TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 – Informační technologie

Studijní obor: 1802R007 – Informační technologie

Webová aplikace pro vytváření, správu a sledování tréninkových plánů

Online training diary management application

Bakalářská práce

Autor: **Patrik Drhlík**

Vedoucí práce: Ing. Petr Kretschmer

Konzultant: Ing. Vojtěch Wrnata

V Liberci dne 17. 5. 2013

Fakulta mechatroniky, informatiky a mezioborových studií

Zadání bakalářské práce

Příjmení a jméno studenta	Patrik Drhlík
Zkratka pracoviště	NTI
Datum zadání BP	18. 10. 2012
Plánované datum odevzdání	17. 5. 2013
Rozsah grafických prací	dle potřeby dokumentace
Rozsah průvodní zprávy	cca 45 stran
Název BP (česky)	Webová aplikace pro vytváření, správu a sledování tréninkových plánů
Název BP (anglicky)	Online training diary management application

Zásady pro vypracování BP

1. Analýza problému, stanovení sledovaných kritérií
2. Volba vhodných prostředků pro realizaci
3. Návrh databázové struktury
4. Realizace aplikace za použití vhodného frameworku
5. Otestování a ověření funkčnosti
6. Vytvoření instalační distribuce a stručného návodu k použití

Seznam odborné literatury

[1] Gilmor, J. W., Velká kniha PHP a MySQL 5 - kompendium znalostí pro začátečníky i profesionály, Zoner Press, 2007

[2] Kosek, J., PHP - tvorba interaktivních internetových aplikací, Grada Publishing, 1999, ISBN 80-7169-373-1

Vedoucí BP	Ing. Petr Kretschmer
Konzultant BP	Ing. Vojtěch Wrnata

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jeho využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Petru Kretschmerovi za pomoc při zodpovídání mých dotazů a poskytování potřebných rad. Dále bych rád poděkoval mému trenérovi Mgr. Petru Jeřábkovi, Ph.D. za poskytnutí podnětů ke zlepšení uživatelského prostředí aplikace a celé mojí tréninkové skupině za pomoc při jejím testování.

Abstrakt

Cílem práce je vytvoření webové aplikace pro tvorbu tréninkových plánů, jejich správu a usnadnění analýzy absolvovaných tréninkových jednotek. To vše na základě možnosti prohlížení starších záznamů, grafů jednotlivých tréninkových ukazatelů, grafů vývoje výkonnosti v závodech a soutěžích za předvolená období a pročítání subjektivních pocitů. Aplikace bude primárně zaměřená na hierarchii v tréninkové skupině, ale nebude vylučovat možnost vedení deníku jednotlivce.

Bude v ní figurovat několik uživatelských rolí. Konkrétně to bude host bez registrace a mezi registrovanými to bude svěřenec, trenér a administrátor. Svěřenec i trenér budou moct být zároveň administrátoři s omezeným či neomezeným oprávněním. Nebude ale možné, aby byl trenér zároveň svěřencem a naopak.

Jednotlivé části rešerše postupně představují problematiku, existující řešení, rozdíly v mém přístupu, následnou realizaci za použití PHP Nette frameworku a popis funkcionality konečné aplikace.

Klíčová slova

Tréninkový deník, tréninkové a závodní statistiky, webová aplikace, trenéři a svěřenci, Nette framework

Abstract

The goal of the work is to create a web application that manages training diaries and simplifies the analysis of completed trainings. The analysis is based on reviewing older entries, training mark charts, race result charts in a specified date range and subjective notes of the user. The application will primarly be focused on a training group but will not exclude the possibility of a single user.

There will be four user roles. First will be an unregistered visitor and amongst the registered there will be a coach, a training group member or an individual and an administrator. Individuals and coaches may also be administrators at the same time who might have limited or unlimited rights. However it will not be possible to be a group member and a coach at the same time.

The thesis is divided into parts that describe the topic, alternative and existing software, differences in my approach and finally the realisation of the application in the PHP Nette framework along with the description of all its sections.

Keywords

Training diary, training and race statistics, web application, coaches and group members, Nette framework

Obsah

Seznam ilustrací

Seznam zkratek, symbolů a pojmů

apod.	a podobně
atd.	a tak dále
např.	například
ms	milisekunda
AJAX	obecné označení pro technologie umožňující měnit obsah stránek bez znovunačítání (*Asynchronous JavaScript and XML*)
CSS	kaskádové styly
framework	softwarová architektura, která poskytuje nástroje pro programování
Google Drive	služba k ukládání dat na webu od společnosti Google
GPS	družicový systém pro určení přesné polohy (*Global Positioning System*)
HTML	jeden z jazyků pro tvorbu webových stránek (*HyperText Markup Language*)
invalidace	obnovení kusu stránky bez potřeby jejího kompletního načtení
JSON	způsob zápisu dat, který je přenositelný mezi rozdílnými systémy (*JavaScript Object Notation*)
Latte	šablonovací jazyk, který využívá Nette framework
plugin	samostatně nefunkční doplněk softwaru
URL	řetězec určující umístění dokumentu na internetu (*Uniform Resource Locator*)
xls	zkratka formátu souboru, který využívá Microsoft Excel
XSS	způsob útoku na webové stránky podstrčením škodlivého kódu (*Cross-Site Scripting*)

1 Úvod

Na světě existuje velké množství sportů a ještě mnohonásobně větší množství lidí, kteří v jejich vykonávání mají svou zálibu. Někteří sporty dělají za účelem prostého udržení v kondici, jiní na závodní úrovni pro radost a většina těch sportovců, kterou vídáme na televizních obrazovkách to dělá pro peníze.

