VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DISTRIBUČNÍ SYSTÉM PRO ELEKTRONICKÉ OBCHODY

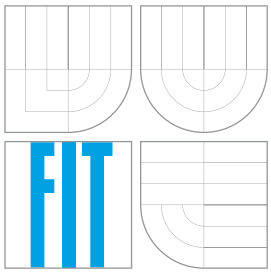
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN GAVENDA

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DISTRIBUČNÍ SYSTÉM PRO ELEKTRONICKÉ OBCHODY

DISTRIBUTION SYSTEM FOR E-SHOPS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN GAVENDA

VEDOUCÍ PRÁCE
SUPERVISOR

Mgr. ZDENĚK MARTÍNEK

BRNO 2008

Abstrakt

Cílem této diplomové práce je navrhnout model informačního systému pro podporu internetových obchodů. Kromě návrhu modelu bude v práci kladen důraz na analýzu možného využití nových technologií, jako je AJAX nebo komunikace pomocí XML. Výsledná práce bude tvořena dvěma aplikacemi, kde první bude celkový katalog zboží a druhou internetový obchod. Pro něj bude hlavní aplikace poskytovat jednotlivé služby.

Klíčová slova

eshop, katalog, podpora, HTML, JavaScript, AJAX, XML, DOM, PHP, MySQL, PostgreSQL

Abstract

The aim of my thesis is to create a model of information system for internet stores support. Together with creating this model I will also analyse the possible use of new technologies such as AJAX or communication with the help of XML. The final work will consist of two applications, where first is a catalogue of products (goods) and the second is an internet store. The main application will provide services for this store.

Keywords

eshop, catalog, support, HTML, JavaScript, AJAX, XML, DOM, PHP, MySQL, PostgreSQL

Citace

Martin Gavenda: Distribuční systém pro elektronické obchody, diplomová práce, Brno, FIT VUT v Brně, 2008

Distribuční systém pro elektronické obchody

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Mgr. Zdeňka Martínka

.....
Martin Gavenda
21. ledna 2008

Poděkování

Rád bych poděkoval panu Mgr. Zdeňku Martínkovi za odbornou pomoc při tvorbě této práce.

Obsah

Část I

Teoretická část

Kapitola 1

Specifikace zadání

1.1 Podrobnější rozbor

Výsledkem práce by měl být model systému podporujícího elektronické obchodování. Systém bude tvořen dvěma aplikacemi.

1.1.1 Serverová aplikace

Tato část systému bude obsahovat kategorizovaný seznam zboží, z něhož bude určité části seznamu poskytovat jednotlivým internetovým obchodům. Aplikace bude obsahovat propracovaný systém uživatelských účtů, na jejichž základě budou mít určití uživatelé přesně specifikované pravomoci. Budou tak odděleny role, jako jsou např. Aministrátor, Product manager nebo Skladník.

Veškeré úkony související se správou systému by měly být implementovány tak, aby umožňovaly co nejrychlejší a nejintuitivnější ovládání.

Aplikace bude poskytovat kromě seznamu nabízeného zboží i vedení objednávek, které budou automaticky zasílány z klientských eShopů. Celý systém by měl mít modulární strukturu a umožňovat další rozšíření.

1.1.2 Klientská aplikace

Hlavní funkcí klientské aplikace bude načtení katalogu zboží ze serveru, jeho zobrazení a možnost vytvoření objednávky. Výsledná interpretace dat a zobrazení zboží bude záviset již pouze na dané implementaci klienta. Klient a server nebudou mít společnou žádnou část.

1.1.3 Komunikace

Jelikož klient ani server nebudou sdílet žádnou část systému jako je např. databáze, bude pro jejich komunikaci užit systém výmeny dokumentů založených na XML. Díky tomuto oddělení se stanout obě části na sobě zcela platformě nezávislé a bude možné vytvářet velké množství implementací dané klientské aplikace.

Kapitola 2

Volba vývojového prostředí

Při volbě vývojového prostředí je důležité uvědomit si, jaký okruh uživatelů bude danou aplikaci používat, v jakém počtu kopií bude nasazena a jak často bude docházet k aktualizacím. U klientské aplikace je zcela zřejmé, že se bude jednat o okruh několika tisíc uživatelů a to teoreticky z celého světa.

Proto je nejlepší použít pro implementaci tenkého klienta, kterým bude komunikovat s aplikací umístěnou na serveru. Díky tomu bude velmi usnadněna aktualizace této aplikace, jelikož se bude nacházet pouze v určitém a předem známém počtu kopií.

U serverové aplikace zvolíme opět tenkého klienta a to z podobného důvodu jako je tomu u klientské aplikace.

2.1 Programovací jazyk

V dnešní době se používá několik možností jak tenkého klienta implementovat. Většinou se jedná o webový prohlížeč, který si neustále načítá aktuální verzi aplikace ze serveru. Toto řešení má hodně výhod, ale také několik nevýhod.

Mezi výhody jistě patří již zmiňovaná aktuálnost dané aplikace, jelikož si klientský prohlížeč při každém http požadavku vyžádá kompletně nové data. My však budeme využívat i technik, které toto tvrzení lehce popírají. Mezi další výhody patří nezávislost na použité platformě, protože se jedná o ustálený standard HTML, který má své prohlížeče implementovány již takřka pro každý možný systém.

Mezi hlavní nevýhody patří možnost rozšíření o periferie. Dost těžko se dá implementovat např. čtečka čárových kódů nebo čipových karet v prostředí HTML. Další velmi podstatnou nevýhodou je velmi špatná podpora ustálených standardů u některých starších prohlížečů a to ať již se jedná o standard HTML, CSS nebo JavaScript.

Pro implementaci skriptů generujících HTML kód jsem zvolil skriptovací jazyk PHP verze 5 a to pro jeho robustnost a hlavně rozšířenost. Mezi další výhody jistě patří i to, že se jedná o OpenSource a že je poskytován zdarma. Ve srovnání např. s konkurečním ASP od společnosti Microsoft nevyžaduje žádný komerční server pro svoji činnost.

2.2 Uložení dat

Pro ukládání dat zvolíme takovou metodu, která umožní jejich perzistentní uložení. K tomu je nejvhodnější použít některý databázový systém. Mezi nejznámější a také nejrozšířenější patří relační databázový systém MySQL, který je pro testovací účely poskytovám zdarma. Jeho výhodou je možnost použít jej na všech nejrozšířenějších platformách a operačních systémech.

Další možností je využití databázového serveru PostgreSQL, který narozdíl od MySQL je již objektově-relační. Je opět poskytován pro testovací účely zdarma a opět jej lze spustit na všech běžných platformách.

Abych mohl porovnat, který systém je pro účely dané aplikace lepší, zvolím pro klientskou aplikaci systém MySQL a pro serverovou část systém PostgreSQL.

2.3 Objektové nebo Imperativní programování

Další téma, na které se zaměřím při analýze aplikací je volba programovací techniky. V zásadě mohu zvolit buď objektově orientovaný přístup nebo starší imperativní programování, často nesprávně zaměňované s procedurálním.

2.3.1 Serverová aplikace

Serverová aplikace bude implementována objektově orientovaným přístupem. Objektově orientovaný přístup vychází z chápání celého systému jako množiny objektů, které se pokoušejí modelovat skutečný svět jedna ku jedné.

Celkový model se skládá z objektů, které mají v každém časovém okamžiku svého života určitý stav, určité vlastnosti a poskytují určité metody. Díky těmto metodám poskytují určité operace, které může daný objekt vykonávat. Každý takový objekt spadá do jisté třídy, která jeho vlastnosti a metody charakterizuje.

Abstrakce u objektů nám umožňuje znázorňovat jednotlivé detaily. Celý objekt pak vystupuje jako černá skříňka, u které není potřeba znát přesnou vnitřní strukturu, ale pouze jisté komunikační rozhraní, které objekt poskytuje.

Objekt je vzhledem k ostatním objektům zapouzdřen a chráněn, aby ostatní objekty nemohly tyto hodnoty přímo měnit. Tyto operace se pak dějí jen poskytovanými metodami.

Třídy objektů je možné řadit do stromové struktury, která nám umožňuje dědit vlastnosti objektů z jejich předků ve struktuře. K této vlastnosti se váže také polymorfismus, který dovoluje přetěžovat funkce implementované u předků novými metodami.

2.3.2 Klientská aplikace

Jelikož hodně programátorů zavrhuje objekty u PHP, vytvořím druhou část tradičním způsobem. Tato metoda je založena na procedurálním programování. Celkový program je podobný např. kuchyňskému receptu, kdy je přesně charakterizován postup v jednotlivých krocích.

Imperativní programování lze rozdělit do tří skupin: Naivní paradigma, Nestrukturované paradigma a Strukturované paradigma. Jazyk PHP využívá poslední - strukturované. To

je založeno na využívání cyklů a podmínek, nikoliv skoků jako tomu je u naivního nebo nestrukturovaného.

Základními prvky takovýchto programů je přiřazení, cyklení a podmíněné dělení toku programu.

2.4 Aplikační servery

Jelikož se jedná o aplikace tvořené tenkými klienty, je k jejich běhu zapotřebí server. K tomu nám poslouží opět volně šiřitelný Apache. Tento server již ve své základní instalaci obsahuje podporu pro běh PHP i napojení na oba databázové servery.

Kapitola 3

Použité technologie a standardy

3.1 Apache

Apache HTTP Server je softwarový webový server s otevřeným kódem pro Linux, BSD, Microsoft Windows a další platformy. V současné době dodává prohlížečům na celém světě většinu internetových stránek.[2]

Celý produkt je šířen pod OpenSource licencí, která dovoluje jeho volné šíření, instalaci a užívání v nezměněné formě. Veškeré změny v něm provedené musí být vyznačeny ve zdrojovém kódu se jménem autora změn.

Apache poskytuje podporu pro nejrůznější technologie, pro které má implementovány jednotlivé knihovny. Tyto knihovny jsou implementovány v modulech, jejichž konfigurace se provádí pomocí konfiguračního souboru serveru. Pro naše účely bude kromě standardních potřeba modul pro interpretaci skriptů PHP.

3.2 MySQL

MySQL je databázový systém, vytvořený švédskou firmou MySQL AB. Jeho hlavními autory jsou Michael „Monty“ Widenius a David Axmark. Je považován za úspěšného průkopníka dvojího licencování – je k dispozici jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí.

MySQL je multiplatformní databáze. Komunikace s ní probíhá – jak už název napovídá – pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními.

Pro svou snadnou implementovatelnost (lze jej instalovat na Linux, MS Windows, ale i další operační systémy), výkon a především díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl na v současné době používaných databázích. Velmi oblíbená a často nasazovaná je kombinace MySQL, PHP a Apache jako základní software webového serveru.

MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení: má jen jednoduché způsoby zálohování, a až donedávna nepodporovalo pohledy, triggery, a uložené procedury. Tyto vlastnosti jsou doplňovány teprve v posledních letech, kdy začaly nejčastějším uživatelům produktu – programátorům webových stránek – již poněkud scházet.[9]

Pro administraci databáze využiji aplikaci phpMyAdmin.

phpMyAdmin je nástroj napsaný v jazyce PHP umožňující jednoduchou správu obsahu databáze MySQL prostřednictvím webového rozhraní. V současné době umožňuje vytvářet/rušit databáze, vytvářet/upravovat/rušit tabulky, provádět SQL příkazy a spravovat klíče. Jedná se o jeden z nejpopulárnějších nástrojů pro správu databáze. Je k dispozici v 52 jazycích.[11]

3.3 PostgreSQL

PostgreSQL je plnohodnotným relačním databázovým systémem s otevřeným zdrojovým kódem. Má za sebou více než patnáct let aktivního vývoje a má vynikající pověst pro svou spolehlivost a bezpečnost. Běží na všech rozšířených operačních systémech včetně Linuxu, UNIXů (AIX, BSD, HP-UX, SGI-IRIX, Mac OS X, Solaris, Tru64) a Windows. Stoprocentně splňuje podmínky ACID[1], plně podporuje cizí klíče, operace JOIN, pohledy, spouštěné a uložené procedury. Obsahuje většinu SQL92 a SQL99 datových typů, např. `INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL a TIMESTAMP`. K systému existuje kvalitní volně dostupná dokumentace včetně českých překladů FAQ a FAQ pro o.s. fy. Microsoft.

