VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

POROVNÁNÍ EXISTUJÍCÍCH PLATFORMEM PRO VÝVOJ WEBOVÝCH APLIKACÍ

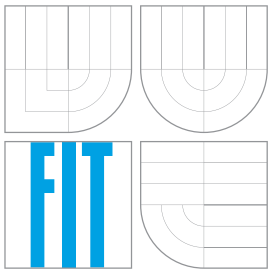
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MILAN BERAN

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

POROVNÁNÍ EXISTUJÍCÍCH PLATFOREM PRO VÝVOJ WEBOVÝCH APLIKACÍ

COMPARE EXISTING PLATFORMS FOR DEVOLEPMENT WEB APPLICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MILAN BERAN
AUTHOR

VEDOUCÍ PRÁCE — Ing. DUŠAN VRÁŽEL
SUPERVISOR

BRNO 2008

Zadání bakalářské práce/6631/2007/xberan25

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Beran Milan**
Obor: Informační technologie
Téma: **Porovnání existujících platforem pro vývoj webových aplikací**
Kategorie: Web

Pokyny:

1. Seznamte se s existujícími platformami a nástroji pro tvorbu webu (PHP, JSP, .NET, CGI skripty).
2. Porovnejte jednotlivé platformy a zhodnoťte výhody a nevýhody jednotlivých přístupů.
3. Navrhněte vhodnou aplikaci, kterou použijete k demonstraci rozdílů a specifik jednotlivých platforem. Při návrhu využijte jazyka UML.
4. Po dohodě s vedoucím vyberte tři platformy a navrženou aplikaci implementujte ve vybraných platformách. Funkčnost aplikace ověřte na vhodně zvoleném vzorku dat.
5. Zhodnoťte dosažené výsledky a shrňte vaše zkušenosti z implementace v jednotlivých platformách.

Literatura:

- Welling L., Thomson L.: PHP a MySQL rozvoj webových aplikací, druhé vydání. Softpress 2004.
- Hall M.: Core Servlets and JavaServer Pages. Prentice Hall 2000.
- Birznieks G. a kol.: CGI Programming with Perl. O'Reilly Media 2000.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Vrážel Dušan, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Milan Beran**
Id studenta: 84463
Bytem: Postřelmovská 465/1a, 789 01 Zábřeh
Narozen: 27. 04. 1985, Zábřeh
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Porovnání existujících platforem pro vývoj webových aplikací
Vedoucí/školitel VŠKP: Vrážel Dušan, Ing.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Práce se zabývá porovnáním existujících nástrojů a platforem pro tvorbu webových aplikací. V úvodu je provedeno teoretické porovnání nejznámějších nástrojů a platforem. Hlavní část práce se věnuje návrhu webové aplikace. Tato aplikace byla naprogramována pomocí ASP.NET C#, PHP a Ruby On Rails. Zkušenosti s implementací pomocí jednotlivých nástrojů jsou shrnuty v závěru práce.

Klíčová slova

ASP.NET C#, CGI skripty, JSP, PHP, PostgreSQL, Ruby On Rails, UML, Webové aplikace

Abstract

Main topic of this work is comparation of existing web application tools and platforms. The most known tools and platforms are theoretically compared in the introduction. The next aim of this work is design of a web application and implementation using tools ASP.NET C#, PHP and Ruby On Rails. Experience with these tools is summarized at the end of this work.

Keywords

ASP.NET C#, CGI scripts, JSP, PHP, PostgreSQL, Ruby On Rails, UML, Web application

Citace

Milan Beran: Porovnání existujících platforem pro vývoj webových aplikací, bakalářská práce, Brno, FIT VUT v Brně, 2008

Porovnání existujících platforem pro vývoj webových aplikací

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Dušana Vrážela.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Milan Beran
13. května 2008

Poděkování

Rád bych poděkoval vedoucímu práce za cenné rady a připomínky při psaní této práce.

Obsah

Kapitola 1

Úvod

V dnešní době je na internetu dostupné velké množství nástrojů a platforem pro tvorbu webových aplikací. Některé jsou velmi rozšířené a známe a o některých slyšela jen malá skupina nadšenců, která se tvorbou webových aplikací zabývá. Tato bakalářská práce se zabývá porovnáním několika těchto nástrojů.

První část práce (kapitola 3) vychází ze studia dostupných materiálů a věnuje se teorii a teoretickému srovnání jednotlivých nástrojů. Tato část vychází převážně z cizích zdrojů a informace v ní nejsou sepsány na základě vlastních zkušeností s jednotlivými platformami, ale pouze z toho, co bylo nastudováno.

V druhé části je představen návrh aplikace, která byla implementována na třech vybraných platformách (kapitola 4). Tyto tři implementace byly dále porovnány a zhodnoceny na základě několika aspektů práce s nimi (kapitola 5). Toto srovnání již vychází z konkrétních osobních zkušeností, které byly při implementaci aplikace pomocí vybraných nástrojů získány.

Hlavním důvodem pro téma srovnávání nástrojů pro tvorbu webových aplikací je zájem o dané téma a zájem naučit se novým technologiím a postupům při tvorbě internetových aplikací.

Kapitola 2

Cíle práce

Hlavním cílem této bakalářské práce je porovnat tři implementace jedné webové aplikace, které byly pro účely práce naprogramovány. Dále je provedeno hodnocení dosažených výsledků, zkušeností s jednotlivými nástroji.

Samotnému srovnávání předcházelo několik důležitých úkolů. Nejprve byly nastudovány informace o existujících nástrojích, platformách a technologiích, které se v současné době využívají pro tvorbu webových aplikací. Práce se zaměřuje hlavně na čtveřici nejrozšířenějších nástrojů a to na PHP, JSP, .NET a CGI skripty. Na základě nastudovaných informací bylo provedeno srovnání a uvedeny výhody a nevýhody jednotlivých přístupů.

Po nastudování obecných informací byl za využití jazyka UML proveden návrh webové aplikace, která slouží k demonstraci a srovnávání rozdílů a specifik vybraných platforem.

Navržená aplikace byla poté implementována do tří vybraných platforem. Vybranými platformami jsou ASP.NET C#, PHP a Ruby on Rails. Zkušenosti s jednotlivými nástroji jsou shrnuty v závěru práce.

Kapitola 3

Teoretická část

Většina informací, která je uvedena v následující kapitole, je čerpána z cizích zdrojů a není založena na osobních zkušenostech. Vlastní zkušenosti má autor pouze s platformou PHP, kterou již delší dobu využívá. Přesto se autor domnívá, že při srovnávání se podařilo zachovat objektivitu. Porovnání na základě osobních zkušeností je uvedeno v kapitole 5.

3.1 Základní teorie a vymezení pojmů

V následující části práce je uvedeno několik definic pojmů, se kterými je v textu dále pracováno. Nejedná se o žádné mohutné definice jednotlivých termínů. Cílem kapitoly je pouze stručné shrnutí základních informace.

3.1.1 Webová aplikace

Tématem práce je porovnání existujících platforem pro tvorbu webových aplikací, a proto je potřeba zde uvést, v jakém smyslu bude s tímto pojmem v následujícím textu nakládáno, a co by webová aplikace měla z pohledu autora splňovat.

Na internetu a v literatuře lze nalézt spoustu definic pojmu webová aplikace[1]. Za všechny uveďme dvě. Na wikipedii[7] lze nalézt definici: „Webová aplikace je aplikace doručená uživateli z webového serveru pomocí služby WWW.“ Druhá definice[12] nám říká: „Webová aplikace je uložena na serveru a doručována uživatelům přes internet.“

Zajímavá je také myšlenka z knihy Jeffa Prosise[3]: „V srdci téměř každé skutečné webové aplikace je nějaký formulář HTML.“

Práce bude tedy vycházet z definice, která splňuje následující body:

- je založena na modelu klient/server
- pro komunikaci mezi klientem a serverem využívá protokol HTTP
- rozhraním pro koncového uživatele je webový prohlížeč
- požadavky klienta zpracovává webový server
- obsahuje formuláře pro odesílání informací uživatelem

[1] Pokud bude v textu mluveno o aplikaci, budu tím stále myšlena webová aplikace. Toto se netýká převzatých definic

3.1.2 Protokol HTTP

Přesnou definici protokolu přenosu hypertextu HTTP (*Hypertext Transfer Protocol*) lze nalézt v RFC 2616[15]. Jedná se o protokol, který definuje rozhraní komunikace mezi webovými servery a klienty (většinou reprezentovanými jako webové prohlížeče). Tento protokol je čistě textový a obvykle je vysílán přes TCP spojení přes port 80.

Když uživatel zažádá o zobrazení internetové stránky, tak dojde k otevření soketového připojení k serveru a odvysílá se požadavek HTTP jehož hlavička obsahuje následující informace:

```
GET /priklad.html HTTP/1.1
Host: www.offlajn.net
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; cs; rv:1.8.1.14)
Accept: */*
Accept-Language: cs,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: windows-1250,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
[prázdný řádek]
```

Server požadavek zpracuje a vrátí klientovi odpověď, která může vypadat takto:

```
HTTP/1.1 200 OK
Date: Tue, 06 May 2008 13:54:24 GMT
Server: Apache
Last-Modified: Tue, 06 May 2008 13:54:08 GMT
ETag: "42886eef-33-44c902e9ec000"
Accept-Ranges: bytes
Content-Length: 51
Connection: close
Content-Type: text/html
[prázdný řádek]
<html>
<body>
Ahoj
</body>
</html>
```

Po přijetí odpovědi prohlížeč analyzuje kód HTML vrácený webovým serverem a zobrazí výslednou webovou stránku.

