VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

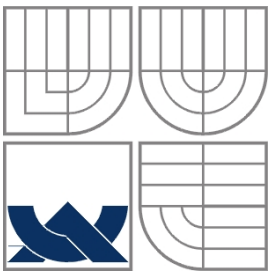
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DATABÁZOVÝ SYSTÉM POSTGRESQL V PROSTŘEDÍ WEBU

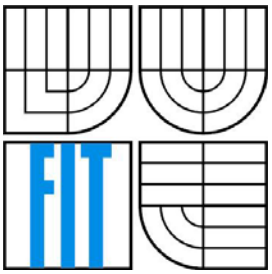
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE JAN PREUSS
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DATABÁZOVÝ SYSTÉM POSTGRESQL V PROSTŘEDÍ WEBU

DATABASE SYSTEM POSTGRESQL
IN WEB ENVIRONMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — Jan Preuss
AUTHOR

VEDOUCÍ PRÁCE — Ing. Dušan Vrážel
SUPERVISOR

BRNO 2008

Abstrakt

Práce pojednává o principech ukládání dat z pohledu tvůrce webových aplikací. Porovnává tři databázové systémy a jejich možnost využití v rámci webových projektů. V dalších částech popisuje návrh aplikace na správu a editaci katastrální mapy malé obce. Její implementaci, strukturu systému a postup řešení. Vysvětluje požívání jazyka PL/pgSQL v databázovém systému PostgreSQL a jeho komunikaci s uživatelským rozhraním napsaném v JavaScriptu. Posední část je zaměřená na popis systému a jeho obsluhu z pohledu uživatele.

Klíčová slova

Databázové systémy, Oracle, MySQL, PostgreSQL, PostGIS, PL/pgSQL, AJAX, JSON, jQuery, MapScript.

Abstract

This work deals with principles of data storage from the web creator's point of view. It compares three major database systems in possibilities of their usage in the broader range of web projects. In the other parts it describes concept of administration and adaptation of cadastral map for small village. It describes implementation, system structure and decision procedure, explains usage of PL/pgSQL language in PostgreSQL database system as well as communication among system and JavaScript user interface. The final part focus on system description and user control ability.

Keywords

Database systems, Oracle, MySQL, PostgreSQL, PostGIS, PL/pgSQL, AJAX, JSON, jQuery, MapScript.

Citace

Preuss Jan: Databázový systém PostgreSQL v prostředí webu, bakalářská práce, Brno, FIT VUT v Brně, 2008

Databázový systém PostgreSQL v prostředí webu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vrážela. Vzorová data jsou převzata z volně dostupných informací Českého úřadu zeměměřičského a katastrálního.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Preuss
13. 05. 2008

Poděkování

Rád bych poděkoval především vedoucímu své bakalářské práce Ing. Vráželovi za jeho odborný dohled a směřování k dokončení celé práce. Dále pak všem, kteří se aktivně podílí na psaní dokumentací k dostupným projektům, jako je PostgreSQL nebo jQuery [5], [8].

Obsah

1 Úvod

Cílem této práce je seznámení se s možnostmi ukládání dat v prostředí webu, podrobné seznámení s databázovým systémem PostgreSQL a jeho rozšířením PostGIS. A porovnání tohoto systému s dalšími databázovými systémy, jmenovitě MySQL a Oracle.

Ukládání prostorových dat v databázových systémech je v dnešní době na vzestupu, především díky veřejné správě, která tato data vždy shromažďovala a nyní je může pohodlně ukládat a lépe se v nich orientovat.

To je také jeden z důvodů, proč po konzultaci s vedoucím práce byla vybrána aplikace pro práci s katastrální mapou malé obce, která má umožnit snadnou práci s pozemky v jejím nejbližším okolí. A tím usnadnit přehled o jednotlivých parcelách v obci.

První část této práce se věnuje teoretické přípravě a rozebírá ukládání dat z pohledu webového prostředí. Podrobněji popisuje a porovnává jednotlivé databázové systémy a všechny technologie zmiňované dále v textu.

Další dvě části se zabývají analýzou a návrhem řešené aplikace a popisují postup při tvorbě představy o celkovém systému. A navrhují řešení dané problematiky s využitím modelovacích technik jazyka UML.

Následující část práce popisuje způsob implementace tohoto systému a nástin jak byla aplikace řešena. Popisuje konkrétní techniky použité při vytváření aplikace a ukázky kódů, na kterých vysvětluje postup řešení.

Předposlední část obsahuje podobný popis z pohledu uživatele přistupujícího k systému a může sloužit jako uživatelská příručka pro práci s aplikací.

V závěru je zhodnocen vytvořený systém a popsán nástin možností dalšího vývoje systému.

2 Teoretická část

2.1 Ukládání dat

V době, kdy vznikal Internet a Tim Berners-Lee přišel se svou vizí hypertextového jazyka a představou provázání dosud separovaných dokumentů, stačilo data ukládat staticky do souborů s příponou *html*. Převratné tehdy bylo především právě to provázání souborů mezi sebou, aby se čtenář mohl snadno dostat k dalším podobným informacím.

V dnešní době, kdy se Internet stal nedílnou součástí mnoha firem a domácností, v době kdy se bez něj neobejde žádná vysoká škola a kdy se Internet stává nejbohatším zdrojem všech informací, už nestačí ukládat data do statických souborů, ale je potřeba je zpracovávat dynamicky.

V počítačových systémech se data fyzicky ukládají různými způsoby, ale zjednodušeně lze říci, že se jedná stále o jeden a tentýž princip ukládání dat. A to binárně na nějaké záznamové médium. Rozdílný je ovšem přístup k těmto datům. Budeme-li se bavit o Word Wide Webu (dále jen WWW), mnoho možností jak data ukládat není. V podstatě se nabízí jen dvě základní možnosti jak data ukládat. Buď formou souborů, ať již textových nebo binárních nebo, což je dnes asi o něco častější, ukládání dat do databázových systémů.

2.1.1 Ukládání do souborů

Do souborů se dají ukládat data, jak již bylo zmíněno, buď binárně, nebo jen textově. Za obecné ukládání binárních informací do souborů můžeme považovat například obyčejné obrázky. S těmito daty se ovšem velmi těžko pracuje, pokud je potřeba je dále členit. O něco univerzálnější je ukládání dat do textových souborů.

Textové soubory

Text lze do souborů ukládat libovolným způsobem. Nejjednodušší je ukládání prostého textu a následné použití celého obsahu souboru. To je ovšem ve většině případů, kdy potřebujeme kromě holých dat i nějaké jejich další atributy, dost nevhodné. Proto, když už někdo přistoupí k tomu ukládat data do textových souborů, většinou si navrhne jednoduchou strukturu, do které informace ukládá. Tu potom prostřednictvím nějakého jazyka generujícího HTML výstup, rozdělí na požadované informace. Tento způsob je ale poměrně nevhodný z pohledu přenositelnosti a kompatibility například s jinými systémy. Právě proto je dnes snaha přecházet na stejný systém ukládání dat do souborů.

XML

Jedním z převažujících způsobů jak uložit textová data do souborů je značkovací jazyk XML (eXtensible Markup Language). Svým způsobem opět dost univerzální a variabilní. Ale jeho výhoda spočívá v systému zapisování informací do jednotlivých elementů, které mohou mít různé atributy, jak je vidět a kódu 2.1. Názvy značek nejsou dopředu nějak dané, jako je tomu třeba u HTML. Proto je možné použít XML na široké spektrum typů dat a nasadit ho v různých situacích. V dnešní době je tento způsob ukládání dat velmi používaný a existují například i obrázky ukládané ve formě XML. Příkladem může být vektorový formát SVG založený právě na jazyku XML.

```
<dokument typ=”textovy” datum=”08.05.2008”>
   <titulek>Titulek dokumentu</titulek>
   <autor>
      <jmeno>Jan Preuss</jmeno>
      <email>xpreus01@stud.fit.vutbr.cz</email>
   </autor>
   <obsah>
      <clanek cislo=“1“>...</clanek>
      ...
   </obsah>
</dokument>
```

Kód 2.1

Bohužel jeho univerzálnost se musí projevit na velikosti souborů. Ta díky režii, způsobené formou zápisu, nabývá u většího množství dat až příliš vysokých hodnot. To může vést například ke zpomalení práce s těmito daty. Také díky možnosti nekonečně mnohokrát zanořovat značky do sebe, může dojít u počítačů s menší operační pamětí k jejímu přetečení a nemožnosti tak dokument rozložit. Největší nevýhodou ovšem je nemožnost uložit do XML jiná než textová data.

Podpora XML je dnes již zabudována ve většině používaných programovacích jazycích. Buď formou přídavných knihoven nebo vestavěných parserů, které převádí data do nejčastěji objektové struktury. Příkladem může být vestavěná podpora XML ve skriptovacím jazyce PHP nebo moduly pro jazyk Perl.

2.1.2 Databázové systémy

Daleko častějším typem ukládání většího množství dat je využití různých databázových systémů. Existuje velké množství těchto systémů, ale následující text se věnuje těmto třem konkrétním databázovým systémům: Oracle, MySQL a PostgreSQL.

Oracle

Databázový systém společnosti Oracle Corporation, vydávaný pod názvem Oracle Database, patří ke špičce mezi databázovými systémy. Jako drtivá většina dnešních systémů dle [1] podporuje relační dotazovací jazyk dle norem ISO/ANSI SQL92 a pozdější SQL 99. Navíc však i vlastní rozšíření jako jsou například hierarchické dotazy. Pro lepší využití možností databázového systému Oracle je možné psát uložené procedury, uživatelské funkce nebo triggery a to ve speciálním jazyce PL/SQL, navrženém právě společností Oracle. Velkou výhodou je podpora prostorových databází a v posledních verzích i objektových databází nebo databází založených na hierarchickém modelu dat. V současné době je aktuální verze Oracle Database 11g, jak je uvedeno na oficiálních stránkách společnosti [2].

Od verze systému Oracle 4 začala společnost s prvními pokusy implementace prostorových dat ve svých databázích. Velký zlom však přinesla až verze Oracle 7, která zavádí Spatial data option, zkráceně SDO. Jedná se o princip indexování a ukládání prostorových dat v databázích Oracle. Následující vývojová verze, tedy Oracle 8, přichází s vlastním odděleným rozšíření nazvaným Oracle Spatial. Kromě jiných výhod přinesl nový způsob indexování prostorových dat a to pomocí R-tree indexů, jejichž princip je založen na stromové struktuře ukládající objekt, rozložen na pravidelné n-rozměrné objekty. V dnešní době je toto rozšíření nabízeno pouze v programovém balíku Oracle Enterprise Edition.

V rámci celého spektra dnešního webu je databázový systém Oracle v menšině. Může za to převážně jeden fakt a to ten, že je tento systém komerční a náklady na jeho pořízení jsou pro většinu běžných vývojářů příliš drahé. Pro větší společnosti, které mají dostatek finančních prostředků a žádají si stabilní databázový systém, je Oracle vhodný a to i s možností napojení na webové služby. Příkladem mohou být informační systémy dvou brněnských vysokých škol.

