VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÁ APLIKACE VODNÍHO PÓLA

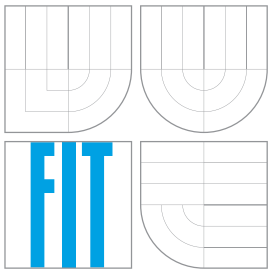
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN LUDVÍK

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

WEBOVÁ APLIKACE VODNÍHO PÓLA

WEB APPLICATION OF WATER POLO

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN LUDVÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ROMAN LUKÁŠ, Ph.D.

BRNO 2007

Zadání

Webová aplikace vodního póla

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 vodního póla. 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 klub vodního póla. Informační systém eviduje informace o zápasech, hráčích, článcích, fotografiích, týmech, atd. Schraňuje informace o výsledcích a dokáže z nich vytvářet statistiky. Přístup k aplikaci je hierarchický podle pravomocí jednotlivých uživatelů autentizovaných heslem. Jde vlastně o jistou odrůdu zjednodušeného redakčního systému. Jednou ze součástí systému je metoda umělé inteligence, která přiřazuje volné rozhodčí k zápasům, které se ještě mají odehrát. Aplikace je vytvořena pro efektivní a přehlednou správu výše uvedených informací.

Klíčová slova

Informační systém, databáze, vodní pólo, 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 an information system for water polo club. Information system stores informations about matches, players, articles, photos, teams, etc. It stores informations about results of this matches too and also it is able to generate some statistics from this records. Access to this application is hierarchic in compliance with rights of an individual user autentificated by password. It is some kind of simple content management system. System also contains one specific method of artificial intelligence, which links free referees to the matches that should be played. Application is created for effective and synoptical management of informations presented above.

Keywords

Information system, database, water polo, entity set, entity, atribut, index, PHP, MySQL, administrator, users rights, XHTML, CSS, JavaScript, artificial intelligence

Citace

Martin Ludvík: Webová aplikace vodního póla, bakalářská práce, Brno, FIT VUT v Brně, 2007

Webová aplikace vodního póla

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Romana Lukáše Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Ludvík
10. května 2007

Poděkování

Děkuji panu Ing. Romanu Lukášovi Ph.D. za vedení a pomoc při tvorbě této bakalářské práce.

Obsah

Kapitola 1

Úvod

Oblast informačních systémů je v poslední době velmi rychle rozšiřující se větví informačních technologií. Je to dáno především díky jejich možnosti propojení s databázovými systémy a snaze většiny podniků, firem a organizací přejít z papírové agendy na plně elektronickou a z větší části automatizovanou, a tím také snížit náklady na lidské i finanční zdroje. Toto je ještě více podporováno tím, že k většině nástrojů pro tvorbu těchto systému, ať už pro tvorbu systému samotného nebo databáze, existují volně dostupné alternativy.

Na jejich tvorbu jsou ovšem kladeny stále větší nároky a to hlavně ze strany uživatelů, kteří si jejich každodenním užíváním zvykli na jistý standard a komfort při jejich použití a ten od tvůrců těchto systémů očekávají, a zadavatelé tento standard přímo vyžadují. Důraz je kladen především na jednoduchost použití, přehlednost a rozšiřitelnost aplikace.

Dalším podstatným pozitivním prvkem webových systémů je, že při použití jistého druhu implementačních nástrojů se nemusí na straně uživatelova zařízení, skrz které přistupuje k systému, instalovat žádné nové aplikace, ani se nemusí klientovo přístupové zařízení nijak zvlášť nastavovat. Pro správnou funkci informačních systému s tenkým klientem založených na běžných webových technologiích stačí jen běžný prohlížeč internetových stránek a připojení k síti, přes které je možno se připojit k serveru. Klientovi poté server poskytne požadované informace (popř. i s připojením k databázovému serveru) a vrátí mu je ve formě běžné webové stránky.

Cílem této práce je nastudování problematiky související s informačními systémy a jednotlivých technologií, jež s touto problematikou souvisejí, a pak následný návrh a vytvoření webového informačního sytému vodního póla.

Tato práce je rozdělena do několika kapitol, v nichž jsou postupně uvedeny problémy při tvorbě aplikace a nástin jejich řešení.

V následující kapitole se budeme zabývat technologiemi pro tvorbu webových informačních systému.

Ve třetí kapitole je provedena analýza systému a rozčlenění problematiky související s jeho návrhem a implementací do jednotlivých částí, kterými se posléze zabývají další kapitoly.

Ve čtvrté kapitole si nastíníme problematiku umělé inteligence, zejména se zaměříme na problém omezujících podmínek a algoritmy pro prohledávání stavového prostoru.

V páté kapitole se budeme věnovat problematice návrhu struktury databáze, která je klíčová ve většině webových aplikacích, jež mají po nějakou dobu schraňovat data.

V šesté kapitole se budeme zabývat samotným návrhem a implementací systému a některými problémy, na které bylo v průběhu vývoje naraženo.

V závěru práce se zaměříme na celkovou zpětnou analýzu vzniklého systému, dodržení

požadavků zadání, na jeho srovnání s obdobnými existujícími systémy a další možnosti jeho rozšíření.

Na konci tohoto dokumentu je vypsána literatura, ze které bylo při tvorbě práce čerpáno, a také je zde uveden seznam příloh, které jsou součástí práce.

Kapitola 2

Technologie pro tvorbu webových informačních systému

V této kapitole bude nastíněna problematika technologií pro tvorbu informačních systému (dále jen IS). A to především IS založených na standardech tvorby webu. A to na standardech, které jsou v dnešní době považovány za závazné.

2.1 Tvorba webových stránek

Webovými stránkami jsou zde myšleny dokumenty, které jsou vytvořeny s pomocí značkovacích jazyků z rodiny *SGML (Standard Generalized Markup Language)* a to konkrétně jazyka *HTML (HyperText Markup Language)*. A v současnosti stále více rozšiřovaného jazyka *XHTML (Extensible HyperText Markup Language)*, jenž se vyvinul pod vlivem rozvoje SGML směrem k XML (eXtensible Markup Language).

Oba tyto jazyky mají svou syntaxi specifikovánu pomocí *DTD (Document Type Definition)*. U jazyka HTML je v době psaní tohoto textu aktuální verze 4.01 specifikace, u jazyka XHTML se jedná o verzi 1.1.

Jazyk XHTML má oproti HTML více striktní syntaxi a snaží se maximálně odstranit značky a atributy pro formátování stránky. Tyto značky byly nahrazeny pomocí *CSS (Cascading Stylesheets – v češtině nazývané kaskádové styly)*. Pomocí kaskádových stylů je poté specifikována sémantika značek používaných v jazyce XHTML.

Pro tvorbu stránek jsem se rozhodl použít jazyka XHTML ve verzi DTD XHTML 1.0 Strict v kombinaci s technologií CSS ve verzi 2.0. K tomuto rozhodnutí přispěla možnost jednoduché změny vzhledu stránek pomocí záměny vkládaného CSS souboru. Další věcí, která přispěla k tomuto rozhodnutí je to, že tato kombinace je v současné době asi nejrozšířenější a zároveň na straně webových prohlížečů nejpodporovanější.

2.2 Využití XML

Extensible Markup Language (XML) je obecný textový formát dat/jazyk resp. datový model. Má přesně danou a jednotnou specifikaci, která je stejně jako HTML nebo XHTML spravována konsorciem W3C (World Wide Web Consorcium).

O jeho využití v projektu bylo rozhodnuto díky jeho snadné analyzovatelnosti a transformovatelnosti. A také proto, že je jednoduše čitelný pro člověka a dokáže přesně vystihnout hierarchický model dat.

Regulární jazyk XML byl použit pro umožnění exportu obsahu databáze do XML souboru, který si administrátor může stáhnout na svůj počítač. A použít jej jako zálohu a nebo jej upravit pomocí transformace do jiného formátu. O tomto exportu do XML se ještě dále zmíníme (viz. 6.1.3).

2.3 Generování webových stránek na straně serveru

Nástroje pro generování webových stránek na straně serveru nejsou tak standardizované, na rozdíl od formátů webových stránek. Prakticky každý programovací jazyk by se dal použít pro generování obsahu webových stránek a jejich syntaktických značek, ale v současnosti existuje opět několik jazyků, které se dostaly do popředí. Těmito jazyky jsou: *PHP (Personal Home Page), Python, ASP (Active Server Pages), JAVA, ...* Ovšem ne všechny technologie jsou poskytovány zdarma a jsou volně šiřitelné.

Momentálně asi nejrozšířenější a zároveň volně použitelný je skriptovací jazyk PHP. Tento je neustále vyvíjen a volně šířen včetně zdrojových kódů, a proto je vyvíjen širokou komunitou uživatelů. V současné době je aktuální verze 5.2.1.

