Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Aplikace pro správu Univerzitní florbalové ligy

Plzeň, 2014 David Košek

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 6. května 2014

David Košek

Poděkování

Touto cestou bych rád poděkoval Ing. Ladislavu Lencovi za odborné vedení bakalářské práce, konzultace, cenné podněty a připomínky, které vedly ke zdárnému vypracování.

Abstrakt

Bakalářská práce se zabývá vytvořením informačního systému pro správu florbalového dění na Západočeské univerzitě. Součástí práce je analýza vhodných technologií pro vývoj webových stránek včetně databázového systému se zaměřením na open source produkty. Z analýzy přecházíme do praktické části, kde jako první navrhneme a vytvoříme databázový model. Poté je implementována webová aplikace včetně administrátorského prostředí. Po vytvoření výsledné aplikace ji otestujeme a zhodnotíme výsledky. V závěru shrneme práci a navrhneme možné rozšíření.

Abstract

The bachelor´s thesis focuses on the creation of an information system for the administration of University of West Bohemia floorball events. The analysis of suitable technologies for web application development is part of the thesis. Mainly the open source products are addressed. From the analysis we go on to practical part where at first a database model is designed and created. Then the web application including administration interface is implemented. After the application has been finished, it is tested and its results are evaluated. In the conclusion we summarize the thesis and we suggest possible extension.

Obsah

1 Úvod

Tato bakalářská práce se zabývá vývojem informačního systému pro správu Univerzitní florbalové ligy na Západočeské univerzitě. Univerzitní florbalová liga je soutěž amatérských týmů, která probíhá pod záštitou Katedry tělesné výchovy a sportu. Soutěže se účastní studenti a absolventi ZČU.

Cílem práce je navrhnout a implementovat informační systém pro správu florbalové ligy. Jednou částí systému bude prezentační část, která zpřístupní důležité informace účastníkům ligy i veřejnosti. Druhou částí bude sada nástrojů pro správu ligy, která bude sloužit administrátorovi. Systém bude navržen podle požadavků správce florbalu na KTS. Součástí práce je rovněž analýza a výběr vhodných prostředků pro tvorbu systému.

Motivací k této práci je především usnadnění a částečná automatizace úkonů spojených s organizací florbalové ligy, zaznamenáváním výsledků a prezentací důležitých statistik. KTS v současnosti nemá obdobný informační systém k dispozici a veškerá agenda je vedena v papírové podobě. Aplikace tedy bude přínosem hlavně pro správce florbalové ligy a také pro vedoucí jednotlivých týmů.

V následujících kapitolách se nejprve seznámíme s požadavky kladených na aplikaci a její funkcionalitu. Poté provedeme analýzu vhodných prostředků pro vytvoření aplikace. Jedno z těžišť této práce je návrh databázového modelu s popisem a zobrazením všech entit, vztahů i zvolených atributů. Na základě dosud získaných informací zdůvodníme výběr použitých prostředků. Hlavní těžiště této práce je v návrhu a implementaci webové aplikace, která bude splňovat veškeré požadavky formulované v seznamu požadavků. Nakonec zhodnotíme výsledky a shrneme práci.

Přílohou bakalářské práce je především uživatelská dokumentace k systému. Je přiloženo i kompletní schéma databázového modelu a také ukázka skriptu generující vybranou část databáze. Kompletní zdrojové kódy i elektronická podoba tohoto dokumentu je rovněž uvedena ve formě přílohy na CD-ROM.

2 Základní popis aplikace

Tato kapitola stručně popisuje základní požadavky na navrhovaný systém. Jejím záměrem je seznámení čtenáře s řešenou problematikou a formulace požadavků na jednotlivé části systému.

Webová aplikace bude rozdělena na tři hlavní části – veřejná část bude poskytovat všem návštěvníkům webu informace o florbalovém dění na ZČU[1]. Aministrátorská část bude sloužit pro správu stránek a univerzitní florbalové ligy. Poslední část bude také neveřejná a bude sloužit registrovaným uživatelům. Registrovaní uživatelé budou mít k dispozici rošířenou funkčnost.

Veřejná část bude zaměřena na prezentaci obecných informací a bude poskytovat i komunikační služby pomocí přístupu do veřejného fóra. Administrátorské prostředí bude přístupno pouze správci florbalu určeného KTS[2]. Bude nabízet snadné a přehledné vedení florbalové univerzitní ligy a možnost editace jednotlivých stránek pro veřejnou část. Poslední část pro registrované uživatele zpřístupní hráčům jednoduchý přehled o svém týmu a pro diskuzi vlastní týmové fórum. Rozhodčím pak umožní zapisovat se na zápasy, které jsou ochotni pískat a ani zde nebude chybět soukromé fórum pro rozhodčí.

Součástí požadavků na systém je nezávislá desktopová aplikace. Ta umožňuje zadávat správci informace i bez internetového přípojení. Desktopová aplikace simuluje světelnou tabuli v hale ukazující hlavně čas utkání a průběžné skóre.

2.1 Základní informace o florbalu na ZČU

Florbal je všeobecně populární sport a čím dál více se rozšiřuje. Na Západočeské univerzitě se mu věnuje nejen spoustu studentů, ale také zaměstnanců. To vedoucího florbalu na univerzitě vede k založení turnajů jak pro studenty, tak pro zaměstnance. Reprezentační univerzitní tým složený ze studentů každoročně reprezentuje Západočeskou univerzitu na ČAH[3], aby poměřily své síly s ostatními týmy v České Republice.

[1] ZČU – University of West Bohemia in Pilsen
[2] KTS – Katedra tělesné výchovy a sportu
[3] ČAH – České akademické hry

2.1.1 Základní informace o UFL

Univerzitní florbalovou ligu mohou hrát studenti a absolventi všech studijních programů (bakalářský, magisterský, doktorandský) veřejných a soukromých vysokých škol ČR a jejich přátelé bez rozdílu věku. UFL[4] se hraje podle pravidel Mezinárodní florbalové federace IFF[5]. Hrací systém se může měnit podle přihlášených týmů a vyřazovací část (play – off) není povinná.

2.2 Požadavky na aplikaci

Jedním z hlavních požadavků je rozšíření informací o florbalovém dění na univerzitě pro studenty, ale i zaměstnance. Uživatel bude moci zaregistrovat svůj tým pomocí připraveného formuláře. Přihlášení do systému mu bude povoleno, až správce potvrdí jeho účast. Poté bude moci, vedoucí týmu přidat hráče na soupisku svého týmu. Bude i možnost registrace rozhodčích. Aplikace bude schopna zobrazovat různé statistické údaje, jako například kanadské bodování hráčů seřazené podle bodů od nejvyššího. Po každém zápase se budou uchovávat kompletní informace tj. hlavně datum, kdy se zápas odehrál, kdo proti komu nastoupil a konečný výsledek. Aplikace bude schopna odesílat emailové a textové zprávy hromadně všem uživatelům, či jednotlivcům. Velký důraz bude kladen na rozlosování zápasů ligy do jednotlivých kol podle přihlášených týmů a zadání časového rozmezí včetně vynechání zimního zkouškového období. Pro přehlednost jsou zbylé požadavky na aplikaci dále rozděleny do menších celků. Z požadavků na systém a z jednotlivých celků budeme vycházet při návrhu databázového modelu.

2.2.1 Veřejná informační část

Informační část webu představují zejména texty a články o univerzitní lize, ale také o celkovém florbalovém dění. Tato část bude poskytována všem, kteří web navštíví. Jedná se především o informace ke hraní UFL, kde se dočteme všechny potřebné informace, jak se do ligy přihlásit, a co všechno je nutné splnit. Dozvíme se i, jak probíhá aktuální sezóna, a jak si který tým i hráč vede v bodování. V případě zájmu o hraní ligy, ale nedostatku hráčů pro za-

[4]UFL – University floorball league
[5]IFF – International Floorball Federation

ložení týmu bude zde i výpis vedoucích přihlášených týmu s jejich kontakty. Je zde uvedeno, co je nutné k tomu stát se rozhodčím. Dále budou k dispozici informace o reprezentaci ZČU, kde se dozvíme všechny potřebné informace o tom, jak tým reprezentuje svojí univerzitu, a jak je možné se do něj dostat. Dále budou zveřejněny i florbalové turnaje, které KTS pořádá, jak pro studenty, tak pro zaměstnance se všemi potřebnými informacemi. Přístup na veřejné fórum má každý a každý zde může zanechat vzkaz. Ve veřejné části se budou nacházet i soubory ke stažení, kde mohou být různé soubory, které administrátor určí. Pro publikaci článků bude nutné kontaktovat správce. Ve veřejné pasáži nebudou chybět ani kontakty na vedoucího florbalu na ZČU. Shrnutí požadavků na informační část:

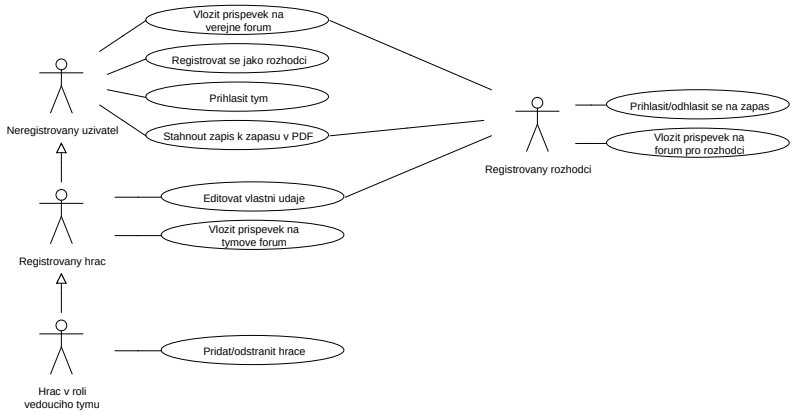
1. Vhodná a přehledná prezentace článků a textů (stránek).
2. Vhodné zobrazení struktury všech bodování.
3. Dokázat rozlišit turnaje studentů od turnajů zaměstnanců.
4. Vytvoření formulářů pro registraci týmu/rozhodčího.
5. Vytvoření jednoduchého veřejného fóra.
6. Možnost stažení souboru/zápisu k zápasu.

2.2.2 Část pro registrované

Tato část webu není veřejná. Je nutné přihlášení pomocí jména a hesla. Aplikace bude schopna sama rozpoznat, zda se jedná o hráče, nebo rozhodčího a podle toho nastaví uživateli práva. Hráči se zobrazí stručný statistický přehled probíhající sezóny. Zde uvidí kanadské bodování v týmu, statistiky brankářů týmu, zápasy odehrané a ještě ty, co tým v sezóně čeká odehrát. Hráč bude mít přístup do týmového fóra, které je soukromé. Každý tým má své vlastní. Pokud je hráč navíc vedoucí týmu, má možnost přidávat hráče do týmu, ale i je mazat. Pokud je hráč zároveň i rozhodčí, tak má práva obou uživatelů. Rozhodčí má přístup ke všem zápasům, které se budou ještě hrát, ale s možností se přihlásit na zápas, který je ochoten pískat. Samozřejmostí je, že rozhodčí mají také své vlastní fórum pro snazší komunikaci. V případě jakéhokoliv problému bude zde připraven výpis všech rozhodčích s jejich kontakty. Všichni přihlášení uživatelé bez rozdílu funkce mají možnost editovat své údaje vyplněné při registraci zejména přihlašovací jméno, heslo

a kontakty. Shrnutí požadavků jsem zobrazil pomocí digramu použití i s poznatky z předchozí části (Obr. 2.1).

Diagram použití zobrazující role uživatelů kromě administrátora.



Obrázek 2.1: Diagram použití uživatelů.

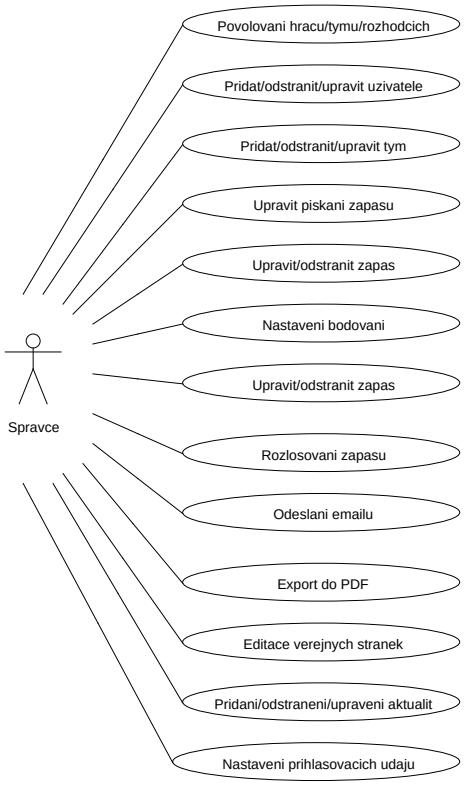
2.2.3 Administrátorská část

Do této části aplikace se dostane pouze správce florbalu určený KTS pomocí zadání uživatelského jména a hesla. Po přihlášení čeká administrátora rychlý přehled probíhající sezóny. Správce bude mít přehled o všech uživatelích rozdělené podle funkcionality (hráč nebo rozhodčí) a bude moci editovat jejich údaje, popřípadě je smazat ze systému. Samozřejmě bude moci uživatele i přidávat na soupisku přihlášených týmů. Dále bude správci zpřístupněno nastavování kanadských bodů včetně statistik brankářů a nastavení bodů týmu za odehrané zápasy. Další schopností administrátora bude úprava rozhodčích k jednotlivým zápasům, to znamená, že bude moci editovat, kdo bude pískat daný zápas. Správce bude schopen odesílat emailové nebo textové zprávy. V neposlední řadě bude pro administrátora připraven export souborů do PDF[6]. Správce bude pověřen rozlosováním zápasů pro nadcházející sezónu. Jeden z hlavních požadavků pro administrátora bude změna textů a článků ve veřejné části. Stávající texty bude moci editovat, články bude schopen přidávat, mazat či editovat. Správce, stejně jako ostatní uži-

[6]PDF – Portable Document Format

vatelé, bude moci měnit své přihlašovací údaje. Pro shrnutí požadavků jsem využil opět digram použití (Obr. 2.2).

Diagram použití zobrazující možnosti administrátora.



Obrázek 2.2: Diagram použití administrátora.

2.2.4 Export souborů do PDF

Export souborů budou využívat hlavně hráči a administrátor. Hráči si budou moci vyexportovat částečně předvyplněný zápis k utkání. V zápise budou předvyplněné soupisky obou mužstev, která mají proti sobě nastoupit. Dále názvy obou týmů rozřazené do kolonek domácí a hosté. Pokud budou k utkání přihlášeni i rozhodčí, nebudou chybět ani jejich jména. V poslední řadě budou předvyplněné zbylé formality, jako jsou číslo utkání, datum utkání, čas utkání a pořádající tým. Druhý styl exportu bude využívat pouze administrá-

tor, jedná se především o stručný seznam všech registrovaných hráčů hrající univerzitní ligu. Dále všechny statistické tabulky (kanadské bodování, statistiky brankářů, celkový pohled na tabulku), seznam rozhodčích a podrobnější soupisky jednotlivých týmů.

