UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Webový portál se zaměřením na bojová umění

Tomáš Fučík

Bakalářská práce
2011

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2010/2011**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš FUČÍK**
Osobní číslo:	**I08044**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Webový portál se zaměřením na bojová umění**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Úkolem bude vytvořit webový portál se zaměřením na bojová umění.

Cílem teoretické části:
- srovnání s podobně fungujícími portály
- zhodnocení současných technologií
- návrh vhodné databáze

Cílem aplikační části bude vytvořit webový portál s následujícími požadavky:
- profily pro jednotlivé uživatele
- registrace a správa klubů a týmů
- možnost psaní článků a jejich správa
- možnost přidávání kategorií a akcí
- administrační rozhraní pro správu celého portálu a jejich modulů.

Aplikace bude postavena na technologii ASP.NET a pro správu relační databáze bude použit Microsoft SQL Server.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

***EVJEN, Bill; HANSELMAN, Scott; RADER, Devon. ASP.NET 3.5 v jazycích C a Visual Basic : Programujeme profesionálně. Praha : Computer Press, 2009. 1600 s. ISBN 978-80-251-2069-9.**

*** HOTEK, Mike. Microsoft SQL Server 2008 : Krok za krokem. Praha : Computer Press, 2009. 488 s. ISBN 978-80-251-2466-6.**

Vedoucí bakalářské práce: **Ing. Lukáš Čegan, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **17. prosince 2010**

Termín odevzdání bakalářské práce: **13. května 2011**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2011

Prohlášení autora

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

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

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 8. 8. 2011 Tomáš Fučík

Poděkování

Touto cestou bych rád poděkoval všem, kteří mě při tvorbě této práce podporovali, nebo mi jakkoliv pomohli. Zejména pak panu Ing. Janu Kellerovi a vedoucímu práce Ing. Lukáši Čeganovi, Ph.D. za cenné rady a připomínky v průběhu realizace této práce. Dále bych rád poděkoval mé rodině a přátelům za podporu během studia.

Anotace

Bakalářská práce je zaměřena na webové technologie. Cílem je vytvoření webového portálu s tématikou bojového umění. Teoretická část se zabývá srovnáním webových portálů z hlediska jejich provedení a nabízeného obsahu. Dále je zde zhodnocení současných webových technologií a seznámení s návrhem databáze. V praktické části je popsána architektura aplikace, konfigurace aplikace, návrh modulů a především nejvíce využité prostředky z technologie ASP.NET.

Klíčová slova

Webová aplikace, relační databáze, architektura aplikace, ASP.NET, SQL, Ajax Control Toolkit.

Title

Web portal with the theme of martial arts.

Annotation

The thesis is focused on web-paging technologies. An objective is to create a web portal about martial arts. Teoretical part deals with comparison of web portals from the perspective of it's design and it's content. In the next part is there an evaluation of nowadays web-paging technologies and introduction to database design. In the practical part is describe the architecture of application, it's configuration, modul's design and especially the most used ASP.NET's elements.

Keywords

Web Application, relational database, application architecture, ASP.NET, SQL, Ajax Control Toolkit.

Obsah

Seznam zkratek

HTTP	Hypertext Transfer Protocol
IIS	Internet Information Services
ASP	Active Server Pages
PHP	Hypertext Preprocessor
HTML	HyperText Markup Language
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
AJAX	Asynchronous JavaScript and Extensible Markup Language
CGI	Common Gateway Interface
IDC	Internet Database Connector
GUID	Globally unique identifier
CLI	Common Language Infrastructure
URL	Uniform Resource Locator
UML	Unified Modeling Language
API	Application Programming Interface
ADO.NET	ActiveX Data Object for .NET
SQL	Structured Query Language
T-SQL	Transact-SQL
PL/SQL	Procedural Language/SQL
ISAPI	Internet Server Application Programming Interface
CSS	Cascading Style Sheets

Seznam obrázků

Seznam tabulek

1 Úvod

1.1 Výběr tématu bakalářské práce

Téma bakalářské práci jsem si zvolil na základě osobního zájmu o vývoj webových aplikací. Abych si rozšířil znalosti, zvolil jsem k vypracování praktické části technologie, které nebyly probrány v rámci bakalářského studia. Při této příležitosti jsem zvolil moderní technologii ASP.NET, s jejíž pomocí byl vytvořen základ webového portálu (webové aplikace) pro příznivce bojových umění.

1.2 Popis a cíl práce

První teoretické části práce se zabývá srovnáním existujících webových portálů. Tento úsek práce je především zaměřena na zpracování různých částí portálů a jejich obsahu, který nabízejí pro potenciální návštěvníky. Dále jsou rozepsány webové technologie, u každé je základní seznámení, případně nějaké klady nebo zápory a následně i celkové zhodnocení. Poslední oddíl teoretické části se zabývá vhodným návrhem databáze. V této části se čtenář seznámí se základními pojmy a fázemi návrhu databáze.

V praktické části je seznámení s použitými nástroji, které byly použity pro vytvoření webového portálu. Následně je zde popsána analýza vytvořené webové aplikace s projektovým názvem XMAonline.cz. V této část se pojednává o architektuře, způsobu zpracování modulů a grafického návrhu webové aplikace. Na tuto část navazuje popis základní konfigurace a nejdůležitějších prvků ASP.NET, které byly použity při zpracování webové aplikace. V posledním úseku práce je seznámení se způsobem zprovoznění potřebné databáze a krátký popis vytvořených tabulek.

2 Srovnání s podobně fungujícími portály

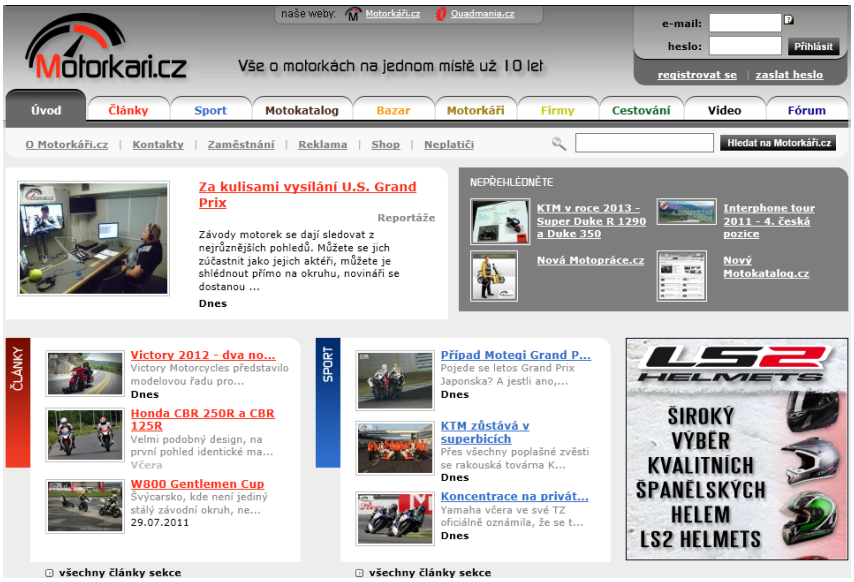
Ke srovnání byly zvoleny už několik let provozované webové portály a to:

- Motorkari.cz,
- Mobilmania.cz.

Oba srovnávané portály si za svou dobu provozu získali spoustu příznivců. Kromě samotného kvalitního provedení stránek nabízejí také velmi kvalitně zpracovaný obsah.

2.1 Motorkari.cz

Při příchodu na stránky je ihned vidět, že se jedná o portál zaměřený na motocykly. Samotná úvodní stránka poskytuje dobrý přehled o nových článcích, sportovních událostech, nahraných videích, a spoustu dalšího. Pokud návštěvníka nějaká sekce zajímá, nemusí ji vyhledávat v menu, ale může rovnou vybrat danou sekci z úvodního obsahu stránky.



Obrázek 1 - Úvodní stránka portálu Motorkari.cz

2.1.1 Články

Na portále jsou články setříděné do kategorií. Celkově tvoří ucelený přehled o novinkách, soutěžích a aktuálním dění ze světa motocyklů. Mezi články jsou zařazeny i redakční testy, které jsou roztříděny do podkategorií podle značek motocyklů. Test každého motocyklu provádí jeden nebo více jezdců. Své zkušenost a celkové ohodnocení motocyklu uvedou do jednoho společného článku.

2.1.2 Katalog motocyklů

Motorkari.cz působí na internetu dlouho, a za celou dobu svého působení se tomuto portálu povedlo vytvořit podrobný katalog motocyklů. Samotný katalog je velmi

propracovaný a uživatelé mohou sami posílat na portál informace pro doplnění katalogu, nechybí zde manuály, hodnocení majitelů, přístup přímo do bazaru k vybranému motocyklu a spousta dalšího.

2.1.3 Motorkáři

Každý zaregistrovaný uživatel může vkládat informace o jeho motocyklu, své zkušenosti a fotografie. Ostatní uživatelé mají možnost k jeho profilu připsat komentáře, dát libovolné hodnocení, nebo si uživatele přidat do přátel. Celá tato sekce je také velmi dobře zpracována.

2.1.4 Ostatní sekce

Další velmi pěkně zpracovaná sekce je bazar, kde se každý den objevují nové nabídky nejen na samotné motocykly, ale i náhradní díly a doplňky. Dále na portálu najdeme katalog firem, kalendář akcí, články z cest a samozřejmostí je také fórum.

2.2 Mobilmania.cz

Webový portál Mobilmania.cz nabízí výborný přehled o nejnovějších mobilních zařízeních a všem, co se kolem nich děje. Mezi velké výhody patří i propojení s dalšími velkými známými portály, jako je zive.cz, navigovat.cz nebo autorevue.cz

Obrázek 2 - Úvodní stránka portálu Mobilamia.cz

2.2.1 Články

Oproti portálu motorkáři.cz zde není podrobné členění článků podle do spousty kategorií, což není až tak na škodu. Protože Mobilmania.cz má rozsáhlou škálu zaměření na různé odvětví v mobilních technologiích, a nějaké smysluplné rozřazování do

podrobnějších kategorií by bylo velmi obtížné. Proto je zde na výběr pouze ze čtyř kategorií a to:

- články,
- bleskovky,
- tiskovky,
- a štítky.

2.2.2 Video

Redaktoři během týdne natáčejí videa, kde se zabývají problematikou a aktuálním stavem mobilních technologií, mobilních operátorů, novými zařízeními a spousty dalších zajímavostí.

2.2.3 Testy

Pro návštěvníky portálu, kteří se chtějí dozvědět něco bližšího o nějakém telefonu, naleznou potřebné informace v sekci testy, kde je na výběr celá řada telefonů různých značek. Jsou zde testovány telefony ze všech možných hledisek, nejen uživatelská prostředí, ale také použité materiály, vzhled telefonu, použitý hardware a příslušenství.

2.2.4 Katalog mobilů

Obsahuje podrobný seznam mobilů, ke každému telefonu jsou vedeny podrobné informace o rozměrech, hmotnosti, hardwarové a softwarové výbavě. Z těchto uvedených informací je sestaven žebříček celkového hodnocení všech telefonů v katalogu.

2.3 Závěrečné porovnání

V následující tabulce (Tabulka 1) je porovnání nabízeného obsahu webových portálů Motorkari.cz, Mobilmani.cz a XMAonline.cz[1].

Tabulka 1 - Srovnání obsahu webových portálů

Obsah	Motorkari.cz	Mobilmania.cz	XMAonline.cz
Články	Ano	Ano	Ano
Aktuality	Ano	Ano	Ano
Testy	Ano	Ano	Ne
Bazar	Ano	Ano	Ne
Katalog	Ano	Ano	Ano
Fotografie	Ano	Ano	Ano
Video	Ano	Ano	Ne
Fórum	Ano	Ano	Ne

[1] XMAonline.cz je cílem praktické části této práce.