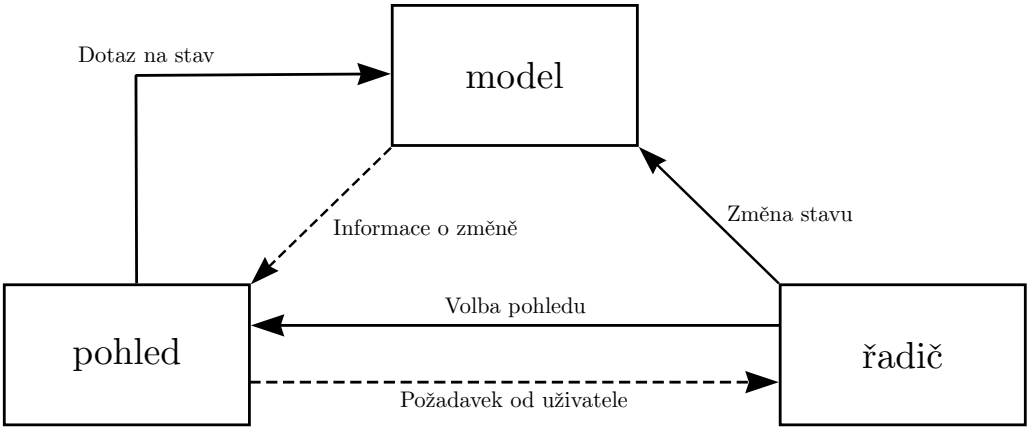
3.1.3 Relační databáze

Nejdříve uveďme definici pro pojem databáze[11]: „Databáze je uspořádaná množina informací (dat) uložená na paměťovém médiu.“ Relační databáze je pak databáze, která je postavena na relačním modelu. To v praxi znamená, že všechna data uložená v databázi jsou uspořádaná do jednotné struktury. Struktura relační databáze se skládá z relací (tabulek), záznamů (řádek tabulky) a atributů (sloupec tabulky).

Pro komunikaci s relační databází je v dnešní době nejčastěji využíván jazyk SQL (*Structured Query Language* = strukturovaný dotazovací jazyk). Tento jazyk podporují téměř všechny dostupné databázové systémy. Tato práce nebude popisovat typické databázové systémy, jako MySQL pro PHP a Microsoft SQL Server pro .NET, ale zaměřím se na PostgreSQL[2]. K popisu PostgreSQL se však blíže dostaneme až v kapitole 5.

3.1.4 Architektura MVC (Model-View-Controller)

Jedná se o softwarovou architekturu, která rozděluje datový model, uživatelské rozhraní a řídící logiku aplikace do tří nezávislých komponent. Rozdělením dojde k tomu, že změna jedné komponenty má minimální vliv na funkci zbylých komponent. V dnešní době se tato architektura využívá především u webových aplikací a to je taky důvod, proč se o ní zmiňujeme.



Obrázek 3.1: Architektura MVC

Na obrázku 3.1 lze vidět závislosti jednotlivých komponent. Silná čára představuje volání metody a přerušovaná čára představuje událost.

- **Model (model)** reprezentuje data nad nimiž aplikace pracuje (u webových aplikací nejčastěji databáze). Model se nestará o to, jak jsou data zobrazována. Proto má pouze minimální závislost na pohledu a posílá mu pouze informace o změnách.
- **View (pohled)** převádí data z modelu do uživatelského rozhraní. Pohledy neobsahují žádná data ani informace o tom, jak s daty pracovat. Obsahují pouze informace o tom, jak data zobrazovat.
- **Controller (řadič)** stojí mezi modelem a pohledem a zpracovává požadavky od uživatele, které jsou vyvolávávány komponenty pohledu (např. odesláním formuláře pomocí tlačítka). Řadič požadavek zpracuje tím způsobem, že přistoupí k modelu a zaktualizuje ho na základě provedené uživatelské akce (např. vloží komentář ke článku).

Bližší informace lze nalézt například na stránkách *O. Čady* [14].

3.1.5 XHTML

XHTML[16] (*eXtensible HyperText Markup Language*) je značkovací jazyk, který se využívá pro tvorbu hypertextových dokumentů v prostředí internetu a je založen na XML. Jazyk XHTML je následníkem staršího HTML, jehož vývoj byl ukončen (i když existuje skupina, která se pokouší o vytvoření nové verze HTML 5[13]). XHTML je založen na univerzálním značkovacím jazyce SGML (*Standard Generalized Markup Language*) a využívá tagy ke strukturování textu (např. nadpisy, odstavce, tabulky).

U XHTML 1.0 Strict se oproti předcházejícím specifikacím a hlavně oproti HTML musí dodržovat přísnější pravidla. Mezi tyto pravidla patří například:

- Všechny tagy a atributy se píší malými písmeny.
- Všechny tagy musí být párové a každý tag musí být uzavřen (např. pro nový řádek nelze použít `<br>`, ale musí se použít uzavřený tag `<br />` nebo párový tag `<br></br>`).
- Hodnoty atributů musí být v uvozovkách.
- Tagy se nesmí křížit. (např. `<p>První odstavec<p>Druhý odstavec</p></p>` musí být přepsáno na `<p>První odstavec</p><p>Druhý odstavec</p>`)
- Nepoužívají se formátovací tagy jako `<font>`, `<b>`, `<i>`, `<u>`. K formátování se využívá CSS.

V praktické části práce se autor snaží dodržovat standard XHTML 1.0 Strict

3.1.6 CSS

Kaskádové styly CSS[18] (*Cascading Style Sheets*) jsou kolekcí metod pro úpravu vzhledu webových stránek. Oproti formátovacím tagům HTML poskytuje CSS širší možnosti formátování vzhledu a rozmístění jednotlivých prvků dokumentu.

Mezi hlavní výhody patří možnost mít definován styl pro webové stránky v jednom souboru. Změnou tohoto souboru pak dojde ke změně vzhledu všech stránek. Díky CSS lze dosáhnout menší datové velikosti stránek a tím i k rychlejšímu načítání.

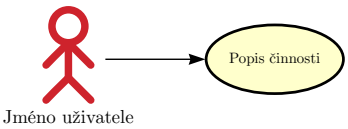
CSS má však i nevýhodu, a tou je omezená podpora u některých prohlížečů, které přesně nedodržují standard pro reprezentaci stylů. Mezi tyto prohlížeče patří hlavně Microsoft Internet Explorer 6.

3.1.7 UML

UML (*Unified Modeling Language*) [1] je modelovací jazyk, který se v softwarovém inženýrství využívá pro vizualizaci, specifikaci, návrh a dokumentaci programových systémů. Jazyk UML dovoluje modelovat jednoduché i složité aplikace pomocí stejné formální syntaxe, a tak umožnuje uživateli sdílet své práce s ostatními. Navržené aplikace jsou dobře pochopitelné i pro zadavatele aplikace a umožnují kvalitní vyjasnění požadavků na vytvářený systém ještě před zahájením práce na vývoji systému.

Use case diagramy

jsou v práci využívány pouze Use case diagramy[2], které slouží ke grafické prezentaci chování systému z pohledu uživatele.



Obrázek 3.2: Jednoduchý Use case diagram

Na obrázku 3.2 můžeme vidět jednoduchý Use case diagram. Postava představuje aktéra systému (aktérem může být osoba i systém) a elipsa znázorňuje případ užití.

Kromě vztahů mezi případy užití a aktéry můžeme identifikovat také několik vztahů mezi případy užití samotnými.

Z těchto vztahů jsou v práci používány následující dva:

1. Relace `<<include>>` vyčleňuje stejné chování ze dvou a více případů užití do samostatného případu užití. Základní případ užití není soběstačný.
2. Relace `<<extend>>` přidává k základnímu případu užití nové, rozšiřující chování. Základní případ užití je zcela soběstačný.

Popis případu užití

Popis případu užití úzce souvisí s Use case diagramy. Jedná se o detailní popis jednotlivých funkcí systému nejčastěji pomocí tabulkově-textové specifikace. Pro popis případů užití je v práci používán formát, který můžeme vidět v následující tabulce.

Název případu užití	**Nějaká činnost**
Identifikace případu užití	UC00
Aktéři	Uživatel
Vstupní podmínky	–
Kroky případu užití	1. První krok
	2. Další krok
	...
Výstupní podmínky	–

3.1.8 ER diagramy

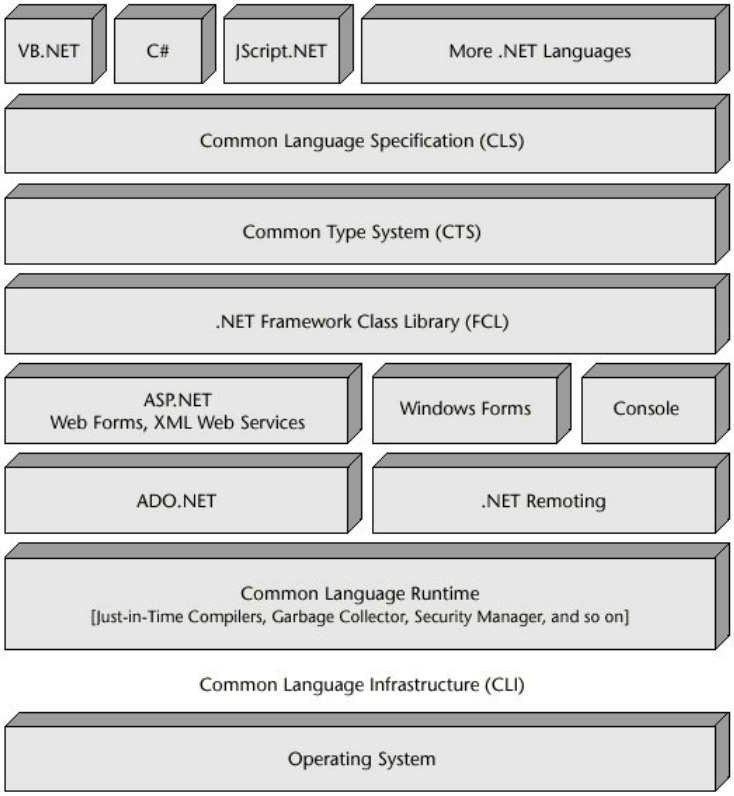
ER diagramy (*Entity Relationship Diagram*) slouží ke grafickému modelování entit, vztahů a omezení pro data navrhované aplikace. Jednoduchý ER diagram můžeme vidět na obrázku 3.3.