Z naší lidské podstaty je patrné, že jsme velice zvídaví tvorové, které po dosažení určité mety zajímá, jak se dostat na tu další. Sám bych se zařadil do kategorie sportovců, kteří závodí pro radost a vlastní potěšení (já osobně konkrétně v atletice). Čím více se zlepším, tím větší mám radost. Abych se ale neustále zlepšoval, tak je potřeba, abych dodržoval určitý tréninkový plán, který mi vymyslí můj trenér. Ten většinou vychází z předchozího tréninkového cyklu a upravuje ho na základě individuálních výsledků jednotlivých svěřenců v celé sezóně. Musí tedy spoléhat na to, že si každý z jeho svěřenců bude poctivě vézt vlastní deník. Jak má ale trenér jednoduše a efektivně zanalyzovat 10 svěřenců, kteří mají motivaci ke zlepšení z prostého přečtení jednotlivých deníků.

První možností je mít pořádnou trpělivost a přibližně měsíc volného času. To je bohužel naprosto nepraktické a neefektivní. Velkým pomocníkem pro atlety bylo vydání knihy [1], která ulehčuje záznam tréninku i různých hodnot, které je potřeba na tréninku měřit. Ulehčuje v tomto případě znamená, že má svěřenec k dispozici tabulky, do kterých vyplňuje hodnoty a následně je potom podle potřeby sčítá. Efektivita je určitě větší, ale stále je to pracné a není mnoho sportovců, kteří by na to měli dostatek trpělivosti. V dnešní době poměrně dost lidí využívá Microsoft Excel nebo podobné tabulkové procesory. Počítání udělá program a když jste šikovní a dobře si tabulky uděláte, tak dále nemáte tolik starostí. Ne každý ale umí Excel ovládat na takové úrovni, aby si to byl schopen zařídit a žádná univerzální šablona pro tyto potřeby neexistuje nebo jsem na ni nikde nenarazil.

Proto jsem po konzultaci s mým trenér došel k závěru, že by se nejen nám hodilo nějaké grafické interaktivní prostředí, které by vedení deníků dokázalo ulehčit a zpříjemnit. Pro realizaci jsem tedy zvolil vytvoření webové aplikace, ke které bude mít přístup kdokoliv a nebude muset podstupovat žádnou instalaci, pouze jednoduchou registraci. Rozhodl jsem se aplikaci pojmout tak, aby nebyla stavěná nikomu na míru, ale aby ji mohl používat opravdu každý. Nebude tedy vůbec záležet na tom, jestli uživatel dělá atletiku, hokej nebo plavání. Bude mít možnost absolvované tréninky zapsat slovně tak, jako by to dělal na papíře, ale navíc si bude moct vymyslet vlastní tréninkové ukazatele a nechat tak program počítat

všechny potřebné náležitosti. Deník si bude moct vézt buď jednotlivec nebo celá skupina sportovců vedená jedním nebo více trenéry. Budou zde tedy vystupovat dvě hlavní role – svěřenec a trenér, který bude moct systém využívat jako plánovací nástroj a zpětně bude moct kontrolovat, jestli svěřencům trénink dělal problém, jestli ho dokázali splnit a podobně.

2 Analýza problému

Nejprve si stanovím, jaké aspekty pro mě jsou nejdůležitější a jaké mám od výsledné aplikace požadavky. Rozeberu všechna kritéria, která budu později srovnávat s ostatními existujícími řešeními a popíši, v čem se ta řešení liší.

2.1 Sledovaná kritéria a požadavky

Jedním z nejdůležitějších sledovaných faktorů pro mě bude flexibilita práce s tréninkovými ukazateli. Záznam těchto ukazatelů řeší každý sportovec jinak a měl by v tomto ohledu dostat co největší volnost. Zároveň je to ta nejpodstatnější věc na celém vedení deníku a v případě psaní na papír také ta nejpracnější. Chci dát uživateli možnost, aby si dokázal jednotlivé ukazatele řadit do větších logických celků (kategorií). Ukážu to na jednoduchém příkladu. Uživatel si vytvoří kategorii ukazatelů se jménem „regenerace“ a přiřadí k ní konkrétní případy regenerace jako je pobyt v sauně, pobyt v páře nebo v bazénu a podobně. Všechny ukazatele v kategorii by měly mít stejnou měrnou jednotku (v tomto případě např. minuty nebo hodiny), aby byla naměřená data konzistentní a bylo možné je graficky zobrazovat jak jednotlivě, tak dohromady, jako celou kategorii.

