České vysoké učení technické v Praze
Fakulta stavební



DIPLOMOVÁ PRÁCE

Webová databáze benediktinských klášterů a její mapová nadstavba

Praha, 2013 **Ondřej Pospíšil**

České vysoké učení technické v Praze
Fakulta stavební
obor Geoinformatika



DIPLOMOVÁ PRÁCE

Webová databáze benediktinských klášterů a její mapová nadstavba

Web database of Benedictine monasteries and its map extension

Vedoucí práce: Ing. Jindřich Hodač, Ph.D.

Katedra geomatiky

Praha, 2013 **Ondřej Pospíšil**

Prohlášení

Prohlašuji, že jsem svou diplomovou práci na téma „Webová databáze benediktinských klášterů a její mapová nadstavba“ vypracoval samostatně a použil jsem pouze podklady uvedené na konci práce v oddílu „Literatura“.

V Praze dne _____ _____

podpis

Poděkování

Děkuji především vedoucímu bakalářské práce Ing. Jindřichovi Hodačovi, Ph.D., dále všem, kteří se na projektu podíleli nebo dosud podílejí, všem, kteří přispěli nápady a dodali inspiraci, případně pomohli konzultací. Jmenovitě zejména Mgr. Janu Kremerovi a Centru medievistických studií Akademie věd, zúčastněným odborníkům, doktorandům a přispěvatelům databáze. Dále děkuji Ing. Petrovi Soukupovi, Ph.D. za výraznou pomoc až spolupráci zejména na počátku projektu, včetně poskytnutí serveru. Také děkuji Majce Dlouhé za pomoc s nekonečnou kontrolou překlepů a v neposlední řadě děkuji také příbuzným, spolubydlícím a přátelům za trpělivost zejména ve chvílích, kdy se nedařilo vše jak mělo.

Abstrakt

Tématem této diplomové práce je tvorba databáze benediktinských klášterů s mapovou nadstavbou. Projekt probíhá ve spolupráci s Centrem medievistických studií. Základem je webová aplikace napsaná v jazyce PHP, s využitím Zend Framework a architektury MVC a databáze MySQL. Dále je využito javascriptové knihovny JQuery a mnoha plug-inů. Předmětem databáze jsou především samotné benediktinské kláštery ve střední a jihovýchodní Evropě, dále i jiné lokality, osobnosti působící v dané době, a také různé artefakty.

Databáze má několik vrstev uživatelů a propracovaný systém správy hesel. Druhou, oddělenou vrstvou je mapová aplikace geograficky zachycující zejména samotné lokality a jejich vztahy. Implementace mapy probíhá v Google Maps JavaScript API rozhraní a je využito i tzv. reverzní geokódování.

Klíčová slova

databáze, PHP, MySQL, informační systém, Zend Framework, MVC, JQuery, GIS, Google Maps API, benediktinské kláštery

Abstract

The subject of this thesis is a creation of an database of Benedictine monasteries with an application map extension. The project is proceeded in cooperation with Centre for Medieval Studies. The base is a web application written in PHP using Zend Framework, MVC architecture and MySQL database management system. It is used JQuery JavaScript library and the several plug-ins. The subjects of this database are the Benedictine monasteries in Central and Southeastern Europe, as well as other habitations, personalities active at that time, and various artifacts.

The database has a few layers of users and sophisticated system of administration of articles. A second and separate layer is the map application geographically depicting especially locations and their relationships. Implementation of mapping application is carried in Google Maps JavaScript API and is used reverse geocoding.

Keywords

database, PHP, MySQL, information system, Zend Framework, MVC, JQuery, GIS, Google Maps API, Benedictine monasteries

vložit originální zadání

(kvůli správnému číslování stránek)

Obsah

1 Úvod **13**
1.1 O projektu . . . 14

2 Volba technologie realizace **15**
2.1 Možnosti tvorby webové databáze . . . 15
2.1.1 PHP . . . 15
2.1.2 Zend Framework . . . 15
2.1.3 MySQL . . . 17
2.1.4 JQuery . . . 17
2.2 Možnosti tvorby mapy . . . 18
2.2.1 Google Maps API . . . 18
2.2.1.1 Mapy API V4 . . . 18
2.2.1.2 aMapy.cz API . . . 19
2.2.1.3 Yahoo Maps API . . . 19
2.2.1.4 Bing Maps . . . 20
2.2.1.5 HERE Maps API . . . 20
2.2.1.6 MapQuest . . . 21
2.2.1.7 OpenStreetMap . . . 21
2.2.1.8 Google Maps API . . . 21
2.2.1.9 Další mapové služby . . . 24

3 Popis informačního systému **25**
3.1 Informační systém z pohledu návštěvníka stránek . . . 29
3.1.1 Hlavní strana . . . 29
3.1.2 Postranní - pravý panel . . . 29
3.1.2.1 Rychlé hledání . . . 29
3.1.3 Uživatelský panel . . . 30
3.1.4 Hlavní menu . . . 30
3.1.4.1 Hledat - lokality . . . 30
3.1.4.2 Hledat - artefakty . . . 31
3.1.4.3 Hledat - osobnosti . . . 31
3.1.4.4 Hledat - fulltext . . . 31
3.1.4.5 Přehledy - Výpis hesel . . . 32
3.1.4.6 Mapa . . . 32
3.1.4.7 Mapa - zobrazení ve velkém okně . . . 32

3.1.5 Detail hesla . . . 35
3.1.5.1 Detail obrázku . . . 36
3.2 Informační systém z pohledu běžného registrovaného uživatele . . . 37
3.2.1 Hlavní strana . . . 37
3.2.1.1 Syntaxe psaní odkazů . . . 38
3.2.2 Postranní - pravý panel . . . 39
3.2.3 Uživatelský panel . . . 39
3.2.3.1 Interní zprávy mezi uživateli . . . 39
3.2.3.2 Uživatelská upozornění . . . 39
3.2.4 Menu . . . 41
3.2.4.1 Uživatel - Seznam uživatelů . . . 41
3.2.4.2 Uživatel - Editace osobních údajů . . . 41
3.2.4.3 Uživatel - Změna přihlašovacího hesla . . . 41
3.2.4.4 Detail uživatele . . . 41
3.2.4.5 Přidat - Lokalitu . . . 42
3.2.4.6 Syntaxe psaní poznámek v textu . . . 43
3.2.4.7 Přidat - Artefakt . . . 43
3.2.4.8 Přidat - Osobnost . . . 44
3.2.4.9 Přidat - Obrázek . . . 44
3.2.4.10 Přidat - Soubor . . . 44
3.2.4.11 Přidat - Návrh . . . 44
3.2.4.12 Přehledy - Návrhy . . . 44
3.2.4.13 Přehledy - Neschválená hesla . . . 44
3.2.4.14 Přehledy - Fotografie . . . 45
3.2.4.15 Přehledy - Soubory . . . 45
3.2.5 Detail hesla - rozdíl oproti návštěvníkovi stránek . . . 45
3.2.6 Editace hesla . . . 46
3.2.7 Editace obrázku nebo souboru . . . 46
3.3 Informační systém z pohledu administrátora . . . 46
3.3.1 Pravá lišta . . . 47
3.3.2 Detail hesla . . . 47
3.3.3 Editace hesla . . . 48
3.4 Informační systém z pohledu hlavního administrátora . . . 48
3.4.1 Hlavní strana . . . 49
3.4.2 Editace osobních údajů . . . 49
3.4.3 Uživatel - registrace uživatele . . . 49
3.4.4 Detail uživatele . . . 49
3.4.5 Další funkce hlavního administrátora . . . 50

4 Implementace **51**
4.1 Implementace databáze jako webové aplikace . . . 51
4.1.1 Návrh a implementace databáze . . . 52
4.1.1.1 Počáteční fáze návrhu . . . 52
4.1.1.2 Popis databáze . . . 52
4.1.1.3 Datové typy . . . 56

4.1.1.4 Správa databáze a změny její struktury 57
4.1.2 MVC aplikace v Zend Framework 57
4.1.2.1 Instalace a základní kroky 57
4.1.2.2 MVC architektura aplikace 64
4.1.2.3 Řadiče s akcemi a pohledy (view) 64
4.1.2.4 Modely 70
4.1.2.5 Ostatní prvky aplikace 74
4.1.3 Využití JavaScriptu 74
4.1.3.1 Využití AJAX 75
4.1.3.2 jQuery Validation Plugin 76
4.1.3.3 Dialog Widget 77
4.1.3.4 jQuery UI MultiSelect Widget 78
4.1.3.5 Image DropDown 78
4.1.3.6 Tablesorter a tablesorter.pager 79
4.1.3.7 Datepicker 80
4.1.3.8 Další využití JavaScriptu 81
4.2 Mapová část aplikace 82
4.2.1 Uložení polohy, reversní geocoding 82
4.2.2 Interaktivní mapa 84
4.2.2.1 MarkerWithLabel 84
4.2.2.2 Filtrování lokalit 87
4.2.2.3 Další funkce mapy 88

5 Závěr **93**

Literatura **98**

A Obsah přiloženého CD **I**

Kapitola 1

Úvod

Webová aplikace, databáze, GIS - to jsou jedny z nejvíce frekventovaných pojmů v projektu *Kultura a umění benediktinského řádu ve střední Evropě 800-1300*, který vede *Centrum medievistických studií* (dále CMS). Požadavkem CMS bylo právě vytvoření webové databáze, která by sloužila odborníkům i veřejnosti, a dále vytvoření mapy s odborným/tematickým obsahem (GIS), která by byla s databází přímo propojena.

Na daný projekt bylo relativně hodně času, a tak spolupráce při tvorbě probíhala průběžně tak, aby informační systém (jak databázi směle můžeme nazvat) odpovídal přesným požadavkům odborníků historiků, lidem kteří databázi budou plnit a uživatelům. To je ostatně jeden z hlavních faktorů při návrhu aplikace, ale i při její implementaci - aplikace je vyvíjena přesně na míru požadavkům, které se navíc v průběhu práce také poněkud měnily. Nelze tedy příliš využívat již hotových řešení, šablon, apod. V konkrétním případě to znamená např. implementaci v jazyce PHP a frameworku s MVC architekturou. Využití redakčního systému by v tomto případě prakticky nebylo možné. O tom ale až později.

Sjednotit myšlení programátora a logiku SQL databáze jako takové s myšlením odborníka historika, ale i s informacemi o klášterech jako takovými není tak jednoduché, jak by se mohlo zdát. Databáze má vždy jasnou logiku, strukturu, ale mnoho údajů a informací nejen o klášterech, které by v ní měly být, má velmi nejednoznačnou strukturu.

Příkladem může být údaj vzniku (datum) kláštera. Ten je často neznámý, a tak bylo potřeba zavést další pojem - *první zmínka*. Je potřeba na jednu stranu databázi navrhnout tak složitou a propracovanou, aby co nejlépe vystihovala realitu, na druhou stranu je potřeba zajistit hranici, kdy už je databáze a informační systém vůbec tak složitá, že by samotná práce s databází vyžadovala složité školení a byla nepřehledná. Úplně vystihnout realitu exaktní databází možné jednoduše není. Navíc čím více možností a různých propojení v databázi je, tím je větší riziko, že systém obsahuje nějaké chyby - a to i malé, které se projeví až při velmi speciálních případech. Příkladem zde může být situace, kdy jsou (v zadaném pořadí) v dané lokalitě (klášteře) vedeni představení (např. opati) - jsou případy, kdy je jeden představený v určitém klášteře více než jednou - a dokonce chronologicky hned v následujícím období. Je třeba skutečné informace někdy až „překroutit“ a „napasovat“ do kolonek databáze, a případnou nejednoznačnost vyřešit např. v textovém popisu hesla. Zde může být příkladem situace, kdy víme, že klášter byl založen někdy po roce 920. Ale nevíme přesně, zda už byl založen v roce 940 nebo 950.

Navíc například jeden zdroj uvádí jasné datum založení rok 925 a jiný jen „kolem roku 920“. V databázi je možné rok založení zadat jako pevný rok (925) anebo jako přibližné rozpětí (např. 920 - 940, tedy v databázi jsou uložené přímo tyto dvě hodnoty), které má sloužit především k tomu, aby bylo možné klášter vyhledat např. zadáním kritéria data založení. „Nestandardní“ situace, jako je tato - případ, kdy existuje hned několik názorů na rok založení - je ale nutné řešit tak, že se vyplní např. přibližné datum 920 - 950, a v textovém popisu dané lokality se rok založení vysvětlí podrobněji.

A především o tom vlastně celá spolupráce byla - hledání optimálního řešení a kompromisu možností databáze a skutečnosti, a o požadavcích, co je třeba ještě evidovat a co je už naopak mimo téma.

1.1 O projektu

Struktura databáze v mnohém připomíná známou *Wikipedii*. Tedy struktura, ne filozofie. Jedním z hlavních principů populární internetové encyklopedie je možnost editovat jednotlivá hesla kýmkoliv, tedy po registraci. Databáze klášterů na rozdíl od Wikipedie umožňuje editaci jednotlivých článků pouze registrovanými uživateli z řad odborníků historiků. Zatímco na Wikipedii je heslo ihned po vytvoření přístupné a viditelné veřejnosti a administrátoři jej mohou až zpětně smazat, v databázi klášterů se každé heslo publikuje pro veřejnost, jen pokud je hotové a schválené administrátorem. Navíc administrátorem s příslušnými právy - o tom ale až později.

Každé heslo má nejméně 4 vývojové fáze:

1. **návrh** - pokud je heslo ve fázi návrhu, jde v podstatě o výzvu ke pracování. Do první úpravy může heslo editovat kdokoliv registrovaný.

2. **rozpracováno** - vzniká buď první úpravou návrhu, nebo jednoduše vytvořením nového hesla. Každé heslo je odemčeno jen pro jednoho uživatele. Navíc jej může editovat jakýkoliv administrátor, pokud k tomu má příslušná práva.

3. **dokončeno** - od předchozího stavu se tato fáze liší jen „značkou“, že je dokončeno a připraveno ke schválení.

4. **schváleno** - pokud administrátor s příslušnými právy heslo schválí, je od této chvíle uzamčeno všem úpravám (to platí i pro administrátory). Zároveň je heslo viditelné pro veřejnost.

Pokud je potřeba heslo po jeho uzamčení opět rozpracovat nebo provést i jen drobnou úpravu, administrátor s příslušnými právy jej může opět odemknout, čímž jej vrátí do fáze *2 (rozpracováno)*. Odemknout heslo je tedy možné opět jen pro jednoho uživatele. Podrobnější popis vztahů a možností informačního systému není předmětem této kapitoly. Jedná se jen o názorné porovnání filozofie databáze ve srovnání s encyklopedií Wikipedia. Databáze z uživatelského hlediska je podrobněji popsaná v kapitole 3.

Kapitola 2

Volba technologie realizace

2.1 Možnosti tvorby webové databáze

Model, který je zde použit - tedy aplikace napsaná v jazyce PHP propojená s databází a mapová nadstavba v API jazyka JavaScript, je relativně populární a běžný. Navíc s podobným přístupem jsem se setkal již při tvorbě své bakalářské práce[1], byť ve znatelně menším rozsahu než zde.

2.1.1 PHP

V případě programovacího jazyka rozhodování neprobíhalo. Díky určité znalosti PHP z předchozích projektů jsem o jiném jazyku příliš neuvažoval, protože náskok daný znalostmi je výhodou použití tohoto jazyka. Každopádně alternativy PHP jsou např. Ruby, Perl, Visual Basic .NET ale i Java či C++ a další.

2.1.2 Zend Framework

Zde je třeba poznamenat, že jsem prakticky přeskočil otázku, zda framework vůbec použít, nebo zda vše psát v samotném PHP. Protože jsem již od počátku tušil rozsah aplikace jako poměrně rozsáhlého informačního systému, padla jednoznačně volba na použití frameworku. Implementace v samotném jazyce PHP má samozřejmě jednu výraznou výhodu - tedy rychlost. U složitějších aplikací jako je tato databáze se ale tento nedostatek zčásti smazává, především ale framework mnohonásobně zkrátí a ulehčí práci, zejména pokud využívá architekturu MVC. V samotném jazyce PHP byla implementována jednoduchá databáze v rámci mé bakalářské práce. Vytvořit aplikaci jako je tato databáze jen s pomocí samotného PHP je v reálném čase téměř nemožné.

Framework by měl kromě architektury MVC nabízet také třídy pro komunikaci s databází, pro tvorbu formulářů, pro překlady do jiných jazyků, zvládat obsluhu ajaxových požadavků, usnadnit tvorbu celkového layoutu, obsluhovat výjimky, hlavní konfigurace evidovat na jednom místě, případně spravovat ACL (seznam pro řízení přístupu). Všechny tyto požadavky splňuje právě *Zend Framework*, zde ale výběr nebyl tak jednoznačný.

K dispozici je velké množství více či méně podobných PHP frameworků, z nichž každý má své výhody a nevýhody. Porovnání frameworků je věc velmi subjektivní a není

předmětem diplomové práce. K jejich přehledu přidávám proto jen stručné základní údaje, příp. základní výhody / nevýhody:

Zend Framework

- open source
- implementovaný v PHP 5
- multiplatformní
- licence New BSD
- výhody: široce rozšířený, aktivně vyvíjený, od samotných vývojářů PHP
- nevýhody: zejména ve verzi 1 je mnoho komponent podle mnohých vývojářů implementována neefektivními postupy [2]; na menší projekty relativně „těžkopádné“ prostředí, složitější implementace formulářů, zejména dekorátorů

Nette Framework

- open source
- multiplatformní
- licence GNU GPL + Nette
- výhody: tvorba formulářů, jednoduchost, vývoj v ČR
- nevýhody: slabší dokumentace

CakePHP

- open source
- multiplatformní
- licence Mit

Symfony

- open source
- multiplatformní
- licence Mit

...a další, např. **Yii**, **CodeIgniter / Kohana**, **Laravel**...

K výběru prvního uvedeného mě vedlo zejména jeho rozšíření a široká komunita vývojářů - tedy aktivní vývoj a velké množství komponent. Nutno poznamenat, že je

využito verze **Zend 1.11.11**. *Zend Framework 2* byl vydán až v průběhu práce na databázi.

2.1.3 MySQL

Výběr *DBMS* (*Systém řízení báze dat*, někdy mylně nazývaný *databázový systém*) proběhl relativně rychle. Byl zvolen ten v současnosti nejčastěji využívaný, alespoň ve světě internetu.

Mezi výhody systému **MySQL** patří zejména velká kompatibilita s PHP a relativně vysoký výkon. Společně s *PHP* a serverem *Apache* tvoří nejčastěji využívanou kombinaci v databázových internetových aplikacích. Pro nekomerční použití je zdarma. Jde o systém multiplatformní a je k dispozici pod bezplatnou licencí GPL i pod komerční placenou licencí.

Mezi další alternativy MySQL patří například:

- **PostgreSQL** - objektově relační databázový systém, primárně vyvíjen pro Linux. Oproti MySQL a vlastně většině DBMS nabízí větší množství pokročilých funkcí. V případě databáze klášterů je ale mnoho těchto funkcí (vč. např. cizích klíčů) implementováno v samotném PHP / Zend Frameworku, tedy v databázovém systému nejsou nutné. Ohledně srovnání rychlosti s MySQL je známo, že pro velmi složité dotazy je výkonnější PostgreSQL, ale pro běžné příkazy `SELECT` nebo `INSERT` (tedy opět případ této databáze) MySQL [3].

- **SQLite** - jednoduchý relační databázový systém napsaný v jazyce C. Každá databáze je zde uložena v jednom samostatném souboru. Šířena je pod licencí *Public domain*.

- **ORACLE Database** - systém podporující nejen SQL, ale navíc i proprietární rozšíření *Oracle* nebo *PL/SQL*.

Pro samotnou aplikaci v Zend Frameworku není podstatné od počátku znát konkrétní DBMS, v konfiguračním souboru se totiž jen změní databázový adaptér.

2.1.4 JQuery

V aplikaci, kde je velké množství formulářů, ajaxových požadavků, a interaktivní mapa, je nezbytné zapojení *Javascriptu*. Bylo použito javascriptové knihovny **JQuery**. Tato knihovna byla vybrána zejména proto, že je zdaleka nejvíce rozšířená, velmi zjednodušuje implementaci ajaxových požadavků a navíc existuje nepřeberné množství pluginů pro všechny myslitelné situace - od validace formulářů, přes fotogalerie, ajaxové našeptávače po grafické efekty. Navíc byl použit i framework JQuery UI na této knihovně závislý. Ten nabízí mj. další formulářové prvky, nebo jejich vylepšení. Jde např. o posuvník, multiselect (tedy select, kde je možné vybrat na principu zakškrtnutí checkboxů i více než jednu možnost), modální i nemodální dialogy, kalendář, nebo efekty jako postupné skrytí nebo zobrazení nějakého elementu. Všem prvkům navíc dodává jednotný

vzhled, který je samozřejmě možné kdykoliv upravit pomocí kaskádových stylů.

V Zend Frameworku existuje třída `ZendX_JQuery`. Název knihovny, potažmo prefixu a složky `ZendX_` značí, na rozdíl od Zend_, že jde o komponenty, které nejsou v samotné distribuci *Zend Framework* - což vychází z licenčních podmínek. Třída `ZendX_JQuery` a její potomci napomáhají zapojení *JQuery* do aplikace. Zatím však ne v plné míře, a tak tato třída v projektu nebyla využita - javascriptový kód je klasicky umístěn ve zvláštním souboru.

Kromě *JQuery* však existují i některé další javascriptové knihovny a frameworky. Dají se rozdělit např. do následujících kategorií:

- knihovny pro manipulaci s DOM (**Dojo Toolkit**, **YUI Library**),
- GUI knihovny (**JQuery GUI**, **Ext JS**),
- grafické knihovny (**Three.js**, **Kinetic.js**),
- knihovny související s webovými MVC aplikacemi (**Google Web Toolkit**),
- knihovny pro AJAX,
- knihovny pro šablony a úpravu stylování (**ZURB Foundation**)
- a další...

Pro komplexní využití v databázi klášterů připadala v úvahu kromě JQuery jen knihovna **DOJO**. Velká výhoda této knihovny je v její velké podpoře Zend Frameworkem. Na rozdíl od JQuery je podpora s pomocí příslušných tříd s prefixem `Dojo_` zahrnutá přímo v základní distribuci frameworku (`Zend_`) a probíhá v této oblasti aktivní vývoj. Přesto pro mě možnosti knihovny JQuery a její rozšíření převážily tuto výhodu *DOJO*.

2.2 Možnosti tvorby mapy

2.2.1 Google Maps API

Vstupním požadavkem tohoto výběru bylo vytvořit javascriptovou mapovou aplikaci napojenou na databázi klášterů, přesto samostatnou a nabízející množství funkcí. Které konkrétně, mělo vyplynout, a skutečně vyplývalo až s vývojem databáze a utvářením požadavků zadavatelů.

V následujících odstavcích je představeno několik základních řešení která se nabízela. Výběr samozřejmě není úplný.

2.2.1.1 Mapy API V4

Základní vlastnosti:

- aktuálně verze 4.8 - Hanzelka a Zikmund (zajímavostí je pojmenování jednotlivých verzí podle známých cestovatelů),
- vlastní mapové podklady,
- není omezen počet požadavků za den [4],
- poměrně rozsáhlá nabídka funkcí, mj. značky, vizitky, shluky značek, vektorové prvky, AJAX, přepínání vrstev, zobrazení GPX, geokódování, plánování trasy, náhledové „minimapy“, přidávání vlastních vrstev včetně jiných projekcí a WMS ad.,
- na území ČR velmi kvalitní mapové podklady, zejména v turistické vrstvě,
- bohužel dostatečně kvalitní mapové podklady jsou k dispozici pouze pro území ČR.

Poslední informace je důležitá, neboť znemožňuje využití API od *Mapy.cz* pro projekt databáze klášterů - databáze totiž zahrnuje střední a jihovýchodní Evropu, včetně dnešního Maďarska, Rumunska, Srbska apod. Nemluvě o případném dalším rozšíření databáze i do dalších částí Evropy. To je velká škoda, protože nejen obsahově (Mapy.cz jsou skutečně na českém území podrobnější než *Google Maps*), ale také z kartografického hlediska (volba mapových značek, barvy, generalizace...) jsou v našich podmínkách kvalitnější než *Google Maps*. V *Google Maps API* je ovšem možné nastavit vlastní mapový klíč, o tom ale až v kapitole 4.2.2.

Zde by bylo ještě vhodné zmínit, že mapové podklady *Google Maps* není možné bezplatně využít v API od *Mapy.cz* [5], ale naopak, tedy mapové podklady *Mapy.cz*, samozřejmě se zobrazením loga „Mapy.cz“ a textem copyrightu použít lze [4].

2.2.1.2 aMapy.cz API

Další česká služba **aMapy.cz** (provozuje *centrum.cz*) do roku 2012 nabízela poměrně kvalitní a přehledné API rozhraní, bohužel také limitované hranicemi České republiky. V roce 2012 byly mapy nahrazeny novými mapami. API rozhraní však s touto změnou bylo ukončeno zejména proto, že vývoj samotný již dříve ustrnul. API mělo kvalitní dokumentaci, žádná omezení dotazů či možnost vložit vlastní mapové podklady, na druhou stranu například nenabízelo geokódování. Mapové podklady byly kvalitní, ale jak už bylo řečeno výše, jen na území České republiky. Samotný server aMapy.cz byl poté ukončen na konci roku 2013, doslova v průběhu psaní této práce.

