VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

POKROČILÁ WEBOVÁ APLIKACE HOROLEZECKÉHO KLUBU

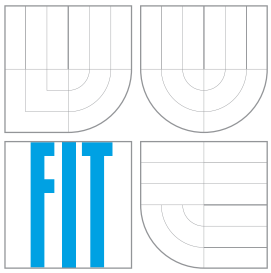
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB KADLAS

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

POKROČILÁ WEBOVÁ APLIKACE HOROLEZECKÉHO KLUBU

WEB APPLICATION OF HILL-CLIMBING CLUB

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB KADLAS

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LUKÁŠ ROMAN, Ph.D.

BRNO 2007

Zadání

Pokročilá webová aplikace horolezeckého klubu

1. Seznamte se s jazyky a prostředky pro tvorbu webových informačních systémů (XHTML, CSS, PHP, Javascript, MySQL).
2. Seznamte se s požadavky kladenými na IS horolezeckého klubu. Rozsah systému konzultujte s vedoucím BP. Požadavky podrobně analyzujte.
3. Proveďte návrh systému. Při analýze požadavků a návrhu využijte vhodných modelovacích technik. Systém musí obsahovat i speciální funkčnost založenou na nějaké metodě z umělé inteligence. Výběr této metody konzultujte s vedoucím BP.
4. Daný systém implementujte.
5. Zhodnoťte dosažené výsledky, porovnejte váš systém s existujícími systémy, navrhněte další možné rozšíření do budoucna.

Kategorie: Web

Licenční smlouva

Licenční smlouva je uložena v archívu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Tato bakalářská práce obsahuje popis návrhu a implementace informačního systému pro horolezecký klub Bezuchov. Informační systém eviduje informace o různých aktivitách v rámci horolezeckého klubu. Jedná se o naplánované akce, horolezecké úspěchy atd. Dále systém shromažďuje informace o členech, jejich registrace na akce a komentáře ke článkům. Uživatel taktéž může vkládat obrázky do fotogalerií a psát články k horolezeckému tématu. Přístup k aplikaci je hierarchický podle pravomocí jednotlivých uživatelů autentizovaných heslem. Aplikace je vytvořena pro efektní a přehlednou správu výše uvedených informací. Součástí systému je také algoritmus pro umělou inteligenci, která slouží k vytváření kalendáře akce.

Klíčová slova

Informační systém, databáze, entitní množina, entita, atribut, index, PHP, MySQL, administrátor, uživatelská oprávnění, XHTML, CSS, JavaScript, umělá inteligence

Abstract

This Bachelor's thesis contains description of a design and implementation of information system for mountain club Bezuchov. The information system audits informations about activities in the mountain club. It contains planned actions, accomplishments in mountains etc. The system accumalates informations about members, their registrations on actions and commnets on articles. User can insert pictures into the photogallery and he can also write an article about mountain theme. Access to this system is hierarchical, based on competence of each user. The system is created for effective and synoptical management of information presented above. System also contains artificial intelligence, which create schedule of actions.

Keywords

Information system, database, entity, attribute, index, PHP, MySQL, administrator, user competence, XHTML, CSS, JavaScript, artificial intelligence

Citace

Jakub Kadlas: Pokročilá webová aplikace

horolezeckého klubu, bakalářská práce, Brno, FIT VUT v Brně, 2007

Pokročilá webová aplikace
horolezeckého klubu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Romana Lukáše.

.....
Jakub Kadlas
9. května 2007

Poděkování

Děkuji Ing. Romanovi Lukášovi za vedení a pomoc při psaní této bakalářské práce.

Obsah

Kapitola 1

Úvod

Informační systém je chápán jako model konkrétního systému reálného světa, slouží k uchovávání a zpracovávání informací a dat. Zpracovávaná data a informace reálného světa tvoří konceptuální model. Tento systém získává informace prostřednictvím jeho uživatelů, kteří je do systému vkládají, upravují a provádějí s nimi různé operace. Z tohoto hlediska je informační systém definován jako tzv. otevřený systém, protože je zpětnovazebně propojen se svým uživatelem. Hlavní úlohou informačního systému je co nejvíce zautomatizovat práci se zdroji a tedy zefektivnit a usnadnit činnost člověku.

Informační systém pro horolezecký klub Bezuchov slouží k uchovávání a správě informací o členech klubu, jejich článcích, fotografií a úspěších. Taktéž slouží pro uchování a zobrazení naplánovaných akcí, na něž se může člen klubu přihlásit. Dalšími vlastnostmi jsou zobrazení plánu činnosti na aktuální rok a zobrazení metodické příručky. Tato aplikace se má snažit ulehčit správu jednotlivých příspěvků, ať už jsou jakékoliv, tak, aby i člověk, který není počítačově znalý, mohl publikovat své dojmy a podělit se o ně s ostatními členy. Taktéž slouží jako informační bod pro svolávání schůzí a dalších akcí, jelikož členové tohoto klubu nepochází z jednoho města a jiné dorozumívání v takovém počtu lidí by bylo příliš složité.

V následujících kapitolách této práce je popsán celý informační systém jak z hlediska jeho návrhu, tak samotné implementace podle konkrétního zadání.

V kapitole č. 2 jsou popsány přesné požadavky na informační systém, které je potřeba striktně dodržet, aby byl systém plně odpovídající účelům jeho vytvoření a aby co nejvíce ulehčil a zefektivnil používání členům.

V kapitole č. 3 jsou probrány technologie vhodné k tvorbě internetových aplikací. Specifikace jejich výhod a nevýhod. Využití v tomto projektu a vhodnost.

V kapitole č. 4 je podrobně popsán celkový návrh a implementace databáze informačního systému. Jedná se zde o řešení a návrh konceptuálního modelu relační databáze a jeho převedení na reálné databázové tabulky. Každý logický celek celého procesu je popsán a uvozen svou podkapitolou.

V kapitole č. 5, zaměřené na umělou inteligenci, je popis a charakteristika vyhledávacích algoritmů ve stavovém prostoru. Také je zde popsán mnou vybraný algoritmus, pro účel vytváření plánu akce.

V kapitole č. 6 je popis návrhu a implementace samotné webové aplikace, v závislosti na databázi a na požadavcích, které jsou definovány pro činnost systému.

V předposlední kapitole je celkové zhodnocení výsledku. Tedy pohled na vytvořený informační systém z hlediska kvality, která je definována požadavky na informační systém a z hlediska jeho uživatelského rozhraní.

V závěru tohoto dokumentu je popsána literatura, ze které bylo čerpáno.

Kapitola 2

Požadavky na informační systém

V požadavcích na vytvoření tohoto systému byla požadována jednoduchost zobrazení, kvůli členům, jejichž přístup k internetu je stále přes modem a velké grafické objekty by nebyly příliš vítané. Dále bylo požadováno zobrazení v prohlížečích Internet Explorer 6.0 a vyšší a také v prohlížeči Mozzila/Firefox. Z pohledu obsahového je zapotřebí implementovat přidávání informací o plánovaných akcích, na které je potřeba přihlašování. Taktéž uchování informací o plánech činnosti klubu na aktuální rok. Mezi příspěvkové prvky bylo nutné přidat zadávání článků od uživatelů s možností základní editace a povolení HTML značek. Informační systém dále obsahuje několik statických stránek, které nejsou generovány kvůli její neměnnosti a zbytečného nezatěžování databázového serveru.

2.1 Informace o horolezeckém klubu

Organizace pro kterou je tento informační sytém vytvářen se jmenuje HOROLEZECKÝ KLUB BEZUCHOV, o. s. (dále jen HK Bezuchov). Sdružuje zájemce o horolezeckou činnost a související sportovní aktivity bez rozdílu národnosti, státní příslušnosti, rasy a přesvědčení. Posláním HK Bezuchov je rozvíjet zájmy a chránit práva svých členů tak, aby se mohli svobodně věnovat horolezectví, skalnímu lezení a dalším sportovním aktivitám. Zároveň je jeho posláním aktivně se podílet na ochraně přírody v místech, kde jsou horolezecké aktivity provozovány.

K dnešnímu dnu tento klub sdružuje přibližně 30 členů, což ho řadí mezi středně velké horolezecké kluby.

2.2 Databáze

2.2.1 Současný stav

V současné době jsou všechny informace o členech uchovávány v papírové podobě. Nevýhoda této metody je určitě jasná. Pomineme-li náchylnost na přenos jednotlivých záznamů, je hlavním nedostatkem neaktuálnost. Úprava papírové databáze je velice zdlouhavá a při častějších změnách téměř nemožná. Taktéž orientace v záznamech každého člena je velice obtížná, vezmeme-li v úvahu že každý člen má nejméně jednu adresu na elektronickou poštu a nejméně jedno telefonní číslo. Tento způsob uložení je také nevhodný pro poskytování kontaktních údajů dalším členům.

2.2.2 Záznam člena

Ve výše uvedené podkapitole byly vysvětleny důvody pro opuštění stávajícího systému ukládání záznamů členů klubu.

Při tvorbě nové databáze na internetovém serveru bylo potřeba určit požadavky na záznamy, které je potřeba o členech schraňovat. Jako nepostradatelný údaj je určitě jméno a příjmení člena. Bez tohoto údaje by veškeré informace byly pro většinu lidí nepochopitelné.

Kvůli zjišťování kontaktů je potřeba také zaznamenávat adresu bydliště a taktéž poštovní směrovací číslo. Protože žijeme v době moderních informačních technologií, je také vhodné uchovávat elektronické adresy uživatelů. U této položky se předpokládá, že uživatel může mít více elektronických adres. Stejné předpoklady platí pro telefonní číslo, ať už pro mobilní, nebo pevnou linku. V rámci vnitřní administrativy klubu je také nutné ukládat informace o platbách poplatků a případné vydání členské průkazky.

2.3 Informační systém

Tento informační systém je dělen na několik sekcí, do nichž je přístup jen podle oprávnění.

