Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Systémy pro správu webového obsahu

Diplomová práce
2009

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra softwarových technologií
Akademický rok: **2008/2009**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jan HŘÍDEL**

Studijní program: **N2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Systémy pro správu webového obsahu**

Zásady pro vypracování:

V úvodní teoretické části práce je nutné provést popis a specifika systémů pro správu webového obsahu. A přehled problematiky moderních webových technologií typických pro Web 2.0. Další částí práce bude tvorba návrhu (v UML) nového systému vyvíjeného firmou Wizards CZ, s.r.o. ve smyslu specifik Web 2.0 vystavěného na architektuře MVC (Model-View-Controller) pro tvorbu a správu webových prezentací a internetových obchodů. Součástí diplomové práce je naprogramování vybrané části systému pro správu webového obsahu za použití technologie PHP 5. Práce bude obsahovat také programátorskou dokumentaci naprogramovaného kódu.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

ARLOW, J., NEUSTADT, I. UML 2 a unifikovaný proces . Brno, Computer Press Books, 2008. SCHUMELLER, J. Myslíme v jazyku UML . Praha, GRADA, 2001. ULLMAN, L. PHP a MySQL . Brno, Computer Press Books, 2004.

Vedoucí diplomové práce: **Ing. Karel Michálek**
Ústav systémového inženýrství a informatiky

Datum zadání diplomové práce: **31. října 2008**

Termín odevzdání diplomové práce: **22. května 2009**

Univerzita Pardubice Česká republika

L.S.

doc. Ing. Simeon Karamazov, Dr.
děkan

doc. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 4. listopadu 2008

PROHLÁŠENÍ AUTORA

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně Univerzity Pardubice.

V Pardubicích dne 17. srpna 2009

Bc. Jan Hřídel

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat především svému vedoucímu diplomové práce, panu Ing. Karlu Michálkovi, DiS., za cenné rady, které mi během vypracování mé práce poskytl. A také Simoně Pietrangeli za pomoc s jazykovým překladem. Mé poděkování patří i všem ostatním, včetně mých nejbližších, kteří mi poskytli příznivé okolní podmínky pro vypracování této práce.

ANOTACE

Tato práce se zaměřuje na moderní webové technologie. Především pak na specifika v souvislosti s termínem *web 2.0*. Výsledkem práce je kompletní analýza zahrnující UML diagramy systému pro správu webových obsahů. Od případů užití až po databázový model. Součástí práce je také naprogramovaná vybraná část systému s odpovídající programátorskou dokumentací.

KLÍČOVÁ SLOVA

správa webového obsahu; web 2.0; analýza; unifikovaný modelovací jazyk; návrh; Zend framework; Ruby on Rails

TITLE

Web content management systems

ANNOTATION

The specialization of this work is modern web technology. It narrows down specifically to term "Web 2.0". The outcome is a complete analysis including UML diagram system with web contents. From use cases to database model. Part of this work is also functional demonstration of programmed components with the appropriate programming documentation.

KEYWORDS

Web content management; web 2.0; analysis; unified modeling language; design; Zend framework; Ruby on Rails

OBSAH

1 Úvod

V podstatě ihned po masovém rozšíření osobních počítačů je začali lidé využívat ke zpracovávání textových dokumentů. A pamětníci mohou potvrdit, že ne vždy to byla jednoduchá záležitost. Počítače pracovaly výhradně v textovém režimu a výsledek nějakého formátovaného textu byl vidět až ve chvíli, kdy vylezl papír z jehličkové tiskárny. Na obrazovce počítače bylo možné vidět jen text prokládaný formátovacími značkami.

Až ve chvíli, kdy se objevovaly první grafické nadstavby k operačním systémům, se také začala velice často skloňovat zkratka WYSIWYG (*What You See Is What You Get*, volně přeloženo: „To co vidíš, je to co dostaneš.“) a světlo světa spatřily první textové editory, které dokázaly nastínit, jak bude výsledek na papíře vypadat.

Toto se ovšem ještě dlouho netýkalo webových stránek, jejichž podstatou dodnes zůstává značkovací jazyk HTML. Ale i zde se s rozvojem Internetu začaly objevovat nástroje, které umožňovaly vytvářet webové stránky také neprofesionálům.

Vývoj webových technologií jde ovšem kupředu mílovými kroky. A tak se rychle objevily nástroje umožňující vizuální editaci textu webové stránky přímo z prostředí internetového prohlížeče. Což mělo za následek vývoj prvních webových aplikací zaměřených na správu webového obsahu. A právě těmto systémům a moderním webovým technologiím se budu věnovat v následujících kapitolách.

2 Základní informace o WCMS

Na začátek je potřeba si říci, co vlastně WCMS je. WCMS – někdy též označováno jako Web CMS znamená *Web Content Management System*, tedy softwarový systém pro správu webového obsahu. Ten je většinou implementován jako samostatná webová aplikace a používá se pro správu rozsáhlého množství materiálů publikovaných na webu. Tedy zejména se jedná o správu HTML dokumentů a k nim přiřazených souborů (obrázků, PDF souborů, …). WCMS usnadňuje vytváření obsahu, jeho spravování a editaci a mnoho dalších základních i pokročilých funkcí pro správu webových prezentací.

Systém obvykle poskytuje řadu nástrojů umožňujících relativně snadné vytváření obsahu webu bez nutnosti znalosti programovacích či značkovacích jazyků.

Ve většině případů je pracovním rozhraním webový prohlížeč, avšak některé WCMS vyžadují například instalaci tenkého klienta.

Systémy pro správu webového obsahu ovšem nelze brát pouze jako systémy pro správu „venkovních“ webů. Především díky kategorizaci a přístupovým právům lze jeden a ten samý zdroj informací různě interpretovat zaměstnancům pomocí intranetu, obchodním partnerům, dodavatelům prostřednictvím extranetu, či běžným uživatelům Internetu (webová prezentace, elektronický obchod).

3 Proč potřebujeme WCMS

Jedním z hlavních důvodů, proč firmy využívají různé distribuční cesty informací a proč vůbec mají webové stránky je fakt, že potřebují firmu, potažmo její produkty aktivně propagovat. A Internet s webovými stránkami je ideální cestou, jak „za málo peněz nabídnout hodně muziky“. Jinými slovy, jak za minimální náklady oslovit co možná nejvíce stávajících i potenciálních zákazníků.

Je to marketing, jehož cílem je zaujmout, sdělit správnou informaci ve správné formě a nakonec i prodat.

Pokud se tedy firma rozhodne používat kvalitní, dobře nakonfigurovaný systém pro správu webového obsahu, dostane do rukou velice silný nástroj, který jí umožní podat přehledně široké množství informací různým skupinám ve vhodné podobě. Což ocení právě zákazník.

Nejde tedy jen o to zobrazit informace na webových stránkách, ale jde o to zobrazit je cíleně a přehledně, aby potenciální zákazník neodešel s tím, že hledanou informaci nenašel, nezorientoval se v tom nepřeberném množství informací, nebo že tam ta informace není.

3.1 Požadavky na prezentované informace

Na prezentované informace, tedy veškeré texty, ceníky, dokumenty, katalogy, obrázky, … klademe obecně zejména tyto nároky:

- Měly by být aktuální, úplné a kvalitní.
- Musí se nechat dobře vyhledat, bez sáhodlouhého procházení webu.
- Zobrazovat je vždy cílené skupině uživatelů.
- Přehledně je prezentovat a vždy je předkládat jednotným stylem.

Od WCMS se pochopitelně také očekává, aby práce s ním byla jednoduchá, intuitivní a rychlá.

3.2 Úspory nákladů

Náklady, které musí firma víceméně pravidelně vynakládat na správu svého intranetu a extranetu mohou být dalším důvodem, proč se rozhodnout pro využití WCMS. Vlastnosti WCMS potom oceníme zejména s rostoucím objemem informací, které je potřeba spravovat a publikovat.

Mezi klady WCMS, které nelze popřít a oceníme je zejména právě při správě velkého rozsahu dat, patří to, že je možné weby spravovat z jednoho centrálního místa a změny je možné provádět poměrně rychle.

Proč je to vůbec možné, vychází už ze základních principů WCMS:

- Obsah mohou spravovat běžní uživatelé, není potřeba programátorů webových aplikací.
- Díky možnosti určovat komu jaká data zobrazit a díky možnosti využít externí zdroj informací není potřeba vytvářet a spravovat několik verzí stránek.

4 Běžné funkční vlastnosti WCMS

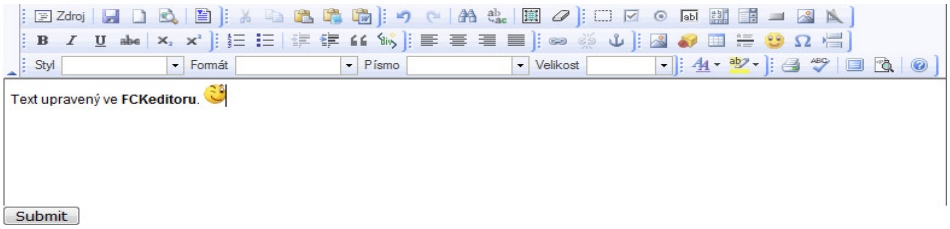
Od WCMS se v první řadě většinou očekává, že správa webových informací se přesune z IT oddělení na příslušné útvary.

Mezi klíčové vlastnosti potřebné pro správu celého životního cyklu webového obsahu patří následující.

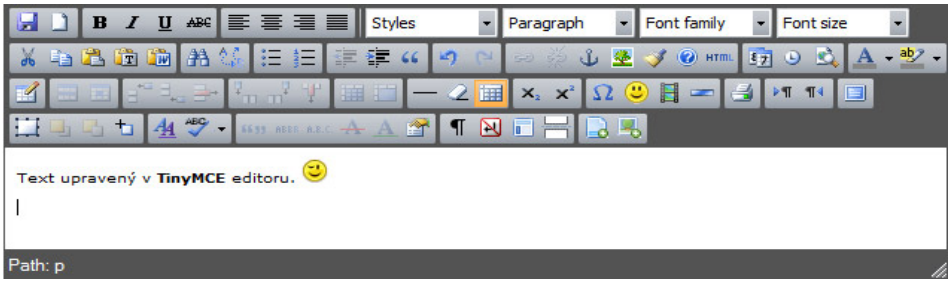
4.1 Snadné vizuální vytváření a editace obsahu

Pokud je již jednou oddělena obsahová a vizuální část prezentace webové stránky, je potom mnohem jednodušší a rychlejší měnit jak obsah, tak výsledný vzhled stránky. Mnoho WCMS obsahuje WYSIWYG editor umožňující i netechnicky zručným uživatelům snadno editovat obsah svých stránek.

Řada těchto WYSIWYG editorů bývá dostupných jako open source projekt umožňující tedy výrobcům WCMS jej upravovat a implementovat do svého systému. Mezi nejznámější open source WYSIWYG editory patří *TinyMCE* z dílny *Mexicode Systems AB* a *FCKeditor* od *FredCK*.



Obrázek 1: FCKeditor[1]



Obrázek 2: TinyMCE editor[2]

4.2 Automatické šablony

Vytváření standardních výstupních šablon – obvykle HTML nebo XML, které mohou být snadno aplikovány na ať už existující nebo na nový obsah.

[1] Zdroj obrázku: vlastní
[2] Zdroj obrázku: vlastní

4.3 Schvalovací workflow

WCMS obvykle umožňují monitorovat workflow obsahu. Workflow je proces na sebe navazujících sekvencí, popřípadě paralelních úkolů, které musí být ukončeny, aby se dostal obsah do své finální fáze a mohl být požadovaným způsobem publikován.

Příklad:

1. Redaktor napíše novou reportáž z právě uskutečněné akce.
2. Fotograf obohatí tuto reportáž o zajímavé snímky.
3. Šéfredaktor zkontroluje celou reportáž, a poté buď schválí její publikování, nebo ji vrátí k přepracování.

Do životního cyklu prezentované informace patří také její datum expirace. I když u běžných článků často bývá doba platnosti nastavena na nekonečno, naopak ankety bývají velice často časově omezené.

4.4 Automatická publikace obsahu na více webech a ve více jazykových mutacích

V dnešní době, kdy Internet v podstatě nezná hranic (pomineme-li nějakou tu čínskou cenzuru), se mnoho firem prezentuje celému světu, a tak není nic neobvyklého narazit na webové stránky, které jsou psané ve více jazycích. To proto, aby se oslovilo „co nejširší obecenstvo“.

A protože by bylo velice nepohodlné používat pro správu každé jazykové mutace obsahu webové stránky jiný (myšleno i jinou kopii téhož) WCMS, řada z nich nabízí přímo správu těchto jazykových mutací.

Nic výjimečného není ani to, že velká firma má několik různých webových prezentací, zaměřených na propagaci jednotlivých dílčích sektorů výroby. Ale některé informace, nebo například akční nabídky se týkají celé výroby, proto nabízí některé WCMS publikovat tento sdílený obsah na více webových stránkách a řídit ho z jednoho centrálního místa.

Pod publikováním informace na více webech si lze také představit zveřejnění informace na Internetu pro širokou veřejnost, na intranetu pro zaměstnance a na extranetu pro obchodní partnery.

4.5 Přebírání a publikace obsahu z jiných podnikových systémů.

Celá řada profesionálních systémů pro správu webových obsahů disponuje funkcemi pro import a export různých dat. A díky standardům a jazykům typu XML je možné tak WCMS provázat s jiným podnikovým systémem. Příkladem může být systém pro evidenci nabídek realitní kanceláře, který umožňuje nabídky vyexportovat do XML a WCMS je pak dovede importovat a velice rychle tak publikovat na webu, aniž by je musel uživatel ručně přepisovat z jednoho systému do druhého.

Ne vždy je potřeba z jiného systému přebírat kompletní informace. Mnohdy nám stačí na webu aktualizovat například pouze ceny produktů podle podnikového systému.

5 Web 2.0

Během pátrání po významu výrazu „web 2.0“[3] často na první pohled dospějeme k názoru, že pojem web 2.0 je termín či fráze, kterou nelze nijak přesně definovat. A při bližším zkoumání zjistíme, že je tomu skutečně tak, a ani to jinak být nemůže. Web 2.0 se spíše než ke klasickým definicím totiž uchyluje k souboru klíčových slov, což hezky znázorňuje tento „tag cloud“ – pole vzájemně provázaných klíčových slov – od Markuse Angermeiera.

Obrázek 3: tag cloud web 2.0[4]

5.1 Pojem web 2.0

Pojem web 2.0 se poprvé objevil, když zástupci firmy *Medialive Internationl* a *Tim O'Reilly* jednali v roce 2004 o názvu konference věnované moderním webovým trendům. Označení web 2.0 jim přišlo vhodné hned ze dvou důvodů.

1. „2.0“ označuje v softwarovém světě novou verzi projektu, která by měla být zásadně lepší než ta původní.
2. Jako metafora pro „druhý dech“, který bylo nutné nabrat po pádu internetových společností. V březnu roku 2000 totiž dosáhl technologický index NASDAQ hodnoty 5 048 bodů a následně se propadl pod 2 000. To vedlo ke krachu mnoha internetových projektů.[5]

[3] Velice často se setkáváme s tím, že se píše Web 2.0 s velkým počátečním písmenem, a to především v anglických textech. Ovšem jelikož se nejedná o žádný název, ale spíš o termín popisující trend, budu psát web 2.0 s malým „w“.

[4] Zdroj obrázku: Markus Angermeier: http://nww.nerdwideweb.com/web20/#web20en

[5] Dot-com crash (dot-com bubble): http://en.wikipedia.org/wiki/Dot_com_crash

Web 2.0 ovšem neznamená nějakou změnu například v technologických standardech, jako změnu HTML apod. Jedná se spíš o změnu přístupu. V dalším pokusu o definici termínu web 2.0 se vyjádřil Tim O'Reilly takto:

„Web 2.0 je revoluce podnikání v počítačovém průmyslu způsobená přesunem k chápání webu jako platformy a pokus porozumět pravidlům vedoucím k úspěchu na této nové platformě. Klíčovým mezi těmito pravidly je toto: tvořte aplikace, které budou díky síťovému efektu s přibývajícím počtem uživatelů stále lepší. (Což jsem jinde nazval , zapřažení kolektivní inteligence '.)"[6]

Osobně však považuji za velice trefné „definování" termínu web 2.0 vyjádření se Rosse Mayfielda:

„Web 1.0 was commerce. Web 2.0 is people."[7]

Toto obecné pravidlo lze rozvést do mnoha specifičtějších, do různých oblastí. Tím pádem ve výsledku vzniká skupina pravidel (vlastností) popisující služby zapadající do kategorie web 2.0. Řada takových aplikací a služeb, které se dnes řadí pod web 2.0, však vznikla už i dříve. Příkladem budiž *Wikipedia* (2001) nebo *del.icio.us* (2003).

Na termín web 2.0 se ovšem můžeme podívat i z jiné stránky, a to sice jako na nálepku, která „prodává". A především ve Spojených státech amerických se na „značku" web 2.0 skutečně slyší. A mnoho vznikajících projektů, jejichž zakladatelé sní o odkoupení firmami jako *Yahoo!* nebo *Google*, tak nese toto označení a nápis „beta" v logu.

5.2 Charakteristiky web 2.0

Jednotlivých charakteristik, které by přesněji popisovaly, co web 2.0 znamená, by mohlo být určitě mnoho. Budu-li se ovšem držet původního O'Reillyho článku[8] ze září 2005 a doplním-li to něčím aktuálnějším, získám následující klíčová vystižení.

5.2.1 Web jako platforma (koncentrace dat)

Pro úspěch jakéhokoliv projektu web 2.0 bývá stěžejní získání velkého množství unikátních dat. A to dokonce tak, že množství dat často bývá důležitější než jejich kvalita. Webový projekt *MySpace* získal dokonce podle renomovaného časopisu *PC World*[9] cenu nejhorší stránky všech dob. Obecně je kritizován ať už pro zabezpečení dat, nebo pro svůj design. Přesto samotné množství uživatelů (více než 200 milionů profilů) vytváří obrovskou hod-

[6] O'Reilly, Tim. 12. 10. 2006. *Web 2.0 Compact Definition: Trying Again.* http://radar.oreilly.com/archives/2006/12/web_20_compact.html (cit. 10. 5. 2009)
[7] Singel, Ryan. 6. 10. 2005 *Are You Ready for Web 2.0?* http://www.wired.com/science/discoveries/news/2005/10/69114 (cit. 10. 5. 2009)
[8] O'Reilly, Tim. 30. 9. 2005. *What Is Web 2.0* http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html (cit. 10. 5. 2009)
[9] Tynan, Dan. 25. 9. 2006. *The 25 Worst Web Sites.* http://www.pcworld.com/article/id,127116-page,7-c,sites/article.html (cit. 10. 5. 2009)

notu (přes 6 miliard dolarů)[10]. Podobně je na tom třeba *eBay* – služba těžící ze své silné uživatelské základny.

V souvislosti s termínem web 2.0 se také často objevuje rčení: „web (Internet) jako platforma". Na svém poměrně prestižním blogu *Read/Write Web* píše Richard MacManus doslova:

„Shromáždil jsem řadu skvělých definic… sám ale preferuji lapidární: Web jako platforma. Umožňuje mi to doplnit to podle toho, s kým mluvím. Pro lidi z korporací: platforma pro obchod. Pro marketing: platforma pro komunikaci. Pro novináře: platforma pro nová média. Pro počítačové nadšence: platforma pro vývoj nového software."[11]

Tomu, abychom chápali Internet jako platformu, nám napomáhá i fakt, že Internet se stal každodenní realitou pro více než miliardu lidí. A už dávno neplatí fakt, že Internet je záležitostí USA a západní Evropy. Dnes se řadí mezi největší skupiny uživatelů především Asiaté (41,2 %).[12]

5.2.2 Many-to-many (změna komunikačního modelu)

Vin Crosbie, coby americký mediální teoretik, dělí média do tří skupin, a to podle jejich povahy:[13]

1. *One-to-one* – interpersonální média
2. *One-to-many* – masová média
3. *Many-to-many* – nová média umožňující vybírat si a personalizovat data

Obrázek 4: Vin Crosbie[14]

[10] Savitz, Eric: *Report Murdoch Says MySpace Worth $6 Billion* http://blogs.barrons.com/techtraderdaily/2006/11/14/report-murdoch-says-myspace-worth-6-billion-sees-200-million-users-by-mid-2007/ (cit. 10. 5. 2009)

[11] MacManus, Richard. 1. 2. 2005. *Web 2.0 Definition and Tagging* http://www.readwriteweb.com/archives/web_20_definiti.php (cit. 20. 5. 2007)

[12] Statistika z března 2009: http://www.internetworldstats.com/stats.htm

[13] Vin Crosbie: *What is New Media?* Dostupné z: http://www.sociology.org.uk/as4mm3a.doc

Podle Vina Crosbieho je tak médium chápáno spíše jako komunikační model a noviny, rozhlas, televize či Internet jsou chápány více jako nositelé médií.

Na jednu stranu řadí první dva modely, které nepotřebují pro svou existenci technologie, ale mohou je využít k překonání bariér. U modelu *one-to-one* to pak může být například telefon nebo dopis (i elektronický). U modelu *one-to-many* potom noviny, rádio či televize. Přičemž jako masové médium je chápána jakákoliv situace, kdy jedinec (nebo skupina) oslovuje masu. Tedy i jakýkoliv proslov, bohoslužba či divadlo. Výhodou prvého modelu je možnost kontroly nad tokem informací a jejich okamžité ovlivňování a přizpůsobení jednotlivci. U masové komunikace potom shledává Vin Crosbie výhodu v obrovském dosahu komunikace, která je už ale jen jednosměrná a nelze ji jednotlivci přizpůsobit.

Na druhou stranu je postaven třetí model (*many-to-many)*, který se bez technologie už neobejde. Crosbie vidí tento model jako možnost překročení hranice k masové individualizaci. S rozvojem Internetu se totiž zvyšuje možnost oslovovat široké publikum se zachováním individuálního přístupu. A web 2.0 toto činí jednodušším.

Pokud bychom si to měli ukázat na příkladech:

Při tvorbě blogu se autor automaticky zapojuje do skupiny mnoha set tisíc či milionů „blogerů“, a jeho příspěvky na blog se indexují v nejrůznějších katalogách a vyhledávačích.[15] V některých se pak vedle sebe objevují odkazy na články jak renomovaných zpravodajských serverů, tak na články z blogů neznámých autorů. Zcela běžně bývá potom oddělena forma od obsahu, a to za pomoci rozšířených a standardizovaných formátů jako jsou *RSS2* nebo ATOM. Ty nám umožňují převzít z původní stránky jen obsah. Uživatel si jen nadefinuje, ať už v nějaké online čtečce[16] nebo v desktopové aplikaci k tomu určené, kanály, které chce pravidelně odebírat. Čtečky RSS kanálů bývají také velice často integrovány přímo do webových prohlížečů a e-mailových klientů. Odebírané zdroje se pak načítají v pravidelných časových intervalech samy, většinou jen titulek, datum, autor a anotace ke článku. A záleží na uživateli, co a kdy si bude chtít přečíst.

A skutečně nejde jen o masová média. Budeme-li mít na jedné straně obsáhlé zpravodajské servery a na druhé osobní deníčky s dosahem na pár desítek uživatelů, vzniká nám někde mezi nimi velké množství zdrojů informací. Ty zaplňují místo po úzce specializovaných časopisech zaměřených na čtenáře s různými zájmy. Ať už to jsou pěstitelé okurek či fanoušci vojenského letectví. Tyto specializované zdroje bývají tedy sledovány jen menší skupinou čtenářů, ale mohou si ve svém oboru vydobýt pozici respektovaných zdrojů.