Dalším důležitým prvkem pro mě bude rozdělení uživatelů do rolí. Nepočítaje hosta, který bude mít možnost registrace a prohlížení článků, se bude program točit kolem dvou hlavních rolí, a to kolem svěřence a trenéra. Ti spolu budou moct vytvořit tréninkovou skupinu nebo tým, kde trenér bude mít možnost plánování tréninků a svěřenci si budou udržovat záznamy o tom, zda-li trénink splnili včetně informací o tom, jaké z něj měli pocity a podobně. Skupinové plánování bude jeden z nejčastějších rozdílů oproti aplikacím, které jsou k dostání.

Budu chtít, aby měl jak trenér, tak svěřenec, na základě svých naměřených dat možnost generování jejich grafického průběhu pro snazší poučení, případně zvědavost. Ze zkušenosti a ohlasu lidí v okolí většina z nich nebude chtít, aby byly tyto záznamy viditelné ostatními uživateli. To znamená, že si uživatel bude moct prohlížet pouze své tréninky, případně potom tréninky svých kolegů ve skupině.

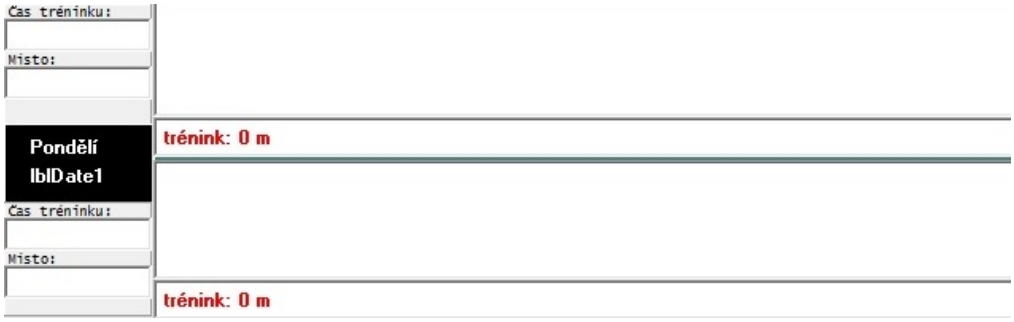
Do tréninkového deníku potom neodmyslitelně patří zaznamenávání absolvovaných závodů či zápasů. Stejně jako z tréninkových ukazatelů by bylo příjemné vidět graficky znázorněnou křivku vývoje výkonnosti v jednotlivých disciplínách. Tato možnost bude ve sportech jako je atletika nebo plavání velmi vítanou možností, která bude mít i výpovědní hodnotu. V týmových sportech, kde je výsledkem skóre zápasu, případně body z turnaje, tato data pravděpodobně nebudou až zas tak zajímavá.

2.2 Existující řešení

Existuje poměrně dost možností, jak si vézt tréninkový deník, ale žádná z nich mě nezaujala natolik, abych ji mohl začít používat nebo jednoduše nedokázala splnit mé stěžejní požadavky. Narazil jsem na pár aplikací pro stolní počítač a dalších pár webových aplikací, které vedení deníku řeší. Každý je zaměřen na něco trochu jiného a práce s nimi se mnohdy velmi liší.

2.2.1 Tréninkový deník TROPOL

Jedná se o aplikaci pro stolní počítač vydanou v roce 2004 jejímž autorem je uváděna webová stránka tropol.cz. Je zde několik záložek, které uživatele postupně vedou od založení účtu až k zapsání prvního tréninku. Když jsem se ale pokusil trénink zapsat, vyskočila na mě hláška o špatném formátu data a já nebyl schopen editovat jediné pole, takže jsem s tímto programem nenabyl moc zkušeností. Narazil jsem zde na možnost grafického zobrazení naměřených údajů, ale bohužel nevím, jak přesně funguje. Uživatel zde má možnost vytvořit vlastní tréninkové ukazatele, ale má omezené možnosti na nastavení jejich měrných jednotek na metry, kilometry a obecnou časovou jednotku. Dalším problémem je, že v jeden den je uživatel omezován počtem vytvořených tréninků na dva (viz Ilustrace 1), což rozhodně nepřidává na flexibilitě. I přes to, že je zde možnost vytvoření ukazatelů, tak jsem nepřišel na to, jak je přiřazovat k jednotlivým tréninkům. Možná bych měl víc štěstí, kdyby samotný program fungoval.



Ilustrace 1: Detail editačního pole pro záznam tréninku v aplikaci TROPOL

2.2.2 SportTracks 3

Jedná se o další desktopovou aplikaci, kterou můžeme získat na stránkách zonefivesoftware.com. Oproti té předchozí vypadá na první pohled lépe. Uživatel si o sobě zadá základní informace jako je jméno, výška a váha, podle čehož potom dokáže počítat spálené kalorie a může Vám udržovat historii vývoje těchto údajů. Je zde možnost přidání libovolného počtu tréninku do jednoho dne, ale jinak je program cílen poměrně konkrétně na vytrvalostní sporty. Každý trénink má předepsané kolonky pro vyplnění vzdálenosti, tempa a dalších

informací, které se hodí jen specifickým sportovcům. Není zde žádná správa na měření libovolných věcí a to je velký rozdíl oproti mé aplikaci.