Rozhodl jsem se použít jazyk PHP – verze 5.2.0 (aktuální v době počátku implementace) pro tvorbu skriptů na straně serveru především z důvodu jeho vysoké rozšířenosti a poměrně vysoké rychlosti skriptů. Konfigurační soubor, s kterým by měly být veškeré skripty plně funkční, je uložen na přiloženém datovém nosiči (`/src/config/php/php.ini`).

2.4 Skriptování na straně klienta

Tvorba webové aplikace nás nutí na straně klienta k použití skriptovacího jazyka, který je široce podporován mezi jednotlivými webovými prohlížeči. Tím je v současné době jednoznačně *JavaScript*, a proto na něj padla volba při tvorbě klientských skriptů.

Klientské skripty jsou používány především pro kontrolu údajů ve formulářích před jejich odesláním na server. Je to hlavně z důvodu ušetření přenášených dat po síti. Tyto skripty ale nelze považovat za stoprocentní ochranu proti nepředpokládaným vstupům, protože JavaScript lze v prohlížeči vypnout. Proto je třeba data kontrolovat i na straně serveru.

2.5 Databáze

Jednou z posledních, ale neméně důležitou součástí informačních systémů je databáze. Je to zapříčiněno hlavně samotnou podstatou webových aplikací a samotného HTTP protokolu. Ten je prvoplánově bezstavový a to je jeden z důvodů využití různých doplňujících technik, které by nám umožňovaly tento "nedostatek" napravit. Těmito technikami mohou být cookies, které si prohlížeč ukládá na klientském zařízení, a nebo různé relace na straně serveru. Tyto techniky ovšem dokáží jen částečně zastoupit nedostatky protokolu a navíc nejsou schopny žádným způsobem vyřešit problém ukládání dat vložených uživatelem.

Tato data, stejně jako u běžných programů, po skončení běhu programu zaniknou, pokud nejsou uloženy na nějaké perzistentní úložiště. U webových i jiných aplikací se tímto úložištěm stává databáze. Jejich výhodou, například před ukládáním do běžných souborů, je rychlost, s jakou dokáže databáze s daty pracovat, a také možnost přesunu databáze na jiný server a tím ulehčení od vytížení například webovému serveru.

Pro mou aplikaci jsem zvolil databázi *MySQL* verze 5.0, která je jazykem PHP dobře podporována a je poměrně široce rozšířena a volně dostupná. Konfigurační soubor databáze je uložen na datovém nosiči (`/src/config/mysql/my.ini`)

2.6 Webový server poskytující generované stránky

Veškeré skripty generující webové stránky v jakémkoliv jazyce, i samotné negenerované stránky, musí být uloženy na serveru, který spravuje jejich přístupnost, ale zároveň musí zajišťovat jejich dostupnost a splňovat všechny požadavky na server jako takový. V dnešní době jsou zde dva nejznámější druhy serveru. Prvním z nich je *IIS*, který je komerčním zástupcem, a druhým je *Apache*, který je opět poskytován zdarma a šířen i se zdrojovými kódy. Je tím opět umožněna správa a vývoj pro širokou komunitu vývojářů a uživatelů.

Vzhledem k zvolení volně šiřitelného jazyka pro psaní skriptů na straně serveru a volně dostupné databáze jsem se rozhodl pro stejné řešení i zde. Byl tedy zvolen server Apache ve verzi 2.2. Dalším důvodem je velmi dobrá spolupráce mezi zvolenými prostředky (serverem, jazykem a databází). Konfigurační soubor, s jehož použitím byla aplikace testována, odladěna a shledána plně funkční, je opět uložen na přiloženém datovém médiu (`/src/config/apache2.2/httpd.conf`).

2.7 Shrnutí

Nástroje zvolené pro implementaci webové aplikace jsou v dnešní době považovány za standard ve tvorbě IS postavených na základě webových technologií. Jsou volně dostupné, stále vyvíjené a udržované, aby odpovídaly stále rostoucím požadavkům a držely krok s trendy ve světě informačních technologií. Vybrané prostředky jsou tedy *jazyk PHP, server Apache, databáze MySQL, jazyk JavaSript*. Pomocí těchto prostředků jsou tvořeny webové stránky odpovídající XHTML (ve formě dle specifikace DTD XHTML 1.0 Strict) s využitím CSS 2.0 pro stanovení vzhledu stránek.

Kapitola 3

Analýza požadavků na systém

V této části se budeme věnovat analýze požadavků na IS. Díky této analýze si aplikaci rozčleníme na dílčí problémy a jejich řešení si poté načrtneme v dalších kapitolách.

3.1 Popis aplikace

Aplikace je určena pro záznam a zveřejňování výsledků zápasů, článků, fotek a komentářů k nim. Systém mimo výsledků zápasů eviduje také detailnější statistiky jednotlivých utkání (jako např. vyloučení, pokutové hody, atd.) a jednotlivých týmů.

Aplikace má jistou strukturu sekcí, kterou může administrátor upravovat a může zakládat nové sekce a nebo jiné mazat. Sekce má šablonu, kterou může administrátor před vytvořením sekce pozměnit. Šablonou je uspořádání podsekcí. A její úpravou se míní zvolení podsekcí, které bude sekce obsahovat.

Součástí systému má být též funkční prvek, který dokáže nasazovat rozhodčí k ještě neodehraným utkáním, a to na základě podmínek předepisujících jaký zápas smí který rozhodčí řídit.

3.2 Role uživatelů v systému

Měly by se zde objevovat 4 druhy rolí, jimiž jsou administrátoři, editoři, uživatelé a hosté. Hierarchie je taková, že na nejnižší úrovni s nejnižším počtem uživatelských práv jsou hosté a nad nimi jsou v následujícím vzestupném pořadí uživatelé, editoři a administrátoři. Uživatelé s vyšším stupněm oprávnění mají mimo jiné také oprávnění všech typů uživatelských účtů na nižších úrovních.

Hosté mají možnost prohlížet stránky (např. výsledky zápasů, rozpisy, články, komentáře, fotky, atd.). Dále mají možnost si vytvořit vlastní účet, aby takto získali práva běžných uživatelů, která budou popsána níže. Pro úspěšné vytvoření je třeba vyplnit unikátní login, heslo a emailovou adresu.

Běžní uživatelé mohou mimo akcí již zmíněných u hostů také přidávat nové komentáře k jednotlivým článkům. Tyto komentáře mohou také mazat, ale to jen pod podmínkou, že komentář je vložen jimi samotnými. Navíc ještě mohou provádět změny hesla svého účtu. Nemohou ovšem měnit svůj login.

Editoři mohou být zaregistrování pouze administrátorem a to tak, že si vytvoří běžný účet a pro ten jim poté administrátor zvýší oprávnění. Mají všechny možnosti předcházejících účtů a navíc mohou vytvářet, editovat a mazat články, výsledky zápasu a detaily zápasu,

ke kterým mohou být připojeny popisy. Dále je možné vkládat a mazat fotky a popisky k nim. Editor také může mazat komentáře ostatních uživatelů. Má možnost vkládat, editovat a mazat termíny v rozpisech zápasů. Také mají možnost vkládat a editovat týmy a jejich hráče. A má možnost vkládat, mazat rozhodčí a k nim doplňovat termíny, ve kterých nemají čas na řízení zápasu.

Administrátoři mohou registrovat nové editory. Mají možnost kompletně spravovat účty (mazat, měnit oprávnění, atd.). Mohou mazat a upravovat veškeré příspěvky, které byly vloženy ostatními uživateli. Administrátor může také zakládat nové sekce. Pro tyto sekce má možnost upravovat šablonu (zmiňováno výše).

3.3 Vlastní analýza

Na základě podrobné analýzy proběhlo vypracování případů užití systému. Tyto případy užití byly následně promítnuty do diagramu případů užití (tzv. use case diagram). Tento diagram je znázorněn na obr. 3.1.

Diagram případů užití znázorňuje jednotlivé aktory a s nimi spojené případy užití systému. Tento způsob modelování je typický v počátku tvorby většiny aplikací (nejen IS). Díky tomuto modelu se dá velmi přesně určit kdo a za jakým účelem bude aplikaci používat.

Dekompozice problémů spojených s aplikací přinesla rozdělení celého úkolu na několik částí. Toto rozdělení vycházelo z již zmiňovaného diagramu případů užití a bylo zvoleno také v souvislosti s řešením problémů pomocí jednotlivých implementačních prostředků a vzájemném napojení jednotlivých problémů na různá specifická odvětví informatiky. Diagram je přiložen i v lepším rozlišení na dodaném datovém nosiči (`/doc/diagrams/uc/usecase.pdf`).