---

[14] Zdroj obrázku: CROSBIE, Vin. *Digital Deliverance staff | Digital Deliverance LLC* (cit. 10. 5. 2009). Dostupný z WWW: http://www.digitaldeliverance.com/blog1/about/

[15] Rozdíly mezi webovými katalogy a fulltextovými vyhledávači popisuji ve své bakalářské práci dostupné v knihovně Univerzity Pardubice nebo na URL adrese: http://tinyurl.com/bp-jan-hridel

[16] Mezi nejznámější online čtečku kanálů patří například Google Reader na adrese: http://www.google.cz/reader

Na tomto poli se na Internetu objevily další zajímavé a úspěšné služby. Jejich podstatou je sdílení různých, podle běžných uživatelů zajímavých, článků a novinek. Dochází tak k velice úzkému prolnutí renomovaných zpravodajských serverů a „obyčejných" blogů. Mezi nejznámější patří servery *Delicious*, *Digg* a *Reddit*. V Česku je to potom server *Linkuj.cz* a na Slovensku portál *Sme*. Všechny tyto službu fungují na velice podobném principu. Zaregistrovaní uživatelé zadávají podle svého mínění zajímavé články (adresu článku, kategorii, krátký popisek) a ty se pak objevují na stránce novinek, kde jej mohou hodnotit ostatní uživatelé. A články, které jsou nejlépe hodnocené, se objevují na titulní straně.

Digg.com byl založen koncem roku 2004 a v březnu 2007 už měl přes více jak milion registrovaných uživatelů. Ti se navíc mohou vzájemně přidávat „do přátel" a uživatelé pak vidí odlišné zprávy ze svého okolí. Digg.com se také zařadil mezi prvních sto nejnavštěvovanějších stránek.

Server *Reddit*, který byl v roce 2006 odkoupen firmou Condé Nast (vlastník také časopisu Wired) je obdobou serveru Digg. Ovšem nabízí zajímavou funkci navíc. A to takzvané „pohřbívání" článků. Čili opak funkce „digg it". Čímž mohou uživatelé článek také označit jako špatný. Pokud ho takto označí více uživatelů, článek z titulní strany zmizí a bude k vidění jen na profilové stránce uživatele. Tím se vlastně samotní uživatelé starají o obsah serveru a bez pomoci editorů „pročišťují" jeho obsah.

Tím je v podstatě umožněno mase, aby individualizovala obsah pro masu. Tedy typ média many-to-many.

5.2.3 Hranice mezi producenty a konzumenty

Spíše se jedná o jakési stírání a rozmlžení té pomyslné hranice mezi konzumenty a producenty. Alvin Toffler (americký spisovatel a futurista) psal již v 80. letech o vzniku takzvaných „prozumentů" (spojení slov producent a konzument).

Tyto vize lze dnes sledovat například v používání hypertextové navigace umožňující nacházet řadu cest v „otevřeném" textu, nebo například v přidávání uživatelských komentářů pod články či hodnocení výrobků. Ale také v možnosti snadného tvoření a distribuci hudby a videa s nástroji technologické kvality, které byly dříve dostupné jen pro specializovaná studia.

Je to právě rozšíření Internetu a nástup blogů, komunitních serverů a komunikační model many-to-many, které uvedly při rozmachu dostupných technologií distribuční cesty i běžným konzumentům. Nemůžeme ovšem tvrdit, že by se ze všech konzumentů stali „prozumenti". Podle statistiky prezentované firmou Hitwise[17] na Web 2.0 expo v dubnu 2007 bylo pouze 0,16 % návštěv na YouTube spojených s nahráním videa, 0,2 % návštěv na Flickru s nahráním obrázku a 4,5 % návštěv Wikipedie bylo spojeno s editací textu. Z tohoto by se mohlo zdát, že ta skupina lidí, která se podílí na tvorbě obsahů webů, je poměrně dosti

[17] Měření provedeno na vzorku 10 milionů Američanů: http://www.hitwise.com/press-center/hitwiseHS2004/web20.php

malá. Ale to je pochopitelné, protože si musíme uvědomit, že uživatelé, kteří videa nahrávají, je ještě o to více sledují. A nesmíme ani zapomenout na fakt, že právě na YouTube není obsah tvořen jen nahráváním videí, ale v poslední době také psaním komentářů, hodnocením příspěvků a vytvářením play-listů z již nahraných videí. Navíc přibyla i možnost publikovat video z YouTube na jiných webových stránkách.

Trochu odlišný pohled na věc mají Dmytri Kleiner a Brian Wyrick, kteří tvrdí ve svém eseji InfoEnclosure 2.0[18], že web 2.0 je v mnoha ohledech primitivní a je jen plynulým následkem vývoje technologií. A že se změnila především povaha projektů, na nichž se pracuje. Většina velkých web 2.0 projektů vznikla jako úspěšné takzvané „start-upy", na nichž jejich zakladatelé zbohatli. Přitom stačilo mít ten správný nápad a vytvořit „jednoduchý" framework, pomocí něhož už uživatelé doplnili obsah a sami tak ve svém volném čase vybudovali „život" a úspěch těchto projektů.

Ale mění se i postoj firem, které si uvědomují sílu komunikace se svými potenciálními zákazníky. Společnosti proto také zakládají firemní blogy, aby ukázaly i svou „lidskou tvář".

S nástupem amatérských producentů souvisí také následující bod: rozvoj wiki systémů.

5.2.4 Wiki systémy

Wiki znamená v havajštině *rychle*. A wiki systémy jsou typické svou možností rychle editovat text na stránce kýmkoliv. Nejznámějším příkladem je zcela jistě encyklopedie *Wikipedia* (datující svůj vznik do roku 2001), ale není jediná. Systémy wiki našly své uplatnění na mnoha platformách a pro různé účely.[19]

Možnost editovat a přidávat stránky je spojena s vlastností mít dostupné všechny předchozí verze stránky. Což je důležité vzhledem k tomu, že otevřený systém může být velice náchylný na situaci, kde se neodbornou editací může poškodit předchozí práce. Přestože se najde mnoho kritiků wiki systémů, ty nejoblíbenější, jimiž jsou *Media Wiki*, *Twiki* či *Kwiki*, přesáhly už v roce 2004 hranici milion stažených kopií ze serveru *Sourceforge*. Wiki systémy veřejně přístupné na webu jsou jen pouhou „špičkou ledovce". Mnohem častěji se tyto systémy používají uvnitř firem, kde nahrazují statické intranety.

Mezi problémy otevřených wiki systému dnes patří záměrné poškozování („vandalismus") cizích obsahů nebo jejich upravování v zájmu prosadit, vlastní například politické, názory. Příkladem může být i český server *Mozek.cz*.

Mezi další neduhy těchto systémů patří šíření spamu. Proti tomu bojuje i Wikipedia, která zavedla atribut „nofollow" u všech odkazů. Stránky, na něž tyto odkazy směřují, tak nezískávají alespoň žádný PageRank.

---

[18] Kleiner, Dmytri – Wyrick, Brian. „InfoEnclosure 2.0" (Web 2.0 – Man's best friendster?): http://www.metamute.org/en/InfoEnclosure-2.0 (cit. 10. 5. 2009)

[19] Seznam dostupných wiki systémů: http://en.wikipedia.org/wiki/List_of_wiki_software

Kritici wiki systémů a zejména Wikipedie varují před přílišnou otevřeností tohoto systému. Obávají se totiž, že možnost volné editace vede ke snížení kvality uveřejněných informací. Tito kritici se domnívají, že je nezbytné, aby pro zachování kvality vyčlenila co možná nejmenší skupina editorů, kteří by projekt řídili.

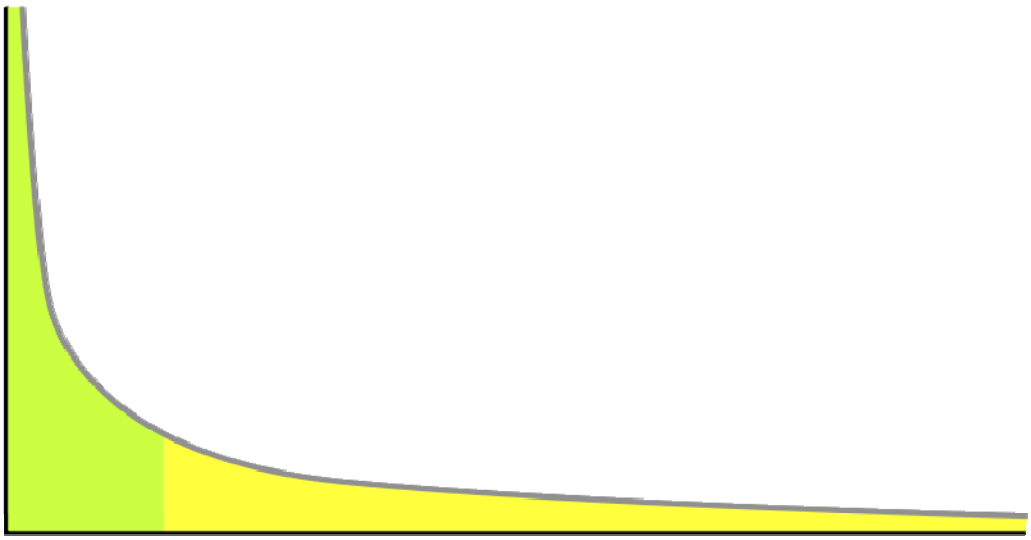
Proti těmto kritikům se veřejně postavili Douglas Rushkoff a Quentin Hardy, kteří zejména vytyčují skutečnost, že wiki systémy dávají možnost přesunout systémy kontroly obsahu (a tedy i možnost opravit chyby) do mnohem širších kruhů než dříve a dávají tím možnost projevit se individualitám. Dále zmiňují fakt, že možnost chyby nahlásit okamžitě, případně je ihned opravovat je velice silnou stránkou. Protože zatímco chyby na Wikipedii mohou být opraveny takřka okamžitě, chyby v encyklopedii *Britannica* zůstanou přinejmenším do jejího dalšího vydání.

V kontextu webu 2.0 je velice zajímavým příkladem využití wiki systému na serveru Last.fm, kde mají uživatelé možnost editovat profily hudebních umělců. Tím se objevuje možnost synchronizovat jak faktické informace, tak třeba problémy s diakritikou. Stejně tak lze celkem jednoduše přidávat informace o chystaných (nebo již uskutečněných) koncertech a festivalech.

5.2.5 Dlouhý chvost (long tail)

Long tail je výraz známý spíše z marketingu. Obecně se jedná o termín, který popisuje prodejnost široké škály výrobků při jejich menší poptávce.

Chris Anderson ve své knize *Long Tail: Why the Future of Business Is selling Less of More* tvrdí, že Internet pomalu mění současnou ekonomiku v ekonomiku a kulturu „okrajových" produktů určených menšinám.



Obrázek 5: Long Tail[20]

[20] Zdroj obrázku: *Dlouhý ocas – Wikipedie, otevřená encyklopedie* [online] dostupné z WWW: http://cs.wikipedia.org/wiki/Long_tail

Na Internetu long tail představují v oblasti publikování blogy, v oblasti inzerce jsou to služby *AdWords* a *AdSense* společnosti Google, která tak dává možnost inzerovat i menším firmám, pro které by byla inzerce v tisku či v televizním vysílání finančně nepřijatelná (služba AdWords). AdSense naopak umožňuje provozovatelům stránek zobrazovat placenou reklamu na svém webu. Tím jim dává možnost příjmů, které by kvůli své malé návštěvnosti neměli vůbec.

S rozšířením nabídky se ovšem vyskytuje problém s volbou. Ten by měly řešit reputační systémy (viz dále).

5.2.6 Reputační systémy

Reputační systém je mechanismus kombinující technologii se sociologií. V podstatě jde o to, že uživatelé z nějaké množiny neustále hodnotí ať už sebe navzájem, tak společně například nějaký konkrétní produkt, službu, firmu, … Výsledkem takovýchto hodnocení bývá nejčastěji nějaká číselná hodnota, která udává reputaci každého konkrétního uživatele, produktu, …

Reputační systémy hrají na poli internetu velice důležitou roli, protože lidé obvykle dají více na osobní zkušenosti ostatních než na vábivou barevnou reklamu.

Takovými průkopníky reputačních systémů na Internetu jsou projekty jako *Slashdot*, *eBay*, *Amazon* a další.

Slashdot je známý zpravodajský portál zaměřený na vědu, technologie, počítačové hry a podobně. Ke článkům vycházejícím na tomto portálů začalo ochotně přispívat tisíce registrovaných uživatelů. Ve chvíli, kdy jejich příspěvky začaly převyšovat únosnou míru, byl zaveden pojem takzvané „karmy“: uživatelé hodnotili sami sebe a své příspěvky co do schopností, tak do důvěryhodnosti. A čím vyšší karmu každý uživatel měl, tím získal větší práva moderovat diskuse. Ovšem pokud získal nějaký uživatel vysokou karmu na dlouhou dobu, byla mu automaticky snížena, aby nedocházelo k prosazování jednoho názoru a tím se stávaly diskuse objektivnější. A tak mohl i obyčejný čtenář nahlížet na každý příspěvek s určitou prioritou. Daná hodnota karmy mu totiž prozrazovala, jak moc oceňují tohoto uživatele a tento příspěvek ostatní.

Principy reputačního systému má ve svém jádru také nejprestižnější internetový vyhledávač Google. Ten na základě mnoha činitelů přiděluje stránkám hodnocení PageRank, které je důležitým faktorem pro konečnou pozici webu ve výsledcích hledání. Umožňuje také například přihlášeným uživatelům, aby si sami ohodnotili jednotlivé výsledky hledání a tím přispěli ke změně reputace konkrétní webové stránky.

5.2.7 Webové aplikace nahrazující desktopové

Kvalita webových aplikací je úzce propojena s kvalitou a rozšířeností vysokorychlostního internetu do domácností. Využívání webových aplikací domácnostmi totiž představuje důležitý psychologický moment. A to sice důvěru v poskytovatele, že osobní data nebudou zneužívána. Ta jsou totiž uložena na straně poskytovatele a ne na lokálním pevném disku.

Nutno podotknout, že mnohdy bývají data na straně poskytovatele zabezpečena mnohem lépe než na straně klienta.

Za předchůdce webových aplikací lze obecně považovat webmailové služby. Ty se začaly na Internetu objevovat již v roce 1996 jako alternativa k poštovním klientům. Revoluci v tomto směru přinesl v roce 2004 Google, když spustil svůj *Gmail* s kapacitou schránky 1 GiB[21], fulltextovým vyhledáváním a ajaxovým rozhraním. Následovaly je služby pro správu internetových záložek, které mohly přidat navíc ještě sociální rozměr, a to díky možnosti sdílení vybraných záložek s ostatními. Podobně to je i s webovými textovými a tabulkovými editory. V dnešní době nechybí například díky projektu *Phixr* ani online grafické editory pro úpravu fotografií. Mezi další patří například RSS čtečky, chatovací programy, organizátory a diáře.

5.2.8 Mashup

Mashup aplikací lze nazvat takový program, který využívá API jiných aplikací a jejich kombinací případně přidáním vlastních algoritmů vytváří nové.

Na počátku vzniku mashup aplikací to byly především mapy (Google Maps, Yahoo! Mapy, u nás mapy od Seznamu a Atlasu), které se staly inspirací pro další aplikace a služby. Takovým mashupem je například projekt *Panoramio*[22], který umožňuje uživatelům přidávat do Google map fotografie z celého světa.

Populárními se staly také mashupy, které jen stahují žhavé aktuality z velkých serverů, které následně hromadně publikují. Příkladem může být povedený projekt *Popurls*[23], který shromažďuje data z více než 15 velkých agregačních serverů.

Oblíbené také jsou služby kombinující výsledky vyhledávání. Například služba na adrese www.musictonic.com nabídne ve výsledcích hledání (zaměřeno na hudební kategorii) video z YouTube, seznam podobných interpretů z Last.fm a alba, která lze zakoupit na adrese nejznámějšího internetového obchodu Amazon.com.

5.3 Příklady web 2.0 služeb

V této části se pokusím přiblížit konkrétní aplikace, které mají prvky společné s termínem web 2.0 tak, jak byl popsán výše. Jelikož by se dalo vybírat ze stovek různých aplikací, pokusím se subjektivně vybrat ty nejnavštěvovanější, nejvíce oblíbené, nebo ty, které jsou něčím skutečně originální a zajímavé.

5.3.1 del.icio.us

Del.icio.us je služba, jež založil v roce 2003 Joshua Schachter. Ten chtěl původně jen malý nástroj, který by mu umožňoval lepší orientaci ve vlastních záložkách. Naprogramoval si

[21] Dnes má emailová schránka Gmail kapacitu 7,36 GiB a stále roste.
[22] Dostupný na URL: http://www.panoramio.com/
[23] Dostupný na URL: http://popurls.com/

tedy aplikaci, která mu umožňovala si ke každému odkazu přiřadit tagy (štítky – klíčová slova), které by ho charakterizovaly.

Když si Joshua Schachter uvědomil, že nástroj by se mohl hodit i jiným uživatelům, zpřístupnil aplikaci pro širokou veřejnost a ta ji vřele přijala.

Pomineme-li sociální aspekt (sdílení záložek s veřejností), služba má význam i pro jednotlivce. Ukládá si záložky svých oblíbených či jinak zajímavých webů na jedno místo, kde je má pomocí štítků přehledně kategorizované a přístupné odkudkoli – z libovolného prohlížeče, z libovolného počítače připojeného k Internetu. Své záložky najde uživatel velice jednoduše, poněvadž hned po jednoduché registraci získá svou unikátní URL adresu, která bude ve tvaru *http://del.icio.us/uzivatelskejmeno* nebo *http://delicious.com/uzivatelskejmeno*, na níž se budou uložené záložky zobrazovat. U každé takto uložené záložky se také zobrazuje její popularita – číslo říkající, kolik uživatelů má tuto záložku uloženou.

Jako nevýhoda této služby bývá často označováno to, že uživatelé zahlcují systém redundantním obsahem i tagy, nebo to, že systém lze využívat k jisté formě spamování.

Naopak mezi výhody se řadívá to, že lze procházet záložky libovolného uživatele. Každý uživatel si může vytvářet síť svých přátel, jejichž záložky chce sledovat dlouhodobě. A i takto vzniklý seznam ze záložek svých přátel je veřejný na adrese ve tvaru *http://delicious.com/network/uživatelskéjmeno*.

Další funkcionalita spočívá v možnosti sledování určitého tagu. Například budeme-li chtít sledovat záložky, které jsou označené tagem „web20“. Najdeme je na adrese *http://delicious.com/tag/web20*. Toto je ideální pokud hledáte informace k nějakému tématu a spíš vám záleží na tom, co považují za dobré lidé a ne vyhledávací roboti. Del.icio.us se tak stává svým způsobem i vyhledávačem, do kterého přidávají výsledky lidé.

Mezi další oblíbené možnosti patří i uveřejňování svých záložek na vlastním webu.

Obdobným projektem jako del.icio.us je například *Furl*[24]. Hlavní rozdíl je v tom, že tady si registrovaný uživatel neukládá pouze odkaz doplněný o tagy, ale přímo si archivuje celou stránku na disk (na serveru má k tomuto účelu prostor o kapacitě 5 GiB).

5.3.2 Last.fm

Last.fm založili původně čtyři studenti jako netradiční internetové rádio už v roce 2002. Chvíli na to vyhráli i několik internetových ocenění[25]. Posluchači začali vytvářet komunitu, která si vyměňovala svoje tipy na oblíbenou hudbu. Fungoval tu takový princip, že když uživatel A poslouchá skupiny 1, 2, 3 a 4 a uživatel B poslouchá skupiny 2, 3, 4 a 5, bylo více než pravděpodobné že bude výhodné vyměnit si informace o skupinách 1 a 5 navzájem. Uživatelé během poslechu hodnotili jednotlivé písně stylem „líbí/nelíbí“, čímž se upravo-

[24] Projekt je dostupný na adrese http://www.furl.net

[25] Konkrétně se jednalo o soutěž Europrix (zaměřenou na studentské práce v oblasti multimedií) a o prestižní ARS Electronika.

val jejich profil, který se řadil mezi ostatní podobné profily, a na jejich základě se vybírala další skladba. Z počátku se jednalo spíš jen o méně známé umělce, což pro ně byla obrovská příležitost prosadit se.

Posluchačů tohoto internetového rádia totiž rapidně přibývalo. Z počátečních pár set se tento počet na konci roku 2006 přehoupl přes 15 milionů.

Dnes má Last.fm své konkurenty. Jedním z nich je například *Pandora*. I když ta se trochu liší způsobem výběru skladeb. Nejde o podobnost založenou na sociální síti (posluchačích), ale na vlastnostech hudby.

5.3.3 YouTube

YoutTube[26] představuje službu pro videohosting a přehrávání videozáznamů. Od svého založení v roce 2005 zaznamenala jeden z nejrychlejších růstů počtu uživatelů vůbec. Úspěch této společnosti vedl k tomu, že už na podzim 2006 ji odkoupil Google za 1,65 miliardy amerických dolarů. Čímž vlastně získal (společně se svým Google Video) dominantní postavení také na poli služeb nabízejících videohosting.

Ačkoli už dříve existovaly podobné služby (MetaCafe.com vzniklo v roce 2003) nebo třeba jiné služby nabízející i lepší kvalitu videí, YouTube si svou oblibu získal lidskou silou – počtem aktivních uživatelů.

Denně YouTube přehraje neuvěřitelných 100 000 000 videí.

YouTube dokázalo nabídnout několik vynikajících vlastností. Ihned po registraci může uživatel začít nahrávat videa v různých formátech (MPEG, MOV, AVI, WMV a další), jedinou podmínkou je, že zdrojové video nesmí být delší než 10 minut a nesmí přesáhnout velikost 1 GiB. Poté je automaticky konvertováno do formátu H.263, takže k jeho přehrání stačí libovolný obyčejný Flash Player, jenž byl už v červnu 2006 součástí více jak 90 % používaných prohlížečů.

Uživatelé si mohou videa přidávat mezi svá oblíbená nebo je řadit do playlistů, které je potom možno sdílet s ostatními uživateli. Dále mohou videa hodnotit (počtem hvězdiček) a komentovat. Samozřejmostí je vyhledávání videí postavené na platformě Google. Také přibyla velice zajímavá funkce přímých „videoodpovědí".

Uživatelé mohou vytvářet komunitní skupiny týkající se téměř jakéhokoliv tématu od politiky a sportu až například po chovatele akvarijních rybiček.

Nechybí ani rozsáhlé statistiky, díky kterým lze zjistit, která videa jsou nejsledovanější, či které má nejrozsáhlejší diskusi.

V minulosti server YouTube nabídl také řadu živých přenosů.

[26] Projekt je dostupný na URL adrese: http://www.youtube.com

Služba nabízí také od počátku možnost vložit si video z YouTube na vlastní webové stránky.

Vedle řady výhod je s YouTube (ale i s jinými videohosting) spojena také řada nevýhod, mezi něž často patří soudní pře (převážně o porušování autorských práv) nebo výskyty videí, které nabádají například k násilí a podobně.