2.2.5 Rozlosování zápasů

Rozlosování zápasů na novou sezónu bude mít na starost administrátor. Zde bude třeba zadat datum začátku sezóny (první zápas), jako další datum bude zadání zimního zkouškového období, jelikož se přes něj liga nehraje a poslední datum zápasu (konec sezóny). Hrací dny budou pouze pondělí a středa vždy od 19:40 první zápas a od 20:25 druhý zápas v hrací den. Systém si sám zjistí počet týmů a podle zadaného časového období rozvrhne zápasy pro celou sezónu. Před konečným potvrzením si administrátor může data zápasů upravit podle svého. Důležité při potvrzení rozlosování je, aby se automaticky doplnily pořádající týmy, to znamená, že první zápas v hrací den pořádá domácí tým z druhého zápasu a analogicky druhý zápas pořádá domácí tým z prvního zápasu, vždy je pořádající domácí tým. Shrnutí požadavků na rozlosování:

1. Rozlosování zápasů pro 4 – 10 týmů.
2. Možnost upravení dat.
3. Zadání libovolného časového období.

2.2.6 Desktopová aplikace

Tato aplikace se bude používat v případě nefunkční světelné informační tabule v hale KTS. Ta zobrazuje aktuální čas zápasu, góly domácích, hostů a aktuální třetinu. Aplikace by měla ukazovat stejné hodnoty.

3 Analýza a problematika prostředků pro tvorbu aplikace

V této kapitole si řekneme, jaké jsou vhodné prostředky pro řešení aplikace obdobného rozsahu. Zmíníme i možnosti, kde by mohly nastat potenciální problémy.

3.1 Programovací jazyky

K programování webových aplikací se nejčastěji používá značkovací HTML[1] jazyk. S tímto jazykem se dají vytvořit statické webové stránky. Značkovacím jazykem se nazývá proto, že začíná znakem (např.) a končí také znakem, před kterým je navíc lomítko (např.).

Pro naformátování stylu webové stránky (barva, umístění textu, písmo...) se používá CSS[2] styly neboli kaskádové styly. Lze jich mít na webu více, ale lze mít i pouze jeden a formátovat tím celý web. Kaskádový styl je vytvořený ze selektorů a deklarace. Selektor určuje, na které HTML prvky stránky se pravidlo vztahuje.

Proč se říká kaskádové styly? „Kaskáda určuje způsob aplikování pravidel. Uživatelské styly potlačí všechny ostatní styly, individuální styly potlačí vložené, připojené a importované styly. Vložené styly mají přednost před připojenými styly. Připojené a importované styly jsou rovnocenné a aplikují se všude, kde nejsou použity jiné druhy stylů. Styly prohlížeče se používají v případě, že nejsou zadány jiné styly.“ [5]

Pro oživení webových stránek se používá JavaScript. Příkladem použití může být například rozbalovací menu, měnící se obrázky a další. JavaScript je závislý na prohlížeči hlavně z důvodu, že si ho uživatel může vypnout, navíc v různých verzích nemusí pracovat správně.

Dále se pro programování webových stránek používá jazyk PHP[3]. S tímto jazykem nabývá web dynamičnosti. Podstatnou vlastností tohoto jazyka je, že

[1] HTML – HyperText Markup Language
[2] CSS – Cascading Style Sheets
[3] PHP – Hyperrtext Preprocessor

neběží na straně klienta, ale na straně serveru, na rozdíl od JavaScriptu. Pomocí PHP můžeme ukládat či měnit data na webu, slouží také pro komunikaci s databází. Další výhodou PHP je, že zdrojový kód nelze zobrazit jako například HTML. Kromě tohoto jazyke lze použít i jiné programovací jazyky (např. Java, C#, Python).

Zatím lze zmíněné jazyky libovolně kombinovat. Zásadní změna je při použití webového frameworku ASP.NET[4]. Při použití tohoto frameworku ho nelze využívat s konkurenčním PHP. ASP.NET je množství knihoven pro vývoj webových aplikací v jazyce C#[5]. Tento jazyk jako jazyk PHP běží na straně serveru a umožňuje také komunikaci s databází. Hlavním rozdílem je, že PHP je dynamický jazyk C# je silně typovaný.

„ASP je vyvíjeno Microsoftem, oproti tomu PHP je opensource – tj. zdrojové kódy jsou každému k dispozici a každý pokročilý programátor si jej může dle libosti upravit. Jazyk se díky tomu může snadno dál rozvíjet.“ [1]

Projekt využívá databázi, proto dalším jazykem, který použijeme je SQL[6]. Tento jazyk je standardem pro komunikaci s databází, často se mu také říká dotazovací (komunikační) jazyk. Syntaxe jazyku je podobná normálnímu jazyku (angličtině).

3.2 Hosting a doména

Jelikož budeme chtít stránky umístit na internet, budeme potřebovat webhosting, který se stává nedílnou součástí této práce. V dnešní době je mnoho firem poskytujících tuto službu. Nabídky se liší v nabízených technologiích bezpečnostních prvků, e-mailových schránek a také podpoře různých datových úložišť (MySQL, PostreSQL). Podle nabízených technologií se odvíjí i cena za poskytnutí hostingu. Součástí každého hostingu je doména, což je vlastně adresa internetových stránek. Doména je pevně definovaná a při jejím zadávání se nerozlišují malá a velká písmena, lze použít i číslice a pomlčky.

[4] ASP.NET – Active Server Pages .NET
[5] C# – C sharp
[6] SQL – Structured Query Language

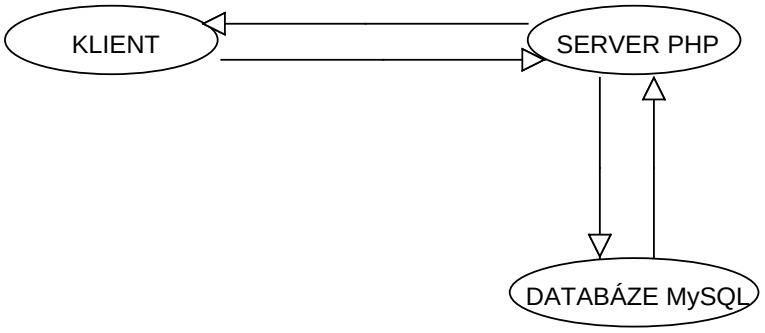
3.3 Databázový systém

Webové aplikace využívají různá datová úložiště. Pro tento projekt jsou vhodnými kandidáty databázové systémy MySQL a PosgreSQL.

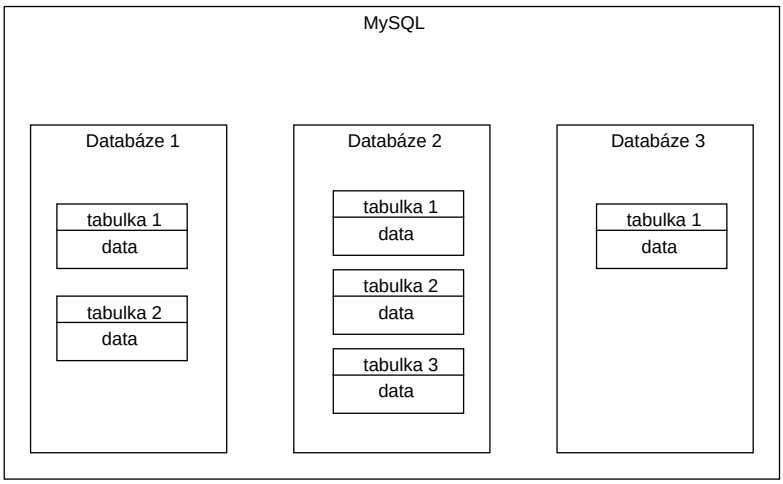
MySQL

MySQL je relační databázový systém, který komunikuje jazykem SQL zmíněným výše. Tento databázový systém je opensource, a je tedy volně šiřitelný. MySQL je více využíván u webových aplikací z důvodu jeho podpory při poskytování hostingu. Pro správu databázového systému MySQL je vhodný phpMyAdmin, což je pokročilý nástroj pro správu systému přes webové rozhraní. MySQL se nejčastěji využívá s PHP jazykem, který umožňuje přístup k datům (Obr. 3.1). Jak jsem zmínil, jde o relační databázi, což se dá vysvětlit tak, že databáze je založená na tabulkách (Obr. 3.2). Každá tabulka obsahuje sloupce nazývané atributy a hodnoty jsou ukládány do řádku. Tato databáze je typovaná, to znamená, že každý sloupec má určený datový typ.

Ukázka komunikace mezi klientem a databází.



Obrázek 3.1: Model komunikace klient – databáze



Obrázek 3.2: Ukázka struktury MySQL

PostgreSQL

PostgreSQL je relační databázový systém. Stejně jako MySQL jde o volně šiřitelný systém, zdrojové kódy jsou veřejné. Jedná se o multiplatformní databázový systém použitelný pod systémy Linux i Windows. Databázový systém má za sebou značný vývoj a i díky tomu patří ke špičce v oboru. PostgreSQL nabízí více datových typů než zmíněný systém, nicméně nepodporuje čísla bez znaménka.

„MySQL a PostgreSQL jsou dvě různé databáze s rozdílnou filozofií a samozřejmě i rozdílným uživatelským rozhraním. Příklad rozdílné funkcionality mezi MySQL a PostgreSQL

```
-Číslo zadané jako řetězecn na MySQL projde na PostgreSQL ne:
SELECT 1+1 AS val;
-> MySql: 2 PostgreSql: 2
SELECT 1+'1' AS val;
-> MySql: 2 PostgreSql: chyba
SELECT 1+CAST('1' AS INT) AS val;
-> MySql: 2 PostgreSql: 2“ [12]
```

3.4 Architektonické vzory

Pro vývoj aplikací lze využít různé návrhové vzory. Výhodou návrhových vzorů je logické rozčlenění aplikace a přehlednější kód. My si zde popíšeme nejčastější případ, se kterým se setkáváme u webových technologií. Je jím MVC[7] model, který srovnáme s jeho velkým bratrem MVP[8] modelem.

MVC

Nyní si popíšeme, jak vypadají komponenty M, V a C v prostředí webu. Pro lepší pochopení a představu je MVC model znázorněn na obrázku níže (obr. 3.3).

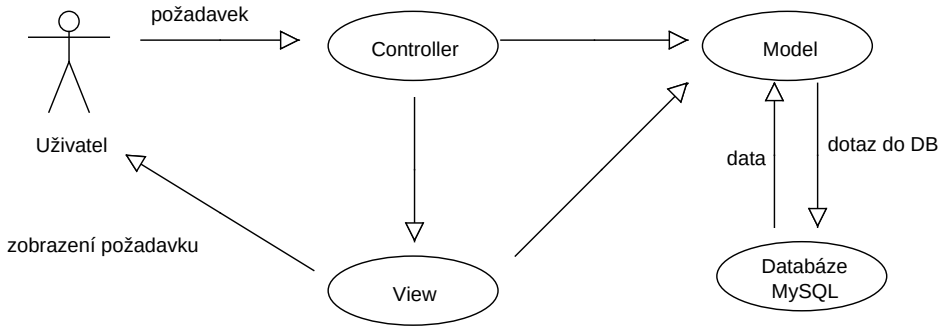
Model
Model obsahuje logiku. Slouží pro komunikaci s databází, mohou zde být výpočty a další. Funkcí modelu je přijetí parametrů, přičemž nezáleží na tom, odkud jsou. Jeho výstupem jsou výstupní data, přičemž netuší, jak budou data zformátována a zobrazena. Model nemá se zobrazením nic společného. [10] [11]

View (pohled)
Jak název napovídá, funkcí této části je zobrazení výstupu uživateli a obsahuje minimum logiky. S větší pravděpodobností je výstupem vyrenderování HTML, ale v některých případech to může být i jiný formát například XML[9]. Tak jako Model, View netuší, odkud mu data přišla. [10] [11]

Controller
Tato komponenta propojuje celou architekturu, Controller se občas považuje za prostředníka, se kterým komunikují všechny komponenty včetně uživatele. Controller se nejčastěji skládá ze dvou částí, první je Front Controller, občas nazývaný Router, který zavolá příslušný Controller na základě parametrů. Volaný Controller podle parametrů pozná, co se po něm žádá. [10] [11]

[7] MVC – Model, View, Controller
[8] MVP – Model , View, Presenter
[9] XLM – Extensible Markup Language – obecný značkovací jazyk.

Zobrazení MVC architektury včetně databáze a uživatele, místo uživatele si lze představit prohlížeč.



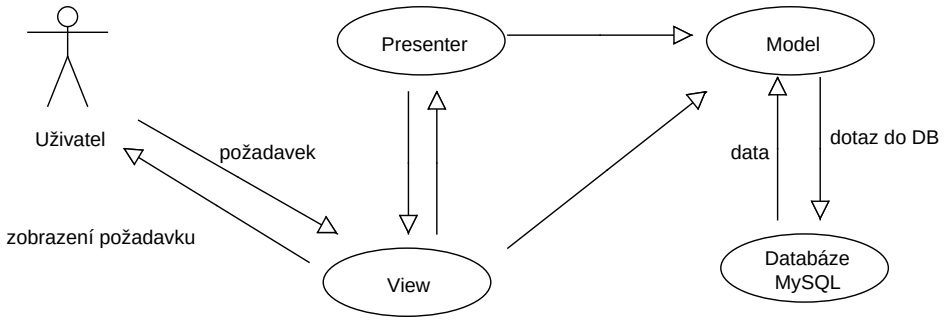
Obrázek 3.3: Schéma MVC architektury.

MVP

Tento model stručně srovnáme s předešlým modelem a pro představu ho opět znázorníme pomocí schématu (obr. 3.4).

Změn oproti MVC je poměrně dost. Uživatel nyní komunikuje pouze s View. Z toho vyplývá, že zpracovává vstup i výstup na rozdíl od MVC, kde vstup zpracovával Controller. View nemá žádnou aplikační logiku, pouze volá příslušné metody v Presenteru na základě vstupu. Presenter už obsahuje aplikační i prezenční logiku. Presenter má vazbu s Modelem, což mu zajišťuje určitou manipulaci, čímž zajistí aktualizaci ve View, nebo přímo sám ovlivňuje View. MVP se dá dále rozdělit podle zodpovědnosti, jakou nese, na dvě variace. První se nazývá Supervising Controller a druhá Passive View. [10] [11]

Zobrazení MVP architektury včetně databáze a uživatele, místo uživatele si lze představit prohlížeč.



Obrázek 3.4: Schéma MVP architektury.

3.5 Knihovny a nástroje

V této části popíšeme knihovny a nástroje, které nám budou užitečné při realizaci práce.

