, zda mu přidělit oprávnění `organiser`, nebo ne.

Po stisknutí tlačítka „Registrovat“ aplikace zkontroluje, zda jsou vyplněny všechny položky, zda se heslo shoduje s jeho potvrzením a zda už neexistuje uživatel se stejným uživatelským jménem. Pokud je nalezen konflikt, je na něj uživatel upozorněn varovným oknem. Pokud k žádnému konfliktu nedojde, je uživatel přidán do databáze a může se ihned přihlásit s oprávněním `guest`. Pokud požádal o oprávnění `organiser`, musí počkat, dokud jeho žádost nepotvrdí administrátor. Do té doby má oprávnění `guest`.

3.4.2 Přihlášení



Obr. 3.12: Informace o přihlášeném uživateli

K přihlášení stačí zadat do přihlašovacího formuláře přihlašovací jméno a odpovídající heslo. Pokud všechno proběhne v pořádku, je uživatel přihlášen a místo přihlašovacího formuláře jsou zobrazeny informace o přihlášeném uživateli a tlačítko pro odhlášení. Pokud je při přihlášení špatně zadané přihlašovací jméno nebo heslo, uživatel není přihlášen

a je upozorněn varovným oknem. V závislosti na oprávnění je také znovu vygenerována hlavní nabídka. Uživatelé s oprávněním `admin` a `organiser` v ní navíc naleznou položky pro přidávání kulturních událostí, míst a typů. Stisknutím tlačítka „`Log Out`“ je uživatel odhlášen. Na místě informací o uživateli se znovu objeví přihlašovací formulář.

4 Aplikace Culture – technické řešení

K tomu, aby vznikla aplikace Culture, bylo zapotřebí použít několik programovacích jazyků a databázi. Data, se kterými aplikace pracuje, jsou uložena v databázi `PostgreSQL`. Použité programovací jazyky jsou `PHP`, `JavaScript` a `HTML`. `HTML` tagy jsou v této aplikaci generovány pomocí `PHP` nebo `JavaScriptu`. Kromě toho je součástí aplikace i `CSS` soubor, který zajišťuje vzhled. V této kapitole rozeberu podrobněji databázi a každou část aplikace podle použitého programovacího jazkyka.

4.1 Databáze PostgreSQL

V této části se budu věnovat obecně databázi `PostgreSQL` a hlavně jejímu využití v této bakalářské práci. `PostgreSQL` je objektově relační databáze s velkým množstvím možností a rozšíření.

4.1.1 Z historie PostgreSQL

Projekt PostgreSQL je jedním z největších a nejpoužívanějších Open Source projektů na světě. Jedná se o objektovou relační databázi, která vznikla na University of California at Berkeley (UCB). V roce 1986 byla profesorem informatiky Michaelem Stonebrakerem vytvořena databáze **Postgres**. Vycházela z projektu **Ingres** a odtud pochází i její název. **Ingres** vznikal na UCB v letech 1977–1985 pro cvičení tvorby databázových systémů. `Postgres` vznikal v letech 1986–1994 s cílem položit nové základy pro databáze jako objektové relace. V roce 1995 byl dosavadní jazyk, kterým byly pokládány dotazy, nahrazen jazykem `SQL` a přejmenován na `Postgres95`. V roce 1996 opustil `Postgres95` zdi univerzity a začal se dále vyvíjet jako Open Source projekt. Dalšího vývoje `Postgresu` se ujala skupina vývojářů mimo Berkeley, která sestávala z lidí z téměř celého světa. Do projektu vložili hodně času a schopností a radikálně `Postgres` změnili. Více jak osm let sjednocovali kód, sestavili mailing list pro oznamování bugů, valnou většinu bugů opravili, přidali mnoho nových vlastností a sestavili rozsáhlou dokumentaci pro vývojáře i uživatele (současná verze dokumentace má téměř dva a půl tisíce stran). Díky jejich tvrdé práci se z `PostgreSQL` stal databázový systém s rozsáhlou podporou a reputací solidního a velice

stabilního projektu. Od té doby je PostgreSQL neustále vylepšován a používán mnoha společnostmi, organizacemi a úřady po celém světě.

Obr. 4.1: Symbolem projektu PostgreSQL je slon, který je známý svou dokonalou pamětí

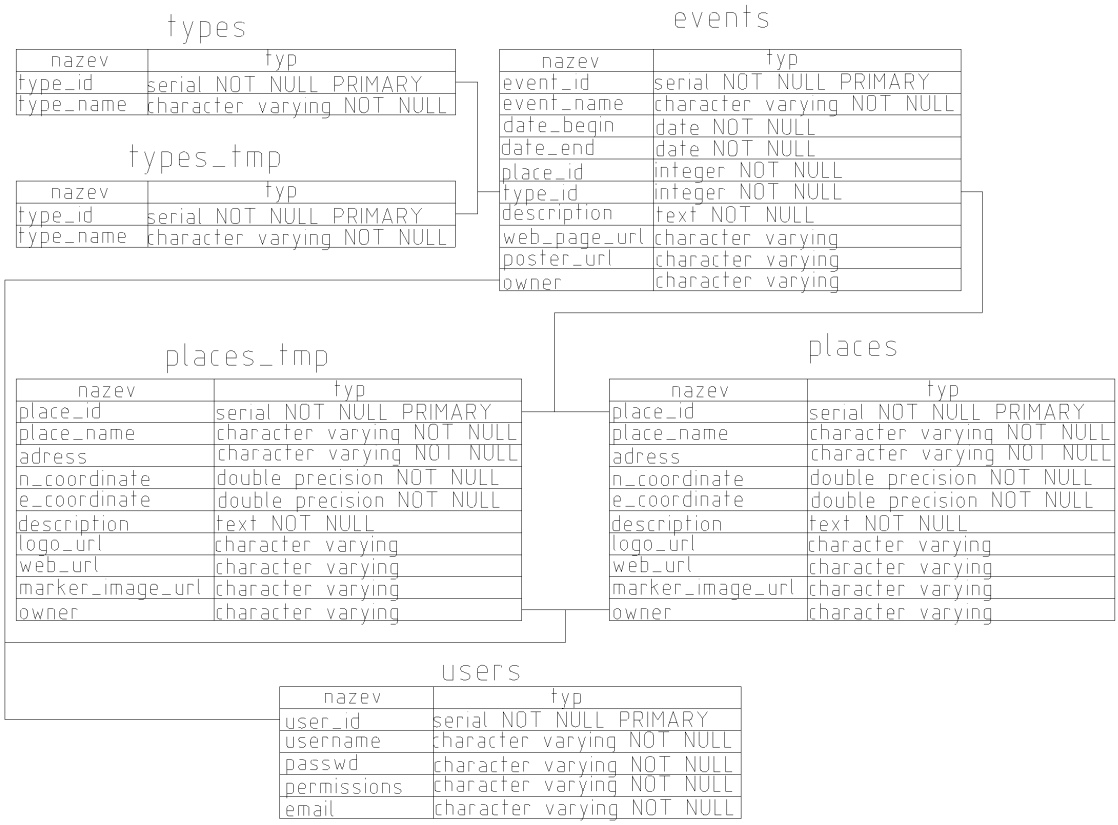
4.1.2 Struktura databáze v aplikaci Culture a vyhledávací dotaz