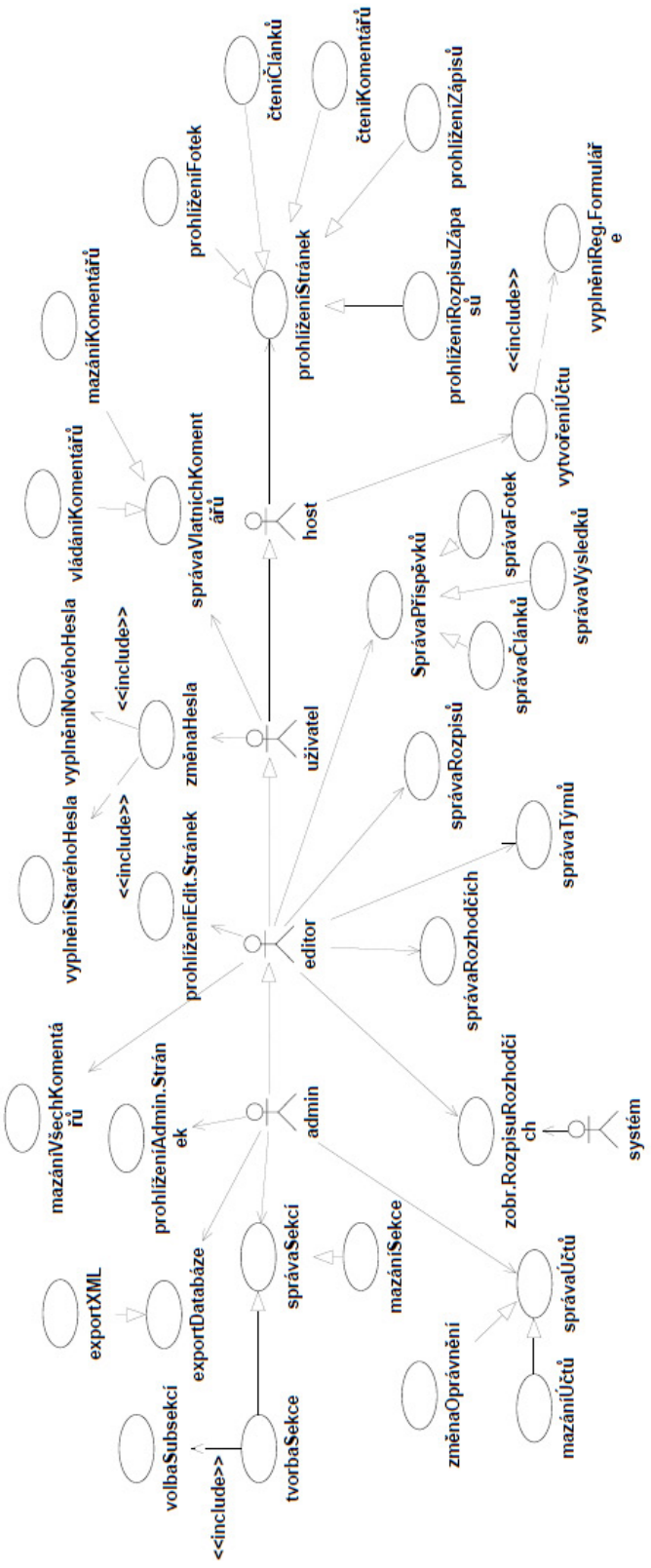
3.4 Shrnutí

Z výše uvedených požadavků na systém vyplývá, že bylo třeba vytvořit alespoň jednoduchou formu redakčního systému (viz. 6.1.1), který by spravoval obsah aplikace. To se muselo projevit jak v tvorbě aplikačního rozhraní, tak se tato skutečnost promítla do struktury databáze, která bude tyto informace schraňovat. Musí zde být také respektována uživatelská práva pro přístup k obsahu a pro jeho editaci.

Systém byl tedy rozdělen na následující části:

- Problémy spojené s hledáním rozhodčích k zápasům (umělá inteligence)
- Návrh a vytvoření struktury databáze, která by odpovídala požadavkům aplikace
- Návrh a tvorba samotné aplikační vrstvy

Jednotlivé problémy a způsoby jejich řešení jsou popsány v následujících kapitolách.



Obrázek 3.1: Diagram případů užití systému

Kapitola 4

Umělá inteligence

Problémy spojené s programováním umělé inteligence (dále jen UI) jsou velmi rozsáhlé a pokrývají širokou škálu oborů. Pro implementaci řešení lze použít prakticky všechny programovací jazyky. Programování se poněkud liší ve stylu přístupu k řešení problému. Při řešení problému spojených s UI záleží na analýze problému ještě více než při řešení klasických algoritmických problémů. Stěžejním bodem v našem případě je snažit se najít akceptovatelnou kombinaci rozhodčích a zápasů, přičemž se snažíme dodržovat omezující podmínky, které limitují rozhodčího v tom, aby směl zápas řídit. Jde vlastně o vytváření rozvrhu rozhodčího. Rozvrhu, který rozhodčímu stanoví místo a datum řízení zápasu.

4.1 Programování s omezujícími podmínkami

Programování s omezujícími podmínkami představuje stále se rozšiřující techniku pro deklarativní popis a efektivní řešení složitých problémů především kombinatorického rázu. Omezující podmínky našly široké uplatnění v mnoha sférách a vyskytují se v mnoha praktických aplikacích, jako jsou například aplikace pro zpracování přirozeného jazyka, počítačové grafiky, návrh plošných spojů, ...

Formálně je úloha s *omezujícími podmínkami* definována takto:

- Nechť existuje množina proměnných $\{X_1, X_2, \ldots, X_n\}$, z nichž každá proměnná X_i může nabývat hodnot z neprázdné domény D_i.
- Nechť existuje množina omezujících podmínek $\{C_1, C_2, \ldots, C_m\}$, z nichž každá podmínka předepisuje vztah mezi nějakou podmnožinou proměnných z výše zmíněné množiny proměnných.
- Nechť stav úlohy je definován hodnotami přiřazenými jednotlivým proměnným a nechť legální stav znamená, že proměnné s přiřazenými hodnotami vyhovují předepsaným podmínkám.
- V počátečním stavu není přiřazena hodnota žádné proměnné.
- Obecný operátor přiřazuje hodnotu libovolné volné proměnné tak, aby následný stav byl legálním stavem.
- Řešením úlohy je legální stav, ve kterém všechny proměnné mají přiřazené hodnoty.

Důležitým prvek v tvorbě podmínek je jejich arita (násobnost). Tato násobnost vyjadřuje počet proměnných v podmínce. Podstatnými typy podmínek jsou binární a vícenásobné. Unární podmínky vedou pouze ke snížení velikosti domény proměnné, a proto nejsou tolik důležité. Binární a n-ární podmínky jsou podstatné díky možnosti převodu každého systému s vícenásobnými podmínkami na systém se samými binárními podmínkami. Takovýto systém pak snižuje složitost popisu řešení i složitost výsledného algoritmu.

V praxi se ovšem tento postup často neuplatňuje a navržený algoritmus je upraven tak, aby mohl pracovat s n-árními podmínkami.

Stanovení podmínek v případě naší aplikace vychází jak z přirozeného logického uvažování, tak i z pravidel soutěže vodního póla v ČR a požadavků na systém. Není zde snaha prosadit, aby systém byl sestaven ze samých binárních podmínek.

Podmínky byly stanoveny takto:

1. Rozhodčí nemůže být delegován na 2 zápasy v jeden den.
2. Dále nemůže rozhodovat zápas týmu, který je jeho domovským klubem.
3. Zápas smí být řízen pouze rozhodčím, jenž je oprávněn posuzovat utkání této kategorie.
4. V systému si může rozhodčí také vyčlenit data, kdy nemá volno pro řízení utkání. Tyto data musí být také systémem respektována jako omezující podmínky a systém na tyto data musí také reflektovat při nasazování rozhodčích.

4.2 Řešení systému podmínek

Systémy s podmínkami můžeme rozdělit do dvou základních skupin. V prvním případě jsou to problémy s konečným počtem proměnných, kde je každá proměnná konečnou doménou a podmínek v takovém systému je konečně mnoho. Každá podmínka snižuje počet kombinací možných hodnot. Z tohoto pohledu se jedná o kombinatorický problém, který je v terminologii programování s omezujícími podmínkami označován jako *Constraint Satisfaction Problem (CSP)*.

Druhou skupinu tvoří problémy s nekonečnými doménami, takovou doménou může být například interval v oboru reálných čísel. Problémy s nekonečnými doménami se ale většinou obor programování s omezujícími podmínkami nezabývá.

