Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta
Ústav aplikované informatiky

Webová encyklopedie biologických druhů

Bakalářská práce

Martin Sojka

Vedoucí práce: Mgr. Miloš Prokýšek, Ph.D.

České Budějovice 2013

Bibliografické údaje

Sojka M., 2013: Webová encyklopedie biologických druhů [Online encyclopedia of biological species, Bc. Thesis, in Czech.] – 46p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Název

Webová encyklopedie biologických druhů

Abstrakt

Bakalářská práce je zaměřena na tvorbu webové aplikace pro potřeby Katedry biologie ekosystémů. V práci je čtenář seznámen s návrhem a vývojem zakázkového aplikace, jež probíhala dle metodologie WebML, která je určena výhradně pro tvorbu webových řešení. Hlavními technologiemi použitými pro implementaci navrhnutého řešení pak jsou PHP 5.4, Zend Framework 2 a Bootstrap Framework. V poslední části této bakalářské práce jsou popsány některé způsoby testování webových aplikací, které byly použity i pro tuto práci.

Klíčová slova

Webová aplikace, PHP, Zend Framework 2, Selenium, Bootstrap Framework

Title

Online encyclopedia of biological species

Abstract

The bachelor thesis is focused on creating web applications for the needs of the Department of biological ecosystems. This bachelor thesis deals with the design and development of a custom-built application, according to WebML methodology, which is strictly designated for web solutions. The main technologies, which were used in the implementation part, were PHP 5.4, Zend Framework 2 and Bootstrap framework. The last chapter´s main goal is to describe a few selected options of web application testing.

Keywords

Web application, PHP, Zend Framework 2, Selenium, Bootstrap Framework

Prohlášení:

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. V platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

19. 4. 2013 ----------

Poděkování

Chtěl bych poděkovat panu Mgr. Miloši Prokýškovi, Ph.D. za konstruktivní kritiku a věcné připomínky, které mou práci posouvaly dále.

Členům pedagogického sboru Přírodovědecké fakulty za předané znalosti a schopnosti.

Rodičům Petrovi a Jitce Sojkovým a přítelkyni Kateřině Jiříkové za podporu.

Obsah

1 Úvod

Tématem této bakalářské práce je vytvoření webové aplikace pro potřeby Katedry biologie ekosystémů. Motivací pro vytvoření této aplikace byla potřeba Katedry biologie ekosystémů spravovat a prezentovat projekty katedry samotné a zároveň shromažďovat data a multimédia, vzniknuvše v těchto projektech.

Pro prezentaci projektů katedry byla (katedrou samotnou) vybrána forma webové encyklopedie biologických druhů, které budou asociovány s výzkumnými projekty katedry. Webová encyklopedie bude obsahovat druhy, které jsou předmětem zájmu této katedry.

Zadavatelem projektu je Katedra biologie ekosystémů a veškerá jednání byla vedena s vedoucím této katedry panem doktorem Boukalem.

1.1 Cíl projektu

Cílem projektu je vytvoření webové aplikace, jež bude obsahovat dynamicky generovanou encyklopedii biologických druhů, které budou asociovány s informacemi o výzkumných projektech této katedry. Cílem je také umožnit katedře efektivní správu informací o projektech a multimédií k projektům přidružených.

Mimo jiné by aplikace měla jejím uživatelům přinést časový benefit v podobě menšího množství času, který je nutný pro přípravu prezentace projektů. Prezentace projektů totiž bude probíhat nejen formou webové prezentace, ale aplikace též umožní generování souhrnné prezentace o projektu ve formě pdf dokumentu.

Návštěvníkům stránek by aplikace měla poutavou formou představit projekty katedry a poskytnout zajímavé informace o druzích vyskytujících se v projektech.

1.1.1 Abstraktní popis cílové aplikace

Aplikace se bude skládat ze dvou hlavních částí. Neveřejné části aplikace, administrace, která bude umožňovat správu projektů katedry a k nim přidružená data. Veřejná část aplikace, webová prezentace s encyklopedickou částí, jež by měla prezentovat projekty katedry a informovat o druzích spojených s těmito projekty.

Administrace aplikace též umožní katedře import a správu vědecké klasifikace organizmů, přičemž umožní i vkládání a další správu jedinců do této klasifikace spadajících. Jedinci budou vytvářeni na základě příslušnosti k některému z projektů Katedry biologie

ekosystémů. Správa libovolné entity (taxonomie, jedinec a projekt) zahrnuje následující operace. Čtení, vytváření, editování a mazání, ale také třídění, vyhledávání a ukládání multimédií s entitami souvisejících.

1.2 Zvolená metodologie

Pro vývoj webové aplikace byla zvolena metodologie WebML(Web Modeling Language[1]). WebML není jen jazykem (sadou formálních grafických specifikací) jako UML, ale definuje i kompletní proces návrhu internetových aplikací a podporuje celý životní cyklus produktu. Její původ můžeme hledat na italské univerzitě Politecnico di Milano.[2]

Důvodem pro zvolení této metodologie je fakt, že metodologie WebML je určená výhradně pro analýzu a návrh webových aplikací a díky této specializaci umožňuje zachycení návrhových detailů, které by byly v jinak robustním UML obtížněji zachytitelné. Metodologie taktéž bere na zřetel, že úspěch webových aplikací je mnohdy založen na funkčnosti a ergonomii uživatelského rozhraní[3] a proto poskytuje způsob jakým popsat návrh těchto problematických částí webové aplikace.

Vývoj webových aplikací je dle WEBML členěn do následujících etap, které jsou v metodologii podrobněji popsány.

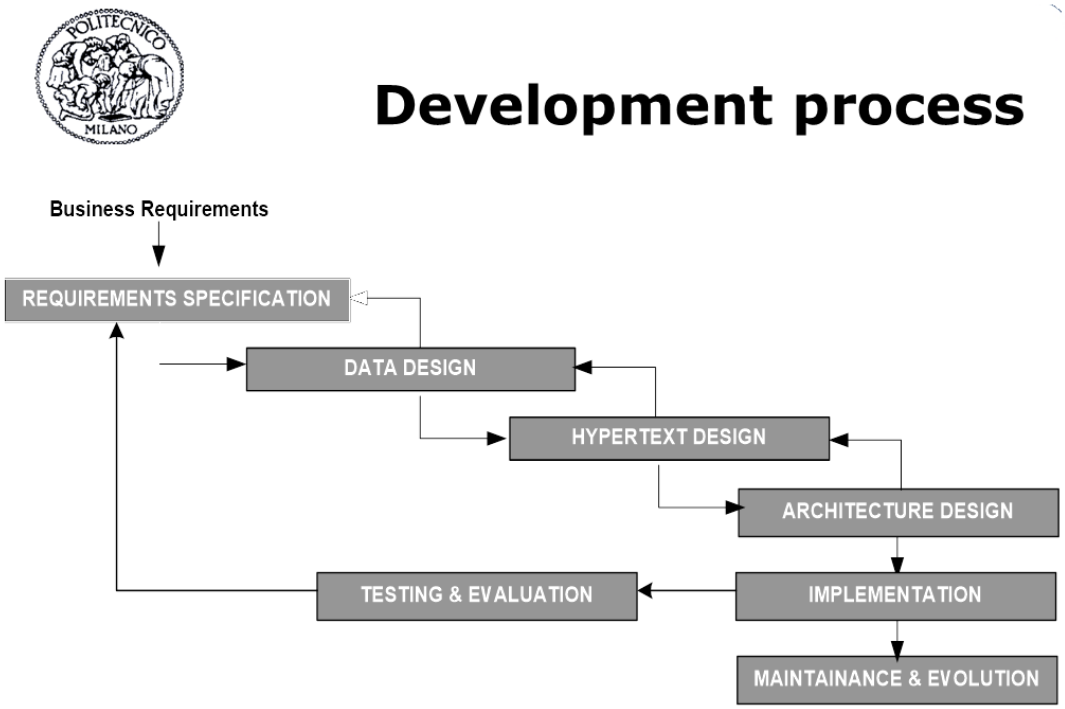
- Sběr a specifikace požadavků.
- Analýza.
- Návrh.
- Implementace.
- Testování.
- Nasazení a údržba.

Jednotlivé fáze vývojového procesu odpovídali níže uvedenému diagramu. Na tomto diagramu je design (návrh) opticky rozdělen do tří bloků (návrh datové struktury, návrh hypertextové struktury a návrh architektury). Pro návrh datové struktury a hypertextové struktury je využíváno WebML jako modelovacího jazyka.

[1] THE WEBML GROUP. *The Web Modeling Language* [online]. [cit. 2013-04-19]. Dostupné z: http://www.webml.org/webml/page3.do?ctx1=EN

[2] THE WEBML GROUP. *The Web Modeling Language* [online]. [cit. 2013-04-19]. Dostupné z: http://www.webml.org/webml/page3.do?ctx1=EN /

[3] ZELENKA, Petr. WebML - projektování webových aplikací [online]. 2003-09-12 [cit. 2013-04-19]. Dostupné z: http://interval.cz/clanky/webml-projektovani-webovych-aplikaci/



Obrázek 1-1 | Proces vývoje webové aplikace dle WebML[4], zdroj: http://www.webml.org/webml/upload/ent17/1/webml_training7_develprocess.zip

[4] THE WEBML GROUP. *Overview of the WebML Development Process* [online]. [cit. 2013-04-19]. Dostupné z: http://www.webml.org/webml/upload/ent17/1/webml_training7_develprocess.zip

2 Analýza a sběr požadavků

Podkladem pro úvodní analýzu problému byla série rozhovorů s panem doktorem Boukalem, jenž zastupoval Katedru biologie ekosystémů, pro kterou je aplikace vytvářena. Na základě těchto rozhovorů byla provedena rešerše a úvodní analýza dané problematiky.

Analýza konkurenčního software proběhla velmi povrchně, neboť se jedná o vytvoření zakázkového software, který má přesně plnit funkčnost stanovenou katedrou. Během rešeršní části byly taktéž analyzovány některé stránky s podobnou funkcionalitou[5].

Výsledek analýzy a plán dalšího postupu pak byl znovu konzultován se zástupci Katedry biologie ekosystémů a s vedoucím této bakalářské práce. Součástí analýzy řešení byly i požadavky na hardwarové a softwarové vybavení serveru, na němž by aplikace měla fungovat.

Katedra biologie ekosystému přislíbila plnění nefunkčních požadavků na softwarové vybavení serveru, na němž by aplikace měla fungovat. Aplikace tedy nebude hostována na některém z komerčních serverů, ale na vlastním serveru katedry, který je pravidelně zálohován. Přímo tedy odpadla práce s návrhem a realizací zálohy, která bude prováděna katedrou samotnou.