První část slouží ke zobrazení informací bez jakékoliv editační možnosti. Je určena široké veřejnosti a k jejímu přístupu není potřeba heslo. Slouží k seznámení se s HK Bezuchov. V této části je zamezeno přihlašování se na akce.

Druhou část tvoří editační stránky a stránky určené jen členům HK Bezuchov. K přístupu na tyto stránky je potřeba přihlášení se pomocí hesla a loginu. Je zde možné editovat vlastní účet a upravovat příspěvky. Samozřejmostí je přístup i na stránky z předchozí části.

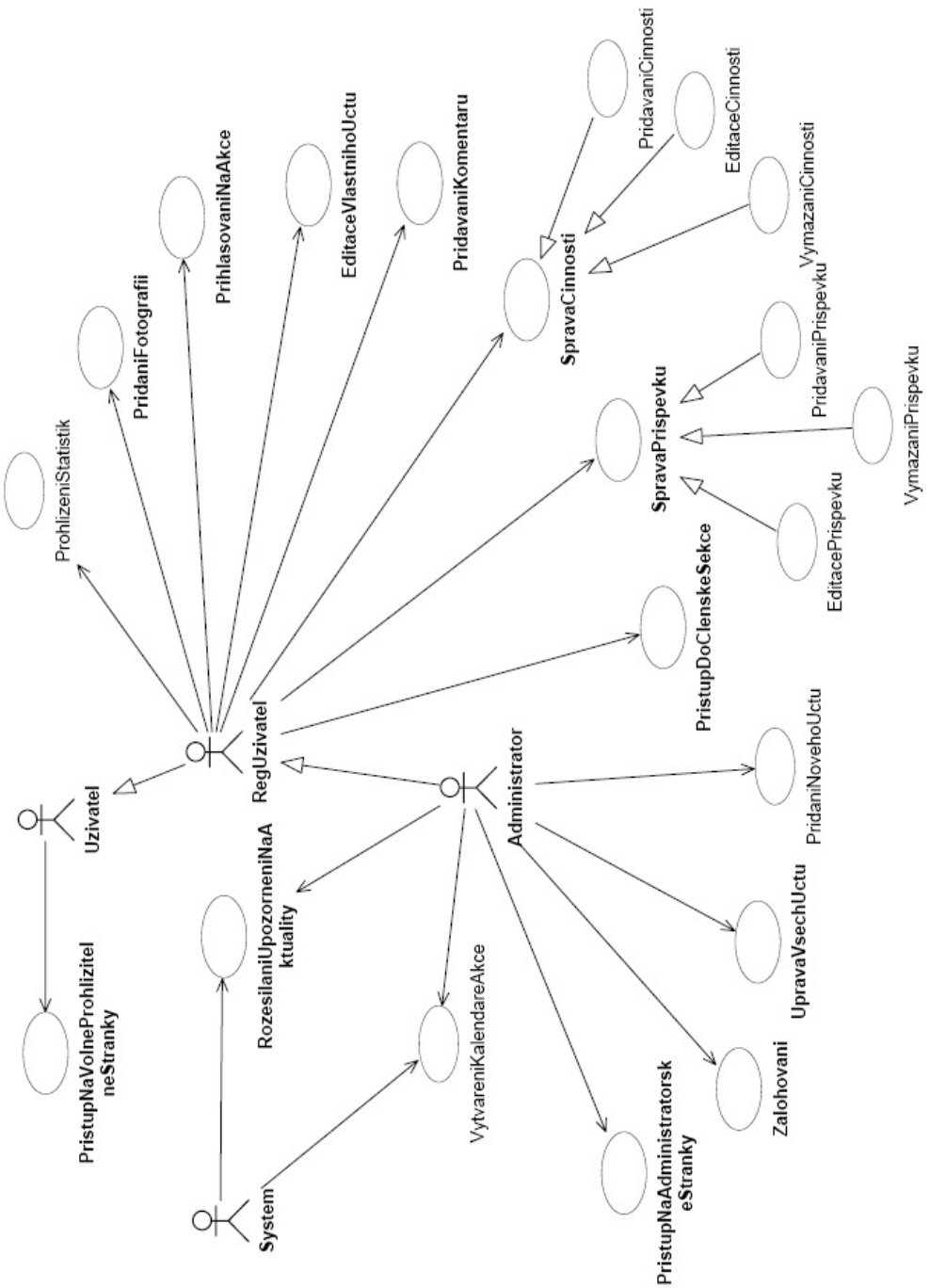
Třetí částí jsou administrátorské stránky. Na tyto stránky je potřeba mít administrátorské oprávnění. V současné době je zde možnost editace všech účtů. Jejich úpravy plateb, stavu průkazek, kontaktů a dokonce i hesel. To je z důvodu zapomnětlivosti některých členů.

2.4 Obecné požadavky

Mezi obecné požadavky patří potřeba tvorby fotogalerií, ať už přímo z akcí klubu, nebo i mimo klub, kdy uživatel může založit fotogalerii a přidávat do ní fotografie. Popřípadě přidávat fotografie do již vytvořených fotogalerií. Samozřejmostí je také možnost popisku.

Dalším požadavkem je vkládání článků. Tyto články musí obsahovat autora, nadpis a vlastní text článku. Také možnost zvýrazňování textů by přispělo k přehlednosti. Tyto požadavky platí také pro vkládání úspěchů klubu.

Také je nutné naimplementovat rozhraní pro přidávání plánu akcí, včetně možnosti přihlašování se na ně, její časové omezení pro přihlášení a zobrazení data konání. S touto funkcí systému také souvisí implementace plánu činnosti na aktuální kalendářní rok. Zde je také nutnost zadávání datumů. Je potřeba zadávat povinné datum začátku akce a volitelné datum konce akce. Také se zde musí objevit seznam odpovědných lidí za danou akci.



Obrázek 2.1: UC diagram kompletní

Kapitola 3

Technologie

V této kapitole budou vysvětleny a probrány technologie na tvorbu internetových stránek podle současného stavu(tj. ke stavu publikování této bakalářské práce).

Hlavní zaměření je na značkovací jazyk *XHTML*, dále na skriptovací programovací jazyk *php* a v poslední řadě na objektově orientovaný skriptovací jazyk *JavaScript*. Vycházel jsem zde z literatury [1] a [4].

3.1 jazyk XHTML

Encyklopedie Wikipedia [10] definuje jazyk *XHTML* (zkratka anglického extensible hypertext markup language - rozšiřitelný značkovací jazyk pro hypertext) jako značkovací jazyk pro tvorbu hypertextových dokumentů v prostředí WWW vyvinutý konsorciem W3C[1]. Jedná se o následníka jazyka *HTML* (3.1.2), jehož vývoj byl ukončen a na rozdíl od svého předchůdce se jedná o aplikaci *XML* (3.1.1).

3.1.1 XML

Než se zaměřím na *XHTML*, chtěl bych věnovat pár řádek jazyku *XML*. Jazyk *XML* (Extensible Markup Language - rozšiřitelný značkovací jazyk) je velmi obecný jazyk pro vytváření dokumentů obsahujících alespoň částečně strukturovaná data.

Formát *XML* je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Jazyk umožňuje popsat jeho strukturu z hlediska věcného obsahu jednotlivých částí, nezabývá se sám o sobě vzhledem nebo jeho částí. Prezentace (vzhled) se potom definuje připojeným stylem. Další možností je pomocí různých stylů provést transformaci do jiného typu dokumentu, nebo do jiné struktury *XML*.

Jazyk *XML* nemá žádné předdefinované značky (tagy, názvy jednotlivých elementů) a také jeho syntaxe je podstatně přísnější, než *HTML*. Pomocí XML značek vyznačujeme v dokumentu význam jednotlivých částí textu. Ty tak obsahují více informací, než kdyby se používalo značkovaní zaměřené na prezentaci (vzhled) - definice písma, odsazení a podobně. *XML* dokumenty jsou informačně bohatší. To lze samozřejmě s výhodou využít v mnoha oblastech. Největší přínos bude samozřejmě pro prohledávání, kdy můžeme určit i jaký význam má mít hledaný text.

[1] World Wide Web Consortium (W3C) - mezinárodní konsorcium vyvíjející webové standardy pro World Wide Web.

3.1.2 HTML

Jazyk *HTML* (HyperText Markup Language - značkovací jazyk pro hypertext) je mrtvý předchůdce jazyka *XHTML*. Dále je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci stránek na Internetu.

Jazyk je podmnožinou dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML [8] (Standard Generalized Markup Language). Vývoj *HTML* byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka.

Jazyk *HTML* je od verze 2.0 aplikací *SGML*. Je charakterizován množinou značek a jejich atributů pro danou verzi definovaných. Mezi značky se uzavírají části textu dokumentu a tím se určuje význam (sémantika) obsaženého textu. Názvy jednotlivých značek se uzavírají mezi úhlové závorky ("<" a ">"). Část dokumentu uzavřená mezi značkami tvoří tzv. element (prvek) dokumentu. Součástí obsahu elementu mohou být další vnořené elementy. Atributy jsou doplňující informace, které upřesňují vlastnosti elementu.

Značky (také nazývané tagy) jsou obvykle párové. Rozlišujeme počáteční a koncové značky. Koncová značka má před názvem značky znak lomítko.

Poslední a konečná revize jazyka *HTML* **4.01** byla vydána 24. prosince 1999. Tato verze opravuje některé chyby předchozí verze a přidává některé nové tagy. Je to poslední verze *HTML*.

3.1.3 XHTML - pokračování

XHTML 1.0 je prvním typem dokumentu rodiny *XHTML*. Jde o reformulaci tři typů dokumentů *HTML* 4 jako aplikací *XML* 1.0. Je zamýšlen jako jazyk pro definici obsahu, který je na jednu stranu ve shodě s *XML* a v případě dodržení některých jednoduchých směrnic je schopen provozu s uživatelskými agenty podporujícími *HTML* 4.

