Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

Model systému pro správu obsahu založeného na technologii ASP.NET

Bakalářská práce

Martin Hlaváč

Vedoucí práce:

PhDr. Milan Novák, Ph.D.

ČESKÉ BUDĚJOVICE 2010

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne 23.4.2010

Abstrakt

Cílem této bakalářské práce je navrhnout a vytvořit univerzální model internetového systému pro správu obsahu (CMS) založeného na technologii ASP.NET. Systém by měl být plně modulární a měl by zohledňovat současné technologické i marketingové trendy. Součástí práce je ukázková aplikace CMS, podle navrženého modelu.

Abstract

The goal of this thesis is to design and create a universal model of a web content management system (CMS), which is based on the ASP.NET technology. The system should be fully modular and should reflect the current technological and marketing trends. A Part of this thesis is sample application of the CMS, according to the proposed model.

Obsah

1. Úvod

Publikování obsahu na internetu je velice rozsáhlé téma. Internet se stal součástí našeho života, stále se mílovými kroky vyvíjí a nejen v komerční sféře je nutné mít výhodu oproti konkurenci, kterou může zajistit kvalitní webová prezentace. Základem kvalitní webové prezentace je dynamicky generovaný obsah s příjemným uživatelským rozhraním, což zajistí systém pro správu obsahu, zkráceně CMS (z anglického Content Management System). Dobrý CMS by měl být snadno použitelný i ovladatelný. V současné době se jako vhodný směr jeví systémy, jejichž funkčnost je založena na modularitě, kde jednotlivé funkce obstarávají samostatné moduly, čímž je dosaženo velké univerzálnosti.

Vhodným řešením pro realizaci CMS je technologie ASP.NET, za kterou stojí silná softwarová společnost Microsoft i velká komunita vývojářů. Tato technologie umožňuje využít všechny výhody prostředí .NET a dále přidává další nástroje určené do webového prostředí. Pomocí volně dostupných knihoven a nástrojů je možné rozšířit základní možnosti o podporu technologie AJAX nebo využívat tzv. „Cleans URL“.

Cílem této práce je navrhnout univerzální model systému pro správu obsahu a ten následně odzkoušet na praktickém řešení. Výsledný systém by měl být modulární a jeho nasazení univerzální.

Práce je rozdělena na dvě části. První část se zaměřuje na návrh univerzálního modelu systému pomocí návrhových vzorů UML a popis jednotlivých technologií. Druhá část popisuje implementaci navrženého modelu, jeho realizaci v prostředí ASP.NET a dále obsahuje postup a řešení všech dalších použitých nástrojů včetně jejich implementace.

Realizace praktického řešení vychází z vytvořených návrhových vzorů UML a jako demonstrace řešení je vytvořena funkční prototypová aplikace v prostředí ASP.NET. V práci jsou využity současné technologické i marketingové trendy obzvláště s důrazem na technologii AJAX a SEO praktiky.

2. Vymezení problematiky

Pro pochopení problematiky publikování obsahu na internetu je nejprve nezbytné vysvětlit některé základní pojmy a alespoň ve stručnosti představit jednotlivé technologické a marketingové techniky i trendy.

2.1 Publikační systém

Systém pro správu obsahu (CMS), neboli také často používané označení publikační systém či redakční systém, je určitý software, který umožňuje spravovat teoreticky libovolný obsah v závislosti na možnostech daného řešení. CMS je nejčastěji využíván pro správu webového obsahu, ovšem nejedná se o nutnou podmínku. Prostředí bývá rozdělené na uživatelskou (Front-End) a administrátorskou (Back-End) část. Uživatelská část je určena široké veřejnosti a slouží k zobrazení obsahu. Administrátorská část je určena k veškeré obsluze obsahu i systému a měla by být dostupná pouze přihlášeným uživatelům s patřičným oprávněním. Mezi základní administrátorské funkce CMS patří správa jednotlivých sekcí systému, např. správa obsahu, uživatelů, nastavení atd. a v jednotlivých sekcích patřičné operace, např. přidání nových záznamů a editace, či smazání stávajících. Moderní CMS by měl být modulární, to znamená, že jednotlivé části systému by měli být samotné části, tzv. moduly, což umožňuje maximálně univerzální nasazení systému (1).

2.2 Technologické trendy

Výběr použitých technologií je velmi důležitý, naštěstí je z čeho vybírat. Jako vhodný kandidát základní platformy se jeví např. prostředí .NET od firmy Microsoft, resp. ASP.NET (2). Pohodlný přístup a ovládání databáze pak programátorovi zařídí technologie LINQ to SQL. Na straně klienta se v poslední době velmi používá populární AJAX, který umožňuje vytvářet a používat nejen líbivé efekty, ale především i asynchronní přístup k datům. Jako vhodný model a způsob řešení se také jeví modulární systém, díky čemuž je možné dosáhnout velké univerzálnosti především v nasazení systému. Takový systém je vyhovující pro využití libovolného scénáře, protože v případě potřeby může být naprogramován modul požadované činnosti. Samozřejmostí by měl být validní xHTML a CSS kód. Dobrým doplňkem kvalitního systému je multi-jazyčnost, resp. možnost vkládat více jazyčný obsah, včetně možnosti přepínání jednotlivých jazyků pro registrované i anonymní uživatele.

2.3 Marketingové trendy

Různé marketingové trendy přispívají ke zvýšení návštěvnosti i kvality webových stránek, proto jsou jejich důležitou součástí. Aby byl CMS úspěšný i po předání zákazníkovi, je nutné už při vývoji myslet a dbát pravidel SEO optimalizace, resp. optimalizace pro internetové vyhledavače. Cílem je dosáhnout při vyhledávání v internetových fulltextových vyhledávačích co nejvyšších pozic a tím pádem i vyšší návštěvnosti. Kvalitní systém by proto měl minimálně generovat validní kód, implementovat čistá URL, tzv. „Cleans URL“, používat popisky a klíčová slova v záhlaví stránky. Dále by mělo být užito (x)HTML tagů podle norem organizace W3C (3), pro SEO důležité především nadpisy (H*x*), titulky (title) a popisy (alt). SEO problematika je nicméně velice rozsáhlá a CMS samotný může pokrýt pouze část, zbytek je závislý na lidské obsluze (4).

3. Analýza a návrh

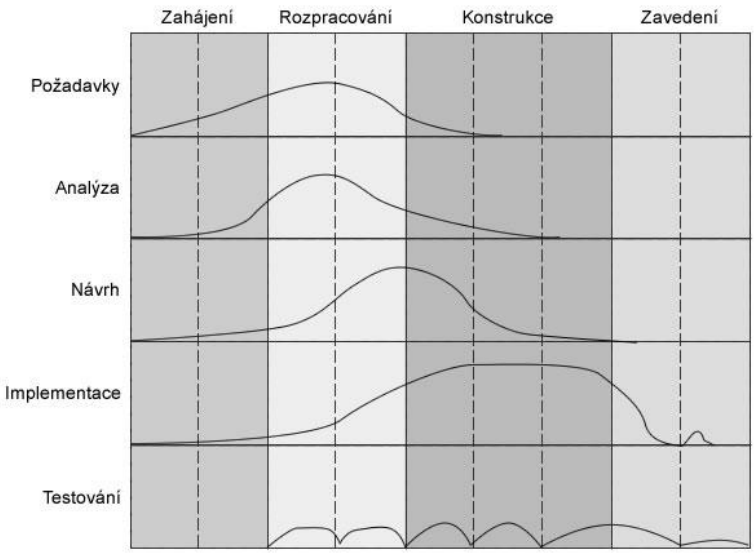
Modelovací jazyk UML (Unified Modeling Language) byl vytvořen již v průběhu 90. let za účelem vyjádření objektově orientované analýzy a návrhu (5, s. 11).

Hlavním cílem analýzy je vytvořit analytický model. Analýza určuje co má systém dělat, ale ne jakým způsobem toho dosáhnout. Rozdíl mezi analýzou a návrhem není jasně definován, vždy závisí na konkrétním případu. Výsledkem analýzy je realizace jednotlivých případů užití z USE CASE diagramu, který ukazuje jejich vzájemnou komunikaci a chování. Analytický model by měl být stručný a jednoduchý, měl by se věnovat zachycení případů užití. Analýza jako taková se zaměřuje na připravení modelu a určení jeho funkcí, kdežto zaměřením návrhu je implementace určených funkcí. Návrh představuje technické řešení navrženého modelu, které je později využito ve fázi implementace (6, s. 137, 330).

> *„Modelování analýzy je strategickou aktivitou, neboť se během ní snažíme o modelování základního chování systému.“ (6, s. 137)*

3.1 Požadavky

Na začátku vývoje celého projektu je nejprve nutné definovat požadavky. Určení požadavků probíhá v prvotních fázích. Je nezbytné mít alespoň základní představu o budoucí podobě systému a především o tom, co bude systém dělat, protože většina požadavků je určena právě na začátků vývoje (6, s. 74).



Obrázek 1 – unifikovaný proces vývoje softwaru, převzato a upraveno (6, s. 74).

Konkrétní základní požadavky na výsledný systém pro správu obsahu jsou určeny takto:

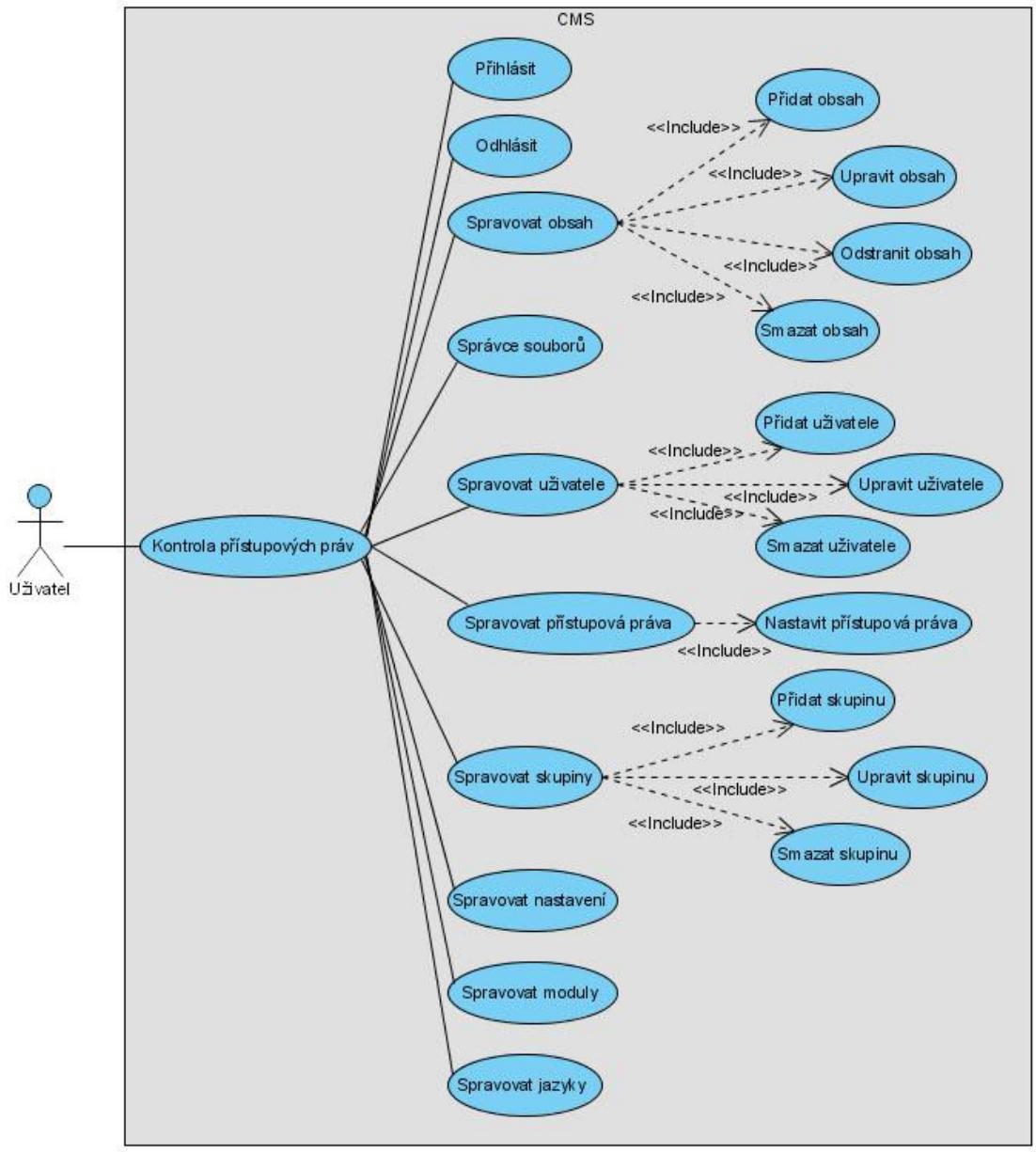
- Možnost přihlášení a odhlášení uživatele do systému.
- Správa obsahu:
  - Přidání nového obsahu.
  - Editace stávajícího obsahu.
  - Odstranění existujícího obsahu.
  - Obnova nebo úplné smazání odstraněného obsahu (obdoba funkce koše ve Windows).
- Knihovna – prohlížeč a správce souborů.
- Správa uživatelů:
  - Přidání nového uživatele.
  - Editace stávajících uživatelů.
  - Smazání existujících uživatelů.
- Správa přístupových práv a jejich nastavení.
- Správa skupin (uživatelských rolí):
  - Přidání nové skupiny.
  - Editace stávajících skupin.
  - Smazání existujících skupin.
- Správa nastavení – pro každý jazyk.
- Správa modulů a jejich editace.
- Správa jazyků a jejich editace.

3.2 USE CASE

Základním a nejjednodušším vyjádřením požadavků je USE CASE diagram (digram případu užití), který umožňuje jednoduše a přehledně vyjádřit funkční požadavky, tedy pouze popsat požadované funkce a služby systému.

Hlavním cílem digramů případů užití je nalézt a určit hranice systému, stanovit aktéry (actors), kteří budou k systému přistupovat, vytvořit jednotlivé případy užití, což jsou možnosti, které mohou aktéři v systému provádět a nakonec vytvořit relace mezi jednotlivými aktéry a případy užití (6, s. 91).

USE CASE diagram vlastně představuje základní model systému, konkrétní požadavky lze tedy graficky vyjádřit takto:



Obrázek 2 – USE CASE model systému.

Díky kontrole přístupových práv je možné dosáhnout velké univerzálnosti. Do systému se může přihlásit libovolný uživatel, ale přístup do administrace či jednotlivých sekcí je povolen pouze s patřičnými přístupovými právy. Model je v podstatě rozdělen do dvou úrovní, kdy první představují jednotlivé hlavní sekce (případy užití) a druhou jejich případné podsekce. Vztah mezi hlavními sekcemi a jejich podsekcemi je znázorněn relací typu `<<include>>`, která značí, že jsou jejich nedílnou součástí (6, s. 121-122).

3.3 Realizace případů užití

Realizace případů užití ukazuje chování a vztah mezi jednotlivými případy užití. Není nutné realizovat všechny případy užití, ale jen ty důležitější. Jedná se o přesnou specifikaci a model chování.

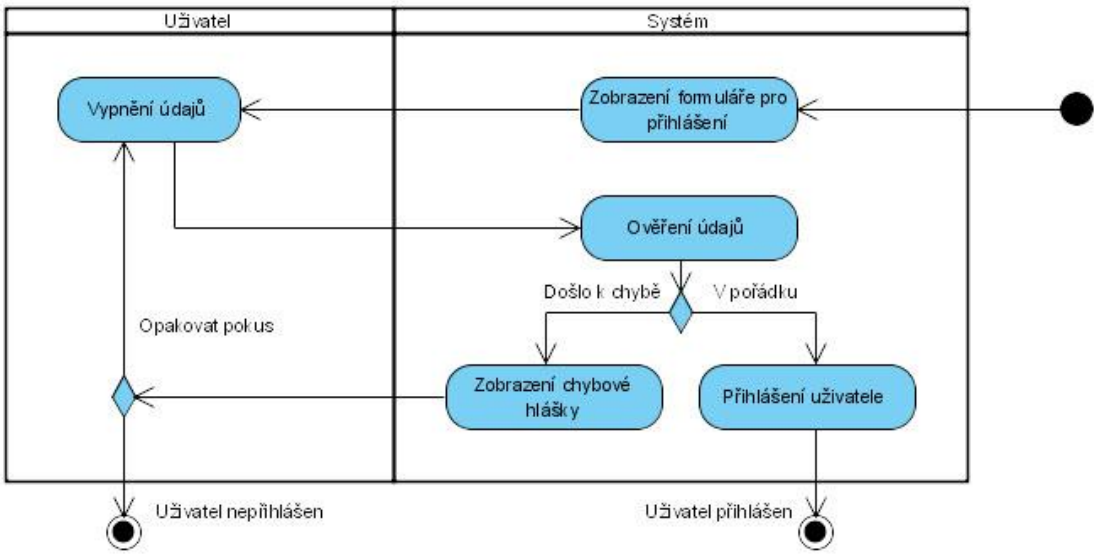
Realizaci případů užití je možné jednoduše vyjádřit pomocí diagramů interakce, například pomocí sekvenčních diagramů, ovšem nejjednodušší a nejpřehlednější vyjádření poskytují diagramy aktivit. (6, s. 246)

3.4 Diagramy Aktivit

Diagramy aktivit jsou objektově orientované vývojové diagramy, které modelování vyjadřují pomocí jednotlivých aktivit. Každá aktivita se skládá z uzlů, jež jsou spojeny hranami. Jedná se o zvláštní případ stavového automatu. Jejich napojení je vhodné na libovolný modelovací prvek, ale nejčastěji jsou používány v kombinaci s případy užití, resp. pro jejich upřesnění. (6, s. 286)

3.4.1 Přihlášení uživatele

Aby bylo možné se systémem pracovat, tak musí dojít nejprve k přihlášení uživatele. Postup přihlášení uživatele probíhá jednoduše. Systém zobrazí formulář pro přihlášení, následně uživatel vyplní své přihlašovací údaje. V případě zadání správných údajů, dojde k jejich úspěšnému ověření a uživatel je přihlášen, jinak systém zobrazí chybovou hlášku s popisem chyby a uživatel má možnost proces buď zopakovat, anebo zrušit, čímž přihlášen nebude.