5.3.4 Google

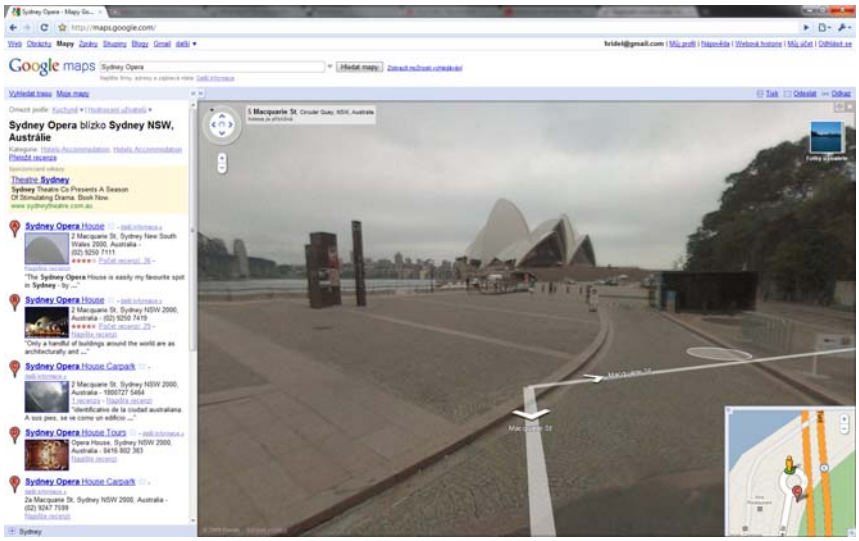
Společnost *Google* představuje bezpochyby internetového giganta, o kterém už asi každý slyšel. Vedle svého populárního vynikajícího vyhledávacího enginu nabízí ale také celou řadu dalších produktů a služeb a celá řada produktů ještě čeká „za dveřmi". Google se stal také v posledních letech nejdražší světovou značkou[27], když překonal takové společnosti, jakými jsou CocaCola, Microsoft, IBM či General Electric.

Mezi nejzajímavější Google produkty patřící do kategorie web 2.0 spadají jistě Gmail, Google Maps, Google Docs, Google Reader, Google Video a další.

Google je považován za hnací motor Webu 2.0. Poskytuje primárně služby (ne software), snaží se své produkty individualizovat, základem produktů je AJAX, využívá long tail (ve vyhledávání i v obchodní strategii), neustále inovuje své produkty.

Google Maps

Osobně považuji Google Maps za jeden ze stěžejních produktů této společnosti a za průkopníka web 2.0. Google přidává postupně do svých map nové funkcionality. Od satelitních snímků, přes ukládání si vlastních map a plánovač trasy až po rozšíření *StreetView*.



Obrázek 6: Pohled na Opera House v Sydney z Google maps[28]

[27] Google se drží na vrcholu žebříčku nejdražších značek světa už 3 roky po sobě. V roce 2006 to bylo s hodnotou 66 miliard dolarů, v roce 2009 již s rovnými 100 miliardami dolarů. Zdroj: http://hn.ihned.cz/c1-36901980-nejdrazsi-znacka-sveta-google

[28] Zdroj obrázku: vlastní

Google Reader
Google Reader představuje populární RSS čtečku, která umožňuje uživateli kategorizovat si své odběry, sdílet je s přáteli a připisovat k nim vlastní poznámky.

Google Docs
Google Docs představuje online kancelářský balík disponujícími nástroji pro úpravu textů (online obdoba MS Word), tabulek (online obdoba MS Excel), prezentací (online obdoba MS PowerPoint)

Google Video
Google Video byl původně vyvíjený projekt, který se měl stát obdobou YouTube, ale jelikož nedosahoval takové popularity, jako se očekávalo, rozhodl se Google YouTube odkoupit a z Google Video se stal vyhledávač videí na Internetu.

5.3.5 Facebook

Facebook[29] je představitelem takzvané sociální sítě, nebo možná přesněji nástrojem na tvorbu sociálních sítí. Sdružující se uživatelé z celého světa zde sdílí se svými přáteli fotografie, videa, zajímavé odkazy, aplikace a osobní informace.

V České republice nyní patří asi k nejrozšířenější službě svého druhu. A to především díky lokalizaci do češtiny. V USA je nyní také sociální sítí číslo jedna (podle nezávislého měření společnosti *Alexa.com* je třetí nejnavštěvovanější stránkou na světě, hned po *google.com* a *yahoo.com*), a to i přes to, že tu má stále silné konkurenty *MySpace.com* a svým způsobem také *Twitter.com* (ten však považuje sám sebe spíše za komunikační síť než za síť sociální).

Facebook je typický svým všeobecně příjemným ajaxovým uživatelským rozhraním.

5.3.6 Twitter

Twitter[30] je stále rychle rostoucí služba nabízející uživatelům sdílet krátké (140 znaků dlouhé) komunikační zprávy – takzvané *tweety*. Nepoužívá klasické označení *Friends*, ale *following* a *followers* (tedy seznam vámi sledovaných uživatelů a seznam uživatelů, kteří sledují vaše tweety). Vedle veřejných tweetu lze vybraným členům posílat také přímé soukromé zprávy, odpovídat veřejně na jejich tweety, sledovat veřejné dění či přeposílat tweety svých přátel.

Díky otevřenému API vzniká celá řada aplikací na twitteru postavených, například služby pro sdílení obrázků, vyhledávání zajímavostí z vašeho okolí, apod.

Každému přihlášenému uživateli se na stránce twitter.com zobrazují jiné zprávy – právě od lidí ze seznamu following. A každý uživatel má svůj veřejný profil na URL adrese ve tvaru *http://twitter.com/uzivatelskejmeno*

[29] Dostupný na adrese: http://www.facebook.com/
[30] Dostupný na adrese: http://www.twitter.com/

Twitter se stal také populární u mnoha veřejně známých osobností ať už ze světa politiky, sportu, hudby, … Namátkou můžu jmenovat tyto: David Heinemeier Hansson, Steve Jobs, Arnold Schwarzenegger, Demi Moore, Britney Spears, Michael Phelps, Barack Obama a mnoho dalších.

5.3.7 Zajímavá čísla

Tady uvedu na závěr této kapitoly několik zajímavých číselných statistik převážně z prostředí sociálních sítí.[22]

- 3 ze 4 Američanů využívají na Internetu sociální sítě.
- Navštěvování sociálních sítí je 4. nejčastější činností na Internetu (častější než posílání emailů).
- 13 hodin – Délka videí nahraných na YouTube každou minutu.
- 100 000 000 – Počet videí shlédnutých na YouTube denně.
- 13 000 000 – Počet dostupných článků na Wikipedii.
- 3 600 000 000 – Počet fotografií uložených na Flickr.com.
- 1 382 % – Měsíční nárůst uživatelů Twitteru od ledna do února 2009.
- 3 000 000 – Počet nových tweetů denně na Twitter.com.
- 5 000 000 000 – Minut strávených na Facebooku každý den.

6 UML

Unified Modeling Language představuje univerzální jazyk pro vizuální modelování systémů. Byl navržen tak, aby jej mohly implementovat veškeré CASE nástroje. Jazyk UML sám o sobě poskytuje pouze vizuální syntaxi, *nenabízí* tedy žádný druh metodiky.

Metodikou, která nám sděluje, jaké pracovníky využít, jaké činnosti vykonat a jaké produkty vyrobit, aby se nám podařilo sestavit model funkčního softwarového systému je například *Unified Process*.

6.1 Zrození UML

Na UML se začalo pracovat v roce 1995. Do té doby existovalo jen několik jazyků pro vizuální modelování a několik metodik. Neexistovaly však žádné standardy a svět objektově orientovaných metod se zmítal spíš v chaosu.

V roce 1996 navrhlo sdružení Object Management Group specifikaci Request For Proposal, ve které bylo UML navrženo jako standard pro objektově orientovaný jazyk pro vizuální modelování. A v roce 1997 byl UML jako standard přijat.

Tím však vývoj UML nekončil a stále se pokračovalo na jeho zdokonalování. Důležitým rokem v této oblasti se stal rok 2005, v němž byla dokončena specifikace UML 2.0, které umožňuje vyspělé modelování.

6.2 Struktura UML

UML je složeno z celé řady grafických prvků, jejichž kombinováním vznikají různé diagramy.

6.2.1 Základní prvky

Za základní prvky UML jsou považovány stavební bloky označované jako *předměty* (things), *vztahy* (relationships) a *diagramy* (*diagrams*)

Předměty

Předměty – samotné prvky označované také jako „věci“ nebo abstrakce – dělíme v UML následovně:

- Strukturální abstrakce – třídy, rozhraní, spolupráce, komponenta, atd.
- Chování – interakce, stav.
- Seskupení – balíčky sloužící k seskupování podobných prvků.
- Poznámky – anotace připojené k modelu zachycující informaci.

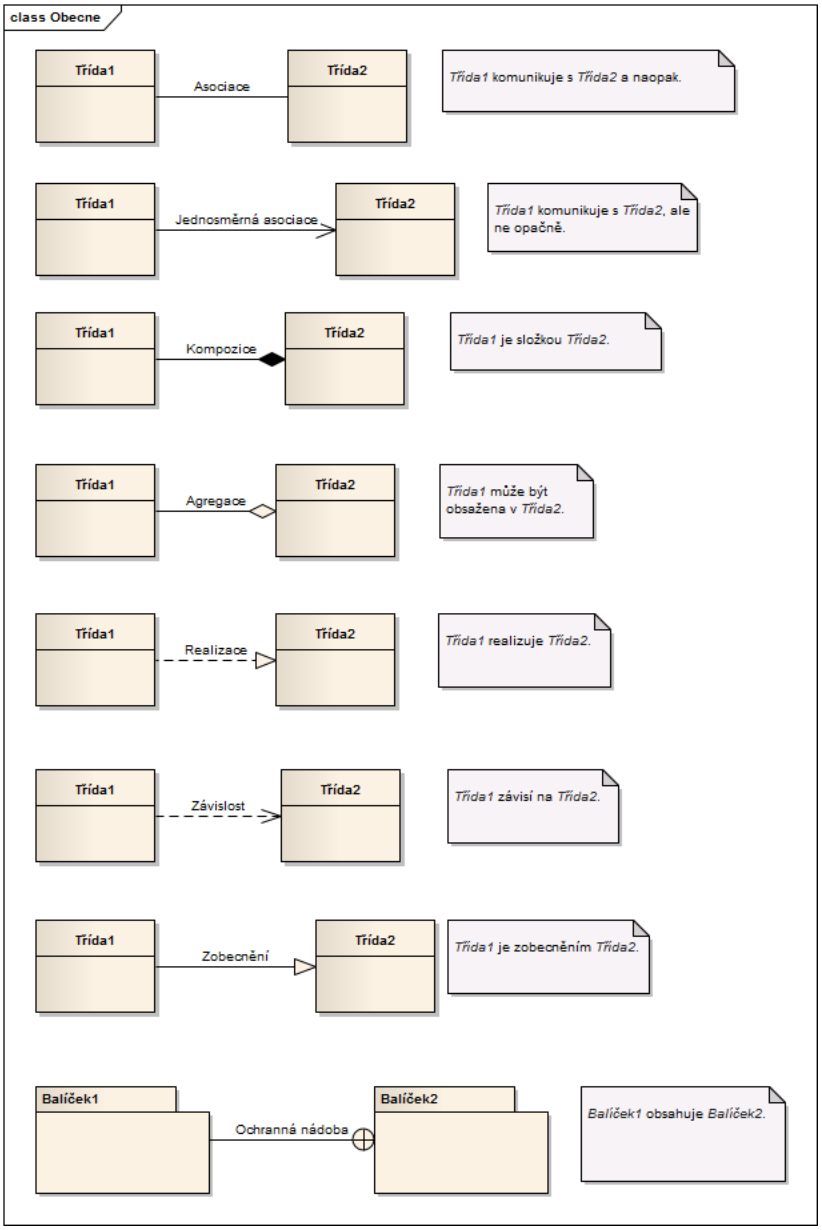
Vztahy

Vyjadřují souvislosti mezi dvěma nebo více předměty. Relací existuje více druhů:

- *Asociace* – volnější vazba mezi prvky (třídami) umožňující jejich vzájemnou komunikaci.
- *Jednosměrná asociace* – vazba odvozena od asociace, omezuje směr komunikace.

- *Kompozice* – silnější vazba mezi částí a celkem. Část nemůže existovat bez celku.
- *Agregace* – volnější vazba mezi částí a celkem. Část může existovat sama o sobě.
- *Realizace* – vztah mezi třídou a rozhraním.
- *Závislost* – Změna jednoho prvku se promítne do prvku závislého.
- *Zobecnění* – Jeden prvek je zobecněním prvku druhého. Znázorňuje se tím tedy dědičnost mezi prvky.
- *Ochranná nádoba* – Zdrojový prvek obsahuje cílový prvek. Využívá se k modelování jmenných prostorů.

Výše popsané vztahy zachycuje následující obrázek.



Obrázek 7: Vztahy v UML[31]

[31] Zdroj obrázku: vlastní

Diagramy
Diagramy v UML představují pohledy na model. Důležité je si uvědomit, že diagram *není* model! Z diagramu můžeme prvek odstranit, ale v modelu stále zůstává. Není žádné pevně dané pořadí, v jakém by se měly diagramy v UML vytvářet. Většinou se však začíná diagramem případů užití. Ve skutečnosti se často pracuje s více diagramy zároveň.

Diagramy struktury:

- *Diagram balíčků* – rozděluje model do logických balíčků, popisuje jejich vzájemnou komunikaci na nejvyšší úrovni.
- *Diagram tříd* – znázorňuje základní stavební bloky modelu (typy, třídy).
- *Objektový diagram* – ukazuje, jaká jsou v daném okamžiku používána spojení rozhraní konstrukčních prvků.
- *Diagram složené struktury* – znázorňuje rozvrstvení prvků a zahrnuje skryté detaily.
- *Diagram komponent* – používán pro modelování složitější struktury (třídy + rozhraní)
- *Diagram nasazení* – znázorňuje fyzické uspořádání důležitých artefaktů, a to včetně jejich nastavení.

Diagramy chování.

- *Diagram případu užití* – znázorňuje vzájemné působení mezi uživatelem a systémem.
- *Diagram aktivit* – má široké pole působnosti od znázornění běhu programu až po specifikaci libovolného procesu.
- *Diagram stavového automatu* – slouží k porozumění okamžiku – určitému stavu projektu.
- *Diagram iterace* – používán především ke znázorňování komunikace.

6.2.2 Obecná mechanika UML

Specifikace
Specifikace představují textovou formu popisu významosloví jednotlivých prvků.

Ornamenty
Ornamenty jsou prvky obohacující základní symboly UML, je-li potřeba znázornit více informací.

Podskupiny
Podskupiny popisují různé způsoby vidění světa. Při zkoumání podskupin v UML se setkáme se dvěma druhy podskupin.

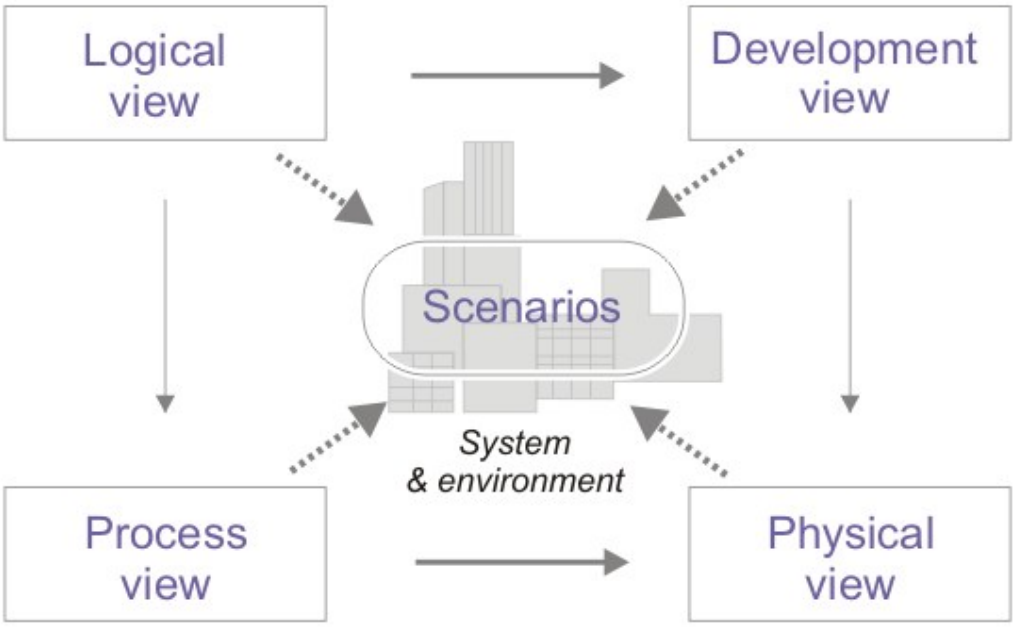
- *Klasifikátor a instance* – klasifikátor je abstraktní vyjádření předmětu; instance představuje konkrétní realizaci předmětu.
- *Rozhraní a implementace* – rozhraní říká, *co* má předmět provádět a implementace říká, *jak* to má provádět.

Mechanismy rozšiřitelnosti
UML je rozšiřitelný modelovací jazyk. Mechanismy rozšiřitelnosti jsou následující:

- *Omezení* – omezující podmínky umožňují přidávat k prvku nová pravidla.
- *Stereotyp* – definuje nový prvek modelu založený na stávajícím prvku.
- *Označené hodnoty* – umožňují rozšířit specifikace prvku tím, že k němu přidávají informaci sestavenou jen k tomuto účelu.

6.2.3 Architektura
UML zachycuje strategické aspekty systému v architektuře „4 + 1“:

- *Logický pohled* – zachycuje slovník jako třídy a objekty.
- *Pohled procesů* – modeluje spustitelná vlákna jako třídy.
- *Pohled implementace* – modeluje soubory a komponenty.
- *Pohled nasazení* – modeluje fyzické nasazení komponent na množinu fyzických výpočetních uzlů.
- *Pohled případu užití* – od tohoto pohledu jsou odvozeny všechny předcházející, zachycuje požadavky uživatele.



Obrázek 8: Model architektury 4+1[32]

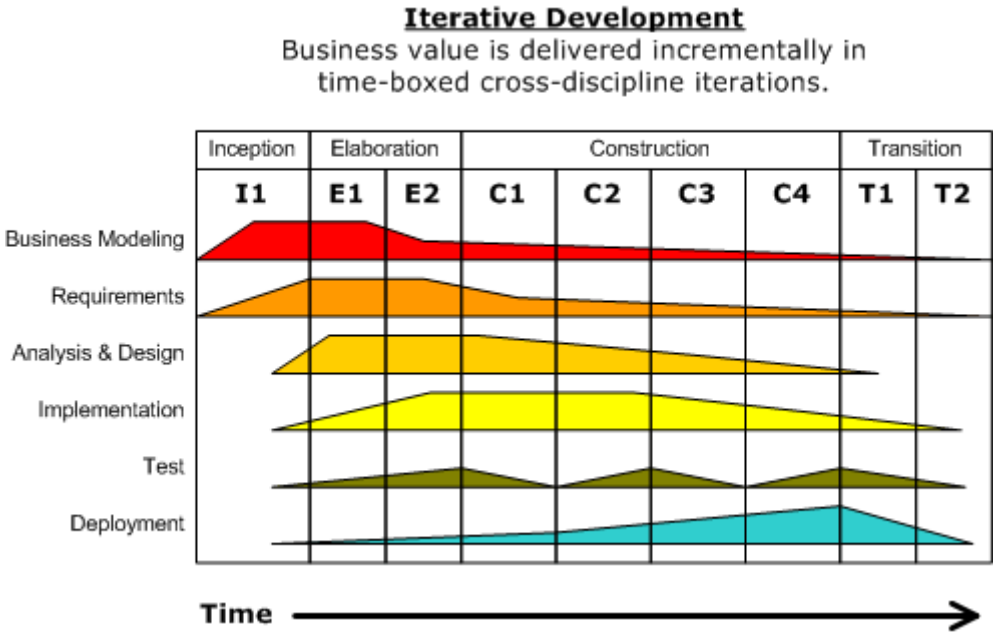
[32] Zdroj obrázku: *4+1 Architectural View Model - Wikipedia, the free encyclopedia* [online] Dostupné z WWW: http://upload.wikimedia.org/wikipedia/commons/f/f2/4%2B1_Architectural_View_Model.jpg

7 Unified Process

Unifikovaná metodika vývoje softwaru (bývá zkráceně označována jako „unifikovaný proces“ – UP) je oblíbenou metodikou pro vývoj softwarových systémů. UP je založen na iteracích, to znamená, že jeden velký projekt je rozdělen do podprojektů, které se řeší v dávkách (po přírůstcích). Nejznámějším unifikovaným procesem je RUP – Rational Unified Process, ten obsahuje i propracovanou úplnou dokumentaci.

Jak UP, tak i RUP jsou rozšiřitelné a přizpůsobitelné systémy, takže mnohdy je téměř nemožné rozeznat, zda zpracovávání procesu proběhlo pomocí UP nebo RUP.

Název UP se tedy často používá spíše v obecné rovině a zahrnuje elementy, které jsou společné mnoha zpracováním. RUP je potom metodika představující konkrétní pojetí UP a její název je registrovanou ochrannou známkou firmy IBM.



Obrázek 9: Iterační vývoj softwarového projektu[33]

Iterativní proces vývoje nám umožňuje vracet se v průběhu tvorby i několikrát ke klíčovým částem. To ovšem není jediná výhoda, kterou nám nabízí. Další nespornou výhodou je i fakt, že tento proces umožňuje běh dvou a více iterací zároveň, což vede k urychlení vývoje.

Životní cyklus každého projektu je složen z několika fází. Každá tato fáze má své milníky, které musí být splněny, aby mohla být fáze ukončena.

[33] Zdroj obrázku: *Unified Process - Wikipedia, the free encyclopedia*, [online] dostupný z WWW: http://upload.wikimedia.org/wikipedia/en/0/05/Development-iterative.gif

- *Fáze zahájení* je zahájením celého projektu.
  Milník: schválení projektu, odsouhlasení rozpočtu, potvrzení proveditelnosti, hrubý náčrt architektury.
- *Fáze rozpracování* je zaměřena na tvorbu částečné, avšak funkční verze systému – spustitelného architektonického systému.
  Milník: vytvoření plánu celého projektu, vytvoření spustitelného prototypu.
- Během *konstrukční fáze* se spustitelný architektonický základ převede na kompletní a funkční systém.
  Milník: produkt je ve fázi, kdy je připraven k nasazení u uživatele, zjištění přijatelnosti rozdílu mezi skutečnými a plánovanými výdaji.
- *Fáze zavedení* spočívá v nasazení hotového systému na pracoviště uživatele.
  Milník: dokončení testování, předání projektu uživateli (zákazníkovi)

Jiný pohled na vývoj software

Vedle výše popsaného modelu vývoje, existují ještě modely *„vodopádový“*, *„spirálový“* a *agilní*. Osobně si myslím, že pro tvorbu webové aplikace se nejvíce hodí právě agilní vývoj. Agilní vývoj taktéž rozděluje projekt do fází, a to vždy do stejných. Příklad budiž shrnutí vývoje eshopu:

- *Fáze 1*
  Týká se: katalog produktů
  Skládá se z: návrh; implementace; testovaní/nasazení
- *Fáze 2*
  Týká se: nákupní košík, objednávka
  Skládá se z: návrh; implementace; testování/nasazení
- *Fáze 3*
  Týká se: historie objednávek, platební brána
  Skládá se z: návrh; implementace; testování/nasazení

Agilní vývoj využívá principy metody SCRUM

1. Krátké vývojové cykly (*sprints*).
2. Denní schůzky „vestoje“ (*stand-up meetings*).
3. Revize cyklu (*sprint review*).
4. Stručný seznam funkcí k implementaci.

SCRUM počítá s vývojovým týmem (obvykle 5–9 lidí). Tvrdí totiž, že úspěch projektu je týmová záležitost – role jednotlivce je velmi omezená.