Jako plus vidím to, že je na oficiálních stránkách komunita, která tvoří pluginy, které rozšiřují funkce již hotového programu. Můžete si tak např. doinstalovat zobrazení výškového profilu uběhnuté trasy, kterou do programu nahrajete, zobrazení na mapě a mnohé další. Tento program má určitě něco do sebe, ale je příliš zaměřený na konkrétní cílovou skupinu, není zde možnost hierarchie tréninkové skupiny, plánování, ani správa vlastních měřených údajů.

2.2.3 Microsoft Excel (tabulkové procesory)

Tabulkové procesory jsou ve vedení deníků velmi oblíbené. Práce s nimi dokáže být poměrně dost flexibilní a vše jde přizpůsobit konkrétním potřebám. Jednotlivé prvky tréninků si můžete rozlišovat různým barevným pozadím, abyste se dokázali na první pohled v dokumentu zorientovat. S Excelem mám vlastní zkušenosti. Dokázali jsme si z něj dokonce udělat skupinový plánovač. Využili jsme k tomu službu Drive od společnosti Google. Nahrál jsem na toto úložiště jeden soubor a zpřístupnil jsem ho všem členům tréninkové skupiny. Každého svěřence reprezentovala jedna záložka v .xls souboru a trenér každému dopředu plánoval tréninky.

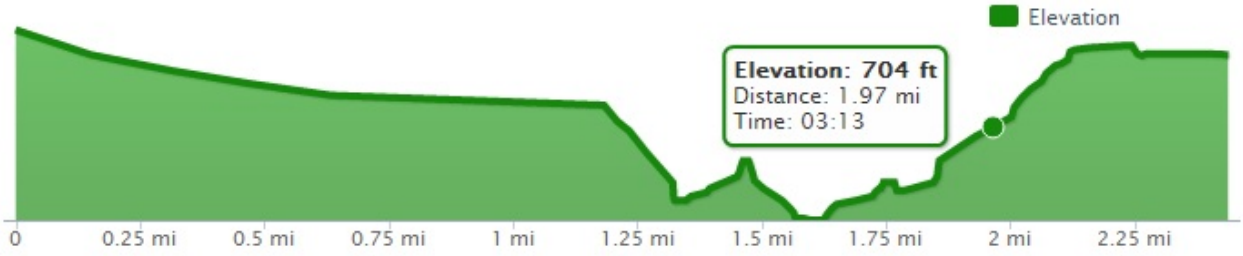
Velkou nevýhodou bylo to, že čím větší množství tréninků bylo v souboru, tím začínalo být všechno méně přehledné a zpětná analýza těchto dat byla pro trenéra velmi zdlouhavá. Navíc jsme ani nezaznamenávali žádné ukazatele do speciálních kolonek, protože jsme nevymysleli, jak to udělat obecně a efektivně.

2.2.4 Runkeeper

Runkeeper je webová aplikace dostupná na adrese runkeeper.com. Podobně jako program v kapitole 2.2.2 je aplikace zaměřená především na vytrvalostní sporty jako je delší běhání, cyklistika a další aktivity, kde se většina tréninkových údajů bere ze zaznamenané trasy nahrané přístrojem s GPS. Pokud GPS nemáte, je zde možnost nakreslení absolvované trasy přímo do mapy. Po dokončení Vám přístroj vše vykreslí do mapy včetně výškového profilu trasy (viz Ilustrace 2) a také grafu rychlosti a vývoje srdečního tepu, pokud jste nahráli soubor s těmito údaji.

Aplikace je velice pěkně provedená a má také mobilního klienta pro telefony se systémy android a iOS, takže když venku běžíte se zapnutým telefonem, tak se po připojení na internet všechny informace automaticky nahrají a zpracují. Je to ale zaměřené jiným směrem, než jsem se vydal já a nesplňuje to požadavky, které bych od takové aplikace měl.

Na tento způsob funguje spousta dalších webových aplikací, které vyvíjí velké společnosti angažující se v prodeji sportovního vybavení jako Nike nebo Adidas.