[2]V textu je používán anglický název místo českého - diagramy případů použití.

Obrázek 3.3: Jednoduchý ER diagram

3.2 .NET

Microsoft .NET [3] [22] je platforma pro vytváření a provozování aplikací pro internet, Windows i Pocket PC. Základním prvkem této platformy je .NET Framework, který představuje prostředí potřebné pro běh aplikací a nabízí potřebné knihovny.



Obrázek 3.4: Architektura .NET Framework [20]

3.2.1 .NET Framework

Na obrázku 3.4 je znázorněna architektura Microsoft .NET Framework. Hlavními komponenty tohoto frameworku jsou společný jazykový běhový modul (*Common Language Runtime* - CLR) a knihovna tříd rámce .NET (*Framework Class Library* - FCL).

3.2.2 ASP.NET

ASP.NET je nástupcem starší technologie Aktivních serverových stránek (*Active Server Pages*). Nejedná se však o vylepšení původního ASP, ale o zcela novou technologii díky začlenění ASP.NET v .NET Frameworku.

3.2.3 Výhody

Programovací jazyky

Programování aplikací v ASP.NET dovoluje použití velkého množství programovacích jazyků. Microsoft nabízí kompilátory pro pět jazyků: C#, J#, C++, Visual Basic a JScript. Ostatní společnosti nabízí kompilátory i pro další jazyky (Perl, Python, Ruby a další).

3.2.4 Nevýhody

Vyšší nároky na znalosti uživatele

Programování pomocí ASP.NET klade vyšší požadavky na znalosti uživatele. Uživatel musí mít základní znalosti o objektově orientovaném programování a musí se naučit používat technologii .NET.

Dostupnost

Aplikace v ASP.NET lze vyvíjet díky dostupným nástrojům zdarma, avšak plné využití funkcí poskytuje až Visual Studio, které je placené.

Dostupnost hostingů, které dovolují provoz aplikace napsané v ASP.NET, je značně menší, než u ostatních platforem. Když už najdeme nějaký dobrý hosting, který podporu .NET má, tak je jeho cena velmi vysoká.

3.3 CGI skripty

CGI (*Common Gateway Interface*) je rozhraní zajišťující komunikaci mezi aplikacemi a webovým serverem. CGI skripty jsou pak aplikace a skripty, které se spouští na straně serveru, zpracovávají informace a požadavky od klienta a vrací odpovědi v požadovaném formátu pro webové prohlížeče.

Rozhraní CGI není závislé na programovacím jazyce, a tak lze pro tvorbu skriptů využít velké množství skriptovacích a programovacích jazyků. Mezi nejrozšířenější patří Perl, Python, C, C++ a v poslední době se mezi ně dostává také Ruby. Toto rozhraní také není závislé na typu webového serveru ani operačním systému.

Obrázek 3.5: Princip obsloužení požadavku CGI skriptem

Na obrázku 3.5 lze vidět, jak funguje obsluha požadavku pomocí CGI skriptu. Uživatel zašle požadavek na zobrazení stránky. Server přijme požadavek a předá parametry CGI skriptu, který je zpracuje a vrátí výstup aplikace. Výstup skriptu vrátí webový server klientovi jako odpověď v podobě HTML kódu.

3.3.1 FastCGI

Hlavním problémem CGI skriptů je nutnost spouštět skript při každém požadavku znovu. Tím dochází k větší časové náročnosti a delším odezvám při zpracovávání požadavků. Při každém spuštění tak skript nahrává konfigurační soubory, přistupuje k databázi a nebo vyhledává informace v lokálním systému. S řešením přišla společnost Open Market, Inc., která vyvinula rozhraní FastCGI[19].

Při použití FastCGI dojde k tomu, že se aplikace spustí na serveru pouze jednou, a to nezávisle na jakémkoliv požadavku. Poté čeká aplikace na požadavek od serveru. Jakmile ho dostane, tak naváže s webovým serverem spojení a zahájí datovou komunikaci.

3.3.2 Výhody

Dostupnost

FastCGI má v dnešní době podporu již ve všech jazycích, které se pro psaní CGI skriptů nejčastěji používají. Stějne tak většina serverů obsahuje moduly pro podporu CGI i FastCGI. Obě rozhraní jsou volně přístupná na internetu, takže není problém je zprovoznit na vlastním serveru.

Díky tomu, že lze pro psaní CGI skriptů použít téměř libovolný skriptovací nebo programovací jazyk, tak je pro programátora lehké takovouto aplikaci vytvořit a provozovat.

3.3.3 Nevýhody

Rychlost

Rychlost velmi závisí na rozsahu aplikace, kterou se snažíme implementovat, a také na použitém jazyce. Odezva skriptů pracujících na CGI je často velmi pomalá. Při použití rozhraní FastCGI je již rychlost trochu lepší, ale v porovnání s jinými nástroji stále není tak dobrá. Velmi výstižně se k rychlosti vyjádřil na svých stránkách J. Kosek[17]. Má zde uvedeno: „Použití klasických CGI skriptů je výhodné v případech, kdy toho potřebujeme hodně spočítat a málo zobrazit.“

Složitost

Napsat rozsáhlejší webovou aplikaci pomocí klasických CGI skriptů je velmi náročné a jiné nástroje se pro tvorbu rozsáhlých aplikací hodí lépe.

3.3.4 Ruby on Rails

Řešením složitosti CGI aplikací může být použití různých frameworků. Mezi tyto frameworky patří i Ruby on Rails [9], kterým se práce zabývá později (5.1.3), protože právě v Ruby on Rails měl autor za úkol vytvořit jednu z implementací své aplikace.

Tento framework je napsán pomocí skriptovacího jazyku Ruby, který je díky jednoduché syntaxi poměrně snadný k naučení. Framework staví na myšlence, že je zbytečné konfigurovat to, co se konfigurovat nemusí. Programátor by měl pouze měnit věci, kde se nastavení liší od běžné konfigurace.

Aplikace napsané pomocí Ruby on Rails jsou automaticky postaveny na architektuře MVC. U ostatních porovnávaných platforem je tato architektura podporována nedostatečne a nebo vůbec.

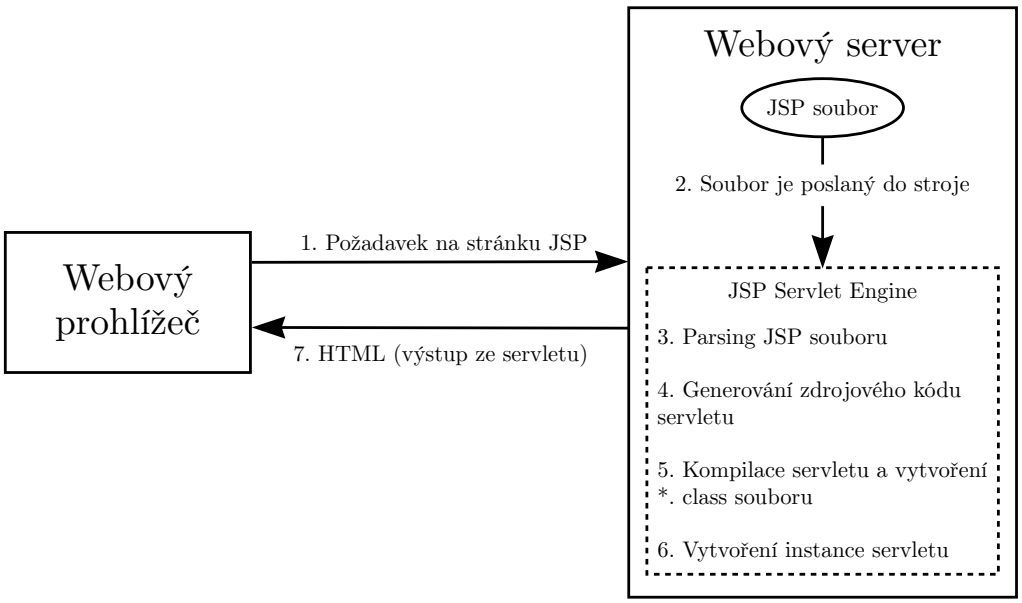
3.4 JSP

JSP (*Java Server Pages*) je nástroj vyvinutý firmou Sun Microsystems za účelem vývoje aplikací na straně serveru. Pro tvorbu webových aplikací se využívá programovací jazyk Java.

Do zdrojového HTML kódu stránky se přímo zapisuje kód jazyka Java, jako je tomu u klasických skriptovacích jazyků.

Např.

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



Obrázek 3.6: Princip činnosti JSP

Zdrojový kód stránky je překládán na Java servlet[3] a následně kompilován. Výsledkem je servlet, jenž generuje HTML kód - ten je zasílán zpět klientovi. Detailní princip fungování je vyjádřen na obrázku 3.6.

[3]Servlety jsou programy psané v Jave a spouštěné na straně serveru podobně jako CGI skripty.

1. Uživatel zažádá o webovou stránku, která byla vytvořena pomocí JSP. Klient zašle dotaz na server.
2. Webový server zjistí, že požadovaná stránka je speciální, protože má koncovku `*.jsp` a přesměruje ho do JSP Servlet Engine
3. V případě, že je tento soubor volaný prvně, tak ho Servlet Engine zpracuje. Jestliže už byl soubor někdy požadován, tak pokračuje bodem 6.
4. Ze souboru se vygeneruje servlet. Veškeré statické HTML je uloženo v `out.println()` příkazech.
5. Zdrojový kód servletu se zkompiluje a vytvoří se `*.class` soubor.
6. Vytvoří se instance servletu a zavolají se metody `init()` a `service()`.
7. Výstupem ze servletu je HTML, které je zasláno klientovi jako odpověď.