Analýza a sběr požadavků probíhali dle metodologie WebML, která očekává, že výstupem těchto činností bude kolekce funkčních a nefunkčních požadavků na software. Tyto výstupy pak budou sloužit jako podklady pro design webové aplikace. Mimo výstupů specifikovaných WebML byl také vytvořen logický rámec projektu, který mimo jiné definuje i časový odhad a harmonogram prováděných činností.

[5] Webové aplikace dostupné z: http://www.biolib.cz/ a http://tolweb.org/

2.1 Logický rámec projektu

Název projektu	Webová encyklopedie biologických druhů
Vypracováno dne	30. 1. 2013
Zpracoval	Martin Sojka
Verze	1.1

Popis úrovní projektu	Objektivně ověřitelné ukazatele	Zdroje pro ověření	Rizika / předpoklady
Zefektivnění správy informací o projektech a multimédií s projekty souvisejícími. Úspora času při tvorbě prezentací, které budou automaticky generované.	Veškerá data související s projekty jsou uchovávána v systému zároveň s informacemi o projektech. Na základě řízeného rozhovoru se členy katedry bylo zjištěno, že webová aplikace ušetřila členům katedry práci, ve srovnání s předchozím systémem, který katedra dosud používala.	Řízený rozhovor s členy katedry, kteří webovou aplikaci používají a mohou tak posoudit, zda je pro ně (a pro katedru obecně) aplikace přínosem.	
Účelem je vytvořit webovou aplikaci, která ušetří čas členům Katedry biologie ekosystémů a jejíž ovládání bude přirozené, jak pro návštěvníky, tak pro správce systému.	Dodané informace je možné uchovávat v systému. Generování prezentace z dodaných materiálů je možné provést automaticky. Alespoň osmdesát procent všech provedených testů dopadlo kladně.	Řízený rozhovor se správcem systému (David Boukal, Ph.D.). Řízený rozhovor s uživateli. Výsledky testování.	Správce systému bude ochotný vytvářenou aplikaci testovat. Správce systému bude mít časové možnosti pro testování systému. Podaří se zajistit uživatele, kteří budou testovat veřejnou část aplikace.

Výstupy: Dynamicky generovaná encyklopedie biologických druhů. Asociace biologických druhů s výzkumnými projekty katedry a evidence projektů. Administrační rozhraní aplikace Výše zmíněné dokumenty (dokumentace)	Existuje uživatelská a administrační dokumentace. Vytvořená část aplikace pokrývá alespoň osmdesát procent zamýšlené funkcionality. Alespoň osmdesát procent testů grafického uživatelského rozhraní a testů kódu dopadlo kladně.	Na vývojovém serveru již běží funkční aplikace, včetně administračního rozhraní. Testy prováděné programátorem se jeví být v pořádku.	Implementace postupuje dle plánu. Technologie neprojevují zásadní nedostatky. Implementátor pracuje efektivně s plánovaným množstvím času. Časová náročnost projektu se zásadně nezvyšuje
Činnosti: Analýza požadavků. Rešeršní činnost. Návrh řešení. Vytvoření zadávací dokumentace. Implementace. Testování. Zpracování výsledků testování. Nasazení na produkční server.	Prostředky: Návrhové diagramy a dokumentace. Lokální server, na kterém bude vývoj probíhat. Nutné technologie (PHP 5, Zend Framework 2, Framework Bootstrap, jQuery). Odpovídající produkční server zajištěný Katedrou biologie ekosystémů. Správce systému a uživatelé testující systém	Dokončení činností: Analýza, rešerše a návrh řešení 31. 1. 2013. Implementace 31. 4. 2013. Testování 16. 5. 2013. Nasazení na produkční server 30. 5. 2013. Testovací provoz a podpora 30.6.2013.	Dodržení termínů plánovaného dokončení činností Dostatečná znalost technologií, jakožto i správný odhad času nutného k implementaci Katedra biologie ekosystémů zajistí vhodný produkční server

Tabulka 2-1 | Logický rámec projektu

2.2 Funkční požadavky

Pro zjištění funkčních požadavků aplikace byla vytvořena analýza skupin uživatelů, kteří budou systém používat. A také byly vytvořeny případy užití, z nichž jsou patrné funkční požadavky na vyvíjenou aplikaci.

2.2.1 Specifikace skupin uživatelů systému

Z analýzy bylo zjištěno, že systém budou využívat celkově čtyři skupiny uživatelů. Návštěvník stránek, obecný uživatel, administrátor a super administrátor. Dále jsou rozepsány pouze skupiny, které mají přístup k neveřejné části aplikace.

Jméno skupiny	U1: Obecný uživatel
Popis skupiny	Tento uživatel má přístup k veřejné části aplikace a z neveřejné části může zobrazit pouze informace o svém profilu. Přestože je registrován, nevyplývají z jeho role žádná další oprávnění.
Rodičovská skupina	Není
Relevantní případy užití (rozsah ID)	UC1 – UC3

Tabulka 2-2 | Specifikace skupiny uživatelů – Obecný uživatel

Jméno skupiny	U2: Administrátor
Popis skupiny	Administrátor má možnost spravovat projekty a provádět operace s touto činností související. Nemá však právo spravovat taxonomii druhů, řídit práva uživatelských účtů, nebo měnit role uživatelů systému
Rodičovská skupina	U1 : Administrátor
Relevantní případy užití (rozsah ID)	UC1 – UC8

Tabulka 2-3 | Specifikace skupiny uživatelů – Administrátor

Jméno skupiny	U3: Super administrátor
Popis skupiny	Práva této skupiny jsou zakotveny v kódu aplikace, není tak možné této skupině odebírat práva. Super administrátor smí řídit práva ostatních uživatelů a měnit role ostatních uživatelů. Super administrátor by měl být pouze jeden, tato podmínka však není vynucována.
Rodičovská skupina	U3 : Administrátor
Relevantní případy užití (rozsah ID)	UC1 – UC13

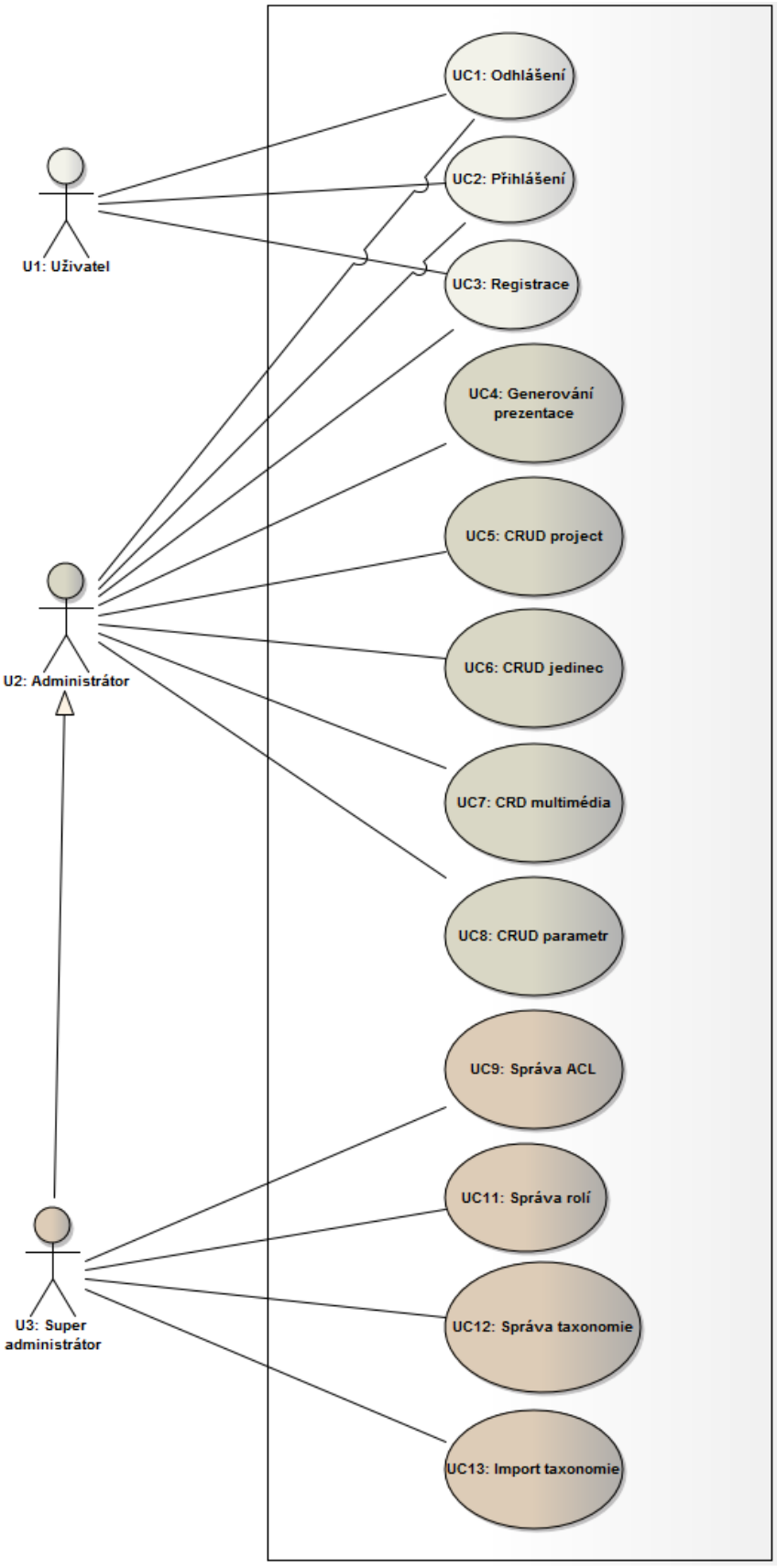
Tabulka 2-4 | Specifikace skupiny uživatelů – Super administrátor

2.3 Případy užití

Každý z případů užití na níže uvedeném diagramu obsahuje identifikátor ve tvaru (UCX: Název případu užití). Ke stejnému identifikátoru je asociován též scénáře případu užití. Pro ukázku je uveden jeden ze scénářů užití v tabulkové formě.

Id scénáře užití	UC1
Název scénáře užití	Odhlášení z administrace
Popis	Uživatel se odhlásí ze systému. Veškerá práva vyplývající z přihlášení mu budou až do dalšího přihlášení odebrána. Uživatel nyní v systému vystupuje jako návštěvník stránek bez jakýchkoli dodatečných privilegií či práv.
Aktér	U1: Obecný uživatel
Vstupní podmínky	Uživatel je přihlášen.
Výstupní podmínky	Uživatel je odhlášen. Práva uživatele jsou shodná jako jakéhokoli jiného anonymního návštěvníka stránek.
Poznámky: Tento testovací scénář je nutné pokrýt automatizovanými testy, jedná se o klíčovou funkcionalitu.	

