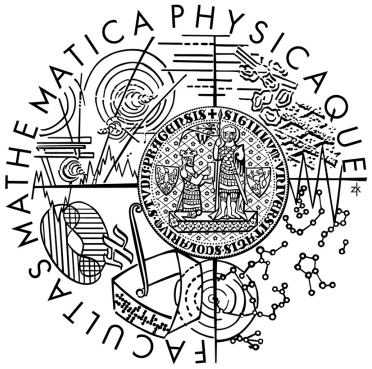
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Antonín Jareš

Informační systém pro sportovní aktivity

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Michal Kopecký, Ph.D.

Studijní program: Informatika
Studijní obor: Obecná informatika

Praha 2015

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne ..... podpis

Na tomto místě bych rád poděkoval svému vedoucímu RNDr. Michalu Kopeckému, Ph.D. za jeho odborné vedení, ochotu a celkovou vstřícnost při konzultacích a vypracováni této bakalářské práce.

Také bych chtěl poděkovat své rodině a přítelkyni za nekonečnou trpělivost a toleranci v průběhu tvorby této práce.

Název práce: Informační systém pro sportovní aktivity

Autor: Antonín Jareš

Katedra / Ústav: Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Michal Kopecký, Ph.D.

Abstrakt: Cílem této práce je navrhnout a implementovat informační systém, který by umožnil amatérským sportovním klubům a klubům nižších soutěží daných sportovních odvětví intuitivní správu obsahu, evidenci událostí, hráčů, statistik a využití fotogalerie na jednom místě. Důraz je přitom kladen na snadnou možnost budoucího rozšiřování aplikace a bezpečnost uživatelů a systému. Výsledný systém pro splnění těchto i dalších požadavků využívá volně šiřitelné technologie – PHP, MySQL, Apache, Framework Nette a další. Aplikace nabízí dostupné funkce s ohledem na úroveň oprávnění příslušející roli uživatele, jako například rozhodčí, trenér, hráč, nebo administrátor.

Klíčová slova: Správa stránek sportovních klubů, PHP, MySQL, Nette

Title: Information system for sports activities

Author: Antonín Jareš

Department: Department of Software Engineering

Supervisor: RNDr. Michal Kopecký, Ph.D.

Abstract: Goal of this thesis is to design and implement an information system allowing amateur sport clubs and clubs participating in lower sport competitions to easily administrate their content, log club events, players, statistics and use a photo-gallery all in one place. This thesis also puts emphasis on easy future extensibility and on its user's security. To achieve these goals, the system uses freely available technologies – PHP, MySQL, Apache, Nette Framework and others. The application provides its functionality with respect to the level of privileges assigned to the user role as referee, coach, player or administrator.

Keywords: Site management for sport clubs, PHP, MySQL, Nette

Obsah

Předmluva

V dnešní době nás počítačové technologie provázejí na každém kroku. Ať už se jedná o nákupy, mezilidský kontakt, či dopravu, v každém odvětví lze nalézt jejich uplatnění. Staly se tak nedílnou součástí našich životů. Technologie samy se každým rokem vyvíjí, mění se, a dostává se jim stále širšího zájmu ze strany veřejnosti. Trendem je převádění dříve plnohodnotných aplikací do prostředí webového prohlížeče, což umožňuje spouštět dané aplikace na široké škále zařízení. I díky tomuto trendu se s webovými technologiemi setkává naprostá většina majitelů počítačů, tabletů a mobilních telefonů.

S tvorbou webových aplikací jsem se v minulosti nesetkal, ale způsob evidence výsledků našeho fakultního basketbalového týmu mne přiměl se nad danou problematikou zamyslet. Nejen náš, ale i naprostá většina ostatních sportovních klubů z nižších soutěží se stále potýká s nutností obtížně a nepohodlně spravovat své webové stránky. Některé kluby proto raději svoje stránky ani nemají. Důsledkem toho je často velmi obtížné, pokud ne nemožné, dohledat jakékoliv výsledky týkající se daného týmu. Proto jsem se rozhodl navrhnout systém, který by možnost správy webových stránek jednoduše zprostředkoval co nejširšímu okruhu uživatelů.

1. Úvod

1.1. Motivace

Cílem této bakalářské práce je zprostředkování jednoduché správy týmů pro sportovní kluby. Většina klubů[1] má i v současné době pouze statické HTML stránky, na které některý člen týmu pravidelně umisťuje výsledky utkání. Pokročilejší projekty potom mohou obsahovat i chatovací místnost[2] pro členy týmu či dokonce fórum[3]. Hlavní nedostatek těchto systémů je nepohodlnost upravování obsahu stránek, které spočívá v přímém přepisování zdrojového kódu webu a zajišťování, popřípadě placení, hostingu. V neposlední řadě hraje svoji roli také nemožnost upravování údajů těmi členy klubů, kteří se ve zdrojovém kódu stránek nevyznají. Obsah webových stránek by přitom měl být upravitelný kýmkoli s dostatečným oprávněním přes samotné webové rozhraní. Tím by byla umožněna pohodlná správa týmu a zároveň úprava obsahu všem jeho členům.

1.2. Cíle práce

Cílem této práce je zprostředkovat pohodlnou a všem dostupnou možnost správy sportovních klubů a soutěží, a tak efektivně eliminovat potřebu každého klubu vést své vlastní webové stránky. Informace o svých klubech a jejich aktivitách by tak uživatelé mohli spravovat na jednom místě. Vzhledem k tomu, že aplikace by mohla sdružovat více klubů kluby určitých soutěží, velmi by při takové konfiguraci usnadnila dohledávání výsledků a orientaci v soutěžích pro širokou veřejnost a kluby samotné. V neposlední řadě by také práce měla klubům poskytnout možnost správy fotogalerií, a tím eliminovat potřebu týmů složitě vytvářet fotogalerie vlastní, popřípadě schraňovat fotografie ve fyzické podobě.

1.3. Struktura práce

V první části práce jsou nastíněny důvody a okolnosti vzniku systému, motivace pro tvorbu aplikace a struktura práce.

Druhá kapitola se zabývá cílenou skupinou uživatelů, popisuje, co se od

[1] http://sokol-kobylisy-hazena-cz.webnode.cz/;http://www.lokovrsovice.cz/; http://www.jirkafaltin.cz/basket/

[2] http://www.basket.matfyz.cz

[3] http://www.basketbalsparta.cz/

dané skupiny osob očekává, a odvozuje omezení s tím spjatá. Následuje rozbor již existujících řešení, od bakalářských prací jiných studentů po komerční projekt větší společnosti.

Třetí kapitola se zabývá specifikací projektu. Zde se nachází bližší pohled na práci včetně zhodnocení jejího omezení a výčet prvků, které by práce měla či mohla implementovat.

Čtvrtá kapitola se věnuje designu a rozboru zvolených technologií. Je zde popsáno rozvrstvení aplikace a komunikace mezi jednotlivými vrstvami. Je zde i nástin dalších použitých technologií, s popisem důvodů jejich zvolení, a jejich krátká charakteristika.

V páté kapitole se nachází popis implementace konkrétních prvků zmíněných v designu a jejich rozbor. Je zde také popsána architektura aplikace a její struktura. V neposlední řadě obsahuje kapitola také sekci věnovanou bezpečnosti.

Šestá kapitola obsahuje programátorskou dokumentaci, která se zabývá instalací a možností rozšiřování aplikace ze strany administrátora.

Poslední část textu je věnována uživatelské dokumentaci. Tato kapitola popisuje navigaci v systému, jeho prvky a možná uživatelská nastavení v něm.

2. Analýza

Tato kapitola se zabývá cílovou skupinou uživatelů, a omezeními, která jsou s touto skupinou spojená. Poté následuje hodnocení již existujících aplikací, které se zabývají podobnou problematikou, a rozbor jejich silných a slabých stránek.

2.1 Cílová skupina

Uživateli, pro které je tato bakalářská práce určena, jsou především neprofesionální sportovci a jejich týmy. Skoro každý profesionální tým[4] má své vlastní stránky na velmi vysoké úrovni a rozhodně nemají potřebu přestoupit na něco jednoduššího. Naproti tomu neprofesionální kluby obvykle nedisponují nadbytečnými finančními prostředky. Je proto žádoucí uvést službu jako bezplatnou.

2.2 Omezení

Vzhledem k tomu, že aplikace cílí na širokou veřejnost bez IT znalostí, služba by měla být intuitivní. Jako nejvhodnější se z tohoto důvodu jeví webová aplikace. Tato volba zároveň nepředstavuje prakticky žádné omezení na znalosti uživatele, protože v dnešní době lze alespoň základní znalost práce s internetem, a připojení k němu, u všech potenciálních uživatelů předpokládat. Navíc toto řešení není příliš omezeno požadavky na operační systém a zajišťuje uživatelům práci s aktuální verzí aplikace bez nutnosti cokoli instalovat. Vzhledem k tomu, že služba má být nabízena zcela zdarma, je potřeba podporovat adekvátní možnost hostingu. V našem případě tedy některou z variant, která je poskytována rovněž zdarma. Tomuto je nutné případně přizpůsobit volbu užitých technologií.

Práce by měla nabízet možnosti správy pro širokou škálu sportů, ale vzhledem k různorodosti pravidel, a celkově rozličným způsobům hry různých sportů, není možné implementovat všechny sporty najednou. Zprvu proto bude služba poskytnuta pro sporty s obdobnými herními systémy. Takové sporty se mohou lišit kupříkladu v bodování, dělení zápasů na části, v počtu hráčů, nebo systému udělování trestů. Implementování rozdílů v určité třídě sportů s podobným systémem fungování by však nemělo být tak náročné jako doplňování zcela nových druhů sportů. Mělo by být v co největší míře pamatováno na umožnění

[4] http://www.basket-nymburk.cz/home/

parametrizace jednotlivých sportů. Dále pak na možnost nezávislé implementace modulů, řídících dané konkrétní aspekty hry. To případně umožní dočasně vynechat některé méně podstatné aspekty. Případné vynechání evidence trestů pro daný sport by nemělo mít významný negativní dopad na funkčnost celé aplikace.

V první verzi se bude tato práce soustředit pouze na jednu skupinu takových sportů, a to na sporty kolektivní. Avšak i v rámci takto vymezené skupiny se sporty v mnoha ohledech liší.

Příkladem mohou být jedny z nejrozšířenějších sportů u nás – fotbal, hokej a basketbal:

Sport	**Tresty**	**Herní úseky**	**Doba úseku**	**Skóre**	**Hráčů**
Fotbal	Červená/žlutá karta	2	45 minut	Bod za gól	11+1
Hokej	2 minuty, 5 minut, ...	3	20 minut	Bod za gól	5+1
Basketbal	Osobní a týmové tresty	4	10 minut	1, 2, 3 body za koš	5

Tabulka 2.0.1 Rozdíl sportů

Je proto žádoucí tuto skupinu sportů nějakým způsobem sjednotit bez ztráty informací o dílčích rozdílech, a umožnit tak budoucím vývojářům jednoduché přidávání sportů do jednotlivých kategorií v aplikaci.

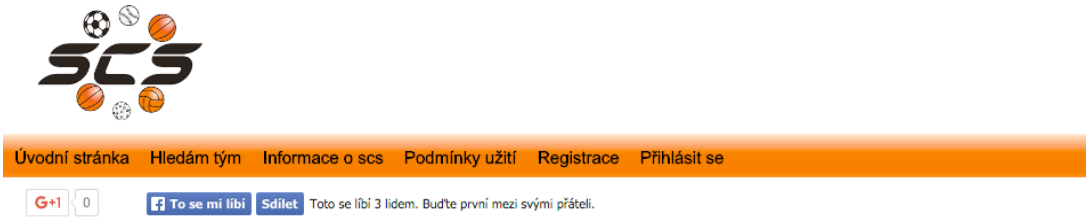
2.3 Jiná existující řešení

Tématem webového informačního systému pro sportovní aktivity se příliš mnoho prací nezabývá, a pokud ano, tak především pouze pro jeden konkrétní klub.

2.3.1 Bc. Jan Krug, VŠE v Praze 2013

Název: Informační systém pro sportovní kluby[5]

Jedná se o diplomovou práci se zaměřením na cloud computing[6] a *Google app engine*[7]. Práce v mnoha ohledech překračuje záměry této bakalářské práce, jak svou složitostí, tak nabízenými funkcemi. Jedná se však o – z mého pohledu zbytečně – komplikovanou záležitost pro běžného uživatele. Dále je systém postaven především na ryzí funkčnosti a na grafický vzhled není kladen takový důraz, což je dle mého názoru u projektu orientovaného na uživatele škoda. Také se na první dojem nejedná o lehce rozšiřitelnou aplikaci. Na druhou stranu práce obsahuje vlastní galerii fotografií a videí, a nabízí široké možnosti spravování tréninků a týmů.



Obrázek 2.0.1 Hlavička systému pana Kruga

[5] http://isis.vse.cz/zp/115259

[6] https://en.wikipedia.org/wiki/Cloud_computing

[7] https://cloud.google.com/appengine/docs

2.3.2 Jan Kolínek, Masarykova Univerzita 2010

Název: Informační systém sportovního klubu[8]

Práce pana Kolínka z Masarykovy Univerzity se již zabývá jak samotným návrhem informačního systému, tak i jeho následnou implementací. Jako ve většině případů se jedná opět o správu jednoho klubu, zde konkrétně házenkářského. Celý systém je napsaný v jazyce *Java* za využití frameworku *Stripes*[9], a na první pohled vypadá velmi obsáhle. Autor se věnuje správě jednoho týmu do detailu, poskytuje možnosti plánování tréninků, evidence docházky, posílání zpráv uživatelům, správu soutěží a evidování zápasů. Práce pana Kolínka se mi, co se funkčnosti týká, zdá jako ideální model této bakalářské práce. Ta by však měla být schopná dané možnosti rozšířit na širší spektrum týmů. Autor práce chytře zvolil cestu webové aplikace, tudíž uživatelé mohou službu lehce využívat bez nutnosti jakékoliv instalace, a mohou tak být s týmem stále v kontaktu. Jako jedno z negativ práce vidím nemožnost rozšíření funkčnosti člověkem, který se na vývoji přímo nepodílel (tedy autor sám) a vůbec využití jazyka Java jako takového. Autor tím přichází o možnost využití spousty dostupných pluginů (například fóra, které si musel sám naprogramovat), které by bylo možné použít, pokud by si zvolil čistě webovou technologii.