2.2.1.3 Yahoo Maps API

Základní vlastnosti:

- 3 základní větve API:
  - **Flash API** - možné vyvíjet v *Javascriptu*, *ActionScriptu* nebo *Adobe Flex*,
  - **Ajax API** - varianta pro prohlížeče bez Flash pluginu - psáno v Javascriptu,
  - **Simple API** - založeno na XML formátu GeoRSS
- mapové podklady *NAVTEQ*,
- omezení na 50,000 dotazů denně,
- v současnosti licenční propojení s *Bing Maps* a *HERE Maps*.

Yahoo! Maps je mapový portál poskytovaný vyhledávačem *Yahoo!*. Ve srovnání s *Mapy.cz* a *Google Maps* v Česku ne tolik využívaný, přesto disponující mnoha funkcemi (kreslení linií a polygonů, přidání značek, infookna, informace o aktuálním provozu, atd.).

2.2.1.4 Bing Maps

Bing Maps (dříve také **Windows Live Maps**) je projekt od společnosti *Microsoft*.

Základní vlastnosti:

- několik základních služeb, hlavní je *Bing Maps AJAX V7*,
- množství funkcí, např. geocoding, reverzní geocoding, vyhledání trasy, několik vrstev (klasická mapa, letecká / družicová, ptačí pohled..), aktuální provoz na silnicích, přidání linií, polygonů... ,
- mapové podklady *NAVTEQ*,
- omezení na 50,000 dotazů na geocoding denně a 250 současně zobrazených bodů zájmu,
- služba je částečně komerční.

2.2.1.5 HERE Maps API

HERE Maps je mapová služba od Nokia, dříve známá jako **Ovi Maps** (do roku 2011), poté **Nokia Maps**. S touto službou souvisí také služba **MAP24**, která v roce 2011 zanikla a částečně přešla na *Nokia Maps*.

- zveřejněno v roce 2013,
- zejména pro Android vývojáře,
- několik stupňů a možností licence,
- poměrně široká nabídka funkcí: několik vrstev (street, terrain, satellite), značky, clustery značek, textové značky, polygony, polylinie, infookna, kontextová menu, vykreslení hustoty, KML objekty, vyhledání nejkratší trasy, geokódování, reverzní geokódování, Data API, animace objektů, minivýřez mapy, přidání vrstev...

2.2.1.6 MapQuest

- vlastní poskytovatelem *AOL*,
- MapQuest úzce spojeno s open-source online mapováním a GPS technologiemi,
- API obsahuje např. možnost přidání bodů, infooken, přidání linií, polygonů, elips, atd. nebo vlastních rastrů, možnost načtení kml a GeoRSS souborů, nastavení vlastního UI, geokódování, reverzní geokódování, vyhledání nejkratší trasy, podpora mobilní verze, výpočty vzdáleností, možnost zjištění vzdálenosti (v čase) i bez zobrazené mapy, atd.

2.2.1.7 OpenStreetMap

Známý projekt volně dostupných geografických map, která může kdokoliv přidávat a editovat. Pro samotné vykreslování mapy server používá aplikaci **OpenLayers**. *OpenLayers* je svojí filozofií značně odlišný od předchozích mapových API rozhraní. Díky licenci *FreeBSD* je možné si vytvořit kompletně vlastní aplikaci, implementace je ale o něco složitější. Hlavním důvodem, proč jsem v databázi dal přednost *Google Maps* oproti *OpenStreetMap*, bylo především to, že *OpenStreetMap* má stále ještě v některých místech nevyrovnanou kvalitu mapových dat. Na většině potřebného území však už kvalitou *Google Maps* dohání.

2.2.1.8 Google Maps API

Na konec jsem si nechal mapové API, pravděpodobně nejznámější, použité v této databázi. Mapy Google jsou nejpoužívanější online mapovou službou s více než 800 000 weby, které využívají API [6]. Tím se prostředí a mapový klíč těchto map stává důvěrně známé pro velké množství uživatelů.

API rozhraní v javascriptu je nejdůležitější službou *Google Maps* a plný název je **Go-**

ogle Maps Javascript API V3[1]. Aktuální verze je *3.14*, experimentální verzí je *3.15*. Další služby jsou např:

- **Google Directions API** - např. podrobný popis cesty (navigaci) z bodu A do bodu B,
- **Google Distance Matrix API** - nabízí informace o vzdálenosti (v km i časové) mezi těmito body,
- **Google Elevation API** - poskytuje informace o nadmořské výšce v daném bodě,
- **Google Geocoding API** - poskytuje převod adresy v textovém formátu na souřadnice, resp. naopak (v případě tzv. *reverse geocoding*) - tato služba je v aplikaci s kláštery využita,
- **Google Time Zone API** - nabízí převod času na základě souřadnic,
- **Google Places API** - poměrně rozsáhlé API, které nabízí přístup k datům *Google Maps*, a umožňuje vývojářům získat např. informace o nejbližších obchodech, firmách, ale také fotografiích apod.

Všechny tyto služby by měly být použity společně s *Google Maps Javascript API V3* a jsou založeny na poskytování dat v XML nebo JSON formátu.

Alternativou pro bezplatnou verzi API je **Google Maps API for Business** nabízející rozšířené služby. Některé základní rozdíly v omezení bezplatné verze a verze API pro firmy jsou shrnuty v následující tabulce:

Tabulka 2.1: Porovnání možností volné a placené verze Google Maps API[6]

Funkce	**Google Maps API**	**Google Maps API for Business**
Geocoding, Directions, Elevation API	2,500 požadavků / den	100,000 požadavků / den
Maximální rozlišení Static Maps API	640 × 640 px	2048 × 2048 px
Technická podpora	NE	ANO
Vrstva demografických dat	NE	ANO
Vkládání do zpoplatněného softwaru a aplikací	NE	ANO

Další související služby Googlu jsou např. **Google Static Maps API V2**. Tato služba na základě požadavku definovaném v URL s parametry vrací s jistým omezením

[1]Do nedávné doby (prosinec 2013) byla také funkční, byť zastaralá, starší verze *V2*. Ta byla však úplně vypnuta a nahrazena verzí *V3*, jejíž syntaxe je velmi podobná, ale ne totožná. Tedy některé aplikace psané ve verzi *V2* od poloviny prosince nemusely fungovat správně. Tato mapová aplikace s lokalitami je nyní plně funkční, tedy psaná ve verzi V3.

výřez mapy. Služba je v této aplikaci také využitá k exportu zadaných výřezů z mapy (viz 4.2.2.3). Služba **Street View Image API** je obdoba této služby aplikovaná na známé *StreetView* - vrací statické obrázky na daných souřadnicích, pod daným úhlem a s daným přiblížením.

Služba **Google Earth API** zase nabízí poměrně rozsáhlé množství funkcí spojených se známou aplikací *Google Earth*. Na jejich výčet zde není prostor.

Jako zastaralé, ale funkční jsou označeny také služby **Google Maps API for Flash** (API pro ActionScript) a **Google Local Search** (tato služba bude každým dnem vypnuta).

Samotné API rozhraní pro tvorbu interaktivní mapy nabízí množství běžných funkcí podobně jako API jiných poskytovatelů. Omezení spočívá u bezplatné verze mj. v počtu načtení map za 24 hodin - 25 000 [7] - což je pro potřeby této aplikace zdaleka dostatečné. Velká výhoda oproti službám od konkurence je *StreetView*. Ačkoliv i někteří další mapoví poskytovatelé podobný projekt nabízejí nebo v současnosti připravují (*Mapy.cz*), pokrytí dat Googlu je zdaleka nejkompletnější. Zde je výčet hlavních funkcí:

- události myši
- ovladače (měřítko, zoom, *StreetView*, typ mapy, ovladač pro otáčení vrstvy 45° pohledu, ad.)
- vlastní stylování mapy - barvy mapových značek, jejich vypnutí apod.
- značky bodů
- InfoWindow
- kreslení i editovatelných polylinií, polygonů, obdélníků, kruhů
- vlastní symboly vč. animací
- vlastní rastrové vrstvy
- další vrstvy + knihovny vrstev (počasí, teplota, provoz, hromadná doprava, cyklomapa, fotografie z databáze *Panoramio*)
- vlastní vrstvy načítané z *Fusion Tables*
- různé mapové typy (silniční, satelitní, hybridní, terénní, 45°), dále vlastní typy map v různých vlastních projekcích
- propojení s dalšími API (hledání trasy, nadm. výška, geokódování, reversní geokódování, ad..)
- další knihovny - *AdSense*, jednoduché kreslení, *Panoramio*, geometrické funkce, místa (obchody, apod., vč. Autocomplete), počasí...

K API existují i další, často open-source knihovny např. pro tvorbu popisů, clustery (skupiny značek, které klasické značky nahrazují při větším množství zobrazených), atd.

Jde především o knihovnu **google-maps-utility-library-v3** [8].

2.2.1.9 Další mapové služby

Kromě těchto mapových služeb nesmíme zapomenout na **Apple Maps**, avšak tato služba neposkytuje API, takže je pro databázi klášterů nepoužitelná. Navíc je obecně známo, že kvalitou dat zatím značně pokulhává za konkurencí (např. [9]).

Kapitola 3

Popis informačního systému

Tato kapitola obsahuje stručný popis funkcí databáze jako *informačního systému* - tak se jeví z pohledu registrovaného uživatele. Aplikace obsahuje relativně velké množství funkcí, formulářů, a tak cílem této kapitoly není popsat vše vyčerpávajícím způsobem. Ačkoliv registrovaných uživatelů v aplikaci nebude velký počet (nejvýše v řádu desítek), s čímž se od počátku počítalo, vzniká postupně zároveň textová nápověda. V určitý moment bude zapojena přímo do systému, aby byla stále při ruce. Přesná forma nápovědy je v tuto chvíli v jednání s CMS.

Uživatelské vrstvy: Systém má 4 různé uživatelské vrstvy, z nichž každá má jiné oprávnění. Navíc každý registrovaný uživatel může být v některé funkci hlavním administrátorem blokovaný a každý administrátor může mít oprávnění na jinou kategorii hesel, čímž se vrstvy a oprávnění ještě více komplikují. Tato kapitola stručně popisuje jednotlivé funkce podle uživatelských vrstev, a to od návštěvníka stránek po hlavního administrátora:

1. neregistrovaný uživatel, návštěvník stránek
2. registrovaný uživatel
3. administrátor
4. hlavní administrátor

Stavy hesla: Každé heslo může dosáhnout 2 - 4 různých stavů:

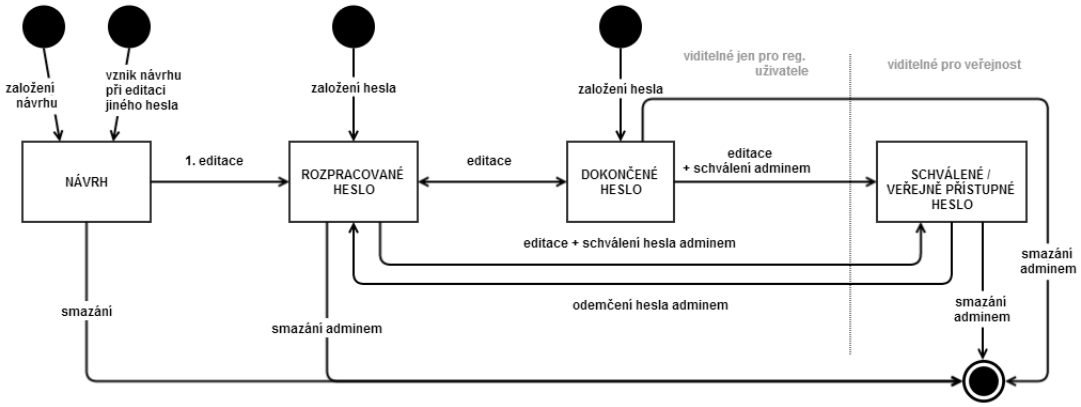
1. návrh
2. rozpracované heslo
3. dokončené heslo
4. schválené / veřejně přístupné heslo

Každé heslo musí projít (pokud není předčasně smazáno) nejméně dvěma stavy - 2.

nebo 3. stavem a poté 4. stavem. Rozdíl mezi stavem „rozpracováno“ a „dokončeno“ je pouze symbolický - dokončené heslo je informace pro administrátora, že je dané heslo připraveno ke schválení. Návrh zase znamená cosi jako „toto heslo by mělo být zpracováno“ - obsahuje pouze základní informace (typ / druh hesla, název, příp. poznámku) a do 1. úpravy jej může zpracovat jakýkoliv registrovaný uživatel. Právě první úpravou návrhu přejde heslo do 2. stavu a stane se odemčeným pouze pro jednoho daného uživatele. Návrh kromě toho může vzniknout také při tvorbě / editaci jiného hesla, jako „vedlejší produkt“. Např. pokud je při úpravě osobnosti přiřazena tato osobnost jako představená k určité lokalitě, ale daná lokalita dosud v databázi není. V takovém případě se lokalita vytvoří, ale jako návrh. Příslušný administrátor může heslo (na jehož kategorii a typ má příslušné právo) nejen schválit, ale také kdykoliv také smazat.

Pokud příslušný administrátor heslo schválí, zpřístupní jej tím pro veřejnost, tedy 1. vrstvu uživatelů. V tu chvíli je heslo pro úpravy úplně uzamčeno, upravit jej nemůže ani hlavní administrátor. Heslo je ale i poté kdykoliv možné znovu odemknout administrátorem - ale vždy jen pro jednoho uživatele.

Na první pohled krajní situací, která ve skutečnosti ale nebude výjimkou, je celý proces hesla v režii jednoho uživatele/administrátora - administrátor má samozřejmě práva jako běžný uživatel, a tak může mít právo i na schválení svého hesla, které sám založil a zpracoval. Stavy hesla by měl lépe popsat upravený stavový diagram:



Obrázek 3.1: Diagram stavů hesla

Typy hesel: Jak už bylo řečeno, hesla se dělí na 3 základní kategorie, tyto kategorie se dále dělí na různé typy. Na typech hesel závisí některé typy propojení (viz níže), některá hesla dále neevidují některé informace (např. typ lokality *sbírková instituce* eviduje jen základní údaje) a na typy hesel jsou navázaná i schvalovací práva administrátorů. U osobností se místo termínu „typ“ užívá termín „druh“ a osobnost, na rozdíl od lokality či artefaktu může být i více druhů zároveň. Lze tedy tvrdit, že jednotlivé typy jsou poměrně zásadní údaj. Zde je přehled všech typů hesel:

- **Lokality**
  - klášter (eviduje se navíc typ kláštera)
  - proboštství
  - inkorporovaná nebo patronátní fara
  - biskupství
  - sbírková instituce (evidují se jen základní údaje)
  - instituce jiných řádů
- **Artefakty**
  - architektonická skulptura
  - architektonický článek
  - architektura stojící
  - architektura zaniklá
  - diplomatický pramen
  - historiografický pramen
  - hrobový nález
  - hudba
  - jiný text
  - kodex
  - krajinné prvky
  - legenda
  - malba knižní
  - malba nástěnná
  - memoriální prameny (nekrologia aj.)
  - normativní prameny
  - pečeť
  - pohřebiště
  - relikvie
  - socha volná
  - správní a hospodářský pramen
  - textil
  - užité umění

- **Osobnosti**
  - biskup
  - mnich
  - panovník
  - představený
  - jiná osobnost
  - nezařazeno

Typy propojení mezi hesly: Hesla v systému mohou být různými způsoby propojena. Zde je výčet a pravidla všech propojení:

- **propojení typu „související“**; může být mezi kterýmikoli hesly, nezávisle na jejich typu / druhu
- **představení+klášter**; představený je osobnost (např. opat), a platí, že osobnost MUSÍ být druhu *představený*, pokud je jako představený napojena u některé lokality, stejně tak naopak. Tedy pokud je osobnost napojena na nějakou lokalitu jako představená, MUSÍ být zároveň druhu *představený*. Tedy např. připojením osobnosti k lokalitě jako představené je u osobnosti automaticky nastaven také druh *představený*. Na straně lokality platí, že představený může být napojen k typu *klášter*, *proboštství* a *instituce jiných řádů*. U napojení je povinně evidováno pořadí představeného, a navíc můžou být evidovány roky *od* kdy *do* kdy představený v lokalitě působil. Představených tedy bývá u jedné lokality více, dokonce se jedna osobnost může u jedné lokality opakovat. Stejně tak osobnost může být vedena jako představená u více lokalit.
- **fundátorství**; Fundátorem může být jedna nebo více osobností u libovolné lokality (kromě typu *sbírková instituce*, *inkorporovaná nebo patronátní fara* a *biskupství*).
- **filiační vazby**; lokality typu *klášter*, *proboštství* a *instituce jiných řádů* mohou mezi sebou být spojeny filiačními vazbami. V každém spojení musí být jedna lokalita mateřská, druhá dceřiná.
- **dnešní uložení artefaktu**; artefakt libovolného typu může být uložen v jedné lokalitě typu *sbírková instituce*.

Přihlášení, automatické odhlášení: Pokud přihlášený uživatel neprovedl žádnou změnu na serveru během 2 hodin, je automaticky odhlášen a stránka přesměrována na přihlašovací. 5 minut před vypršením této lhůty se zobrazí varovné vyskakovací okno s upozorněním a uživatel má možnost přihlášení prodloužit, případně zrušit. Odhlášení tedy proběhne na serveru, ale i na straně klienta (v prohlížeči) se provede přesměrování na přihlašovací stránku.

3.1 Informační systém z pohledu návštěvníka stránek

Tato podkapitola popisuje aplikaci z pohledu návštěvníka stránek, který není registrovaný. Protože je skutečná databáze nyní ještě ve vývoji a obsahuje zatím velmi málo dat, nepočítá se v tuto chvíli s přístupem „náhodných“ návštěvníků stránek. Navíc nyní databáze ještě nemá vlastní doménu. Některé stránky a části aplikace jsou tak dočasně pro návštěvníka vypnuty. Týká se to především detailů hesel.

3.1.1 Hlavní strana

Pokud se návštěvník stránek dostane na hlavní stranu aplikace, uvidí především „uvítací“ text. Zda bude pevný jako doposud, nebo výpis novinek formou blogu, je zatím v jednání. Všechny podstránky aplikace (kromě hlavní mapy) mají ale stejný layout, který si můžeme rozdělit na několik částí:

- samotný obsah stránky
- uživatelský panel (úplně nahoře)
- hlavička (v tuto chvíli obsahuje pouze fotografii a nadpis databáze, slouží zároveň jako odkaz na hlavní stranu)
- menu, podmenu
- postranní pravý panel
- patička

3.1.2 Postranní - pravý panel

Pro návštěvníka stránek je v pravém panelu viditelný pouze **oddíl rychlého hledání**.

3.1.2.1 Rychlé hledání

Jde o jednoduchý formulář, který nabízí v první řadě volbu, zda hledat podle názvu, nebo podle ID hesla (toto má význam spíše pro registrované uživatele). V prvním případě se hledá mezi všemi kategoriemi hesel a systém bere v úvahu jak hlavní název, tak vedlejší názvy hesla. Výsledek hledání se zobrazí v tabulce, ve které je kromě názvu hesla i kategorie a typ / druh. Tabulku lze řadit podle všech parametrů (má význam v případě většího množství výsledků). Pokud chce uživatel řadit podle více kritérií (např. nejprve podle kategorie, poté podle názvu), stačí kliknout na hlavičky daných sloupců a zároveň držet tlačítko SHIFT. Názvy hesel současně slouží jako odkazy na jejich detail. Ve druhém případě (hledání podle ID) a po volbě základní kategorie se zobrazí přímo detail hesla, pokud tedy heslo s takovým ID existuje.

Připomeňme, že v současné době návštěvníci nemají přístup k detailu hesla, a tak systém v takovém případě provede přesměrování na přihlašovací stránku.

3.1.3 Uživatelský panel

V odhlášeném stavu je v uživatelském panelu pouze odkaz na přihlašovací stránku.

3.1.4 Hlavní menu

Hlavní menu je dvouúrovňové, přičemž odpovídající druhá úroveň se zobrazí až po kliknutí na položku první úrovně - což byl přímo požadavek CMS. Tedy ne box, který by se objevil po najetí myši.

3.1.4.1 Hledat - lokality

Hledání lokalit podle parametrů nabízí vyhledání ve všech lokalitách v databázi, pokud známe jen určité údaje. Výrazným pomocníkem je filtrování. V praxi to znamená, že po každém ve formuláři nově zadaném omezení se prohlížeč spojí s databází (viz kap. 4.1.3.8) a následně omezí zbývající parametry tak, aby bylo možné vybrat už jen ta omezení, ze kterých systém vyhledá alespoň jednu lokalitu. Navíc určité parametry nejsou vůbec evidovány u některých typů lokality (např. u *sbírkových institucí*) se evidují jen základní informace, a tak se formulář průběžně přizpůsobuje. Nad formulářem je také odkaz na formulář fulltextového vyhledávání (viz kap. 3.1.4.4). V lokalitách je možné vyhledávat podle těchto parametrů:

- název lokality
- typ lokality
- příp. typ kláštera
- stav lokality
- první zmínka
- založení
- světec
- fundátor
- představený
- související hesla

U některých parametrů, např. rok založení, je navíc možné nastavit, zda se mají do vyhledaných lokalit započítat i ty, u kterých údaj není vůbec zadán. Po odeslání formuláře se zobrazí jednoduchá tabulka obsahující vyhledané lokality a jejich typ. V případě nepřihlášeného uživatele se zobrazí jen lokality, které jsou schválené.

3.1.4.2 Hledat - artefakty

Vyhledání artefaktů funguje na podobném principu jako vyhledání lokalit. Vyhledávat je možné podle těchto parametrů:

- název artefaktu
- typ artefaktu
- vznik
- datace
- současné uložení
- související hesla

3.1.4.3 Hledat - osobnosti

I hledání osobností podle parametrů funguje stejně. Zde se vyhledání provádí podle těchto parametrů:

- jméno osobnosti
- druh osobnosti
- narození
- úmrtí
- fundace
- příp. působení představeného
- souvisejí hesla

3.1.4.4 Hledat - fulltext

Formulář pro hledání fulltextu je velmi jednoduchý, obsahuje pouze textové pole pro vzor a odesílací tlačítko. Systém prohledává hlavní i vedlejší názvy a hlavní texty všech hesel - tedy u lokalit jde o historii a stavební vývoj, u artefaktů popis, rozměry a u osobností biografie. Hledání je společné pro všechny kategorie a výsledky jsou tak v jedné tabulce, kterou je opět možné řadit. Navíc se zde počítá i s větším počtem výsledků, a tak je tabulka stránkovaná a je možné zvolit počet hesel na jedné stránce. V této tabulce je kromě názvu, kategorie a typu hesla uvedeno i procento správnosti výsledku, tedy jak moc odpovídá zadanému vzoru.

3.1.4.5 Přehledy - Výpis hesel

Tato stránka zobrazuje přehledně v jedné tabulce všechna hesla, která jsou schválená a tedy viditelná pro veřejnost. Tabulku je opět možné pro větší přehlednost řadit dle všech parametrů, případně stránkovat. V tabulce jsou zobrazeny u každého hesla tyto údaje:

- název / jméno
- kategorie (lokalita, artefakt, osobnost)
- typ / druh
- uživatel, který heslo založil
- datum, kdy bylo heslo založeno
- uživatel, který heslo schválil
- datum, kdy bylo heslo schváleno

3.1.4.6 Mapa

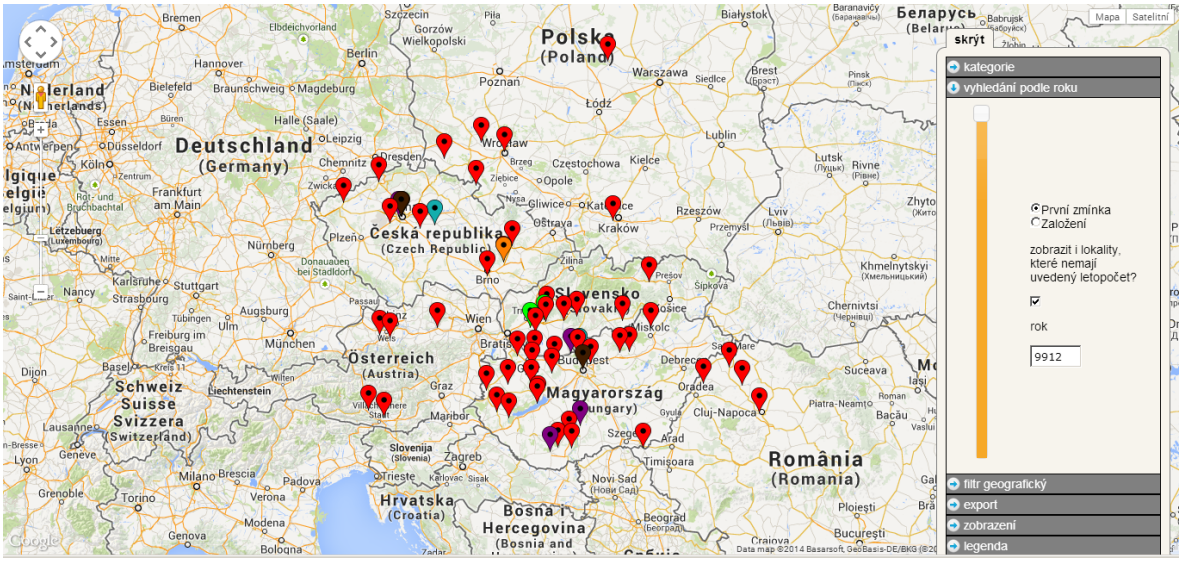
Mapa, která se zobrazí po kliknutí na příslušnou položku v menu, je velikostí i možnostmi značně omezená, ale pravděpodobně ještě dozná změn. I tak umožňuje filtrování zobrazených lokalit podle jejich typu. Kliknutím na jednotlivá zaškrtávací pole se vyberou, případně vyřadí typy, které chce uživatel zobrazit. Pokud chce uživatel rychle zobrazit jen jednu kategorii, stačí kliknout jen na text *zobrazit*. Možnost *zobraz vše* není třeba rozebírat. Po každé změně se u mapy nastaví střed a zoom tak, aby mapa akorát pokryla zobrazené lokality. Zároveň se vypisuje počet zobrazených lokalit. Zajímavější mapa se ale objeví po kliknutí na odkaz *zobrazit mapu v novém okně*.