MySQL

MySQL je v dnešní době jedním z nejrozšířenější databázový systém používaný ve webových službách. Především díky jeho propojením se skriptovacím jazykem PHP. Většina začínajících vývojářů sáhne právě po MySQL především díky větší propagaci tohoto systému společně s PHP a jeho rychlosti na úkor menší funkcionality.

Jedná se o systém vytvořený Švédskou společností MySQL AB, která ho šíří nejen pod komerční licencí, ale především pod bezplatnou licencí GPL, velmi oblíbenou mezi nezávislými vývojáři. V současné době se firma MySQL AB spojila s gigantem na poli informačních technologií, společností SUN Microsystems Inc.

Systém podporuje standardní jazyk SQL92 s vlastními úpravami. Nabízí možnost ukládat data v různých typech databázových tabulek. Mezi ty nejznámější patří MyISAM a InnoDB. V připravovaných verzích se počítá i s novým systémem Maria, který se právě vyvíjí. Nejčastěji se

používá typ MyISAM, protože je rychlejší a primárně nastavený. Tento typ tabulek ovšem nepodporuje cizí klíče či transakce. Proto je pro náročnější uživatele vhodnější InnoDB, které tyto vlastnosti podporuje. Bohužel však na úkor rychlosti. Od verze 4.1 s využitím rozšíření podporuje i prostorové databáze, zatím ovšem ne příliš použitelně.

Současná poslední stabilní verze MySQL 5.0 dle [3] přináší práci s pohledy a kurzory. Ale především zavádí možnost psaní vlastních uložených procedur, jako je tomu u ostatních databázových systémů, v jazyku vycházejícím z jazyka SQL/PSM. Od této verze je možné psát i triggery, spouštěné při provedení příkazů INSERT, UPDATE nebo DELETE.

Připravované nové verze 5.1 a 6.0 obsahují další nová zlepšení jako například dělení tabulek a záznamů, plánovač událostí a mnoho dalších.

PostgreSQL

Databázový systém PostgreSQL dle [4] vznikl z původně vyvíjeného systému na kalifornské univerzitě v Berkeley pod názvem Ingres v letech 1977 – 1985. Od té doby prošel dlouhým vývojem až k podobě, kterou známe dnes. V současnosti je projekt PostgreSQL vyvíjen jako open source, velkou skupinou nezávislých vývojářů po celém světě. Díky tomu je šířen pod nejvolnější licencí BSD, která dovoluje využít systém nejen v nekomerční sféře, ale dokonce ho nabízet i jako součást komerčních produktů.

Jako většina systémů založených na jazyce SQL podporuje i PostgreSQL všechny funkce SQL92 a rozšiřuje ho o vlastní modifikace. Krom toho plně podporuje cizí klíče, uložené procedury, pohledy atd. Výhodou je psaní uložených procedur nejen v jednom jazyce, ale v libovolném jazyce, jako například Perl, Python, C nebo třeba i speciálním jazyku PL/pgSQL, vycházejícím z jazyka navrženého firmou Oracle.

Současné verze PostgreSQL 8 podle [5] obsahují vlastnosti, které ho díky své stabilitě a bezpečnosti umožňují nasadit v podnikových aplikacích. Jako například dělení tabulek a záznamů, informační schémata, částečné a funkcionální indexy anebo integrované fulltextové vyhledávání.

Díky možnosti psát funkce v různých jazycích, nabízí PostgreSQL možnost rozšiřitelnosti o mnoho zajímavých doplňků. Těchto rozšíření je obrovské množství a systém tak dokáže přesně to, co uživatel požaduje. Počínaje simulováním prostředí Oracle nebo MySQL, až po prostorové databáze, rozšíření PostGIS. Mezi další užitečné rozšíření patří Slony-I nebo TSearch2 pro fulltextové vyhledávání na bázi lexémů. Velké množství těchto rozšíření je na serveru www.pgfoundry.org, zabývajícího se právě touto problematikou.

Rozšíření PostGIS implementuje do databázového systému PostgreSQL možnost pracovat s prostorovými daty tak, jak to předpokládá standard OpenGIS, popisující aplikační rozhraní a datové formáty v GIS systémech. Vychází ze SQL92 rozšířeného o geometrické typy, definuje metadata pro práci s prostorovými daty a jejich datové schéma. Je tak možné silně konkurovat komerčním systémům, jako je Oracle. PostGIS přináší rozšíření standardního PostgreSQL o nové datové typy,

operátory a funkce pracující s prostorovými daty a dvě nové systémové tabulky *geometry_columns* a s*patial_ref_sys* obsahující metadata.

Aktuální verze PostgreSQL již nepracuje u prostorových dat s indexy typu R-tree, ale využívá tzv. GiST (Generalized Search Tree) indexů. Nevýhodou R-tree indexů byla jejich nutnost pracovat s minimálně dvou rozměrnými daty. GiST indexy ovšem mohou obsahovat jak R-tree indexy, tak i B-tree indexy a díky tomu je možné je použít daleko univerzálněji. Proto jsou v PostgreSQL používány pro fulltext, indexaci obsahu polí a samozřejmě pro geometrické typy.

Srovnání

Základní práce s databázovými systémy je podobná, ovšem každý nabízí jiné možnosti Jejich soupis je v tabulce 2.1. Důraz je kladen především na využití těchto systémů ve webových aplikacích a práci s prostorovými daty.

Srovnání z pohledu přístupu z různých programovacích jazyků je v dnešní době zhruba stejné. Například jazyk Perl obsahuje modul DBI (Database independent interface for Perl), který je platformě nezávislý a je možné se s ním připojit k různým databázovým systémům. O připojení k jednotlivým databázovým systémům se potom starají tzv. ovladače, které definují konkrétní komunikaci s rozhraním databáze. Oproti tomu jazyk PHP definuje pro každý databázový systém vlastní modul a stejně jako jiné systémy, využívá aplikační rozhraní těchto jednotlivých systémů.

Hlavní nevýhodou systému Oracle oproti MySQL a PostgreSQL je šíření pod komerční licencí. Což z pohledu vývojářů menších systémů je dosti zásadní. Proto většina dnešních poskytovatelů webhostingu Oracle nenabízí a preferují i kvůli nižším cenám nabízených služeb databázové systémy MySQL a PostgreSQL. Bohužel většina běžných poskytovatelů webhostingu nabízí PostgreSQL jen v jeho základní instalaci a za využití jeho rozšíření je nutné si připlatit.

	Oracle	**MySQL**	**PostgreSQL**
Licence	Komerční	Komerční a GPL	BSD
Cizí klíče	Ano	Pouze v InnoDB	Ano
Transakce	Ano	Pouze v InnoDB	Ano
Poddotazy	Ano	Ano	Ano
Hierarchické dotazy	Ano	Ne	S využitím rozšíření
Uložené procedury	Jazyk PL/SQL	Jazyk podobný SQL/PSM	Různé jazyky např. Perl, Python, PL/pgSQL
Triggery	Ano	Ano	Ano
Pohledy	Ano	Ano	Ano
Prostorové databáze	Oracle Spatial	Rozšíření Spatial Extensions	Rozšíření PostGIS

Tabulka 2.1

2.2 Model-view-controller

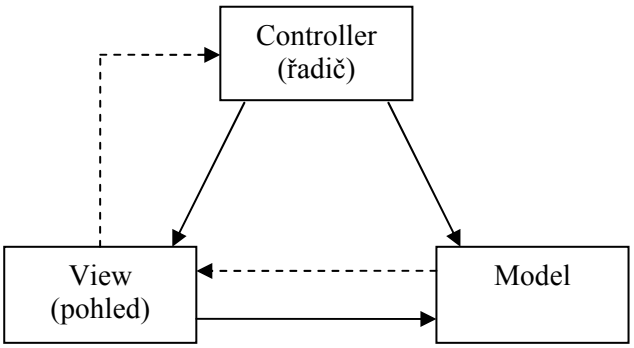
Model-view-controller (MVC) je jeden z návrhových vzorů softwarové architektury, někdy také nazývaný Model 2. Obecně řečeno rozděluje aplikaci na tři základní komponenty, které mezi sebou komunikují. Jak již z názvu vyplývá, jedná se o komponenty nazvané model, view (pohled) a controller (řadič). Tento návrhový vzor je často používaný právě ve webových aplikacích a slouží jako ucelený model, jak při realizaci postupovat.

Model je komponenta starající se o práci s daty. Jejich aktualizaci v datovém modelu. Nestará se ovšem o samotné fyzické ukládání dat. Tato činnost není návrhovým vzorem MVC pokryta. Jedná se pouze o přístup k editaci, ukládání popřípadě zobrazování dat z datové vrstvy.

View, česky nazývaný pohled, se stará o prezentaci uživateli. Neřeší vůbec žádné aplikační problémy, pouze přebírá data reprezentovaná modelem a zobrazuje je uživateli. Ve webových systémech je výsledkem obsah v HTML formátu.

Controller, v češtině řadič, je část systému starající se o odchytávání a zpracovávání událostí od uživatele. Podle situace se stará o změny v modelu nebo pohledu.

Všechny tyto tři komponenty jsou propojeny a navzájem se ovlivňují, ale oddělují jednotlivé logické části systému, jak je vidět na obrázku 2.1.



Obrázek 2.1

2.3 Použité technologie

Tato kapitola se zabývá krátkým teoretickým popisem jednotlivých technologií použitých při konečné implementaci navrženého systému.

PL/pgSQL

Jedná se o jeden z nejpoužívanějších jazyků pro psaní uložených procedur v databázovém systému PostgreSQL. Vychází z procedurálního jazyka PL/SQL navrženého firmou Oracle, postaveného na programovacím jazyku Ada. Formát psaní uložených procedur je ukázán na kódu 2.2 a srozumitelně popsaný v dokumentu [6].

```
CREATE [OR REPLACE] FUNCTION název (typ_argumentu [,...])
   RETURNS návratový_typ AS $$
      tělo_funkce
   $$ LANGUAGE 'plpgsql'
   [EXTERNAL SECURITY INVOKER | EXTERNAL SECURITY DEFINER]
```

Kód 2.2

PHP

Skriptovací jazyk určený pro programování dynamických webových stránek a rozsáhlejších webových aplikací. Původně navržený Rasmusem Lerdorfem pro jeho vlastní potřebu. Dnes jeden z nejpoužívanějších jazyků pro tvorbu dynamického webu. Současná verze PHP 5 dle [7] je již plně objektová a skýtá obrovské množství funkcí pro práci s různými datovými zdroji. Interpretace jazyka probíhá na serveru a uživateli se posílá již vygenerovaný, nejčastěji HTML kód. Části kódu, které mají být interpretovány, se uvádí mezi speciální značky, jak je vidět na příkladu 5.3.

```
<?php
   echo "Hello World";
?>
```

Kód 5.3

AJAX

Neboli Asynchronous JavaScript and XML, je princip komunikace mezi serverem a webovým prohlížečem klienta. Výhoda spočívá v tom, že přenos probíhá asynchronně, čili na pozadí a uživatel ani nemusí vědět, že se aplikace připojuje k serveru. Této technologie se využívá při tvorbě webových stránek, kde není nutné a žádoucí obnovovat celou stránku, ale stačí dynamicky doplňovat jen její části.