3.5.1 Knihovna DIBI

DIBI je knihovna pro komunikaci s databází. Podporuje mnoho databázových systémů i dva vhodné kandidáty zmíněné v kapitole 3.3. Jedná se o jednoduchou knihovnu pro elegantní zápis příkazů. DIBI lze svobodně používat i v komerčních projektech. Jejím cílem je maximálně ulehčit práci programátorům (snadný přístup k metodám, zjednodušit zápis SQL příkazů...), eliminovat výskyt chyb, přenositelnost mezi databázovými systémy a zachovat maximální jednoduchost. Třída dibi má jednu výhodu, kvůli které ji využívám, a to, že je vždy po ruce. Nemusím získávat instanci připojení. [8]

3.5.2 Knihovny pro export PDF souborů

Aplikace má být schopna generovat PDF soubory. K tomu je často využívána některá z knihoven pro generování PDF dokumentů. Při využití PHP

je jedna z nejznámějších knihoven mPDF, samozřejmě existují i jiné, například TCPDF knihovna. Při využití ASP.NET to je iTextSharp knihovna.

mPDF

Tato knihovna je volně použitelná. Knihovna vytváří PDF soubory v UTF–8 kódování, tedy včetně českých znaků. Funguje na principu, že vezme HTML kód, popřípadě i CSS, a vytvoří z něj PDF soubor. Dá se říct, že vytiskne do souboru naprogramovanou stránku v HTML.

iTextSharp

Tato knihovna umožňuje vytvářet PDF soubory na základě dat z XML nebo databáze. Knihovna je open source, tudíž je volně použitelná.

3.5.3 HTML editor

Jedním z požadavků na aplikaci je úprava stávajících HTML stránek ve veřejné části a možnost přidání či úprava článků. HTML editory se dají rozdělit na textové, objektové a WYSIWYG[10]. Pro náš účel využijeme WYSIWYG editor, protože nevyžaduje žádnou znalost HTML jazyka. Těchto editorů existuje nespočetné množství, my si zde popíšeme pouze jeden, který později využijeme v implementaci.

CLEditor

Editor je přehledný a jeho největší výhodou oproti ostatním HTML editorům je jeho velikost, kterou zabírá. Mezi další výhody patří jeho rychlost načítání a snadné používání. Tento nástroj je vydáván pod svobodnou licencí a nebrání nám v jejím využití. Editor je postaven nad knihovnou jQuery. [9]

[10]WYSIWYG – What You See Is What You Get

3.6 Odesílání zpráv

Z požadavků vyplývá, že administrátor může odesílat e-mailové a textové zprávy. Zde by se mohly vyskytnout problémy při realizaci systému. Odeslat e-mailovou zprávu lze pomocí funkce v PHP jazyce, pokud to poskytovatel umožňuje. V opačném případě je nutné si napsat vlastního SMTP[11] klienta a následně využít funkce PHP pro odeslání.

Pro odesílání textových zpráv se využívá tzv. SMS brána. Dříve s tím nebyl takový problém jako dnes. Princip byl v tom, že se zpráva odeslala jako e-mail, kde se část adresy před zavináčem vyplnila telefonním číslem a část za zavináčem byl název operátora. Někteří čeští operátoři tuto možnost nabízejí i dnes, někteří částečně a někteří vůbec. Řešením tohoto problému může být placená SMS brána, kde se platí za každou odeslanou zprávu. Po konzultaci se zadavatelem jsme tuto možnost vyřadili. Odesílání textových zpráv jen části uživatelům nemá smysl.

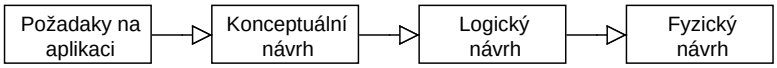
[11] SMTP – Simple Mail transfer Protocol

4 Návrh databázového modelu

Základem webové aplikace je správný návrh databáze. Její struktura je popsána pomocí datového modelu, který obsahuje kompletní definici všech uvažovaných entit, atributů a relací vycházejících z požadavků na aplikaci. Z výsledného modelu lze již přímo vygenerovat prázdnou databázi pro florbalové dění na ZČU, kterou je poté možno plnit daty. V této kapitole se tedy seznámíme s návrhem databáze na požadovanou aplikaci.

Při vytváření datového modelu jsem se snažil co nejvíce vycházet z požadavků na aplikaci. Cílem je navrhnout kvalitní datovou strukturu pro konkrétní aplikaci a databázový systém, který bude tato aplikace využívat k uložení dat.

Standardní fáze návrhnu databáze.



Obrázek 4.1: Model návrhu databáze

4.1 Konceptuální návrh schématu

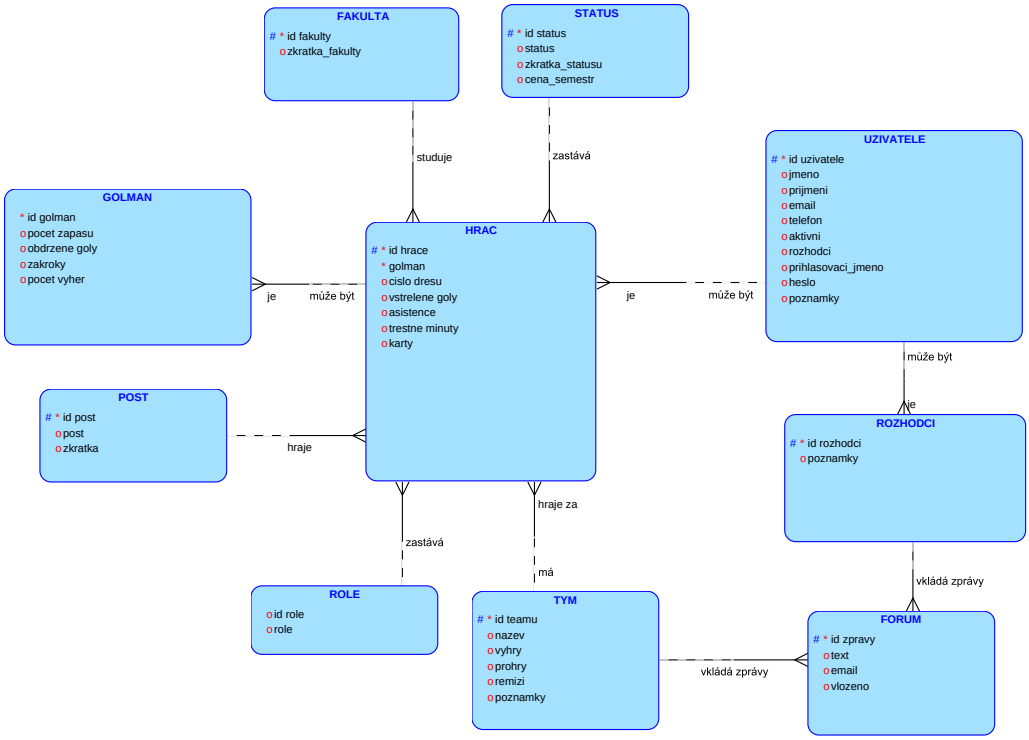
V této části si řekneme, co je obsahem databáze. Nebereme v úvahu pozdější implementaci ani použitý databázový systém. Schéma si pro usnadnění rozdělíme do více úrovní podle funkcionality, ze kterých pak vznikne výsledný databázový model. Složené schéma nalezneme v Příloze A na obr. 7.1.

„Zadání často explicitně neříká, co má systém obsahovat, ale spíše to, co od něj klient očekává nebo jak by měl fungovat. Naším úkolem bude v zadání identifikovat tzv. základní entity. Jedná se o prvky, které jsou pro nás zajímavé a o kterých má z databázového hlediska smysl uchovávat data. Model, ve kterém jsou vyobrazeny tyto entity a vztahy mezi nimi, se nazývá konceptuální.“ [4]

4.1.1 Správa uživatelů

Část pro správu uživatelů slouží především k tomu, aby bylo zřejmé, jakou funkci daný uživatel zastává. Z požadavků plyne, že uživatel může být hráč nebo rozhodčí, či oboje, proto jsem se rozhodl udělat nadřazenou strukturu, abych zamezil duplicitě dat (např. je zbytečné mít jméno a příjmení u rozhodčího nebo u hráče, když jde o jednoho a toho samého uživatele).

V tuto chvíli máme dostatek informací pro vytvoření potřebných entit, které jsou nutné pro aplikační funkcionalitu této části. Na následujícím obrázku jsou zobrazeny všechny entity včetně jejich vztahů a atributů sloužící pro uchování informací o uživateli.

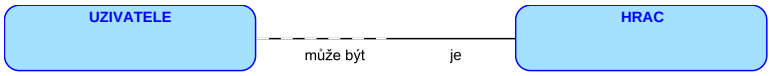


Obrázek 4.2: Konceptuální schéma části pro rozdělení uživatelů.

Vztahy mezi entitami

Po vytvoření entit v předchozí části si popíšeme jednotlivé vztahy. Pro přehlednost si definujeme strukturu popisu, která bude vypadat následovně, a budeme ji využívat i u ostatních částí. V první řadě bude slovní prohlášení, poté následuje schéma a nakonec bližší popis.

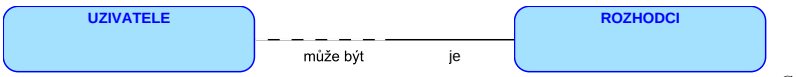
Jeden uživatel může být jeden hráč. Hráč je právě jeden uživatel.



Obrázek 4.3: Vztah UZIVATELE – HRAC

Uživatel je nadřazená struktura pro registrované osoby. Slouží k rozdělení na hráče nebo rozhodčí. Uživatel může být hráčem i rozhodčím zároveň.

Jeden uživatel může být jeden rozhodčí. Rozhodčí je právě jeden uživatel.

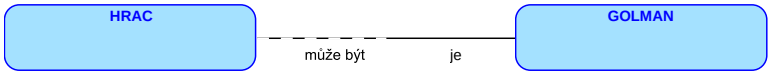


S

Obrázek 4.4: Vztah UZIVATELE – ROZHODCI

Popis této vazby je stejný jako popis vztahu UZIVATELE – HRAC.

Jeden hráč může být brankářem. Brankář je jeden hráč.



Obrázek 4.5: Vztah HRAC – GOLMAN

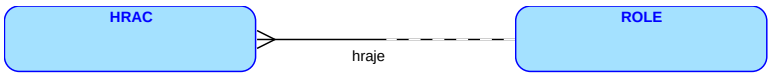
Tato vazba vznikla z důvodu toho, že uživatel může být brankářem i hráčem zároveň. Pravidla nezakazují, aby hráč jednou hrál v roli útočníka a podruhé nastoupil do brány jako brankář. V případě, že by se toto pravidlo zakázalo, tak by se uživatel dělil ještě na brankáře.

Jeden tým má více hráčů. Více hráčů hraje za jeden tým.

Obrázek 4.6: Vztah HRAC – TYM

Každý hráč hraje právě za jeden určitý tým. Hráč nemůže hrát za více týmů zároveň. Každý tým může mít libovolný počet hráčů.

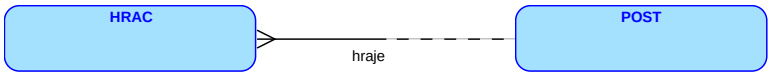
Více hráčů zastává jednu roli. Jednu roli může zastávat více hráčů.



Obrázek 4.7: Vztah HRAC – ROLE

Každý hráč zastává určitou roli v týmu (např. vedoucí týmu), tato vazba nám přiřazuje ke každému hráči jeho roli. Každý hráč může zastávat pouze jednu roli.

Jeden post může hrát více hráčů. Více hráčů hraje jeden post.



Obrázek 4.8: Vztah HRAC – POST

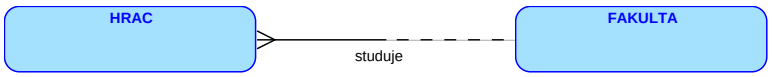
Každý hráč má určitý post (např. útočník). Pomocí této vazby rozpoznáme posty u jednotlivých hráčů. Každý hráč může hrát oficiálně jeden post, nýbrž v zápase se nekontroluje, zda je hráč na daném postu.

Více hráčů má jeden status. Jeden status může mít více hráčů.

Obrázek 4.9: Vztah HRAC – STATUS

Každý hráč má svůj určitý status (např. zda je student zapsaný ve stagu). Díky této vazbě víme, jaký má hráč status. Každý hráč může mít pouze jeden status.

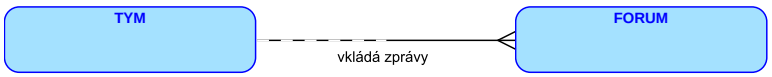
Jedna fakulta má více hráčů. Více hráčů může studovat jednu fakultu.



Obrázek 4.10: Vztah HRAC – FAKULTA

Hráči studují různé fakulty a tato vazba nám přiřazuje ke každému jednu z předdefinovaných fakult. Hráč může mít přiřazenou pouze jednu fakultu, i když ve skutečnosti může studovat více fakult. Je na uživateli, kterou fakultu zvolí.

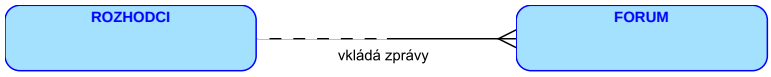
Jeden tým může vložit více příspěvků. Více příspěvků může být od jednoho týmu.



Obrázek 4.11: Vztah TYM – FORUM

Každý tým má své soukormé fórum pro komunikaci mezi hráči. Hráči nejsou omezeni v počtu vložených zpráv. Díky této vazbě dokážeme určit tým, který vložil příspěvek.

Jeden rozhodčí může vložit více zpráv. Více zpráv může být od jednoho rozhodčího



.

Obrázek 4.12: Vztah ROZHODCI – FORUM

Rozhodčí mají pro lepší komunikaci svoje soukromé fórum. Díky této vazbě dokážeme určit, které zprávy vložili rozhodčí.

Atributy jednotlivých entit

V této části si popíšeme jednotlivé atributy z navrhnutých entit. Všechny atributy vycházejí z požadavků na aplikaci. Stejně jako u vztahů mezi entitami i zde si definujeme určitou strukturu, kterou budeme dodržovat. Nejprve si o určité entitě řekneme, k čemu slouží, a poté popíšeme jednotlivé atributy entity včetně datových typů. V případě jedná-li se o primární klíč, bude za datovým typem zkratka PK, jedná-li se o cizí klíč, bude zkratka FK. Nakonec bude následovat stručný popis konkrétního atributu. Atributy konkrétní entity budou vyznačovány bezpatkovým písmem, `datové typy` budou vyznačovány strojovým písmem. Název ENTITY bude označen velkým písmem a kapitálkami.

UZIVATELE
Tato entita slouží pro uchování informací o uživatelích, jde především o jméno a příjmení, kontaktní údaje ve formě telefonního čísla a emailové adresy, přihlašovací údaje a případný komentář. Každý, kdo se registruje a zastává činnost rozhodčího nebo hráče má v této tabulce záznam.