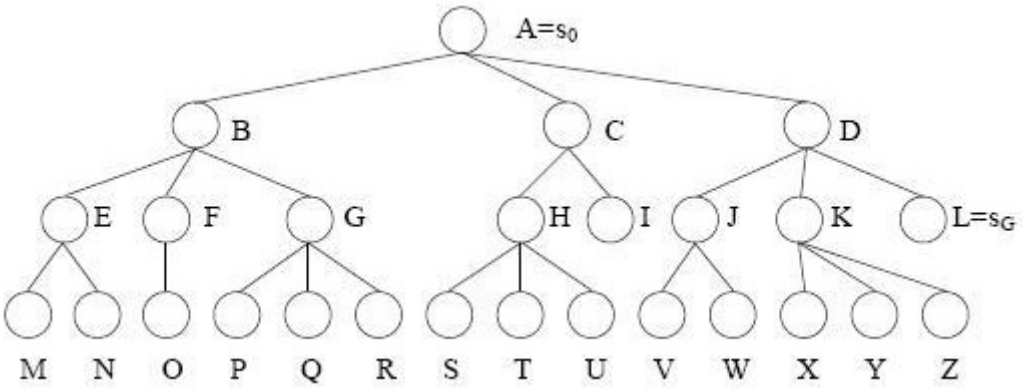
V naší aplikaci se tedy také budeme zabývat pouze systémem s konečným počtem proměnných.

Budeme se snažit prohledávat stavový prostor, který nám poskytnou všechny možné kombinace proměnných. Pro činnost prohledávání stavového prostoru existuje poměrně široká škála algoritmů, jejich dělení a vlastnosti budou zmíněny v následujících odstavcích.

4.3 Algoritmy pro řešení systému podmínek

Algoritmy prohledávající stavový prostor a řešení systému podmínek mohou být použity k řešení široké škály problémů. Jejich použití je poměrně obecné a převážná část problému jejich tvorby spočívá v nalezení podmínek, které musí být splněny, stanovení počátečního a cílového stavu a specifikace stavu jako takového.

Typické hledání řešení vyjádřené v grafické formě má podobu stromového grafu (viz. obr. 4.1), kde uzly zobrazují jednotlivé stavy a hrany jsou zobrazením přechodů, při kterých musejí být splňovány stanovené podmínky. Startovací stav je označen S_0 a cílový stav je označen jako S_G. Cílem algoritmu je za splnění všech podmínek projít ze startovacího stavu do stavu cílového.



Obrázek 4.1: Ukázka stavového prostoru

Nejjednoduššími algoritmy pro prohledávání stavového prostoru jsou algoritmy typu - *generate & test*. Tento typ algoritmů jednoduše prochází postupně jednotlivé uzly stromu a ty rozgenerovává a zkouší, jestli jsou dodrženy v novém uzlu všechny podmínky a zda není nový stav cílovým. Takovými algoritmy jsou např. *Breadth First Search, Depth First Search, Uniform Cost Search, atd.*

Existují ale také pokročilejší metody hledání řešení CSP problémů. Tyto metody jsou výrazně efektivnější. Zmíníme zde několik nejznámějších.

Těmito metodami jsou:

1. Backtracking
2. Forward checking
3. Min-conflict

4.3.1 Backtracking

Princip metody je velmi jednoduchý: V každém kroku přiřazuje hodnotu neoznačené proměnné a pokud nový stav není legálním stavem, přiřazuje další hodnotu atd. Není-li možné této proměnné přiřadit žádnou hodnotu, pak se okamžitě vrací o krok zpět.

4.3.2 Forward checking

Princip algoritmu Forward checking spočívá v tom, že po každém přiřazení hodnoty nějaké proměnné vyřazuje z množin přípustných hodnot dosud volných proměnných ty hodnoty, které jsou s právě přiřazenou hodnotou v konfliktu.

4.3.3 Min-conflict

Algoritmus Min-conflict je lokálním algoritmem, který vychází z libovolného úplného (všem proměnným jsou přiřazeny nějaké hodnoty), ale nelegálního (přiřazené hodnoty nesplňují podmínky) stavu a snaží se zmenšovat počet konfliktů (tj. počet porušených podmínek). Tento jinak velmi jednoduchý přístup je překvapivě efektivní pro mnoho CSP.

4.4 Zvolený algoritmus pro vyhledávání

Pro hledání cílové kombinace zápasů a rozhodčích byl zvolen algoritmus *Backtracking*. Tento algoritmus využívá datovou strukturu zásobník (možné je použít třeba i strom), ve kterém si uchovává jednotlivé stavy, které mají být rozgenerovány (tento zásobník bude označován jako OPEN). Tato metoda prohledávání stavového prostoru byla zvolena z důvodu nepříliš mnoha podmínek, které mají být dodržovány, a jeho dobré implementovatelnosti pro náš problém. Další výhodou je to, že není třeba problém komplikovat propagací podmínek do dalších uzlů.

Tento algoritmus není hodnocen jako optimální, to znamená, že algoritmus nenalezne v každém případě nejlepší řešení. Časová náročnost tohoto algoritmu je exponenciální a paměťová náročnost je extrémně nízká, protože si algoritmus udržuje kromě zpětné cesty ke kořenovému uzlu jen aktuálně rozgenerovávaný uzel.

Princip algoritmu:

1. Sestroj zásobník OPEN (bude obsahovat všechny uzly určené k expanzi) a umísti do něj počáteční uzel.
2. Je-li zásobník OPEN prázdný, pak úloha nemá řešení a proto ukonči 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.

4.5 Pořadí vyhodnocování podmínek

Ať už je použit jakýkoliv algoritmus pro hledání cílového stavu, tak dochází k postupnému vyhodnocování omezujících podmínek. Pořadí vyhodnocování podmínek může výrazně ovlivnit rychlost nalezení cílového stavu.

Pro tuto oblast existuje opět několik možných přístupů. Na tomto místě si zmíníme dva. Prvním z nich je *first-fail*, druhým *succeed first*.

Princip *first-fail* spočívá ve výběru takové proměnné, která půjde nejhůře ohodnotit, například z toho důvodu, že je tato proměnná obsažena v nejvíce podmínkách nebo proto,

že je její doména nejmenší. Motivací snahy vybrat nejhůře ohodnotitelnou proměnnou je fakt, že pokud její ohodnocení nepovede k řešení, je dobré to zjistit co nejdříve.

Princip *succeed first* přistupuje k tomuto problému z opačného pohledu. To znamená, že se budeme snažit vždy vybrat tu hodnotu, která by nás mohla nejrychleji dovést k cíli. Princip vychází z iterativního přístupu programovacího algoritmu. Pokud budeme muset projít všechny její hodnoty, je jedno, v jakém pořadí je budeme brát. Pokud nás ale některá z hodnot povede k nalezení řešení, budeme se snažit tuto hodnotu vybrat co nejdříve.

Pro náš případ byla zvolena první varianta čili metoda first-fail. Spojením podmínek pro zákaz řízení dvou zápasů v jeden den a respektovaní dní, kdy si rozhodčí zvolil volno, získáváme pravděpodobně nejhůře splnitelnou podmínku. A proto ji použijeme jako první, abychom snížili počet řešení co nejrychleji. Zbývající dvě podmínky mají prakticky stejnou váhu, pokud systém nemá výrazně vyšší počet týmů nebo kategorií, a proto se jejich pořadí v rychlosti řešení příliš neprojeví.

4.6 Shrnutí

Předpokladem správné funkčnosti vyhledávání kombinací je nejprve stanovení rozpisu zápasů a teprve následovného doplnění obsazených termínů pro rozhodčí. Samozřejmou skutečností v realitě, která je promítnuta do systému je na počátku vyhodnocování, vyšší počet rozhodčích, kteří by mohli zápasy řídit, než je maximální počet zápasů hraných v jednom dni. Tento počet se může stát nižším, až dodatečným přidáváním volných termínů rozhodčích. Toto přidávání ale nemůže probíhat před vložením uceleného rozpisu zápasů.

Problémem návrhu by se mohla zdát špatná vytíženost některých rozhodčích. To je způsobeno tím, že algoritmus nebere v potaz počet zápasů řízených určitým rozhodčím. Tento problém by mohl být opraven zavedením heuristik na počet zápasů. Další možnou heuristikou by se mohla stát vzdálenost města, kde má sídlo domovský klub rozhodčího a místa, kde má rozhodčí řídit zápas.

Implementační komplikací se projevilo, kromě stanovení podmínek (řešeno výše 4.1) a volby algoritmu (část 4.4), hledání struktury stavu, který je používán pro udržování částečných řešení na zásobníku. Struktura se nakonec skládá jak ze zápasů, kterým se snažíme přiřadit rozhodčího, tak i z rozhodčích, jež považujeme za operátory. Je to z důvodu měnících se možností přiřazení rozhodčího na základě jeho volných dnů. Udržováním aktuálních dnů volna každého rozhodčího v každém stavu nám zajišťuje následné správné používání rozhodčího jako operátoru.

V této části bylo čerpáno z literatury [4], [1].

Kapitola 5

Návrh struktury databáze

Databáze má za úkol schraňovat data, jenž mají být uloženy v aplikaci. Vývoj databází jako takových ovšem nebyl jednoduchý.

5.1 Volba typu databáze z pohledu datového modelu

V začátcích tvorby aplikací (nejen webových) byla veškerá data součástí procesů, definice dat byla součástí definice procesů a data nemohla být sdílena více procesy, což bylo komplikací pro tvorbu složitějších aplikací.