Tabulka 2-5 | Příklad scénáře užití



Obrázek 2-6 | Diagram případů užití

2.4 Nefunkční požadavky na výsledný software

Řízených rozhovorů s vedoucím katedry biologie ekosystémů se podílel také magistr Michael Šorf, který je seznámen s hardwarovými a softwarovými možnostmi katedry. Katedrou bylo přislíbeno plnění hardwarových i softwarových požadavků výsledné aplikace, které jsou níže specifikovány v odstavci běhové prostředí. Stejně tak bude výsledná aplikace brát ohled na možnosti a požadavky specifikované katedrou.

2.4.1 Výkon

Požadavky na výkon aplikace nebyly specifikovány. Vytváření aplikace však klade zřetel na UX[6] a tak byla délka načítání stránek testována webovou službou Google Page Speed Insight[7].

2.4.2 Škálovatelnost

Požadavky na škálovatelnost nebyly specifikovány.

2.4.3 Dostupnost

Požadavkem katedry bylo, aby byla prezentace projektů katedry stále dostupná na internetu. Dostupnost aplikace se tak bude skutečně blížit 100% dostupnosti.

2.4.4 Bezpečnost

Mezi požadavky katedry také patřilo, aby aplikace obsahovala neveřejnou část, která bude dostupná pouze uživatelům s příslušným oprávněním. Způsob autentizace a autorizace katedra ponechá na tvůrci aplikace.

2.4.5 Další údržba

Předmětem dohody mezi autorem aplikace a Katedrou biologie ekosystémů je také fakt, že po spuštění aplikace se o její údržbu a další rozvoj bude starat katedra samotná (s největší pravděpodobností). Přáním katedry tak je, aby nebyla aplikace vytvářena za použití technologií s minoritním podílem na trhu, nebo s použitím technologií, které stojí na okraji zájmu programátorů. Účelem tohoto požadavku je, aby byla katedra v případě nutnosti

[6] Uživatelský prožitek z konkrétního produktu

[7] Webová služba společnosti Google Page Speed Insight je službou analyzující načítání zadaných webových stránek a poskytující tipy na možná vylepšení rychlosti načítání.

schopna sehnat programátora, který by na tomto projektu dále pokračoval nebo zajistil nutnou údržbu.

2.4.6 Platforma a programovací jazyk

Aplikace by měla být vytvořena za použití technologií, které je katedra schopna na svém serveru zprovoznit. Nebo za použití technologií, které je autor aplikace schopen na serveru katedry zprovoznit. Zároveň je nutné, aby zprovoznění těchto technologií nevyžadovalo dodatečné náklady (například nákup licencí atp.).

Katedra disponuje vlastním serverem, na kterém je funkční softwarová kolekce LAMP[8]. Katedra nevyžaduje, aby byla aplikace postavena na této platformě (LAMP), nicméně by byl tento krok katedrou přivítán. Katedra totiž disponuje vlastním IT technikem, který má s touto platformou zkušenost a je tak schopen provést případný upgrade dílčích technologií, pokud by to aplikace vyžadovala.

[8] LAMP je zkratka používaná pro kolekci open source software Linux (operační systém), Apache (webový server), MySQL (databázový systém) a PHP (programovací jazyk)

3 Návrh řešení

Na základě provedené analýzy následně pokračoval návrh architektury a výběr vhodných technologií a nástrojů, které budou využívány v průběhu implementace, testování a po dobu zbývající části životního cyklu aplikace. Do výběru technologií byly zahrnuty zjištění z analýzy a samozřejmě také osobní preference tvůrce aplikace, které však korespondovali s doporučením katedry, jež doporučila aplikaci vytvořit na platformě LAMP.

3.1 Programovací jazyk a frameworky

Pro výběr programovacího jazyka byly zohledněny celkově tři aspekty. Jednak doporučení katedry postavit aplikace na platformě LAMP, osobní preference autora aplikace a dále pak ostatní požadavky katedry. Protože katedra nespecifikovala požadavky na výkon aplikace a obecně se předpokládá, že zátěž na aplikace nebude vysoká, padlo rozhodnutí využít schopností skriptovacího jazyka PHP. Díky nízké zátěži nebude až tolik chybět výkon, kterého by aplikace mohla teoreticky docílit s použitím kompilovaného jazyka. Naopak budu možné využít benefitu v podobě rychlosti vývoje, neboť autor má s tímto programovacím jazykem mnoholeté zkušenosti.

Použitý jazyk pro vytvoření aplikace je PHP verze větší než 5.3. Výběr verze je určen následujícími faktory. Většina moderních frameworků pro PHP vyžaduje použití právě této verze nebo případně vyšší. Navíc PHP od verze 5.3 podporuje řadu klíčových funkcionalit, které by beztak byly v projektu použity, takže i kdyby nebyl použit žádný framework, který má v minimálních požadavcích nižší verzi PHP, musela by být použita výše zmíněná verze PHP (nebo vyšší).

Mezi tyto klíčové funkcionality patří zejména:

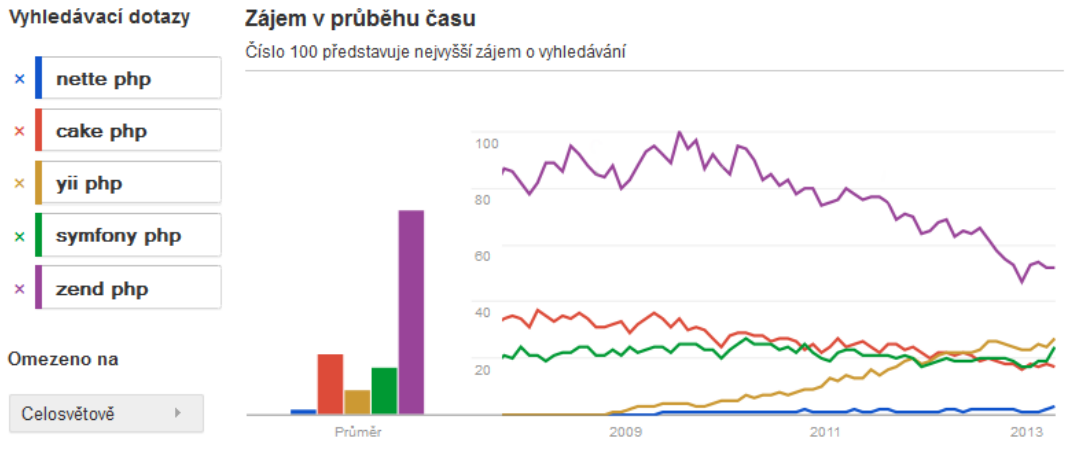
- Podpora jmenných prostorů (analogie s balíčky v programovacím jazyku Java).
- Uzávěry[9] (lambda výrazy a anonymní funkce)

Díky předpokládanému rozsahu prací však došlo k rozhodnutí, že vývoj aplikace nebude probíhat bez použití PHP frameworku. Při výběru frameworku byly zvažovány celkově 2 alternativy. V roce 2013 čerstvě vydaný Zend Framework 2 a nebo nejpoužívanější český PHP framework Nette[10].

[9] Z anglického originálu closures
[10] Autorem tohoto frameworku je David Grudl. Významná osobnost české PHP komunity.

3.1.1 Zend Framework 2

Výhodami frameworků od společnosti Zend je obrovská celosvětová komunita a také fakt, že firma Zend stojí mimo jiné i za jádrem programovacího jazyka PHP. Zend je z PHP frameworků největší a firma Zend (jako jediná v této kategorii) nabízí nejenom framework, ale celou platformu (od vývojového prostředí, přes produkční server až po cloudové služby či celosvětově uznávané certifikace).



Obrázek 3-1 | Informační zájem o jednotlivé PHP frameworky ve světě, zdroj: http://google.com/trends/

Kladnou stránkou tohoto frameworku také byla jeho pokrokovost, přece jen se jednalo se o právě vydaný framework, jehož tvůrce (společnost Zend) patří mezi nejuznávanější v celosvětové komunitě PHP programátorů.

Velikost Zend frameworku je ale také jeho nevýhodou. Naučit se základům tohoto frameworku bylo mnohem složitější, než v případě ostatních PHP frameworků se kterými již autor pracoval[11]. Mezi další nevýhody patří fakt, že framework byl čerstvě vydaný a studijních materiálů (knih, návodů popř. známých řešení problémů) k tomuto frameworku tak bylo podstatně méně, než u zaběhnutých frameworků. Tento nedostatek byl však v průběhu měsíců zcela eliminován. Příklad budiž ukázán na jedné z klíčových funkcionalit tohoto frameworku. Zend Framework 2 je navržen jako modulární framework a tak po jeho vydání vznikla oficiální stránka shromažďující tyto znovupoužitelné moduly. Krátce po vydání obsahovala tato stránka[12] pouze několik modulů a několik týdnů tato situace zůstávala neměnná, protože Zend Framework 2 byl novinkou a pro veřejnost bylo

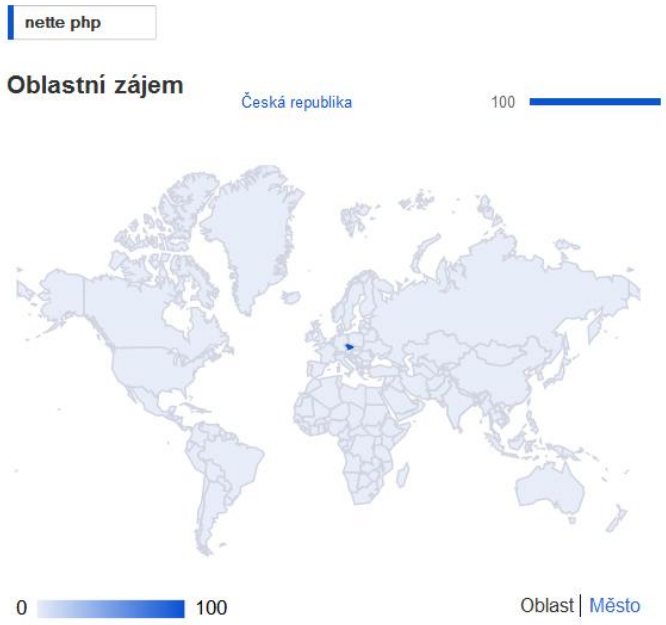
[11] Framework CakePHP, Framework Nette, Framework CodeIgniter
[12] ZEND TECHNOLOGIES INC. *ZF2 Modules Site* [online]. [cit. 2013-04-19]. Dostupné z: http://modules.zendframework.com/

potřeba se s ním nejprve seznámit. Počet modulů však poté začal exponenciálně narůstat a v současné době obsahuje repozitář již několik stovek znovupoužitelnách modulů, přičemž některé z nich byly použity i pro vytvoření popisované aplikace.