3.4.1 Výhody

Oblíbenost jazyka

Jazyk Java je v dnešní době velmi rozšířený a oblíbený. Java poskytuje třídy pro práci se sítí, databázemi, e-maily a spoustu dalších tříd. VVětšina uživatelů se s ním již setkala, a tak se při programování webových aplikací rozhodne právě pro JSP.

Dostupnost

Jazyk Java je přenositelný a uživatel není limitován použitým operačním systémem nebo serverem. JSP lze stáhnout na internetu a nainstalovat jeho podporu na domácí server. Pokud chce uživatel umístit svoji aplikaci na internet, existuje spousta dostupných hostingů, které podporu JSP poskytují.

Objektový přístup

Jazyk Java je objektově orientovaný programovací jazyk a tak i JSP využívají objektový přístup při tvorbě webových aplikací.

3.5 PHP

Název PHP [21] je rekurzivní zkratka pro *PHP: Hypertext Preprocessor* (kde původní význam zkratky PHP je *Personal Home Pages*). Jedná se o skriptovací programovací jazyk, který je určený především pro tvorbu dynamických webových stránek, ale lze ho využít i pro vytváření skriptů.

Jazyk PHP vychází ze skriptovacího jazyka Perl a syntakticky je asi nejvíce podobný jazyku C. Zdrojový kód se nejčastěji zapisuje přímo do zdrojového kódu HTML dokumentu, podobně jako je tomu u ostatních skriptovacích jazyků.

Např.

```
<html>
<body>
```

```
Dnešní datum je: <?php echo date("d.m.Y", time());?>
</body>
</html>
```

3.5.1 Výhody

Dostupnost

PHP i server, na kterém lze PHP provozovat jsou volně ke stažení na internetu. Instalace a zprovoznění prostředí pro provoz PHP aplikací jsou velmi jednoduché. Navíc, pokud uživatel nemá zkušenosti s konfigurací serveru, může si stáhnout některý z balíčků. Ty často obsahují server, PHP a databázový systém (Např. WampServer [8]).

Pokud uživatel nechce aplikaci napsanou v PHP provozovat pouze na lokálním serveru, ale chce ji umístit na internet, tak nemá téměř žádný problém, protože podporu PHP nabízí většina hostingů i u nejlevnějších programů.

Jednoduchost

Naučit se PHP je velmi jednoduché díky dobré dokumentaci a spoustě návodů dostupných na internetu. Navíc existuje spousta uživatelů, kteří poskytují hotová řešení a návody ostatním uživatelům.

3.5.2 Nevýhody

„Spaghetti code“

PHP má nedostatečnou podporu oddělení prezentační a řídící logiky. Vkládáním PHP kódu přímo mezi kód HTML dochází k horší čitelnosti zdrojových textů („Spaghetti code“).

Špatný systém pojmenování funkcí

PHP obsahuje velké množství funkcí, ale tyto funkce nedodržují žádná pravidla pro pojmenování. Některé funkce jsou pojmenovány celým názvem, některé pouze zkratkou. Někdy je v názvu použito podtržítko a někdy ne (Například funkce pro práce s řetězci `str_split` a `strcmp`). Pro uživatele je tak někdy obtížné nalézt funkci, kterou potřebuje.

Nedostatečný objektový model

Knihovna základních funkcí PHP je procedurální a většina funkcí nevyužívá objekty. PHP obsahuje i podporu objektově orientovaného programování, ale v porovnání s ostatními nástroji je tato podpora velmi špatná.

3.6 Shrnutí

Každý z nástrojů má své přednosti a výhody. Pro uživatele, který teprve začíná s tvorbou webových aplikací, může být těžké se rozhodnout, pomocí kterého nástroje svoji aplikaci napíše.

Úplný začátečník se nejčastěji rozhodne pro PHP, protože se o tomto nástroji nejvíce mluví a je na českém internetu nejvíce rozšířený. Během chvíle se může zaregistrovat na nějakém free hostingu a začít vyvíjet svoji aplikaci.

Uživatel, který má zkušenosti s jazykem Java, se většinou rozhodne pro JSP, protože už má znalosti syntaxe tohoto jazyka.

CGI skripty se již v dnešní době příliš nevyužívají, a tak si uživatel vybere spíše některý z frameworků, které využívají rozhraní FastCGI.

Platformu ASP.NET si vybere nejčastěji uživatel, který již má zkušenosti s vývojem aplikací (například pomocí PHP) a chce se naučit novým věcem.

3.7 Výběr platforem pro implementaci

Jako první nástroj pro implementaci byl zvolen jazyk PHP, jelikož s ním má autor práce již dřívější osobní zkušenosti.

Jako druhý nástroj byl vybrán ASP.NET C#, protože se jedná o platformu s PHP často srovnávanou.

Jako třetí platforma byla zvolena Ruby on Rails, a to především kvůli zaujetí, které u autora vyvolala.

Kapitola 4

Návrh webové aplikace

Pro aplikaci byl vymyšlen pracovní název *mpBlog*, který je zkratkou pro multi-platformní blog.

4.1 Specifikace

Webová aplikace, kterou měl autor za úkol navrhnout a vytvořit je zaměřena na demonstraci jednotlivých platforem a rozdílů práce s nimi. Nemá za cíl ohromit množstvím svých funkcí, pro které jsou většinou použity stejné funkce a postupy programování. [1].

Aplikace představuje jednoduchý systém pro publikování článků na internetu. Neregistrovaný uživatel může využívat pouze základní funkce. Po vyplnění požadovaných údajů (jméno, heslo, e-mail) dojde k registraci uživatele do systému. Po registraci se uživatel může přihlásit a využívat rozšířené funkce aplikace. Mezi hlavní funkce patří: práce s články, práce s komentáři, hodnocení článků a správa uživatelského účtu.

4.2 Funkční požadavky

1. Správa implementací

   Webová aplikace musí umožnovat přepínání mezi jednotlivými implementacemi.

2. Správa uživatelů

   V systému jsou 3 role uživatelů. Každá role má definována práva a funkce, které může využívat. Jednotlivá práva a funkce jsou vyznačeny u každého požadavku zvlášť.

   - Neregistrovaný uživatel (N)
     Uživatel, který vstoupí na stránky aplikace a není přihlášen. Má omezené funkce.
   - Registrovaný uživatel (R)
     Uživatel, který je zaregistrovaný v aplikaci a přihlásí se. Po přihlášení má registrovaný uživatel možnost využívat veškeré funkce systému kromě funkcí dostupných pouze pro administrátora.

[1] Např. aplikace neobsahuje diskuzi, protože ta by byla naprogramováná téměř stějně jako komentáře u jednotlivých článků.

- Administrátor (A)
  Administrátor je správce systému, který má stejná práva a funkce jako registrovaný uživatel a k tomu má speciální práva pro správu aplikace.

Správa uživatelů obsahuje následující funkce:

- Registrace (N)
  - Pro registraci je potřeba zadat tyto údaje (uživatelské jméno, heslo, e-mail).
  - Pro větší bezpečnost aplikace bude heslo šifrováno.
- Přihlášení (R, A)
- Odhlášení (R, A)
- Zobrazení údajů o uživateli (N, R, A)
- Editace údajů o uživateli (R - pouze vlastní údaje, A - údaje u všech registrovaných uživatelů)
- Smazání uživatele (A)

3. Správa článků

   Tento požadavek určuje funkce pro práci s články.

   - Vložení článku (R, A)
   - Editace článku (R - pouze vlastní články, A - všechny články)
   - Smazání článku (R - pouze vlastní články, A - všechny články)
   - Zobrazení článku (N, R, A)

4. Správa komentářů

   Komentáře slouží k diskuzi o jednotlivých článcích a budou zobrazovány spolu s článkem.

   Funkce pro práci s komentáři jsou následující:

   - Vložení komentáře (N, R, A)
   - Editace komentáře (R - pouze vlastní komentáře, A - všechny komentáře)
   - Smazání komentáře (R - pouze vlastní komentáře, A - všechny komentáře)

5. Hodnocení článků

   Každý článek bude moci být hodnocen (R, A) a bude zde možnost měnit hodnocení. (R, A)

6. Logování systému

   Systém bude vytvářet záznamy v databázi, kde budou uvedeny jednotlivé akce uživatelů.

   Každý záznam bude obsahovat datum, IP adresu, druh události a ID uživatele (jestliže se bude jednat o registrovaného uživatele, nebo administrátora).

   Zaznamenávat se budou následující události:

   - Registrace (N)

- Příhlášení (R, A)
- Odhlášení (R, A)
- Zobrazení článku (N, R, A)
- Přidání a editace článku (R, A)
- Přidání komentáře (N, R, A) a editace komentáře (R, A)

Uživatelé si budou moci jednotlivé záznamy zobrazit:

- Registrace (A - všechny registrace)
- Příhlášení (R - pouze svoje přihlášení, A - přihlášení všech uživatelů)
- Odhlášení (R - pouze svoje odhlášení, A - odhlášení všech uživatelů)
- Zobrazení článku (R - kdo četl jeho články a jaké články četl, A - u všech uživatelů, kdo a jaké články četl)
- Přidání a editace článku (R - pouze údaje o svých článcích, A - údaje u všech článků)
- Přidání komentáře a editace komentáře (R - pouze údaje o svých komentářích, A - údaje o všech komentářích)

4.3 Nefunkční požadavky

1. Implementace

   Webová aplikace bude implementována pomocí 3 platforem. Tyto platformy jsou ASP.NET C#, PHP a Ruby on Rails.