Obrázek 2.0.2 Uživatelský profil v práci pana Kolínka

[8] http://is.muni.cz/th/208275/fi_b/bc.pdf
[9] https://stripesframework.atlassian.net/wiki/display/STRIPES/Home

2.3.3 Michal Augustini, VUT v Brně 2011

Název: Analýza a návrh webového informačního systému pro sportovní klub[10]

Práce pana Augustiniho je vcelku obsáhlá a velmi pěkně graficky navržená. Jako ve většině případů se jedná o správu stránek jednoho sportovního klubu. Autor se vydal cestou webové aplikace za využití technologií PHP a MySQL. Funkčnost stránek je navržena tak, že obsah se dynamicky mění pouze v prostředním rámci. Tento způsob je zřejmě rychlejší, ale omezuje možnosti stránky a jejich funkčnost. Systém neeviduje žádné fotografie, neobsahuje svoji fotogalerii, ale pouze odkazuje na galerie na facebookových stránkách. Výsledek této bakalářské práce by naproti tomu měl uživateli nabídnout i jednoduché obrázkové galerie, popřípadě jednoduché zobrazování fotek příslušných k daným událostem (zápasy, tréninky).

Dále jsou v práci pana Augustiniho sekce, ke kterým mají přístup pouze administrátoři a musejí je pravidelně aktualizovat (video měsíce, fotka měsíce, a podobně). Tyto sekce považuji vzhledem k účelu aplikace za zbytečné. Mimo jiné stránky také vedou tabulku nejúspěšnějších hráčů, nominace na nadcházející zápasy a přihlašování na události. Tyto funkčnosti rovněž nejsou nezbytně nutné, ale uživatelům práci se systémem jistě zpříjemní.

Obrázek 2.0.3 Návrh úvodní stránky pana Augustiniho

[10] http://is.muni.cz/th/256598/fi_b/text_prace__lqvovtdo.pdf

2.3.4 Pavel Procházka, VUT v Brně 2012

Název: Návrh informačního systému pro sportovní klub [11]

V této práci se pan Procházka zaměřuje pouze na návrh informačního systému, ne na jeho implementaci. Návrh obsahuje i některé komplexnější služby, než které se objeví v rámci této práce, například evidence plateb a správu docházky na tréninky. Tyto funkce by mohly být možným rozšířením této práce, avšak v první verzi jistě obsaženy nebudou. Jedná se tedy více než o tvorbu konkrétního systému spíše o výzkum možnosti realizace daného informačního systému a kalkulaci potřebných prostředků s tím spojených, nikoliv jeho skutečné naprogramování či realizaci. Jako většina publikovaných prací k tomuto tématu se i tato zaobírá pouze jedním sportovním klubem, konkrétně zaměřeným na florbal.

2.3.5 Sportvia[12]

S tímto placeným softwarem nemám zkušenosti, ale firma na svých stránkách zmiňuje, že se jedná o excelentní řešení správy sportovního klubu. V základní verzi však nabízí pouze správu jednoho klubu a navíc je systém navržen formou offline aplikace šiřitelnou pouze pod operačním systémem Windows. Na první pohled se jedná o aplikaci, která je výhradně určena správci týmu, a nikoliv běžným hráčům, tím spíš ne veřejnosti k nahlížení výsledků a sledování sportovních událostí. Na rozdíl od softwaru Sportvia by měla být tato bakalářská práce dostupná k užití zdarma, přístupná veřejnosti, a ne zaměřena na jeden klub.

---

[11] https://dspace.vutbr.cz/bitstream/handle/11012/11955/2012_BP_Prochazka_Pavel_120360.pdf?sequence=1

[12] http://www.sportvia.eu/

3. Specifikace

Tato kapitola se zabývá bližší specifikací bakalářské práce. V první části je detailněji popsána idea aplikace a její možnosti. Jsou zde diskutovány funkční požadavky aplikace, včetně seřazení podle priority.

3.1 Bližší pohled

Navrhovaná aplikace si klade za cíl poskytnout nebo nahradit systém pro správu týmu klubům, které se účastní nižších sportovních soutěží. Funkčností by tato práce měla nabídnout oproti již dostupným systémům jednodušší správu obsahu, údajů, a organizaci týmu samotného. V důsledku by tak měla aplikace zpřístupnit klubům jednoduchou a intuitivní možnost jak organizovat celý tým nebo soutěž. Aplikace by proto měla být určena jak trenérům, tak hráčům, jejichž role a oprávnění se budou zákonitě lišit (více v sekci 3.2 Nutné funkční prvky).

V neposlední řadě by tato centralizace měla mít za také následek snadnou orientaci v dění neprofesionálních soutěží pro veřejnost. První verze práce se bude zaobírat pouze několika populárními kolektivními míčovými hrami, typu fotbal, basketbal, florbal. Technicky by ale mělo být možné zakomponovat libovolný sport, který dodržuje podobná pravidla jako výše uvedené sporty.

3.2 Nutné funkční prvky

Následující prvky jsou zcela základní a vynechání jakéhokoliv má za následek fatální omezení možností aplikace.

1) Evidence a správa týmů: Jednotlivé sportovní týmy se budou moci do aplikace zaregistrovat a následně spravovat svůj obsah, který bude v základu obsahovat seznam hráčů, informace o týmu, fotogalerii. O informace se bude každý tým starat sám. Po registraci se týmy přihlásí do soutěže. Jejich účast bude následně povolena či zamítnuta daným správcem soutěže.
2) Správa soutěží: Administrátor bude moci evidovat nové soutěže v různých sportech, do kterých se následně budou přihlašovat týmy a vytvářet události.

3) Správa hráčů: Každý hráč bude mít vlastní profilovou stránku s osobními údaji a profilovou fotkou a bude moct být členem více týmů. Pokud se hráči neregistrují sami, budou zaregistrováni trenérem týmu. Profilovou stránku určitého hráče by neměl mít právo upravovat každý, takže je nutné implementovat určité oblasti přístupů podle typu uživatele.
4) Evidence událostí: Po každém utkání, či jiné důležité události, bude událost zaevidována do databáze, popřípadě s popisem.
5) Konfigurovatelnost: Týmy by měly mít možnost vzhledově upravit své domovské stránky, co se týče loga, barvy stránek, či textu.
6) Role přístupu: Každý uživatel bude mít rozdílná oprávnění k přístupu na stránky a jejich správu, v závislosti na jejich roli – administrátor, správce soutěže, trenér týmu, hráč, zaregistrovaný uživatel, nezaregistrovaný návštěvník.

	Správa týmů	**Správa výsledků týmů**	**Přispívání (fotografie, komentáře)**	**Odběr novinek týmů**	**Zobrazení týmů a obsahu**
Administrátor	Všech	Všech	Všude	Všech	Všech
Správce soutěže	V soutěži	V soutěži	-	Všech	Všech
Trenér týmu	Svého	-	Svůj tým	Všech	Všech
Hráč	-	-	Svůj tým	Všech	Všech
Zaregistrovaný	-	-	-	Všech	Všech
Nezaregistrovaný	-	-	-	-	Všech

Tabulka 3.0.1 Popis přístupových práv uživatelů

3.3 Důležité funkční prvky

Následující prvky nejsou nepostradatelné, ale jejich vynechání by znatelně snížilo kvalitu služby pro koncové uživatele.

1) Upozornění: Hráči (a popř. odběratelé) by měli dostávat upozornění o nadcházejících událostech, které se týkají jejich týmu. Je proto potřeba vytvořit nějaký jednoduchý systém pro zprávy, který rovněž bude moci fungovat jako dorozumívací kanál mezi uživateli.
2) Fotogalerie: Každý tým a každá událost bude moci mít svoji fotogalerii, kam budou moci uživatelé s dostatečným oprávněním (správce týmu, hráči, rozhodčí) nahrávat fotografie.
3) Fórum/komentáře: Aplikace by měla obsahovat možnost zpětné vazby uživatelů vůči událostem, popřípadě vůči uživatelům samotným. Fórum by mohlo být zbytečně složité pro tak jednoduchou zpětnou vazbu, proto by bylo pravděpodobně vhodnější využít možnosti komentování, ať už se jedná o kanál *Disqus* [13], či komentáře přímo prostřednictvím služby Facebook. Možné je také zkombinování obou prostředků, kde uživatelé budou moci zanechávat komentáře přímo na stránkách událostí/uživatelů, a fórum využijí pro delší příspěvky a debaty.
4) Evidence docházky na události: Každý hráč se bude moci přihlásit na nadcházející události a zpětně sledovat, čeho se účastnil, a čeho ne.
5) Hráčské statistiky v lize: Každému hráči budou automaticky aktualizované údaje o vstřelených gólech, asistencích, atd. po doplnění údajů do událostí.

[13] http://www.disqus.com

3.4 Zpříjemňující funkční prvky

Následující prvky nejsou ke správné a celkové funkčnosti aplikace nezbytně nutné, avšak jejich implementace by uživatelům mohla zpříjemnit užívání služby.

1) SMS upozornění: Hráči a odběratelé týmu by mohli dostávat SMS upozornění na nadcházející události týkající se jejich týmu.
2) Statistiky nejúspěšnějších hráčů v lize: V každé lize by mohl být seznam nejúspěšnějších hráčů, seřaditelný podle různých kritérií.
3) Hlasování o nejlepší hráče: Uživatelé a veřejnost by mohli hlasovat o nejlepší hráče (v libovolném smyslu) ve vybraných soutěžích.
4) Grafy úspěšnosti a vývoje: Historie úspěchů a výsledků týmů by mohly být evidovány a např. jako grafy zobrazeny v závislosti na čase. Výsledky by mohly být zobrazitelné pouze za určité období. U hráčů by bylo možné zobrazit jejich snažení, schopnosti, a hodnocení za uvedený časový úsek.
5) Chatovací místnost pro každý tým zvlášť: Možnost jak komunikovat s celým týmem najednou. Mohlo by se jednat o sub-forum, které by bylo přístupné pouze danému týmu, či přímo o místnost, kam by mohli lidé psát (forma shoutbox [14]).

[14] http://www.shoutbox.com

4. Design

Tato kapitola rozebírá zvolené technologie, důvod jejich výběru a návrh vrstev systému. Konkrétní implementace jednotlivých technologií a rozdělení systémových vrstev následuje v kapitole *5. Implementace* na straně 22.

4.1. Použité technologie

V dnešní době Webu 2.0 je na výběr z řady technologií a modulů, nabízejících možnost tvorby webových stránek. Konkrétní výběr určitých technologií proto není přímočarý a jednoduchý, ale je třeba se nad zvolením prostředků zamyslet a vybrat ty nejvhodnější.

Vzhledem k omezením nastíněným v sekci 2.2 Omezení na straně 4 jsem se rozhodl pro implementaci pomocí PHP frameworku *Nette* a databáze *MySQL*. Samotný framework a programování se však nepostará o vzhled stránek a jejich interaktivitu. K těmto potřebám jsem proto zvolil knihovnu CSS stylů a JavaScriptu nazvanou *Bootstrap*[15].

4.1.1 Proč framework?

Framework je jednoduše řečeno nadstavba nějakého programovacího jazyka. *"Softwarová struktura, která slouží jako podpora při programování, vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovny a podporu pro návrhové vzory."* [1]

4.1.2 Proč PHP?

- Zprovoznění systému s PHP je velmi jednoduché.
- Drtivá většina hostingů PHP podporuje. Hostingy zdarma podporují téměř výhradně pouze PHP.
- Je jedním z nejpoužívanějších jazyků k tvorbě webových aplikací.
- K PHP existuje velmi obsáhlá snadno dostupná dokumentace[16].
- Zvolení PHP má za následek velmi nízké finanční nároky, jak z hlediska softwaru (většina nástrojů spojených s PHP je zdarma), tak z hlediska hardwaru, protože PHP může být spuštěno na téměř jakémkoliv serveru.

[15] http://getbootstrap.com/

[16] http://php.net/docs.php

- PHP nabízí velkou flexibilitu, co se týče databází, na rozdíl od aplikací postavených na ASP. NET, které využívají zpravidla MS-SQL databáze.
- Vzhledem k obrovské komunitě vývojářů PHP je dostupných mnoho open source knihoven a nalézt odpovědi při řešení problémů je mnohem snazší.

4.1.3 Proč Nette?

- Jedná se framework postavený nad PHP, který jsem vybral jako hlavní jazyk pro psaní aplikační logiky.
- Je vyvíjen českou komunitou s uživateli, kteří se aktivně podílí na jeho vývoji, vylepšeních, a kteří pomáhají začátečníkům.
- Implementuje takzvaný MVP (Model-View-Presenter) koncept, který velmi usnadňuje práci a následnou správu projektu.
- Je jednoduchý na naučení a pochopení.
- Díky jazyku *Latte*[17], který je využit při tvorbě HTML šablon, velmi urychluje a zlehčuje tvorbu jednotlivých stránek.

4.1.4 Proč MySQL?

- Implementace a správa MySQL databází je jednoduchá, rychlá a intuitivní.
- Předpokládaná velikost dat nevyžaduje nasazení nějaké extra výkonné databáze.
- MySQL databáze patří pro očekávanou velikost dat mezi co do funkčnosti dostačující a přitom nejrychlejší[18], co se výkonu týká.
- Rozsáhlá komunita podporující vývojáře 24/7.
- MySQL je opensource a zdarma.
- Hostingy zdarma podporují především MySQL.