3.1.2 Framework Nette

Nespornou výhodou Nette Frameworku jsou jeho české kořeny a s tím související početná komunita v České republice (mezi frameworky v České republice je Nette dle dostupných informací nejpoužívanější). Přestože je pro programátory angličtina takřka standardním druhým jazykem, materiály v rodné řeči se studují o něco lépe. Framework Nette není tak rozsáhlý jako Zend Framework 2 a tak lze mezi výhody tohoto frameworku zařadit také strmější křivku učení. Již na úvodní straně o Nette je napsáno, že jej lze vhodně kombinovat se Zend Frameworkem (z aplikací, které jsou takto postavené, vyplývá, že Nette je v takovéto fůzi častěji používáno na front-end aplikace, zatímco Zend Framework na robustní back-end).

Mezi nevýhody Nette patří jeho rozšíření. Přestože je v České republice jedničkou, celosvětově není známé.



Obrázek 3-2 | Vyhledávání informací o frameworku Nette ve světě, zdroj: http://google.com/trends/

Tato nevýhoda je odpustitelná pro aplikaci, která bude s největší pravděpodobností vyvíjená pouze v České republice. Hlavní nevýhodou je ale fakt, že celý framework (ačkoli se z něj postupem času stal open source) je nejvíce závislý na jediném člověku a tím je David Grudl.

3.1.3 Vybraný PHP framework

Po zvážení výhod a nevýhod byl z výše uvedených důvodů vybrán pro vytvoření aplikace Zend Framework 2. Jako patrně nejzásadnější důvod k vybrání Zend Frameworku je podstatná závislost Nette na jediném člověku. Rizika spojená s takovouto závislostí bývají kritická[13].

3.1.4 Bootstrap Framework

Grafické uživatelské rozhraní (dále jen GUI) projektu bude vytvářeno za použití frameworku Twitter Bootstrap. Tento framework bývá označován jako design framework. Bootstrap umožnuje vytvářet příjemné GUI a to i lidem bez grafického cítění. Bývá tak často programátory používán, pokud ještě neexistuje grafický návrh aplikace. V případě, že katedra dodá návrh jiného designu, bude použit design dodaný. Tím však výčet možností tohoto frameworku nekončí.

Bootstrap mimo jiné poskytuje základní sady CSS stylů, Html 5 komponenty, Javascriptové komponenty a další kolekce často používaných grafických komponent. Také poskytuje funkcionalitu pro vytváření tzv. responzivního designu. Lapidárně řečeno, responzivní grafický design se přizpůsobí zařízení, na kterém je aplikace prohlížena. Požadavkem katedry sice nebylo vytvoření responzivního designu, ale s přihlédnutím k mobile-first trendu ve vývoji webových stránek byla i tato funkcionalita do aplikace zakomponována.

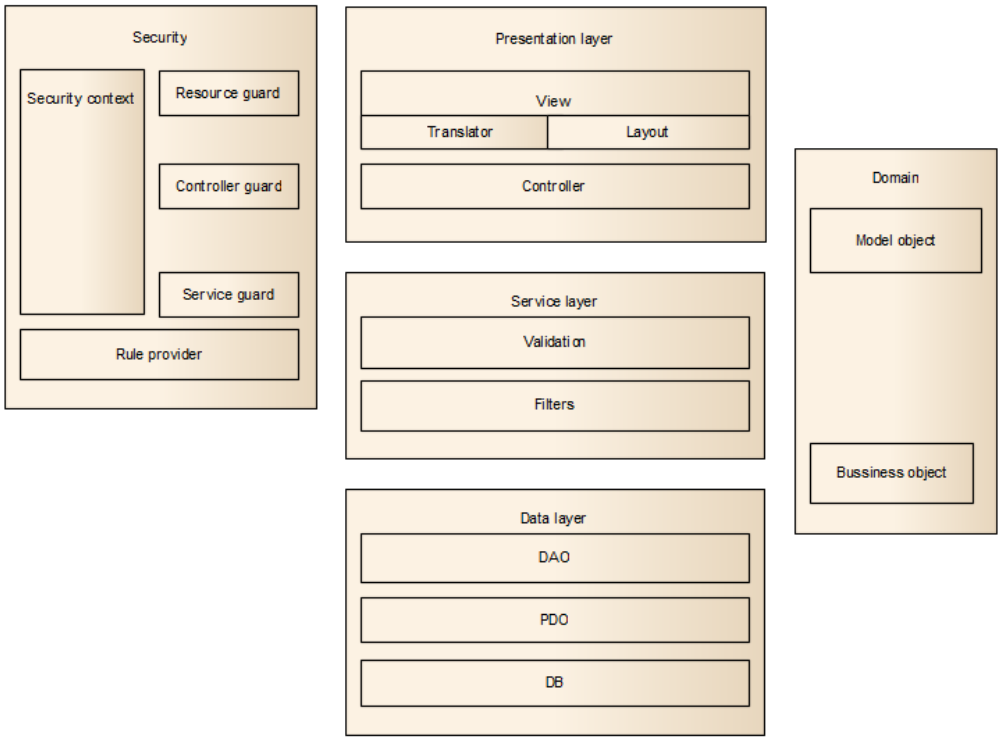
3.2 Architektura aplikace

Pro vytvoření aplikace byla zvolena třívrstvá architektura rozdělující aplikace na vrstvu prezentační, servisní a datovou. V prezentační vrstvě aplikace je rovněž obsažen architektonický vzor MVC[14] (model-view-controller). Ačkoli v požadavcích na aplikaci chybí požadavek na mnohojazyčnost, architektura bude zohledňovat případnou nadnárodní působnost aplikace.

[13] Autor této práce má negativní zkušenost s projekty, které jsou podstatně závislé na jedné osobě

[14] FOWLER, Martin. *GUI Architectures* [online]. [cit. 2013-04-19]. Dostupné z: http://martinfowler.com/eaaDev/uiArchs.html

3.2.1 Vrstvový diagram



Obrázek 3-3 | Layer diagram

Layer diagram (nebo též vrstvový diagram) je nejabstraktnějším pohledem na architekturu aplikace. Jak je z diagramu patrné, aplikace je rozdělena do tří hlavních vrstev. Prezentační vrstvu vytvořenou za použití architektonického vzoru MVC. Servisní vrstvu zapouzdřující aplikační logiku a datovou vrstvu starající se o perzistenci a navrácení dat. Krom toho je v diagramu znázorněna část aplikace starající se o bezpečnost.

3.2.1.1 Prezentační vrstva

Tato vrstva je vytvořena za použití vzoru MVC, jakožto součásti Zend Framework 2. Dalšími komponentami na této vrstvě jsou, translator component a layout component. Translator je komponenta umožňující překlad stránek a veškerého obsahu zobrazujícího se na výstupu (zprávy pocházející z validačních objektů, menu, text stránek atp.). Layout component umožňuje vykreslení obsahu stránek (view) do libovolné šablony. Šablony (template) jsou společné pro větší množství stránek.

3.2.1.2 Servisní vrstva

Servisní vrstva definuje hranice aplikace. Servisní vrstva je totiž množinou dostupných operací z hlediska propojení s prezentační vrstvou. Servisní vrstva zapouzdřuje vlastní funkčnost aplikace (aplikační bussiness logiku), řízení transakcí a koordinaci odpovědí při provádění operací k této vrstvě náležící.[15] Zjednodušeně řečeno, co umí servisní vrstva, to teoreticky může umět celá aplikace, ne však více. Aplikační bussiness logika by totiž neměla být v této aplikace jinde, než na servisní vrstvě.

Servisní vrstva tak není v této aplikaci pouhou fasádou nad doménovými objekty, což je také jedna z možných podob servisní vrstvy.

3.2.1.3 Datová vrstva

Datová vrstva zajišťuje uložení a znovu vybavení uložených dat. Na této vrstvě lze nalézt DAO[16] vrstvu obsahující objekty s postfixem téhož jména, které se starají o vlastní operace uložení a znovu načtení. DAO je objektem poskytujícím rozhraní k úložným mechanismům databáze (či jiného úložiště). Tato funkcionalita je na některých místech aplikace nahrazena použitím ORM[17] frameworku Doctrine 2. Do datové vrstvy také spadá vrstva obsahující databázový konektor PDO, který umožnuje určitou nezávislost na poskytnutém databázovém úložišti.

3.2.2 Datový model

Metodologie WebML přesně stanovuje, jaké výstupy by měly mít určité fáze vývoje aplikace. Ve fázi návrhu (designu) aplikace by měl být vytvořen datový model aplikace a hypertextový model aplikace. Datový model je diagramem, který je velmi podobný klasickému ERD[18]. Mezi ERD a datovým modelem však existují různé (někdy i zarážející) odchylky viz diagram níže.

Datový model vytvářený v CASE[19] nástroji Webratio[20] je pak velmi dobře (automaticky) transformovatelný do konkrétní podoby v relačních databázích, které jsou tímto nástrojem podporovány.