Název atributu	**Datový typ**	**Popis**
id_uzivatele	`INTEGER`, PK	Identifikační číslo uživatele
jmeno	`VARCHAR(32)`	Jméno uživatele
prijmeni	`VARCHAR(32)`	Příjmení uživatele
email	`VARCHAR(64)`	Emailová adresa uživatele
telefon	`VARCHAR(32)`	Telefonní číslo na uživatele
aktivni	`BOOLEAN`	Identifikátor, zda je uživatel aktivní
rozhodci	`BOOLEAN`	Identifikátor, zda je uživatel rozhodčí
prihlasovaci_jmeno	`VARCHAR(32)`	Přihlašovací jméno uživatele do systému
heslo	`VARCHAR(16)`	Heslo k přihlášení uživatele
poznamky	`TEXT`	Místo pro případné poznámky k uživateli

Tabulka 4.1: Atributy entity UZIVATELE

ROZHODCI
Zde si uchováváme všechny rozhodčí, které se přihlašují na zápasy, které chtějí pískat. Rozeznáváme je podle jejich identifikátorů.

Název atributu	**Datový typ**	**Popis**
id_rozhodci	`INTEGER`, PK	Identifikační číslo rozhodčího
id_uzivatele	`INTEGER`, FK	Identifikační číslo uživatele
poznamky	`TEXT`, FK	Místo pro případné poznámky k rozhodčímu

Tabulka 4.2: Atributy entity ROZHODCI

HRAC
Entita HRAC uchovává určité informace o hráči, jedna z nejdůležitějších je, za jakým tým hráč hraje, poté jeho bodování. Nechybí zde ani informace například o tom, co hráč studuje za fakultu nebo jaká je jeho role v týmu.

Název atributu	**Datový typ**	**Popis**
id_hrace	INTEGER, PK	Identifikační číslo hráče
id_uzivatele	INTEGER, FK	Identifikační číslo uživatele
id_tymu	INTEGER, FK	Identifikační číslo týmu, za který hráč hraje
id_status	INTEGER, FK	Identifikační číslo určující status hráče
id_fakulty	INTEGER, FK	Identifikační číslo fakulty hráče, kterou hraje
id_post	INTEGER, FK	Identifikační číslo postu, který hráč zastává
id_role	INTEGER, FK	Identifikační číslo role, kterou hráč zastává
golman	BOOLEAN	Identifikátor, zda je hráč golman
cislo_dresu	INTEGER	Číslo dresu s kterým hráč hraje
vstrelene_goly	INTEGER	Počet vstřelených gólů
asistence	INTEGER	Počet asistencí
trestne_minuty	INTEGER	Počet trestných minut
karty	INTEGER	Počet červených karet

Tabulka 4.3: Atributy entity HRAC

GOLMAN
Zde jsou ukládány všechny potřebné informace pro vedení statistik brankářů, ze kterých lze vypočítat např. procentuální úspěšnost a další.

Název atributu	**Datový typ**	**Popis**
id_golman	INTEGER, PK	Identifikační číslo brankáře
id_uzivatele	INTEGER, FK	Identifikační číslo uživatele
pocet_zapasu	INTEGER	Počet odchytaných zápasů
obdrzene_goly	INTEGER	Počet obdržených branek
zakroky	INTEGER	Celkový počet zákroků
pocet_vyher	INTEGER	Počet výher

Tabulka 4.4: Atributy entity GOLMAN

TYM
Entita pro správu týmu, zde si uchováváme potřebné informace k jednotlivým týmům pro vyhodnocení tabulky. Zde je uchovává i název týmu.

Název atributu	**Datový typ**	**Popis**
id_tymu	`INTEGER`, PK	Identifikační číslo týmu
nazev	`VARCHR(32)`, FK	Název týmu
vyhry	`INTEGER`	Počet výher týmu
prohry	`INTEGER`	Počet proher týmu
remizy	`INTEGER`	Počet remíz týmu
poznamky	`TEXT`	Případné místo pro poznámky k týmu

Tabulka 4.5: Atributy entity TYM

FAKULTA
V této entitě máme uchované předdefinované zkratky fakult na Západočeské univerzitě v Plzni.

Název atributu	**Datový typ**	**Popis**
id_fakulty	`INTEGER`, PK	Identifikační číslo fakulty
zkratka_fakulty	`VARCHR(16)`	Zkratka fakult ZČU

Tabulka 4.6: Atributy entity FAKULTA

POST
Zde máme uchované předdefinované posty, které můžou hráči zastávat.

Název atributu	Datový typ	Popis
id_post	INTEGER, PK	Identifikační číslo postu
post	VARCHR(16)	Název hracího postu
zkratka	VARCHR(2)	Zkratka hracích pozic

Tabulka 4.7: Atributy entity POST

ROLE
V této entitě máme uchované předdefinované role, které můžou hráči zastávat.

Název atributu	Datový typ	Popis
id_role	INTEGER, PK	Identifikační číslo role
role	VARCHR(32)	Název role

Tabulka 4.8: Atributy entity ROLE

STATUS
Zde máme uložené předdefinované statusy, které můžou hráči vlastnit.

Název atributu	Datový typ	Popis
id_status	INTEGER, PK	Identifikační číslo statusu
status	VARCHR(16)	Název statusu
zkratka_statusu	VARCHR(4)	Zkratka statusu
cena_semestr	INTEGER	Cena, kterou má hráč zaplatit za semestr

Tabulka 4.9: Atributy entity STATUS

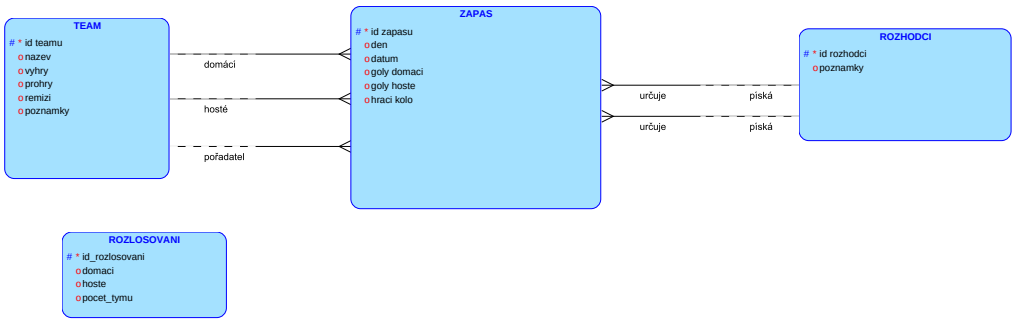
FORUM
V této entitě si ukládáme příspěvky ze všech fór (pro týmy, rozhodčí a veřejné).

Název atributu	Datový typ	Popis
id_zpravy	INTEGER, PK	Identifikační číslo zprávy
vlozil	VARCHR(32)	Jméno, přezdívka pod kterou se zobrazí zpráva
text	TEXT	Vložený příspěvek
email	VARCHR(32)	Kontaktní email
vlozeno	DATETIME	Vložený příspěvek

Tabulka 4.10: Atributy entity FORUM

4.1.2 Správa zápasů

Tato část se zabývá strukturou zápasu, z požadavků plynou všechny potřebné informace pro vytvoření jednotlivých zápasů. Na následujícím obrázku jsou zobrazeny všechny entity potřebné k vytvoření zápasu včetně jejich vazeb. Ke každému zápasu jsou potřeba tři týmy, jeden domácí, druhý hostí a třetí pořádající. K pískání zápasu jsou nutní dva rozhodčí, kteří se budou moci na zápas sami zapisovat. Dvě entity na tomto obrázku byly již popsány dříve (viz. 5.1.1 Správa uživatelů), tudíž je nebudeme znovu popisovat.

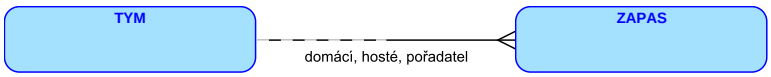


Obrázek 4.13: Konceptuální schéma správy zápasů.

Vztahy mezi entitami

Jak již bylo řečeno výše, popíšeme vztahy mezi entitami pomocí definované struktury. Entita rozlosování nemá žádnou vazbu s ostatními entitami.

Jeden tým může hrát více zápasů jako domácí, hosté nebo být pořadatelem. Více zápasů může hrát, či pořádat jeden tým.



Obrázek 4.14: Vztah TYM – ZAPAS

K odehrání zápasu jsou nutné tři různé týmy, proto je tato vazba ve schématu třikrát, my si jí znázorníme pouze jednou, protože ostatní dvě jsou analogické k té první. Tato vazba nám přiřazuje týmy k utkání. Zápasy jsou generovány automaticky s přihlášenými týmy.

Jeden rozhodčí může pískat více zápasů. Více zápasů může pískat jeden rozhodčí.

Obrázek 4.15: Vztah ROZHODCI – ZAPAS

Ke každému zápasu jsou potřeba dva rozhodčí, kteří určují zápas. Tak jako u předchozí vazby si znázorníme pouze jednu vazbu, i když v koncovém návrhu jsou dvě. Jde opět o analogii vazby, jelikož jsou třeba dva rozhodčí.

ZAPAS
Zde máme uchováváme informace o zápasech vygenerované pomocí rozlosování. Pouze rozhodčí se na zápas přihlašují sami, v případě nouze může administrátor rozhodčí k zápasu upravit. Všechny informace jsou upravitelné (např. z důvodu rektorského volna) kromě identifikátoru zápasu.

Název atributu	**Datový typ**	**Popis**
id_zapasu	`INTEGER`, PK	Identifikační číslo zápasu
den	`VARCHR(8)`	Zkratka hracího dne, vždy (po,st)
datum	`DATETIME`	Datum zápasu
cas	`VARCHAR(16)`	Čas v kolik začíná zápas, vždy (19:40, 20:25)
id_domaci	`INTEGER`, FK	Identifikační číslo domacího týmu
id_hoste	`INTEGER`, FK	Identifikační číslo hostujícího týmu
id_poradatel	`INTEGER`, FK	Identifikační číslo pořádajícího týmu
id_rozhodci_1	`INTEGER`, FK	Identifikační číslo rozhodčího
id_rozhodci_2	`INTEGER`, FK	Identifikační číslo rozhodčího
goly_domaci	`INTEGER`	Počet vstřelených gólů domácím týmem
goly_hoste	`INTEGER`	Počet vstřelených gólů hostujícím týmem
hraci_kolo	`INTEGER`	Číslo hracího kola

Tabulka 4.11: Atributy entity ZAPAS

Atributy jednotlivých entit

Všechny atributy vycházejí z požadavků na aplikaci a jsou nutné pro správný chod aplikace. Navíc dvě zobrazené entity jsou popsány výše (tab. 4.2 a 4.5)

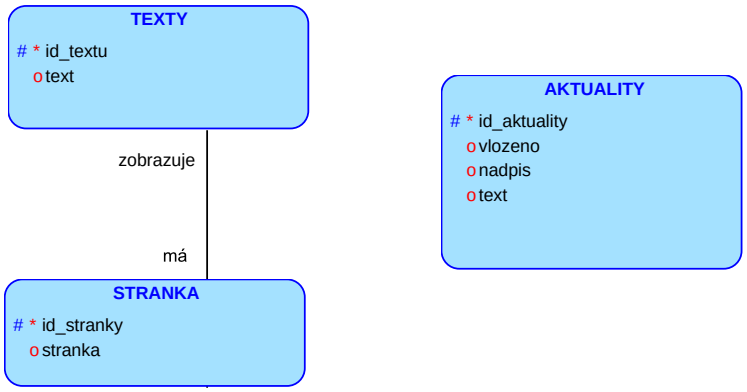
ROZLOSOVANI
Tato entita slouží pro správné rozlosování zápasů. Jsou zde uloženy modely zápasů pro rozlosování v rozmezí 4–10týmů.

Název atributu	Datový typ	Popis
id_rozlosovani	`INTEGER`, PK	Identifikační číslo rozlosování
domaci	`INTEGER`	Číslo týmu hrající jako domací
hoste	`INTEGER`	Číslo týmu hrající jako hosté
pocet_tymu	`VARCHAR(8)`	Čísla pro kolik je daný zápas týmů (např. 4, 5)

Tabulka 4.12: Atributy entity ROZLOSOVANI

4.1.3 Správa textů a aktualit

Tato část se zaměřuje na texty a články zobrazené ve veřejné části webové aplikace. Z požadavků na aplikaci je zřejmě, že správce bude měnit obsahy jednotlivých stránek a také, že buce přidávat aktuální články o dění na ZČU. Jelikož pro editaci obsahu je nutné si uchovávat jiné parametry než pro editaci aktualit, je potřeba vytvořit dvě různé entity.



Obrázek 4.16: Konceptuální schéma správy textů a aktualit.

TEXTY
V této entitě si ukládáme jednotlivé obsahy stránek ve formě HTML.

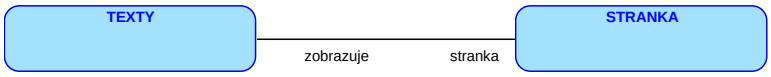
Název atributu	Datový typ	Popis
id_textu	`INTEGER`, PK	Identifikační číslo textu (obsahu)
text	`TEXT`	Text, který se má zobrazit na dané stránce

Tabulka 4.13: Atributy entity TEXTY

Vztahy mezi entitami

Popíšeme vztah mezi entitami pomocí definované struktury. Entita aktuality nemá žádnou vazbu s ostatními.

Jeden stránka má jeden obsah. Jeden obsah je zobrazen na jedné stránce.



Obrázek 4.17: Vztah TEXTY – STRANKA

Tato vazba nám přiřazuje k jednotlivým textům obsahy stránek, které mají zobrazit.

Atributy jednotlivých entit

Všechny atributy jednotlivých entit jsou potřebné pro splnění požadavků a funkcionality.

STRANKA
Zde máme uložené názvy stránek, které lze editovat.

Název atributu	**Datový typ**	**Popis**
id_stranky	`INTEGER`, PK	Identifikační číslo stránky
stranka	`TEXT`	Název stránky, které lze měnit obsah

Tabulka 4.14: Atributy entity STRANKA

AKTUALITY
V této entitě si ukládáme jednotlivé články, aktuality ve formě HTML.

Název atributu	**Datový typ**	**Popis**
id_aktuality	`INTEGER`, PK	Identifikační číslo aktuality
vlozeno	`DATETIME`	Datum přidání nové aktuality/článku
nadpis	`VARCHAR(64)`	Nadpis aktuality/článku
text	`TEXT`	Text (obsah) aktuality/článku

Tabulka 4.15: Atributy entity AKTUALITY

4.2 Závislost vztahů

V této části se seznámíme se všemi závislostmi záznamů u jednotlivých entit. Jedna ze závislostí vychází z UZIVATELE, pokud odstraníme záznam z této entity, je nutné odstranit záznamy i v ostatních entitách. Odstraníme-li tedy uživatele (např. bude hráč i rozhodčí), neodstraníme příslušný záznam jen v UZIVATELE, ale také v ROZHODCI a HRAC, který určíme podle cizího klíče. Analogicky pokud dojde k odstranění hráče, odstraní se i záznam v UZIVATELE, pokud není daný uživatel rozhodčí. V tomto případě dojde k odstranění záznamu pouze v HRAC, záznam v UZIVATELE zůstane kvůli vazbě s rozhodčím. Stejně to funguje i s rozhodčím, dojde-li ke smazání rozhodčího, odstraní se záznam v ROZHODCI, a pokud je uživatel hráčem, záznam v UZIVATELE zůstane, v opačném případě se odstraní.