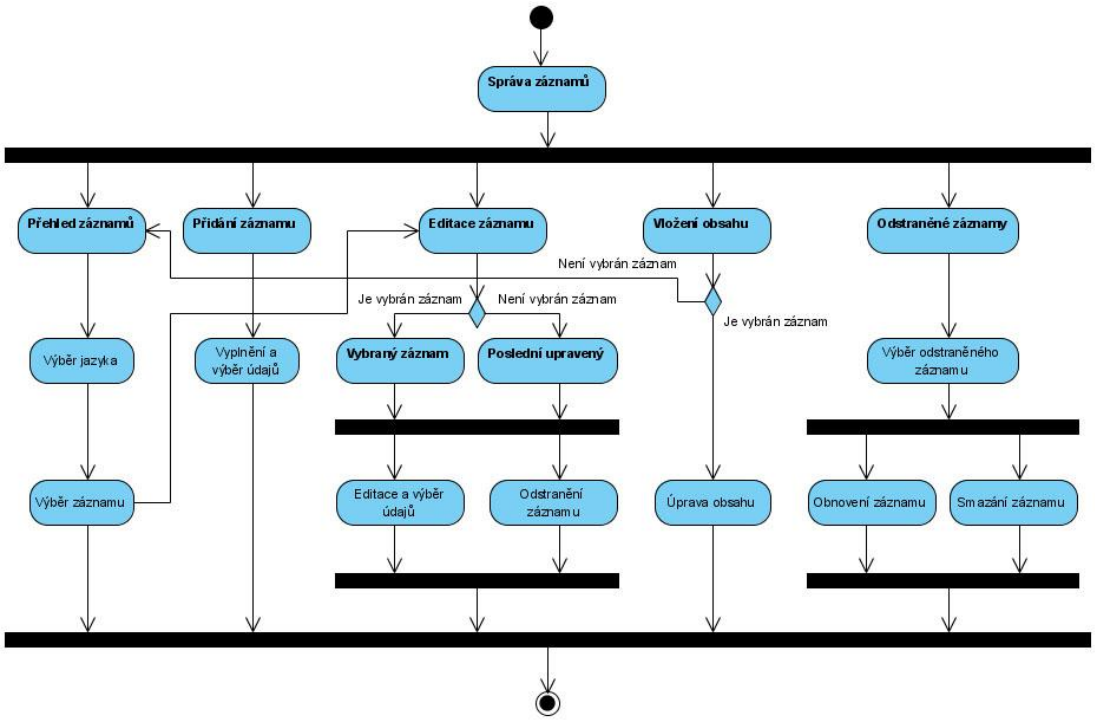


Obrázek 3 – přihlášení uživatele.

3.4.2 Správa obsahu

Základní jednotkou systému pro správu obsahu je záznam (item), který slouží především ke kategorizaci. K jednotlivým záznamům může být pomocí modulů přiřazen libovolný obsah. Při vytvoření záznamu je vybrán typ modulu, který je k danému záznamu po vytvoření pevně připojen a později jej není možné změnit. Díky konceptu záznam-obsah (modul) je dosaženo maximální univerzálnosti. Celá sekce správy obsahu je rozdělena do několika částí:

- Přehled záznamů – přehled všech záznamů pro aktuálně vybraný jazyk. Záznamy jsou zobrazeny vždy pouze pro vybraný nebo výchozí jazyk.
- Přidání záznamu – přidání nového záznamu. Po vyplnění a výběru patřičných údajů dojde k přidání nového záznamu.
- Editace záznamů – úpravy nebo odstranění již existujících záznamů. Pokud není vybraný žádný záznam k úpravě, použije se naposledy upravený.
- Vložení obsahu – pokud je vybraný záznam, tak je mu možné přidat nový obsah nebo upravit stávající.
- Odstraněné záznamy – přehled odstraněných záznamů, obdoba funkce koše ve Windows. Odstraněné záznamy je možné buď obnovit, nebo úplně smazat. Odstraněné záznamy v systému nejsou jinak zobrazeny.

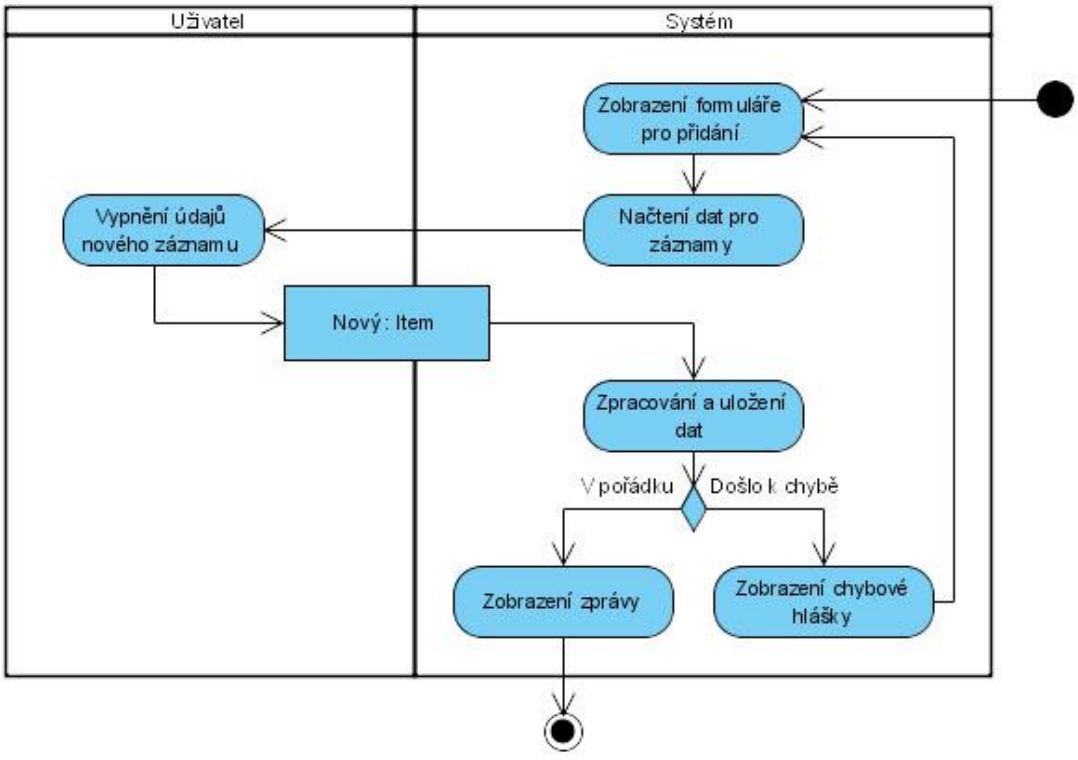


Obrázek 4 – schéma správy obsahu.

K upřesnění případů užití slouží také doplňující charakteristiky. Vybrané doplňující charakteristiky případů užití pro správu obsahu společně s diagramy aktivit:

A) Přidání záznamu

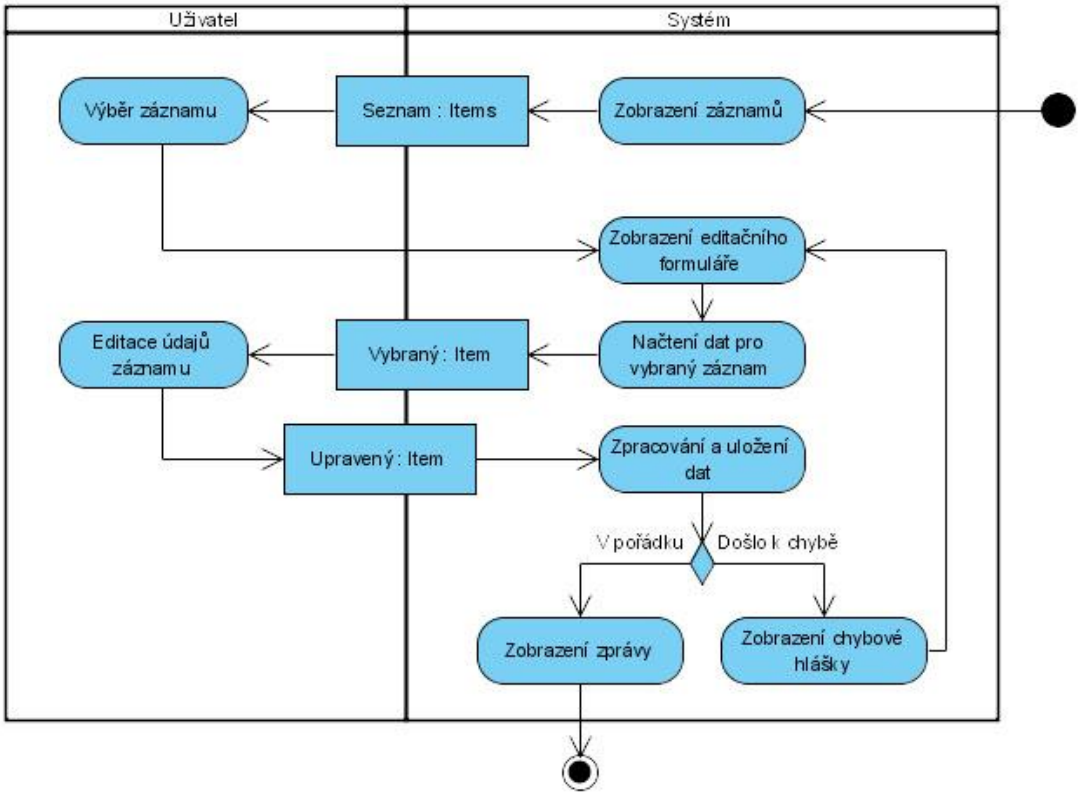
- **Stručný popis:** Postup přidání nového záznamu do systému.
- **Hlavní aktéři:** Uživatel
- **Vedlejší aktéři:** Systém
- **Vstupní podmínky:** Uživatel je přihlášen a má práva pro přístup do dané sekce.
- **Hlavní scénář:**
  1. Systém zobrazí formulář pro přidání a načte úvodní data.
  2. Uživatel vyplní údaje nového záznamu a odešle formulář.
  3. Systém zpracuje a uloží data. O výsledku informuje uživatele.
- **Výstupní podmínky:** Záznam je přidán do databáze. Uživatel je informován.
- **Alternativní scénáře:** Žádné



Obrázek 5 – scénář pro přidání nového záznamu.

B) Editace záznamu

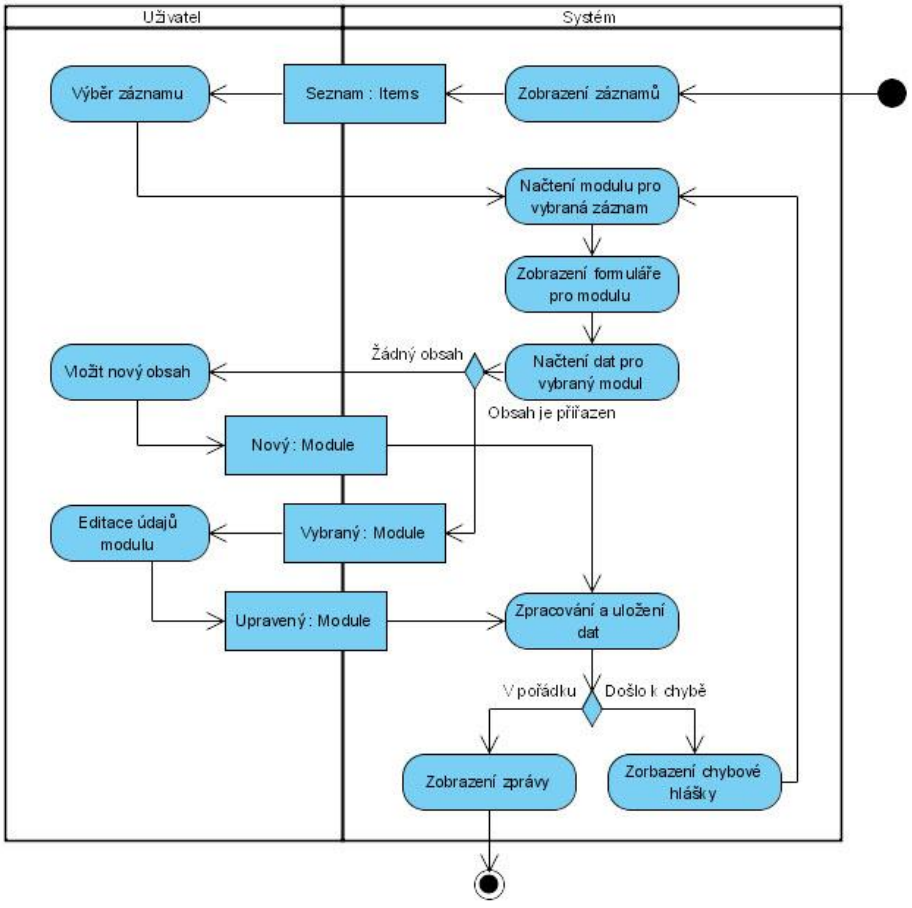
- **Stručný popis:** Postup editace existujícího záznamu v systému.
- **Hlavní aktéři:** Uživatel
- **Vedlejší aktéři:** Systém
- **Vstupní podmínky:** Uživatel je přihlášen a má práva pro přístup do dané sekce. V systému existuje alespoň jeden záznam.
- **Hlavní scénář:**
  1. Systém zobrazí přehled všech záznamů pro vybraný jazyk.
  2. Uživatel vybere libovolný záznam k úpravě.
  3. Systém zobrazí formulář pro editaci a načte data vybraného záznamu.
  4. Uživatel upraví údaje vybraného záznamu a odešle formulář.
  5. Systém zpracuje a uloží data. O výsledku informuje uživatele.
- **Výstupní podmínky:** Údaje záznamu jsou v databázi upraveny. Uživatel je informován.
- **Alternativní scénáře:** Žádné



Obrázek 6 – scénář pro editaci existujícího záznamu.

C) Vložení obsahu

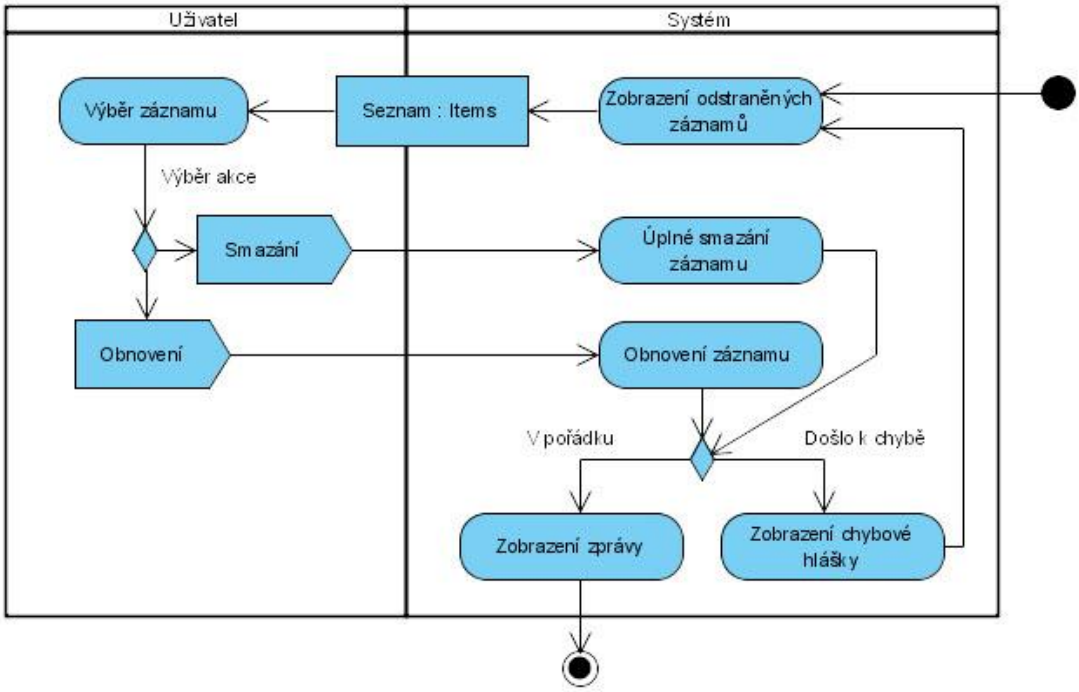
- **Stručný popis:** Postup vložení obsahu k vybranému záznamu.
- **Hlavní aktéři:** Uživatel
- **Vedlejší aktéři:** Systém
- **Vstupní podmínky:** Uživatel je přihlášen a má práva pro přístup do dané sekce. V systému existuje alespoň jeden záznam.
- **Hlavní scénář:**
  1. Systém zobrazí přehled všech záznamů pro vybraný jazyk.
  2. Uživatel označí libovolný záznam a vybere operaci vložení obsahu.
  3. Systém pro vybraný záznam zobrazí formulář modulu a jeho data.
  4. Uživatel upraví nebo vloží nové údaje a odešle formulář.
  5. Systém zpracuje a uloží data. O výsledku informuje uživatele.
- **Výstupní podmínky:** Do databáze je přidán nový záznam modulu nebo upraven již existující. Uživatel je informován.
- **Alternativní scénáře:** Žádné



Obrázek 7 – scénář pro vložení obsahu k záznamu.