Postupem času se projevila snaha oddělit definici dat i data vlastní od procesů a také umožnit přístup k datům více procesům současně. Ta byla v současnosti dotažena k odstranění všech výše uvedených překážek. Byla dokonce překonána a to tak, že v současné době je možné vyčlenit mimo procesy i definici uživatelského rozhraní a samotnou definici procesů lze také vytáhnout mimo vlastní aplikaci.

Databáze jsou členěny také podle typu datového modelu, který definuje logickou organizaci dat v databázi.

Nejznámější typy modelů:

1. Hierarchický
2. Síťový
3. Relační
4. Objektový

V dnešní době jsou již překonány první dva zmiňované modely a nejrozšířenějším se stal model relační, i když vývoj databázových technologií začíná směřovat k objektovému modelu a objevuje se také stále širší spektrum tzv. objektově-relačních databází, které kombinují výhody těchto dvou přístupů.

V našem případě budeme používat databázi relační. Pro vytvoření struktury relační databáze z popisu požadavků systému existuje propracovaná metodologie *Konceptuálního modelování*, i když její použití nemusí být vždy zcela triviální záležitostí. V případě špatné analýzy a nebo chybě při aplikaci metodiky může dojít k vytvoření špatně modifikovatelné a

těžko použitelné struktury databáze. Námi použitá metodika povede na vytvoření *ER (entity relationship) diagramu* z popisu požadavků na systém a na následné převedení diagramu na strukturu tabulek databáze.

ER diagram se snaží modelovat svět, kdy entity jsou chápány jako objekty reálného světa rozlišitelné od jiných objektů, o nichž chceme mít informace v databázi a vztahy tvoří asociace mezi těmito objekty.

5.2 Nástin požadavků na databázi naší aplikace

Databáze musí být schopna udržovat údaje o loginech, heslech a rolích uživatelů. Podstatné je také propojení mezi uživatelem a relací, které se vytváří při jeho přihlášení do systému. Ukládají se informace o uživatelských příspěvcích do systému, kterými mohou být obrázky, články, zápasy a komentáře. Databáze musí udržovat strukturu sekcí a subsekcí, které musejí mít propojení se svým obsahem. Důležité pro zápasy jsou týmy, jejich hráči, trenéři, události zápasu, data jejich konání, komentáře k zápasům a rozhodčí, kteří je řídili. Důležitým pro každý vstup je čas jeho vložení a jeho autor. V rámci vytváření rozpisů rozhodčích pomocí metody umělé inteligence je třeba mít uloženy i data, kdy si rozhodčí rozhodl vzít volno od pískání zápasu. Další požadavky vyplývají z popisu aplikace (viz. 3.1)

5.3 Tvorba ER diagramu

Na následujícím obrázku 5.1 je znázorněn ER diagram struktury databáze. Tento diagram je v lepším rozlišení přiložen na datovém nosiči (`/doc/diagrams/er/er.pdf`). V obrázku jsou vyznačeny entity s jejich atributy a také vztahy mezi těmito entitami. Tyto entity a vztahy jsou popsány v anglickém jazyce, aby následný převod na tabulky vedl také na anglická pojmenování. Tato pojmenování jsou výhodná z důvodu nepříliš dobré kompatibility kódování diakritiky u některých databázových systémů.

V struktuře ER diagramu měl poměrně důležité slovo zamýšlený redakční systém. Struktura musela být tvořena tak, aby nedocházelo ke zbytečné složitosti databáze, a také aby se předešlo případným potížím při následném rozšiřování systému (např. nový druh příspěvků). Tato vlastnost se projevuje v generalizaci entit *article, photo, match, comment* do entity *entry* a jejím následném spojením s entitou *user* a *subsection*. Ostatní entity se již z větší části starají o ligu vodního póla (soupisky, rozhodčí, statistiky, . . .). Z tohoto trochu více vybočuje již jen entita *refunavailable*, která má spíše spojitost s vytvářením kombinací rozhodčích a zápasů.

5.4 Převod ER diagramu na systém tabulek

Následným krokem v postupu tvorby struktury databáze je převod ER diagramu na tabulky. Ty jsou zobrazeny na následujícím obrázku 5.2.

V těchto tabulkách již jsou vystiženy veškeré atributy databázových tabulek včetně *cizích klíčů* (klíčů, které se používají k vzájemnému provázání tabulek). Tyto tabulky byly následně vloženy do databáze a byly mezi nimi vytvořeny i příslušné vazby.

V tabulkách jsou pomocí označení PK zvýrazněny primární klíče tabulek a pomocí FK jsou označeny cizí klíče.

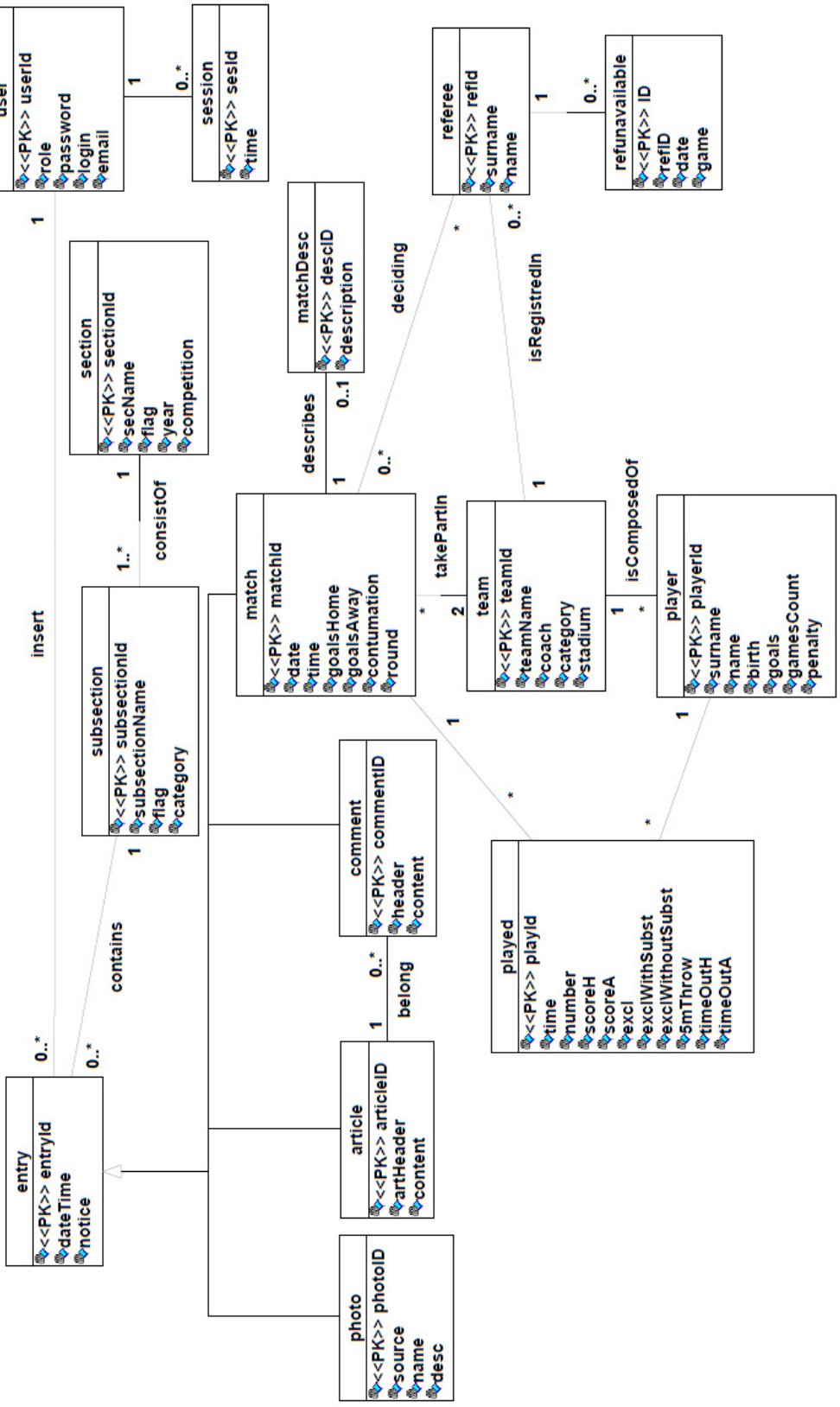
Při přeměně ER diagramu na tabulky jsem se rozhodnul, že každá specializace příspěvku (entry) bude reprezentována vlastní tabulkou. Toto řešení bylo zvoleno z důvodu jednodušího přidávání tabulek pro nové druhy příspěvku. Zjednodušení je ukryto v tom, že není třeba nijak upravovat již existující tabulky, ale stačí jen dodat tabulku novou.