PostgreSQL je šířen pod BSD licencí, která je nejliberálnější ze všech open source licencí. Tato licence umožňuje neomezené používání, modifikaci a distribuci PostgreSQL. PostgreSQL je možno šířit se zdrojovými kódy nebo bez nich, zdarma nebo komerčně.

PostgreSQL umožňuje běh uložených procedur napsaných v několika programovacích jazycích, v Perlu, v Pyhon, v jazyku C nebo ve speciálním PL/pgSQL – jazyku vycházejícím z PL/SQL fy. Oracle. Existují PostgreSQL varianty JDBC, ODBC, dbExpress, Open Office, PHP, .NET Perl nativních rozhraní. K PostgreSQL existuje překladač Embedded SQL pro C a C++.

Předností systému PostgreSQL je rozšiřitelnost. Systém může být bezproblémově rozšiřován o nové datové typy, funkce operátory, agregační funkce, procedurální jazyky. Díky tomu mohly vzniknout následující rozšíření: PostGIS – podpora pro geografické informační systémy, TSearch2- podpora fulltextového vyhledávání, Slony-I – master to multiple slaves replikace. Na serveru pgfoundry je k dispozici několik desítek doplňků včetně doplňků rozšiřující o funkcionalitu MySQL, SQL Serveru a Oraclu.[12]

Pro práci s databází PostgreSQL využiji program pgAdmin III společnosti The pgAdmin Development Team. Software je možné získat na webové adrese

`http://www.pgadmin.org/download/`

.

3.4 DOM

DOM (akronym anglického Document Object Model – objektový model dokumentu) je objektově orientovaná reprezentace XML nebo HTML dokumentu. DOM je API umožňující přístup či modifikaci obsahu, struktury, nebo stylu dokumentu, či jeho částí.

[1] ACID – v tomto případě zkratka ACID (Atomičnost – Atomicity, Konzistence – Consistency, Izolovanost – Isolation, Trvalost – Durability) charakterizuje 4 základní vlastnosti databáze, které garantují, že jsou databázové transakce prováděny spolehlivě.

Původně měl každý webový prohlížeč své vlastní specifické rozhraní k manipulaci s HTML elementy pomocí JavaScriptu. Vzájemná nekompatibilita těchto rozhraní však přivedla konsorcium W3C[2] k myšlence standardizace, a tak vznikl W3C Document Object Model (zkráceně W3C DOM). Tato specifikace je platformně a jazykově nezávislá. Předchozí specifická rozhraní byla nazvána Intermediate DOM (anglicky přechodný DOM).

DOM umožňuje přístup k dokumentu jako ke stromu, což je zároveň datová struktura používaná ve většině XML parserů (Xerces, MSXML) a XSL procesorů (Xalan). Tato technologie, nazývaná grove (Graph Representation Of property ValuEs), vyžaduje nahrání celého parsovaného dokumentu do paměti, z čehož plyne, že její optimální použití je tam, kde je k jednotlivým elementům dokumentu přistupováno v náhodném pořadí nebo opakovaně. Existuje i alternativní technologie pro případ, že je potřeba postupná, nebo jednorázová úprava – sekvenční model SAX, který má v těchto případech výhodu rychlejšího zpracování a nižší paměťové náročnosti.[5]

3.5 PHP

PHP (rekurzivní zkratka PHP: Hypertext Preprocessor, „PHP: Hypertextový preprocesor“, původně Personal Home Page) je skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek. Nejčastěji se začleňuje přímo do struktury jazyka HTML, XHTML či WML, což je velmi výhodné pro tvorbu webových aplikací. PHP lze ovšem také použít i k tvorbě konzolových a desktopových aplikací.

PHP skripty jsou prováděny na straně serveru, k uživateli je přenášen až výsledek jejich činnosti. Syntaxe jazyka kombinuje hned několik programovacích jazyků (Perl, C, Pascal a Java). PHP je nezávislý na platformě, skripty fungují bez úprav na mnoha různých operačních systémech. Obsahuje rozsáhlé knihovny funkcí pro zpracování textu, grafiky, práci se soubory, přístup k většině databázových serverů (mj. MySQL, ODBC, Oracle, PostgreSQL, MSSQL), podporu celé řady internetových protokolů (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP, . . .)

PHP se stalo velmi oblíbeným především díky jednoduchosti použití a tomu, že kombinuje vlastnosti více programovacích jazyků a nechává tak vývojáři částečnou svobodu v syntaxi. V kombinaci s databázovým serverem (především s MySQL nebo PostgreSQL) a webovým serverem Apache je často využíván k tvorbě webových aplikací. Díky velmi častému nasazení na serverch se vžila zkratka LAMP – tedy spojení Linux, Apache, MySQL a PHP nebo Perl.

S verzí PHP 5 se výrazně zlepšil přístup k objektově orientovanému programování podobný Javě. [10]

3.5.1 Požadované moduly

Samotné PHP obsahuje velkou řadu funkcí. Moje aplikace bude využívat kromě těch standardních také knihovnu pro MySQL, PostgreSQL a pro konverzi kódové stránky řetězců.

[2] World Wide Web Consortium (W3C) je mezinárodní konsorcium jehož členové společně s veřejností vyvíjejí webové standardy pro World Wide Web. Cílem konsorcia je „Rozvíjet World Wide Web do jeho plného potenciálu vývojem protokolů a směrnic, které zajistí dlouhodobý růst Webu“. W3C se také zabývá vzděláním a přístupností, vyvíjí software a nabízí otevřenou diskuzi o Webu prostřednictvím fóra.[13]

php_mysql.dll

V zásadě budu používat jen základní funkce této knihovny. První z nich je *mysql_connect*, která slouží pro připojení k databázovému serveru. Obvykle bývá volána se třemi parametry – adresa serveru, jméno uživatele, heslo uživatele. Její návratová hodnota je ukazatel na spojení.

Po navázání spojení se serverem je třeba zvolit databázi, se kterou se bude pracovat. Toto se provádí pomocí příkazu *mysql_select_db*, která má jeden povinný parametr, kterým je jméno databáze, a jeden nepovinný, kterým je ukazatel na spojení. Při úspěchu vrací `True` jinak `False`.

Pro vykonávání SQL dotazů slouží funkce *mysql_query*, která provede dotaz pomocí aktuálního spojení. Jako parametr se zadává SQL dotaz a výsledek je vrácen pomocí ukazatele.

Pro procházení výsledku dotazu se nejčastěji volá funkce *mysql_fetch_array*, která prochází jednotlivé řádky výsledku a vrací je jako pole. Jako parametr se zadává ukazatel na výsledek. Jednotlivé položky pole jsou indexovány pořadovými čísly sloupců a také názvy sloupců výsledku.

Při vkládání do tabulek pomocí SQL příkazu `INSERT` je vhodné znát primární klíč, který byl danému řádku přiřazen. Tuto hodnotu zjistí funkce *mysql_insert_id*. Má jeden nepovinný parametr, kterým je ukazatel na spojení se serverem.

php_pgsql.php

Knihovna pro práci s databází PostgreSQL v prostředí PHP má opět velkou spoustu funkcí. Já však budu využívat jen ty základní. Od databáze MySQL se liší hlavně vytvářením jednoznačného identifikátoru `OID`, který jednoznačně charakterizuje jednotlivé záznamy v celém kontextu působnosti databáze.

Pro navázání spojení se serverem se používá funkce `pg_connect`, která má jeden povinný parametr `connection_string` a jeden nepovinný, kterým je `connect_type`. Textový řetězec `connection_string` je složen z `host`, `hostaddr`, `port`, `dbname`, `user`, `password`, `connect_timeout`, `options`, `tty`, `sslmode`, `requiressl` a `service`. Ne všechny jsou povinné a jejich nastavení záleží na dané instalaci. Návratovou hodnotou funkce je ukazatel na spojení.

Pro odeslání dotazu slouží funkce `pg_query`, která má jako povinné parametry ukazatel na spojeni a SQL dotaz. Návratová hodnota je ukazatel na výsledek.

Pro procházení výsledkem používám funkci `pg_fetch_array`, která navrací výsledky jako pole. Její první povinný parametr je ukazatel na výsledek. Druhý parametr je nepovinný a označuje o kolikátý řádek máme zájem. Třetím parametrem, který je také nepovinný, dovoluje určit, zda se mají prvky v návratovém poli indexovat čísly (`PGSQL_NUM`), jmény sloupců (`PGSQL_ASSOC`) nebo obojím (`PGSQL_BOTH`).

Stejně jako u MySQL i zde budu chtít zjistit označení naposledy vloženého prvku. U této knihovny je implementována funkce, která vrátí nikoliv primární klíč naposledy vloženého záznamu, ale jeho `OID`. Jde o funkci `pg_last_oid`) a jako parametr se vkládá výsledek naposledy vykonaného příkazu funkcí `pg_query`.

DOM API v PHP 5

Pro práci s DOM bylo v PHP verze 5 vytvořeno API, které nahrazuje používání knihovny DOM_XML z PHP verze 4. Toto API obsahuje několik desítek funkcí, které zaručují velkou výkonnost spojení PHP a DOM. Pro moji aplikaci bude stačit opět několik základních.

Jelikož se již jedná o plnohodnotnou třídu, pro vytvoření nového DOM objektu použiji standardní konstruktor `DOMDocument`. Při jeho volání mohu uvést dva nepovinné parametry pro určení verze DOM a pro určení kódování.

Dalším objektem implementovaným v PHP je objekt elementu DOM. K jeho vytvoření slouží konstruktor `DOMElement`, který má jeden povinný parametr jméno elementu, a dále nepovinně hodnotu a `Namespace` platnosti elementu.

Pro vytvoření nového atributu, který se chová opět jako objekt, slouží konstruktor třídy `DOMAttr`. Jako povinný parametr se uvádí jméno nového atributu a dále můžeme jako nepovinný parametr uvést i hodnotu tohoto nového atributu.

Pro spojování elementů do stromu existují metody objektů `DOMDocument` a `DOMElement` `appendChild`, která má jako parametr ukazatel na připojovaný element.

K připojení atributu k elementu slouží metoda `setAttributeNode`, které předáme ukazatel na přidávanou vlastnost jako povinný parametr.

Pro výpis struktury vzniklého objektu můžeme použít několik různých metod objektu `DOMDocument`, které se liší formátem výstupu. Mezi nejpoužívanější patří `saveXML`, která navrací výsledný dokument jako řetězec.

Pro načtení již existujícího XML dokumentu slouží metoda objektu `DOMDocument` `load`, která má jako parametr cestu k danému souboru. Může se jednat o soubor na disku nebo soubor přístupný pomocí URL.

```
$dom = new DOMDocument('1.0', 'iso-8859-1');
$element = $dom->appendChild(new DOMElement('root'));
$attr = $element->setAttributeNode(new DOMAttr('attr', 'attrvalue'));
echo $dom->saveXML();
```

3.6 XHTML

XHTML (zkratka anglického extensible hypertext markup language – „rozšiřitelný hypertextový značkovací jazyk“) je značkovací jazyk pro tvorbu hypertextových dokumentů v prostředí WWW vyvinutý konsorciem W3C. Původně se předpokládalo, že se stane nástupcem jazyka HTML, jehož vývoj byl verzí 4.01 ukončen. V roce 2007 však došlo k založení pracovní skupiny, která má za cíl vytvořit novou verzi HTML, která ponese označení HTML 5.0. XHTML je stále paralelně vyvíjeno a nyní se pracuje na verzi 2.0.[14]

3.6.1 CSS

CSS je zkratka pro anglický název Cascading Style Sheets, česky tabulky kaskádových stylů. Je to jazyk pro popis způsobu zobrazení stránek napsaných v jazycích HTML, XHTML nebo XML.