D) Smazaní nebo obnova záznamu

- **Stručný popis:** Postup smazání nebo obnovy odstraněných záznamů.
- **Hlavní aktéři:** Uživatel
- **Vedlejší aktéři:** Systém
- **Vstupní podmínky:** Uživatel je přihlášen a má práva pro přístup do dané sekce. V systému existuje alespoň jeden odstraněný záznam.
- **Hlavní scénář:**
  1. Systém zobrazí přehled odstraněných záznamů pro vybraný jazyk.
  2. Uživatel vybere libovolný záznam a zvolí operaci:
     - buď úplné smazání,
     - nebo obnovu záznamu.
  3. Systém zpracuje a provede operaci. O výsledku informuje uživatele.
- **Výstupní podmínky:** Údaje záznamu jsou v databázi upraveny nebo je v databázi smazán. Uživatel je informován.
- **Alternativní scénáře:** Žádné

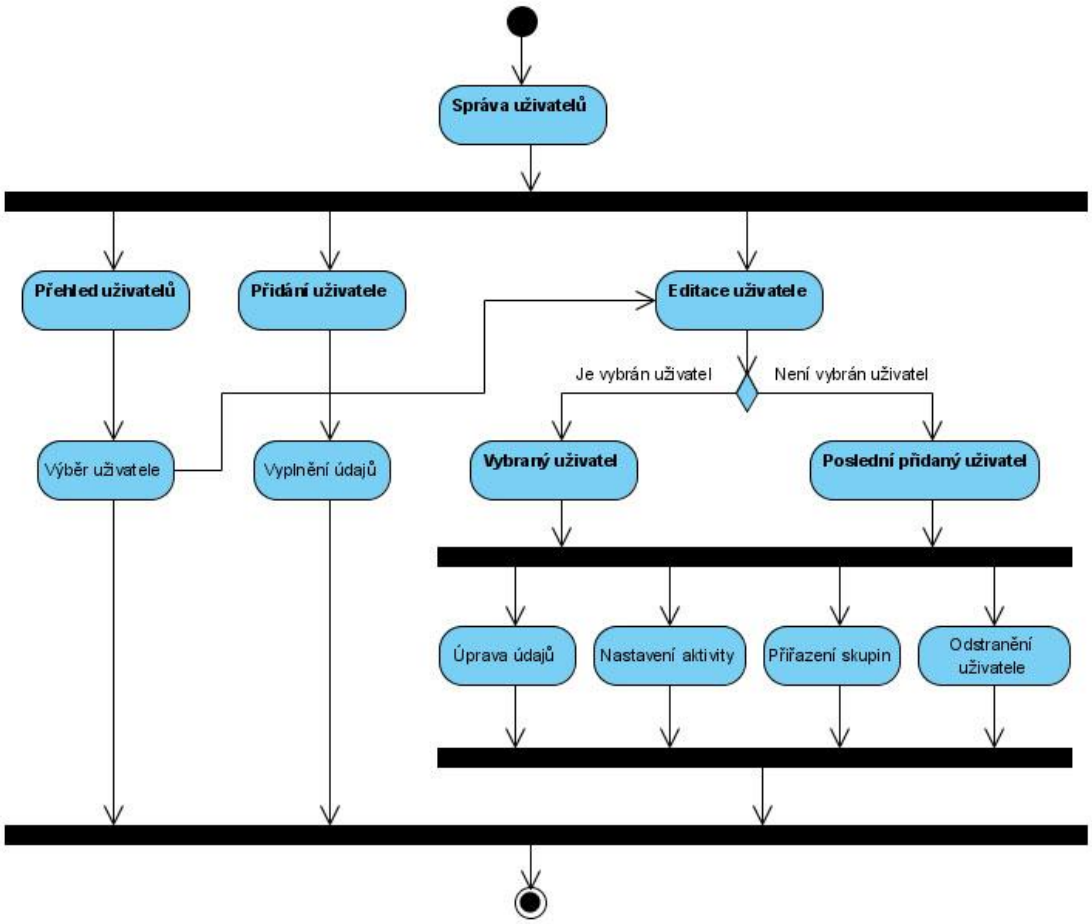


Obrázek 8 – scénář smazání nebo obnovy záznamu.

3.4.3 Správa uživatelů

Sekce správa uživatelů je určena jak pro manipulaci s již registrovanými uživateli, tak i pro ruční (administrátorské) vytvoření uživatelů nových.

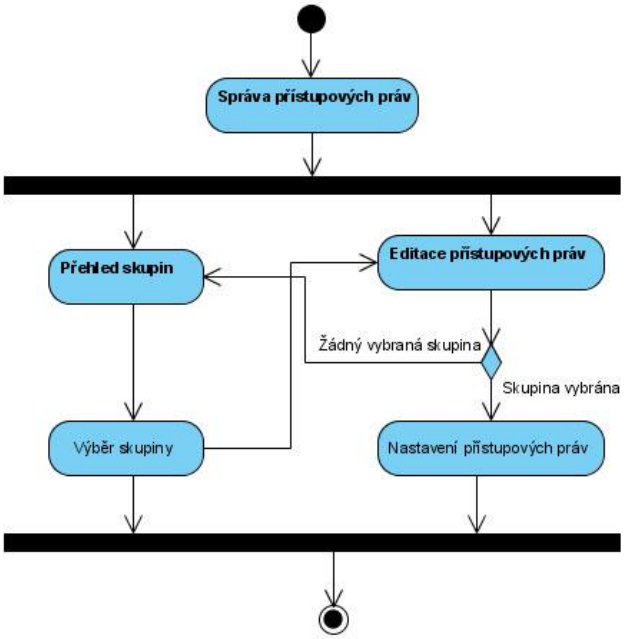
- Přehled uživatelů – přehled všech uživatelů, slouží k výběru konkrétního uživatele pro editaci.
- Přidání uživatele – přidání nového uživatele podle vyplněných údajů.
- Editace uživatele – úprava nebo smazání existujícího uživatele. Pokud není žádný uživatel vybrán, použije se naposledy přidaný.



Obrázek 9 – schéma správy uživatelů.

3.4.4 Správa přístupových práv

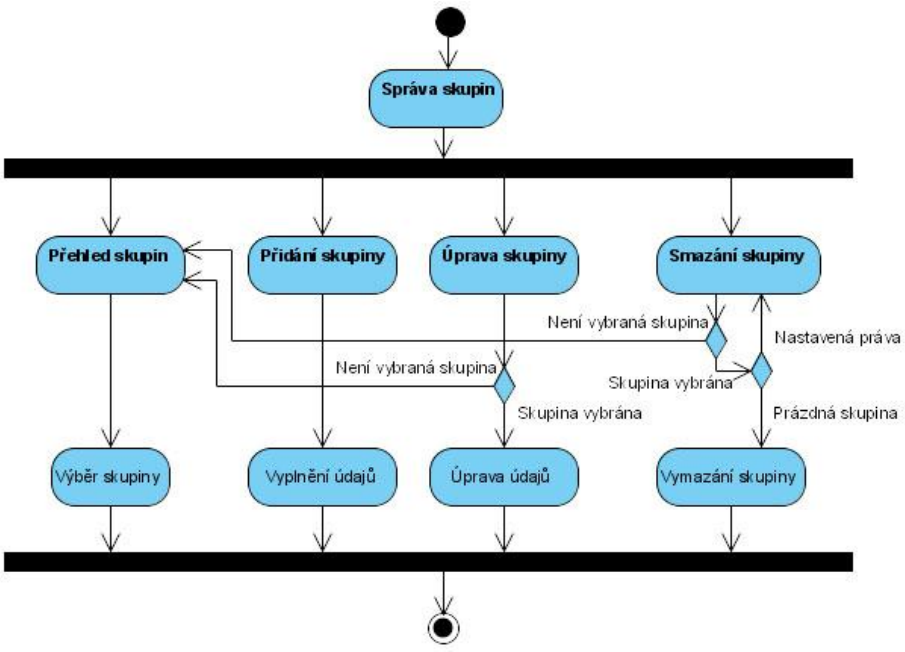
Pomocí nastavení přístupových práv jednotlivým uživatelským skupinám je možné povolit či zakázat jejich přístup do dané sekce, popř. k dané funkci. Práva mohou být nastavena libovolně jednotlivým skupinám, přičemž počet skupin není nijak omezen.



Obrázek 10 – schéma správy přístupových práv.

3.4.5 Správa skupin

Užití skupin je vhodné k libovolné kategorizaci jednotlivých uživatelů. Systém by měl obsahovat alespoň jednu skupinu, např. skupinu určenou pro administrátory. Počet skupin by měl být ovšem větší, přiměřený počtu uživatelů. Skupiny je možné buď přidávat, upravovat nebo mazat. Ke smazání skupiny může dojít pouze tehdy, když je nějaká vybrána a zároveň pouze pokud je prázdná, resp. především pokud v ní není zařazen žádný uživatel.

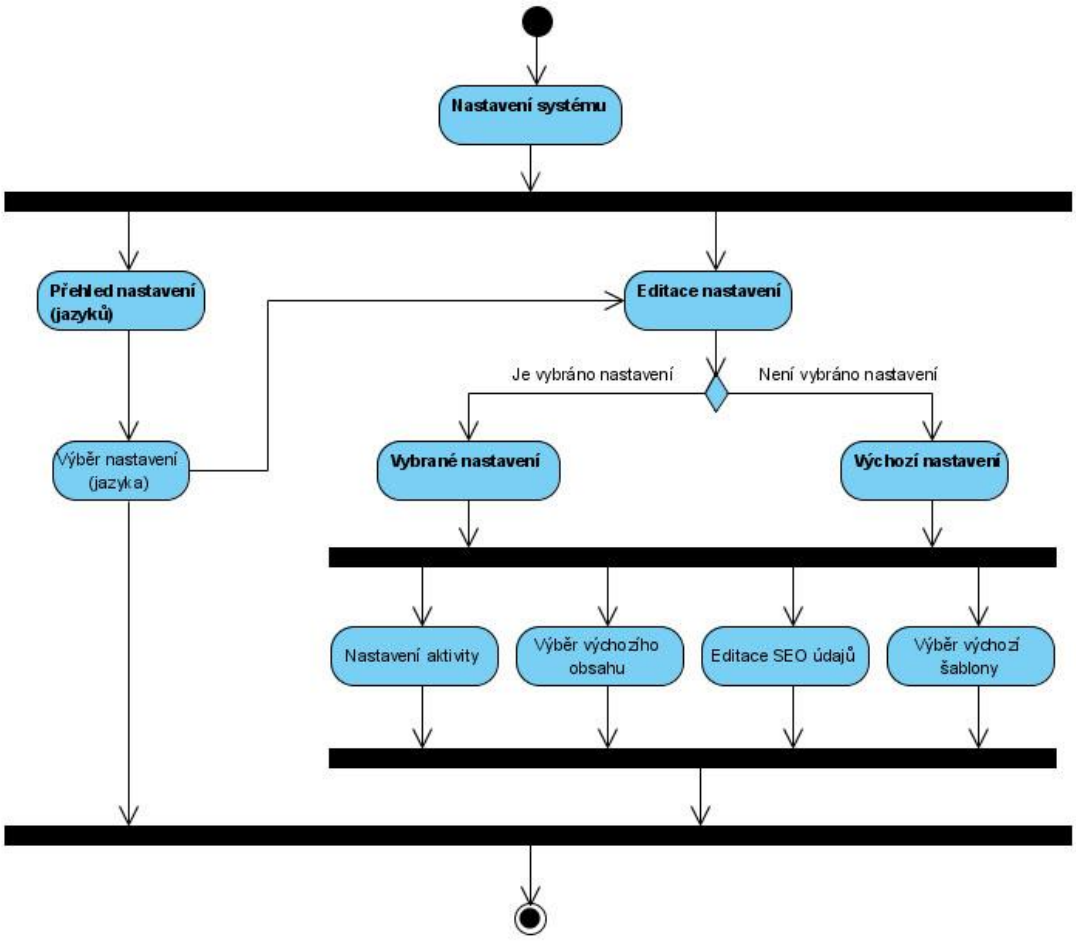


Obrázek 11 – schéma správy skupin.

3.4.6 Správa nastavení

Ke každému jazyku v systému by mělo být právě jedno nastavení, resp. záznam nastavení. Jednotlivé záznamy nastavení obsahují pro daný jazyk základní údaje, jako např. výchozí šablonu, výchozí obsahový záznam (item) a SEO záležitosti. Právě jedno nastavení by mělo být výchozí, které je použito v případě, že není vybrán žádný konkrétní jazyk.

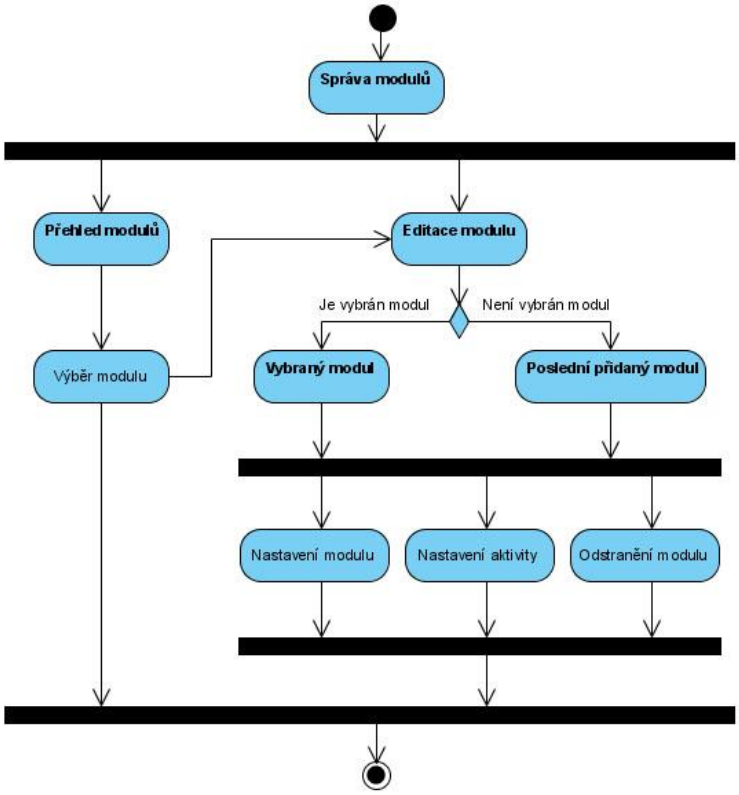
- Přehled záznamů – přehled záznamů nastavení pro všechny jazyky v systému, slouží k výběru konkrétního nastavení k editaci.
- Editace nastavení – úpravy hodnot pro vybraný záznam nastavení, pokud není žádné nastavení vybráno, použije se výchozí.



Obrázek 12 – schéma správy nastavení.

3.4.7 Správa modulů

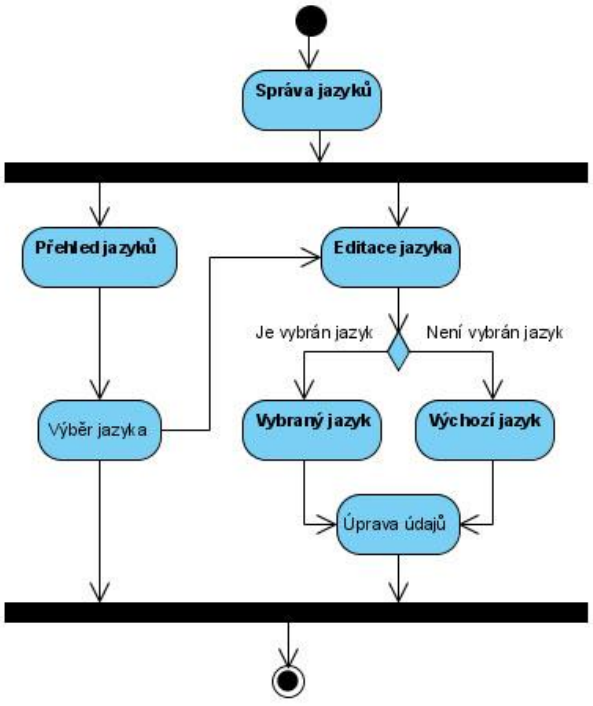
Administrace a nastavení jednotlivých nahraných modulů slouží vždy ke změně nastavení a aktivitě, nebo k jejich odstranění ze systému. Pokud není žádný modul vybrán, použije se naposledy přidaný.



Obrázek 13 – schéma správy modulů.

3.4.8 Správa jazyků

Administrace a nastavení jednotlivých nahraných jazyků. Je možné měnit pouze jejich údaje. Pokud při editaci není vybrán žádný jazyk, použije se výchozí.

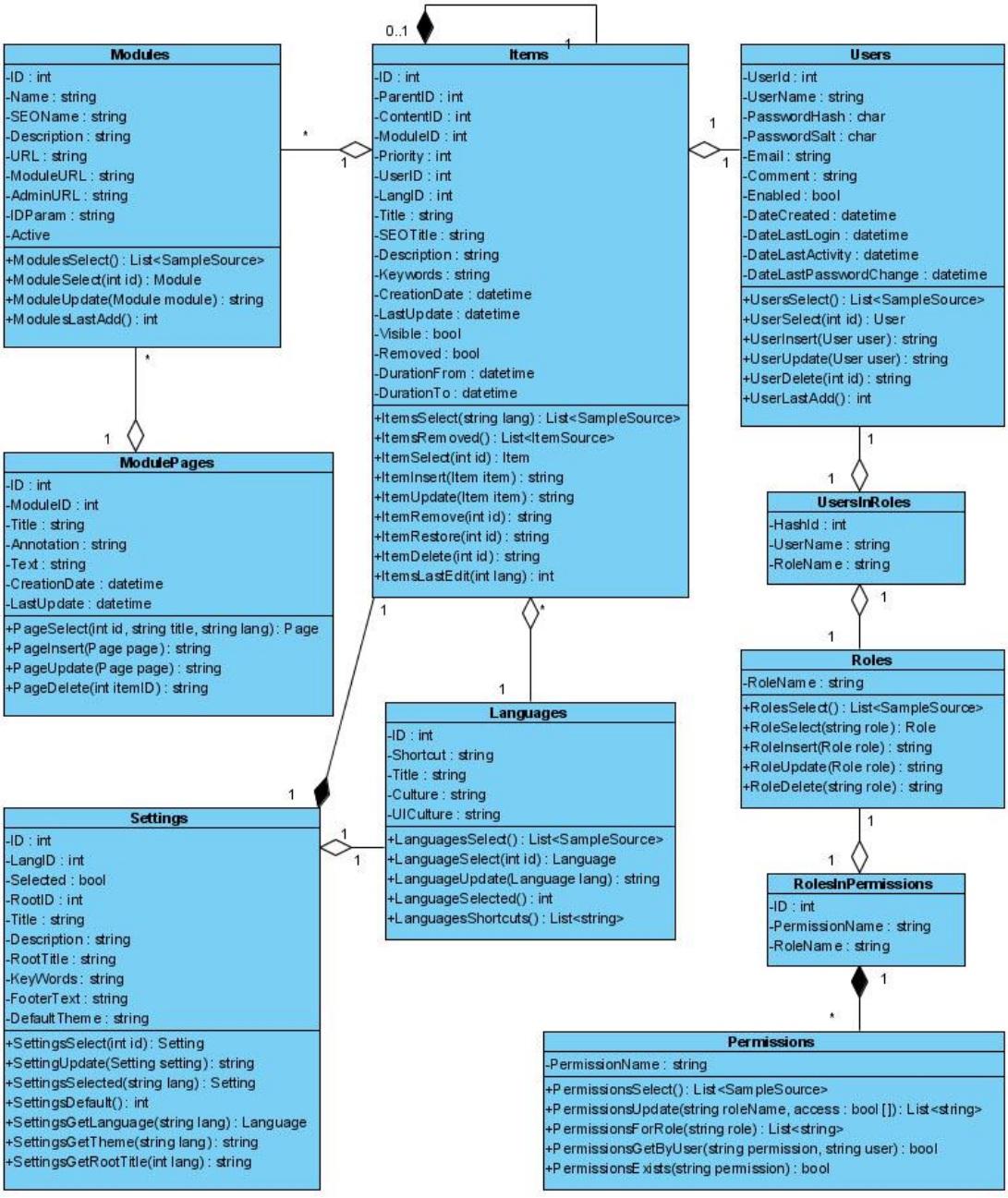


Obrázek 14 – schéma správy jazyků.