8 Nástroje pro modelování

Jedná se o nástroje CASE (Computer-Aided Software Engineering). Tyto nástroje jsou určené ke komplexní podpoře vývoje software. Tedy zahrnují v sobě podporu pro celý proces vývoje od analýzy až po testování.

Tyto nástroje využívají ve většině případů UML k vizuálnímu znázornění vývoje projektů.

Často zmiňované výhody CASE nástrojů:

- Usnadňují údržbu vyvíjeného systému.
- Možnost spolupráce více lidí a nástrojů během vývoje.
- Generování kostry zdrojových kódů a SQL v závislosti na modelu.
- Generování a vznik dokumentace.
- Podpora reverzního inženýrství.

Pro vytvoření modelu WCMS (a jeho znázornění prostřednictvím diagramů) jsem se rozhodl pro použití CASE nástroje *Enterprise Architect*, který je vyvíjen firmou Sparx. Jako nevýhodu tohoto systému shledávám především to, že není multiplatformní a je primárně určen jen pro systémy Windows. Naštěstí díky projektu *Wine* je možné jej spustit i na systémech GNU/Linux, na kterých velice často probíhá fáze implementace webových aplikací.

9 Tvorba návrhu nového WCMS

Před započetím samotné tvorby návrhu přichází na řadu velice důležitá část celého projektu, a to je sběr požadavků.

9.1 Sběr požadavků

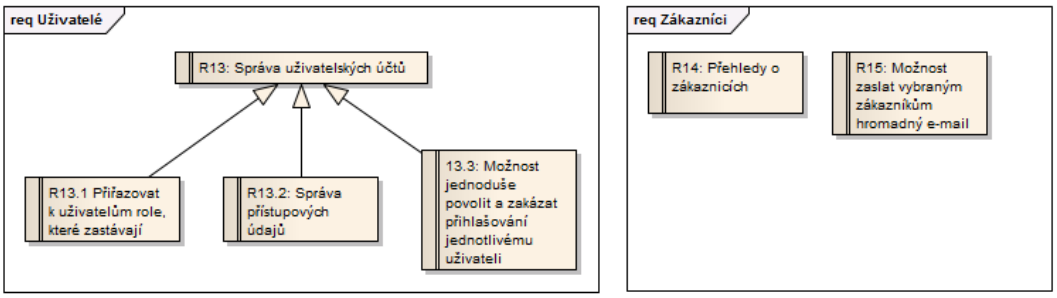
Jak už bylo řečeno, jedná se o zcela stěžejní část celého projektu. Této fáze se účastní obvykle jak zadavatel (budoucí uživatel systému), tak zhotovitel (analytik). Sběr požadavků může probíhat vícero způsoby:

- Konzultace obou stran,
- vyplňování dotazníků,
- dílnou požadavků (diskusí) – všechny nápady v diskusi jsou považovány za dobré a zapíší se.

Požadavky se obecně dělí na funkční a nefunkční. Funkční jsou ty, které určují chování samotného systému, nefunkční požadavky jsou potom ty, co určují omezení vytvářeného systému.

9.1.1 Funkční požadavky

Kompletní posbírané funkční požadavky na WCMS jsou k dispozici v UML modelu na přiloženém CD. Funkční požadavky byly rozděleny do několika skupin, podle oblastí WCMS.



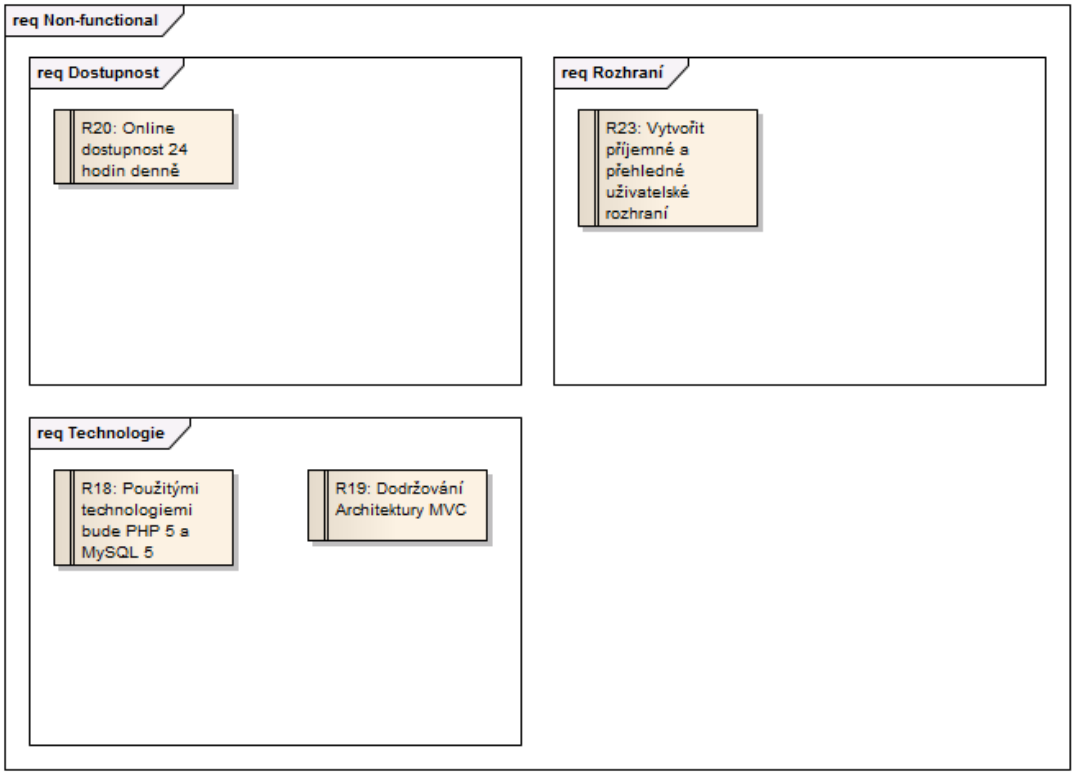
Obrázek 10: Ukázka funkčních požadavků[34]

9.1.2 Nefunkční požadavky

Nefunkční požadavky jsou požadavky specifikované na použité technologie, na dostupnost a výkonnost systému.

Obrázek 11 znázorňuje nefunkční požadavky na konkrétní WCMS.

[34] Zdroj obrázku: vlastní

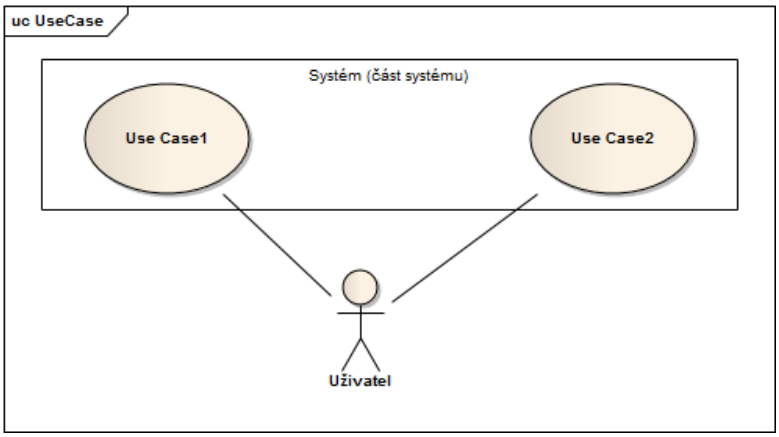


Obrázek 11: Ukázka nefunkčních požadavků WCMS[35]

9.2 Use Case diagramy

Diagramy případů užití popisují, jak uživatelé pracují se systémem. V roli koncového uživatele může vystupovat jak konkrétní osoba, tak stroj, čas nebo jiný systém.

Na následujícím obrázku je znázorněný obecný Use Case diagram, kde postava představuje uživatele systému (aktéra), obdélník ohraničený černou linií znázorňuje systém (nebo jeho část) a elipsy už znázorňují jednotlivé případy užití. Mezi jednotlivými případy užití a aktérem vzniká komunikační relace.



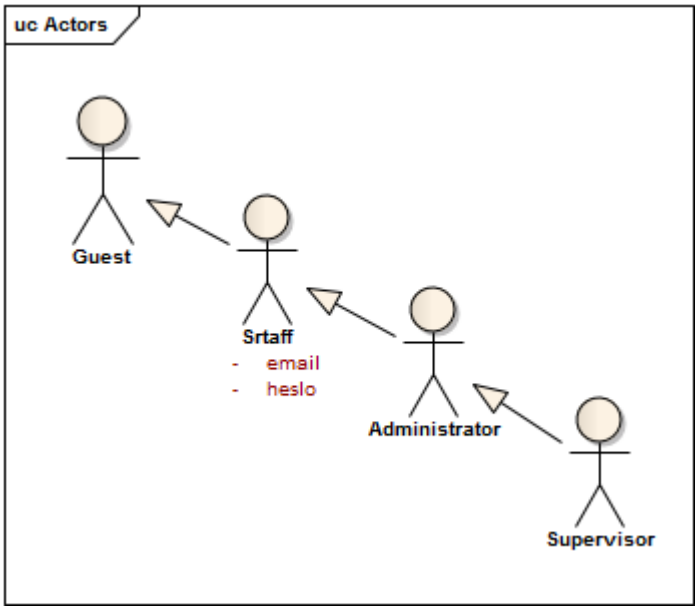
Obrázek 12: Obecný případ užití[36]

[35] Zdroj obrázku: vlastní

Při tvorbě návrhu WCMS jsem se zaměřil na případy užití jednotlivých uživatelských rolí, které představují aktéry v systému.

- *Guest* – host systému – nepřihlášený uživatel; unikátní oprávnění: možnost se přihlásit, nechat si zaslat zapomenuté heslo.
- *Staff* – zaměstnanci – dědí oprávnění od Guest; vlastní práva: přidávat a editovat obsah.
- *Administrator* – správce systému – dědí oprávnění od Staff (potažmo Guest); vlastní práva: publikovat obsah, mazat obsah, správa uživatelů.
- *Supervisor* – skrytý hlavní správce systému – dědí oprávnění od Administrator; vlastní práva: konfigurace systému (například zakládání nových jazykových mutací).

Na následujícím obrázku je zachycen vztah mezi jednotlivými aktéry systému. Je tedy patrné dědění oprávnění, které je popsáno výše.



Obrázek 13: Znázorněná struktura uživatelů systému[37]

V rámci tvorby případů užití by se měl nejprve pořídit pohled na využívání systému jako na celek. Ten se poté může (v závislosti na rozsahu modelovaného systému) rozdělit do více specifičtějších modelů.

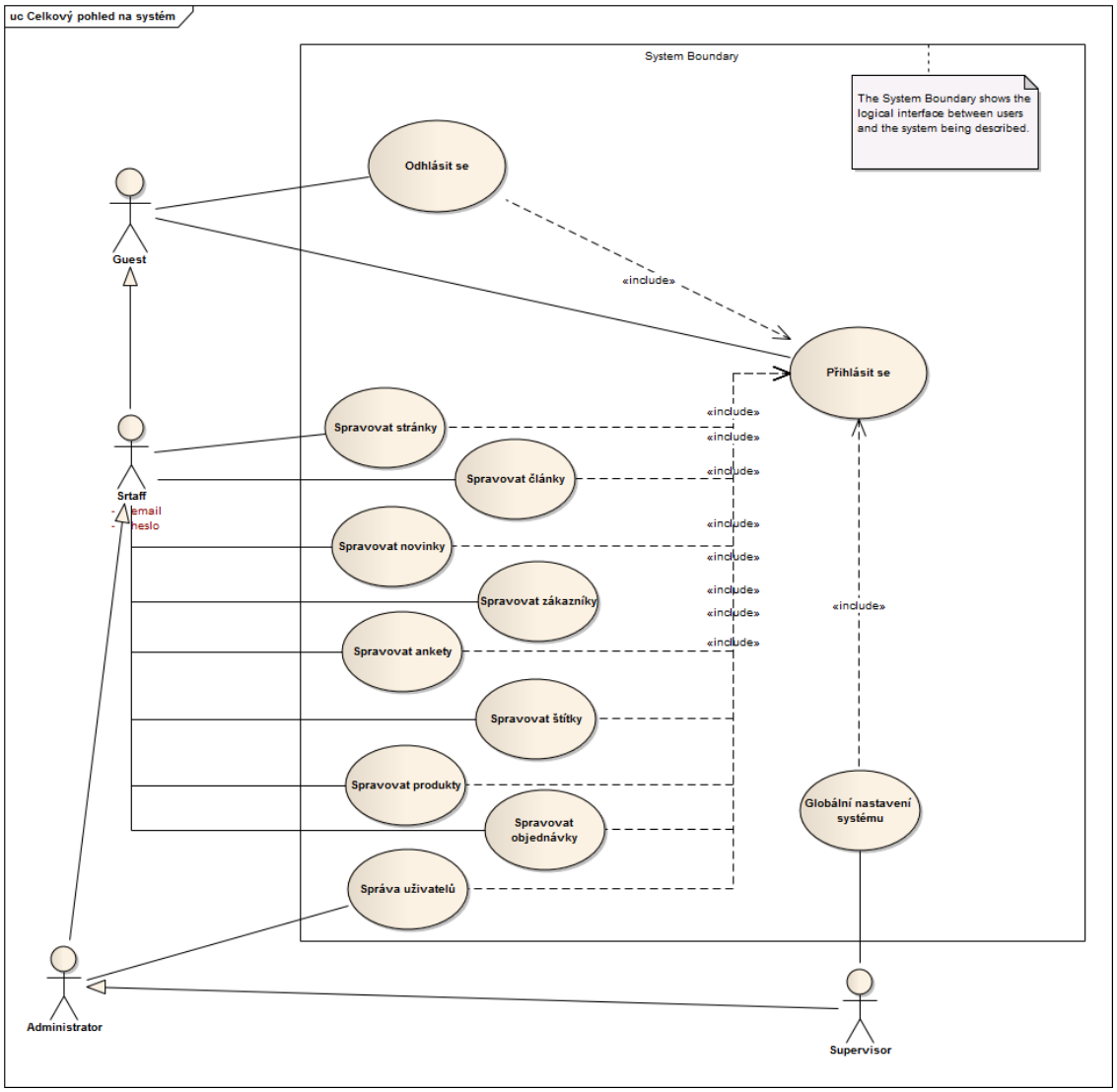
Například případ užití *Spravovat novinky* může být dále rozložen na podrobnější diagram případu užití, kde budou zachyceny například skutečnosti jako: *Založit novou novinku*, *přidat úvodní obrázek k novince*, *přidat anotaci k novince*, *editovat anotaci novinky*, *přidat obsahovou část novinky*, *přidat fotogalerii k obsahové části novinky*, *přidat soubory ke stažení k obsahové části*

[36] Zdroj obrázku: vlastní
[37] Zdroj obrázku: vlastní

novinky, *editovat obsahovou část novinky*, *změnit pořadí obsahových částí novinky*, *publikovat novinku*, a tak podobně.

Případy užití v sobě také velice často zahrnují i takzvané scénáře naznačující pracovní tok událostí vedoucích ke konečnému stavu či výsledku. Scénáře se dále dělí na hlavní a alternativní. Přičemž hlavní scénář případu užití je vždy jen jeden a vedlejších scénářů může být libovolné množství. Je ovšem doporučováno redukovat počet alternativních scénářů na nezbytné minimum.

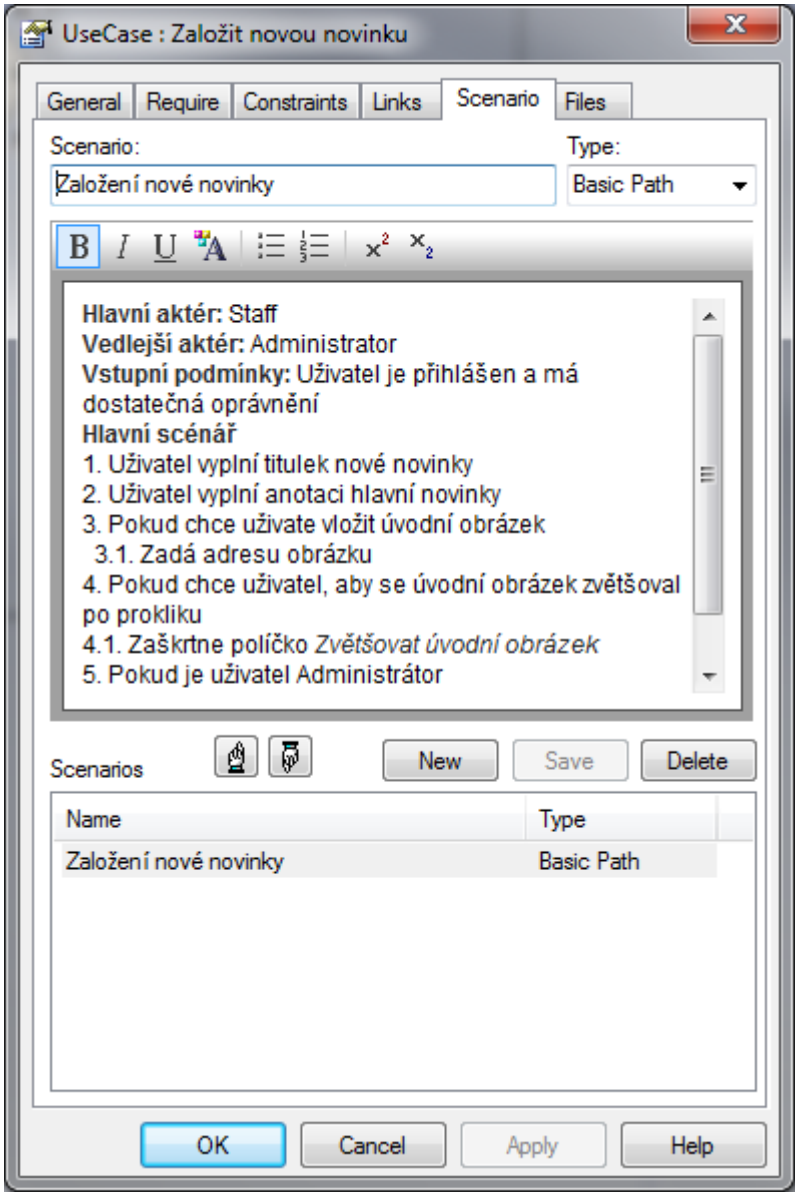
Na následujícím *obrázku 14* je znázorněn diagram případu užití, který zobrazuje souhrnný náhled na využívání navrhovaného Web Content Management Systému z pohledu jednotlivých aktérů.



Obrázek 14: Celkový pohled na případy užití WCMS[38]

[38] Zdroj obrázku: vlastní

Na následujícím *obrázku 15* je znázorněno přidání hlavního scénáře k případu užití *Založit novou novinku*.



Obrázek 15: Přidání scénáře k případu užití[39]

9.3 Analytický model

Analytický model vzniká během fáze analýzy. Hlavním aktérem tu bývá systémový analytik, který se účastnil i sběru požadavků při konzultacích se zákazníkem. Je tedy dobře obeznámen s aktuálním tématem a může navrhnout odpovídající analytické řešení. Během této fáze mohou vznikat pro lepší dokreslení problému i diagramy aktivit.

Analytický model vychází z požadavků a opírá se o diagramy aktivit. Bývá na vysoké abstraktní úrovni a nezaobírá se přílišnými detaily. Při tvorbě analytického modelu se zodpovídají otázky: *„Co by měl systém dělat? Co by měl systém umět?“*.

[39] Zdroj obrázku: vlastní

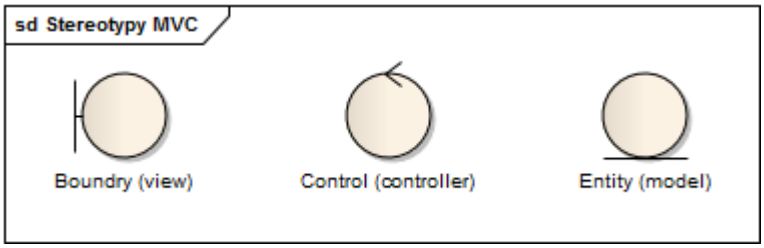
Během tvorby analytického modelu se s úspěchem využívá stereotypů ke ztvárnění chování aplikace založené na architektuře MVC.

9.3.1 Analýza MVC

Podrobnějším principům fungování a smyslům architektury MVC se věnuji později v kapitole 10.3.1 Modely, pohledy a řadiče.

Pro znázornění principů architektury MVC pomocí UML diagramů lze využít takzvaných stereotypů. Právě pomocí nich lze znázornit uživatelské rozhraní, kontroléry a datové složky.

Tyto stereotypy lze v prostředí CASE nástroje Enterprise Architect vyjádřit následovně, jak ukazuje obrázek.



Obrázek 16: Stereotypy MVC[40]

S objektem *Boundry* komunikuje aktér systému. Představuje tedy veškerá uživatelská rozhraní jako dialogy, menu, formuláře, obrazovky, apod. Představuje tedy vrstvu *pohled (view)* z architektury MVC.

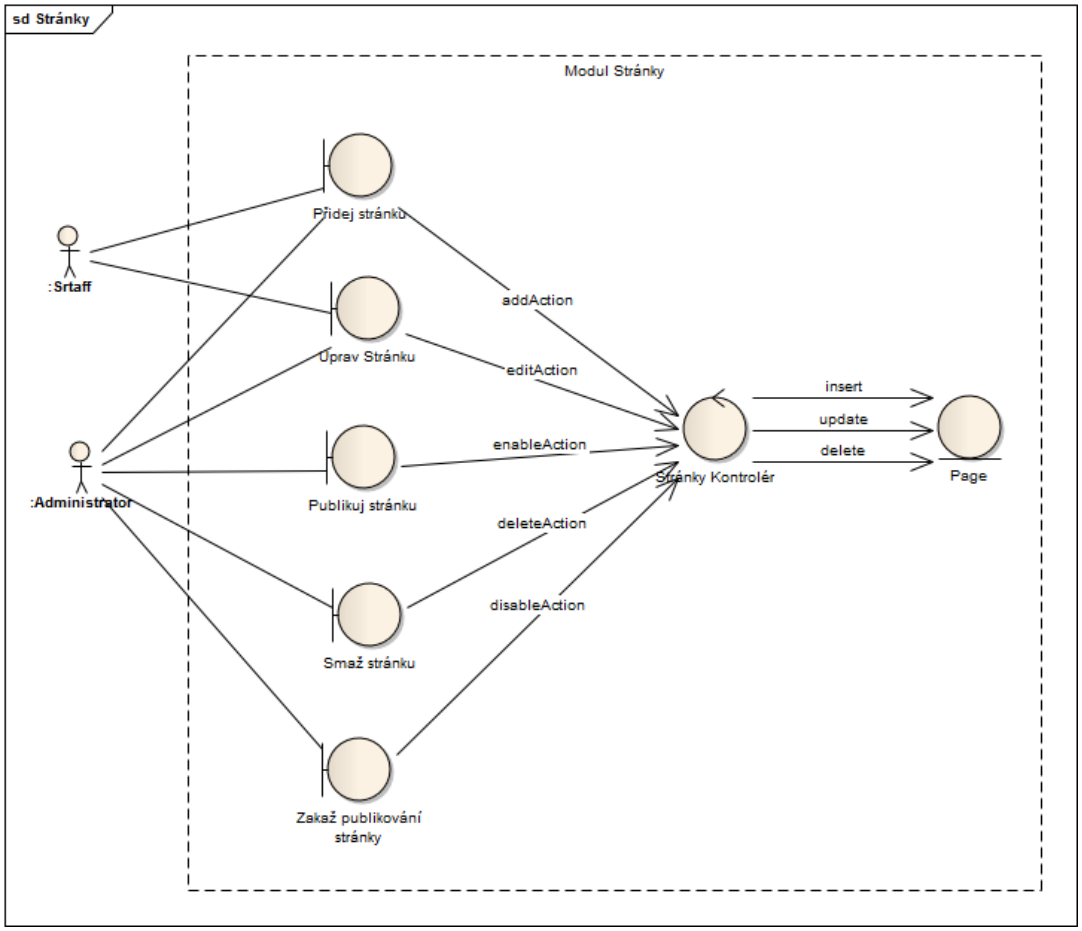
Control je objekt, který disponuje řídící funkcí. V podstatě jde o souhrn pravidel, která filtrují data tak, jak si je přeje uživatel zobrazit. V rámci architektury MVC odpovídá vrstvě *řadič (controller)*.