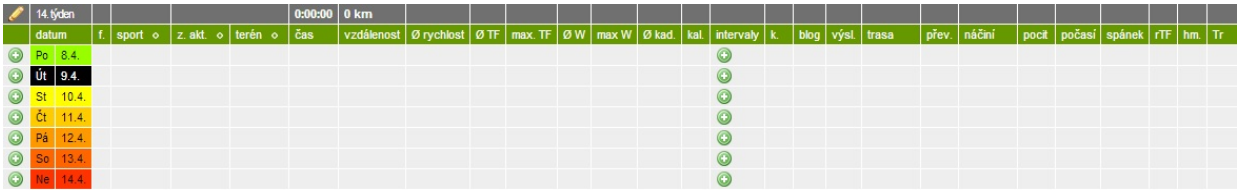


Ilustrace 2: Ukázka výškového profilu trasy z aplikace Runkeeper

2.2.5 MyTreneek

Jedná se o českou webovou aplikaci dostupné z webu na mytreneek.cz, která je pravděpodobně nejvíce podobná mému konceptu. Členství, které je zdarma nabízí možnost záznamu tréninku v rozdílných formách. Ty jsou ale pevně dané a pro někoho tudíž zbytečné. Uživatel zde není vázaný na konkrétní sport, ale každý trénink si může založit v duchu jiného sportu. Výběr sportů je omezený, ale je zde možnost přidat si vlastní. V této verzi si není možné vézt deník soukromě, ale každý Váš záznam okamžitě uvidí ostatní uživatelé na hlavní stránce. Tréninkové ukazatele si zde uživatel může vytvořit, resp. doplnit sadu ukazatelů, která už je v programu po registraci. Není mi však na první pohled jasné, jestli jsou to ukazatele společné s ostatními uživateli nebo jestli jsou dostupné pouze mě.

Aplikace vypadá zajímavě, ale působí na mě tak, že se cítím svázaný i přes to, že má uživatel možnosti na vytvoření vlastních prvků tréninku. Pravděpodobně je to trochu chaotickým uspořádáním formulářů a přehledů (viz Ilustrace 3 – týdenní přehled). Hlavním nedostatkem je pro mě omezenost volné verze. Není zde možnost rozdělení rolí na trenéra a svěřence a následné skupinové plánování. Údajně je tato možnost v placené verzi, kterou jsem neměl možnost vyzkoušet, takže nevím, jak přesně funguje, ale na základě zpracování této základní verze usuzuji, že ji autor pojal jiným způsobem, než by to vyhovovalo mě.



Ilustrace 3: Ukázka týdenního přehledu v aplikaci MyTreneek

2.3 Shrnutí hlavních rozdílů

Po prozkoumání ostatních nalezených alternativ je patrně vidět několik hlavních rozdílů, ve kterých se liší od mých požadavků. Ani jeden z existujících projektů mi nevyhovoval

po stránce flexibilní správy vlastních tréninkových ukazatelů. Představoval bych si, že v tomto ohledu bude mít uživatel naprostou volnost.

Velkým rozdílem je to, že žádný z projektů není zaměřen na vedení deníku pro tréninkové skupiny. Všechny se zabývají pouze jednotlivci a když už týmy (viz kapitola 2.2.5), tak je to zpoplatněné. Na skupinové plánování jsem tedy nikde nenarazil a budu na to klást hlavní důraz. Budu vycházet z toho, jak to chodí v atletice a tyto principy aplikuji i na ostatní sporty, ve kterých je to velmi podobné. Tréninková skupina nebo tým má jednoho či více trenérů, kteří jim mohou ať už společně nebo zvlášť tréninky plánovat.

Nejlépe zpracované tréninkové statistiky měl podle mě projekt Runkeeper (viz kapitola 2.2.4). Byl ale zaměřen na sporty, kde člověk 90 % všech údajů bere z tras naměřených GPS přístrojem. Tato data potom ale dokázal zpracovat do toho nejpodrobnějšího detailu a zobrazit např. vývoj rychlosti nebo nadmořskou výšku do interaktivního grafu, který byl synchronizován s uloženou trasou na mapě. Tuto možnost do své aplikace zahrnovat nebudu, ale jinak chci, aby si uživatel mohl zobrazit veškeré naměřené údaje za přesně určená období jak z tréninkových ukazatelů, tak z jednotlivých výsledků v závodech.

3 Návrh aplikace

Po stanovení vlastních požadavků na aplikaci se podívám na to, jak ji celou navrhnout. Nejdříve si vyberu technologie, které budu k její tvorbě využívat a vysvětlím, proč jsem si vybral právě ty. Po zvolení těch správných technologií popíši návrh databázové struktury od konkrétních entit se všemi atributy až po jejich vzájemné relace. Nakonec bude zbývat popsání konkrétní struktury aplikace.

3.1 Použité technologie

Pro serverovou část aplikace jsem zvolil jazyk PHP s velmi oblíbeným frameworkem Nette. Klientskou část bude obsluhovat JavaScript, především potom JavaScriptová knihovna jQuery. Budu také využívat CSS framework zvaný Twitter bootstrap pro zjednodušení a urychlení práce s rozvržením vzhledu stránek. Mezi poslední větší použité komponenty patří jQuery plugin jqPlot pro vykreslování dynamických grafů a TinyMCE grafický textový editor.

3.1.1 PHP Nette framework

Podle [2] je Nette framework „populární nástroj pro vytváření webových aplikací v PHP 5. Při programování Vám vychází vstříc a nepřidělává vrásky. Eliminuje bezpečnostní rizika, podporuje AJAX a znovupoužitelnost.“ Je to tedy nadstavba jazyka PHP, která nám poskytuje velké množství nástrojů na postavení webové aplikace. Framework je postaven na návrhovém vzoru MVC (Model-View-Controller).