Jazyk byl navržen standardizační organizací W3C. Byly vydány zatím dvě verze specifikace CSS1 a CSS2 (plus CSS 2.1), pracuje se na verzi CSS3.

Hlavním smyslem je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Původně to měl umožnit už jazyk HTML, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak. Starší verze HTML obsahují celou řadu elementů, které nepopisují obsah a strukturu dokumentu, ale i způsob jeho zobrazení. Z hlediska zpracování dokumentů a vyhledávání informací není takový vývoj žádoucí.

Hlavní nevýhodou CSS je zatím stále špatná podpora v majoritních prohlížečích. Různé prohlížeče interpretují stejný CSS kód jinak a je někdy velmi obtížné jej napsat tak, aby se na všech (resp. na několika vybraných) prohlížečích výsledek zobrazil stejně. Situace se ale v roce 2006 značně zlepšuje, v souvislosti s tím se s napětím očekával příchod Internet Exploreru 7, který by měl postupně vytlačit svého předchůdce IE 6, který byl častým zdrojem problémů. Nicméně ani IE 7 se striktně nedrží definice CSS 2.1.[4]

Pro vyřešení chybné interpertace CSS v různých prohlížečích, zejména problému s box-modelem, lze využít několik různých řešení, které využívají dalších nedokonalostí interpretace. Občas se používá vkládání „podtžítka“ do označení vlastnosti což však není v souladu s normou CSS.

Lepší řešení je specifikovat zanoření elementu pomocí dvou označení a mezi ně vložit znak „>“. Prohlížeč IE6 neumí tuto vlastnost správně vyhodnotit, a umožňuje nám tak specifikovat vlastnosti, které se použíjí jen prohlížeči typu FireFox nebo Opera.

```
div .prvek{color: red;}
div>.prvek{color: black;}
```

Výsledkem je, že v prohlížečích založených na IE 6 bude barva červená, v ostatních černá.

3.6.2 JavaScript

JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk, jehož autorem je Brendan Eich z tehdejší společnosti Netscape.

Nyní se zpravidla používá jako interpretovaný programovací jazyk pro WWW stránky, často vkládaný přímo do HTML kódu stránky. Jsou jím obvykle ovládány různé interaktivní prvky GUI (tlačítka, textová políčka) nebo tvořeny animace a efekty obrázků.

Jeho syntaxe patří do rodiny jazyků C/C++/Java. Slovo Java je však součástí jeho názvu pouze z marketingových důvodů a s programovacím jazykem Java jej vedle názvu spojuje jen podobná syntaxe. JavaScript byl v červenci 1997 standardizován asociací ECMA (Europen Computer Manufacturers Association) a v srpnu 1998 ISO (International Standards Organization). Standardizovaná verze JavaScriptu je pojmenována jako ECMAScript a z ní byly odvozeny i další implementace, jako je například ActionScript.

JavaScript byl původně obchodní název implementace společnosti Netscape, kde byl vyvíjen nejprve pod názvem Mocha, později LiveScript, ohlášen byl společně se společností Sun Microsystems v prosinci 1995 jako doplněk k jazykům HTML a Java. Pro verzi firmy Microsoft je použit název JScript.

Program v JavaScriptu se obvykle spouští až po stažení WWW stránky z Internetu (tzv. na straně klienta), na rozdíl od ostatních jiných interpretovaných programovacích jazyků (např. PHP a ASP), které se spouštějí na straně serveru ještě před stažením z Internetu. Z toho plynou jistá bezpečností omezení, JavaScript např. nemůže pracovat se soubory, aby tím neohrozil soukromí uživatele.

JavaScript je možné použít i na straně serveru. První implementací JavaScriptu na straně serveru byl LiveWire firmy Netscape vypuštěný roku 1996, dnes existuje několik možností včetně opensource implementace Rhinola založená na Rhino, gcj a Apache.[7]

Pro práci s DOM je v JavaScriptu implementováno několik základních funkcí. Celý dokument se skládá z jednotlivých elementů, které mají určité vlastnosti a metody. K odkázání na urřitý element se použije buď klíčové slovo `this`, které se vkládá do volání funkce jako parametr a odkazuje na aktuální element, nebo je třaba použít hiearchii DOM stromu. V následující části uvedu nejpoužívanější vlastnosti a metody JavaScriptu.

DOM Document Object properties

body Tato vlastnost odkazuje na element BODY HTML dokumentu.

DOM Document Object Methods

createAttribute("attributename") Tato metoda vytvoří novou vlastnost aktuálního dokumentu. Jako návratová hodnota je odkaz na novou vlastnost.

createElement("tagName") Metoda vytvoří nový element. Typ nového elementu se uvádí jako parametr. Návratová hodnota je odkaz na tento nově vytvořený element.

createTextNode(text) Metoda vytvoří nový textový prvek, který lze zařadit do výsledného DOM stromu. Návratová hodnota je odkaz na nově vytvořený prvek.

getElementById(id) Metoda slouží k přístupu k libovolnému elementu v DOM na základě uvedeného ID.

getElementsByTagName(tagname) Vrátí všechny elementy z dokumentu daného typu.

DOM Element properties

attributes[] Vrátí všechny vlastnosti daného elementu jako pole. IE6 vrací nejen vlastnosti definované autorem, ale také elementy definované v DTD[3]. Ve FireFoxu se navrací jen vlastnosti definované uživatelem, případně změněné pomocí skriptu.

childNodes[] Navrátí všechny synovské prvky. Jsou vráceny jako objekty a seřazeny v poli. Další vlastnosti lze získat pomocí atributů `NodeName` a `NodeType`.

className Pomocí této vlastnosti můžeme číst nebo měnit aktuální `class` daného elementu. Lze tak měnit vizuální vlastnosti definované pomocí CSS.

[3]DTD (Document Type Definition, česky Definice typu dokumentu) je jazyk pro popis struktury XML případně SGML dokumentu. Omezuje množinu přípustných dokumentů spadajících do daného typu nebo třídy. DTD tak například vymezuje jazyky HTML a XHTML. Struktura třídy nebo typu dokumentu je v DTD popsána pomocí popisu jednotlivých značek (nebo též elementů) a atributů. Popisuje jak mohou být značky navzájem uspořádány a vnořeny. Vymezuje atributy pro každou značku a typ těchto atributů.[6]

id Umožňuje číst, případně změnit ID daného elementu.

nodeName Navrátí typ elementu jako je například `DIV` nebo `P`.

nodeValue Navrátí hodnotu daného elementu. Pro textové prvky navrátí text daného prvku, pro vlastnosti navrátí hodnotu dané vlastnosti. Pomocí této vlastnosti lze měnit hodnoty prvků.

parentNode Jako návratová hodnota je předán ukazatel na nadřazený prvek v DOM hiearchii.

DOM Element methods

addEventListener(eventType, listener, useCapture) Asociuje funkci k události JavaScriptu pro aktuální prvek. Tato metoda nefunguje v prohlížečích Internet Explorer. Jako první parametr se uvádí typ události, který má stejné označení jako v JavaScriptu avšak bez počátečního `on` (např. `click`, `focus`, `change`). Jako druhý parametr se uvádí jméno funkce, která bude volána. Zde nesmí být uvedeny žádné parametry ani závorky. Tímto postupem lze k jednomu prvku a jedné události přiřadit více funkcí.

attachEvent(eventType, function) Tato metoda je obdobou metody `addEventListener` pro prohlížeče Internet Explorer. Zde se však uvádí celé jméno události včetně `on`.

```
if (window.addEventListener)
  window.addEventListener("load", statusreport, false)
if (window.attachEvent)
  window.attachEvent("onload", statusreport)
```

appendChild(node) Připojí element předaný parametrem na konec seznamu synovských prvků.

detachEvent(eventType, function) Odstraní funkci ze seznamu určité události. Je určena pro prohlížeče Internet Explorer.

getAttribute(attributeName) Vrátí hodnotu daného atributu pro aktuální prvek.

getElementsByTagName(tagName) Vrátí pole ukazatelů na synovské objekty daného typu.

removeAttribute(attributename) Odstraní atribut daného jména z aktuálního elementu.

removeChild(childreference) Odstraní synovský prvek zadaný referencí.

removeEventListener(eventType, listener, useCapture) Odstraní funkci ze seznamu dané události. Je určena pro prohlížeče Firefox a podobné. Obdoba funkce `detachEvent` u IE.

replaceChild(newChild, oldChild) Přepíše v aktuálním prvku synovský prvek novým. Oba se zadávají referencí.

setAttribute(attributename, value, [iecaseflag]) Slouží pro nastavování parametrů aktuálního prvku. Třetí parametr zapíná a vypína rozpoznávání velikosti písmen pro prohlížeče Internet Explorer, které rozeznávají rozdíl např. mezi `align` a `Align`.

3.6.3 AJAX

AJAX (Asynchronous JavaScript and XML) je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů.

Podobně jako DHTML, LAMP nebo SPA, Ajax ve skutečnosti není konkrétní jednotlivá technologie, ale pojem označující použití několika technologií dohromady s určitým cílem.

Mezi výhody patří odstranění nutnosti znovunačtení a překreslení celé stránky při každé operaci, které jsou nutné u klasického modelu WWW stránek. Pokud například uživatel klikne na tlačítko pro udělení hlasu v nějaké anketě, celá stránka se musí znovu načíst ze serveru, třebaže se na ní jen například aktualizují výsledky hlasování a veškerý zbytek obsahu zůstává stejný. Prostřednictvím AJAXu proběhne odeslání hlasu uživatele na pozadí, server zašle jen ty části stránky, které se změnily, a jen tyto části se uživateli na stránce aktualizují a překreslí. Uživatel tak má pocit mnohem větší plynulosti práce, která se blíží běžným desktopovým aplikacím.

Z toho vyplývá také potenciál snížit zátěž na webové servery a síť obecně. Jelikož není potřeba při každém požadavku sestavit celý HTML dokument, ale pouze provedené změny, je množství vyměňovaných dat výrazně nižší a teoreticky to může mít příznivý vliv i na zátěž databázových serverů či dalších backendových systémů. AJAX však naopak může zvýšit počet vyměňovaných HTTP požadavků, třebaže přenášejí nižší množství dat tak při nevhodné implementaci zátěž neklesne.

Mezi nevýhody patří hlavně změny v paradigmatu používání webu: webové stránky se chovají jako plnohodnotná aplikace se složitou vnitřní logikou, nikoli jako posloupnost stránek, mezi kterými se lze navigovat i pomocí tlačítek Zpět a Další. Moderní AJAXové aplikace jsou schopny funkce těchto tlačítek (přinejmenším částečně) obnovit za použití různých technik (např. využití části adresy za znakem `#` či pomocí neviditelných IFRAMEs).

Problémem AJAXových aplikací také může být síťová latence: potřeba komunikace přes Internet má negativní dopady na rychlost odezvy a interaktivitu uživatelského rozhraní. Pokud uživateli není jasně signalizováno, že aplikace zpracovává jeho požadavek (a na pozadí komunikuje se serverem), jediné, co zaregistruje, je zpožděná reakce (mezitím se dokonce může snažit operaci spustit znovu, neboť se domnívá, že systém jeho příkaz ignoroval).

Další nevýhodou AJAXu je nutnost používat moderní grafické prohlížeče, které podporují potřebné technologie. (Všechny dnešní běžné prohlížeče však tyto technologie alespoň v základu podporují.)[3]

Objekt XMLHttpRequest

Historie AJAXu sahá již k prohlížeči Internet Explorer 5, uvedeného na jaře roku 1999, kde byl uveden jako ovládací prvek Active X. Původně byl podporován pouze Internet Explorerem, ale s příchodem Mozilly 1.0 a Safari 1.2 se podpora rozšířila.