3.5 Návrhové třídy

Návrhové třídy představují takové třídy, jež jsou specifikovány na úrovni, po které je možná jejich implementace. Jedná se o další, přesnější vyjádření případů užití. Model návrhových tříd může být přímo předán programátorovi a v závislosti na míře jeho zpracování je možné přejít k přímé implementaci (6, s. 341).

Model návrhových tříd může být vyjádřen pomocí diagramu tříd (Class diagram):

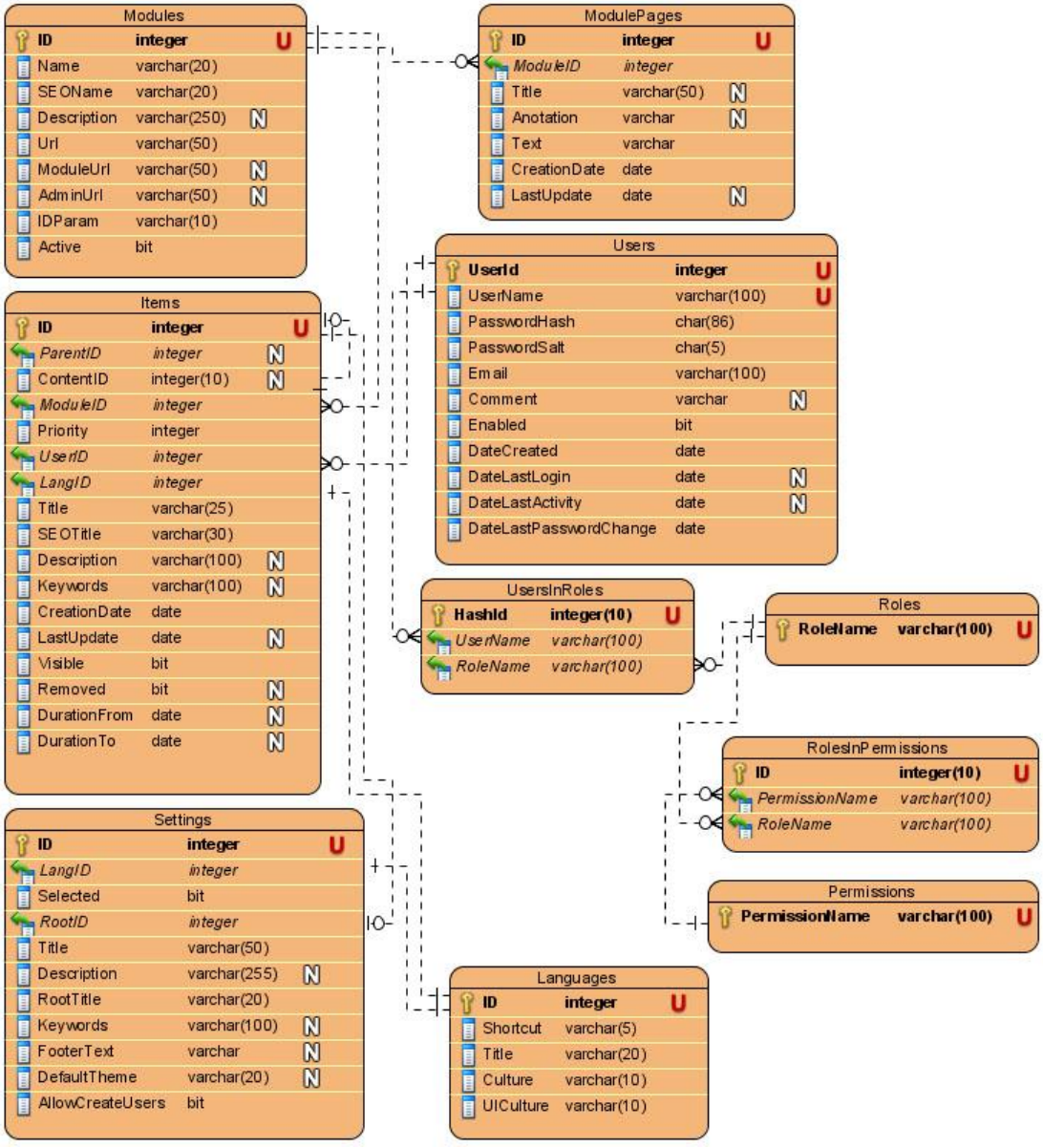


Obrázek 15 – class diagram knihovny.

3.6 Model databáze

K vyjádření modelu databáze slouží E-R diagram (Entity-Relation), který ilustruje vztah mezi jednotlivými entitami v databázi. Jedná se vlastně o grafickou dokumentaci databázových tabulek. Tento diagram zobrazuje všechna pole v každé tabulce, navíc může popisovat typy dat u těchto polí a znázorňovat vztahy mezi nimi. Vztah určuje, jakým způsobem jsou data mezi entitami sdílena, přičemž existují tři typy vztahů: 1:1, 1:N a M:N. Tyto diagramy jsou užitečné pro návrháře i správce databáze a slouží k lepšímu pochopení struktury databáze (6, s. 185-189).

Databázi systému lze pomocí E-R diagramu vyjádřit takto:

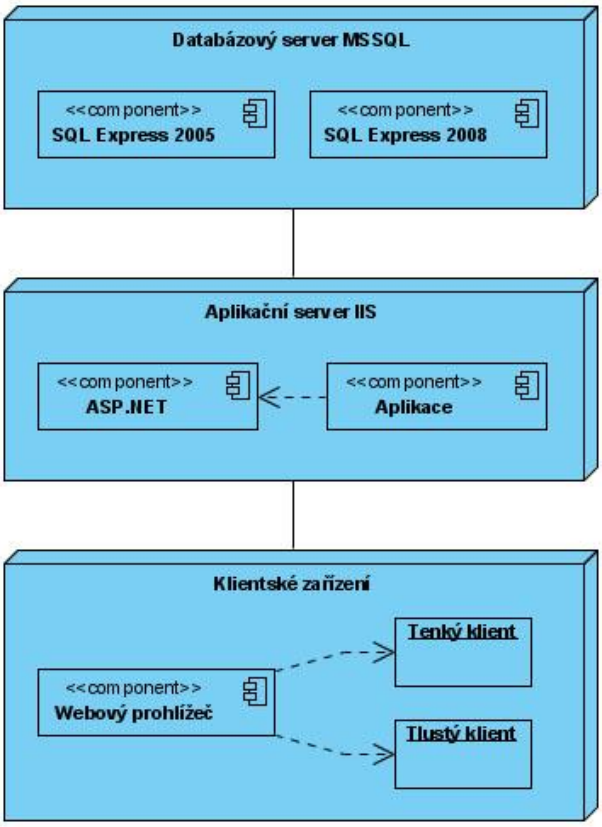


Obrázek 16 – Entity-Relation diagram databáze.

3.7 Model nasazení

Model nasazení systému se vyjadřuje pomocí diagramu nasazení (deployment diagram), který ukazuje hardware i software a vztah mezi nimi. Jedná se vlastně o vyjádření nasazení softwarové architektury na architekturu fyzickou (6, s. 469).

Nasazení systému může být vyjádřeno následovně:



Obrázek 17 – diagram nasazení systému.

4. Použité technologie a nástroje

Přehled a popis jednotlivých použitých technologií, metod a nástrojů při implementaci navrženého modelu.

4.1 ASP.NET

ASP.NET je skriptovací jazyk běžící na straně serveru, je součástí .NET Frameworku a slouží k vývoji webových stránek či aplikací. Vychází z původního jazyka ASP (Active Server Pages) jehož je nástupcem (7, s. 38).

> *„Webové aplikace ASP.NET jsou plnohodnotné aplikace .NET, které vykonávají komplikovaný kód a jež udržují runtime .NET. ASP.NET také používá všechny schopnosti .NET Frameworku – což je vyčerpávající soubor nástrojů tříd – právě tak snadno jako obyčejná aplikace Windows. ASP.NET v podstatě odstraňuje dělicí čáru mezi vývojem aplikací a webovým vývojem, protože rozšiřuje nástroje a technologie, které výhradně používali vývojáři desktopových aplikací, do světa webového vývoje.“ (7, s. 36)*

Nejdůležitější vlastnosti a výhody ASP.NET:

- Přímá integrace s .NET Frameworkem – je možné používat stejné třídy pro jakýkoliv druh aplikace. Webový vývojáři mají stejné možnosti a nástroje, jako vývojáři klientských aplikací.
- ASP.NET se kompiluje – pro psaní kódu stránek se používají interpretované skriptovací jazyky, nejprve tedy dochází k přeložení do strojového kódu nižší úrovně, což má za následek mírné zpomalení.
- Velký výběr programovacích jazyků – webové aplikace v ASP.NET je možné psát v libovolném .NET jazyku, nejčastěji se používá C# nebo VB.NET.
- Engine ASP.NET běží uvnitř společného runtime jazyků CLR (Common Language Runtime). Na .NET Framework se odkazuje jako na celek řízeného kódu. To přináší výhody, jako např. automatickou správu paměti (garbage collection), typovou bezpečnost při kompilaci, apod.
- Oproti klasickému ASP je ASP.NET objektově orientované.
- Podpora různých zařízení a prohlížečů.
- Snadná konfigurace a rozmístění – stačí zkopírovat pouze soubory webové aplikace do virtuální složky na severu (7, s. 39-47).

K vývoji systému je použita verze ASP.NET 3.5, jež vychází z verze 2.0, oproti které přidává pouze některé nové funkce, především se jedná o přidání podpory LINQ a AJAX. Jako programovací jazyk systému byl zvolen C#, jakožto v současnosti výchozí jazyk .NET Frameworku.

Celé této i příbuzným technologiím je věnován komunitní web (2), který zároveň slouží jako vstupní brána k technologiím rodiny ASP.NET.

4.2 LINQ to SQL

LINQ (z anglického Language Integrated Query) je dotazovací jazyk, určen pro rozšíření nejen jazyka C#. Umožňuje psát příkazy jazyka C#, za účelem získání dat z nějaké databáze, v případě LINQ to SQL se jedná o databázi MSSQL.

Obsahuje sadu dotazovacích operátorů podobnou jazyku SQL, pomocí které je možné dotaz vykonat nad libovolným typem dat a dotaz se vykoná až v případě přístupu k datům. Konstrukce v LINQu se řešení pomocí tzv. Lambda výrazů, které jsou jednodušší metodou zápisu anonymních metod. Klíčové slovo var umožňuje využit anonymní třídy (7, s. 49).

4.3 Ajax Control Toolkit

Ajax Control Toolkit je volně dostupná knihovna serverových ovládacích prvků založených na Microsoft AJAX.NET, dříve projekt Atlas, jež je součástí ASP.NET 3.5. Jedná o plnohodnotný balík moderních ovládacích prvků webových stránek šířený jako open-source. Celý projekt je výsledkem společného úsilí mezi Microsoft ASP.NET AJAX a komunitou. Ajax Control Toolkit obsahuje více než 30 ovládacích prvků, které pomáhají snadno vytvářet bohaté, interaktivní webové stránky (8).

4.4 CSS Friendly Control Adapters

ASP.NET samo o sobě generuje poněkud zastaralý kód, což je dáno především tím, že většina ovládacích prvků pochází z původní, několik let staré verze 2.0. CSS Friendly Control Adapters Toolkit, jak už samotný název napovídá, tento problém řeší. Jedná se o zdarma dostupnou sadu adaptérů, které upravují vykreslování většiny původních ovládacích prvků tak, aby generovaly CSS validní kód a zároveň z nich odstraňují tabulkové layouty (9).

4.5 Altairis Web Security Toolkit

ASP.NET obsahuje vestavěnou podporu pro práci s uživateli, uživatelskými rolemi a profily. Základní řešení Membership, Profile a Role providerů je zbytečně složité a nemotorné. Tento problém řeší Simple SQL Providers z balíku Altairis Web Security Toolkit od českého autora Michala Valáška. Celík balík je volně dostupný v rámci komunitního projektu CodePlex (10).

4.6 FCKeditor.Net

FCKeditor.Net umožňuje používat volně dostupný WYSIWYG FCKeditor v rámci ASP.NET a slouží k jeho přímé integraci. Díky tomu může být na stránkách použit ovládací prvek pro editaci nejen textů, jehož způsob ovládání i možnosti jsou podobné klasickým textovým editorům. FCKeditor navíc obsahuje správce souborů, který pomáhá snadno nahrávat a přidávat obrázky i další soubory (11).

4.7 Visual Studio

Visual Studio je profesionální vývojářský nástroj pro platformu .NET od firmy Microsoft. Obsahuje sadu designérských nástrojů a především funkci IntelliSence, která již při psaní kódu zachytává možné chyby a nabízí jejich opravu. Visual studio umožňuje psát nejen webové stránky, ale i aplikace v ASP.NET velice pohodlně a efektivně (7, s. 55). Základní edice Express je zdarma, ale projekty MSDN AA nebo Dreamspark přináší studentské licence vyšších verzí pro nekomerční použití.

4.8 Visual Paradigm for UML

Visual Paradigm for UML je návrhový nástroj, který podporuje všechny UML diagramy a schémata. Poskytuje rozsáhlé možnosti využití v libovolném případě modelování funkcí jazyka UML, včetně plně funkčních USE CASE a dalších diagramů. Výsledné diagramy mohou být exportovány do všech možných standardně používaných formátů. Pro nekomerční účely je k dispozici zdarma verze Community Edition (12).

6. Implementace

Hlavní část implementace tvoří systémová knihovna a samotná webová aplikace. Při implementaci byly nalezeny a vyřešeny různé problémy.

6.1 Knihovna

Základní výhodou prostředí .NET je možnost využívat DLL knihovny. Celá datová část systému je založena na dynamické knihovně (Class Library Project) `WebSystemLibrary.dll`, která má na starosti obsluhu veškerých dat. Nejlepší vlastností proto je možnost použít takovouto knihovnu v libovolném prostředí .NET Frameworku. Systém jako takový je sice řešen v prostředí ASP.NET, ale stejně tak by mohlo být možné vytvořit mutaci i pro Windows Forms, Silverlight nebo WPF.

6.1.1 Repositář

Podle vzoru ASP.NET MVC (13) a návrhového vzoru Repository je hlavní součástí knihovny třída `Repository.cs`, jejíž hlavní funkcí je právě zpracovávání datových dotazů. Bylo by možné jednotlivé části systému oddělit do vlastních repositářů (tříd), ale pro větší jednoduchost je použita třída pouze jedna. Jednotlivé části ve třídě jsou pro větší přehlednost odděleny do regionů:

```
#region
//část kódu
#endregion
```

Repositář obsahuje část pro inicializaci, různé systémové funkce, zdroje dat ovládacích prvků (datasources) a jednotlivé části administrace i uživatelské části:

- Obsah (Items)
- Obsahové moduly:
  - Modul stránek (ModulePages)
- Uživatelé (Users)
- Oprávnění (Permissions)
- Uživatelské skupiny (Roles)
- Nastavení (Settings)
- Moduly (Modules)
- Jazyky (Languages)

Přístup k datům z databáze zajišťuje technologie LINQ to SQL, kde u malých projektů by přímý přístup k DataContextu nebyl až takový problém, ovšem s postupným rozšiřováním aplikace by přímý přístup představoval čím dál tím větší problém, nehledě na údržbu a testování. Celý tento problém řeší právě systémový repositář (13).

6.1.2 Singleton

Pro přístup k repositáři je použit obecně platný návrhový vzor typu Singleton (jedináček).

Singleton patří do skupiny tvořivých objektových vzorů. Jeho hlavním smyslem je tvorba objektů, přičemž k následné nebo případné změně chování objektu dochází výměnnou instance.

Vhodně použitý singleton se nachází všude, kde se požaduje pouze jedna instance třídy v rámci projektu. Instance třídy musí být snadno dostupná pomocí známého, resp. veřejného přístupového bodu. Singleton objekt zpřístupňuje pomocí veřejné statické proměnné a navíc přidává jednoduše případné rozšíření pomocí dědičnosti. Dále se skládá pouze z jediné třídy, která sdílenou instanci zapouzdřuje a vytváření její instance (14). Řešení singletonu v systému:

```
public class Singleton
{
    private static Repository _Repository;
    private static string _ConnectionStringm;

    public static Repository Repository
    {
        get
        {
            if (_Repository == null)
            {
                if (_ConnectionString != null) _Repository =
                    new Repository(_ConnectionString);
                else
                    return null;
            }
            return _Repository;
        }
    }

    public static string ConnectionString
    {
        get { return Singleton._ConnectionString; }
        set { Singleton._ConnectionString = value; }
    }
}
```

6.1.3 SampleSource

Třída `SampleSource.cs` je výchozí systémový datový typ zdrojů dat pro ovládací prvky určené k zobrazení libovolných záznamů a jejich výběru. V systému je využívána především pro ovládací prvek TreeView.