Další závislosti nám plynou z předchozích příkladů, při odstranění rozhodčího je ještě třeba upravit utkání, na které je rozhodčí přihlášen v entitě ZAPAS. Dojde pouze k nastavení hodnoty na `null`, celý záznam se neodstraní. V případě odstranění hráče je nutné zjistit, zda nebyl brankářem, pokud ano, dochází k odstranění záznamu v GOLMAN.

Nejkomplikovanější odstranění přichází při smazání týmu. Dochází tím ke smazání všech hráčů na soupisce týmu. Odstranění hráče, brankáře, uživatele a rozhodčího jsem zmínil již výše. V entitě ZAPAS se odstraní příslušné zápasy s tímto týmem. V poslední řadě se odstraní týmové zprávy vložené na týmové fórum, jelikož nemá smysl uchovávat tyto zprávy i nadále. Tyto zprávy se odstraní z FORUM.

4.3 Výchozí hodnoty atributů

U některých z atributů jsou nastaveny výchozí hodnoty. To znamená, že při ukládání nových záznamů jsou určité hodnoty vyplněny výchozími hodnotami, pokud nejsou zadané explicitně jinak. Při zvolení vhodných hodnot to může být velmi účinné. Například při přidávání hráče má bodování (vstřelené góly, asistence...) nastavené na 0. Defaultně byl tento atribut prázdný, a tudíž při vypisování kanadského bodování se zobrazí prázdná kolonka. Po nastavení výchozí hodnoty se zobrazí 0, což vypadá lépe. Analogicky jsou nastaveny brankářské statistiky a týmové statistiky.

4.4 Fyzický datový model

Po realizaci konceptuálního schématu vycházejícího z požadavků na aplikační funkcionalitu lze vytvořit fyzický datový model. Informace shromážděné z popisu jednotlivých vztahů a entit se zobrazí do celého návrhu datového modelu. Pro navržení modelu jsem zvolil prostředí Oracle SQL Developer 4.0 – Data Modeler. Schéma datového modelu ukazuje celkové řešení návrhu a je uvedeno v Příloze B na str. (ještě nevím) obr. 7.2. Následuje Příloha C, kde jsou ukázky skriptů generující část navržené databáze. Kvůli rozsahu databáze jsou kompletní zdrojové kódy pro navržení databázového modelu na disku CD-ROM, který nalezneme v Příloze D.

5 Implementace aplikace

V předchozí části jsem navrhl databázový model, který lze nyní plnit daty. Stručně popíši vybrané prostředky pro implementaci a samotnou implementaci pro vytvoření webové aplikace a administrátorského prostředí.

Pro vývoj aplikace jsem využil všechny programovací jazyky zmíněné v kapitole 3.1. Programovací jazyky, kromě jazyka ASP.NET, který nelze použít v případě využití PHP. PHP jsem si zvolil z důvodu větší podpory u hostingů a také díky jeho rozšíření mezi webovými aplikacemi. Osobně upřednostňuji PHP. Ostatní vybrané jazyky jsou standardem při vytváření obdobných projektů.

Jako datové úložiště využiji databázový systém MySQL popsaný výše. Ke komunikaci s databází využívám knihovnu DIBI.

Z konkrétních vybraných technologií lze vybrat poskytovatele hostingu. Po předložení mého návrhu a po konzultaci s katedrou tělesné výchovy a sportu jsme se rozhodli pro založení webového projektu na CIVu[1], který využijeme pro hosting webových stránek. Webový hosting poskytovaný CIVem nám nabízí, jak využití PHP tak MySQL technologii. Největší výhodou tohoto hostingu je vybrání libovolné neobsazené domény a jeho cena za poskytnutí. Je zcela zdarma. Jako adresu stránek si KTS vybrala www.florbal.zcu.cz, kde je spuštěný projekt.

Pro export požadovaných dokumentů jsem využil knihovnu mPDF, jelikož jsem zvolil k tomu přizpůsobený jazyk PHP.

5.1 Webová aplikace

Tato část aplikace je vytvořena pomocí procedurálního programování. Struktura implementace je rozdělená do více složek podle jejich účelu. Kaskádový styl se nachází ve složce css. Soubory ke stažení jsou ve složce download. Použité speciální písmo najdeme ve složce fonts. Použité obrázky v img, knihovna pro práci s databází je ve složce lib. Poté následuje složka pages, která obsahuje všechny obsahy stránek a skriptů, které může uživatel vidět. Poslední

[1] CIV – Centrum informatizace a výpočetní techniky.

složka je zapis, kde je vše potřebné pro vygenerování zápisu k utkání v pdf, tudíž zde nalezneme knihovnu mPDF. Poslední soubor je **index.php**, který se načítá jako první při navštívení webu.

Všechny stránky vycházejí právě ze zmíněného **index.php**. Jedná se o kostru webu, do které je načítán obsah jednotlivých stránek pomocí PHP funkce *include()*. Stránka je rozdělena na hlavičku (head) a tělo (body).

V hlavičce jsou tzv. *meta tagy*, kde se nastavuje kódování, klíčová slova, popisek pro vyhledávače a další. Ve své práci jsem použil všechny zmíněné tagy. Dále jsem v hlavičce nastavil cestu ke kaskádovému stylu pomocí *linku*, který používám pro naformátování obsahu stránek. Jako poslední věc v hlavičce jsem nastavil *title*, což je nadpis stránek v záložce ve webovém prohlížeči.

V těle kostry dochází k načtení skriptu pro přihlášení do databáze opět pomocí funkce *include()*. Následně dochází k nastartování *SESSION* funkcí *session_start()*, což je super globální proměnná, ve které můžeme mít uložené více parametrů. Tuto proměnnou využívám pro uchování informací o přihlášeném uživateli. Pomocí toho dokáži rozlišit, zda je uživatel přihlášen, popřípadě nastavím informační panel na stránce o tom, kde uživatel vidí svoje přihlašovací jméno a možnost upravit své údaje. Následuje vytvoření menu pomocí HTML tagů, kde se nachází všechny záložky (např. UFL, REPREZENTACE, TURNAJE, KONTAKY. . .). Pokud je přihlášen hráč, má o záložku navíc, která ho odkazuje na souhrn informací o jeho týmu, což rozpoznám díky nastavení *SESSION*. Pod menu se nachází tabulka týmů seřazená podle jejich získaných bodů. Týmy si načtu z databáze pomocí knihovní funkce dibi a to *dibi::fetchAll()*, která nám vrátí pole všech řádků (záznamů) v tabulce TYM. Díky možnostem jazyka SQL nám databáze vrátí již seřazené týmy podle bodů (k seřazení řádků v dotazu slouží příkaz *ORDER BY*, pro získání záznamů *SELECT*). Poté získané hodnoty procházím cyklem *foreach* a postupně vypisuji týmy do tabulky. Jako poslední na stránce jsou tři odkazy v podobě ikon na facebookové stránky KTS, následuje odkaz na stránky KTS a jako poslední je ikona pro přihlášení. Pokud je hráč přihlášen, ikona o přihlášení zmizí, to rozpoznám díky nastavení *SESSION*. Dosud všechno zmíněné je zobrazeno na všech stránkách webu a mění se pouze jejich střed (obsah) pomocí již zmíněné funkce PHP *include()*. Defaultně je nastaveno načtení úvodní stránky. V případě upravení adresy a nenalezení požadované stránky je načtena opět úvodní strana.

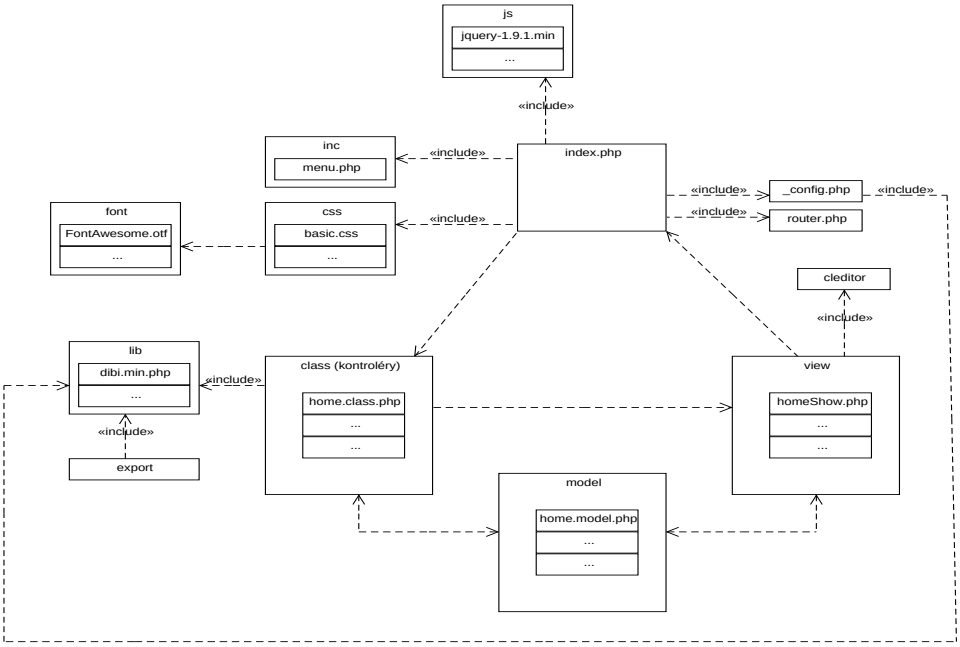
Jednotlivé stránky obsahují vždy nadpis, který naznačuje jejich účel. Poté následuje informační text, případné tlačítka odkazující se na jiné stránky či formuláře.

Při zobrazení některých textů je nutné získat daný obsah stránky z databáze, protože obsah některých stránek je editovatelný. Pomocí funkce *dibi::fetch* a SQL dotazu získám jeden záznam z tabulky TEXTY uložený ve formě HTML, který následně zobrazím. Stránky, kde je zobrazený text, který nelze editovat, jsou také ve formě HTML, ale již bez přístupu do databáze.

Všechny formuláře na webu vždy zpracovává příslušný skript. Například při registraci týmu dojde k vyplnění formuláře a následně ho uživatel odešle. Skript (**registrujTym.php**) jako první ověří, zda jsou vyplněny všechny povinné údaje, pokud ne vypíše chybu, která nastala, jinak pokračuje dál. Pro tuto kontrolu údajů jsem využil funkci *empty()*, která vrací *true* v případě, že je parametr prázdný, *false* v případě, když parametr něco obsahuje. Následně když jsou všechny údaje vyplněné, tak se zkontrolují hesla, zda se shodují. Zde jsem využil funkci *strcmp()* pro porovnání dvou řetězců, která opět vrací *true* v případě, že se řetězce shodují a *false*, když ne. Poté se zkontroluje délka přihlašovacího jména a hesla. K tomu nám slouží funkce *strlen()*, která nám vrací počet znaků zadaného parametru. Následuje ověření správnosti tvaru emailu a telefonu. Kde jsem využil regulární výrazy a funkci *preg_match()*, která vrací *true*, pokud parametr splňuje definovaný tvar a *false*, pokud ne. Pak si z databáze načtu počet přihlašovacích jmen, emailů, telefonů a týmů, kde jsou údaje stejné jako ve vyplněném formuláři. Pomocí *dibi::fetchAll()* získám všechny tyto záznamy a pomocí *count()* zjistím počet všech těchto záznamů. Pokud je již některý z údajů v databázi, je uživatel požádán o změnu daného údaje. Po důkladné kontrole vyplněných údajů je daný tým s vedoucím registrován a vložen do databáze. K tomu jsem opět využil DIBI a její funkci *dibi::query()*. Následně vypíši uživateli zprávu o přidání a pomocí funkce *header()* ho přesměruji na úvodní stranu. Všechny ostatní skripty fungují na velmi podobné logice. Jediný skript na generování zápisu k utkání funguje na jiné logice. Ten funguje na principu získání informací z databáze o vybraném utkání, jako je soupiska hráčů, názvy týmů atd. Poté se nastaví jméno PDF souboru, kódování (UTF–8) a formát souboru (A4–L – A4 na šířku). Pomocí funkce *file_get_contents()* získáme obsah CSS souboru uvedený jako parametr. Následně aktivujeme pomocí funkce *ob_start()* výstupní buffer. Pak HTML tagy vypíšeme potřebné údaje. Poté do proměnné pomocí funkce *ob_get_contents()* uložíme obsah bufferu. Ještě před vygenerováním výsledného PDF vyprázdníme po sobě výstupní buffer funkcí *ob_end_clea()*. Posledním krokem vygenerujeme PDF soubor se zadanými parametry. K tomu jsem použil funkci *getSimplPdfWindow()*, kam jako první parametr je vložen název souboru, následuje HTML kód a použité CSS.

5.2 Administrátorské prostředí

Tato část na rozdíl od předchozí části je vytvořena pomocí objektově orientovaného programování s využitím MVC architektury. Použitá architektura vychází ze společného projektu ze ZSWI. Struktura administrátorského prostředí je komplikovanější a tak místo popisu jsem se rozhodl ji zachytit pomocí následujícího schématu (obr. 5.1).



Obrázek 5.1: Struktura administrátorského prostředí.

Pokud se chce správce dostat do prostředí, musí se nejdřív přihlásit v předchozí části, poté je přesměrován do administračního prostředí. Všechny stránky opět vycházejí ze souboru **index.php**. Zde jako první věc zapneme SESSION funkcí *session_start()*, abychom získali informace o uživateli, který se snaží do prostředí dostat. Následuje podmínka, která nám ošetřuje, zda je daný uživatel oprávněn ke vstupu a funkcí *isset()* zjistíme, zda jsou SESSION vůbec nastaveny. Při záporném vyhodnocení je uživatel vrácen na veřejnou část. Pokud je přihlašovaný správcem, příkazem *require_once()* načteme příslušný konfigurační soubor pro připojení do databáze **_config.php** a soubory na nastavení routování **router.php** a kontrolér **controler.class.php**. Příkaz *require_once()* jsem využil z důvodu toho, že pokud byl kód ze souboru již dříve vložen, nevloží se znovu.

Funkce routování **router.php** je určení, jaký kontrolér zavolat. Jako první si funkcí *expole()* rozdělím adresu do pole podle oddělovače *&*. Funkcí *array_shift()* si vezeme první prvek z pole, který mi určuje název stránky. Tato funkce nám vyjme první prvek a posune všechny prvky dopředu, funguje na principu fronty. Potom zjistíme, zda název stránky není prázdný funkcí *empty()*, pokud je, za stránku označíme úvodní **home** a za kontrolér **home.class.php**. V opačném případě označíme daný kontrolér. Potom zbylé argumenty procházím cyklem *foreach* a rozděluji je dál, které si ukládám do jednotlivých proměnných. Poté dochází k ověření, zda daný kontrolér existuje funkcí *file_exists()*, pokud ne je vypsána chyba a skript je ukončen pomocí funkce *die()*. Pokud daný kontrolér existuje, je načten pomocí *include_once()*. Poté převedu první písmeno kontroléru na velké funkcí *ucfirst* a to označím za název třídy kontroléru. Následně funkcí *class_exists()* ověřím, její existenci. V případě, že třída neexistuje, je skript ukončen s výpisem chybové hlášky funkcí *die()*. Pokud existuje, dojde k vytvoření nové instance třídy kontroléru.