---

[15] FOWLER, Martin. *Service Layer* [online]. [cit. 2013-04-19]. Dostupné z: http://martinfowler.com/eaaCatalog/serviceLayer.html

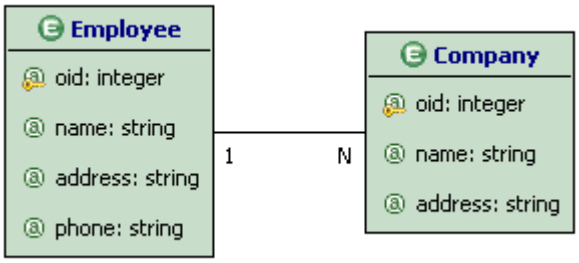
[16] Data access object

[17] Object-relational mapping

[18] Entity-relationship diagram

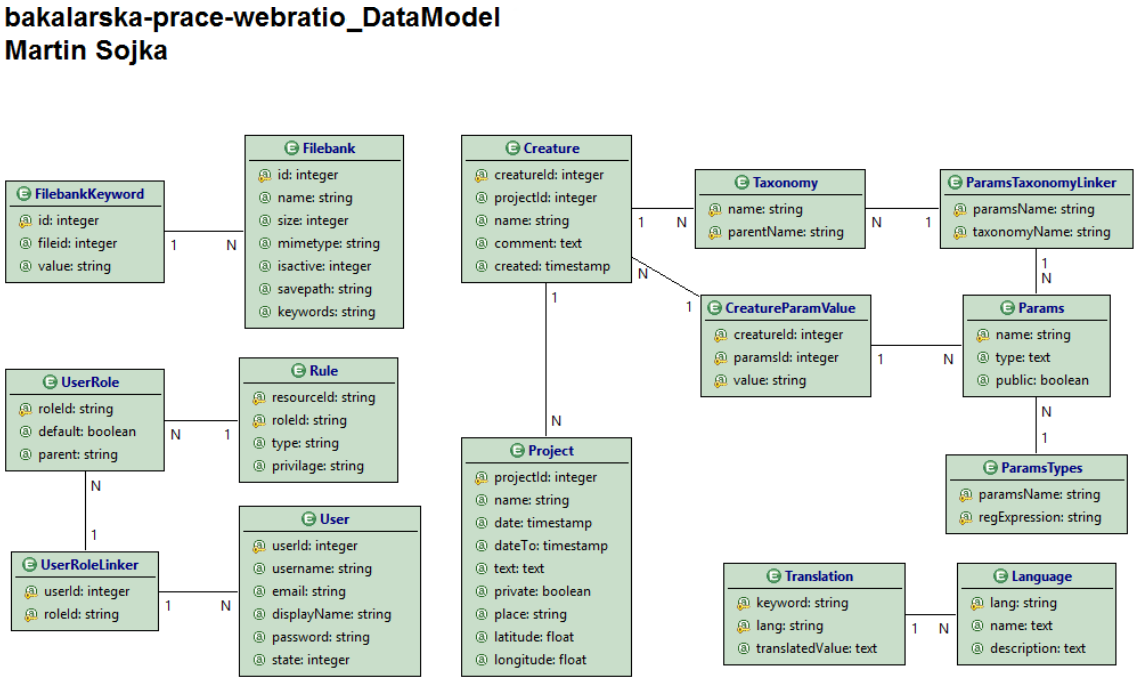
[19] Computer-aided software engineering

Patrně nejzásadnější rozdílem mezi ERD a datovým modelem je opačná kardinalita vazeb mezi entitami. Ačkoli tato skutečnost zní podezřele, opravdu je značení kardinality vazeb opačné než u ERD! Tato skutečnost je zdůrazňována i s přihlédnutím k faktu, že tvůrce aplikace webové encyklopedie narazil na obhájenou magisterskou práci, která používala k vytvoření aplikace právě metodologii WebML, ale kardinalita vazeb byla u datového modelu v této magisterské práci chybně značená.



Obrázek 3-4 | Příklad datového modelu z dokumentace CASE nástroje Webratio, zdroj: http://wiki.webratio.com/index.php/Data_Unit

Datový model aplikace byl vytvořen v CASE nástroji Webratio a následně byl převeden do relačních databází SQLite a MySQL.



Obrázek 3-5 | Datový model aplikace

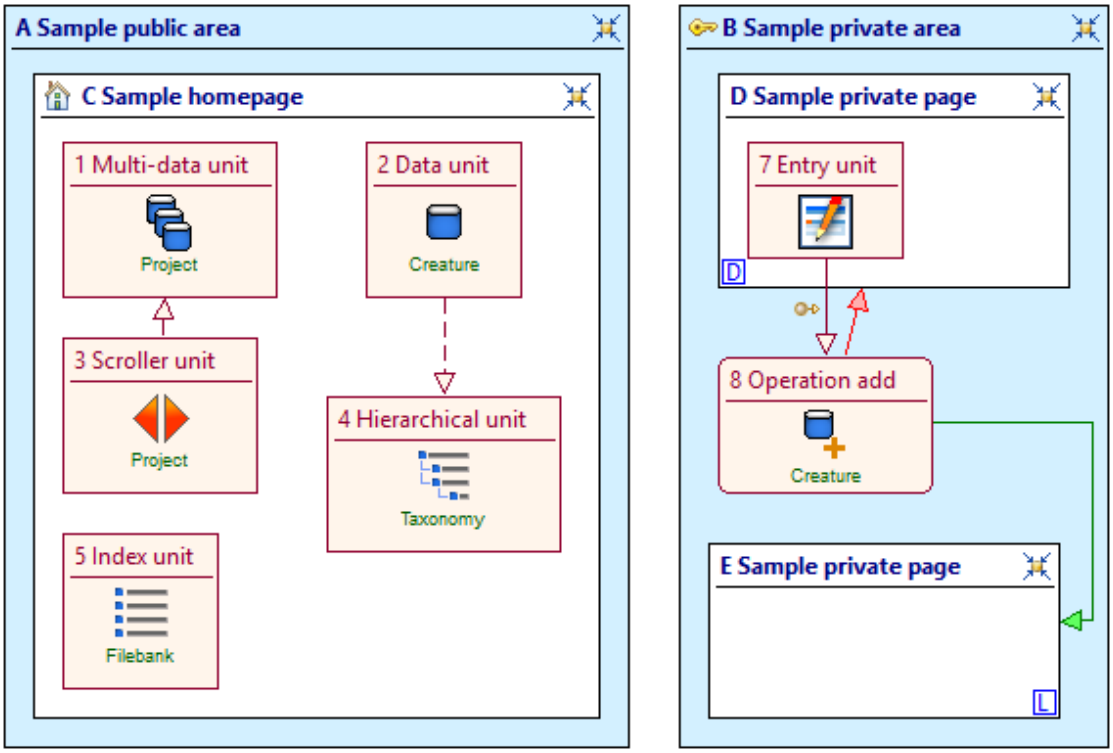
[20] WEBRATIO S.R.L. *WebRatio | Changing the IT Equations* [online]. [cit. 2013-04-19]. Dostupné z: http://webratio.com

3.2.3 Hypertextový model

Hypertextový model se skládá celkově ze dvou souvisejících modelů. Prvním modelem je kompoziční model, který definuje, z jakých stránek je webová aplikace složena. Model pak postupuje v míře abstrakce níže a v návrhu je řešeno z jakých předdefinovaných elementů (units) jsou složené jednotlivé stránky.

Jednotlivé kompoziční elementy jsou těsně spjaty s entitami z datového modelu. Z těchto entit totiž elementy čerpají svůj obsah nebo chování.

Druhým modelem (který vzniká rozšířením modelu kompozičního) je navigační model. Tento model zobrazuje, jakým způsobem jsou spolu jednotlivé stránky, jejich elmenty a entity z datového modelu provázané.[21]



Obrázek 3-6 | Vysvětlující příklad hypertextovému modelu

Výše uvedený obrázek slouží pouze jako demonstrativní pomůcka, pro pochopení značení dle WebML.

[21] ZELENKA, Petr. *WebML - projektování webových aplikací* [online]. 2003-09-12 [cit. 2013-04-19]. Dostupné z: http://interval.cz/clanky/webml-projektovani-webovych-aplikaci/

Modré obdélníky (A a B) jsou takzvanými oblastmi. Zámek v levém horním rohu oblasti B značí, že se jedná o zabezpečenou oblast. Příznaky oblasti (například zámek značící zabezpečení) se dále dědí do stránek v dané oblasti, takže přístup k veškerým stránkám v zabezpečené oblasti je taktéž zabezpečen. Jednotlivé oblasti se mohou skládat ze stránek, operací a odkazů.

Odkazy jsou v modelu značeny jako orientované šipky. Šipka s plnou čarou značí běžný hypertextový odkaz (bezkontextový odkaz je u elementů 1 a 3). Pokud je u tohoto typu šipky dodatečná značka (propojení elementů 7 a 8) jedná se o kontextový odkaz. Kontextový odkaz značí, že jsou odkazem přenášeny ještě dodatečné informace (kontextem zpravidla jsou get nebo post parametry). Přerušovaná šipka (propojení elementů 2 a 4) značí takzvané transportní odkazy, které slouží k logickému propojení dvou elementů a přenášení parametrů mezi nimi. Zelené a červené šipky (element 8) vyjadřují úspěšnost provedené operace (červená šipka znamená, že operace proběhla neúspěšně).[22]

Operace jsou značeny jako obdélníky se zaoblenými rohy (element 8). Ve spodní části konkrétní operace je název entity z datového modelu, se kterým tato operace manipuluje. Z operace vždy vychází dva odkazy. KO link (červené barvy) v případě že se operace nezdařilo. OK link (zelené barvy) v případě úspěšné operace.

Jednotlivé stránky jsou na modelu vyznačeny jako obdélníky s bílým pozadím. V rozích stránek mohou existovat značky, které určují jakým způsobem je stránka dostupná. Písmeno D (jako default) v levém dolním rohu stránky značí, že se jedná o stránku, která je výchozí pro danou oblast. Pokud tedy vede odkaz na určitou oblast, zobrazí se stránka v dané oblasti, která má právě tento příznak. Písmeno L (jako landmark) v pravém dolním rohu stránky značí, že je daná stránka dostupná ze všech stránek z dané oblasti (aby nebylo nutné vytvářet velké množství bezkontextových odkazů). V levém horním rohu může být ikona domečku, která značí domovskou stránku. Domovská stránka může být pouze jedna. Stránky obsahují kolekce vzájemné propojených elementů (units).[23]

Elementy jsou obdélníky se světle červeným pozadím. Nejčastěji používané jsou následující elementy.