Princip je založen na komunikaci se serverem pomocí XMLHttpRequest. V praxi to znamená, že uživatel pomocí JavaScriptu odešle požadavek na server a dál jen na pozadí kontroluje, jak server odpoví. A podle odchycených stavových kódů zaslaných serverem se opět nějak zařídí. Data se posílají především ve formátu XML, ale je možné posílat jak HTML, tak například prostý text nebo JSON.

Následující příklad 5.4 demonstruje vytvoření XMLHttpRequest požadavku a následné odchycení odpovědi od serveru.

```
function nový_požadavek(adresa) {
   var httpRequest;
   if (window.XMLHttpRequest) { // Mozilla, Safari, ...
      httpRequest = new XMLHttpRequest();
   } else if (window.ActiveXObject) { // IE
      httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
   }

   httpRequest.onreadystatechange = function() {
      zobraz_odpověď(httpRequest);
   };
   httpRequest.open('GET', adresa, true);
   httpRequest.send('');
}

function zobraz_odpověď(httpRequest) {
   if (httpRequest.readyState == 4) {         // požadavek dokončen
      if (httpRequest.status == 200) {        // stavový http kód 200
         alert(httpRequest.responseText);     // vypíše odpověď od serveru
      } else {
         alert('Špatná odpověď od serveru');
      }
   }
}
```

Kód 5.4

JSON

Zkratka JSON znamená JavaScript Object Notation. V češtině přesně JavaScriptová objektová notace. Jedná se o odlehčený textový formát pro výměnu dat, založený na principu zapisování dat snadno čitelných člověkem i strojem. Jak již název napovídá, vychází z jazyka JavaScript, ale je použitelný a dnes již implementovaný v různých dalších jazycích. Data se zapisují ve formátu pár název:hodnota, oddělené čárkou a uzavřené ve složených závorkách, jak je vidět na příkladu.

```
{ položka1:hodnota, položka2:hodnota, položka3:hodnota, ... }
```

Pole hodnot se zapisuje pomocí hranatých závorek následně:

```
[ {položka1:hodnota, položka2:hodnota}, {položka1:hodnota}, ...]
```

V JavaScriptu se k položkám přistupuje jako k normálním objektům tedy pomocí běžné tečkové notace.

```
var data1 = { položka1:hodnota1, položka2:hodnota2 };
var data2 = [ {položka1:hod11, položka2:hod12}, {položka1:hod21 } ];

alert(data1.položka1);            // vypíše hodnota1
alert(data2[0].položka2);         // vypíše hod12
alert(data2[1].položka1);         // vypíše hod21
```

V PHP jsou pro práci s daty v JSON formátu určeny funkce *json_decode*/*json_encode*, které slouží pro převod JSON formátu do/z pole, objektu, popřípadě jiného datového typu.

jQuery

Jedná se o silný a pohodlný JavaScriptový framework, pro usnadnění práce se strukturou HTML dokumentu, práci s událostmi, AJAXem anebo například snadnou tvorbu jednoduchých, ale efektních animací. Jeho síla spočívá ve snadné práci s DOM strukturou jak je vidět v dokumentaci [8]. K jednotlivým prvkům lze přistupovat pomocí podobných selektorů jako v případě CSS nebo i XPath. Což ho tvoří velmi snadno použitelným a poměrně přehledným. Na elementy se lze nejen lehce odkazovat, ale i s nimi v rámci DOM struktury manipulovat, vytvářet nové či je odstraňovat. Stejně snadno lze na libovolné elementy napojit různé události a pak je obsluhovat.

Jádro jQuery obsahuje plno funkcí pro základní práci a většině běžných uživatelů bude určitě stačit. Navíc s každou novou verzí se základní funkce obsažené v této knihovně rozrůstají. Pro náročnější uživatele je možnost psát si i vlastní pluginy a funkcionalitu jQuery tak rozšiřovat. Těchto pluginů v současné době existují stovky a každým dnem přibývají nové.

Vše se točí kolem funkce jQuery(), která se dá taky zapsat ve zkrácené podobě formou symbolu $(). Ale to může kolidovat s jinými frameworky a tak u systémů, kde chcete používat i jiné knihovny, je lepši, psát delší formát.

Na ukázku, jak se s jQuery pracuje, následuje pár jednoduchých příkladů.

Všem prvkům SPAN obsažených v tagu P nastavíme barvu písma na červenou:

```
jQuery("p span").css({color: "red"});
```

Na konec prvního prvku DIV vložíme nový DIV.

```
jQuery("div:first").append("<div>Nový<\div>");
```

Všem odkazům nastavíme na událost kliknutí funkci, která zobrazí, kam odkaz směřuje a zastaví přesměrování na toto místo, čímž de facto zneaktivní odkazy.

```
var callback = function () {
   var href = jQuery(this).attr("href");       // jquery(this) vrací prvek,
                                               // nad kterým byla fce volaná
   alert(href);
}
jQuery("a").click(callback);
```

Nevýhodou starších verzí frameworku jQuery bylo to, že byly docela pomalé. Tento problém se ale vývojářům podařilo poměrně dobře odstranit a v dnešní době dokáže na běžných počítačích pracovat dosti svižně.

Nejlepším zdrojem inspirace a především kompletní referenční příručka všech funkcí včetně příkladů se nachází na oficiálních stránkách projektu jQuery [8].

MapScript

PHP MapScript [9] je dynamicky načítaný modul skriptovacího jazyka PHP, který zpřístupňuje vlastnosti UNM MapServeru. Díky tomuto rozšíření je možné připojit se k vzdáleným serverům pomocí standardu WMS a využívat tak různé mapové podklady. Stejně tak MapScript podporuje připojení k databázovému systému PostgreSQL a jeho rozšíření PostGIS. Všechny tyto data pak dokáže navrstvit do jednoho výsledného obrázku různých datových typů, popřípadě i formátu *swf* společnosti Macromedia. Požadované vrstvy a další konfigurace se nastavují v souboru s příponou *map*, který se poté volá v PHP funkcích MapScriptu.

Část obsahu souboru s příponou *map* může vypadat asi tak, jak je naznačeno v kódu 5.5.

```
MAP                                          # začátek sekce MAP
   NAME mapa
   UNITS meters
   EXTENT -668300 -1080950 -667700 -1080520  # požadovaná oblast
   SIZE 650 360
   IMAGECOLOR 240 240 240
   IMAGETYPE PNG
   FONTSET 'C:/mapscript/fonts/fonts.list'   # absolutní cesta k seznamu fontů
   CONFIG "PROJ_LIB" 'C:/mapscript/nad/'     # seznam souřadných systémů
   LAYER
      NAME "kat_mapa"                        # název vrstvy
      TYPE RASTER                            # typ vrstvy – rastr, vektor
      STATUS ON                              # vrstva je aktivní
      CONNECTIONTYPE WMS
      CONNECTION "http://wms.cuzk.cz/wms.asp"
```

```
      METADATA
         "wms_title"  "Katastrální mapa"
         "wms_srs" " EPSG:102067"
         "wms_name" "KN,RST_DKM,RST_GPL"
         "wms_server_version" "1.1.0"
         "wms_format" "image/png"
      END
   END
   LAYER
      NAME "pozemky"                          # název vrstvy
      TYPE POLYGON
      STATUS ON                               # vrstva je aktivní
      CONNECTIONTYPE  POSTGIS                 # připojení k DB serveru
      CONNECTION "host=lo dbname=db user=postgres password=heslo port=5432"
      DATA "geom FROM tabulka"                # data, která se mají zobrazit
   END
END
```

Kód 5.5

3 Analýza systému

Jak ze zadání bakalářské práce jasně vyplývá, jedná se o vytvoření webové aplikaci využívající rozšíření databázového systému PostgreSQL pro práci s prostorovými daty.

Systém by měl umožňovat intuitivní a graficky přívětivé zobrazování katastrálních dat, primárně určené pro malé obce. Předpokladem využití je zobrazování katastrálních dat obyvateli obce a jejich správa vedením obce. Používání aplikace by mělo zjednodušit a sjednotit katastrální data v blízkém okolí obce.

Z důvodu snadného přístupu uživatelů k datům by měl být systém navržen jako webová aplikace, bez potřeby instalace dalších rozšíření do prohlížeče.

Následuje podrobnější přehled specifikace požadavků na daný systém.

3.1 Specifikace požadavků

Základní požadavky na webovou aplikaci pracující s katastrální mapou malé obce jsou vyjmenovány v následujícím seznamu:

- Intuitivní přístup přes webové rozhraní
- Snadná práce s daty dané katastrální oblasti
- Systém oprávnění přístupů s možnostmi editace
- Volné nahlížení do systému bez nutnosti registrace
- Jednoduchá práce s mapou a možnost volby zobrazených vrstev v mapě
- Úprava jednotlivých pozemků
- Možnost zadat jednoduše nový pozemek
- Editace vlastníků a dalších parametrů jednotlivých pozemků
- Rychlé odezva systému

4 Návrh systému

Následující podkapitoly popisují návrh systému vycházející ze základní analýzy. Tento návrh se detailněji zabývá strukturou systému a jeho funkcemi. Jsou zde rozebrány veškeré požadavky kladené na systém z pohledu návrhu aplikace. Taktéž tato kapitola obsahuje datovou strukturu a model případů užití systému dle jednotlivých rolí.

4.1 Slovník pojmů

Při návrhu požadované aplikace byla použita terminologie popsaná níže, aby se tak usnadnila orientace v následujícím textu.

Systém – kompletní aplikace jako celek.

Pozemek – katastrální území definované prostřednictvím systému a uložené jako prostorová informace s dalšími vlastnostmi v databázi.

Atribut – vlastnost pozemku jako například vlastník, parcelní číslo, druh pozemku atd.

Filtr – seznam pravidel určených k vymezení určité skupiny pozemků v rámci všech zobrazených pozemků.

Mapový náhled – část mapy zobrazující konkrétní vybranou oblast.

Vrstva – jedna z vrstev mapového podkladu, které se navzájem překrývají a tvoří tak mapový náhled. Přičemž nejvyšší vrstvou je vždy vrstva obsahující pozemky.

Návštěvník – libovolný uživatel systému, který není přihlášen.

Uživatel – uživatel systému, který je přihlášen a má možnost editovat pozemky.

Administrátor – uživatel, který může přidávat a editovat účty ostatních uživatelů

4.2 Funkční požadavky

Na základě analýzy požadavků na danou aplikaci byly stanoveny funkční požadavky sepsané níže. Tyto požadavky jsou členěny dle rolí uživatele v systému.

Návštěvník může přistoupit k systému a provádět tyto operace:

- Procházet mapovými podklady
- Přibližovat a oddalovat mapový náhled
- Vybírat pozemky
- Zobrazovat informace o pozemcích
- Možnost filtrovat zobrazení pozemků dle více kritérií
- Zobrazování více mapových podkladů

Uživatel má stejné možnosti jako návštěvník, rozšířené o následující možnosti:

- Definovat a přidávat nové pozemky
- Editovat informace o pozemcích
- Mazat pozemky
- Přidávat, upravovat a mazat všechny atributy pozemků

Administrátor je speciální typ uživatele s dalšími rozšířenými možnostmi.