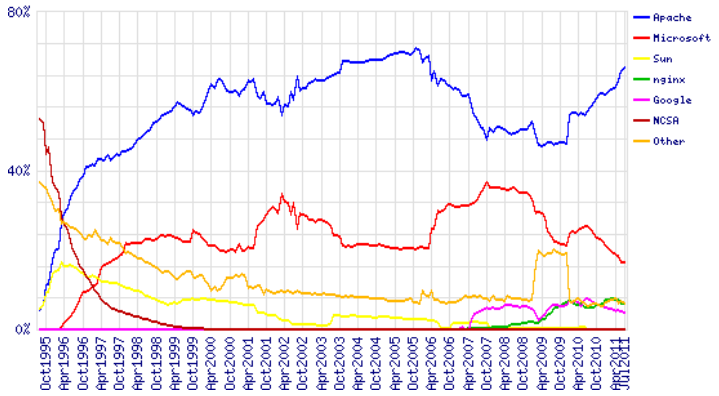
3 Zhodnocení současných technologií

Dnešní webové technologie přinášejí hodně možností, díky kterým se dnes zřídka dá narazit na klasické statické stránky, které se hojně objevovaly před pár lety. Nové technologie přispěly také k interaktivnější práci se stránkami a postupem času začal přesun klasických desktopových aplikací na webové servery v podobě webových aplikací.

Při tvorbě webových aplikací je na výběr z široké škály technologií a programovacích jazyků. Nabízejí se technologie zcela zdarma, až po profesionální placené. Protože pro nezkušeného člověka je těžké si správně vybrat, je v příloze (Příloha B) tabulka, kde je porovnání webových technologií ve vztahu k velikosti výsledných řešení vytvářené webové aplikace.

3.1 Webový server

Při procházení internetových stránek pomocí internetový prohlížeč dochází při každém kliknutí na odkaz k odeslání HTTP požadavku na webový server, který přijatý požadavek zpracuje a pošle odpověď. Tento webový server nebývá pouze samotný hardware, ale také balík složený z několika druhů softwaru. Na následujícím obrázku (Obrázek 3) je vykreslen graf znázorňující podíl webových serverů na trhu od srpna roku 1995 až po červenec roku 2011. Z tohoto grafu je jednoznačně vidět, že největší podíl má Apache.



Obrázek 3 - Podíl webových serverů na trhu

Zdroj: http://news.netcraft.com/

3.1.1 Apache

Je nejpopulárnější internetový webový server od roku 1996. Patří mezi nejznámější multiplatformní HTTP servery s otevřeným zdrojovým kódem pro moderní operační systémy včetně UNIX a Windows NT. Cílem projektu The Apache HTTP Server Project je poskytnout bezpečný, efektivní a rozšiřitelný server, který poskytuje HTTP služby v souladu s platnými HTTP standardy. Apache je určen pro práci s širokou škálou jazyků, k nejznámější patří PHP, Perl a Python. (1) (2)

3.1.2 Internet Information Services (IIS)

IIS je volitelná součást operačního systému Windows Server od vydání Windows NT 4.0. Ve své podstatě je službou Windows, která má na starosti zpracování požadavků přicházejících na konkrétních portech. IIS není pouze samotný webový server, ale kolekce nástrojů, z nichž jedna část poskytuje webové služby. Má modulární architekturu a je k dispozici řada samostatných modulů, které lze zapnout nebo vypnout dle potřebné funkcionality. Kromě využití na plnohodnotný webový server, který je schopen řešit složité úlohy vyžadující výkon, lze jej také díky poměrně snadnému nastavení použít i pro jednoduché projekty. Microsoft IIS je také doplněn o IDC[2] pokračuje v podpoře tradičních metod CGI[3] spolu s vlastní sadou filtrování a realizace systémů v podobě filtrů ISAPI[4]. (2) (3)

Tabulka 2 - Srovnání IIS a Apache

Vlastnosti	IIS	Apache
ASP	Ano	s Chilisoft, Apache::ASP, nebo Mod_Mono
CGI	Ano	Ano
Perl	Ano	Ano
Python	Ano	Ano
PHP	Ano	Ano
JSP	Ano	Ano
Zabudovaný .NET	Ano	Ne

3.2 Webové technologie statického obsahu

Společenstvím národů na Internetu je organizace World Wide Web Consortium, často označováno jako W3C. Jejím záměrem je přesvědčovat webovou komunitu o důležitosti univerzálnosti, ale současně i nutnosti uspokojit chuť po líbivém vzhledu stránek. Hlavní myšlenkou je, aby důležité společnosti jako je Apple, Adobe, Opera, Microsoft a několik dalších významných společností se spojily a nevytvářeli si každý své standardy. (4)

Základním prostředkem pro vytváření webových stránek je jazyk HTML, který podléhá neustálému vývoji. Současné vyjádření v doporučeních W3C HTML 4.01 směřují k tomu, aby byl obsah oddělen od formy zobrazení. Proto byly zavrženy všechny atributy prvků, které stanovují formu zobrazení, mezi nejznámější z nich patří bgcolor, font nebo align. Formátování dokumentů by se mělo dosahovat pomocí CSS, neboli kaskádových stylů. (5) (6)

[2] IDC - Internet Database Connector je nástroj, který lze používat k posílání dotazů do databáze.
[3] CGI - Common Gateway Interface je standardní způsob pro propojení externích aplikací s webovým serverem.
[4] ISAPI - Internet Server Application Programming Interface

3.2.1 HTML

HTML je značkovací jazyk, který slouží k popisu obsahu webové stránky. Veškeré příkazy v HTML nazýváme značky (nebo také tagy). Uzavírají se ve špičatých závorkách, do kterých jsou spolu se značkou zahrnuty případné parametry. Vše ostatní mimo tyto značky je text, který internetový prohlížeč zobrazí. Používají se jak párové, tak i nepárové značky. Párové mají vliv na určitou část dokumentu, kde první část se nachází před ovlivňovanými prvky a druhá za nimi. Tímto způsobem se vymezí část, na kterou se aplikuje vlastnost párové značky. Sem se zahrnují například značky, které ovlivňující text (tučnost, zarovnání, kurzíva). I když se dopustíte nějakých chyb, dokáže většinou prohlížeč zobrazit prakticky celou stránku tak, jak má vypadat. Ovšem pokud narazí na zásadní chybu, nedokáže si s ní poradit. (7)

Obecný zápis párové značky může vypadat následovně:

```
<značka>
      text, na který bude aplikována značka
</značka>
```

a nepárové značky takto:

```
<značka>
```

Celková struktura dokumentu je rozdělena na *html, head* a *body*.

Dokumenty HTML mají tři společné značky vymezující určité oddíly dokumentu.

1. Značka <html>

   Značka <html> ohraničuje celý dokument HTML. Tudíž určuje, kde daný dokument začíná a končí.

   ```
   <html>
   … obsah dokumentu …
   </html>
   ```

2. Značka <head>

   Značka <head> vymezuje záhlaví dokumentu HTML. Do této části dokumentu se umísťují metedata, název dokumentu nebo skripty použité v dokumentu. Data umístěná ve značce <head> jsou pro koncového uživatele „neviditelné“.

3. Značka <body>

   Do značky <body> se umísťuje hlavní obsah, který je zobrazen koncovému uživateli. (4)

3.2.2 XHTML

„Písmeno X značí ve zkratce XHTML slovo eXtensible a je, co se týká jazyka, založeno na XML. A přesně to má XHTML dokázat – stát se vzorem jazyka HTML založeným na XML.“ (8)

Aby byly dokumenty bez problému přenositelné a rozšířitelné, je třeba základní pravidla jazyka XHTML vždy dodržovat a to:

- Procesní instrukce XML – Jedná se o značku *<?xml ?>*, která říká, že se v dokumentu používá XML. Z důvodů zpětné kompatibility se však nevynucuje.

  ```
  <?xml vision="1.0" ?>
  ```

- DOCTYPE – Je nezbytné vždy specifikovat DOCTYPE, aby prohlížeč věděl, jakou verzi XHTML vyžadujete.

  ```
  <!DOCTYPE html PUBLIC "-//W3C/DTD XHTML 1.0 Strict//EN"
  "http://www.w3c.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  ```

- Zápis značek – Všechny značky musí být korektně zapsány v souladu s pravidly XML (není dovolené křížení značek).

  ```
  <b><i>Text tučný s kurzívou</b></i>
  ```

  Značky, které mají nějaký obsah, musí být vždy uzavřeny.

  ```
  <ul>
  <li>Položka 1</li>
  <li>Položka 2</li>
  </ul>
  ```

  Tyto zásady je nutné dodržovat i u značek *
* pro odsazení řádku, *<hr>* pro vodorovnou čáru nebo *<img>* pro vložení obrázku. Korektní zápis vypadá následovně *
*, *<hr />*. Značky bez obsahu můžeme zapsat dvěma způsoby:

  ```
  <img src="obrazek.gif"></img>
  <img src="obrazek.gif" />
  ```

- Velikost písmen – Stejně jako v XML, tak i v XHTML se smějí značky a jejich parametry psát pouze malými písmeny.

  ```
  <p>Odstavec</p>
  ```

- Hodnoty parametrů musí být v uvozovkách, a pokud parametr uvedete, musí mít nastavenou také danou hodnotu.