3.1.4.7 Mapa - zobrazení ve velkém okně

Tato mapa má vlastní layout, tedy žádné menu ani postranní panely. Plocha mapy zabírá celý prohlížeč, jen v pravé části je menu, které lze kdykoliv dočasně skrýt a znovu zobrazit. Skrytí probíhá kliknutím na příslušný text a je užitečné zejména při větším počtu lokalit - žádnou tak nezakrývá. U mapy jsou ponechány všechny základní ovladače - tedy zoom, navigace, přepínání mezi vrstvami map (základní, fotomapa) a StreetView. Mapa samotná je mírně upravena - je skrytá vrstva čísel silnic a dálnic. Na mapě s kláštery tento údaj působil spíše rušivě.

Položek v menu je aktuálně 7, tento počet se však pravděpodobně bude ještě měnit. První položkou je **filtr dle typů lokalit**, který byl popsán již u jednoduché verze mapy (kap. 3.1.4.6).

Druhou položkou je **filtr dle roku**. Oknu vévodí svislý posuvník, jehož posunem se mění letopočet v textovém okně a zároveň se lokality v mapě zobrazují a skrývají. Vždy platí, že jsou zobrazeny lokality, které již v daném roce byly založeny, tedy které mají



Obrázek 3.2: Náhled mapy s lokalitami

příslušný rok stejný nebo nižší než je zadán. Lze samozřejmě upravit i přímo rok v textovém poli. Přepínacími tlačítky je možné filtrovat dle roku první zmínky nebo podle roku založení. Protože je pravděpodobné, že část lokalit nebude tyto údaje obsahovat, je možnost takové lokality skrýt / zobrazit.

Geografický filtr funguje na podobném principu jako *filtr dle typů lokalit*. U lokalit se v databázi ukládá i stát, administrativní celek, ve kterém leží i obec, a tak není problém např. zobrazit lokality jen na území Maďarska.

Export mapy je funkce umožňující potřebný výřez „tisknout“ do rastrového formátu (PNG). Díky jistým omezením ze strany Googlu (viz kap. 4.2.2.3) je ale proces komplikovanější. Navíc tato funkce v tuto chvíli není plně dokončena. Kliknutím na odkaz *nastavit výřez* se zobrazí editovatelný obdélník zabírající velkou část mapy. Ten je možné táhnutím stran či rohů nastavit na požadovaný výřez. Pokud je však žádoucí exportovat plochu celého aktuálně zobrazeného okna, klikne uživatel na *celé okno*, čímž se obdélník objeví v celé ploše okna, tentokrát needitovatelný. Kliknutím na *exportovat mapu* se zobrazí odkaz na daný výřez mapy, který lze jednoduše stáhnout. Ovšem stále bez popisů. Zmiňované omezení spočívá především v maximální možné velikosti jednoho výřezu, a tak pokud je výřez větší, výřezů je více. Druhé omezení spočívá v nemožnosti zpracovat mapu na serveru. Finální řešení se tedy předpokládá takto:

1. uživatel nastaví zobrazení popisů, mapové značky lokalit ap. (viz níže) - vše, co požaduje v exportované mapě
2. zároveň nastaví, zda chce do mapy přidat titulek, měřítko a legendu (příp. kam)
3. uživatel vybere požadovaný výřez (viz výše) - bere se v potaz také měřítko, maximální povolená plocha je tedy velikost mapy, resp. prohlížeče.
4. uživatel kliknutím na příslušné tlačítko vytvoří odkaz / odkazy na stažení statické mapy
5. vygenerovaný statický obrázek / obrázky (zatím bez popisu ap.) stáhne běžným způsobem (pravé tlačítko myši)
6. obrázky nahraje na server - elementy pro vložení souboru budou přehledně vedle odkazů na statické mapy
7. po odeslání server již do těchto obrázků přidá popisy, legendu a požadované prvky, příp. více obrázků spojí do jednoho, a vzápětí nabídne odkaz ke stažení hotového výřezu mapy. Výřez může být na serveru určitou dobu uložen (např. den) a po tuto dobu by měl být uživatelem znovu stáhnutelný.

V sekci **zobrazení** je možné skrýt nebo zobrazit popis lokalit (užitečné zejména pro export mapy) a nastavit sadu značek pro označení lokalit v mapě.

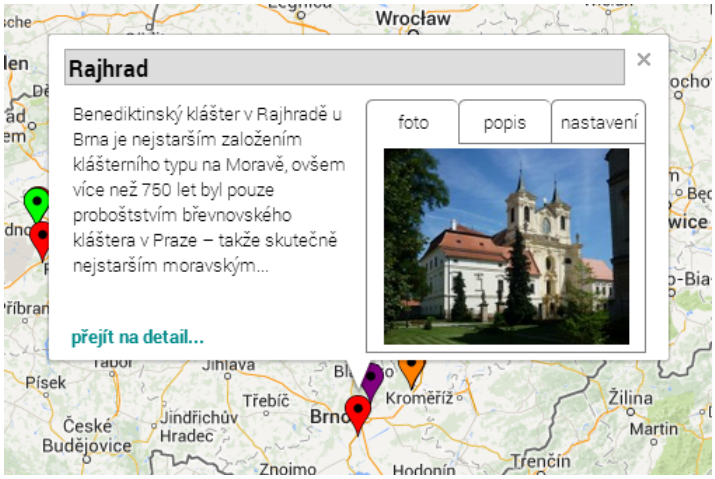
Sekce **legenda** poskytuje mapový klíč typů lokalit - pokud se změní sada značek (viz výše), změní se tím i legenda.

V sekci **statistiky** se v současné době zobrazuje pouze počet zobrazených lokalit, nabízí se ale zobrazení množství jiných zajímavých informací.

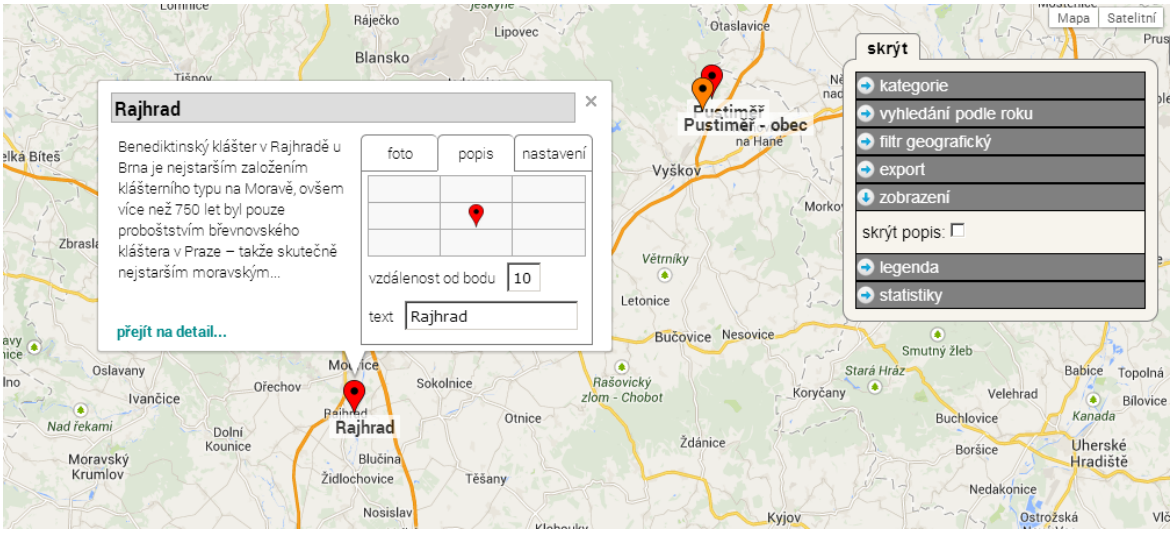
Pokud uživatel klikne na značku dané lokality, zobrazí se informační okno. To obsahuje název lokality a v levé části zkrácený (prvních 35 slov) hlavní text. Pod ním potom odkaz na detail lokality. V pravé části se nacházejí 3 záložky. Na té **první** je titulní fotografie lokality, pokud existuje.

Na **druhé** záložce se nachází nastavení popisu lokality. Nejprve je samozřejmě potřeba popis u lokalit zobrazit (v menu mapy), poté je s ním možné manipulovat. To se může hodit např. v situaci, kdy automaticky generovaný popis zakrývá jinou lokalitu, nebo se popisy mezi sebou překrývají. Je možné tímto způsobem zajistit čitelnost mapy podle základních pravidel kartografie. Základních poloh popisu je 8, odpovídají světovým stranám. Devátou možností je ponechat původní polohu. Přepnutí na danou polohu se provede jednoduchým kliknutím na příslušné pole. Také lze jednoduše nastavit vzdálenost popisu od bodu lokality v pixelech, či změnit samotný text popisu. To může být výhodné, pokud je pro potřeby výřezu mapy příliš dlouhý.

Záložka **nastavení** potom bude nabízet další možnosti zobrazení, v tuto chvíli je volná.



Obrázek 3.3: InfoWindow lokality

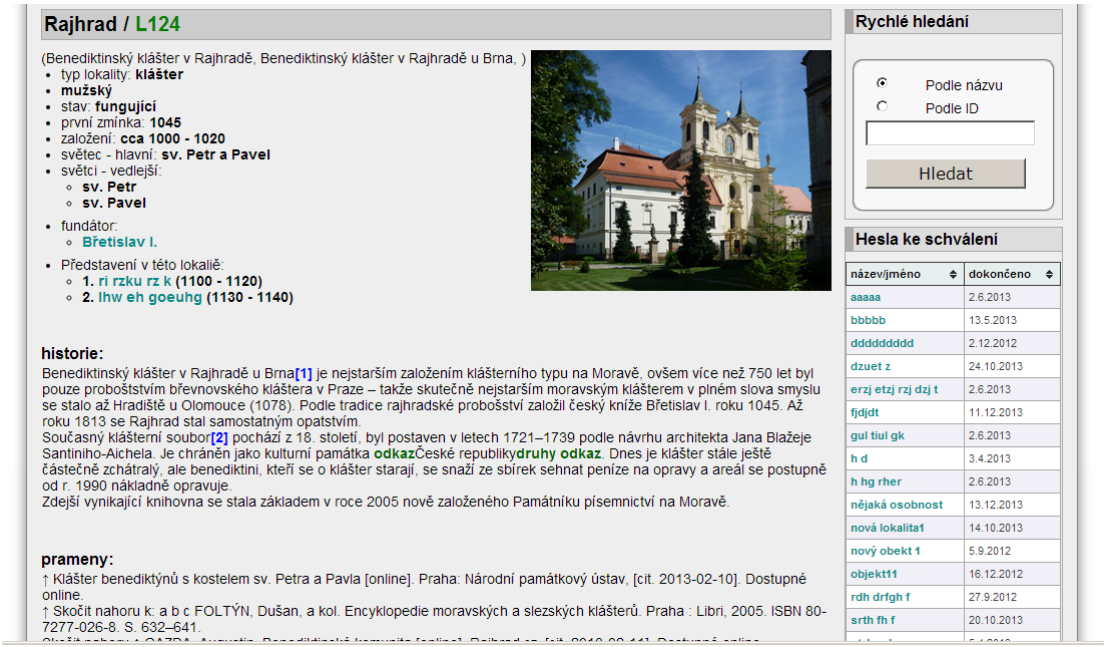


Obrázek 3.4: Popis lokality a jeho nastavení

3.1.5 Detail hesla

Jde prakticky o nejdůležitější stránky databáze. Nabízí se srovnání se stránkou s libovolným článkem na Wikipedii. Obsah stránky lokality, artefaktu a osobnosti se mírně liší, základ ale zůstává stejný. Pod názvem hesla je bodový výčet základních údajů. Nejprve *další názvy*, poté *typ* lokality / artefaktu, resp. *druhy* osobnosti (v případě osobností může druhů být i více), příp. *typ kláštera* a *stav lokality*, dále letopočty. V případě lokality se jedná o *první zmínku* a *založení*, v případě artefaktu *vznik* a *dataci* a v případě osobnosti *narození*, *úmrtí* a *první zmínku*. Ve všech případech nemusí být uveden přesný rok, ale přibližné rozmezí. U lokality je dále uveden *světec hlavní*, příp. výčet *světců vedlejších*, *fundátor* a přehled *představených* vč. roků působení - slouží zároveň jako odkazy na osobnosti. V případě osobnosti jsou naopak uvedeny lokality / *kláštery*, ve kterých případný představený působil, nebo které osobnost zakládala. U osobnosti a artefaktu je zde uvedeno i *místo* a *stát původu*, v případě artefaktu potom lokalita (*sbírková instituce*),

ve které je artefakt uložen. V pravé části se potom nachází *fotografie k heslu*. Pokud jich je u hesla evidováno více, tak ta, která je nastavena jako titulní.



Obrázek 3.5: Detail lokality

Následují hlavní texty. V případě lokality jde o *historii* a *stavební vývoj* (jen u některých typů), v případě artefaktu *popis* a *rozměry*, v případě osobnosti *biografie*. V textech mohou být odkazy. Ty interní - tedy na jiná hesla v databázi - jsou zvýrazněny tmavě zeleně, externí světle zeleně. Pokud text obsahuje poznámky pod čarou, jsou propojeny odkazy podobně jako na Wikipedii. Seznam poznámek je pod texty. Následují výčty souvisejících lokalit, artefaktů a osobností, fungující jako odkazy. Pod výčty jsou externí odkazy a v případě lokality mateřské a dceřiné kláštery. Následují všechny obrázky, které jsou k heslu přiřazené. Stejně jako titulní obrázek v horní části stránky, fungují zároveň jako odkazy na stránku s daným obrázkem (viz kap. 3.1.5.1). Pokud jsou k heslu přiřazeny nějaké soubory (PDF), jsou pod obrázky vypsány. Kliknutím na *zobrazit detail* u jednotlivých souborů se rozbalí podrobnější informace (formát, velikost, datum nahrání, kdo soubor nahrál a stručný popis).

Posledním bodem v detailu hesla pro neregistrovaného uživatele je v případě lokalit zobrazení její polohy na interaktivní mapě.

3.1.5.1 Detail obrázku

Pokud návštěvník klikne na některý obrázek, zobrazí se na zvláštní stránce. Ta obsahuje název obrázku, jeho větší náhled - po kliknutí na obrázek se zobrazí v plné velikosti v novém okně - dále rozměry původního obrázku i zobrazeného náhledu, velikost v kB (po najetí myši přesná velikost v bajtech), formát a hesla, ke kterým je přiřazen. V případě více hesel totiž není potřeba tentýž soubor nahrávat několikrát, stačí jej jen přiřadit (opět podobnost s Wikipedií, resp. *WikiMedia Commons*).

3.2 Informační systém z pohledu běžného registrovaného uživatele

Po přihlášení systém uživatele přesměruje na hlavní stranu, která je ale jiná, než v případě návštěvníka stránek.

3.2.1 Hlavní strana

Hlavní stránka je jakýsi blog článků, které slouží k hlavní komunikaci mezi uživateli a k řešení interních a organizačních záležitostí. Na hlavní straně je zobrazeno 10 posledních článků, další jsou uloženy v archivu. Stránky archivu jsou přístupné odkazem z hlavní strany. Pokud se uživatel do archivu dostane, má možnost listovat mezi stránkami archivu, nebo vyhledat články v požadovaném rozmezí podle data. Výběr data (od / do) je jednoduchý - stačí kliknout na příslušný den v kalendáři, přičemž pokud např. uživatel zadá určité datum „od“, datum „do“ už mu systém nedovolí vybrat dřívější než „od“.

Zpět na hlavní stranu se uživatel dostane:

- kliknutím na položku *Home* v menu
- kliknutím na hlavičku stránky (v současné době s leteckou fotografií Sázavského kláštera)

Pokud chce uživatel sám napsat nový článek, klikne na odkaz **Nový článek** na hlavní straně. Zobrazí se formulář se dvěma poli - titulkem a textem. Článek se uloží kliknutím na odesílací tlačítko.

Obecně ve všech formulářích v aplikaci platí, že červeně zvýrazněné údaje jsou povinné. Pokud zadávané údaje nejsou v pořádku, nebo chybí, systém uživatele upozorní ještě před odesláním formuláře - většinou červeným zvýrazněním problematického elementu a červeno-žlutým textem v horní části stránky.

Delší články na hlavní straně jsou vždy kvůli přehlednosti zkráceny. Pravidla pro krácení jsou následující:

- Pokud je v textu článku znakem dvojité roury (svislítka - ||) označeno místo pro zkrácení textu, a toto místo leží do 1500 znaku, je text článku zkrácený v označeném místě a celý je viditelný až po rozkliknutí článku.
- Pokud je znak dvojité roury dále než 1500 znaků od počátku, nebo není v textu vůbec, je text uměle zkrácen po 500. znaku a celém slově.
- Pokud je text kratší než 500 znaků a neobsahuje znak dvojité roury, není krácen vůbec.

Zde je příklad textu článku, jak je vidět v poli formuláře, pokud chceme na hlavní straně zobrazit jen první větu:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit.|| Vestibulum
```

```
faucibus accumsan fringilla. Maecenas faucibus odio sed accumsan vulputate.
Donec adipiscing turpis ut dui cursus mattis. Donec volutpat in arcu a
fermentum. Morbi ut fermentum nisi, sit amet vulputate risus.
```

V přehledu článku na hlavní straně je kromě zmiňovaného textu a titulku viditelný ještě autor článku, datum přidání článku a počet příspěvků v diskuzi. Po jeho rozkliknutí (ať už odkazem na diskuzi, odkazem v titulku nebo odkazem „více...“) se článek zobrazí v plné délce a pod ním diskuze s jednoduchým formulářem pro přidání diskuzního příspěvku.

3.2.1.1 Syntaxe psaní odkazů

V článcích, delších textech hesel, diskuzích k heslům a k článkům a ve zprávách lze psát odkazy na jiné stránky. HTML tagy jsou u textových polí v těchto případech zakázaná, a tak má vkládání odkazů v textu vlastní syntaxi. Odkazy se dělí na interní (určité heslo) a externí (např. stránky farnosti. kláštera, apod.).

Interní odkazy: Tento typ odkazu je možné do textu vložit dvěma způsoby.

První je např. tento:

```
[text odkazu|L124]
```

`text odkazu` je text, který se v textu zobrazí jako odkaz, `L124` je ID hesla, v tomto případě určité lokality (`L` symbolizuje lokalitu, `B` artefakt, `S` osobnost). Toto ID je v případě, že je uživatel přihlášen, viditelné hned za titulkem hesla na stránce daného hesla. Důležité jsou hranaté závorky a symbol tzv. roury neboli svislítka. Tento způsob zadání odkazu je *doporučený*.

2. způsob zadání interního odkazu je např. tento:

```
[text odkazu|Rajhrad]
```

Tedy místo ID je zde přímo název hesla. Toto funguje, ale je doporučeno využívat první způsob, mj. proto, že pokud se např. změní název hesla, na které se odkazuje, odkaz se stane nefunkčním. Naopak ID zůstává stejné.

Externí odkazy: Princip externího odkazu je podobný:

```
[text odkazu|http://www.klaster-sazava.cz/]
```

Princip by měl být jasný - ID hesla zde nahradí samotné url odkazu. Je nutné nezapomenout na protokol (zde „http://“). Pokud je jakýkoliv odkaz syntakticky zadán špatně, zvýrazní se v zobrazeném textu červeno-žlutě a po najetí myši se zobrazí informace, že je odkaz zadán špatně.

3.2.2 Postranní - pravý panel

V pravém panelu registrovaný uživatel kromě formuláře rychlého hledání vidí také tzv. aktuality všech uživatelů. Jde o rychlou krátkou zprávu, kterou má možnost jakýkoliv uživatel zobrazit v pravé liště - vždy jen jednu, kterou přepisuje, příp. maže. Zprávy by měly být typu : „do konce měsíce nejsem na příjmu“, nebo „prosím o zpracování hesla o Sázavském klášteru“. Aktuality jsou řazeny od nejnovější, zobrazeno je jich nejvíce deset, a u každé je mimo autora i datum.

Krátký formulář s aktualitou je pod aktualitami ostatních uživatelů.

3.2.3 Uživatelský panel

Uživatelský panel (úplně nahoře) je zaplněn různými odkazy, pokud je uživatel přihlášen.

Prvním údajem je uživatelské jméno právě přihlášeného uživatele, které zároveň funguje jako odkaz na „profil“ uživatele (viz kap. 3.2.4.4). Následuje role uživatele (běžný uživatel, administrátor, hlavní administrátor). Dalším odkazem (*změnit uživatele*), se zobrazí rovnou přihlašovací formulář. Kliknutím na zprávy se přímo v aktuální stránce (pod menu) zobrazí posledních 10 přijatých a odeslaných zpráv (kap. 3.2.3.1).

3.2.3.1 Interní zprávy mezi uživateli

Systém nabízí jednoduchý komunikační prostředek mezi registrovanými uživateli - posílání zpráv. Kliknutím na *poslat zprávu* v uživatelském panelu se zobrazí formulář pro odeslání zprávy. Uživatel musí v každém případě vybrat jednoho příjemce, dále vyplnit předmět zprávy nebo text. Doporučené je samozřejmě vyplnit obě textová pole. Po odeslání se zpráva objeví v odeslaných, pokud uživatel klikne v už. panelu na *zprávy*. Dokud si adresát zprávu nepřečetl, je modře zvýrazněna. Naopak, pokud má uživatel nějakou příchozí zprávu nepřečtenou, již v už. poli je nápis červeně zvýrazněn a v závorce zobrazen počet nepřečtených zpráv, např. „zprávy(1)“. Nepřečtené zprávy jsou poté ve výpisu také červeně zvýrazněny. V přehledu zpráv je u každé datum a čas odeslání, uživatel-odesílatel, resp. uživatel-příjemce, předmět zprávy a začátek textu samotné zprávy. Tedy podobně jako v případě mnoha e-mailových klientů. Kliknutím na datum, předmět nebo text zprávu otevřeme a tím je označena za přečtenou.

Na zprávu je možné přímo odpovědět, v takovém případě se pod jejím textem zobrazí formulář pro odpověď s chybějícím výběrem odesílatele - ten už je daný.

3.2.3.2 Uživatelská upozornění

Protože má každý uživatel v systému na starost určité množství hesel a zároveň by měl např. sledovat hlavní nástěnku, bylo nutné vytvořit systém upozornění. Níže je výčet - uživatel *Josef*, resp. administrátor *Jan* dostanou upozornění pokaždé, když dojde k jedné z následujících situací:

- došlo k editaci hesla, (příp. zpracování návrhu, viz kap. 3.2.6), které založil *Josef*, nebo které je pro uživatele *Josef* odemčeno, jiným uživatelem

- administrátor schválil heslo, které *Josef* založil, nebo které je pro uživatele *Josef* odemčeno
- administrátor odemkl již uzamčené heslo uživateli *Josef* pro úpravy
- nějaký uživatel editoval a zároveň dokončil - tedy označil jako dokončené - heslo, na které má administrátor *Jan* schvalovací právo
- administrátor smazal heslo, které *Josef* založil, nebo které je pro uživatele *Josef* odemčeno
- administrátor předal určité odemknuté heslo od jiného uživatele uživateli *Josef*
- administrátor předal určité odemknuté heslo od uživatele *Josef* jinému uživateli
- došlo ke změně typu / druhu hesla [1], které založil *Josef*, nebo které je pro uživatele *Josef* odemčeno, jiným uživatelem
- jiný uživatel přidal příspěvek do diskuze k heslu, které založil *Josef*, nebo které je pro uživatele *Josef* odemčeno, nebo pokud uživatel *Josef* již u hesla diskutoval, nebo pokud je uživatel *Josef* hlavním administrátorem - hlavní administrátor je totiž informován o všech příspěvcích
- došlo k odemknutí již uzamčené osobnosti kvůli odpojení představeného od lokality (ze strany lokality) - osobnost uživatel *Josef* zakládal, nebo pro uživatele *Josef* byla odemčená
- došlo ke změně druhu osobnosti vlivem odpojení od lokality jako působiště představeného[2].
- hlavní administrátor upravil údaje o uživateli *Josef*
- hlavní administrátor nastavil určitý typ blokování uživateli *Josef* (viz kap. 3.4.4)
- hlavní administrátor zrušil určitý typ blokování uživateli *Josef* (viz kap. 3.4.4)
- hlavní administrátor nastavil uživateli *Josef* jinou roli (např. roli administrátora)
- hlavní administrátor upravil administrátorovi *Jan* práva na schvalování určitých typů nebo druhů hesel
- v diskuzi nějaký uživatel přidal příspěvek ke článku, který uživatel *Josef* psal, nebo ke kterému již uživatel *Josef* přidal diskuzní příspěvek
- libovolný uživatel kromě uživatele *Josef* přidal na nástěnku nový článek

Posledních 20 upozornění se na jakékoliv stránce objeví pod menu (podobně jako zprávy) po kliknutí na *upozornění* v uživatelském panelu. U každého je uvedeno datum, čas a stručně shrnuta událost.

Poslední položkou v uživatelské liště je odkaz na odhlášení uživatele ze systému.

[1] toto není považováno za běžnou úpravu hesla, protože s typem je spojeno právo schvalování

[2] pokud je osobnost druhu představený, MUSÍ mít zároveň přidáno nejméně jedno působiště

3.2.4 Menu

3.2.4.1 Uživatel - Seznam uživatelů

Stránka nabízí jednoduchý přehled všech uživatelů v jedné tabulce. Uvedeno je už. jméno, celé jméno, role a datum posledního přihlášení do systému. Tabulku lze opět řadit a odkazem se uživatel dostane na detail příslušného uživatele (viz kap. 3.2.4.4).

3.2.4.2 Uživatel - Editace osobních údajů

Na této stránce se nachází jednoduchý formulář pro úpravu vlastních údajů. Běžný uživatel má možnost nastavit / upravit pouze email a stručné informace o sobě. Email je povinný a musí jej mít zadaný každý uživatel. Pod formulářem je potom odkaz na stránku pro změnu přihlašovacího hesla (viz kap. 3.2.4.3).

3.2.4.3 Uživatel - Změna přihlašovacího hesla

Na jednoduché stránce je pro úspěšnou změnu nutné do formuláře zadat 1x původní heslo a 2x nové heslo. To by mělo eliminovat možnost překlepu při zadání hesla. Heslo musí mít nejméně 5 znaků - formulář to hlídá a uživatele upozorní ještě před odesláním dat na server.

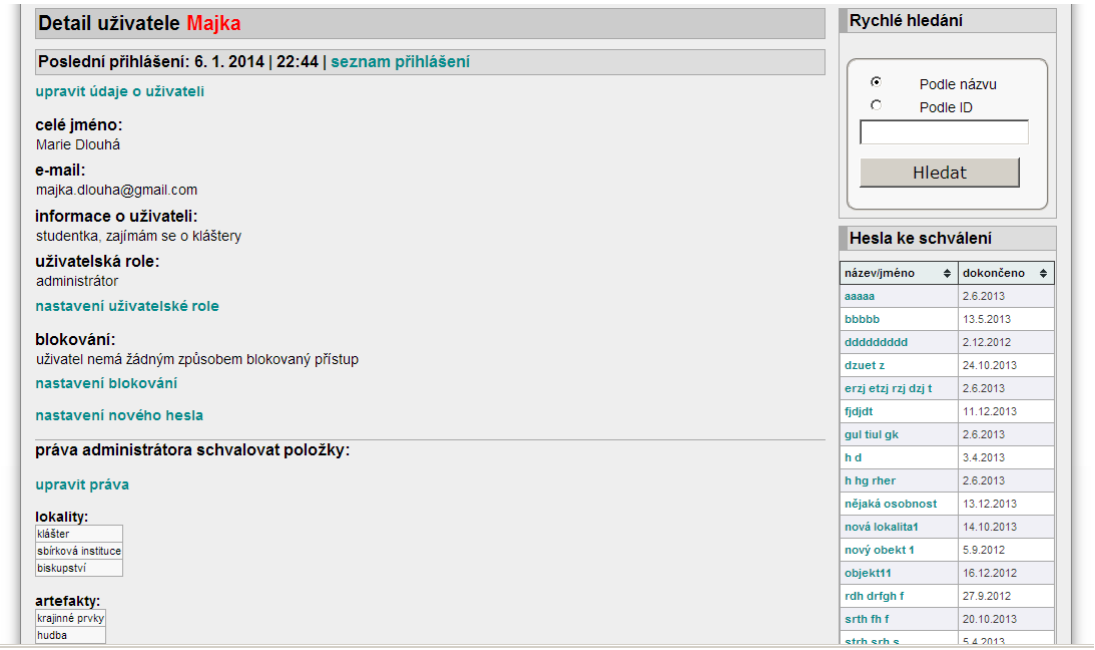
3.2.4.4 Detail uživatele

Na stránku s detaily uživatele je možné se dostat několika způsoby. Přes seznam všech uživatelů (viz kap. 3.2.4.1), přes různé odkazy v jiných seznamech, na stránku s vlastním profilem také odkazem (už. jméno) v uživatelském panelu (viz kap. 3.2.3) nebo v menu *Uživatel - Informace o mně*.

Stránka nabízí vyčerpávající informace o daném uživateli na jednom místě. Hned v horní části stránky je viditelné datum a čas posledního přihlášení. Vedle údaje je odkaz na stránku, kde je stránkovaný seznam všech přihlášení (datum a čas). Dále je na stránce viditelné vlastní jméno, e-mail, role a informace o tom, zda má uživatel nastavené určité blokování (viz kap. 3.4.4). Pokud je zobrazený profil právě přihlášeného uživatele, jsou zde také odkazy na stránku editace vlastních informací (viz kap. 3.2.4.2) a na stránku editace přístupového hesla (viz kap. 3.2.4.3). Pokud jde o profil administrátora, je zde i sekce, ve které jsou vypsány všechny typy lokalit a artefaktů a druhy osobností, na které má daný administrátor schvalovací práva.

V každém profilu uživatele je ve spodní části stránky především obsáhlý seznam (stránkovaná tabulka) všech hesel, která nějakým způsobem souvisí s daným uživatelem. Jde o hesla, která uživatel:

- založil
- editoval
- která jsou pro daného uživatele odemčena



Obrázek 3.6: Detail lokality

Tabulku lze jako obvykle řadit a odkazem se prokliknout na všechna hesla. Obsahuje u každého hesla jeho typ, stav a údaje o vytvoření / editaci, příp. zda je heslo uživateli odemčeno.

3.2.4.5 Přidat - Lokalitu

Jedná se o jednu z nejdůležitějších stránek. Obsahuje poměrně komplexní formulář pro přidání hesla lokality do systému. Jedinými povinnými údaji jsou *název lokality* a její *typ*. V závislosti na tom, jaký typ je vybrán, jsou poté některé další pole zobrazena nebo skryta. *Další názvy* se vepíšou do jednoho text. pole a oddělí středníkem. U *první zmínky* a *založení* platí, že je možné vyplnit přesný rok, nebo přibližné rozmezí - tedy 2 letopočty. *Světce* je možné vybrat z nabídky, nebo přidat do databáze nového. Pokud je světců více, evidují se v části *Světci - vedlejší*. Přidávají se jeden po druhém stejně jako ten hlavní.

U *představených* přidaných k lokalitě je proces komplikovanější. Stejně jako u vedlejších světců, je možné přidat představených neomezeně. Přidává se opět jeden po druhém, u každého je volba zda vybrat z osobností, které jsou již v databázi, nebo zda přidat novou osobnost. V takovém případě bude po odeslání formuláře osobnost vytvořena jako návrh (viz začátek této kapitoly). Pokud se uživatel rozhodne touto cestou osobnost/návrh přidat, musí pokaždé vyplnit jméno a druh (druh *představený* je již vybrán a nelze ho měnit, viz začátek této kapitoly), případně přidat k osobnosti poznámku, například ujasňující, o jakou osobnost se jedná. V každém případě má uživatel možnost u každého představeného nastavit letopočty jeho působení v dané lokalitě. Následně lze ještě s pořadím představených manipulovat. Před odesláním formuláře systém zkontroluje nejen letopočty právě zadané, ale také letopočty všech působení všech představených, a varuje uživatele, pokud v některém případě dochází k časovému překrytí. Uložit takovou

lokalitu ale uživateli nebrání.

Připojení *fundátorů* probíhá jednodušeji - nezávisí zde na pořadí. Podobným způsobem je možné přidat *mateřské*, resp. *dceřiné* kláštery - napojit již existující lokality nebo přidat nové. Ty se opět vytvoří jako návrhy. Následují velká textová okna pro hlavní texty (*Historie*, v případě některých typů lokalit *stavební vývoj*, dále *prameny* a *literatura*). Dále jsou přiřazeny případná *související hesla*. V elementech pro výběr je možné pro snadnější hledání text filtrovat. Také je možné vložit neomezené množství *externích odkazů*, např. na webové stránky kláštera a u každého odkazu zadat text pro odkaz a podrobnější popis. K lokalitě lze v tomto formuláři nahrát jednu *fotografii* (a vepsat název a podrobnější popis obrázku), nebo je možné jich více připojit, pokud jsou již v databázi nahrané. Slouží k tomu speciální *drop-down* element s náhledy obrázků. Soubor je možné pouze nahrát, přiřazení již nahraného probíhá přes jiný formulář (viz kap. 3.2.7). Před odesláním je možné vepsat interní poznámku, která slouží pro informaci ostatním uživatelům, ale veřejnosti se zobrazovat nebude. Dále, pokud již uživatel tímto lokalitu rovnou dokončil, označí příslušný zakškrtávací box a připraví ji tak administrátorovi ke schválení.

Pokud je známa poloha lokality, má uživatel před odesláním formuláře také možnost označit ji na mapě jednoduše přesunutím značky a kliknutím na tlačítko *Uložit pozici*. Uložení lokality potom probíhá tlačítkem *Uložit lokalitu*, ale pokud systém najde ještě před odesláním chybějící nebo nevalidní data, upozorní uživatele příslušným varováním v horní části stránky. Po uložení je uživatel přesměrován přímo na detail hesla.

3.2.4.6 Syntaxe psaní poznámek v textu

U delších textů je možné přidávat poznámky, které se zobrazí až pod samotnými texty. Podobně jako na Wikipedii, k tomu slouží vlastní syntaxe. Jednoduše se v textu poznámka uzavře do složených závorek:

```
...leží ve městečku Sázava{tato Sázava se nachází v okrese Benešov} nad
   stejnojmennou řekou.
```

V textu v detailu hesla bude tedy toto:

```
...leží ve městečku Sázava[1] nad stejnojmennou řekou.
```

...a příslušná poznámka bude až pod samotnými texty.

3.2.4.7 Přidat - Artefakt

V případě přidání nového artefaktu je formulář velmi podobný jako u lokality. Zde jsou tedy uvedeny jen odlišnosti. Jako letopočty se u artefaktu uvádí *První zmínka* a *datace*. V sekci *dnešní sbírkové uložení* je vybrána lokalita, nebo založená nová jako návrh (typu *sb. uložení*) - princip podobný jako např. u fundátorů ve formuláři lokalit. Jako *původ* má možnost uživatel vybrat nebo přidat nové místo a zemi. Budoucnost tohoto údaje v databázi je však nejistá. Hlavní texty se zde označují *popis* a *rozměry*.

3.2.4.8 Přidat - Osobnost

I v tomto případě je formulář podobný. *Druhů osobností* je možné vybrat více, pokud je ale jeden z druhů *představený*, musí být zároveň přiřazeno alespoň jedno *působení*. Přiřadit představeného lze jen k již existující lokalitě, přičemž pokud daná lokalita již nějaké představené přiřazené má, je zároveň nabídnut drop-down seznam pro chronologické zařazení působení. Opět je možné zde zadat i letopočty působení. Jako letopočty osobnosti mohou být uvedeny hned tři - *narození*, *úmrtí* a *první zmínka* - opět jako konkrétní roky nebo přibližné rozpětí. *Původ osobnosti* - viz původ artefaktu. Hlavní text je zde jen jeden, a to *Biografie*.

3.2.4.9 Přidat - Obrázek

Obrázek lze do systému přidat jak s určitým heslem, tak samostatně. K tomu slouží právě tento formulář. Je možné zadat název, popis, a rovnou vybrat hesla, ke kterým má být připojen. Vždy platí, že první připojený obrázek k heslu je nastaven jako titulní. Pokud je obrázků u hesla více a uživatel by chtěl jako titulní nastavit jiný, může to následně provést ve formuláři editace hesla.

3.2.4.10 Přidat - Soubor

Formulář pro samostatné přidání souboru je téměř shodný s formulářem pro přidání obrázku.

3.2.4.11 Přidat - Návrh

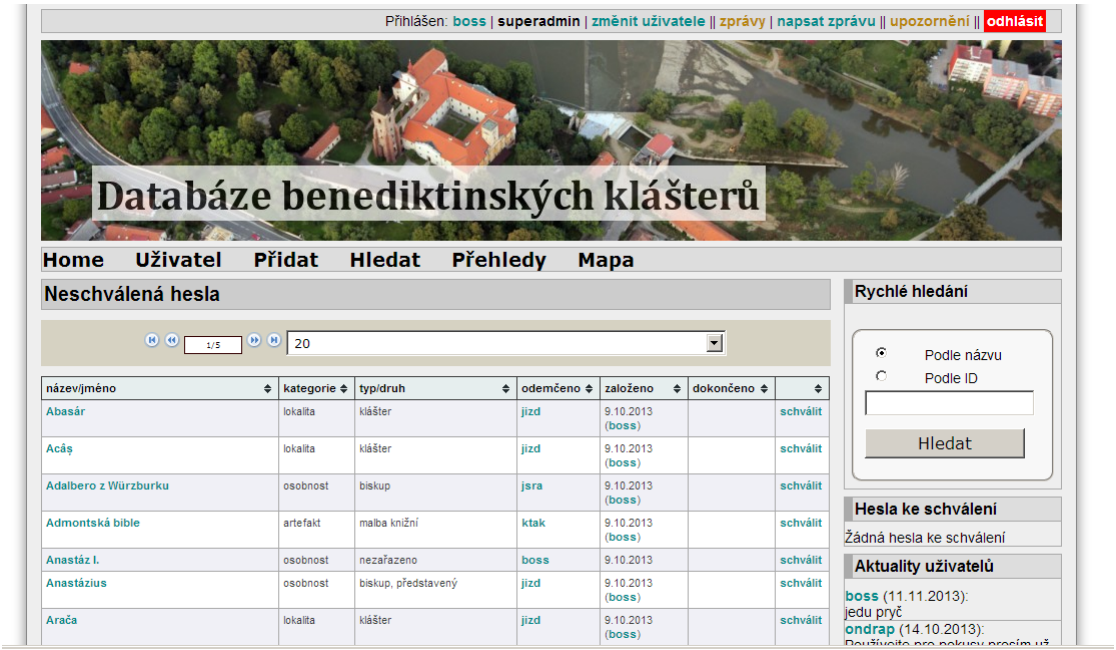
Heslo je také možné vytvořit ještě před jeho zpracováním jako návrh (viz začátek této kapitoly). Slouží k tomu jednoduchý formulář, kde musí být zadána kategorie hesla a konkrétní typ. Navíc je vhodné další podrobnosti rozepsat v poli poznámka, ať je např. ostatním uživatelům jasné, o kterou lokalitu či osobnost jde.

3.2.4.12 Přehledy - Návrhy

Vytvořené návrhy, které dosud čekají na první úpravu (a tím změnu na jiný stav hesla), jsou přehledně vypsány v tabulce na této stránce. Tabulka je opět stránkovaná a umožňuje řazení. U každého návrhu je uveden autor, datum založení a kategorie / typ. Podrobnější informace (ona poznámka) se zobrazí po kliknutí na „info“ vedle názvu návrhu. Kliknutím na *zpracovat* se uživatel dostane přímo na editační formulář.

3.2.4.13 Přehledy - Neschválená hesla

Tato stránka s obsáhlou tabulkou zobrazuje přehledně všechna hesla, která jsou typu „rozpracováno“ nebo „dokončeno“ - tedy nejedná se o návrhy a dosud čekají na schválení. U všech hesel je uveden název, kategorie, typ, uživatel pro kterého je odemčeno, datum založení a u těch které jsou dokončeny také datum dokončení. Podle všech těchto parametrů lze stránkovanou tabulku řadit.



Obrázek 3.7: Výpis neschválených hesel - výřez

3.2.4.14 Přehledy - Fotografie

Stránka s přehledem všech obrázků zakládá na jejich náhledech. Nejedná se tedy o tabulku s názvy, ale přímo čtvercovou síť náhledů obrázků. Název obrázku se objeví až po najetí myši a detailnější informace se uživatel dozví až po kliknutí na něj. Odkaz totiž vede přímo na stránku s detailem obrázku (viz kap. 3.1.5.1).

3.2.4.15 Přehledy - Soubory

Přehled všech nahraných souborů (formátu *PDF*) uživatel najde na této stránce. U každého má možnost jedním kliknutím soubor zobrazit, a nebo kliknout pro zobrazení stránky s editací (viz kap. 3.2.7). Avšak název i podrobnější popis je viditelný již v tomto přehledu. Pokud je popis delší, stačí na oddíl se souborem najet myší.

3.2.5 Detail hesla - rozdíl oproti návštěvníkovi stránek

Stránka s detailem hesla obsahuje všechny informace jako ty, které vidí nepřihlášený uživatel. Ale navíc obsahuje ještě další informace, které jsou důležité jen pro registrované uživatele. Všechny tyto informace jsou přehledně odděleny vodorovnou čárou. Jedna odlišnost je však i v horní části stránky - hned za nadpisem hesla je uvedeno barevně jeho *ID*, např. pro potřeby vkládání interních odkazů do textu (viz kap. 3.2.1.1). Pod čarou je nejprve informace kdo a kdy heslo založil. Dále oddíl s několika posledními úpravami hesla a odkazem na stránku s výpisem všech úprav. Taková stránka (možnost řadit) zobrazuje přehledně všechny úpravy hesla, datum a čas editace, editora, případně poznámku k editaci[3]. Editace, při které bylo heslo schváleno administrátorem, je barevně zvýrazněna. Te-

[3] u každé editace je možné a žádoucí uvést stručnou poznámku o tom, co bylo editováno (viz kap. 3.2.6)

oreticky však mohou být v přehledu úprav schváleného hesla i úpravy po tomto schválení - za editaci se považuje např. i připojení představeného k lokalitě (ze strany lokality).

Dále je na stránce hesla interní poznámka k heslu, jeho stav, případně uživatel, pro kterého je heslo odemčeno. Pod tím je k dispozici seznam administrátorů, kteří mají právo heslo tohoto typu schválit. Ke každému heslu je také připravena jednoduchá diskuze s polem pro odeslání příspěvku. Pokud je aktuálně přihlášen uživatel, pro kterého je dané heslo odemčeno (případně se jedná o návrh - ten může editovat jakýkoliv uživatel), je zde možné také nastavit typ hesla. Toto nastavení není součástí formuláře úpravy hesla kvůli závislosti typů hesel na právech administrátorů. V případě osobnosti může být tato sekce složitější. Protože změnou druhu osobnosti na *představeného* je nutné i osobnost k lokalitě jako představeného přiřadit, zobrazí se v takovém případě i možnost přidat libovolný počet působení představeného. A to včetně letopočtu a zařazení v chronologii lokality. Dále je na stránce odkaz na jeho propojení s fotografiemi.

Na stránce propojení hesla s fotografiemi je možné naklikat libovolné množství fotografií které jsou v databázi a tím je přiřadit k heslu - kliknutím na odesílací tlačítko. Je zde i odkaz na formulář přidání nové fotografie, ve kterém je již předvyplněno toto související heslo.

Důležitým odkazem je *editovat* - to uživatele dostane na stránku s editačním formulářem.

3.2.6 Editace hesla

Editační formulář hesla je velmi podobný tomu pro přidání nového hesla. Neumožňuje ale změnit typ / druh hesla a pokud má heslo již připojeny nějaké obrázky, je zde také možnost nastavit, který má být titulním - s využitím přepínačů a náhledy obrázků. Každá změna navíc umožňuje - a je to doporučeno - stručně popsat, co bylo při editaci hesla změněno. Nad formulářem jsou pro přehled data posledních editací hesla od každého uživatele - tedy pokud heslo již bylo editováno. Také je možné v případě lokality upřesnit polohu mapy a kliknutím na tlačítko *Uložit pozici* také uložit do databáze stát a podrobnější administrativní oblasti (viz kap. 4.2.1).

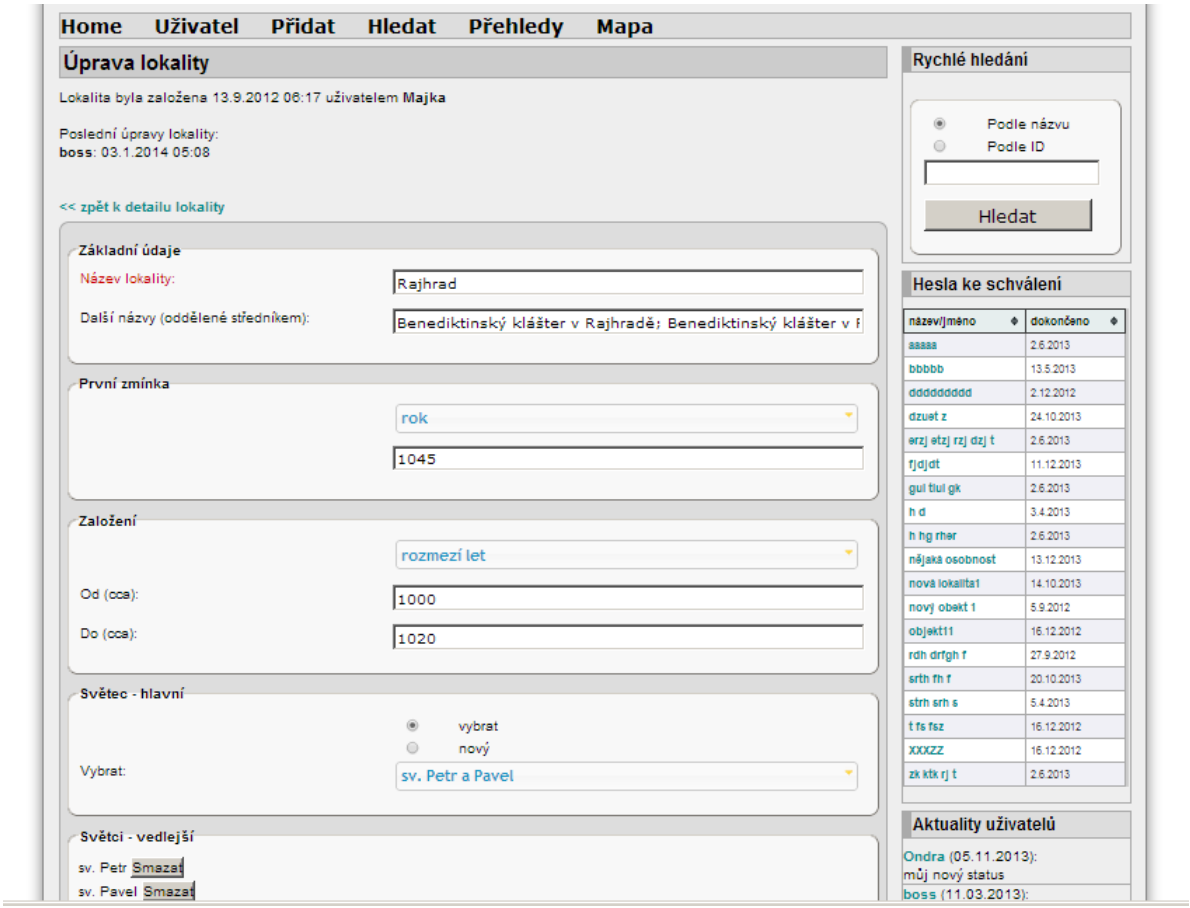
Pokud je editováno heslo ve stavu *návrh*, je třeba pamatovat, že uložením změn se návrh mění na rozpracované, příp. dokončené heslo a uzamyká se editujícímu uživateli.

3.2.7 Editace obrázku nebo souboru

Již na stránce detailu fotografie má registrovaný uživatel navíc možnost vidět, kdo a kdy obrázek vložil. Různými cestami - ať už z detailů hesel nebo ze stránek přehledů fotografií či souborů - se uživatel dostane na stránku s úpravou obrázku nebo souboru. V jednoduchém formuláři je možné upravit název i stručný popis obrázku či souboru a dále jeho přiřazení ke konkrétním heslům.

3.3 Informační systém z pohledu administrátora

Pohled administrátora se příliš neliší od toho, co má možnost vidět běžný registrovaný uživatel. Jak už bylo řečeno, administrátor má na rozdíl od běžného uživatele práva



Obrázek 3.8: Editace hesla - výřez formuláře

na schvalování určitých typů / druhů hesel. Jakých, to určuje hlavní administrátor.

3.3.1 Pravá lišta

První funkce navíc, kterou je možné zaregistrovat, je v pravé liště. Mezi panelem rychlého hledání a aktualitami uživatelů je vždy při ruce seznam několika *hesel ke schválení*. Jedná se o jednoduchou tabulku (možnost řazení) všech hesel, která jsou označena jako dokončená a zároveň má aktuálně přihlášený uživatel právo na jejich schválení. Zobrazen je název, resp. jméno a datum dokončení hesla. Některé další informace jsou k dispozici po najetí myši na název hesla. To slouží jako odkaz na detail daného hesla.

3.3.2 Detail hesla

Pokud administrátor zobrazí stránku s detailem hesla, na které má právo schvalování, má stejné možnosti editace jako uživatel, pro kterého je heslo odemčeno. Tedy editovat ve speciálním formuláři a nastavit propojení na fotografie. Odkaz na editaci hesla je však tentokrát text *editovat/schválit*. Stejně tak má možnost u hesla změnit jeho *typ / druh*. Logicky, pokud změní typ nebo druh na takový, na který nemá schvalovací práva, po této změně již heslo editovat nemůže.

Dalším rozdílem na této stránce je možnost heslo definitivně smazat, tedy opět pokud má administrátor schvalovací právo. Jednoduchý formulář smazání je na stránce úplně dole a pro úspěšné smazání musí administrátor zadat své přihlašovací heslo. Smazané heslo již poté nejde „vrátit zpět“.

Pokud je stav hesla *rozpracováno* nebo *dokončeno* pod sekcí, kde je uveden uživatel, pro nějž je heslo odemčeno, je odkaz *předat jinému uživateli*. Vede na stránku s jednoduchým formulářem, ve kterém lze vybrat jakéhokoliv (tedy i sebe samého) uživatele a jemu heslo předat.

Pokud je heslo schválené a uzamčené, nemůže jej před odemčením ani administrátor editovat. Pokud ale administrátor má schvalovací práva, má možnost heslo odemknout (a tím jej zneviditelnit pro veřejnost). Odkaz na stránku odemknutí je v sekci *odemčeno pro uživatele*. Na stránce odemčení hesla opět stačí vybrat jednoho uživatele a odesílacím tlačítkem heslo jednoduše odemknout. Tím je heslu nastaven stav *rozpracováno*.

3.3.3 Editace hesla

Pokud administrátor není sám tvůrce a editor hesla a chystá se heslo schválit, ve formuláři editace hesla pouze zkontroluje údaje. V případě administrátora jsou však odesílací tlačítka hned 2 rovnocenná. První s textem *Uložit změny a neschvalovat* a druhé s textem *Uložit změny a schválit*. Význam je jasný - první tlačítko funguje obdobně jako v případě běžného uživatele (například situace, kdy administrátor po uživateli před schválením ještě požaduje úpravy hesla) a druhé kromě uložení změn heslo schválí a uzamkne - tedy zobrazí pro veřejnost.

Na stránce s přehledem všech neschválených hesel je pro administrátora zobrazen další sloupec, v němž je u vybraných hesel odkaz *schválit* - jednoduše u těch, které schválit může (ale nemusí být ještě dokončená). Tím, že lze i podle tohoto sloupce tabulku řadit, je lze snadno vytřídit. Odkazy však vedou na detaily hesel.

Stejně tak v tabulce již schválených hesel vidí administrátor u vybraných hesel odkaz *odemknout* vedoucí na stránku s odemčením daného hesla konkrétnímu uživateli. Opět lze tato hesla řazením vytřídit.

3.4 Informační systém z pohledu hlavního administrátora

Hlavní administrátor má všechna práva administrátora a k tomu další navíc. Zatímco funkce běžných administrátorů je orientována spíše na odborná hesla a samotný obsah databáze, funkce hlavního administrátora je především ve správě uživatelů. V zásadě hlavní administrátor nemusí být na rozdíl od ostatních administrátorů odborník historik. Avšak má jejich schvalovací práva. Pokud kdokoliv přidá příspěvek diskuze k libovolnému heslu, hlavní administrátor je na to upozorněn.

3.4.1 Hlavní strana

Již na hlavní straně je viditelný rozdíl u hlavního administrátora. Jednotlivé články na nástěnce lze včetně diskuzí smazat, např. pokud už nemají význam apod. Odkazy pro smazání článku jsou vždy hned v hlavičce za odkazem do diskuze.

3.4.2 Editace osobních údajů

Tato stránka se liší jen možností editovat kromě základních informací a e-mailu také vlastní a uživatelské jméno.

3.4.3 Uživatel - registrace uživatele

Tato stránka je viditelná pouze pro hlavního administrátora a umožňuje s pomocí formuláře do systému přidat dalšího uživatele. Všechny položky jsou povinné - *uživatelské jméno*, *celé jméno*, *e-mail*, *uživatelská role* (administrátor nebo běžný uživatel) a dvakrát *heslo* - podruhé pro kontrolu. Heslo si poté sám uživatel má možnost samozřejmě po přihlášení změnit (viz kap. 3.2.4.3). Registrace tedy probíhá vždy jen ze strany hlavního administrátora, což souvisí i s faktem, že uživatelů bude „pouze“ několik desítek. Většina uživatelů ve finální verzi databáze je navíc již v tuto chvíli hlavním administrátorem registrovaná.

3.4.4 Detail uživatele

Stránky detailů jednotlivých uživatelů jsou z pohledu hlavního administrátora rozšířeny o několik odkazů. Hned v horní části je odkaz na stránku úpravy uživatelských údajů - *upravit údaje o uživateli*. Ve formuláři na této stránce má možnost upravit uživateli veškeré údaje vč. uživatelského jména.

Pod údajem uživatelské role (běžný uživatel / administrátor) je odkaz na stránku s jednoduchým formulářem úpravy role *nastavení uživatelské role*. Zde u uživatele přepíná mezi rolí administrátora a ostatního uživatele.

V sekci *blokování* je zase odkaz na stránku, na které je nastavení blokování daného uživatele. V jednoduchém formuláři může zaškrtnout i více z následujících možností:

- **Přihlášení** - uživatel nemá možnost ani se do systému přihlásit, stále je ale registrován
- **Editace položek** - uživatel nemůže editovat žádná hesla, nastavovat jiné typy, ani editovat údaje o obrázcích a souborech
- **Zakládání nových hesel** - uživatel nemůže zakládat nová hesla (tedy kromě návrhů)
- **Vytváření nových návrhů** - uživatel nemůže vytvářet nové návrhy hesel
- **Přidávání článků na nástěnku** - uživatel nemůže přidat nový článek na nástěnku

Pod odkazem blokování se nachází odkaz pro nastavení nového přihlašovacího hesla uživateli - *nastavení nového hesla*. Je nutné jednou zadat přihlašovací heslo hlavního administrátora a dvakrát (pro kontrolu) nové přihlašovací heslo daného uživatele.

V případě, že jde o profil administrátora, v oddílu *práva administrátora schvalovat položky* je odkaz *upravit práva* na stránku, kde hlavní administrátor může s pomocí zaškrtávacích boxů naklikat jednotlivé typy a druhy hesel, na která má mít příslušný administrátor schvalovací práva. Odesílacím tlačítkem se opět uloží změny.

Pokud by administrátor měl určitá schvalovací práva a hlavní administrátor tomuto administrátorovi změnil roli na běžného uživatele, tato schvalovací práva zůstanou uložena, takže pokud poté opětovně nastaví uživateli roli administrátor, schvalovací práva zůstanou ta naposled nastavená.

3.4.5 Další funkce hlavního administrátora

Co se týká editací a schvalování hesel, hlavní administrátor má absolutní práva editovat, schvalovat, odemykat, předávat jinému uživateli, případně mazat jakékoliv heslo (musí před smazáním jen zadat své přihlašovací heslo). Editovat heslo ovšem nemůže ani hlavní administrátor, pokud je uzamčené.

Na stránce detailu obrázku je také jednoduchý formulář pro jeho smazání - hlavní administrátor zadá své přihlašovací heslo a tlačítkem obrázek nenávratně smaže z databáze i z pevného disku serveru.

Kapitola 4

Implementace

Při tvorbě podobné aplikace, resp. databáze, je nutné dodržet chronologii některých základních kroků a také oddělit jednotlivé složky - tedy grafický vzhled aplikace, její datový model a funkcionalitu. Nejinak tomu bylo i zde. První prací tedy bylo sepsání základních požadavků zadavatele, vytvoření co nejvíce propracovaného návrhu databáze, a až poté postupná implementace jednotlivých controllerů (dále řadičů), akcí a modelů MVC aplikace a jejich propojování. V další fázi přišla na řadu veškerá implementace javascriptových aplikací a pluginů na straně klienta, včetně AJAXu. To však až ve chvíli, kdy základ aplikace na straně serveru byl samostatně funkční. Jinak řečeno, aplikace by měla být co nejméně závislá na využití JavaScriptu.

Nakonec přišla na řadu implementace mapové aplikace a její propojení se zbytkem databáze.

Naopak samotná grafika aplikace je doposud velmi strohá a je žádoucí řešit ji až nakonec - pravděpodobně v jednotném stylu s webovými stránkami související výstavy (viz kap. 5). Právě oddělené soubory s HTML výstupem a soubory layoutu takové konečné úpravy (např. přidáváním CSS tříd k elementům, příp. úpravou samotného HTML) velmi usnadní.

Vývoj aplikace probíhal po celou dobu na serveru *Apache*, který mám instalovaný na svém osobním počítači, stejně jako *PHP* a *MySQL*. Po nějaké době byla celá aplikace zkopírována na server *geo5.fsv.cvut*, který běžel prakticky jen kvůli tomuto projektu, a po další době byla zkopírována na server Filosofického ústavu *web.flu.cas.cz*. Od té doby již byla aktualizovaná pouze aplikace na serveru filozofického ústavu, školní server byl vypnut. Po každé větší změně, nebo více změnách, které byly provedeny na *localhostu*, byla aplikace přehrána na server, případně změněna struktura databáze. Jediný soubor, který musel být na serveru vždy upraven zvlášť, je konfigurační, ve kterém jsou definovány přihlašovací údaje do databáze.

4.1 Implementace databáze jako webové aplikace

Prvotní popis požadované databáze přišel ze strany dodavatele ve formě jednoduchého *.doc* dokumentu, ve kterém byly popsány 3 základní typy hesel a údaje, které je potřeba u těchto hesel evidovat. Naopak třeba o vrstvách uživatelů, jejich právech atd. bylo se nutné na schůzkách se zadavatelem přímo domluvit. Na mně, jako tvůrci databáze

tedy bylo např. určit, co evidovat jako sloupce tabulky, na co využít tzv. propojovací tabulky, jaké datové typy použít apod.

4.1.1 Návrh a implementace databáze

4.1.1.1 Počáteční fáze návrhu

Základní návrh databáze byl postupně sestavován na základě požadavků Centra medievistitckých studií (CMS). Probíhal nejprve na papíře, poté v programu *MySQL Workbench*. Tento program umožňuje již v návrhu, tedy aniž by se databáze skutečně vytvořila, sestavit *model databáze*. Tedy popsat jednotlivé tabulky, jejich vztahy, cizí klíče, přidávat rutiny, generovat ER diagram a další. Na základě tohoto modelu lze poté jednoduše vygenerovat *SQL CREATE* soubor pro vytvoření skutečné databáze.

Struktura databáze se usměrňováním požadavků CMS od návrhu mnohokrát změnila, podstatné ale zůstalo - tabulky *lokality*, *objekty*, *osobnosti* a *users*, které tvoří základ databáze.

4.1.1.2 Popis databáze

Následující tabulka přehledně popisuje všechny tabulky použité v modelu databáze, resp. v relační databázi samotné.

Tabulka 4.1: Přehled tabulek relační databáze

název tabulky	**popis**
adminsprava_druhyosobnosti	tabulka práv administrátorů na druhy osobností - jedná se o propojovací tabulku uživatelů s druhy osobností
adminsprava_typylokality	tabulka práv administrátorů na typy lokalit - jedná se o propojovací tabulku uživatelů s typy lokalit
adminsprava_typyobjektu	tabulka práv administrátorů na typy artefaktů - jedná se o propojovací tabulku uživatelů s typy artefaktů
clanky	tabulka uživatelských článků na nástěnce - eviduje nadpis, samotný text (typ BLOB - binární), začátek textu zvlášť a informace, kdo a kdy článek přidal
diskuzeclanky	tabulka příspěvků diskuze k článkům - eviduje čas, uživatele, text přípěvku a ID článku
druhyosobnosti	tabulka druhů osobností
filiace	propojovací tabulka filiačních vztahů mezi lokalitami - tedy mateřskými a dceřinými

formaty	tabulka formátů vkládaných souborů; v současné době je pevně dána možnost vkládat jen PDF soubory, v budoucnu je zde ale možnost rozšíření např. na dokumenty Word, komprimované soubory apod.
fotografie	tabulka fotografií - ukládají se názvy, popis, časy přidání a uživatel, který obrázek přidal, koncovky, velikosti a jejich hash součty, které odpovídají názvům souborů uložených v adresářové struktuře na disku; nejedná se tedy o ukládání samotných binárních dat v tabulce
funkcnosti	jednoduchá tabulka několika stavů lokality - zrušená / zničená / stojící, apod.
itemsnotes	tabulka příspěvků diskuze k jednotlivým heslům v databázi
links	tabulka url odkazů z jednotlivých hesel; eviduje text linku, jeho url a popis
logging	tabulka, do které se ukládají všechny časy přihlášení jednotlivých uživatelů
lokality	tabulka lokalit, jedna ze 3 základních tabulek s hesly; eviduje mj. název, typ lokality, typ kláštera, stav lokality, informace o první zmínce a založení lokality (rok, rozmezí let), hlavního světce, dále hlavní texty k heslu (historie, stavební vývoj, prameny, literatura), uživatele, který heslo vytvořil, uživatele, který má heslo odemčeno pro úpravy, čas vytvoření hesla, stav hesla, interní poznámku k heslu, souřadnice lokality, stát, administrativní region a sídlo - generováno pomocí tzv. reverzního geokódování, viz kap. 4.2.1
lokalityothernames	tabulka dalších názvů jednotlivých lokalit
lokality_fotografie	propojovací tabulka lokalit a fotografií; eviduje navíc informaci o tom, zda je obrázek u hesla nastaven jako titulní
lokality_fundatori	propojovací tabulka lokalit a osobností, kde propojení znamená fundaci u dané lokality
lokality_lokality	propojovací tabulka lokalit s jinými lokalitami - na úrovni „související“

lokality_opati	propojovací tabulka lokalit a osobností, kde propojení znamená působení představeného v dané lokalitě; ukládá se nejen ID osobnosti a lokality, ale také její pořadí v lokalitě a roky působení
lokality_plany	propojovací tabulka lokalit a plánů; protože tabulka plánů pravděpodobně nebude využita a nahradí ji tabulka souborů, bude zřejmě odstraněna i tato nevyužitá tabulka
lokality_soubory	propojovací tabulka lokalit a souborů
lokality_svetci	propojovací tabulka lokalit a tzv. vedlejších světců
messages	tabulka textových zpráv mezi uživateli; eviduje od koho a komu je zpráva určena, čas vytvoření, předmět, text zprávy, zda jde o odpověď na jinou zprávu a informaci o tom, zda byla zpráva již přečtena
mista	tabulka míst, či sídel, např. míst původu osobností nebo artefaktů; tato tabulka je stále sporná, v zásadě bude buď zrušena a nahrazena tabulkou lokalit, nebo bude naopak rozšířena o souřadnice a údaje např. o původu osobnosti, čímž je bude možné zobrazit v mapě.
objekty	tabulka artefaktů; další klíčová tabulka, s druhým typem hesla; původní požadavek byl u artefaktů použít výraz „objekt“, „artefakt“ ale lépe vystihuje předmět evidence - a tak byl v kódu a samotné databázi ponechán výraz „objekt“, ale v HTML výstupu (frontend) je již všude využit výraz „artefakt“; tabulka eviduje mj. název artefaktu, časové informace o vzniku a dataci artefaktu (rok, rozmezí let), typ objektu, místo a stát původu, místo dnešního uložení (lokalita typu sbírková instituce), hlavní texty (popis, literatura, prameny), čas vytvoření hesla, uživatele, který jej vytvořil a uživatele, který má heslo odemčeno pro úpravy, stav hesla a interní poznámku k heslu
objektyothernames	tabulka dalších názvů jednotlivých artefaktů
objekty_fotografie	propojovací tabulka artefaktů a fotografií; eviduje navíc informaci o tom, zda je obrázek u hesla nastaven jako titulní

objekty_lokality	propojovací tabulka artefaktů a lokalit na úrovni „související“
objekty_mista	propojovací tabulka artefaktů a míst coby původu artefaktu (viz tabulka *mista* výše)
objekty_objekty	propojovací tabulka artefaktů s dalšími artefakty na úrovni „související“
objekty_soubory	propojovací tabulka artefaktů a souborů
osobnosti	tabulka osobností; třetí hlavní tabulka se samotnými hesly; eviduje mj. jméno osobnosti, časové údaje o první zmínce, narození a úmrtí (rok, rozmezí let), místo původu osobnosti, hlavní texty (biografie, prameny, literatura), čas vytvoření hesla, uživatele, který heslo založil, uživatele, pro kterého je heslo odemčeno k úpravám, stav hesla a interní poznámku k heslu
osobnostidalsijmena	tabulka dalších jmen jednotlivých osobností
osobnosti_druhyosobnosti	propojovací tabulka osobností a tabulky druhů osobností - osobnost, na rozdíl od lokality či artefaktu může mít zadaných více druhů
osobnosti_fotografie	propojovací tabulka osobností a fotografií; eviduje navíc informaci o tom, zda je obrázek u hesla nastaven jako titulní
osobnosti_lokality	propojovací tabulka osobností a lokalit na úrovni „související“
osobnosti_objekty	propojovací tabulka osobností a artefaktů na úrovni „související“
osobnosti_osobnosti	propojovací tabulka osobností s dalšími osobnostmi na úrovni „související“
osobnosti_soubory	propojovací tabulka osobností a souborů
plany	tabulka plánů; bude pravděpodobně zrušena a nahrazena tabulkou souborů
soubory	tabulka souborů - eviduje jen informace o souborech, uživatele, který jej nahrál, formát, velikost, název, popis a další údaje, soubor samotný je uložen v adresářové struktuře na disku; stejně jako u fotografií, nejedná se tak o tabulku s binárními daty
staty	tabulka států; využitá jak pro polohu lokalit - a jejich následné filtrování v mapě - tak pro původ artefaktu a osobnosti
svetci	tabulka světců lokalit

typylokality	tabulka typů lokalit; eviduje navíc také informaci, zda může mít daný typ lokality představeného, a zda může mít lokalita tohoto typu definovány filiační vztahy
typyobjektu	tabulka typů artefaktů
updatelokality	tabulka úprav jednotlivých lokalit - eviduje čas, uživatele, který heslo edituje a také stručné shrnutí editace
updateobjekty	tabulka úprav jednotlivých artefaktů - eviduje čas, uživatele, který heslo edituje a také stručné shrnutí editace
updateosobnosti	tabulka úprav jednotlivých osobností - eviduje čas, uživatele, který heslo edituje a také stručné shrnutí editace
upozorneniclanky	tabulka upozornění, která se týkají článků na nástěnce
upozornenientity	tabulka upozornění, která se týkají jednotlivých hesel
upozornenientity_users	propojovací tabulka upozornění, která se týkají jednotlivých hesel s tabulkou uživatelů; některé upozornění se totiž např. týká všech příslušných administrátorů ap.
upozorneniusers	tabulka upozornění, která se týkají uživatelů samotných
users	tabulka uživatelů; eviduje mj. jejich jméno, už. jméno, základní informace, e-mail, roli, aktuální informaci, čas registrace, informace o případném blokování uživatele, hashované heslo pro přihlášení a tzv. sůl pro šifrování hesla

Databáze má tedy celkem 53 tabulek, z nichž přibližně jedna polovina využívá formát (storage engine) *MyISAM*, druhá polovina *InnoDB*. Hlavní výhodou prvního formátu jsou především fulltextové funkce, druhý, modernější formát zase podporuje transakce, cizí klíče ad. Výchozí porovnání databáze je *utf8_czech_ci*. Poznamenejme, že nastavení cizích klíčů a jiné prvky referenční integrity jsou zajištěny především v samotné MVC aplikaci v PHP.

4.1.1.3 Datové typy

Jako datové typy sloupců v jednotlivých tabulkách jsou vždy zvoleny takové, aby co nejlépe vyhovovaly předpokládanému obsahu.

- **INT(11)** - v případě klíčů
- **TINYINT(n)** - v případě potřeby rozlišení několika možností, příp. jako náhrada za *boolean* (1/0 - **TINYINT(1)**)
- **SMALLINT(n)** - v případě letopočtů (unsigned)
- **FLOAT(15,11)** - v případě souřadnic
- **VARCHAR(n)** (řetězec dynamické velikosti) - v případě nadpisů, podnadpisů a krátkých textů
- **BLOB**, resp. **MEDIUMBLOB** - (binární datový typ) v případě delších textů
- **DATETIME** - v případě potřeba uložení data a času nějaké události

MySQL nabízí také datové typy speciálně pro uložení geografických informací, jako např. **POINT**, **LINE**, **CURVE**, **SURFACE**, ad. Jejich využití v této aplikaci by však nebylo takovou výhodou, a tak použity nebyly.

4.1.1.4 Správa databáze a změny její struktury

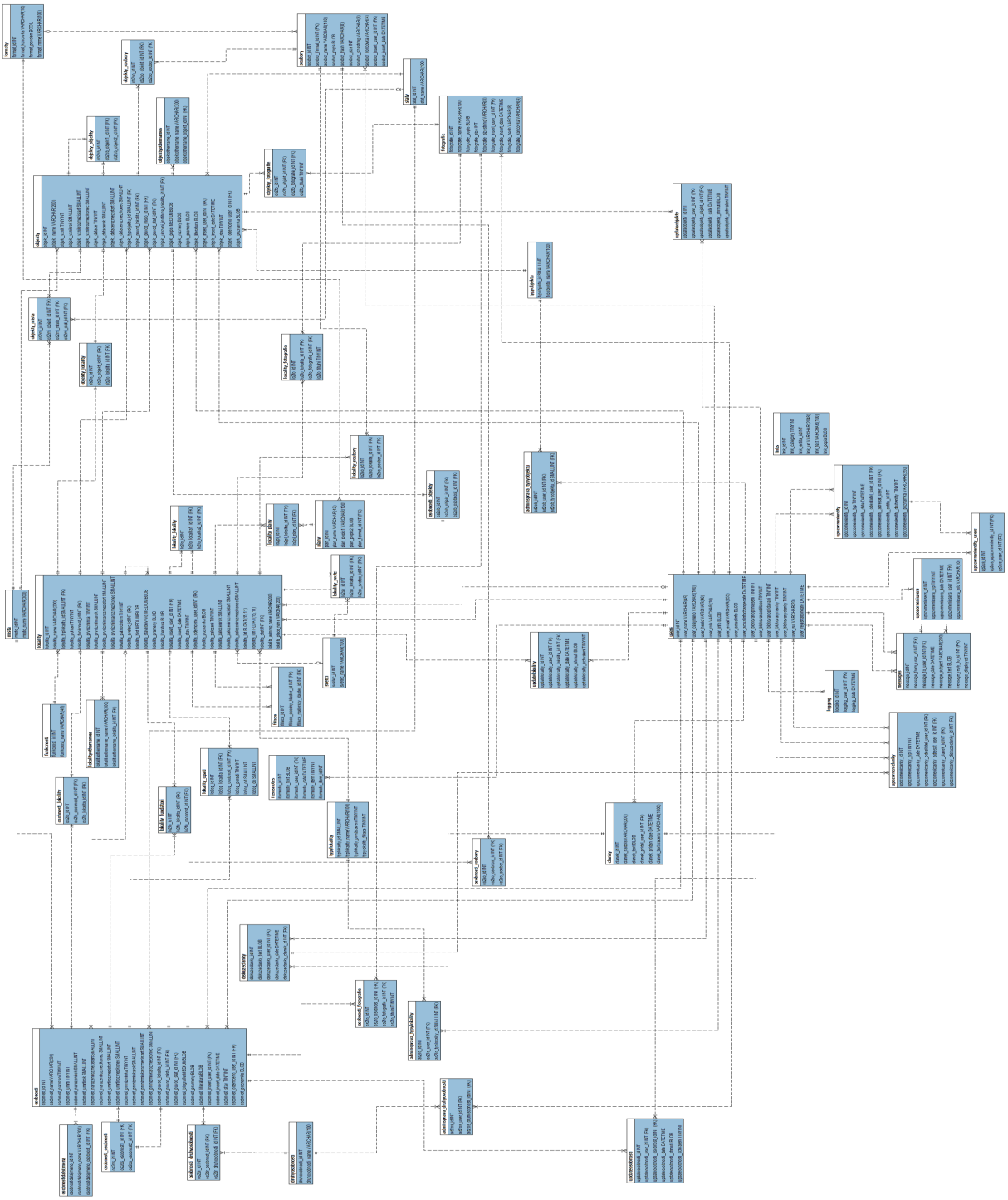
Jak už bylo řečeno, návrh databáze probíhal v software *MySQL Workbench*. S pomocí tohoto programu také proběhl první export do *SQL CREATE* souboru a byla databáze vytvořená. V první fázi také byla databáze tímto programem spravovaná, později se využíval známý nástroj pro správu *phpMyAdmin*, ver. 3.5. Umožňuje mj. ruční úpravy tabulek, úpravy s pomocí SQL příkazů, správu uživatelů i import a export. Jak školní server *geo5*, tak stávající server filozofického ústavu AV, kde databáze běží nyní, *phpMyAdmin* nabízí, a tak není problém změny struktury provedené na *localhostu* jednoduše přenést na server.

4.1.2 MVC aplikace v Zend Framework

K psaní aplikace bylo využito open source integrovaného vývojového prostředí (IDE) *NetBeans IDE*, jenž je součástí rozsáhlejšího projektu *NetBeans*. Podporuje množství programovacích jazyků, zejména *Java* a *PHP*, obsahuje kromě obvyklých funkcí, jako doplňování kódu, šablony pro tvorbu tříd, navigaci, formátování ap. také nástroje pro ladění a dále množství pluginů, mj. právě pro usnadnění práce se *Zend Framework* (dále ZF). Kód, ať už se jedná o phtml skripty, PHP, Javascript, nebo kaskádové styly, je poměrně důkladně okomentován pro rychlejší orientaci.

4.1.2.1 Instalace a základní kroky

Po rozbalení frameworku do požadované složky je potřeba vytvořit několik základních souborů. Jedná se o konfigurační soubor `application.ini` uložený ve složce `/application/config`, soubor `bootstrap.php` ve sožce `/application`, a samozřejmě `index.php`,



Obrázek 4.1: ER diagram databáze - v plné velikosti je k dispozici v příloze - viz příl. A

uložený ve složce `/public`. Tato složka poté slouží jako root pro web, do ní je potřeba přesměrovat případnou doménu. Obsahuje také další podsložky `/css`, `/files`, `/images`, `/js`, jejichž obsah by měl být podle názvu jasný, dále také soubor `favicon.ico`, a kořenová složka i podsložky mají přímý přístup omezený s pomocí souborů `.htaccess`.

Výchozí soubor **index.php** slouží ke spuštění celé aplikace. Definuje základní cesty do knihovny tříd, konfiguračního souboru a nastaví prostředí aplikace (development / production).

Soubor **bootstrap.php** slouží k základní konfiguraci. Od verze ZF 1.8 se ale samotné konfigurační údaje přesunuly do souboru *application.ini*. Soubor *bootstrap.php* tak slouží jen k inicializaci *ActionHelperu*, třídy `Klastery_Acl` a nastavení maximální doby pro vykonání akce. Níže je výpis *bootstrap* souboru:

```
<?php

class Bootstrap extends Zend_Application_Bootstrap_Bootstrap {

    protected function _initConfig() {
        $config = $this->getOptions();

        Zend_Registry::set('config', $config);
        Zend_Controller_Action_HelperBroker::addPath('Klastery/Controller/
            Action/Helper', 'Helper');

        ini_set('max_execution_time', 300);

        return $config;
    }

    protected function _initAcl() {
        $this->bootstrap('config');
        $config = $this->getResource('config');

        Zend_Registry::set('acl', new Klastery_Acl($config['acl']));
    }

    protected function _initJQuery() {
        $this->bootstrap("view");
        $view = $this->getResource("view");

        ZendX_JQuery::enableView($view);

        return $view->jQuery();
    }
}
```

V souboru **application.ini** je tedy definováno vše podstatné k aplikaci na jednom místě. Jedná se především o:

- základní nastavení *PHP*
- nastavení prostředí a časové zóny
- cesta do adresáře knihovny *Zend*
- cesta k souboru `bootstrap.php`
- nastavení cest dalších složek, např. řadičů, vlastních helperů, složky s layoutem, k pluginům, k souborům a fotografiím..
- nastavení databázového adaptéru
- nastavení složek pro ukládání indexů fulltextového vyhledávače *Lucene*
- nastavení routování jednotlivých akcí, tedy která url adresa spouští kterou akci a co znamenají parametry
- nastavení navigace - slouží pro generování i víceúrovňového menu, drobečkové navigace apod.
- nastavení rolí, řadičů, a definice přístupových práv - tedy která role má přístup ke které akci
- v závislosti na nastavení prostředí aplikace zobrazí či skryje zobrazení výjimky, popis chyby či tzv. *stack trace*.

Následuje malá ukázka konfiguračního souboru. Zahrnuje nastavení databázového adaptéru, nastavení routování na detail hesla a nastavení přístupových práv pro administrátora.