Již od prvopočátku je AJAX založen na objektu XMLHttpRequest, který obstarává komunikaci se serverem. Abychom jej mohli využívat, je třeba nejdříve vytvořit jeho instanci pomocí JavaScriptového kódu. Jelikož tento objekt není prozatím standardizován, jeho vytvoření se v jednotlivých prohlížečích různí. Prohlížeče Internet Explorer jej implementují jako prvek Active X, jiné prohlížeče jako objekt JavaScriptu.

Detekce toho, zda budeme vytvářet jedním, či druhým způsobem je velmi triviální a je založena na detekci, zda lze vytvářet objekty Active X.

```
var xmlHttp;
function vytvorXMLHttprequest(){
  if (window.ActiveObject){
    xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
  }
  else if (window.XMLHttpReuest){
    xmlHttp = new XMLHttpRequest();
  }
}
```

Metody a hodnoty objektu XMLHttpRequest

abort() Tato metoda přeruší aktuální požadavek.

getAllResponseHeaders() Vrátí všechny hlavičky HTTP požadavku ve formě párů klíč/hodnota jako jsou např. `Content-Length`, `Date` nebo `URL`.

getResponseHeader() Tato metoda je podobná metodě getAllResponseHeaders pouze s tím rozdílem, že bere argument specifikující konkrétní hlavičku o kterou máme zájem. Tuto hodnotu vrací jako řetězec.

open() Tato metoda nastavuje parametry volání serveru. Jako jediná je určena pro inicializaci požadavku. Má dva povinné a tři nepovinné argumenty. Je nutné zadat požadovanou metodu přenosu (`GET`, `POST` nebo `PUT`) a `URL` prostředku, který voláme. Volitelně je možné zadat také argument typu Boolean specifikující, jestli má být volání asynchronní – výchozí hodnota je `true`, tzn. asynchronní požadavek. Pokud zadáme `false`, čeká se na odpověď serveru a až poté se pokračuje ve zpracování. Vzhledem k tomu, že asynchronní volání je jedním z hlavních přínosů AJAXu, nastavení tohoto argumentu na `false` ztrácí použití objektu XMLHttpRequest smysl. Za určitých okolností, jako je např. ověření uživatelského vstupu před pokračováním dále, to však může být užitečné. Poslední dva argumenty umožňují zadat uživatelské jméno a heslo.

send() Tato metoda provede vlastní požadavek na server. Pokud byl požadavek deklarován jako asynchronní, vrací tato metoda řízení okamžitě, v opačném případě čeká

na příchod odpovědi od serveru. Nepovinný argument může být instance objektu DOM, vstupní tok nebo řetězec. Argument předávaný metodě je odeslán jako část těla požadavku.

setRequestHeader() Tato metoda nastavuje hodnotu specifikované hlavičky požadavku. Jako argumenty bere řetězec reprezentující název požadované hlavičky a řetězec reprezentující hodnotu, na kterou má být hlavička nastavena. Tato metoda může být volána až po zavolání metody `open()`.[1]

Kromě techto metod obsahuje objekt XMLHttpRequest také několik standardních atributů.

onreadystatechange Ukazatel na obslužný kód, který je spuštěn při každé změně interního stavu objektu, typicky se jedná o funkci v JavaScriptu.

readyState Stav požadavku. Pět možných hodnot je: 0 – neinicializováno, 1 – načítání, 2 – načteno, 3 – interaktivní, 4 – dokončeno.

responseText Odpověd serveru ve formě řetězce.

responseXML Odpověď serveru ve formě XML. Tento objekt může být dále zpracován a zkoumán jako DOM objekt.

status Stavový kód získaný od serveru.

statusText Textová verze stavového kódu (`OK`, `Not Found` atd.).[1]

Část II

Implementační část

Kapitola 4

První iterace

4.1 Požadovaný výsledek

Výsledkem první iterace bude základ serverové aplikace, který umožní modulární architekturu. Umožní přihlášení uživatelů, jejich administraci a také umožní nastavení přístupových práv k jednotlivým modulům. V této iteraci bude také navržen vzhled celé aplikace a budou definovány základní grafické prvky.

4.1.1 Základní obrazovka

Jako základní obrazovka pro nepřihlášeného uživatele bude jednoduchý formulář obsahující pole pro vložení uživatelského jména a hesla a tlačítko pro přihlášení do systému. Pokud bude uživatel úspěšně přihlášen, zobrazí se mu již seznam všech možných aplikací, které jsou pro jeho uživatelský účet zpřístupněny.

Seznam přístupných aplikací je zobrazen v levé části obrazovky a jednotlivé moduly jsou seskupovány do oken, které tvoří logické celky (např. okno pro práci s uživatelskými účty obsahuje část pro výpis, editaci atd.). Jelikož může těchto oken být velké množství, bude možné je „sbalit“ a tím zmenšit prostor, který zabírají. Informace o aktuálním rozložení by měla být perzistentně uložena k danému uživatelskému účtu.

4.1.2 Správa uživatelů

Tato část systému umožní správu uživatelských účtů. Bude sestávat z modulů pro zobrazení seznamu uživatelů a jejich editaci, modulu pro přidání nového uživatele, seznamu uživatelských skupin, do kterých je možno jednotlivé uživatele řadit, modulu pro tvorbu skupin a modulu pro nastavování přístupových práv k jednotlivým modulům.

Po kliknutí na položku Seznam uživatelů se zobrazí ve středovém sloupci seznam již vytvořených uživatelů. Po zvolení některého z nich se zobrazí v pravé části informace o daném uživatelském účtu. Tyto políčka bude možné jednoduchým kliknutím zpřístupnit a měnit informace v nich obsažené. Po dokončení změny se tato informace okamžitě uloží do databáze.

Pod výpisem základních dat bude uveden seznam skupin, do kterých je možné uživatele zařadit. Změna členství ve skupině se provede jednoduchým kliknutím na ikonu aktuálního stavu. Uživatelské skupiny umožní lepší administraci přístupových práv k jednotlivým

částem systému. Pod seznamem skupin se bude nachát možnost deaktivovat daný uživatelský účet.

4.1.3 Vytvoření nového uživatele

Po zvolení této funkce se v pravé části zobrazí formulář, který umožní zadat základní informace o uživateli a jejich odeslání. Po odeslání se v dalším kroku vloží heslo pro daný uživatelský účet a v poslední části se nastaví jednotlivá členství ve skupinách. Pokud nebude některá položka při vytváření správně vyplněna, bude uživatel o tom informován a bez správného zadání nebude možné pokračovat v dalším kroku.

4.1.4 Seznam skupin

Po kliknutí na tuto položku se ve středovém sloupci zobrazí seznam již vytvořených skupin. Po zvolení některé se v pravé části zobrazí okno, ve kterém bude možno editovat její jméno.

4.1.5 Vytvoření nové skupiny

V této sekci je možné vytvořit novou skupinu zadáním jejího jména.

4.1.6 Nastavení přístupových práv

Po zvolení možnosti nastavení přístupových práv se ve středovém sloupci zobrazí dvě okna. V prvním je uveden seznam uživatelů, ve druhém seznam skupin. Po zvolení některého záznamu, ať již jména uživatele nebo jména skupiny, se v pravém sloupci zobrazí seznam všech modulů a jeho částí. Jednoduchým kliknutím bude možné aktuální stav změnit.

4.2 Specifikace případů použití

Případ použití:	Přihlášení do systému
ID	1
Stručný popis	Uživatel se pomocí formuláře přihlásí do systému
Primární aktéři	Uživatel
Sekundární aktéři	nejsou
Předpoklady	Žádný uživatel není přihlášen
Hlavní tok	1. Zobrazí se přihlašovací formulář 2. Uživatel vloží přihlašovací jméno 3. Uživatel vloží přihlašovací heslo 4. Uživatel klikne na tlačítko Přihlásit
Následné podmínky	Uživatel s daným uživatelským jménem a heslem byl nalezen v databázi a byl přihlášen. Zobrazí se úvodní strana aplikace se seznamem modulů
Alternativní tok	Zobrazí se informace o neúspěšném přihlášení. Přihlašovací formulář se zobrazí znovu.

Případ použití:	Seznam uživatelů a jejich editace
ID	2
Stručný popis	Uživatel pomocí modulu zobrazí a edituje informace o uživatelích
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	Ostatní uživatelé
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel vybere jiného uživatele ze seznamu 2. Zobrazí se seznam údajů vybraného uživatele 3. Uživatel může měnit zobrazené údaje, změny se okamžite ukládají v databázi
Následné podmínky	nejsou
Alternativní tok	není

Případ použití:	Vytvoření nového uživatele
ID	3
Stručný popis	Uživatel vytvoří ve 3 krocích nového uživatele
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	nejsou
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel vyplní zobrazený formulář obsahující základní informace o novém uživateli 2. Odešle jej pomocí tlačítka 3. Uživatel v dalším formuláři vyplní heslo k novému uživatelskému účtu 4. Uživatel v posledním kroku nastaví, do kterých skupin nový účet patří
Následné podmínky	Nový účet je vytvořen
Alternativní tok	Pokud nebyl některý krok zcela splněn, nelze přejít ke kroku následujícímu. Uživatel je o tom informován

Případ použití:	Zobrazení a editace uživatelské skupiny
ID	4
Stručný popis	Uživatel může zobrazit a měnit informace o uživatelské skupině
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	nejsou
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel vybere některou ze seznamu skupin 2. Uživatel může měnit zobrazené informace o skupině. Změny se okamžitě ukládají do databáze
Následné podmínky	nejsou
Alternativní tok	Pokud není žádná skupina dosud vytvořena je o tom uživatel informován

Případ použití:	Vytvoření nové uživatelské skupiny
ID	5
Stručný popis	Uživatel může vytvářet nové uživatelské skupiny
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	nejsou
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel vyplní zobrazený formulář a odešle jej
Následné podmínky	Nová skupina je vytvořena
Alternativní tok	není

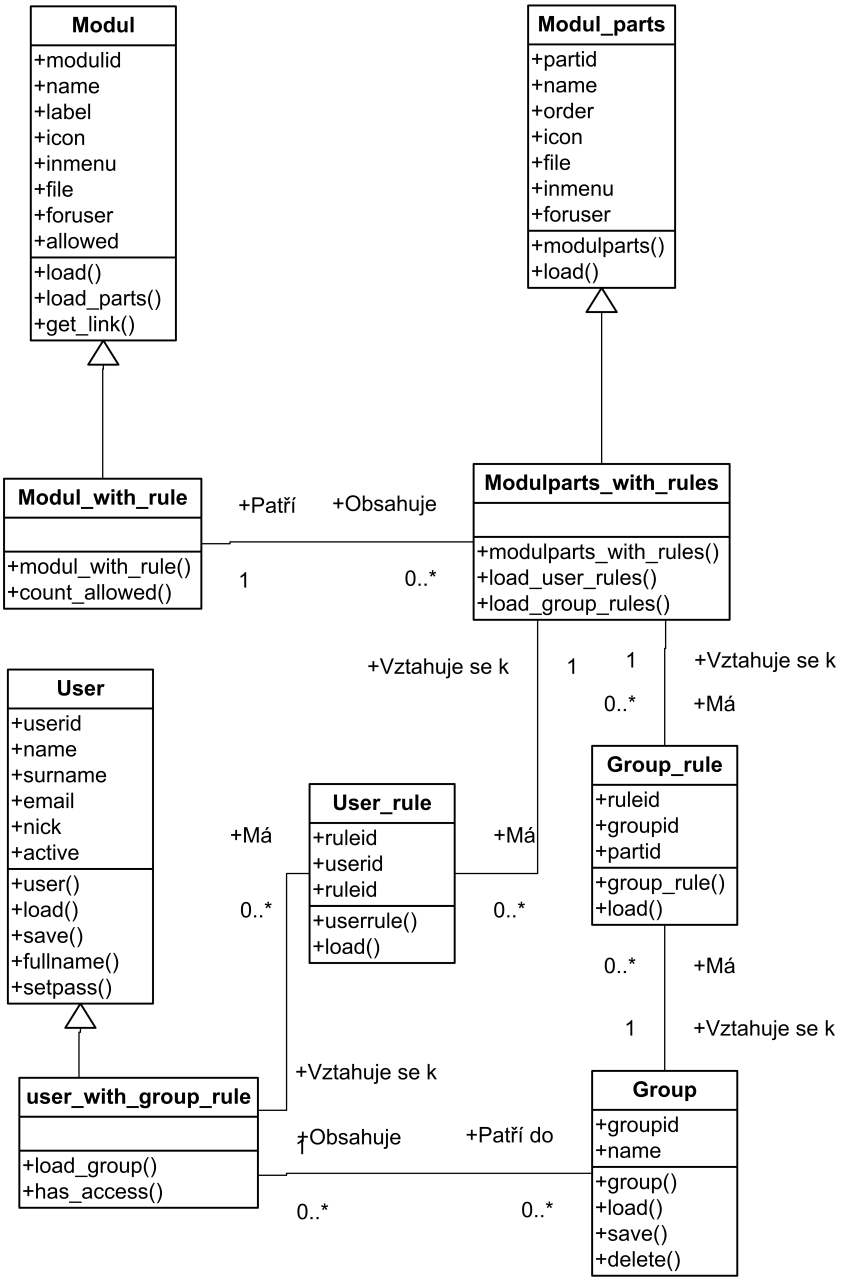
Případ použití:	Nastavení práv
ID	6
Stručný popis	Uživatel může nastavovat přístupová práva k modulům
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	Ostatní uživatelé
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel vybere ze seznamu uživatelů nebo uživatelských skupin 2. V seznamu modulů, které se zobrazí v pravé části obrazovky, může uživatel měnit přístupová práva
Následné podmínky	nejsou
Alternativní tok	není

4.3 Diagram tříd

Základní schéma použitých tříd v prvni iteraci je uvedeno na obráku 4.1.

