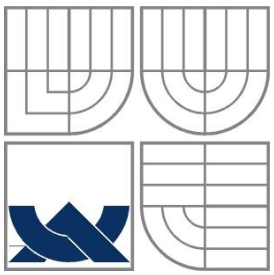
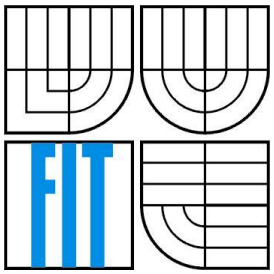


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PLAVECKÉHO ODDÍLU

INFORMATION SYSTEM OF A SWIM CLUB

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN VENCEL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADEK BURGET, Ph.D.

BRNO 2016

Zadání bakalářské práce/18225/2015/xvence00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Vencel Jan**
Obor: Informační technologie
Téma: **Informační systém plaveckého oddílu**
Information System of a Swim Club
Kategorie: Web

Pokyny:

1. Seznamte se s technologiemi pro tvorbu webových informačních systémů na platformě PHP.
2. Analyzujte požadavky plaveckého oddílu na informační systém pro evidenci plavců, jejich výsledků, nominací, přihlášek apod.
3. Na základě provedené analýzy navrhněte architekturu informačního systému.
4. Po dohodě s vedoucím implementujte navržený systém s využitím vhodných klientských a serverových technologií.
5. Implementujte možnost importu dat ze serverů Českého svazu plaveckých sportů a FINA a exportu dat pro další analýzu.
6. Proveďte testování vytvořeného systému.
7. Zhodnoťte dosažené výsledky.

Literatura:

- Gutmans, A., Rethans, D., Bakken, S.: Mistrovství v PHP 5, Computer Press, 2012
- Dokumentace k projektu Nette: http://doc.nette.org/cs/2.2/

Pro udělení zápočtu za první semestr je požadováno:

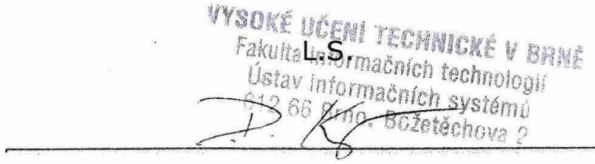
- Body 1 až 3

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Burget Radek, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016



L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato bakalářská práce se věnuje informačnímu systému plaveckého oddílu ASK Blansko. Jedná se o webovou aplikaci postavenou na trojici webových technologií – webserver Apache, databáze MySQL a jazyk PHP. Účelem systému je umožnit zástupcům oddílu administraci těchto databázových dat a v přístupné podobě je nabídnou také běžnému uživateli. Při vývoji systému byl použit PHP rámec Nette a rámec Bootstrap pro uživatelské rozhraní. K vizualizaci dat uživateli bylo užito nástroje Google Charts. Statistiky návštěvnosti byly získány díky službě Google Analytics. Práce se zabývá analýzou, návrhem, implementací a testováním konkrétního informačního systému. Za zmínku stojí také funkcionalita související s exportem vybraných dat a importem dat z ČSPS a FINA.

Abstract

This bechelor thesis is devoted to an information system of ASK Blansko swim club. It is a web application built on the web technology trio – webserver Apache, database MySQL and PHP language. The main goal of the system is to provide a possibility to administrate these database data and also to make the data accessible to a common user. During the development, the Nette PHP framework and the Bootstrap front-end framework were used. The database data were vizualized using the Google Charts tool and statistical data about the user‘s attendance were collected by the Google Analytics tool. The thesis discusses analysis, design, implementation and testing of the information system. The funcionality related to an export of the selected data and also an import from ČSPS and FINA servers is also worth mentioning.

Klíčová slova

Informační systém, databáze, plavecký oddíl, PHP, Nette Framework, MySQL, Bootstrap, HTML, CSS, JavaScript, JQuery, Google Charts, Google Analytics, AJAX, Git, ERD, UCD, MVC, responzivní design.

Keywords

Information systém, database, swim club, PHP, Nette Framework, MySQL, Bootstrap, HTML, CSS, JavaScript, JQuery, Google Charts, Google Analytics, AJAX, Git, ERD, UCD, MVC, responzive design.

Citace

VENCEL, Jan. *Informační systém plaveckého oddílu*. Brno, 2016. 51 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Informační systém plaveckého oddílu

Prohlášení

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

……
Jan Vencel
24. února 2016

Poděkování

V této sekci bych chtěl poděkovat vedoucímu práce Ing. Radkovi Burgetovi, Ph.D za poskytnutí odborné pomoci.

Obsah

1 Úvod

Každým rokem se rozrůstající databáze plaveckého oddílu ASK Blansko způsobila zrození myšlenek na vytvoření informačního systému, který by sdružoval a přehledným způsobem prezentoval tato, za dlouhá léta nashromážděná, data. Další motivací pro vznik programu bylo poskytnout oddílu jednoduchý způsob vedení evidence a analýzy výkonů plavců.

Výsledkem těchto úvah byl požadavek na vytvoření webové aplikace, která by eliminovala nutnost používat systém na specifickém zařízení. Ze stejného důvodu byl také kladen důraz na použití technologií využívaných u většiny moderních prohlížečů.

Uváděný program je první svého druhu, co se týče internetových aplikací na správu oddílové databáze plavců v republice. Podobnou funkcionalitu má pouze informační systém Českého svazu plaveckých sportů, který schraňuje data týkající se registrovaných plavců a závodů na území České republiky. Výsledky ze závodů v textové podobě uložené na serverech ČSPS jsou využívány touto aplikací k importu zaplavaných časů plavci z místního oddílu.

Jelikož je aplikace volně přístupná na internetu mohou ji užívat nejen trenéři, ale hlavně také plavci, kterých se vlastně samotná data v systému týkají. Tito plavci a jejich rodiče mají tak možnost nahlížet do objemné oddílové databáze, vybírat a třídit si data podle svého zájmu a následně si tato data vyexportovat pro další analýzu v programu MS Excel nebo pouze k tisku.

Obsah práce je rozdělen do několika kapitol, které popisují tvorbu aplikace od jejího návrhu přes implementaci až k jejímu otestování a následnému nasazení. Na první kapitolu, kterou představuje tento úvod, navazuje druhá kapitola, která má uvést do problematiky představením současného systému administrace plaveckého oddílu a v návaznosti na to uvedením cílů aplikace. Dále je objasněna konkrétní metodologie návrhu aplikace. Následující kapitola se zabývá popisem využitých technologií při tvorbě a implementaci informačního systému. Ve třetí kapitole jsou rozepsány postupy při analýze a návrhu systému, které jsou doplněny diagramy. Další kapitola se už věnuje typu architektury systému a zvláště popisu implementace programu a to zejména jeho stěžejních funkcí. Pozornost je zde věnována také vytvoření uživatelsky přívětivého uživatelského rozhraní nebo způsobu přihlašování uživatelů. Šestá kapitola je zaměřena na testování funkcionality aplikace při zkušebním provozu a její následné nasazení do ostrého provozu. Obsažena je také kapitola shrnující statistická data nashromážděná při sledování návštěvnosti aplikace. Poslední dvě kapitoly potom tvoří závěr doplněný možnými vylepšeními aplikace a seznam použité literatury.

2 Úvod do problematiky

Začneme objasněním současných způsobů administrace oddílových dat s drobným nahlédnutím do minulosti. V návaznosti na rozbor současného stavu jsou vytyčeny cíle aplikace.

2.1 Současný stav

Oddíl závodního plavání v Blansku začal psát svoji historii ve válečném roce 1940. Celou dobu si trenéři museli nějakým způsobem vést záznamy. Až do nedávné minulosti byly veškeré údaje o plavcích a jejich výsledcích uchovávány pomocí papíru a tužky. Takto vzniklá kartotéka byla velmi závislá na pečlivosti trenérů, obtížně se udržovala v aktuálním stavu, těžce se v ní vyhledávaly požadované informace. Tzv. sdílení dat bylo nemožné nebo minimálně hrozilo ztrátou či pomícháním jednotlivých listů.

Situace se začala měnit k lepšímu po rozšíření výpočetní techniky do běžného života. V roce 2004 zprovoznil Český svaz plaveckých sportu na svých internetových stránkách aplikaci pro centrální evidenci závodních plavců a závodů konaných na území České republiky. Ze dne na den tak trenéři ale i plavci získali komfortní přístup k datům s možností jednoduché tvorby přihlášek.

Přes popsaný pokrok ale i nadále nebylo možné u konkrétního plaveckého oddílu podrobněji analyzovat výsledky jednotlivých plavců a vytvářet souhrnné statistiky a žebříčky. Tohoto úkolu se v Blansku od roku 2006 ujal desktopový program pracující nad lokální databází MS Access. Od svého vzniku až do dnešních dnů nastřádal přes čtrnáct tisíc záznamů. Velmi usnadňuje hlavnímu trenérovi přehled o plavcích, archivaci výsledků a tvorbu přihlášek včetně všech analytických rozborů. Jeho hlavní nevýhodou je jeho omezená působnost. Veřejnost nemá k datům v databázi přístup do té chvíle, dokud hlavní trenér nevytvoří konkrétní sestavu a nezveřejní jí na internetových stránkách nebo na oddílové nástěnce na bazéně.

2.2 Cíle

Cílem je vytvořit aplikaci, která umožní hlavnímu trenérovi administraci oddílové databáze odkudkoli. Aplikace už nebude sloužit pouze hlavnímu trenérovi, ale také ostatním trenérům a předně plavcům. Záměrem je poskytnout veřejnosti nástroje k prohlížení a vyhledávání v databázi a umožnit sestavení vlastních výběrů a žebříčků podle určitých kritérií a zároveň tak snížit nároky na hlavního trenéra, který musí doposud po každé sezóně ručně vytvářet aktuální žebříčky.

Dalším důležitým cílem aplikace je vytvořit nový způsob evidence osobních informací o plavcích a podpořit nebo nahradit tak papírovou kartotéku. Informace by byly na kartě detailu plavců přístupny pro všechny trenéry, ale upravovat by je mohl pouze hlavní trenér.

Protože se jedná o veřejnou aplikaci, tak je záměrem nabídnout plavcům a jejich rodičům funkce, které se týkají samotných plavců. Řeč je o statistikách a grafech vytvořených na základě databázových dat. Zmíněné funkce by představovaly další lákadlo aplikace, protože každého nejvíce zajímají statistiky a grafy sestavené konkrétně na jeho osobu.

S ohledem na fakt, že hodně lidí v dnešní době přistupuje na internet přes mobilní zařízení, je také cílem aplikace schopnost přizpůsobit se těmto zařízením a poskytovat stejné funkce jako desktopovým uživatelům.

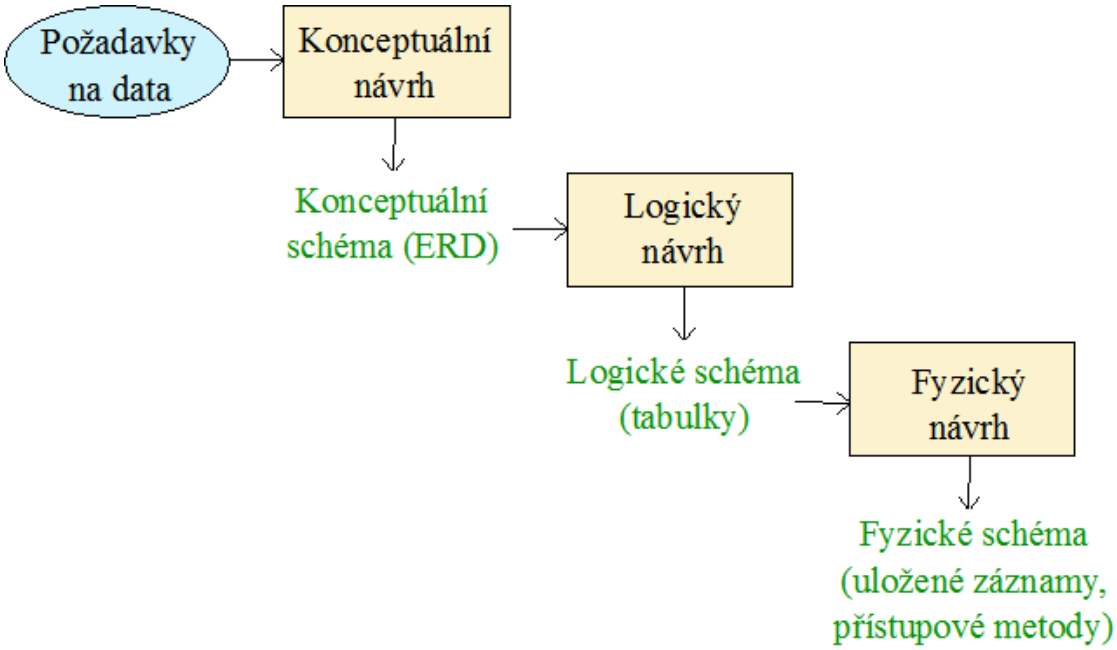
3 Zvolená metodologie návrhu

Aplikace byla vyvíjena v následujících etapách, které jsou součástí životního cyklu obecného informačního systému. Jedná se o posloupnost za sebou jdoucích období. Pro každé období byl stanoven určitý cíl a k jeho dosažení byly v tomto období nasměrovány veškeré činnosti [2].

- **Předběžná analýza a specifikace cílů** - Základem celkového návrhu a vývoje stávajícího systému jsou požadavky uživatelů a cíle organizace. V této části byly dané požadavky shromážděny, rozebrány a byla odhadnuta doba realizace s náklady.
- **Analýza systému a specifikace požadavků** - Tato část cyklu rozebírá předchozí část. Je velmi důležité odhalit veškeré chyby ve struktuře dat i systému v této fázi. Chyby, které se zde neodhalí, jsou později velice obtížně odstranitelné. Výsledkem byl popis návrhu a realizace IS, časový harmonogram a konkrétní implementace systému. Patří sem také logický datový model a fyzický datový model
- **Implementace** – Dále přišlo na řadu vlastní programování.
- **Testování** - V této etapě byly provedeny připravené testy na hotovém informačním systému. Bylo potřeba otestovat veškeré možné reakce systému na vstupní data a zjištěné nedostatky odstranit.
- **Zavádění systému** - Zaváděním systému se myslí především jeho instalace a zavedení do provozu organizace nebo školení uživatelů.
- **Rutinní provoz a údržba** – V této etapě je systém provozován, používán a udržován. Informační systém, o kterém je tato práce, se aktuálně nachází v tomto bodě.
- **Ukončení provozu** - do této fáze se informační systém dostává, pokud již nesplňuje požadavky uživatelů. Může probíhat současně se zaváděním nového informačního systému.

3.1 Konceptuální modelování

Návrh databáze zahrnoval tři hlavní fáze (obrázek 3.1). V této podkapitole se zaměříme na tu počáteční, která je výsledkem procesu nazývaného konceptuální modelování [3].



Obrázek 3.1: Fáze návrhu databáze (převzato z [3])

Jakmile byly známy požadavky na uložená data v budoucím informačním systému, byl potřeba vytvořit model popisující data v databázi, nezávisle na logické úrovni i fyzickém uložení databáze. Tento tzv. konceptuální model je model, zakreslující funkční a informační potřeby oddílu. Zabývá se pouze potřebami oddílu a nezabývá se problémem jejich implementace.

Pod takovýmto model si lze představit známý Entity Relationship Diagram nebo modely vytvořené v jazyce UML. Pokud se nám podaří takovéto konceptuální schéma vytvořit, usnadní to společné chápání objektů aplikace uživateli a projektanty. Pomáhá to také usnadňovat diskuzi mezi těmito lidmi předcházet tak chybám a nedorozuměním.