Databáze aplikace Culture obsahuje šest propojených databázových tabulek. Tři z těchto tabulek (tabulky `events`, `places` a `types`) jsou používány pro vyhledávání kulturních akcí, dvě z nich (`places_tmp` a `types_tmp`) slouží k přidávání nových záznamů uživateli s oprávněním `organiser`. Poslední tabulka `users` obsahuje data o uživatelích a slouží mimo jiné k přihlašování. Při vyhledávání kulturních událostí pomocí formuláře je přes PHP skript proveden následující databázový dotaz:

```
SELECT p.place_name AS p_name, p.description AS p_desc, e.event_name AS e_name, * FROM places
AS p JOIN events AS e ON p.place_id = e.place_id JOIN types ON e.type_id = types.type_id WHERE
e.event_name LIKE '$event_name%' AND p.place_id = $place_id  AND (e.date_begin, e.date_end)
OVERLAPS (DATE '$beg_date', DATE '$end_date') AND e.type_id=$genre ORDER BY e.date_begin;
```

Pro úplnost musím doplnit, že tvar dotazu závisí na tom, které položky vyhledávacího formuláře jsou vyplněny. Proměnné `$event_name`, `$place_id`, `$beg_date`, `$end_date` a `$genre` jsou převzaty z vyhledávacího formuláře metodou `GET`. Z dotazu je vidět, že zatímco jméno události (proměnná `$event_name`) je předáváno jako `string`, typ události a místo jsou předávány jako číslo. To je způsobeno tím, že ve vyhledávácím formuláři jsou jména místa a typu události vybírány z roletového menu. Z tohoto menu se do PHP skriptu nepředává název, ale hodnota zvolené položky, která je v těchto roletových menu reprezentována hodnotou `id`. V dotazu jsou použity následující příkazy a operátory jazyka SQL:

- `SELECT` – Značí, že následující dotaz bude v databázi vyhledávat.



Obr. 4.2: Názorné zobrazení tabulek a vztahů mezi nimi

- `AS` – Alias. Slouží pouze ke zkrácení dotazů. Libovolná tabulka nebo atribut může být pomocí příkazu `AS` v dotazu nahrazena názvem aliasu.
- `FROM` – Určuje, z které tabulky či tabulek budou hledány odpovídající výsledky.
- `JOIN ON` – Slouží k propojování tabulek. Parametrem klauzule `JOIN` je jméno připojované tabulky. Po příkazu `ON` musí být uvedeny atributy, kterými budou propojeny řádky tabulek.
- `WHERE` – Po klauzuli `WHERE` jsou uvedeny omezující podmínky dotazu.
- `LIKE` – Operátor sloužící k porovnání textových řetězců (datové typy `character varying`, `text` a další). Číslené hodnoty jsou porovnávány operátory $=$, $<$ a $>$.
- `AND` – Operátor spojující jednotlivé podmínky. Při použití operátoru `AND` musí být splněny všechny podmínky. Dalším použitelným operátorem je `OR`, při jehož použití stačí, aby byla splněna alespoň jedna podmínka.

- `OVERLAPS` – Operátor sloužící k porovnání časového rozsahu. Z každé strany operátoru `OVERLAPS` musí být dvojice hodnot typu `DATE`. Operátor `OVERLAPS` porovnává, zda se časové rozsahy určené těmito dvojicemi překrývají, nebo ne.
- `ORDER BY` – Tato klauzule slouží k seřazení výsledků podle hodnoty zadaného parametru.

V tomto příkazu jsou pomocí klauzule `JOIN` propojeny tabulky `events`, `places` a `types`. Klíčem k propojení tabulek `events` a `places` je atribut `place_id`, k propojení tabulek `events` a `types` je použit klíčový atribut `type_id`. Výsledky jsou seřazeny podle atributu `date_begin`. K tomu je použita klauzule `ORDER BY`.

Pro vyhledávání událostí slouží ještě jeden dotaz. Je proveden vždy, když je spuštěno automatické vyhledávání. Vzhledem k tomu, že je spouštěn automaticky, nemá žádné volitelné parametry. Vybírá z databáze položky pouze v závislosti na aktuálním datu.

```
SELECT p.place_name AS p_name, p.description AS p_desc, e.event_name AS e_name, * FROM places
AS p JOIN events AS e ON p.place_id = e.place_id JOIN types ON e.type_id = types.type_id WHERE
(e.date_begin, e.date_end) OVERLAPS (DATE '$beg_date', DATE '$end_date') ORDER BY e.date_begin;
```

Výsledkem tohoto dotazu budou informace o událostech, které se odehrávají v určitém časovém úseku a jejich typech a místech konání. Proměnné `$beg_date` a `$date_end` jsou v `JavaScriptu` nastaveny na hodnotu aktuálního data a hodnotu data týden po aktuálním datu.

4.1.3 Podrobný rozbor tabulky `events` a dotaz pro přidání události

Tabulka `events` obsahuje veškeré informace o událostech. Obsahuje následující atributy, které reprezentují jednotlivé informace události:

- `event_id` – typ `SERIAL` – Primární klíč tabulky. Jednoznačně definuje každý záznam. Každému záznamu je automaticky přiděleno unikátní číslo, které ještě žádnému záznamu nebylo přiděleno. To znamená, že i v případě, kdy jsou všechny záznamy smazány, další vložený záznam dostane přidělené `id` o jedna vyšší, než bylo `id` posledního záznamu, který byl přidán.