2. Jednotná data

   Všechny tři implementace musí mít společná data, která budou uložena pomocí databázového systému PostgreSQL.

3. Jednotný výstup

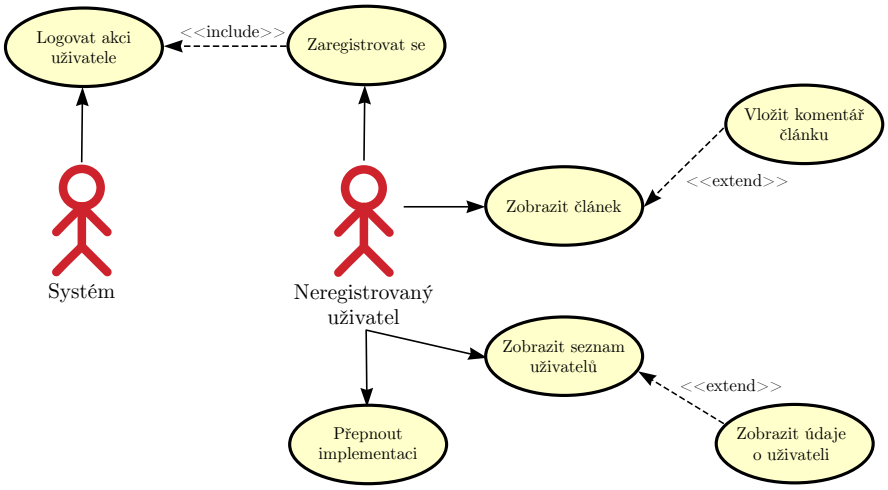
   Výstup všech tří implementací musí být stejný. Tento výstup bude ve formátu *XHTML 1.0 Strict* a bude formátován pomocí *CSS*.

4.4 Use case diagram

Na základě specifikace a zadaných požadavků byly vytvořeny následující Use case diagramy. První zachycuje procesy pro Neregistrovaného uživatele (obrázek 4.1). Na druhém Use case diagramu můžeme vidět procesy pro Registrovaného uživatele a Administrátora (obrázek 4.2). U obou diagramů je navíc aktér, který představuje Systém.

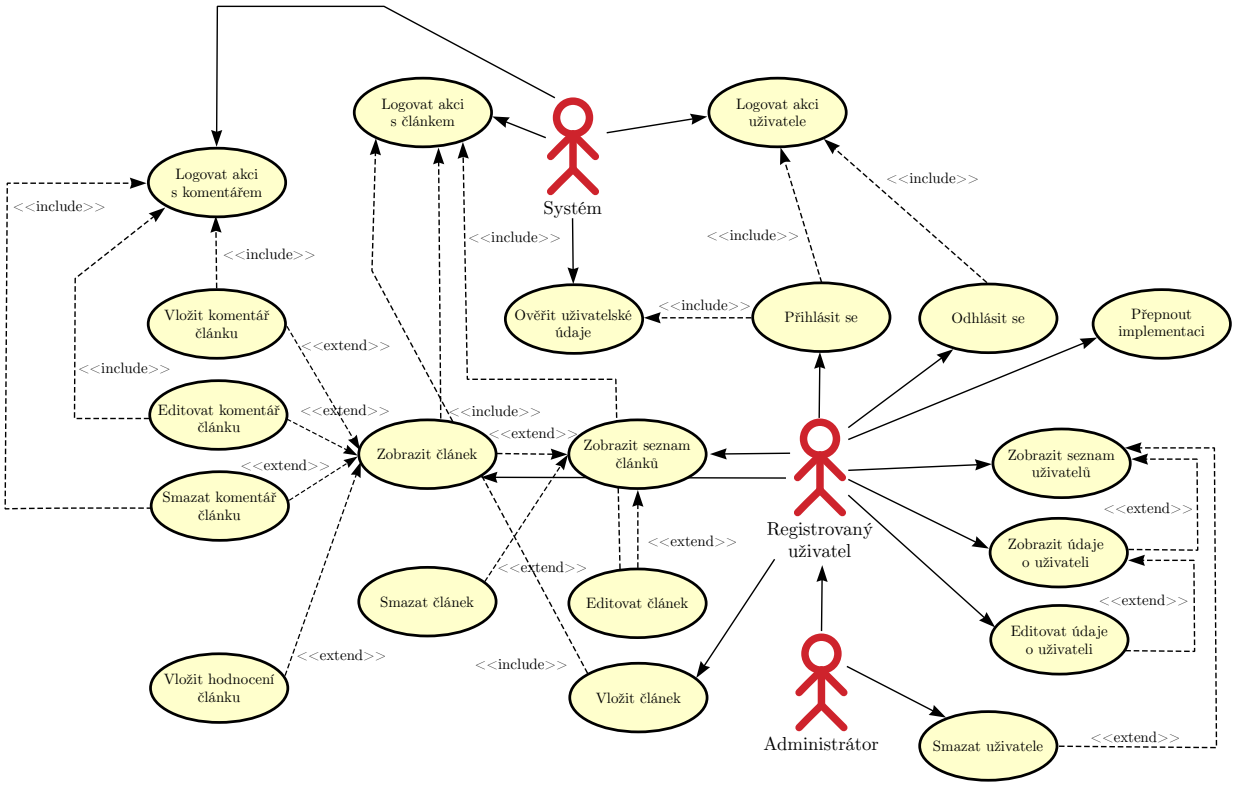
4.5 Případy užití

Pro lepší představu o fungování výsledného systému jsou na tomto místě uvedeny příklady popisu případů užití. Případ užití UC01 představuje proces *Registrace uživatele* a je v něm použit odkaz na vazbu typu `<<include>>`. Pomocí této vazby je odkazováno na případ užití UC02.



Obrázek 4.1: Use case diagram pro Neregistrovaného uživatele

Název případu užití	**Registrace uživatele**
Identifikace případu užití	UC01
Aktéři	Neregistrovaný uživatel, Systém
Vstupní podmínky	Uživatel je neregistrovaný.
Kroky případu užití	1. Uživatel otevře stránku *Zaregistrovat se*. 2. Systém zobrazí formulář pro registraci uživatele. 3. Uživatel vyplní údaje ve formuláři (uživatelské jméno, heslo, kontrola hesla a e-mail) a formulář odešle stisknutím tlačítka *Zaregistrovat se*. 4. Systém ověří správnost zadaných údajů. Data jsou v pořádku. 5. Systém uloží údaje o uživateli do databáze a potvrdí úspěšné uložení do databáze informativním hlášením. 5a1. Systém zobrazí formulář pro registraci a upozorní na položky, které jsou špatně vyplněny. 5a2. Uživatel opraví údaje a formulář odešle stisknutím tlačítka *Zaregistrovat se*. 5a3. Systém uloží údaje o uživateli do databáze a potvrdí úspěšné uložení do databáze informativním hlášením. 6. Systém uloží do databáze log záznam o registraci uživatele - viz případ užití Logovat akci uživatele.
Výstupní podmínky	Uživatel je zaregistrován.



Obrázek 4.2: Use case diagram pro ostatní uživatele

Název případu užití	**Logovat akci uživatele**
Identifikace případu užití	UC02
Aktéři	Systém
Vstupní podmínky	Uživatel spustí akci, která vyvolá požadavek pro zápis log záznamu.
Kroky případu užití	1. Systém zjistí o jaký požadavek uživatele se jedná. Může jít o registraci, přihlášení nebo odhlášení. 2. Systém uloží do databáze log záznam o akci uživatele.
Výstupní podmínky	Je vytvořen log záznam o akci uživatele.

Na následujících případech užití UC03a a UC03b je demonstrováno spuštění stejného procesu jinými aktéry.

Název případu užití	**Zobrazit seznam uživatelů**
Identifikace případu užití	UC03a
Aktéři	Neregistrovaný uživatel, Registrovaný uživatel, Systém
Vstupní podmínky	–
Kroky případu užití	1. Uživatel otevře stránku *Seznam uživatelů*. 2. Systém zobrazí seznam s uživatelskými jmény a e-maily všech uživatelů. **Bod rozšíření: *Zobrazit údaje o uživateli***
Výstupní podmínky	–

Název případu užití	**Zobrazit seznam uživatelů**
Identifikace případu užití	UC03b
Aktéři	Administrátor, Systém
Vstupní podmínky	–
Kroky případu užití	1. Uživatel otevře stránku *Seznam uživatelů*. 2. Systém zobrazí seznam s uživatelskými jmény a e-maily všech uživatelů. **Bod rozšíření: *Zobrazit údaje o uživateli*** **Bod rozšíření: *Smazat uživatele***
Výstupní podmínky	–

4.6 ER diagram

Komentáře k jednotlivým tabulkám, které jsou zobrazeny na obrázku 4.3:

- **users** - Uživatelé

  Každý uživatel má jedinečné číslo `id`, vlastní uživatelské jméno `email` a po registraci mu je přiřazena funkce 'Registrovaný uživatel'.

  Existuje zde uživatel 'Nepřihlášený uživatel', který se nemůže do systému přihlásit, ale je nutné ho mít v databázi, aby nedocházelo k problémům s cizími klíči.

- **articles** - Články

  Každý článek má jedinečné číslo `id`.

- **comments** - Komentáře

  Každý komentář má jedinečné číslo `id` a vztahuje se k jednomu článku `articles_id`.

- **ratings** - Hodnocení

  Každé hodnocení je jedinečné díky kombinaci čísla uživatele `users_id` a čísla článku `articles_id`.