Obrovskou výhodou Nette je jeho propracované zabezpečení, na které klade velký důraz. Je stavěn tak, že odolává nejrůznějším úrokům a uživatel se sám o tyto záležitosti nemusí starat. Je např. odolný vůči XSS (Cross-Site Scripting) – využití neošetřených výstupů, díky nimž je útočník schopen do stránky vložit a vykonat svůj škodlivý kód. Nette všechny výstupy ošetřuje automaticky tak, aby k útoku nemohlo dojít. Automaticky jsou také ošetřeny všechny vstupy, takže nemusíme ani nic nastavovat. Nette se také automaticky stará o ochranu session proti odcizení či podstrčení session ID, který zajišťuje identifikaci uživatele.

Další příjemnou věcí na Nette jsou jeho ladící nástroje. Autoři frameworku si tyto nástroje sami velice pochvalují a mají proč. Je zde k dispozici tzv. Laděnka. Ta nám buď při chybě nebo nezachycené výjimce vygeneruje přehledný náhled toho, kde chyba nastala včetně všech detailních informací např. o tom, v jaké konfiguraci serveru chyba nastala nebo jaké proměnné byly v době výskytu chyby k dispozici. Pokud něco takového nastane, tak se vygeneruje HTML soubor s chybou a uloží se pro případnou pozdější analýzu. Máme možnost si natavit, jestli aplikace běží v produkčním či vývojovém režimu. Při nastání chyby v produkčním režimu

uživatele neotravují chybové hlášky, ale automaticky je mu zobrazena přívětivá zpráva, která oznamuje, že nastala určitá chyba. Dále zde ještě nalezneme nástroje na měření času nebo informační výpis proměnných.

Posledním ladícím nástrojem, který nesmí být opomenut, je tzv. debugger bar. Jedná se o drobnou lištu, která se automaticky zobrazuje v pravé dolní části okna prohlížeče. Na ní jsme na první pohled rychle schopni zjistit základní informace o průběhu vykonání aktuální stránky jako je celkový čas, množství alokované paměti, doba a počet vykonání dotazů na databázi a další.

Další silná stránka Nette je jeho šablonovací systém zvaný Latte. Latte byl první šablonovací systém, se kterým jsem se kdy setkal a oslnil mě svou jednoduchostí. Stačilo se naučit tzv. makra, která zastupují základní PHP funkce a psaní šablon a vzhledu začalo být rázem velmi snadné.

Nakonec musím zmínit, že je Nette dělaný tak, aby se v něm snadno stavěly Ajaxové aplikace. Bez frameworku je tvorba takové aplikace poměrně složitá a časově náročná. V Nette vše funguje tak, že se napíše funkční základ, který po doplnění pár řádků kódu kompletně změní chování aplikace na Ajaxovou.

3.1.2 Twitter bootstrap CSS framework

Tento framework se používá k tvorbě vzhledu samotné stránky. Jedná se o sadu CSS stylů, JavaScriptových knihoven a velkého množství ikon, které nám při tvorbě webu dokáží velmi urychlit a především usnadnit práci.

Použitelné komponenty se dělí na ty, které vyžadují JavaScript či ne. První příjemnou pomůckou je mřížkový systém. Může mít libovolný počet řádků, kde každý řádek je rozdělen na dvanáct stejně velkých částí. Tyto části můžeme různě spojovat a rozdělit tak stránku např. na dva stejně velké sloupce, jeden větší než druhý nebo klidně pět různě dlouhých sloupců. Takto by podle [3] vypadal řádek, který by byl rozdělen na dva sloupce, přičemž levý by zabíral třetinu řádku a pravý dvě třetiny:

```
<div class="row">
   <div class="span4">...</div>
   <div class="span8">...</div>
</div>
```

Rodičovský element musí mít třídu `.row` a každý z jeho potomků musí mít třídu `.span*`, kde se místo hvězdičky použije číslo od jedné do dvanácti. Pro vyplnění celé šířky rodiče je potřeba, aby čísla u těchto tříd měla dohromady součet roven dvanácti.

Ze základních komponent jsou zde dále k dispozici velmi dobře zpracované tabulky. Základní tabulka musí mít třídu `.table`. Pokud chceme tabulku trochu vylepšit, např. automaticky lehce podbarvit každý druhý řádek, tak pouze stačí přidat další třídu – `.table-striped`. Pro tabulku se zobrazenými okraji potom stačí použít třídu `.table-bordered` a je jich ještě víc. Všechny se dají vzájemně kombinovat a je tak možno snadno vytvořit tabulku podle našich představ.

Stejně dobře jsou zpracované třídy na formátování formulářů, u kterých máme např. možnost nechat celý formulář vykreslit v jedné řádce. Stejně jednoduše můžeme ovlivňovat velikosti jednotlivých prvků formuláře jako jsou textová pole, rozbalovací seznamy apod. Dále máme k dispozici několik základních stylů pro různá zobrazení tlačítek, např. na změnu jejich barvy nebo velikosti.

Mimo těchto základních jsou tu i složitější hotové komponenty připravené k použití. Mezi ně patří třeba skupina tlačítek nebo hotová navigační lišta, která se dá fixně umístit k horní části stránky. Systém záložek s nadpisy, které po zvolení zobrazí na stejném místě jiný obsah. Výstražné boxy s různými odstíny barev a mnohé další.