3.1.1 Entity Relationship Diagram

Již z názvu je nám jasné, že hlavní komponenty používané v ER diagramu jsou entity, vztahy a omezení. Entity modelují objekty, které se vyskytují v modelovaném fyzickém systému. Chápeme je jako aspekty reálného světa, které jsou odlišitelné od ostatních aspektů. Mohou jimi být třeba studenti, profesoři nebo předměty na vysoké škole. Každá entita je reprezentována množinou atributů. Vztahy modelují spojení mezi těmito entitami – například profesoři učí předměty. Na

libovolných entitách a vztazích můžeme specifikovat integritní omezení. Tím může být například fakt, že student si může zapsat maximálně 186 kreditů za bakalářské studium.

3.1.2 UML

Další typ konceptuálních schémat s grafickou syntaxí lze modelovat v jazyce UML. Univesal Modeling Language představuje soubor inženýrských praktik, které byly ověřeny jako úspěšné pro modelování rozsáhlých a komplexních systémů. Kromě četných notací k různým účelům zahrnuje UML také diagramy pro statické třídy, stavové přechody nebo komponenty. V oblasti analýzy a návrhu je UML dnes standardem a je tedy dobré se v něm orientovat.

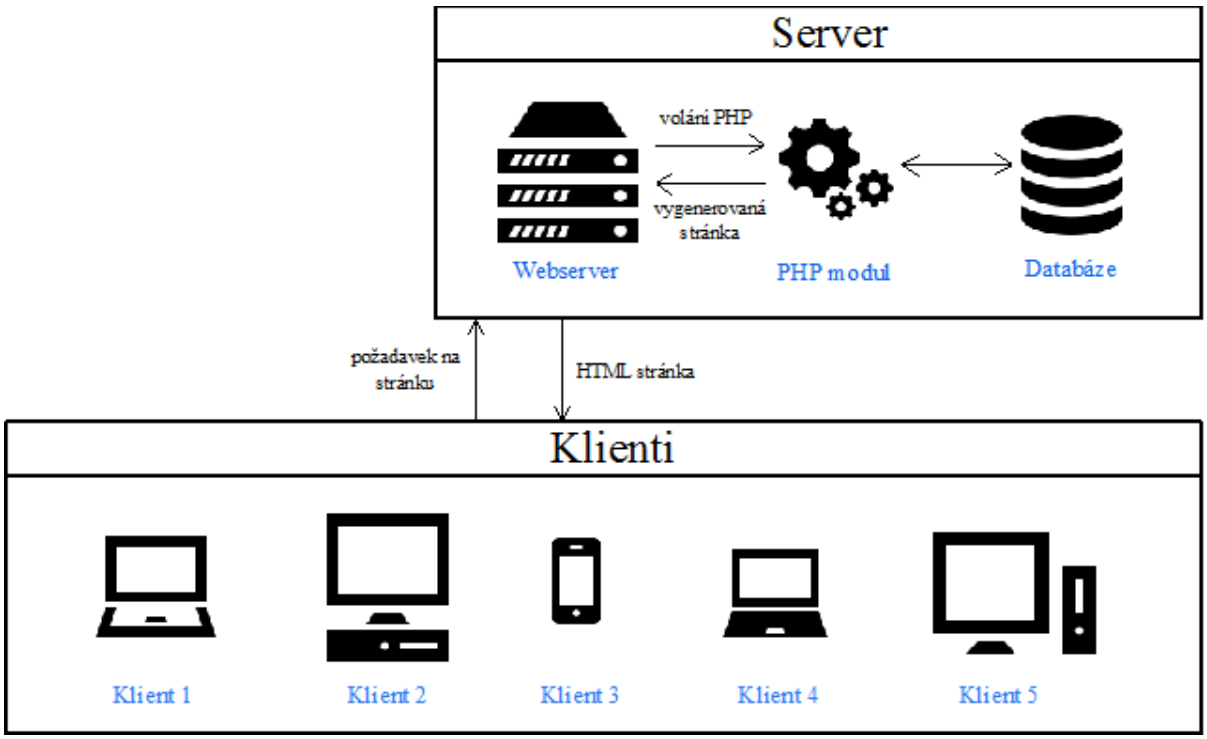
4 Použité technologie

Následuje jednoduchý popis a představení technologií, které byly využity při tvorbě informačního systému.

4.1 PHP

Od chvíle, kdy vznikla potřeba přidávat do internetových stránek dynamickou funkčnost, došly naše možnosti tak daleko, že jsme schopni vytvořit webovou aplikaci s chováním shodným jako u desktopové aplikace. Využíváme k tomu technologie pro programování dynamických internetových stránek a webových aplikací, mezi které neodmyslitelně patří skriptovací jazyk PHP.

Jak je vidět na obrázku 4.1, PHP provádí skripty na straně serveru a k uživateli je pouze přenášen výsledek jejich činnosti. Stránka webové aplikace jako taková, kterou vidíme v prohlížeči, neleží již na serveru ale je vytvářena dynamicky podle toho co uživatel požaduje. Na základě tohoto požadavku načte PHP data z databáze a vygeneruje příslušnou HTML stránku.



Obrázek 4.1: Obsluha klientů serverem s PHP modulem (převzato z itnetwork.cz)

Prvenství tohoto hypertextového preprocesoru v rozšířenosti mezi skriptovacími jazyky pro web můžeme přisuzovat jeho jednoduchosti, rozsáhlého souboru funkcí v základní knihovně, nativní podpoře mnoha databázových systémů nebo multiplatformní použití.

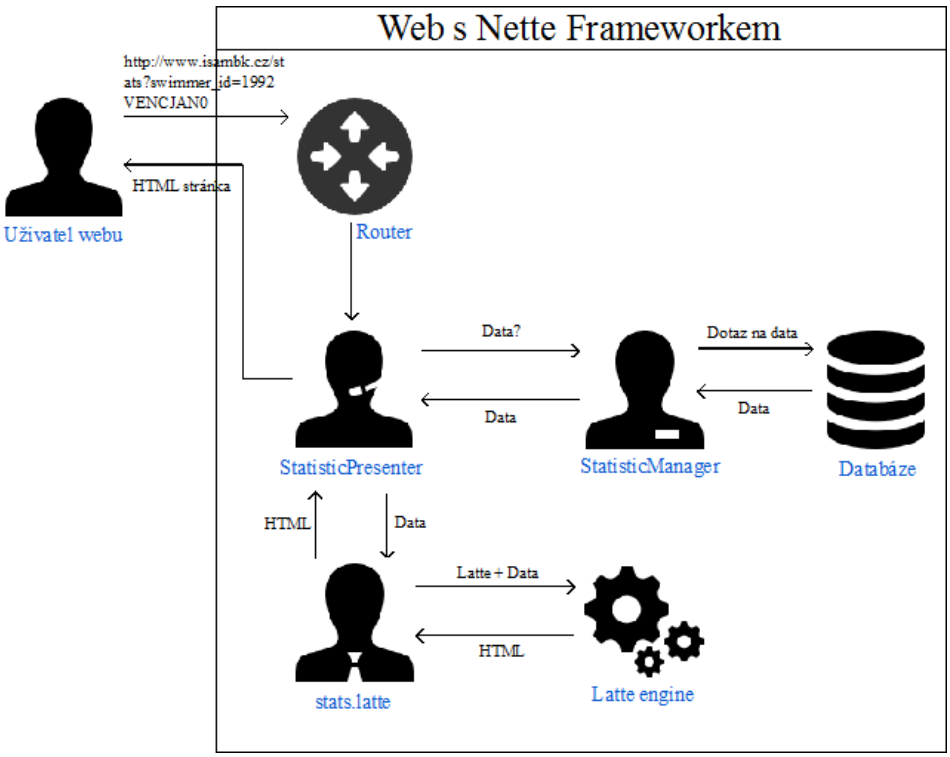
Odpůrcům PHP může naopak vadit nekonzistentní pojmenování funkcí, nejednotné pořadí parametrů u některých funkcí, nebo že ve standardní distribuci chybí ladící nástroj [4].

4.1.1 Nette Framework

Framework obecně je knihovna, která má ulehčit práci programátorovi. Vývojář je pak schopen psát rychleji a přehledněji s minimem úsilí. V zásadě jde o to, abychom se nezabývali programováním pořád stejných a rutinních funkcí od začátku, ale věnovali se místo toho vymýšlení nových zajímavých funkcí.

Nette Framework je dílem českého vývojáře Davida Grudla. Byl vytvořen jako open source framework pro tvorbu webových aplikací v PHP 5 s důrazem na eliminaci bezpečnostních rizik. Když programujeme v Nette, musíme počítat s objektovým přístupem a také znát základy a principy tzv. MVC architektury, kterou si přiblížíme v kapitole 6.1.

V Nette tuto architekturu představuje princip MVP. Písmenko M zastupuje model. Model je PHP soubor, který obsahuje třídu obsluhující logiku aplikace. Třeba práce s databází nebo různé výpočty. Písmeno P představuje presenter. Tato komponenta komunikuje s uživatelem a je takovým prostředníkem mezi komponentami modelem a šablonou. Od modelů získává data, která posílá výstupu (šabloně). Vše na základě parametrů od uživatele, kterému nakonec vrátí HTML stránku. Šablona (písmenko V jako výstup) obsahuje Latte šablony s HTML kódem. Pomocí šablonovacího jazyka Latte nám Nette umožňuje vkládat do HTML kódu data z PHP [5]. Pro názornost si uvedeme životní cyklus aplikace v Nette na příkladu zobrazení statistik o plavci.



Obrázek 4.2: Podstata fungování webu s Nette Frameworkem (převzato z itnetwork.cz)

Požadavek *http://www.isambk.cz/stats?swimmer_id=1992VENCJAN0* od uživatele se dostane k routeru, který z něho pozná, že má zavolat StatisticPresenter. Ten s parametrů url adresy zjistí, kterou akci (metodu) má provést. Zavolá tedy model StatisticManager, který získá data z databáze a vypočítá statistiky o daném plavci. Data vrátí do presenteru a ten je zase předá šabloně. Ta má pro ně připravenou HTML stránku s Latte značkami, do kterých je Latte engine automaticky vloží. StatisticPresenter potom obrží už hotovou HTML stránku se statistikami, kterou pošle uživateli.

4.2 MySQL

Multiplatformní databáze MySQL je v dnešní době jeden z nejpoužívanějších databázových systémů [4]. A to hlavně díky své rychlosti, bezplatném licenci GPL[1] nebo kompatibilitě s ostatními systémy (Apache, PHP). Ovládána je jazykem SQL, který je podobný jako u všech SQL databází. Pro uživatelsky jednodušší správu databáze se používají různé nástroje, např. phpMyAdmin.

Webové aplikace často používají čtveřici - linuxový operační systém, webserver Apache, databázi MySQL a programovací jazyk PHP. Tato čtveřice se zkráceně pojmenovává jako LAMP. Existuje také WAMP pokud tyto technologie provozujeme pod operačním systémem Windows.

Pro potřeby vývoje a testování tohoto informačního systému byl zřízen lokální web server s nastavením WAMP. WAMP byl instalován a nakonfigurován s využitím instalačního balíčku XAMPP, který nabízí tyto technologie napříč platformami a je zdarma je stažení.

4.2.1 InnoDB

Databáze MySQL podporuje mnoho typů tabulek. Některé jsou určeny pro informace s dlouhodobou platností. Jiné existují jen proto, aby usnadnily údržbu a prohlížení kolekce tabulek. Zmíněné a jiné typy tabulek určuje tzv. ukládací engin databáze. Jedním takovým enginem je i InnoDB, který byl použit v práci. InnoDB má za sebou již více jak desetiletou historii a je využíván i velikány jako jsou společnosti Yahoo! nebo Google. Velký důraz klade na efektivnost v oblasti transakčních aplikací. Díky tomuto ukládacímu enginu MySQL podporuje například uzamykání na úrovni řádků nebo cizí klíče. Za zmínku stojí také schopnost automatického zotavení po havárii. Dalšími ukládacími enginy jsou například MyISAM, IBMDB2I, MEMORY nebo MERGE [4].

4.3 HTML

HTML je zkratka pro HyperText Markup Language, což je značkovací jazyk pro tvorbu internetových stránek. Společně s CSS a Javascriptem je HTML základním stavebním kamenem

[1] General Public License (všeobecně veřejná licence)

všech moderních webových stránek a aplikací. Původ jazyka najdeme v univerzálním značkovacím jazyce SGML.

Pomocí HTML vytváříme strukturovaný textový dokument. Užíváme tzv. tagy (značky) a jejich atributy (vlastnosti) a tím určujeme sémantiku (význam) našeho dokumentu. Dvojice tagů, uzavřených do úhlových závorek (<, >), tvoří element. Ukončovací tag je odlišen od toho otevíracího znakem lomítko. Některé tagy mohou být také nepárové (<hr>,
). Ukázka párového tagu s atributem, který se používá jako odkaz:

Informační systém iSAM

Do HTML dokumentu lze také vkládat kód skriptovacích jazyků nebo kaskádové styly, ale z důvodu přehlednosti a udržovatelnosti se tento postup nedoporučuje. Ideální je si vytvořit zvláštní soubor se stylovým předpisem a soubor se skripty. Ty se potom akorát přilinkují v hlavičce dokumentu.

Struktura HTML dokumentu je jasně daná. Nejprve je nutné sdělit prohlížeči s jakým typem dokumentu má co dočinění. To provedeme deklarací typu dokumentu pomocí značky <!DOCTYPE html>. Následně definujeme kořenový element značkami <html> a </html>, který obepíná celý dokument. Dále vložíme hlavičku mezi tagy <head> a </head>. Tento element naplníme metadaty o dokumentu (kódování, název dokumentu, autora, popis, popř. klíčová slova, …). Samotné tělo (<body>, </body>) obsahuje vlastní náplň naší internetové stránky.

Aktuálně používaná verze je HTML 5.0 z roku 2014, což je patnáct let po předchozí verzi.

4.4 CSS

Pokud se nám podaří napsat internetovou stránku v HTML, měli bychom k ní vytvořit stylový předpis v jazyce CSS a definovat tím způsob zobrazení elementů. Samotné HTML také umí zvýraznit text nebo kurzívu, ale nám to určitě nestačí. Jak již bylo řečeno, je lepší mít samostatný soubor pro styly a v něm definovat jednotlivá pravidla. Celý stylový předpis je vlastně soubor pravidel vztahujících se ke konkrétním elementům. Pravidla se skládají ze selektoru a bloku deklarací uzavřeného ve složených závorkách. Uvnitř tohoto bloku uvádíme deklarace oddělené středníkem. Deklaraci tvoří identifikátor vlastnosti, následuje dvojtečka a hodnota vlastnosti. Pokud máme více selektorů k jednomu bloku deklarací, musíme je oddělit čárkami.

selektor1, selektor2 {vlastnost: hodnota}

Důležitou vlastností, s kterou musíme počítat, je dědičnost. Většina vlastností se dědí a element, který nemá vlastnost definovanou, jí dědí po nadřazeném elementu. Týká se to především vlastností písma. Dále je dobré vědět, váha předpisu (důležitost) je určena místem jeho definice. To znamená, že v případě dvou pravidel, které mají stejný selektor ale jedno je na začátku dokumentu a druhé na konci, se uplatní vlastnosti onoho druhého pravidla [6].

4.4.1 Bootstrap Framework

Při tvorbě uživatelského rozhraní a designu internetové aplikace pomocí CSS a JavaScriptu si můžeme ušetřit práci. Upustíme od přístupu, kdy si pro každou komponentu našeho webu musíme vymýšlet vlastní pravidla do stylového předpisu. Místo toho použije jeden z front-end[2] frameworků, který nám usnadní práci s typografií, tvorbou layoutu, vytváření elementů uživatelského rozhraní a zároveň ošetří zobrazování napříč platformami.