- Přidávat a mazat uživatele
- Upravovat detaily uživatelů

4.3 Nefunkční požadavky

Na základě analýzy požadavků na danou aplikaci byly stanoveny i tyto nefunkční požadavky, které zahrnují předpoklady pro základní běh systému.

- Databázový systém PostgreSQL 8.2
- Rozšíření PostGIS 1.3.2
- Vytvořena databáze s rozšířením o prostorové tabulky
- Webový server (Apache, IIS…)
- Skriptovací jazyk PHP 5.0.2
- Modul MapScript 4.6
- Framework jQuery pro práci s AJAXovými požadavky pro rychlejší práci se systémem
- Webový prohlížeč podporující webové standardy (nejlépe Mozilla Firefox)
- Podpora cookies v prohlížeči
- Podpora JavaScript v prohlížeči
- Předpokládaná zátěž systému je do 100 návštěvníků za den
- Předpokládaný počet návštěvníků připojených v jeden moment je do 20
- Předpokládaný počet registrovaných uživatelů je do 30

4.4 Případy užití

Výsledkem zpracování funkčních a nefunkčních požadavků je návrh případů užití, který je naznačen v příloze A. Pro lepší pochopení následuje příklad jednoho případu užití z tohoto návrhu v tabulce 4.1.

Případ použití: Editace pozemku
ID5
Stručný popis: Editace údajů o pozemku
Primární aktéři: Uživatel
Sekundární aktéři: Žádní
Předpoklady: Přihlášení uživatele
Hlavní tok: 1. Případ použití se spustí, když je požadavek na editaci pozemku 2. Zahrnout Získej informace o pozemku 3. Dokud jsou zadané údaje neplatné 3.1. Systém požaduje zadání informací o pozemku – parcelní číslo, další atributy 3.2. Systém ověří zadané údaje o pozemku 4. Zahrnout Ulož pozemek
Následné podmínky: Žádné
Alternativní toky: 1. Nezadané povinné údaje 2. Špatný výběr pozemku

Tabulka 4.1

4.5 Databázový model

Zpracováním funkčních a nefunkčních požadavků vznikl i návrh databázového modelu zobrazeného v příloze B. Pro lepší pochopení tohoto diagramu a funkcí jednotlivých tabulek následuje jejich podrobnější popis.

Databáze obsahuje dvě systémové tabulky *geometry_columns* a s*patial_ref_sys*, kterými se tento text nezabývá, protože jsou součástí prostorového rozšíření PostGIS databázového systému PosgreSQL. Ostatní tabulky jsou podrobně popsány níže.

t_pozemky

Tabulka obsahující všechny pozemky v systému a jejich atributy. Prostorová data určující vzhled daného pozemku jsou uložena v položce *pgeom* speciálního datového typu geometry. Položky *powner* a *ptype* jsou cizí klíče určující vlastníka respektive druh pozemku.

t_owner

Tabulka vlastníků pozemků uchovávajících jméno, příjmení a adresu jednotlivých vlastníků určené unikátním identifikátorem sloužící k rozšíření tabulky *t_pozemky*.

t_poz_type

Tabulka se seznamem jednotlivých druhů pozemků, obsahují název druhu a unikátní identifikátor pro provázání s pozemky.

t_group

Jména skupin uživatelů, která jen definují, jedná-li se o návštěvníka, uživatele či administrátora systému. Tato tabulka není nutná, slouží jen pro lepší orientaci v pevně definovaných číselných označeních skupin uživatelů.

t_user

Tabulka obsahující všechny uživatele systému, jejich jména a příjmení, přihlašovací údaje a skupinu, ke které náleží. Dodatečné informace jsou poslední přihlášení a je-li uživatel právě online. Do této tabulky se ukládají i všichni návštěvníci, samozřejmě bez jména a dalších údajů pro ně nepotřebných.

t_select

Obsahuje pro každého uživatele nebo návštěvníka vždy maximálně jeden záznam s vybraným pozemkem. Daný identifikátorem vybraného pozemku, jeho geometrií a identifikátorem uživatel, který pozemek vybral. Podrobnější popis práce s touto tabulkou je v následující kapitole implementace.

t_filter

Seznam všech pozemků vyfiltrovaných jedním uživatelem. Struktura položek je stejná jako u tabulky *t_select* a také je podrobněji popsána dále.

t_filter_rule

Seznam pravidel, podle kterých se ukládají pozemky do tabulky *t_filter* a podle nichž se následně filtruje. Obsahuje cizí klíč sloužící k provázání na uživatele, kterému pravidla patří, pravidlo zapsáno ve formátu JSON pro prezentaci v systému a ve formátu pro SQL dotaz, podle kterého se zrcadlí pozemky do tabulky *t_filter*.

t_map_detail

Uchovává parametry mapového náhledu pro snazší práci s mapou. Obsahuje identifikátor uživatele, kterému zadaná data patří a detailní informace o zobrazené části mapy včetně přiblížení.

4.6 Návrhy řešení

Následující text obsahuje nástin řešení jednotlivých netriviálních požadavků, jejichž vyřešení bylo během návrhu potřeba domyslet.

Rozšíření MapScript bylo použito pro snadné vykreslení vektorových dat do webové aplikace. Pracuje na principu vrstvení obrázků na sebe a vrácení výsledného obrázku do prohlížeče, složeného ze všech požadovaných rastrových a vektorových vrstev dostupných z různých zdrojů.

Použití tabulek *t_select* a *t_filter* bylo nutné z důvodu omezené možnosti spojování dat z více tabulek při zobrazování pomocí rozšíření MapScript. Do těchto tabulek se proto zrcadlí data, která se později vypisují do vektorové vrstvy zmiňovaným rozšířením. Toto řešení je jistým ústupkem snižujícím rychlost práce s databází, ale v předpokládaném rozsahu využívání aplikace poměrně zanedbatelným.

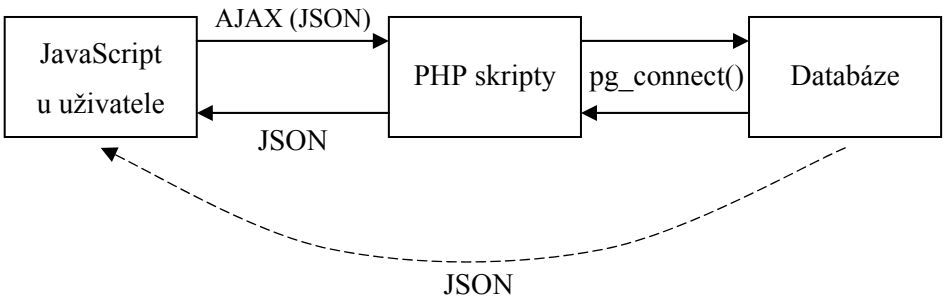
Kvůli snazší orientaci v zobrazovaných pozemcích byla mezi požadavky funkce filtrování. Princip filtrování je založen na výběru požadovaných kriterií vycházejících z vlastností pozemků. Vytvářejí se pravidla specifikovaná pro jednotlivé vlastnosti, která se ukládají do databáze. Kombinací těchto pravidel se vyberou jen odpovídající pozemky a ty se následně zrcadlí do tabulky *t_filter*.

Pro rychlejší a snazší práci s celým systémem bylo nutné zvolit technologii AJAX, aby se eliminovalo zbytečné načítání všech částí aplikace a mohla se obnovovat jen ta nejnutnější. Především není nutné neustálé načítání mapového náhledu při činnostech, během kterých se nemění. Aktualizace mapy znamená největší zátěž pro server, protože je nutné se připojovat ke vzdáleným serverům a pracovat s grafickou knihovnou.

5 Implementace

Aplikace je rozdělena do více souborů, především kvůli přehlednosti. Samotná struktura systému je pak tvořena třemi pomyslnými částmi podobně jako je tomu v návrhovém vzoru MVC. Jen s tím rozdílem, že byl kladen důraz především a co největší přenesení aplikační logiky na model.

V prvé řadě uloženými procedurami v databázi, které provádí většinu aplikační logiky. Tyto procedury se volají pomocí PHP, které přistupuje k databázi. Volání těchto skriptů pak probíhá formou AJAX požadavků posílaných pomocí jQuery. O zobrazování přijatých dat se starají JavaScriptové funkce využívající jQuery. Všechny tyto tři části při posílání dat používají JSON formát, aby práce s nimi v JavaScriptu byla co nejsnazší. Pro ilustraci je systém volání znázorněn na obrázku 5.1.



Obrázek 5.1

5.1.1 Datová část

Je to nejdůležitější část celého systému. Samotný datový model rozložení tabulek najdete v příloze A. Kromě běžných tabulek uživatelů, pozemků, nebo například vlastníků pozemků, na kterých není mnoho co popisovat a jejichž vazby jsou znázorněny ve zmíněné příloze, bych rád upozornil na tabulky *t_select* a *t_filter*. Jedná se o dvě pomocné tabulky pro ukládání vybraných popř. vyfiltrovaných pozemků.

Při každém vybrání pozemku se tento pozemek zrcadlí do tabulky *t_select* s identifikátorem uživatele. Z ní se pak zobrazuje do mapy ve vrstvě *pozemky_selected*. Je to především proto, že rozšíření MapScript špatně pracuje s dotazy na více než jednu tabulku. Pro každého uživatel tabulka obsahuje vždy maximálně jeden záznam, protože není možné mít zároveň vybráno více pozemků.

Obdobně se chová i tabulka *t_filter*, která ovšem pro jednoho uživatele může obsahovat záznamů více, protože je možné, že filtrovanému pravidlu bude odpovídat více záznamů, popřípadě zadaných pravidel bude větší množství. Samotná pravidla se pak ukládají do tabulky *t_filter_rule*.

Databáze obsahuje větší množství funkcí na práci s daty a vytváří tak jednoduché rozhraní pro obsluhu databázových tabulek. Funkce začínají vždy předponou *poz*, aby se tak snadno odlišily od funkcí v databázovém systému obsažených. Všechny činnosti prováděné systémem se volají jednoduše, jen jako název uložené procedury napsané v PL/pgSQL. Ta přebírá data nejčastěji jako textový řetězec v JSON formátu a pak ho rozdělí na požadované položky, nebo jako konkrétní datové typy u jednoduchých parametrů. Vrací opět nejčastěji data ve formátu JSON, kde jednou z častých položek je položka error, obsahující číslo chyby, popřípadě 0 pokud vše proběhlo úspěšně.