Kontrolér **controler.class.php** má dvě metody, které ostatní kontroléry od něj dědí. Jsou to metody *main()* a *render()*. V první zmiňovaný metodě si kontrolér zjistí, jakou metodu má volat (např. *show*). Druhá zmiňovaná metoda, slouží pro zobrazení příslušného view. Nejprve si načteme daný view, potom ověříme jeho existenci opět metodou *file_exists()*. Pokud neexistuje, označíme za view chybovou stránku, kterou následně zobrazíme. Následně dochází k zapnutí bufferování výstupu funkcí *ob_start()*, načteme příslušnou stránku pomocí *include()* a do proměnné si uložíme obsah interního bufferu pomocí *ob_get_contents()*. Následně po sobě obsah bufferu vymažeme pomocí *ob_end_clean()* a zobrazíme příkazem *echo*.

Nyní se vracíme zpět k souboru **index.php**. Tak jako v předchozím souboru

index.php, i zde je stránka rozdělena na tělo (body) a hlavičku (head). V hlavičce jsou použité podobné meta tagy, jako v předchozí části. Navíc je zde pouze tag pro načtení javasriptů. Poté dochází k zavolání dotyčné metody *main()* z již vytvořené instance třídy. Jako poslední k čemu na stránce dochází je načtení menu pomocí funkce *include()*.

Vše doteď zmíněné nám poskytuje kostru administrátorského prostředí. Dokáže načíst kteroukoliv stránku, z kteréhokoliv kontroléru a zobrazit jí. Poslední, co nám zbývá zmínit je použití modelu. Do příslušného kontroléru vždy načteme příslušný model. V kontroléru si poté vytvoříme instanci modelu, abychom dokázali z něj využívat příslušné metody. V modelu jsou nejčastěji metody pro získání/odstranění/aktualizaci dat z databáze. Ve view si opět načítáme model, který budeme potřebovat funkcí *require_once()*. Pro získání dat si nejprve vytvoříme instanci modelu a díky němu, poté voláme příslušné metody.

Struktura jednotlivých stránek je podobná těm z předchozí části. Jako první na stránce je nadpis a následuje informační text, výpis záznamů, formulář či tlačítka odkazující se na další stránky. Zde pouze nevyhodnocuje stránku příslušný skript, ale kontrolér. Pro kontrolu dat z formulářů jsem opět využil regulární výrazy a funkce zmíněné výše. Celkově použité funkce v této části jsou velmi podobné funkcím z části předchozí. Pro práci s databází jsem využil funkce knihovny dibi, opět zmíněné výše. Export souborů je stejný, jako v předchozí části. V případě nejasností je celý zdrojový kód důkladně okomentován.

6 Testování

Testování se zavádí z důvodu ověřování kvality hotového systému. Testování se zabývá vyhledáváním chyb, to je zřejmé, ale liší se v jeho velikosti záběru. Testování lze rozdělit do více testů např. testy použitelnosti, testy spolehlivosti, testy podpory a jiné. Webové aplikace se liší od desktopových zejména v tom, že webové aplikace využívají prohlížeč. Už to lze považovat za první problém, na který bychom si měli dát pozor. Uživatelé mají na webu větší volnost pohybu, než u desktopových aplikací. My se budeme soustředit na testy kompatibility mezi různými prohlížeči, při různém rozlišení obrazovky a samozřejmě na funkcionalitu aplikace. Webovou aplikaci je nejlepší testovat ručně než pomocí automatických testů.[6]

6.1 Testování výsledného systému

Zabývat se budeme pouze dynamickou kontrolou systému, to znamená, že aplikace odpovídá požadavkům na aplikaci. Dynamické testování vyžaduje testování na běžící aplikaci a probíhá hlavně na základě různých vstupů a posuzování výstupů. Aplikace je testována na stroji Lenovo ThinkPad T520, Intel(R) Cote(TM) i7-2670QM CPU @ 2.20GHZ, 8,00 GB RAM na systému Windows 7 Proffesional 64bitové verzi.

Jako první jsem testoval kompatibilitu mezi prohlížeči. Abych věděl, na jakých prohlížečích bych měl aplikaci testovat, tak pro získání statistik používání prohlížečů jsem využil stránky uvedené v literatuře [7].

Jako prvním testovaným prohlížečem byl Google Chrome – verze 34.0, který je v České Republice nejrozšířenější. Následoval Mozilla Firefox – verze 28.0. Jako třetí prohlížeč jsem zvolil Internet Explorer 10 – verze 10.0. Poslední prohlížeč jsem vybral Operu – verze 20.0.

Shrnutí výsledků kompatibility mezi prohlížeči. Podle očekávání Google Chrome dopadl pro danou aplikaci jako nejlepší. To především z důvodu, že aplikace byla vyvíjena pro tento prohlížeč. Firefox jako druhý nejpoužívanější prohlížeč prokázal výbornou kompatibilitu a s aplikací neměl sebemenší problém. První problém přichází s prohlížečem Internet Explorer 10. Zde bylo nalezeno špatné zobrazení při stahování zápisu k utkání, kolem ikony ke stažení je navíc rámeček. V administrátorském prostředí u položek menu jsou vidět odrážky. Všechny nalezené chyby v tomto prohlížeči nemají vliv na

funkcionalitu, ale pouze na vzhled. Jako poslední prohlížeč byl Opera, kde nenastal žádný problém.

Po testování různých prohlížečů jsem testoval různé rozlišení obrazovky. Pro získání nejpoužívanějších rozlišení jsem opět využil stránky jako u statistiky prohlížečů. Ukázalo se, že nejpoužívanější rozlišení je 1366x786, následuje 1920x1080, 1280x1024 a 1280x800. Rozlišení 1024x768 již není, tak aktuální jako před lety, tudíž jsem se rozhodl jej do testování nezahrnout.

Při testování zmíněných rozlišeních jsem nenarazil na problém s komponenty či jejich zobrazením. Pouze u nižších rozlišení se může uživateli zdát zobrazená stránka trochu větší, než by bylo třeba. Doporučená rozlišení jsou dvě nejpoužívanější 1920x1080 a 1366x786.

Dosud jsme testovali různá prostředí s různými možnostmi zobrazení. Nyní otestujeme funkcionalitu aplikace podle požadavků na systém (viz. 2. Základní popis aplikace). Začnu testováním formulářů. Poté budu testovat možné role uživatelů. Dále budu zkoušet měnit parametry v URL[1] adrese.

Při vyplňování veškerých formulářů jsem nenarazil na žádný problém a vše funguje podle požadavkům. Při testování jsem nejprve zadával očekávané hodnoty a poté neočekávané, pokaždé aplikace reagovala korektně. V případě role hráče, či rozhodčího se žádný problém nevyskytl. V roli administrátora jsem narazil na chybu. Chyba byla v odhlášení se z administrátorského prostředí, aplikace nereagovala na tento požadavek korektně. Jiné problémy v rolích uživatelů nebyly zjištěny. Při posledním testování měnění parametrů nebo celé URL adresy jsem narazil na více problémů. Jednalo se o chyby stejného typu, jenom na více stránkách. Například po přihlášení rozhodčího jsem se mohl dostat na správu týmu, i když rozhodčí žádný tým nemá a v žádném nehraje. Stejně tak po přihlášení hráče, nebo rozhodčího jsem se po upravení URL adresy dostal do administrátorského prostředí.

6.2 Zhodnocení testovacích výsledků

Jedná-li se o prostředí, kde máme spuštěnou aplikaci, lze ze zjištěných informací vyvodit závěr, že vytvořená aplikace je bezproblémová.

Při testování funkcionality aplikace byly nalezeny chyby. Důsledkem chyb jsou pouze menší nepříjemnosti. Všechny objevené problémy vznikly nepozorností programátora nebo špatným odhadem daného problému. Všechny

[1]URL – Uniform Resource Locator.

chyby byly opraveny a aplikace je připravena na předání koncovému uživateli.

7 Závěr

Tato bakalářská práce se zabývala navržením a implementováním webové aplikace pomocí vhodných prostředků. V požadavcích na systém je zmíněna i desktopová aplikace, která je považována za sekundární část. Po domluvě s KTS nakonec tato aplikace není požadována z důvodu malé pravděpodobnosti výpadku funkčnosti aktuální informační tabule v hale KTS. Vývojem aplikace chceme dosáhnout větší informovanosti studentů a zaměstnanců ZČU o florbalovém dění. Správce díky administračnímu prostředí dokáže řídit celou univerzitní florbalovou ligu.

Na začátku práce jsem se zaměřil na technologie potřebné pro tvorbu webu včetně databázového systému vycházející ze studia odborných publikovaných článků a literatury.

Na základě požadavků jsem navrhl konceptuální schéma se všemi potřebnými entitami pro správnou funkcionalitu včetně jejich vztahů a atributů. Jeho grafické znázornění je v Příloze A (obr. 7.1). Pro bližší pochopení popisuji vztahy mezi jednotlivými entitami a výchozí hodnoty atributů. Poté byl vytvořen fyzický model, který specifikuje vlastnosti databáze již konkrétních datových typů.

Nad navrženým databázovým modelem byla implementována webová aplikace. Jako první jsem začal implementaci veřejné části s možností přihlášení uživatelů. To jsem vytvořil pomocí procedurálního programování. Druhou část webu tvoří administrátorské prostředí, které jsem vytvořil pomocí objektově orientovaného programování. Po vytvoření aplikace lze tyto dva způsoby programování srovnat. Pro menší weby se dá určitě využít procedurálního programování. Pro weby rozsáhlejší bych rozhodně doporučil objektově orientované programování, z důvodu lepší architektury a lepší přehlednosti. Běžný návštěvník webu nepozná metodu programování, tudíž vybraná metoda nemá na funkcionalitu žádný vliv. U administrátorského prostředí jsem využil návrhového vzoru MVC.

Po vytvoření aplikace jsem provedl její testování. Díky tomu jsem odhalil chyby, které byly následně opraveny. Chyby u rozsáhlé webové aplikace vzniknou velmi snadno. Hlavním zdrojem chyb bylo opomenutí programátora.

Stránky byly vytvořeny s použitím programovacího jazyka PHP. Pro

uchování dat byla použita databáze MySQL. V současné době web slouží pro poskytnutí informací o florbalovém dění na ZČU. Od nového školního roku bude sloužit i pro správu florbalové univerzitní ligy. Webové stránky jsou dostupné na http://www.florbal.zcu.cz. Ukázky rozhraní jsou uvedeny v Příloze E.

Web by se mohl do budoucna rozšířit o galerii obrázků. Další možností rozšíření je využití OCR[1] pro automatické načítání zápisů o utkání. To by přineslo další automatizaci a zjednodušení práce pro administrátora.

[1] OCR – Optical Character Recognition

Literatura

[1] Tvorba WWW - o webdesignu, grafice a reklame: ASP - Active Server Pages [online]. [cit. 2014-04-11]. Dostupne z: http://www.tvorba-webu.cz/php/asp.php

[2] BC. VIKTOR MLADĚNKA. Charakteristika databázových systémů a tvorba databáze [online]. [cit. 2014-04-11]. Dostupné z: http://www.vn.cz/ mladenkav/izm/mpi/MPI%2012.doc

[3] ČÁPKA, David. 1. díl - Úvod do ASP.NET. Devbook.cz [online]. [cit. 2014-04-12]. Dostupné z: http://www.devbook.cz/tutorial-uvod-do-asp-dot-net

[4] ČÁPKA, David. Databáze: Tvorba konceptuálního modelu z business zadání. Devbook.cz [online]. [cit. 2014-04-12]. Dostupné z: http://www.devbook.cz/databaze-navrh-konceptualni-model-z-business-zadani

[5] HOLZSCHLAG, Molly E. HTML a CSS jdi do toho. Praha: Grada Publishing, a.s., 2006. ISBN 80-247-1451-X.

[6] Testování softwaru: základy testování. BOROVCOVÁ, Anna. [online]. [cit. 2014-04-19]. Dostupné z: http://testovanisoftwaru.blogspot.cz/

[7] StatCounter: GlobalStat. [online]. [cit. 2014-04-19]. Dostupné z: http://gs.statcounter.com/

[8] Dibi is Database Abstraction Library for PHP 5. | dibi [online]. 2008, 2014 [cit. 2014-04-20]. Dostupné z: http://dibiphp.com/cs/

[9] OZANA, Roman. CLEditor - fajný WYSIWYG HTML Editor. [online]. [cit. 2014-04-20]. Dostupné z: http://www.nabito.net/cleditor-fajny-wysiwyg-html-editor/

[10] MVC architektura. Devbook.cz [online]. [cit. 2014-04-20]. Dostupné z: http://www.devbook.cz/mvc-architektura-navrhovy-vzor

[11] Prezentační vzory z rodiny MVC. Zdrojak.cz [online]. [cit. 2014-04-20]. Dostupné z: http://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/

[12] Přechod z MySQL. Postgres.cz [online]. 2013 [cit. 2014-04-20]. Dostupné z: http://postgres.cz/wiki/Přechod_z_MySQL

Seznam použitích zkratek

ZČU *(University of West Bohemia)*
: Zkratka pro Západočeskou univerzitu v Plzni.

KTS
: Katedra tělesné výchovy a sportu.

ČAH
: České akademické hry.

UFL *(University floorball league)*
: Univerzitní florbalová liga.

IFF *(International Floorball Federation)*
: Mezinárodní florbalová federace.

PDF *(Portable Document Format)*
: Soubory formát pro ukládání dokumentů.

HTML *(HyperText Markup Language)*
: Hypertextový značkovací jazyk..

CSS *(Cascading Style Sheets)*
: Kaskádovací styl pro desing webu.

PHP *(Hyperrtext Preprocessor)*
: Skriptovací programovací jazyk.

ASP.NET *(Active Server Pages .NET)*
: Součást .NET Frameworku pro tvorbu webových aplikací a služeb.

C# *(C sharp)*
: Vysokoúrovňový objektově orientovaný programovací jazyk.

SQL *(Structured Query Language)*
: Strukturovaný dotazovací jazyk používaný pro práci s daty v databázích.

MVC *(Model, View, Controller)*
: Třívrstvá architektura, pro oddělení logiky od výstupu.

MVP *(Model , View, Presenter)*
: Třívrstvá architektura.

XLM *(Extensible Markup Language)*

Obecný značkovací jazyk.

WYSIWYG *(What You See Is What You Get)*
Překládáno jako „Co vidíte, to dostanete“.

SMTP *(Simple Mail transfer Protocol)*
Internetový protokol pro přenos zpráv.

CIV
Centrum informatizace a výpočetní techniky.