  ```
  <p align="right">Odstavec</a>
  ```

(8)

3.2.3 HTML versus XHTML

HTML a XHTML používají stejné elementy, parametry a hodnoty. Rozdíl je pouze v syntaxi. Jestliže je HTML jedno, zda použije elementy html, head, body a s nimi DOCTYPE, pak jazyku XHTML to jedno není. Jestliže HTML umožňuje občas nějakou tu značku neukončit, XHTML trvá na uzavření všech elementů, i těch prázdných. „Jestliže vám HTML umožňuje vypustit uvozovky okolo hodnot parametrů, které obsahují pouze písmena, číslice a čtyři jednoduché symboly (-, ., _, a :), jazyk XHTML má noční můry (a zobrazuje chyby), pokud uvozovky na toto místo nezapíšete.“ (4)

3.3 Webové technologie dynamického obsahu

Základní rozdělení skriptů je podle toho, zda běží na straně klienta nebo serveru. Klientské skripty běží na straně klienta, tedy na straně s webovým prohlížečem, naopak serverové skripty se spouštějí na straně serveru. (9)

3.3.1 Klientské skripty

Hlavní myšlenkou je, aby webové aplikace vypadaly a chovaly se jako desktopové aplikace. Jak už je z názvu patrné, jsou spouštěny na straně klienta tzv. klientským softwarem. Výhodou je, že můžou pracovat na pozadí a komunikovat se serverem, abychom získali data bez nutnosti obnovení stránky v prohlížeči. (9)

JavaScript

JavaScript je programovací jazyk, jehož činnost zabezpečuje prohlížeč webových stránek. Tudíž k provádění kódu dochází na straně klienta až po načtení webové stránky. Ranou verzi JavaScriptu původně nazývaného LiveScriptu vytvořil v roce 1995 vývojář Brendan Eich pracující pro Netscape Comminucations. I když obsahuje v názvu slovo „Java“, nemá s programovacím jazykem Java příliš společného. (10)

Aby byla zachována bezpečnost, autoři do jazyka JavaScript nezabudovali funkce umožňující zápis a čtení ze souborů. Uživatel se tak nemusí obávat, že by mu někdo smazal obsah disku nebo zneužil diskrétní informace. „Na druhou stranu to, že je JavaScript jazyk interpretovaný na straně klienta, do značné míry svazuje ruce tvůrcům stránek, jelikož je ochuzuje o možnost zpracovávat centrálně a uchovávat údaje týkající se provozu webové stránky.“ (10) I přes uvedená omezení je JavaScrcipt silnám nástrojem jak zinteraktivnit webovou stránku. (10)

Ajax

Samotné označení Ajax nám samo o sobě moc neřekne, ale pokud se podíváme podrobněji na celý název Asynchronous JavaScript and XML, ze kterého zkrat vzešla, je hned patrné, že Ajax představuje kolekci technologií.

Slovo asynchronous značí, že prohlížeč je schopen odpověď zpracovat, až v okamžiku, kdy k ní skutečně dojde, a nemusí čekat na odpověď od serveru. Jinak řečeno, přenos dat se odehrává na pozadí, aniž by musel prohlížeč pozastavit prováděné operace a

na něco čekat. V názvu nalezneme i slovo „JavaScript“, které je v termínu Ajax velmi důležité, protože právě JavaScript je tím, co řídí úkony probíhající v prohlížeči. JavaScript má v případě Ajaxu na starosti spojení se serverem i při zpracování dat přijatých od serveru. Po přijetí potřebných dat od serveru, může být opět použit JavaScript pro zpracování těchto dat a jejich zobrazení či jiné využití. Jazyk XML se stal základním komunikačním prostředkem pro web umožňující odesílání dat v textové podobě skrze Internet, z tohoto důvodu se aplikace využívající Ajax nejčastěji vytváří tak, aby zpracovávali informace přijaté od serveru ve formátu XML. (11)

Výhodou technologie Ajax je, že není nutné znovu načítat celé stránky při každém volání serveru, což také přispívá k interaktivnějšímu webu a zpříjemňuje tak samotné procházení stránky. Ale vše má samozřejmě své nevýhody, a tak se lze setkat při použití určitých komponent například z *AjaxControlToolkitu* s nekompatibilitou a nemožností plně využít všech jejich funkcí. (11)

3.3.2 Serverové skripty

Skripty spouštěné na straně webového serveru se všeobecně nazývají serverové skripty. K jejich výhodám patří, že nezatěžují počítač na straně klienta. Samotný klient ani nemusí vědět, že na straně serveru se daný skript spustil.

PHP

První základy pro vznik jazyka PHP položil Rasmus Lerdorf v roce 1994. Rasmus si vytvořil v Perlu jednoduchý systém na evidování přístupu k jeho stránkám. Později kvůli zátěži serveru, přepsal systém do jazyka C. Dále kvůli požadavkům uživatelů, jazyk uvolnil pod názvem Personal Home Page Tools, který se později změnil na Personal Home Page Construction Kit. (12)

PHP je interpretovaný jazyk. Což přináší výhodu rychlosti vývoje aplikací. Stačí jen napsat skript a bez jakékoliv zdlouhavé kompilace si prohlédnout jeho výsledek ihned v prohlížeči. Právě pro tuto výhodu má i velkou nevýhodu související s tím, že se jedná o interpretovaný jazyk. Pokud bychom jazyk kompilovali, sám kompilátor by odhalil všechny syntaktické chyby. Tyto chyby odhalí PHP až při vykonávání samotného skriptu, kdy vypíše chybové hlášení a přeruší běh skriptu. (12)

ASP.NET

ASP.NET se používá k tvorbě webových aplikací a služeb. Je součást Microsoft .NET Frameworku[5]. Nabízí úplný, objektově orientovaný programovací model, který obsahuje architekturu řízenou událostmi, založenou na ovládacích prvcích, což podporuje zapouzdření kódu a jeho opětovné využívání. NET Framework je rozčleněn do propracované kolekce funkčních částí, zahrnující více než 10 000 typů (třídy, struktury, rozhraní…), které lze při vývoji využívat. V ASP.NET lze psát kód v kterémkoli z podporovaných jazyků .NET. Dalším významným aspektem ASP.NET je, že se

[5] Microsoft .NET Framework – Aplikační rozhraní pro operační systém Windows.

neinterpretuje, ale kompiluje, což vede ke zvýšení výkonu oproti předchozímu ASP. (13) (14)

Kdybych měl ASP.NET nějakým způsobem zhodnotit, určitě bych vyzvedl jednoduchý vývoj webových aplikací za využití vývojového prostředí Microsoft Visual Studio. Také se nabízí možnost propojení s dalšími technologiemi poskytovanými přímo od Microsoftu. Samozřejmě že ASP.NET není dokonalé, stejně jako u všech technologií se i zde najdou negativa. Jedním z nich je určitě ViewState, ve kterém se ukládají hodnoty vlastností komponent a při špatném zacházení může dojít ke zbytečně velkému nárůstu přenášených dat mezi klientem a serverem.

3.4 Závěrečné zhodnocení

Technologií, ze kterých si vybrat je mnoho, a neexistuje jednoznačné vodítko, které by nám řeklo, jaké technologie si zvolit, nebo zda se jedná o dobrou nebo špatnou technologii. Každá najde své využití a pro výběr nám může být nápomocná tabulka v příloze (Příloha C). V této tabulce jsou zhodnoceny technologie, vzhledem k velikosti výsledných webových aplikací.

4 Návrh vhodné databáze

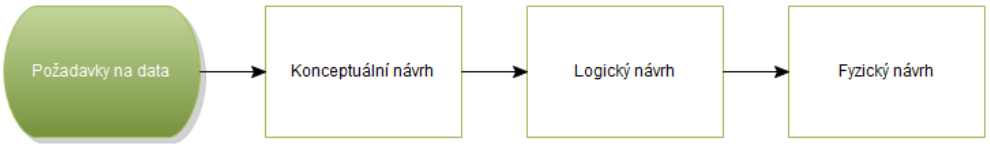
Existuje celá řada možností, jak uchovávat data (informace), dnes je velice rozšířený způsob uchovávání dat pomocí relačních databázových systémů. V poslední době se také objevují objektově orientované databáze, které představují úplně jiný myšlenkový pohled na data.

V této práci se budeme dále věnovat návrhu týkající se relační databáze, což je sada nástrojů nejen pro ukládání dat, ale i samotnou manipulaci s těmito daty. Ještě před samotným návrhem je zde seznámení se základními databázovými objekty.

Tabulky jsou nejzákladnější databázový objekt. Každá tabulka se skládá z takzvaných doménových dat (sloupce) a entitových dat (řádky). Ty bývají nejlépe znázorněny pomocí diagramu, který představuje vizuální reprezentaci návrhu databáze, která zahrnuje samotné tabulky, názvy sloupců v tabulkách a relace mezi tabulkami. Dalšíu užitečným objektem jsou pohledy. Představují virtuální tabulku a většinou se také jako tabulka používá, ale sama o sobě na rozdíl od tabulky žádné data neobsahuje. Hlavním důvodem jejich použití je bezpečnost, kdy při volání provádějí mapování a následnou reprezentaci dat nad jednou nebo více tabulkami. Základem programování v SQL jsou procedury. Jedná se o uspořádanou posloupnost příkazů jazyka T-SQL nebo PL/SQL. Uživatelsky definované funkce se velice podobají procedurám, rozdíl nalezneme například v možnosti vracet většinu hodnot libovolného datového typu. (3)

4.1 Fáze návrhu databáze

Před samotným návrhem databáze je potřeba získat maximální přehled informací o datech, které se budou v samotné databázi uchovávat. Pak začne návrh, který musí projít řadou kroků vyobrazených na následujícím obrázku (Obrázek 4), než dojde k finálnímu fyzickému návrhu.



Obrázek 4 - Fáze návrhu databáze

4.1.1 Konceptuální návrh

Výsledkem konceptuálního návrhu je schéma obecně aplikovatelné v jakémkoli technicko-programovém prostředí nezávisle na jejich fyzickém uložení. Ve většině případů se prezentuje v podobě Use Case diagramu[6], který definuje chování systému, aniž by odhaloval jeho vnitřní strukturu a datového diagramu, který definuje třídy prvků, atributy a vztahy mezi nimi. Mezi nejznámější konceptuální datové modely patří E-R model. Zjednodušeně je konceptuální model formální popis modelované reality. (3) (15)

[6] http://en.wikipedia.org/wiki/Use_case_diagram

V prvním kroku ER modelování se vymezují entity, vztahy a atributy, dokud návrh nevyhovuje potřebám zadavateli a tomu kdo bude databázi realizovat. V druhém kroku se formulují atributy a klíče entit.

Entita

Je objekt reálného světa, který je schopen nezávislé existence a je jednoznačně odlišitelný od ostatních objektů**.** Každou entitu musíme přesně definovat slovním popisem a množinou atributů.

Atributy

Přiřazující entitám či vztahům hodnotu popisného typu, určující některou podstatnou vlastnost entity nebo vztahu. (např. jméno, plat, hmotnost).

Vztahy mezi entitami

Pokud existují vztahy (relace) mezi entitami, pak u těchto vztahů se definují dvě základní vlastnosti a to:

- kardinalita
- a parcialita.

Kardinalita vztahu vyjadřuje skutečnost, kolik (jeden či mnoho) řádků jedné tabulky může vstoupit do vztahu s kolika řádky druhé tabulky. Rozlišují se následující tři druhy vztahů a to:

- 1:1 – Ve vztahu jedna k jedné je s každým záznamem v jedné tabulce svázán právě jeden odpovídající záznam ve druhé tabulce (např. jeden občan dané země má právě jedno rodné číslo).
- 1:N – Vztah jedna k více, kde na jedné straně je jediný objekt, který je ve vztahu s jedním nebo více objekty na straně druhé (např. jeden člověk může vlastnit více automobilů, ale automobil má pouze jednoho vlastníka).
- M:N – Vztah více k více, kde vystupuje více objektů na obou stranách (např. student může být zapsán na více předmětech a zároveň jeden předmět může mít zapsáno více studentů).

Parcialita vyjadřuje povinnost či nepovinnost existence ve vztahu. To znamená, jestliže pro vztah dvou entit platí, že výskyt jedné entity je podmíněn výskytem druhé, označujeme vztah jako totální. Pakliže se entity mohou vyskytovat zcela nezávisle, používáme výraz parciální vztah. Totální vztah popisujeme jako 1:N a parciální jako 0:N.

(3) (16) (17)

4.1.2 Logický návrh

Výsledku logického návrhu je dosaženo transformací objektů z ER diagramu. Následuje formulace definic tabulek v relačním databázovém systému obsahujících atributy, primární a cizí klíče.

Kandidátní klíč

Určitému atributu, případně množina atributů říkáme kandidátní klíč, který jednoznačně identifikuje řádek. Atribut, který, je součástí kandidátního klíče se nazývá klíčový. Každá tabulka musí mít alespoň jeden kandidátní klíč, pokud budeme vkládat do tabulky další atributy, musíme ověřit skutečnost, zda navržený kandidátní klíč skutečně je jednoznačnou identifikací po celou dobu životnosti dané aplikace. (3) (15)