Jedním takovým frameworkem je Bootstrap , který byl použit v této práci. Tato volně stažitelná sada nástrojů obsahuje množství HTML, CSS a JavaScriptových komponent, které stačí jenom použít. Je nutné pouze stáhnout CSS stylopis (popř. JavaScript) Bootstrapu a nalinkovat ho do hlavičky HTML souboru. Potom už stačí dodržovat strukturu HTML dokumentu popsanou v podrobné dokumentaci frameworku a přidáváním určených tříd vytvářet profesionálně a moderně vypadající web. Framework nám také zajistí, že rozložení stránky se dynamicky přizpůsobí s ohledem na používané zařízení (stolní PC, tablet, mobilní telefon). Takovému designu se říká responzivní.

4.5 JavaScript a jQuery

Jak JavaScript tak jQuery jsou technologie, pomocí kterých definujeme chování webu. Chováním jsou myšleny různé efekty a animace, výběr a změna objektů DOM [3], obsluha událostí (kliknutí na tlačítko) nebo manipulace s CSS. Rozdíl mezi těmito technologiemi je ten, že JavaScript je programovací jazyk a jQuery je JavaScriptová knihovna (framework), která usnadňuje práci s JavaScriptem. jQuery klade důraz na jednoduchost, čitelnost a rychlost. Automaticky řeší nekompatibilitu mezi prohlížeči a je multiplatformní.

V této práci byla použita knihovna jQuery, s kterou také pracuje Bootstrap Framework. Byl použit také princip neobtruzivního (nevtíravého) JavaScriptu, což znamená, že JavaScriptový kód je umístěn do zvláštního souboru a není smíchán s HTML. Stejně jako je zvykem u CSS.

4.5.1 Google Charts

Google Charts je nástroj k vizualizaci dat na internetových stránkách od společnosti Google. Před použitím je nutno načíst určité knihovny specifikované v dokumentaci. Následně se pomocí JavaScriptu určí data k vizualizaci a definuje nastavení grafu. A konečně můžeme instanciovat a vykreslit graf s využitím funkcí k tomu určeným. Potom už stačí v HTML dokumentu vytvořit element s identifikátorem stejným jaký jsme zvolili při instanciaci [7]. Google Charts nám na toto místo vloží interaktivní graf nebo tabulku. Postup je podrobněji popsán v kapitole 6.4.3.

[2] Front-end je část, kterou vidí návštěvník stránek. Na rozdíl od back-endu, což je například aministrační rozhraní eshopu sloužící k přidávání zboží.
[3] Document Object Model (objektový model HTML dokumentu)

Pro potřeby vizualizace statistik plavců v této práci byl použit graf čárový, koláčový a sloupcový. Data jsou do těchto grafů plněny dynamicky podle toho, jakého plavce (popř. disciplínu a rok) si uživatel zvolí. K zobrazení žebříčků oddílu byly zvoleny tabulky nabízené taktéž touto službou.

Výhodou nástroje je, že grafy jsou renderovány s využitím HTML5/SVG[4], což zajišťuje kompatibilitu napříč platformami a webovými prohlížeči. Díky vektorové grafice je rovněž bez problémů zvětšování/zmenšování obrázků.

4.5.2 Google Analytics

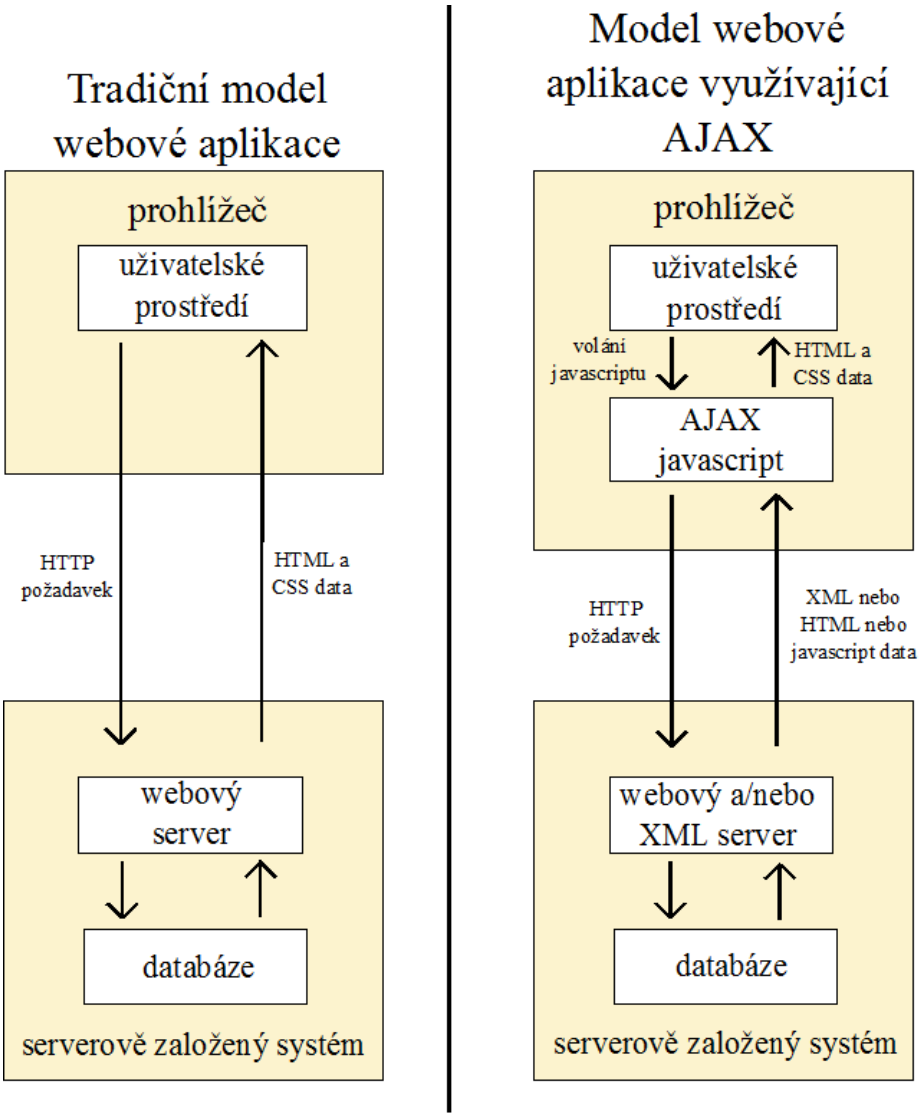
Další službou nabízenou společností Google je nástroj pro vlastníky internetových stránek, který jim umožňuje získávat statistická data o uživatelích těchto stránek, jejich aktivitách a chování. Dále lze monitorovat například dění na webu v reálném čase, průměrnou dobu trvání návštěv, pohyb uživatelů na stránkách nebo zařízení, ze kterých se uživatelé připojují nejčastěji. Používání Google Analytics je zdarma. Stačí se pouze zaregistrovat a zkopírovat přidělený měřící kód do hlavičky svých HTML dokumentů, které chceme měřit. Výsledky měření můžeme procházet přes webové rozhraní služby nebo také v mobilní aplikaci.

4.6 AJAX

AJAX nebo-li *Asynchronous JavaScript and XML* označuje soubor technologií spolupracujících s určitým cílem. Jedná se o HTML, CSS JavaScript, HTTP a XML, které se starají o asynchronní zpracování webových stránek. AJAX dokáže kontaktovat server a získat od něj libovolná data v XML, bez toho, aby se musela celá stránka znovu nahrávat. To nám umožňuje překreslit jenom konkrétní část stránek, s kterou uživatel právě pracuje. Na rozdíl od klasického přístupu, kdy na základě každého HTTP požadavku pošle webový server celý HTML dokument (obrázek 4.3). Například kvůli zobrazení výpočtu bodové kalkulačky není potřeba aktualizovat celý obsah stránky a tím uvést všechny parametry nastavené uživatelem do defaultního stavu. AJAX tak výrazně přispívá k plynulosti a rychlosti internetových stránek. S omezením potřeby přenášet kompletní HTML dokument souvisí také snížení zátěže webových serverů [8].

Jako nevýhodu AJAXu můžeme vidět to, že užíváním AJAXových komponent na stránce se nemění url adresa a tím pádem použitím tlačítka zpět v prohlížeči se vrátíme na předchozí url bez ohledu na aktivity provedené pomocí AJAXu. Nebo pokud uživatel zadá AJAXový požadavek s delší dobou zpracování (složitý dotaz do databáze), tak nepozná, jestli systém jeho žádost ignoruje nebo něco nefunguje. V těchto případech je vhodné požádat uživatele o trpělivost vhodným textem nebo animací.

[4] Scalable Vector Graphics (škálovatelná vektorová grafika) - značkovací jazyk a formát souboru, který popisuje dvojrozměrnou vektorovou grafiku pomocí XML



Obrázek 4.3: Porovnání tradiční webové aplikace s aplikací využívající AJAX (převzato z en.wikipedia.org/wiki/Ajax_(programming))

Podporu AJAXu nabízí také Nette Framework, který díky speciální knihovně jeho používání výrazně usnadňuje.

4.7 Git

Když skupina programátorů pracuje na větším projektu, potřebují si někde uchovávat výsledky své činnosti tak, aby k nim měli přístup všichni členové týmu kdykoli a odkudkoli. K tomu se používají SCM [5] systémy. Jedním takovým systémem je i Git. I když autor Linus Torvalds zásadně odmítá přirovnávat Git k tradičnímu SCM díky systému větvení, našim účelům to postačí. Git je tedy distribuovaný verzovací systém s možností vytváření větví a jejich slučování. Možnost větvení podporuje vývojáře zkoušet nové nápady bez nutnosti sdílet je s ostatními. Zmíněná distribuovanost

[5] Source control management (systémy pro správu zdrojových kódů)

systému poskytuje každému vývojáři lokální kopii celé historie vývoje. Změny se kopírují z jednoho úložiště do jiného a importují se v podobě dalších vývojových větví [9].

V tomto projektu nebyl Git použit z důvodu spolupráce více programátorů, nýbrž kvůli potřebě organizovaně zálohovat odvedenou práci a mít přístup k předešlým verzím aplikace. Jako hosting pro Git repozitář byla použita webová služba GitHub, která nabízí bezplatné uložení open-source projektů.

5 Analýza a návrh informačního systému

Nyní už víme, jaká je motivace k vytvoření informačního systému. Také jsme se seznámili s technologiemi, díky kterým můžeme tento projekt uskutečnit. Nic nám tedy nebrání v tom, představit si počáteční fázi vývoje informačního systému pro plavecký oddíl.

5.1 Analýza požadavků

Na základě rozhovorů s trenéry a správcem databáze vznikly požadavky na informační systém, který má být přístupný jak plavcům a trenérům z oddílu tak veřejnosti. Jeho základním stavebním kamenem bude databáze plavců, závodů a časů. Údaje budou veřejně přístupné k prohlížení. V datech půjde vyhledávat a třídit podle libovolného sloupce. Uživatel si vybraná a setříděná data bude kromě prohlížení moci exportovat do souborů typu PDF a CSV dle zvolených sloupců. Požadavek na export do formátu CSV (Comma-separated values) vznikl z důvodu možnosti provádět vlastní analýzu dat např. v programu MS Excel. Uživatel bude mít rovněž možnost vygenerovat statistiku s grafy o každém plavci v databázi nebo zobrazit kartu detailu se základními údaji a fotkou. U každého závodu bude dostupný výpis zúčastněných plavců z oddílu. Aplikace bude obsahovat seznam všech disciplín bazénového plavání s možností porovnání oddílového, českého i světového rekordu u každé disciplíny. Systém bude schopen vygenerovat z databáze různé typy žebříčků oddílu a nabídnou je uživateli ke stažení v PDF. Mezi možnosti aplikace bude také patřit bodová kalkulačka, která dokáže uživateli vypočítat body ze zadaného času zvolené disciplíny a naopak.

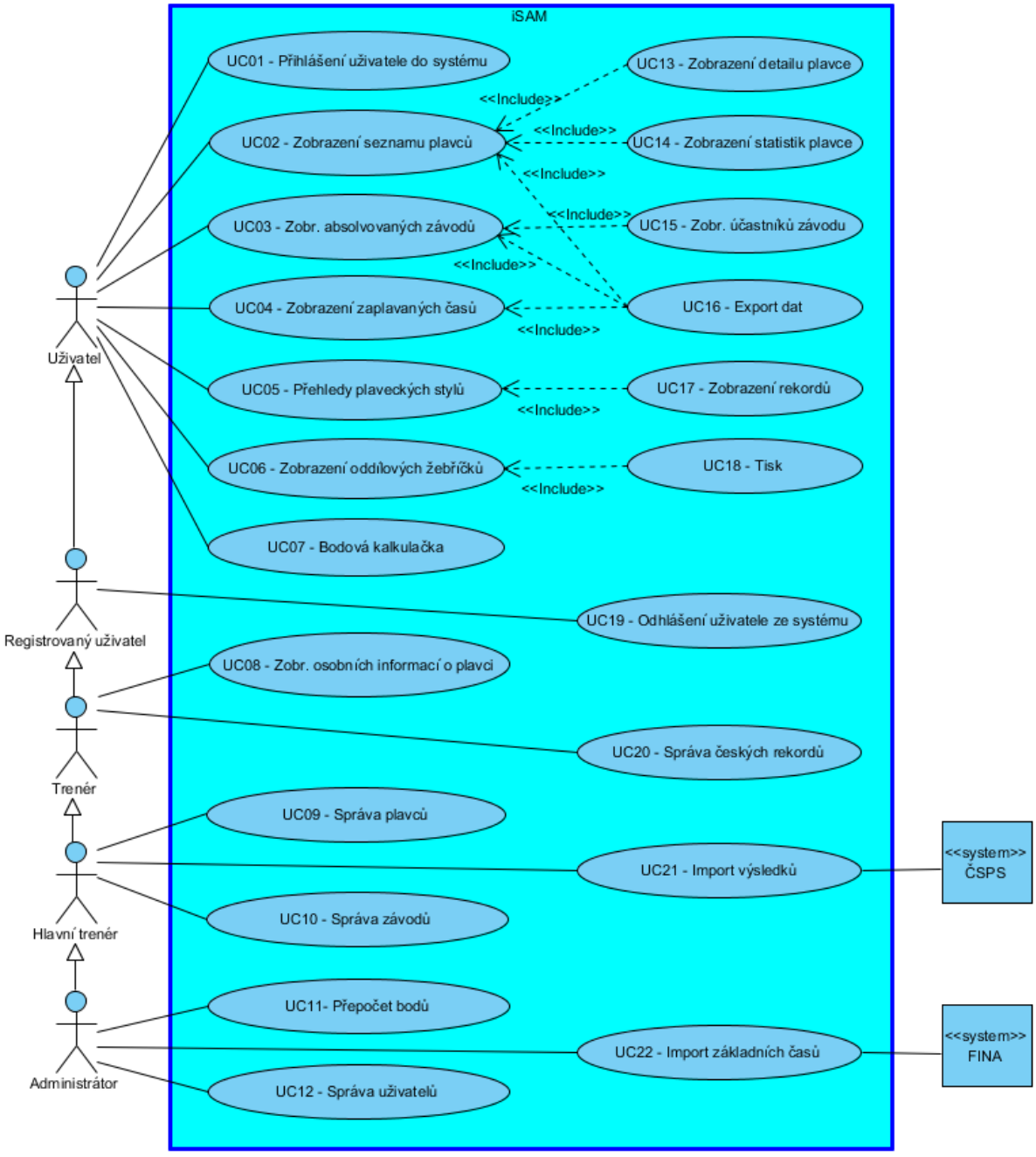
Systém bude rozlišovat čtyři typy uživatelů – administrátor, hlavní trenér, trenér a běžný uživatel. Administrátor bude mít přístup ke všem funkcím aplikace. Administrátor i hlavní trenér mohou být pouze jeden. Výše uvedené požadavky na funkčnost budou dostupné také pro běžného uživatele bez nutnosti autentizace. Následující požadavky už vyžadují přihlášení. Hlavní trenér bude mít možnost načíst výsledky ze závodů do databáze aplikace. Výsledky jsou v textové podobě dostupné na serveru Českého svazu plaveckých sportů. Také byl vysloven požadavek na uchovávání adres a kontaktních údajů plavců jako náhrada doposud vedené papírové kartotéky. Tyto údaje budou moci zobrazit trenéři a hlavní trenér, nicméně jejich editaci bude moci provádět pouze hlavní trenér. To samé se týká fotek plavců. Systém umožní hlavnímu trenérovi vkládat (popř. mazat) do databáze nové plavce a závody. Trenéři a hlavní trenér budou moci upravovat hodnoty českých rekordů.