XHTML 1.0 má oproti *HTML* tyto výhody:

- Dokumenty *XHTML* jsou shodné s *XML*. Jako takové jsou snadno zobrazitelné, upravovatelné a validovatelné standardními *XML* nástroji.
- *XHTML* dokumenty mohou být napsány tak, že fungují stejně nebo ještě lépe než dříve s existujícími uživatelskými agenty s podporou *HTML* 4 a taktéž v nových, *XHTML* 1.0 podporujících, uživatelských agentech.
- Dokumenty *XHTML* mohou využívat aplikací (např. skriptů, appletů), který pracují s *HTML* objektovým modelem dokumentu nebo *XML* objektovým modelem dokumentu *DOM*. [2]
- I postupem času budou dokumenty shodné s *XHTML* 1.0 pravděpodobně schopné spolupráce s různými *XHTML* prostředími.

V současné době se pracuje na verzi *XHTML* 2.0

XHTML je ideální jazyk na vytváření statických internetových stránek, které není nutné dynamicky měnit. V kombinaci s *php* a *JavaScriptem* tvoří většinu internetových aplikací.

[2] *DOM* je objektově orientovaná reprezentace *XML* nebo *HTML* dokumentu.

3.2 jazyk PHP

K definici jazyka *PHP* využiji opět Wikipedii. [6] Ta popisuje *PHP* (*PHP*: Hypertext Preprocessor, "*PHP*: Hypertextový preprocesor", původně Personal Home Page) jako skriptovací programovací jazyk určený především pro programování dynamických internetových stránek. Nejčastěji se začleňuje přímo do struktury jazyka *HTML*, *XHTML* či *WML*, což je velmi výhodné pro tvorbu webových aplikací. *PHP* lze ovšem také použít i k tvorbě konzolových a desktopových aplikací.

3.2.1 Historie

PHP je skriptovací jazyk pro tvorbu dynamického webu a jeho počátky spadají do roku 1994. Tehdy se pan Rasmus Lerdorf rozhodl vytvořit jednoduchý systém pro počítání přístupu ke svým stránkám, který byl napsán v *PERL*u. Za nějakou dobu byl systém přepsán do jazyka *C*, protože perlovský kód zatěžoval server. Sada těchto skriptů byla ještě později téhož roku vydána pod názvem "Personal Home Page Tools", zkráceně PHP.

V polovině roku 1995 se systém *PHP* spojil s jiným programem stejného autora, a to sice s nástrojem "Form Interpreter" neboli zkráceně FI. Tak vzniklo PHP/FI 2.0, systém, který si postupně získal celosvětovou proslulost a byl velmi rozšířen.

Koncem roku 1998 byla již k dispozici verze *PHP* 3.0, která byla mnohem rychlejší a byla k dispozici rovněž pod operačními systémy Windows. Počet webů které používaly *PHP* se zvyšoval, až dosáhl cca 150 000. *PHP* verze 4 přidávají do jazyka mnoho nových funkcí a rovněž přinášejí přepracované a podstatně rychlejší jádro Zend.

PHP verze 5 obsahuje jádro Zend II, kompletně přepsanou podpora XML, nové objektové rozhraní pro práci s databází MySQL a mnoho dalšího. Tento systém je stále vybavován novými technologiemi a je aktivně vyvíjen.

Podle údajů z dubna 2004 běží *PHP* na více než 15 000 000 doménách a je to bezkonkurenčně nejčastěji používaný modul webového serveru Apache.

3.2.2 Princip jazyka

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

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

S verzí *PHP* 5 se výrazně zlepšil přístup k objektově orientovanému programování podobný *Javě*.

3.2.3 Vlastnosti jazyka PHP

- Jazyk *PHP* je dynamicky typový, tzn. že datový typ proměnné se určí v okamžiku přiřazení hodnoty.
- Díky tomu má *PHP* dva typy porovnání, '==' stejný jako v *C*, a '===' který platí jen když jsou oba dva výrazy stejného typu.
- Pole jsou heterogenní, mohou tedy obsahovat jakékoli údaje stejně tak jako jejich indexy.
- Řetězce lze uzavírat jak do uvozovek (obsah je parsován), tak do apostrofů (obsah není parsován).

3.3 jazyk JavaScript

JavaScript je na Wikipedii [5] definován jako multiplatformní, objektově orientovaný skriptovací jazyk. Nyní se zpravidla používá jako interpretovaný programovací jazyk pro WWW stránky, vkládaný přímo do *HTML* kódu stránky. Jsou jím obvykle ovládány různé interaktivní prvky *GUI* (tlačítka, textová políčka) nebo tvořeny animace a efekty obrázků. Bohužel se na *JavaScript* nelze vždy spoléhat, protože může být u uživatele jeho podpora deaktivována.

3.3.1 Historie

Jazyk *JavaScript* vymyslel v roce 1995 Brendan Eich ve firmě Netscape. Byl určen především pro použití v prohlížeči Netscape Navigator 2.0, ale využíván byl i v serverových produktech firmy. S dalšími verzemi prohlížeče se jazyk vyvíjel a časem bylo jeho jádro standardizováno organizací ECMA. Dnes *JavaScript* existuje v mnoha implementacích.

3.3.2 Princip jazyka

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

3.3.3 Vlastnosti jazyka JavaScript

Obsahuje literály pro regulární výrazy, lambda funkce, dědičnost pomocí prototypů a další. Díky jeho jednoduchosti se ho lze snadno naučit a zároveň je díky zmiňovaným funkcím velmi efektivním pracovním nástrojem pro pokročilé programátory.

3.4 jazyk CSS

CSS (Cascading Style Sheets - tabulky kaskádových stylů) je jazyk pro popis způsobu zobrazení stránek napsaných v jazycích *HTML*, *XHTML* nebo *XML*. Jeho hlavním smyslem je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Původně to

měl umožnit už jazyk *HTML*, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak.

3.4.1 Vlastnosti jazyka CSS

- rozsáhlejší možnosti - rozsáhlejší formátovací možnosti než samotné HTML
- konzistentní styl - ve všech dokumentech mají objekty stejný vzhled
- oddělení struktury a stylu
- dynamická práce se styly
- formátování XML dokumentů
- větší kompatibilita alternativních webových prohlížečů
- kratší doba načítání stránky
- stále špatná podpora v majoritních prohlížečích typu Internet Explorer

Kapitola 4

Databáze informačního systému

Databáze je registr (papírová agenda, kartotéka) v elektronické podobě, v podobě perzistentních dat. Perzistentní data jsou data trvalá, tedy z hlediska aplikace jsou to data, která jsou uchovávána i tehdy, pokud právě aplikace samotná neběží. V tom je rozdíl od dat, se kterými pracují ostatní nedatabázové aplikace, které získávají data teprve za běhu a po skončení provádění aplikace jsou tato data ztracena.

Data uložená v databázi jsou integrovaná, což můžeme chápat jako data sjednocená s úplným nebo částečným odstraněním redundance, data sdílená, což umožňuje přístup více uživatelů. Co se týká bezpečnosti dat, tak je možné snadněji omezit přístup k datům omezením práv uživatelů, a lze také snadněji zajistit integritu (správnost z hlediska splnění omezení) dat. Jedná se o tzv. integritní omezení, která reflektují reálný svět. V této kapitole jsem vycházel z literatury [3].

4.1 Postup při návrhu databáze

Definice: Databáze je správně navržena, pokud je integrovaná, tedy zbavená nadbytečných údajů a duplicit.

Návrh databáze informačního systému se skládá z několika etap:

První etapou je vytvoření tzv. konceptuálního modelu. Tento model je vytvořen na základě znalostí získaných při specifikaci systému, což je analýza systému na základě konkrétních požadavků. Z těchto požadavků můžeme navrhnout základní konceptuální model, se kterým bude vyvíjený systém pracovat. Nejznámější a nejčastěji používanou modelovací technikou pro návrh relačních databází je *entitně-vztahové*[1] modelování, jehož výsledkem je *entitně-vztahový* diagram (ER diagram z anglického entity-relationship). Modelují se pouze perzistentní data, která budou uložena v databázi.

Druhá etapa je transformace konceptuálního modelu na tabulky relační databáze. V tomto kroku je velice důležité odstranit ze vznikajících tabulek redundanci dat. Hrozí zde, že se jistá informace opakuje nebo se potenciálně může opakovat. Ta nám zbytečně zabírá místo v databázi, ale především způsobuje složitou kontrolu správnosti dat, tedy odpovídajícího integritního omezení. Před každým vložením nebo modifikací řádku tabulky by SŘBD [2]

[1] Entita - objekt (koncept) aplikační domény světa, který je rozlišitelný od jiných objektů, o nichž potřebujeme mít informace v databázi.

[2] Systém řízení báze dat

nebo aplikace musel kontrolovat, že ve všech řádcích tabulky s vkládanou hodnotou bude i stejná hodnota redundantní.

Taktéž je nutné si stanovit jako cíl zohlednění výkonnostního hlediska tím, že nebudeme vytvářet zbytečné tabulky. Je také nutné zkontrolovat, zda atributy všech entitních množin a vztahů nabývají atomických hodnot. Pokud ne, je potřeba provést úpravu diagramu nebo pamatovat na tuto skutečnost při transformaci.

Dalším krokem je transformace entitních množin, kdy každá množina je v databázi reprezentovaná tabulkou. Následuje transformace vztahových množin. Zde záleží na kardinalitě vztahu. Pokud nejde o vztah s kardinalitou N:M, lze jej reprezentovat pouhou vazbou mezi tabulkami vytvořenou pomocí cizího klíče. Při kardinalitě N:M se neobejdeme bez další tabulky.

V některých případech je vhodné využít transformace generalizace/specializace. Zde existuje několik možných variant transformace a výběr nejvhodnější vyžaduje posouzení dalších faktorů, z nichž některé nejsou zjistitelné z ER diagramu. Zde je opět důležitým prvkem vyvarovat se redundanci a zohlednit efektivní provádění častých operací z hlediska spojování tabulek, případně velikosti záznamů.