- `event_name` – typ `CHARACTER VARYING` – Hodnota tohoto atributu definuje jméno každé události.
- `date_begin` – typ `DATE` – Definuje datum, ve kterém událost začíná. Datum je ve formátu `YYYY-MM-DD`.
- `date_end` – typ `DATE` – Definuje datum, ve kterém událost končí. Datum je ve formátu `YYYY-MM-DD`.
- `description` – typ `TEXT` – Tento atribut obsahuje podrobné informace o události.
- `place_id` – typ `INTEGER` – Obsahuje `id` místa, ke kterému je událost přiřazena. Pomocí tohoto atributu jsou ke každé události přiřazovány informace o místu, ke kterému je přiřazena.
- `type_id` – typ `INTEGER` – Obsahuje `id` typu, ke kterému je událost přiřazena. Pomocí tohoto atributu jsou ke každé události přiřazovány informace o typu, ke kterému je přiřazena.
- `web_page_url` – typ `CHARACTER VARYING` – V tomto atributu jsou uchovávány internetové adresy.
- `poster_url` – typ `CHARACTER VARYING` – Obsahuje relativní cestu ke grafickému souboru s plakátem uloženému na serveru.
- `owner` – typ `CHARACTER VARYING` – Obsahuje jméno uživatele z tabulky `users`, který událost vytvořil.

Uživatelé mohou v tabulce `events` údaje vyhledávat nebo je do ní vkládat. Vyhledávání záznamů jsem popsal již dříve. K vkládání záznamu je používán skript v jazyce `PHP`, který obsahuje následující `SQL` dotaz:

```
INSERT INTO events (event_name, date_begin, date_end, place_id, type_id, description,
web_page_url, poster_url, owner) VALUES ('$event_name', DATE '$date_begin', DATE
'$date_end', $place_id, $type_id, '$description', '$event_URL', '$event_poster',
'$owner');
```

Příkaz `INSERT` slouží k přidávání záznamů do databáze. Klauzule `INTO` definuje tabulku, do které budou hodnoty vloženy. Za názvem tabulky je v závorce výčet atributů, do kterých

budou záznamy přidávány, a který také určuje jejich pořadí. Klauzule `VALUES` definuje hodnoty, které budou do tabulky zapsány. Podle pořadí hodnot v závorce jsou hodnoty přiřazovány k atributům. Hodnoty proměnných přidávaných do databáze jsou převzaty z `HTML` formuláře metodou `POST`.

4.1.4 Podrobný rozbor tabulek `places` a `places_tmp` a dotaz pro přidání místa

Tabulka `places` obsahuje informace o místech. Slouží pro vyhledávání akcí a vkládají se do ní nová místa. Tabulka `places_tmp` je totožná, až na to, že neslouží k vyhledávání, ale pouze k vkládání. Uživatelé s oprávněním `organiser` totiž do tabulky `places` nemohou přidávat události přímo. Místa jimi přidaná jsou přidána pouze do tabulky `places_tmp`. V případě, že administrátor potvrdí přidání místa, je spuštěn následující `PHP` skript (zkráceno).

```
<?php
require("src/class.pgsql.php");
$p = new pgsql;
$place_id = $_GET['place_id'];
$p->query("SELECT * FROM places_tmp WHERE place_id=$place_id");
...
}
$p->query("SELECT * FROM places WHERE place_name LIKE '$place_name'");
if($p->nextrow()){
    $p->query("DELETE FROM places_tmp WHERE place_name LIKE '$place_name'");
    ...
}
else{
    $p->query("DELETE FROM places_tmp WHERE place_id = $place_id");
    $p->query("INSERT INTO places (place_name, adress, n_coordinate, e_coordinate, description,
    marker_image_url, logo_url, web_url, owner) VALUES ('$place_name',  '$adress', $n_coordinate,
    $e_coordinate, '$description', '$icon_target', '$image_target', '$place_url', '$owner');");
    ...
}
?>
```

Skript se připojí k databázi (k tomu slouží třída `Pgsql`) a zkontroluje, zda v tabulce `places` už místo stejného názvu neexistuje. Pokud ano, je záznam z tabulky `places_tmp` vymazán a do tabulky `places` není vložen. V opačném případě je záznam odstraněn z tabulky `places_tmp` a je vložen do tabulky `places`.

Obě dvě tabulky tedy obsahují stejné atributy.

- `place_id` – typ `SERIAL` – Primární klíč tabulky. Jednoznačně definuje každý záznam v tabulce `places` nebo `places_tmp`. Každému záznamu je automaticky přiděleno unikátní číslo podle stejných pravidel, jako je přidělováno `event_id` v tabulce `events`.
- `place_name` – typ `CHARACTER VARYING` – Název místa. V databázi není ošetřeno, že by se dvě místa nemohla jmenovat stejně, ale při přístupu do databáze přes `PHP` skripty aplikace Culture (což je pro všechny uživatele vlastně jediná možnost) to ošetřeno je. V tabulce by se tedy neměly objevit dvě místa se stejnou hodnotou atributu `place_name`.
- `adress` – typ `CHARACTER VARYING` – Tento atribut uchovává adresu každého místa.
- `n_coordinate` – typ `DOUBLE PRECISION` – Tento atribut uchovává hodnotu zeměpisné šířky v systému `WGS-84`. Vzhledem k zeměpisným šířkám, pro které je aplikace určena, název odkazuje na severní polokouli, ale samozřejmě je možné vkládat i místa na polokouli jižní. Stačí k tomu zapsat hodnotu se záporným znaménkem.
- `e_coordinate` – typ `DOUBLE PRECISION` – Uchovává hodnotu zeměpisné délky v systému `WGS-84`. Stejně jako předchozí atribut může obsahovat i záporné hodnoty.
- `description` – typ `TEXT` – Obsahuje podrobné informace o místě.
- `logo_url` – typ `CHARACTER VARYING` – Obsahuje relativní cestu ke grafickému souboru s logem místa uloženému na serveru.
- `web_url` – typ `CHARACTER VARYING` – V tomto atributu je uchovávána internetová adresa místa.
- `marker_image_url` – typ `CHARACTER VARYING` – Obsahuje relativní cestu ke grafickému souboru s mapovou ikonou uloženému na serveru.
- `owner` – typ `CHARACTER VARYING` – Obsahuje jméno uživatele z tabulky `users`, který místo vytvořil.