Z důvodů komplikovanější přípravy dat bude pouze administrátor oprávněn nahrát do systému aktuální ročník základních časů (base time). Hodnoty jsou ve formátu excelovských tabulek dostupné na serveru Mezinárodní plavecké federace (FINA). Systém si bude ukládat historii těchto časů podle roku vytvoření. Administrátorovi tak bude umožněno přepočítat bodové hodnoty časů v databázi podle požadovaného ročníku základních časů. Tím bude zajištěna aktuálnost bodových hodnot ve

všech funkcích systému. Na základě osobní nebo písemné (email) žádosti od hlavního trenéra bude mít administrátor v aplikaci možnost založit účet pro nového trenéra.

5.2 Diagram případů užití

Následující diagram případů užití (use case diagram) zobrazuje chování systému z hlediska uživatele. Některé případy užití (např. UC09 - Správa plavců) jsou zobecněny z důvodu zajištění přehlednosti diagramu.



Obrázek 5.1: Use Case Diagram

5.3 Detaily případů užití

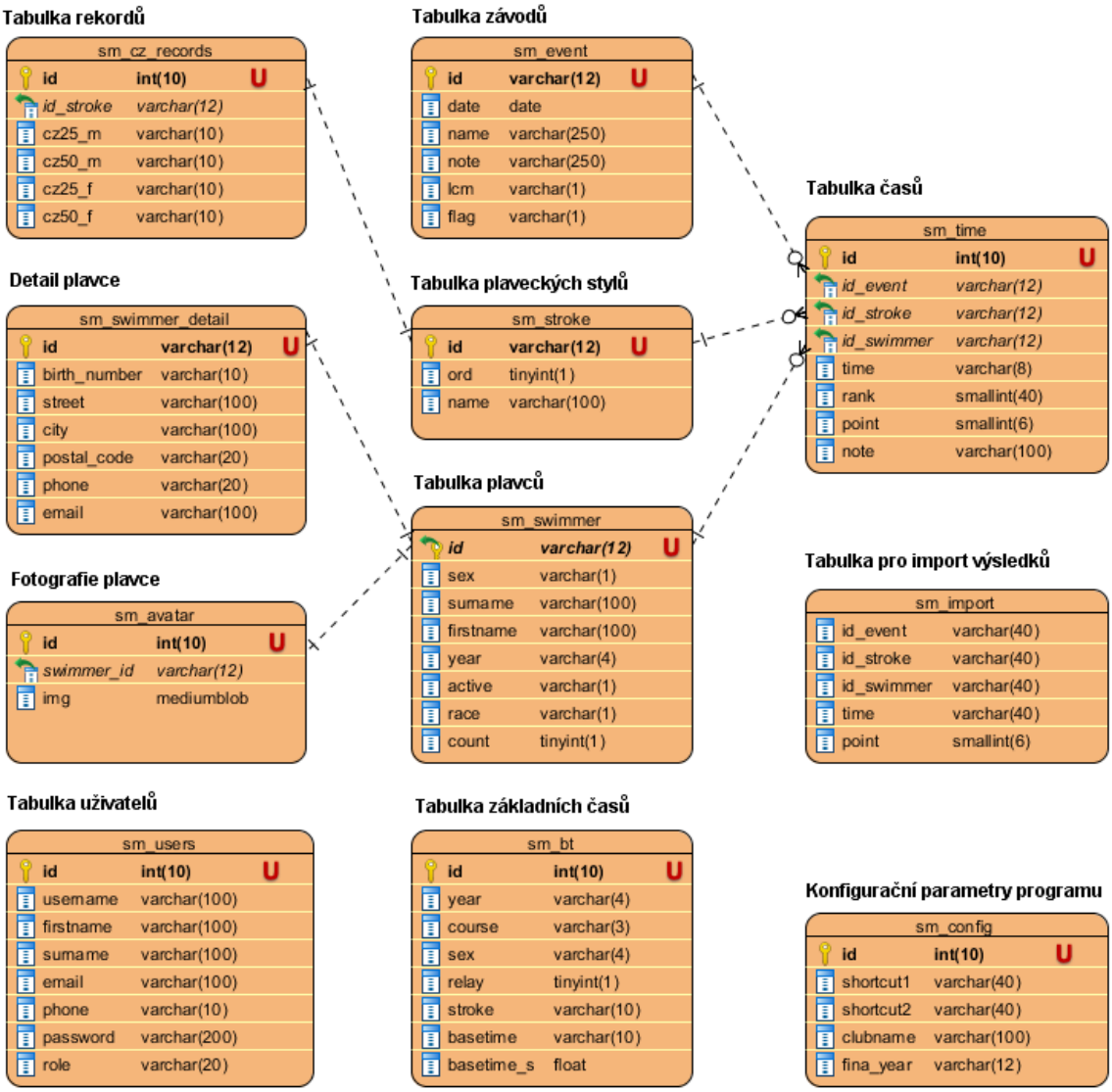
Případ užití	**Import výsledků**	**ID:**	**UC21**
Popis	Import výsledků ze závodů dostupných na serveru ČSPS.		
Uživatelé	Administrátor, Hlavní trenér		
Vstupní podmínky	1. Aktér je přihlášen do systému. 2. Aktér vytvořil nový závod.		
Následné podmínky	1. Systém importuje výsledky do tabulky časů.		
Hlavní tok	1. Aktér vyhledá odpovídající identifikátor závodu, který neobsahuje výsledky. 2. Aktér nahraje soubor s výsledky. 3. Aktér zahájí kontrolu souboru stiskem tlačítka „Zkontrolovat“ 4. KDYŽ soubor obsahuje chyby 4.1 Systém oznámí aktérovi konkrétní chyby a vyzve aktéra k jejich opravě ve zdrojovém souboru. 4.2 Případ užití končí. 5. Aktér zahájí import výsledů stiskem tlačítka „Importovat“ . 6. Systém importuje výsledky do tabulky časů.		
Výjimky	Selhání systému, Storno		
Frekvence	Několikrát do měsíce		

Případ užití	**Zobrazení účastníků závodu**	**ID:**	**UC15**
Popis	Zobrazí se výčet všech zúčastněných plavců zvoleného závodu.		
Uživatelé	Administrátor, Hlavní trenér, Trenér, Běžný uživatel		
Vstupní podmínky	Žádné		
Následné podmínky	1. Systém zobrazí výčet zúčastněných plavců.		
Hlavní tok	1. Aktér zvolí položku nabídky „Závody“. 2. Aktér klikne na ikonu „Zobrazit účastníky“ u konkrétního závodu. 3. Systém zobrazí výčet zúčastněných plavců.		
Výjimky	Selhání systému, Storno		
Frekvence	Několikrát do měsíce		

Případ užití	**Správa českých rekordů**	**ID:**	**UC20**
Popis	Provede se aktualizace českého rekordu.		
Uživatelé	Administrátor, Hlavní trenér, Trenér		
Vstupní podmínky	1. Aktér je přihlášen do systému.		
Následné podmínky	1. Systém provede aktualizaci českého rekordu.		
Hlavní tok	1. Aktér zvolí položku nabídky „Styly“. 2. Aktér klikne na ikonu „Aktualizovat český rekord“. 3. Aktér vybere disciplínu. 4. Aktér zvolí pohlaví. 5. Aktér zvolí typ bazénu. 6. Aktér zadá novou hodnotu českého rekordu 7. Aktér uloží změnu kliknutím na tlačítko „Uložit“. 8. Systém provede aktualizaci českého rekordu.		
Výjimky	Selhání systému, Storno		
Frekvence	Jednou do roka		

5.4 ER diagram

Uvedený ER Diagram popisuje schéma databáze. Žlutým klíčem jsou označeny primární klíče tabulek a položky se zelenou šipkou představují cizí klíče. Červené písmeno U u primárních klíčů reflektuje jejich unikátnost.



Obrázek 5.2: ER Diagram

6 Implementace

Dostáváme se ke stěžejní části vývoje informačního systému, kterou je praktické uskutečnění výše uvedených požadavků a návrhů. Cizím slovem implementace. Jak vyplývá z předchozího textu, k implementaci byl použit PHP framework Nette. K vývoji aplikace bylo použito vývojové prostředí NetBeans IDE.

6.1 Architektura systému

Internetové stránky, kterým se věnuje tato práce, nejsou jednoduchou webovou prezentací, ale webovou aplikací s určitou funkcionalitou. Tuto funkcionalitu zajišťuje PHP v podobě různých logických operací a dotazů do databáze. Kdybychom PHP kód mísili s HTML kódem, výsledek by byl velmi nepřehledný a špatně udržovatelný. Proto vznikla architektura MVC, která si dává za úkol logiku od výstupu oddělit. Výhodou takového architektonického vzoru je také snazší kooperace skupiny vývojářů, kde se každý specializuje na jednu věc (tvorba logických funkcí, prezentace dat na webu).

MVC architektura se skládá s komponent tří typů. Jak napovídá název, jsou jimi Model, View a Controller [10]. Dalším přínosem tohoto rozdělení je možnost upravovat jednotlivé komponenty zvlášť s minimálním ovlivňováním ostatních. Snadno také můžeme napojit více různých pohledů, aniž bychom museli duplikovat kód tzv. bussines logiky (kód důležitý pro samotnou funkčnost aplikace). MVC architektura je součástí většiny dnešních PHP frameworků. Nette Framework používá MVP architekturu, která je od MVC odvozena a vysvětlena v kapitole 4.1.1.

6.1.1 Model

Model reprezentuje data a obsahuje veškerou logiku aplikace. Pod tím si lze představit komunikaci s databází nebo různé výpočty. Tato komponenta se nestará o původ parametrů, které jsou jí předány a ani neovlivňuje způsob formátování svých výstupních dat.

6.1.2 View

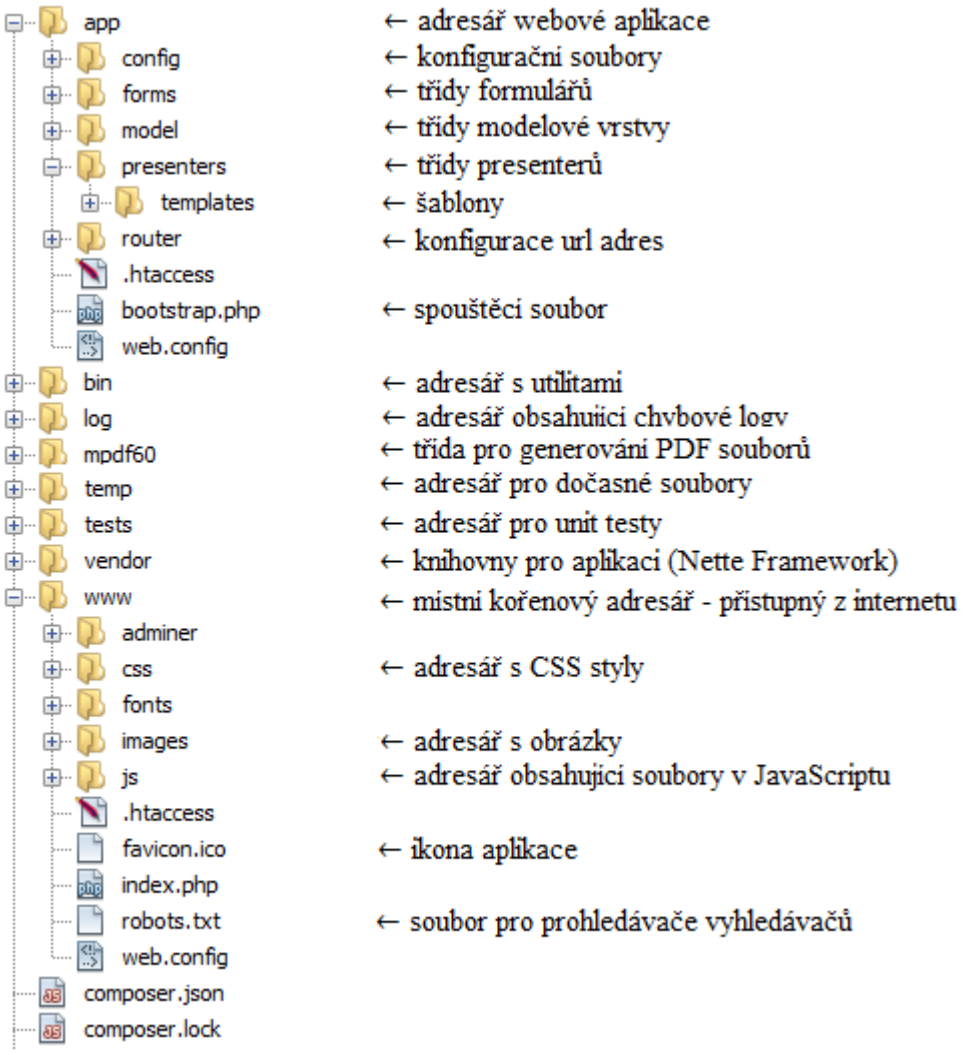
View nebo-li pohled získaná data převádí do podoby vhodné k prezentaci uživateli. View obsahuje HTML šablonu, do které jsou pomocí tagů nějakého značkovacího jazyka vkládána konkrétní data. K tomu můžeme využít různé cykly a podmínky. View je závislý pouze na datech, která jsou mu poskytnuta. Neměl by mít možnost si je sám brát.

6.1.3 Controller

Controller je propojující prvek obou komponent, který vyřizuje požadavky od uživatele. Na základě těchto požadavků komunikuje s modelem a pohledem a stará o celkové provázání funkcí aplikace.

6.2 Adresářová struktura

Adresářová struktura (obrázek 6.1) projektu vychází z doporučené adresářové struktury Nette Frameworku, známé jako Sandbox.



Obrázek 6.1: Adresářová struktura projektu

6.3 Výběry z databáze

Výběry z databáze jsou jedněmi z nejdůležitějších funkcí tohoto informačního systému. Jsou přítomny téměř na každém rohu nabídky aplikace. Ať už se jedná třeba o výpis závodů podle určitých kritérií, sestavení žebříčků oddílu nebo zobrazení statistik o plavci. Nette nám k těmto účelů nabízí metody, které komunikaci s databází značně zjednodušují [11].

6.3.1 Přístup k databázi

Než můžeme tyto metody použít, musíme si nejprve nakonfigurovat přístup k databázi. V souboru *config.local.neon* nastavíme jméno host serveru, jméno databáze, uživatelské jméno a heslo. Následně tam, kde chceme komunikovat s databází, vytvoříme instanci třídy *Nette\Database\Context* a v konstruktoru ji uložíme do privátní proměnné $database.

```
/** @var Nette\Database\Context */
private $database;

public function __construct(Nette\Database\Context $database)
{
    $this->database = $database;
}
```

Od této chvíle můžeme do databáze přistupovat přes tuto proměnnou. Při interakci s databází byla nejčastěji využívána metoda *query*, která generuje a provede SQL dotaz do databáze. Jako parametr přijímá SQL dotaz v podobě řetězce popř. hodnoty zastoupené v dotazu otazníky. Následuje ukázka takového dotazu, který vybere z databáze jméno a příjmení plavce na základě jeho identifikátoru.

```
$plavec = $this->database->query("SELECT surname, firstname FROM
sm_swimmer WHERE sm_swimmer.id = ? ", "1992VENCJAN0");
```

Samozřejmě pomocí této metody lze provádět také příkazy INSERT, DELETE nebo UPDATE.