Pravidla logického návrhu databáze jsou jasná a většinou jednoznačná. Pro daný ER diagram zpravidla vedou na jediné dobré řešení. Naproti tomu ER model v podobě ER diagram, který modeluje řešený problém, může mít řadu různých podob. Kvalita diagramu se promítá i do kvality návrhu databáze. Chyby v ER diagramu, zejména chybné určení kardinality vztahů může vést na chybný návrh databáze. A přímý logický návrh databáze bez použití konceptuálního modelu je možný pouze u menších databází. U rozsáhlých systémů takový postup vede na výsledek, který obsahuje chyby a je obtížně udržovatelný.

Třetí etapa je variantou ke druhé etapě a předpokládá, že všechny informace budou uloženy v jedné tabulce a tu normalizujeme. Normalizaci můžeme chápat jako postup návrhu databáze, jehož cílem je dostat výsledek, který nebude trpět nedostatky chybného návrhu. Vzniká otázka, jak se pozná, že tabulka není navržena dobře. K ohodnocení kvality návrhu tabulky byly zavedeny tzv. *normální formy*. V čím vyšší normální formě tabulka je, tím je její návrh kvalitnější. Normální formy definují, jaké vlastnosti tabulka musí mít s ohledem na závislosti mezi jejími atributy. Nejjednodušším typem takových závislostí jsou závislosti funkční.

4.2 Realizace databáze

Tato kapitola se zaměřuje na implementaci databáze, transformaci jejich tabulek a vzájemných vztahů mezi nimi.

Zajištění správné návaznosti tabulek tak, aby odpovídaly správnému návrhu databáze není jednoduché. K úspěšnému vytvoření je zapotřebí postupu, který je uveden v předchozí kapitole (4.1).

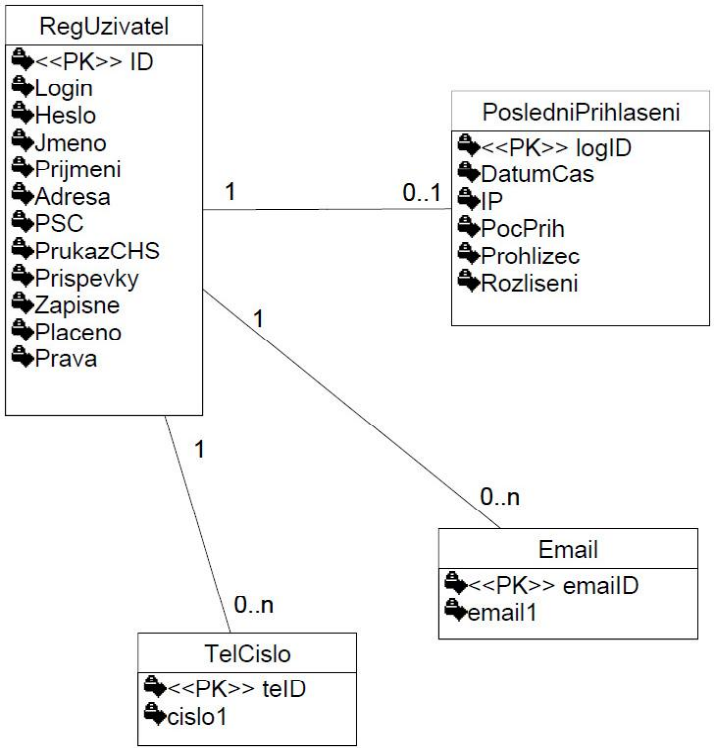
4.2.1 Uživatel

Základním stavebním kamenem této databáze je entita obsahující informace o uživatelích (v případě tohoto informačního systému se jedná o členy HK Bezuchov).

Při její tvorbě se musely zohlednit nároky na obsah informací střádaných u každého člena. Jak je vidět z obrázku(4.1), bylo nutné kromě klasických údajů typu Jméno, Příjmení, také ukládat kontaktní údaje, přihlašovací údaje a v poslední řadě stav členství v klubu HK Bezuchov.

Zdá se sice, že tyto záznamy jsou kompletní, ale opak je pravdou. Pro větší konektivitu mezi členy je nutné ukládat informace o telefonních číslech a také elektronické adresy (dále jen e-mail). Zde nastala otázka, zda tyto informace namodelovat jako část atributy entitní množiny `RegUzivatel`, nebo vytvořit novou entitu, jak pro telefonní čísla, tak pro e-maily.

Při realizaci první varianty by nastal problém s přidáváním hodnot u uživatelů, kteří mají více kontaktních údajů. Při vkládání například dvou telefonních čísel by se musel uživatel opakovat v databázi dvakrát. Tato redundance by ale byla v rozporu se správným návrhem databáze. Po zjištění takovéhoto rozporu zbývá jen druhá varianta – vytvoření samostatných entitních množin. Po analýze možných vztahů jsem nastavil kardinalitu vztahu jako 1:N (viz obrázek 4.1). Tento vztah lze jednoduše přetransformovat do vzniklých tabulek databáze.



Obrázek 4.1: Výřez z ER diagramu: Entita Registrovaný uživatel ve vztahu k Email, Telefonní číslo a Poslednímu přihlášení

Jelikož je systém hierarchický podle přístupových práv a hesel, je nutné ukládat informace o přihlášení kvůli kontrole přístupu na stránky a jejich následné odhlašování po jisté

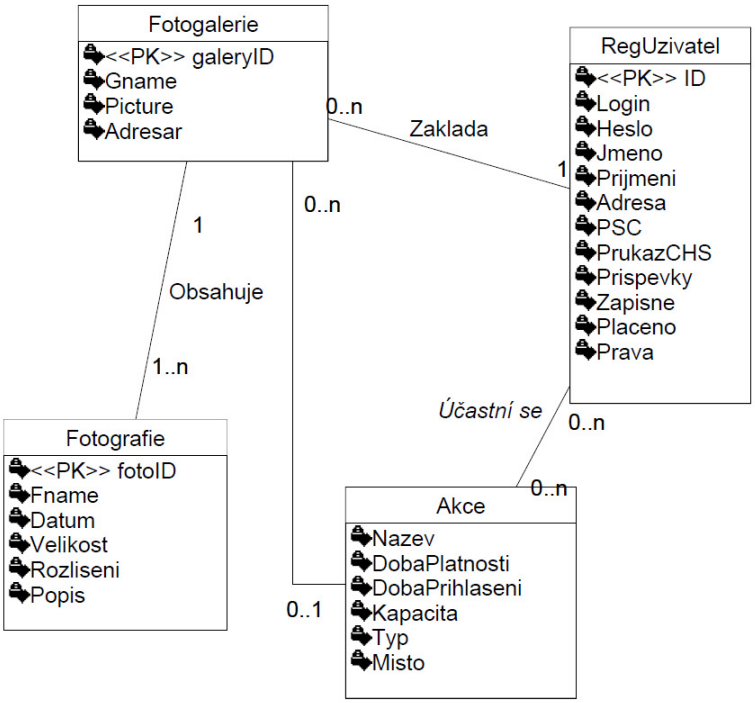
době dané automatickým odhlašováním.

Zde byla opět možnost tyto informace vložit jako atributy do entity `RegUzivatel`. Z důvodu častého přepisu těchto informací by se ale zpomaloval databázový server. Protože někteří uživatelé, co zde sice figurují jako registrovaní kvůli kontaktním adresám, nemusí mít přístup k internetu, zabíraly by jejich informace o přihlášení zbytečně místo v databázi. Z tohoto důvodu se tato varianta s jednou entitou netváří jako nejlepší řešení. Proto byla vytvořena entita `PosledniPrihlaseni` (viz obrázek 4.1). Jsou zde také přidány informace pro statistiku.

4.2.2 Fotogalerie

Při tvorbě entity `Fotogalerie` bylo nutné zajistit návaznost na entitu `RegUzivatel` z důvodu zjišťování autora. Vztah mezi těmito entitami je 1:N, kdy jeden uživatel může vytvořit více fotogalerií (viz obrázek 4.2). Taktéž je tato entita spjata s entitou `Akce` pro zjištění, zda daná fotogalerie navazuje na akci klubu. Vztah je 0..1:0..N z důvodu možnosti vytvořit fotogalerii i bez navazující akce.

Na entitu `Fotogalerie` navazuje entita `Fotografie`. Zde jsou zapsány informace o fotografii. K jaké fotogalerii patří, její informace a popis. Vztah mezi `Fotogalerie` a `Fotografie` je 1:N (viz obrázek 4.2). Předpoklad, že by fotografie byla obsažena ve více fotogaleriích není podporován.



Obrázek 4.2: Výřez z ER diagramu: Entita Fotogalerie ve vztahu k Registrovaný uživatel, Fotografie, Akce

4.2.3 Činnosti

Entita `Cinnosti` obsahuje seznam činností naplánovaných v rámci klubu HK Bezuchov. Z důvodu potřeby záznamu informace o uživatelích, kteří jsou za danou akci zodpovědní, bylo nutné spojit tuto entitu s entitou `RegUzivatel` (viz obrázek 4.3).

Jelikož by vztah mezi těmito entitami obsahoval kardinalitu N:M, bylo ho nutné při implementaci do databáze přeměnit na tzv. vazební entitní množinu. Postup odstranění této kardinality je uveden dále v kapitole 4.2.6.

4.2.4 Příspěvky

Tato sekce je věnována entitám na ukládání informací o příspěvcích. Jednotlivé části jsou odděleny do podkapitol, ve kterých jsou specifikovány detaily.