```
...

resources.db.adapter = "pdo_mysql"
resources.db.params.charset = "utf8"
resources.db.params.host = "localhost"
resources.db.params.username = "root"
resources.db.params.password = "melnik"
resources.db.params.dbname = "klastery_db"
resources.db.isDefaultTableAdapter = true

...

resources.router.routes.item_show.type = "Zend_Controller_Router_Route"
resources.router.routes.item_show.route = ":controller/show/:id"
resources.router.routes.item_show.defaults.module = "default"
resources.router.routes.item_show.defaults.action = "show"
resources.router.routes.item_show.defaults.id = null
resources.router.routes.item_show.regs.controller = "(lokality|objekty|
```

```
    osobnosti)"
resources.router.routes.item_show.regs.id = "\d+"

...

acl.rules.allow.admin.error = all
acl.rules.allow.admin.index = all
acl.rules.allow.admin.items = all
acl.rules.allow.admin.lokality = all
acl.rules.allow.admin.navrhy = all
acl.rules.allow.admin.objekty = all
acl.rules.allow.admin.osobnosti = all
acl.rules.allow.admin.user.1 = change
acl.rules.allow.admin.user.2 = logout
acl.rules.allow.admin.user.3 = showall
acl.rules.allow.admin.user.4 = show
acl.rules.allow.admin.user.5 = edit
acl.rules.allow.admin.user.6 = changepassword
acl.rules.allow.admin.user.7 = showme
acl.rules.allow.admin.user.8 = listlogin
acl.rules.allow.admin.user.9 = ajaxislogged
acl.rules.allow.admin.user.10 = logoutloginredirect
acl.rules.allow.admin.clanky.1 = archiv
acl.rules.allow.admin.clanky.2 = archivbydate
acl.rules.allow.admin.clanky.3 = insert
acl.rules.allow.admin.clanky.4 = show
acl.rules.allow.admin.fotografie.1 = show
acl.rules.allow.admin.fotografie.2 = update
acl.rules.allow.admin.fotografie.3 = insert
acl.rules.allow.admin.fotografie.4 = prirazeni
acl.rules.allow.admin.fotografie.5 = showall
acl.rules.allow.admin.soubory.1 = insert
acl.rules.allow.admin.soubory.2 = update
acl.rules.allow.admin.soubory.3 = showall
acl.rules.allow.admin.messages = all
acl.rules.allow.admin.map = all
```

Dalším zásadním souborem je soubor jednotného **layoutu**. Tedy vše, co se na jednotlivých stránkách opakuje - HTML tagy `head`, `body`, hlavička stránky a menu, postranní sloupec, patička. V aplikaci jsou využity layouty dva. Jeden pro zobrazení mapové aplikace v samostatném okně, druhý pro všechny ostatní stránky. V souboru layoutu je samozřejmě možné zpětné doplnění některých řetězců, které závisí na konkrétní stránce, např. titulek stránky, připojení příslušného souboru JavaScriptu apod. Soubory layoutu se nacházejí ve složce `/application/layouts/scripts`

Další podstatné soubory doplňují knihovnu tříd `Zend` vlastními třídami a nachází

se ve složce `/library/Klastery`. Vedle této složky jsou ještě dvě - `/library/Zend` a `/library/ZendX`. První obsahuje samotné třídy ZF, druhá třídy, které nejsou vyvíjené přímo vývojáři ZF, ale s jádrem úzce souvisí.

Knihovna `Klastery` tedy obsahuje vlastní třídu `Klastery_Acl`, která definuje na základě konfiguračního souboru přístupová práva pro jednotlivé role. Dál třídu `Klastery_Form` odvozenou od třídy `Zend_Form` a definující mj. vlastní dekorátory formulářů a propojení formuláře s modelem.

Také je zde definováno několik *Action Helperů* - tedy vlastních tříd, které obsahují opakující se části kódu v akcích. Pokud třída helperu obsahuje metodu `direct()`, tato metoda se spustí, pokud je v akci možné tuto funkci volat jménem helperu, viz např:

```
Zend_Controller_Action_HelperBroker::getStaticHelper('TextViewMarkNotes');
$osobnostBiografie = $this->_helper->TextViewMarkNotes($dataOsobnost[
    'osobnost_biografie'], 0);
```

V ukázce je použití helperu pro zobrazení poznámek pod čarou v textu hesla. Jsou použity tyto vlastní helpery:

- **Helper_EncryptPassword** - asymetricky šifruje zadané heslo za pomoci tzv. *soli*. Je použito hešovací funkce *SHA1* a helper je využit při každém zadání hesla - ať už při přihlášení, změně přihlašovacího hesla, mazání hesla (myšleno článku) nebo registrace nového uživatele.

- **Helper_ImageMerge** - helper spojí několik obrázků do jednoho - vstupními parametry funkce je počet obrázků v jednom a druhém směru a pole s adresami obrázků. Helper využívá *GD knihovnu PHP*, a sám je využitý při sestavení vlastního výřezu mapy.

- **Helper_InMultiArray** - funkce v helperu slouží k prohledávání vícerozměrného pole.

- **Helper_SetNewFile** - funkce slouží k uložení nově nahraného souboru, jeho přejmenování a s ním spojené zařazení do databáze.

- **Helper_SetNewPhoto** - funkce slouží k uložení nově nahrané fotografie, jejímu přejmenování a s ním spojenému zařazení do databáze. Zároveň vytvoří kopii fotografie velikosti náhledu a tu uloží do příslušné složky.

- **Helper_SortByNameAscii** - funkce řadí pole podle vnitřního pole. Zároveň správně seřadí i řetězce s diakritikou a znaky jiných abeced a to tak, že nejprve podle pole definovaného v externím souboru převede všechny znaky na znaky tabulky *ASCII*, poté řadí klasickým způsobem voláním callback funkce.

- **Helper_SortByValue** - podobný případ jako helper výše, řadí však i podle druhého pole, pokud se hodnoty v prvním shodují.

- **Helper_TextCancelIncompleteLink** - funkce smaže neuzavřené závorky s odkazem (viz kap. 3.2.1.1), odkaz samotný smaže také - text tedy ponechá samotný. Použito např. v *infoWindow* v mapové aplikaci, kde je k popisu lokality použito prvních n (např. 30) slov hlavního textu hesla.

- **Helper_TextCancelNotes** - funkce smaže neuzavřené závorky s poznámkami, podobně jako v helperu výše.

- **Helper_TextViewMarkLink** - funkce projde daný text a změní odkaz psaný vlastní zjednodušenou syntaxí na odkaz HTML, tedy link `<a></a>` (viz kap. 3.2.1.1).

- **Helper_TextViewMarkNotes** - funkce také projde daný text, najde poznámky psané vlastní syntaxí, vrátí text s URL odkazy na poznámky a k tomu pole s poznámkami - výsledek je podobný jako u hesel na Wikipedii (viz kap. 3.2.4.6).

Také jsou zde dva pluginy, třídy `Klastery_Controller_Plugin_ActionSetup` a `Klastery_Controller_Plugin_Auth`. První slouží mj. ke spuštění akce postranní lišty a nastavení expirace přihlášení uživatele. Druhý plugin kontroluje, zda má daný uživatel práva na danou akci. Pokud ne, provede přesměrování na jinou akci.

Třída `Klastery_Loader_ModelLoader` zařídí načtení modulů (moduly v tomto případě využity nejsou).

Interface `Klastery_Model_Interface` deklaruje základní metody `getStorage()`, `insert()`, `update()`, `delete()`, `fetchEntry()`, `fetchEntries()`. Ty jsou potom definovány v jednotlivých modelech. Třída `Klastery_Model_Abstract`, potomek interface `Klastery_Model_Interface`, je abstraktní třída, která definuje metody mj. `loadModel()`, `loadForm()`, `getForm()` ad. Z třídy `Klastery_Model_Abstract` je odvozená další třída `Klastery_Model_Database`, která jak definuje např. metody `insert()`, `update()`, tak deklaruje metody `_preInsert()` a `_postInsert()` ad., což jsou vlastně triggery definované v odvozených modelech. Další třídou je `Klastery_Model_Exception` odvozená od `Zend_Exception`, obsluhující výjimky.

V neposlední řadě je ve složce `/library/Klastery` několik vlastních validátorů, tedy tříd odvozených od třídy `Zend_Validate_Abstract` a obstarávajících validaci zejména formulářů. Jedná se o tyto validátory:

- **Klastery_Validate_ChangePassword** - validátor pro změnu hesla. Pokud se zadává nové heslo, musí být pro kontrolu zadáno dvakrát stejně.

- **Klastery_Validate_ChangeUserName** - validátor kontroluje, zda je zadané i heslo při změně uživatelského jména. Tento validátor již není využit a pravděpodobně bude odstraněn.

- **Klastery_Validate_FileExist** - validátor zkontroluje, zda daný soubor (či obrázek) již na serveru není nahraný, byť pod jiným názvem.

- **Klastery_Validate_FileNameRequired** - validátor zkontroluje, zda je zadán název souboru či obrázku - pokud je název povinný.

- **Klastery_Validate_IfCheckedNotNull** - validátor kontroluje, zda je jeden určitý textový element formuláře vyplněn - avšak jen pokud je jiný element - checkbox - zaškrtnutý.
- **Klastery_Validate_MultiInArray** - validátor kontroluje, zda je v selectu vybrán některý z daných prvků pole.
- **Klastery_Validate_NarozeniUmrti** - validátor kontroluje, zda není u osobností rok narození větší než rok úmrtí.
- **Klastery_Validate_RequiredDependSelect** - validátor kontroluje, zda jsou ve formuláři nového návrhu správně vybrány všechny potřebné kategorie.
- **Klastery_Validate_UniqueUserName** - validátor kontroluje, zda dané uživatelské jméno již neexistuje. Použit např. při registraci nového uživatele, nebo změně údajů.
- **Klastery_Validate_Url** - validátor s pomocí regulárního výrazu kontroluje, zda je (např. ve formuláři) zadaná URL adresa platná.
- **Klastery_Validate_VznikRok** - validátor kontroluje, zda je rozmezí let zadáno správně, tedy zda jsou obě hodnoty vyplněny a zda hodnota *do* je větší než hodnota *od*.

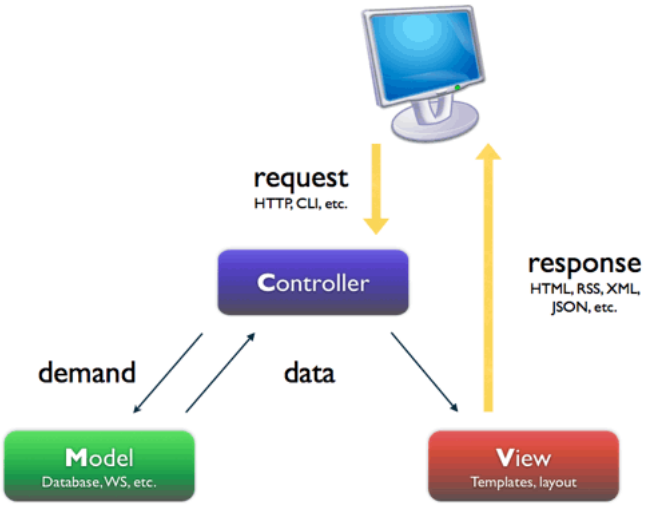
Poznamenejme, že veškerá validace formulářů samozřejmě probíhá jak na straně serveru, tak na straně klienta. Na straně klienta (v JavaScriptu) probíhá, aby klient na server zbytečně neodesílal neplatné údaje, na druhou stranu se tato validace dá snadno obejít (např. jen vypnutím JavaScriptu v prohlížeči), musí tedy probíhat druhá validace na straně serveru. A validátory, které nejsou již připravené vývojáři ZF, bylo potřeba implementovat - což je tento případ.

4.1.2.2 MVC architektura aplikace

Controller, neboli *řadič* je v MVC aplikaci třída, která reprezentuje nějakou obecnou entitu - např. uživatele - a s pomocí *akcí*, tedy metod této třídy řídí aplikaci - tedy složce *view (pohled)* určí, co má zobrazit. Použije k tomu *modely*, tedy třetí složku, která se stará o data - např. komunikuje s databází. Nemusí ale jít jen o databázi, ale např. indexované výsledky fulltextového vyhledávání, formuláře, apod. Také není nutné model nebo pohled při požadavku využít. Řadič se obejde i bez modelu (např. při požadavku zobrazení statické stránky) i bez pohledu (např. při odhlášení, kdy dojde v průběhu vykonání akce k přesměrování na jinou akci).

4.1.2.3 Řadiče s akcemi a pohledy (view)

Řadičem tedy může být zmiňovaný *uživatel*, akcí potom např. *zobrazení detailu uživatele*, nebo např. odhlášení. Akce ale nemusí být jen spojena se zobrazením určité webové stránky. Akcí může být např. zobrazení postranní lišty, indexování fulltextového vyhledávače, nebo různé AJAXové pořadavky. Většina akcí má své odpovídající *view* soubory, což jsou PHP soubory ve formě HTML s vloženým PHP kódem. Mají svou



Obrázek 4.2: Znázornění principu MVC architektury - zdroj [10]

adresu danou názvem řadiče a koncovku *.phtml*. To, že aplikace propojí podle názvu správnou akci se správným souborem pohledu, zařídí komponenta `Zend_Loader`.

Prvním krokem bylo samozřejmě určení entit, které by mohl dobře reprezentovat řadič, a následně určení akcí. Takové entity se částečně překrývají s hlavními tabulkami v databázi, ale také s databází vůbec souviset nemusí, jako např. `Index` nebo `Error`. Následuje přehled všech 13 řadičů a stručný popis jednotlivých akcí aplikace.

- **ClankyController** - řadič článků na nástěnce
  - *archivAction()* - akce pro zobrazení archivu starších článků z nástěnky
  - *archivbydateAction()* - akce pro zobrazení archivu starších článků z nástěnky s parametry datumu (*od* - *do*)
  - *insertAction()* - akce pro vložení nového článku na nástěnku
  - *removeAction()* - akce pro smazání článku - *neexistuje view*
  - *showAction()* - akce pro zobrazení detailu článku vč. diskuze
- **ErrorController** - řadič obsluhující výjimky
  - *errorAction()* - akce pro zobrazení chyby
- **FotografieController** - řadič pro fotografie
  - *insertAction()* - akce pro vložení nové fotografie pomocí speciálního formuláře
  - *prirazeniAction()* - akce pro přiřazení fotografií k danému heslu
  - *showAction()* - akce pro zobrazení detailu fotografie
  - *showallAction()* - akce pro zobrazení všech fotografií
  - *updateAction()* - akce pro úpravu údajů o fotografii

- **IndexController** - řadič pro hlavní stranu aplikace, boční lištu, a další všeobecné akce
  - *ajaxguestAction()* - (akce pro AJAX) - vrací informaci pro nepřihlášeného uživatele v případě jakéhokoliv AJAXového požadavku
  - *indexAction()* - akce pro zobrazení hlavní strany pro přihlášeného uživatele
  - *publicindexAction()* - akce pro zobrazení hlavní strany pro nepřihlášeného uživatele
  - *sidebarAction()* - akce pro zobrazení postranní lišty
  - *userbarAction()* - akce pro zobrazení informací o aktuálně přihlášeném uživateli v horní liště
  - *usermessagesAction()* - akce pro zobrazení panelu přehledu zpráv
  - *usernoticeAction()* - akce pro zobrazení panelu s upozorněními pro přihlášeného uživatele
- **ItemsController** - řadič pro hesla obecně - tedy zahrnující lokality, artefakty i osobnosti
  - *ajaxaddlinkAction()* - (akce pro AJAX) - vrátí soubor elementů pro přidání hypertextového odkazu
  - *ajaxjoinfotografieAction()* - (akce pro AJAX) - vrátí element, umožňující propojení hesla s dalšími fotografiemi - vč. náhledů fotografií
  - *buildAction()* - akce pro vytvoření indexů všech hesel pro fulltextové vyhledávání
  - *searchfulltextAction()* - akce pro fulltextové vyhledání hesel
  - *showAction()* - akce pro zobrazení všech neschválených hesel
  - *showapprovedAction()* - akce pro zobrazení hotových a schválených hesel
  - *showfoundAction()* - akce pro zobrazení všech hesel vyhledaných s pomocí rychlého hledání v postranním panelu
  - *showupdatesAction()* - akce pro zobrazení historie všech úprav daného hesla
- **LokalityController** - řadič pro lokality
  - *ajaxadddcerinyklasterAction()* - (akce pro AJAX) - vrátí soubor elementů pro napojení / přidání dceřiného kláštera
  - *ajaxaddfundatorAction()* - (akce pro AJAX) - vrátí soubor elementů pro napojení / přidání fundátora
  - *ajaxaddmaterskyklasterAction()* - (akce pro AJAX) - vrátí soubor elementů pro napojení / přidání mateřského kláštera
  - *ajaxaddopatAction()* - (akce pro AJAX) - vrátí soubor elementů pro napojení / přidání představeného
  - *ajaxaddsvetecAction()* - (akce pro AJAX) - vrátí soubor elementů pro napojení / přidání světce

- *ajaxgetopatchronologieAction()* - (akce pro AJAX) - vrací ve formátu JSON roky působení daných představených
- *ajaxsearchhelperAction()* - (akce pro AJAX) - vrací ve formátu JSON upřesnění voleb do formuláře hledání podle parametrů (zužování nabídky)
- *buildAction()* - akce pro vytvoření indexů lokalit pro fulltextové vyhledávání - v současné době nevyužito
- *changeuserAction()* - akce pro změnu uživatele, který má danou lokalitu odemčenou pro úpravy
- *createAction()* - akce pro zobrazení stránky s formulářem pro vytvoření nové lokality
- *searchAction()* - akce pro vyhledání lokalit dle zadaných kritérií
- *searchfulltextAction()* - akce pro vyhledání lokalit s pomocí fulltextu - v tuto chvíli nevyužito, nahrazeno hledáním mezi všemi hesly
- *showAction()* - akce pro zobrazení detailu lokality
- *unlockAction()* - akce pro odemčení již uzavřeného hesla lokality danému uživateli
- *updateAction()* - akce pro zobrazení formuláře úpravy hesla lokality

- **MapController** - řadič pro mapu
  - *ajaxgetFiliaceAction()* - (akce pro AJAX) - akce vrací ve formátu JSON případné filiační vazby mezi danými lokalitami - ve fázi vývoje
  - *ajaxgetlokalityAction()* - (akce pro AJAX) - akce vrací ve formátu JSON všechny lokality, které mají zadané souřadnice, vč. potřebných údajů.
  - *ajaxmakeimgfrommaptilesAction()* - (akce pro AJAX) - akce vrací odkaz na obrázek sestavený z daných mapových výřezů - ve fázi vývoje
  - *showAction()* - akce pro zobrazení zjednodušené mapy
  - *showmaximizedAction()* - akce pro zobrazení plnohodnotné mapové aplikace - akce má přiřazený speciální layout

- **MessagesController** - řadič pro zprávy mezi uživateli
  - *insertAction()* - akce pro přidání nové zprávy
  - *showAction()* - akce pro zobrazení jedné zprávy
  - *showsentAction()* - akce pro zobrazení již odeslané zprávy

- **NavrhyController** - akce pro návrhy (nultý stav hesla)
  - *createAction()* - akce pro vytvoření nového návrhu
  - *showAction()* - akce pro zobrazení všech návrhů

- **ObjektyController** - řadič pro artefakty

- *changeuserAction()* - akce pro změnu uživatele, který má daný artefakt odemčenou pro úpravy
- *createAction()* - akce pro zobrazení stránky s formulářem pro vytvoření nového artefaktu
- *searchAction()* - akce pro vyhledání artefaktů dle zadaných kritérií
- *showAction()* - akce pro zobrazení detailu artefaktu
- *unlockAction()* - akce pro odemčení již uzavřeného hesla artefaktu danému uživateli
- *updateAction()* - akce pro zobrazení formuláře úpravy hesla artefaktu

- **OsobnostiController** - řadič pro osobnosti
  - *ajaxaddanotherpusobeniAction()* - (akce pro AJAX) - akce vrací soubor elementů pro působení představeného
  - *ajaxaddpusobeniAction()* - (akce pro AJAX) - akce vrací soubor elementů pro působení představeného (použito v jiném formuláři)
  - *ajaxgetlokalitachronologieopatuAction()* - (akce pro AJAX) - akce vrací ve formátu JSON roky působení představených v daných lokalitách
  - *ajaxselectpredstaveniAction()* - (akce pro AJAX) - akce vrací select element s již přiřazenými představenými k dané osobnosti (slouží k chronologickému zařazení dalšího představeného)
  - *changeuserAction()* - akce pro změnu uživatele, který má danou osobnost odemčenou pro úpravy
  - *createAction()* - akce pro zobrazení stránky s formulářem pro vytvoření nové osobnosti
  - *searchAction()* - akce pro vyhledání osobností dle zadaných kritérií
  - *showAction()* - akce pro zobrazení detailu osobnosti
  - *unlockAction()* - akce pro odemčení již uzavřeného hesla artefaktu danému uživateli
  - *updateAction()* - akce pro zobrazení formuláře úpravy hesla artefaktu

- **SouboryController** - řadič pro soubory
  - *insertAction()* - akce pro přidání nového souboru
  - *showallAction()* - akce pro zobrazení přehledu všech souborů
  - *updateAction()* - akce pro úpravu informací o souboru

- **UserController** - řadič pro uživatele
  - *ajaxisloggedAction()* - (akce pro AJAX) - akce vrací informaci o tom, zda je uživatel na serveru ještě přihlášen

- *blokovaniAction()* - akce pro nastavení blokování uživatele
- *changeAction()* - akce pro změnu uživatele
- *changepasswordAction()* - akce pro změnu vlastního hesla uživatele
- *editAction()* - akce pro editaci údajů uživatele
- *editpravaAction()* - akce pro editaci práv administrátora
- *editroleAction()* - akce pro změnu role uživatele
- *listloginAction()* - akce pro zobrazení seznamu přihlášení daného uživatele
- *loginAction()* - akce pro přihlášení (login) uživatele
- *logoutAction()* - akce pro odhlášení uživatele - *neexistuje view*
- *logoutloginredirectAction()* - akce pro odhlášení uživatele, zároveň přesměrování na stránku s přihlášením uživatele - *neexistuje view*
- *registrationAction()* - akce pro registraci nového uživatele hlavním administrátorem
- *setnewpasswordbysuperadminAction()* - akce pro nastavení nového hesla hlavním administrátorem
- *showAction()* - akce pro zobrazení detailních informací o daném uživateli
- *showallAction()* - akce pro zobrazení přehledu všech uživatelů
- *showmeAction()* - akce pro zobrazení informací o právě přihlášeném uživateli - v akci přesměrování na akci `showAction()` s parametrem. *Neexistuje view*
- *superadmineditAction()* - akce pro úpravu informací o uživateli hlavním administrátorem

Následuje ukázka akce pro přidání nového souboru. Formulář je definován jako třída v jiném souboru (viz kap. 4.1.2.4). Pro samotné fyzické uložení souboru je použit helper `Helper_SetNewFile`.