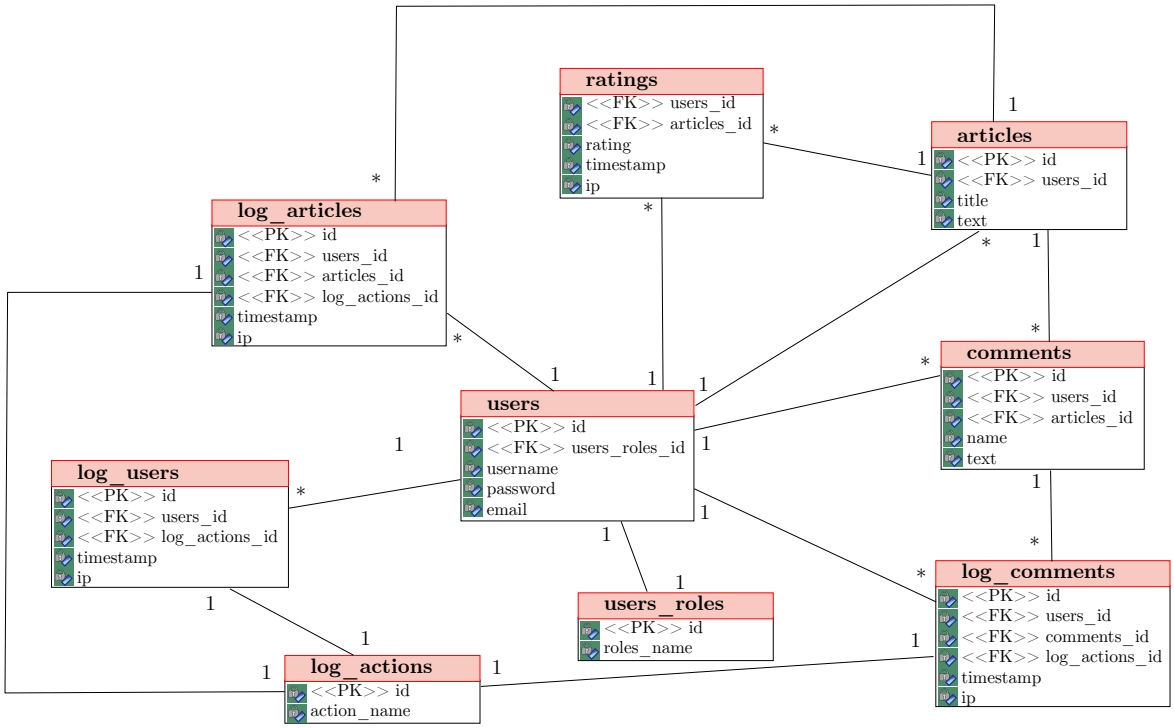
- **users_roles** - Role uživatelů

  Tabulka obsahuje jednotlivé role uživatelů.

- **log_actions** - Popis akcí

  Tabulka obsahuje popis akcí, které se budou přiřazovat ostatním tabulkám.

  Např. `1 - Přihlášení uživatele, 4 - Vložení článku,...`



Obrázek 4.3: ER diagram

- **log_users** - Operace uživatelů

  Do této tabulky se zaznamenávají operace uživatelů (Registrace, Přihlášení, Odhlášení)

- **log_articles** - Operace s články

  Tabulka, která zaznamenává operace s články (Vytvoření, Zobrazení, Editace, Smazání).

- **log_comments** - Operace s komentáři

  Tabulka zaznamenávající operace s komentáři (Vytvoření, Editace, Smazání).

Kapitola 5

Implementace

5.1 Použité nástroje

5.1.1 ASP.NET C#

Serverové ovládací prvky

Jednou z výhod ASP.NET jsou serverové ovládací prvky. Jedná se o třídy .NET Frameworku, které třídy reprezentují vizuální prvky webového formuláře. Tyto prvky automaticky generují kód HTML.

Verze

Pro vývoj aplikace bylo použito prostředí Microsoft Visual Studio 2005, jehož součástí je .NET Framework 2.0.

Server

Aplikace byla testována na ASP.NET Development Server, který je také součástí Visual Studia.

5.1.2 PHP

Objektový přístup

U implementace aplikace v PHP je využíván objektový přístup, stejně jako u zbylých dvou nástrojů.

Verze

Pro implementaci bylo použito PHP verze 5.2.5.

Server

Aplikace byla vyvíjena a testována na lokálním serveru Apache HTTP Server 2.2.6.

5.1.3 Ruby on Rails

Pravidla pojmenování

Ruby on Rails používá pro návrh databáze několik pravidel:

- Pro tabulky se využívají anglické názvy objektů, které reprezentují množným číslem a malými písmeny (např. tabulka pro uživatele je `users`).
- Každá tabulka by měla mít numerický primární klíč pojmenovaný `id`.
- Cizí klíč sestává z názvu tabulky, do níž referuje, v jednotném čísle, podtržítka a id - `user_id`.

Generátory

Generátory slouží k vygenerování základní struktury aplikace. Mezi nejpoužívanější patří `scaffold` generátor. Pomocí následujícího příkazu Ruby on Rails vytvoří strukturu webu, kde je již naprogramováno základní přidáváni, editace a mazání článků.

```
script/generate scaffold article
```

Scaffold generátor je v implementaci využit na několika místech (např. `articles`). V implementaci aplikace jsou však využívány hlavně jednoduché generátory (`controller`, `model`).

Existuje i generátor, který vytvoří základní strukturu pro správu uživatelů `login_generator`. Tento však nebyl v implementaci použit pro svou špatnou kompatibilitu s Ruby on Rails verze 2.0.2.

Dokumentace a zdroje

Ruby on Rails ve verzi 2.0.2 vyšla v prosinci roku 2007 a přinesla hodně změn celého frameworku. Mnoho ukázek práce a publikací o Ruby on Rails, které lze nalézt na internetu, pracuje se starší verzí frameworku. Pro některé problémy, na které autor při tvorbě aplikace narazil, bylo velmi obtížné najít řešení a některé se mu nepodařily vyřešit vůbec.

Verze

Pro implementaci aplikace byl použit jazyk Ruby 1.8.6, framework Rails 2.0.2 a balíčkovací systém Gem 1.1.1.

Server

Jako testovací server byl použit WEBrick 1.3.1, který je součástí Ruby on Rails.

5.1.4 PostgreSQL

Databáze celkem obsahuje 9 tabulek a 8 procedur. Struktura databáze odpovída navrženému ER diagramu (obrázek 4.3).

Procedury

U většiny tabulek je jako primární klíč používána buňka `id`, která má automaticky generovanou hodnotu. PostgreSQL však nepodporuje vlastnost `AUTO_INCREMENT`, kterou autor zná z jiných SQL databází (např. MySQL). Tato vlastnost však může být nahrazena pomocí sekvence.

Příklad použití:

```
CREATE SEQUENCE users_id_seq;
CREATE TABLE "users"(
"id" integer PRIMARY KEY DEFAULT nextval('users_id_seq'),
...
);
```

U některých tabulek nebylo nutné, aby se atribut `id` generoval automaticky, protože jsou tyto tabulky neměnné. Jedná se o tabulky `users_roles` a `log_actions`. Jednotlivé záznamy těchto tabulek jsou vkládány pomocí skriptu, který slouží k vytvoření databáze.

Verze

Pro všechny tři implementace byl využit databázový systém PostgreSQL v8.3.1[10].

Pro správu databáze byl použit program pgAdmin Version 1.8.2, který je součástí instalace PostgreSQL.

5.2 Srovnání

Zde jsou uvedeny postupy k řešení několika problémů, tak jak jsou naimplementovány pomocí jednotlivých nástrojů.

5.2.1 Pohledy

Šablona

Šablona je vytvořena pomocí XHTML 1.0 Strict a CSS. Šablona byla vytvořena a testována v prohlížeči Mozilla Firefox. V tomto prohlížeči se zobrazuje správně. Ostatní prohlížeče mohou šablonu zobrazovat s menšími rozdíly.

Všechny tři aplikace využívají stejnou šablonu a jsou odlišeny pouze barevnými prvky. Původní šablona se nachází na přiloženém CD ve složce \`sablona`\.

ASP.NET

V ASP.NET jsou jako pohled využívány Master pages. Každá master page obsahuje alespoň jeden prvek `contentplaceholder`. Do tohoto prvku je poté vkládán obsah stránek, které danou master page dědí.

Soubor `MasterPage.master`:

```
<%@ Master Language="C#" AutoEventWireup="true"
  CodeFile="MasterPage.master.cs" Inherits="MasterPage"%>
...
```

```
<head runat="server">
  ...
  <title></title>
</head>
...
<asp:contentplaceholder id="contentPlaceHolder" runat="server">
</asp:contentplaceholder>
...
```

Stránka, která dědí master page:

```
<%@ Page Language="C#" MasterPageFile="~/Template/MasterPage.master"
  AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default"
  Title="Název stránky" %>
...
<asp:Content ID="Content1" ContentPlaceHolderID="contentPlaceHolder"
  runat="server">
         <p>Text, který se vkládá na stránku.</p>
</asp:Content>
...
```

Pomocí `Title` je nastaven název stránky, který se vloží mezi tagy `<title>` v master page.

PHP

PHP nemá žádnou podporu pohledů a ani je ničím nenahrazuje. Řešením však může být použití některého ze šablonovacích systému napsaných pomocí PHP. Mezi nejrozšířenější patří Smarty [4].

V implementaci je použita jednoduchá náhrada šablonovacího systému. Jedná se o stránky, kterým se předávají proměnné a generování HTML kódu je řešeno až v těchto souborech.

Ruby on Rails

Ruby on Rails jako jediný nástroj dodržuje architekturu MVC. Pohledy jsou tedy již součástí aplikace a pracuje se s nimi naprosto jednoduše. Pro všechny pohledy lze nastavit společnou šablonu. Tato šablona je umístěna v adresáři `\views\layouts\`.

```
...
<head>
...
  <%= stylesheet_link_tag 'style' %>
  <title>mpBlog &ndash; <%= @title %></title>
<head>
...
<%= @content_for_layout %>
...
```

Pomocí `style_link_tag` určíme, který `*.css` soubor se má využívat. Soubory s kaskádovým stylem se nachází v adresáři `\public\stylesheets\`. Obsah pohledů se automaticky vkládá na místo, které je určeno pomocí prvku `@content_for_layout`.