Primární klíč (Primary key)

„Primární klíč zajišťuje jedinečnost dat v rámci sloupců, deklarovaných jako součást tohoto primárního klíče, přičemž tato jedinečná hodnota slouží zároveň jako identifikátor každého jednotlivého řádku tabulky.„ (3)

Cizí klíč (Foreign key)

Představuje metodu zajištění datové integrity, realizují vztahy neboli relace mezi tabulkami. Pokud do tabulky přidáme cizí klíč, vytvoříme tak závislost mezi tabulkou, v níž cizí klíč definujeme, a tabulkou, na kterou se tento cizí klíč odvolává. Po přidání cizího klíče již ke každému záznamu, který vložíme do odkazující se tabulky, musí buďto v odkazované tabulce existovat odpovídající záznam se stejnou hodnotou v odkazovaném sloupci (sloupcích), nebo se hodnota sloupce (sloupců) cizího klíče musí rovnat NULL. (3) (15)

Tabulka s odkazem sama na sebe

Může nastat situace, kdy sloupec na který je potřeba se odkazovat není součástí jiné tabulky, ale je obsažen přímo v tabulce, z níž odkaz směřuje. Využití si lze představit například u tabulky se zaměstnanci, kde je třeba určit podřízeným, kdo je jejich nadřízený. (3)

Normalizace

Normalizace je jedním z nejdůležitějších základních stavebních kamenů moderního návrhu databází, která obsahuje řadu pravidel. Někdy podle jejích pravidel potřebujeme data normalizovat, ale jindy je zase musíme zcela úmyslně denormalizovat. Provádíme postupnou dekompocizi tabulek, do vyhovujícího tvaru. (3)

„*Jakmile se jednou rozhodneme implementovat normalizovanou nebo denormalizovanou strategii, vždy dostaneme v cíli nějakou databázi – a snad je to ta nejlepší databáze, jakou umíme sestrojit. Nesnažte se ale v žádném případě nějak úzkostlivě držet něčeho, co vám doporučují v knihách. Držte se toho, co je pro vaši danou konkrétní situaci správné.*“ (3)

Nultá normální forma

Tabulka je v nulté normální formě tehdy, existuje-li alespoň jedno neatomické pole. Tudíž pole obsahuje alespoň jednu dále dělitelnou hodnotu. Příkladem může být tabulka, obsahující v atributu současně jméno i příjmení, což se na první pohled může být v pořádku. Jenže může nastat situace, kdy uživatel při zadávání prohodí jméno s příjmením. To samo o sobě se také nemusí zdát být kritické, protože stále lze při pohledu na uložená data rozeznat typické české jméno a příjmení, ale opět se najde výjimka, uživatel se může jmenovat Jiří Adam, z čehož lze stěží soudit, zda je Adam jméno a Jiří příjmení nebo naopak. (3)

První normální forma

První normální forma říká, že všechny atributy musí být dále nedělitelné, neboli atomické. Tudíž „problém“ z předchozí nulté normální formy je vyřešen rozdělením jednoho atributu na dva samostatné atributy. Tedy jméno a příjmení se stanou samostatnými atributy. (3)

Druhá normální forma

Tabulka je v druhé normální formě, pokud splňuje pravidla první normální formy a každý sloupec musí být závislý na celém klíči. (3)

Třetí normální forma

Aby tabulka byla ve třetí normální formě, musí být ve druhé normální forma a všechny neklíčové sloupec tabulky, nesmí být závislé na žádném jiném neklíčovém sloupci tabulky. (3)

Ostatní normální formy

Ostatní normální formy se považují alespoň v akademickém světě ještě za součást modelu normalizace. Případy jejich využití bývají vzácné. Čtvrtá normální forma vyhovuje pravidlům třetí normální formy, avšak jeden sloupec z primárního klíče je sám o době závislý na jiných sloupcích primárního klíče. Pátá normální forma řeší bezztrátové a ztrátové dekompozice. Šestá normální forma řeší možnosti vzniku tzv. modifikačních anomálií, kdy změna v datech na jednom místě se musí rozšířit do všech ostatních potřebných míst, takže se v systému neriskuje vznik vzájemně neodpovídajících dat. (3) (16)

4.1.3 Fyzický návrh

Fyzický návrh je poslední fázi návrhu databáze. V této fázi se zohledňují charakteristické vlastnosti databáze, jako jsou datové typy atributů a integritní omezení. (17)

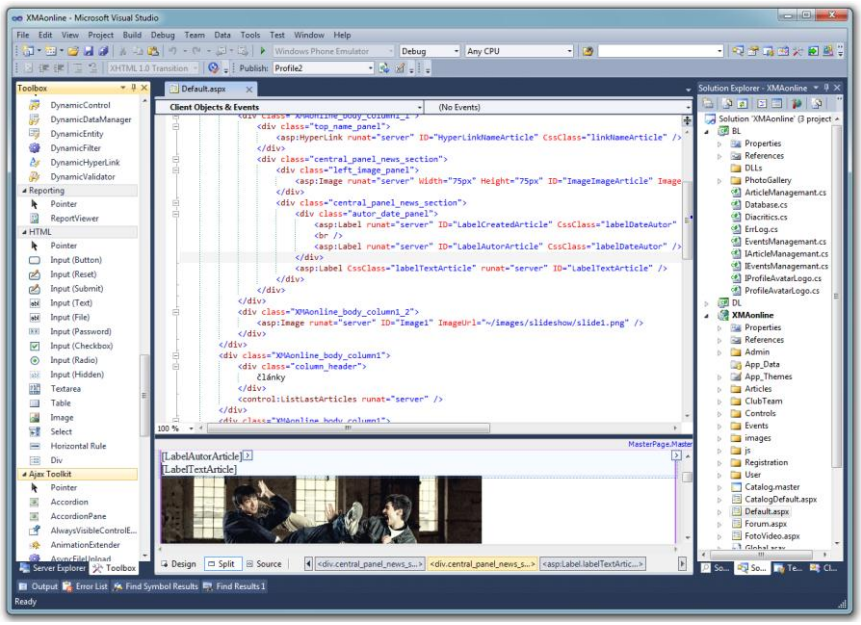
5 Praktická část

5.1 Použité nástroje

5.1.1 Microsoft Visual Studio 2010 Professional

V současné době se Visual Studio využívá po celý proces vývoje softwaru od návrhu až po nasazení. Obsahuje pokročilé nástroje pro testování, tvorbu prototypů, ladění, automatizaci apod. Základem každého projektu ve Visual Studiu je tzv. solution neboli řešení. V solution se nachází uložené projekty a výsledkem jejího sestavení může být jedna nebo více aplikací, případně i několik knihoven.

Visual Studio obsahuje propracovaný editor zdrojových kódů s možností zvýraznění syntaxe mnoha programovacích jazyků. Nechybí ani vizuální editor pro návrh vzhledu aplikací s grafickými komponentami.



Obrázek 5 - Microsoft Visual Studio 2010 Professional

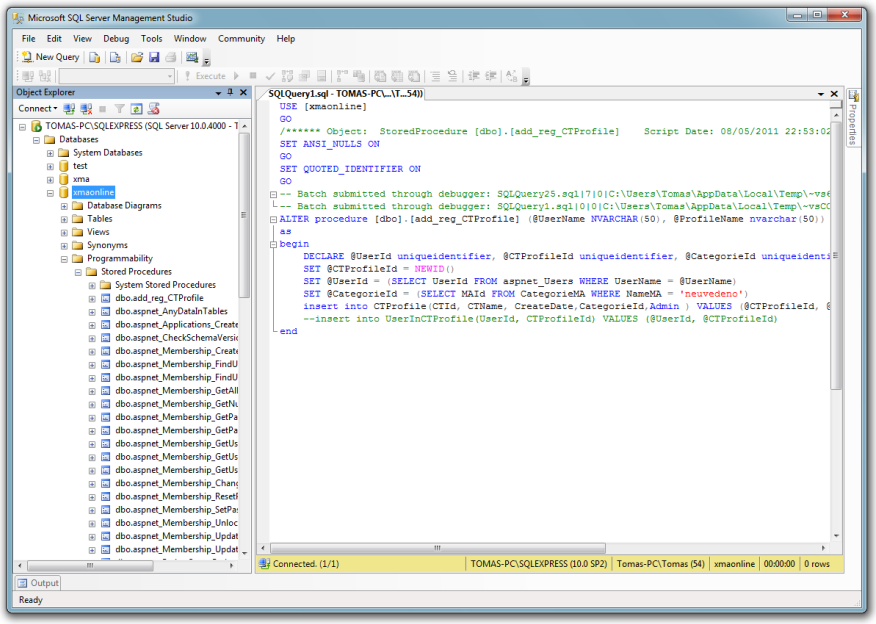
Protože má každý uživatel specifické požadavky na vývojové prostředí je možné spoustu výchozího nastavení měnit, nebo Visual Studio rozšiřovat o další nástroje. Mezi nejvíce stahované nástroje patří:

- VSCommands 2010,
- Productivity Power Tools,
- a NuGet Package Manager.

Většinou se jedná o nástroje, které zpřehledňují zdrojový kód a napomáhají při práci s projektovými soubory, třídami a metodami. Většinu rozšiřujících nástrojů je možné instalovat přímo z Visual Studia z nabídky *Tools->Extension Manager->Online Galler*.

5.1.2 Microsoft SQL Server Managemant Studio Express

Je snadno použitelný grafický nástroj pro správu Microsoft SQL Serveru ve verzi 2005 a 2008.



Obrázek 6 - Microsoft SQL Server Managemant Studio Express

Toto studio obsahuje editoru kódu, který zvýrazňuje syntaxi jazyka SQL. Nově od verze SQL Serveru 2008 podporuje také technologii IntelliSense, která dokáže při psaní SQL dotazů napovídat strukturu tabulek. Kromě editoru obsahuje také grafický editor. Pomocí něj lze bez nutnosti psaní SQL dotazů snadno upravovat databázi v rámci vytváření tabulek, atributů, integritních omezení a podobně.

5.2 Analýza aplikace

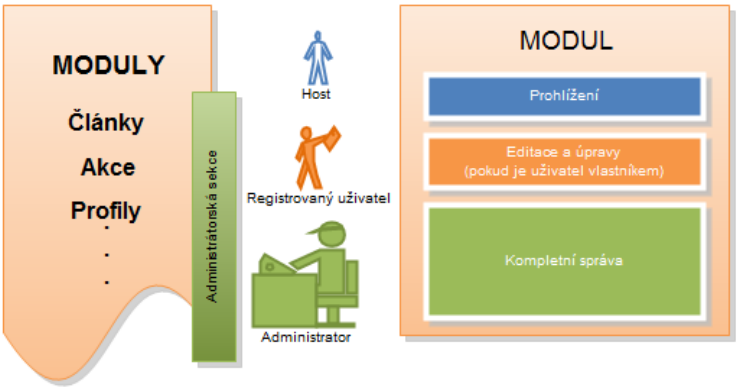
Protože webová aplikace musí splňovat zadané požadavky, bylo potřeba kvůli přehlednosti a snadné správě dodržet jednotnou architekturu. Proto byla aplikace navržena na třívrstvé architektuře, do které jsou zakomponovány jednotlivé moduly. Z těchto modulů se skládají různé části webové aplikace, jako je správce článků, uživatelské profily, fotogalerie a podobně. Webová aplikace je vytvořena v ASP.NET, pro programování samotné logiky aplikace byl použit jazyk C#. Vzhled stránek je stylován kaskádovými styly, nebo upravenými vlastnosti použitých komponent. Pro správu báze dat se využívá Microsoft SQL Server a komunikaci s touto databází obstarává ADO.NET[7].

5.2.1 Moduly

Webová aplikace je rozdělena do několika modulů, které musí nějaký uživatel spravovat, zde je má na starosti administrátor, který má přístup do administrátorské sekce,

[7] ADO.NET - Microsoft ActiveX Data Objects .NET

kde může vytvářet články, akce nebo schvalovat uživateli nahrané fotografie. Každý modul má možnost zpřístupnit registrovanému uživateli určité úpravy. Ale vždy záleží na konkrétním modulu, jaké úpravy jsou povoleny registrovanému uživateli. Nejlepším příkladem jsou samotné uživatelské profily, kdy modul rozpozná, že se jedná o registrovaného uživatele a současně je vlastníkem tohoto profilu, tudíž je mu dovoleno nahrávat fotografie a měnit osobní informace. Na následujícím obrázku (Obrázek 7) je vidět na jeho levé straně seznam modulů, nad kterými je postavena administrátorské sekce. Základní stavba modulu je navržena tak, aby vyhovovala konkrétním potřebám z hlediska úpravy obsahu administrátorovi, registrovanému uživateli nebo návštěvníkovi webových stránek.



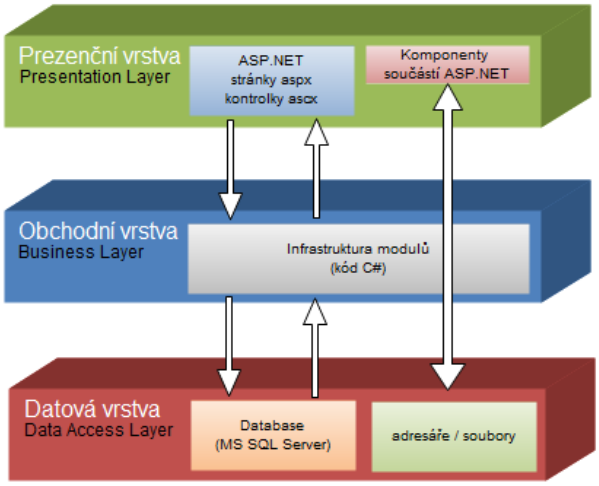
Obrázek 7 - Stavba modulů

5.2.2 Architektura aplikace

Webová aplikace je postavena na třívrstvé architektuře, její stavba je vidět na obrázku (Obrázek 8). Jak už sám název napovídá, aplikace se skládá ze tří vrstev a to:

- Prezenční vrstva,
- Obchodní vrstva,
- a Datová vrstva.

Komunikace mezi vrstvami se uskutečňuje vždy se sousedící vrstvou, jedinou výjimkou jsou hotové komponenty v ASP.NET. Mezi takové nepatří speciální komponenty, které při jejich použití automaticky komunikují s databází nebo ukládají soubory na webovém serveru.



Obrázek 8 - Architektura aplikace

Prezenční vrstva

Hlavním úkolem prezenční vrstvy je předat uživateli srozumitelnou formou obsah uložený na serveru. Opětovně musí i zaručit, aby uživatel mohl komunikovat s webovou aplikací, ale přitom nesmí uživateli dovolit jeho zásahem způsobit poškození uložených dat, proto je zde celá řada kontrolních opatření. Hlavním obsahem této vrstvy jsou webové stránky, tyto soubory používají příponu *aspx*. Do těchto souborů se definují prvky, které budou na stránce zobrazeny, s nimi jsou vždy svázány ještě další dva soubory. Důležitý pro práci je jeden z nich, a to soubor s příponou *cs*, skrz něj lze komunikovat s webovým souborem a jeho ovládacími prvky. V této vrstvě je umístěn i hlavní konfigurační soubor, kaskádové styly, mapa webu a další soubory související s webovou prezentací.

Obchodní vrstva

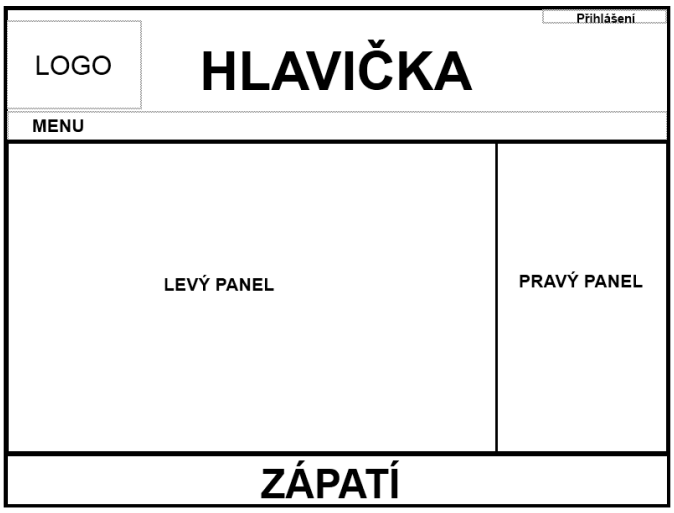
Obchodní vrstva známá také pod názvem aplikační vrstva slouží jako prostředník, mezi krajními vrstvami. Na první pohled by se mohla zdát zbytečná, protože vše by se dalo naprogramovat v samotné prezenční vrstvě, ale celý projekt by byl velmi nepřehledný a další modifikace by byly dále velmi obtížné. Obsahuje funkce, které zpracovávají data jak z datové, tak prezenční vrstvy, a na základě naprogramované funkcionality se rozhodne, jak s nimi dále naloží. Data, které poslala prezenční vrstva, nakonec nemusí do datové vrstvy dorazit, protože jsou zde další opatření, které se snaží zaručit integritu uložených dat.

Datová vrstva

Datová vrstva obstarává funkce spojené s ukládáním dat na straně webového serveru a komunikaci s databází.

5.2.3 Návrh základní stránky

Rozvržení je realizováno XHTML prvky, které jsou stylovány kaskádovými styly. XHTML prvky tvoří základní kostru stránky. Která je rozdělena na hlavičku s menu, logem a komponentou pro přihlášení uživatelů. Pod hlavičkou je umístěný levý a pravý panel. Levý panel se používá na zobrazení obsahu vybraných stránek z menu. Pravý panel je nyní prázdný a lze jej dále využít na naplnění dalším obsahem. Zápatí stránky obsahuje pouze informační text o práci.



Obrázek 9 - Rozložení základních prvků stránek

5.3 Základní konfigurace

Základní nastavení na úrovni aplikace se uskutečňuje pomocí souboru *web.config*. Celý obsah konfiguračního souboru ASP.NET je vnořen do kořenového prvku `<configuration>`. Tento prvek obsahuje prvek `<system.web>`. Zde jsou pak samotné prvky pro jednotlivé aspekty konfigurace, týkající se bezpečnosti, ladění, providery a dalších konfigurací. Pro vlastní nastavení je zde prvek `<appSettings>`. Do prvku `<connectionString>` můžeme vkládat naše připojovací řetězce k databázím.

5.3.1 Spojení s databází

Protože aplikace potřebuje komunikovat s databází, je zde nakonfigurován jeden poskytovatel členství, který obsahuje nastaveného providera. Tento provider v sobě zahrnuje konfigurační parametr *connectionStringName* a ten se dále odkazuje do prvku `<connectionString>`. Připojovací řetězec, spíše známý pod původním anglickým názvem connection string obsahuje údaje pro přístup do databáze. Jeho konfiguraci je na následujícím kódu, kde je konkrétní použité připojení v době programování webové aplikace.