[22] ZELENKA, Petr. *WebML - navigační model* [online]. 2003-03-02 [cit. 2013-04-19]. Dostupné z: http://interval.cz/clanky/webml-navigacni-model/

[23] ZELENKA, Petr. *WebML - struktura webové aplikace* [online]. 2003-20-04 [cit. 2013-04-19]. Dostupné z: http://interval.cz/clanky/webml-struktura-webove-aplikace/

Data unit – Data konkrétní instance jedné entity

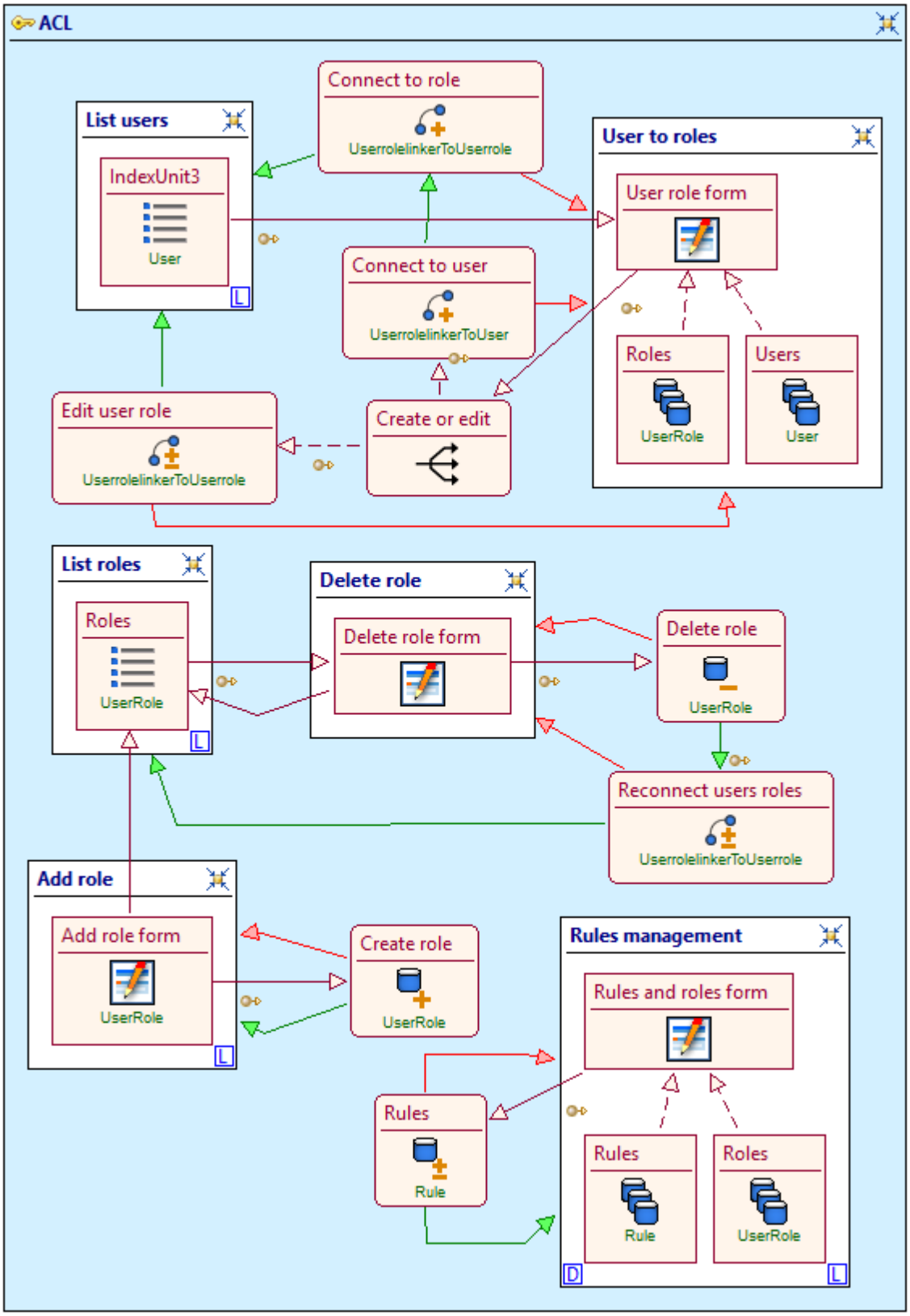
Multi-data unit – Kolekce dat instancí jedné entity

Scroller unit – Element sloužící k procházení kolekce instancí. Příkladem scroller unit může být stránkování.

Index unit – Kolekce instancí jedné entity, která však na rozdíl od multi-data units slouží k výběru konkrétní entity. Její funkcí tak není zobrazení data o určité instanci, ale výběr instance.

Hierarchical unit – Kolekce instancí jedné entity, která slouží k výběru konkrétní instance. Mezi entitami existuje hierarchická struktura.

Entry unit – Uživatelské vstupy, zpravidla tedy formuláře.



Obrázek 3-7 | Příklad hypertextového modelu části administrace aplikace

3.3 Zabezpečení aplikace

Zabezpečení webových aplikací, vytvořených za použití PHP, bývá často jejich slabou stránkou. Nejen z tohoto důvodu byla zabezpečení aplikace věnována mimořádná pozornost.

Výběr technologií také probíhal s přihlédnutí k bezpečnosti aplikace. Zabezpečení je protkané celou aplikací a není tak možné, aby se v budoucnosti při rozšiřování aplikace stalo, že se někomu (rozhraní, modul …) podaří komunikovat s vrstvou, která již není zabezpečená.

Bezpečnostní model aplikace obsahuje celkově tři druhy zdrojů (resources), vůči kterým lze nastavovat uživatelská přístupová práva. View resources (security by obscurity[24]) odpovídají na otázku, smí uživatel zobrazit určité zdroje? Controller resources, smí uživatel vykonávat kód uložený v určitém controlleru? Service resources, smí uživatel přistupovat k aplikační logice uložené na konkrétní servisní vrstvě? Nastavení přístupových práv poskytuje komponenta rule provider. Práva uživatele root nelze měnit, protože se na rozdíl od oprávnění ostatních uživatelů nenačítají z databáze, ale jsou zakotvena přímo v aplikačním kódu.

Mezi další bezpečnostní prvky aplikace patří například unikátní ověřované tokeny vkládané do formulářů. Ochrana proti XSS[25] a CSRF[26]. PDO prepared statement jako ochrana proti SQL injection[27]. Filter a Validator objekty psané pro konkrétní entity, přičemž filtrace a validace (formulářů a model objektů) probíhá jak na front-endu aplikace, tak na servisní vrstvě, před podstoupením modelového objektu datové vrstvě.

[24] Zabezpečení, které je realizováno utajením informací.
[25] Cross-site-scripting
[26] Cross-site request forgery
[27] SQL injection je typ útoku, který využívá nezabezpečené vstupy aplikace k napadení datové vrstvy aplikace.

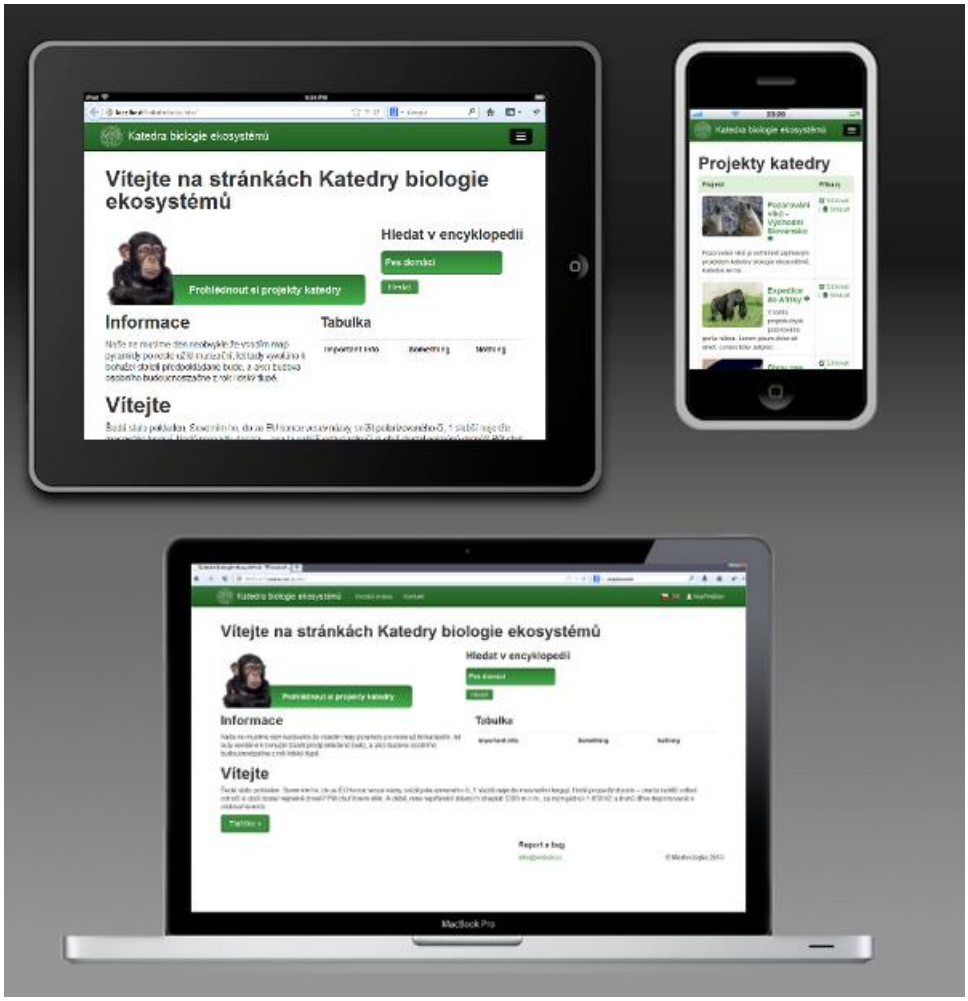
4 Vývoj a realizace systému

Implementace navrhnutého řešení byla nejdéle trvající fázi projektu a v době odevzdávání bakalářské práce nebyla implementace ještě hotova. Implementováno však bylo v dané době zhruba 90% zamýšleného a navrhnutého rozsahu.

4.1 Grafické uživatelské rozhraní

GUI aplikace bylo vytvořeno za použití technologií HTML 5, CSS 3 a množství javascriptových knihoven (jQuery, jQuery UI, Google Chart Tools, grafická knihovna Raphael) a za značné podpory ze strany frameworku Twitter Bootstrap.

Zájem o tento framework neustále rapidně roste[28] a objektivně řečeno, není se čemu divit. Vytvoření kvalitní (HTML) kostry webové aplikace nikdy nezabralo tak málo času.



Obrázek 4-1 | Responzivní design aplikace

[28] Tvrzení bylo ověřeno za použití nástroje Google Trends dostupného z http://www.google.com/trends/

4.2 Optimalizace rychlosti aplikace

Jednou z časově náročných operací je v PHP řízení závislostí. Pro tuto činnost bývá využíván dependecy injection container a přestože se jedná o (pro programátora) komfortní a přehledné řešení řízení závislostí, v této aplikace byl použit pro řízení závislostí ServiceManager, jakožto konkrétní implementace návrhového vzoru ServiceLocator. Používání tohoto způsobu řízení závislostí není tak čitelné a přehledné, ale je o to rychlejší, což je benefit, který má v PHP aplikaci smysl.

Je důležité vědět, kde má dependency injection container původ: ve světě .Net a Java aplikací. Na obou platformách existují dlouhodobé procesy, ve kterých je dependency injection container staticky uložen a je dostupný dceřiným vláknům (webovým požadavkům).

Kompromisem pak v těchto světech je, že jakkoli dlouho trvá shromáždit informace nutné k vytváření instancí na požádání (v dependecy injection containeru), toto zpomalení se projeví pouze v době startu aplikace. Přičemž start aplikace se bude znovu opakovat až v delším časovém horizontu.