Pro převod dat z JSON formátu do položky typu RECORD slouží funkce *poz_JSONToRecord*. Výsledkem je položka, na kterou se lze v PL/pgSQL odvolávat stejně jako například na objekt v JavaScriptu. Čili uložím-li si výsledek této funkce v libovolné jiné funkci do proměnné *a*, k jednotlivým položkám se odvolávám *a.polozka*. Nástin funkce na převod je uveden v kódu 5.1.

```
CREATE OR REPLACE FUNCTION poz_JSONToRecord(text)
  RETURNS RECORD AS $$
    DECLARE rec RECORD; row text; pair text; var text;
             value text; sel text := '';
    BEGIN
    IF length($1) > 0 THEN
        row := $1;
        row   := substring(row from E'^{(.*)}$');
        WHILE length(row) > 0 LOOP
          pair  := substring(row from
                    E'([^"]*"[^"]*"\\s*:\\s*({[^}]*}|"[^"]*"|[^,]*))');
          var   := substring(pair from E'"([^"]*)"');
          value := substring(pair from
                    E'"[^"]*"\\s*:\\s*"?(({[^}]*})|([^"]*))"?\\s*');
          row   := substr(row,length(pair)+1);
          IF length(sel) > 0 THEN
            sel := sel || ', ';
          END IF;
          sel := sel || '''' || value || '''::text AS ' || var;
        END LOOP;
        EXECUTE 'SELECT ' || sel INTO rec;
        RETURN rec;
    ELSE
        RETURN NULL;
    END IF;
    END;
  $$ LANGUAGE plpgsql;
```

Kód 5.1

Naopak data, která jednotlivé funkce volané ze systému vrací, jsou převáděna opět do JSON formátu a ve výsledku už rovnou prostřednictvím PHP předávány JavaScriptu. Jako příklad může sloužit funkce *poz_selectUsers* vracející pole všech uživatelů jak je vidět v kódu 5.2.

```
CREATE OR REPLACE FUNCTION poz_selectUsers()
   RETURNS text AS $$
      DECLARE row text; rec text := '{title:''new user'', uid: -1, '
                                 || ' uname:''name'', usurname:''surname'', '
                                 || 'ulogin:''login'', upass:''pass''},';
    BEGIN
      FOR row IN
         SELECT '{title: ''' || uname || ' ' || usurname || ''', uid: '
                || uid || ',' || ' uname:''' || uname || ''', usurname:'''
                || usurname || ''',' || ' ulogin:''' || ulogin
                || ''', upass: ''''}'
                FROM t_user WHERE ugroup = 2 ORDER BY usurname
      LOOP
         rec := rec || row || ',';
      END LOOP;
      RETURN '[' || rec || ']';
    END;
  $$ LANGUAGE plpgsql;
```

Kód 5.2

Ukládání nových pozemků do databáze využívá možnosti vkládat data ve formátu WKT, neboli Well Know Text. Je to textový formát dobře čitelný pro člověka, ze kterého se vytvoří binární formát pro uložení do vnitřní struktury databáze, jak jej specifikuje PostGIS. Zapisuje se počátečním klíčovým slovem jako je POINT, LINE nebo POLYGON a následují čárkou oddělené souřadnice jednotlivých bodů. Tohoto systému využívá aplikace při ukládání pozemků ve funkci *poz_saveAcre* nebo i při výběru pozemků ve funkci *poz_selectPozemek* naznačené v kódu 5.3.

```
CREATE OR REPLACE FUNCTION poz_selectPozemek (text, integer)
   RETURNS boolean AS $$
      BEGIN
      IF EXISTS ( SELECT sid FROM t_select
                  WHERE
                    CONTAINS(sgeom, GeomFromText('POINT(' || $1 || ')', 2065))
                    AND uid = $2)
      THEN
         DELETE FROM t_select WHERE uid = $2;
         RETURN TRUE;
      ELSE
         DELETE FROM t_select WHERE uid = $2;
```

```
            INSERT INTO t_select (pid, uid, sgeom)
               SELECT pid,$2,pgeom FROM t_pozemky
               WHERE CONTAINS(pgeom, GeomFromText('POINT(' || $1 || ')', 2065));
            RETURN TRUE;
         END IF;
         END;
   $$ LANGUAGE plpgsql;
```

Kód 5.3

Jak ze zadání problému vyplývá, není možné mít v databázi uložené pozemky, které by se překrývaly. Proto bylo nutné vyřešit tento problém při vkládání nových pozemků do databáze. A tak se ve funkci *poz_saveAcre* od požadovaného tvaru pomocí prostorové algebry odečítají všechny v databázi již obsažené pozemky. Pro názornost je část funkce, starající se o toto ořezávání, zobrazena v kódu 5.4.

```
polygon := 'POLYGON((' || polygon || substring($1 from E'^([^,]*),') ||'))';
SELECT GeometryFromText(polygon, 2065) INTO geom;
FOR geom2 IN SELECT pgeom FROM t_pozemky LOOP
   SELECT difference(geom, geom2) INTO geom;
END LOOP;
```

Kód 5.4

Toto ořezávání ale vede k jednomu problému, který bylo také nutné ošetřit. A to když se při tomto ořezávání původní jeden celistvý objekt rozloží na více menších částí. Kdyby se tento problém neošetřoval, do databáze by se pozemek uložil jako jeden objekt, fyzicky ovšem složen z několika nezávislých částí. Toto není ale v rámci reálného světa možné a tak bylo nutné tuto skutečnost kontrolovat a při vytvoření více objektů je do databáze uložit postupně. O této skutečnosti informovat uživatele a označit pouze první z takto vytvořených objektů. Část funkce řešící tento problém, je naznačena v kódu 5.5.

```
SELECT NumGeometries(geom) INTO num;
...
FOR num IN 1..num LOOP
   IF NOT EXISTS(
          SELECT pid FROM t_pozemky WHERE pgeom like GeometryN(geom,num))
   THEN
      INSERT INTO t_pozemky (pgeom)
             VALUES (GeometryN(geom,num))
             RETURNING pid INTO curr_id;
      IF id = 0 THEN
         id := curr_id;
         END IF;
```

```
        ELSE
            inDB := true;
    END IF;
END LOOP;
```

Kód 5.5

5.1.2 Obsluha pomocí PHP

Systém využívá PHP jen k těm nejnutnějším činnostem, jako je například připojování k databázi, ošetření vstupů, aby systém odolal útokům typu SQL injection a podobě. Pro práci s databází je připravená funkce, která jako parametr přebere název uložené procedury včetně parametrů, odešle požadavek databázi a vrací její odpověď. Protože formát JSON pracuje s kódováním ve formátu unicode, které databáze nedokáže zpracovat, před odesláním se znaky převádí na UTF8.

Díky přenesení veškeré aplikační logiky na databázi obsahují PHP skripty jen velmi málo kódu. Ale pro přehlednost jsou členěny podle funkcí, i když by se daly sepsat do jednoho a jen přepínat podle vhodně zvolených parametrů. Výsledkem je například soubor, ukládající informace o pozemku ukázaný v kódu 5.6.

```
<?php
/*
 $Id: save.php 22 2008-04-18 03:32:58Z $
 */

include_once "../../config.php";
include_once "../../db.php";
include_once "../../session.php";

echo exec_sql_func("poz_savePozDetail('" .
                   pg_escape_string(json_encode($_POST)) . "')");
?>
```

Kód 5.6

Nejrozsáhlejší využití PHP je ve skriptu *map.php*, kde se pracuje s funkcemi rozšíření MapScript. To slouží k pohybu v mapě a jejímu vykreslování. V podstatě se jedná o funkce pro manipulaci s daty obsaženými v souboru s příponou *map* využívaného MapScriptem. Více informací najdete v již zmíněném souboru nebo na stránkách referenční příručky k tomuto PHP modulu [9]. Obecný princip vykreslování mapového náhledu je založen na kombinaci obrázků z různých zdrojů vrstvených na sebe pomocí grafických knihoven PHP a zobrazení výsledného obrázku v systému.

Posledním využitím PHP je tvorba hierarchické struktury menu a jejich položek. Tyto informace není nutné mít uložené v databázi, protože s daty nijak výrazně nesouvisí. A protože je žádoucí, aby i tyto údaje byly reprezentovány formátem JSON, předávají se tyto informace o

struktuře také pomocí PHP. Aby byl zápis přehlednější a snazší než přímý zápis v JSON, využívá se na převod PHP funkce *json_encode*. Data se přehledně napíší ve formě několika rozměrného pole a poté pomocí zmíněné funkce převedou na požadovaný formát, který se jen vypíše pro použití v JavaScriptových funkcích.

5.1.3 Prezentace uživateli

Toto je pro uživatele asi ta nejpodstatnější část. Jedná se o JavaScriptové funkce, které přebírají data v JSON formátu a prezentují je uživateli. Pro implementaci většiny funkcí a jednodušší práci s DOM strukturou dokumentu jsem použil již výše zmíněný framework jQuery.

Základní struktura dokumentu je definována v souboru *index.php*, který také slouží jako jediný soubor zobrazovaný uživateli. Díky technologii AJAX, se spouští všechno z této stránky, příchozí data se zpracovávají a vykreslují dynamicky přímo do těla stránky.

Tato jedna stránka se skládá ze čtyř základních částí, které jsou na sobě nezávislé, jen se občas ovlivňují. Jmenovitě to jsou záhlaví, kde se zobrazuje datum, čas a přihlášený uživatel. Lišta záložek sloužící k přepínání mezi jednotlivými částmi systému. Boční informační panel nazvaný sidebar, pro zobrazování všech potřebných informací dané sekce. A jako poslední, oblast obrázku s mapou, která slouží pro navigaci na mapovém podkladu a práci s pozemky. Všechny tyto části nejsou pevnou součástí zdrojového kódu stránky *index.php*, ale po načtení DOM struktury se spojí se serverem a požádají ho o strukturu, podle které se do stránky doplní.

Všechny požadavky posílané AJAXem jsou řešeny co nejjednodušeji a to formou obyčejných odkazů v HTML. Jen s tím rozdílem, že po načtení se všem nastaví na událost kliknutí daná JavaScriptová funkce. Ta způsobí vyvolání AJAX požadavku odeslaného na adresu danou atributem HREF a vracení logické hodnoty *false*, která zapříčiní, že je odkaz z pohledu prohlížeče neaktivní. Tímto způsobem se snadno zabrání odchodu ze stránky a přitom jednoduchému předání potřebné hodnoty pro vyslání požadavku. Výsledek zpracovaný pomocí PHP a vracený jako řetězec JSON, je dále zpracováván a doplněn do požadovaného místa daného konkrétní situací. Takto se načítání dat větví a i výsledek prvního dotazu může vést k odeslání dalšího požadavku na data vnitřní struktury zobrazeného prvku.

Příkladem může být například přepínání mezi jednotlivými záložkami. Při kliknutí na záložku se pošle požadavek na strukturu dat v sidebaru. Ta je obecně dána několika položkami, mezi kterými se dá přepínat. A aby nebylo nutné načítat data pro všechny i neviditelné položky, při načtení dat do sidecaru, se první aktivuje a to vyvolá další požadavek na obsah konkrétní položky. A takto i tato položka může o část svých dat požádat server a počkat na odpověď. Příkladem je třeba roletové menu, které svůj obsah načítá samostatně z databáze.