4.3.1 Třída `Modul`

Třída `modul` modeluje jednotlivé aplikační části celého systému. Charakterizuje základní vlastnosti jako id, jméno zobrazované v menu, ikonu zobrazovanou v menu. Dále obsahuje seznam částí, které do dané aplikace spadají. Jejím rozšířením je třída `modul_with_rule`, která jej rozšiřuje o možnost načtení částí, ke kterým má určitý uživatel přístup a také metodou pro zjištění počtu těchto přístupných částí.



Obrázek 4.1: Diagram tříd – 1. iterace

4.3.2 Třída `Modul_parts`

Tato třída charakterizuje jednotlivé části modulů. Mimo jiné obsahuje vlastnosti jako pořadí (`order`) nebo `inmenu`, čímž určuje zda se daná položka zobrazuje v menu. Jejím rozšířením je třída `modulparts_with_rules`, která ji rozšiřuje o seznam přístupových pravidel vztažených k aktuálnímu uživateli a další seznam pravidel vztažených ke skupinám, do nichž aktuální uživatel patří. Dále rozšiřuje také o dvě metody pro načítání těchto pravidel.

4.3.3 Třída `User`

Třída modeluje jednotlivé uživatelské účty. Charakterizuje základní vlasntosti jako jméno (`name`), příjmení (`surname`), email (`email`) a zda je účet aktivní (`active`). Dále poskytuje metody pro načtění účtu z databáze, jeho uložení, nastavení hesla a zobrazení plného jména. Jejím rozšířením je třída `user_with_group_rule`, která poskytuje navíc seznam skupin, do kterých daný uživatel patří. K tomu je obsažena metoda pro načtení těchto skupin a metoda, která zjistí, zda má daný uživatel povolen přístup k určité funkci systému.

4.3.4 Třída `Group`

Třída group modeluje uživatelské skupiny. Mezi atributy patří jméno (`name`). Jako metody poskytuje možnost načtení se serveru, uložení na server a smazání ze serveru.

4.3.5 Třída `User_rule`

Třída modeluje možnost propojení uživatelského účtu s částí modulu a tím i modeluje přístupové právo mezi danými objekty.

4.3.6 Třída `Group_rule`

Třída modeluje podobně jako předchozí možnost definovat přístupová práva k modulům, tentokrát však pro uživatelské skupiny.

4.4 Entitně relační schéma databáze

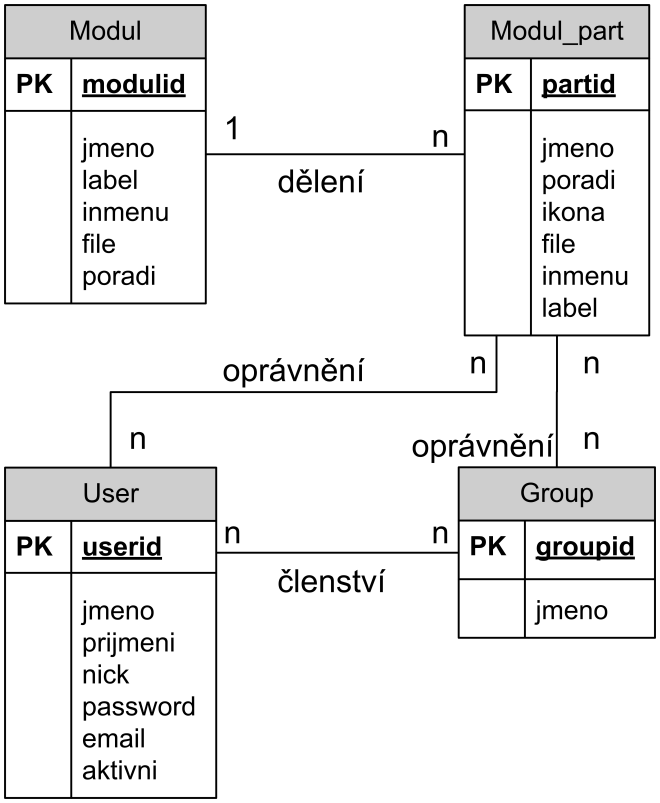
Pro první iteraci je uvedeno schéma entit na obrázku 4.2

4.5 Popis implementace

Protože některé požadavky jsou vzhledem k použité technoligii méně časté, popíši nyní jejich implementaci. Většina z nich vychází z použití technologie AJAX a práce s DOM stromem.

4.5.1 „Sbalení“ oken

Většina zobrazovaných údajů bude umístěna do nejrůznějších oken. Každý uživatel však bude mít jiné nejčastěji používané okna. Aby mu okna, která nepoužívá příliš často zbytečně nepřekážela a nezabírala místo, bude moci je jednoduše skývat, případně rozekrývat.



Obrázek 4.2: Entitně relační schéma databáze – 1. iterace

Pro tento účel bude mít každé takovéto okno ve svém pravém horním rohu ikonu, po jejímž kliknutí se daná akce provede. Aby tato změna byla perzistentní, tzn. při novém načtení stránky zůstal tento stav nezměněn, bude se vše ukládat na server do databáze. Díky uložení na jediném místě bude mít každý uživatel vždy poslední známe rozložení a to ať se přihlásí z libovolného počítače.

V databázi bude proto vytvořena nová tabulka, která tyto změny uchovává. Aby bylo možné jednotlivé okna identifikovat, bude mít každé z nich jedinečný identifikátor. Ten bude sestaven z jedinečného identifikátoru uživatele a také z označení okna. Toto označení okna bude jedinečné v rámci celé aplikace.

Při načítání stránky se pro každé okno provede databázový dotaz, který zjistí v jakém stavu se dané okno pro daného uživatel nachází. Na základě této informace se pro takové okno vybere CSS styl, pomocí něhož bude okno buď rozbalené nebo sbalené.

Pokud uživatel bude chtít změnit stav okna, klikne na již zmiňovanou ikonu, čímž se pomocí JavaSriptu změní CSS styl daného okna a pomocí AJAXu se vyvolá HTTP požadavek na server, kde se PHP skriptem změní informace v databázi. Díky tomu bude při novém načtení okno opět ve stejném stavu.

4.5.2 Editace informací

Při zobrazení údajů např. o uživateli, bude možné je přímo měnit. Aby se provedené změny ihned ukládaly na server, bude zde implementována JavaSriptová funkce, která se o uložení

postará. Po kliknutí na některou textovou informaci se zachycením události onClick spustí JavaScriptová funkce, která změní grafickouo podobu daného pole. Tato změna se provede změněním CSS stylu, který bude globálně definován.

Po provedení změn se vyvolá akcí onBlur jiná JavaScriptová funkce, která změní zpět styl daného prvku a vyvolá HTTP požadavek, na jehož základě PHP skript uloží provedené změny do databáze. Díky tomu odpadne nebezpečí, že uživatel zapomene provedené změny potvrdit jejich odesláním pomocí tlačítka.

Jelikož je provádění změn asynchronní, není třeba čekat na zaznamenání změny do databáze. Zde však dochází k riziku, že kvůli nějakým technikým potížím, nebude daná změna uložena. K tomu však může dojít i při použití klasických technik editace záznamů.

4.6 Testování 1. iterace

Pro otestování funkčnosti první iterace použiji testy, které prověří všechny implementované vlastnosti. Jelikož se jedná pouze o modelovou aplikaci, nebudou všechny vlastnosti, zvláště ty méně důležité, implemetovány. V zásadě jde pouze o analýzu možností nových technologií a jejich uvedení na praktických příkladech pro zvolenou aplikaci.

4.6.1 Test přihlášení do systému

Popis

Test prověří, zda je funkční přihlašování do systému.

Instrukce

1. Do formulářové položky `jméno` vlož uživatelské jméno
2. Do formulářové položky `heslo` vlož uživatelské heslo
3. Klikni na tlačítko `přihlásit`

Očekávaný výsledek

Pokud systém obsahuje uživatelský účet s daným jménem a heslem, bude uživatel přihlášen a zobrazí se úvodní obrazovka pro přihlášeného uživatele. V opačném případě se formulář pro přihlášení zobrazí znovu s informací o nezdařeném pokusu o přihlášení.

4.6.2 Test odhlášení ze systému

Popis

Test prověří, zda je funkční odhlašování ze systému.

Instrukce

1. Přihlaš se do systému

2. Klikni na tlačítko `odhlásit`

3. Klikni na tlačítko `zpět` okna prohlížeče

Očekávaný výsledek

Systém odhlásí aktuálně přihlášeného uživatele a zobrazí úvodní obrazovku pro nepřihlášeného uživatele – přihlašovací formulář. V daném prohlížeči by neměla zůstat žádná informace o předchozí práci uživatele. V historii navštívených stránek prohlížeče sice zůstane seznam provedených akcí, ale jejich opětovným vyvoláním se zobrazí opět pouze úvodní obrazovka nepřihlášeného uživatele.

4.6.3 Test skrývání obsahu oken

Popis

Test prověří, zda je funkční sbalování a rozbalování oken

Instrukce

1. Přihlaš se do systému

2. Změn libovolně stav jednotlivých oken, která jsou zobrazena v menu na levé straně

3. Klikni na tlačítko `odhlásit`

Očekávaný výsledek

Po opětovném přihlášení by měly mít všechna okna stejné zobrazení jako tomu bylo před odhlášením ze systému a to na jakémkoliv počítači.

4.6.4 Test editace uživatele

Popis

Test prověří, zda je funkční editování již uložených záznamů o uživatelích.

Instrukce

1. Přihlaš se do systému

2. Klikni na tlačítko `Seznam` v okně `Uživatelé` – pokud zde taková položka není, patrně nemáte přístup k dané části systému

3. Vyber některého ji vytvořeného uživatele

4. Kliknutím a změnou údajů edituj některou položku

5. Kliknutím na stav členství v určité skupině tento stav změn

6. Odhlaš se ze systému

Očekávaný výsledek

Po opětovném zobrazení informací o editovaném uživateli budou zřetelné provedené změny.

4.6.5 Test vytvoření nového uživatele

Popis

Test prověří, zda je funkční vytváření nových uživatelů.