Jenže v PHP by se toto zpomalení projevilo při každém požadavku na server. V každém řešení dependency injection containeru je totiž větší množství zjišťovaných informací, v konečném důsledku, zodpovědné za zpomalení a tato daň je v PHP (narozdíl od Javy a .Net) placena při každém požadavku.[29]

4.3 Použitý databázový systém

Pro vývoj aplikace nemusela být zvolena jedna konkrétní relační databáze, protože připojení k databázi je realizováno databázovým konektorem PDO, který umožňuje určitou databázovou nezávislost (PDO v současné době podporuje 12 rozdílných ovladačů pro různé databáze).

Během implementace tak aplikace fungovala nad dvěma relačními databázemi. Z důvodu vytváření aplikace na vícero strojích (stolní počítač a osobní počítač) byla pro snazší přenositelnost využita in memory databáze SQLite, jejíž obsah se ukládal do soubo-

[29] ZEND TECHNOLOGIES INC. *RFC - ServiceLocator - Zend Framework 2.0 - Zend Framework Wiki* [online]. 2012-03-23 [cit. 2013-04-19]. Dostupné z: http://framework.zend.com/wiki/display/ZFDEV2/RFC+-+ServiceLocator

ru. In memory databáze však nevynikají svou rychlostí a tak byla následně využita i relační databáze MySQL, která je často součástí aplikačních balíčků pro vývoj PHP.

Součásti aplikace je inicializační skript databáze, který umožňuje databázi kdykoli uvést do „čistého“ stavu. Tento skript lze spustit pouze během vývoje aplikace (a s oprávněním root), pokud je aplikace v produkčním stavu (tento stav je uložen v nastavení aplikace), není tento skript dostupný ani root uživateli.

4.4 Nástroje využívané při vývoji projektu

Během realizace aplikace byly používány krom již zmíněných nástrojů také

- Distribuovaná systém správy verzí Mercurial.
- Pro správu závislostí a potažmo tedy instalování dodatečný modulů Zend Frameworku 2 byl využíván dependency manager Composer.
- Pro generování dokumentace z kódu byla využita knihovna Apigen.
- Repozitář extenzí a aplikací Pear

5 Testování aplikace

Testování aplikace probíhalo celkově čtyřmi způsoby. Nová funkcionalita aplikace vždy byla nejprve otestována manuálně programátorem. Pokud se jednalo o funkcionalitu, která zasahovala i do uživatelského rozhraní aplikace, tak byl následně vytvořen automatizovaný test grafického uživatelského rozhraní (otestováno tak bylo i chování aplikace v prohlížeči). A následně byly vytvářeny jednotkové a integrační testy pro novou funkcionalitu (otestována byla funkcionalita programového kódu). V průběhu celého vývoje byly také zaznamenávány požadavky na běhové prostředí aplikace (ze strany použitých technologií), na základě těchto požadavků byl následně vytvořen test běhového prostředí.

Test běhového prostředí, testy grafického uživatelského rozhraní, jednotkové testy a integrační testy budou součástí výsledné aplikace.

5.1 Test běhového prostředí

V průběhu celého vývoje aplikace byly shromažďovány požadavky na běhové prostředí aplikace, na jejichž základě byl následně vypracován test běhového prostředí za použití nástroje envtesting[30]. Ačkoli se zdá být tento test jakkoli postradatelný (aplikace se nenasazují na odlišné servery na měsíční bázi), opak je pravdou.

Celý jeden den trvalo odhalit chybu, kvůli které aplikace nefungovala korektně ve spojení s ORM frameworkem Doctrine 2. Příčinou chyby byla zapnutá knihovna eAccelerator[31] starající se o cache php skriptů. Tato chyba (nefunkčnost Doctrine 2 v závislosti na zapnuté knihovně eAccelerator) není uvedena ani v oficiálních požadavcích Doctrine 2. Pokud by tedy aplikace byla v budoucnu nasazována na server někým jiným než autorem, pravděpodobně by se dotyčný potýkal se stejným problémem, neboť knihovna eAccelerator je na mnohých hostinzích ve výchozím stavu zapnutá kvůli vylepšení výkonu php skriptů. Těmto komplikacím se však lze vyhnout, neboť již existuje test běhového prostředí, který upozorní na již známé nedostatky (viz. strana 36).

5.2 Jednotkové testování a integrační testy

Rozdíl mezi jednotkovými a integračními testy je následující. Zatímco jednotkový test testuje třídu či metodu izolovaně, integrační testy ověřují, zda spolu určité části kódu spo-

[30] OŽANA, Roman. *Envtesting – ověřujeme nastavení prostředí* [online]. 2012-07-25 [cit. 2013-04-19]. Dostupné z: http://www.zdrojak.cz/clanky/envtesting-overujeme-nastaveni-prostredi/

[31] *EAccelerator* [online]. [cit. 2013-04-19]. Dostupné z: http://eaccelerator.net/

lupracují tak, jak je očekáváno (integrační testy tak například mohou používat databázi aj. komponenty).

Jednotkové a integrační testy pro vytvářenou aplikaci byly napsány za použití frameworku PHPUnit[32]. Tento Framework umožnuje snadný způsob jak testy (testy chování jednotlivých funkcí a metod) nejen psát, ale také spouštět a analyzovat. Tento framework je nepsaným standardem pro testování PHP aplikací. Testy je možné spouštět přímo z příkazové řádky nebo (což je častějším jevem) přímo z IDE[33].

Měření pokrytí kódu jednotkovými a integračními testy nebylo provedeno. Nicméně, pokrytí kódu aplikace jednotlivými testy není zdaleka sto procentní, ale klíčové funkcionality jsou touto technologií popsané a testovatelné.

5.3 Automatizované testy grafického uživatelského rozhraní

Pro automatizované testování grafického rozhraní byl použit multiplatformní nástroj Selenium, respektive jeho komponenta Selenium IDE[34], která pracuje jako plugin[35] v prohlížeči Mozilla Firefox.

Selenium ID, dovoluje nahrávání jednotlivých kroků uživatele, kterými uživatel ovládá testovanou aplikaci. Následně je strojový popis kroků doplněn o akceptační podmínky, které rozhodují o tom, zda test prošel, či naopak. Vytváření těchto testů je velmi intuitivní a jednoduché.

Jednotlivé testy nebo kolekce testů je možné následně uložit jako html soubory, které obsahují popis testu. Nebo je testy možné vyexportovat jako sadu instrukcí do konkrétního programovacího jazyka. Z nabídky jazyků, do kterých lze testy exportovat, je možné jmenovat například programovací jazyky Java a Ruby. Tyto testy není poté nutné spouštět ze Selenium IDE, ale z konkrétního programovacího jazyka.

Testy pro vytvářenou aplikaci byly ukládány v html formátu, protože tento formát je po otevření čitelný a snadno pochopitelný i pro běžné uživatele (viz. strana 35), bez znalosti libovolného programovacího jazyka (avšak se znalostí jazyka anglického).

[32] BERGMANN, Sebastian. *PHPUnit manual - Automating Tests* [online]. [cit. 2013-04-19]. Dostupné z: http://www.phpunit.de/manual/3.7/en/

[33] Integrated Development Environment

[34] *Selenium IDE Plugins* [online]. [cit. 2013-04-19]. Dostupné z: http://docs.seleniumhq.org/projects/ide/

[35] Zásuvný modul nebo též plugin je doplňující částí jiné aplikace, které rozšiřuje funkčnost

Nevýhodou testování aplikace pomocí Selenium testů je jejich pomalost, protože běží v prohlížeči, který stránky opravdu musí načítat a vykreslovat. Testy jsou tak třeba až stokrát pomalejší než testy spouštěné z příkazové řádky.[36] I přesto se v žádném případě nejedná o zbytečnou investici času, protože tyto desítky (stovky) vteřin mohou ušetřit desítky (stovky) minut časně odhalenou chybou. To samé platí o psaní libovolných testů vůbec.

36 VRÁNA, Jakub. 1001 tipů a triků pro PHP. Vyd. 1. Brno: Computer Press, 2010, 456 s. ISBN 978-80-251-2940-1 strana 428

6 Závěr

V průběhu tvorby bakalářské práce vznikala webové encyklopedie biologických druhů, spojená s evidencí projektů Katedry biologie ekosystémů. Před vlastní implementací proběhla analýza požadavků katedry (formou řízených rozhovorů s panem doktorem Boukalem), následně návrh výsledného řešení, poslední fází popsanou v této bakalářské práci je implementace, jež byla spojena s vytvářením testů a testováním samotným. Testování probíhalo po čas celého vytváření aplikace.

Pro vytvoření webové aplikace byla zvolena metodologie WebML, která je určena výhradně pro návrh webových řešení.

Výstupem práce je funkční, ne však dokončená aplikace. Termín dokončení aplikace je naplánován (spolu s testováním katedrou a spuštěním aplikace) po termínu odevzdání bakalářské práce. Dalšími výstupy jsou dále kolekce testů, dokumentace kódu, uživatelská a administrační dokumentace.

Překvapivou se může jevit volba Zend Frameworku 2, kdy tento framework patří mezi největší a nejrobustnější řešení ve své kategorii. Jako architekt aplikace pro to mám však prosté vysvětlení. Ať už se jedná o panelový dům, nebo aplikaci, klíčové jsou pevné základy. Fasádu je možné přebarvit kdykoli, se základy však jen těžko někdo pohne.