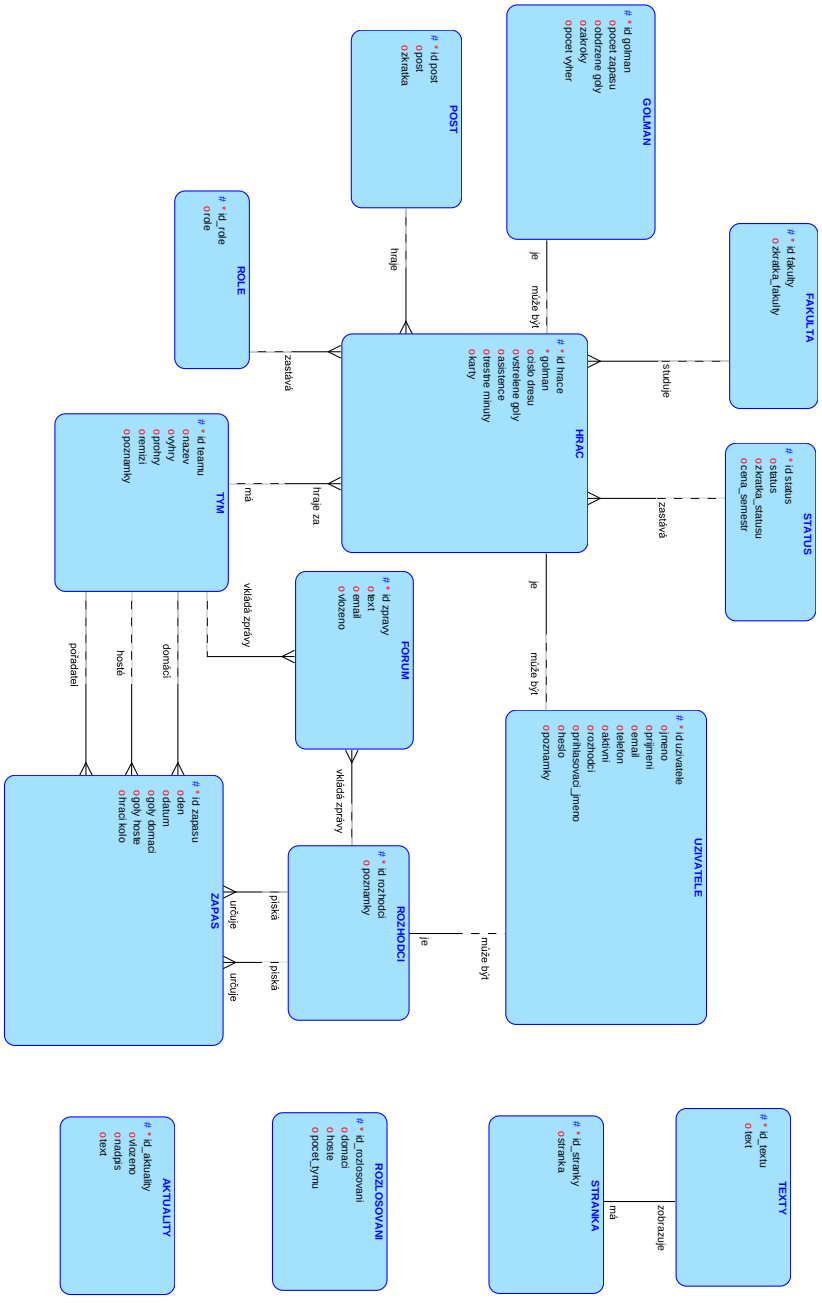
URL *(Uniform Resource Locator)*
Definuje adresu určující umístění dokumentu na internetu.

OCR *(Optical Character Recognition)*
Metoda, která pomocí scanneru umožňuje digitalizaci tištěných textů..

Seznam tabulek

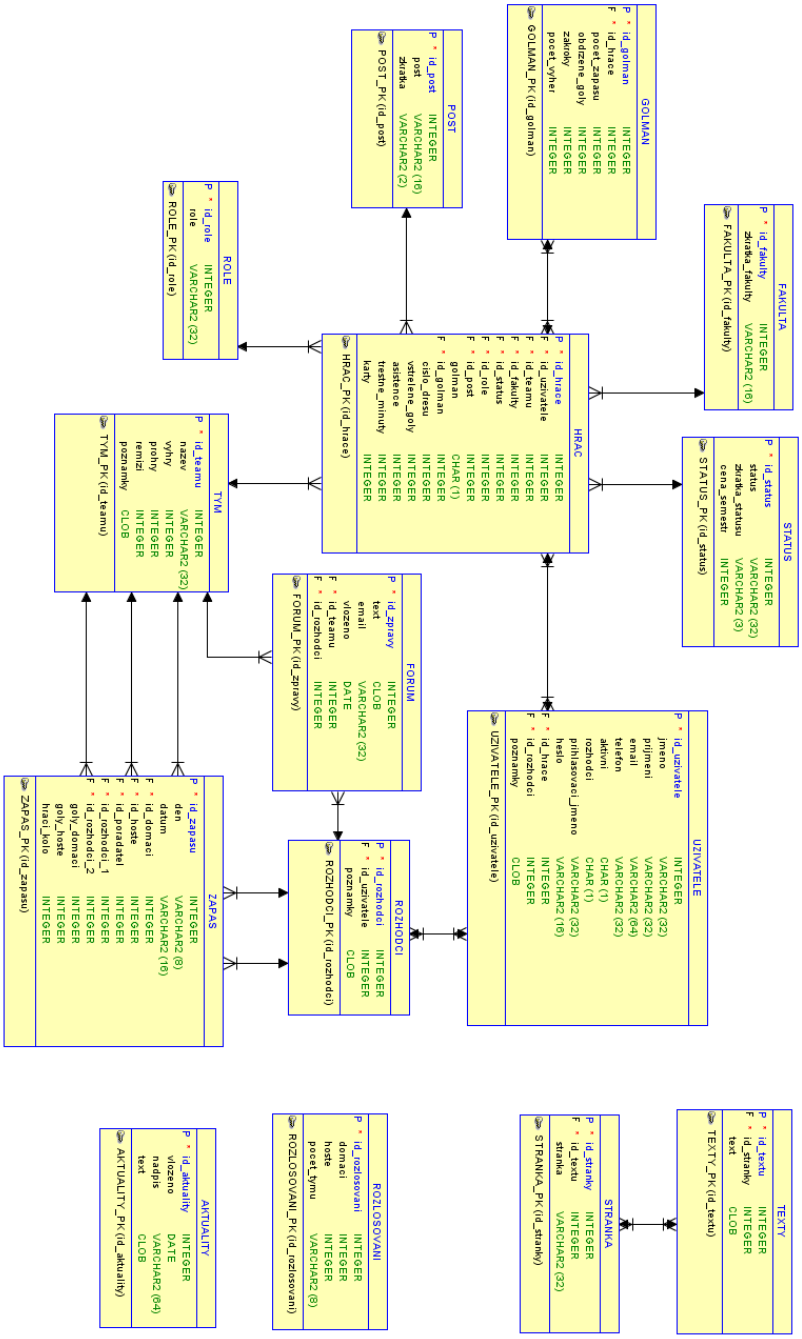
Seznam obrázků

Příloha A – Konceptuální schéma



Obrázek 7.1: Kompletní konceptuální schéma navržené pro aplikaci.

Příloha B – Fyzický model



Obrázek 7.2: Fyzický datový model pro navrženou aplikaci.

Příloha C – ukázka skriptů

Definice tabulek

Ukázka vytvoření tabulky HRAC

```
CREATE TABLE HRAC(
    id_hrace        INTEGER NOT NULL ,
    id_uzivatele    INTEGER NOT NULL ,
    id_teamu        INTEGER NOT NULL ,
    id_fakulty      INTEGER NOT NULL ,
    id_status       INTEGER NOT NULL ,
    id_role         INTEGER NOT NULL ,
    id_post         INTEGER NOT NULL ,
    golman          CHAR (1) ,
    id_golmana      INTEGER ,
    cislo_dresu     INTEGER ,
    vstrelene_goly  INTEGER ,
    asistence       INTEGER ,
    trestne_minuty  INTEGER ,
    karty           INTEGER

   );
```

Definice primárních klíčů, cizích klíčů

Nastavení primárního klíče k dané tabulce:
`ALTER TABLE HRAC ADD CONSTRAINT HRAC_PK PRIMARY KEY ( id_hrace );`

Nastavení cizích klíčů:

```
ALTER TABLE HRAC ADD CONSTRAINT HRAC_FAKULTA_FK FOREIGN KEY
( id_fakulty ) REFERENCES FAKULTA ( id_fakulty );


ALTER TABLE HRAC ADD CONSTRAINT HRAC_GOLMAN_FK FOREIGN KEY
( id_golman ) REFERENCES GOLMAN ( id_golman );
```

```
ALTER TABLE HRAC ADD CONSTRAINT HRAC_POST_FK FOREIGN KEY
( id_post ) REFERENCES POST ( id_post );


ALTER TABLE HRAC ADD CONSTRAINT HRAC_ROLE_FK FOREIGN KEY
( id_role ) REFERENCES ROLE ( id_role );


ALTER TABLE HRAC ADD CONSTRAINT HRAC_STATUS_FK FOREIGN KEY
( id_status ) REFERENCES STATUS ( id_status );


ALTER TABLE HRAC ADD CONSTRAINT HRAC_TYM_FK FOREIGN KEY
( id_teamu ) REFERENCES TYM ( id_teamu );


ALTER TABLE HRAC ADD CONSTRAINT HRAC_UZIVATELE_FK FOREIGN KEY
( id_uzivatele ) REFERENCES UZIVATELE ( id_uzivatele );
```

Definice zbylého nastavení

U určitých atributů nastavíme výchozí hodnoty:

```
ALTER TABLE ‘hrac‘ CHANGE ‘vstrelene_goly‘ ‘vstrelene_goly‘ INT(
11 ) NULL DEFAULT ’0’;


ALTER TABLE ‘hrac‘ CHANGE ‘asistence‘ ‘asistence‘ INT( 11 ) NULL
DEFAULT ’0’;


ALTER TABLE ‘hrac‘ CHANGE ‘trestne_minuty‘ ‘trestne_minuty‘ INT(
11 ) NULL DEFAULT ’0’;


ALTER TABLE ‘hrac‘ CHANGE ‘karty‘ ‘karty‘ INT( 11 ) NULL DEFAULT
’0’;
```

Příloha D – Obsah CD-ROM

CD-ROM je umístěný v na přední straně desek bakalářské práce. Obsah disku je uveden níže:

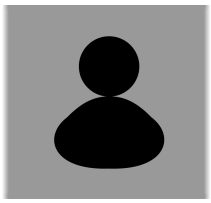
Dokumentace	Adresář s tímto dokumentem a jeho zdrojovým textem.
Skripty	Skripty pro vytvoření databázového modelu.
Schéma	Schéma databázového modelu.
Zdrojové kódy	Adresář se všemi zdrojovými kódy.

Příloha E – Uživatelská příručka

Uživatelská dokumentace je rozdělena podle jednotlivých rolí. V příloze F jsou zobrazeny screenshoty webové aplikace. Hráčům registrovaným vedoucím týmu je doporučeno, aby své přihlašovací údaje při první návštěvě změnili! Po přečtení uživatelské dokumentace by mělo být čtenáři jasné, jak danou aplikace ovládat.

Pro hráče

Aby se mohli hráči přihlásit, je nejprve nutné zaregistrovat tým. Tento úkol má na starost vedoucí týmu, který podá přihlášku pomocí formuláře na stránce pro přihlášení týmu, na kterou se dostane následující cestou *UFL->PŘIHLÁŠKA->Přihlas tým!*. Pokud vedoucí týmu vše vyplní správně, přihláška se odešle do systému, kde na její potvrzení čeká správce. Po potvrzení správcem přijde vedoucímu týmu email o aktivaci přístupu. Do této doby se hráč nemohl přihlásit! Pokud správce tým vymaže, vedoucí týmu o tom bude informován emailovou zprávou. Po aktivačním emailu se může vedoucí přihlásit do systému pomocí kliknutí na ikonu pod tabulkou na stránkách (obr. 7.3).



Obrázek 7.3: Ikona pro přihlášení.

Po úspěšném přihlášení přibyde uživateli v menu nová záložka *MŮJ TÝM*. Kliknutím na tuto záložku je možné se dostat do stručného přehledu svého týmu. Zde se nachází aktuální informace a dvě možnosti.

První z nich je možnost vstoupit na týmové fórum pomocí tlačítka *TÝMOVÉ FÓRUM* a komunikovat s ostatními hráči týmu. Pro přidání vzkazu je zapotřebí kliknout na tlačítko *PŘIDAT VZKAZ*, tlačítkem *ZPĚT* se dostaneme zpátky na informace o týmu. Při přidávání nové zprávy nám naběhne textové pole, do kterého může napsat naši zprávu. Tlačítkem *SMAZAT* odstraníme veškerý text z textového pole. Tlačítkem *ODESLAT* se zpráva odešle a dojde k přesměrování zpět na týmové fórum. Pro pořádek se jako autor

zprávy bere automaticky příjmení hráče.

Druhá z možností je tlačítko *SPRÁVA TÝMU*. Tuto možnost má pouze vedoucí týmu! Po kliknutí na toto tlačítko se zobrazí seznam hráčů v týmu včetně vedoucího. Zde má vedoucí týmu další možnosti. Jedna z nich je tlačítko *PŘIDEJ HRÁČE*, která po stisknutí nabídne formulář pro přidání hráče na soupisku. Při správném vyplnění údajů a odesláním formuláře pomocí tlačítka *Přidej hráče* je hráč zapsán na soupisku týmu. Přidanému hráči přijde upozornění na email, že byl registrován i s přihlašovacími údaji. Přidaný hráč je aktivní, správce nemusí nic povolovat. Vedoucí týmu po přidání hráče je vrácen na soupisku týmu. Další z možností vedoucího je, mazání hráčů pomocí tlačítka *Smazat*, které je vždy vedle každého hráče na soupisce. Po stisknutí tlačítka je hráč nevratně smazán, pro jeho opětovnou účast v lize ho musíte znovu přidat pomocí příkazů zmíněných výše. Opět i zde je tlačítko *ZPĚT*, které nás vrátí na informační stránku o týmu.

Při přihlášení se uživateli v pravém horním rohu stránky objeví informační panel s možnostmi odhlášení, k tomu slouží tlačítko *Odhlásit se*, po stisknutí je uživatel odhlášen. Druhé tlačítko *Nastavení* slouží k nastavení svých údajů. Stisknutím tohoto tlačítka se uživateli načtou jeho údaje, které může měnit. Pro potvrzení nových údajů o hráči slouží tlačítko *Aktualizuj údaje*, pro nastavení přihlašovacích údajů slouží tlačítko *Aktualizuj přihlašovací údaje*. Vedoucímu týmu je doporučeno aktualizovat si své data, které po něm nebyly u registrace požadovány!

Pokud se chce hráč stát rozhodčím a nechce mít dvoje přihlašovací údaje, je dobré být přihlášen a registrovat se. Údaje se automaticky vyplní z již registrovaného uživatele a je automaticky zařazen mezi rozhodčí. Více informací k této věci jsou v následující části věnované rozhodčím.