```
<connectionStrings>
    <add name="xmaonlineConnectionString"
        connectionString="Data Source=TOMAS-PC\SQLEXPRESS;
                          Initial Catalog=xmaonline;
```

```
                    Integrated Security=True"
        providerName="System.Data.SqlClient"/>
</connectionStrings>
```

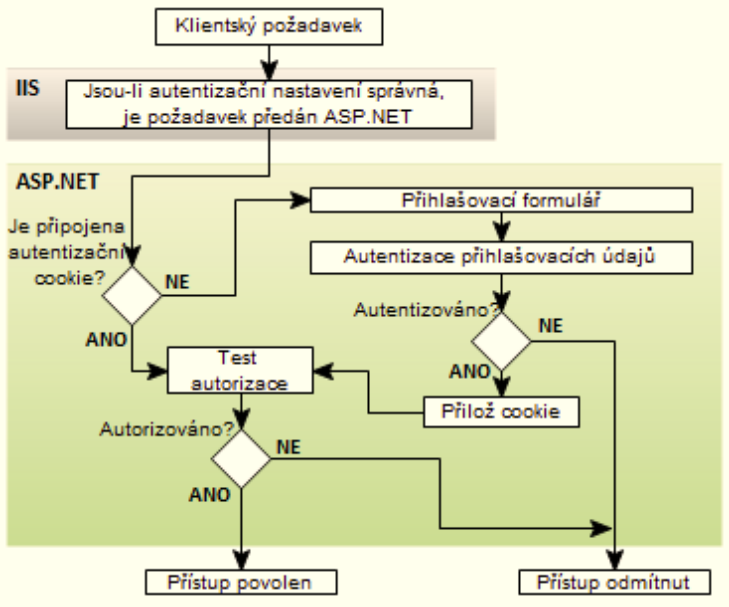
Možnost takovéto konfigurace přináší výhodu v době, kdy se aplikace nasazuje přímo na hosting. Pokud je jednou hotová konfigurace v souboru *web.config*, stačí pak vygenerovat připojovací řetězec přímo u provozovatele hostingu, kde se bude webová aplikaci provozovat. Tento vygenerovaný připojovací řetězec se zamění s původním, který se používal na lokálním počítači. Dále se konfigurační soubor nahraje na hosting, na lokálním počítači se nastaví zpět původní připojovací řetězec a není potřeba s tímto konfiguračním souborem nijak manipulovat. Bez problémů lze webovou aplikaci modifikovat a opětovně nasadit upravenou verzi do ostrého provozu.

5.3.2 Autentizace a autorizace

Do webové aplikace se mohou registrovat, a následně přihlašovat uživatelé. Z těchto důvodů se musí při přihlašování do aplikace ověřit jejich identita. Proto musí prvně dojít k autentizaci uživatele, což je proces, při kterém se zkoumá identita přihlašovaného uživatele, který se musí prokázat nějakými přihlašovacími údaji pro ověření identity. Následně přichází na řadu autorizace. Při tomto procesu se autentizovanému uživateli stanovují práva a omezení. K těmto účelům je možné využít různé autentizační systémy. První systémem je autentizace Windows, která předává zodpovědnost za autentizaci IIS, které požádá o autentizaci prohlížeč. Prohlížeč poskytne zpětně IIS přihlašovací údaje a pokud je autentizace uživatele úspěšná, IIS povolí žádost o webovou stránku a předá informace o roli ASP.NET. Formulářová autentizace je na rozdíl od autentizace Windows integrována v ASP.NET. Její systém je založen na šifrovaných cookie, ve kterých jsou uloženy informace o uživateli. V případě použití formulářové autentizace zůstává zodpovědnost za správu uživatelů na samotném správci aplikace, tedy na konkrétní osobě.

5.3.3 Formulářová autentizace

V praktické části je použita formulářová autentizace z nutnosti mít vlastní autentizační infrastrukturu spolu s uložištěm informací o registrovaných uživatelích. Proces autentizace je vidět na obrázku (Obrázek 10). Vstupní bránou k autentizaci uživatele je přihlašovací stránka, kde se vyplní přihlašovací jméno a heslo. Následně po odeslání údajů dochází k formulářové autentizaci na straně serveru, ta je založen na „lístcích“, které dostává uživatel po úspěšné autentizaci.



Obrázek 10 - Formulářová autentizace

V souboru *web.config* je použit prvek *<authentication>*, který má vlastnost *<mode>* nastavenou na hodnotu *Forms*, tímto způsobem se ASP.NET dozví, že se bude používat formulářová autentizace.

5.3.4 Členství, Role a Profily

ASP.NET poskytuje kompletní systém pro správu uživatelů, tento systém se jmenuje API členství. S jeho pomocí lze vytvářet, editovat nebo odstraňovat uživatele. Pro jednoduchou práci s tímto systémem lze použít konfigurační nástroj Web Site Administration Tool, který je vidět na následujícím obrázku (Obrázek 11).

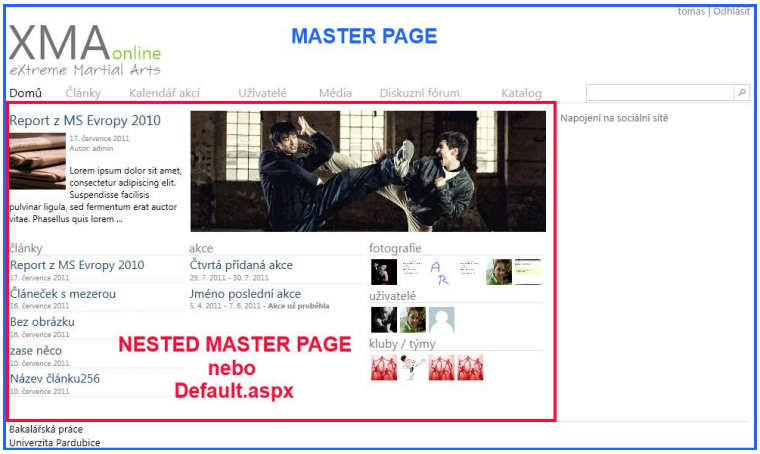
Obrázek 11 - Web Site Administration Tool

Pomocí této webové aplikace jsou vytvořeny dvě role. První role je pro administrátora a druhá pro zaregistrované uživatele. Kromě vytváření uživatelských rolí a správy uživatelů jsou zde i další možnosti nastavení. Jako je typ Role Providera, Membership Providera, zamezení přístupu konkrétním uživatelům nebo vytvořeným rolím do určitého adresáře webové aplikace.

Je zřejmé, že s pomocí Web Site Administration Tool by bylo vytváření každého uživatele velmi pracné. Proto je v ASP.NET několik hotových komponent, které se dají vložit na webovou stránku vytvářené webové aplikace, kde každý uživatel vyplní registrační formulář a systém API členství se o vše postará. Nabízené komponenty z ASP.NET ve většině případů nevyhovují, a proto je třeba udělat konkrétní modifikace. Případně se nabízí možnost napsat vlastní komponenty a implementovat do nich API členství, rolí a profilů.

5.4 Master Page a Nested Master Page

Jelikož se webový portál skládá z různých stránek, na kterých se opakuje stejný obsah, jako je menu, logo a podobně, lze si nadefinovat jednu základní stránku. Tato stránka bude pořád stejná, a pouze v místě kde se má měnit obsah, se bude vkládat nová stránka s konkrétním obsahem. Jak je vidět na následujícím obrázku (Obrázek 12), v modrém orámování se nachází základní stránka s neměnným obsahem, a do ní jsou na místo v červeném orámování následně vkládány další stránky s požadovaným obsahem, který si uživatel zvolí z menu.



Obrázek 12 - Master Page a Nested Master Page

5.4.1 Master Page

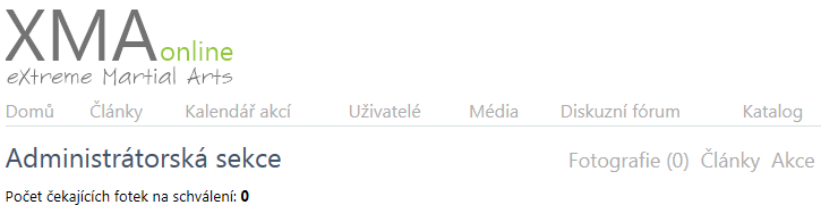
ASP.NET počítá s tím, že většina webů je řešena základní stránkou, která se nemění. A pro tyto účely je výhodné použít Master Page[8], kde se nadefinuje základní obsah stránky spolu s místem, do kterého se bude načítat dále požadovaný obsah. Samostatnou stránku Master Page nedokáže prohlížeč zobrazit, protože se jedná o jakousi

[8] http://interval.cz/clanky/aspnet-20-master-pages/

šablonu. Do této stránky je zapotřebí pomocí speciální komponenty Content Place Holder[9] nadefinovat, kde se bude zobrazovat požadovaná stránka. Aby byla tato stránka začleněna do Master Page, je třeba ji propojit přes Content Place Holder z Master Page.

5.4.2 Nested Master Page

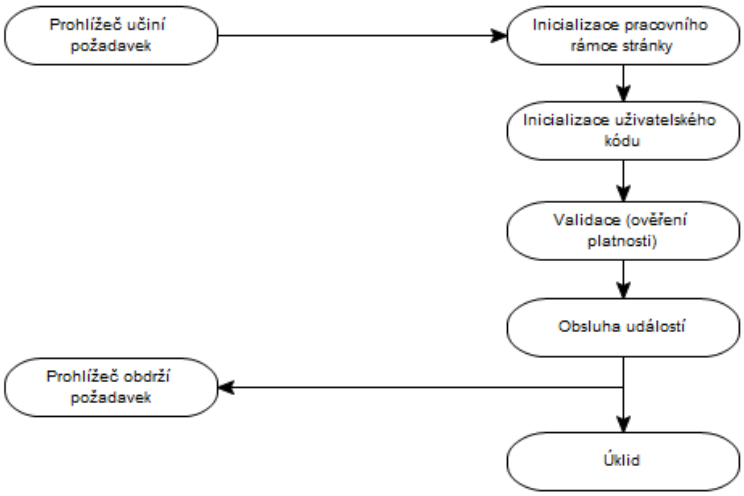
Nested Master Page je téměř totožná s běžnou Master Page, Používají se k dalšímu rozdělení stránky. Jsou založeny na stejném principu jako Master Page, ale na rozdíl od Master Page se dají použít na jednom webovém portálu vícekrát. To se samozřejmě hodí, pokud je potřeba vytvořit stránku obsahující další neměnné prvky, a pouze v konkrétní části by se měnil obsah tvořený dalšími stránkami. Příkladem mohou být další nabídky, jak je vidět na následujícím obrázku (Obrázek 13). V základní Master Page je vnořený Nested Master Page s menu pro sekci administrátora. A do Nested Master Page se opět vkládají stránky vybrané z menu.



Obrázek 13 - Nested Master Page

5.5 Životní cyklus stránky

Webová stránka v ASP.NET se zpracovává na serveru po etapách. V každé etapě se volají různé události, což umožňuje, aby se stránka zapojila do toku zpracování a mohla zareagovat, jak uznáme za vhodné.



Obrázek 14 - Životní cyklus stránky

[9] http://msdn.microsoft.com/cs-cz/library/system.web.ui.webcontrols.contentplaceholder.aspx

5.5.1 Inicializace pracovního rámce

V této první etapě dochází k vytvoření stránky a probíhá vygenerování ovládacích prvků definovaných na webové stránce. Dále dochází k vyvolání události *Page.Init*. Událost *Page.Init* jsem použil jen zřídka, protože je ještě velmi brzy na vykonání jakékoliv práce s komponentami vloženými na webové stránce. Ale občas jsem ji využil k vytvoření instance třídy *SqlConnection*, nebo nějaké jiné. V praktické části byla pro ilustraci událost *Page.Init* použita pro vytvoření instance třídy *EventManagemant*. Implementace je vidět na následujícím kódu.