Instrukce

1. Přihlaš se do systému
2. Klikni na tlačítko `Přidat uživatele` v okně `Uživatelé` – pokud zde taková položka není, patrně nemáte přístup k dané části systému
3. Ve formuláři postupně vyplň všechny položky
4. Kliknutím na tlačítko `Uložit` přejdi k dalšímu kroku
5. Ve formuláři vyplň heslo pro nového uživatele a to do obou políček
6. Kliknutím na tlačítko `Uložit` přejdi k dalšímu kroku
7. Vyber skupiny, do kterých nový uživatel patří
8. Odhlaš se ze systému

Očekávaný výsledek

Po opětovném zobrazení seznamu již vytvořených uživatelů bude seznam rozšířen o nového uživatele. Pokud nebude některá položka v průběhu vytváření uživatele vyplněna správně, nebude možné přejít k dalšímu kroku.

4.6.6 Test editace uživatelské skupiny

Popis

Test prověří, zda je funkční editování již vytvořených uživatelských skupin.

Instrukce

1. Přihlaš se do systému
2. Klikni na tlačítko `Seznam skupin` v okně `Uživatelé` – pokud zde taková položka není, patrně nemáte přístup k dané části systému
3. Vyber některou již vytvořenou skupinu – pokud zde žádná není, je ji třeba nejprve vytvořit
4. Kliknutím a změnou údajů edituj některou položku
5. Odhlaš se ze systému

Očekávaný výsledek

Po opětovném zobrazení dané skupiny budou patrné provedené změny.

4.6.7 Test vytvoření nové uživatelské skupiny

Popis

Test prověří, zda je funkční vytváření nových uživatelských skupin.

Instrukce

1. Přihlaš se do systému
2. Klikni na tlačítko `Přidat skupinu` v okně `Uživatelé` – pokud zde taková položka není, patrně nemáte přístup k dané části systému
3. Vyplň jméno nové skupiny
4. Kliknutím na tlačítko `Uložit` ulož novou skupinu
5. Odhlaš se ze systému

Očekávaný výsledek

Po opětovném zobrazení seznamu již vytvořených uživatelských skupin bude seznam rozšířen o novou skupinu.

4.6.8 Test nastavování uživatelských práv

Popis

Test prověří, zda je funkční nastavování uživatelských práv.

Instrukce

1. Přihlaš se do systému
2. Klikni na tlačítko `Nastavení práv` v okně `Uživatelé` – pokud zde taková položka není, patrně nemáte přístup k dané části systému
3. Zvolte uživatelský účet nebo skupinu uživatelů
4. V seznamu modulů kliknutím na aktuální stav změň nastavení oprávnění
5. Odhlaš se ze systému

Očekávaný výsledek

Po přihlášení změněným uživatelským účtem bude pravé menu odpovídat novému nastavení.

4.7 Závěr

Výsledkem první iterace je základní část serverové aplikace, která poskytuje základní rozhraní pro libovolné moduly. K těmto modulům je možné díky podpoře uživatelských účtů nastavovat přístupová oprávnění. Díky velké flexibilitě je možné vytvářet nejrůznější skupiny uživatelů, pro které bude již předem nastaven omezený přístup.

Kapitola 5

Druhá iterace

5.1 Požadovaný výsledek

V této části vývoje bude implementován katalog zboží.

5.1.1 Kategorie

Pro řazení zboží bude vytvořen seznam kategorií. Jednotlivé kategorie budou do sebe hiearchicky řazeny.

5.1.2 Slovník vlastností

Každá kategorie bude mít přiřazeny určité vlastnosti, které bude poskytovat seznamu zboží v ní obsažené. Každá vlastnost bude buď textová nebo bude mít určen výčet přípustných hodnot. Z těchto hodnot bude možné určit, které jsou jednotlivým kategoriím přístupné.

5.1.3 Zboží

Jednotlivé druhy zboží budou zařazeny do příslušných kategorií. Tato kategorie jim zároveň určí, jaké mají vlastnosti a také vymezí, které hodnoty jsou jim dostupné.

5.2 Specifikace případů použití

Případ použití:	Seznam kategorií a jejich editace
ID	7
Stručný popis	Uživatel si zobrazí stromový seznam kategorií a bude moci editovat jejich vlastnosti
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	nejsou
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel vybere některou z kategorií 2. Uživatel mění jednotlivé vlastnosti včetně slovníku, jejího zanoření a položek, které jsou dostupné. Změny se okamžitě ukládají do databáze
Následné podmínky	nejsou
Alternativní tok	Pokud není zatím žádná kategorie vytvořena, je o tom uživatel informován.

Případ použití:	Vytvoření nové kategorie
ID	8
Stručný popis	Uživatel vytvoří novou kategorii
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	nejsou
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel vyplní jméno nové kategorie 2. Uživatel určí, do které části stromu nová kategorie patří 3. Uživatel vybere slovníkové vlastnosti a příslušné položky
Následné podmínky	nejsou
Alternativní tok	Pokud není jméno kategorie nebo její zařazení vyplněno, není možné pokračovat

Případ použití:	Editace slovníku
ID	9
Stručný popis	Uživatel edituje slovníkové vlastnosti
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	nejsou
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel vybere vlastnost, kterou chce editovat 2. Uživatel změní jméno dané vlastnosti 3. Uživatel změní, zda se jedná o vlastnost textovou nebo výčtovou 4. Pokud se jedná o výčtovou vlastnost, uživatel může editovat jednotlivé položky vlastnosti, případně přidávat nové
Následné podmínky	nejsou
Alternativní tok	není

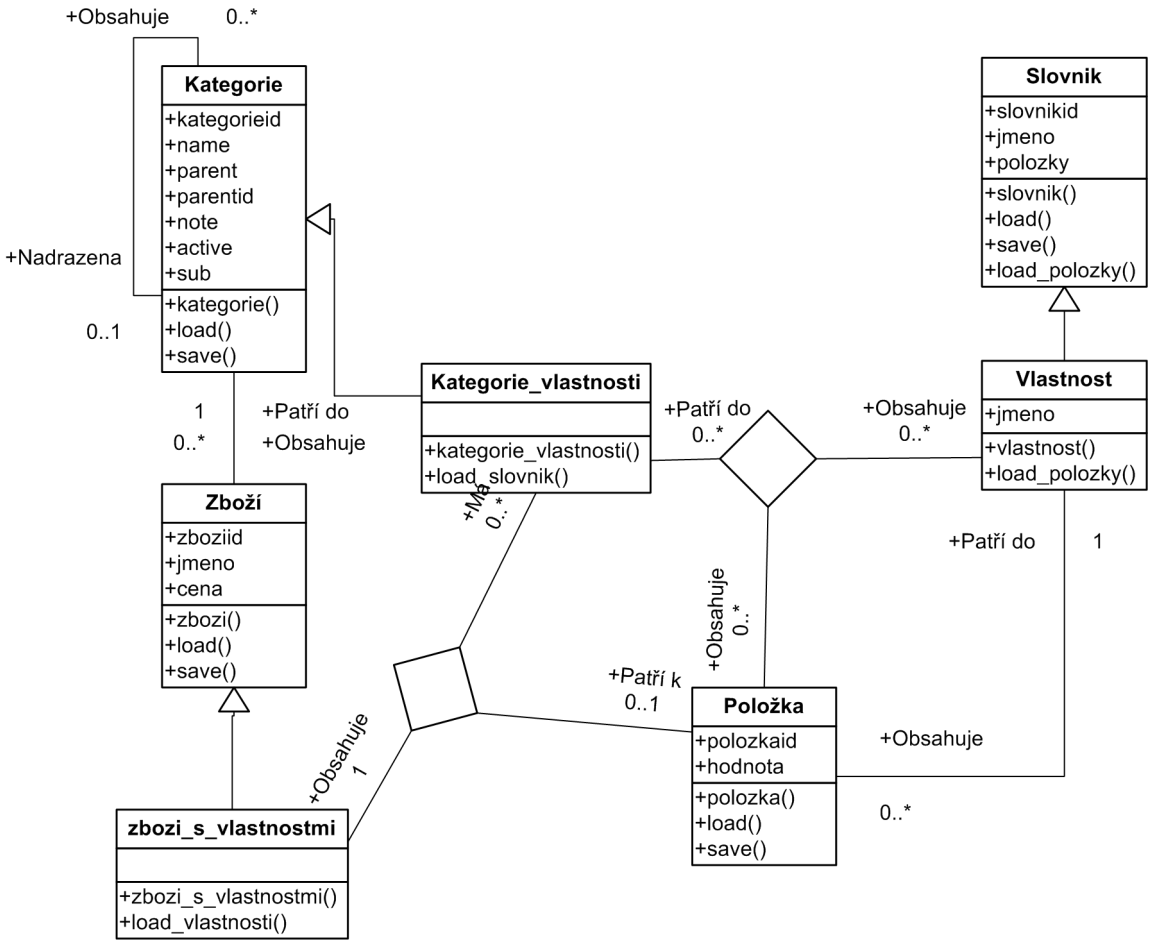
Případ použití:	Editace slovníku
ID	10
Stručný popis	Vytvoření nové slovníkové položky
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	nejsou
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel vyplní jméno nové položky 2. Uživatel klikne na `Uložit` a přejde k dalšímu kroku 3. Uživatel změní, zda se jedná o vlastnost textovou nebo výčtovou 4. Pokud se jedná o výčtovou vlastnost, uživatel může přidávat jednotlivé položky vlastnosti, případně je editovat
Následné podmínky	nejsou
Alternativní tok	není

Případ použití:	Zobrazení a editace zboží
ID	11
Stručný popis	Uživatel může zobrazit jednotlivé zboží z kategorií a měnit jejich vlastnosti
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	nejsou
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel vybere kategorii, ze které chce zobrazit zboží 2. Ze seznamu zboží vybere to, o které má zájem 3. Uživatel edituje jednotlivé vlastnosti. Změny se okamžitě ukládají do databáze.
Následné podmínky	nejsou
Alternativní tok	Pokud není žádné zboží v dané kategorii vytvořeno, je o tom uživatel informován

Případ použití:	Vytvoření nového zboží
ID	12
Stručný popis	Uživatel může vytvořit nové zboží
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	nejsou
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel vybere kategorii, do které chce vložit zboží 2. Uživatel klikne na tlačítko vložit zboží 3. Uživatel vyplní formulář, který obsahuje všechny vlastnosti z kategorie. 4. Tlačítkem `Uložit` uloží nový produkt do katalogu
Následné podmínky	nejsou
Alternativní tok	Pokud není žádné zboží v dané kategorii vytvořeno, je o tom uživatel informován

5.3 Diagram tříd

Základní schéma použitých tříd v druhé iteraci je uvedeno na obráku 5.1.



Obrázek 5.1: Diagram tříd – 2. iterace

5.3.1 Třída `Kategorie`

Třída modeluje jednotlivé kategorie. Umožňuje také modelovat zanoření jednotlivých kategorií do hiearchického stromu. Obsahuje základní vlastnoti jako jméno, ID, zda je kategorie aktivní, informace o nadřazené kategorii a seznam vnořených kategorií. Její metody slouží pro načtení a uložení kategorie do databáze. Jejím rozšířením je třída `Kategorie_s_vlastnostmi`, která ji rozšiřuje o seznam jednotlivých vlastností, které jsou pro ni dostupné ze slovníku. Dále ji rozšiřuje i o metodu pro načtení těchto vlastností z databáze.

5.3.2 Třída `Zboží`

Třída modeluje jednotlivé produkty. Umožňuje jejich zařazení do kategorií. Jejím rozšířením je třída `Zboží_s_ vlastnostmi`, která ji rozšiřuje o možnost uvést hodnoty slovníkových vlastností.

5.3.3 Třída `Slovník`

Třída slovník charakterizuje jednotlivé vlastnosti, které lze přiřazovat jednotlivým kategoriím případně zboží. Mezi vlastnosti patří určení zda se jedná o textovou nebo výčtovou vlastnost. Pokud se jedná o výčtovnou vlastnost, je třída rozšířena třídou `Vlastnost`. Ta umožňuje vypsat jednotlivé položky, kterých může nabývat.