4.1.5 Proč Boostrap?

- Samostatná knihovna se snadnou možností nasazení.
- Rozmanitost a možnost vzhledově prvky upravovat.
- Zdarma, šířeno pod volnou licencí.
- Aktivní komunita podílející se na vývoji.
- Automatické přizpůsobení mobilním zařízením.

[17] http://doc.nette.org/cs/2.2/templating#toc-latte

[18] http://arxiv.org/pdf/1205.2889.pdf;
http://db-engines.com/en/system/Microsoft+SQL+Server%3BMySQL%3BOracle

4.2 Základní struktura aplikace

Celá aplikace by měla být rozdělena do několika samostatných celků nebo vrstev pro zřetelné oddělení povinností a usnadnění její následné správy. Tento přístup odpovídá takzvanému *Single Responsibility Principle*[19], podle kterého by se každá část aplikace měla zabývat jednou dílčí odpovědností v systému a ničím jiným. Stejně tak by více částí nemělo sdílet odpovědnosti mezi sebou, aby se tak předešlo možnému vzniku konfliktů.

Nejnižší takovou vrstvou je *vrstva model*, která se stará o úschovu dat a jejich výdej zbytku aplikace. Pod tuto vrstvu také spadají třídy, které se podílejí na úpravě dat v rámci běhu aplikace, ale ne na jejich prezentaci. Jsou to například třídy starající se o přihlašování uživatelů nebo směrování příchozích požadavků na odpovídající části aplikace.

Další vrstvou je *vrstva prezentační*. Do této vrstvy spadají všechny třídy typu *Presenter*, které řídí nahrávání dat do HTML šablon, tvorbu formulářů a spouštění aplikační logiky na straně serveru.

Poslední vrstvou je *vrstva view*, neboli front-end. Skládá se ze všech komponent aplikace, které se přímo podílí na úpravě vzhledu finálně generovaných stránek. Do této vrstvy patří především *.latte* šablony a JavaScriptové soubory či CSS formátování.

Výše zmíněné rozložení odpovídá návrhovému vzoru *MVP (Model-View-Presenter)*, který je základním předpokladem pro práci ve frameworku Nette. Vzor *MVP* je velmi podobný[20] známějšímu vzoru *MVC (Model-View-Controller)*. Rozdělení aplikace do těchto tří vrstev je v dnešní době standardem napříč tvorbou webových aplikací.

[19] https://en.wikipedia.org/wiki/Single_responsibility_principle

[20] http://stackoverflow.com/questions/2056/what-are-mvp-and-mvc-and-what-is-the-difference

4.2.2 Vrstva Model

Většina projektů v dnešní době je úzce spjatá s daty, se kterými pracuje. Proto je velmi důležité tato data správně uchovávat. Při špatném navrhnutí modelu, ať už se jedná o opomenutí sloupců v tabulce databáze, neoptimální zvolení typu dat, či implementaci pomalých algoritmů pro vyhledání, případně úpravu dat, je projekt vystaven riziku dramatického snížení rychlosti a mnoha dalším problémům. Model se dá zpětně upravit, avšak takové úpravy bývají mnohdy vyčerpávající a mají zpravidla za následek dlouhodobé výpadky, potřebu upravit stávající zdrojový kód prezentační vrstvy a zvýšené nebezpečí poškození dat. V takovém případě se na opravách zbytečně plýtvá časem, který by mohl být využit k zlepšení aplikace nebo údržbě. Správné navržení modelu je tedy základní nutností, jejíž zanedbání může způsobit zbytečné komplikace.

Model se stará o úschovu a správu dat a jejich vydávání, pokud je požádán. Měl by jako jediný komunikovat s databází a mít k ní přístup. Také může implementovat algoritmy týkající se přímo dat. V rámci této práce se o úschovu dat stará relační databáze *MySQL*. Datové schéma databáze by mělo být ve smysluplném formátu (třetí normální forma[21]) a být schopné ukládat všechny potřebné údaje o stavu systému. V ideálním případě by se databáze měla rovněž sama postarat o potřebné reakce na mazání a úpravu záznamů pomocí triggerů a integritních omezení.

Do této vrstvy patří také všechny prvky systému, které obstarávají přímou komunikaci mezi databází a zbytkem aplikace, popřípadě manipulují s aplikačními daty, ale nemají nic společného s jejich prezentací. V prvním případě se jedná o dvě skupiny tříd, nazvané *Repositories* a *Services,* které se chovají jako prostředník mezi databází a prezentační vrstvou. Třídy ve skupině *Repositories* smějí data číst a předávat výše, zatímco třídy ve skupině *Services* data smí upravovat a mazat. Všechny dotazy/příkazy směřující na databázi z prostředí aplikace jsou vedeny výhradně přes tyto třídy. Další samostatnou třídou je tzv. *Authenticator*, který má na starosti přihlašování uživatelů a kontrolu jimi zadaných údajů. V neposlední řadě vrstva modelu obsahuje třídu *RouterFactory*, která se stará o směrování požadavků na odpovídající *presentery* v prezentační vrstvě.

[21] https://en.wikipedia.org/wiki/Third_normal_form

4.2.2.1 Vrstva Model – Skupina Repository

Skupina tříd *Repository* má za úkol získávat záznamy z databáze a následně je předávat výše, dle požadavků *Presenteru*, který metody daného *Repository* volá.

Tato část modelu by data neměla nijak upravovat, ale pouze je vydávat na základě přijatých požadavků. Každé databázové tabulce, ze které je potřeba získávat data, by měla příslušet právě jedna třída *Repository*, která danou tabulku obsluhuje. Zpravidla je vhodné, aby všechny třídy *Repository* dědily svá rozhraní od jedné rodičovské třídy *BaseRepository*, která implementuje základní metody, jež jsou pro všechny potomky shodné (například vypsání všech položek tabulky, obstarání položky podle jejího id, a jiné).

4.2.2.2 Vrstva Model – Skupina Service

Skupina tříd *Service* má za úkol záznamy databáze upravovat na základě požadavků *Presenterů*. Opět by každé databázové tabulce měla příslušet právě jedna třída *Service*, která ji obsluhuje. Jakýkoliv požadavek na úpravu dat by měl projít výhradně přes třídu *Service*. Tím je správa všech dat efektivně centralizována do jedné skupiny tříd, díky čemuž není potřeba ošetřovat bezpečnost zacházení s databází na více úrovních aplikace.

4.2.2 Vrstva View

Vrstva *View* má na starost vzhled aplikace a její vykreslování. Obsahuje všechny vizuální komponenty systému a je úzce propojená s prezentační vrstvou. Ve frameworku Nette je vrstva *view* efektivně řešena pomocí takzvaných *Latte*[22] HTML šablon, které sami implementují několik bezpečnostních opatření proti narušení stránek (například proti XSS[23] pomocí zneškodnění „nebezpečných“ znaků). Vrstva však sama od sebe neovlivňuje vzhled aplikace (barvy, rozvržení stránky atd.), ale pouze zdrojový kód stránek. Je proto žádoucí do aplikace zakomponovat JavaScript a CSS styly, aby stránky mohly být interaktivní a mít propracovanější vzhled.

Správný návrh a implementace vrstvy *View* má znatelný dopad na funkčnost, rychlost a vzhled celého projektu. Vrstva by měla být psána dynamicky v závislosti

[22] http://doc.nette.org/cs/2.2/templating#toc-latte

[23] http://cs.wikipedia.org/wiki/Cross-site_scripting

na příchozích datech z vrstvy *Presenterů*, nikoliv „natvrdo“ jako statické HTML stránky.

Vestavěný systém šablon vývojáři velmi usnadňuje práci. V podstatě se jedná o možnost propojení funkčnosti PHP se psaním HTML kódu. Šablony nejen umožňují využívat cykly a podmínky, ale mohou i zjistit, zda je aktuální uživatel přihlášen, v jaké je roli, a následně podle daných informací upravit svůj obsah. Šablony lze vzájemně vkládat do sebe, čímž lze docílit společných prvků na více šablonách, například zobrazování menu na každé stránce aplikace.

Při správné implementaci této vrstvy by měla být její následná údržba vcelku jednoduchá. Šablony lze také rozšiřovat vlastními makry, kde definování makra má za následek vykonání PHP kódu nad určitým blokem dat. O samotný vzhled se starají CSS šablony, které jsou standardem ve stylování HTML stránek.

Namísto ručního psaní CSS šablon a JavaScriptových souborů jsem se rozhodl využít volně dostupnou knihovnu *Boostrap*, která obsahuje řadu předem naprogramovaných prvků, jež lze jednoduše zakomponovat. V *Latte* šablonách tak stačí jednotlivým HTML elementům přiřadit určitou třídu a jejich vzhled se změní podle CSS šablon knihovny *Bootstrap*.

4.2.3 Prezentační vrstva

Prezentační vrstva je složena ze všech tříd typu *Presenter* a představuje propojení mezi vrstvou *view* a *modelem.* Pracuje s daty pomocí tříd *Repository* a *Service*, popřípadě data zpracovává a upravuje (lokálně, nikoliv v databázi) a odesílá do šablon. Stará se o zpracování požadavků uživatele, bezpečnost a celkově většinu běhu aplikace.

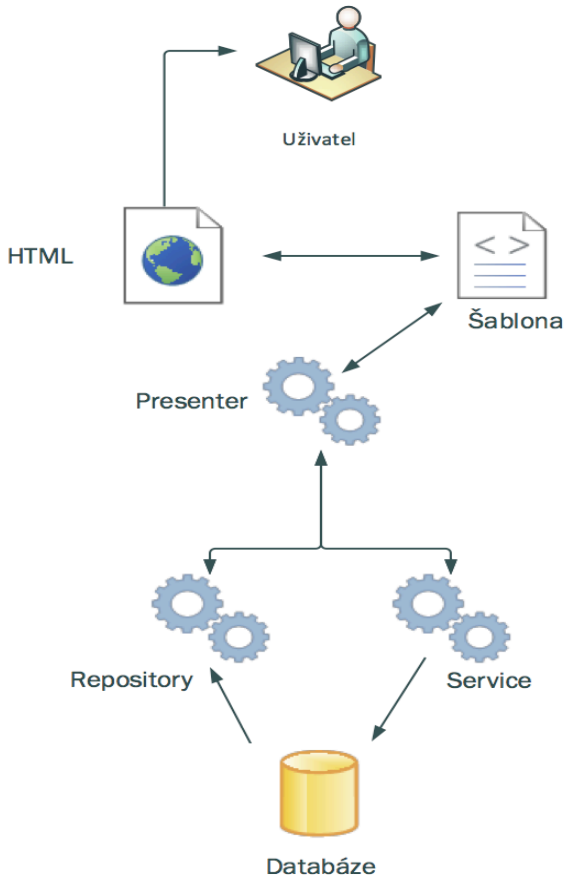
Správná implementace prezentační vrstvy je stěžejní. Jakékoliv nedostatky v této vrstvě mohou mít pro aplikaci fatální následky. V lepším případě nefunkčnost určitých komponent, v horším případě nefunkčnost celých stránek a výdej nekorektních dat. V případě nejhorším pak hrozí ohrožení celé databáze kvůli mezerám v bezpečnosti.

Tato vrstva také implementuje bezpečnostní opatření zamezující nesprávným skupinám uživatelů spuštění jim nepovolených případů užití. Všechny uživatelské požadavky jsou přes tuto vrstvu směřovány.

Zpravidla je pro každou specifickou skupinu stránek (například stránky zabývající se profilem uživatelů) vytvořena jedna třída *Presenter*, která následně spravuje jí příslušné šablony a zpracovává další požadavky. Při jakékoliv činnosti uživatele je z kontextu šablony zavolána příslušná metoda odpovídající třídy *Presenter*, v rámci které jsou výsledky činnosti uživatele dále zpracovány. Následně probíhá komunikace s vrstvou modelu, pokud je potřeba. Prezentační vrstva se zpravidla také stará o vytváření formulářových komponent, určených k vykreslení v jí příslušných šablonách.

4.2.4 Komunikace mezi vrstvami

Komunikace v aplikaci by tedy měla probíhat následovně: Uživatel vyšle na server požadavek. Požadavek je systémem přesměrován na třídu *Presenter*, která na jeho základě zvolí odpovídající HTML šablonu k vykreslení. Do šablony *Presenter* nahraje data potřebná ke správnému zobrazení výsledku, a následně je z této šablony vygenerován kód stránky ve formátu HTML. Tento kód je v konečném kroku odeslán na zařízení uživatele, kde ho může prohlížeč interpretovat.



Obrázek 4.0.1 Komunikace vrstev aplikace

4.3 Rozdílnost sportů

Jak již bylo nastíněno v sekci *2.2 Omezení* na straně 4, tato práce má za cíl zprostředkovat možnost správy týmů pro širokou škálu sportů z určité skupiny. Sporty v jedné skupině se však mohou v mnoha aspektech lišit viz *Tabulka 2.0.1 Rozdíl sportů* na straně 5, a je proto potřeba umožnit systému na tyto rozdíly nějak reagovat.

Ideálním přístupem by bylo evidování rozdílů jednotlivých sportů a vytvoření modulu, který by specifikace daných sportů zpracoval a načetl odpovídající šablonu, popřípadě specificky reagoval na akce uživatele. Zmíněná šablona by tak mohla data zobrazit stylem specifickým pro daný sport. Na takovou parametrizaci se jednoznačně hodí formát JSON[24], ve kterém by mohly být uloženy jak jednotlivé specifikace pravidel pro daný sport, tak i průběhy celých událostí. Tento soubor by byl následně načten modulem, který by dále nahrál buď standardní šablonu, nebo šablonu zvláště vytvořenou pro daný sport. Tento způsob by měl umožnit adekvátní promítnutí odlišností daných sportů v aplikaci. Také by měl zajistit zjednodušenou možnost přidávání nových sportů do aplikace dalšími vývojáři.