Pomocí řadiče ještě nastavíme proměnnou, která představuje název stránky.

```
...
def index
  ...
  @title = "Název stránky"
  ...
end
```

5.2.2 Připojení k databázi

ASP.NET

Pro připojení je potřeba nainstalovat connect providera, který se bude starat o předavání dat. Součástí instalace PostgreSQL je i Npgsql 1.0.0, který jednoduše nainstalujeme. K připojení k databázi nám stačí nahrát do adresáře `bin` knihovny `Npgsql.dll` a `Mono.security.dll` a vytvořit spojení.

```
NpgsqlConnection db = new NpgsqlConnection("server=localhost;
 port=5432; user=postgres; password=heslo; Database=mpBlog");
db.Open();
```

PHP

V `php.ini` stačí povolit knihovnu `php_pgsql.dll` a restartovat Apache Server. Poté se již můžeme k databázi připojit pomocí příkazu `pg_connect`.

```
$db = pg_connect("host=localhost port=5432 user=postgres password=heslo
 dbname=mpBlog");
```

Ruby on Rails

Pomocí příkazu `gem install postgres-pr` stáhneme potřebný balíček a nastavíme správné údaje pro připojení k databázi.

soubor `config/database.yml`

```
development:
 adapter: postgresql
 database: mpBlog
 username: postgres
 password: heslo
 host: localhost
 port: 5432
```

5.2.3 Výpis dat z databáze

Pro ukázku si necháme z databáze vypsat id, uživatelské jméno a email všech registrovaných uživatelů a vypíšeme je do tabulky.

Dotaz, který nám vypíše tyto údaje je následující:

```
SELECT id, username, email FROM users WHERE users_roles_id = 2
```

ASP.NET

Pro výpis dat do tabulky v ASP.NET je použit datový prvek DataGrid, což je datově vázaná tabulka. V prvku DataGrid je také implementováno rozhraní pro řazení jednotlivých sloupců a pro stránkování.

```
...
protected void Page_Load(object sender, EventArgs e)
{
  Database db = new Database();

  System.Data.IDbCommand cmd = db.CreateCommand();
  cmd.CommandText = "SELECT id, username, email FROM users
  WHERE users_roles_id = 2";

  System.Data.IDataReader reader = cmd.ExecuteReader();
    grid.DataSource = reader;
    grid.DataBind();
    reader.Close();
    reader = null;

  cmd.Dispose();
  cmd = null;
}
...
<asp:DataGrid id="grid" runat="server">
</asp:DataGrid>
...
```

- Vytvoříme si dotaz, který pošleme databázi.
- Vytvoříme si proměnnou `reader`, která slouží ke čtení dat z databáze.
- Určíme proměnnou `reader` jako zdroj dat pro prvek `DataGrid`

PHP

Pro zpracování dotazu v PHP jsou v implementaci použity příkazy `pg_query()`, `pg_fetch_object()` a `pg_fetch_row()`. Tyto příkazy jsou obsaženy v metodě `query()` třídy `Database`, která je zde v příkladu uvedena.

```
...
$q = "SELECT id, username, email FROM users WHERE users_roles_id = 2";
if($result = $database->query($q))
{
  $text = "";
  foreach($result as $r)
  {
    $text .= "<tr>";
    $text .= "<td>$r->id</td>";
    $text .= "<td>$r->username</td>";
    $text .= "<td>$r->password</td>";
    $text .= "</tr>";
  }
}

echo "<table>";
echo $text;
echo "</table>";
...
```

- Vytvoříme si dotaz, který pošleme databázi.
- Databáze nám vráti pole objektů `$result`.
- Jeden objekt představuje jeden záznam.
- Záznamy vypíšeme.

Ruby on Rails

V Ruby on Rails nejde vypsat tabulku pomocí jednoho souboru, ale musíme upravit řadič, model i pohled.

Řadič `ruby_controller.rb`

```
...
def index
  ...
  @users = User.find(:all, :conditions => "users_roles_id = 2")
  ...
end
```

Model `user.rb`

```
class User < ActiveRecord::Base
belongs_to: users_role

has_many: article
...
end
```

Pohled `index.rhtml`

```
<table>
<% for user in @users %>
  <tr>
  <td><%= user.id %></td>
  <td><%= user.username %></td>
  <td><%= user.email %></td>
  </tr>
<% end %>
</table>
```

- Řadič nám najde požadované záznamy. Pomocí `:conditions` si určíme požadované omezení.
- Model může obsahovat pouze mezitabulkové vazby.
- Pohled obsahuje šablonu pro vypsání jednotlivých atributů.

5.2.4 Vytvoření formuláře

ASP.NET

V implementaci pomocí ASP.NET jsou formuláře tvořeny pomocí serverových ovládacích prvků `<asp:TextBox>` a `<asp:Button>`.

```
...
<form runat="server" id="register">
  ...
  <asp:TextBox id="username" runat="server">
  </asp:TextBox>
  ...
  <asp:TextBox ID="password" runat="server" TextMode="Password">
  </asp:TextBox>
  ...
  <asp:Button id="btnForm" Text="Zaregistrovat se" runat="server">
  </asp:Button>
  ...
</form>
...
```

PHP

Tradičně jsou formuláře v PHP vypisovány pomocí HTML kódu. V implementaci je pro jednodušší práci použito pole, které se předává speciálnímu souboru. Tento soubor je částečnou náhradou za šablonovací systém.

Soubor s inicializací formuláře:

```
...
$form_inputs = array(
  array("title" => "Uživatelské jméno", "id" => "username",
    "type" => "text", "value" => $username)
  , array("title" => "Heslo", "id" => "password",
    "type" => "password", "value" => $password)
  , array("title" => "", "id" => "btnForm",
    "type" => "submit", "value" => "Zaregistrovat se")
  ...
);
...
```

Soubor `form.tpl.php`:

```
...
foreach($form_inputs as $f)
{
  if(empty($f['title'])) $f['title'] = " ";
  echo "<td><label for=\"".$f['id']."\">".$f['title']."</label></td>\n";
  echo "<td><input id=\"".$f['id']."\" name=\"".$f['id']."\"
    type=\"".$f['type']."\" value=\"".$f['value']."\" /></td>\n";
}
...
```

Ruby on Rails

Nejdříve musíme v řadiči vytvořit proměnnou `@user`. Inputy formuláře se generují pomocí `text_field`, `password_field` a `submit_tag`.

Řadič:

```
...
def register
  @user = User.new()
end
...
```

Názvy jednotlivých vstupů musí odpovídat názvům atributů tabulky v databázi. Jestliže chceme použít pomocné proměnné (například pro kontrolu hesla), je potřeba je nadefinovat v modelu.

Model:

```
class User < ActiveRecord::Base
  ...
  attr_accessor :confirm_password
  ...
end
```

Pohled:

```
<%= form_tag :action => "create" %>
  ...
 <%= text_field 'user', 'username' %>
 ...
 <%= password_field 'user', 'password' %>
 ...
 <%= submit_tag 'Zaregistrovat se'%>
 ...
</form>
```

5.2.5 Zpracování formuláře

ASP.NET

Pro kontrolu údajů ve formuláři se v ASP.NET používají serverové ovládací prvky `<asp:ValidationSummary>`, `<asp:RequiredFieldValidator>`, `<asp:CompareValidator>` a `<asp:RegularExpressionValidator>`. První se umístí na začátek formuláře a vypisuje se do něj souhrn chyb. Další uvedené prvky se umístí nejčastěji za pole vstupu a v případě chyby se vypíše hvězdička.

```
...
<form runat="server" id="register">
  <asp:ValidationSummary ID="ValidationSummary1" runat="server" />
  ...
  <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
    Display="Dynamic" ErrorMessage="Nebylo vyplněno uživatelské jméno!"
    ControlToValidate="username">*</asp:RequiredFieldValidator>
  ...
  <asp:CompareValidator ID="CompareValidator1" Display="Dynamic"
    ErrorMessage="Hesla se neshodují!" ControlToValidate="password"
    ControlToCompare="confirm_password" runat="server>
    *</asp:CompareValidator>
  ...
  <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
    runat="server" ControlToValidate="email" Display="Dynamic"
    ValidationExpression="\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"
    ErrorMessage="E-mail nebyl vyplněn správně!">
    *</asp:RegularExpressionValidator>
  ...
</form>
...
```

PHP

PHP nenabízí žádné metody pro kontrolu zadaných údajů, jako ostatní dva nástroje. Proto musíme vše kontrolovat pomocí vlastních funkcí.

```
...
$error_msg = array();
...
if(empty($username))
{
  $error_msg[] = "Nebylo vyplnÄ>no uĹľivatelskÃ© jmÃ©no!";
}
...
return $error_msg;
```

Ruby on Rails

V Ruby on Rails se v modelu určí pole, která se budou kontrolovat. Metody Metoda `validates_presence_of` a `validates_uniqueness_of` slouží k ověření zadání a jedinečnosti odeslaných údajů. V metodě `validate_before_create` si můžeme nastavit vlastní pravidla pro kontrolu údajů.

Model:

```
...
validates_presence_of :username, :email
       validates_uniqueness_of :username, :email
       ...
def validate_on_create
  @email_format = Regexp.new(/^([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})$/)
  errors.add(email, "E-mail nebyl vyplněn správně!")
    unless @email_format.match(email)
  errors.add(confirm_password, "Hesla se neshodují!")
    unless password == confirm_password
  errors.add(password, "Heslo nebylo vyplňeno")
    unless !password or password.length > 0
end
...
```

V pohledu poté zadáme místo pro výpis chyb pomocí `<%= error_messages_for 'user' %>`.