```
protected void Page_Init(object sender, EventArgs e)
{
     this.actionManagemant = new EventsManagemant(Server.MapPath(
                                                  eventsRootDirectory));
}
```

5.5.2 Inicializace uživatelského kódu

Tato etapa je pro praktické použití nejdůležitější, dochází k vyvolání události *Page.Load*, která se využívá k inicializaci ovládacích prvků. V tuto chvíli lze nastavit ovládacím prvkům vlastnosti dle konkrétních potřeb. Například vyplnit textová pole, nebo provádět libovolné konfigurace.

Protože se událost *Page.Load* vyvolá vždy, bez ohledu na to, zdali se stránka požaduje poprvé, nebo už došlo k opětovnému přeposlání stránky, je třeba rozlišit mezi prvním načítáním stránky od následných načítání. To je velmi důležité, protože stav zobrazení se udržuje automaticky po načtení z nějakého dynamického zdroje pouze z prvního načítání stránky. Aby bylo možné určit aktuální stav stránky, je zde zavedena statická vlastnost *IsPostBack*, která je při prvním požadavku na server o stránku nastavena na hodnotu *false*.

```
protected void Page_Load(object sender, EventArgs e)
{
      if (Page.IsPostBack)
        DosloKPostBacku();
      else
        NastaveniHodnotPriPrvnimNacteni();
}
```

V praktické části je se využívá tato vlastnost velmi často, bez kontroly postbacku by byla práce s ovládacími prvky velmi obtížná. Sice v pozdějších volaných událostech ASP.NET nebo nějakého tlačítka, by bylo možné odchytit stav ovládacích prvků, ale už by bylo pozdě, protože událost *Page.Load* by stihla provést nastavení, které jsme si sami naprogramovali pro první spuštění stránky.

5.5.3 Validace

ASP.NET obsahuje ovládací prvky, které slouží k ověření platnosti hodnot vstupních ovládacích prvků, a můžou následně reagovat zobrazením chybové zprávy. Použité validační prvky v praktické části, jsou soběstačné a není třeba ošetřovat validační

události. Takovéto ovládací prvky jsou například implementovány následujícím kódem v přihlašovací stránce.

```
<asp:TextBox ID="UserName" runat="server" Width="150px"></asp:TextBox>
<asp:RequiredFieldValidator ID="UserNameRequired" runat="server"
  ControlToValidate="UserName" ErrorMessage="User Name is required."
  ToolTip="Uživatelské jméno je povinné."
  ValidationGroup="Login1">*</asp:RequiredFieldValidator>
```

Předchozí kód definuje editační pole *TextBox*, s jeho identifikačním jménem *UserName*, hned za ním je definován validátor, který má vlastnost *ControlToValidate* nastavenou na identifikátor pro *TextBox*. Takto se dá validátoru vědět, jaký ovládací prvek bude kontrolovat. *RequiredFieldValidator* kontroluje, zda uživatel editační pole vyplnil, v případě kdy bude žádoucí validovat i jiné vlastnosti, je možné přidat jiný validátor.

5.5.4 Obsluha událostí

Tato část byla zpočátku problematická, než jsem dostatečně pochopil kdy ASP.NET danou událost volá. V praktické části jsem nejčastěji použil události:

- Page.Init,
- Page.Load,
- událost ovládacího prvku,
- Page.PreRender.

Události se provádí ve stejném pořadí, jak jsem je výše uvedl. Ze seznamu událostí je vidět, že události ovládacích prvků se provádějí až po provedení události *Page.Load*, na to je třeba si dát pozor, a používat vlastnost *IsPostBack*, aby se zabránilo opětovným změnám na výchozí hodnoty nastavené při inicializaci nebo zbytečnému dotazování do databáze v události *Page.Load*. Příkladem z praktické části je vytvořená komponenta *ControlAction*, která při spuštění v režimu editace kontroluje, zda má nějaká sportovní událost nastavenou úvodní fotografii, jméno a ostatní hodnoty. Tato kontrola se provádí v případě, že nedošlo k postbacku, tedy jde o první načtení stránky. Pomocí následující podmínky se zkontroluje a dojde k nastavení všech ovládacích prvků na požadované hodnoty.

```
if (!Page.IsPostBack)
{
    SetAllComponents();
}
```

Pak je možné provést potřebné změny a kliknout na tlačítko Uložit. Dojde k postbacku a události se spustí znovu. V události *Page.Load* se při kontrole podmínky rozpozná, že se jedná o postback, zbytečně se nebude vykonávat spousta akcí spojených s prvním nastavením ovládacích prvků, a hlavně nedojde k přepsání hodnot, které se v ovládacích prvcích provedly. Protože bylo kliknuto na tlačítko, vyvolá se dále jeho událost, kde se provede načtení nových hodnot z ovládacích prvků a dojde k jejich změně i v uložištích.

V události *Page.PreRender* se mohou udělat ještě nějaké změny v nastavení ovládacích prvků, protože události ovládacích prvků mohly ovlivnit výsledek, který se má uživateli zobrazit. Takovým příkladem může být kliknutí na tlačítko pro odstranění fotografie. V obsluze události tlačítka se fotografie odstraní z uložiště, ale ASP.NET o tom neví, protože podle událost *Page.Load* při prvním spuštění byla fotografie k dispozici a komponenta umožňující nahrávání fotografií byla skryta. Tudíž po odstranění budeme chtít tuto komponentu pro nahrávání fotografií zobrazit, což lze právě ošetřit v události *Page.PreRender*, kde se uskuteční kontrola, zda opravdu není fotografie k dispozici. Pokud není, nastaví se komponentě pro nahrávání fotografií viditelnost a vše je v pořádku.

5.6 Třída Page

Kromě užitečné vlastnosti *IsPostBack* třídy *Page* zmíněné výše, jsou zde další užitečné vlastnosti, které jsem v projektu využil.

5.6.1 Object Request

Je instancí třídy *Systém.Web.HttpRequest* a reprezentuje hodnoty a vlastnosti HTTP požadavku. Využil jsem parametrů URL v téměř každé části webu, které tento objekt nese, když bylo třeba zobrazit profil uživatele, konkrétní článek a podobně. Jeho použití je velmi jednoduché, pokud se zažádá o stránku s nějakým parametrem, kterým může být například profilové jméno uživatele, je tento parametr uložen v URL a tím pádem o něm ví i *Request*. Samotné přečtení a uložení parametru můžeme provést pomocí následujícího kódu.

```
String strUserTmp = Request.QueryString["User"];
```

5.6.2 Object Response

Je instancí třídy *Systém.Web.HttpResponse* a reprezentuje odpověď webového serveru na požadavek klienta. Z tohoto objektu jsem často využíval metodu *Redirect()*, která umožňuje uživatele poslat na jinou stránku. To je výhodné ve chvíli, kdy je třeba ošetřit, aby se uživatel nedostal na stránku, kam nemá mít přístup. Na následujícím kódu je v podmínce kontrola, zda je uživatel přihlášený, v případě že tomu tak není, je automaticky přesměrován na úvodní stránku.

```
if (HttpContext.Current.User.Identity.IsAuthenticated)
{
     Response.Redirect("~/Default.aspx");
}
```

5.7 Web User Control

Hlavním důvodem programování vlastní komponenty je opětovná použitelnost na více místech ve webové aplikaci, a nejen v jednom samotném webovém projektu, ale je možnost dalšího využití v dalších webových aplikací. Vlastní komponenta se musí vždy zaregistrovat do stránky, kde se bude používat. Na následujícím kódu je vidět způsob registrace komponenty, kde je uvedený zdroj, kterým je samotná komponenta, druhá

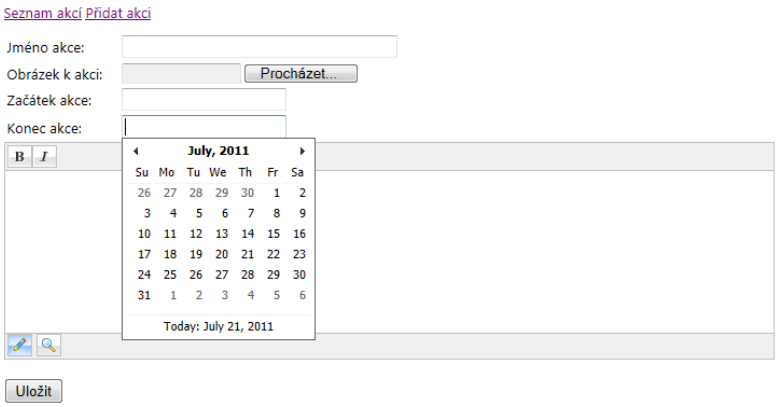
vlastnost nastavuje pojmenování prefixu, ze kterého budeme komponentu vybírat, a poslední vlastnost je samotné pojmenování pro vytvořenou komponentu.