Jako poslední věc, kterou je potřeba zmínit k tabulkám `places` a `places_tmp`, je způsob vkládání nových záznamů do těchto tabulek. Ke vkládání záznamů slouží `PHP` skript, který nejprve zkontroluje oprávnění uživatele, který nové místo přidává. Podle toho je nový

záznam přidán buď do tabulky `places`, nebo `places_tmp`. `SQL` dotaz sloužící k přidání záznamu do tabulky se potom liší jenom v tabulce, do které je záznam přidán.

```
INSERT INTO places_tmp (place_name, adress, n_coordinate, e_coordinate, description,
marker_image_url, logo_url, web_url, owner) VALUES ('$place_name',  '$adress',
$n_coordinate,$e_coordinate, '$description', '$icon_target', '$image_target',
'$place_url', '$owner');
```

V tomto případě jsou záznamy vloženy do tabulky `places_tmp`. Hodnoty proměnných přidávaných do databáze jsou převzaty z `HTML` formuláře metodou `POST`.

4.1.5 Podrobný rozbor tabulek `types` a `types_tmp` a dotaz pro přidání typu

Tabulky `types` a `types_tmp` obsahují stejné atributy, rozdíl mezi nimi je stejný jako mezi tabulkami `places` a `places_tmp`. Místa přidaná uživatelem s oprávněním `organiser` do tabulky `types_tmp` jsou po potvrzení administrátorem převedena do tabulky `places` pomocí `PHP` skriptu. Skript pracuje stejně jako skript pro přesunování záznamů z tabulky `places_tmp` do tabulky `places`.

Tabulky `types` a `types_tmp` obsahují jen dva atributy.

- `type_id` – typ `SERIAL` – Primární klíč tabulky. Jednoznačně definuje každý typ. Pravidla přidělování hodnoty jsou stejná jako u předchozích tabulek.
- `type_name` – typ `CHARACTER VARYING` – Název typu. Databáze nikomu nebrání, aby byly dva názvy totožné, ale v `PHP` skriptech, které k tabulce přistupují, je to ošetřeno, takže by se v tabulce neměly objevit dva záznamy se stejným názvem. Vzhledem k tomu, že název je jediným atributem tabulky, který může uživatel vidět, nedávalo by ani smysl, aby v tabulce byly dva typy se stejnou hodnotou atributu `type_name`.

Dotazy, kterými jsou přidávány záznamy do tabulek `types` a `types_tmp`, se liší zase jenom v tom, do které tabulky jsou data přidávána. To závisí na oprávnění uživatele. Dotaz je velice krátký.

```
INSERT INTO types (type_name) VALUES ('$genre_name');
```

4.1.6 Podrobný rozbor tabulky `users`, dotaz pro registraci a dotaz pro přihlášení

Tabulka `users` se od ostatních tabulek v mnohém liší. V tabulce jsou uchovávány informace o uživatelích. Při registraci uživatele jsou do tabulky uloženy přihlašovací údaje. Při přihlašování jsou tyto údaje v tabulce vyhledány a porovnány s hodnotami z přihlašovacího formuláře. Tabulka `users` obsahuje tyto atributy:

- `user_id` – typ `SERIAL` – Primární klíč tabulky. Jednoznačně definuje každého uživatele. Přiřazování hodnoty `user_id` se řídí stejnými pravidly jako v předchozích tabulkách.
- `username` – typ `CHARACTER VARYING` – Uživatelské jméno. Při přihlašování je podle něj vyhledán záznam v tabulce. Při přidávání nových kulturních událostí a míst je vloženo do nového záznamu do atributu `owner`.
- `passwd` – typ `CHARACTER VARYING` – Zašifrované heslo. Kvůli bezpečnosti není do tabulky zadáváno heslo tak, jak ho uživatel napíše, ale je zašifrováno `PHP` funkcí `crypt(string $str [, string $salt ] )`. Při ověřování hesla je potom podle stejného klíče zašifrována hodnota zadaná uživatelem do přihlašovacího formuláře a je porovnána s hodnotou atributu `passwd` pro odpovídající uživatelské jméno.
- `permissions` – typ `CHARACTER VARYING` – Podle hodnoty tohoto atributu získává uživatel přístup k formulářům pro přidávání údajů do tabulek `events`, `places`, `places_tmp`, `types` a `types_tmp`. Hodnoty oprávnění jsou následující:
  - `guest` – nesmí přidávat záznamy do žádné tabulky
  - `organiser` – smí přidávat záznamy do tabulek `events`, `places_tmp` a `types_tmp`
  - `admin` – smí přidávat záznamy do tabulek `events`, `places` a `types`
- `email` – typ `CHARACTER VARYING` – Obsahem tohoto atributu je emailová adresa uživatele. Emaily, které jsou generované při žádosti o oprávnění `organiser` nebo při přidávání míst do tabulky `places_tmp` a `types_tmp`, mají v položce odesílatel tuto adresu.

Využití této tabulky v aplikaci je především při registraci nových uživatelů a při jejich přihlašování. Pokud uživatel nemá uživatelské jméno ani heslo, může se zaregistrovat.

K registraci slouží následující PHP skript (zkrácený o kód, který není pro problematiku databáze důležitý):

```
<?php
require("src/class.pgsql.php");
$p = new pgsql;
...
if (!empty($username) && !empty($new_passwd) && !empty($check_passwd) && !empty($email))
{
    if($new_passwd === $check_passwd)
    {
        $q->query("SELECT username FROM users;");
        while($q->nextrow()){
            $act_username = $q->getfield("username");
            if ($act_username == $username)
            $unique_name = false;
        }
        if ($unique_name == true)
        {
            $p->query("INSERT INTO users (username, passwd, permissions, email) VALUES ('$username',
            '$crypted_passwd', '$permissions', '$email');");
            ...
        }
        ...
    }
    ...
}
...
?>
```