5.5 Shrnutí

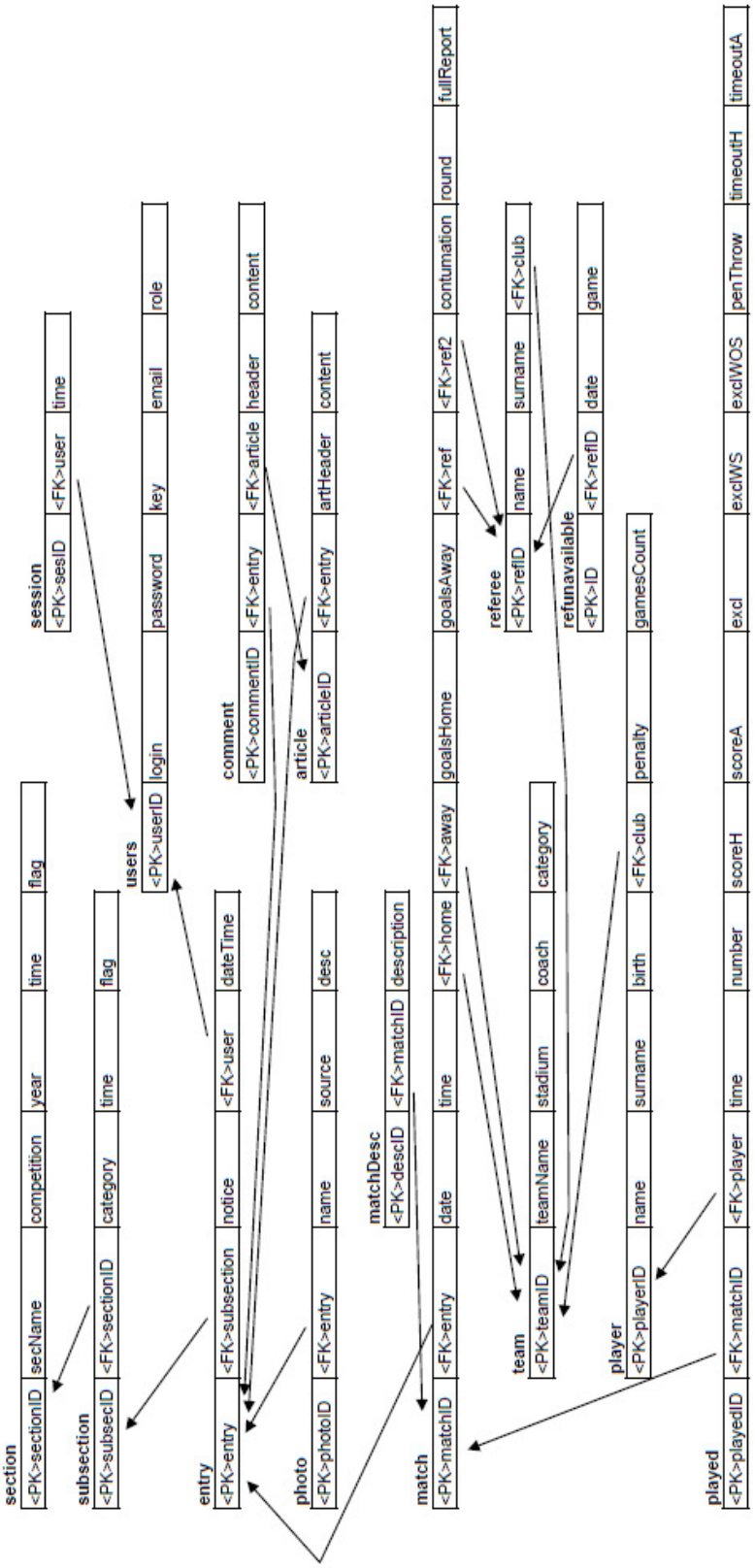
V této sekci jsme si nastínili vytvoření databázové struktury pomocí strukturovaného konceptuálního modelování. Vytvořili jsme podle požadavků systému ER diagram, který jsme následně převedli na tabulky, které velmi přesně reprezentují tabulky v relační databázi a jsou zde znázorněny i vztahy mezi těmito tabulkami.

Tato část byla poměrně důležitá a vzhledem k dobré analýze a návrhu struktury databáze, již nebylo třeba více do struktury databáze zasahovat, vyjma přidání několika potřebných atributů v průběhu implementace (jsou již v diagramech promítnuty), a aplikační vrstva na ní mohla poměrně bezproblémově stavět.

V této kapitole jsem čerpal z následujících zdrojů [2], [8], [6].



Obrázek 5.1: Výsledný ER diagram použitý pro transformaci na tabulky



Obrázek 5.2: Tabulky, na které byl převeden ER diagram

Kapitola 6

Návrh a implementace aplikační vrstvy systému

Aplikační vrstvou v našem případě je myšleno webové rozhraní informačního systému, ale také skripty pracující nad databází ale i bez ní, které vykonávají i jinou než zobrazovací činnost spojenou se zobrazením uživatelského rozhraní.

Grafické rozhraní je tvořeno pomocí knihovny `page.php`, která vytváří objektovou nadstavbu nad možnostmi generování webových stránek pomocí jazyka PHP. Tato nadstavba zajišťuje plnou validitu vzhledem k používané verzi DTD (v našem případě XHTML 1.0 strict). V případě nevalidního dokumentu tato knihovna vyvolá příslušnou výjimku a generování příslušné stránky se nezdaří. Tato knihovna mnou byla pouze upravena (původní vývoj: Vojtěch Orgoň) a doplněna o určité funkce, které jsem v ní postrádal (vkládání obrázků, JavaScriptu, ...). Její další výhodou je kromě neustálé validace webových stránek možnost přehledněji vytvářet stránky.

6.1 Přiblížení požadavků na aplikační vrstvu

Většina požadavků na aplikační vrstvu byla již rámcově zmíněna v této sekci, a nebo v předchozím textu (viz. 3.1). Nyní se budeme jednotlivým požadavkům věnovat zvlášť.

6.1.1 Redakční systém

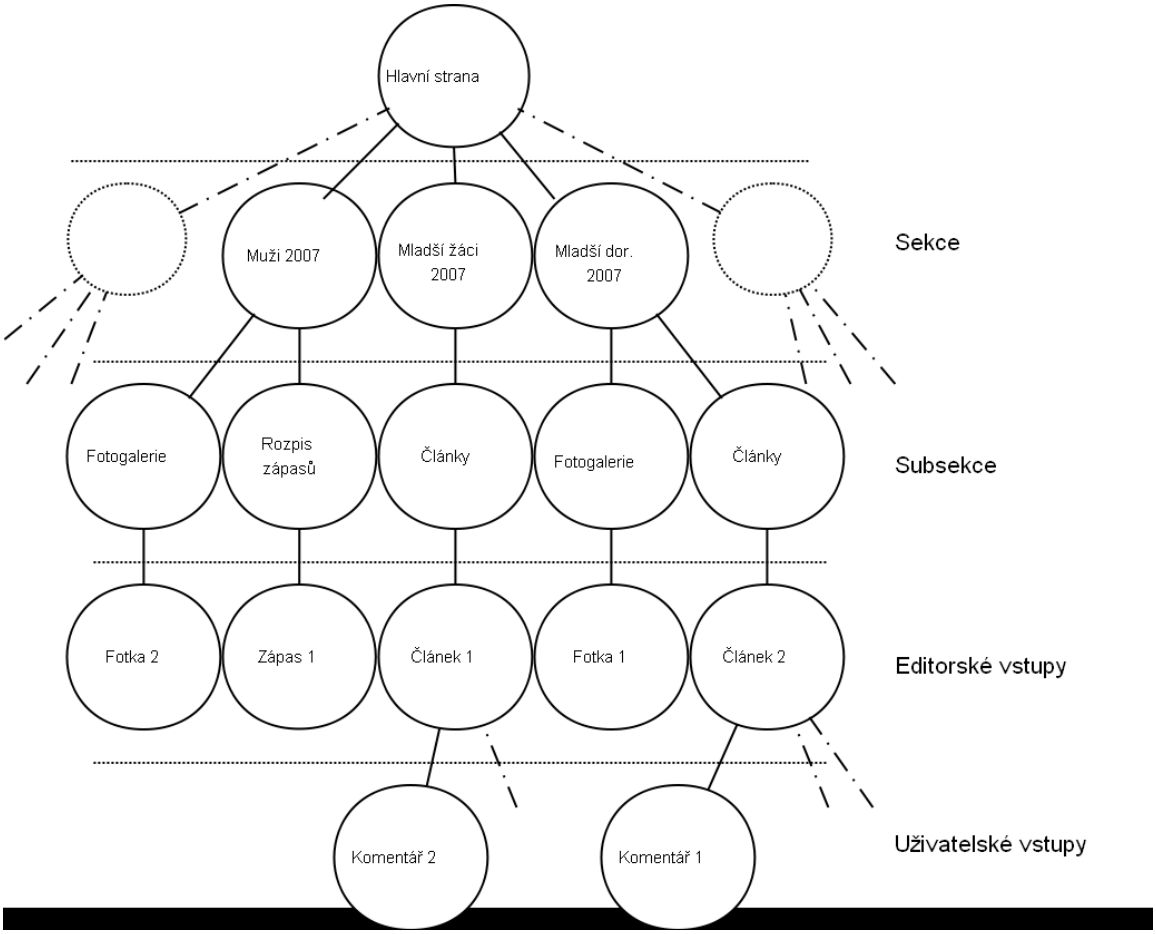
Redakční sytém (CMS), zkratka anglického content management system (systém pro správu obsahu), je označení pro software zajišťující správu dokumentů, nejčastěji webového obsahu. V dnešní době se jako CMS zpravidla chápou webové aplikace, někdy s případným doplňkovým programovým vybavením u klienta.