```
<%@ Register Src="~/Controls/ControlAction.ascx" TagPrefix="myControl"
 TagName="ControlAction" %>
```

Komponenta je nyní ve stránce zaregistrována a nyní ji stačí jen použít ve stránce. Aby bylo možní s ní pracovat i v kódu, je třeba uvést vlastnost *ID*.

```
<myControl:ControlAction runat="server" ID="ControlAction" />
```

Předchozí ilustrační kód je použitý pro komponentu obstarávající vytváření akcí, které se při zveřejnění administrátorem objeví v kalendáři akcí, dostupného v hlavním menu. Vzhled vlastní komponenty je vidět na následujícím obrázku (Obrázek 15).



Obrázek 15 - Komponenta akce

Komponenta obstarává základní operace pro vytváření a úpravu dané akce. Rozhodování, v jakém režimu nyní komponenta běží, je řešeno následujícím výčtovým typem.

```
public enum TypeEvents
{
    Add, Edit
}
```

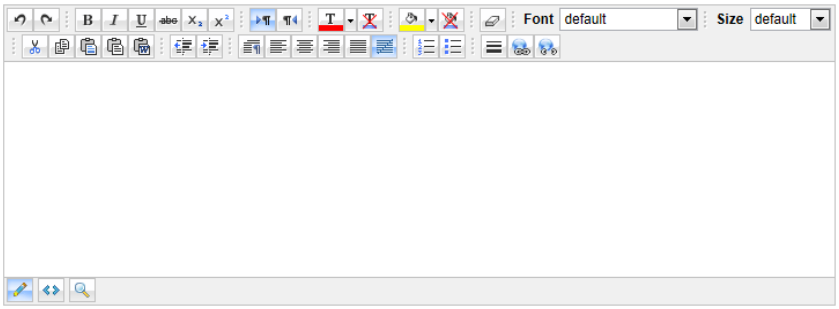
5.8 Ajax Control Toolkit

Ajax Control Toolkit obsahuje bohatou sadu ovládacích prvků, které se dají použít ve webových formulářích ASP.NET. Ajax Control Toolkit je přidán do Visual Studia připojením knihovny AjaxControlToolkit.dll k referencím v projektu.

5.8.1 Komponenta *Editor*

Jedná se o klasický WYSIWYG HTML editor. Tato komponenta je velmi rozsáhlá a obsahuje spoustu prvků pro úpravu textu, proto je použita pro psaní článků. V dolní liště na levé straně má tři ikonky. Tyto ikonky slouží k přepínání způsobu výpisu vloženého textu do editoru. Důležitá je zde prostřední ikonka, ta přepne editor do režimu, kde je text

editován s HTML značkami, které se právě využívají k nastavení fontu, vykreslení tabulek a podobně.



Obrázek 16 - WYSIWYG HTML Editor

5.8.2 Komponenta LiteEditor

Pro psaní sportovních akcí se používá méně vybavený WISIWIG HTML editor. Protože u akcí není potřeba využívat tolik rozsáhnout komponentu, jako pro psaní článků, byla pro tyto akce napsána vlastní komponenta, která je vidět na následujícím obrázku (Obrázek 17).

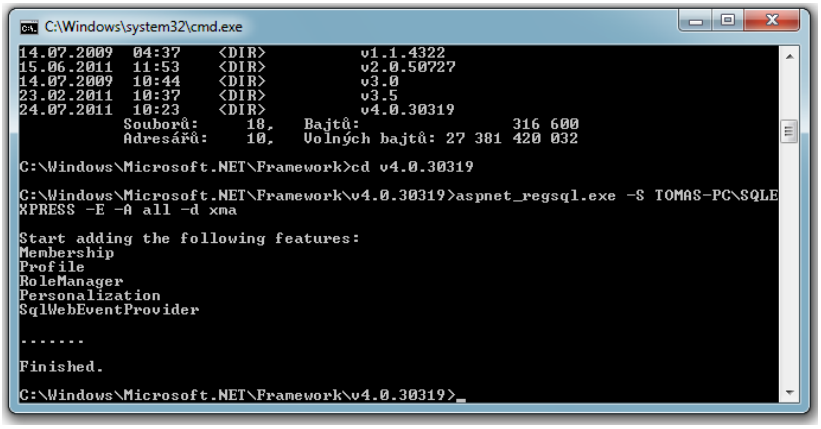
Obrázek 17 - WYSIWYG HTML LiteEditor

5.9 Databáze

Implementace na straně databáze je komplikovanější, protože se v projektu používá *SqlMembershipProvider*. Na straně SQL Serveru bylo třeba vytvořit databázi, tabulky a procedury potřebné k běhu API členství. Nabízejí se dva způsoby, jak takovou databázi vytvořit. Prvním je využít SQL skripty poskytované od ASP.NET, které umožní ručně vytvořit potřebnou databázi. Druhou možností je použít nástroj *aspnet*_regsql, který vše potřebné obstará sám. K vytvoření potřebné databáze byl vybrán druhý způsob, a to nástroj *aspnet_regsql*. Spuští se z příkazové řádky následujícím příkazem.