Každý příspěvek má svého autora, který je dán záznamem z entity `RegUzivatel`. Uživatel může vložit neomezené (pokud bude volné místo pro ukládání na serveru) množství příspěvků. Každý příspěvek patří do jisté sekce. Ta je reprezentována jako další entita. Vztah mezi entitami `Sekce` a `Prispevek` je 1:N. Detail vztahu viz obrázek (4.3)

Článek

Entita `Clanek` navazující na entitu `Prispevek` udává obsah a nadpis vloženého příspěvku. Jde o specifikaci entity `Prispevek`. (viz obrázek 4.3)

Komentář

Entita `Komentar` navazující na entitu `Prispevek` obsahuje informace o uloženém komentáři a jeho příslušnost k danému článku. Jelikož `Komentar` má vazbu na sebe sama, lze navazovat komentáře na již existující komentář. Jde o reflexivní vztah mezi entitami. (viz obrázek 4.3)

Akce

Entita `Akce` navazující na entitu `Prispevek` obsahuje informace o zadávaných akcích. Vztah mezi entitami `Akce` a `RegUzivatel` je 0..N:0..M. Proto při její implementaci do databáze musela být vytvořena pomocná tabulka `Ucastnik`, která obsahuje uživatele přihlášené na tuto akci. Tato entita je generalizací dvou entit `Schuze` a `VystupNaHoru`. (viz obrázek 4.3)

4.2.5 Tabulky pro umělou inteligenci

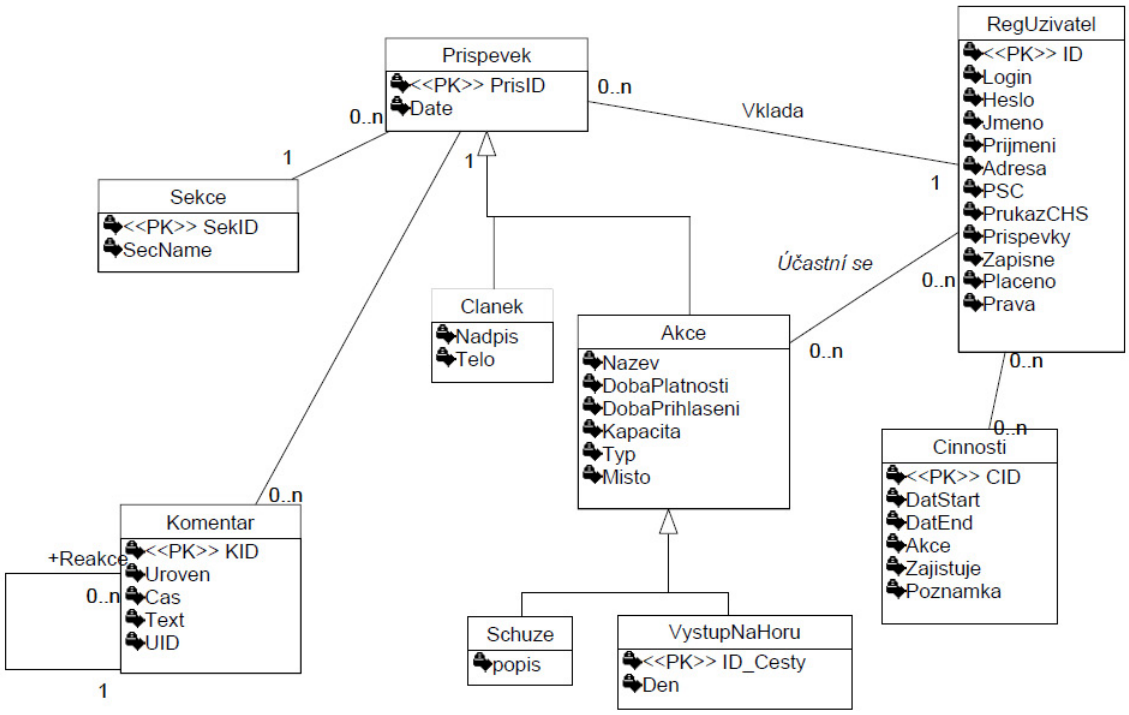
V této sekci se zaměřím na část tabulek v databázi, které jsou věnovány algoritmu pro umělou inteligenci. Viz obrázek (4.4).

Hory

Entita `Hory` obsahuje informace o horách, jejich název a výšku.

Výstup na horu

Entita `VystupNaHoru`, která nám udává den výstupu na horu, také spojuje entity `hory` a `Pruvodce` pro zjišťování výstupů.



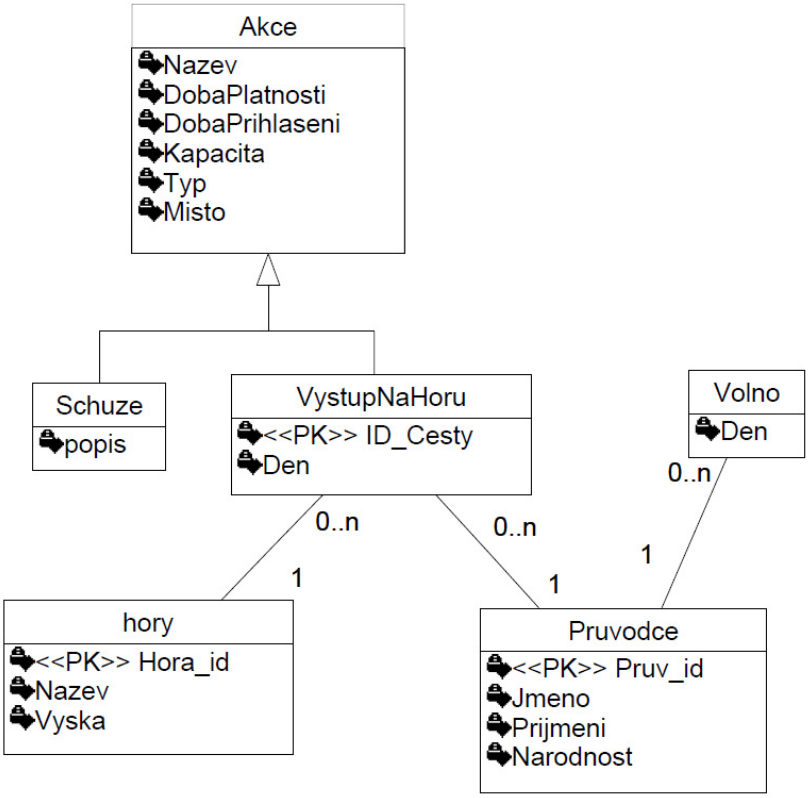
Obrázek 4.3: Výřez z ER diagramu: Entita Článek a navazující entity

Průvodce

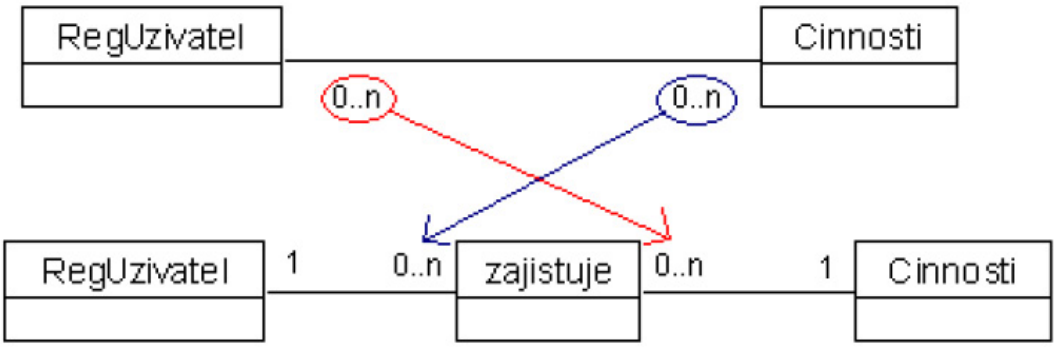
Tabulka `Pruvodce` obsahuje popis průvodce a jako závislá entita je k ní připojena entita `volno` udávající volné dny, kdy nemůže jít průvodce na žádný výlet.