Struktura jednotlivých položek v bočním menu je dána následujícím formátem:

```
Title:      Detail pozemku                  // titulek položky v sidebaru
Type:       acre                            // typ položky (určuje CSS třídu)
Href:       include/sidebar/acre/detail.php // adresa dat pro danou ploţku
Data:       „textová data“                  // případně jen textová data
                                            // pokud není předán href
Callback:   acreDetailDraw()                // funkce zpracovávající obdržená data
```

Podobný formát dat se používá i u ostatních částí stránky, které se načítají dynamicky a je jejich obsah potřeba vytvořit pomocí JavaScriptu. Díky tomuto systému předávání dat se může vysílat větší množství požadavků, protože přenos je odlehčen od zbytečných dat popisujících, jak má daná položka vypadat a přenáší se opravdu jen to podstatné.

6 Popis systému

Popis systému z pohledu uživatele je nutné začít vysvětlením principu práce s uživateli. Neboť jak již bylo zmíněno pro vybírání nebo filtrování pozemků je potřeba uživatele nějak unikátně odlišit od ostatních, aby si navzájem nezasahovali do práce.

Tady se nabízí dvě možná řešení, jak se problému vyhnout. Buďto zakázat nepřihlášenému uživateli vybírat a filtrovat pozemky. Což je v našem případě dosti nevhodné. Znamenalo by to, že uživatel, který se na tento systém dostane, by vlastně nemohl nic dělat, dokud by se nepřihlásil. To by řešil systém registrace, kdy by se nový uživatel zaregistroval a později se mohl přihlásit. Ale z pohledu systému na evidenci katastrálních pozemků malé obce, kde by mohly na radnici tento systém zpřístupnit libovolným návštěvníkům svých stránek, je to poměrně nevhodné. A především pro návštěvníka dosti nepohodlné, aby se jen kvůli prohlížení musel registrovat. Většina uživatelů internetu se registrace zalekne a může je i odradit.

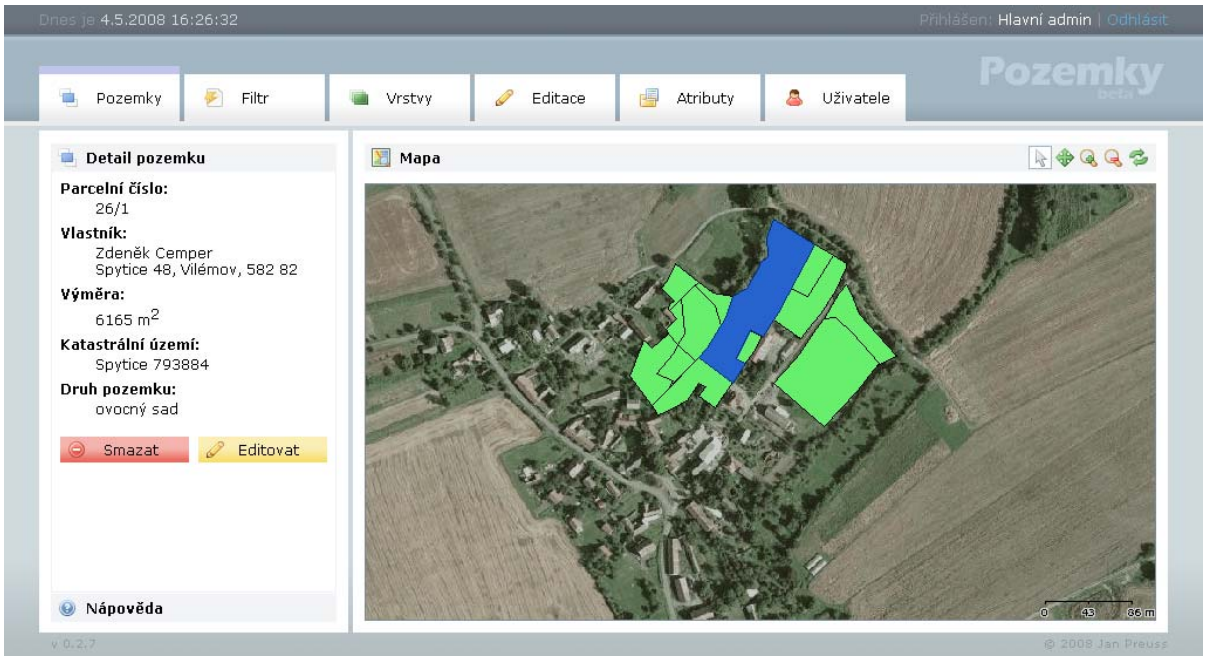
Jiným řešením je každému nově příchozímu návštěvníkovi v databázi vytvořit účet a od té doby s ním pracovat. Tento systém má nevýhody, že eviduje všechny příchozí návštěvníky v databázi a časem může být tato databáze uživatelů až příliš velká. Vycházejme však z předpokladu, že se jedná o systém malé obce a návštěvníků nikdy nebude velké množství. Navíc se počet návštěvníků v databázi eliminuje a pravidelně se mažou návštěvníci starší než 30 dní. A především při první návštěvě se uživateli vytvoří cookies s informací o tom, že už je veden v databázi a při příští návštěvě se použije tento účet. To má i tu výhodu, že když uživatel opustí systém a měl vybrán nějaký pozemek, ten zůstane do příští návštěvy evidován v databázi. A při dalším přihlášení se zobrazí stejné pracovní prostředí, jako když ho uživatel opustil.

Samozřejmostí je, že takto automaticky vytvořený uživatel má zpřístupněny jen ty nejzákladnější funkce systému, aby do něj nemohl nijak zasahovat, ale měl možnost si prohlížet dostupné informace.

Aby ovšem bylo možné s objekty pracovat i jinak, než si je jen prohlížet, bylo nutné zavést uživatele s vyšší prioritou, kteří budou moci pozemky i editovat a upravovat jim vlastnosti. V tomto případě je registrace nevhodná, aby se nemohl registrovat kdokoliv. Proto je při instalaci v databázi vytvořen jeden speciální uživatel *admin*, který má oproti běžným uživatelům možnost i přidávat, mazat a editovat všechny uživatele systému. Jednotlivé role systému lze dobře vidět v příloze B.

6.1 Uživatelské rozhraní

Uživatelské rozhraní je, jak již bylo zmíněno, rozděleno na čtyři základní oddělené části, které plní každá svou funkci v rámci systému. Jak je vidět na obrázku 6.1.

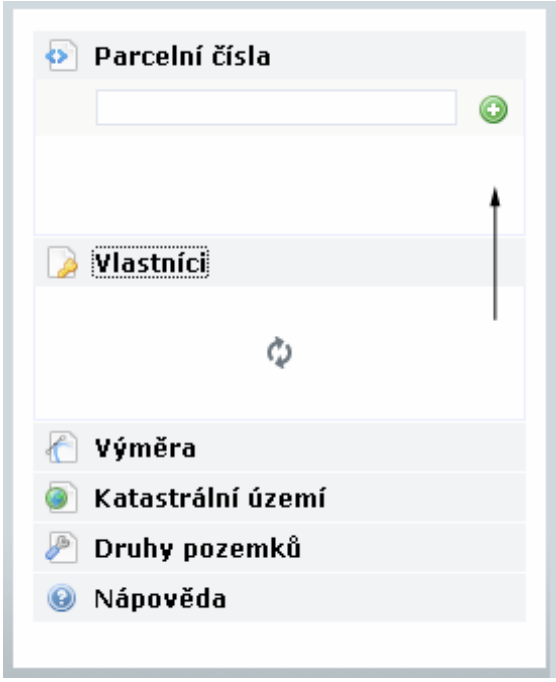


Obrázek 6.1

Horní lišta zobrazuje nalevo čas a datum a napravo informace o přihlášeném uživateli. Pokud je uživatel přihlášen, zobrazí se tam jeho jméno a tlačítko odhlásit. Pokud je uživatel pouze jako návštěvník, vypíše se slovíčko *guest* a číslo, které mu systém přidělil následované tlačítkem přihlásit. Po kliknutí na toto tlačítko se vysune dialog pro přihlášení.

Pod tímto pruhem je lišta záložek pro přepínání mezi jednotlivými funkcemi systému. Aktivní záložka má nad sebou barevný pruh, odlišující jednotlivé funkce od sebe. Podrobný popis jednotlivých záložek je dále v textu.

Na levé straně stránky je informační panel zobrazující data potřebná pro danou sekci. Obecně může obsahovat několik položek, mezi kterými se dá přepínat. Při prvním načtení se zobrazí první tato položka a ostatní jsou naskládány dole. Po kliknutí na libovolnou další položku se současně schová doposud aktivní položka a pomocí animace se rozsune nová položka, jak je naznačeno na obrázku 6.2.



Obrázek 6.2

Největší část zabírá oblast s mapou, která slouží k manipulaci s mapovým náhledem, pro vybírání pozemků a jejich možnou editaci. Nad kterou je nástrojová lišta s tlačítky pro práci s mapou viz. obrázek 6.3. Jednotlivá tlačítka slouží pro výběr objektu, posun mapy, přiblížení, oddálení a obnovení mapového náhledu. Právě aktivní nástroj je orámován a podbarven bíle.

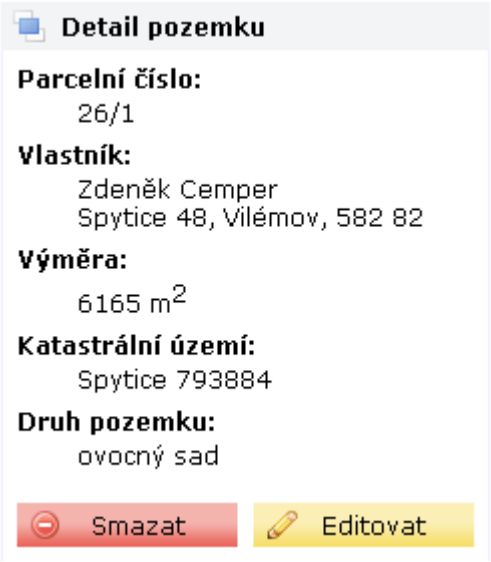
Obrázek 6.3

6.1.1 Záložka pozemky

V této záložce je možné si prohlížet aktuálně vybraný pozemek. K výběru slouží symbol šipky z nástrojové lišty nad mapou. Je-li tento nástroj aktivní, pak stačí kliknout do libovolného místa na mapě, a pokud je na tomto místě nějaký pozemek, tak se vybere a jeho detail se zobrazí nalevo. Pokud byl do teď nějaký pozemek aktivní a uživatel klikne mimo všechny pozemky nebo opět na vybraný pozemek, výběr se automaticky ruší. Jinak je pozemek vybraný po celou dobu práce a i po odhlášení se tato informace uchovává v databázi pro pozdější přihlášení. Vybrané pozemky se zvýrazňují modrou barvou, aby se odlišily od ostatních pozemků.

Je-li uživatel v systému přihlášen a ne pouze jako návštěvník, má možnost pozemek nejen prohlížet, ale také smazat a editovat jeho vlastnosti. Jak je vidět na obrázku 6.4. Kliknutím na tlačítko editace, se detail automaticky přepne do editačního režimu a uživatel může snadno měnit detail

pozemku. Pokud změny nechce ukládat, použije tlačítko *Zrušit*, které je umístěno na místě původního tlačítka *Editovat*, aby když uživatel omylem klikne při editaci dvakrát, nehrozilo žádné nebezpečí. Naopak potvrzovací tlačítko *Uložit* je na druhé straně bočního panelu.