```
// akce na pridani noveho souboru
public function insertAction() {
    $modelSoubory = Klastery_Model_Database::loadModel('soubory');
    $modelUsers = Klastery_Model_Database::loadModel('users');

    $form = $modelSoubory->getForm('insert');
    $id_user = Zend_Auth::getInstance()->getIdentity()->user_id;

    // ma uzivatel blokovane zakladani novych prispevku (a tim i nahravani
    // souboru)?
    $blokovaneZalozeni = $modelUsers->getBlokovanizakladaniByUserId($id_user);

    if ($this->getRequest()->isPost() && false !== $this->getRequest()
        ->getPost('soubor_submit_insert', false)) {
```

```
        if ($form->isValid($this->getRequest()->getPost())) {

            $upload = new Zend_File_Transfer_Adapter_Http();
            $upload->setDestination('./files/pdf');

            if ($upload->receive()) {

                Zend_Controller_Action_HelperBroker::
                    getStaticHelper('SetNewFile');
                $souborId = $this->_helper->SetNewFile($form, NULL,
                    $upload, NULL);
            }

            $this->_redirect('/');
        }
    }

    $this->view->form = $form;
    $this->view->blokovane_zalozeni = $blokovaneZalozeni;

    $this->view->action = 'insert';
    $this->view->controller = 'soubory';
}
```

Zde je potom odpovídající *view*:

```
<h1>Přidání nového souboru</h1>
<?php if ($this->blokovane_zalozeni == FALSE) { ?>
    <?php print $this->form->render(); ?>
<?php } else { ?>
    Funkci přidávání souborů máte hlavním administrátorem blokovanou!
        Kontaktujte jej prosím.
<?php } ?>
```

4.1.2.4 Modely

Třídy modelů můžeme v zásadě rozdělit na 4 skupiny.

1. Třídy reprezentující jednotlivé tabulky v databázi, vč. definic cizích klíčů. Např. `Model_Db_Clanky`.

2. Třídy samotných modelů. Modely z velké části odpovídají tabulkám v databázi. Výjimkou jsou např. propojovací tabulky, které pochopitelně nemají svůj odpovídající model, naopak existuje model `Model_Items`, který zahrnuje lokality, artefakty i osobnosti společně.

3. Třídy reprezentující formuláře. Každý formulář je zařazen v definici určitého modelu, obsahuje vlastní filtry, validátory i dekorátory a větší formuláře mají elementy seskupeny do tzv. *DisplayGroup*. Formulářů je použito v aplikaci celkem 48. Příkladem může být formulář / třída s názvem `Model_Form_OsobnostUpdate`.

4. Třídy reprezentující dokumenty pro fulltextové vyhledávání, např. `Model_Document_Lokality`

Následuje ukázka z první skupiny tříd, třída `Model_Db_Messages`, která reprezentuje tabulku zpráv v databázi. Vlastnosti třídy popisují název tabulky, primární klíč a cizí klíče:

```
<?php
class Model_Db_Messages extends Zend_Db_Table_Abstract {

    protected $_name = 'messages';
    protected $_primary = 'message_id';
    protected $_referenceMap = array(
        'Model_Db_Users_From' => array(
            'columns' => 'message_from_user_id',
            'refTableClass' => 'Model_Db_Users',
            'refColumns' => 'user_id'
        ),
        'Model_Db_Users_To' => array(
            'columns' => 'message_to_user_id',
            'refTableClass' => 'Model_Db_Users',
            'refColumns' => 'user_id'
        ),
        'Model_Db_Reply' => array(
            'columns' => 'message_reply_to_id',
            'refTableClass' => 'Model_Db_Messages',
            'refColumns' => 'message_id'
        ),
    );
}
```

Těchto tříd je stejný počet, jako tabulek databáze - tedy 53.

Ukázkou z druhé skupiny tříd je jednoduchá metoda, která vrátí počet příspěvků diskuze k danému článku na nástěnce:

```
public function fetchCountByClanekId($clanekId) {
    $db = $this->getStorage()->getAdapter();
    $select = $db->select();
    $select->from('diskuzeclanky', array('COUNT(*)'));
    $select->where('diskuzeclanky_clanek_id = ?', $clanekId);
    return $db->fetchOne($select);
}
```

Těchto tříd v aplikaci existuje celkem 32, tedy téměř všechny souvisí s odpovídající tabulkou v databázi. V modelu jsou definovány formuláře, odpovídající tabulka, příp. definovány tzv. háky - tedy metody, které se spouští např. zavoláním rodičovské metody `parent::insert($data);` (viz kap. 4.1.2.1) apod. Takovým příkladem je metoda `_preInsert()`. Metod je v některých třídách i více než 130 - např. v modelu lokalit.

Formulářů, jak už bylo řečeno, je v aplikaci využito celkem 48 - od jednoduchých přihlašovacích až po komplexní formuláře, pomocí nichž se např. zakládá nové heslo. Každý formulář je ve vlastní třídě a jeho definice probíhá v metodě `init()`, např.:

```
class Model_Form_LokalitaCreate extends Klastery_Form {

    public function init() {
/*
definice formulare
*/
    }
}
```

Nejprve se s pomocí příslušných modelů získají data do formuláře, např.:

```
...
$svetciOptions = $this->getModel()->fetchSvetci();
...
asort($svetciOptions);
...
```

...poté se vytvoří jednotlivé elementy, např.:

```
$lokalitaSvetecHlavniSelect = new Zend_Form_Element_Select('lokalita_
   svetec_hlavni_select', array(
    'size' => '1',
    'required' => FALSE,
    'maxlength' => 100,
    'multiOptions' => $selectImplicitniVolbaSvetec + $svetciOptions,
    'label' => 'Vybrat:',
    'class' => 'lokalita_svetec_hlavni_select',
    'id' => 'lokalita_svetec_hlavni_select',
));
```

...přířadí připravené i vlastní validátory a filtry:

```
...
$lokalitaImage->addValidator('Extension', false, 'jpg,png,gif,jpeg');
$lokalitaImage->addValidator(new Klastery_Validate_FileExist('image'));
...
$lokalitaDalsinazvy->addFilter('StringTrim', ';');
...
```

...jednotlivé elementy přidají do samotného formuláře, např.:

```
...
$this->addElements(array(
    ...
    $lokalitaName,
    ...
));
```

... seskupí do tzv. groupboxů, např.:

```
$this->addDisplayGroup(array(
    'lokalita_hist',
    'lokalita_stavebnivyvoj',
    'lokalita_prameny',
    'lokalita_literatura',
        ), 'texty_group', array('legend' => 'Texty'));
```

...a nakonec přiřadí tzv. dekorátory, definované v rodičovské třídě `Klastery_Form`:

```
...
$lokalitaSvetecHlavniRB->setDecorators($this->radioElementDecorators);
...
```

Dekorátory slouží k popisu toho, jak se má formulář vygenerovat, tedy např. který HTML tag má obalit zmiňované groupboxy, jaké má mít CSS třídy, apod.

Dokumenty pro indexování a následné rychlejší vyhledávání ve fulltextu jsou třídy, které se skládají z jednotlivých polí vytvářených z jednotlivých sloupců tabulky z databáze. Pole zahrnují ID hesel, hlavní a vedlejší názvy a hlavní text. Samotné indexování i vyhledávání obstarává třída `Zend_Search_Lucene`, což je vyhledávací engine k tomu určený. Zde je ukázka dokumentu `Model_Document_Objekty`, který umožňuje indexování artefaktů:

```
<?php
class Model_Document_Objekty extends Zend_Search_Lucene_Document {

    public function setData(array $data) {

        $this->addField(Zend_Search_Lucene_Field::keyword(
                        'key', $data['objekt_id'], 'utf-8'
        ));

        $this->addField(Zend_Search_Lucene_Field::text(
                        'name', $data['objekt_name'], 'utf-8'
        ));
```

```
        $this->addField(Zend_Search_Lucene_Field::text(
                        'other_name', $data['other_name'], 'utf-8'
        ));

        $this->addField(Zend_Search_Lucene_Field::unStored(
                        'popis', $data['objekt_popis'], 'utf-8'
        ));
    }
}
```

4.1.2.5 Ostatní prvky aplikace

Přístupová práva pro jednotlivé role (autorizaci) obstarává komponenta `Zend_ACL`, autentizaci potom komponenta `Zend_Auth`.

4.1.3 Využití JavaScriptu

V okamžiku, kdy základní aplikace napsaná jen v PHP za pomoci ZF fungovala, bylo možné postupně zapojovat také JavaScript, který má za úkol práci v systému ulehčit zejména na straně klienta. Bylo v maximální míře využito javascriptové knihovny *JQuery*, která je velmi rozšířená a má širokou podporu prohlížečů. Hlavním využitím samotné knihovny je zejména interakce mezi HTML a JavaScriptem, jehož syntaxi tím velmi zjednodušuje. Projevuje se to např. u AJAXových požadavků. Zde je na ukázku porovnání jednoduchého volání AJAXového požadavku v samotném JavaScriptu:

```
// Initialize the Ajax request
var xhr = new XMLHttpRequest();
xhr.open('get', 'send-ajax-data.php');

// Track the state changes of the request
xhr.onreadystatechange = function(){
    // Ready state 4 means the request is done
    if(xhr.readyState === 4){
        // 200 is a successful return
        if(xhr.status === 200){
            alert(xhr.responseText); // 'This is the returned text.'
        }else{
            alert('Error: '+xhr.status); // An error occurred during
                                         the request
        }
    }
}

// Send the request to send-ajax-data.php
xhr.send(null);
```

Zde je stejný případ, tentokrát s využitím knihovny *JQuery* a funkce `$.get`:

```
$.get('send-ajax-data.php', function(data) {
    alert(data);
});
```

Ukázka pochází ze stránek Wikipedie ([11]).

Je vidět, že práce s *JQuery* zejména usnadňuje psaní v JavaScriptu, navíc nabízí základní funkce pro animaci, efekty, manipulaci s CSS a především nepřeberné množství pluginů na této knihovně postavených. Dál je využito také související framework *JQuery UI*. Ten nabízí mnoho grafických efektů a zejména velmi užitečné widgety. Použitý je např. tzv. *multiselect* (alternativa klasického formulářového selectu, jen s možností výběru více položek najednou, formou zaškrtávacích checkboxů), *Datepicker* (pro výběr data v kalendáři) a *Slider*.

Kód JavaScriptu byl nejprve psán do jednoho souboru, který byl jednoduše zahrnut v hlavičce `head` v souboru layoutu. Jak kód JavaScriptu rostl do mnoha tisíců řádků, bylo kvůli přehlednosti i rychlejšímu načítání nezbytné jej rozdělit pro každou stránku do samostatného souboru. Nyní je tedy jeden soubor JavaScriptu, který je společný pro všechny stránky (akce), a navíc se u každé stránky načítá javascriptový soubor pojmenovaný po příslušném řadiči a akci, ve kterém je kód nutný pouze pro danou stránku.

4.1.3.1 Využití AJAX

V mnoha formulářích (a nejen formulářích) je zbytečné načítat např. element `select` s velkým množstvím lokalit, pokud je pravděpodobné, že nebude využit. Jedná se např. o přiřazení lokalit jako míst působení k představeným. A navíc, pokud má být možnost připojit k lokalitě neomezené množství představených či fundátorů, jediná rozumná možnost, jak formulář na toto připravit, je dynamické načítání dalších elementů přes *AJAX*. V takovém případě server vrátí část HTML kódu, který je následně přidán na patřičné místo ve formuláři. Jindy ale po serveru klient požaduje jen samotné, určitým způsobem strukturované informace, v takovém případě server může vrátit data např. ve formátu XML nebo JSON. Zde je ukázka využití AJAXu pro načtení několika elementů pro přidání světce do formuláře vytvoření lokality:

```
...
url = 'lokality/ajaxaddsvetec/format/html';

$.ajax({
    type: 'POST',
    url: url,
    data: {
        id: pocitadlo_svetcu
    },
    success: function(data) {
        $("#svetec_vedlejsi_" + pocitadlo_svetcu).html(data);
        ...
```

Serveru je zde odeslán požadavek na danou url, metodou POST je přidán parametr (v tomto případě číslo světce), a po úspěšném obdržení odpovědi se spustí callback funkce a zpracování dat (zde v proměnné (data)). Jiný příklad využití technologie AJAX je při načítání mapy (viz kap. 4.2.2), zde je použita i jiná funkce:

```
...
var url_json_get_lokality = 'map/ajaxgetlokality/format/json';

var jsonData = $.getJSON(url_json_get_lokality)
        .done(function(return_data) {
    $.each(return_data.lokality_list, function(key, value) {
        displayLocation(value);
    });
});

jsonData.complete(function() {
...
```

V tomto případě klient obdrží data ve formátu *JSON*, která ihned v cyklu `$.each` zpracuje.

4.1.3.2 jQuery Validation Plugin

Pro validaci formulářů na straně klienta slouží plugin *jQuery Validation Plugin*. Obsahuje sadu standardních validátorů a navíc možnost si vlastní validátory přidat a ty potom použít. Následuje ukázka vytvoření validátoru na kontrolu data, které je větší než určitá hodnota, a následné použití tohoto validátoru:

```
$.validator.addMethod("greaterThan",
        function(value, element, params) {

            if ($(params).val() === ''
                    || value === '') {
                return true;
            }

            if (!/Invalid|NaN/.test(new Date(value))) {
                return new Date(value) >= new Date($(params).val());
            }

            return isNaN(value) && isNaN($(params).val())
                    || (Number(value) >= Number($(params).val()));
        }, 'Must be greater than {0}.');

$("#clanek_date_search").validate({
    focusInvalid: false,
```

```
    rules: {
        ...
        clanek_do: {
            greaterThan: "#clanek_od",
                ...
            }
        },
        messages: {
            ...
        clanek_do: {
            greaterThan: 'Datum "Od" musí být starší než datum "Do".',
            ...
        }
    },
    errorPlacement: function(error, element) {
        $("#errors_claneksearchbydate ul").html('<li>' + error.text()
            + '</li>');
    },
    submitHandler: function(form) {
        form.submit();
    }
});
```

Handler `submitHandler` slouží k definici funkce, která se provede před samotným úspěšným odesláním formuláře. V několika případech je to využito pro kontrolu s využitím AJAXu, zda se např. nepřekrývají roky působení představeného u jiných lokalit. V takovém případě je pouze zobrazeno varování, ale systém nebrání v odeslání formuláře. Jako dialog pro varování je využitý *Dialog Widget*.

4.1.3.3 Dialog Widget

Tento dialog využívá přímo již zmiňovaný *JQuery UI CSS framework* a nabízí široké možnosti nastavení. V ukázce je jeho využití jako modálního dialogu právě při kontrole chronologie představených:

```
var def1 = $.Deferred();

$("#dialog-confirm").dialog({
     resizable: false,
     height: 200,
     width: 500,
     modal: true,
     buttons: [{
             text: "V pořádku",
             click: function() {
                 ...
```

```
                def1.resolve("true");
            },
        }, {
            text: "Opravit",
            click: function() {
                ...
                def1.resolve("false");
        },
    }],
});
```

V ukázce je také vidět řešení asynchronní operace, což ajaxový požadavek je. Slouží k tomu objekt `$.Deferred`, který zajistí, že se určitá část kódu vykoná až po nějaké úspěšné události (ve chvíli, kdy je definována určitá hodnota proměnné).

4.1.3.4 jQuery UI MultiSelect Widget

Tento widget je v aplikaci využitý mnohokrát - usnadňuje manipulaci uživatele se selectem, u kterého je možnost vybrat více položek najednou. Fungování standardního elementu select s tímto atributem totiž nemusí být pro každého hned jasný [1]. A protože widget nabízí navíc také možnost filtrování, je využitý i pro klasické elementy select. V ukázce je využití pro výběr artefaktů souvisejících s danou lokalitou:

```
$("#lokalita_objekty").multiselect({
    noneSelectedText: 'Vybrat artefakty',
    selectedList: 100,
    checkAllText: 'Vybrat vše',
    uncheckAllText: 'Zrušit vše'
}).multiselectfilter({
    placeholder: 'Zadej text',
    label: 'Filtruj:'
});
```

4.1.3.5 Image DropDown

Dalším jQuery pluginem využitým v aplikaci je tento plugin pro „drop-down“ nabídku (tedy select), ale s možností k jednotlivým položkám také zobrazovat obrázky [2]. Využitý je v případě připojení již existujících obrázků k novému heslu. Následuje ukázka:

```
$("#join_fotografie").msDropDown({
     showRows: 10,
     rowHeight: 30,
     multiple: true,
     openDirection: 'auto',
```

[1] viz příklad na http://www.w3schools.com/tags/att_select_multiple.asp
[2] viz http://www.marghoobsuleman.com/jquery-image-dropdown

```
     enableCheckbox: true,
     on: {
         change: function() {
             refresh_fotografie_selected();
         },
         open: function() {
             refresh_fotografie_selected();
         },
         click: function() {
             refresh_fotografie_selected();
         },
         close: function() {
             refresh_fotografie_selected();
         ...
     }
}});
```

4.1.3.6 Tablesorter a tablesorter.pager

Pro řazení obsahu tabulek (např. seznamy hesel, činnosti uživatelů apod.) je využit tento plugin. Pro obvyklé obsahy sloupců a obvyklé způsoby řazení plugin obsahuje standardní *parsery* (např. číslo, řazení podle abecedy bez diakritiky apod.), pro jiné sloupce však bylo potřeba vytvořit a přidat vlastní parsery. Jedná se např. o řazení data v českém formátu, řazení textu s diakritikou, nebo řazení uživatelských rolí. V následující ukázce je vytvoření parseru pro stav hesla a jeho aplikace na tabulku:

```
$.tablesorter.addParser({
    id: 'stav-hesla',
    is: function(s) {
        return false;
    },
    format: function(s)
    {
        // v pripade nevyplnenych policek
        if (s === '') {
            return 99999999;
        }

        return s.toLowerCase()
                .replace(/schváleno/, 0)
                .replace(/dokončeno/, 1)
                .replace(/rozpracováno/, 2)
                .replace(/návrh/, 3);
    },
    type: 'numeric'
});
```

```
if ($('#showupdates tbody tr').length) {
     $("#showupdates").tablesorter({
         headers: {
             0: {
                 // sorter na text s diakritikou
                 sorter: 'datum'
             },
             1: {
                 // zakazano
                 sorter: false
             },
             3: {
                 // zakazano
                 sorter: false
             }
         },
         widthFixed: true,
         widgets: ['zebra']
     });
}
```

Plugin *TablesorterPager* je potom další nadstavba pluginu *Tablesorter*, je použit u obsáhlejších tabulek a umožňuje jejich stránkování:

```
if ($('#items_showapproved tbody tr').length) {
    $("#items_showapproved").tablesorter({
        ...
    }).tablesorterPager({
        container: $("#pager"),
        positionFixed: false,
        size: 20
    });
}
```

4.1.3.7 Datepicker

Widget *Datepicker* je součástí frameworku *JQuery UI* a umožňuje do textového pole jednoduše vybrat datum v kalendáři, místo ručního psaní. Použitý je v archivu článků, při výběru článků dle data vloženi:

```
$("#clanek_od").datepicker({
    changeMonth: true,
    changeYear: true,
    numberOfMonths: 3,
    minDate: new Date(minDate), maxDate: 0,
    dayNamesMin: ["Po", "Út", "St", "Čt", "Pá", "So", "Ne"],
```

```
    monthNames: ["Leden", "Únor", "Březen", "Duben", "Květen", "Červen",
        "Červenec", "Srpen", "Září", "Říjen", "Listopad", "Prosinec"],
    monthNamesShort: ["Led", "Úno", "Bře", "Dub", "Kvě", "Črn", "Črc",
        "Srp", "Zář", "Říj", "Lis", "Pro"],
    onClose: function(selectedDate) {
        $("#clanek_do").datepicker("option", "minDate", selectedDate);
    }
});
```

4.1.3.8 Další využití JavaScriptu

Na stránce vyhledání hesla dle různých parametrů je implementován „pomocník“, který při výběru určitého parametru odfiltruje ty možnosti u dalších parametrů, při jejichž výběru by nebylo nalezeno ani jedno heslo. Podobný princip často funguje u některých internetových obchodů a hledání výrobku podle různých parametrů. V tomto případě nebylo využito žádného speciálního pluginu, ale vše bylo implementováno přímo, s pomocí AJAXu. Při každém výběru se na server odešle požadavek s informacemi o tom, které parametry jsou vybrány, PHP skript na straně serveru požadavek s použitím metod modelu zpracuje a vrátí informace o tom, které atributy výběru filtrovat (tedy nastavit neaktivní), které ponechat.

V ukázce je pouze javascriptová funkce, která se spustí po výběru nějakého vyhledávacího parametru:

```
function ajaxSearchHelper() {
    $('#search_waiting').show();

    var url_ajax_search_helper = 'lokality/ajaxsearchhelper/format/json';

    var jsonData = $.getJSON(url_ajax_search_helper, {
        typlokality: $('#lokalita_search_typlokality_id').val(),
        pohlavi: $('#lokalita_search_pohlavi').val(),
        funkcnost: $('#lokalita_search_funkcnost_id').val(),
        svetec: $('#lokalita_search_svetec').val(),
        fundator: $('#lokalita_search_fundator_id').val(),
        souvisejici_objekt: $('#lokalita_search_souvisejici_objekt').val(),
        souvisejici_osobnost: $('#lokalita_search_souvisejici_osobnost')
            .val(),
        souvisejici_lokalita: $('#lokalita_search_souvisejici_lokalita')
            .val(),
        predstaveny: $('#lokalita_search_opati').val()
    })
    .done(function(return_data) {
        processTheFilters(return_data);

        checkDisabledSelects('lokalita_search');
    })
```

```
    .success(function() {
        $('#search_waiting').hide();
    });
}
```

Dále je v aplikaci JavaScript hojně využit v mapové nadstavbě, o tom ale až v další kapitole.

4.2 Mapová část aplikace

Mapová nadstavba databáze přímo využívá *Google Maps Javascript API* a týká se především lokalit. Pro využití tohoto API je nejprve potřeba vygenerovat klíč [3], a ten pak použít v odkazu na externí skript API od Google Maps v tagu:

```
<script type="text/javascript" src="http://maps.googleapis.com/maps/api/
    js?key=API_KEY&sensor=false&libraries=places"></script>
```

Tím je prakticky zprovozněno celé mapové API.

Těžištěm mapové aplikace je samozřejmě mapa zobrazená v celé ploše prohlížeče, tedy s vlastním layoutem. Avšak nejprve probíhá interaktivní zadávání polohy lokalit.

4.2.1 Uložení polohy, reversní geocoding

To se odehrává ve formuláři přidání / úpravy hesla (lokality) a je využito vlastnosti `draggable: true` objektu `Marker`. Po kliknutí na příslušný button se teprve souřadnice markeru uloží do skrytého elementu ve formuláři a později s formulářem odešlou serveru. Zajímavější je zde ale využití tzv. *reversního geokódování (reverse geocoding)*. Současně se zmiňovaným kliknutím je odeslán ajaxový požadavek na server Google[4] s parametry souřadnic markeru. Je tak využito dalšího API Googlu, a to *Google Geocoding API*. Jednou ze základních podmínek použití tohoto API rozhraní (kromě limitu 2 500 požadavků denně) je ta, že musí být zároveň dané souřadnice zobrazeny v mapě [13], což se také samozřejmě děje. Server vrátí data ve formátu JSON, obsahující vyčerpávající informace o geografické poloze bodu, adrese, nejbližším sídle apod. Podrobnosti jsou k nalezení v oficiální dokumentaci ([12]), v aplikaci se však data projdou v několikanásobném cyklu `$.each` a zobrazí:

- stát
- administrativní dělení
- obec či sídlo

Sídel / obcí a administrativních regionů, ve kterých značka leží, může být logicky hned několik. Tyto údaje je možné ještě upravit a jsou poté na server odeslány spolu

[3] zde: https://code.google.com/apis/console

[4] http://maps.googleapis.com/maps/api/geocode/

se souřadnicemi a zbytkem formuláře. V databázi se tyto údaje u lokalit ukládají a slouží např. k následnému filtrování lokalit. V následující ukázce je část kódu s událostí kliknutí na button *Ulož pozici* a obsahuje zkráceně výše popsaný proces:

```
$('#save_position').bind("click", function() {
    var lat = homeMarker.getPosition().lat();
    var lon = homeMarker.getPosition().lng();