5.3.4 Třída `Položka`

Třída `Položka` umožňuje modelovat jednotlivé položky, které je možno přiřadit výčtovým instancím třídy `Slovník`.

5.4 Entitně relační schéma databáze

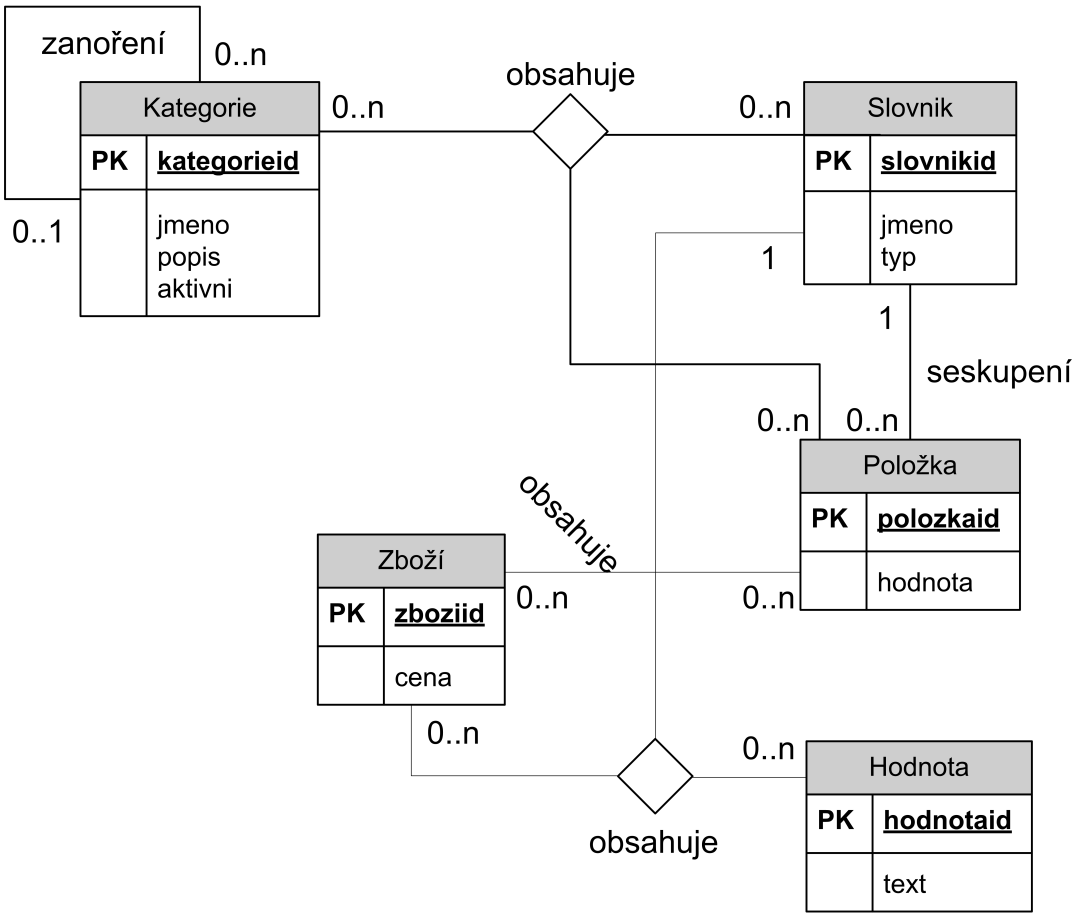
Pro druhou iteraci je uvedeno schéma entit na obrázku 5.2

5.5 Popis implementace

5.5.1 Stromové řazení

Některé položky, které mají hiearchické řazení, budou zobrazeny ve stromu, který je možné libovolně rozbalovat a opětovně sbalovat. Aby tato funkce měla opodstatnění, musí se jednat o vlastnost, která je perzistentní. Pro tento účel bude implementována funkce v JavaScriptu, která bude nejen měnit strukturu, ale také ukládat toto nastavení do databáze. Každý takový prvek bude mít jedinečné označení v rámci celé aplikace. Stav každého takového prvku se bude vztahovat nejen k označení prvku, ale také k identifikaci uživatele.

Po kliknutí na prvek značící rozbalení nebo sbalení prvku se nejprve zavolá funkce JavaScriptu, která provede změnu v DOM přepsáním CSS a následně uloží pomocí požadavku AJAXu tuto změnu do databáze. Při novém načtení daného prvku se z databáze načte jeho aktuální stav.



Obrázek 5.2: Entitně relační schéma databáze – 2. iterace

5.5.2 Editace vlastností

Všechny vlastnosti, které jsou definovány výčtem, bude možné změnit pomocí elementu `SELECT`. Každá takové položka bude nejprve zobrazena jako prostý text, po kliknutí se však změní na formulářovou položku. Po změně hodnoty se vyvolá událost JavaScriptu `onChange`, která dané změny uloží pomocí AJAXu do databáze a následně změní zpět položku na prostý text.

U textových vlastností se opět po kliknutí změní prvek na element `INPUT`, který umožní editovat hodnotu. Po změně se vyvolá událost JavaScriptu `onBlur`, která značí zrušení aktivního přístupu k položce (opak `onFocus`). Tato funkce uloží provedené změny do databáze a následně změní opět prvek na prostý text.

5.6 Testování 2. iterace

5.6.1 Test vytvoření nové kategorie

Popis

Test prověří, zda je funkční vytváření nových kategorií.

Instrukce

1. Přihlaš se do systému
2. Klikni na tlačítko `Přidat` v okně `Kategorie` – pokud zde taková položka není, patrně nemáte přístup k dané části systému
3. Ve formuláři vyplň jméno nové kategorie
4. Vyber nadřazenou kategorii
5. Kliknutím na tlačítko `Uložit` přejdi k dalšímu kroku
6. Ve formuláři postupně přidávej vlastnosti, které bude daná kategorie obsahovat
7. Kliknutím na tlačítko `Uložit` ulož novou kategorii do databáze
8. Odhlaš se ze systému

Očekávaný výsledek

Po opětovném zobrazení seznamu již vytvořených kategorií bude seznam rozšířen o novou kategorii. Pokud nebude některá položka v průběhu vytváření kategorie vyplněna správně, nebude možné přejít k dalšímu kroku.

5.6.2 Test editace kategorie

Popis

Test prověří, zda je funkční editování již vytvořených kategorií.

Instrukce

1. Přihlaš se do systému
2. Klikni na tlačítko `Seznam` v okně `Kategorie` – pokud zde taková položka není, patrně nemáte přístup k dané části systému
3. V seznamu již vytvořených kategorií zvol kategorii k editaci
4. Postupně změn jednotlivé položky
5. Podle potřeby přidej nebo edituj již vložené vlastnosti
6. Odhlaš se ze systému

Očekávaný výsledek

Po opětovném zobrazení informací o kategorii budou zobrazeny provedené změny.

5.6.3 Test vytvoření nové slovníkové položky

Popis

Test prověří, zda je funkční vytváření slovníkových položek.

Instrukce

1. Přihlaš se do systému
2. Klikni na tlačítko `Slovník` v okně `Kategorie` – pokud zde taková položka není, patrně nemáte přístup k dané části systému
3. Klikni na Přidat položku
4. Vyplň jméno nové vlastnosti
5. Kliknutím na tlačítko `Uložit` přejdi k dalšímu kroku
6. Zvol zda jde o textovou nebo výčtovou vlastnost
7. Pokud jde o výčtovou vlastnost, přidej jednotlivé položky
8. Odhlaš se ze systému

Očekávaný výsledek

Po opětovném zobrazení seznamu vlastností bude seznam rozšířen o novou položku.

5.6.4 Test editace slovníkové položky

Popis

Test prověří, zda je funkční vytváření slovníkových položek.

Instrukce

1. Přihlaš se do systému
2. Klikni na tlačítko `Slovník` v okně `Kategorie` – pokud zde taková položka není, patrně nemáte přístup k dané části systému
3. Vyber některou již vytvořenou položku
4. Postupně změn jednotlivé položky – změny se okamžitě ukládají do databáze
5. Odhlaš se ze systému

Očekávaný výsledek

Po opětovném zobrazení změněné vlastnosti budou provedené změny patrné.

5.6.5 Test přidání zboží

Popis

Test prověří, zda je funkční přidávání zboží.

Instrukce

1. Přihlaš se do systému
2. Klikni na tlačítko `Katalog` v okně `Zboží` – pokud zde taková položka není, patrně nemáte přístup k dané části systému
3. Vyber některou kategorii
4. Klikni na přidat produkt
5. Vylň zobrazený formulář
6. Odhlaš se ze systému

Očekávaný výsledek

Po opětovném zobrazení katalogu zboží bude nový produkt uveden v seznamu.

5.6.6 Test editace zboží

Popis

Test prověří, zda je funkční editování zboží.

Instrukce

1. Přihlaš se do systému
2. Klikni na tlačítko `Katalog` v okně `Zboží` – pokud zde taková položka není, patrně nemáte přístup k dané části systému
3. Vyber některou kategorii
4. Zvol některý produkt
5. Uprav požadované vlastnosti
6. Odhlaš se ze systému

Očekávaný výsledek

Po opětovném zobrazení měněného zboží budou změny patrné.

5.7 Závěr

V druhé iteraci byly vytvořeny moduly, které umožňují katalogizovat zboží. Díky slovníku vlastností, je možné ke každé kategorii vytvořit relevantní vlastnosti, které umožní přesně specifikovat produkty v ní obsažené. Této vlastnosti by šlo dále využít při vyhledávání v produktech, jelikož by se daly přesně specifikovat požadavky na výsledek.

Kapitola 6

Třetí iterace

6.1 Požadovaný výsledek

V této části vývoje systému bude vytvořen vzorový eshop, který umožní zobrazování zboží a jeho objednávku. Pro konfiguraci jednotlivých obchodů v hlavní aplikaci bude vytvořen modul, který umožní nastavení kategorií.

6.1.1 Eshop

Každý eshop bude implementován na vlastní databázové struktuře. Pro komunikaci využije výměnu dokumentů XML. Bude umožňovat jen základní zobrazení zboží v kategoriích. Po objednání zašle hlavní aplikaci danou objednávku, která ji zaznamená.

6.1.2 Komunikace

Komunikace bude probíhat na základě požadavku HTTP, ve kterém bude předán dotaz na požadované informace. Tyto informace budou chráněny proti změně pomocí hashe, který bude sestaven z parametrů a dále z tajného klíče.

6.1.3 Objednávky

V hlavní aplikaci bude implementován modul, který umožní zobrazování jednotlivých objednávek.

6.2 Specifikace případu použití

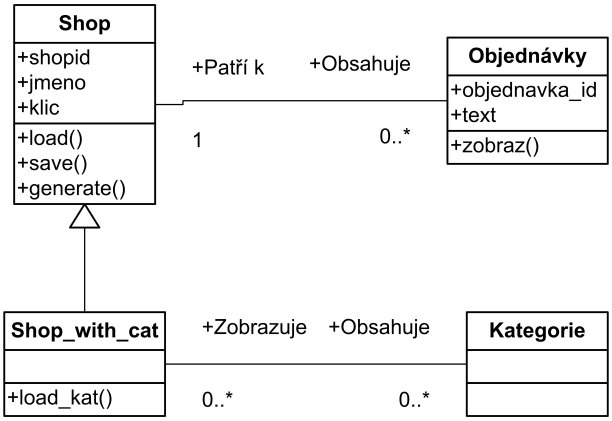
Případ použití:	Seznam eshopů a jejich editace
ID	7
Stručný popis	Uživatel si zobrazí stromový seznam eshopů a bude moci editovat jejich vlastnosti
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	nejsou
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel vybere některý eshop 2. Uživatel mění jednotlivé vlastnosti včetně tajného klíče a zobrazovaných kategorií
Následné podmínky	nejsou
Alternativní tok	Pokud není zatím žádný eshop vytvořen, je o tom uživatel informován.