6.3.2 Dynamická tvorba dotazů

V příkladu výše je SQL dotaz jasně dán. V aplikaci ale existují funkce, které vyžadují dynamické sestavení dotazu na základě zvolených výběrových a třídících kritérií. Uživatel zadá tyto kritéria do připraveného formuláře, který následně odešle. Po sérii kontrol správnosti zadaných kritérií se provede jejich rozdělení podle typu (WHERE nebo ORDER BY) a vložení do příslušných dvojrozměrných polí. Ta mohou vypadat například takto:

```
$par_where = array (
        0 => array ("surname", "like", "V%"),
        1 => array ("sex", "=", "M"),
        2 => array ("year", "<", "2000")
)
$par_order = array (
        0 => array ("year", "DESC")
)
```

Tato pole se potom použijí k vygenerování dotazu, který je následně předán metodě *query* a proveden. Výsledkem je výběr plavců, kterým příjmení začíná na písmeno V, pohlavím jsou to muži a jejich ročník je menší jak 2000. Celý tento výběr je seřazen podle ročníku sestupně.

6.3.3 Složitější dotazy

Další zajímavé dotazy jsou dotazy, které dávají vznik oddílovým žebříčkům. Jako příklad si uvedeme a popíšeme dotaz, který se používá k sestavení pořadí plavců podle času v určité disciplíně.

```
SELECT
        @rownum := @rownum + 1 AS rank, sel_2.cas, sel_2.prijmeni, sel_2.jmeno,
        sel_2.rocnik, sel_2.datum, sel_2.zavod
FROM
        (SELECT -- 2. select
                MIN(sel_1.cas) AS cas, plavci.surname AS prijmeni,
                plavci.firstname AS jmeno, plavci.year AS rocnik,
                zavody.date AS datum, zavody.name AS zavod
        FROM
                sm_time casy, sm_event zavody, sm_swimmer plavci,
                ( SELECT  -- 1. select
                        a.id_swimmer AS plavec, MIN(a.time) AS cas
                FROM
                        sm_time a, sm_event b, sm_swimmer c
                WHERE
                        a.id_event = b.id AND a.id_swimmer = c.id AND c.sex = ? AND
                        b.lcm = ? AND a.id_stroke = ?
                 GROUP BY
                        a.id_swimmer
                ) sel_1
        WHERE
                casy.id_swimmer = sel_1.plavec AND casy.time = sel_1.cas AND
                casy.id_event = zavody.id AND casy.id_swimmer = plavci.id AND
                plavci.sex = ? AND zavody.lcm = ? AND casy.id_stroke = ?
        GROUP BY
             plavci.surname, plavci.firstname, zavody.date, zavody.name, plavci.year
        ORDER BY  1
```

```
) sel_2,
(SELECT @rownum := 0) sel_3
```

Uvedený dotaz se skládá ze tří dotazů, z toho dvou vnořených. Modrý dotaz nám vybere ke každému identifikátoru plavce jeden nejlepší čas v dané disciplíně, pohlaví a délky bazénu. Červený dotaz dohledá k dvojici (id, čas) z modrého dotazu zbývající položky (příjmení, jméno, název závodu, ročník, …). Zelený dotaz je tu od toho, aby tyto výsledky očísloval. Definuje proměnnou *@rownum*, kterou postupně zvyšuje o jedničku a označuje *rank.*

V dotazu je na dvou místech použita agregační funkce *MIN ().*Ve vnitřním (modrém) dotazu najde minimální čas. Ve vnějším (červeném) odstraní duplicity, které vzniknou, pokud plavec zaplave ve více závodech ve stejné disciplíně stejný čas. Agregační funkce nám umožňují seskupit vybrané řádky dotazu a aplikovat na ně určitou aritmetickou operaci. Ve spojení s agregačními funkcemi nesmíme zapomenout na konstrukci GROUP BY, kde vyjmenujeme neagregované položky [12]. Pro větší názornost je uvedena ukázka kousku výsledných dat odpovídajících disciplíně sto metrů motýlek na krátkém bazénu.

Pořadí	Čas	Příjmení	Jméno	Ročník	Datum	Závod
1	00:57,74	Vencel	Jan	1992	10. 12. 2015	Zimni MČR dorostu a dospělých v Plzni
2	01:00,14	Vencel	Michal	1996	28. 11. 2015	Vánoční cena Zlína - 10. kolo ČP
3	01:04,40	Kučera	Milan	2003	23. 04. 2016	Cena města Blanska
4	01:07,07	Špaček	Dominik	1998	06. 10. 2012	Velká cena města Trutnova
5	01:10,00	Chemčuk	Tomáš	1998	23. 04. 2016	Cena města Blanska
6	01:10,70	Švarc	Radim	2002	23. 04. 2016	Cena města Blanska
7	01:11,0	Kopřiva	Michael	1992	22. 10. 2011	Velká cena města Vysokého Mýta
8	01:11,16	Přikryl	Adam	1997	29. 03. 2014	Jarni cena Boskovic
9	01:12,40	Odehnal	Pavel	1999	09. 05. 2015	Cena města Blanska
10	01:12,72	Musil	Milan	2002	16. 04. 2016	Cena Třebovské lokomotivy

Tabulka 6.1: Žebříček prvních deseti plavců v disciplíně 100m motýlek na 25m bazénu

6.4 Zobrazení vybraných dat

Jakmile máme požadovaná data vybrána s databáze, měli bychom je přehledným a názorným způsobem zobrazit uživateli. V práci je implementována prezentace většího objemu dat pomocí tabulek a grafů. Všechna implementace tohoto typu probíhá výhradně v příslušné Nette šabloně s pomocí jazyka Latte. Na rozdíl od předchozích výběrů, které patří do modelu.

6.4.1 Tabulky

Tabulky pro zobrazení vybraných plavců, závodů nebo časů mají konkrétní výšku a CSS vlastnost *overflow* nastavenou na hodnotu *auto*. To způsobí schovávání obsahu, který při rolování

přesahuje rámec tabulky. Například obsah tabulky plavci je dynamicky naplňován velmi podobnou konstrukcí, jako je ta následující.

```
{foreach $swimmers as $swimmer}
	<tr id="{$swimmer->id}">
		<td>{if ($swimmer->sex === 'M')}Muž{else}Žena{/if}</td>
		<td>{$swimmer->surname}</td>
		<td>{$swimmer->firstname}</td>
		<td>{$swimmer->year}</td>
		<td>{if ($swimmer->active === '1')}Ano{else}Ne{/if}</td>
		<td><a n:href="zobraz! $swimmer->id" class="ajax" click="showDet()">
			<span class="glyphicon glyphicon-user"></span>
		    </a>
		</td>
	</tr>
{/foreach}
```

Objekt *$swimmers* získáme z příslušného presenteru, který se postará o jeho naplnění voláním odpovídajícího modelu. Následně využijeme Latte makra *foreach* [13], které představuje známý cyklus. Jako podmínku zase můžeme použít makro *if*. Výše uvedený kód by samozřejmě šel napsat i v čistém PHP, ale v Latte je to v mnoha ohledech (pohodlnost, přehlednost, součást Nette) výhodnější, tak proč ho nepoužít.

Poslední datová buňka v našem příkladu obsahuje odkaz, který otevře modální okno detailu plavce. Hodnotou atributu n:href není URL, ale akce presenteru, které bude jako parametr předána hodnota *$swimmer->id*. Tato akce za použití AJAXu pošle do šablony informace o konkrétním plavci. Ty se zobrazí v modálním okně aktivovaném funkcí *showDet()*. Element *span* obsahuje ikonu, kterou poskytuje Bootstrap framework.

6.4.2 AJAX

Často využívanou technologií je AJAX. S jeho pomocí je naprogramováno zobrazování obsahu tabulek po uplatnění výběrových nebo třídících kritérií. Díky tomu, mimo jiné, zůstanou tato kritéria v tom stavu, jak je uživatel aplikoval a nenastaví se do stavu defaultního. Dále je AJAX využíván např. při zobrazování rekordů u jednotlivých disciplín nebo výsledku výpočtu bodové kalkulačky. Vzhledem k tomu, že Nette neobsahuje implementaci AJAX požadavků na straně klienta, stáhneme si veřejně dostupnou knihovnu *nette.ajax.js*. Knihovna automaticky „zAJAXuje“ všechny odkazy a formuláře s CSS třídou *ajax* [14]. V následujícím příkladu se jedná o tlačítko k odeslání formuláře výběrových a třídících kritérií.

```
<button n:name=show type="submit" class="ajax">Aplikovat</button>
```

V příslušné akci presenteru vložíme kód, který zjistí, jestli se jedná o AJAXový požadavek. Pokud ne, provede přesměrování, jinak provede zneplatnění konkrétního snippetu (neplatné se přenášejí).

```
public function selectingOptionFormSucceeded($form)
{
	//logika na základě přijatých vyplněných položek formuláře
	...
	$this->template->swimmers = //naplníme z databáze


	if (!$this->isAjax())
		$this->redirect('this');
	else{
		$this->invalidateControl('tabulkaPlavci');
	}
}
```

Části, které chceme, aby AJAX překresloval, označíme v šabloně párovým makrem {snippet} a {/snippet}, které pojmenujeme. Zde použijeme dříve uvedený kód pro naplnění tabulky plavců.

```
{snippet tabulkaPlavci}
	{foreach $swimmers as $swimmer}
		...
	{/foreach}
{/snippet}
```

Vše dohromady nám zajistí okamžité překreslení pouze tabulky plavců po odeslání výběrových a třídících kritérií.

6.4.3 Grafy

Pro lepší názornost jsou určitá data zobrazována ve formě grafů. K tomu je použit nástroj Google Charts, který je představen v kapitole 3.5.1. Každému plavci, který má dostatek záznamů v databázi, je možné vygenerovat grafy týkající se např. vývoje jeho výkonosti, počtu absolvovaných závodů a startů, četnosti jednotlivých disciplín, atd.

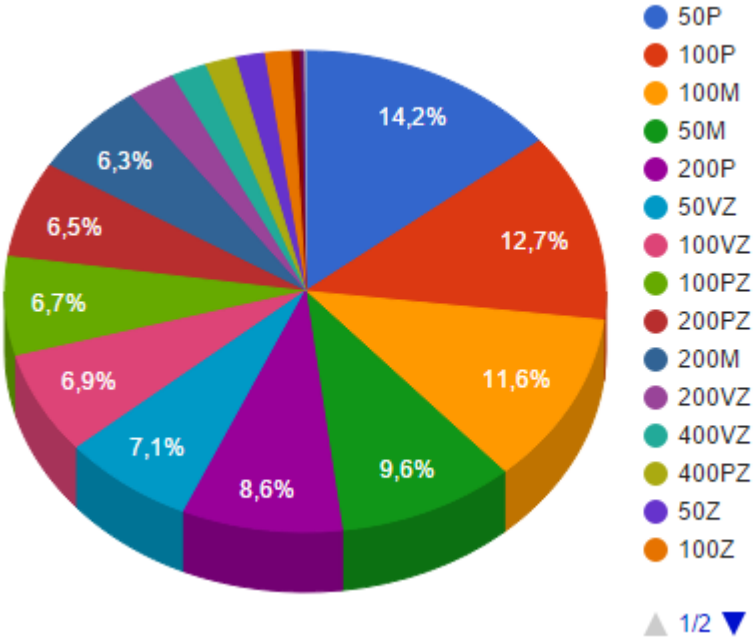
Než začneme samotné grafy programovat, musíme v hlavičce HTML dokumentu načíst AJAX API, která je uvedena v dokumentaci. Jako příklad si uvedeme tvorbu grafu na četnost jednotlivých disciplín.

```
<script type="text/javascript">

//načteme Visualization API corechart balíček a můžeme nastavit jazyk
//grafu
google.charts.load('current', {'language': 'cs',
'packages':['corechart']});

//nastavíme callback, který se spustí až když je Visualization API načteno
google.charts.setOnLoadCallback(drawStrokeCountChart);

//funkce vytvářející graf
function drawStrokeCountChart()
{
      //vytvoření proměnné data (instance třídy DataTable)
      var data = new google.visualization.DataTable();

      //vytvoření sloupců tabulky
      data.addColumn('string', 'Styl');
      data.addColumn('number', 'Četnost');

      //přidávání jednotlivých řádků v cyklu procházením proměnné
      //$stroke_count (poslána presenterem), která obsahuje pole disciplín
      //a odpovídající počet startů v ní
      {foreach $stroke_count as $stroke => $count}
            data.addRow([{$stroke}, {$count}]);
      {/foreach}

      //naplnění proměnné options požadovanými nastaveními
      var options = {
            height: 350,
            is3D: true,
            width: 546,
            chartArea: {
                  left:60,top:20,width:'90%',height:'90%'
            }
      };

      //instanciace a vykreslení grafu
      var chart = new
google.visualization.PieChart(document.getElementById('stroke-count'));
      chart.draw(data, options);
}
</script>
```

Tento skript je součástí šablony. V ní ještě musíme vytvořit div element, který bude obsahovat tento graf.

```
<div id="stroke-count"></div>
```

Na tomto místě se následně vykreslí nějak takto vypadající interaktivní graf.



Obrázek 6.2: Koláčový graf četnosti jednotlivých disciplín

Některé grafy umožnují uživateli zvolit si disciplínu nebo rok. Tyto grafy se vytvářejí a vykreslují s využitím AJAXu. To řeší problém časově náročného generování všech grafů na stránce jenom kvůli změně disciplíny u jednoho grafu.

6.5 Import dat

Aplikace umožnuje importovat dva typy dat. Prvním typem jsou výsledky ze závodů uložené na serveru Českého svazu plaveckých sportů (ČSPS) a druhým typem jsou základní časy ze serveru Mezinárodní plavecké federace (FINA).

6.5.1 Výsledky

Pořadatelé registrovaných závodů by měli po jejich skončení nahrávat na stránky ČSPS výsledky. A to jak v PDF, tak TXT formátu. Nás pro účely této aplikace zajímá především TXT formát, který vypadá takto.

```
160312
VELKÁ CENA HRADCE KRÁLOVÉ A KRÁLOVÉHRADECKÉHO KRAJE V PLAVÁNÍ - 2.ročník
12.03.2016
Hradec Králové
50
13.03.2016 08:13
C

27;"50P";1;"CHOCOVÁ";"Petra";"1986";"Z";"PKČL";00:32,20;767;6;4;13.03.2016
;"";"";"";"";"";" ";" ";" ";" "
27;"50P";2;"JORDOVÁ";"Eva";"1989";"Z";"KomBr";00:32,69;733;6;5;13.03.2016;
"";"";"";"";"";" ";" ";" ";" "
27;"50P";3;"ZAMAZALOVÁ";"Veronika";"1997";"Z";"ASKBl";00:34,08;647;6;3;13.
03.2016;"";"";"";"";"";" ";" ";" ";" "
27;"50P";4;"CHRÁPAVÁ";"Edita";"1999";"Z";"FaBr";00:34,61;618;6;6;13.03.201
6;"";"";"";"";"";" ";" ";" ";" "
27;"50P";5;"JELÍNKOVÁ";"Barbora";"1998";"Z";"TJZn";00:35,13;591;5;7;13.03.
2016;"";"";"";"";"";" ";" ";" ";" "
27;"50P";6;"GERŽOVÁ";"Dominika";"1999";"Z";"TJKr";00:35,19;588;6;2;13.03.2
016;"";"";"";"";"";" ";" ";" ";" "
```
atd.