Pro rozhodčí

Stát se rozhodčím může kdokoliv, kdo splňuje povinnosti rozhodčího uvedené na stránce pro registraci a v regionálních propozicích UFL pro Plzeň (cesta *UFL->PROPOZICE->Regionální propozice UFL pro Plzeň*). Pro registraci rozhodčího je nutné vyplnit formulář na stránce *UFL->ROZHODČÍ->Registruj se!*. Nezapomeňte, pokud jste hráčem, přihlaste se a až poté se registrujte! Po správném vyplnění stiskněte tlačítko *Registruj se*, v případě bezproblémového vyplnění formuláře rozhodčí čeká na aktivační email. Ten se mu odešle, až správce potvrdí jeho možnou účast, jako rozhodčí. V případě smazání registrace rozhodčího je o tom uživatel informován opět emailovou

zprávou.

Tak jako hráči se po přihlášení zobrazí v pravém horním rohu informační panel s možnostmi odhlášení a nastavení údajů (viz. Příručka pro hráče). Přihlásit se do systému jde kliknutím na ikonu pod tabulkou na stránkách (obr. 7.3). Pro přihlášení na zápas musí rozhodčí jít zpět na stránku, kde se registroval. Zde už není možná registrace jako taková, ale rozhodčí uvidí seznam zápasů, které se ještě neodehrály. Ke všem těmto zápasům existují tři možnosti, které můžou nastat. Pro přihlášení na zápas máme možnost *PŘIHLÁSIT*, stisknutím této možnosti se přihlásíme na zápas, který chceme pískat. U daného zápasu následně vidíme své příjmení mezi rozhodčími a možnost *ODHLÁSIT*. Po stisknutí této možnosti se odhlásíme ze zápasu. Poslední možnost, která může nastat, je, pokud bude zápas již obsazen rozhodčími, poté nejde se na daný zápas dostat a vidíme *OBSAZENO*.

Stejně jako týmy mají rozhodčí také své fórum pro komunikaci, do této části se dostaneme tlačítkem *R-FÓRUM*, které je umístěno nad zápasy, na které se lze přihlásit. Ovládání je stejné jako u fóra týmu (viz. Příručka pro hráče). Poslední možnost, kterou rozhodčí má, je zobrazení kontaktů na ostatní rozhodčí, na které se dá dostat pomocí tlačítka *KONTAKTY NA OSTATNÍ* umístěné vedle tlačítka *R-FÓRUM*.

Pro administrátora

Administrátor se nemusí nijak registrovat. Přihlašovací účet je automaticky vygenerován. Pro přihlášení do administrátorského prostředí je opět nutné kliknout na ikonu pod tabulkou na stránkách (obr. 7.3). Po správném vyplnění údajů je správce přesměrován do svého prostředí, kde jako první vidí stručný přehled. Převážně všechny vypsané informace jsou zobrazitelné i dále, po kliknutí na kterýkoliv údaj, se zobrazí detailnější popis toho, na co správce zmáčkl. V prostředí je menu umístěno nahoře stránky po celé délce. Jednotlivé záložky jsou rozřazeny podle jejich funkcionality.

První položkou je *Florbal*, která nás po kliknutí přesměruje zpět na úvodní stránku administrátorského prostředí.

Následuje položka *Hráči*, ve které jsou podzáložky *Přidat hráče* a *Seznam hráčů*. Jak už název napovídá, tak pro přidání hráče zvolíme první zmíněnou možnost. Po správném vyplnění a odeslání formuláře tlačítkem *Přidat uživatele* je hráč zapsán na soupisku vybraného týmu. Červené položky jsou povinné, modré nepovinné. Doporučení je, ale vyplnit popřípadě vybrat z možností všechny informace. Podzáložkou *Seznam hráčů* se nám zobrazí stručný seznam všech registrovaných hráčů, kde jsou vedle každého záznamu

tři možnosti. První je tlačítko *Detail*, které nás přesměruje na stránku s detaily vybraného hráče, zde je možnost přejít rovnou k jeho editaci, či smazání pomocí tlačítek *Upravit* a *Smazat*. Stejné tlačítka jsou i dvě zbylé možnosti při výpisu hráčů. Tlačítkem *Smazat* se hráč odstraní. Tlačítkem *Upravit* se zobrazí formulář pro jeho úpravu informací.

Analogicky to je u záložky *Týmy* a *Rozhodčí*. U týmů je třeba si dát pozor, že při smazání týmu se odstraní i jeho všichni hráči! Rozhodčí mají jednu podzáložku navíc a to *Přehled pískání*. Po kliknutí na tuto možnost se nám zobrazí seznam zápasů s výpisem rozhodčích. U každého zápasu je tlačítko *Upravit*, které slouží pro nastavení rozhodčích k danému zápasu. Po kliknutí na toto tlačítko máme možnost pozměnit rozhodčí výběrem jiného, popřípadě uvolnit pozici. Pro uvolnění pozice je zde možnost - - - - - - -. Tlačítkem *Upravit pískající* se nastaví vybraný rozhodčí k zápasu.

Další položkou v menu je možnost *Statistiky*, která má podzáložky *Kanadské bodování*, *Brankářské statistiky* a *Tabulka*. Kliknutím na první zmíněnou možnost se nám zobrazí seznam hráčů seřazený podle celkového počtu bodů. Ke každému záznamu jsou možnosti *Přidej G*, který na kliknutí přidá hráči gól. Další je *Přidej A*, který po stisknutí přidá hráči asistence. Následuje tlačítko *Přidej 2 TM*, který přidá dvě trestné minuty. Poslední možností je *Nastav bodování*, které nás po kliknutí přesměruje na formulář pro ruční zadání hodnot. Tlačítkem *Uprav statistiky* se nastaví zadané hodnoty. Pro usnadnění je zde vyhledávač, podle soupisek jednotlivých týmů. Možností *Hledej hráče* se nám zobrazí seznam hráčů vybraného týmu. Tlačítkem *Zpět na výpis* se dostaneme zpět na výpis všech hráčů. Brankářské statistiky fungují podobně. Po kliknutí na tuto podzáložku se nám zobrazí seznam brankářů. Tlačítkem *Přidej hodnoty* se zobrazí formulář pro přidání hodnot, pro přidání je nutné stisknout tlačítko *Uprav statistiky*. Touto možností přidáme hodnoty ke stávajícím. Tlačítkem *Uprav statistiku* lze upravit celkové hodnoty brankáře. Po kliknutí na toto tlačítko se nám nabídne formulář pro zadání nových hodnot, pro nastavení je nutné zmáčknout tlačítko *Uprav statistiky*. Po kliknutí na poslední podzáložku *Tabulka* se nám zobrazí seznam týmů s možnosti *Přidej výhru*, *Přidej prohru* a *Přidej remízu*. Všechny zmíněné možnosti přidají týmu daný atribut. Posledním tlačítkem je *Nastav bodování*, které slouží pro ruční nastavení hodnot jako v kanadském bodování.

Záložka *Zápasy* slouží pro přehled a správu zápasů. Po kliknutí na podzáložku *Seznam zápasů* se nám zobrazí všechny naplánované zápasy. Opět každý z nich má dvě možnosti *Smazat*, po kliknutí na tuto možnost se daný zápas odstraní a druhá možnost je *Upravit*. Po kliknutí na toto tlačítko se nám zobrazí formulář s daty o zápasu, která lze měnit. Pro upravení zápasu je nutné formulář potvrdit tlačítkem *Uprav zápas*. Další podzáložka je *Rozlosování*. Po volbě této možnosti je správci zobrazen seznam přihláše-

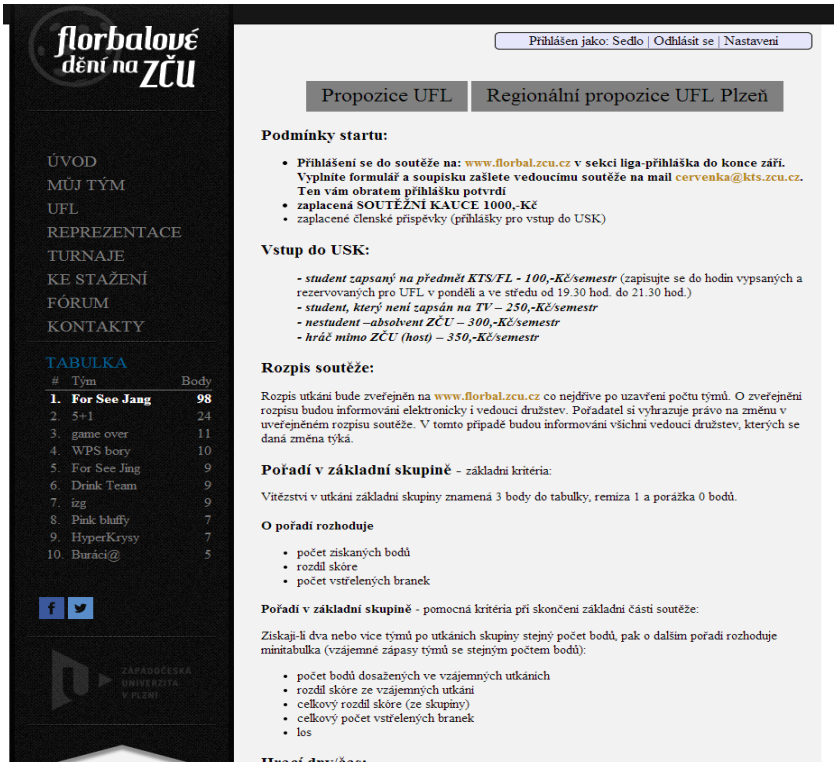
ných týmů a možnost zadání požadovaných kalendářních dat. Po vyplnění dat a zmáčknutí tlačítka *ROZLOSUJ ZÁPASY* se nám zobrazí seznam rozlosovaných zápasů. Před uložením zápasů je možná jejich editace v kolonce *Datum*. Pokud správci nevyhovuje vygenerované datum, lze ho změnit právě kliknutím na konkrétní datum a vybráním nového data. Pro uložení zápasů je nutné zápasy potvrdit, k tomu slouží tlačítko *POTVRĎ ZÁPASY*. Poslední podzáložkou je *Výsledky zápasů*, po stisknutí se nám zobrazí seznam zápasů s jedinou možností *Uprav výsledek*. Toto tlačítko nám nabídne formulář pro vyplnění výsledku zápasu, pro nastavení je nutné potvrdit výsledek tlačítkem *Uprav zápas*. Pro odeslání emailů je zde záložka *Zprávy*, ve které se nachází podzáložka *Odešli email*. Po kliknutí se nám zobrazí stránka s možností vyhledání uživatele podle příjmení (nebo jeho části). Pro vyhledání uživatele je nutné zmáčknout tlačítko *OK*. V případě nalezení uživatele je zobrazen pomocí seznamu s možností *Odešli email*. Další z možností je odeslat emailovou zprávu všem členům vybraného týmu. Pro využití této možnosti je nutné zmáčknout tlačítko *Odešli týmu*. Následují tři rychlé přístupy pro odeslání zprávy uživatelům. Tyto tlačítka jsou *Odešli všem*, které slouží pro odeslání všem uživatelům. Tlačítko *Odešli rozhodčím* je pro odeslání zprávy všem registrovaným rozhodčím. Tlačítko *Odešli vedoucím* slouží pro odeslání zprávy všem vedoucím týmu. Poslední na stránce je seznam všech hráčů i rozhodčích s jedinou možností *Odešli email*. Po kliknutí na kteroukoliv volbu tlačítka (kromě tlačítka *OK*) na této stránce je správce vyzván pro zadání předmětu zprávy a obsahu zprávy. Pro odeslání zprávy je nutné zmáčknout tlačítko *ODEŠLI*.

Předposlední položkou v menu je *Export*. Zde se nachází podzáložky pro vygenerování PDF souboru. Podle názvu podzáložky se vygeneruje příslušné PDF. Jediná podzáložka *Soupiska týmu* nám dává na výběr, kterou soupisku týmu chceme exportovat. K tomu slouží tlačítko *Exportovat*, které nalezneme u každého záznamu.

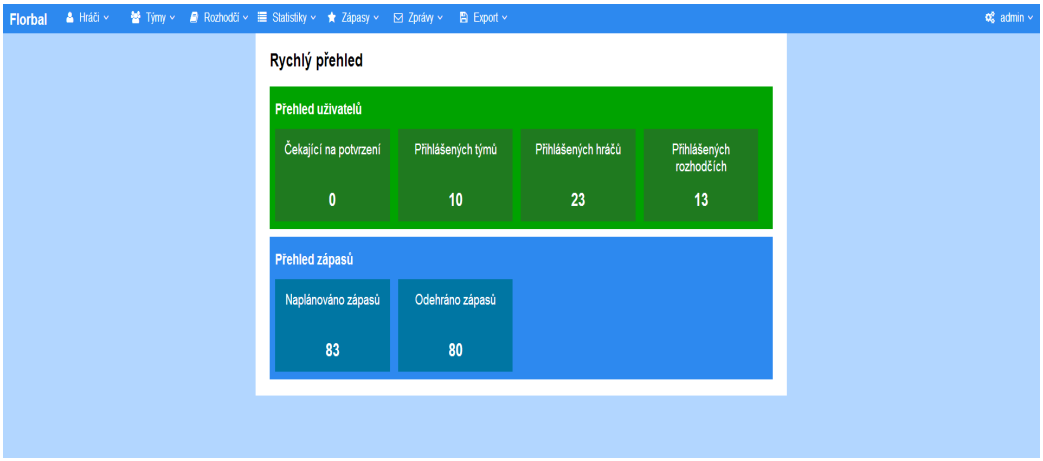
Poslední záložkou je *Admin*. Zde jsou tři podzáložky. První z nich je *Editovat profil*, která po stisku nabídne možnost změny přihlašovacích údajů. Pro nastavení nových údajů je nutné stisknout tlačítko *Změň údaje*. Další podzáložkou je *Nastavení*, zde jsou vypsány stránky, kterým lze měnit obsah. Kliknutím na vybranou stránku se nám zobrazí aktuální obsah stránky, který lze pomocí připraveného editoru měnit. Pro uložení nového obsahu stránky je nutné zmáčknout tlačítko *Ulož změny*. Následuje nastavení aktualit. K tomu slouží tlačítko *Přidej novou* a *Uprav stávající*. Po stisknutí první možnosti jsme vyzváni pro zadání nadpisu aktuality a je připraven editor pro zadání textu článku. Pro její uložení je nutné kliknout na tlačítko *Přidej*. Po kliknutí na druhou možnost se nám zobrazí seznam všech přidaných aktualit s možnostmi *Smazat*, která po stisknutí odstraní danou aktualitu a možnost

Upravit. Po kliknutí na toto tlačítko se nám zobrazí formulář stejný jako pro přidání aktuality. Po jejím upravení je nutné uložit daný článek. K tomu je nutné kliknout na tlačítko *Ulož*. Poslední možností je tlačítko *Nastav nový ročník*. Po kliknutí na toto tlačítko proběhne reset aplikace. Všechna nezbytná data budou odstraněna! Poslední podzáložkou je *Odhlásit se*. Po kliknutí na tuto možnost je administrátor odhlášen z aplikace.

Příloha F – ukázka webového rozhraní



Obrázek 7.4: Ukázka veřejné části po přihlášení.



Obrázek 7.5: Ukázka administrátorské části po přihlášení.