Tento skript provede připojení k databázi a do proměnných předá hodnoty z registračního formuláře. Poté zkontroluje, zda jsou vyplněny všechny položky, zda jsou hodnoty hesla a potvrzeného hesla stejné a zda uživatelské jméno v databázi již není. Pokud jsou všechny podmínky splněny, jsou údaje o uživateli přidány do tabulky `users` a uživatel se již může přihlásit. Přihlašování probíhá přes přihlašovací formulář. Hodnoty z formuláře jsou předány metodou GET tomuto PHP skriptu (zde je zkrácený o kód, který se nevztahuje k problematice databází):

```
<?php
require("src/class.pgsql.php");
$p = new pgsql;
...
if (!empty($username) && !empty($crypted_passwd)){
    $p->query("SELECT * FROM users WHERE username LIKE '$username';");
```

```
        ...
        elseif($passwd == $crypted_passwd) {
            session_start();
            $_SESSION['permissions']=$permission;
            $_SESSION['username']=$username;
            ...
        }
        else {
        ...
        }
    }
    ...
    ?>
```

Tento skript se připojí k databázi, vyhledá zašifrované heslo náležící uživatelskému jménu a porovná se zašifrovaným heslem z přihlašovacího formuláře. V případě, že jsou hesla shodná, je do asociativního pole `$_SESSION` zapsáno uživatelské jméno a oprávnění. Tím je uživatel přihlášen. Další chování stránky po přihlášení uživatele řídí `PHP`.

4.2 Použití `PHP` v aplikaci Culture

`PHP`, nebo–li „*Hypertext Preprocessor*“, je skriptovací jazyk, který je spouštěn na straně serveru (narozdíl třeba od `JavaScriptu`, který je spouštěn z prohlížeče). Kód je kompilován na straně serveru, takže na serveru musí být nainstalován kompilátor. V aplikaci Culture je `PHP` použito k sestavení `HTML` kódu stránky a k práci s databází.

Obr. 4.3: Logo `PHP`

4.2.1 Třída `Builder` a sestavení stránky

Pro sestavení kódu stránky jsem napsal třídu `Builder`, která obsahuje řadu členských metod generujících `HTML` kód. Metody jsou vytvořeny tak, aby dokázaly vygenerovat stránku pro uživatele všech oprávnění. Členské metody třídy `Builder` všechny obsahují pouze `HTML` kód. Na příkladech ukážu dvě z nich, všechny ostatní se chovají a jsou používány úplně stejným způsobem.

- `build_head()` – Tato metoda sestaví hlavičku dokumentu. Obsahuje seznam stylů (`CSS`) a `JavaScript`ových souborů a také <`meta`> tagy. Jediné co obsahuje je `HTML` kód, stejně jako všechny ostatní metody třídy `Builder`.

```
function build_head() {
?>
<!DOCTYPE html PUBLIC>
<html>
<head>
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <title>Culture</title>
    <link rel="stylesheet" type="text/css" media="all" href="styles/jsDatePick_ltr.min.css" />
    <link rel="stylesheet" type="text/css" href="styles/style.css" />
    <script type="text/javascript" src="js/main_functions.js">
    </script>
    <script type="text/javascript" src="js/jsDatePick.min.1.3.js">
    </script>
    <script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?sensor=false">
    </script>
</head>
<?php
}
```

- `build_body_onload()` – Velice jednoduchá metoda, která pouze otevře část <`body`> a nastaví, které metody se mají spustit při události `onLoad`.

```
function build_body_onload() {
    ?>
        <body onLoad="initialize();">
    <?php
    }
```

  V případě, že by se uživatelům s určitým oprávněním měla při načtení těla `HTML` dokumentu spustit jiná funkce, stačilo by nahradit metodu `build_body_onload()` novou metodou, která by při načtení těla provedla jinou funkci.

Členské metody třídy `Builder` jsou volány z instance této třídy, která je vytvořena v souboru `index.php`, tedy souboru, který je zkompilován a spuštěn při otevření aplikace. Zde je zobrazen okomentovaný zdrojový kód tohoto souboru:

```
<?php

// spustí novou session, nebo zavolá stávající
session_start();

// vytvoří novou instanci třídy builder
require("src/class.builder.php");
$builder = new builder;

// je sestavena hlavička dokumentu, otevřeno tělo dokumentu a záhlaví stránky
$builder->build_head();
$builder->build_body_onload();
$builder->build_body_header();

// pokud je uživatel přihlášen, zobrazí jeho přihlašovací údaje
if (isset($_SESSION['username'])) {
    $builder->build_logout_form();
}
// pokud není přihlášen, je sestaven přihlašovací formulář
else {
    $builder->build_login_form();
}

// je sestaven seznam výsledků
$builder->build_result_box();

// pokud je uživatel přihlášen, zkontroluje oprávnění, a podle něj sestaví menu
if (isset($_SESSION['username'])) {
    if ($_SESSION['permissions'] == 'guest')
    {
        $builder->build_guest_menu();
    }
    elseif ($_SESSION['permissions'] == 'organiser')
    {
        $builder->build_organiser_menu();
    }
    elseif ($_SESSION['permissions'] == 'admin')
    {
        $builder->build_admin_menu();
    }
}
else {
    $builder->build_guest_menu();
```

```
}

// je sestaven prostor pro formuláře a pro výsledky a zápatí dokumentu
$builder->build_permanent_elements();

// uzavírá tělo dokumentu a celý dokument
$builder->build_tail();
?>
```

Tento krátký soubor otevře asociativní pole `$_SESSION` a podle uživatelského jména a oprávnění volá ty členské metody třídy `Builder`, které jsou přístupné uživateli s odpovídajícím oprávněním. Tím je sestaven `HTML` kód celé stránky.

Ke správnému sestavení stránky je zde využito asociativní pole `$_SESSION`. Mezi asociativní pole v `PHP` patří třeba i dříve zmiňované `$_GET` a `$_POST`, nebo `$_FILES`, které je v aplikaci Culture použito pro nahrávání obrázků na server. Konkrétně `$_SESSION` je specifické tím, že obsahuje proměnné relace, které jsou přístupné v celém skriptu do té doby, než vyprší platnost relace (`timeout`). To je výborná věc pro uchovávání dat, které je třeba uchovat po celou dobu běhu skriptu, jako je například oprávnění uživatele. Před zapisováním do nebo čtením z asociativního pole musím pole vytvořit nebo zavolat již vytvořené. K obojímu slouží metoda `session_start()`. Mezi další metody asociativního pole `$_SESSION` použité v aplikaci Culture patří metody `isset($_SESSION[$var])` a `session_unset()`. Metoda `isset($_SESSION[$var])` vrací hodnotu `true` v případě, že je do proměnné `$var` dosazena nějaká jiná hodnota než `null`. V případě, že není, vrací hodnotu `false`. Tato metoda byla použita při stavění stránky v kódu výše. Metoda `session_unset()` vyprázdní všechny proměnné asociativního pole. V aplikaci Culture je použita v souboru `logout.php`, který slouží k odhlášení uživatele. Vypadá takto:

```
<?php
session_start();
session_unset();
?>
```

V aplikaci Culture jsou v asociativním poli `$_SESSIONS` uchovávány informace o přihlášení, konkrétně uživatelské jméno a oprávnění. Uživatelské jméno i oprávnění jsou po přihlášení zobrazeny na hlavní stránce spolu s tlačítkem `Log Out`.