[24] https://en.wikipedia.org/wiki/JSON

5. Implementace

Tato kapitola se zabývá popisem vytvořené aplikace z technické stránky. V první části je popsána architektura aplikace z pohledu vrstev *modelu*, *presenterů* a *views*, které byly popsány v 4.2 Základní struktura aplikace na straně 16.

Následuje seznámení se strukturou aplikace a nakonec detailní rozbor implementace jednotlivých vrstev.

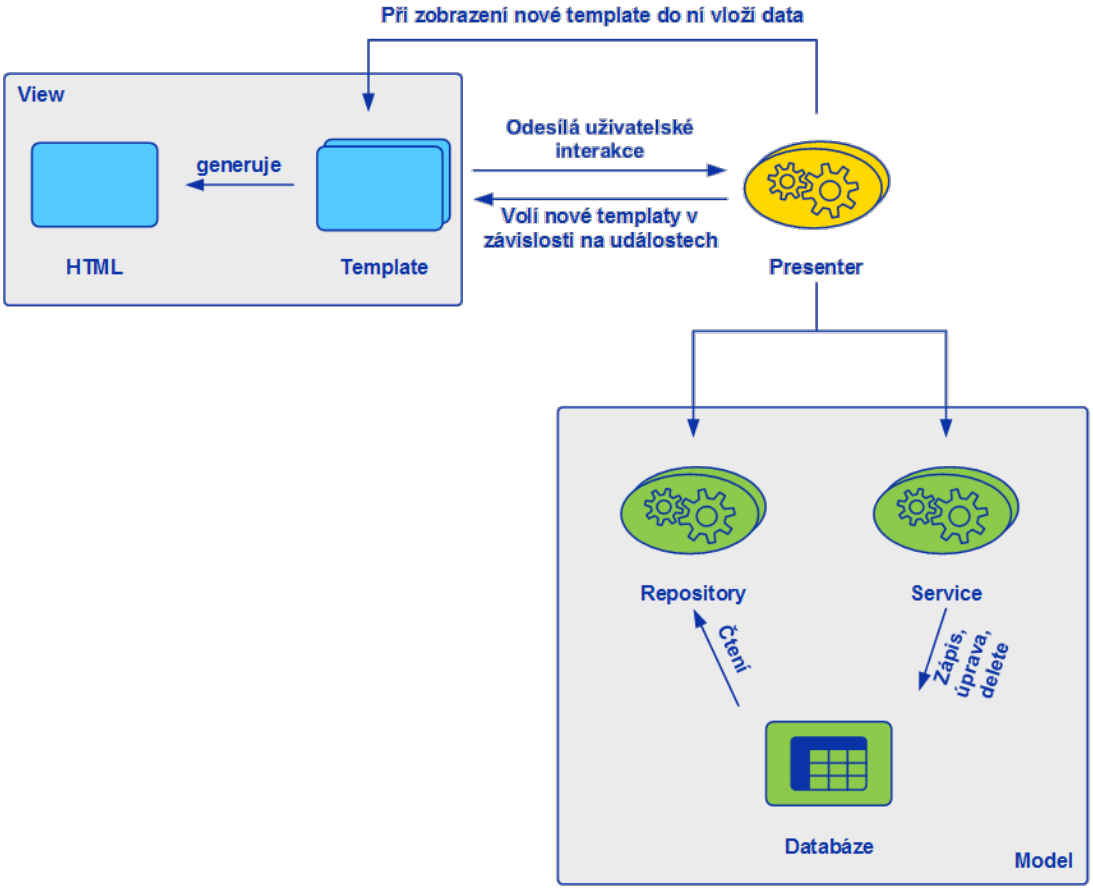
5.1. Architektura aplikace

Aplikace je rozdělena na tři vrstvy MVP(Model-View-Presenter), kde každá vrstva má na starosti jinou část aplikace (viz. 4.2 Základní struktura aplikace).

Model se stará o ukládání dat v databázi a jejich následnou správu.

Presenter se stará o volbu správných šablon, které se mají uživateli interpretovat, o nahrání dat do nich, a o obsluhu uživatelské interakce.

View obsahuje šablony, které jsou na serveru intepretovány a převedeny do HTML kódu, který je zobrazen koncovému uživateli.

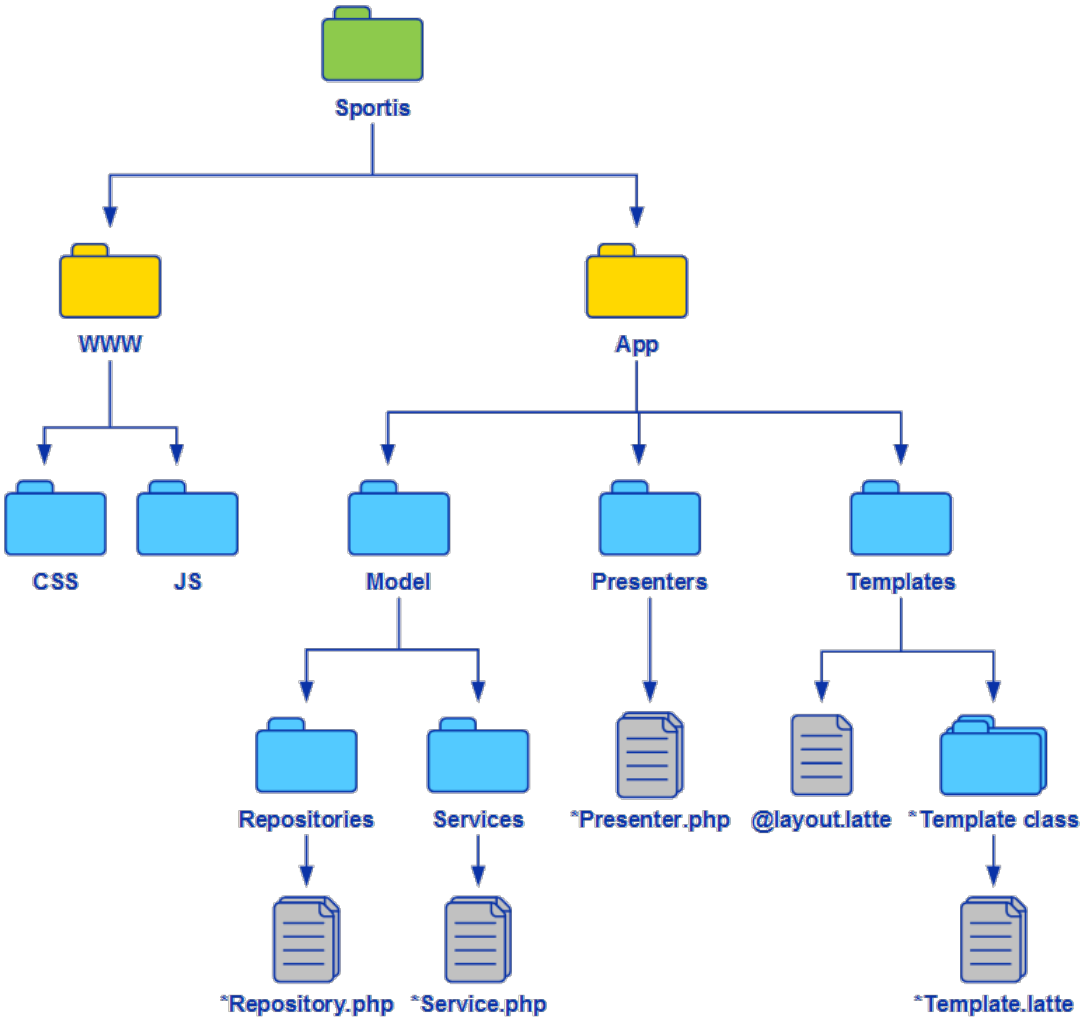


Obrázek 5.0.1 Architektura aplikace

5.1.1 Struktura aplikace

Adresářová struktura aplikace se skládá z hlavní složky *App*, která v dalších podsložkách obsahuje veškerou aplikační logiku spouštěnou na straně serveru. Zde se nacházejí všechny .php soubory, ať už se jedná o implementaci *Presenterů*, nebo tříd *Repository* a *Service*. Dále se zde nacházejí také *.latte* soubory, které reprezentují jednotlivé šablony.

Složka *WWW* naopak obsahuje soubory, které aplikaci ovlivňují pouze ze strany klienta, například CSS šablony a JavaScriptové soubory. Ty se starají o zobrazování modálních oken, možnosti změn barev v nastavení a celkově o zpříjemnění užívání stránek uživatelem.



Obrázek 5.0.2 Struktura aplikace

5.1.2 Model

Vrstva modelu obsahuje databázi a dále dvě skupiny tříd *Repository* a *Service*. *Repository* a *Service* jsou jediné skupiny tříd, které mohou s databázovou vrstvou komunikovat. *Repository* mohou nad databází provádět operaci read, zatímco *Service* mohou provádět operace create, update, delete.

5.1.2.1 Databáze

Správné uchovávání dat patří mezi jeden z nejdůležitějších požadavků na webovou aplikaci. V této práci je vrstva modelu reprezentována relační databází MySQL. Databázové schéma splňuje třetí normální formu[25] a obsahuje triggery, které obstarávají pomocnou funkčnost. Díky použití kaskádovitého mazání[26] není potřeba explicitně mazat data ve spojovacích tabulkách.

Například tento trigger smaže ze systému notifikace, které už žádný uživatel nemá ve svém archivu:

```
CREATE TRIGGER `trigger_after_notification_user_delete`
AFTER DELETE ON `notification_user`
FOR EACH ROW
BEGIN
SET @t1 = (
  SELECT COUNT(*)
  FROM notification_user
  WHERE notification_id=OLD.notification_id);
CASE
  WHEN @t1 = 0 THEN
    DELETE FROM notification WHERE id = OLD.notification_id;
  ELSE BEGIN END;
END CASE;
END
```

Popis databázových tabulek je k nalezení na konci této práce v části *3. Popis databázových tabulek.*

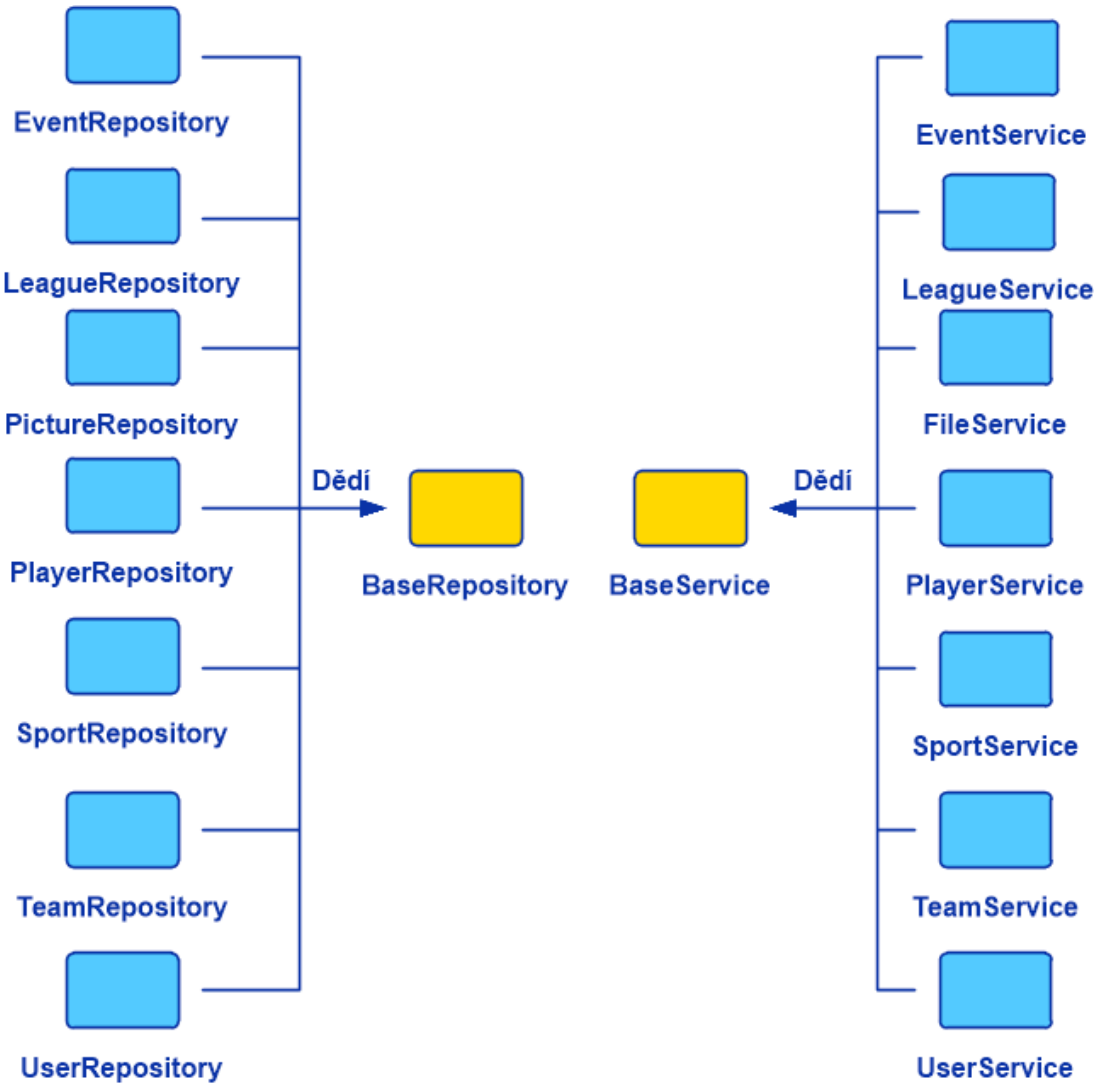
[25] https://en.wikipedia.org/wiki/Third_normal_form

[26] http://dba.stackexchange.com/questions/44956/good-explanation-of-cascade-on-delete-update-behavior

5.1.2.2 Repositories a Services

Komunikace s databází je striktně prováděna pouze přes tyto třídy. Každá tabulka, která pouze nereprezentuje vztah mezi entitami, na sebe má navázány právě jednu třídu typu *Repository* a právě jednu třídu typu *Service*. K tabulkám, které reprezentují vztahy mezi entitami, mohou přistupovat obě třídy, k nimž entity patří. Navíc aplikace obsahuje rodičovské třídy *BaseRepository* a *BaseService*, které implementují základní funkčnosti nad tabulkami a od nichž všechny ostatní třídy *Repository* a *Service* dědí.