Velmi silnou stránkou bootstrapu jsou JavaScriptové komponenty. Dokáží z webu udělat naprosto interaktivní aplikaci. Příkladem mohou být dialogová vyskakovací okna, která se dají použít pro zobrazení detailnějšího popisu nebo pro požádání uživatele o potvrzení jeho akce. Záložky, které už jsem zmiňoval v odstavci výše by bez JavaScriptu nemohly fungovat. Dále jsou zde hotové nastylované bubliny zobrazující se po na najetí na element s HTML atributem `title`, které přepisují výchozí vzhled. A v neposlední řadě ještě např. našeptávač, který nám po zadávání textu v textovém poli radí, jaké jsou možnosti.

3.1.3 JavaScriptová knihovna jQuery

jQuery je podle [4] rychlá, malá a bohatá JavaScriptová knihovna, která díky jednoduchému API ulehčuje manipulaci a průchod objektem dokumentu, animace, Ajax a obsluhu událostí, a která je kompatibilní s celou řadou prohlížečů.

Ohromnou silou a důvodem, proč je tato knihovna tak široce používána je její princip výběru elementů v dokumentu. Využívá k tomu CSS selektory. Tímto způsobem tedy získáme přístup k libovolnému elementu, na který potom můžeme aplikovat další funkce. V případě, že nám výsledek selektoru vrací více než jeden výsledek, tak se vrátí všechny a pokud na tento výsledek aplikujeme funkci, tak se aplikuje na všechny elementy najednou. Máme dokonce možnost funkce řetězit a aplikovat jich několik za sebou: Např. takto:

```
$("input").hide(300).css("background", "#F00").show(300);
```

Příklad výše nám na celé stránce nalezne všechny elementy typu `input`, aplikuje na ně funkci `hide`, která nám s číselným parametrem zajistí, že se všechny elementy za 300 ms schovají. Funkce `css` všem nastaví červené pozadí a poslední funkce show všechny prvky opět za 300 ms zobrazí. V tuto chvíli už ale budou mít všechny červené pozadí.

Na následujících dvou příkladech ukážu rozdíl mezi čistým JavaScriptem a jQuery v získávání dat ze serveru. Oba budou dělat stejnou věc, a to stahovat text článku na základě jeho id v databázi po kliknutí na tlačítko a načtou ho do elementu `div` s `id="ajax"`.

JavaScript:

```
function nactiTextClanku(idClanku) {
var xmlhttp;
if (window.XMLHttpRequest) {
   xmlhttp=new XMLHttpRequest();
} else {
   xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.onreadystatechange = function() {
if (xmlhttp.readyState==4 && xmlhttp.status==200) {
document.getElementById("ajax").innerHTML= xmlhttp.responseText;
}
}
xmlhttp.open("GET", "nactiTextClanku.php?idClanku=" + idClanku,
true);
xmlhttp.send();
}
```

jQuery:

```
$("button#nactiClanek").click(function() {
   var idClanku = 3;
   $.get("nactiTextClanku.php", { "idClanku" : idClanku},
   function(textClanku) {
        $("#ajax").html(textClanku);
   });
});
```

Na první pohled je vidět, že kód v JavaScriptu je pomalu dvakrát tak delší a není zdaleka tak přehledný. Navíc jsme museli vytvořit funkci, která se o načítání stará. V tuto chvíli samotný kód ještě není funkční, protože ho musíme odněkud zavolat. V HTML budeme tedy mít např. následující tlačítko, po jehož kliknutí se vykoná naše funkce:

```
<button id="nactiClanek" onclick="nactiTextClanku(3);">Načti
článek</button>
```

První `if` podmínka v naší funkci zajišťuje, aby komunikace se serverem fungovala i se staršími prohlížeči. To vše už je v jQuery vyřešené a nám stačí jednoduché volání funkce $.get, která přejímá tři parametry – url, odesílaná data a funkci, která se vykoná po úspěšném vykonání dotazu.

Kód v jQuery už je takto hotový a nepotřebuje již nijak zasahovat do HTML. To je jednou z dalších výhod této knihovny. Toto použití se nazývá jako „nevtíravé“ (z anglického unobtrusive).