Obrázek 6.4

6.1.2 Záložka filtr

Filtrování probíhá na zcela jednoduchém principu. Podle vlastností, které mají jednotlivé pozemky je lze snadno filtrovat. Lze tedy filtrovat podle parcelních čísel, vlastníků, výměry, katastrálního území a druhu pozemku. Princip filtrování je založen na kombinaci pravidel, které uživatel zadá. A podle těchto pravidel se poté vyberou ty pozemky, které tyto pravidla splňují a zobrazí se do vrstvy v mapě žlutou barvou. Stejnou jako je barva záložky, aby bylo jasné, které pozemky jsou vybrané a které vyfiltrované.

Jednotlivá pravidla lze velmi snadno přidávat, stačí si z boční nabídky vybrat požadovanou skupinu pravidel a pak přidat konkrétní pravidlo, kterým chce uživatel pozemky filtrovat. Jak je znázorněno na obrázku 6.5. Pravidlo pak odstraní kliknutím na něj. Tímto stylem se kombinují všechny pravidla a nastaví-li uživatel pravidlo v sekci vlastníků a druhu pozemku, výsledkem budou všechny pozemky odpovídající alespoň jednomu pravidlu. Zjednodušeně se dá říci, že čím více pravidel uživatel použije, tím více pozemků se vybere.

Pravidla u vlastníků a druhu pozemku se definují jen vybráním z roletového menu a kliknutím na tlačítko se symbolem přidání.

Obrázek 6.5

U přidávání pravidel v sekci parcelních čísel je možné napsat buď konkrétní číslo anebo použít zástupný znak %, který znamená libovolný počet dalších znaků. Například pokud chci vybrat všechny pozemky začínající číslem 30, můžu napsat 30%. Systém je ovšem ošetřen na jiné nedovolené vstupy a tak nehrozí například útoky typu SQL injection.

V sekci výměra se jednotlivá pravidla definují se speciálním symbolem na začátku, který určuje, jestli se mají vybrat pozemky s přesnou výměrou, větší nebo menší než je zadaná hodnota. Význam jednotlivých operátorů je vcelku jasný. Znaménko > následované číslem vybere jen ty pozemky, které mají výměru větší než zadané číslo. Znaménko < dělá přesný opak, a pokud uživatel napíše = popřípadě znaménko vynechá úplně, vyberou se pozemky s výměrou přesné hodnoty. Pokud uživatel místo povolených symbolů anebo číslic použije jiné znaky, řetězec se automaticky považuje za hodnotu nula a s ní se dál pracuje.

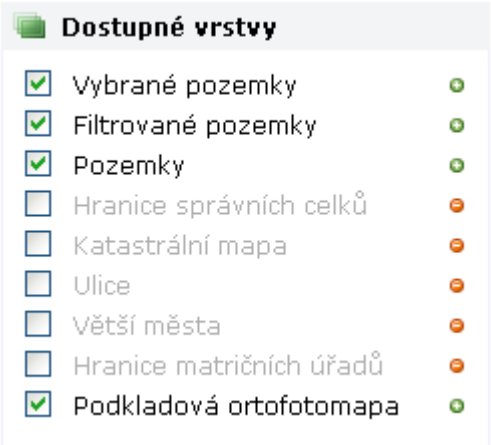
U filtrování katastrálního území se nedá předpokládat, že uživatel použije přesný název, takže se zadaný text hledá v celém názvu katastrálních území jednotlivých pozemků. Vytvoří-li uživatel například nové pravidlo obsahující pouze jeden znak *a*, vyberou se všechny pozemky, které spadají do katastrálního území, obsahujícího kdekoliv v názvu právě toto písmeno.

6.1.3 Záložka vrstvy

Pro usnadnění práce s mapou je vhodné mít aspoň částečnou možnost volby mapového podkladu pod vrstvou vykreslující pozemky. Užitečná je podkladová ortofotomapa, čili mapa leteckých snímků krajiny. Z této mapy moc informací vyčíst nelze, ale doplňuje prostorovou představu o konkrétních pozemcích a vytváří tak dojem, že se nejedná pouze o seskupení polygonů určité barvy na jednobarevném pozadí. Naopak podkladová vrstva rastrové katastrální mapy není příliš estetická, ale nese sebou obrovské množství informací potřebných pro editaci pozemků. Jako jsou hraniční čáry pozemků udávající vymezení jednotlivých pozemků v prostoru a pak čísla jednotlivých parcel pro možné vyhledání v katastru nemovitostí. Ostatní dostupné vrstvy slouží spíš jako pomocné vrstvy při

orientaci v prostoru celé České republiky, protože většina map se zobrazuje jen v určitých měřítcích. Například zmiňovaná ortofotomapa se zobrazuje jen při větším detailním přiblížení, takže při zobrazení celé republiky není vidět vůbec. Proto jsou zde vrstvy s hranicemi správních celků a matričních úřadů, které se zobrazují i při větších oddáleních. Nebo větší města pro snazší orientaci v prostoru celé republiky. Poslední vrstva nazvaná ulice, pak zobrazuje ulice ve větších městech, kdyby bylo nutné se v nich orientovat.

Kliknutím na konkrétní mapovou vrstvu ji uživatel může buďto aktivovat anebo naopak deaktivovat. Pro lepší přehlednost jsou neaktivní mapy vypisovány jen šedou barvou, aby se tak odlišily od aktivních vrstev a na konci řádku každé vrstvy je ještě dodatečná ikonka symbolizující totéž. Přesně jak ilustruje obrázek 6.6. Vykreslování vrstev do obrázku probíhá v pořadí, jak jsou seřazeny v bočním panelu. Ty nejvýše napsané se vykreslí nahoře a naopak ty na spodu výpisu budou při vykreslování úplně nejníže.



Obrázek 6.6

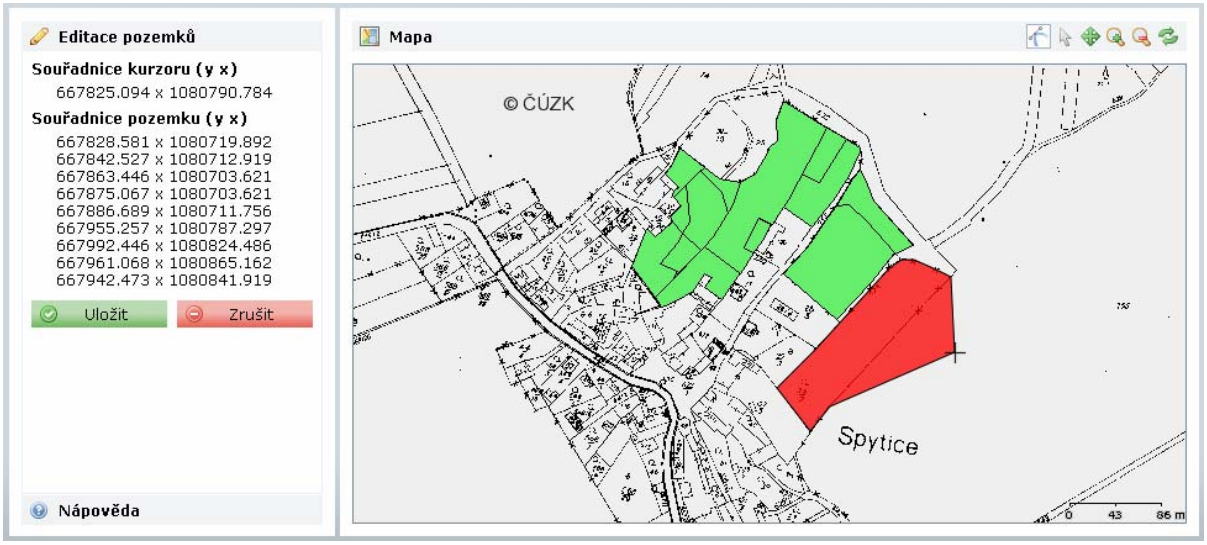
6.1.4 Záložka editace

Záložka editace slouží k přidávání nových pozemků do databáze. Přístup k ní mají jen přihlášení uživatelé. Protože by nemělo smysl zadávat pozemky například souřadnicemi, bylo nutné vymyslet jiný systém. Bohužel současná norma HTML, ani se sebelepším využitím JavaScriptu, nedokáže kreslit objekty. Prvek CANVAS by měla obsahovat až nová verze HTLM 5, která ale zatím není oficiálně uvolněná. Naštěstí prohlížeče, jako třeba Firefox nebo Safari, již kreslení objektů pomocí JavaScriptu zvládají a například u Internet Exploreru se dá pomocí přídavného JavaScriptu tato funkce taky téměř věrohodně simulovat. Díky tomu je možné přímo v prohlížeči kreslit základní vektorové objekty a tak pěkně a přehledně umožnit editaci pozemků přímo nad danou mapou.

V okamžiku, kdy uživatel přepne na záložku editace, v nástrojové liště nad mapou přibude nový nástroj pro editaci pozemků ukázaný na obrázku 6.7. Je-li tento nástroj aktivní, změní se kurzor nad mapou na kříž, aby bylo snazší přesně určit požadovaný bod. Používání tohoto nástroje je možné aktivovat a deaktivovat i kliknutím pravým tlačítkem myši nad plochou mapy. To aby se usnadnila práce se zadáváním nových tvarů.

Obrázek 6.7

Pokud je nástroj vybrán, v levé části se vypisují aktuální souřadnice kurzoru přepočtené na reálné souřadnice v mapě ve formátu *y, x*, tak jak je tomu v souřadném systému S-JTSK, který se používá v kartografických mapách v České republice. Kliknutím na libovolné místo na mapovém náhledu se aktuální souřadnice kurzoru uchová a vypíše opět do levé části stránky. Od tohoto okamžiku se za kurzorem táhne čára spojující uložený bod a aktuální pozici kurzoru. Při následném kliknutí do mapy se bod opět uloží a spojí s předchozím. V tento moment se již netvoří jen čára mezi aktuální pozicí kurzoru a posledním bodem, ale vytváří se trojúhelník, jehož vrcholy tvoří naklikané body a pozice kurzoru. Poslední vrchol trojúhelníku, se plynule pohybuje s kurzorem. Stejným způsobem se vytváří i složité mnohoúhelníky, kdy poslední vrchol se vždy pohybuje za kurzorem. V momentě deaktivace nástroje pro editaci pozemků se přestane polygon pohybovat za kurzorem a spojí se pouze první a poslední vybraný bod, čímž se docílí znázornění požadované oblasti. Barva vybraného mnohoúhelníku je červená, a aby bylo vidět i původní podklad, je prvek poloprůhledný. Příklad použití nástroje pro editaci pozemků je vidět na obrázku 6.8.



Obrázek 6.8

Výhodou používání reálných souřadnic je možnost posouvat s mapovým podkladem. Definovaný objekt se při posunu, zvětšování nebo zmenšování pohybuje zároveň s mapou. To je výhoda především při definici větších objektů, kde je potřeba pracovat s takovým přiblížením, které neumožňuje zobrazit celý objekt najednou. Uživatel by sice mohl mapu oddálit, ale tím by se dopouštěl zbytečně velkých nepřesností.