Kompletní popis tříd Repository a Service a jejich metod, stejně jako další detaily implementace, je k nalezení ve vygenerované dokumentaci na přiloženém CD.



Obrázek 5.0.3 Hierarchie tříd Service a Repository

5.1.3 Presentery

Prezentační vrstva se skládá z následujících tříd:

1. *BasePresenter* -> rodičovská třída, od které všechny ostatní dědí. Všechny dostupné třídy typu *Repository* a *Service* se načítají v této třídě. Jelikož jsou ostatní *presentery* potomky *BasePresenteru,* mají k nim tak přes dědičnost také přístup. Dále tato třída implementuje načtení notifikací uživatele a správného CSS stylu, který si uživatel zvolil.
2. *CliPresenter* -> třída starající se o spouštení cron úloh při spuštění aplikace z prostředí konzole
3. *ErrorPresenter* -> třída starající se o vykreslování všech chyb v aplikaci (využívá různé šablony v závisloti na http kódu chyby)
4. *EventPresenter* -> třída, která se stará o všechny možné případy užití nad událostmi (evidence, úprava, a jiné)
5. *HomepagePresenter* -> třída, která se stará pouze o úvodní stránku aplikace
6. *LeaguePresenter* -> třída, která má na starosti prezentaci sportovních soutěží, například zobrazení aktuální tabulky výsledků
7. *NotificationPresenter* -> třída, která má na starosti mazání notifikací a jejich označení za přečtené
8. *PlayerPresenter* -> třída starající se o hráčské profily. Stará se o jejich vytváření, změnu a nahrávání fotografií.
9. *ProfilePresenter* -> třída, která má na starosti uživatelské profily. Všechna uživatelská nastavení, změnu stránek profilu, změnu avatara, má na starosti tato třída.
10. *SignPresenter* -> třída starající se o registraci nových uživatelů a jejich přihlašování.
11. *TeamPresenter* -> třída, která má na starosti vše týkající se týmů. Ať už se jedná o jejich výpis, registraci, smazání, či editaci.

5.1.4 Views

Vrstva *View* obsahuje všechny šablony (v jazyce *Latte*) pro vykreslování HTML stránek a volitelné styly ve formě CSS souborů. Každému *presenteru* přísluší právě jedna šablona za každou HTML stránku, kterou může zobrazit.

Šablony tedy představují všechny stránky, které jsou v rámci aplikace zobrazitelné. Tato aplikace obsahuje následující šablony:

1. *@layout.latte* -> kostra šablon implementující prvky, které jsou sdíleny napříč celou aplikací (například horní lišta s menu). Ostatní šablony do této kostry pouze nahrávají svůj obsah pomocí předem definovaných bloků
2. *Error*
    a. *403,404,XXX.latte* -> šablona definující obsah, který se má vykreslit na základě chyb v aplikaci
    b. *errorLayout.latte* -> kostra chybových šablon obsahující jednoduché formátování
3. *Event*
    a. *checkIn.latte* -> šablona umožňující úpravu výsledků zápasů mezi týmy
    b. *checkInList.latte* -> šablona zobrazující seznam zápasů, kterým může rozhodčí spravovat výsledky
    c. *default.latte* -> standardní šablona zobrazující informace o události
    d. *edit.latte* -> šablona umožňující editaci dané události
    e. *editList*.latte -> šablona zobrazující seznam událostí, které daný uživatel může editovat
    f. *layout.phmtl* -> kostra pro ostatní šablony třídy *Event*
    g. *mine.latte* -> šablona s událostmi, kterých se daný uživatel účastnil/ bude účastnit v budoucnosti
    h. *register.latte* -> šablona pro evidování nové soukromé události
    i. *registerMatch*.latte -> šablona umožňující rozhodčím evidovat nové utkání
    j. *upcoming.latte* -> šablona zobrazující nadcházející události
    k. *Football* -> adresář s příkladem využití vlastního zobrazení událostí pro určité sporty, viz 6.2 Přidání nového sportu na straně 33
4. *Homepage*
    a. *default.latte* -> šablona domovské stránky aplikace
5. *League*
    a. *tables.latte* -> šablona zobrazující aktuální pořadí a statistiky týmů napříč aplikací

6. *Player*
    a. *default.latte* -> šablona zobrazující hráčský profil
    b. *edit.latte* -> šablona umožňující editaci hráčského profilu
    c. *playerLayout.latte* -> kostra šablon *default.latte* a *edit.latte*
7. *Profile*
    a. *default.latte* -> šablona zobrazující uživatelský profil
    b. *edit.latte* -> šablona umožňující editaci uživatelského profilu
    c. *myTeams.latte* -> šablona zobrazující uživatelovy týmy, ve kterých hraje, a ve kterých působí jako trenér
    d. *profileLayout.latte* -> kostra šablon *default.latte* a *edit.latte*
    e. *settings.latte* -> šablona s uživatelským nastavením
8. *Sign*
    a. *in.latte* -> šablona pro přihlášení uživatele do systému
    b. *register.latte* -> šablona pro registraci nového uživatele
    c. *signLayout.latte* -> kostra šablon *in.latte* a *register.latte*
9. *Team*
    a. *addCoach.latte* -> šablona modálního okna, ve kterém lze zaregistrovat nového trenéra pro tým
    b. *addPlayer.latte* -> šablona modálního okna, ve kterém lze zaregistrovat do týmu nového hráče
    c. *bulletinBoard.latte* -> šablona reprezentující soukromou nástěnku týmu, kam členové mohou přidávat příspěvky
    d. *default.latte* -> šablona zobrazující údaje o týmu
    e. *edit.latte* -> šablona umožňující editaci daného týmu
    f. *editVisual.latte* -> šablona umožňující úpravy vzhledu domovské stránky týmu
    g. *gallery.latte* -> šablona galerie týmu
    h. *layout.phtml* -> kostra šablon *TeamPresenteru*
    i. *list.latte* -> šablona obsahující výpis všech týmů v systému
    j. *register.latte* -> šablona umožňující registraci nového týmu

Mimo šablon obsahuje aplikace také několik složek s CSS soubory, které určují styl celé aplikace. Různé styly si mohou uživatelé navolit v nastavení.

5.3 Bezpečnost

Velmi důležitým aspektem každé webové aplikace je především bezpečnost. Ať už se jedná o možnost narušení celého systému pomocí spuštění cizího kódu na serveru, či odcizení uživatelských údajů a jejich následné dešifrování, bezpečnost hraje významnou roli při zavádění webových aplikací. Díky použití frameworku Nette je při implementaci těchto bezpečnostních opatření vývojáři ulehčena práce.

5.3.1 XSS[27], SQL Injection[28], CSRF[29]

„Nette Framework klade velký důraz na bezpečnost aplikací, a proto úzkostlivě dbá i na dobré zabezpečení formulářů. Dělá to zcela transparentně a nevyžaduje manuálně nic nastavovat. Ochrání vaše aplikace před útokem Cross Site Scripting (XSS) i Cross-Site Request Forgery (CSRF), odfiltruje ze vstupů kontrolní znaky, ověří validitu UTF-8 kódování nebo jestli nejsou položky vybrané v select boxech podvržené atd.“ [2]

O hrozby typu XSS je automaticky postaráno díky užití formulářů frameworku Nette, které ve všech textových řetězcích získaných od uživatele automaticky escapují[30] speciální znaky, jejichž použití by mohlo mít za následek kompromitaci systému.

Hrozby typu CSRF programátor může lehce ošetřit přidáním jednoho řádku kódu při vytváření formulářů:

```
$form->addProtection('Vypršel ochranný časový limit, odešlete prosím formulář ještě jednou');
```

O hrozbu SQL injection se musí postarat programátor sám, ale díky frameworku Nette lze využít při volání vrstvy databáze speciální formát textových řetězců, které prevenci usnadňují. V takovém formátu je za každý znak otazníku v řetězci postupně dosazen zadaný parametr, který je správně escapován pro použití v SQL.

[27] https://cs.wikipedia.org/wiki/Cross-site_scripting

[28] https://cs.wikipedia.org/wiki/SQL_injection

[29] https://cs.wikipedia.org/wiki/Cross-site_request_forgery

[30] https://cs.wikipedia.org/wiki/Escapov%C3%A1n%C3%AD

Příklad:

```
#$teamId je parametr získaný z url (display/team?teamId=9)

$this->database->query("SELECT * FROM team WHERE id=" .
$teamId . ";")

#špatně -> v takovém případě může útočník zapsat do GET
#parametru například „; DROP TABLE user;“ a zrušil by tím
#celou tabulku s uživateli

$this->database->query("SELECT * FROM team WHERE id=?;",
$teamId)

#správně -> za využití Nette formátu textových řetězců
#v takovém případě je řetězec $teamId převeden na jeden
#validní SQL příkaz, což v tomto případě znamená využití
#pouze příkazu SELECT a žádného jiného, který by mohl
#následovat vložením speciálních textových řetězců
```

5.3.2 Solení hesel

Vždy se však může stát, že některé hrozby do systému proniknou, a poté by mohly být uživatelské údaje získány třetí stranou. V takovém případě by se mohlo například podařit rozšifrovat uživatelská hesla (ať už pomocí metodou *rainbow tables*[31], útoku *brute-force*[32], či *slovníkového útoku*[33]) a následně by mohli být uživatelé ohroženi i na jiných portálech, kde mají zaregistrovaný účet se stejným emailem a heslem. Aby se takovému případu předešlo, hesla v aplikaci využívají solení[34].

Při registraci každého uživatele je vytvořena náhodná[35] sůl, a pomocí PHP funkce crypt[36] je heslo solí zašifrováno. Následně je celé heslo zahašováno pomocí algoritmu SHA-1[37] a hash se solí je následně uložen do databáze. Při každém přihlášení je poté sůl stejným způsobem zakomponována do hesla zadaného uživatelem a tento výsledek je porovnán vůči hashi uložené v databázi.

---

[31] https://en.wikipedia.org/wiki/Rainbow_table

[32] https://en.wikipedia.org/wiki/Brute-force_attack

[33] https://cs.wikipedia.org/wiki/Slovn%C3%ADkov%C3%BD_%C3%BAtok

[34] https://cs.wikipedia.org/wiki/S%C5%AFl_(kryptografie)

[35] pomocí funkce mt_rand() -> http://php.net/manual/en/function.mt-rand.php

[36] http://php.net/manual/en/function.crypt.php

[37] https://en.wikipedia.org/wiki/SHA-1

```
$salt = sha1(mt_rand());
$this->userService->insert(array(
"password" => sha1(crypt($values->password, $salt)),
"salt" => $salt));
```

5.3.3 Dvoustranné omezení přístupu uživatelů

Tabulka 3.0.1 na straně 10 nastiňuje uživatelská práva a možnosti pro různé skupiny přístupů. Omezení těchto práv je v první řadě implementováno skrytím příslušných odkazů uživatelům, kteří nemají právo danou akci vykonat.

U webových aplikací jako je tato bakalářská práce však nestačí pouze uživatelům odkazy na dané funkčnosti (smazání týmu, přidání hráče atd.) nezobrazit, ale je velmi důležité mít ochranu dvojí. V případě, že by uživatel měl přístup k obdobné funkčnosti jinde v rámci systému, mohl by zjistit, jak odkaz na takovou funkčnost sestavit, a malou úpravou pak spouštět případy užití, ke kterým původně neměl být oprávněn. Proto je nezbytné implementovat ochranu i v rámci funkčností na straně serveru. Tím pádem pokud uživatel odkaz vytvoří a pokusí se případ spustit, na serverové straně se úloha nevykoná a server o této skutečnosti uvědomí uživatele chybovým hlášením.

Příklad – odebrání hráče z týmu na straně serveru:

```
public function actionDeletePlayer($teamId, $playerId)
{
     if ($this->isCoach($teamId, $this->user->id)) {
$this->teamService->deletePlayerInTeam($playerId,   $teamId);
     $this->redirect('Team:edit', $teamId);
     }
     else {
$this->error("Nemáte povolení mazat hráče tohoto týmu.",403);
     }
}
```

6. Programátorská dokumentace

V této kapitole je vysvětlen způsob instalace aplikace na webový server a popsána možnost přidání nových sportů včetně ukázkového příkladu.