Jednou ze základních snah při vytváření IS bylo vytvořit alespoň jednoduchý redakční systém, který by umožnil editorům minimálně spravovat fotky, články a komentáře. A to z toho důvodu, aby administrátorovi odpadla většina starostí s tímto spojená. Když vyjdeme z ukázkové stromové struktury webu na obrázku 6.1 (možná větší rozvětvenost je naznačena čerchovanými čarami a tečkovanými kružnicemi), tak lze vidět, že první patro stromu tvoří sekce. Tato část byla ponechána plně v rukou administrátora.

Druhé patro tvoří subsekce, které se váží vždy k adekvátní sekci a jejich tvorba probíhá zároveň se sekcemi. A tím pádem je opět spravuje administrátor systému.

Ve třetím patře už můžeme vidět jednotlivé příspěvky (články, fotky, ...). A až na tomto třetím patře se do hry dostávají i editoři a mohou se starat o obsah.

Ve čtvrtém patře jsou pak už jen komentáře ke článkům, které mohou přidávat a mazat i jednotliví uživatelé. Celý tento obsah se geneneruje z databáze a vytváří nám strukturu systému. Na stromu nejsou znázorněny statické části webu (ty, které nejsou generovány z databáze).



Obrázek 6.1: Ukázka možné stromové struktury webu

Pro redakční systém je důležité respektovat již zmiňovaná uživatelská práva. Jejich úroveň je třeba udržovat i během přechodu mezi jednotlivými stránkami a po celou dobu používání IS. Existuje několik přístupů. Jako například udržovaní stavu relace pomocí cookies a nebo pomocí udržování relace na serveru. Třetí, a v našem případě použitou možností, je řešení celého problému na aplikační úrovni s pomocí databáze.

Při přihlášení do databáze je vygenerováno náhodné číslo pomocí generátoru *Mersenne Twister*, které je použito jako identifikační číslo relace. Číslo je uloženo do databáze a při každém přechodu z jedné stránky IS na druhou dojde k jeho porovnání s číslem uvedeným v adrese. Při odhlášení uživatele je číslo smazáno z databáze a tím je znemožněno jeho zneužití.

Pro zajištění bezpečného uchování hesla v databázi jsou hesla uložena v zahashované podobě. A to pomocí algoritmu *SHA1 (Secure Hash Algorithm)*. Tento algoritmus je v dnešní době široce používán pro zabezpečení hesel a také v jiných šifrovacích úlohách.

6.1.2 Uživatelské rozhraní

Hlavní snahou při tvorbě uživatelského rozhraní bylo snažit se udržet jednoduchost a přehlednost. Stěžejními body rozhraní jsou ovládací prvky. O uživatelově názoru na IS totiž rozhoduje především jeho spokojenost s ovládáním systému. To jak rychle a pohodlně se dostane k požadovaným informacím a kolik ho to bude stát úsilí. Vzhledem k tomu, že se aplikace člení na jednotlivé sekce a subsekce, tak bylo potřebné vytvořit minimálně dva ovládací prvky. Jeden pro navigaci mezi jednotlivými sekcemi a druhý pro jejich samotné ovládání. Hlavní navigační menu pro přepínání mezi jednotlivými sekcemi je umístěno ve vodorovném pruhu při dolním okraji hlavního banneru (úvodního nápisu) a menu pro lokální ovládání bylo umístěno do sloupečku na levém okraji zobrazovací plochy. Obě tato místa jsou velmi často používána pro umístění ovládacích prvků na drtivé většině webových prezentací, a proto si myslím, že tato místa byla zvolená vhodně a uživateli jednoznačně usnadní orientaci v organizační struktuře webu.

Dalšími podstatnými prvky, které jsou při tvorbě webu důležité, jsou nadpisy a podnadpisy jednotlivých stránek a atribut *title*. Tento atribut se zobrazuje v záhlaví okna prohlížeče a je velmi podstatný pro vyhledávače, ale i pro alternativní typy médií (např. zvukový výstup). Tyto součásti stránek, především atribut title, by měli zůstávat aktuální v rámci každé stránky a ne jen v rámci celého webu a usnadnit tak uživatelům orientaci v IS.

Dalším problémem bylo vytvoření velkého množství formulářů, aby uživatel mohl do systému přispívat a tím ho měnit. Formuláře musely zůstat dostatečně přehledné, i přesto, že v některých případech jsou velmi složité. A bylo nutno snažit se kontrolovat všechny vstupy. Většina vstupů je ošetřena již na straně klienta pomocí JavaScriptu, ale ten si může uživatel vypnout, a proto musí být většina vstupů kontrolována i na straně serveru. I přes tuto skutečnost se uživatelům vypnutí JavaScriptu důrazně nedoporučuje, protože mohou přijít o celý pracně vyplňovaný formulář v případě jeho kontroly až na straně serveru.

Dalším problémem bylo formátování textu v delších příspěvcích, kde je třeba docílit jistého členění textu. Vzhledem k tomu, že tato část se týká v praxi pouze článků, a ty mohou vkládat pouze editoři a nebo administrátor, tak bylo rozhodnuto, že formátování se nechá zcela na vůli tvůrce článku a použije se pro něj HTML element $<\ pre\ >\ Text <\ /\ pre\ >$, jenž umožňuje volné formátování textu ve smyslu odsazení a zalamování řádků. Články nepodporují žádné druhy zvýrazňování textu nebo změny barvy nebo typu písma.

Podstatný je také soubor `scripty.js`, který obsahuje většinu funkcí, které se starají o JavaScriptovou část aplikace. Tímto jsou myšleny především funkce pro kontrolování jednotlivých formulářů a dalších uživatelských vstupů.

Pro samotný vzhled rozhraní je důležitý soubor `main.css`. Ten obsahuje styl, jakým budou jednotlivé prvky vyobrazeny. Záměnou souboru lze bez jakékoliv úpravy funkčnosti zcela změnit vzhled IS.

6.1.3 Skripty nad databází

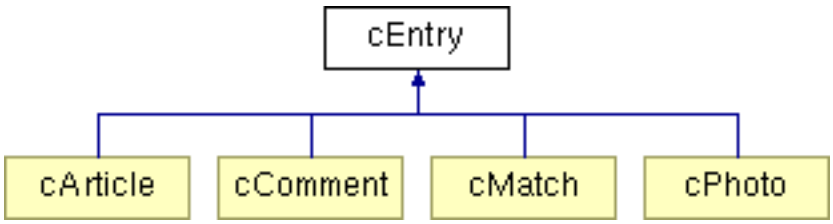
Celý systém má těžiště v databázi. Již dříve jsme si ukázali, jak struktura této databáze vznikla (viz. 5). Při zobrazování uživatelského rozhraní je třeba provádět také tyto skripty, které získávají data z databáze. Většina těchto dat je potřebná už pro samotné vytvoření ovládacích prvků systému.

Byla vytvořena knihovna `db.php`. Ta pracuje jako objekt *cDB* zapouzdřující práci s databází. V jejím samotném konstruktoru dochází k vytvoření spojení s databází. Toto spojení

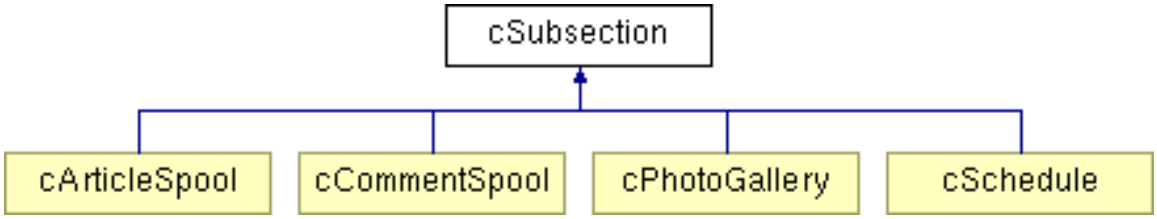
je poté použito vždy, když dojde k provádění nějakého SQL dotazu na databázi. K provádění dotazů slouží metoda *query*. Při selhání prováděného dotazu objekt vyvolá výjimku, která usnadňuje ladění aplikace.