Na prvních řádcích je uvedena hlavička, která obsahuje identifikační číslo dokumentu, název závodů, datum závodů, místo konání, typ bazénu a datum a čas vystavení výsledků. Následují vlastní výsledky. Platí, že co řádek, to jeden výkon konkrétního plavce. Každý řádek má většinou 22 polí. Existují totiž programy pro administraci plaveckých závodů, které generují výsledky s jedním polem navíc.

Každý řádek obsahuje tato pole – číslo disciplíny v programu závodů, zkratka disciplíny, umístění, příjmení, jméno, ročník, pohlaví, plavecký klub, čas, body, číslo rozplavby, číslo dráhy, datum. Zbylá pole se využívají u štafet. Výsledky štafet aplikace při importu ignoruje, protože dosavadní struktura databáze na tato data není uzpůsobena. To však neplatí pro První Úsek Štafety, který se bere jako klasický start jednotlivce a importuje se s příznakem „PÚŠ“.

Import výsledků může provádět buď administrátor, nebo hlavní trenér. Trenérům je import výsledků zakázán, aby se zamezilo importu stejných výsledků vícekrát z důvodu nedostatečné komunikace mezi trenéry. Aplikace není schopna rozpoznat, že se jedná o stejné výsledky, pokud jsou nahrávány do prázdného závodu (závodu neodpovídají žádné záznamy v tabulce časů).

Pověřená osoba tedy nejprve musí vytvořit prázdný závod a identifikovat ho výstižným identifikátorem. Potom nahraje stažené textové výsledky. Aplikace nejdříve vybere jenom ty řádky, které se týkají oddílu ASK Blansko. Využívá k tomu osmé pole řádku, které určuje oddíl. U vybraného řádku pak provede různé kontroly týkající se např. správnosti zkratky disciplíny, existenci plavce a disciplíny v databázi nebo smysluplnosti času. Také provede nutnou konverzi češtiny z kódování Windows-1250 do UTF-8. Pokud je vše bez chyby, tak aplikace spočítá k času body podle vzorce:

$$P = 1000 * (B/T)^3$$

Kde B je aktuální základní čas z tabulky základních časů a T je zaplavaný čas. Obě hodnoty časů jsou v sekundách. Pokud je v některé položce chyba, tak je konkatenována se znaky „E#“, které později poslouží k upozornění uživatele na chybu. Po provedení popsaných akcí se načtené položky z řádku uloží do pole a přistoupí se k dalšímu řádku.

Jakmile je hotovo a pole naplněno, tak se vše uloží do pomocné tabulky v databázi. Její obsah se zobrazí uživateli, který je uvědomen, zda jsou načtené položky bez chyby nebo nikoli. Chybné položky jsou označeny zmíněnými znaky „E#“ a obarveny červeně. Uživatel je v tomto případě nucen opravit chyby ve vstupním textovém souboru a provést jeho načtení znovu. Pokud je vše v pořádku, je mu umožněn import načtených hodnot do tabulky časů. To se provede jednoduchým překopírováním obsahu pomocné tabulky do tabulky časů. Pomocná tabulka má z tohoto důvodu stejnou strukturu jako tabulka časů.

```
$this->database->query('INSERT INTO sm_time(
                                id_event,
                                id_stroke,
                                id_swimmer,
                                time,
                                rank,
                                point
                            )SELECT * FROM sm_import');
$this->database->query('DELETE FROM sm_import');
$this->flashMessage('Data byla importována.', 'success');
```

Následuje výmaz pomocné tabulky a oznámení uživateli o úspěchu.

Díky tomuto aparátu importu výsledů rychle a jednoduše zajistíme aktuálnost dat v databázi a na nich závislých funkcí v aplikaci.

6.5.2 Základní časy

Základní čas (base time) je stanoven pro každou disciplínu a udává se v sekundách. Samozřejmě závisí na typu bazénu a pohlaví. Nové základní časy jsou vydávány organizací FINA každý rok. Jsou to vlastně hodnoty světových rekordů pro ten daný rok. Pro krátký bazén jsou základní časy definovány 31. srpna a pro dlouhý 31. prosince.

Z výše uvedeného vztahu pro výpočet bodů vyplývá potřeba mít základní časy uloženy v nějaké tabulce v databázi, abychom s nimi mohli jednoduše pracovat. Proto vznikla tabulka základních časů, která uchovává veškerou jejich historii. Základní časy staršího data využívá aplikace např. ve funkci bodová kalkulačka, kde umožňuje uživateli volbu roku těchto časů. Nahrávat základní časy může pouze administrátor a očekává se to jednou až dvakrát do roka. Soubor se základními časy

je dostupný na stránkách FINA pouze ve formátu XLS. Následuje ukázka několika prvních řádků z takového souboru.

FINA Points Table - Base times

YEAR	COURSE	GENDER	RELAY-COUNT	DISTANCE	STROKE	BASETIME	BASETIME IN SECONDS
2008	SCM	M	1	50	FREE	21.10	21,10
2008	SCM	M	1	100	FREE	46.45	46,45
2008	SCM	M	1	200	FREE	1:42.47	102,47
2008	SCM	M	1	400	FREE	3:37.85	217,85
2008	SCM	M	1	800	FREE	7:36.85	456,85
2008	SCM	M	1	1500	FREE	14:27.07	867,07
2008	SCM	M	1	50	BACK	23.46	23,46
2008	SCM	M	1	100	BACK	50.47	50,47
2008	SCM	M	1	200	BACK	1:50.18	110,18

Z důvodu pouhé dostupnosti souboru v tomto formátu je nutné provést v programu MS Excel export do souboru v TXT formátu. Takto vzniklý soubor už lze přes webové rozhraní aplikace naimportovat do tabulky základních časů. Proces importu je podobný, jako je tomu v případě importu výsledků. Avšak nepoužívá se zde pomocná tabulka a záznamy jsou přímo vkládány do tabulky základních časů.

6.6 Export dat

Aplikace poskytuje uživateli možnost exportu sestav a žebříčků do souborů ve formátech PDF a CSV. Formát PDF je vhodný přímo pro tisk. Použití CSV formátu se předpokládá při další analýze dat např. v programu MS Excel.

6.6.1 PDF

Pří vytváření PDF dokumentů se využívá služeb knihovny mPDF, která je volně dostupná na internetu. Obsah staženého archivu rozbalíme do složky v rootu naší aplikace jak je vidět v adresářové struktuře v kapitole 6.2. Teď už se můžeme přepnout do příslušného modelu a pustit se do tvorby PDF. Nejprve je nutné načíst třídu mPDF.

```
\Composer\Autoload\includeFile(dirname(__DIR__).'/../mpdf60/mpdf.php');
use mPDF;
```

potom už lze vytvořit objekt jako instanci třídy mPDF s určitým nastavením – papír formátu A4, písmo Times velikosti 9pt, okraje.

```
$mpdf = new mPDF('', 'A4', 9, 'times', 15, 15, 25, 15, 8, 8);
```

Obrázek do záhlaví vložíme tímto způsobem.

```
$mpdf->SetHTMLHeader('<div align="center"><img src="images/logo_small.png"
width="80" height="50"/></div>');
```

Ještě nastavíme patičku, která bude obsahovat datum tisku, autora, adresu aplikace a číslo stránky.

```
$mpdf->setFooter(date("j. n. Y")."| Jan Vencel © www.isambk.cz| Strana
{PAGENO}");
```

A nyní už zbývá jenom vložit obsah a poslat výsledek na výstup.

```
$mpdf->WriteHTML($content);
$mpdf->Output($nazev.'.pdf', 'I');
```

Proměnná *$název* obsahuje název PDF dokumentu podle aktuální potřeby a je konkatenována s příponou .pdf. Druhý parametr metody *Output()* zajistí, že dokument je poslán prohlížeči, pokud je dostupný plugin pro podporu PDF v prohlížeči, jinak vynutí jeho stažení. Proměnnou *$content* musíme samozřejmě nejdříve naplnit HTML kódem, než ji předáme metodě *WriteHTML()*. Tvorba takového HTML kódu se provádí v aplikaci dvěma způsoby. První způsob je ten, kdy uživatel exportuje data z tabulek plavců, závodů nebo časů a je o něco složitější u důvodu možnosti volby sloupců pro export. Druhý způsob se uplatňuje při exportu žebříčků. Obecně se HTML kód skládá z nadpisů různé úrovně a tabulky, která je v cyklu plněna daty závislými na typu exportovaného žebříčku. Způsob plnění řádků tabulky je vidět na následujícím příkladu.

```
//$data_array je pole řádků
//$content obsahuje vytvářený HTML dokument
foreach ($data_array as $row) {
      $content .= '<tr>';
      foreach ($row as $field) {
            $content .= '<td>'.$field.'</td>';
      }
      $content .= '</tr>';
}
```

Výsledný vzhled PDF dokumentu můžeme stylizovat klasicky pomocí CSS.

6.6.2 CSV

CSV, nebo-li hodnoty oddělené čárkami (Comma-separated values) je typ textového souboru určený pro výměnu tabulkových dat. Obsahuje řádky a v nich jednotlivé položky oddělené oddělovačem (čárka, středník). V této aplikace je použit jako oddělovač středník z důvodu přítomnosti čárek v desetinných číslech.

Pro tvorbu CSV dokumentu není potřeba žádná externí knihovna, jako tomu bylo v případě PDF, ale vystačíme si pouze s funkcemi standardně zabudovaných do PHP. A to zejména s funkcí *fputcsv()*. Tu voláme v cyklu a předáváme jí jednotlivé řádky.

```
foreach ( $data as $row ) {
      //zformátujeme datum do požadovaného tvaru
      if($row['date']){
            $row['date'] = $row['date']->format('Y-m-d');
      }
      fputcsv($fh, $row, ";", '"');
}
```

Funkce ukládá CSV data na místo v paměti popsané ukazatelem *$fh*. Tato data si následně vyzvedneme a pomocí funkce *stream_get_contents()* z nich vytvoříme string formát.

```
$csv = stream_get_contents($fh);
```

V proměnné *$csv* nyní máme data v požadované CSV formátu. Zbývá jenom přinutit prohlížeč, aby provedl stažení souboru do uživatelova file systému. To provedeme následovně.

```
//nesmíme zapomenout zahrnout třídu TextResponse
use Nette\Application\Responses\TextResponse;

$httpResponse = $this->presenter->getHttpResponse();
$httpResponse->setContentType('text/plain');
$httpResponse->setHeader('Content-Disposition', 'attachment;
filename="'.$nazev.'"');
$httpResponse->setHeader('Content-Length', strlen($csv));
$this->presenter->sendResponse(new TextResponse($csv));
```

6.7 Přepočet bodů

Už jsme si říkali, že organizace FINA vydává každý rok nové hodnoty základních časů, které tato aplikace využívá k výpočtu bodových hodnot jednotlivých výkonů. Vzhledem k této skutečnosti je potřeba mít v aplikaci možnost přepočítat bodové hodnoty všech výkonů v tabulce časů a mít tak záznamy v aktuálním stavu. V současné době má tabulka přibližně čtrnácti tisíc řádků. Přepočet je proto relativně zdlouhavý proces, který může trvat až několik minut. Za účelem zkrácení této doby na minimum byla aplikována určitá nastavení a vypnuty některé kontroly.

```
// Vypnutí automatických COMMITů
$this->database->query('SET autocommit=0');
//vypnutí kontrol unikátnosti omezení na sekundárních klíčích
$this->database->query('SET unique_checks=0');
//zrušení omezení (constraints)
$this->database->query('SET foreign_key_checks=0');
//zahájení transakce
$this->database->query('START TRANSACTION');

// Postupně procházeni všech řádku tabulky
$tabulka = $this->database->table($this->tableName);
foreach ($tabulka as $radek) {
      ...//volání funkce na přepočet bodů
}
// provedeme COMMIT za celou transakcí, nikoli po updatu jednoho řádku
$this->database->query('COMMIT');
//vrátíme zpátky původní nastavení
$this->database->query('SET foreign_key_checks=1');
$this->database->query('SET unique_checks=1');
$this->database->query('SET autocommit=1');
```

Transakce s uvedeným nastavením trvá už příjemných patnáct vteřin.

6.8 Uživatelé

Aplikace rozlišuje čtyři typy uživatelů – administrátor, hlavní trenér, trenér a běžný uživatel. První tři jmenovaní se musí do aplikace přihlašovat. Jejich přihlašovací údaje, název role a další doplňující informace jsou uloženy ve zvláštní tabulce v databázi. Nyní si ukážeme, jak probíhá autentifikace.

```
//vybereme řádek z tabulky uživatelů podle uživatelského jména zadaného
//uživatelem
$row = $this->database->table(self::TABLE_NAME)->where(self::COLUMN_NAME,
$username)->fetch();

//pokud se nám řádek nepodařilo vybrat, znamená to špatně zadané
//uživatelské jméno
if (!$row) {
     throw new Nette\Security\AuthenticationException('Chybné uživatelské
     jméno.', self::IDENTITY_NOT_FOUND);
```

```
//jestliže uživatelské jméno odpovídá, porovnáme zadané heslo s hashovaným
//heslem v tabulce
}elseif (!Passwords::verify($password, $row[self::COLUMN_PASSWORD_HASH])){
      throw new Nette\Security\AuthenticationException('Chybné heslo.',
      self::INVALID_CREDENTIAL);
}

//převedeme do pole a odstraníme položku s heslem
$arr = $row->toArray();
unset($arr[self::COLUMN_PASSWORD_HASH]);

//vrátíme identitu uživatele
return new Nette\Security\Identity($row[self::COLUMN_ID],
$row[self::COLUMN_ROLE], $arr);
```

Získaná identita je objekt představující soubor informací o uživateli. Díky ní můžeme později v aplikaci přistupovat k uživatelovu identifikátoru nebo jménu. Po úspěšné autentizaci přichází na řadu autorizace, která nám ověří, že uživatel má k určité operaci oprávnění [15]. V aplikaci jsou použity jako autorizační kritéria podmínka přihlášení a příslušnost určité roli. Například takto je podmíněno zobrazení formuláře pro ruční vkládání plavců do tabulky.

```
{if $user->loggedIn}
      {if ($user->isInRole('admin') or $user->isInRole('head_coach'))}
      ...
      {/if}
{/if}
```

Proměnná *$user* je objekt třídy *Nette\Security\User*.

6.9 Uživatelské rozhraní

Při vytváření uživatelského rozhraní byl kladen důraz na jednoduchost a intuitivnost. Celá aplikace je laděná do modra a pozadí vytváří podvodní fotka s plavcem. Obrázek byl stažen ze stránek *www.pixabay.com,* kde jsou nabízeny fotky s licencí Creative Commons (CC), která umožňuje jejich volnou distribuci. Navigaci v aplikaci zajišťuje vodorovná nabídka v horní části. Její obsah se mění podle toho, v jakém stavu se aplikace nachází (uživatel přihlášen nebo nepřihlášen) nebo jaká práva má přihlášený uživatel (jestli má přístup k importu výsledků nebo nikoli). Úvodní stranu tvoří krátké představení aplikace spolu s logem oddílu, které je zároveň logem aplikace. Jednotlivé stránky nabídky se skládají z mírně transparentních panelů, které se v některých případech dají sbalit. Některé funkce aplikace (přihlášení, detail plavce, editace detailů o plavci) byly

zprostředkovány pomocí modálního okna. Nepovolené nebo naopak úspěšné operace aplikace oznamuje uživateli příslušně zbarvenými zprávami ve stavovém panelu aplikace. Některé odkazy, které jsou z důvodu nedostatku prostoru zastoupeny pouze ikonou, jsou opatřeny tzv. tooltipem. Ten zajistí zobrazení popisu odkazu, když přes něho uživatel přejede myší.