Případ použití:	Vytvoření eshopu
ID	7
Stručný popis	Uživatel vytvoří nový eshop
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	nejsou
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel vyplní jméno nového eshopu 2. Uživatel mění jednotlivé vlastnosti včetně tajného klíče a zobrazovaných kategorií
Následné podmínky	nejsou
Alternativní tok	Pokud není jméno vyplněno, není možné přejít k dalšímu kroku.

Případ použití:	Zobrazení objednávek
ID	7
Stručný popis	Uživatel si zobrazí seznam objednávek
Primární aktéři	Oprávněný uživatel
Sekundární aktéři	nejsou
Předpoklady	Uživatel má právo přístupu k tomuto modulu
Hlavní tok	1. Uživatel si zobrazí objednávku z celkového seznamu
Následné podmínky	nejsou
Alternativní tok	Pokud nejsou prozatím žádné objednávky vytvořeny je o tom uživatel informován.

6.3 Diagram tříd

Základní schéma použitých tříd ve třetí iteraci je uvedeno na obráku 6.1.



Obrázek 6.1: Diagram tříd – 3. iterace

6.3.1 Třída `Shop`

Tato třída modeluje jednotlivé elektronické obchody v hlavní aplikaci. Obsahuje atributy id, jméno a tajný klíč používaný při komunikaci. Metody slouží pro načtení a uložení dat do databáze a také metodu pro generování tajného klíče.

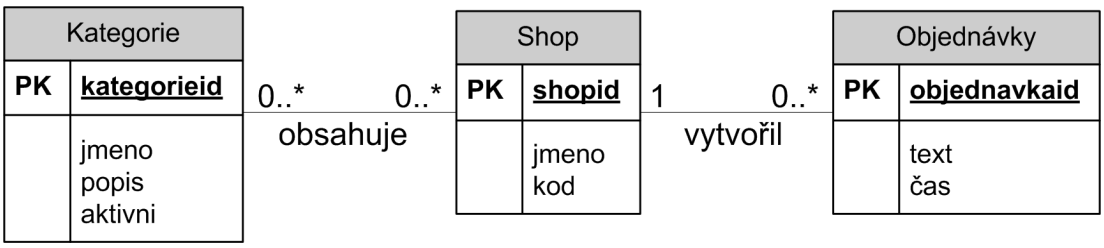
Rozšířením této metody je metoda `Shop_with_cat`, která základní třídu rozšiřuje o možnost zobrazení jednotlivých kategorií, které do daného eshopu patří.

6.3.2 Třída `Objednávky`

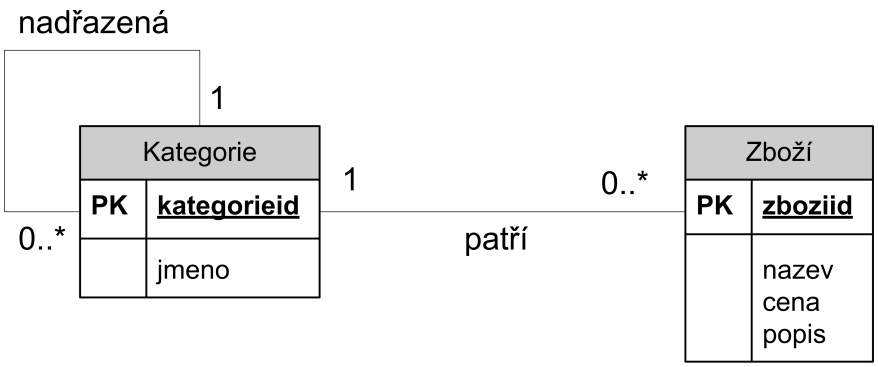
Třída `Objednávky` modeluje jednotlivé objednávky vytovřené jednotlívými eshopy. Tato třída umožňuje pouze základní zobrazení objednaného zboží v textové podobě.

6.4 Entitně relační schéma databáze

Pro třetí iteraci jsou uvedena schémata entit na obrázku 6.2 pro hlavní aplikaci a na obrázku 6.3 pro klientský eshop.



Obrázek 6.2: Entitně relační schéma databáze – 3. iterace, hlavní aplikace



Obrázek 6.3: Entitně relační schéma databáze – 3. iterace, eshop

6.5 Popis implementace

6.5.1 Komunikace

Pro zachování integrity budou parametry předávané v HTTP požadavku chráněny pomocí tajného klíče a MD5[1] hashe. Tento otisk bude vytvořen z řetezce sestaveného z parametrů a tajného klíče, který bude následně připojen za předávané paramerty. Po přijetí serverem se opět sestaví řetězec parametrů a daného klíče, vytvoří se otisk a porovná se s předaným.

[1] Message-Digest algorithm je rozšířená rodina hashovacích funkcí, která vytváří ze vstupních dat výstup (otisk) fixní délky. Otisk je též označován jako miniatura, kontrolní součet (v zásadě nesprávné označení), fingerprint, hash (česky někdy psán i jako haš). Jeho hlavní vlastností je, že malá změna na vstupu vede k velké změně na výstupu, tj. k vytvoření zásadně odlišného otisku.[8]

Na základě této jednoduché autentizace se sestaví XML dokument podle předaných parametrů a ten se vrátí zpět do klientské aplikace. Tato aplikace si daný dokument rozparsuje a aktualizuje příslušné databázové struktury.

6.5.2 Vytváření objednávky

Při vytváření objednávky se budou jednotlivé položky ukládat do session. Následně k nim zákazník přidá osobní údaje a celá tato informace se odešle na server. Zde se uloží do databáze.

6.5.3 Aktualizace dat

Pro aktualizaci dat se odešle HTTP požadavek na dané adresy. Ty zajistí předání nových dat, které si následně klientská aplikace zaktualizuje. Tato aktualizace se bude provádět automaticky pomocí automatického spouštění skriptů.

6.6 Testování 3. iterace

6.6.1 Test vytvoření nového eshopu

Popis

Test prověří, zda je funkční vytváření nového eshopu.

Instrukce

1. Přihlaš se do systému
2. Klikni na tlačítko `Seznam` v okně `eShop` – pokud zde taková položka není, patrně nemáte přístup k dané části systému
3. Klikni na tlačítko `Nový eshop` – pokud zde taková položka není, patrně nemáte přístup k dané části systému
4. Ve formuláři vyplň jméno nového eshopu
5. Kliknutím na tlačítko `Uložit` přejdi k dalšímu kroku
6. Vygeneruj nový tajný klíč pro komunikaci
7. Postupně zvol kategorie, které bude daný eshop obsahovat – údaje se automaticky aktualizují v databázi
8. Odhlaš se ze systému

Očekávaný výsledek

Po opětovném zobrazení seznamu již vytvořených eshopů bude seznam rozšířen o nově vytvořený. Pokud nebude některá položka v průběhu vytváření eshopu vyplněna správně, nebude možné přejít k dalšímu kroku.

6.6.2 Test editace eshopu

Popis

Test prověří, zda je funkční editace již vytvořeného eshopu.

Instrukce

1. Přihlaš se do systému
2. Klikni na tlačítko `Seznam` v okně `eShop` – pokud zde taková položka není, patrně nemáte přístup k dané části systému
3. Zvol některý již vytvořený eshop ze seznamu – pokud zde žádný není, je třeba nejprve některý vytvořit
4. Ve formuláři edituj jméno eshopu
5. Vygeneruj nový tajný klíč pro komunikaci
6. Postupně zvol kategorie, které bude daný eshop obsahovat – údaje se automaticky aktualizují v databázi
7. Odhlaš se ze systému

Očekávaný výsledek

Po opětovném zobrazení seznamu již vytvořených eshopů budou provedené změny patrné.

6.6.3 Test aktualizace dat

Popis

Test prověří, zda je funkční aktualizace dat.

Instrukce

1. Otevři URL `adresa/xml/load.php?akce=kategorie`
2. Otevři URL `adresa/xml/load.php?akce=zbozi`

Očekávaný výsledek

Po zobrazení kategorií daného eshopu budou zobrazeny ty, které jsou nastaveny v hlavní aplikaci, včetně zboží.

6.7 Závěr

Cílem této práce bylo zanalyzovat problematiku centrální distribuce zboží pro elektronické obchody. Byly stanoveny základní požadavky na funkčnost takové aplikace jako je jednoduchá správa, použití velmi jednoduchého a vysoce funkčního rozhraní, možnost jednoduchého přidání nového eshopu a také určitým způsobem oddělení jednotlivých aplikací.

Dále byl postupně ve třech iteracích vytvářen model daného systému s postupnou implementací nových vlastností. První iterace vytváří základní model systému, který umožní zapracování nejrůznějších modulů. Tuto část systému je v budoucnu možné použít i pro další aplikace, a to díky velmi flexibilní struktuře.

Ve druhé iteraci byl vytvořen katalog zboží. Tento katalog se od ostatních běžně používaných (viz nejrůznější již implementované obchody na internetu) vyznačuje možností tvořit kategorie podle těch nejpřesnějších požadavků. Všechny vlastnosti, které může kategorie obsahovat, jsou obsaženy ve slovníku, který lze opět velmi snadno rozšiřovat.

Díky této velmi robustní vlastnosti celého systému, lze u každého produktu specifikovat jeho přesné vlastnosti. Při výběru konkr0tního produktu si tedy zákazník může vybrat přesně takový produkt, který mu bude vyhovovat.

V poslední části vývoje byl navržen jednoduchý internetový obchod, který umožňuje komunikovat s hlavní aplikací. Tato aplikace může být dále nasazována v předem neomezém množství a je tedy možné vytvořit internetové obchody s nejrůznějším zaměřením. Jde vždy již pouze o zpracování předaných informací z hlavní aplikace.

Celý systém však může být rozšířen o další vlastnosti, které by však již překračovaly účel této práce a to analyzovat základní požadavky na takovéto systémy. Mezi další rozšíření by bylo jistě vhodné zahrnout bezpečnější způsob komunikace HTTPS, který by chránil přenášené informace přes internet.

Dalším rozšířením by bylo předávání všech slovníkových položek klientským aplikacím, který by následně dovolovaly pomocí těchto informací např. vyhledávání. Takovéto obchody by se staly vysoce uživatelsky přívětivými.

Dalším, ne však určitě posledním, rozšířením by byla možnost doplnění jednotlivých produktů o jejich fotografie a tím poskytnout maximální servis klientským eshopům.

Tato práce byla pro mne přínosná v tom, že jsem si osvojil některé nové prvky programování, poznal nové techniky ať již použitého jazyka nebo komunikace. Dále jsem poznal i jisté prvky projektového řízení, jako je tvorba v iteracích nebo psaní dokumentace.

V této práci se také ukázalo, že pro tvorbu rozsáhlejších projektů je vhodné použít objektový přístup. U menších to již tak zřejmé není vzhledem k náročnosti analýzy u objektového návrhu a i možného většího nároku na rychlost aplikace. Volba vhodné databáze je vždy spíše otázkou příležitosti.

Literatura

[1] Nathaniel T. Schutta a Ryan Asleson. *AJAX Vytváříme vysoce interaktivní webové aplikace*. Computer Press, 2006. ISBN 80-251-1285-3.

[2] WWW stránky. Apache http server - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/Apache`.

[3] WWW stránky. Asynchronous javascript and xml - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/AJAX`.

[4] WWW stránky. Cascading style sheets - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/Cascading_Style_Sheets`.

[5] WWW stránky. Document object model - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/DOM`.

[6] WWW stránky. Document type definition - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/DTD`.

[7] WWW stránky. Javascript - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/JavaScript`.

[8] WWW stránky. Message-digest algorithm - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/MD5`.

[9] WWW stránky. Mysql - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/Mysql`.

[10] WWW stránky. Php - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/PHP`.

[11] WWW stránky. Phpmyadmin - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/PhpMyAdmin`.

[12] WWW stránky. Postgresql - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/Postgresql`.

[13] WWW stránky. World wide web consortium - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/W3C`.

[14] WWW stránky. Xhtml - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/XHTML`.

Seznam obrázků