Takto definovaný objekt je následně možné zrušit a začít editovat nový pozemek anebo uložit do databáze mezi ostatní pozemky. V tom případě systém automaticky vybere vytvořený objekt a přepne na záložku pozemky, kde je možné takto vytvořenému objektu hned editovat vlastnosti.

Při definování pozemků není nutné snažit se přesně trefit hranici sousedního pozemku, klidně uživatel může přesáhnout do sousedních pozemků. Protože při ukládání se od definovaného tvaru ořízne všechny v databázi již uložené pozemky a výsledném je nový objekt, který přesně sousedí s ostatními, ale s žádným se nepřekrývá. Toto ořezávání ovšem vede k problému, kdy ořezáním vzniknou z původně jednoho objektu dva, nebo dokonce i více. V tom případě systém všechny části uloží jako jednotlivé pozemky a uživatele o tom informuje. Vybrán je potom první ze všech takto vytvořených objektů.

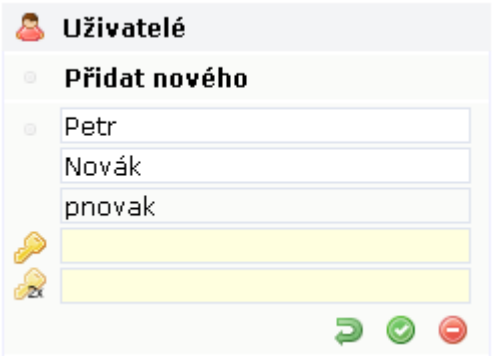
6.1.5 Záložka atributy

Definice a úpravy jednotlivých atributů pozemků je možné provádět v záložce nazvané atributy. Stejně jak u předešlé záložky, možnost měnit atributy mají opět jen přihlášení uživatelé. Tato záložka je rozdělena na dvě části, vlastníky a druhy pozemků. Princip editace je u obou typů podobný. Při prvním načtení se zobrazí výpis konkrétních atributů pod sebou. Kliknutím na libovolný z nich se tento záznam rozsune a je možné ho editovat. Po rozsunutí se objeví tři tlačítka, pro návrat k předchozím hodnotám a zrušení editace, uložení změn popřípadě smazání atributu. Mazat je možné jen atributy, které nejsou nastavené u žádného objektu. První položka v seznamu atributů je volba pro přidání nového. Tato položka se chová stejně jako ostatní, jen s tím rozdílem, že ji nelze smazat. Příklad editace atributu vlastníka je vidět na obrázku 6.9.

Obrázek 6.9

6.1.6 Záložka uživatelé

V záložce uživatelé je možné vidět všechny momentálně přihlášené návštěvníky a samozřejmě editovat jejich detailní informace. S tím opět souvisí jistá omezení. Tuto záložku vidí pouze uživatel *admin*, aby kdokoliv nemohl spravovat všechny uživatelské účty. Výpis a editace uživatelských účtů probíhá obdobně jako je tomu u zadávání atributů. Snad jen s tím rozdílem, že na konci jsou dvě speciální kolonky na zadání nového hesla. Pokud nejsou vyplněny, heslo zůstává nezměněno, jinak se uloží do databáze heslo nové. Pro kontrolu jsou tyto kolonky dvě, aby nedocházelo k omylům při změně hesla. Uživatele lze stejně jako v předchozím případě editovat, uložit či smazat. Tentokrát se při mazání kontroluje jen, jestli uživatel není právě přihlášen. Pro představu následuje opět ukázka v podobě obrázku 6.10.



Obrázek 6.10

7 Závěr

Úkolem této bakalářské práce bylo popsat možnosti ukládání dat v prostředí webu, porovnat jednotlivé způsoby s databázovým systémem PostgreSQL a vytvořit ukázkovou aplikaci využívající daného rozšíření systému.

Aplikace využívající rozšíření PostGIS by měla sloužit vedením menších obcí k organizaci pozemků v nejbližším okolí obce a usnadnit tak přehled a správu jednotlivých parcel.

Díky zaměření na snadnou obsluhu a možnost ovládání přes webové rozhraní, je práce s aplikací snadná a poměrně intuitivní, tak jak bylo specifikováno v požadavcích. Využitím technologie AJAX na komunikaci se serverem, urychlilo práci se systémem a zmenšilo počet přenášených dat. Takže je systém poměrně rychlý a omezený především rychlostí připojení konkrétního uživatele.

I přesto, že je systém zcela funkční, je možné ho i nadále rozšiřovat. Jedná se ovšem spíše o menší úpravy pro ještě intuitivnější ovládání systému, jako například posun mapy tažením obrázku nebo například přichytávání k vrcholům již definovaných pozemků a tak usnadnit volbu bodů nových pozemků.

Kromě přiložené verze je možné projekt vyzkoušet i v testovacím provozu na webové adrese http://pozemky.pavuciny.com. Kde se v době odevzdání práce aplikace nachází. Pro ukázku uživatelů je zde vytvořen smyšlený uživatel Petr Novák s přihlašovacím jménem *pnovak* a heslem *heslo*. Pro přístup k administrátorskému účtu slouží přihlašovací jméno *admin* a heslo taktéž *admin*.

Přínosem této práce bylo především osvojení si technik práce s jazykem PL/pgSQL a tvorby uložených procedur v databázovém systému PostgreSQL. Návrh struktury komunikace databáze s uživatelským rozhraním pomocí AJAX požadavků a vytvoření vlastní struktury přenášených dat. Zároveň i úvod do práce s rozšířením MapScript a obecně práce s geografickými informačními systémy, od kterých se očekává ještě velká budoucnost a je to v dnešní době rychle se rozvíjející odvětví.

Literatura

[1] Abbey, M., Corey M., Abramson I. *Oracle9i*. Praha, SoftPress 2002.

[2] Oracle Česká republika [online]. Oracle [cit. 10. 05. 2008].
Dostupné z http://www.oracle.com/global/cz/index.html

[3] MySQL [online]. MySQL [cit. 10. 05. 2008].
Dostupné z http://www.mysql.com

[4] Momjian, B. *PostgreSQL, praktický průvodce*. Brno, Computer Press 2003.

[5] PostgreSQL: Documentation [online]. PostgreSQL [cit. 05. 05. 2008].
Dostupné z http://www.postgresql.org/docs/

[6] Jemný úvod do jazyka PL/pgSQL PostgreSQL [online]. Pavel Stěhule [cit. 05. 05. 2008]
Dostupné z http://postgresql.ok.cz/doc/plpgsql.html

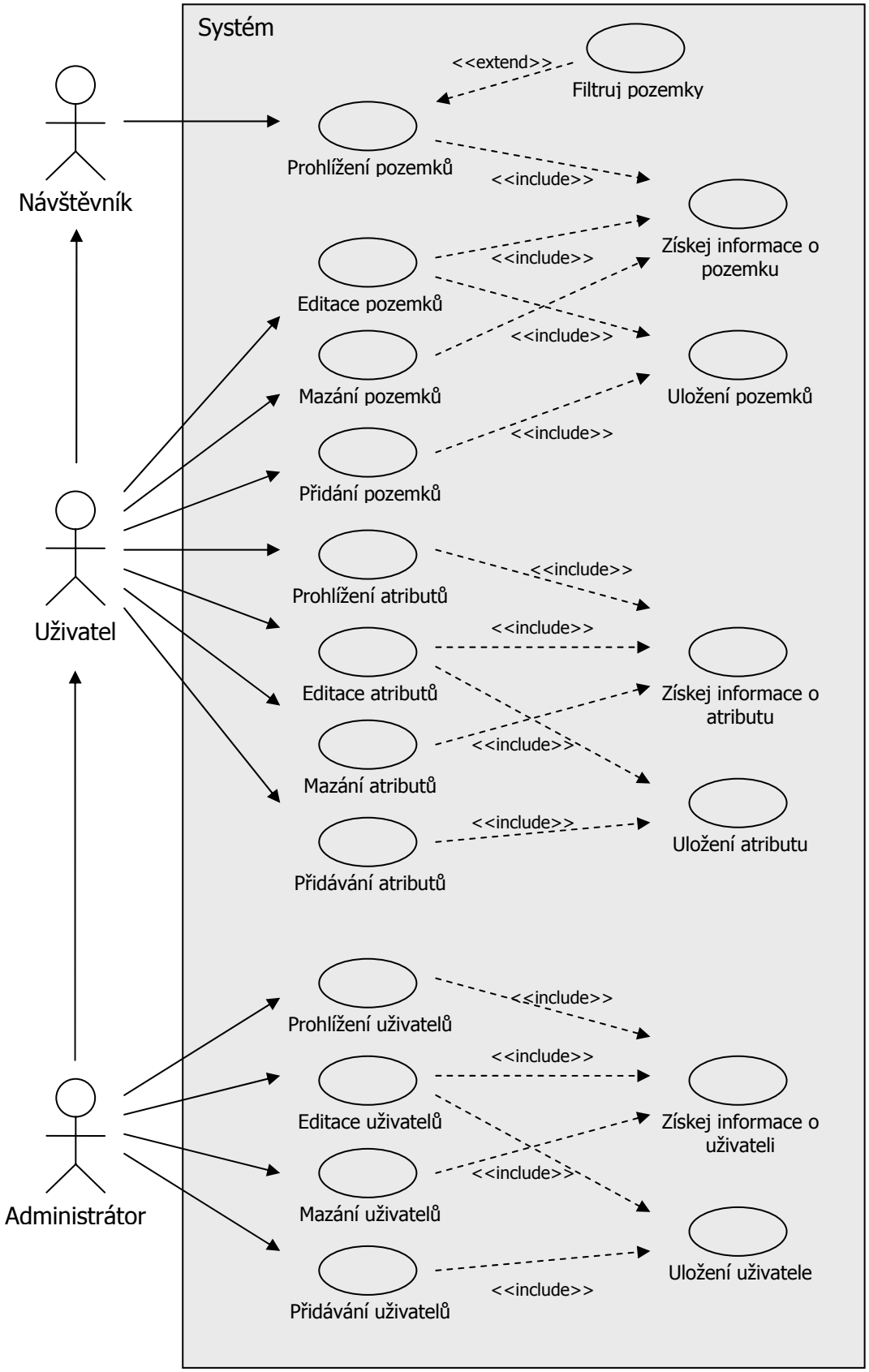
[7] PHP: manuál PHP [online]. PHP [cit. 05. 05. 2008]
Dostupné z http://www.php.net/manual/cs/

[8] jQuery JavaScript Library [online]. jQuery [cit. 05. 05. 2008]
Dostupné z http://docs.jquery.com/Main_Page

[9] PHPMapScript.MapTools.org [online]. MapTools [cit. 05. 05. 2008]
Dostupné z http://www.maptools.org/php_mapscript/index.phtml?page=phpmapscript-class-guide.html

Seznam příloh

Příloha A. Diagram případů užití



Příloha B. Databázový model