4.2.6 Postup při odstranění vazby typu N:M

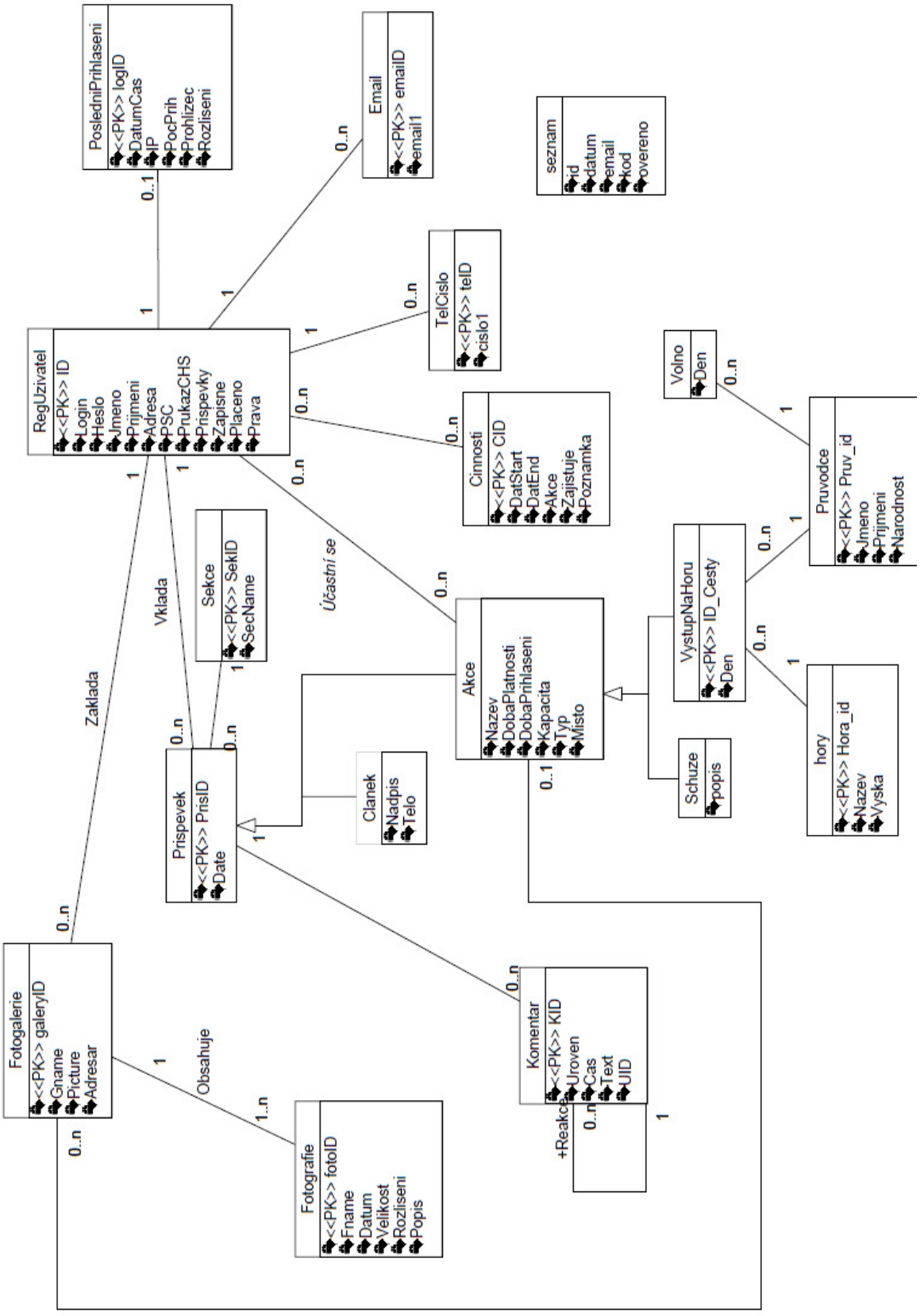
Postup pro odstranění vazby N:M si ukážeme na dvou entitách `RegUzivatel` a `Cinnosti`. Nejprve si vytvoříme vazební entitní množinu `zajistuje`. Ta bude mít na straně `Cinnosti` kardinalitu 1 a na straně `zajistuje` hodnotu, které byla původně u `RegUzivatel`. Naopak při vytvoření vztahu mezi `zajistuje` a `RegUzivatel` bude na straně `RegUzivatel` kardinalita 1 a na straně `zajistuje` bude původní kardinalita ze strany `Cinnosti`, čili 0..M. Ilustrace úpravy je uvedena na obrázku (4.5).



Obrázek 4.4: Entity pro umělou inteligenci



Obrázek 4.5: Odstranění kardinality N:M



Obrázek 4.6: ER diagram kompletní

Kapitola 5

Umělá inteligence

Cílem této kapitoly je stručné obeznámení s problematikou umělé inteligence, přičemž hlavní pozornost věnuji těm oblastem, které nejvíce ovlivnily moji práci. Použitá literatura [11], [2].

Uznávaná nejobecnější definice, již vytvořil Marvin Minsky, definuje Umělou inteligenci (Artificial Intelligence), zkráceně UI (AI) jako vědu, která se zabývá tím, jak přinutit stroje projevovat se takovým chováním, které by v případě člověka vykazovalo potřebu inteligence.[9]

5.1 Vývoj UI

Postupně s rozvojem techniky si lidé kladli otázku, jestli je možné vytvořit systém, který by dosahoval takové chování, které je u živých organismů označováno jako inteligentní. Otázku "Mohou stroje myslet ?" řešili již v 17. století významní filozofové jako Descartes, Pascal, Hobbes a další.

Později, až ve druhé polovině 20. století, na základě výsledků matematické logiky a teorie algoritmů, spolu s prudkým rozvojem výpočetní techniky, jsou navrhovány a experimentálně ověřovány metody, postupy a algoritmy napodobující inteligentní chování. Vzniká tak mladá vědní disciplína, jejímž cílem je napodobit inteligentní lidské chování, nazvaná umělá inteligence.

V oblasti umělé inteligence se objevuje několik přístupů. Jeden z úspěšně aplikovaných přístupů při realizaci UI, vychází z hypotézy formulované Newellem a Simmonem v roce 1976, že požadovaného efektu lze dosáhnou vhodnou manipulací se znaky.

Z biologie, resp. neurologie, vychází další směr tzv. konekcionismus, který využívá vysoce paralelní výpočetní techniky realizované sítí jednoduchých počítačových elementů vzájemně si předávajících zprávy (např. neuronové sítě).

5.2 Úvod do UI

Klasická UI úloha je definovaná počátečním stavem, množinou cílových stavů a množinou operátorů, které umožňují stavy této úlohy měnit. Řešením úlohy je nalezení posloupnosti operátorů, jejichž postupnou aplikací se dostaneme z počátečního stavu do některého z množiny cílových stavů. Často je nutné výchozí úlohu rozdělit na jednodušší podúlohy a ty potom řešit relativně samostatně.

Metody řešení úloh nám nabízejí postupy, kterými lze cílové stavy, resp. posloupnosti operátorů vedoucí k cílovým stavům, nalézat a představují tak prostředky nepostradatelné ve všech aplikačních oblastech UI.

Metody řešení úloh lze rozdělit do čtyř skupin a to na:

- Metody založené na prohledávání stavového prostoru
- Metody řešení úloh s omezujícími podmínkami
- Metody založené na rozkladu úloh/problémů na podproblémy
- Metody hraní her

Stavový prostor si můžeme představit jako orientovaný graf/strom, jehož uzly představují jednotlivé stavy úlohy a jehož hrany reprezentují přechody mezi těmito stavy způsobené definovanými operátory. Cesta z počátečního stavu do některého cílového stavu je zřejmě řešením úlohy. V řadě úloh je nutné minimalizovat cenu této cesty, která se získá součtem cen jednotlivých přechodů. U některých úloh naopak cesta není vůbec důležitá a důležitý je pouze cílový stav.

Při výběru správné metody se můžeme orientovat podle hodnotících kritérií a na jejich základě zvolit pro nás nejvýhodnější metodu.

Metody řešení úloh se hodnotí podle čtyř kritérií, kterými jsou:

- Úplnost (nalezne metoda řešení, pokud toto existuje?).
- Časová náročnost.
- Paměťová náročnost.
- Optimálnost (nalezne metoda nejlepší řešení?).

5.3 Metody založené na prohledávání stavového prostoru

Metody prohledávání stavového prostoru se dělí na dvě základní skupiny:

- Neinformované/slepé
- Informované

Neinformované metody:

- Metoda prohledávání do šířky (BFS - Breadth First Search)
- Metoda stejných cen (UCS - Uniform Cost Search)
- Metoda prohledávání do hloubky (DFS - Depth First Search)
- Metoda omezeného prohledávání do hloubky (DLS - Depth Limited Search)
- Metoda postupného zanořování do hloubky (IDS - Iterative deepening DFS)
- Metoda zpětného navracení (Backtracking)
- Metoda obousměrného prohledávání (BS - Bidirectional BFS Search)

Informované metody (Jejich názvy se obvykle nepřekládají):
Těmito metodami se nebudu dále zabývat, pro zjištění dalších informací doporučuji nahlédnout do citované literatury.
Metody Best First Search:

- Metoda Greedy search
- Metoda A* search

Metody lokálního prohledávání (Local search)

- Metoda Hill-climbing
- Metoda simulovaného žíhání (Simulated annealing)
- Metody založené na genetických algoritmech (Genetic algorithms)

5.3.1 Neinformované metody

Neinformované metody nemají k dispozici žádnou informaci o cílovém stavu a nemají ani žádné prostředky, jak aktuální stavy hodnotit. Protože v naimplementované umělé inteligenci využívám postupy z této kapitoly, je dobré se seznámit s některými variantami.

Metoda slepého prohledávání do šířky (BFS)

Při tomto způsobu prohledávání máme jistotu, že vždy nalezneme koncový stav. Musíme ale projít značně velký počet uzlů (procházíme všechny uzly, které mají hloubku menší, než je hloubka koncového uzlu). Každý uzel v grafu navštívíme nejvýše jednou.

Je-li počet bezprostředních následníků každého uzlu konečný, pak algoritmus BFS je úplný a optimální. Časová i paměťová náročnost algoritmu BFS je však exponenciální.

I když je algoritmus BFS velmi jednoduchý, je pro svou časovou a paměťovou náročnost pro řešení složitějších úloh prakticky nepoužitelný. Přesto je algoritmus BFS považován za základní algoritmus. Navíc jeho praktické použití na jednoduchých úlohách ukazuje na jeden závažný a obecný problém, kterým je opakované generování již expandovaných uzlů.

Metoda stejných cen (UCS)

Metoda vychází z předpokladu ohodnocení uzlů cenami, které odpovídají vynaloženým nákladům na vygenerování těchto uzlů. K expanzi se vždy vybírá uzel s minimálním ohodnocením. Jsou-li ohodnocení uzlů rovna jejich hloubce, je metoda stejných cen shodná s metodou prohledávání do šířky.

Pro hodnocení metody UCS platí stejné závěry jako pro metodu BFS. Je-li počet bezprostředních následníků každého uzlu konečný, pak algoritmus UCS je úplný a optimální, jeho časová i paměťová náročnost je exponenciální.

Metoda slepého prohledávání do hloubky (DFS)

Tento způsob prohledávání může vést k cíli mnohem rychleji, než prohledávání do šířky (zvláště když se vydáme správným směrem), ale nemáme zaručeno (v případě nekonečné větve), že vždy nalezneme koncový stav. Na rozdíl od prohledávání do šířky můžeme některými uzly procházet vícekrát, neboť se často musíme navracet.

Algoritmus DFS není, na rozdíl od výše popsaných algoritmů BFS a USC, ani úplný, ani optimální. Časová náročnost algoritmu DFS zůstává exponenciální.

Metoda omezeného prohledávání do hloubky (DLS)

Pokud řešíme úlohu, u které dokážeme odhadnout hloubku řešení, můžeme problém s opakujícím se generováním stejných stavů vyřešit omezením hloubky prohledávání. Jde tedy o upravený algoritmus DFS.