    $("#lokalita_lat").val(lat);
    $("#lokalita_lon").val(lon);

    var url = "http://maps.googleapis.com/maps/api/geocode/json?latlng="
        + lat + "," + lon + "&sensor=false";

    $.getJSON(url, {
        format: "json"
    }).done(function(json) {
        ...
        var count_of_administrative_area = 0;

        if (json.status == 'OK') {
            $.each(json.results, function(key, value) {
                $.each(value.address_components, function(key, value) {
                    if (value.types.indexOf("sublocality") >= 0) {
                        locality_sublocality = value.long_name;
                    } else if (value.types.indexOf("locality") >= 0) {
                        locality_locality = value.long_name;
                    } else if (value.types.indexOf("country") >= 0) {
                        locality_country = value.long_name;
                    } else if (value.types.indexOf(
                                "administrative_area_level_1") >= 0) {
                        if (locality_administrative_area_level_1 === '') {
                            locality_administrative_area_level_1
                                = value.long_name;
                            count_of_administrative_area++;
                        }
...
                });
            });
        }
        ...
    });
});
```

V detailu každého hesla lokality, které má zadanou polohu, je potom jednoduchá mapa s jedním markerem, který má již parametr `draggable: false`, tedy není možné

jej přemístit.

4.2.2 Interaktivní mapa

Jak už bylo řečeno, hlavní mapa je jádrem a těžištěm celé mapové nadstavby a má vlastní layout `layoutmapmaximized.phtml`. Jedná se o akci `showmaximized` řadiče `map`. Po zavolání akce a načtení DOM struktury stránky se ihned zadá ajaxový požadavek na geograficky lokalizované lokality. Server vrátí data ve formátu *JSON*, která jsou následně v cyklu zpracována. To však již bylo popsáno v kap. 4.1.3.1. U každé lokality je tak vytvořen tzv. *marker*, *infoWindow*, tyto prvky jsou propojeny a připojeny do mapy. Jednotlivé markery jsou zároveň ukládány do pole a protože se jedná o objekty, lze u nich také evidovat i další informace, než jen ty povinné. Eviduje se tak např. stát, roky založení či typ lokality. Dle typu lokality se navíc k markeru připojí příslušná ikona. Mapa samotná má mírně upravený mapový klíč - číselné popisy silnic a dálnic jsou zneviditelněny:

```
var roadAtlasStyles = [
    {
        featureType: 'road.highway',
        elementType: 'labels',
        stylers: [
            {visibility: 'off'}
        ]
    },
    {
        featureType: 'road.arterial',
        elementType: 'labels',
        stylers: [
            {visibility: 'off'}
        ]
    },
];

var styledMapOptions = {
    name: 'Roads without labels'
};

var roadMapType = new google.maps.StyledMapType(
        roadAtlasStyles, styledMapOptions);

map.mapTypes.set('roadatlas', roadMapType);
map.setMapTypeId('roadatlas');
```

4.2.2.1 MarkerWithLabel

Běžný marker je objekt třídy `google.maps.Marker`. V tomto případě však byla využita rozšířená třída `MarkerWithLabel`, která je rozšířením té zmiňované. Je součástí open

source knihovny *google-maps-utility-library-v3* [5]. Třída umožňuje generovat k lokalitám popisky, což může být výhodné při tisku mapy. V tomto případě se tedy u markerů evidují ještě údaje o těchto popiscích. Navíc také logické hodnoty, zda je lokalita v nějakém parametru filtrována (viz kap. 4.2.2.2) či ne. Následuje právě ukázka definice markeru:

```
var marker = new MarkerWithLabel({//set marker
    position: latLng, //set marker position coordinates
    map: map, //set map to be used by the marker
    draggable: false, //make the marker draggable
    title: locality.lokalita_name,
    icon: icon,
    class: 'typ' + locality.lokalita_typlokality_id,
    pz: pz_type, //locality.lokalita_prvnizminka,
    rok_pz: locality.prvni_zminka, // prvni zminka
    za: za_type, //locality.lokalita_zalozeni,
    rok_za: locality.zalozeni, // zalozeni
    id: locality.lokalita_id, // id lokality / markeru
    label_distance: 10, // vzdalenost popisu od bodu
    label_position: 'or', // implicitni pozice popisu
    filter_typlokality: true, // je lokalita ve filtru typu lokalit
                                zobrazena (true - je, false - neni)
    filter_datace: true, // filtr zobrazeni podle let
    filter_geografie: true, // filtr geograficky
    stat: locality.lokalita_stat, // stat
    adregion: locality.lokalita_admreg_name, // administrativni region
    place: locality.lokalita_place_name, // sidlo

    labelText: '',
    labelContent: locality.lokalita_name,
    // the CSS class for the label
    labelAnchor: new google.maps.Point(22, 0),
    labelClass: "maplabels" + locality.lokalita_id + " maplabels",
    labelStyle: {
        opacity: 0.75,
        padding: 20,
    },
    labelInBackground: false,
    labelVisible: false,
});

markers.push(marker);
```

Jak je vidět, jeden z parametrů popisu (konkrétně `labelAnchor`) je vztažný bod, tedy poloha vzhledem k bodu. Protože se markery ukládají do pole, lze tento údaj u jednotlivých lokalit kdykoliv měnit. V následující ukázce je událost, kdy uživatel klikne

[5] viz https://code.google.com/p/google-maps-utility-library-v3/

na tlačítko, které má popis posunout vpravo nahoru. Zmiňme ještě, že hodnotu `label_distance` lze také měnit a že se jedná o vzdálenost popisu od bodu.

```
$("div.set_label_position_div .rightup").live("click", function() {
    var iw_id = $(this).closest(".infoWindow").attr('rel');

    for (var i = 0; i < markers.length; i++) {
        if (markers[i].get("id") == iw_id) {
            var label_distance = parseInt(markers[i].get("label_distance"));
            var height = $("div.maplabels" + iw_id).innerHeight();
            markers[i].set("labelAnchor", new google.maps.Point(
                -label_distance, height + label_distance));
            markers[i].set("label_position", "ru");
            break;
        }
    }
});
```

Stejným způsobem lze text popisu měnit:

```
$("#label_text").live("keyup", function() {
    var iw_id = $(this).closest(".infoWindow").attr('rel');

    for (var i = 0; i < markers.length; i++) {
        if (markers[i].get("id") == iw_id) {
            markers[i].set("labelContent", $(this).val());
            markers[i].set("labelText", $(this).val());
        }
    }
});
```

Shrňme si tedy marker, infoWindow, a jejich vzájemné propojení s mapou:

```
function displayLocation(locality) {
    ...
    var content = '<div class="infoWindow" rel="' +
                  locality.lokalita_id + '">'
            + '<h2 class="iwtitle">' + locality.lokalita_name + '</h2>'
            ...
            + '</div>';

    // coordinates
    var latLng = new google.maps.LatLng(locality.lokalita_lat,
                 locality.lokalita_lon);
        ...
    var marker = new MarkerWithLabel({//set marker
```

```
        position: latLng, //set marker position coordinates
        map: map, //set map to be used by the marker
        ...
        labelVisible: false,
    });

    markers.push(marker);
    ...
    google.maps.event.addListener(marker, 'click', function() {
        var id = marker.get("id");

        // implicitni nastaveni menu v IW
        setDefaultIWMenu();
        infowindow.close();

        infowindow.setContent($('#' + locality.lokalita_id + '_iwc')[0]);
        infowindow.open(map, marker);

        ...
    });
});
```

4.2.2.2 Filtrování lokalit

Většinu funkcí mapy obstarává menu, které je možné s pomocí JavaScriptu skrýt, aby neubíralo plochu mapy. Filtrovat lokality je možné podle tří základních kritérií:

1. podle typu lokality
2. podle států (příp. v budoucnu podle jiných adm. jednotek)
3. podle roku založení ap.

Po každé změně zobrazených lokalit proběhne funkce `setMapPositionDependShowMarkers()`, která nastaví zoom a střed mapy podle zobrazených lokalit a zároveň zobrazené lokality sečte:

```
var countOfMarkers = setMapPositionDependShowMarkers();
$('#count_of_loc_num').text(countOfMarkers);
...
function setMapPositionDependShowMarkers() {

    var bounds = new google.maps.LatLngBounds();
    var countOfMarkers = 0;

    for (var i = 0; i < markers.length; i++) {
```

```
        if (markers[i].getVisible()) {
            bounds.extend(markers[i].position);
            countOfMarkers++;
        }
    }

    if (countOfMarkers > 0) {
        map.fitBounds(bounds);
    }

    return countOfMarkers;
}
```

První a druhý typ filtru je jednoduchý - v cyklu se projdou všechny markery, a podle požadovaných parametrů jsou buď skryty nebo zobrazeny. Ve třetím případě, tedy filtrování podle roku (založení nebo první zmínky) jsou zobrazeny ty lokality, které mají daný rok stejný nebo menší. Tedy např. ty, které v daném roce již byly založeny. Je využito dalšího widgetu z frameworku *JQuery UI*, tedy *Slideru*. Posunutím se změní rok a v mapě se objeví / zmizí určité lokality:

```
$("#slider").slider({
    orientation: "vertical",
    range: "min",
    value: 1000,
    min: 0,
    max: 2000,
    slide: function(event, ui) {
         $("#year").val(ui.value);
        viewByYear(ui.value);
    },
    stop: function(event, ui) {
        var countOfMarkers = setMapPositionDependShowMarkers();
        $('#count_of_loc_num').text(countOfMarkers);
    }
});
```

Filtry obsahují i další parametry a funkcionalitu, na jejichž popis zde není prostor.

4.2.2.3 Další funkce mapy

Export mapy - vč. lokalit a popisů je v době psaní práce ve fázi vývoje. V každém případě - základní myšlenka byla zpočátku zhruba následující: V mapě se provede obdélníkový výběr s pomocí třídy `google.maps.Rectangle` (příp. jinak) a daný výřez se využitím další služby od Googlu, *Static Maps API* (viz kap. 2.2.1.8) vygeneruje ve stejném měřítku, jako je hlavní mapa. Server by potom (např. s pomocí PHP a knihovny GD) do výřezu doplnil zobrazené lokality, jejich popis, příp. měřítko a legendu a hotový

výřez vrátil uživateli ke stažení. Toto však narazilo na zásadní omezení právě služby *Static Maps API*.

Prvním omezením je maximální možná velikost vygenerovaného obrázku, která je 640 x 640 pixelů [14]. Řešení se zprvu jevilo v tom, že server postupně načte více výřezů, které budou maximální velikosti právě 640 x 640 px, a dohromady dají požadovaný velký výřez, a tyto výřezy sloučí do jednoho právě s pomocí GD knihovny. Takové plně automatické zpracování však není možné, což je druhé omezení. Obrázek statické mapy např. přístupný pod odkazem

```
http://maps.googleapis.com/maps/api/staticmap?center=40.714728,
    -73.998672&zoom=12&size=400x400&maptype=roadmap&sensor=false
```

je určen pro přímé zobrazení na určitém webu, nelze jej použít pro serverové zpracování [15] a jediná možnost, jak z generovaných obrázků vytvořit jeden velký, je ta, že uživatel na každý klikne a ručně stáhne. Potom příp. odešle na server, který už obrázky jako mozaiku spojí dohromady a připojí lokality (viz výše). To je jediné legální řešení, a cílem je zprovoznění funkce, ale s omezením velikosti výřezu (např. do velikosti obrazovky).

Následuje ukázka vytvoření obdélníkového výřezu a následného vytvoření URL dotazu na službu *Static Maps API*. Obdélník v mapě je zadáván geografickými souřadnicemi (zeměpisná šířka a délka), ale výřez statické mapy naopak zeměpisnými souřadnicemi středu, přiblížením a velikostí výřezu v pixelech. Je tedy nutný přepočet těchto souřadnic:

```
$("#vyrez").bind("click", function() {

    var northEast = map.getBounds().getNorthEast();
    var southWest = map.getBounds().getSouthWest();

    var frame_width = (northEast.lat() - southWest.lat()) / 10;

    var rectangle_bounds = new google.maps.LatLngBounds(
            new google.maps.LatLng(southWest.lat() + frame_width,
                southWest.lng() + frame_width),
            new google.maps.LatLng(northEast.lat() - frame_width,
                northEast.lng() - frame_width)
            );

    rectangle.setEditable(true);
    rectangle.setBounds(rectangle_bounds);

    makeRectangle();
});

...

function makeRectangle() {
```

```
    rectangle.setMap(null);
    rectangle.setMap(map);
}

$("#export_map").bind("click", function() {

    var center = rectangle.getBounds().getCenter();
    var zoom = map.getZoom();

    var scale = Math.pow(2, map.getZoom());

    var northEast = rectangle.getBounds().getNorthEast();
    var southWest = rectangle.getBounds().getSouthWest();

    var northEastPoint = map.getProjection().fromLatLngToPoint(northEast);
    var southWestPoint = map.getProjection().fromLatLngToPoint(southWest);

    var x_size = Math.floor((northEastPoint.x - southWestPoint.x) * scale);
    var y_size = Math.floor((northEastPoint.y - southWestPoint.y) * scale)
        * (-1);

    ...

    var url_static_map = 'http://maps.googleapis.com/maps/api/staticmap?';
    url_static_map += 'center=' + center.lat() + ',' + center.lng();
    url_static_map += '&zoom=' + zoom;
    url_static_map += '&size=' + x_size + 'x' + y_size;
    url_static_map += '&style=feature:road.highway|element:labels|
        visibility:off';
    url_static_map += '&sensor=true';

    $("#static_map_link a").attr('href', url_static_map);
    $("#static_map_link").show();

    $("#static_map_links").html('');
    $("#static_map_links").hide();
});
```

V tuto chvíli se zobrazí link pro zobrazení potřebného výřezu, zatím však bez lokalit. Pokud je výřezů více, zobrazí se odkazů více, tato varianta je však zatím rozpracovaná, stejně jako celý systém získávání výřezů z mapy. Do určité míry je to také dáno tím, že CMS je v současné době zaměstnáno především plněním databáze, požadavky na mapu nejsou prioritou.

Dalšími funkcemi jsou zobrazení jednoduchých statistik mapy - počet zobrazených

lokalit se již zobrazuje (viz kap. refsec:filtrovanilok). Také je možné popis skrýt, resp. zobrazit - implicitní nastavení je skrytý popis, protože má větší význam pouze u exportu mapy.

```
$("#popis_hidden").live("click", function() {

    if ($(this).is(':checked')) {
        hideLabels();
         ...
        labels_visible = false;
    } else {
        showLabels();
        ...
        labels_visible = true;
    }
});

function showLabels() {
    for (var i = 0; i < markers.length; i++) {
        markers[i].set("labelVisible", true);
    }
}

function hideLabels() {
    for (var i = 0; i < markers.length; i++) {
        markers[i].set("labelVisible", false);
    }
}
```

Kapitola 5

Závěr

Pokud se podíváme do zadání této diplomové práce, cíl byl splněn. Alespoň v rozsahu potřebném pro tuto diplomovou práci - jinak projekt pokračuje dál.

Centrum medievistických studií se nyní soustředí na plnění databáze, mezitím já i uživatelé postupně odhalujeme poslední kritická místa aplikace. Jen dlouhodobějším užíváním se totiž přijde na některá nelogická - nebo naopak logická a efektivní řešení. A to nemám na mysli kvalitu kódu, ale manipulaci s databází vůbec, a to, zda dostatečně pokryla požadavky historiků. V průběhu vývoje proběhlo mnoho schůzek jak v CMS, tak s lidmi, kteří budou databázi aktivně plnit, resp. už ji plní. Pohled na databázi se zprvu utvářel, poté vyvíjel a měnil, což by mělo zajistit systém, v němž má každý údaj své místo, a nic zde není zbytečně.

Nápady na vylepšení, nebo naopak zjednodušení různých dílčích situací přicházejí z CMS stále, a stále je o čem diskutovat. V následujících odstavcích tedy shrnu základní možnosti do budoucna. Pokud jde o nejbližší události - které často vyplynuly až v závěru psaní této práce - lze je rozdělit na ty související s databází a na ty, které souvisí s mapou.

- Související s databází:
  - rozdělení dalších názvů hesel podle jazyků - *čeština*, *němčina* a *latina*
  - zkvalitnění systému zálohování databáze - jak automatické, tak např. předchozí verze hesla
  - automatické indexování dokumentů pro fulltextové vyhledávání (formou automatického spouštění indexovací akce -např. *Cron*, nebo formou např. spuštění indexování při prvním načtení hlavní stránky každý den).
  - zasílání e-mailu hlavnímu administrátorovi při zvlášť důležitých událostech
  - kvalitnější implementace hlavní stránky pro návštěvníky, možnost její snadné editace
  - vytvoření komplexní nápovědy - zvláštní stránky i nápověda, která se zobrazí po najetí myší na příslušný element
  - atd.
- Související s mapou:

- finální implementace nejméně uživatelsky náročného systému exportu / tisku výřezů z mapy, vč. měřítka, legendy, popisu atd.
- zobrazení měřítka i přímo v on-line aplikaci - především grafického
- možnost zrušení výřezu
- měření vzdálenosti
- rozšíření obsahu infooken u lokalit
- různé sady mapových značek typů lokalit
- možnost zobrazení v mapě filiačních vztahů
- zobrazování některých geografických informací také o osobnostech a artefaktech
- atd.

Dále v určitý okamžik bude potřeba databázi graficky upravit tak, aby bylo také na co se dívat. Přijde na řadu stylování s pomocí CSS. V tuto chvíli soubor se styly spíše drží aplikaci přehlednou, ale v budoucnu by měl zajistit grafiku, která bude tematická, bude odpovídat např. stránkám s připravovanou výstavou v Národní galerii k výročí 700 let benediktinů v Čechách. Výstava nese jméno „Otevři zahradu rajskou - Benediktini v srdci Evropy 800-1300“ a bude s databází do jisté míry spojena. Díky MVC architektuře aplikace není takový problém v rozumné době vytvořit novou grafiku - manipuluje se totiž jen s částí *pohledu* (*view*).

Jsou zde i nápady a plány, které patří spíše do vzdálenější budoucnosti a nemusí ani souviset s tímto projektem NAKI. Především jde o vícejazyčnost aplikace. Databáze s největší pravděpodobností bude zároveň i v angličtině a němčině. K překladu se výborně hodí komponenta `Zend_Translate`. Statické překlady (např. menu, popisy ve formulářích atd.) budou umístěny v externím souboru (např. PHP pole, TMX soubory, CSV soubory nebo INI soubory), dynamické budou vedeny jako další sloupce v databázi.

Druhým směrem pro pozdější vývoj, na který přišla řeč, je rozšíření databáze i do jiných řádů. Zda to je dobrý nápad, ukáže až čas, každopádně při takovém rozšíření bude pravděpodobně potřeba ve velké míře upravit strukturu databáze. Přece jen každý řád má svá specifika, která je potřeba v databázi vystihnout. Také není k zahození nápad rozšířit databázi i do ostatních částí Evropy.

V průběhu práce na databázi jsem příspěvek prezentoval také na Studentské konferenci v Telči v listopadu 2013.

K vývoji bylo použito prostředí NetBeans IDE, okrajově také editor PSPad a k psaní tohoto textu systém LaTeX. Vše je volně dostupné ke stažení. Technická poznámka: ke správnému fungování aplikace je potřeba provést tato opatření PHP na serveru:

- povolit funkce `mb_string`,
- povolit knihovnu `GD`,
- nastavit `magic_quotes_gpc=on`

Poslední větou v popisu zadání je: „Shrnutí a kritické zhodnocení zvolených řešení“. Tak tedy řešení, které jsem zvolil - tedy kombinace *Zend Framework*, *MySQL*, *JQuery*, *Google Maps* je myslím pro tento projet plně vyhovující a podruhé bych použil stejnou kombinaci. Byť např. novou verzi ZF - *ZF 2*. V implementaci bych pravděpodobně více zapojil databázi, funkci některých modelů bych přesunul na databázové požadavky. Ale ani model použitý zde nepovažuji za špatný. Ovšem oblast, kterou bych podruhé implementoval jinak, jsou interaktivní formuláře - zejména části kódu, které se starají o AJAXové načítání různých typů propojení (představení ap.) se mnohdy opakují, jen v jiném kontextu. Opakovaná implementace např. tlačítek vložení, smazání představeného, fundátora, mateřského kláštera, apod. zabírá mnoho stovek řádků kódu. Pravděpodobně bych příště vytvořil univerzální funkce pro vytvoření elementů a ty jen podle kontextu naplnil daty z AJAXu a umístil na správné místo ve formuláři. Další zásadní nedostatky v projektu nevidím - vše, co se ukázalo jako „slepá ulička“ bylo v průběhu práce smazáno a nahrazeno jiným řešením. Často došlo i k rušení mnoha databázových tabulek nebo některých akcí v řadiči.

Protože databáze, která by podobným způsobem mapovala benediktiny v Česku pravděpodobně neexistuje, nebylo příliš mnoho projektů, od kterých by se dalo snadno „odrazit“. V zahraničí existuje podobná databáze např. na adrese `http://monasticwales.org/` - ta se na úplném počátku i stala částečně vzorem pro tuto databázi benediktinských klášterů. Jedná se o databázi velšských klášterů, která také obsahuje mapu. Dnes je myslím možné s úspěchem říci, že - alespoň v některých oblastech (např. mapa) je databáze benediktinských klášterů propracovanější [1].

[1] samozřejmě toto lze špatně hodnotit. Např. nevíme, jak propracovaný systém editace článků mají autoři zmiňované databáze, nebo zda-li vůbec nějaký.

Literatura

[1] POSPÍŠIL O.: *Via Sancta - informační systém o poutní cestě, bakalářská práce*, Praha 2011. Dostupné z <http://gama.fsv.cvut.cz/ cepek/proj/bp/2011/ondrej-pospisil-bp-2011.pdf>.

[2] ŠÁRA M.: *Webová prezentace reklamní agentury s e-shopem - bakalářská práce*, Pardubice 2012.
Dostupné z <http://dspace.upce.cz/bitstream/10195/46133/4/SaraM_Webova Prezentace_IR_2012.pdf>.

[3] WIKIVS: *MySQL vs PostgreSQL* [online]
Dostupné z <http://www.wikivs.com/wiki/MySQL_vs_PostgreSQL>.

[4] *Mapy.cz - FAQ – často kladené otázky k API* [online].
Dostupné z <http://api4.mapy.cz/view?page=faq>.

[5] *Google Maps/Google Earth APIs Terms of Service* [online]. Dostupné z <https://developers.google.com/maps/terms>.

[6] *Google - Mapy pro firmy* [online]. Dostupné z <http://www.google.com/intl/cs/enterprise/mapsearth/products/mapsapi.html>.

[7] *Google Maps JavaScript API v3 - Usage Limits and Billing* [online]. Dostupné z <https://developers.google.com/maps/documentation/javascript/usage>.

[8] *Google-maps-utility-library-v3 - Libraries* [online]. Dostupné z <http://code.google.com/p/google-maps-utility-library-v3/wiki/Libraries>.

[9] PIKOVER J.: *Google Maps vs. Apple Maps iOS (comparison)* [online], 2013. Dostupné z <http://www.gadgetreview.com/2013/01/google-maps-vs-apple-maps-ios-comparison.html>.

[10] DUNN I.:*Implementing the MVC Pattern in WordPress Plugins* [online]. Dostupné z <http://iandunn.name/content/presentations/wp-oop-mvc/mvc.php>.

[11] *Wikipedia: Ajax* [online]. Dostupné z <http://en.wikipedia.org/wiki/Ajax_(programming)>.

[12] *The Google Geocoding API: Documentation* [online]. Dostupné z <https://developers.google.com/maps/documentation/geocoding>.

[13] *The Google Geocoding API: Usage Limits* [online]. Dostupné z <https://developers.google.com/maps/documentation/geocoding/#Limits>.

[14] *Static Maps API V2 Developer Guide: Image Sizes* [online]. Dostupné z <https://developers.google.com/maps/documentation/staticmaps/#Imagesizes>.

[15] *Static Maps API V2 Developer Guide: Limits* [online]. Dostupné z <https://developers.google.com/maps/documentation/staticmaps/#Limits>.

[16] BÖHMER M.: *Zend Framework - Programujeme webové aplikace v PHP* Computer Press, a.s.: 2010, 416 s. ISBN 978-80-251-2965-4

[17] GUTMANS A., BAKKEN S.S., RETHANS D.: *Mistrovství v PHP 5*, 2. vydání Computer Press, a.s.: 2008, 655 s. ISBN 978-80-251-1519-0

[18] MARGORÍN M.: *jQuery bez předchozích znalostí - Průvodce pro samouky*, 1. vydání Computer Press, a.s.: 2011, 255 s. ISBN 978-80-251-3379-8

Příloha A

Obsah přiloženého CD

K této práci je přiloženo CD, na kterém jsou uloženy zdrojové kódy a tato diplomová práce. Má následující strukturu:

- diplomova_prace
  - dp.pdf - text diplomové práce ve formátu PDF
  - zdrojové soubory LaTeXdiplomové práce
- zdrojove_soubory - celý adresář projektu včetně knihovny Zend, soubor `index.php` se nachází ve složce `/public` a v souboru `/application/config/application.ini` lze nastavit databázový adaptér (viz kap. 4.1.2.1)
- sql - složka s **.sql* souborem pro vytvoření databáze včetně tabulek
- er_diagram.png - ER diagram databáze v plné velikosti (viz obr. 3.1)