5.2.6 Šifrování

Pro zvýšení bezpěčnosti bylo heslo před uložením do databáze zašifrováno pomocí algoritmu `SHA1` [6].

ASP.NET

Pro zašifrování hesla v ASP.NET byla použita funkce nalezená na internetu [5]. Zdrojový kód této funkce je značně složitý.

PHP

Oproti tomu pro zašifrování hesla v PHP nám postačí jeden příkaz:

```
$password = sha1($password);
```

Ruby on Rails

V Ruby on Rails musíme nejdříve nahrát potřebnou třídu a poté můžeme použít funkci k šifrování.

```
require "digest/sha1"
...
  @pass = Digest::SHA1.hexdigest(@pass)
...
```

5.2.7 Práce s časem

U několika relací v databázi je použit datový typ timestamp (časová známka), který v aplikaci slouží pro uložení aktuálního času provedení akce.

Každý z nástrojů generuje timestamp v jiném formátu.

	ASP.NET	PHP	Ruby on Rails
Funkce	`DateTime.Now`	`time()`	`Time.now`
Formát	`11.5.2008 9:29:36`	`1210490979`	`Sun May 11 09:29:35 +0200 2008`

Jak můžeme vidět, každý z nástrojů vrací timestamp v jiném formátu, a proto k zápisu do databáze byly použity přímo funkce databáze.

Např.

```
SELECT count(id) FROM log_users
 WHERE users_id = '1'
 AND now() - timestamp < '00:30:00.00';
```

Tento dotaz vrátí počet uživatelů, kteří mají `id` 1 a byli aktivní v posledních 30ti minutách. V praxi tento dotaz slouží ke kontrola, zda daný uživatel nebyl příliš dlouho neaktivní. Příkaz `now()` vrací aktuální datum a čas ve formátu `2008-05-11 09:29:38.894`.

5.3 Omezení

Kvůli značné časové náročnosti implementace aplikace pomocí jednotlivých platforem se nepodařilo splnit všechny požadavky, které byly předloženy v návrhu webové aplikace (kapitola 4). Veškeré funkce a problémy, uvedené ve srovnání, jsou ve všech třech implementacích naprogramovány.

5.4 Shrnutí

ASP.NET je velmi mocný nástroj, který obsahuje spoustu již hotových komponent pro tvorbu webových aplikací. ASP.NET však vyžaduje dobré pochopení práce v .NET Frameworku. Při implementaci praktické části práce docházelo ke spoustě problémům. Tyto problémy byly často způsobeny databází. Databáze PostgreSQL není typická pro ASP.NET a práce s ní je na internetu špatně dokumentována.

PHP je dobrý nástroj pro začínajícího uživatele. Poskytuje spoustu funkcí a není tak přísné na správnost napsaného kódu. Programování v PHP je však náročnější na čas. Metody a funkce, které jsou již v ASP.NET a Ruby on Rails připraveny k použití, je v PHP potřeba nejdříve naprogramovat a až poté je lze používat.

PHP má oproti zbylým dvěma nástrojům jednu velikou výhodu. Je velmi rozšířený mezi uživateli internetu a většina problému práce s PHP je již na internetu vyřešena a tak není těžké nalézt potřebné řešení.

Ruby on Rails je mladý a stále ještě se rozvíjející nástroj. Jeho kouzlo spočívá v jednoduchosti vytváření aplikací. Tuto jednoduchost uživateli poskytují generátory. Během chvíle uživatel může mít funkční aplikaci pro přidávání článků. Pokud ovšem chce přidávat další funkce a propojovat je mezi sebou, stává se práce s aplikací složitější. Je to způsobeno právě mládím této platformy, kdy se spousta uživatelů učí s ní pracovat. Ke spoustě problémům je proto velmi obtížné najít na internetu řešení, které by fungovalo.

Kapitola 6

Závěr

Hlavním cílem práce bylo provést porovnání nástrojů a platforem pro tvorbu webových aplikací. Toto porovnání bylo nejdříve provedeno na základě nastudovaných informací (kapitola 3). V práci je dále uvedeno řešení problémů tvorby webové aplikace pomocí ASP.NET C#, PHP a Ruby on Rails. Porovnání těchto nástrojů je v kapitole o implementaci (kapitola 5). Navržená aplikace bohužel nebyla implementována v plném rozsahu, tak jak je specifikována v návrhu (kapitola 4).

Za vlastní přínos autor považuje seznámení se s problematikou návrhu a tvorby webové aplikace pomocí různých nástrojů. Práce také seznamuje čtenáře s problematikou řešení základních funkcí, které se objevují ve většině běžných webových aplikacích.

Možné rozšíření práce spočívá v porovnání většího počtu platforem, které se k tvorbě webových aplikací využívají. Stejně tak jako v implementaci komplexnější aplikace, jež by zahrnovala větší množství funkcí (například práce s obrázky, soubory, generování grafů a další).

Literatura

[1] Hana Kanisová; Miroslav Müller. *UML srozumitelně*. Computer Press a.s., 2004. ISBN 80-251-0231-9.

[2] Bruce Momjian. *PostgreSQL - Praktický průvodce*. Computer Press a.s., 2003. ISBN 80-7226-954-2.

[3] Jeff Prosise. *Programování v Microsoft .NET*. Computer Press a.s., 2003. ISBN 80-7226-879-1.

[4] WWW stránky. Smarty : Template engine. `http://www.smarty.net/`, Naposledy navštíveno 11. 5. 2008.

[5] WWW stránky. Encrypting passwords with sha1 in .net and java. `http://authors.aspalliance.com/thycotic/articles/view.aspx?id=2`, Naposledy navštíveno 12. 5. 2008.

[6] WWW stránky. Secure hash algorithm - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/Secure_Hash_Algorithm`, Naposledy navštíveno 12. 5. 2008.

[7] WWW stránky. Webová aplikace - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/Webová_aplikace`, Naposledy navštíveno 2. 4. 2008.

[8] WWW stránky. Install php 5 apache mysql on windows : Wampserver. `http://www.wampserver.com/en/index.php`, Naposledy navštíveno 20. 4. 2008.

[9] WWW stránky. Ruby on rails. `http://www.rubyonrails.org/`, Naposledy navštíveno 20. 4. 2008.

[10] WWW stránky. Postgresql: The world's most advanced open source database. `http://www.postgresql.org/`, Naposledy navštíveno 21. 4. 2008.

[11] WWW stránky. Databáze - wikipedie, otevřená encyklopedie. `http://cs.wikipedia.org/wiki/Databáze`, Naposledy navštíveno 5. 6. 2008.

[12] WWW stránky. Glossary. `http://www.simply.com.au/glossary.php?alphabet=W`, Naposledy navštíveno 6. 5. 2008.

[13] WWW stránky. Html 5. `http://www.w3.org/html/wg/html5/`, Naposledy navštíveno 6. 5. 2008.

[14] WWW stránky. Model, view, controller. http://www.ocs.cz/text/Cocoa/MVC.html, Naposledy navštíveno 6. 5. 2008.

[15] WWW stránky. Rfc 2616 (rfc2616) - hypertext transfer protocol – http/1.1. http://www.faqs.org/rfcs/rfc2616.html, Naposledy navštíveno 6. 5. 2008.

[16] WWW stránky. Xhtml 1.0: The extensible hypertext markup language (second edition). http://www.w3.org/TR/2002/REC-xhtml1-20020801/, Naposledy navštíveno 6. 5. 2008.

[17] WWW stránky. Aplikace na webu: 6. cgi-skripty. http://www.kosek.cz/clanky/iweb/06.html, Naposledy navštíveno 7. 5. 2008.

[18] WWW stránky. Css - kaskádové styly. http://www.jakpsatweb.cz/css/, Naposledy navštíveno 7. 5. 2008.

[19] WWW stránky. Fastcgi: rychleji, lépe, bezpečněji. http://www.ics.muni.cz/zpravodaj/articles/112.html, Naposledy navštíveno 7. 5. 2008.

[20] WWW stránky. Codeguru: The .net architecture. http://www.codeguru.com/csharp/sample_chapter/article.php/c8245/, Naposledy navštíveno 8. 5. 2008.

[21] WWW stránky. Php: Hypertext preprocessor. http://www.php.net/, Naposledy navštíveno 8. 5. 2008.

[22] Andrew Troelsen. *C# a .NET 2.0 profesionálně*. Zoner Press, 2006. ISBN 80-86815-42-0.

Seznam příloh

Dodatek A Ukázka vzhledu aplikace

Dodatek B Popis zprovoznění jednotlivých implementací

Dodatek C CD se zdrojovými kódy

Dodatek A

Ukázka vzhledu aplikace

Obrázek A.1: PHP - Seznam článků

Obrázek A.2: Ruby on Rails - Registrace

Dodatek B

Popis zprovoznění jednotlivých implementací

PostgreSQL

Vytvoříme databázovou strukturu pomocí skriptu `\databaze\databaze.sql`.

ASP.NET C#

1. Nakopírujeme soubory se zdrojovými kódy aplikace na server. Zdrojové kódy se nachází ve složce `\aspnet\`.
2. V souboru `\aspnet\App_Code\Database.class.cs` nastavíme jednotlivým proměnným správné hodnoty pro připojení k databázi.

PHP

1. Nakopírujeme soubory se zdrojovými kódy aplikace na server. Zdrojové kódy se nachází ve složce `\php\`.
2. V souboru `\php\include\constants.php` nastavíme jednotlivým konstantám správné hodnoty pro připojení k databázi.

Ruby on Rails

1. Nakopírujeme soubory se zdrojovými kódy aplikace na server. Zdrojové kódy se nachází ve složce `\ruby\`.
2. V souboru `\ruby\config\database.yml` nastavíme správné hodnoty pro připojení k databázi.