Algoritmus DLS není, stejně jako algoritmus DFS, ani úplný, ani optimální. Časová náročnost algoritmu DLS zůstává exponenciální.

Metoda postupného zanořování do hloubky (IDS)

Nelze-li stanovit hloubku řešení a nemáme-li k dispozici dostatek paměti pro použití metody BFS, můžeme se pokusit vyřešit tento problém použitím metody postupného zanořování do hloubky. Princip této metody spočívá v opakovaném použití metody DLS s postupným zvyšováním hloubky prohledávání.

Metoda postupného zanořování do hloubky se zdá být na první pohled velmi neefektivní, protože při každém dalším zanoření opakuje plně prohledávání, které již provedla při předcházející hloubce.

Metoda zpětného navracení (Backtracking)

Metoda zpětného navracení je speciální metodou prohledávání do hloubky, kdy rozšiřuje nalezené částečné řešení směrem k úplnému řešení pomocí postupného ohodnocování jednotlivých proměnných možnými hodnotami. V každém kroku je kontrolováno splnění všech podmínek mezi již ohodnocenými proměnnými. Pokud některá z podmínek nevyhovuje, program jde zpět k poslední ohodnocované proměnné, která má ještě jinou, dosud nezkoušenou hodnotu. Pokud je tedy nalezen konflikt, není již dané řešení dále rozšiřováno, proto je tato metoda efektivnější než metoda generuj a testuj.

Metoda má extrémně nízkou paměťovou náročnost, protože v paměti (tj. v zásobníku OPEN) jsou uloženy pouze uzly aktuální cesty spolu označením právě použitých operátorů. Pro časovou náročnost, úplnost a optimálnost platí stejné závěry jako pro metodu DFS: časová náročnost je exponenciální, metoda není ani optimální, ani úplná.

Algoritmus je následující:

1. Sestroj zásobník OPEN (bude obsahovat uzly určené k expanzi) a umísti do něj počáteční uzel.

2. Je-li zásobník OPEN prázdný, pak úloha nemá řešení a ukonči proto prohledávání jako neúspěšné. Jinak pokračuj.

3. Jde-li na uzel na vršku zásobníku aplikovat první/další operátor, tak tento operátor aplikuj a pokračuj bodem 4, v opačném případě odstraň testovaný uzel z vrcholu zásobníku a vrať se na bod 2.

4. Je-li nový vygenerovaný uzel, tj. uzel vzniklý aplikací operátoru na uzel na vršku zásobníku, uzlem cílovým, ukonči prohledávání jako úspěšné a vrať cestu od kořenového uzlu k uzlu cílovému (vrací se posloupnost stavů, nebo operátorů). Jinak ulož nový uzel na vršek zásobníku a vrať se na bod 2.

Metoda obousměrného prohledávání (BS)

U úloh, které používají inverzní operátory je možné prohledávat stavový prostor oběma směry metodou BFS - od počátečního k cílovému stavu a současně od cílového stavu ke stavu počátečnímu. Pokud se pak v každém směru prohledá poloviční hloubka, klesne paměťová složitost. Pro časovou složitost toto však neplatí, protože v každém kroku je nutné porovnávat všechny aktuální koncové stavy jednoho prohledávání se všemi aktuálními koncovými stavy druhého prohledávání.

5.4 Implementovaný vyhledávací algoritmus

Pro řešení této části bakalářské práce jsem si zvolil algoritmus Metoda zpětného navracení (Backtracking) 5.3.1. Tento algoritmus využívá zásobníku, jistou modifikací by mohl také pracovat na stromové struktuře. Obě tyto implementace jsou si rovnocenné.

Při hledání rozvrhu akcí pro zadané skupiny (implicitní nastavení počítá se čtyřmi skupinami), dochází k prohledání stavového prostoru a následné prokombinování průvodců tak, aby se každá skupina dostala na kopec o který má zájem a s průvodcem kterého preferuje. V případě nalezení kombinace, která splňuje požadavky, dojde k výpisu kalendáře. V opačném případě bude uživatel informován o nenalezení a vyzván k novému zadání požadavku.

Nevýhoda této implementace spočívá v nerovnoměrném rozložení využití průvodců a to tak, že jsou preferováni průvodci umístěni výše v tabulce. K rovnoměrnému rozložení by bylo potřeba zajistit ukládání dodatečných informací o počtu navštívených tras. Současná implementace je tedy výhodná v případě upřednostňování zkušenějších průvodců před méně zkušenými, kteří by v dané agentuře působili jen jako pomocná síla.

Jak už bylo zmíněno dříve, tato metoda není ani optimální, ani úplná. To znamená, že se nemusí nelézt nejlepší výsledek a také nemusí najít výsledek, i když řešení existuje. Její časová exponenciální náročnost a lineární paměťová náročnost je pro tento projekt dostačující.

5.5 Vyhodnocování podmínek

Při vyhodnocování podmínek pro vyhledávací algoritmus je možné určit dva nejčastější druhy podmínek. Jedná se o princip *succeed first* a *first-fail*.

Pokud by jsme si zvolili princip *succeed first*, snažili bychom se najít hodnotu, která by nás mohla dovést nejrychleji k cíli. Princip tohoto způsobu je postaven na předpokladu, že pokud má algoritmus na vyhodnocování interaktivní přístup, pak je jedno v jakém pořadí budeme muset projít prohledávané hodnoty. Proto zde není důvod jako první nezvolit tu, která nás nejvíce zajímá.

Druhý princip nazvaný *first-fail* je založen na nejhůře ohodnocavatelné proměnné. Jedná se o proměnnou obsahující nejvíce omezení. Pokud se nám podaří správně takovou proměnnou vybrat, zajistíme si, že dané nesprávné řešení bude co nejdříve odhaleno.

V mém případě jsem vybral princip *first-fail*. Z důvodu omezení ze strany obsazených průvodců je nejvýhodnější. U každého průvodce totiž může být záznam o obsazených dnech a možnost, zda je na danou horu požadován. Jako hlavní podmínku bude nejvhodnější zvolit, zda je na danou horu požadován. V tomto jednoduchém případě by mohla být zvolena i druhá podmínka jako hlavní a přesto by nedošlo k výrazné změně rychlosti vyhledávání.

Kapitola 6

Návrh a implementace informačního systému

V této kapitole se zaměřuji na implementaci webového rozhraní, skriptů pro práci s databází a dalších obslužných skriptů. Většina stránek v tomto informačním systému (dále jen IS) je generována přes php server. Jazyk PHP (viz 3.2) byl vybrán z důvodu jeho veliké rozšířenosti a také díky bezplatnému používání php serveru. Grafická stránka IS byla vytvořena pomocí technologie CSS (Cascading Style Sheets) (viz 3.4), která je v současnosti nejpoužívanější pro tvorbu vzhledu internetových aplikací.

Většina specifikací pro tvorbu IS byla zmíněna v kapitole 2.

6.1 Redakční systém

Redakční systém, také známý pod zkratkou CMS (content management system - systém pro správu obsahu), je označení pro software zajišťující správu dokumentů, nejčastěji webového obsahu. V dnešní době se jako CMS zpravidla chápou webové aplikace, někdy s případným doplňkovým programovým vybavením u klienta. (Definice převzata z [7])

Cílem této bakalářské práce bylo vytvoření IS, který by odpovídal alespoň jednoduchému CMS. Mezi prvky, které je nutné v tomto IS mít patří možnost vkládání fotografií, článků a komentářů. Mezi nutné požadavky také bylo zařazeno přihlašování na akce.

Veškeré snahy o vytvoření CMS jsou způsobeny snahou snížit zatížení Administrátora při vkládání výše zmíněných informací. Administrátor se stará pouze o funkčnost stránek a vkládání stránek, které nejsou závislé na uživatelích. Taktéž se stará o šablony, do kterých jsou načteny informace z databáze.

6.2 Uživatelské rozhraní

Při návrhu a tvorbě uživatelského rozhraní byl nutný předpoklad snadné orientace a přehlednosti. Toto kritérium, je ale velice subjektivní, protože každý uživatel má jiné názory. Z mého hlediska bylo nejvýhodnější vytvořit prostředí o jednom ovládacím prvku umístěném v levé části stránky a přihlašovací formulář v horním pravém rohu. Toto umístění platí za jedno ze základních a nejpoužívanějších rozložení ovládacích prvků na internetu. Některé položky menu jsou dále členěny pomocí dynamického vysouvacího menu.

6.2.1 Nadpis

Každá stránka je jednoznačně definována podle svého nadpisu. Ten je v případě podsekcí článku načítán z databáze a v případě změny názvu sekce se nemusí přetvářet celá stránka, ale jen jeden řádek v tabulce. Veškeré stránky v IS mají jednotnou hlavičku, neboli *title*.

6.2.2 Formulář

Hlavním prvkem editovatelných stránek je formulář. Ten je využíván jak pro přidávání nových příspěvků, tak pro editaci stávajících a samozřejmě také pro smazání záznamů. Tyto editační prvky jsou povoleny všem uživatelům, kteří mají alespoň status registrovaného uživatele.

Některé formuláře bylo nutné ošetřovat kvůli omezeným vlastnostem na vstupu, pomocí JavaScriptu. Ať už šlo o omezení počtu písmen pro vkládání nebo o omezení vstupních znaků jen na číslice, nebo písmena. Tento způsob ověřování má také svou stinnou stránku a to je možnost deaktivace podpory JavaScriptu na straně uživatele. Proto je u některých závažných položek, jako je například změna hesla, implementován algoritmus kontroly i přímo v php scriptu běžícím na serveru.

6.2.3 Formátování

Jelikož se jedná o CMS, je zde velice pravděpodobné, že bude dán velký nárok na vkládání textů a jeho formátování. Naimplementovaný IS dovoluje u příspěvků zobrazování zalamování textů podle toho, jak byly uloženy do databáze. Jelikož se do databáze uloží předformátovaný text obsahující informace o konci řádků a mezerách mezi slovy, je pak velice jednoduché použít funkci *nl2br*, která převede reprezentaci konce řádků na tag `<br/>` a zajistí požadované formátování.