Třída má dva konstruktory, které používají vždy dva parametry z pouze třech možných atributů třídy:

- id – celočíselná identifikace záznamu,
- title – titulek nebo text zobrazovaný pro daný záznam,
- value – textová identifikace nebo hodnota záznamu, zpravidla URL.

Řešení třídy `SampleSource.cs`:

```
public class SampleSource
{
    private int id;

    public int ID
    {
        get { return id; }
        set { id = value; }
    }

    private string title;

    public string Title
    {
        get { return title; }
        set { title = value; }
    }

    private string value;

    public string Value
    {
        get { return this.value; }
        set { this.value = value; }
    }

    public SampleSource(string title, string value)
    {
        this.title = title;
        this.value = value;
    }

    public SampleSource(string title, int id)
    {
        this.title = title;
        this.id = id;
    }
}
```

6.1.4 Features

Třída `Features.cs` obsahuje různé systémové funkce, především ty, které jsou v kódu použity vícekrát, a proto je výhodné jejich vyčlenění do jednoho místa. Konkrétně se jedná o metody:

- IsNumeric – veřejná statická metoda, jejíž funkcí je určení, zdali zadaný parametr v textové podobě je možné převést na číselnou hodnotu, tedy jestli se jedná o číslo. Tato funkce je v systému používána především pro kontrolu správnosti hodnoty parametru ID v URL adresách.

```
public static bool IsNumeric(string numberString)
{
    char[] ca = numberString.ToCharArray();
    for (int i = 0; i < ca.Length; i++)
    {
        if (!char.IsNumber(ca[i]))
            return false;
    }
    return true;
}
```

- ThemeExist – veřejná statická metoda, která ověřuje, jestli zadaný název šablony z parametru je obsažen v seznamu, resp. v poli názvů všech šablon z druhého parametru. Při inicializaci stránky v systému se tato funkce používá pro ověření existence šablony, resp. pro ověření existence názvu šablony.

```
public static bool ThemeExist(string[] themes, string theme)
{
    bool exist = false;
    if (themes.Length > 0)
    {
        foreach (string s in themes)
        {
            if (s.Equals(theme))
                exist = true;
        }
    }
    return exist;
}
```

- RemoveDiacritics – opět veřejná statická metoda určená pro odstranění diakritiky z textového parametru. Funkce této metody je využívána při vytváření SEO titulku pro URL adresu, jelikož zde diakritika nemá co pohledávat, tak ji automaticky odstraňuje a mezery nahrazuje pomlčkami. Autorem původního nápadu a řešení je Michal Valášek (15).

```
public static string RemoveDiacritics(string s)
{
    s = s.Normalize(NormalizationForm.FormD);
    StringBuilder sb = new StringBuilder();

    for (int i = 0; i < s.Length; i++)
    {
    if (CharUnicodeInfo.GetUnicodeCategory(s[i]) !=
        UnicodeCategory.NonSpacingMark) sb.Append(s[i]);
    }

    string source = Regex.Replace(sb.ToString(),
        @"[^\w\ ]", "");

    return source.Replace(" ", "-").ToLower();
}
```

6.1.5 ItemSource

Třída `ItemSource.cs` je datový typ pro zkrácené zobrazení záznamů (itemů), jelikož záznamy mají nadefinovány velké množství atributů, tak při jejich přehledu je na snaze zobrazit jen nejdůležitější informace. V systému je třída používána pro přehled odstraněných záznamů v ovládacím prvku GridView jako jeho zdroj (datasource).

Třída obsahuje pouze definice pro:

- ID – číselná identifikace záznamu,
- Title – titulek záznamu,
- Date – datum vytvoření nebo datum poslední úpravy záznamu, pokud není prázdný,
- UserName – uživatelské jméno uživatele, který záznam vytvořil,
- Priority – priorita záznamu, slouží jako výchozí položka pro seřazení.

6.1.6 SystemInfo

Třída `SystemInfo.cs` obsahuje základní informace o aktuálně používané verzi knihovny. Pomocí veřejných statických proměnných je možné zjistit:

- verzi knihovny,
- build vydání,
- datum vydání,
- URL adresu,
- autora.

6.2 ASP.NET praktiky

Vytváření webových aplikací a stránek v ASP.NET má své zažité praktiky a konvence, které mohou být použity dle vlastního uvážení. Při vývoji webové aplikace v ASP.NET stojíme na začátku před rozhodnutím jak nový projekt založit:

- Projekt (vývoj založený na projektu) – podobně jako u desktopových aplikací, projekt je kompilován do DLL assembly.
- Web Site (webový projekt) – jednotlivé soubory jsou umístěny v adresářové struktuře webu a každý soubor se považuje za součást aplikace.

Systém je založen jako webový projekt, především kvůli jednoduššímu rozmístění aplikace a správě souborů nebo pro jednodušší ladění (7, s. 58).

6.2.1 Adresáře

Každá webová aplikace by měla mít dobře navrženou adresářovou strukturu. ASP.NET má několik předdefinovaných adresářů určených k různým účelům. V systému jsou použity následující:

- App_Browsers – umožňuje nadefinovat pomocí speciálních XML souborů různé akce pro webové prohlížeče a umožňuje konfiguraci chování pro různé webové aplikace. V systému je použit v souvislosti s CSS Control Adapters.
- App_Code – obsahuje třídy a soubory, které se dynamicky kompilují. Zkompilovaný kód je v rámci APS.NET umístěn do dočasných adresářů, takže se v adresáři Bin neobjeví. Pro účely systému jsou zde umístěny třídy, které přímo závisí na ASP.NET a webové aplikaci.
- App_GlobalResources – zde jsou umístěny globální zdroje dostupné z celé aplikace. ASP.NET obsahuje velmi dobrou podporu pro multi-jazyčné webové aplikace i stránky. Globální zdroje fungují na principu klíč-hodnota (slovník), největší využití mají jako soubory pro jednotlivé překlady (jazyky).
- App_Themes – zde jsou umístěny jednotlivé webové šablony (themes) v samostatných složkách.
- Bin – obsahuje všechny zkompilované .NET assembly, především DLL knihovny, které webová aplikace využívá. V systému je zde umístěn odkaz na knihovnu a všechny další externí nástroje (7, s. 204-205).

Dále systém obsahuje vlastní adresáře pro lepší přehlednost a strukturu:

- Admin – obsahuje vše co je nutné k administraci systému. Pomocí vlastního souboru `Web.config` jsou nastavena základní bezpečností pravidla pro přístup pouze přihlášeným uživatelům a administrační webová šablona.
- Library – zde jsou umístěny adresáře pro jednotlivé systémové souborové knihovny včetně souborů, které obsahují. Jednotlivé soubory, jež nejsou součástí systému, ale obsahu, by zde měly být umístěny.
- Parts – seskupení jednotlivých uživatelských ovládacích prvků; v kořenové pozici obsahuje součásti pro uživatelské rozhraní, v administračním adresáři obsahuje jednotlivé součásti administrace.
- Scripts – zde jsou umístěny jakékoliv skripty používané v systému, například aktuálně používaný WYSIWYG editor.

6.2.2 Web User Controls

Uživatelské ovládací prvky jsou výborným nástrojem jak složitý, nepřehledný a roztáhlý kód rozdělit do samostatných bloků, které je možné ve výsledku poskládat do jednoho celku. Jejich použití je podobné jako u serverových ovládacích prvků, ale způsob jejich definice se podstatně lišný. Prázdný ovládací prvek vypadá podobně jako webový formulář, stejně se do něho vkládají libovolné ovládací prvky, kód může být oddělen a zanořování různých uživatelských ovládacích prvků vzájemně do sebe je také povoleno. Ovšem na rozdíl od webového formuláře nemůže uživatelský ovládací prvek být volán a používán samostatně, protože slouží jen jako stavební kámen (7, s. 699-700).

V systému jsou jednotlivé uživatelské ovládací prvky umístěny ve složkách Parts a to buď přímo v administračním anebo kořenovém adresáři aplikace, v závislosti na tom, pro jaký účel jsou určeny.

Pro větší přehlednost je použit řádný způsob pojmenovávání, název tedy začíná nejprve názvem sekce, kam patří a následně akcí, kterou vykonává. Například uživatelský ovládací prvek pro vyhledávání (vyhledávací panel) a v administraci pro editaci a smazání záznamů jsou pojmenovány (včetně cesty):

```
Parts/SearchPanel.ascx
Admin/Parts/ItemsEdit.ascx
Admin/Parts/ItemsDelete.ascx
```

6.2.3 Master Pages

Kromě samotných webových šablon (themes) pro vzhled stránky a uživatelských ovládacích prvků je pro další zjednodušení struktury stránky ASP.NET vybaveno dalším mocným nástrojem – vzory stránek (Master Pages).

Master Page slouží k vyčlenění definice webového formuláře, kterou následně může používat libovolný počet webových formulářů a v případě, že v ní dojde ke změně, tak se tato změna projeví na všech webových formulářích, které tento daný vzor používají. Jsou to vlastně šablony webových stránek, jež definují pevný obsah webové stránky a deklarují části pro vkládání vlastního obsahu webové stránky.

Hlavní rozdíl při deklaraci Master Page oproti webovému formuláři je v zahajovací deklaraci direktivy Master místo Page, a navíc například nemůže obsahovat definici pro šablonu:

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

<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Error.aspx.cs" Inherits="Error" Theme="Error" %>
```

V systému jsou definovány dvě základní šablony:

- `Default.master` – výchozí vzor stránek určený uživatelské části.
- `Admin.master` – vzor stránek pro administraci.

K definici vlastního obsahu webové stránky (formuláře) slouží ovládací prvek ContentPlaceHolder. Ten je určen a povelen pouze pro tuto funkci (7, s. 721).

6.2.4 Web.config

Konfigurační soubor webové aplikace založený na způsobu zápisu pomocí XML. Obsahuje definice nastavení pro bezpečnost, přihlašování, chybové hlášky a mnoho dalších. V kořenovém adresáři musí být právě jeden konfigurační soubor, který zároveň slouží jako výchozí pro celou aplikaci a v ostatních adresářích může být vždy další doplňující konfigurační soubor pro daný adresář (7, s. 70).

V systému je použit pouze výchozí konfigurační soubor a doplňující administrační, který upřesňuje pravidla pro administraci.

6.2.5 Global.asax

Globální aplikační soubor má na starosti hlavně definice globálních proměnných a reakce, resp. provedení globálních událostí. Pro potřeby systému je použita pouze globální událost `Application_Start`, která se aktivuje při startu aplikace, tedy při startu systému a nastavuje repositáři z knihovny pomocí singletonu připojovací řetězec. Dále je zde obsažena veřejná statická metoda `RegisterRoutes`, která slouží k definici routovacích pravidel pro čisté URL (7, s. 70).

```
void Application_Start(object sender, EventArgs e)
{
    Singleton.ConnectionString = ConfigurationManager
    .ConnectionStrings["DefaultConnectionString"]
    .ConnectionString;
    RegisterRoutes(RouteTable.Routes);
}

public static void RegisterRoutes(RouteCollection routes)
{
    routes.Add("pages", new Route("pages",
        new BaseRouteHandler(@"~/Pages.aspx")));
    routes.Add("pagesLang", new Route("{lang}/pages",
        new BaseRouteHandler(@"~/Pages.aspx")));
    routes.Add("page", new Route("page/{title}",
        new BaseRouteHandler(@"~/Pages.aspx")));
    routes.Add("pageLang", new Route("{lang}/page/{title}",
        new BaseRouteHandler(@"~/Pages.aspx")));
    routes.Add("pageLang2", new Route("page/{title}/{lang}",
        new BaseRouteHandler(@"~/Pages.aspx")));
    routes.Add("searchDefault", new Route("search",
        new BaseRouteHandler(@"~/Search.aspx")));
    routes.Add("search", new Route("search/{text}",
        new BaseRouteHandler(@"~/Search.aspx")));
    routes.Add("searchLang", new Route("{lang}/search/{text}",
        new BaseRouteHandler(@"~/Search.aspx")));
    routes.Add("errors", new Route("error",
        new BaseRouteHandler(@"~/Error.aspx")));
    routes.Add("error", new Route("error/{code}",
        new BaseRouteHandler(@"~/Error.aspx")));
}
```

6.2.6 Custom Error Pages

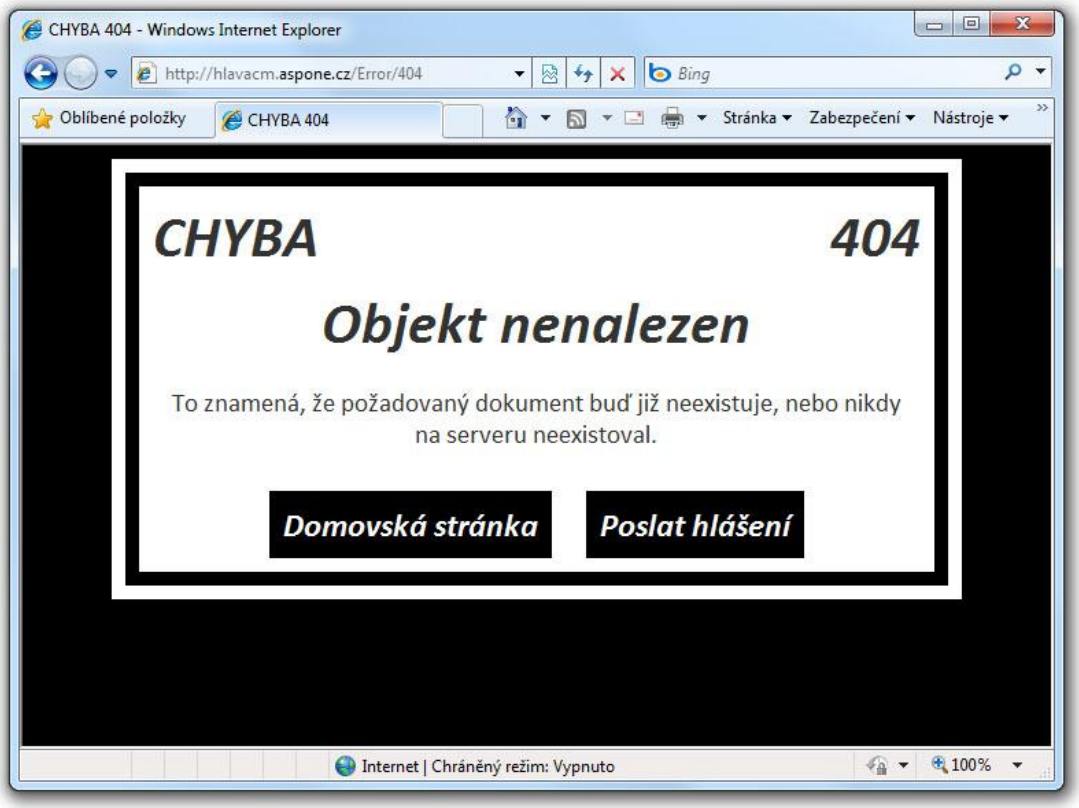
ASP.NET obsahuje vestavěné funkce pro používání vlastních chybových stránek. V systému je za tímto účelem vytvořen webový formulář `Errors.aspx`, který má nastavenou vlastní šablonu Errors a pomocí parametru `?code=xxx` nebo routovacího pravidla `Error/xxx`, kde `xxx` značí jeden z nadefinovaných třímístných celočíselných kódů, se zobrazí příslušná chybová hláška.

V systému jsou nadefinovány pouze nejběžnější chybové hlášky (16).

Definice probíhá v konfiguračním souboru `Web.config` za použití již nadefinovaných routovacích pravidel:

```
<system.web>
  <customErrors mode="On" defaultRedirect="~/Error">
    <error statusCode="400" redirect="~/Error/400" />
    <error statusCode="401" redirect="~/Error/401" />
    <error statusCode="402" redirect="~/Error/402" />
    <error statusCode="403" redirect="~/Error/403" />
    <error statusCode="404" redirect="~/Error/404" />
    <error statusCode="408" redirect="~/Error/408" />
    <error statusCode="414" redirect="~/Error/414" />
    <error statusCode="500" redirect="~/Error/500" />
    <error statusCode="503" redirect="~/Error/503" />
  </customErrors>
</system.web>
```

Šablona pro chové hlášky je úmyslně odlišná, grafika bez obrázků pouze v černé a bílé barvě:



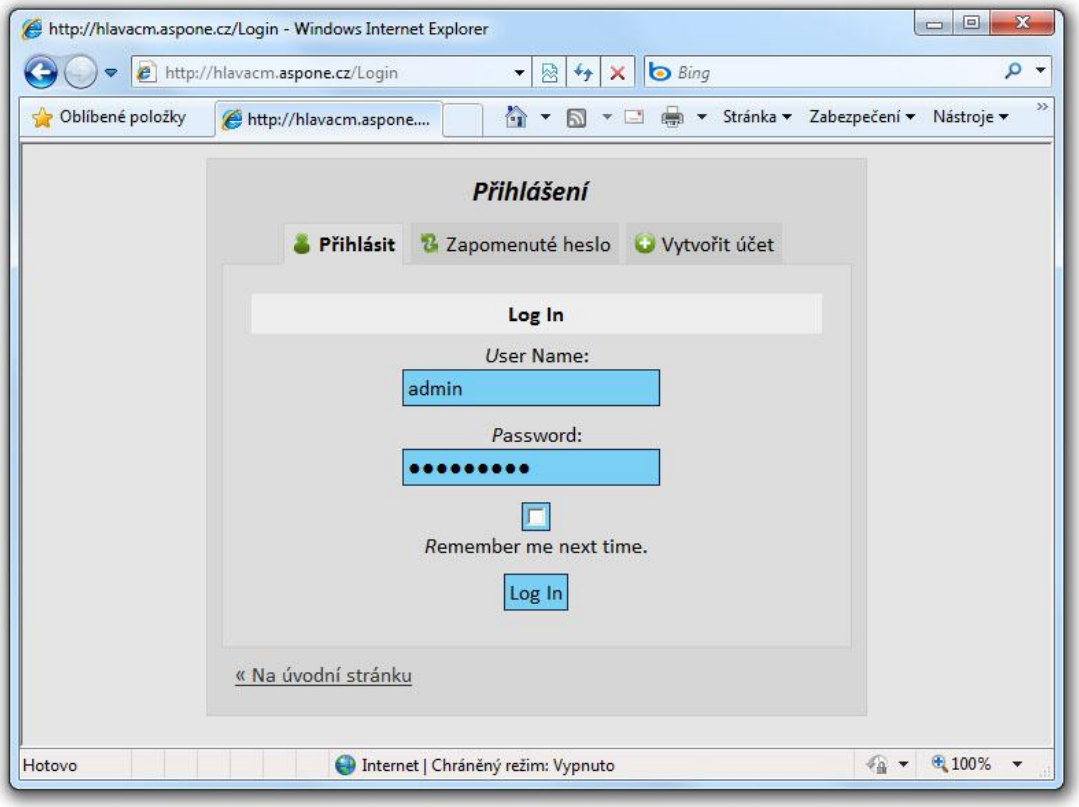
Obrázek 18 – ukázka chybového hlášení 404.

6.2.7 Membership

ASP.NET obsahuje velice slušnou podporu pro práci s uživateli, kterou je možné snadno a rychle integrovat přímo do aplikace. Není tak nutné složitě vytvářet vlastní řešení pro přihlašování a další práci s uživateli, ale plně stačí používat již hotové rozhraní členství podle svých představ.

Celý systém členství se skládá ze čtyř vrstev, které na sebe navazují:

- Ovládací prvky – ASP.NET obsahuje celou řadu ovládacích prvků připravenou pro práci s členstvím. V systému jsou použity všechny základní, pro přihlášení, obnovu hesla, změnu hesla, či registraci nového uživatele a další.
- API členství – Především třídy Membership a MembershipUser, které obsahují informace o uživatelích a jejich správu. Třída Membership je v systému používána pro ověřování a získání informací o aktuálně přihlášeném uživateli.
- Poskytovatelé členství (Membership Providers) – provádí ověřování totožnosti jednotlivých uživatelů z používaného úložiště. Místo nativního řešení SqlMembershipProvider je v systému použit již zmíněný externí Simple SQL Provider, resp. část pro členství (MembershipProvider) a uživatelské role (RoleProvider).
- Úložiště členství – libovolné úložiště dat pro uživatele v závislosti na použitém poskytovateli, nejčastěji se jedná o SQL server (7, s. 949-951).



Obrázek 19 – ukázka přihlašovací stránky.

6.2.8 LinqSiteMapProvider

Ve výchozím nastavení ASP.NET se jako základní datový zdroj ovládacích prvků určených k navigaci používá prvek SiteMapDataSource a jeho výchozím zdrojem dat je XML soubor s mapou webu, např. `Web.sitemap`. Ovšem v celém systému se pro přístup k datům používá LINQ to SQL, což zajišťuje vysokou dynamičnost. LinqSiteMapProvider je proto vylepšená verze vlastního poskytovatele mapy webu podle SqlSiteMapProvider, ze kterého přímo vychází, ale je upravena pro potřeby systému. Přímo pracuje se systémovou databází a zpracovává zobrazení hierarchických dat (7, s. 778-788).

V souboru `Web.config` je nutné nadefinovat minimálně jeden záznam poskytovatele mapy webu typu LinqSiteMapProvider a ten označit jako výchozí, dále pro správnou funkci multi-jazyčnosti musí být přidán pro každý jazyk v systému vlastní záznam poskytovatele mapy webu, ovšem s nadefinovanou zkratkou konkrétního jazyka:

```
<system.web>
  <siteMap defaultProvider="DefaultSiteMapProvider">
    <providers>
      <add name="DefaultSiteMapProvider"
        type="LinqSiteMapProvider"
        cacheTime="10" />
      <add name="SiteMapProvider-cz"
        type="LinqSiteMapProvider"
        cacheTime="10"
        langUrl="cz" />
      <add name="SiteMapProvider-en"
        type="LinqSiteMapProvider"
        cacheTime="10"
        langUrl="en" />
      <add name="SiteMapProviderMenuAdmin"
        type="System.Web.XmlSiteMapProvider"
        siteMapFile="~/Admin/Admin.sitemap"
        securityTrimmingEnabled="true" />
    </providers>
  </siteMap>
</system.web>
```

Název jednotlivých poskytovatelů může být libovolný, ale v případě konkrétních jazyků musí končit pomlčkou, za kterou následuje správná zkratka jazyka použitá v systému, aby mohlo dojít k identifikaci.

V systému je dále ještě použit pro účely administrace statický XML soubor s mapou webu `Admin.sitemap`, jelikož v administraci se nepředpokládá dynamická změna struktury.

6.2.9 Global Resources

Globální zdroje fungují na principu slovníku klíč-hodnota (a případně ještě popis). Jsou ideálním prostředkem pro realizaci vícejazyčnosti, i protože je jejich dostupnost možná z jakéhokoliv místa v aplikaci. Ukládají se do systémové složky ASP.NET App_GlobalResources v kořenovém adresáři aplikace (7, s. 205). Jejich koncová implementace se provádí v libovolném textovém ovládacím prvku, nejčastěji se ale pro zobrazení textů používá ovládací prvek Localize:

```
<asp:Localize ID="lcEditTitle" runat="server"
Text="<%$ Resources: Lang, Title %>" />
```

Lokalizačních souborů může být libovolně mnoho, ale je vhodné používat pouze několik souborů určených ke specifickým účelům. Samotná funkce probíhá tak, že výchozí lokalizační soubory jsou nějak pojménovány a končí příponou `.resx`. Pro jakýkoliv další jazyk jsou ty samé soubory pojmenovány stejně, ovšem s tím rozdílem, že před koncovkou ještě obsahují označení konkrétního jazyka. V systému jsou použity celkem tři jazykové soubory pro každý jazyk:

- Lang – výchozí systémový lokalizační soubor.
- AdminNavigation – lokalizační soubor pro navigaci v administraci.
- CustomErrors – lokalizace vlastních chybových hlášek.

Konkrétní lokalizační soubory v systému včetně cesty:

```
~/ App_GlobalResources/Lang.resx
~/ App_GlobalResources/Lang.en-US.resx
~/ App_GlobalResources/AdminNavigation.resx
~/ App_GlobalResources/AdminNavigation.en-US.resx
~/ App_GlobalResources/CustomErrors.resx
~/ App_GlobalResources/CustomErrors.en-US.resx
```

Výchozím jazykem v systému je čeština, tudíž není potřeba přidávat zkratku `cs-CZ`, ale pro všechny ostatní jazyky již konkrétní soubory musí obsahovat specifický identifikátor. Druhým jazykem je americká angličtina. Bylo by možné použít obecnou identifikaci angličtiny `en`, jenže ta by pak byla sdílená pro všechny další společné kultury a pro správnou funkci globalizace je na snaze jasně definovat konkrétní kulturu. V případě, že má stránka nastavenou kulturu, ale lokalizační soubor neexistuje nebo není dostupný, použije se výchozí lokalizační soubor. Stejně tak, pokud nějaký záznam v lokalizačním souboru chybí, použije se stejný záznam z výchozího souboru. Tam ale chybět nesmí, jinak by kompilátor vyhodil chybu.

6.3 Webová aplikace

Samotný systém je rozdělen na administrační a uživatelskou část. Je tedy využito konceptu, kde je administrace oddělena od zbytku aplikace.

6.3.1 Administrace

Jednotlivé součásti administrace jsou umístěny v adresáři `~/Admin`. Pomocí vlastního souboru `Web.config` mají všechny administrační webové formuláře nastaven jednotný vzhled a omezen přístup pouze na přihlášené uživatele, dále je přístup do daných sekcí řešen pomocí přístupových práv.

Všechny administrační sekce mají vždy vlastní webový formulář, jehož jednotlivé funkce obstarávají patřičné uživatelské ovládací prvky. Uživatelské ovládací prvky jsou podle použité adresářové struktury umístěny v adresáři Parts s výjimkou souborů pro moduly, které jsou umístěny v adresáři Modules.

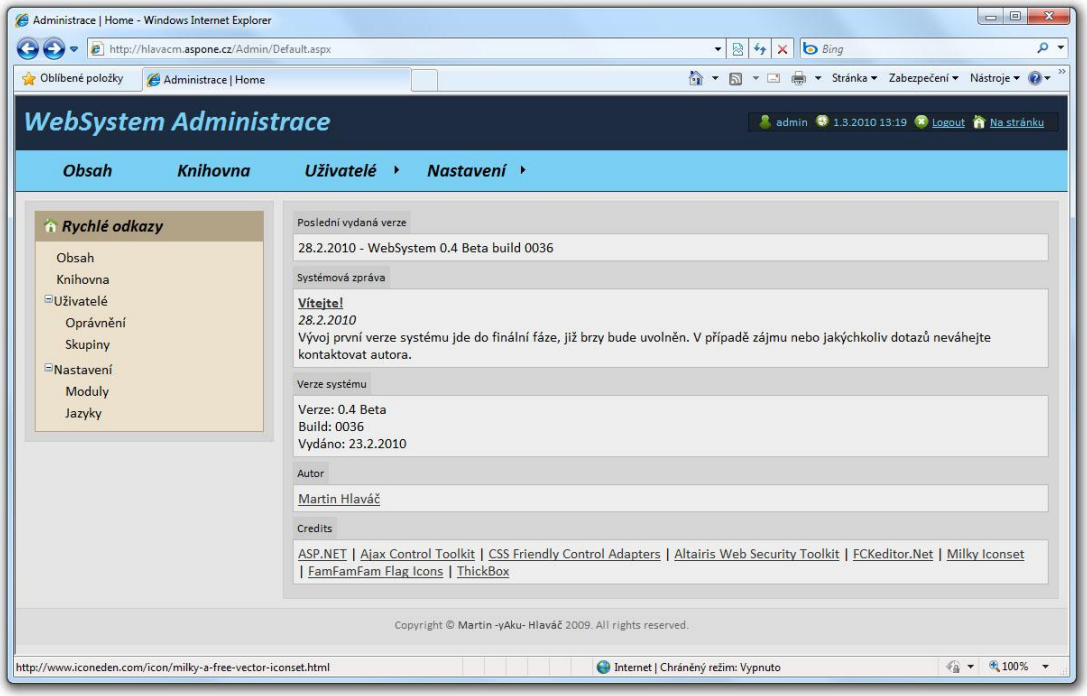
Navigace v administraci je řešena pomocí statického souboru `Admin.sitemap`, jelikož se zde nepředpokládá dynamická změna dat ani struktury. Soubory mapy webu navíc poskytují velmi dobrou podporu integrace globálních zdrojů, takže není problém s vícejazyčností.

Administrace má i vlastní vzor stránek `Admin.master`, který používají všechny zdejší webové formuláře. Tento vzor stránek obsahuje kromě definice vzhledu a rozložení ovládacích prvků také kontrolu zapnutého Javascriptu v prohlížeči, ověření přihlášení uživatele, nastavení titulku stránky a další.

Jednotlivé části administrace:

- Default – výchozí stránka.
- Content – správa obsahu.
- Library – prohlížeč souborů v knihovně a jejich správa.
- Users – správa uživatelů.
    - Permissions – správa oprávnění.
    - Roles – správa uživatelských skupin.
- Settings – jednotlivá nastavení a jejich editace.
    - Modules – správa modulů.
    - Languages – správa jazyků.

Ukázka úvodní obrazovky administrace:



Obrázek 20 – úvodní stránka administrace.

6.3.2 Uživatelská část

Jednotlivé součásti uživatelské části jsou umístěny v kořenovém adresáři. Přístup do uživatelské části není omezen, kromě sekce pro osobní nastavení uživatelů, která je dostupná pouze konkrétnímu uživateli po přihlášení. Pomocí souboru `Web.config` je celé aplikaci nastaven výchozí vzhled, pokud není uvedeno jinak.

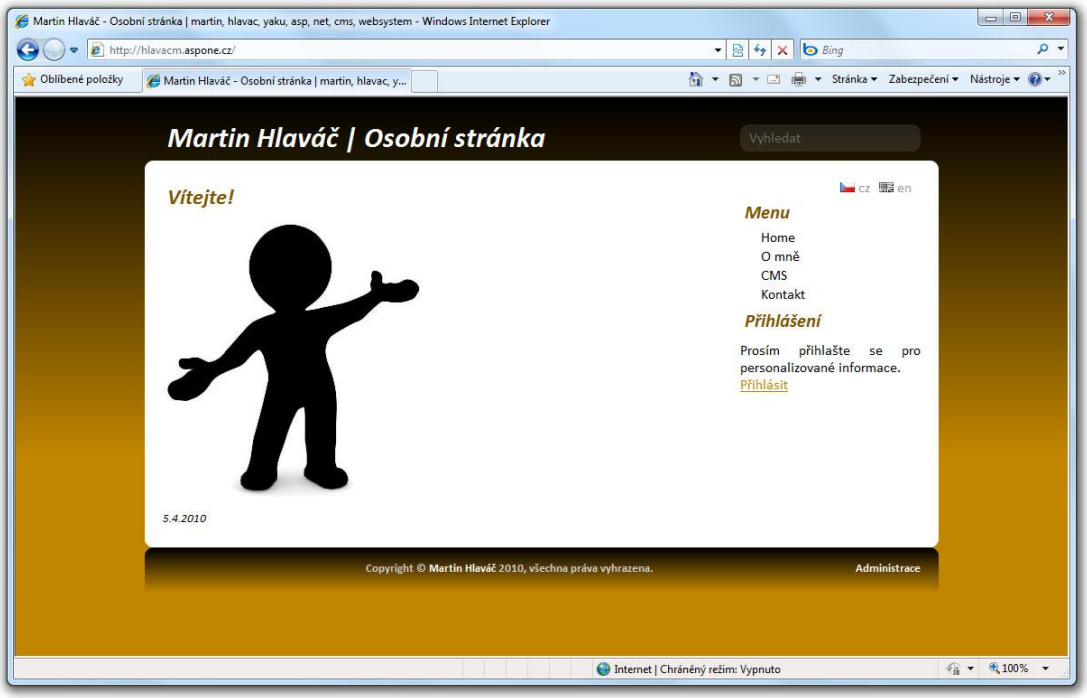
Navigace je vyřešena pomocí ovládacího prvku TreeView, jehož datovým zdrojem je LinqSiteMapProvider, který podporuje dynamický obsah a vícejazyčnost, což je zde na rozdíl od administrace víc než žádoucí. Stejně jako v administraci, i zde pro jednotlivé sekce existuje vždy webový formulář složený z patřičných uživatelských ovládacích prvků.

Výchozí vzor stránek pro uživatelskou část `Default.master` obsahuje samozřejmě definice vzhledu a rozložení ovládacích prvků na stránce a na straně kódu mimo jiné inicializaci správného jazyka včetně jeho navigace, načtení správného záznamu nastavení, výpis titulku, klíčových slov nebo implementaci pro aktuálně používaný javascriptový box skript.

Rozdělení uživatelské části:

- Default – výchozí stránka.
- Pages – webový formulář pro zobrazení obsahu modulu stránek.
- Personal – správa osobního nastavení.
- Search – zpracování vyhledávání.
- Login – přihlášení, obnova hesla nebo vytvoření nového uživatele.
- Error – zobrazení vlastních chybových hlášek.

Ukázka úvodní stránky webu:



Obrázek 21 – úvodní stránka webu.

6.3.3 Hlášení

Systém je vybaven vlastním řešením pro zobrazování chybových hlášek. Jedná se o uživatelský ovládací prvek `Report.ascx`, který zobrazuje libovolný text ve dvou režimech:

- Staticky – text je staticky zobrazen a zůstává neměnný.
- Dynamicky – text je dynamicky zobrazen a po nastaveném čase zmizí.

Jednotlivé chybové hlášky jsou umístěny v globálních zdrojích, odkud jsou podle potřeby volány.

Dynamické zobrazení umožňuje kombinace ovládacích prvků Timer a AnimationExtender, které umožňují v libovolném čase zobrazení a zmizení chybové hlášky, navíc s pěknými efekty:

```
<asp:Timer ID="Timer" runat="server" Interval="5000"
    Enabled="false" OnTick="Timer_Tick" />
<asp:Panel ID="pReport" runat="server" Visible="false">
    <div class="info-panel">
        <asp:Label ID="lReport" runat="server" Text="" />
    </div>
</asp:Panel>
<ajaxToolkit:AnimationExtender ID="pReportAE" runat="server"
    TargetControlID="pReport">
    <Animations>
        <OnLoad>
            <Sequence>
                <FadeIn Duration="1" Fps="24" />
                <FadeOut Duration="4" Fps="24" />
            </Sequence>
        </OnLoad>
    </Animations>
</ajaxToolkit:AnimationExtender>
```

Interval dynamického zobrazení je nastaven na 5 vteřin. Zobrazování chybových hlášek tímto způsoben je v celém systému.

6.3.4 Vyhledávání

Do uživatelské části systému je zabudována i jednoduchá funkce vyhledávání v obsahu. Vzhledem k tomu, že všechen obsah je vždy přiřazen k vlastnímu itemu (záznamu), může být základní vyhledávání řešeno právě přes údaje v itemech.

Vyhledávání probíhá podle zadaného textového řetězce o délce 3-20 znaků, kratší nejsou akceptovány a delší naopak zkráceny. Výsledky vyhledávání jsou vždy pro aktivní jazyk, pokud žádný jazyk není vybraný, použije se výchozí.

6.3.5 Osobní nastavení

Po přihlášení může každý uživatel využít osobní stránku, kde kromě přehledu osobních informací, má možnost změnit si heslo. Zde by mohly být přidány další funkce, například různé profilové nastavení, ovšem pro základní účely systému použité možnosti plně stačí.

Odkaz na osobní nastavení se zpřístupní po přihlášení v osobním (login) panelu nebo jednoduše na adrese `/Personal`. Celá sekce je dostupná pouze přihlášeným uživatelům.

6.3.6 Přístupová práva

Problém přístupových práv je v systému vyřešen pomocí statických konstant a jejich ověřování v databázi. Každý webový formulář i uživatelský ovládací prvek mají pevně přidělenou statickou konstantu, pro lepší přehlednost a jednoduchost se skládá z cesty umístění daného souboru. V databázi je seznam všech těchto potřebných konstant, které jsou přidělovány skupinám. Přístup do libovolné sekce je pomocí patřičné konstanty ověřen podle skupiny aktuálně přihlášeného uživatele a jeho přístupových práv:

```
if (dr.PermissionsGetByUser(permissionName,
    HttpContext.Current.User.Identity.Name) == false ||
    HttpContext.Current.User.Identity.IsAuthenticated == false)
{
    tcContent.Enabled = false;
    tcContent.Visible = false;
    Report.ShowStatic(Resources.Lang.MessageNoAccess);
}
```

Přičemž `permissionName` je právě statická konstanta dané sekce. Pro ověření se používá metoda `PermissionsGetByUser` z repositáře, která podle zadaného přístupového práva a uživatelského jména povolí či zakáže přístup a případný neúspěch je zobrazen pomocí statické chybové hlášky.

6.4 Problémy při implementaci

Při vytváření systému byly odhaleny různé problémy v prostředí ASP.NET. Za některými problémy stály chyby z externích knihoven, za jinými způsob implementace nebo samotné prostředí.

6.4.1 TreeView

U navigačního ovládacího prvku TreeView s využitím CSS Control Adapters se objevila chyba v události `SelectedNodeChanged`, která reaguje na změnu vybraného uzlu. Událost je ve výchozím stavu identifikována jako `protected`, což se ukázalo za problémové, protože identifikátor je nutné nastavit jako `public`. Navíc se musí jako u události `OnSelectedNodeChanged` přidat stejný odkaz události `OnAdaptedSelectedNodeChanged`.