4.2.2 Třída `Pgsql` a připojení k databázi

Třídu `Pgsql` jsem nenapsal sám, poskytl mi ji vedoucí této bakalářské práce Ing. Jiří Cajthaml, Ph.D. a já jsem ji poupravil pro potřebu této bakalářské práce. Třída `Pgsql` obsahuje řadu členských proměnných, které definují připojení, kromě nich obsahuje i členské metody, které zprostředkovávají komunikaci mezi skriptem `PHP` a samotnou databází. V třídě jsou využity `PHP` metody předdefinované pro práci s databází `PostgreSQL`. Členskými proměnnými třídy `Pgsql` jsou:

- `$host`, `$port`, `$db`, `$user`, `$password` – obsahují informace o připojení
- `$link` – do této proměnné je uložena návratová hodnota metody `pg_connect`, do které vstupují jako parametry informace o připojení
- `$resource` – do této proměnné je uložen výsledek metody `pg_query($query)`
- `$row` – do této proměnné je uložen výsledek metody `pg_fetch_assoc($resource)`
- `$rowno` – označuje číslo řádky při prohledávání výsledků `SQL` dotazu

Členské metody třídy `Pgsql` slouží k manipulaci s členskými proměnnými a k členění výsledků. Konstruktor třídy je napsán tak, aby při vytvoření instance bylo rovnou vytvořeno připojení k databázi.

```
function pgsql() {
    $this->link = pg_connect("host=".$this->host." port=".$this->port." dbname=".$this->db."
    user=".$this->user." password=".$this->password);
    if($this->link === FALSE) {
    return;
    }
    register_shutdown_function( array( &$this, "end" ) );
  }
```

Dalšími metodami třídy `Pgsql` jsou:

- `query($query)` – Tato metoda provede `SQL` dotaz a výsledek uloží do proměnné `$resource`. Zároveň nastaví číslo řádku na 0 a proměnnou `$resource` vrátí.
- `nextrow()` – Tato metoda převezme aktuální řádek výsledků dotazu. Potom zkontroluje, zda ve výsledcích existuje další řádek. Pokud ano, číslo řádku inkrementuje, pokud ne, je metoda ukončena bez inkrementace.

- `getfield($col)` – Tato metoda vrací hodnotu atributu z aktuálního řádku.
- `end()` – Tato metoda uzavírá spojení s databází.

Instance třídy `Pgsql` jsou v aplikaci Culture používány ve všech ostatních `.php` souborech. Slouží jak k přidávaní, tak k vyhledávání záznamů v databázi. Těchto souborů je poměrně dost a některé jsou kvůli velkému množství podmínek a vedlejším efektům, které se provedou při určitých podmínkách, dost složité, ale princip využití třídy `Pgsql` je ve všech případech stejný. Uvedu zde proto jako příklad jen jeden jednoduchý skript. Jedná se o skript, který vrátí všechny položky z tabulky `types`.

```
<?php
require("src/class.pgsql.php");
$p = new pgsql;

$p->query("SELECT * FROM types");
while($p->nextrow()) {
  $type_name =  $p->getfield("type_name");
  $type_id = $p->getfield("type_id");
  echo "$type_id|$type_name|";
}
?>
```

Nejprve je vytvořena instance třídy `Pgsql`, poté je proveden `SQL` dotaz a nakonec jsou ze seznamu výsledků po řádcích vypisovány nalezené hodnoty. Co se s nimi bude dít dále, je věcí `JavaScript`u.

4.3 Použití `JavaScript`u v aplikaci Culture

`JavaScript` je objektově orientovaný skriptovací jazyk používaný jako interpretovaný programovací jazyk pro tvorbu WWW stránek. Na rozdíl od `PHP` je `JavaScript` interpretován na straně klienta, takže všechny skripty v tomto jazyce jsou při spuštění stránky dočasně staženy do prohlížeče. To způsobuje určitá bezpečnostní rizika, takže `JavaScript` například nemůže pracovat se soubory. Obecně platí, že `HTML` zajišťuje obsah dokumentu, `CSS` jeho vzhled a `JavaScript` funkčnost. `JavaScript` byl původně vyvíjen ve společnosti Netscape vývojářem Brendanem Eichem pod názvem `Mocha`. Poprvé byl podporován v beta verzi prohlížeče `Netscape Navigator 2.0` (vydaným v září 1995) pod jménem

LiveScript. Jeho současný název JavaScript vyvolává dojem, že má něco společného s programovacím jazykem Java. Společnou mají pouze podobnou syntaxi, což je důsledkem toho, že oba jazyky vycházejí z jazyka C++. Jednalo se tedy spíše o marketingový tah. V současnosti je JavaScript obchodní známkou společnosti Oracle Corporation.

Obr. 4.4: Neoficiální logo JavaScriptu použité na konferenci JSConf EU 2011

V aplikaci Culture je JavaScript používán především k zobrazování mapy a práci s ní, k práci s výsledky PHP skriptů vracejících výsledky z databáze pomocí AJAXu, k práci s kalendářem a dále k práci s uživatelským prostředím.

4.3.1 Zobrazení mapy a práce s ní

K práci s mapou používám **Google Maps JavaScript API V3**, které bylo podrobně popsáno v samostatné kapitole. Zde se tedy zaměřím hlavně na konkrétní příklady použití v aplikaci Culture. V aplikaci jsou dvě instance třídy Map. První z nich je mainMap a je zobrazena v hlavním okně aplikace při načtení stránky, po každém vyhledávání a při volbě „Mapa“ v hlavním menu. Slouží k zobrazování míst nalezených SQL dotazem. Druhá mapa, addPlaceMap, je součástí formuláře pro přidávání míst. Slouží k výběru polohy nového místa. Obě dvě mapy jsou zobrazeny jako ROADMAP a mají nastaven střed na stejné souřadnice.