```
aspnet_regsql -S (local)\SQLEXPRESS -E -A all -d JmenoDatabaze
```

Na následujícím obrázku (Obrázek) je vidět konkrétní použití příkazu pro lokální SQL Server Express.



Obrázek 15 - Instalace databáze

V příloze (Příloha C) je diagram s tabulkami vytvořené databáze, na první pohled zaujme velké množství tabulek. Takto rozsáhlé řešení je zbytečné pro menší projekty, které využívají jen základní funkce API členství. V takových případech je možné použít providery třetích stran, které jsou upravena jen pro základní potřeba registrace a správy uživatelů.

Aby se zbytečně nevytvářely nové tabulky pro informace o uživatelích, byly do vygenerované tabulky aspnet_Users přidány následující atributy:

- Avatar,
- UserDescription,
- a RegDate.

Databáze byla doplněna o další potřebné tabulky pro ukládání dat. Jelikož vygenerované tabulky používaly u primárních klíčů datový typ *uniqueidentifier*[10], rozhodl jsem se tento datový typ použít i pro mnou vytvořené primární klíče. E-R diagram používaných tabulek je v příloze (Příloha B).

Tabulka 3 - Popis tabulek

Tabulky	popis
Photos	Obsahuje informace o nahraných fotografiích samotnými uživateli. Kdo fotografii nahrál, datum nahrání, typ obrázku (BMP, JPG, PNG) a zda je fotka schválena.
CTProfile	Obsahuje informace o registrovaných klubech, týmech a na jaký bojový styl jsou zaměřeni.
CategorieMA	Seznam stylů bojových sportů.
Articles	Obsahuje informace o článcích. Kdo je autorem, zda je článek publikován, kdy byl vytvořen, obsah článku a úvodní fotografii.
Events	Obsahuje informace o sportovních akcích. Podobné článkům, ale je zde přidán start a konec akce.
CategorieEA	Obsahuje možné kategorie pro roztřídění akcí a článků.
Ostatní tabulky	Jsou vygenerovány nástrojem aspnet_sqlreg nebo mezi tabulky pro vyřešení vztahů M:N.

[10] http://msdn.microsoft.com/en-us/library/ms187942.aspx

6 Závěr

Při zpracování teoretické části jsem se dozvěděl ze zde uvedené literatury mnoho zajímavých informací. Nejvíce mě však zaujala část zabývající se návrhem databáze a pohled Roberta Vieira na normalizaci, kde doporučuje držet se zdravého rozumu a nesnažit se za každou cenu striktně dodržovat teorie, které se píší v literatůrách. Přesto, že je tato práce převážně zaměřena na technologie, byl jsem nucen v prvním úseku teoretické části, kde je srovnání již fungujících webových portálů se zaměřit na samotný obsah a zpracování. Protože jsem na žádost o zaslání podrobnějších informací ohledně využitých technologií na vybraných portálech neobdržel odpověď.

Cílem praktické části bylo vytvořit webový portál se zaměřením na bojová umění. Tuto část se mi podařilo splnit v zadaném rozsahu. Na této části jsem si rozšířil znalosti s programováním v ASP.NET, jazyku C#, vytvářením vlastních komponent a možnosti práce s databází pomocí jazyka SQL a Transact-SQL. Nyní se webový portál nachází ve funkčním stavu a byl otestován na webhostingu ASPone[11]. Do budoucna je možné portál rozšířit o další moduly a nasadit na ostrý běh, aby byl přístupný veřejnosti.

[11] www.aspone.cz

Literatura

1. Apache HTTP SERVER PROJECT. *Apache.* [Online] [Citace: 7. 7 2011.] http://httpd.apache.org/.

2. **Brown, Martin.** IIS vs. Apache, Looking Beyond the Rhetoric. *ServerWatch.* [Online] 10. Leden 2008. [Citace: 7. Červenec 2011.] http://www.serverwatch.com/tutorials/article.php/3074841/IIS-vs-Apache-Looking-Beyond-the-Rhetoric.htm.

3. **Vieira, Robert.** *SQL Server 2000 Programujeme profesionálně.* Praha : Computer Press, 2001. ISBN 80-7226-506-7.

4. **Castro, Elisabeth.** *HTML, XHTML a CSS - Názorný průvodce tvorbou WWW stránek.* Brno : Computer Press, 2007. ISBN 978-80-251-1531-2.

5. **Buranský, Imrich.** *HTML a DHTML - hotová řešení.* Brno : Computer Press, 2003. ISBN 80-7226-841-4.

6. **Písek, Slavoj.** *HTML a XHTML - začínáme programovat.* Praha : Grada Publishing, 2003. ISBN 80-247-0571-0.

7. **Broža, Petr.** *Programování WWW stránek pro úplné začátečníky.* Praha : Computer Press, 2000. ISBN 80-7226-278-5.

8. **Hauser, Marianne, Hauser, Tobias a Wenz, Christian.** *HTML a CSS - Velká kniha řešení.* Praha : Computer Press, 2006. ISBN 80-251-1117-2.

9. **Schafer, Steven M.** *HTML, XHTML a CSS - Bible pro tvorbu WWW stránek.* Praha : Grada Publishing, a. s., 2009. ISBN 978-80-247-2850-6.

10. **Škultéty, Rastislav.** *JavaScript Programujeme internetové aplikace.* Brno : Computer Press, 2004. ISBN 80-251-0144-4.

11. **Holzner, Steven.** *Mistrovství v AJAXu.* Brno : Computer Press, a. s., 2007. ISBN 978-80-251-1850-4.

12. **Kosek, Jiří.** *PHP - tvorba interaktivních internetových aplikací.* Praha : Grada Publishing, spol. s r.o., 2009. ISBN 80-7169-373-1.

13. Common Language Runtime. *Wikipedia.* [Online] 11. Červenec 2011. [Citace: 18. Červenec 2001.] http://en.wikipedia.org/wiki/Common_Language_Runtime.

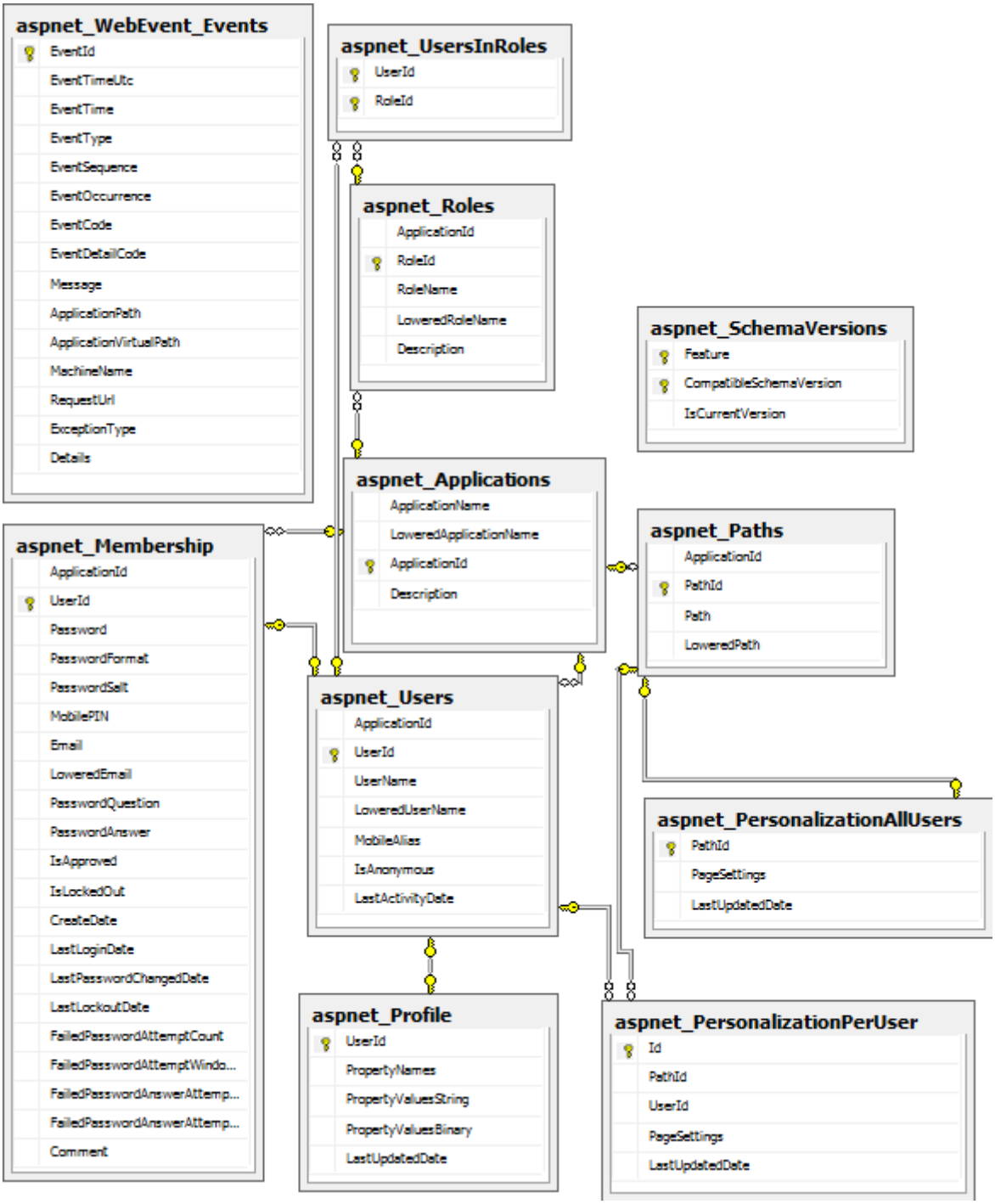
14. **MacDonald, Matthew a Szpuszta, Mario.** *ASP.NET 3.5 a C# 2008 tvorba dynamických stránek PROFESIONÁLNĚ.* Brno : Zoner Press, 2008. ISBN 978-80-7413-008-3.

15. **Kaluža, Radovan.** Konceptuální datový model | owebu.cz , webhosting , domény , hosting. *owebu.cz.* [Online] 7. Září 2004. [Citace: 4. Červenec 2011.] http://owebu.bloger.cz/Databaze/Konceptualni-datovy-model.

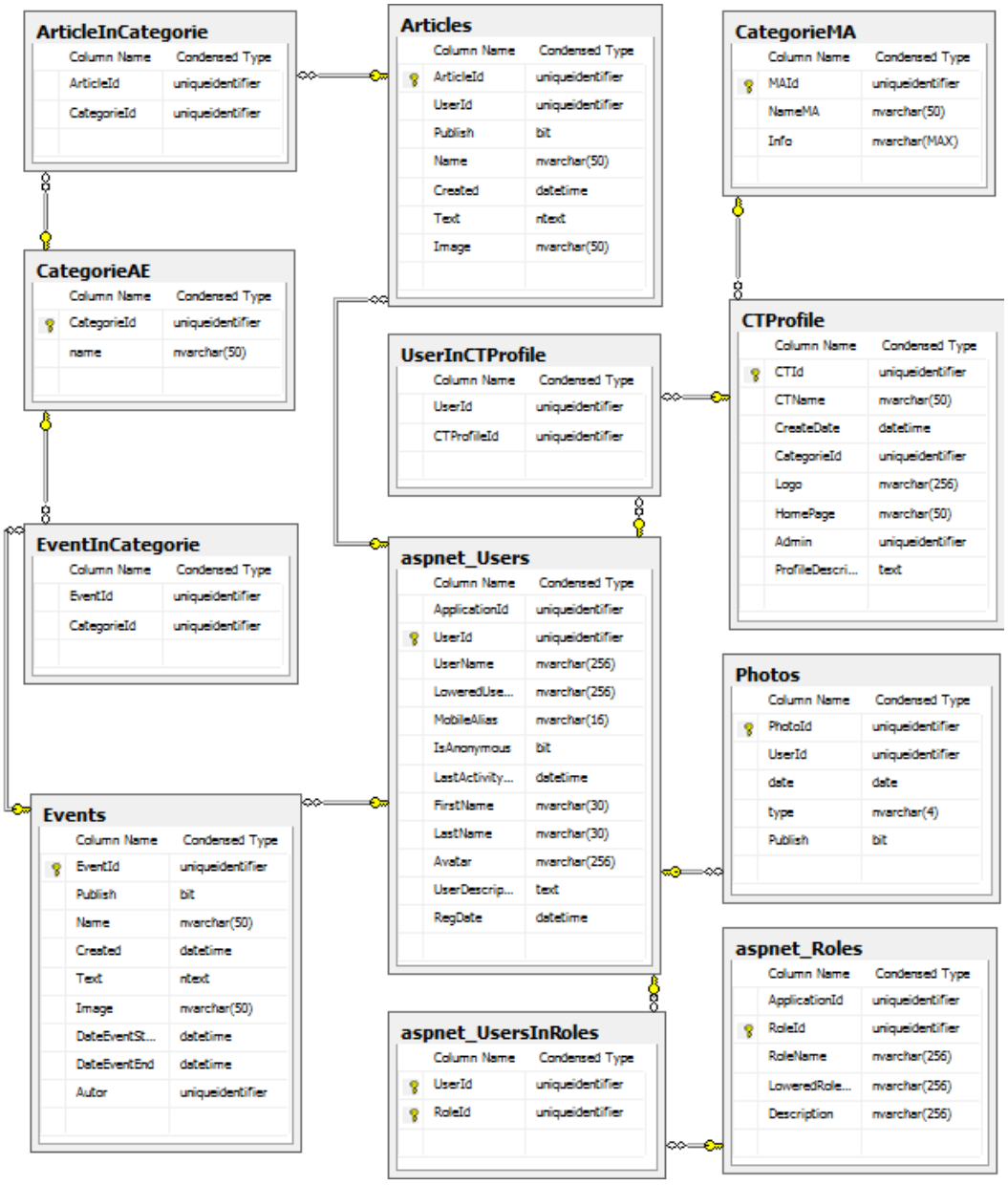
16. Konceptuální datový model. *owebu.cz.* [Online] 7. Září 2004. [Citace: 27. Červenec 2001.] http://owebu.bloger.cz/Konceptualni-datovy-model.

17. **Zendulka, J.** *Fakulta informačních technologií VUT v Brně.* [Online] [Citace: 4. Červenec 2011.] http://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/2_kmod.pdf.

Příloha A – Tabulky pro členství, role a profily



Příloha B – E-R diagram



Příloha C – Využití technologií vzhledem k velikosti řešení

Typ aplikace	JAVA / J2EE	GROOVY	PHP	RUBY	PYTHON	.NET
Webová prezentace s převažujícím statickým obsahem (bez redakčního systému)	nevhodné Větší náklady na vývoj	nevhodné Větší náklady na vývoj	vhodné Rychlá a snadná implementace	vhodné Rychlá a snadná implementace. Podobné charakteristiky jako PHP	vhodné Rychlá a snadná implementace. Podobné charakteristiky jako PHP	nevhodné Větší náklady na vývoj
Aplikace s krátkou životností - microsite (bez redakčního systému)	nevhodné Větší náklady na vývoj	nevhodné Větší náklady na vývoj	vhodné Rychlá a snadná implementace	vhodné Rychlá a snadná implementace. Podobné charakteristiky jako PHP	vhodné Rychlá a snadná implementace. Podobné charakteristiky jako PHP	nevhodné Větší náklady na vývoj
Menší aplikace se speciální či specifickou funkcionalitou	velmi vhodné Perfektní při napojení na externí systémy, podpora mnoha druhů exportů, tisků a perfektní zabezpečení aplikací	velmi vhodné Rychlý vývoj. Přebírá vlastnosti J2EE technologií a lze s nimi kombinovat. Podpora exportů do PDF a dalších formátů, perfektní zabezpečení aplikací	vhodné Možné problémy při komunikaci s externími systémy. Podpora exportů do PDF a dalších formátů. Možné problémy s bezpečností	velmi vhodné Rychlý vývoj. Dobré zabezpečení. Umí komunikovat s externími systémy. Podpora exportů do PDF a dalších formátů	vhodné Možné problémy při komunikaci s externími systémy. Podpora exportů do PDF a dalších formátů. Možné problémy s bezpečností	velmi vhodné Perfektní při napojení na externí systémy. Umí exporty a dokáže perfektně zabezpečit aplikace
Středně velká aplikace (menší informační systémy, publikační systémy nebo CMS)	velmi vhodné Architektura pomůže s vývojem aplikace. Poměrně snadná údržba a rozšiřitelnost aplikací	velmi vhodné Poměrně rychlý vývoj a snadná údržba. Přebírá vlastnosti J2EE technologií a lze s nimi kombinovat	méně vhodné Nutno použít MVC framework. Problémy při úpravách, udržování aplikace a vývoje v týmu	méně vhodné Nutno použít MVC framework. Problémy při úpravách, udržování aplikace a vývoje v týmu	méně vhodné Nutno použít MVC framework. Problémy při úpravách, udržování aplikace a vývoje v týmu	velmi vhodné Architektura pomůže s vývojem aplikace. Poměrně snadná údržba a rozšiřitelnost aplikací
Aplikace s dlouhou životností (redakční systém, CMS, elektronický obchod nebo informační systém)	velmi vhodné Aplikace složitější a dražší na vývoj. Úpravy a udržování poměrně snadné a levné	velmi vhodné Poměrně rychlý vývoj a snadná údržba. Lze kombinovat s Java technologiemi. Možné problémy s rošiřitelností	méně vhodné Levné vytvoření, ale obtížné úpravy a udržování aplikace	méně vhodné Pro menší aplikace použitelné	méně vhodné Pro menší aplikace použitelné	velmi vhodné Aplikace složitější a dražší na vývoj. Úpravy a udržování poměrně snadné a levné
Rozsáhlý systém (informační sytém, velký CMS, CRM, DMS nebo velmi výkonné aplikace)	velmi vhodné Technologie určena pro rozsáhlé projekty. Efektivní vývoj i údržba. Podpora pro vysoce výkonné aplikace	méně vhodné Mohou nastat problémy z hlediska efektivity, přehlednosti a rozšiřitelnosti	nevhodné Komplikovaný vývoj z hlediska efektivity a přehlednosti	nevhodné Komplikovaný vývoj z hlediska efektivity a přehlednosti	nevhodné Komplikovaný vývoj z hlediska efektivity a přehlednosti	velmi vhodné Technologie určena pro rozsáhlé projekty. Efektivní vývoj i údržba. Podpora pro vysoce výkonné aplikace

Zdroj: http://www.morosystems.cz/porovnani-technologii-pro-webova-reseni

Příloha D – Přiložené CD

Přiložené CD obsahuje text bakalářské práce, zdrojové kódy a publikovanou verzi webového portálu spolu s potřebnými SQL skripty pro zprovoznění databáze na SQL Serveru.