```
<asp:TreeView ID="tvContent" runat="server" ExpandDepth="1"
   OnSelectedNodeChanged="tvContent_SelectedNodeChanged"
   OnAdaptedSelectedNodeChanged="tvContent_SelectedNodeChanged"
   CssSelectorClass="SimpleTreeView">
</asp:TreeView>
```

6.4.2 ToolkitScriptManager

Aby na stránce bylo možné používat ajaxové ovládací prvky, musí stránka obsahovat právě jednu referenci prvku ToolkitScriptManager, který se stará o jejich správu. Ovšem ten má ve výchozím stavu vypnutou podporu globalizace a některé prvky závislé na rozdílném nastavení kultur, nemusí správně pro konkrétní kultury pracovat, proto je nutné nastavit jeho vlastnost `EnableScriptGlobalization` na hodnotu `True`:

```
<ajaxToolkit:ToolkitScriptManager ID="ToolkitScriptManager"
    runat="server" EnableScriptGlobalization="true"/>
```

6.4.3 CollapsiblePanelExtender

Pro skrývání a odkrývání části obsahu slouží celá řada ovládacích prvků, nabízí se například prvek typu Accordition, ovšem jeho řešení v rámci Ajax Control Toolkitu je poněkud krkolomné, především přístup k dalším ovládacím prvkům, jež obsahuje, není standardní. Proto je vhodné za tímto účelem používat raději ovládací prvek CollapsiblePanelExtender, který spolupracuje s klasickými prvky Panel a stará se pouze o funkci jejich skrývání a odkrývání:

```
<ajaxToolkit:CollapsiblePanelExtender ID="cpeExtended"
    runat="Server" ExpandControlID="pExtendedHeader"
    CollapseControlID="pExtendedHeader"
    TargetControlID="pExtended" TextLabelID="lcExtendedText"
    ExpandedText="<%$ Resources: Lang, HideDetails %>"
    CollapsedText="<%$ Resources: Lang, ShowDetails %>"
    Collapsed="True" SuppressPostBack="true" />
```

6.4.4 FileUpload

Ovládací prvek FileUpload slouží k výběru souboru a jeho následnému nahrání do uvedené lokace (na server). Ovšem jeho použití v UpdatePanelu je značně omezené. Pro správnou funkci se musí tlačítko určené k funkci nahrávání registrovat u UpdatePanelu jako PostBackTrigger a dále musí obsahovat specifikovanou událost `OnClientClick` na straně klienta (17):

```
<asp:Button ID="bUpload" runat="server" OnClick="bUpload_Click"
    Text="<%$ Resources: Lang, Upload %>"
    OnClientClick="javascript:document.forms[0].encoding =
    'multipart/form-data';" />

<asp:UpdatePanel>
    <Triggers>
        <asp:PostBackTrigger ControlID="bUpload" />
    </Triggers>
</asp:UpdatePanel>
```

6.4.5 FCKeditor

Pro větší komfort a jednoduchost je vhodné při editaci obsahu používat libovolný webový WYSIWYG editor. Jako nejlepší kandidát se ukázal FCKeditor.NET (10), především kvůli jeho maximální integraci v rámci ASP.NET. Pro jeho správnou funkci se ale musí provést několik kroků. Prvním je stažení správného balíku samotného editoru z archivu ve verzi FCKeditor 2.6.3. Dále má ovládací prvek FCKeditorV2 funkční omezení práce v UpdatePanelu, což vyřeší jeho registrace u ScriptManageru při inicializaci v metodě `Page_Load` (18):

```
ScriptManager.RegisterOnSubmitStatement(fckEditText,
    fckEditText.GetType(), "editor", "FCKUpdateLinkedField('" +
    fckEditText.ClientID + "');");
```

Dále musí být ve hlavičce stránky přidána javascriptová funkce (18):

```
<script type="text/javascript">
    function FCKUpdateLinkedField(id) {
        try {
            if (typeof (FCKeditorAPI) == "object") {
              FCKeditorAPI.GetInstance(id).UpdateLinkedField();
            }
        }
        catch (err) { }
    }
</script>
```

Pro správné fungování nahrávání souborů pomocí FCKeditoru musí existovat adresář s libovolným názvem, např. v systému použitý adresář Library, uvnitř kterého musí být složky pro výchozí sekce: file, flash, image, media.

Protože výchozím konektorem pro nahrávání je nastaveno php, musí se v konfiguračním souboru `fckconfig.js` provést změna na hodnou aspx:

```
var _FileBrowserLanguage = 'aspx' ;  // asp | aspx | perl | php
var _QuickUploadLanguage = 'aspx' ;  // asp | aspx | perl | php
```

Dále je nutné upravit konfiguraci pro aspx konektor tak, že metoda `CheckAuthentication` by měla vracet hodnotu `true`, jinak bude obsah nepřístupný a proměnné `UserFilesPath` správně nastavit cestu (19), např.:

```
UserFilesPath = Page.ResolveUrl("~/Library/");
```

Nakonec musí být provedena poslední úprava, deaktivace vzhledu v souborech aspx konektoru `connector.aspx` a `upload.aspx`:

```
<%@ ... StylesheetTheme="" Theme="" %>
```

6.4.6 Prohlížeče

Během testování systému v různých prohlížečích, se objevil problém s načítáním AJAXových ovládacích prvků v prohlížečích založených na jádře Webkit (Chrome, Safari), kdy nedocházelo ke správné inicializaci prohlížeče. Pro vyřešení problému je nutné vytvořit javascriptový soubor (20), např: `Webkit.js`:

```
Sys.Browser.WebKit = {};
if( navigator.userAgent.indexOf( 'WebKit/' ) > -1 )
{
  Sys.Browser.agent = Sys.Browser.WebKit;
  Sys.Browser.agent = Sys.Browser.WebKit;
  Sys.Browser.version = parseFloat(
    navigator.userAgent.match(/WebKit\/(\d+(\.\d+)?)/)[1]
  );
  Sys.Browser.name = 'WebKit';
}
```

Tento soubor se následně musí v příslušném webovém formuláři přiřadit ke Script Manageru:

```
<ajaxToolkit:ToolkitScriptManager ID="ToolkitScriptManager"
    runat="server" EnableScriptGlobalization="true">
    <Scripts>
        <asp:ScriptReference Path="~/Scripts/Webkit.js" />
    </Scripts>
</ajaxToolkit:ToolkitScriptManager>
```

6.4.7 Vlastní chybové hlášení 404 na IIS

Jediný problém, který se nepodařilo vyřešit, se vyskytl v souvislosti s použitím vlastních chybových hlášení v kombinaci s hezkými URL pomocí routování. Problém je v nastavení IIS. Když běží v integrated módu, tak vlastní chybové hlášení 404 (not found) fungují pouze pro soubory webových formulářů (koncovka `.aspx`). Na webhostingu běží webové aplikace bohužel právě v tomto módu (21).

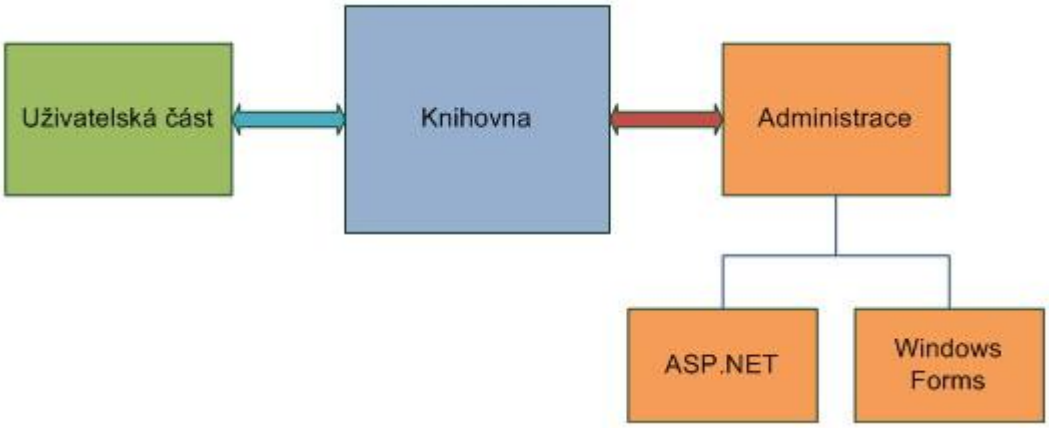
6.5 Rozšíření aplikace

Přístup a ovládání systému pomocí webového rozhraní je primární cíl této práce, ovšem koncepce systému je taková, že v případě potřeby může být vytvořeno rozhraní k systému v libovolném prostředí .NET Frameworku. Vzhledem k tomu, že se technologie v současné době začíná prosazovat do náročnějších implementací, tak musíme mít na paměti, že ne vždy bude webové rozhraní dostačující. Při nasazení CMS systému v kombinaci například s modulem pro internetové obchodování je dost pravděpodobné, že bude požadováno napojení na další externí řešení.

V moderních systémech, zejména ve velkých B2B aplikacích je zapotřebí například napojení na další hardware jako jsou čtečky nebo externí software jako jsou různé ekonomické a informační systémy (ERP) nebo systémy pro řízení vztahů se zákazníky (CRM). Tento proces se nazývá systémová integrace. Cílem je spojit všechny softwarové, popř. i hardwarové komponenty v jeden fungující celek. Využití systémové integrace se nabízí v případě nasazení nového systému, při implementaci nového distribučního kanálu nebo pro zajištění konzistence mezi různými systémy. A to právě umožňuje desktopová aplikace, která využívá jádra systému (22).

Pro jasnou ukázku univerzálnosti řešení systému slouží demoverze desktopové aplikace WinForms klient, která prostřednictvím technologie Windows Forms dokáže zprostředkovat stejné možnosti jako vytvořený webový klient za použití stejných metod a postupů v přístupu k systémové knihovně. Rozdíl je pouze v řešení uživatelského rozhraní.

Desktopové Windows Forms prostředí nabízí kromě jiného oproti webovému prostředí ASP.NET v základu větší sadu ovládacích prvků. Výhodou je, že většinu základních ovládacích prvků obsahují obě platformy stejně a navíc i způsob používání se mezi nimi příliš neliší.

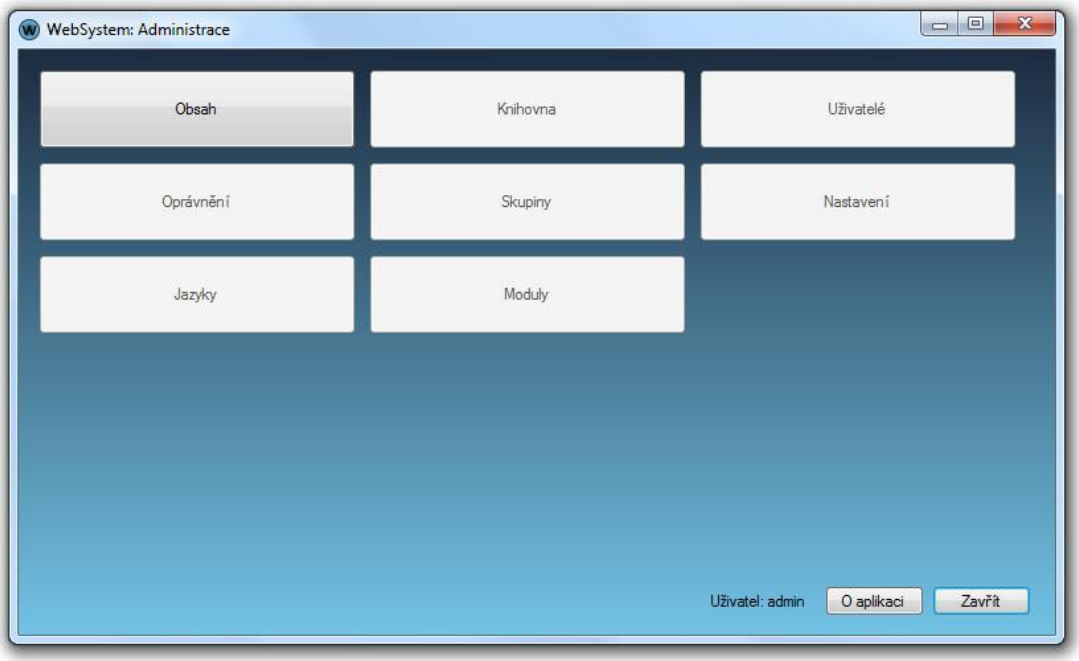


Obrázek 22 – obecné schéma systému.

Samotné jádro systému tvoří knihovna, která představuje datový most mezi uživatelskou částí a administrací. To právě způsobuje, že administrace může být klidně přístupná buď pomocí webového prostředí, nebo jako desktopová aplikace. Zároveň díky zvolené technice naprogramování knihovny může být administrace napsána v libovolném prostředí .NET Frameworku.

Pro realizaci stačí pouze importovat potřebné knihovny, tedy systémovou a Altairis Web Security. Pak přijdou na řadu nezbytné jmenné prostory a nastavení konfigurační části souboru `App.config`. Nakonec už stačí postupovat stejně jako na webu a vytvořit uživatelské ovládací prostředí.

Konkrétně se jedná o jmenné prostory System.Web a System.Web.Security, které zajistí stejnou podporu Memberships jako ve webové aplikaci. Pomocí třídy MembershipUser lze pak jednoduše pracovat s existujícími uživateli. Není tak problém implementovat funkci pro přihlášení nebo vytvoření nového uživatele. V souboru `app.config` jde především o zkopírování částí definic connectionStrings a Memberships (membership, roleManager) ze souboru `Web.config` webové aplikace. U připojovacích řetězců musí být hlavně správně nastaven ten výchozí.



Obrázek 23 – úvodní obrazovka demo verze WinForms klienta.

Stejně jako u webové aplikace je vhodné pro každou sekci vytvořit formulář a pro jejich případné podsekce uživatelské ovládací prvky. Navigace musí být naopak pozměněna, kvůli rozdílné funkčnosti prvku TreeView. Protože Windows Forms neobsahují podporu pro vzory stránek, tak uspořádání administrace musí být vyřešeno jiným způsobem, například pomocí úvodního formuláře se seznamem odkazů na další sekce. Běžným prostředkem interakce v desktopových aplikacích jsou tlačítka.

V prostředí Windows Forms není možné přímo využít pro formátování textu FCKeditor.NET, jelikož je jeho určení výhradně do webového prostředí. Proto musí být použit nějaký jiný editor určený pro desktopové aplikace, kde je ovšem podstatně menší výběr možných kandidátů. Nejjednodušší možností je použít editor Xstandard ve verzi Lite (23), která je zdarma, ovšem kromě textové reklamy obsahuje i podstatná funkční omezení. V bezplatné verzi editoru je omezeno především nahrávání obrázků a souborů na server.

Aby mohlo být vytvořeno spojení s databází na serveru, tak musí být na webhostingu povolena funkce vzdáleného připojení, jinak připojení k SQL serveru nebude možné otevřít. Bohužel freehotingové služby mají tuto funkci zakázanou.

7. Nároky a instalace systému

Systém je výhradně určen pro webový server IIS (Internet Infomation Server) s podporou ASP.NET a databáze MSSQL Express. Pro provoz systému je možné využít libovolnou webhostingovou službu podporující ASP.NET nebo vlastní server, pokud budou splněny následující základní parametry:

- IIS 7.0 nebo vyšší
- ASP.NET 3.5
- databáze MSSQL Express 2005 nebo 2008
- podpora LINQ to SQL a ASP.NET AJAX
- alespoň 15MB prostoru pro aplikaci a 3 MB pro databázi

7.1 Instalace systému na webhosting

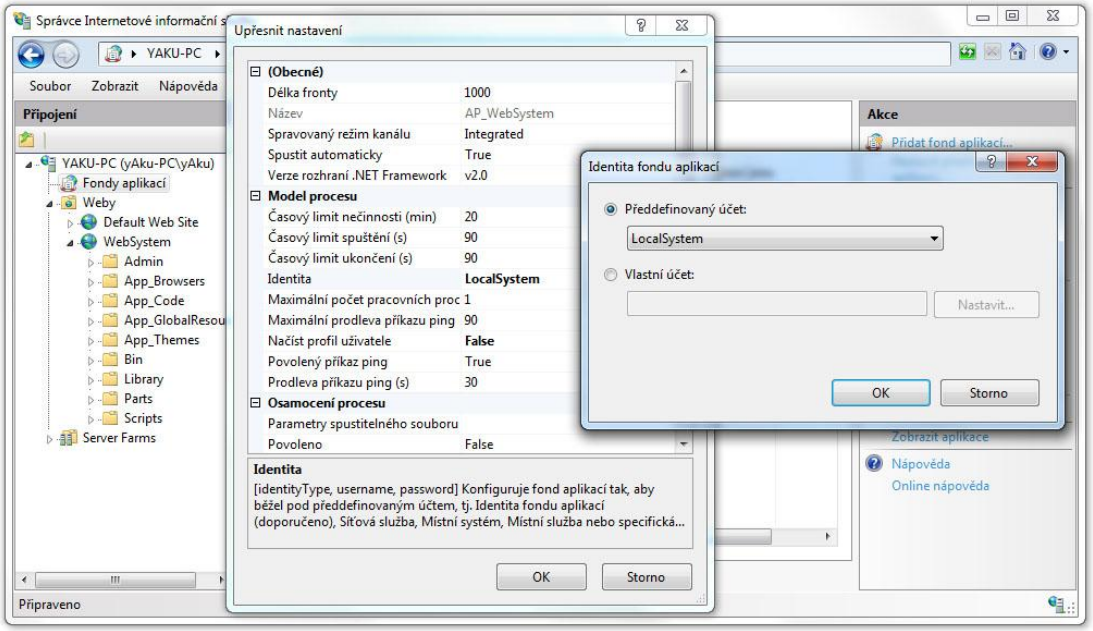
- Obsah kořenového adresáře aplikace (WebSystem) zkopírujte do WWW adresáře na serveru.
- Adresář Library a všechny jeho podřízené musí mít nastavena přístupová práva pro zápis.
- Na serveru vytvořte novou databázi a zkopírujte do ní data:
    - buď jednoduše pomocí nástroje SQL Database Publishing wizard (24), díky kterému je možné zkopírovat celou databázi ze souboru `websystem.mdf` včetně dat,
    - nebo ručně pomocí připravených instalačních SQL skriptů `db.sql` (pro strukturu) a následně `data.sql` (pro základní data).
- Upravte konfigurační soubor `Web.config` podle parametrů vašeho webhostingu, především je nutné správně nastavit výchozí připojovací řetěz k databázi (DefaultConnectionString).
- Přihlaste se pomocí výchozího a administračního uživatelského účtu:
    - uživatelské jméno: admin,
    - heslo: hesslo123

Údaje výchozího uživatelského účtu je možné změnit po přihlášení nebo rovnou během ruční instalace v instalačním skriptu `data.sql`.

Pozor! V případě, že by po úspěšné instalaci nebylo možné se přihlásit, tak je nutné kontaktovat technickou podporu webhostingu s prosbou o zvýšení práv kvůli použité knihovně Altairis Web Security pro správu uživatelů a přihlašování.

7.2 Zprovoznění na lokálním počítači

Instalační postup na lokálním počítači je podobný jako na webhostingu, ale databázi je možné přidat i přes nástroj SQL Management Studio a dále pro správný chod systému na lokálním počítači v rámci internetové informační služby IIS je nutné založit nový APPPOOL, např. AP_WebSystem a v jeho sekci Upřesnit nastavení, nastavit vlastnost Identita na hodnotu "LocalSystem" (24).



Obrázek 24 – IIS localhost upřesnění nastavení.

Na lokálním počítači může být také pro testovací účely systém spuštěn ve Visual Studiu v rámci ASP.NET Development Server.

8. Testování aplikace

Hotový systém byl otestován na lokálním počítači v rámci internetové informační služby IIS, dále na veřejně dostupných serverech ASPone a ASP 2 v rámci freehostingových variant (26, 27).

Cílem testování bylo nalezení možných chyb, jejich případné odstranění, otestování přehlednosti uživatelského rozhraní a práce se systémem.

Testování probíhalo na různých počítačích a mobilních zařízeních s operačními systémy:

- Windows 7
- Windows Vista
- Windows XP
- Arch Linux

Při testování byly použity všechny známé a nejpoužívanější internetové prohlížeče podle statistiky gemiusRanking pro Českou republiku (28):

- Internet Explorer
- Firefox
- Opera
- Chrome
- Safari

Před samotným testováním byla provedena maximální optimalizace systému, kterou následoval test v uzavřené skupině uživatelů dle připraveného scénáře.

Během testování nedošlo k žádným závažným nedostatkům. Nalezené menší chyby byly opraveny a připomínky uživatelů vyslyšeny. Na základně provedení a vyhodnocení testování byl systém optimalizován a měl by být snadno ovladatelný i přístupný.

Systém byl úspěšně otestován i na mobilním zařízení HTC Diamond 2 s operačním systém Windows Mobile 6.5 v mobilním internetovém prohlížeči Opera 10 s plnou funkčností systému.

8.1 Testovací scénář

Celkem bylo osloveno 10 vybraných uživatelů s různými zkušenostmi práce na počítači a internetu. Každý uživatel dostal k dispozici vlastní administrátorský účet do testovací verze systému a byl vyzván ke splnění několika úkolů:

- Přihlásit se do administrace systému.
- Vytvořit nový záznam, ke kterému přidat vlastní obsah včetně nahrání obrázku z počítače do knihovny systému.
- Vytvořený záznam v editaci odstranit a následně ze seznamu odstraněných zase obnovit.
- Vytvořit nového uživatele, vytvořit novou skupinu a uživatele do ní zařadit.
- Vytvořené skupině nastavit libovolně přístupová práva.
- Odhlásit se ze systému a v uživatelské části systému pomocí vyhledávacího formuláře najít vlastní přidaný záznam.
- Vyhodnotit práci se systémem v připraveném dotazníku.

8.2 Zhodnocení testování

Po dokončení testovacího scénáře byl uživatelům předložen dotazník, pomocí kterého celý průběh testu shrnuli. Všech 10 oslovených testerů by systém případně doporučili ostatním k používání nebo ho sami používali. Připravený testovací scénář zvládli v pořádku všichni oslovení. Provedený test ukázal, že je systém snadno ovladatelný i pro méně zkušené uživatele, kteří se správou webových stránek nemají zkušenosti.

Samotné přihlášení do systému nepředstavuje problém, pouze přechod z uživatelské části do administrace pomocí skrytého odkazu je na první pohled přehlédnutelný. Co se týče správy záznamů a obsahu, tak nejjednodušším krokem se ukázalo být přidání nového záznamu. Následné operace jako úprava, odstranění, či obnova záznamů již byly méně přehledné. Vytvoření nového uživatele i skupiny bylo pro testující převážně jasné, menší zádrhele, především v komfortním přístupu, vykázaly kroky zařazení uživatele do skupiny a nastavení přístupových práv. S odhlášením ze systému a následným vyhledáním přidaného záznamu, naopak nikoho problém neměl.

9. Závěr

Pro úspěšnou realizaci redakčního systému byla provedena analýza, na kterou navázala specifikace návrhových vzorů UML, pomocí které je model systému velmi dobře čitelný. Základní model představuje USE CASE diagram, jehož jednotlivé části jsou popsány textovou formou a dále vysvětleny graficky, pomocí příslušných diagramů interakce a dalších.

Samotná implementace navrženého modelu v prostředí ASP. NET vycházela z výsledků analýzy a UML návrhu. Hlavní součástí vytvořené aplikace je systémová knihovna, která má na starosti práci s daty. Důležitým úsekům samotné aplikace i knihovny je v práci věnována patřičná pozornost, problematické části kódu jsou vysvětleny a doplněny komentářem. Popsány jsou také využité ASP.NET praktiky a jejich konkrétní nasazení. Pro účely systému byl mimo jiné vytvořen vlastní jednoduchý systém chybových hlášení, přístupových práv a datový zdroj navigačních ovládacích prvků. Během vytváření aplikace byly nalezeny různé problémy. Za některými stály chyby z externích knihoven, za jinými způsob implementace nebo samotné prostředí, ale jejich řešení je vždy vylíčeno.

Výsledný systém je modulární a jeho nasazení univerzální. Díky modulárnímu návrhu je možné systém dále libovolně rozšířit vytvořením nových modulů a spravovat tak teoreticky jakýkoliv obsah.

I když se použitá implementace navrženého modelu v prostředí ASP.NET ukázala jako vhodná volba, nejen kvůli možnostem, skvělému přístupu, ale i díky celkovému výsledku, tak se neomezuje pouze na nějakou konkrétní platformu. Návrh modelu je takový, že pro vytvořenou systémovou knihovnu je možné vytvořit administrační část v některém z mnoha prostředí .NET Frameworku, nejen tedy v požadovaném ASP.NET. V přiložené demo aplikaci alternativního klienta je ukázána stejná funkčnost i v prostředí Windows Forms, stejně tak by bylo možné využít např. Windows Presentation Foundation nebo Silverlight a další.

Uživatelská i administrační část systému jsou snadno dostupné odkudkoliv na světě pomocí některého z běžně používaných internetových prohlížečů i operačních systémů. Ovládání systému je jednoduché a přehledné, systém může tedy bez sebemenších problémů ovládat i běžný uživatel.

Díky použité technologii AJAX je systém rychlý a není potřeba vždy celé stránky znovu načítat. Pro kvalitní provoz webových prezentací jsou implementovány běžné SEO praktiky, v čele s hezkými URL. Vestavěná podpora prostředí ASP.NET pro práci s uživateli zajišťuje bezpečnost celého systému. Uživatelská data jsou bezpečně uložena v databázi a na hesla je navíc kromě klasického šifrování použit tzv. solený hash (salted hash), který zajišťuje odlišnost a ještě horší čitelnost každého kódu.

Jasnou výhodou webového prostředí je jednoduchý přístup k systému pouze pomocí internetového prohlížeče napříč operačními systémy. Desktopová aplikace místo toho nemusí být závislá na internetovém připojení a připravuje lepší půdu pro případné rozšíření na další hardware nebo externí softwarové systémy.

Instalace a zprovoznění systému spočívá pouze v několika krocích, především se jedná o nakopírování souborů, nastavení individuálních parametrů jako je přístupový řetězec k databázi a ruční vytvoření nebo automatické zkopírování databáze. Správná funkce systému byla otestována na bezplatných variantách webhostingů dvou společností působících na našem trhu. I když je systém realizován v prostředí ASP.NET, není omezen na drahé, individuální webhostingové varianty.

10. Literatura

[1] *MediaCentrik* [online]. 2007-04-10 [cit. 2010-02-02]. CMS. Dostupné z WWW: <http://www.mediacentrik.cz/slovnik/cms.aspx>.

[2] *ASP.NET : The Official Microsoft ASP.NET Site* [online]. c2010 [cit. 2010-02-02]. Dostupné z WWW: <http://www.asp.net>.

[3] *W3C* [online]. c2009 [cit. 2010-02-02]. Standards. Dostupné z WWW: <http://www.w3.org/standards>.

[4] NOVÁK, Milan. *VOXCAFE - internet full service* [online]. 2006-11-01 [cit. 2010-02-02]. Desatero optimalizace stránek. Dostupné z WWW: <http://www.voxcafe.cz/clanky/optimalizace-stranek/desatero-optimalizace-stranek.html>.

[5] KANISOVÁ, Hana; MÜLLER, Miroslav. *UML srozumitelně*. 2. aktualiz. vyd. Brno : Computer Press, a. s., 2007. 176 s. ISBN 80-251-1083-4.

[6] ARLOW, Jim; NEUSTADT, Ila. *UML2 a unifikovaný proces vývoje aplikací*. 1. vyd. Brno : Computer Press, a. s., 2007. 568 s. ISBN 978-80-251-1503-9.

[7] MACDONALD, Matthew; SZPUSZTA, Mario. *ASP.NET 3.5 a C# 2008 : tvorba dynamických stránek PROFESIONÁLNĚ*. 1. vyd. Brno : Zoner Press, ZONER software, s.r.o., 2008. 1584 s. ISBN 978-80-7413-008-3.

[8] *Ajax Control Toolkit* [online]. 2009-11-18 [cit. 2010-02-11]. Dostupné z WWW: <http://ajaxcontroltoolkit.codeplex.com>.

[9] *ASP.NET 2.0 CSS Friendly Control Adapters 1.0* [online]. 2006-04-27 [cit. 2010-02-11]. Dostupné z WWW: <http://www.asp.net/CssAdapters>.

[10] *Altairis Web Security Toolkit* [online]. 2009-09-06 [cit. 2010-02-11]. Dostupné z WWW: <http://altairiswebsecurity.codeplex.com>.

[11] *FCKeditor.Net : ASP.Net Control to easily integrate FCKeditor on ASP.NET pages* [online]. 2008-07-26 [cit. 2010-02-11]. Dostupné z WWW: <http://ckeditor.com/download>.

[12] *Visual Paradigm for UML : UML tool for software application development* [online]. c2010 [cit. 2010-02-11]. Dostupné z WWW: <http://www.visual-paradigm.com/product/vpuml>.

[13] KOTTNAUER, Jakub. *Programujte.com* [online]. 2009-08-15 [cit. 2010-02-13]. ASP.NET MVC v praxi od A do Z, 3. díl – Model. Dostupné z WWW: <http://programujte.com/?akce=clanek&cl=2009061000-asp-net-mvc-v-praxi-od-a-do-z-3-dil-%96-model>. ISSN 1801-1586.

[14] DRLÍK, Zdeněk. *Interval.cz* [online]. 2005-03-25 [cit. 2010-02-13]. Používáme návrhové vzory v .NET - Singleton. Dostupné z WWW: <http://interval.cz/clanky/pouzivame-navrhove-vzory-v-net-singleton>.

[15] VALÁŠEK, Michal. *ASPNET.CZ* [online]. 2006-05-10 [cit. 2010-02-14]. Jak spolehlivě odstranit v .NET z řetězce diakritiku. Dostupné z WWW: <http://www.aspnet.cz/Articles/92-jak-spolehlive-odstranit-v-net-z-retezce-diakritiku.aspx>.

[16] SIL, Sanjit. *ASP Alliance* [online]. 2006-09-12 [cit. 2010-02-16]. Working with Custom Error Pages Using ASP.NET. Dostupné z WWW: <http://aspalliance.com/1008_Working_with_Custom_Error_Pages_Using_ASPNET>.

[17] *Stack Overflow* [online]. 2008 [cit. 2010-02-28]. FileUpload control inside an UpdatePanel without refreshing the whole page?. Dostupné z WWW: <http://stackoverflow.com/questions/35743/fileupload-control-inside-an-updatepanel-without-refreshing-the-whole-page>.

[18] *JLCoady.net* [online]. 2007-03-30 [cit. 2010-02-28]. Easy way to get FCKEditor to work inside an ASP.Net AJAX UpdatePanel. Dostupné z WWW: <http://jlcoady.net/aspnet/fckeditor-work-inside-updatepanel>.

[19] CHOWDHURY, Iqbal. *ASP Alliance* [online]. 2010-01-20 [cit. 2010-02-28]. Solution to ASP.net Theme Conflict with FCKEditor. Dostupné z WWW: <http://aspalliance.com/1908_Solution_to_ASPnet_Theme_Conflict_with_FCKEditor>.

[20] TURLOV, Alexander. *Web.NET* [online]. 2009-01-29 [cit. 2010-03-18]. ASP.NET AJAX Compatibility Patch for Safari 3.x and Google Chrome. Dostupné z WWW: <http://blog.turlov.com/2009/01/aspnet-ajax-compatibility-patch-for.html>.

[21] *RuslanY Blog* [online]. 2008-09-30 [cit. 2010-03-18]. Wildcard script mapping and IIS 7 integrated pipeline. Dostupné z WWW: <http://ruslany.net/2008/09/wildcard-script-mapping-and-iis-7-integrated-pipeline/>.

[22] VAIC, Miroslav; SOUKENÍK, Jiří. Systémová integrace. *BUSINESS WORLD* [online]. 2006, Duben, [cit. 2010-04-08]. Dostupný z WWW: <http://www.trask.cz/DeliverLive/digitalAssets/455_SystÃ©movÃ¡%20integrace_BW_0406.pdf>.

[23] *XStandard* [online]. c2010 [cit. 2010-03-01]. Free standards-compliant XHTML WYSIWYG editor. Dostupné z WWW: <http://www.xstandard.com>.

[24] *ASPone Help Wiki* [online]. 2008-12-04 [cit. 2010-02-28]. Jednoduchý přenos databáze z Vašeho počítače na ASPone SQL server. Dostupné z WWW: <http://wiki.aspone.cz/Jednoduchý_přenos_databáze_z_Vašeho_počítače_na_ASPone_SQL_server>.

[25] *CodeToday.NET* [online]. 2009-01-12 [cit. 2010-02-28]. How to: Login failed for user \'IIS APPPOOL\\DefaultAppPool\'. Dostupné z WWW: <http://www.codetoday.net/default.aspx?g=posts&t=1595>.

[26] *ASPone.cz* [online]. c2010 [cit. 2010-03-01]. Freehosting. Dostupné z WWW: <http://www.aspone.cz/20300/Webhosting-plan-freehosting.aspx>.

[27] *ASP 2* [online]. c2009 [cit. 2010-03-01]. Webhosting zdarma. Dostupné z WWW: <http://www.asp2.cz>.

[28] *GemiusRanking CZ* [online]. c2010 [cit. 2010-03-01]. Web browsers – groups. Dostupné z WWW: <http://www.rankings.cz/en/rankings/web-browsers-groups.html>.

11. Seznam obrázků

Příloha – obsah CD

Adresář (soubor)	**Popis**
Docs	Bakalářské práce ve formátech *.docx* a *.pdf*.
Images	Všechny použité obrázky pro bakalářskou práci v původním rozlišení.
SampleDB	Vzorová databáze.
SQL	Instalační SQL skripty.
Test	Výsledky testování systému.
Video	Video ukázky práce se systémem a WinForms klientem.
WebSystem	Systémový adresář včetně zdrojových kódů. Obsah toho adresáře tvoří systém a je určen k nahrání na web.
WebSystemForms	Zdrojový projekt a soubory demo verze klienta ve Winforms.
WebSystemLibrary	Zdrojový projekt a soubory systémové knihovny.
ReadMe.txt	Informační soubor.