Objekt *Entity* je náhrada *modelu* z architektury MVC. Jsou to informace, které jsou hledány prostřednictvím boundry. Jde tedy o data, která mohou být uložena v databázi, souboru apod.

To, jakým způsobem probíhá komunikace mezi objekty v rámci analytického modelu, je určeno několika pravidly:

1. Aktéři smí komunikovat se systémem pouze prostřednictvím uživatelského rozhraní (objekt Boundry).
2. Objekty typu Boundry komunikují jednak s objekty typu Control a jednak s aktérem.
3. Objekty Control komunikují se všemi mimo aktérů. Tedy komunikují jednak s objekty Boundry, s objekty Entity a jednak s jinými objekty typu Control.
4. Entity poskytují data pouze objektům Control.

[40] Zdroj obrázku: vlastní



Obrázek 17: MVC analýza modulu stránky[41]

9.3.2 Analytické třídy

Důležitou částí analýzy systému je definování analytických tříd. K jejich identifikaci lze dospět několika způsoby, přičemž nejběžnější jsou dva, a sice:

Metoda CRC štítků

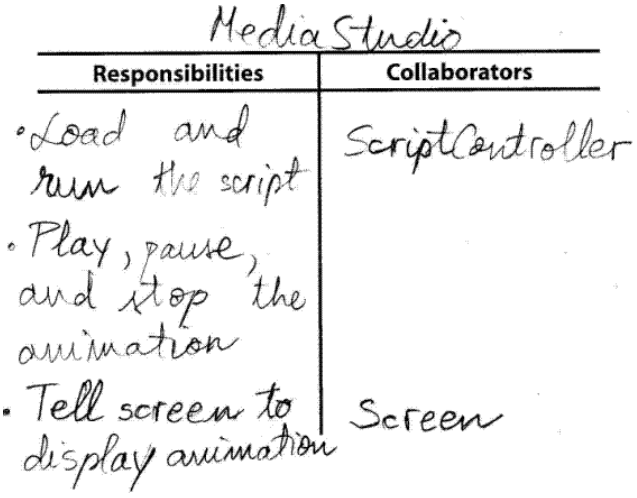
Tato metoda bývá v anglické literatuře nazývána Class Responsibility Collaboration card, odtud zkratka CRC.

Tato metoda spočívá v pojmenovávání předmětů z problémové domény, následně se těmto předmětům přiřadí odpovědnosti (*Responsibilities*) a nakonec se ještě označí třídy, které by mohly spolupracovat (*Collaboration).*

V druhé fázi se pak analytici rozhodují, které lístečky budou zastupovat třídy a které budou zastupovat jejich atributy.

Ukázku podoby CRC štítku ukazuje obrázek 18.

[41] Zdroj obrázku: vlastní



Obrázek 18: Ukázka CRC štítku[42]

Metoda podstatných jmen a sloves

U této metody se vychází s textové podoby specifikace navrhovaného systému. Z tohoto textu se vyjmou podstatná jména jako adepti na třídy a atributy a slovesa jako adepti na metody.

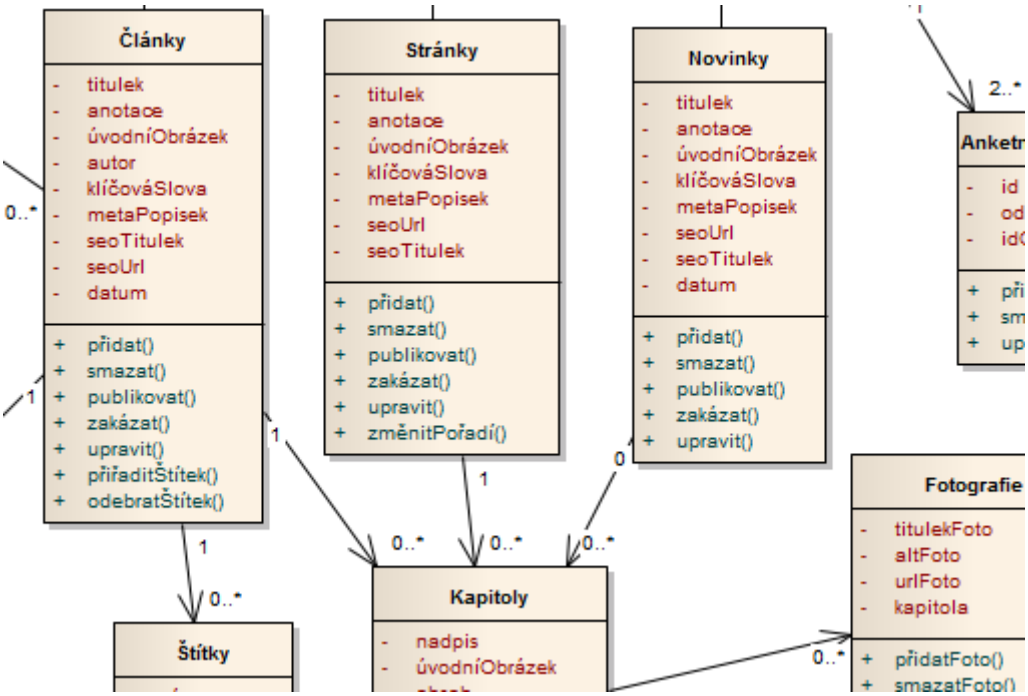
V analytickém modelu se mezi třídami řeší jen to, které spolu komunikují, ale většinou se už neřeší jak.

Základní podobu analytických tříd si lze představit takto:

- Název – povinný (měl by být výstižný).
- Atributy povinné – ale jen nejdůležitější množina atributů.
- Operace jen obecně – bez parametrů a návratových typů.
- Není určen typ viditelnosti.
- Počet vazeb na ostatní třídy je minimální.

Výřez z diagramu analytických tříd je vidět na obrázku 19. Kompletní diagram je potom v elektronické podobě na přiloženém CD.

[42] Zdroj obrázku: *How to make good CRC cards and Scenarios*. [online] Dostupný z WWW: http://coweb.cc.gatech.edu/cs2340/5583



Obrázek 19: Výřez z diagramu analytických tříd[43]

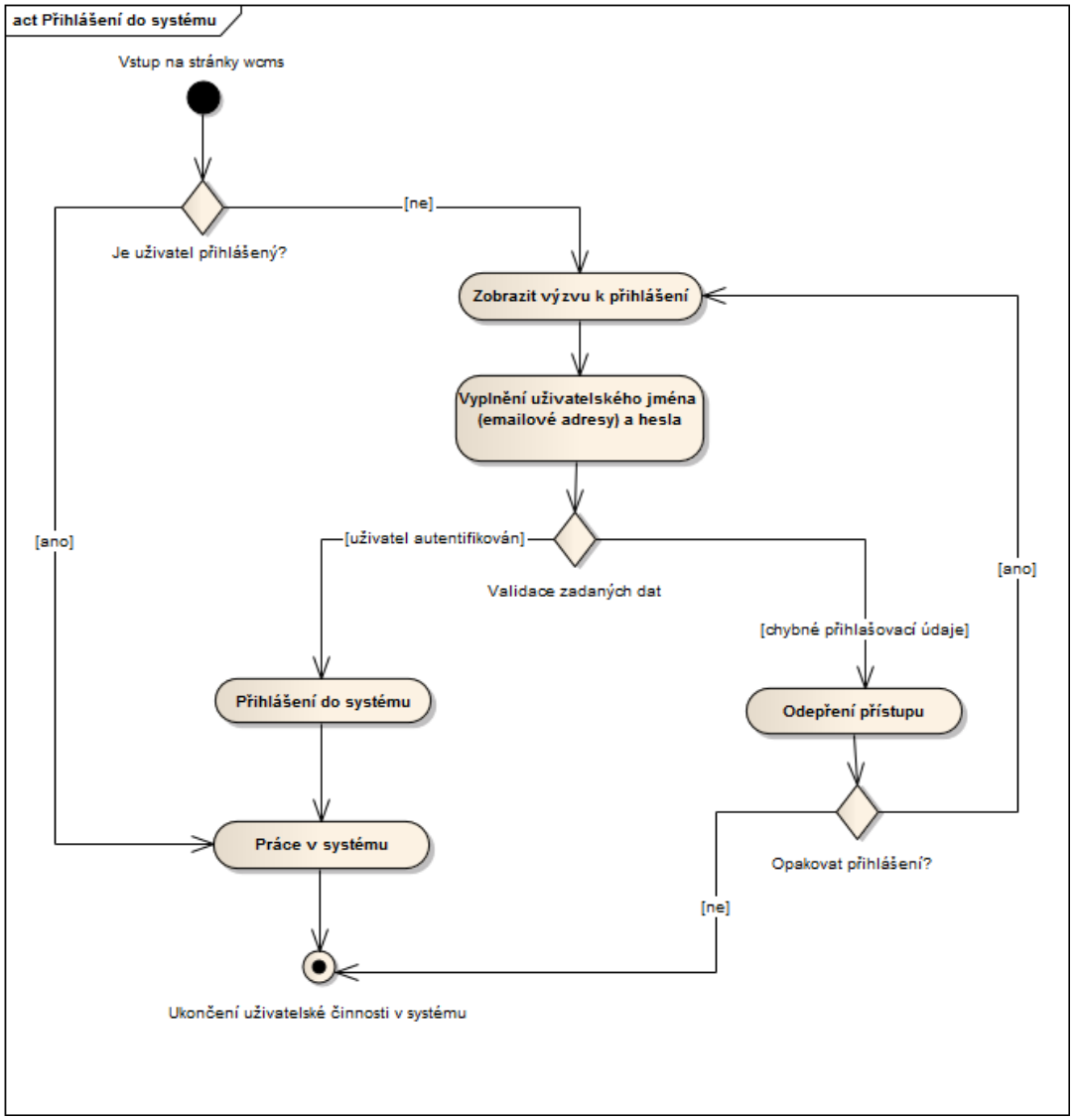
9.3.3 Diagramy aktivit

V systému obvykle probíhá celá řada aktivit. Tyto aktivity probíhají buď sériově, nebo paralelně. A právě jejich průběh znázorňují diagramy aktivit. Diagram vždy začíná počátečním bodem a je ukončen bodem koncovým. Koncových bodů může mít diagram i více, záleží na průběhu aktivit.

Diagramy aktivit se také často používají ke znázornění obchodních (business) procesů a workflow.

Obrázek 20 zachycuje diagram aktivit znázorňující přihlášení a obecně práci v systému. Více diagramů aktivit navrhovaného WCMS je v elektronické podobě na přiloženém CD.

[43] Zdroj obrázku: vlastní



Obrázek 20: Diagram aktivit[44]

9.4 Návrhový model

Z analytického modelu vychází model návrhový. Ten by měl být již na takové úrovni, aby podle něho šel systém implementovat. Tento návrhový model často tvoří návrhář, avšak do jisté míry ho může vytvářet i analytik, který má o systému dokonalý přehled a nemusí tak podávat vysvětlení návrháři.

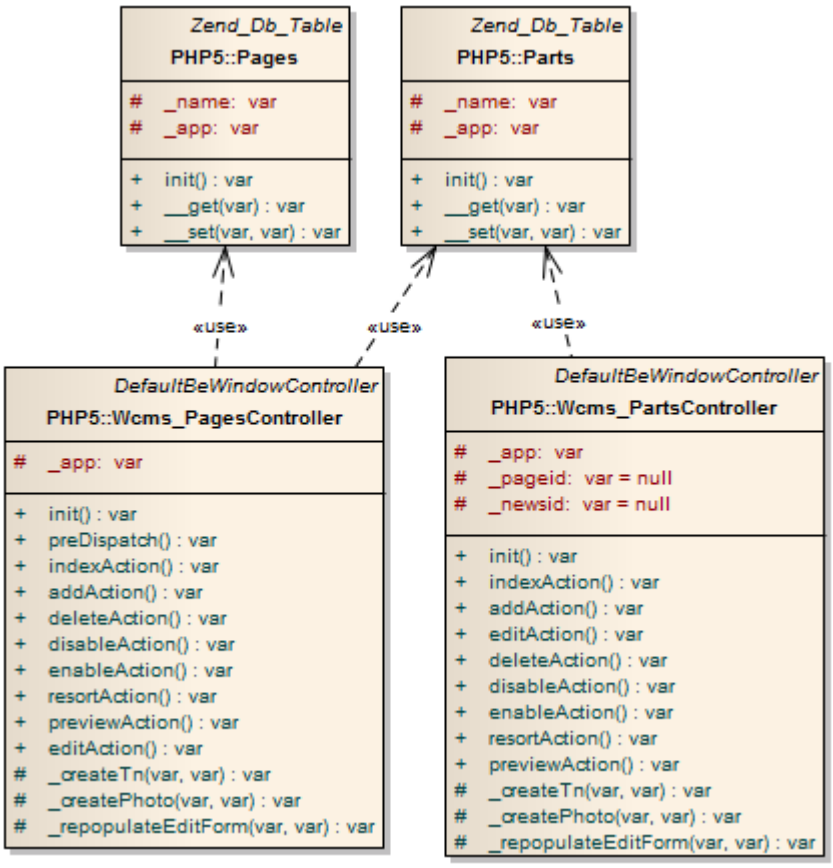
Moderní nástroje CASE v sobě zahrnují podporu mnoha programovacích jazyků a jsou schopny na základě dobře postaveného návrhového modelu vygenerovat funkční kostru aplikačního kódu.

Mnou používaný CASE nástroj Enterprise Architect podporuje generování do všech běžně používaných programovacích jazyků, jako jsou Java, C, C# nebo pro webové aplikace jazyk PHP.

[44] Zdroj obrázku: vlastní

V rámci tvorby návrhového modelu často vzniká několik verzí modelu. Přičemž jeden model je více abstraktní a neobsahuje žádná specifika konkrétního programovacího jazyka.

Další návrhové modely (vycházející z tohoto abstraktního) bývají označovány jako modely *implementační*, které v sobě zahrnují rysy programovacího jazyka, ve kterém bude systém implementován.



Obrázek 21: Ukázka implementačního diagramu pro jazyk PHP[45]

9.4.1 Návrhové třídy

Na rozdíl od tříd analytických jsou třídy návrhové mnohem detailnější a vychází ze dvou oblastí:

- První oblastí je odvození ze tříd analytických, které jsou upřesňovány.
- Druhou oblastí jsou knihovny již existujících tříd a komponent.

Návrhový model se soustředí na zodpovězení otázky: *„Jakým způsobem má systém činnost vykonat?“* Metody z tříd analytických jsou doplněny o parametry a návratové typy, atributy analytické třídy jsou pak doplněny o konkrétní typy (případně i výchozí hodnoty) a přístupnost.

[45] Zdroj obrázku: vlastní

Správná návrhová třída by měla být pokud možno úplná, jednoduchá (poskytuje dále nedělitelné služby), soudržná (měla by obsahovat jen ty metody, které nezbytně potřebuje).

V rámci tvorby návrhových tříd se také upřesňují relace mezi třídami z analytického modelu. Upřesňují se zejména násobnosti, názvy rolí, části a celky (agregace, kompozice), jednostranné průchodnosti.

Pokud mezi sebou mají třídy asociaci typu M:N, je nutné vytvořit asociační třídu, která bude mít s první třídou vztah M:1 a s druhou vztah 1:N.

9.5 Datový model

Datový model znázorňuje strukturu úložiště dat. Nejčastěji se jedná o relační databázi. Na datový model existují dva pohledy.

1. *Logický model* na datový model nebere v úvahu konkrétní relační databázi.
2. *Fyzický model* zahrnuje v sobě konkrétní relační databázi.

Během analýzy datového modelu jsou důležité následující kroky: identifikace entit, identifikace klíčů, určení vazeb a kardinality, samotné modelování

Datový model by měl vycházet z modelu návrhového, a to mapováním tříd na tabulky a mapováním asociací a agregací, mapováním dědičnosti.

Obrázek 22: Výřez z datového modelu WCMS, splňující 3. normální formu[46]

[46] Zdroj obrázku: vlastní

10 Implementace

Tato kapitola je věnována principům tvorby webových aplikací, přičemž největší důraz bude kladen na skriptovací programovací jazyk PHP, architekturu MVC, Zend Framework a databázi MySQL. Tato specifika byla zadána jako nefunkční požadavky na systém během analýzy.

Implementace bývá plně v rukou programátora, který aplikuje své dovednosti na návrhový model. V případě nejasností či pochybností o správnosti návrhového modelu konzultuje programátor toto právě s autorem návrhu.

10.1 Webové aplikace

Webová aplikace je aplikace typu klient-server poskytovaná uživateli prostřednictvím webového serveru v rámci sítě Internet, případně v rámci vnitropodnikového intranetu. Oblíbenost webových aplikací je rozšířena především kvůli minimálním nárokům na klienta. Těmto nárokům ve většině případů vyhovuje běžný moderní webový prohlížeč[47] rozšířený o prostředky pro podporu[48] aplikací třetích stran. Webový prohlížeč tak vystupuje jako takzvaný *tenký klient* – sám o sobě nezná logiku aplikace.

Podobu webových aplikací tak nabírá mnoho podnikových i jiných informačních systémů, ale i e-mailových rozhraní, internetových obchodů, weblogů, diskusních fór či v dnešní době už běžně také online aukčních síní, radiových a televizních vysílání.

Nedocenitelnou výhodou je, že webovou aplikace lze aktualizovat pouze na jednom místě[49] (server) a ne u milionů uživatelů (klientů).

10.1.1 Statické stránky

Pro takovýto druh stránek je charakteristický statický, čili neměnný, obsah. Dříve takto vypadala většina webových prezentací. Na serveru bylo uloženo mnoho dokumentů označkovaných HTML značkami (dnes běžně nazývané poangličtěle „tagy“). Stránky byly vzájemně provázány pevně danými hypertextovými odkazy.

A i přesto, že vznikly nástroje, které za člověka doplňovaly HTML značky do textu automaticky, manipulace se statickými stránkami byla s rostoucím množstvím obsahu stále náročnější.

Proto začali výrobci webových prohlížečů podporovat takzvané rámce, které autorům stránek značně zjednodušily práci s větším množstvím provázaných souborů. Ovšem ani

[47] V dnešní době to jsou hlavně tyto: Mozilla Firefox 3+, Google Chrome 3+, Apple Safari 4+, Konqueror, Opera 9+, SeaMonkey 1.1+, Microsoft Internet Explorer 8+.
[48] Jedná se zejména o podporu Javy a Adobe Flash.
[49] Velké aplikace pochopitelně nemusí běžet jen na jednom jediném serveru, ale i na více serverech zároveň.

rámce nebyly zcela spásné a navíc se příliš „nekamarádily“ s internetovými vyhledávači[50]. Byla tudíž nutnost přejít na tvorbu dynamických webových stránek.

Ukázka zdrojového kódu stránky složené z rámců:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>WCMS - Programátorská dokumentace</title>
  <meta http-equiv='Content-Type' content='text/html; charset=utf-8'/>
</head>

<FRAMESET rows='120,*'>
       <FRAME src='packages.html' name='left_top' frameborder="1">
       <FRAMESET cols='25%,*'>
              <FRAME src='li_WCMS.html' name='left_bottom' frameborder="1">
              <FRAME src='blank.html' name='right' frameborder="1">
       </FRAMESET>
       <NOFRAMES> Alternativní obsah pro prohlížeče bez podpory rámců </NOFRAMES>
</FRAMESET>
</HTML>
```

10.1.2 Dynamické stránky

Dynamické stránky vznikají generováním na serveru. Data pro tyto stránky bývají nejčastěji uložena v nějaké databázi – převážně MySQL, SQLite, PostgreSQL, MSSQL, Oracle. Ovšem zdrojová data pro obsah dynamických stránek mohou být uložena také v souborech (XML, HTML, TXT).

O generování obsahu na straně serveru se v závislosti na zaslaném požadavku starají skripty a programy zpracovávané interpretem daného programovacího jazyka – nejčastěji PHP, Ruby, ASP, JSP, Python a další.

Technologie pro tvorbu dynamických stránek

Dynamické webové stránky bývají nejčastěji naprogramovány v nějakém z následujících programovacích jazyků. Pro urychlení vývoje se často využívá nejrůznějších frameworků.

Proč používat webový framework?

> *„Toto je sněhová vločka. Vaše aplikace není jedna z nich. Většina věcí, které většina lidí dělá, není nijak unikátní. Vaše potřeby nejsou nijak ‚zvláštní‘.“*[51]

[50] Více o této problematice naleznete v mé bakalářské práci dostupné v univerzitní knihovně Univerzity Pardubice nebo online na adrese: http://tinyurl.com/bp-jan-hridel

[51] David Heinmeier Hansson – přednáška na konferenci Future Of Web Applications, únor 2006

Vývoj webových frameworků

Obrázek 23: Vývoj webových frameworků[52]

Mezi zmiňované programovací jazyky pro webové aplikace patří:

- **PHP**
  Jedná se asi o velice často (ne-li nejčastěji) používaný programovací jazyk pro tvorbu webových dynamických stránek. Jeho oblíbenost vychází zejména z následujícího:
  - Open-source technologie od PHP Group.
  - Velmi volná syntaxe (vychází z jazyků C, Java, Perl, Pascal).
  - Velice solidní nabídky hostingových programů (a to včetně freehostingů)
  - Podpora pro mnoho databázových systémů.

  Jazyk PHP má však i své nevýhody, a to hlavně:

  - PHP skript se znovu na serveru překládá při každém volání.
  - Podpora Unicode je zatím dostupná jen z přídavné knihovny. Ve verzi PHP 6 by měl být řetězec Unicode základním datovým typem.
  - Chybí debugovací nástroj. (ve výchozí distribuci)
  - PHP se stále vyvíjí a to přináší i občasné změny či ukončení platnosti nějakých funkcí. To může způsobit nefunkčnost starších aplikací na serverech s novou verzí PHP.

  Mezi významné projekty implementované v PHP můžeme zařadit tyto:

  - Software *MediaWiki* – např. Wikipedia.
  - *phpBB* – Balík pro provoz webového fóra.
  - *WordPress* – Publikační systém pro provoz weblogů a podobných aplikací.
  - *phpMyAdmin* – Webová aplikace pro správu MySQL databáze.
  - *Texy!* – Překladač intuitivní syntaxe pro formátování textu na HTML.
  - Částečně také *Facebook* – rozsáhlá sociální síť, používá vlastní upravenou verzi PHP plus některé funkce z jazyka C.

[52] Zdroj obrázku: Karel Minařík – přednáška Efektivní vývoj webových aplikací v Ruby On Rails na konferenci Webexpo, říjen 2008

Používané frameworky v PHP:

- Zend Framework,
- CakePHP,
- Nette,
- a další.

Historie PHP se datuje od roku 1994, kdy byla poprvé napsána binární část CGI. Autorem této prvotní části je dánský programátor Rasmus Lerdorf, který vydal první veřejnou verzi PHP 8. června 1995. Tato verze byla vydána pod číslem 2 a již měla základní funkčnost jako dnešní PHP[53].

Dalším významným milníkem u PHP byl rok 1997, kdy byl dvěma izraelskými programátory přepsán hlavní parser a byl tak položen základ pro PHP 3. Také došlo ke změně významu zkratky z původního *Personal Home Page* na nové rekurzivní PHP = *PHP: Hypertext Preprocessor*. Verze PHP 3 pak byla oficiálně uvolněna v červnu 1998.

PHP 4 bylo vydáno v květnu 2000, kde největší změnou bylo zavedení pokročilého dvoustupňového systému parse/exekute syntaktické analýzi tagu – Zend engine I a zavedení takzvaný superglobálních proměnných ($_GET, $_POST, $_SESSION, …).

Verze PHP 5 přichází v červenci 2004 a nabízí Zend engine II s novým objektovým modelováním, což přineslo z pohledu PHP velkou změnu v objektovém přístupu k aplikaci. Ve verzi 5.3 pak přibyly také dlouho postrádané jmenné prostory.

- **ASP**

  ASP je skriptovací jazyk vytvořený společností Microsoft za primárním účelem zpracovávat dynamické webové stránky na serverové straně aplikace. Koncepce ASP spočívá ve strukturovaném programování webových aplikací, kde se využívá jazyků VBScript (původ z Visual Basicu) a JScript (serverová obdoba JavaScriptu).

  První verze ASP byla vydána v roce 1996. Poslední verze potom v roce 2000. Poté bylo ASP nahrazeno svým nástupcem ASP.NET.

- **ASP.NET**

  Je součást .NET frameworku, který je potřeba pro vývoj, nasazení a běh aplikace. Jedná se o přímého nástupce ASP (vznikl jako přímý konkurent JSP), který se liší zejména v objektově orientovaném přístupu k aplikaci a podpoře asi 20 jazyků, z nichž nejčastěji jsou používány C#, Visual Basic.NET, JScript.NET či Managed C++. Tato podpora je možná díky tomu, že ASP.NET založen na CLR, který je sdílen všemi aplikacemi postavenými na .NET Frameworku a stará e mimo jiné také na-

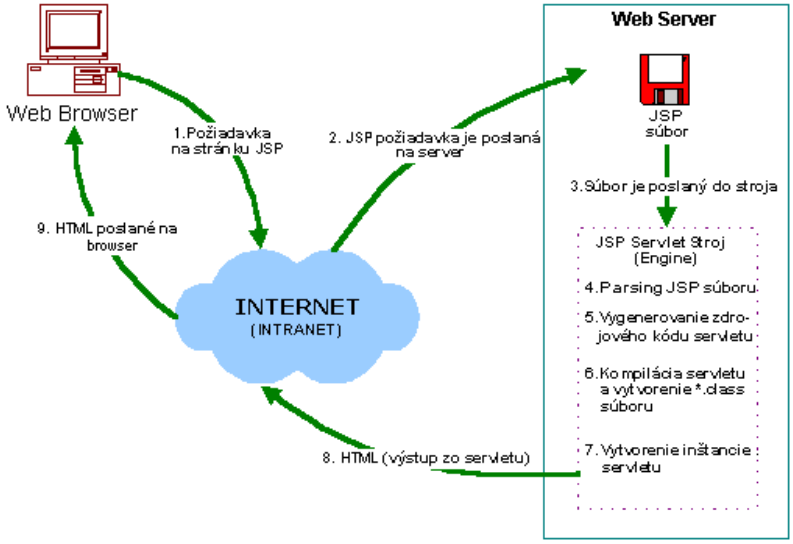
[53] Dnes je poslední stabilní verze PHP označena jako 5.3.0 a byla uvolněna 30. června 2009.

příklad o správu paměti, bezpečnost kódu a načítání potřebných komponent do paměti.

Na ASP.NET byl postaven také oficiální framework ASP.NET MVC, který umožňuje snadněji vyvíjet webové aplikace podle architektury MVC.

- **JSP**

Bylo vyvinuto společností Sun Microsystems za účelem vývoje dynamických webových aplikací. JSP je založené na programovacím jazyku Java a má podobu HTML dokumentu, ve kterém se ve speciálních značkách nachází Java kód, který generuje dynamický obsah.



Obrázek 24: Princip činnosti JSP[54]

Ukázka kódu JSP stránky:

```
<html>
	<body>
		Dnešní datum:  <%= new java.util.Date() %>
	</body>
</html>
```

- **Ruby**

Ruby je z mého pohledu velice zajímavý interpretovaný skriptovací programovací jazyk, který je především díky frameworku *Ruby on Rails* vhodný pro tvorbu dynamických webových aplikací.

Ruby má velice jednoduchou syntaxi, přesto je však dostatečně výkonný, aby dokázal konkurovat známějšímu *Perlu* či *Pythonu*. Ruby je zcela objektově orientovaný jazyk. Lze tedy říci, že v Ruby je vše objekt.

[54] Zdroj obrázku: *JavaServer Pages pro všechny | Interval.cz* (cit. 13. 8. 2009). Dostupný z WWW: http://interval.cz/podklady/1999-2008/branicky/jsp1/architecture.gif

Na samém počátku vzniku jazyka Ruby byl v roce 1993 japonský zastánce objektově orientovaného programování Yukihiro Matsumoto (přezdívaný *Matz*). Ten hledal skriptovací jazyk, který by splňoval jeho nároky. A vzhledem k tomu, že Perl mu tehdy připadal málo výkonný a Python málo objektový, začal vytvářet jazyk vlastní. A tak v roce 1995 spatřil světlo světa Ruby.

Dnes je Ruby nejvíce rozšířený v Japonsku, ale v současné době prudce roste (také díky Ruby on Rails) zájem o tento jazyk jak ve Spojených státech amerických, tak i v Evropě.

„Věřím, že – alespoň do jisté míry – je smyslem života být šťastný. Na základě tohoto přesvědčení je Ruby navrženo tak, že je nejenom snadné, ale i zábavné v něm programovat. Ruby vám umožňuje soustředit se na kreativní stránku programování, a nepřidělává vám další starosti.“[55]

Obrázek 25: Yukihiro Matsumoto (Matz)[56]

Nyní si dovolím malou ukázku jazyka Ruby v porovnání s jazykem Java. Zadání programu je shodné, a sice vypsat číslo s nejvyšší hodnotou z neutříděného pole.

[55] Yukihiro Matsumoto, předmluva k prvnímu vydání knihy Programming Ruby
[56] Zdroj obrázku: Karel Minařík – přednáška č. 3 – Úvod do programování aneb Do nitra stroje, 2007

V jazyce Java by kód vypadal takto:

```
class MaxApp {
	public static void main (String args[]) {
		int[] input = {1, 5, 3, 95, 43, 56, 32, 90, 2, 4, 19};
		int largest = input[0];
		for (int i = 0; i < input.length; i++) {
			if (input[i] > largest)
			largest = input[i];
		}
		System.out.println("Nejvyšší číslo je: " + largest + "\n");
	}
}
```

V jazyce Ruby by se stejný úkon napsal takto:

```
input = [1, 5, 3, 95, 43, 56, 32, 90, 2, 4, 19]
largest = input.first
input.each do |i|
	largest = i if i > largest
end
print "Nejvyšší číslo je: #{largest} \n"
```

Z uvedené ukázky je patrná jednoduchost a krása jazyka Ruby.

Pro vývoj dynamických webových aplikací se často využívá již zmíněný framework *Ruby on Rails*, jehož autorem je dánský programátor David Heinemeier Hansson. O síle tohoto frameworku vypovídá i legendární videoukázka[57] toho, jak si vytvořit vlastní weblog za 15 minut.

10.2 Nástroje k tvorbě dynamické webové aplikace

Kvalitní webová aplikace musí být také podpořena kvalitními nástroji nutnými pro její vývoj, spuštění a běh. Těmito nástroji jsou nejčastěji webový server, databázový server a vývojové prostředí.

V rámci implementace WCMS byly využity konkrétně tyto nástroje:

- *Webový server:* Apache 2.2 na operačním systému GNU/Linux s jádrem 2.6.28-14-generic (distribuce Ubuntu 9.04 x86_64). Na serveru byla zprovozněna technologie PHP 5.2.6-3ubuntu4.1 with Suhosin-Patch 0.9.6.2.
- *Databázový server:* MySQL 5.2 for dekan-linux-gnu (x86_64).
- *Vývojové prostředí:* Eclipse SDK 3.5 s rozšířením Aptana Studio.

10.2.1 Webový server

Webový server může mít podobu počítače či programu (démona), který má na starosti vyřizování *http* požadavků od klientů (webových prohlížečů). Vyřízením požadavku se rozumí odeslání odpovídající webové stránky (nejčastěji html dokumentu) a stavového kódu. Tento stavový kód nese informaci o tom, zda byl požadavek vyřízen dle očekávání, nebo jestli došlo k nějaké chybě. Tyto kódy se dělí do dvou velkých skupin (úspěšné a chybové)

[57] Toto video je dostupné online z: http://media.rubyonrails.org/video/rails_blog_2.mov

a jsou definovány v RFC 2616 sekce 6.1.1. Jejich výčet naleznete v příloze této diplomové práce.

Nejčastěji používanými webovými servery jsou:

Apache HTTP Server
Jedná se o softwarový webový server s otevřeným kódem spustitelný na téměř libovolné platformě. V současnosti se jedná asi o nejrozšířenější webový server.

Apache začal být vyvíjen na Illinoiské univerzitě v roce 1993 a v roce 2005 dosáhlo jeho používaní úctyhodných 69 %[58].

Velice často se používá ve spojení s Linuxem, PHP či Perlem a databázovým systémem MySQL. Pro tuto sadu svobodného softwaru se vžilo označení LAMP.

Internet Information Services
Internet Information Services (zkráceně IIS) je webový server od firmy Microsoft s podporou nejrůznějších internetových služeb jako jsou: FTP, FTPS, SMTP, NNTP a HTTP/HTTPS. Nevýhodou tohoto webového serveru je orientace pouze na platformu Microsoft Windows.

NginX
NginX se vyslovuje jako „engin X" a jedná se o vysoce výkonný webový server a e-mail server licencovaný jako BSD. Je to multiplatformní webový server, který se hojně používá na serverech s Mac OS X. Podporuje celou řadu protokolů pro web a e-mail (POP3, IMAP, SMTP).

10.2.2 Databázový server

Pod označením databázový server můžeme opět chápat buď celý počítač, na němž je uložen databázový soubor, a zároveň je zde spuštěn SQL server, který pro klienty zpracovává veškeré manipulace s daty. Nebo může být databázový server chápán jako program, který poskytuje databázové služby jiným programům či počítačům. Mezi nejčastěji používané databázové servery patří:

Oracle Database
Databázový systém od firmy Oracle Corporation představuje multiplatformní relační databázi. Označení Oracle 10g napovídá, že databázová platforma Oracle má již něco za sebou. Vývoj se datuje od roku 1977, kdy firma SDL (Software Development Laboratory) pracovala na vývoji produktu Oracle pro CIA. Tato organizace tehdy potřebovala uchovávat velké množství informací a rychle v nich podle požadavků vyhledávat.

PostgreSQL
PosgreSQL je relační databázový systém, který si zakládá na spolehlivosti a bezpečnosti. Má otevřený zdrojový kód, který je šířen pod licencí BSD. Jedná se o databázovou platfor-

[58] Vychází z výsledků měření Netcraft. Dostupné online z: http://tinyurl.com/9ppnn

mu vyvíjenou již více jak 15 let a je dostupná pro architektury Windows 32bit i Unixové systémy.

Microsoft SQL Server
MSSQL je relační databázový systém vyvíjený firmou Microsoft. Jako hlavní dotazovací jazyky jsou pro tuto databázi využívány SQL a T-SQL. Tento databázový systém běží výhradně na systémech Microsoft Windows.

SQLite
SQLite je relační databázový systém obsažený v knihovně napsané v jazyce C. Umožňuje tak „přilinkování databáze" k téměř jakékoliv aplikaci napsané například v C/C++, Delphi, PHP, Java, Python a v neposlední řadě také v již zmiňovaném Ruby. *SQLite3* poskytuje Rails dostatek metadat, aby bylo například zřejmé, že *cena* obsahuje číslo a Rails jej tak vnitřně ukládají jako BigDecimal. Z jiných databází by se mohlo stát, že *cena* přijde jako řetězec a je nutné ji konvertovat.

MySQL
MySQL je multiplatformí relační databázový systém vyvíjený švédskou firmou MySQL AB. MySQL bylo od počátku (první vydání v květnu 1995) orientováno především na rychlost, a to i za cenu některých zjednodušení. Věci jako podpora triggerů, pohledů a uložených procedur je záležitost posledních pár let.

MySQL je také považováno za průkopníka dvojího licencování, neboť je poskytováno jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí.

MySQL bývá součástí již zmiňované sady softwaru LAMP pro webové servery. Běží na něm i několik dnes populárních internetových řešení, jako jsou například Flickr, Facebook[59], Wikipedia, YouTube či Google (není myšleno pro hlavní vyhledávací službu).

10.2.3 Vývojové prostředí
Vývojové prostředí bývá často označováno anglickou zkratkou „IDE". Jde o softwarový produkt, který se snaží usnadnit práci programátorům. Obsahuje většinou editor zdrojového kódu, kompilátor, debugger, dokumentaci k programovacímu jazyku a prohlížeč struktury projektu. Některá vývojová prostředí se úzce specializují na jeden programovací jazyk, jiná například pomocí pluginů umožňují vyvíjet v různých jazycích.

Mezi nejznámější vývojová prostředí patří:

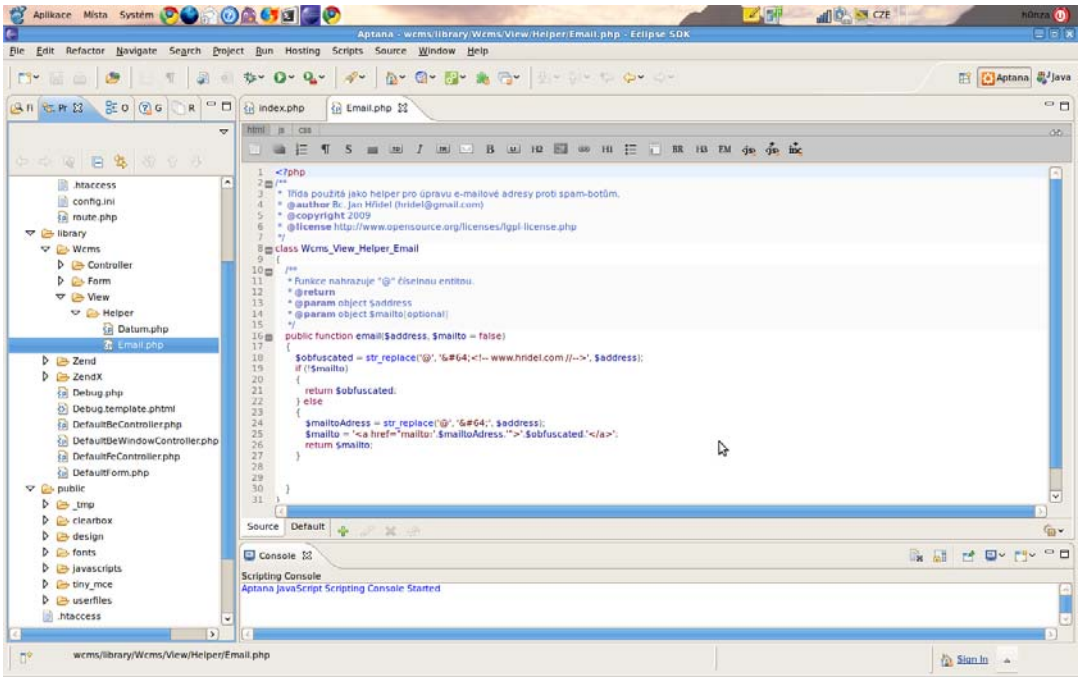
Eclipse
Eclipse je multiplatformní prostředí původně určeno pro vývoj v jazyce Java. Jeho flexibilní povaha však již od začátku počítala s tvorbou rozšiřujících pluginů, a tak je dnes eclipse využíváno k vývoji v téměř jakémkoliv programovacím jazyce. A to nejen včetně PHP či Ruby, ale dokonce také například umožňuje tvorbu UML návrhu.

[59] Zdroj informace: http://blog.facebook.com/blog.php?post=7899307130

Jelikož Eclipse je v podstatě na existenci pluginů postaveno, vznikla přímo pod křídly Eclipse celá řada „subprojektů“ orientujících na jednotlivá odvětví softwarového vývoje. Mezi oficiální subprojekty patří:

- *Enterprise Development* – vývoj J2EE
- *Embedded + Device Development* – vývoj J2ME
- *Rich Client Platform* – vývoj nad platformou RCP
- *Application Frameworks* – frameworky pro vývoj nástrojů postavených nad Eclipse
- *Language IDE* – vývoj v dalších podporovaných jazycích jako C++ nebo PHP

Od verze Eclipse 3 se přešlo na novou architekturu pluginů a ty jsou nyní dynamicky nahrávány až ve chvíli, kdy jsou skutečně potřeba. To přineslo celkově menší paměťové nároky a zrychlení nejen startu aplikace.



Obrázek 26: Vývojové prostředí Eclipse na Ubuntu[60]

NetBeans

NetBeans je open source projekt s rozsáhlou komunitou uživatelů i vývojářů. Hlavním sponzorem projektu je firma Sun Microsystems a na vývoji se nejvíce podílí čeští vývojáři v pražské pobočce na Chodově.

NetBeans je neprogramováno v jazyce Java a je také primárně určeno pro vývoj v jazyce Java. Od verze 6.0 však už plně podporuje také například vývoj v C++, PHP či Ruby. Usnadňuje práci s frameworky jako jsou Struts, Ruby on Rails a umožňuje mimo jiné také vývoj SOAP aplikací a webových služeb.

[60] Zdroj obrázku: vlastní

V současné době je možné NetBeans využívat zdarma bez jakéhokoliv omezení na systémech Microsoft Windows, GNU/Linux, Mac OS X a Solaris.

JDeveloper
Vývojové prostředí od firmy Oracle Corporation. Specializované na vývoj v jazyce Java (včetně podpory HTML a JSP).

Zend Studio
Rozhraní od firmy Zend navržené pro dosažení maximální produktivity PHP programátorů. Toto vývojové prostředí je nabízené s pronájmem licence na 1 rok ($ 399) nebo na 3 roky ($ 717)[61].

KDevelop
KDevelop je svobodné velmi sofistikované vývojové prostředí určené pro KDE. V současné verzi 3.5 podporuje celou řadu jazyků (Ada, Bash, C, C++, Fortran, Java, Pascal, Perl, PHP, Python, Ruby a další). Prostředí nedisponuje vlastním kompilátorem, ale využívá kompilátorů externích (např. GCC).

Microsoft Visual Studio
Microsoft Visual Studio je velice propracované vývojové prostředí od společnosti Microsoft. Umožňuje vývoj konzolových i formulářových aplikací pro platformy Microsoft Windows, Windows Mobile, Windows CE, .NET a Microsoft Silverlight. V rámci .NET frameworku umožňuje také relativně efektivní vývoj webových aplikací.

Programovací jazyky jsou ve Visual Studiu podporovány prostřednictvím jazykových služeb. Díky tomu mohou editor i debugger podporovat libovolný programovací jazyk. Integrovanými jazyky jsou C/C++, VB.NET, ASP.NET a C#. Ostatní jazyky je třeba prostřednictvím jazykových služeb doinstalovat zvlášť.

10.3 Architektura webových aplikací

V době, kdy vývojová prostředí nabízí možnost navrhovat webové aplikace způsobem „drag & drop“, hrozí nebezpečí, že kód zajišťující vzhled aplikace bývá promíchán s kódem pracujícím s datovým úložištěm, což může vést ke značným problémům jako je například:

- Velice obtížné hledání chyb v obrazovce s tisíci řádky.
- Několikanásobné zobrazování stejných dat (seznam zákazníků a detail zákazníka).
- Opakování zdrojového kódu na více místech.

S řešením nejen těchto problémů přichází architektura MVC.

[61] Oficiální ceny aktuální v srpnu 2009.

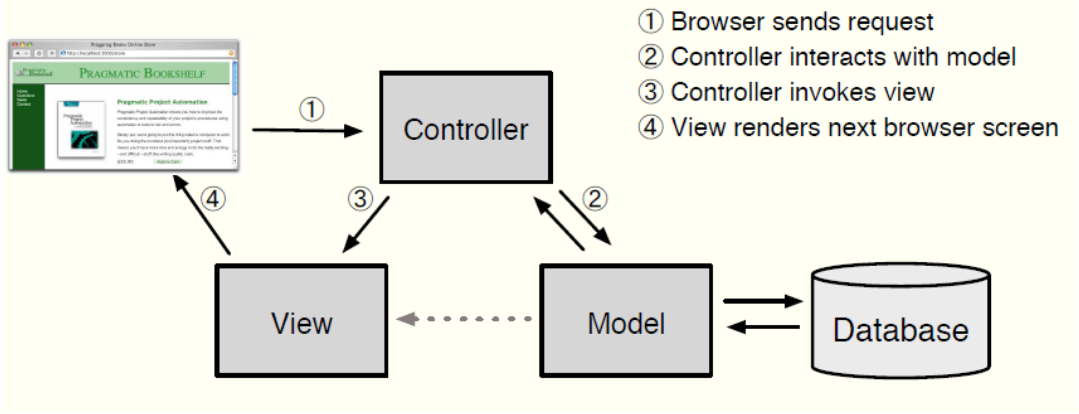
10.3.1 Modely, pohledy a řadiče

Už v roce 1979 přišel Trygve Reenskaug s novou architekturou pro rozvoj interaktivních aplikací. Podle jeho návrhu byly aplikace rozděleny do tří typů komponent: modely-pohledy-řadiče (models-views-controllers).

Model bývá zodpovědný za udržování stavu aplikace. Někdy je tento stav přechodný a trvá jen několik málo interakcí s uživatelem. Někdy však bývá trvalý a ukládá se mimo aplikaci – často do databáze.

Model je víc než jen data. Uplatňuje také veškerá bussines pravidla, která jsou na data aplikována. Například: Pokud nemůže být sleva použita u objednávek menších než sto korun, model bude prosazovat toto omezení. Umístěním těchto omezujících pravidel do modelu se ujišťujeme, že nic jiného v aplikaci nemůže naše data zneplatnit.

Pohled bývá zodpovědný za vytváření uživatelského rozhraní, obvykle založeného na datech z modelu. Například e-shop má seznam produktů, který se zobrazuje na obrazovce katalogu. Tento seznam bude přístupný přes model, ovšem bude to právě pohled, který bude data formátovat pro koncového uživatele. Přestože pohled může zobrazovat uživateli různá data různými způsoby. Pohled nikdy nebude nijak manipulovat s příchozími daty. Práce pohledu končí ve chvíli, kdy jsou data úplně zobrazena. Může proto existovat mnoho pohledů, které budou přistupovat ke stejné množině dat, často pro různé účely. V e-shopu budou určitě existovat jednak pohledy zobrazující produkty v produktovém katalogu a jednak pohledy pro administrátory, kteří produkty přidávají a editují.



Obrázek 27: Architektura MVC[62]

Řadiče (Controllers) organizují žádosti. Kontroléry přijímají události z vnějšího světa (běžného uživatelského vstupu). Spolupracují s modelem a zobrazují vhodný pohled uživateli.

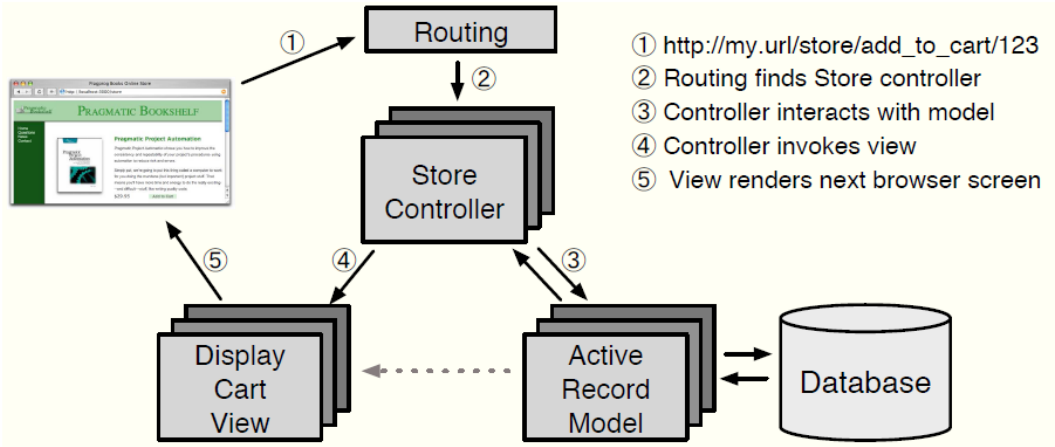
Tento triumvirát (model, pohled a řadič) tvoří dohromady architekturu známou jako MVC.

[62] Zdroj obrázku: RUBY, Sam, THOMAS, Dave, HANSSON, David Heinemeier. *Agile Web Development with Rails*. 3rd edition. United States of America: The Pragmatic Programmers LLC, 2008. 730 s. ISBN 978-1-9343561-6-6.

Architektura MVC byla původně určena klasickým GUI aplikacím, kde toto rozdělení vedlo k používání mnohem méně propojení. To ve výsledku znamenalo opět jednodušší psaní a udržování kódu.

Ve světě softwarového vývoje se často ignorují dobré nápady z minulosti, a tak i vývojáři webových aplikací původně psali dlouhé monolitické kódy, které v sobě kombinovali přístup do databáze, business logiku a manipulaci s událostmi. Ovšem nápady z minulosti se začaly opět pomalu objevovat a lidé začali s architekturami experimentovat i u webových aplikací, což začalo odrážet 20 let staré nápady v MVC. Výsledkem byl vznik frameworků, jako jsou WebObjects, Struts a JavaServer Faces. Všechny jsou s různým stupněm věrnosti založeny na myšlenkách MVC.

Velice dobře podporují architekturu MVC webové frameworky *Ruby on Rails* a *Zend Framework*, jehož vývojáři se Rails v mnohém inspirují. Jak funguje architektura MVC v Rails znázorňuje následující obrázek.



Obrázek 28: Architektura MVC v Ruby on Rails[63]

Jak je z obrázku patrné, jakýkoliv požadavek z webového prohlížeče je v Rails (ale taktéž v Zend Frameworku s využívanou podporou MVC) zaslán nejprve routeru, který vyhodnotí, kam do aplikace požadavek poslat a jak by měl být požadavek analyzován. Nakonec tato fáze identifikuje konkrétní metodu (v Rails i v Zend Frameworku označovanou jako „akce“) někde v kontroléru. Tato akce může a velice často komunikuje s modelem, což může mít za následek volání dalších metod. Nakonec akce nachystá informace pro zobrazení, které odešle do pohledu, jenž je vyobrazí uživateli.

Na obrázku jsou konkrétně znázorněny akce, které se vykonají po kliknutí na tlačítko „Přidat do košíku“ na detailní stránce produktu. Stisknuté tlačítko vyšle http požadavek ve tvaru *http://my.url/store/add_to_cart/123*. Router tento požadavek vyhodnotí tak, že bude zavolána metoda *add_to_cart* v kontroleru *StoreControlleru* (o konvencích pojmenovávání – i

[63] Zdroj obrázku: RUBY, Sam, THOMAS, Dave, HANSSON, David Heinemeier. *Agile Web Development with Rails*. 3rd edition. United States of America: The Pragmatic Programmers LLC, 2008. 730 s. ISBN 978-1-9343561-6-6.

ta je v Zend Frameworku částečně převzata z Rails – se zmíním později v kapitole o Zend Frameworku). Poslední část URL adresy (*123*) označuje vnitřní identifikátor vybraného produktu.

Akce (metoda) *add_to_cart* začne zpracovávat požadavek uživatele. V tomto případě nejprve najde košík aktuálního uživatele (což je objekt spravovaný modelem). Zároveň požádá model, aby našel informace o produktu s id *123*. Potom řekne košíku, aby do sebe přidal tento produkt. Takže model má na starosti veškerou business logiku a práci s daty. Kontrolér jen říká modelu to, *co* má dělat. A model ví, *jak* to má dělat.

Ve chvíli, kdy je do nákupního košíku vložen nový produkt, ho můžeme zobrazit uživateli. Kontrolér řídí věci tak, že pohled má skrz něj přístup k objektu košík v modelu a vyvolá kód pro zobrazení. V Rails a v Zend Frameworku je toto volání často implicitní – podle úmluv o pojmenovávání souborů v aplikaci.

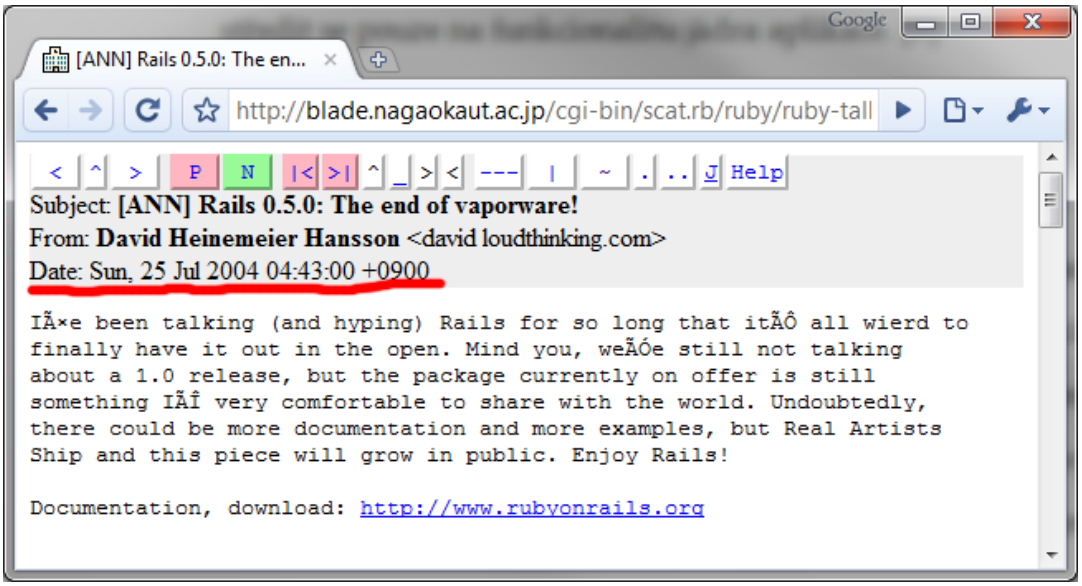
Takto v podstatě dnes funguje architektura MVC u webových aplikací. Díky dodržování konvencí a správného členění funkcionality se stává kód aplikace mnohem přehlednější a jednodušší na údržbu a rozšiřování.

Mohlo by se zdát, že při dodržování správného členění kódu můžeme dosáhnout MVC architektury i bez nutnosti používat framework typu Rails nebo Zend Framework. Ovšem používání takovéhoto frameworku je výhodné neboť framework sám se stará o veškerá propojení jednotlivých vrstev. Umožňuje tak vývojáři soustředit se pouze na funkcionalitu jádra aplikace.

10.3.2 Zend Framework

Zend Framework (často označovaný jako ZF) je objektově orientovaný webový aplikační framework implementovaný v PHP 5 a licencovaný pod New BSD License. ZF je orientovaný na rychlý vývoj webových aplikací. Je konstruován modulárně, což umožňuje vývojáři využívat jen ty třídy, které skutečně potřebuje, ovšem musí dbát na některé závislosti. Například nelze využívat komponentu *Zend_Db_Table* bez *Zend_Db*. ZF si lze představit jako knihovnu s mnoha třídami připojenou k vyvíjené aplikaci. ZF disponuje nástroji pro práci s databází, lokalizačními (jazykovými) nástroji, podporou architektury MVC a dalšími. Jeho vývoj započala firma Zend Technologies Inc. v roce 2005, tedy ve chvíli kdy už například framework Ruby on Rails začal ve světě (převážně tedy ve Spojených státech amerických) získávat na popularitě. Jak ukazuje obrázek 29, první vydání Rails bylo již v roce 2004.

Jelikož Rails jsou skutečně famózním frameworkem, není divu, že právě Rails se ZF inspiruje nejvíce.



Obrázek 29: První zmínka o Rails na Internetu v roce 2004[64]

Požadavky

Od verze 1.7.0 vyžaduje Zend Framework pro bezproblémový chod minimálně PHP verzi 5.2.4. V produkčním prostředí však vývojáři důrazně doporučují používat PHP 5.2.3 nebo novější z důvodu většího zabezpečení a výkonu. Ke spouštění testů je potřeba mít nainstalovanou minimálně knihovnu PHPUnit 3.0.

Klíčové vlastnosti

- Veškeré komponenty v Zend Frameworku jsou plně objektově orientované a vyhovují direktivě E_STRICT.
- Modulární architektura minimalizuje křížové zavislosti mezi komponentami.
- Podpora architektury MVC se šablonovým systémem a podporou Layouts.
- Poskytuje abstraktní vrstvu nad databázemi MySQL, Oracle, IBM DB2, MSSQL Server, PostgreSQL, SQLite a Informix Dynamix Server.
- Podpora cache.

Často používané komponenty

- Zend_Acl – Jednoduchý a flexibilní systém pro správu uživatelských oprávnění
- Zend_Auth – Autentizace uživatelů s mnoha druhy úložišť
- Zend_Cache – Implementace cache systému s úložišti ve formě paměti, souboru, APC, SQLitea, atd.
- Zend_Config – Slouží k nastavení aplikace skrze konfigurační soubory
- Zend_Controller – Implementace Model-View-Controller (MVC) architektury
- Zend_Date – Komponenta pro práci s daty
- Zend_Db – Implementace multidatabázové vrstvy
- Zend_Dojo – Knihovna pro práci s javascriptovým frameworkem Dojo
- Zend_Filter – Komponenta pro filtrování uživatelských dat s velkým množstvím filtrů

[64] Zdroj obrázku: vlastní

- Zend_Form – Objektový vývoj webových formulářů včetně filtrování hodnot a jejich valildace
- Zend_Layout – Správa layoutů aplikace
- Zend_Log – Komponenta pro logování určitých dat s množstvím backendů
- Zend_Mail – Tvorba e-mailů, správa e-mailoých schránek
- Zend_Memory – Podpora pro zpracování dat s omezeným množstvím dostupné paměti
- Zend_OpenID – Implementace OpenID klienta i serveru
- Zend_Paginator – Komponenta pro práci se stránkováním dat
- Zend_Pdf – Objektový přístup a vytváření PDF souborů
- Zend_Registry – Komponenta pro uchovávání objektů a hodnot v aplikační vrstvě
- Zend_Translate – Podpora pro překlady a různé jazykové mutace aplikací
- Zend_View – Šablonový systém
- ZendX_Jquery – Podpora javascriptoveho frameworku jQuery

ZendX_Jquery je třída podporující práci s javascriptovým frameworkem jQuery, který velice usnadňuje zapojit moderní uživatelsky příjemné prvky do webové aplikace. Tato třída není součástí klasické knihovny *Zend*, ale rozšířené knihovny *ZendX*. To proto, že vývojáři ZF se rozhodli oficiálně podporovat jiný javascriptový framework a sice *Dojo Toolkit*. Existuje tedy oficiální třída *Zend_Dojo*. Oba frameworky si jsou svou funkcionalitou velice podobné.

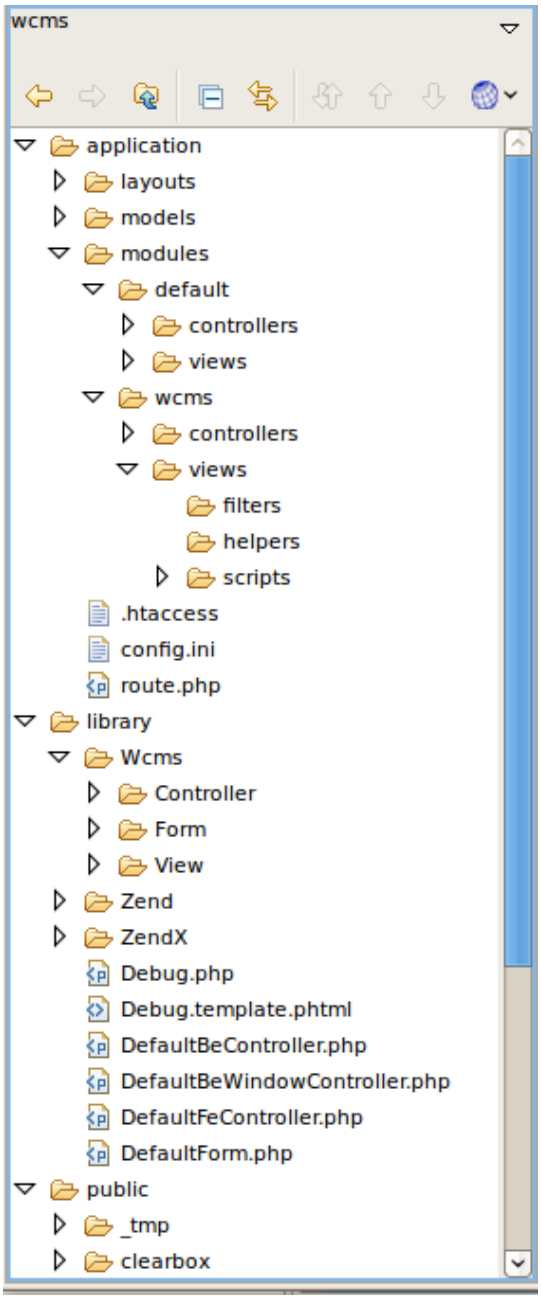
Při implementaci WCMS jsem se rozhodl pro jQuery neboť mi mnohdy přijde značně rychlejší a také dle mého názoru nabízí lepší podporu komunitou uživatelů.

Adresářová struktura

Pro implementaci WCMS jsem zvolil následující adresářovou strukturu aplikace, která vychází z obecných doporučení adresářové struktury pro práci se Zend Frameworkem. V „root adresáři" webové aplikace jsou dva nejdůležitější adresáře, a sice *application* a *library*. Potom je vhodné vytvořit ještě adresář, který se nejčastěji nazývá *public*. Zatímco pro adresáře application a library platí jistá pravidla, co se obsahu a podadresářů týká, pro adresář public už žádná pravidla neplatí a je zcela v rukou programátora, jak s jeho obsahem naloží.

Já v něm zvolil podadresář pro dočasné soubory *_tmp*, adresář *javascripts* pro soubory obsahující JavaScriptový kód, adresář *design* pro soubory s kaskádovými styly a pro obrázky tvořící vizuální vzhled aplikace (rozděleno v dalších podadresářích).

Do adresáře application patří většina vývojářem naprogramovaného kódu, a to soubory tvořící layout, hlavní modelové třídy a kontroléry rozdělené do příslušných modulů (já využívám dva hlavní moduly: *default* – pro frontend aplikace a *wcms* – pro backend aplikace). Vedle kontrolérů se zde také vyskytují soubory spojené s pohledovou vrstvou architektury MVC.



Obrázek 30: Adresářová struktura implementované aplikace[65]

V adresáři library se potom nachází samotné knihovny *Zend* potažmo *ZendX* a případně také vlastní knihovny.

V mém případě jsem vytvořil adresář *Wcms*, ve kterém jsou především specifikované formuláře a „hlepery" pro zobrazení data v hezky českém formátu a emailových adres.

Konvence pojmenování souborů a tříd

K pojmenování běžných souborů je povoleno používat výhradně alfanumerické znaky, podtržítka a spojovníky „-". Používání mezer v názvech souborů je přísně zakázáno.

[65] Zdroj obrázku: vlastní

Veškeré soubory, které obsahují PHP kód, by měly končit příponou „.php“, s výjimkou souborů pohledů (obvykle v adresáři *view/skript/*), které mívají příponu „.phtml“. Následující příklady ukazují povolené pojmenování souborů pro Zend Framework:

```
Zend/Db.php
Zend/Controller/Front.php
Zend/View/Helper/FormRadio.php
```

Pojmenování funkcí a metod se řídí pravidlem, které bývá často nazýváno jako „camelCase“ formátování. To znamená, že jsou povoleny v názvech funkcí pouze alfanumerické znaky, přičemž od používání číslic se ve většině případů odrazuje a vždy se začíná malým písmenem. Pokud je název složen z více slov, je každé počáteční písmeno následujícího slova psáno verzálkou. Příklad správně pojmenovaných funkcí:

```
filterInput()
getElementById()
widgetFactory()
```

Pojmenování proměnných je povoleno alfanumerickými znaky, přičemž je doporučeno se číslicím vyhýbat. Proměnné instancí, které jsou deklarovány jako „private“ či „protected“ by měly začínat jedním podtržítkem. Stejně jako u funkcí by měl název proměnné začínat minuskou, která by měla být následována ve stylu „camelCaps“. Samozřejmostí je pochopitelně dodržování syntaxe jazyka PHP, takže úplně prvním znakem je vždy „`$`“.

Obecně se doporučuje být „upovídaný“, tedy aby název proměnné zřetelně vystihoval data, která obsahuje. Všeobecně oblíbené proměnné typu `$i` nebo `$n` by měly být použity pouze v případech, kdy značí počítadlo iterací v nějakém cyklu. Ten by ovšem neměl být delší než 20 řádků kódu. Jinak by se měl použít výstižnější název.

Názvy konstant by měly obsahovat pouze alfanumerické znaky a podtržítka, přičemž ta by měla sloužit výhradně k oddělení veškerých slov, tedy správně takto „`EMBED_SUPPRESS_EMBED_EXCEPTION`“. Pojmenování, kde nejsou oddělena všechna slova jako například „`EMBED_SUPPRESSEMBEDEXCEPTION`“, není povoleno. Konstanty by měly být definovány jako atributy třídy pomocí modifikátoru `const`. Použití `define` je povoleno, není však příliš doporučováno.

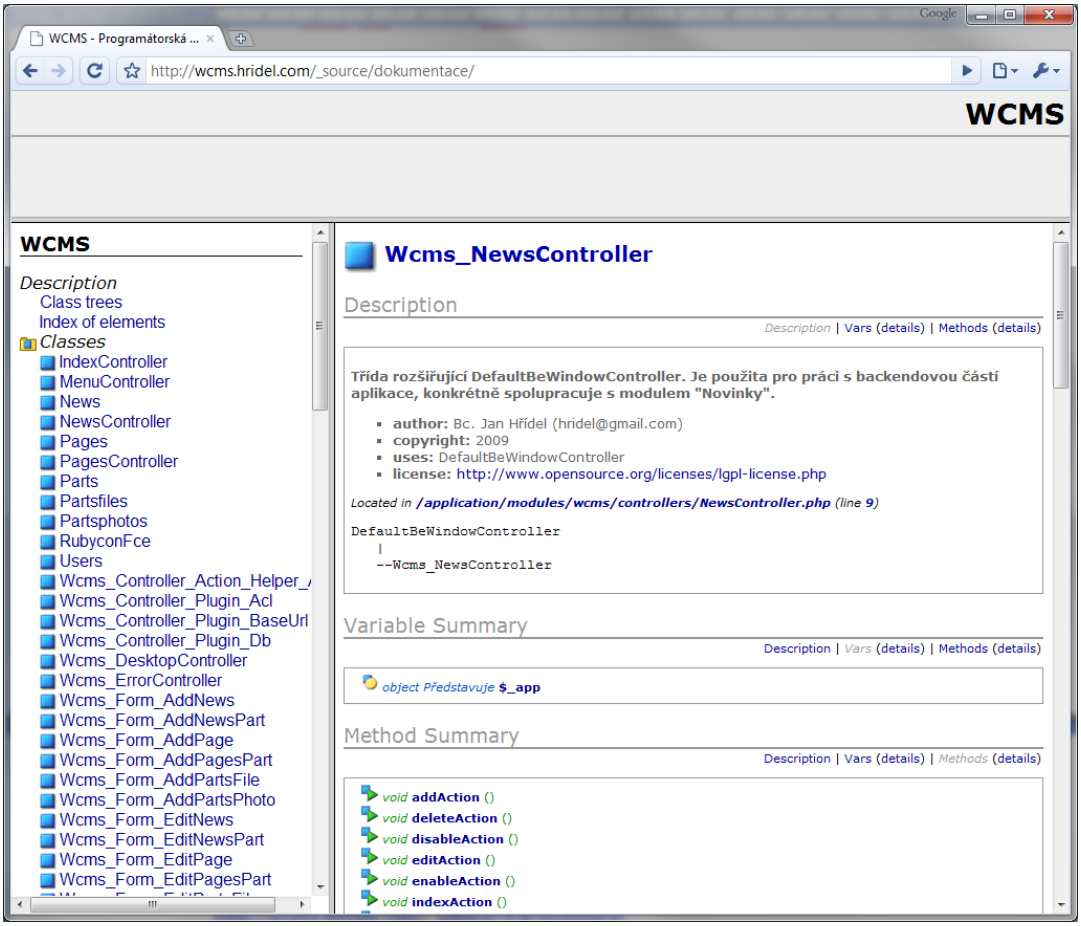
Zend Framework standardizuje pojmenování tříd způsobem, kde název třídy mapuje cestu k adresáři, ve kterém je třída uložena. Povoleno je používat alfanumerické znaky a podtržítka, přičemž jsou upřednostňována jen písmena a podtržítka by se měla používat pouze v místě, kde normálně lomítko značí novou úroveň adresářového stromu. Tedy například třída uložená v *Zend/Db/Table.php* by měla nést název `Zend_Db_Table`. Pokud je název třídy složen z více slov, mělo by každé slovo začínat velkým písmenem. Jestliže nastane situace, že třída je pojmenována zkratkou z více slov, jen první písmeno zkratky se píše velké, tedy správně `Zend_Pdf`, nikoli ~~`Zend_PDF`~~.

Tato konvence definuje jakýsi mechanismus pseudo-jmenných prostorů, které až do vydání verze 5.3[66] v PHP chyběly.

Všechny třídy, které představují kontroléry, by měly obsahovat na konci názvu slovo *Controller*. Tedy třída by měla být pojmenována například takto: `Wcms_PagesController` a měla by být uložena v adresáři *modules/wcms/controllers/* a soubor by se měl jmenovat *PagesController.php*.

Konkrétní implementace analyzovaného WCMS

Implementace WCMS je postavena na Zend Frameworku v maximální míře. Bližší představu o jednotlivých souborech a struktuře naprogramovaného kódu poskytne zcela jistě programátorská dokumentace v HTML formátu, která je přiložena na CD, případně dostupná online na adrese: http://wcms.hridel.com/_source/dokumentace/



Obrázek 31: Programátorská dokumentace[67]

Nyní bude popsáno několik klíčových částí pro funkčnost celé aplikace. Aplikace je postavena na architektuře MVC, přičemž model přistupuje k datům uloženým v MySQL databázi. Proto třídy v modelu jsou poděděny od třídy `Zend_Db_Table`. Aby však bylo možné databázové spojení využívat, musí být nejprve navázáno, a k tomu se využívají přístupové

[66] Verze PHP 5.3 s podporou jmenných prostorů byla vydána 30. června 2009
[67] Zdroj obrázku: vlastní

údaje, které jsou uloženy v souboru `config.ini` v adresáři `application`. Tento konfigurační soubor je zpracováván třídou `Zend_Config_Ini`. V souboru `index.php` voláním:

```
$config = new Zend_Config_Ini('./application/config.ini', UMISTENI);
```

`UMISTENI` představuje konstantu, která představuje parametr určující, jaká část konfiguračního souboru se načte. Konfigurační soubor tedy vypadá takto:

```
[local]
db.adapter = PDO_MYSQL
db.host = 127.0.0.1
db.username = lokální uživatel
db.password = lokální heslo
db.dbname = wcms

[live]
db.adapter = PDO_MYSQL
db.host =   localhost
db.username = live uživatel
db.password = live heslo
db.dbname = wcms
```

Protože soubor *config.ini* není nijak šifrovaný, je třeba zakázat webovému prohlížeči přístup k tomuto souboru (nebo všem souborům v adresáři application). Toho je docíleno serverovým lokálním konfiguračním souborem *.htaccess*, který je uložen taktéž v adresáři *application*. Soubor *.htaccess* vypadá velice jednoduše následovně:

```
deny from all
```

Serverový soubor *.htacces* musí být uložen také v kořenovém adresáři webové aplikace. Jeho úkolem je zajistit, aby všechny požadavky od klienta směřující na danou doménu byly – pokud nevedou na existující soubor – přesměrovány hlavní na soubor *index.php*. Mimo to může tento soubor také konfigurovat nějaké nastavení PHP.

Výpis kořenového souboru *.htaccess*:

```
#php nastaveni
php_value arg_separator.output &amp;
php_value session.use_trans_sid 0
php_value upload_max_filesize 20M
php_value post_max_size 20M
php_value max_execution_time 200
php_value max_input_time 200

php_flag magic_quotes_gpc off
php_flag register_globals off

RewriteEngine On
RewriteCond %{REQUEST_FILENAME} -s [OR]
RewriteCond %{REQUEST_FILENAME} -l [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^.*$ - [NC,L]
RewriteRule ^.*$ index.php [NC,L]
```

Do souboru *index.php* je potom připojen soubor *route.php* z adresáře application. Tento soubor řeší za pomoci třídy `Zend_Controller_Router` pravidla vyhodnocování URL adres. To usnadňuje optimalizovat tvar frontendových URL adres pro internetové vyhledávače.

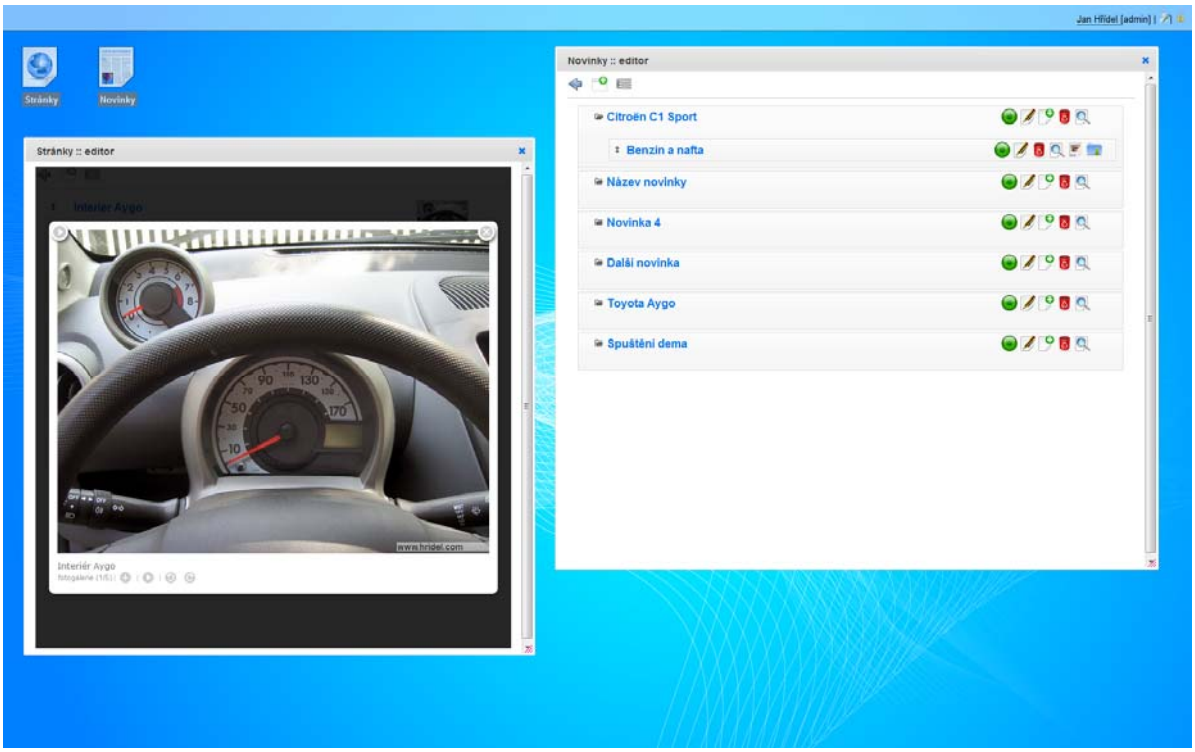
Zend Framework pracuje ve výchozím stavu s URL adresami, které bývají ve tvaru: *http://domena/controller/action/param1/value1/param2/value2/...* Konkrétně by například adresa pro zobrazení detailu stránky věnující se vozidlům Toyota Aygo vypadala takto: *http://wcms.hridel.com/pages/index/seo_url/toyota-aygo*. To není špatné, ale díky třídě `Zend_Controller_Router_Route` může být snadno nastaveno, že oddíl stránky bude mít tvar: *http://domena/:seo_url*. Tím docílíme toho, že skutečná URL adresa stránky o Toyotě Aygo bude vypadat následovně: *http://wcms.hridel.com/toyota-aygo*, což je mnohem hezčí jak pro vyhledávače, tak zajisté i pro uživatele. V *route.php* tuto situaci řeší pravidlo:

```
$router = $frontController->getRouter();
    /**
     * Routování klasických stránek
     * @var route
     */
    $route = new Zend_Controller_Router_Route(
        ':seo_url',
        array('module' => 'default','controller' => 'pages','action' => 'index')
    );
    $router->addRoute('pages', $route);
```

Obrázek 32: Ukázka URL adresy detailu stránky[68]

[68] Zdroj obrázku: vlastní

Při implementaci WCMS byl kladen důraz kromě podpory SEO také na příjemné uživatelské rozhraní. V tomto směru jsem se inspiroval moderními desktopovými prostředími a snažil jsem se tak přizpůsobit uživatelské rozhraní webové aplikace – WCMS – pracovnímu prostředí tak, jak jsou na něj uživatelé zvyklí z Microsoft Windows, KDE či Gnome. Rozhraní proto disponuje velkými pohyblivými ikonami, plovoucími nemodálními dialogovými okny s možností měnit jejich velikost. Pohyblivá okna umožňují mít otevřeno více oken zároveň (například současně editovat novinky a fotogalerii stránek). Systém využívá také technologie Ajax pro zlepšení interaktivity s uživatelem. U monitorů s menším rozlišením doporučuji použití zobrazení webového prohlížeče na celou obrazovku. Potom může uživatelské rozhraní vypadat následovně.



Obrázek 33: Uživatelské rozhraní implementovaného WCMS[69]

[69] Zdroj obrázku: vlastní

11 Testování produktu

Testování je velice důležitá část, která provází celý proces implementace systému. Proces testování bývá podmnožinou procesu ověřování a plánování kvality. Úkoly testovacího týmu tedy mohou být značně široké. Testování by mělo ve výsledku odhalit chyby před finálním nasazením systému a tím ušetřit případné náklady na opravení později uživateli objevených chyb.

11.1 Kategorie testů

11.1.1 Testování statické a dynamické

Při tomto druhu rozdělení se vychází toho, jestli je k testování potřeba software spouštět.

Statické testování lze úspěšně provádět ještě před vytvořením prvního prototypu. Výsledkem takovýchto testů může být například zpřesnění odhadu náročnosti na čas (člověkohodin) a zdroje (strojohodin).

Dynamické testování vyžaduje spustitelnou verzi produktu a většinou probíhá na základě poskytování různých vstupů a hodnocení výstupů.

11.1.2 Černá a bílá skříňka

Smyslem testování černé skříňky (v anglické literatuře uváděno jako „black-box test“) je testování softwaru jako celku z pohledu uživatele, tedy bez znalosti konkrétní implementace. Zjišťuje se, zda software disponuje takovým chováním, jaké je od něho očekáváno.

Při testování bílé skříňky (v angličtině označované výrazy „glass-box test“, „white-box test“ případně „clear/transparent-box test“) už má tester přístup i ke zdrojovému kódu. Ztrácí tedy částečně pohled uživatele, ale může mu pomoci odhalit, kde hledat chybu.

11.1.3 Automatické a manuální testování

Manuální testy se používají, pokud test vyžaduje lidské ohodnocení a úsudek nebo rozličné přístupy, které není potřeba často opakovat.

Naproti tomu moderní webové frameworky jako Ruby on Rails nebo Zend Framework nám umožňují psát také automatické testy. Ty jsou používány zejména pro zátěžové testy, kdy je generováno velké množství dat.

Automatizované testování nekontroluje správnost výsledků, ale změnu chování od předešlého testu.

Proč tedy psát testy?

- Testy jsou nejmocnějším pomocníkem agilního vývoje.
- Testy jsou obrana „proti sobě samému“.
- Testy jsou obrana proti příšerám (legacy code).

12 Nasazení systému

Fáze nasazení softwarového produktu následuje po ukončení fází vývoje a testování a zpřístupňuje finální verzi aplikace uživatelům, pro které byl software vyvíjen. Jelikož každý produkt je svým způsobem specifický, měl by proces nasazení odpovídat taktéž specifickým požadavkům pro nasazení. Tyto požadavky mohou vycházet z činností, ze kterých bývá proces nasazení složen.

Vydání
Vydání je fáze následující po dokončení vývoje. Obsahuje veškeré operace nutné k uskutečnění přenosu systému na stranu zákazníka. Zahrnuje taktéž nutné informace o zdrojích nutných pro chod systému.

Zavedení a spuštění
Jedná se o činnost, kdy jsou do provozu uvedeny veškeré spustitelné komponenty systému.

Ukončení
Ukončením je nazývaná fáze, která zajišťuje „vypnutí" systému – znemožnění jeho spouštění. Například těsně před provedením aktualizace.

Přizpůsobení
Jedná se o fázi adaptace systému na straně zákazníka. Například změna prostředí.

Aktualizace
Fáze označovaná také jako update systému. Jedná se o aktualizaci celku nebo části systému na novou verzi.

Vestavěný update
Vestavěným updatem se chápe systém plné nebo částečné automatické aktualizace.

Odinstalace
Odinstalování představuje kompletní odebrání softwarového produktu. Obvykle kvůli dlouhodobému nepoužívání systému.

Stažení
Stažení je fáze, ve které je produkt označen za zastaralý a je ukončena jeho podpora. Označuje tak konec celého životního cyklu softwarového produktu.

Proces nasazení demoverze implementovaného WCMS byl spojen se založením domény třetího řádu[70] a se založením MySQL databáze na straně poskytovatele webhostingu, který zajišťuje dostupnost aplikace z Internetu.

Jelikož systém byl vyvíjen na lokálním serveru, bylo nutné před nasazením provést jednu změnu v konfiguračním souboru *conf.php*, a to sice změnit hodnotu konstanty `LOCAL` z `true` na `false`.

[70] Demo WCMS systému je spuštěno na URL adrese http://wcms.hridel.com/

13 Závěr

Cílem této práce bylo zmapovat moderní webové technologie související s termínem web 2.0. Následně provést analýzu (s využitím UML a UP) nového WCMS, který by umožňoval podporu těmto specifikům a zároveň plnil nefunkční požadavky zadavatele, kterými bylo především využití technologií PHP 5 a MySQL 5.

Součástí zadání bylo také vytvoření funkční demo verze tohoto WCMS s programátorskou dokumentací a okomentovaným kódem. WCMS mělo být postaveno na architektuře MVC, což by mělo umožňovat snadné rozšíření o další funkcionalitu.

Na začátku analýzy byly zjišťovány požadavky na nový systém, na jejichž základě byly vytvořeny diagramy případů užití. Vybrané případy užití byly obohaceny o základní a alternativní scénáře.

Při návrhu analytického modelu bylo využito metody štítkování pro nalezení analytických tříd. Tento analytický model byl rozšířením o více atributů a metod převeden do modelu návrhového.

Jelikož požadavkem bylo uchovávat data v databázovém systému MySQL, byl před samotnou implementací vytvořen ještě datový model.

V další části byly již popisovány možnosti samotné implementace. A vytvořena elektronická podoba programátorské dokumentace.

Během vypracování této práce jsem se sám seznámil s novými prvky agilního vývoje webových aplikací, což shledávám obrovským přínosem pro mé další působení a zdokonalování se v tomto oboru.

Zároveň by měl výsledek této práce posloužit jako podklad pro vybudování nové verze Web CMS – stěžejního produktu pro dvě nyní již spolupracující firmy na poli tvůrců profesionálních webových řešení *eBrána s. r. o.* a *Wizards CZ s. r. o.*

POUŽITÁ LITERATURA

[1] PETRÁŇ, Milan. *Systémy pro správu a publikaci webového obsahu* [online]. 2008 [cit. 2009-04-03]. Dostupný z WWW: <http://businessworld.cz/ecm-dm/systemy-pro-spravu-a-publikaci-weboveho-obsahu-2295>.

[2] PHP - Wikipedie, otevřená encyklopedie [online]. 2009 , 7. 7. 2009 [cit. 2009-08-12]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/PHP>.

[3] *Active Server Pages - Wikipedie, otevřená encyklopedie* [online]. 2009 , 18. 6. 2009 [cit. 2009-08-12]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/ASP>.

[4] *ASP.NET - Wikipedie, otevřená encyklopedie* [online]. 2009 , 14. 7. 2009 [cit. 2009-08-13]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/ASP.NET>.

[5] BRANICKÝ, Marek. *JavaServer Pages pro všechny | Interval.cz* [online]. 2002 , 6. 8. 2002 [cit. 2009-08-14]. Dostupný z WWW: <http://interval.cz/clanky/javaserver-pages-pro-vsechny/>.

[6] *Ruby (programovací jazyk) - Wikipedie, otevřená encyklopedie* [online]. 2009 , 12. 6. 2009 [cit. 2009-08-13]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Ruby_%28programovac%C3%AD_jazyk%29>.

[7] LACKO, Luboslav. *Oracle: Správa, programování a použití databázového systému*. Marek Kocan. 2. dopl. vyd. Brno: Computer Press, a.s., 2007. 576 s. ISBN 978-80-251-1490-2.

[8] *PostgreSQL - Wikipedie, otevřená encyklopedie* [online]. 2009 , 6. 7. 2009 [cit. 2009-08-13]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/PostgreSQL>.

[9] RUBY, Sam, THOMAS, Dave, HANSSON, David Heinemeier. *Agile Web Development with Rails. 3rd edition*. United States of America: The Pragmatic Programmers LLC, 2008. 730 s. ISBN 978-1-9343561-6-6.

[10] WELLING, Luke, THOMSON, Laura. *PHP a MySQL: Rozvoj webových aplikací*. Miroslav Klíma; Jan Kulínek. 3. autoriz. vyd. Praha: SoftPress s.r.o., c2005. 830 s., CD-ROM. ISBN 80-86497-83-6.

[11] *MySQL - Wikipedie, otevřená encyklopedie* [online]. 2009 , 3. 8. 2009 [cit. 2009-08-13]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/MySQL>.

[12] *Eclipse (vývojové prostředí) - Wikipedie, otevřená encyklopedie* [online]. 2009 , 8. 8. 2009 [cit. 2009-08-14]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Eclipse_%28v%C3%BDvojov%C3%A9_prost%C5%99ed%C3%AD%29>.

[13] *NetBeans - Wikipedie, otevřená encyklopedie* [online]. 2009 , 12. 7. 2009 [cit. 2009-08-14]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Netbeans>.

[14] *Zend Studio - Zend.com* [online]. c2009 [cit. 2009-08-14]. Dostupný z WWW: <http://www.zend.com/products/studio/>.

[15] *KDevelop - Wikipedie, otevřená encyklopedie* [online]. 2009 , 1. 6. 2009 [cit. 2009-08-14]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/KDevelop>.

[16] *Microsoft Visual Studio - Wikipedie, otevřená encyklopedie* [online]. 2009 , 16. 6. 2009 [cit. 2009-08-14]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Microsoft_Visual_Studio>.

[17] *Zend Framework - Wikipedie, otevřená encyklopedie* [online]. 2009 , 28. 7. 2009 [cit. 2009-08-14]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Zend_Framework>.

[18] *Zend Framework: Documentation* [online]. [2008] [cit. 2009-08-15]. Dostupný z WWW: <http://framework.zend.com/manual/en/coding-standard.naming-conventions.html>.

[19] *Bod zlomu: Dlouhý ocas* [online]. 2007 , 31. 3. 2007 [cit. 2009-08-15]. Dostupný z WWW: <http://bodzlomu.typepad.com/my_weblog/2007/03/chris_anderson_.html>.

[20] ZBIEJCZUK, Adam. *Web 2.0-charakteristika a služby*. Brno, 2007. 71 s. Masarykova Univerzita v Brně, Fakulta sociálních studií. Vedoucí diplomové práce Mgr. David Kořínek.

[21] *Interview with Martin Stiksel of last.fm | Blog | Econsultancy* [online]. 2006 , 8. 11. 2006 [cit. 2009-08-15]. Dostupný z WWW: <http://econsultancy.com/blog/485-interview-with-martin-stiksel-of-last-fm>.

[22] KAGAN, Marta. *What the F**K is Social Media: One Year Later* [online]. 2009 , červenec 2009 [cit. 2009-08-14]. Dostupný z WWW: <http://www.slideshare.net/mzkagan/what-the-fk-is-social-media-one-year-later>.

[23] *Testování softwaru - Wikipedie, otevřená encyklopedie* [online]. 2009 , 12. 8. 2009 [cit. 2009-08-15]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Testov%C3%A1n%C3%AD_softwaru>.

[24] ARLOW, Jim, NEUSTADT, Ila. *UML 2 a unifikovaný proces vývoje aplikací*: Objektově orientovaná analýza a návrh prakticky. Bogdan Kiszka. 2. aktualizované a doplněné vyd. Brno: Computer Press, a.s., 2008. 568 s. ISBN 978-80-251-1503-9.

SEZNAM ZKRATEK

- AJAX – *Asynchronous JavaScript and XML*.
- API – *Application Programming Interface*.
- ASP – *Active Server Pages*. Aktivní serverové stránky.
- ATOM – *Atom Syndication Format*.
- BSD – *Berkeley Software Distribution*. Odvozenina Unixu distribuovaná Kalifornskou univerzitou v Berkeley.
- CASE – *Computer-Aided Software Engineering*.
- CGI – *Common Gateway Interface*. Protokol pro propojení externích aplikací s webovým serverem.
- CIA – *Central Intelligence Agency*. Zpravodajská služba USA.
- CLR – *Common Language Runtime*.
- CMS – *Content Management System*. Systém pro správu obsahu.
- CRC – Class *Responsibility Collaboration*.
- FTP – *File Transfer Protocol*.
- FTPS – *FTP Secure / FTP-SSL*.
- GCC – *GNU Compiler Collection*.
- GNU – *GNU's Not Unix*.
- GNU GPL – *GNU General Public License*.
- HTML – *HyperText Markup Language*. Značkovací jazyk pro hypertext.
- HTTP – *HypeText Transfer Protokol*. Internetový protokol pro výměnu hypertextových dokumentů.
- HTTPS – *HTTP Secure*.
- IDE – *Integrated Development Environment*.
- IIS – Internet *Information Services*.
- IMAP – *Internet Message Access Protocol*. Protokol pro přístup k elektronické poště.
- J2EE – Totéž co Java EE – *Java Enterprise Edition*.
- J2ME – Totéž co Java ME – *Java Micro Edition*.
- JSP – *JavaServer Pages*.
- KDE – *K Desktop Environment*.
- LAMP – *Linux, Apache, MySQL, PHP*.
- MVC – *Model-view-controller*. Model-pohled-řadič.
- PHP – *Hypertext Preprocessor*, původně: *Personal Home Page*.
- POP3 – *Post Office Protocol*. Protokol pro příjem elektronické pošty.
- RCP – *Rich Client Platform*.
- RFC – *Request for comments*. Žádost o komentáře – používá pro označení řady standardů.
- RSS, RSS2 – *Really Simple Syndication* nebo *Rich Site Summary*.
- RUP – *Rational Unified Process*.
- SEO – *Search Engine Optimization*.
- SMTP – *Simple Mail Transfer Protocol*. Protokol pro odesílání elektronické pošty.
- SQL – *Structured Query Language*.

- SSL – *Secure Sockets Layer*.
- NNTP – *Network News Transfer Protocol*.
- T-SQL – *Transact-SQL*.
- TXT – Označuje běžný textový soubor.
- UP – *Unified Process*.
- URL – Uniform Resource Locator.
- USA – *United States of America*. Spojené státy americké.
- WCMS – *Web Content Management Sy*stem. Systém pro správu webového obsahu.
- WYSIWYG – *What You See Is What You Get*. Co je vidět, to je výsledek.
- XML – *eXtensible Markup Language*. Rozšiřitelný značkovací jazyk.

SEZNAM OBRÁZKŮ

Přílohy

Příloha A

Výčet stavových kódů webového serveru.

Příloha B

CD obsahující:

- elektronickou verzi tohoto dokumentu,
- model navrženého WCMS v programu Enterprise Architect,
- zdrojové kódy naprogramované aplikace,
- elektronickou podobu programátorské dokumentace naprogramované aplikace.

Příloha A

Stavové kódy webového serveru jsou vždy třímístná celá čísla, která spadají do následujících skupin:

- **1xx**: Informační – požadavek obdržen, proces pokračuje
- **2xx**: Úspěch – akce byla úspěšně přijata, srozumitelná a akceptována
- **3xx**: Přesměrování – Musí následovat další akce, aby mohl být požadavek dokončen
- **4xx**: Chyba klienta – Požadavek má špaatnou syntaxi nebo nemůže být splněn.
- **5xx**: Chyba serveru – Zdánlivě platný požadavek se nepodařilo vyřídit.

Výčet stavových kódů webového serveru.

- 100 – Continue
- 101 – Switching Protocols
- 200 – OK
- 201 – Created
- 202 – Accepted
- 203 – Non-Authoritative Information
- 204 – No Content
- 205 – Reset Content
- 206 – Partial Content
- 300 – Multiple Choices
- 301 – Moved Permanently
- 302 – Found
- 303 – See Other
- 304 – Not Modified
- 305 – Use Proxy
- 307 – Temporary Redirect
- 400 – Bad Request
- 401 – Unauthorized
- 402 – Payment Required
- 403 – Forbidden
- 404 – Not Found
- 405 – Method Not Allowed
- 406 – Not Acceptable
- 407 – Proxy Authentication Required
- 408 – Request Time-out
- 409 – Conflict
- 410 – Gone
- 411 – Length Required
- 412 – Precondition Failed
- 413 – Request Entity Too Large
- 414 – Request-URI Too Large
- 415 – Unsupported Media Type
- 416 – Requested range not satisfiable
- 417 – Expectation Failed
- 500 – Internal Server Error
- 501 – Not Implemented
- 502 – Bad Gateway
- 503 – Service Unavailable
- 504 – Gateway Time-out
- 505 – HTTP Version not supported