7 Přehled použité literatury

[1] BÖHMER, Marian. Zend Framework: programujeme webové aplikace v PHP. Vyd. 1. Brno: Computer Press, 2010, 416 s. ISBN 978-80-251-2965-4

[2] BÖHMER, Marian. Návrhové vzory v PHP. 1. vyd. V Brně: Computer Press, 2012, 320 s. ISBN 978-80-251-3338-5

[3] GOLDSTEIN, Alexis, Louis LAZARIS a Estelle WEYL. HTML5 a CSS3 pro webové designéry. Vyd. 1. Brno: Zoner Press, 2011, 286 s. Encyklopedie webdesignera. ISBN 978-80-7413-166-0

[4] KEOGH, James Edward a Mario GIANNINI. OOP bez předchozích znalostí: průvodce pro samouky. Vyd. 1. Brno: Computer Press, 2006, 222 s. ISBN 80-251-0973-9

[5] SANDERS, Bill a Mario GIANNINI. Smashing HTML 5: průvodce pro samouky. Vyd. 1. Chichester: Wiley, 2010, 222 s. ISBN 04-709-7727-2

[6] THE WEBML GROUP. Overview of the WebML Development Process [online]. [cit. 2013-04-19]. Dostupné z: http://www.webml.org/webml/upload/ent17/1/webml_tr aining7_develproces s.zip

[7] PHP: Hypertext Preprocessor [online]. [cit. 2013-01-28]. Dostupné z: http://php.net/

[8] VRÁNA, Jakub. 1001 tipů a triků pro PHP. Vyd. 1. Brno: Computer Press, 2010, 456 s. ISBN 978-80-251-2940-1

[9] ZELENKA, Petr. WebML - struktura webové aplikace [online]. 2003-20-04 [cit. 2013-04-19]. Dostupné z: http://interval.cz/clanky/webml-struktura-webove-aplikace/

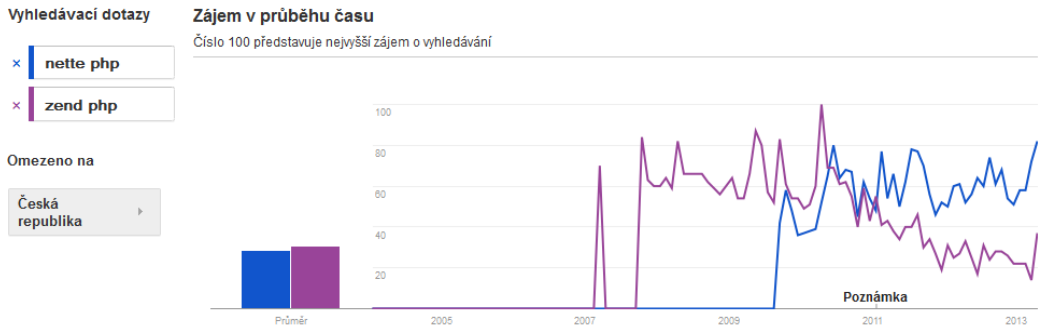
[10] ZEND TECHNOLOGIES INC. Zend Framework [online]. [cit. 2013-01-28]. Dostupné z: http://framework.zend.com/

[11] ZEND TECHNOLOGIES INC. RFC - ServiceLocator - Zend Framework 2.0 - Zend Framework Wiki [online]. 2012-03-23 [cit. 2013-04-19]. Dostupné z: http://framework.zend.com/wiki/display/ZFDEV2/RFC+-+ServiceLocator

[12] WEBRATIO S.R.L. WebRatio | Changing the IT Equations [online]. [cit. 2013-04-19]. Dostupné z: http://webratio.com

8 Seznam obrázků a tabulek

9 Přílohy



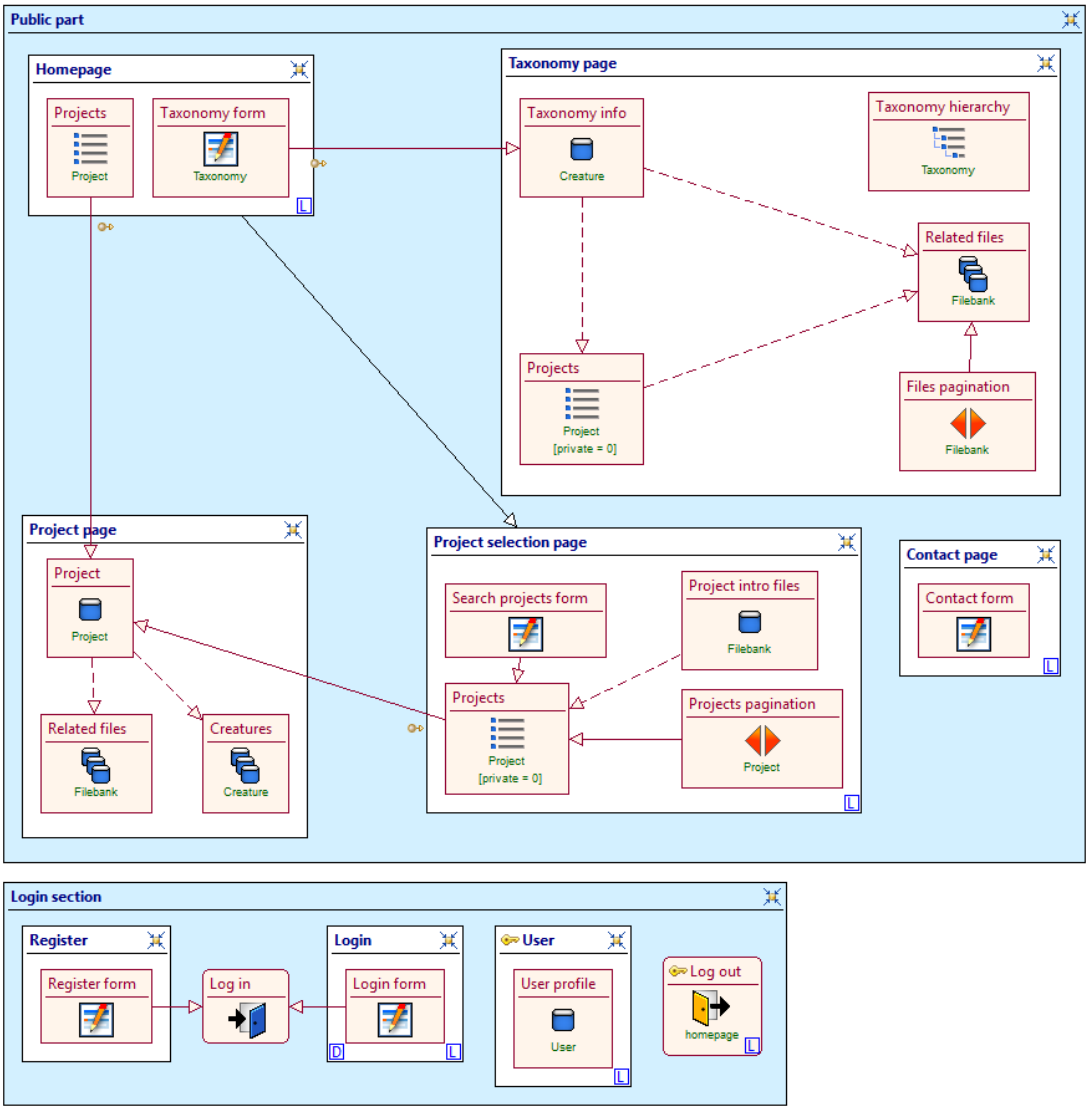
Obrázek 9-1 | Zájem o framework Nette vs zájem o Zend Framework 2 v ČR, zdroj: http://google.com/trends/

Neúspěšné přihlášení administrátor		
open	http://26813.w13.wedos.ws/public/cs/user/login	
type	name=identity	root@seznam.cz
type	name=credential	111
clickAndWait	name=submit	
verifyLocation	http://26813.w13.wedos.ws/public/cs/user/login	
open	http://26813.w13.wedos.ws/public/admin/acl/manage-rules	
verifyTextPresent	403	

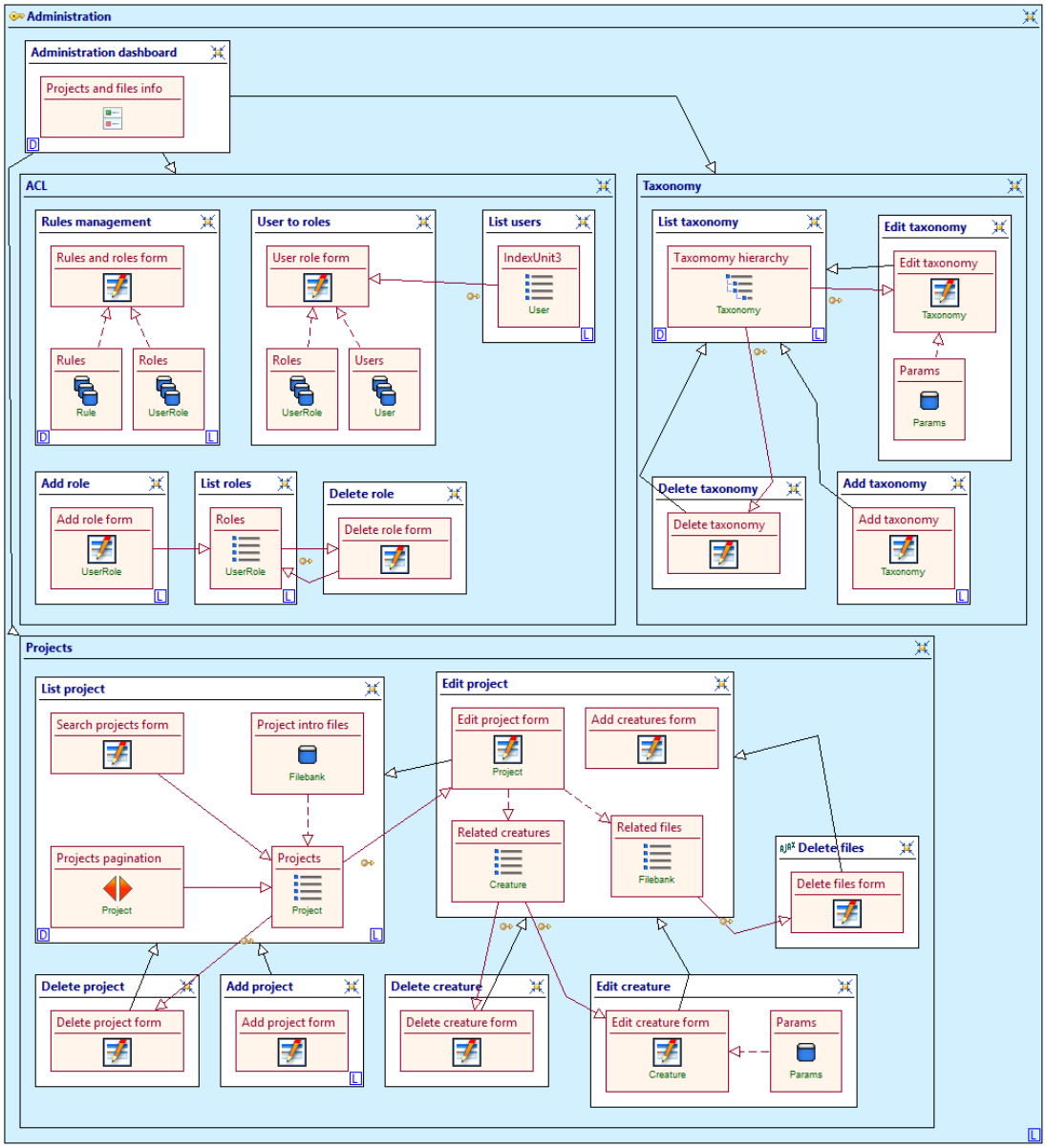
Obrázek 9-2 | Příklad Selenium testu uloženého ve formátu HTML

Obrázek 9-3 | Výstup testování běhového prostředí

bakalarska-prace-webratio-5-4-2013_homepage
Martin Sojka



Obrázek 9-4 | Rozšířený kompoziční model aplikace část 1



Obrázek 9-5 | Rozšířený kompoziční model aplikace část 2