U některých příspěvků je navíc možnost vkládání zvýraznění písma, odkazů na jiné stránky, emailů a také obrázků, pokud autor zná cestu k danému obrázku. Pokud je uživatel znalý ve psaní XHTML kódu, může pro editaci používat přímo XHTML kód. Tato varianta editace je sice velice nebezpečná co se týče zabezpečení stránky, ale v případě uzavřené společnosti lidí se tato hrozba minimalizuje.

6.2.4 JavaScript skripty

Jak bylo zmíněno v kapitole 6.2.2, je zde naimplementována podpora JavaScriptů. Tyto scripty jsou uloženy v adresáři `js` a jsou rozděleny do více souborů pro větší přehlednost.

Nejobsáhlejší soubor s funkcemi JavaScriptu `tags.js` je používán pro editační účely při vkládání textů. Je volán při kliknutí na tlačítko reprezentující danou funkci (např. tučný text). Pokud uživatel používá prohlížeč Internet Explorer, může využít i možnosti vložení tagů přímo na zvýrazněný text. V jiných prohlížečích se tyto tagy vloží za poslední napsaný znak.

V dalším souboru `porovnejbunky.js` jsou naimplementováný funkce na porovnání buněk. Ať už na porovnání datumů při vkládání do akcí a činností, tak také pro kontrolu stejnosti hesel, vyplněných buněk a v poslední řadě kontrola, zda vyplněné datum doopravdy v daném měsíci existuje.

Další zajímavé funkce slouží pro zobrazení kalendáře. Nachází se v souboru `kalendar.js`. Při zavolání tohoto skriptu se vyplní předpřipravená stránka aktuálními dny a rozprostře

se podle pořadí v týdnu podle českých zvyků. Při výběru data kliknutím na něj, se zapíše do buňky v předchozí stránce.

6.2.5 CSS

Protože je celá stránka tvořena pomocí CSS stylů, je také důležité si tyto informace někde uchovávat. K tomuto účelu slouží adresář `css`, ve kterém jsou umístěny soubory s CSS. Z důvodu nekompatibility prohlížečů Internet Explorer, musely být některé styly umístěny do souboru ie.css, který je speciálně využíván těmito prohlížeči. Ostatní prohlížeče využívají soubor `styles.css`. V tomto adresáři jsou také umístěny soubory s CSS styly pro zobrazování kalendáře.

6.2.6 Databázové skripty

Pro komunikaci s databází je nutné využít některou funkci z balíčku funkcí jazyka php. Jelikož připojovaná databáze je typu *MySQL*, omezil se výběr na dva balíčky. Jedním je balíček funkcí využívající příkazů *MySQL* a druhým *mysqli*.

K dalším operacím s databází jsem si vybral *mysqli*. Jedná se o vylepšení *MySQL*. Přesný název v angličtině zní *MySQL Improved Extension*. Toto vylepšení obsahuje objektový přístup a také některé menší zlepšení funkcí.

Skript pro připojení je integrován v souboru `connect.php`. Pro přístup k tabulkám jsou využity standardní příkazy. To samé platí také pro jakoukoliv úpravu v tabulkách (vkládání, mazání, atd.). Přístup k databázi je díky *mysqli* brán objektově.

6.2.7 Skripty pro úpravu obrázků

Pro úpravy obrázků je vytvořena skupina scriptů v jazyce php, které jsou umístěny v souboru `fotgrafie_create.php`. Tyto skripty mají za úkol po načtení obrázku pro vytvoření fotogalerie zjištění jeho validace a následné vytvoření nového adresáře pro tuto fotogalerii. Pokud vytvoření proběhne v pořádku, dojde k vytvoření obrázku sloužícího jako náhled fotogalerie.

Následné nahrávání fotografií do fotogalerie se děje pomocí skriptů obsažených v souboru `fotgrafie_upload.php`.

Při zobrazení fotografií v galerii jsou zavolány skripty obsažené v souboru `g.php`. Tento soubor se stará o zmenšení fotografií do náhledových velikostí.

6.2.8 Ostatní skripty

Pro informování návštěvníků o aktualizaci stánky slouží skript na odesílání emailů s aktualitami. Pokud dojde k provedení skriptu, zjistí se z databáze poslední vložené články. Aktuální hodnota hledání je nastavena na jeden den. Pokud je nějaký článek nalezen, odešle se upozornění o změně. Bohužel u tohoto skriptu nebyla ověřena funkčnost, protože je nutné k jejímu běhu využít fungující poštovní server.

6.2.9 Prvek umělé inteligence

V tomto systému je naimplementována umělá inteligence, která rozhoduje o vytváření rozvrhu výletů na hory. Pro zjednodušení systému bylo zvoleno zaměření jen na hory ve Vysokých Tatrách. Po zadání požadavku na navštívené hory a preferované horské vůdce, dojde k vyhledání a výpisu kalendáře akce.

6.2.10 Export do XML

Pro zazálohování databáze do souboru ve formátu XML je v menu položka Záloha DB. Zde se může zazálohovat buďto celá databáze, nebo také jen jednotlivé tabulky. Obnova databáze ze struktury XML není zatím plně naimplementována, a proto je deaktivována.

```
<email>
<uzel emailID=‘‘1’’ ID=‘‘999’’ email=‘‘pokus.i%40pokus.cz’’/>
<uzel emailID=‘‘8’’ ID=‘‘3’’ email=‘‘ahoj%40a.cz’’/>
<uzel emailID=‘‘9’’ ID=‘‘999’’ email=‘‘pokud%40pokus.cz’’/>
</email>
<prispevek>
<uzel PrisID=‘‘1’’ ID=‘‘999’’ SekID=‘‘1’’ Date=‘‘2007-04-03’’/>
</prispevek>
<clanek>
<uzel PrisID=‘‘21’’ Nadpis=‘‘Nadpis’’ Telo=‘‘%3A%29’’/>
</clanek>
```

Kapitola 7

Závěr

7.1 Shrnutí

Bakalářská práce se zabývá problematikou tvorby informačního systému a také využitím umělé inteligence ve spojení s tímto systémem. V prvních dvou kapitolách je podáno vysvětlení technologie tvorby internetových stránek a jejich výhody. Další kapitoly se více zaměřují na její realizaci a implemetaci, popřípadě na varianty pro danou implementaci.

Při její realizaci bylo využito nejdostupnějších a nejběžnějších technologií, ve verzi k vydání této bakalářské práce. To jest PHP Version 5.2.0, MySQL Version 5.0.22, XHTML 1.0.

7.2 Splnění požadavků

Naimplementovaný systém sice neobsahuje veškeré náležitosti, které byly při prezentaci uváděny, ale z důvodu jejich přetransformování do jiných částí nebo zjištění, že daná náležitost by nebyla přínosná pro IS, byly buďto odstraněny nebo sjednoceny s jinou funkcí.

Naopak specifikace zadané v kapitole 2 byly splněny do posledního požadavku a některé dokonce rozšířeny nad zadaný požadavek.

7.3 Porovnání s jinými systémy

Při porovnání tohoto systému s již naimplementovanými nestatickými systémy pro horolezecké kluby vyšlo najevo, že systémy, které jsou v provozu delší dobu, jsou velice dobře přizpůsobeny uživatelům, než systémy, jejichž stáří je jen několik dnů. V těchto mladých systémech nebývají odstraněny chyby a také nejsou dostatečně přizpůsobeny potřebám uživatelů. Zde platí, že každý uživatel i programátor má vlastní názor na výsledný produkt. Proto nejlepší výsledky budou dosaženy tam, kde je interakce s uživatelem. Na druhou stranu i některé systémy s dlouhou dobou existence jsou schopné dosáhnout jen základních kvalit. Z tohoto poznatku usuzuji, že mnou naimplementovaný systém není sice úplně ideálně vytvořen, ale také nepatří k těm nejhorším. Odpovídá středně kvalitním systémům pro účely horolezeckých klubů.

7.4 Náměty na rozšíření

Pro další rozvoj systému lze vylepšit vkládání článků a jeho formátování na vyšší úroveň. Zavést vyhledávání v daných článcích s inteligentním řazením. Také vylepšení grafické podoby a bezpečnosti při ukládání dat do databáze, je dobrý námět na další změny.

Literatura

[1] J. Havelka a kolektiv. *Vytváříme WWW stránky a spravujeme moderní web site.* Computer Press, a.s., 2004.

[2] F. Zbořil ml. F. Zbořil. *Základy umělé inteligence.* 2006.

[3] P. N. Weinberg J. R. Groff. *SQL - Kompletní průvodce.* Computer Press, a.s., 2004.

[4] J. Kosek. *PHP - Tvorba interatktivních internetových aplikací.* GRADA Publishing, spol. s r.o., 2005.

[5] WWW stránky. Javascript. [online] [cit. 2007-05-01] `http://cs.wikipedia.org/wiki/PHP`.

[6] WWW stránky. Php. [online] [cit. 2007-05-01] `http://cs.wikipedia.org/wiki/PHP`.

[7] WWW stránky. Redakční systém. [online] [cit. 2007-05-01] `http://cs.wikipedia.org/wiki/CMS`.

[8] WWW stránky. Sgml. [online] [cit. 2007-05-01] `http://cs.wikipedia.org/wiki/SGML`.

[9] WWW stránky. Umělá inteligence. [online] [cit. 2007-05-01] `http://cs.wikipedia.org/wiki/Umělá_inteligence`.

[10] WWW stránky. Xhtml. [online] [cit. 2007-05-01] `http://cs.wikipedia.org/wiki/XHTML`.

[11] J. Lažanský V. Mařík, O. Štěpánková. *Umělá inteligence I.* Nakladatelství Academia, 1993.