Celková vizáž webu byla inspirována praktikami tzv. „flat designu“. Flat design je aktuálním trendem při tvorbě nejen internetových stránek a aplikací, ale i různých log či ikon. Hlavními rysy jsou jednoduchost, jednolitost a nízká variabilita barev, kvalitní typografické zpracování nebo větší fonty bez efektů a stínů. Příkladem mohou být nové verze operačního systému MS Windows (MS Windows 8 a 10).

Většina výše zmíněných funkcí je implementována s využitím Bootstrap frameworku. Díky tomuto frameworku je aplikace schopná přizpůsobit se menším displejům jako jsou na tabletech a mobilech. Když dodržujeme stanovený „grid systém“ o dvanácti sloupcích, Bootstrap nám automaticky přizpůsobuje elementy stránky velikosti displeje. Někdy bylo ovšem také nutné použít modulu Media Queries, který umožní aplikovat jen určitou část CSS deklarací podle určitého rozlišení displeje v podmínce. Media Queries lze vložit přímo do CSS dokumentu a může vypadat například takto.

```
@media only screen and (max-width: 1199px) and (min-width: 1100px){

	#header_wrapper{
		width: 100%;
		margin: 0px;
	}
}
```

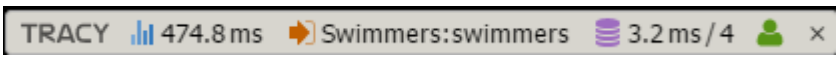
Uvedený příklad nám zajistí nastavení uvedených vlastností elementu *#header_wrapper* v případě, že se šířka viewportu (prostoru pro stránku v okně prohlížeče) dostane mezi hodnoty 1199 a 1100 pixelů.

7 Testování a nasazení systému

7.1 Testování

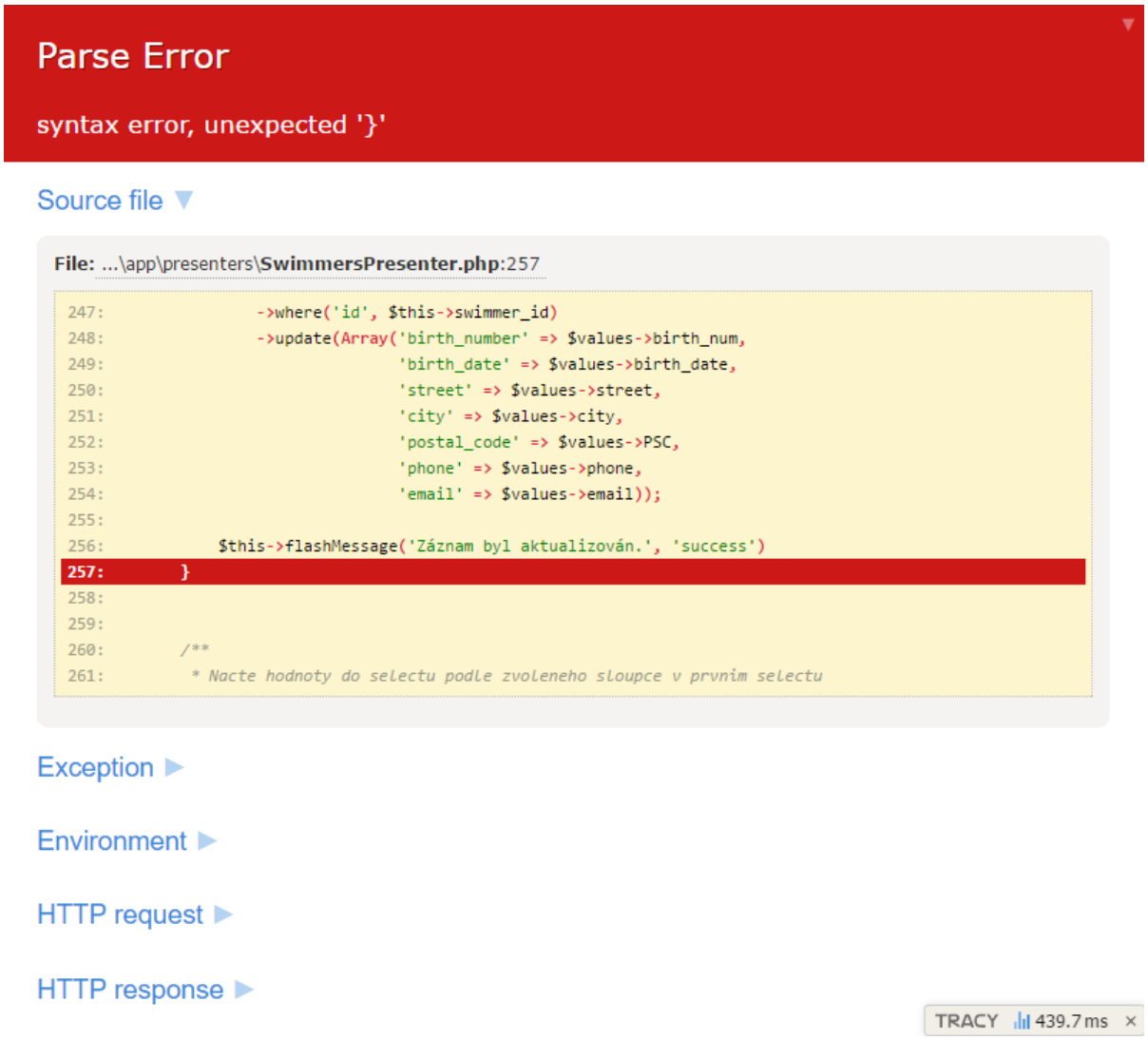
Testování aplikace probíhalo z největší části v průběhu samotného vývoje. Po implementování nové funkce byla kompletně otestována její funkcionalita. Po dokončení aplikace byla otestována funkcionalita těchto funkcí jako celku. Na vstupy funkcím byly posílány takové vstupy, jaké je schopen vyprodukovat tradiční i netradiční uživatel. Na základě výsledků tohoto testování byly funkce nastaveny tak, aby na chybné vstupy reagovaly uživatelsky srozumitelnou a pochopitelnou odpovědí (zpráva).

Nette Framework nabízí pro potřeby testování vhodný nástroj, který byl ve velké míře také využíván. Tím nástrojem je knihovna zvaná Tracy nebo také Laděnka [16]. Přítomnost Tracy můžeme na první pohled zaznamenat v pravém dolním rohu naší stránky. Tam se nachází tzv. Debugger Bar, kde hned vidíme, kolik milisekund trvalo načtení stránky, v jaké akci kterého presenteru se stránka nachází, komunikaci s databází nebo jestli je přihlášen uživatel.



Obrázek 7.1: Debbuger Bar

Tracy nahrazuje nepřehledné chybová hlášení jazyka PHP, který je vypisuje přímo do zdrojového kódu stránky. Chyba nebo výjimka se v prohlížeči zobrazí v takovéto podobě.



Obrázek 7.2: Zpráva o chybě

Hned vidíme typ chyby s její podrobnější specifikací. Také se nám nabízí pohled do zdrojového souboru, na kterém řádku chyba vznikla. K dalším podrobnostem, jako jsou například hodnoty proměnných, se dostaneme po pár kliknutích. Takový typ chyby, co je na obrázku, byl často odhalen ještě před spuštěním aplikace díky vývojovému prostředí NetBeans, které provádí syntaktickou analýzu psaného textu.

Podrobné ladící informace jsou velmi důležité pro vývojáře ale pro uživatele už nikoli. Proto se Tracy při spuštění aplikace na produkčním serveru automaticky deaktivuje a všechny chyby se od té chvíle zaznamenávají do logovacího souboru *log/error.log*. Zajímavou funkcí Tracy je možnost zaslat email vývojáři, když přibude do logovacího souboru nový záznam.

7.2 Nasazení

Po důkladném otestování byl systém připraven na nasazení do provozu. Nejprve byla přes společnost Wedos registrována doména *www.isambk.cz*. Následně byl u té samé společnosti zřízen také hosting. Potom bylo nutné vytvořit databázi. Díky dříve vytvořeným vytvářecím skriptů a datům v textové podobě nekladlo vytváření větší odpor. Po úspěšném přenesení zdrojových souborů na server zbývalo už jenom oznámit existenci webové aplikace příslušníkům plaveckého oddílu. To bylo provedeno prostřednictvím internetových a facebookových stránek oddílu.

8 Sledování návštěvnosti

Následující tabulka vznikla z dat naměřených službou Google Analytics. Časové období měření bylo od 28. února 2016 do 5. května 2016.

Přehled	
Celkový počet návštěv během daného časového období	234
Uživatelé, kteří měli během daného časového období aspoň jednu návštěvu	138
Celkový počet zobrazených stránek (i opakovaná zobrazení)	1224
Průměrný počet stránek zobrazených během návštěvy	5,23
Průměrná doba trvání návštěvy	6 min 12s
Procentuální podíl návštěv, kdy uživatel opustil web už na vstupní stránce	25,27%
Top 3 nejnavštěvovanějších stránek	1. / (24,28%) 2. /swimmer (17,00%) 3. /event (7,95%)

Noví vs. vracející se uživatelé	
Procento nových uživatelů (navštívili web pouze jednou)	53,20%
Procento vracejících se uživatelů	46,80%

Zařízení	
Procento přístupů do aplikace z desktopu	74,73%
Procento přístupů do aplikace z mobilu	23,12%
Procento přístupů do aplikace z tabletu	2,15%

V sekci Tok uživatelů bylo zjištěno, že přibližně 50% uživatelů, kteří vstoupí do aplikace přes hlavní stránku, míří na Přehled plavců. Odtud asi 75% uživatelů pokračuje na statistiku konkrétního plavce.

9 Závěr

Záměrem této bakalářské práce bylo vytvořit informační systém plaveckého oddílu ASK Blansko. Mohu konstatovat, že stanovené cíle se podařilo splnit. Jedinou funkcionalitou, která nebyla kompletně implementována podle původního záměru, je systém vytváření přihlášek na závody. A to z toho důvodu, že oddíly, včetně toho blanenského, využívají již existující systém na stránkách Českého svazu plaveckých sportů. Proto by byla tato funkce v aplikaci zbytečná a je tak pouze nahrazena odkazem na stránky ČSPS. Systém je v současné době plně v provozu a slouží cílové skupině uživatelů.

Na počátku této práce jsme se seznámili se současným stavem databáze oddílu a s cíli aplikace. Uvedena byla také konkrétní metodologie návrhu. Následovalo krátké seznámení s technologiemi využitými při vývoji systému, kterému předcházelo delší samostudium. Funkčnost některých technologií ujasňují obrázky. V kapitole o analýze a návrhu sytému byly stanoveny požadavky, které byly následně promítnuty do diagramů. Konkrétní implementaci daných požadavků se věnuje další kapitola. Detailněji je zde popsána funkcionalita, na které si aplikace z podstatné části zakládá. Tento popis je prokládán komentovanými výňatky ze zdrojového kódu. V závěru práce je představen způsob testování a nasazení systému do běžného provozu. Data týkající se návštěvnosti jsou vyhodnocena v kapitole sledování návštěvnosti.

Aplikace je schopna se velice rychle přizpůsobit jiným oddílům. Jedná se v podstatě jenom o změnu názvu a popisu oddílu na úvodní straně. Avšak musí být splněn důležitý požadavek existence databáze a její správné struktury.

Uživatelé mobilních zařízení a tabletů by neměli mít problém z těchto zařízení aplikaci ovládat. Ta je naprogramována tak, aby se přizpůsobovala různým velikostem displeje a nabízela tak této nemalé skupině uživatelů stejné funkce a možnosti jako mají uživatelé desktopových zařízení.

9.1 Možná vylepšení

Databáze, ze které aplikace vychází, vznikla před deseti lety. V té době se vůbec nepočítalo se zpřístupněním dat veřejnosti. Databázová struktura byla navržena tak, aby její podoba vyhovovala pouze nárokům, které vyžaduje administrace oddílových dat (plavců závodů, časů) zajišťovaná jedním určeným správcem přes desktopový program. Vytvořená internetová aplikace je postavena také na této databázi, z důvodu jejího velkého objemu. I když se aplikace snaží tato data co nejvíce zpřístupnit uživateli, tak svůj administrační charakter nezakryje.

Pokud bychom tedy chtěli vylepšovat, musíme začít databází. Funkce založené na té stávající jsou téměř vyčerpány. To znamená vytvořit tabulky např. pro evidenci přihlášek do oddílu, placení příspěvků, trenérů a jejich tréninkových skupin, docházky plavců nebo tréninkových deníků. Takto

nově vytvořená databáze by poskytovala potenciál pro nové funkce aplikace. Takový systém by pravděpodobně přesáhl rámec bakalářské práce.

Literatura

[1] HRUŠKA, T., KŘIVKA, Z. Informační systémy (IIS, PIS), Pojem informačního systému, Data, Procesy, Transakce. Brno: FIT VUT v Brně, 2012. 17-28 s.

[2] ŠMÍD, V. Životní cyklus informačního systému [online]. 2002 [cit. 2016-02-15], Dostupné na URL: <http://www.fi.muni.cz/~smid/mis-zivcyk.htm>

[3] ZENDULKA, J., RUDOLFOVÁ, I. Databázové systémy IDS. Brno: FIT VUT v Brně, 2006. 33-65 s.

[4] JASON GILMORE, W. Velká kniha PHP 5 a MySQL*: Kompendium znalostí pro začátečníky i profesionály*. Vyd. 3. Brno: Zoner Press, 2011. ISBN 978-80-7413-163-9.

[5] Nette Foundation. MVC aplikace & presentery [online]. 2008 [cit. 2016-03-01], Dostupné na URL: <https://doc.nette.org/cs/2.3/presenters>

[6] VĚTROVSKÁ, P. Úvod do CSS [online]. 2004 [cit. 2016-02-19], Dostupné na URL: <http://www.webtvorba.cz/css/uvod-do-css.html#zaciname>

[7] Google, Inc. Using Google Charts [online]. 2015 [cit. 2016-02-21], Dostupné na URL: <https://developers.google.com/chart/interactive/docs/>

[8] MÁČEL, L., KUŽELA, A., HRUŠKA, T. Internetové aplikace (WAP) V., část AJAX. Brno: FIT VUT v Brně, 2012. 13-18 s.

[9] Git. About [online]. [cit. 2016-02-21], Dostupné na URL: <https://git-scm.com/about>

[10] ČÁPKA, D. MVC architektura [online]. [cit. 2016-02-29], Dostupné na URL: <http://www.itnetwork.cz/navrhove-vzory/mvc-architektura-navrhovy-vzor/>

[11] Nette Foundation. Databáze [online]. 2008 [cit. 2016-03-08], Dostupné na URL: <https://doc.nette.org/cs/2.3/database>

[12] STEPHENS, R., PLEW, R., JONES ARIE, D. Naučte se SQL za 28 dní. Brno: Computer Press, 2010. ISBN 978-80-251-2700-1.