Obr. 4.5: Ukázka mapy s markerem a otevřeným informačním oknem

Hlavní mapa se inicializuje funkcí `initializeMap()`, která obsahuje proměnnou `mainMap Options` s nastavením mapy. Jediná věc, kterou funkce nakonec vykoná, je zobrazení mapy se zvoleným nastavením v prvku <`div`> definovaným id `mapCanvas`.

```
function initializeMap() {
  var mainMapOptions = {
    center: new google.maps.LatLng(48.975, 14.474),
    zoom: 14,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
mainMap = new google.maps.Map(document.getElementById("mapCanvas"), mainMapOptions);
}
```

K inicializaci mapy `addPlaceMap` je použita funkce `initializePlaceMap`, která je v základu totožná (v nastavení je hodnota `zoom` nastavena na 15 a mapa je zobrazena v jiném prvku <`div`>), ale navíc mapě přidává `Listener`, který při kliknutí do mapy vytvoří marker. V kódu vypadá přidání `Listeneru` následovně:

```
google.maps.event.addListener(addPlaceMap, 'click', function(event) {
addPlaceMarker(event.latLng);
});
```

Při kliknutí do mapy `addPlaceMap` je tedy spuštěna funkce `addPlaceMarker(location)` s hodnotou parametru `event.latLng`, která obsahuje souřadnice místa v mapě, kam uživatel kliknul. Podobnou funkci jako je `addPlaceMarker(event.latLng)` využívá i mapa `mainMap`. Jedná se o funkci `showMarker(location, p_name, content, icon)`. Tato metoda má kromě parametru `location`, který definuje polohu markeru, ještě parametry `p_name` a `icon`, které obsahují název markeru a cestu k ikoně, a parametr `content`, který je uvnitř funkce předáván dál funkci `createInfoWindow(map, marker, content)`. Oproti tomu funkce `addPlaceMarker(location)` zobrazí marker bez názvu, bez informačního okna a s implicitní ikonou, ale souřadnice markeru uloží do `HTML` tagů typu `<input type='hidden' />`. Při potvrzení formuláře pro přidání místa jsou tyto souřadnice uloženy do tabulky `places` jako poloha nového místa. Všechny tyto funkce jsou rozepsány zde:

```
function showMarker(location, p_name, content, icon){
    var marker = new google.maps.Marker({
        position: location,
        map: mainMap,
        title: p_name,
    });
    if (icon)
    {
    marker.setIcon(icon);
    }
    markersArray.push(marker);
    createInfoWindow(mainMap, markersArray[markersArray.length-1], content);
}

function addPlaceMarker(location) {
    clearMarkers(addMarkersArray);
    var marker = new google.maps.Marker({
        position: location,
        map: addPlaceMap
    });
    addMarkersArray.push(marker);
    var coordsLatLng = addMarkersArray[0].getPosition();
    document.getElementById("addNCoord").value = coordsLatLng.lat();
    document.getElementById("addECoord").value = coordsLatLng.lng();
}

function createInfoWindow(map, marker, content){
var infowindow = new google.maps.InfoWindow({
    content: content
    });
    infoWindowsArray.push(infowindow);
```

```
    google.maps.event.addListener(marker, 'click', function() {
        for (var i = 0 ; i < infoWindowsArray.length; i++)
        {
            infoWindowsArray[i].close();
        }
        infowindow.open(map, marker);
    });
}
```

Funkce `createInfoWindow(map,marker, content)` vytvoří nové informační okno a přidá mu `Listener`, který při kliknutí na marker zavře všechna otevřená informační okna a otevře jen to, které je přiřazeno tomuto markeru.

4.3.2 Práce s AJAXem

Pod zkratkou `AJAX` se skrývá „*Asynchronous JavaScript and XML*“ a je to označení pro technologie, které provádí změny ve stránce dynamicky, tedy bez nutnosti znovunačítání. V aplikaci Culture je `AJAX` používán k odeslání `PHP` skriptu a k dalšímu zpracování textu, který `PHP` vypíše. Jinými slovy – `JavaScript`ová funkce převezme hodnoty z `HTML` formuláře, metodou `GET` je předá `PHP` skriptu. Výsledky, které `PHP` skript vypíše konstruktem `echo`, uloží do instance třídy `XMLHttpRequest`, se kterou potom může `JavaScript` dále pracovat. Použití `XMLHttpRequest`u v aplikaci Culture je vždy reprezentováno tímto obecným kódem:

```
var xmlhttp;
xmlhttp.open("GET","skript.php?promenna1="+promenna1+"&promenna2="+promenna2,true);
xmlhttp.send();
  if (window.XMLHttpRequest)
  {// pro IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
  }
    else
  {// pro IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
    xmlhttp.onreadystatechange=function()
  {
    if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
      var info = xmlhttp.responseText; // výpis z^HTML je uložen do proměnné info
      // další zpracování výsledků JavaScriptem
```

```
    }
  }
```

V aplikaci Culture je PHP skriptem zpracován SQL dotaz a výsledky jsou odeslány jako text konstruktem echo. Členská metoda třídy XMLHttpRequest responseText tento text přijme a dále zpracuje.

4.3.3 Práce s kalendářem JsDatePick

Kalendář, který se zobrazuje ve formuláři pro vyhledávání a ve formuláři pro přidávání nových kulturních událostí, napsal v roce 2009 izraelský vývojář Itamar Arjuan a zpřístupnil ho pod licencí GNU General Public License. Použití kalendáře obnáší přidání těchto dvou řádek do hlavičky kódu:

```
<link rel="stylesheet" type="text/css" media="all" href="styles/jsDatePick_ltr.min.css"/>
<script type="text/javascript" src="js/jsDatePick.min.1.3.js"></script>
```

Tím je do dokumentu přidán zdrojový kód kalendáře. Kalendář je do dokumentu přidán vytvořením instance třídy JsDatePick s nastavenými vlastnostmi. V aplikaci Culture jsou následujícím vlastnostem přidány tyto hodnoty:

- useMode:1 – nastavení zobrazení celého kalendáře rovnou (další možnost je zobrazení kalendáře až po kliknutí do prvku cíle)
- isStripped:true – nastavení zobrazení kalendáře bez nadbytečných ovládacích prvků, jako je například pro tento případ nežádoucí křížek, který kalendář zavře
- target:DOM element – nastavení HTML prvku, ve kterém bude kalendář zobrazen

Tím je kalendář zobrazen na stránce. Další věcí, kterou je důležité ošetřit, je získání vybraného data. K tomu slouží členská metoda getSelectedDate(). Všechny funkce, které jsou potřeba k zobrazení kalendáře a získání vybraného data, jsou obsaženy ve funkci initializeSearchCalendar() pro zobrazení kalendáře ve formuláři pro vyhledávání a ve funkci initializeAddCalendar() pro zobrazení kalendáře ve formuláři pro přidávání kulturních akcí. Tyto funkce jsou spouštěny po vygenerování odpovídajícího formuláře.