6.1 Instalace

Kroky instalace jsou následující:

1) V závislosti na operačním systému je potřeba naistalovat odpovídající balíček pro zprovoznění Apache, PHP a MySQL serveru. Na CD přiloženém k této práci jsou dostupné odkazy na balíky *LAMP, MAMP a WAMP* pro odpovídající operační systémy Linux, Mac OS X a Windows.
2) Upravení souboru *httpd.conf* v adresáři nainstalovaného Apache serveru tak, aby řádek *#LoadModule rewrite_module modules/mod_rewrite.so* nebyl zakomentovaný (umístění souboru se liší v závislosti na operačním systému a instalačním balíku serveru)
3) Vytvoření aplikačního schématu v MySQL
    a. Vytvoření nové databáze s porovnáváním *utf8_czech_ci*
    b. Nahrání aplikačního schématu pomocí *backup.sql* z obsahu CD do nově vytvořené databáze
    c. (dobrovolné) Vytvoření nového uživatele, který bude toto schéma obsluhovat
4) Dále je nutné upravit soubor *sportis/app/config/config.local.neon*, kde uživatel vyplní IP adresu a port, na kterých je databáze dostupná. Dále vyplní název databáze a přihlašovací údaje uživatele, který databázi obsluhuje.

    dsn: 'mysql:host=<ip>:<port>;dbname=<jméno databáze>'
    user: <přihlašovací jméno uživatele, který má do databáze přístup>
    password: <heslo uživatele >
5) (dobrovolné) Pokud je aplikace jediná na nainstalovaém serveru, lze v souboru *httpd.conf* změnit řádek s *DocumentRoot* na následující:

    DocumentRoot "<cesta k aplikaci>/sportis/www"
6) Pokud všechny kroky instalace proběhly v pořádku, po přístupu na adresu *localhost/sign/in*, popřípadě *127.0.0.1/sign/in* se zobrazí přihlašovací obrazovka aplikace

6.2 Přidání nového sportu

Aplikace je koncipována pro sporty, ve kterých se utkání účastní dva týmy a o vítězi rozhoduje skóre. V rámci těchto kritérií lze do aplikace přidávat další sporty a postupem níže docílit využití vlastních šablon pro zobrazení událostí v daném sportu, či změny šablony pro zápis jejich výsledků.

Aplikace v základu obsahuje standardní šablony pro vykreslování událostí a evidenci výsledků. Tím pádem lze (pokud administrátor nechce využít možností využití vlastních šablon) zaevidovat nový sport pouze přidáním záznamu do tabulky *sport* v databázi se jménem daného sportu. Pokud se však administrátor rozhodne přidat vlastní způsob zobrazování událostí nového sportu a vlastní způsob jejich zápisu, je potřeba provést následující:

1) V databázi přidat jméno sportu do tabulky *sport*
2) Vytvořit nové *.latte* šablony pro zobrazení události a pro zápis výsledku pro daný sport (předlohou mohou být standardní *default.latte*, *checkIn.latte* a *layout.phtml* v *templates/Event*), nejlépe v nové podsložce vůči *templates/Event*, například *templates/Event/Football*
3) Upravit atributy třídy *EventPresenter* aby odráželi tuto skutečnost (relativní cesty se uvádějí bez přípony *.latte*)
    a. Do *$checkInTemplates* přidat záznam ve formátu „Název sportu“ => „Relativní cesta z *templates/Event* k šabloně se zápisy výsledků“
    b. Do *$defaultTemplates* přidat záznam ve formátu „Název sportu“ => „Relativní cesta z *templates/Event* k šabloně zobrazující události“
    c. Do *$sportToId* vložit záznam ve formátu „Id sportu v DB“ => „Název sportu“
4) Pokud je potřeba, tak v třídě *EventPresenter* implementovat vlastní beforeRender metodu pro daný sport a přidat její volání do switch tabulky v metodě *beforeRenderForSportWithId($sportId, $event)*.
5) Pokud je potřeba, tak v třídě *EventPresenter* implementovat vlastní veřejné metody pro vytvoření formulářů[38], které se mají použít v šablonách z kroku 2, a jejich callbacky. Metody renderDefault a renderCheckIn už jsou nastaveny, aby vykreslovaly nastavené šablony z kroku 3.

[38] https://doc.nette.org/en/2.3/forms

6.2.1 Příklad – přidání fotbalu

V následující části je postup ze sekce *6.2 Přidání nového sportu* předveden na přidání sportu fotbal do aplikace. Postup je předveden na aplikaci, v níž zatím žádné jiné sporty tímto způsobem přidány nejsou. V základní verzi této aplikace je příklad implementování fotbalu zakomponován do zdrojového kódu.

1) Přidání sportu fotbal do databáze a vytvoření odpovídající soutěže:

```
INSERT INTO sport(name) VALUES (‘Fotbal’);
INSERT INTO league(name,league_id) VALUES (‘Extraliga’,1);
// 1 odpovídá id, se kterým se nám vytvoří záznam o
fotbalu
```

2) Vytvoření šablony *checkIn.latte*, *default.latte* v umístění *templates/Event/Football*. Je vhodné do tohoto umístění zkopírovat šablonu *layout.html*, v níž je potřeba upravit cestu k *@layout.latte,* dle aktuálního umístění souboru.
3) Dále je potřeba adekvátně upravit atributy třídy *EventPresenter* na následující:

```
$checkInTemplates = [ "Fotbal" => 'Football/checkIn'];
$defaultTemplates = [ "Fotbal" => 'Football/default'];
$sportToId = [ 1 => "Fotbal"]; // 1 je id sportu v DB
```

4) V tomto případě není potřeba využívat dodatečnou metodu *beforeRender*, ale pro účel příkladu lze vytvořit prázdnou metodu *footballBeforeRender($event)*, která je následně odkázána ve switch tabulce v metodě *beforeRenderForSportWithId($sportId, $event)*:

```
switch ($sportName) {
    case "Fotbal":
        $this->footballBeforeRender($event);
        break;
    default:
        break;
}
```

5) Následuje implementace metody *createComponentFootballCheckInForm()* využité na tvorbu formuláře použitého v šabloně *checkIn.latte*. Formulář je potřeba spojit s novou metodou *footballCheckInFormSuccess($form, $values)*. V obou těchto metodách lze k získávání a ukládání informací o událostech využívat mimo jiné sloupec *json_details* z tabulky *event*, díky

kterému lze docílit ukládání většího množství informací o události, například stav utkání po prvním poločase.

6) Po vytvoření nové události pod soutěží Extraliga je nyní pro standardní zobrazení události využita nově vytvořená šablona *default.latte.* Při zápisu výsledku je využita šablona *checkIn.latte*.

6.3 Parametrizace soutěží

V rámci aplikace lze přidávat jak nové sporty, tak sportovní soutěže. Vzhledem k možnosti existence rozdílů mezi soutěžemi v rámci jednoho sportu, například v délce herní doby různých věkových skupin, je žádoucí implementovat možnost parametrizování jednotlivých soutěží. V tomto případě je opět využit formát JSON a sloupec *json_details* databázové tabulky *league*, ve kterém mohou být parametry jednotlivých soutěží uloženy a následně zohledňovány napříč aplikací.

6.3.1 Příklad – parametrizace extraligy

Po vytvoření soutěže v kroku 1 ze sekce *6.2.1 Příklad – přidání fotbalu na straně* 34 lze do databáze vložit specifikaci soutěže extraligy ve formátu JSON následovně:

```
UPDATE league SET json_details='{
     "periods": 2,
     "periodLength": 45,
     "penalties": ["Žlutá karta", "Červená karta"],
     "players": 11,
     "points": { "win": 2, "draw" : 1, "loss" : 0 }
}' where name='Extraliga';
```

Následně lze ve třídě *LeaguePresenter* při generování pořadí týmů v soutěžích na dané nastavení reagovat určením zisku bodů za vítězství, remízu a prohru:

```
$json = $this->leagueRepository->
     getJsonDescriptionByLeagueId($event->league_id);
$winPoints = json['points']['win'];
$drawPoints = json['points']['draw'];
$lossPoints = json['points']['loss'];
```

Aplikace takto může být snadno rozšířena o přizpůsobení různým parametrům soutěží. V základní verzi lze přidáním JSON řetězce parametrizovat bodové ohodnocení za vyhrané utkání, remízu a prohru. Zohlednění těchto parametrů se následně projeví na pořadí týmů v tabulkách jednotlivých soutěží.

6.3 Cron úlohy

Aplikace také umožňuje spouštění vlastních cron úloh. Toho je v základní verzi využito pro odesílání upozornění na události, kterých se uživatel účastní, a konají se v následujících 24 hodinách.

Pro implementaci těchto cron úloh je vytvořena třída *CliPresenter*, která je pomocí routování[39] využita při každém interpretování souboru *index.php* z konzole. Pro zavedení vlastní cron úlohy, kterou má aplikace vykonat, stačí splnit následující:

1) Ve třídě *CliPresenter* implementovat private metodu, která má být pomocí cron úlohy spuštěna.
2) Přidat volání metody v rámci *actionCron* do sekce *if (...isConsole())*, eventuálně po zjištění předaných argumentů z konzole pomocí *getopt*[40]
3) Nastavení automatického spouštění skriptu s danými parametry na serveru

6.3.1 Příklad – nadcházející události

Výše popsané odeslání upozornění na události, které se konají v následujících 24 hodinách lze tedy naimplementovat takto:

1) Implementování private metody *notifyOfUpcomingEvents* ve třídě *CliPresenter*.
2) Volání této metody v rámci *actionCron* v *if* sekci v závislosti na parametru „notifyOfUpcoming“

```
$options = getopt("", ["notifyOfUpcoming"]);
if (Nette\Environment::isConsole()) {
   if(isset($options["notifyOfUpcoming"]))
      $this->notifyOfUpcomingEvents();
   $this->terminate();
} ...
```

3) Vytvoření cron skriptu a nastavení spouštění v určitý čas (závislé na konkrétním hostingu):

```
php sportis/www/index.php --notifyOfUpcoming
```

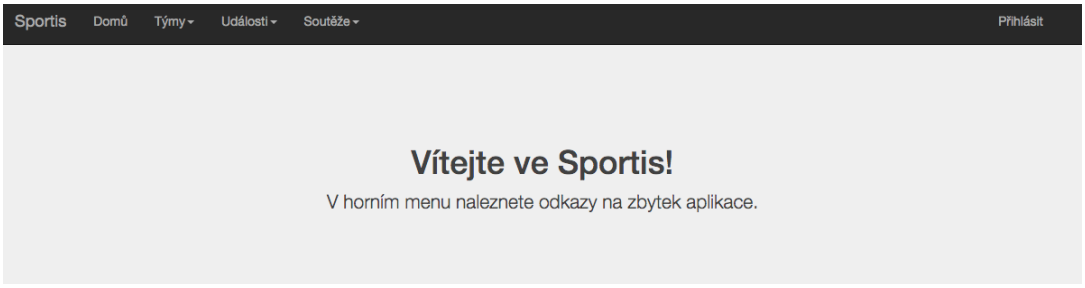
[39] https://pla.nette.org/en/routovani-v-cli

[40] http://php.net/manual/en/function.getopt.php

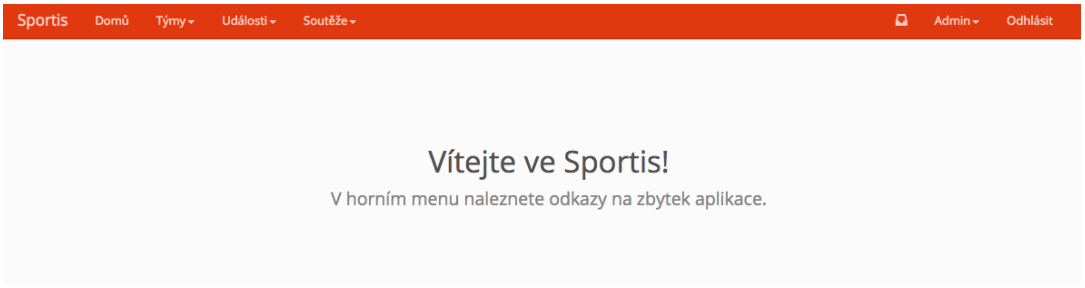
7. Uživatelská dokumentace

V této kapitole je nastíněno použití základních funkčností aplikace, jako jsou: menu, registrace, registrace týmu, úprava nastavení, změna vzhledu stránek týmu, tvorba události, systém zpráv a notifikací.

7.1 Ovládací panel



Obrázek 7.0.1 Úvodní obrazovka nepřihlášeného uživatele

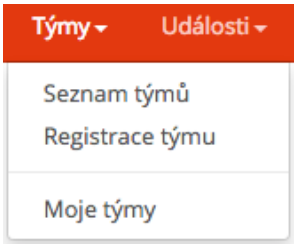


Obrázek 7.0.2 Úvodní obrazovka přihlášeného uživatele

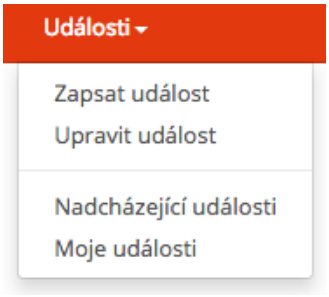
Úvodní obrazovka aplikace, kterou může vidět každý uživatel. Pomocí horního menu se uživatelé můžou v aplikaci orientovat:

1. Domů → Vrátí uživatele odkudkoliv na tuto stránku

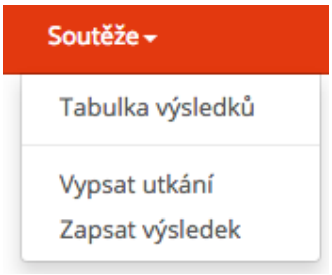
2. Týmy → Umožňuje výpis seznamu týmů v aplikaci, výpis uživatelových týmu, popřípadě registraci týmu, pokud je uživatel v roli trenéra



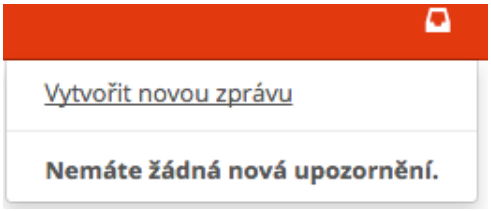
3. Události → Umožňuje uživatelům zobrazit jejich události, nadcházející události, popřípadě vytvoření události, či její úpravu.



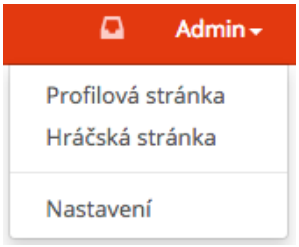
4. Soutěže → Umožňuje běžným uživatelům zobrazení aktuální tabulky pořadí týmů v soutěžích. Rozhodčím umožňuje vypsat nové utkání v soutěži, popřípadě zapsat výsledek.