[13] Nette Foundation. Latte makra [online]. 2008 [cit. 2016-03-04], Dostupné na URL: <https://latte.nette.org/cs/macros>

[14] Nette Foundation. AJAX [online]. 2008 [cit. 2016-03-11], Dostupné na URL: <https://doc.nette.org/cs/2.3/ajax>

[15] Nette Foundation. Přihlašování & oprávnění uživatelů [online]. 2008 [cit. 2016-03-20], Dostupné na URL: <https://doc.nette.org/cs/2.3/access-control>

[16] Nette Foundation. Debugování a zpracování chyb [online]. 2008 [cit. 2016-03-18], Dostupné na URL: <https://tracy.nette.org/cs/>

Seznam příloh

Příloha A

Ukázka vzhledu aplikace

Obrázek A.1: Úvodní stránka aplikace

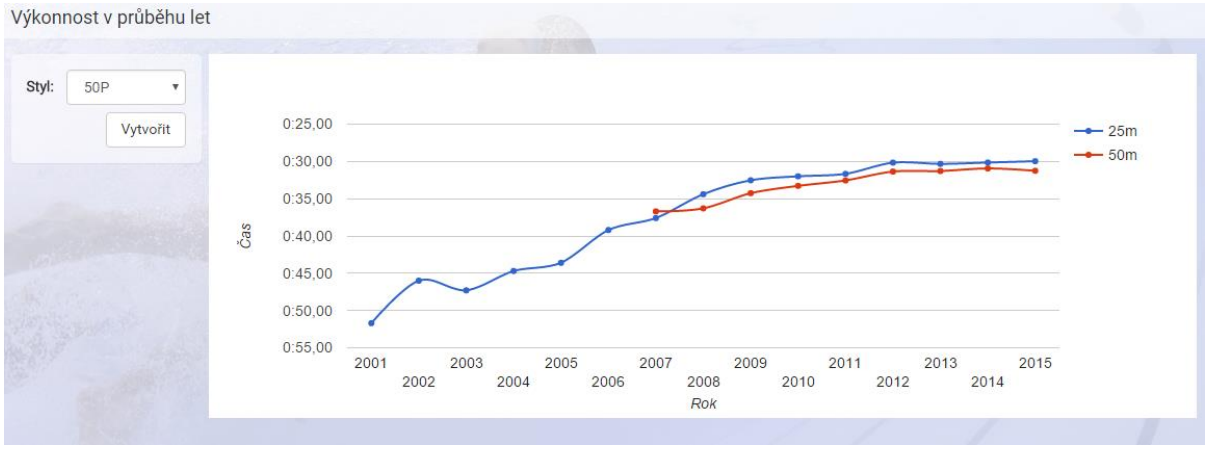
Nejlepší výkony podle bodů - krátký bazén

Muži i Ženy

Pořadí	Body	Příjmení	Jméno	Ročník	Styl	Čas	Datum	Závod
1	685	Zamazalová	Veronika	1997	50P	00:32,67	10. 12. 2015	Zimní MČR dorostu a dospělých v Plzni - 11. kolo ČP
2	646	Vencel	Jan	1992	200P	02:19,35	18. 12. 2014	Zimní mistrovství ČR open v Plzni
3	584	Vencel	Michal	1996	100VZ	00:53,75	10. 12. 2015	Zimní MČR dorostu a dospělých v Plzni - 11. kolo ČP
4	575	Pokorná	Petra	1998	50P	00:34,64	28. 11. 2013	Zimní MČR 2013 dorostů a dospělých, Plzeň - Slovany
5	569	Formánková	Anna	1997	200VZ	02:13,70	29. 05. 2010	Letní krajský přebor žactva v Brně
6	556	Špaček	Dominik	1998	100P	01:07,61	03. 11. 2012	AXIS CUP 2012 - 6.kolo ČP, Jihlava
7	521	Kopřivová	Hana	1998	200VZ	02:17,70	10. 10. 2015	Memoriál Jaroslava Starého
8	507	Grmelová	Viktorie	2000	50P	00:36,12	07. 11. 2015	Axiscup v Jihlavě - 8. kolo ČP
9	484	Pokorná	Bára	2000	50VZ	00:29,60	10. 10. 2015	Memoriál Jaroslava Starého
10	480	Špaček	Michal	1973	100VZ	00:57,38	03. 03. 2012	Jarní cena města Boskovice

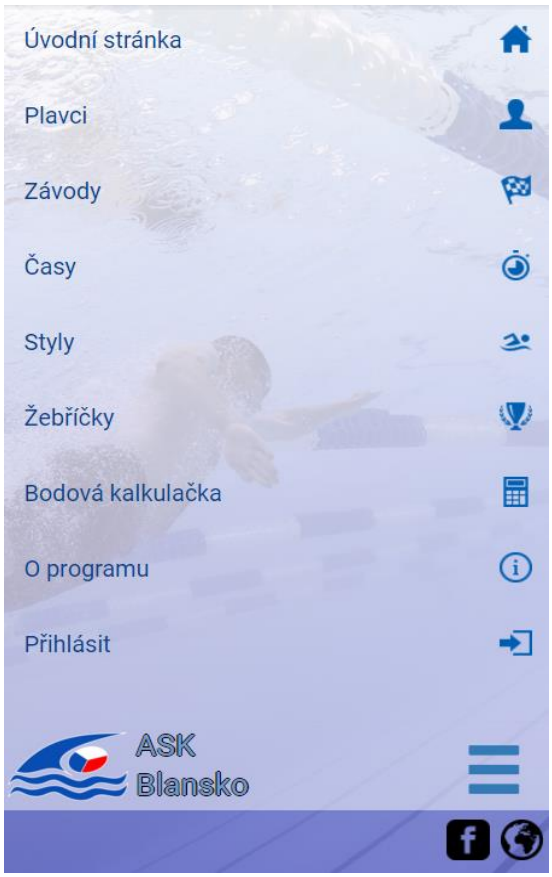
1 2 10 15

Obrázek A.2: Tabulka v sekci Žebříčky – Nejlepší výkony podle bodů

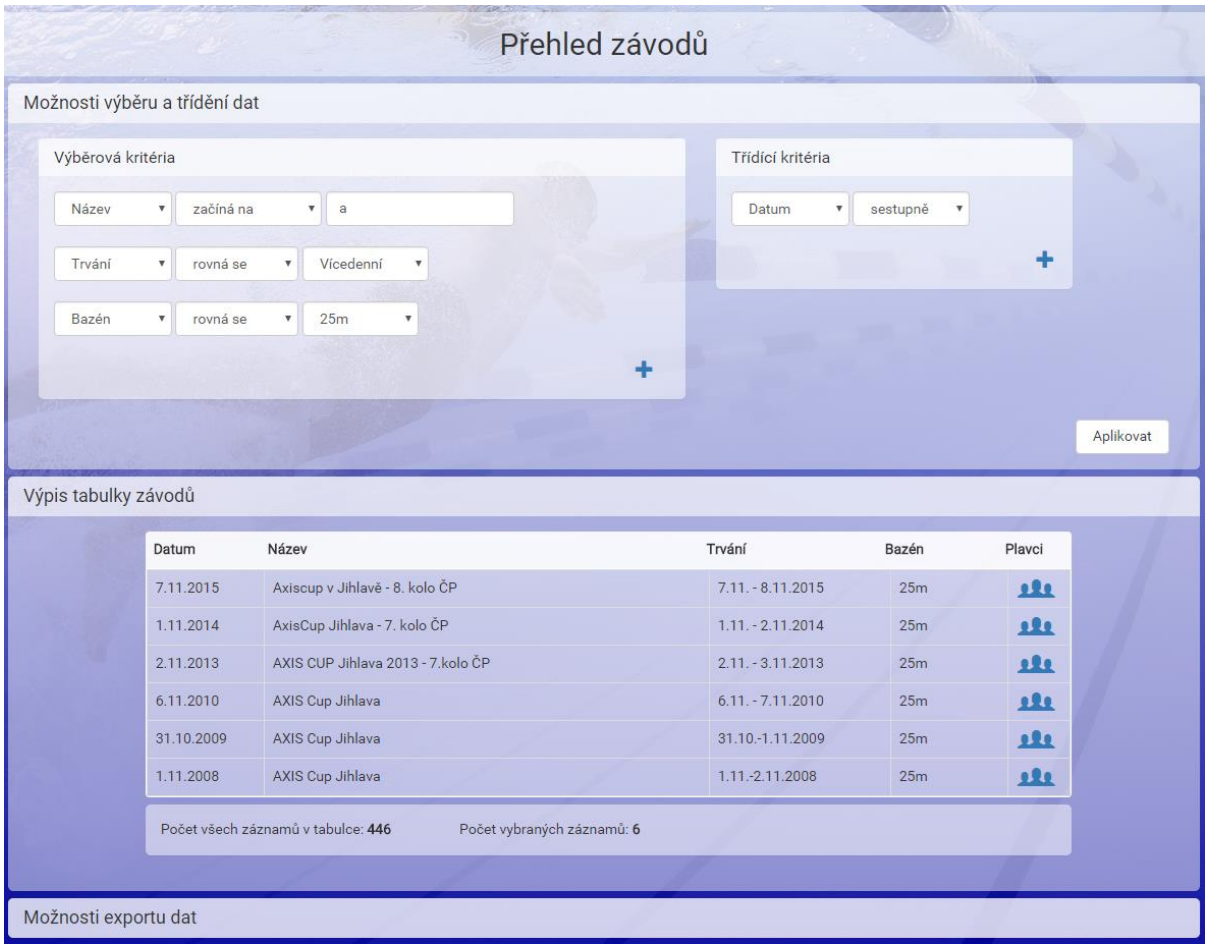


Obrázek A.3: Graf v sekci Statistiky – Výkonnost v průběhu let

Obrázek A.4: Přehled plavců



Obrázek A.5: Vysouvací nabídka v mobilních zařízeních



Obrázek A.6: Konkrétní výběr závodů

Příloha B

Vytvářecí skripty

```
CREATE TABLE sm_users (
  id        INT UNSIGNED NOT NULL AUTO_INCREMENT,
  username  VARCHAR(100) NOT NULL,
  firstname VARCHAR(100) NOT NULL,
  surname   VARCHAR(100) NOT NULL,
  email     VARCHAR(100) NOT NULL,
  phone     VARCHAR(10) NOT NULL,
  password  VARCHAR(200) NOT NULL,
  role      VARCHAR(20) NOT NULL,
  PRIMARY KEY (id))
ENGINE = InnoDB CHARACTER SET utf8 COLLATE utf8_czech_ci;

CREATE TABLE sm_swimmer (
 id         VARCHAR(12) NOT NULL ,
 sex        VARCHAR(1) NOT NULL ,
 surname    VARCHAR(100) NOT NULL ,
 firstname  VARCHAR(100) NOT NULL ,
 year       VARCHAR(4) NOT NULL ,
 active     VARCHAR(1) NOT NULL ,
 race       VARCHAR(1) NOT NULL ,
 count      TINYINT UNSIGNED NOT NULL ,
 PRIMARY KEY  (id) )
 ENGINE = InnoDB CHARACTER SET utf8 COLLATE utf8_czech_ci;

CREATE TABLE sm_swimmer_detail (
 id           VARCHAR(12) NOT NULL,
 birth_number VARCHAR(10) NOT NULL,
 birth_date   VARCHAR(10) NOT NULL ,
 street       VARCHAR(100) NOT NULL,
 city         VARCHAR(100) NOT NULL,
 postal_code  VARCHAR(20) NOT NULL,
 phone        VARCHAR(20) NOT NULL,
 email        VARCHAR(100) NOT NULL ,
 PRIMARY KEY (id),
 CONSTRAINT fk_swimmer_detail FOREIGN KEY (id) REFERENCES sm_swimmer(id) ON DELETE
CASCADE ON UPDATE CASCADE)
ENGINE = InnoDB CHARACTER SET utf8 COLLATE utf8_czech_ci;
```

```
CREATE TABLE sm_event (
 id         VARCHAR(12) NOT NULL ,
 date       DATE NOT NULL ,
 name       VARCHAR(250) NOT NULL ,
 note       VARCHAR(250) NOT NULL ,
 lcm        VARCHAR(1) NOT NULL ,
 flag       VARCHAR(1) NOT NULL ,
 PRIMARY KEY  (id) )
 ENGINE = InnoDB CHARACTER SET utf8 COLLATE utf8_czech_ci;

CREATE TABLE sm_stroke (
 id         VARCHAR(12) NOT NULL ,
 ord        TINYINT UNSIGNED NOT NULL ,
 name       VARCHAR(100) NOT NULL ,
 PRIMARY KEY  (id) )
 ENGINE = InnoDB CHARACTER SET utf8 COLLATE utf8_czech_ci;

CREATE TABLE sm_bt (
 id           INT NOT NULL AUTO_INCREMENT,
 year        VARCHAR(4) NOT NULL ,
 course       VARCHAR(3) NOT NULL ,
 sex          VARCHAR(4) NOT NULL ,
 relay        TINYINT UNSIGNED NOT NULL ,
 stroke       VARCHAR(10) NOT NULL ,
 basetime     VARCHAR(10) NOT NULL ,
 basetime_s   FLOAT(6,2) NOT NULL ,
 PRIMARY KEY (id) )
 ENGINE = InnoDB CHARACTER SET utf8 COLLATE utf8_czech_ci;

CREATE TABLE sm_cz_records (
 id        INT NOT NULL ,
 id_stroke VARCHAR(12) NOT NULL ,
 cz25_m    VARCHAR(10) NOT NULL ,
 cz50_m    VARCHAR(10) NOT NULL ,
 cz25_f    VARCHAR(10) NOT NULL ,
 cz50_f    VARCHAR(10) NOT NULL ,
 PRIMARY KEY (id) ,
 CONSTRAINT fk_stroke_cz_records FOREIGN KEY (id_stroke) REFERENCES sm_stroke(id)
ON DELETE CASCADE ON UPDATE CASCADE)
ENGINE = InnoDB CHARACTER SET utf8 COLLATE utf8_czech_ci;
```

```
CREATE TABLE sm_time (
 id         INT UNSIGNED NOT NULL AUTO_INCREMENT,
 id_event   VARCHAR(12) NOT NULL ,
 id_stroke  VARCHAR(12) NOT NULL ,
 id_swimmer VARCHAR(12) NOT NULL ,
 time       VARCHAR(8) NOT NULL ,
 rank       VARCHAR(40) NOT NULL ,
 point      SMALLINT UNSIGNED NOT NULL ,
 note       VARCHAR(100) NOT NULL ,
 PRIMARY KEY  (id) ,
 CONSTRAINT fk_event FOREIGN KEY (id_event) REFERENCES sm_event(id) ON DELETE
CASCADE ON UPDATE CASCADE,
 CONSTRAINT fk_stroke FOREIGN KEY (id_stroke) REFERENCES sm_stroke(id) ON DELETE
CASCADE ON UPDATE CASCADE,
 CONSTRAINT fk_swimmer FOREIGN KEY (id_swimmer) REFERENCES sm_swimmer(id) ON DELETE
CASCADE ON UPDATE CASCADE)
ENGINE = InnoDB CHARACTER SET utf8 COLLATE utf8_czech_ci;

CREATE TABLE sm_import (
 id_event   VARCHAR(40),
 id_stroke  VARCHAR(40),
 id_swimmer VARCHAR(40),
 time       VARCHAR(40),
 rank       VARCHAR(40),
 point      SMALLINT UNSIGNED)
ENGINE = InnoDB CHARACTER SET utf8 COLLATE utf8_czech_ci;

CREATE TABLE sm_config (
 id         INT UNSIGNED NOT NULL AUTO_INCREMENT,
 shortcut1  VARCHAR(40) NOT NULL ,
 shortcut2  VARCHAR(40) ,
 clubname   VARCHAR(100) NOT NULL ,
 fina_year  VARCHAR(12) NOT NULL ,
 PRIMARY KEY  (id) )
 ENGINE = InnoDB CHARACTER SET utf8 COLLATE utf8_czech_ci;

CREATE TABLE sm_avatar (
 id         INT UNSIGNED NOT NULL AUTO_INCREMENT,
 swimmer_id VARCHAR(12) NOT NULL ,
 img       MEDIUMBLOB NOT NULL ,
 PRIMARY KEY  (id) )
 ENGINE = InnoDB CHARACTER SET utf8 COLLATE utf8_czech_ci;
```

Příloha C

Obsah CD

db/

- data/ #databázová data v době nasazení systému
- create_scripts.txt #vytvářecí skripty

doc/

- technicka_zprava.pdf #tato technická zpráva
- technicka_zprava.docx #tato technická zpráva ve zdrojové podobě

src/

- app/ #modely, presentery, šablony
- www/ #avatary, css a js soubory, obrázky