4.3.4 Práce s uživatelským prostředím

Prací s uživatelským prostředí mám na mysli běžnou funkčnost stránky, jako to, že při potvrzení formuláře pro vkládání nových záznamů je zobrazen kontrolní formulář, nebo že při výběru položky ze seznamu nalezených kulturních akcí jsou v hlavním bloku zobrazeny podrobnosti o události a místě. To je řešeno vložením události `onClick` k prvku dokumentu.

```
<input type="button" onClick="javascript:login(document.getElementById('username').value,
document.getElementById('pwd').value)" class="myButton" value="Log In" />
```

Tento tag vytvoří tlačítko s popisem „Log In“. Při stisknutí tlačítka je spuštěna funkce `login(username, passwd)` s hodnotami parametrů převzatými z prvků definovaných id `username` a `pwd`. Vzhled prvků dokumentu má v režii `CSS`.

4.4 Vzhled a styly

Vzhled dokumentu zajišťuje jazyk `CSS`, tedy „Cascade Style Sheets“. Jazyk navrhl norský vývojář Håkon Wium Lie (v současnosti CTO ve společnosti Opera Software) z podnětu standardizační organizace W3C („World Wide Web Consortium“), která vyvíjí standardy pro World Wide Web. Smyslem `CSS` je oddělení vzhledu dokumentu od jeho obsahu a funkčnosti. Původně měl vzhled zajišťovat `HTML` kód a v některých starších verzích `HTML` opravdu obsahuje tagy, které se starají i o vzhled stránky. V dnešní době je však standardem k zajišťování vzhledu stránky jazyk `CSS`.

K řešení vzhledu aplikace Culture byla použita šablona `Consistent`, kterou jsem stáhnul z webu http://www.freecsstemplates.org/ a upravil pro potřeby aplikace. Jedná se o dvousloupcový vzhled s pevnou šířkou a tmavým barevným schématem. Šablona je distribuována zdarma pod licencí Creative Commons Attribution 3.0.

5 Závěr

V rámci této bakalářské práce byla vytvořena aplikace, která umožňuje vyhledávání v databázi, zobrazování výsledků na mapě, přidávání nových záznamů do databáze a obsahuje správu uživatelů. Prvky v aplikaci jsou uspořádány tak, aby aplikace byla přehledná a ovládání jednoduché.

Při zpracovávání této práce jsem se naučil přemýšlet nad problémy jiným způsobem. Klíčovou částí řešení problému je nalezení nejlepšího řešení. Jeho implementace do kódu je potom už spíše jen manuální prací. Na problém je potřeba dívat se komplexně z pohledu celé aplikace a ne jenom z pohledu jednotlivých částí. Když jsem začal dodržovat tato pravidla, práce mi šla mnohem rychleji a problémy jsem řešil efektivněji. Kromě toho jsem si potvrdil svoji teorii, že při znalosti několika programovacích jazyků je mnohem jednodušší naučit se další.

Aplikace Culture je v současném stavu provozuschopná, ale pořád je možné ji vylepšit. Během práce jsem neustále vymýšlel novinky a vylepšení a některé z nich se mi do aplikace ve vymezeném čase podařilo přidat. Mezi ty ostatní patří například:

- Přidání možnosti editovat informace o kulturních událostech a míst těm uživatelům, kteří je vytvořili.
- V podrobnostech o kulturních událostech evidovat počet uživatelů, kteří se události zůčastní.
- Pro rychlejší vkládání kulturních akcí přidat možnost exportu kulturních akcí z aplikace **Google Calendar**.
- Vytvoření diskuze u kulturních událostí a míst pro zaregistrované uživatele.
- Vytvoření dalšího oprávnění, jehož držitelé by měli možnost psaní reportáží a recenzí na kulturní události. Ty by se zobrazovaly při načtení stránky v hlavním panelu aplikace.
- Vytvoření nové tabulky pro archivaci akcí, které už nejsou aktuální.
- Vytvoření aplikace pro mobilní zařízení.

6 Seznam citací a zdrojů

Citace

1. Google Places API References. In: *Google Developers* [online]. 2011 [cit. 2012-05-15]. Dostupné z: https://developers.google.com/maps/documentation/places/

Publikace

- ZAKAS, Nicholas C., Jeremy MCPEAK a Joe FAWCETT. *AJAX Profesionálně*. Brno: Zoner Press, 2007. ISBN 978-80-86815-77-0.
- MOMJIAN, Bruce. *PostgreSQL*: *praktický průvodce*. 1. vyd. Brno: Computer Press, 2003, 402 s. ISBN 80-722-6954-2.

Web

- PostgreSQL 9.1.3 Documentation. POSTGRESQL GLOBAL DEVELOPMENT GROUP. *Dokumentace PostgreSQL* [online]. 2011 [cit. 2012-05-13]. Dostupné z: http://www.postgresql.org/files/documentation/pdf/9.1/postgresql-9.1-A4.pdf
- STACK EXCHANGE INC. *Stack Overflow* [online]. [cit. 2012-05-15]. Dostupné z: http://stackoverflow.com/
- REFSNES DATA. *W3Schools* [online]. 1999 [cit. 2012-05-15]. Dostupné z: http://www.w3schools.com/
- GOOGLE INC. *Google Developers* [online]. 2011 [cit. 2012-05-15]. Dostupné z: https://developers.google.com/
- PHP GROUP. *PHP* [online]. 2001 [cit. 2012-05-15]. Dostupné z: http://php.net/
- POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL* [online]. 1996 [cit. 2012-05-15]. Dostupné z: http://www.postgresql.org/

- WIKIMEDIA. *Wikipedia* [online]. 2001 [cit. 2012-05-15].
  Dostupné z: http://www.wikipedia.org/