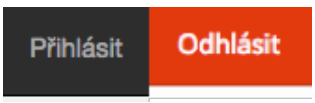
5. Box →Obsahuje uživatelovy zprávy a notifikace (ať už od hráčů, či od uživatelových týmů) a umožňuje zasílání zpráv jiným uživatelům



6. Tlačítko s názvem profilu →Zpřístupňuje nastavení profilové a hráčské stránky a nastavení přihlášeného uživatele

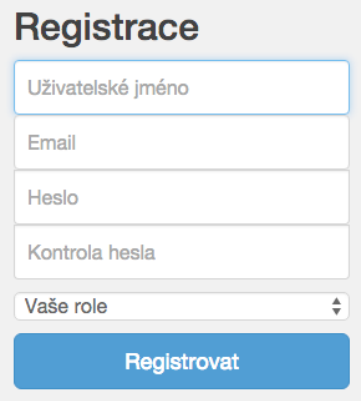


7. Přihlásit / odhlásit →Přihlásí/odhlásí uživatele



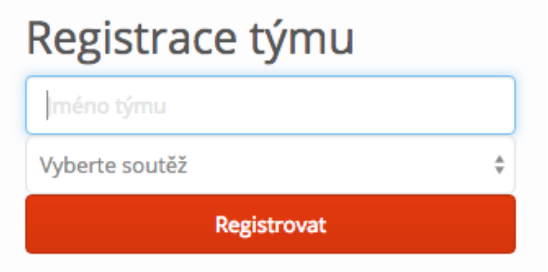
7.2 Registrace uživatele

V případě, že uživatel není do systému zaregistrovaný, klikne na tlačítko “Přihlásit” v pravém horním rohu a následně na “Zaregistrovat”. Zobrazí se mu klasický registrační formulář, po jehož vyplnění bude zaregistrován do systému, pokud dané uživatelské jméno nebo e-mail nejsou již zaregistrované.



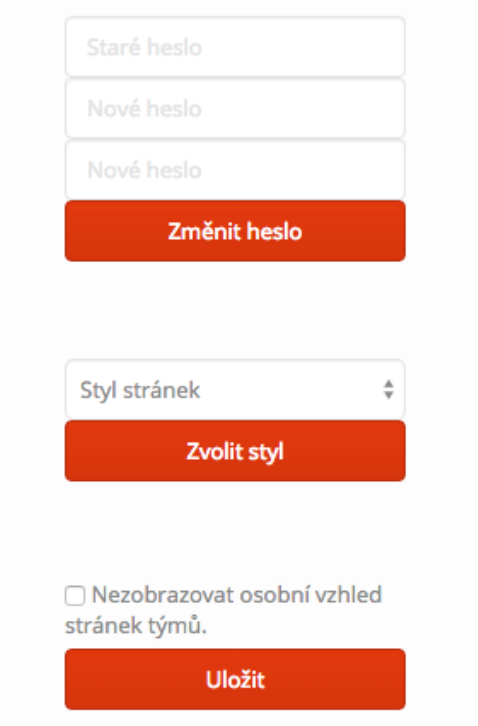
7.3 Registrace týmu

V případě, že je uživatel registrován v roli trenéra a chce založit nový tým, klikne na tlačítko „Týmy“ v navigační liště a vybere „Registrace týmu“. Následně je přesměrován na obrazovku, kde zadá název týmu a vybere soutěž, do které má být tým zaregistrován. Po odeslání registrace jsou notifikování rozhodčí příslušné soutěže a žádost mohou akceptovat nebo zamítnout. Uživatel je v případě akceptace o skutečnosti informován automatickou zprávou.



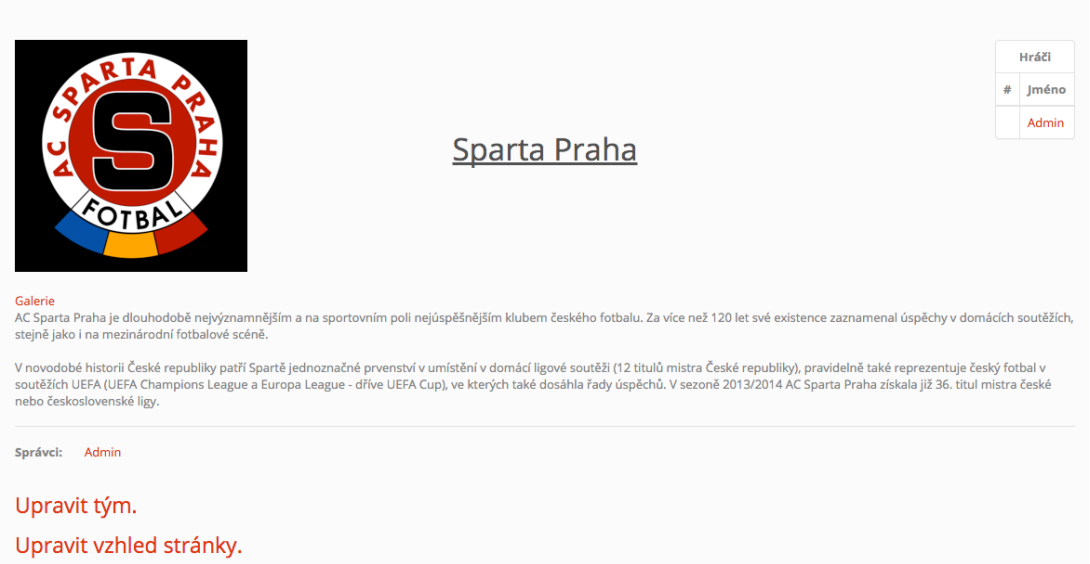
7.4 Uživatelské nastavení

Přes tlačítko s názvem uživatelova profilu a volbu „Nastavení“ se uživatel dostane na své nastavení v rámci aplikace. Zde může změnit své heslo, vybrat si jiný styl zobrazení stránek, popřípadě může potlačit zobrazování osobního vzhledu stránek týmů.



7.5 Stránky týmu

Na stránkách týmu mohou uživatelé vidět informace o týmu, hráče, logo, a pokud mají dostatečná oprávnění (jsou jedním z trenérů), tak také odkaz na úpravu týmu a jeho vzhledu.



“Upravit tým” odkáže uživatele na stránku, kde může vyplnit nové informace o týmu, přidávat/odebírat hráče a správce, popřípadě změnit logo týmu.

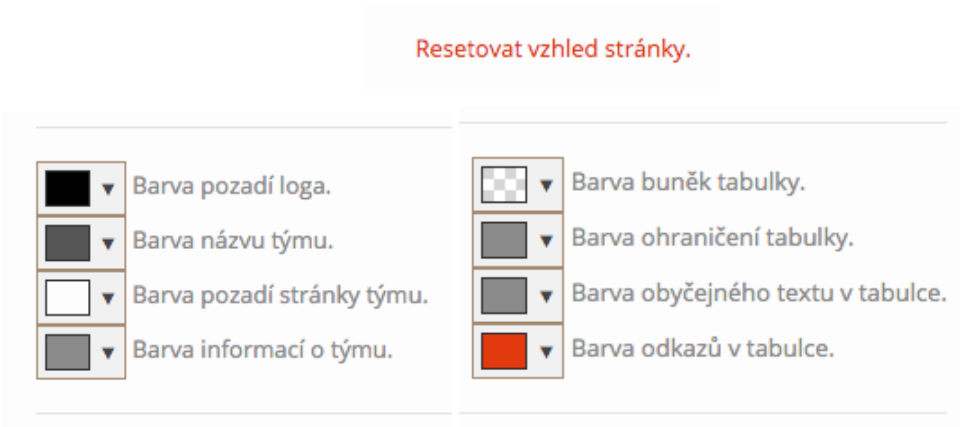
“Upravit vzhled stránky” umožní uživateli změnit zobrazení stránek napříč systémem. Toto nastavení však mohou uživatelé na svém profilu potlačit.

7.6 Upravení vzhledu týmu

Pokud uživatel zvolí „Upravit vzhled stránky“, je mu nabídnuta tabulka, ve které si může zvolit barvy jednotlivých prvků stránky týmu.

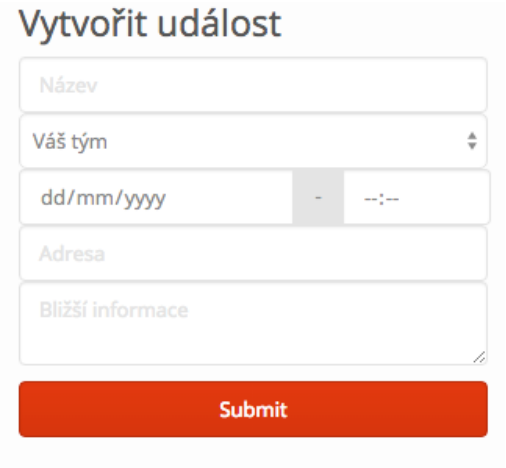
Po změně prvků a navrácení na stránku týmu jsou barvy uloženy do systému, a vzhled stránek týmu je tak zachován.

V případě, že by se uživatel chtěl vrátit k původnímu systémovému vzhledu, může tomu tak učinit stisknutí odkazu „Resetovat vzhled stránky“ v horní pravé části.



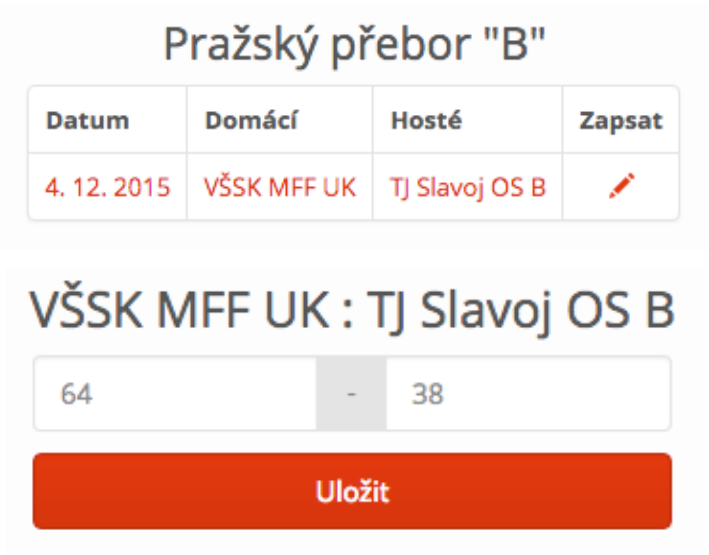
7.7 Tvorba soukromé události a utkání

Uživatel s dostatečnými právy má možnost pro svůj tým vytvořit novou událost, na níž bude každý hráč týmu upozorněn. Může tomu tak udělat přes tlačítko „Události“ v menu, a následně „Zapsat událost“. Následně událost pojmenuje a zvolí jeden ze svých týmu, kterému je událost určena. Dále vyplní datum události a bližší informace. Analogicky lze vypsat nové utkání. Tato možnost je však zpřístupněna pouze rozhodčím.



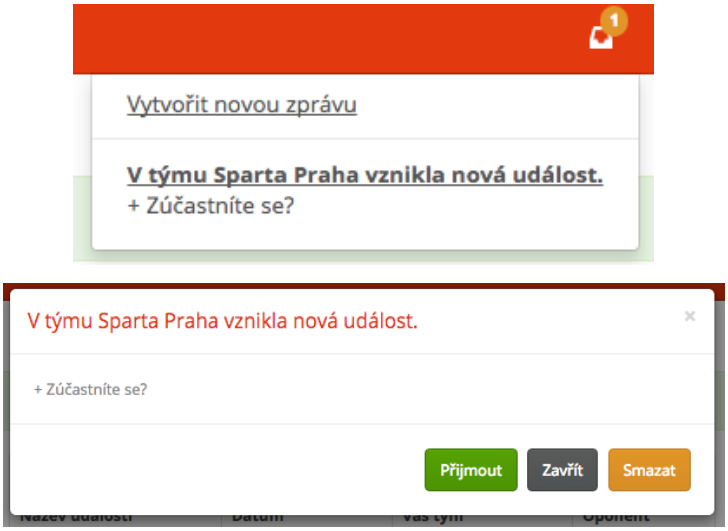
7.8 Zapsání výsledku utkání

Uživatel v roli rozhodčího má přes volby „Soutěže“ → „Zapsat výsledek“ možnost zaevidovat výsledek jednotlivých utkání. Učiní tak může stisknutím tlačítka ve formě tužky u příslušné události. Následně se uživateli zobrazí formulář, po jehož vyplnění a úspěšném odeslání bude výsledek utkání zapsán do systému.



7.9 Systém notifikací

Pokud uživateli někdo zašle zprávu, nebo v některém z týmů, ve kterých je uživatel aktivní, vznikne nová událost, uživatel je na tuto skutečnost upozorněn svítící ikonou nad tlačítkem zpráv. Pozvání na událost může uživatel přijmout, nebo smazat. V obou případech může svoji účast později změnit na stránce události.



Závěr

V rámci této práce byl vytvořen informační systém, který umožňuje klubům především nižších sportovních soutěží jednoduchou správu svých webových stránek. Registrace a používání systému jsou pro uživatele jednoduché a intuitivní.

Díky snadné konfigurovatelnosti si týmy mohou své domácí stránky vzhledově přizpůsobit svým individuálním potřebám. V případě zájmu může systém spravovat nejen jeden tým, ale i všechny týmy dané soutěže centrálně na jediném místě. Dají se v něm evidovat jednotlivé události týmů, ať už se jedná o zápasy nebo události jiného charakteru, například tréninky.

Každý uživatel systému v něm má svoji vlastní profilovou stránku. Hráči pomocí interního systému zpráv dostávají upozornění na nové události a mohou na nich evidovat svoji účast. Každý tým má přístup k vlastní nástěnce a fotogalerii.