Při návrhu a implementaci těchto skriptů jsem se snažil aplikovat na základní přístup k databázi postupy objektového návrhu a to nejen vytvořením objektové nadstavby nad databází. A proto vznikly abstraktní třídy *cEntry* a *cSubsection*. Tyto třídy ve svých konstruktorech provádějí akce společné pro všechny vstupy, respektive subsekce. Tyto třídy deklarovány také metody, které jsou poté definovány ve všech třídách, které od těchto abstraktních tříd dědí.

Dědění třídy *cEntry* je zobrazeno na obrázku 6.2 a následovníci třídy *cSubsection* jsou znázorněny na obrázku 6.3



Obrázek 6.2: Ukázka hierarchie objektů ve vztahu ke třídě cEntry



Obrázek 6.3: Ukázka hierarchie objektů ve vztahu ke třídě cSubsection

Zajímavá položka menu *Export databáze* souvisí jak s touto sekcí, tak i s předcházejícím uživatelským rozhraním. Tato položka nám dává možnost vyexportovat obsah databáze do souboru XML s charakteristickou strukturou, která je nastíněna v ukázce 6.4. Tento soubor se dá využít jako záloha databáze a to díky možnosti nepříliš složité transformace XML souboru do SQL skriptu, který by se dal nahrát zpět do databáze. Podstatnou podmínkou ale je, aby administrátor měl vytvořenu zálohu struktury databáze, protože transformace by byla schopna vytvořit ze struktury XML i strukturu tabulek, ale nebyla by schopna postihnout vazby mezi jednotlivými tabulkami. Tato záloha by mohla být prováděna i automatizovaně (např. pomocí nástroje *Cron*).

Další možností využití takto vyexportovaných součástí databáze by mohlo být vytvoření *RSS (Rich Site Summary) kanálu*. Tato technologie se dříve používala pro předávání aktuálních novinek mezi jednotlivými servery a až později se začala využívat i pro předávání aktualit uživatelům, kteří používají RSS čtečky a přihlásí se k odběru aktuálních informací z konkrétního serveru.

```
<?xml version="1.0" encoding="UTF-8"?>
<backup>
<users userID="9" role="admin" login="dz"
password="fc3a095ca67b2d6d892f7aca91b4d16064b13d2b"
key="1555763665" email="dz\%40dz.com" />
<entry entryID="190" user="9" subsection="4"
dateTime="2007-04-23+21\%3A2\%3A33" notice=""/>
<team teamID="7" teamName="Praha" stadium="" coach=""
category="starsiD"/>
<player playerID="12" name="říJi" surname="áNovk" birth="" club="3"
penalty="0" photo="" position="player" goalsF="0" goalsA="0"
gamesCount="0"/>
.
.
.
</backup>
```

Obrázek 6.4: Ukázka struktury exportovaného XML

6.1.4 Ostatní skripty

V této části si zmíníme práci ostatních skriptů, jenž by se daly označit v systému jako pomocné. Většinu těchto skriptů tvoří ošetření přechodu mezi jednotlivými stránkami a volání konstrukce objektu *cDisplay* a následné volání příslušné zobrazovací metody pro danou sekci.

Samotný objekt *cDisplay* by se dal v našem případě nazvat jako zobrazovací. A to na základě toho, že všechny součásti, které se snaží zobrazit nějakou část systému to činí skrze tento objekt. Objekt ve svém konstruktoru vytvoří hlavní kostru stránky (včetně menu, zápatí, atd.). Do této kostry je poté pomocí volání adekvátní zobrazovací metody vložen vlastní obsah stránky a prvky specifické pro každou konkrétní stránku.

Dalším typem pomocných skriptů jsou soubory obsahující různé pomocné funkce, respektive skripty zpracovávající výstupy ostatních, nad databází pracujících, algoritmů. Nejobsáhlejším souborem obsahujícím pomocné funkce je soubor `scripts.php`.

6.2 Shrnutí

V této kapitole jsme si ukázali řešení některých problémů, které nám při tvorbě informačního systému vyvstaly. Těchto problému bylo samozřejmě více, ale v této části jsme si zmínili jen ty, u nichž bylo zvoleno alespoň do určité míry zajímavé řešení z pohledu návrhu. Celkové hodnocení implementace bude rozebráno v závěru.

V této části bylo čerpáno z těchto zdrojů [8], [6], [7], [5], [3].

Kapitola 7

Závěr

V této části proběhne zhodnocení celého systému z několika pohledů a dojde také ke srovnání s obdobnými IS, které v současné době existují.

7.1 Splnění požadavků zadání

Požadavky uvedené v popisu aplikace 3.1 se podařilo v průběhu implementace úspěšně splnit.

Podařilo se vytvořit redakční systém, který umožňuje vkládání fotek, článků a komentářů k nim. Implementace zahrnuje zaznamenávání týmů, jejich soupisek, zápasů, jejich výsledků a také rozhodčích. Systém je také schopen automaticky vytvořit rozpis nasazení rozhodčích na zápasy.

Celý IS se bez problémů korektně zobrazuje v dnešní době v nejvíce rozšířených webových prohlížečích, jimiž jsou Internet Explorer 6, Opera 9.0 a vyšších, Mozilla Firefox 2.0 a vyšších.

7.2 Srovnání s podobnými systémy

Celkové srovnání s jinými webovými systémy je prakticky nemožné, protože webových systému existuje příliš velký počet, přičemž jejich kvalita se výrazně liší systém od systému. Ovšem podstatné klady vytvořeného systému jsou v tom, že je validní vzhledem k webovým standardům, snaží se udržovat dobrou přehlednost a ovladatelnost, atd.

Srovnání s jinými weby vodního póla v ČR je také poměrně komplikované a to z toho důvodu, že většina těchto webů je koncipována jako statické webové prezentace. Zbylých několik systému používá povětšinou nějaký druh univerzálních redakčních systémů. Oproti těmto univerzálním redakčním systémům náš poněkud zaostává právě kvůli jejich univerzálnosti a modularitě, kdežto náš systém je zaměřen příliš jednostranně.

7.3 Možnosti rozšíření

Systém poskytuje mnoho možností k jeho rozšíření. A to prakticky ve všech součástech. Je to dáno poměrně vydařeným návrhem databáze i aplikační vrstvy.

Systém je v porovnání s jinými redakčními systémy poměrně jednoduchý, ale pro potřeby našeho IS je zcela dostačující. Při jeho dalším rozvoji na základě existujícího databázového návrhu by se systém mohl stát dostatečně modifikovatelným a použitelným pro různé druhy

aplikací. Vylepšení redakčního systému by mohlo proběhnout rozšířením typů příspěvků a subsekcí a umožněním jejich komplikovanější vzájemné vazby.

Ve směru generování rozpisu zápasů by se dal zaměnit algoritmus, který zajišťuje přiřazování rozhodčích k zápasům za jiný, protože i zvolený algoritmus Backtrackingu má své jisté nevýhody (jako např. časovou složitost). Nahrazení by bylo možné např. pomocí algoritmu Forward-checking, ale vznikly by nám tak jiné problémy, které by se musely dále řešit.

Dalším možným rozšířením by se mohlo stát vytvoření propracovaného lexikálního analyzátoru, který by dokázal podle značek ve vstupním textu složitěji formátovat text článku, popřípadě by umožňoval vkládání odkazů a obrázků přímo do jednotlivých článků, atd.

Kromě možností úpravy funkčnosti je zde široké pole možností pro úpravy vzhledu systému, které je poskytováno použitím kaskádových stylů při původním vývoji.

7.4 Shrnutí

S použitím implementačních prostředků zmíněných výše (viz. 2) se podařilo vytvořit informační systém, který splňuje veškerá očekávání, která na něj byla v době formulace požadavků kladena. Systém má další možnosti rozvoje a v případě dalšího vývoje by se mohl stát jedním z nejlépe použitelných systému zabývajících se vodním pólem v ČR.

Literatura

[1] F. Zbořil jr. F. Zbořil. *Studijní opora předmětu IZU*. 2006.

[2] I. Rudolfová J. Zendulka. *Studijní opora předmětu IDS*. 2006.

[3] J. Kosek. *PHP - Tvorba interaktivních internetových aplikací*. GRADA Publishing, 1999. ISBN 80-7169-373-1.

[4] T. Müller. *Diplomová práce-Interaktivní tvorba rozvrhu*. 2001.

[5] WWW stránky. Otevřená encyklopedie. [online] `http://www.wikipedia.cz`.

[6] WWW stránky. Stránky o webových technologiích. [online] `http://www.interval.cz`.

[7] WWW stránky. Užitečné rady a pokyny při psaní webových prezentací. [online] `http://www.jakpsatweb.cz`.

[8] Z. Křivka T. Hruška. *Studijní opora předmětu IIS*. 2006.