Způsob navržení systému a implementace jednotlivých prvků umožňuje budoucím vývojářům jednoduchou implementaci dalších rozšíření. Jedním z takových rozšíření by mohl být například modul povolující kompletní správu sportovních soutěží přímo z webového rozhraní aplikace. V současné verzi jsou soutěže parametrizovatelné, avšak o jejich zaevidování se musí postarat sám administrátor. Dalším modulem by mohla být evidence plateb hráčů, popřípadě kasový systém určený pro celou soutěž.

Systém je také vzhledem k rozdílnosti rolí uživatelů navržen s velkým důrazem na bezpečnost. Nejenže se odkazy, na které uživatel nemá mít oprávnění, nezobrazují, ale ani pokud uživatel získá daný odkaz jiným způsobem, jeho spuštění nevykoná požadovanou úlohu díky všudypřítomné back-end re-verifikaci.

Při ohlédnutí zpět by určité aspekty této práce mohly být naimplementovány jiným způsobem. Například každá sportovní soutěž by mohla mít přidělenou vlastní subdoménu v rámci systému, a tak povolit uživateli pohybovat se pouze v prostředí, ve kterém je jeho tým registrovaný. Dále by systém zpráv mezi uživateli mohl mít propracovanější vzhled a zpřístupňovat real-time konverzaci podobnou sociálním sítím. V neposlední řadě by bylo rovněž vhodné více zdokonalit základní vzhled stránek.

I přes některé nedostatky je však aplikace vhodným kandidátem na nahrazení aktuálně používaných systémů evidence sportovních týmů. A to nejen těch, které dosud evidují své informace pouze pomocí statických HTML stránek.

Seznam použité literatury

[1] https://cs.wikipedia.org/wiki/Framework

[2] https://doc.nette.org/cs/2.3/forms

[3] **Robin Nixon: Learning PHP, MySQL, JavaScript, and CSS: A Step-by-Step Guide to Creating Dynamic Websites**, O'Reilly Media, ISBN 978-1449319267

[4] **Rasmus Lerdorf, Kevin Tatroe**: Programming PHP, O'Reilly Media, ISBN 978-1449392772

[5] **Nette Framework Dokumentace [20.8.2015]**: http://doc.nette.org/cs/

[6] **PHP Documentation [20.8.2015]:** http://php.net/docs.php

Seznam tabulek

Seznam obrázků

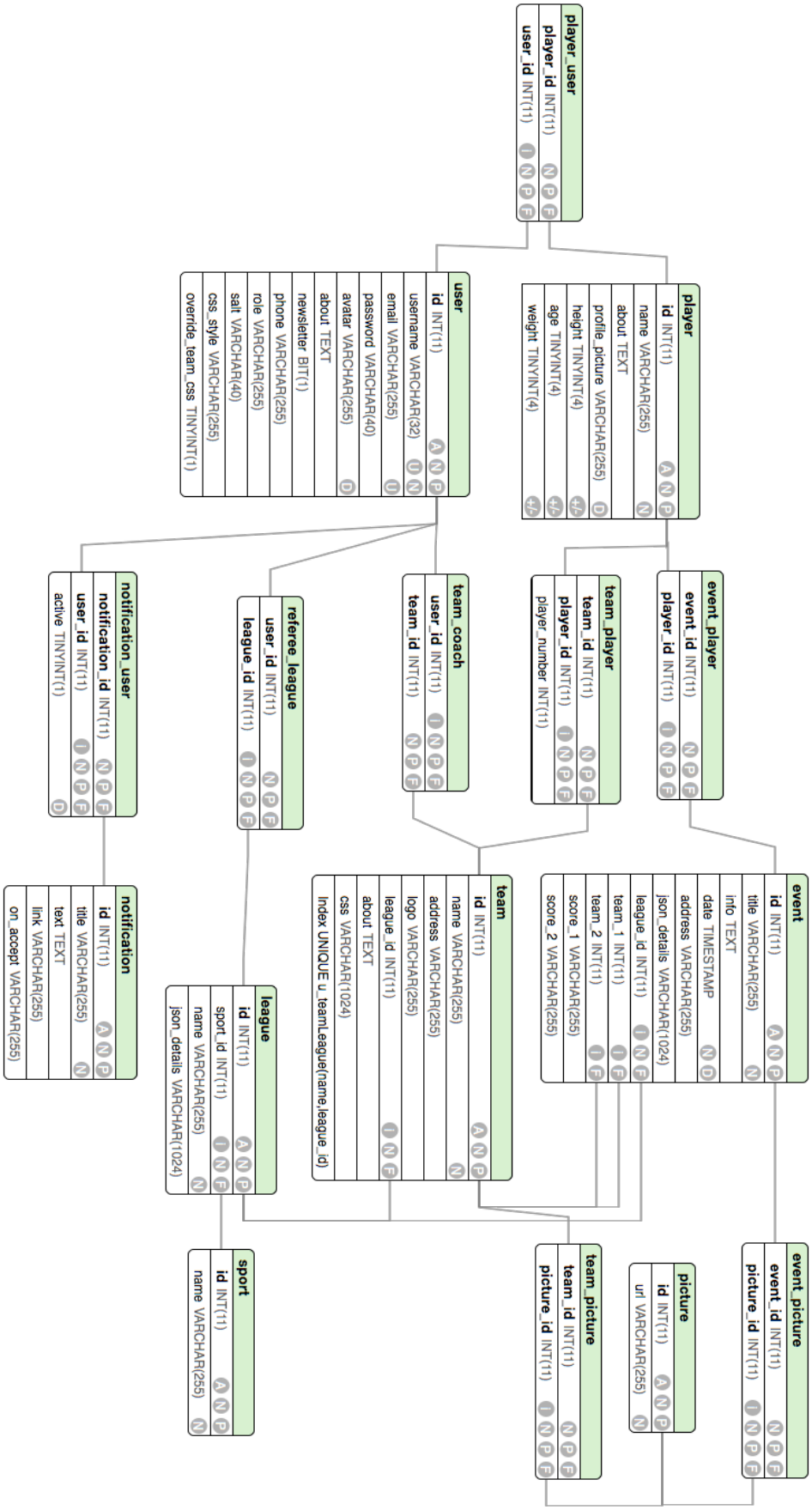
Přílohy

1. CD

Na přiložené CD se nacházejí následující soubory a složky:

- *index.html* – popisuje obsah CD a zadání bakalářské práce
- *Docs* – adresář s vygenerovanou dokumentací aplikace, zdrojový kód vyvinutý v rámci této práce je obsažen v namespace App
- *Packages.txt* – obsahuje odkazy na instalační balíčky pro servery s různými operačními systémy
- *text.pdf* – text této práce
- *backup.sql* – SQL skript pro tvorbu databázového schématu
- *testData.sql* – SQL skript pro naplnění aplikace testovacími daty
- *sportis.zip* – archiv se zdrojovými kódy aplikace

2. Schéma databáze



3. Popis databázových tabulek

Event – uchovává data o všech zaznamenaných událostech v systému, jako jsou utkání, tréninky, popřípadě jiné události, zaevidované uživateli.

Název sloupce	Typ	Popis
id	int(11)	- unikátní id události
title	varchar(255)	- název události
info	mediumtext	- informace o události
date	date	- datum konání události
address	varchar(255)	- adresa místa události
json_details	varchar(1024)	- JSON řetězec umožňující zachycení detailů události
league_id	int(11)	- cizí klíč, který odkazuje na id soutěže události
team_1	int(11)	- cizí klíč, který odkazuje na id prvního týmu
team_2	int(11)	- cizí klíč, který odkazuje na id druhého týmu
score_1	varchar(255)	- výsledné skóre týmu 1
score_2	varchar(255)	- výsledné skóre týmu 2

Tabulka 0.1 – Databázová tabulka event.

Event_picture – tabulka, která spojuje události s jejich obrázky

Název sloupce	Typ	Popis
event_id	int(11)	- cizí klíč odkazující na id události
picture_id	int(11)	- cizí klíč odkazující na id obrázku

Tabulka 0.2 – Databázová tabulka event_picture.

Event_player – tabulka, která spojuje události s hráči, kteří se jí účastnili

Název sloupce	Typ	Popis
event_id	int(11)	- cizí klíč odkazující na id události
player_id	int(11)	- cizí klíč odkazující na id hráče

Tabulka 0.2 – Databázová tabulka event_player.

League – tabulka, která popisuje jednotlivé soutěže

Název sloupce	Typ	Popis
id	int(11)	- unikátní id soutěže
sport_id	int(11)	- cizí klíč odkazující na id sportu
name	varchar(255)	- název soutěže
json_details	varcharo(1024)	- JSON řetězec umožňující parametrizaci soutěže

Tabulka 0.3 – Databázová tabulka league.

Notification – tabulka znázorňující uživatelská oznámení

Název sloupce	Typ	Popis
id	int(11)	- unikátní id oznámení
title	varchar(255)	- název oznámení („Byl jste přidán do týmu!“ atd.)
text	text	- zpráva obsažená v oznámení
link	varchar(255)	- odkaz, který může zpráva obsahovat (je vytvářen pouze uvnitř aplikace, ne uživateli)
on_accept	varchar(255)	- odkaz, který je vykonán při označení zprávy za odsouhlasenou

Tabulka 0.4 – Databázová tabulka notification.

Notification_user – tabulka, která přiřazuje oznámení uživatelům

Název sloupce	Typ	Popis
notification_id	int(11)	- cizí klíč odkazující na id oznámení
user_id	int(11)	- cizí klíč odkazující na id uživatele
active	boolean	- příznak zda uživatel oznámení již zobrazil, či nikoliv

Tabulka 0.5 – Databázová tabulka notification_user.

Picture – tabulka obsahující cestu k obrázkům

Název sloupce	Typ	Popis
id	int(11)	- unikátní id obrázku
url	varchar(255)	- systémová cesta k souboru s obrázkem

Tabulka 0.6 – Databázová tabulka picture

.

Player – tabulka obsahující údaje o jednotlivých hráčích

Název sloupce	Typ	Popis
id	int(11)	- unikátní klíč hráče
name	varchar(255)	- jméno hráče
about	text	- info o hráči, které si sám vyplní
profile_picture	varchar(255)	- systémová cesta k profilovému obrázku hráče
height	tinyint(4)	- výška hráče
age	tinyint(4)	- věk hráče
weight	tinyint(4)	- váha hráče

Tabulka 0.7 – Databázová tabulka player

.

Player_user – tabulka, která spojuje hráčské profily s uživateli

Název sloupce	Typ	Popis
player_id	int(11)	- cizí klíč odkazující na id hráčského profilu
user_id	int(11)	- cizí klíč odkazující na id uživatele

Tabulka 0.8 – Databázová tabulka player_user

.

Referee_league – tabulka spojující rozhodčí s danými soutěžemi

Název sloupce	Typ	Popis
user_id	int(11)	- cizí klíč odkazující na id uživatele
league_id	int(11)	- cizí klíč odkazující na id soutěže, ve které je uživatel rozhodčím

Tabulka 0.9 – Databázová tabulka referee_league.

Sport – tabulka popisující jednotlivé zaregistrované sporty

Název sloupce	Typ	Popis
id	int(11)	- unikátní id sportu
name	varchar(255)	- název sportu

Tabulka 0.10 – Databázová tabulka sport.

Team – tabulka popisující jednotlivé týmy

Název sloupce	Typ	Popis
id	int(11)	- unikátní id týmu
name	varchar(255)	- jméno týmu
address	varchar(255)	- adresa, kde tým sídlí
logo	varchar(255)	- systémová cesta k logu týmu
league_id	int(11)	- cizí klíč odkazující na soutěž, které se tým účastní
about	text	- informace o týmu
css	varchar(1024)	- CSS, podle kterého se styluje domovská stránka týmu (lze potlačit uživatelem)

Tabulka 0.11 – Databázová tabulka team.

Team_coach – tabulka, která přiřazuje trenéry k týmům

Název sloupce	Typ	Popis
team_id	int(11)	- cizí klíč odkazující na tým
user_id	int(+)	- cizí klíč odkazující na uživatele, který je v roli trenéra týmu

Tabulka 0.12 – Databázová tabulka team_coach.

Team_picture – tabulka spojující týmy s jejich obrázky

Název sloupce	Typ	Popis
team_id	int(11)	- cizí klíč odkazující na tým
picture_id	int(+)	- cizí klíč odkazující na obrázek, který patří k týmu

Tabulka 0.14 – Databázová tabulka team_picture.

Team_player – tabulka spojující týmy s hráči

Název sloupce	Typ	Popis
team_id	int(11)	- cizí klíč odkazující na tým
player_id	int(11)	- cizí klíč odkazující na profil hráče, který v týmu hraje
player_number	int(11)	- číslo dresu hráče v daném týmu

Tabulka 0.15 – Databázová tabulka team_player.

User – tabulka uchovávající data o uživatelích

Název sloupce	Typ	Popis
id	int(11)	- unikátní id uživatele
username	varchar(32)	- unikátní uživatelské jméno
email	varchar(255)	- unikátní email uživatele
password	varchar(40)	- heslo uživatele zašifrováno pomocí SHA1
avatar	varchar(255)	- systémová cesta k profilovému obrázku uživatele
about	text	- informace o uživatel
newsletter	boolean	- příznak, zda uživatel chce odebírat newsletter
phone	varchar(255)	- telefonní číslo uživatele
role	varchar(255)	- přístupové role uživatele, oddělené čárkou (administrator, coach, atd.)
salt	varchar(40)	- sůl, která byla použita při solení hesla
css_style	varchar(255)	- systémová cesta k css složce, kterou si uživatel nastavil
override_team_css	boolean	- příznak, zda se má uživateli zobrazovat vlastní nastavení vzhledu stránek týmů

Tabulka 0.16 – Databázová tabulka user.