3.1.4 Ostatní komponenty

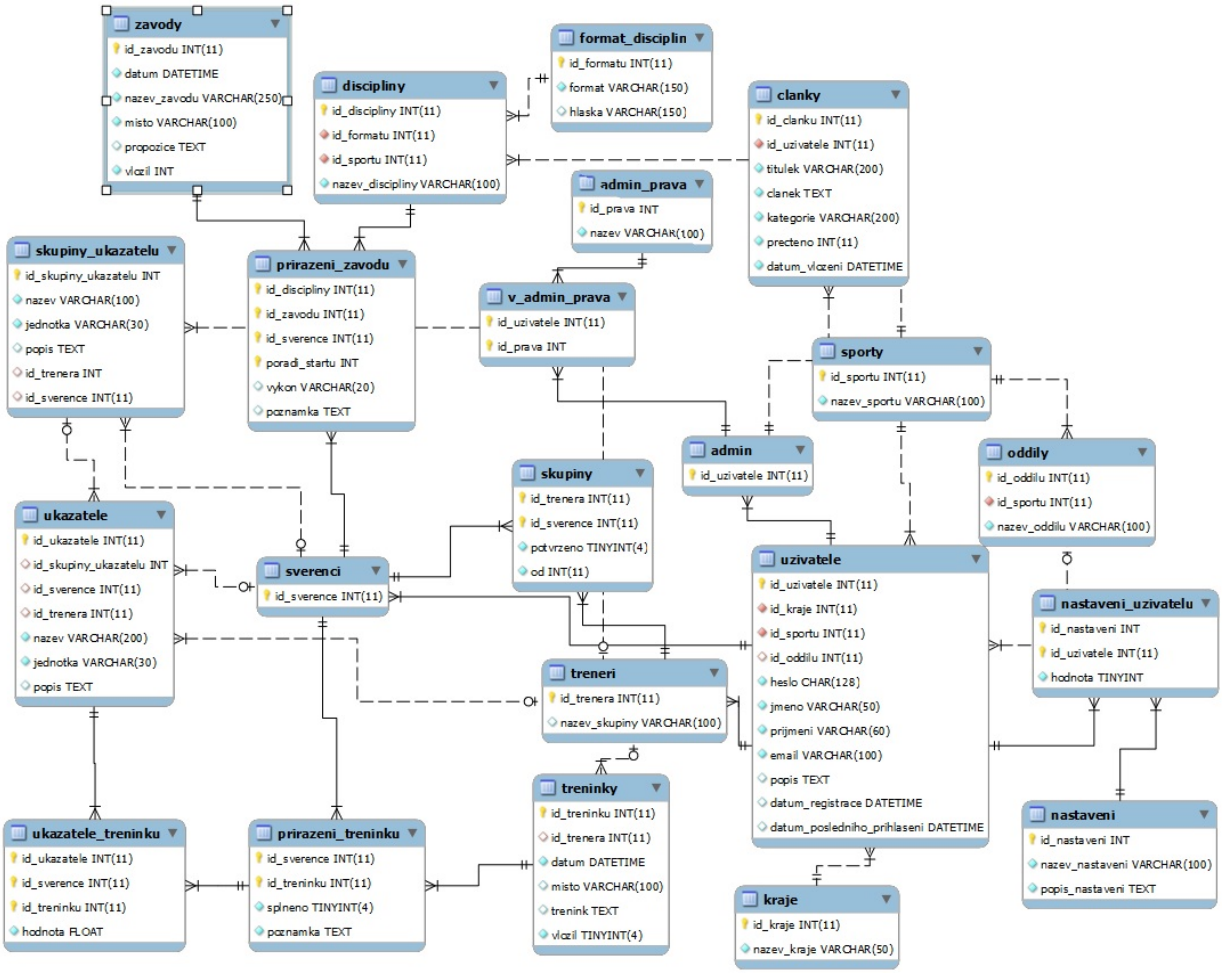
Další použité nástroje jsou už jen konkrétní hotové komponenty, které jsem využil pro ulehčení práce. Patří sem plugin jqPlot [5], což je rozšíření knihovny jQuery pro vykreslování grafů. Komponenta má nepřeberné množství nastavení a s její pomocí jsme schopni naše data zobrazit ve sloupcových, koláčových a dalších grafech.

Využívám také textový editor TinyMCE [6], který dokáže živě formátovat psaný text tak, jako by ho uživatel psal např. v programu Microsoft Word. Umožní uživateli vlastní text snadno naformátovat a strukturovat, zvýraznit důležité části apod.

Z další JavaScriptové knihovny jQuery UI [7] využívám drobnější komponenty na výběr data nebo pomocné šipky k textovému poli, které po stisknutí zvyšují nebo snižují číselnou hodnotu uvnitř pole. Plugin pro výběr data bohužel nativně nepodporuje výběr času, který také potřebuji, takže k tomu navrch využívám doplněk Timepicker [8].

3.2 Databázová struktura

Návrh databáze byl v mém případě velmi stěžejním krokem. Potřeboval jsem v ní zaznamenat potřebné uživatelské informace a vyřešit rozdělení rolí. Z návrhu nakonec vzešli role tři – administrátor, trenér a svěřenec. Následně bylo potřeba vytvořit dvě skupiny tabulek, kde jedna má za úkol vyřešit ukládání a přiřazování tréninků a ta druhá záznam ze závodů či zápasů. Do tréninků bylo navíc potřeba zakomponovat ukazatele tak, aby je mohli uživatelé využívat individuálně. Při návrhu jsem také potřeboval vyřešit globální správu disciplín ve sportech a článků v kategoriích, abych se v budoucnu nemusel o všechno starat já, ale aby si to dokázali obstarat sami uživatelé. Později jsem si ještě uvědomil, že bych mohl vytvořit uživatelské individuální nastavení. Vše by mělo být patrné z ER diagramu (viz Ilustrace 4).



Ilustrace 4: ER diagram databázové struktury

3.2.1 Uživatelé a jejich role

Všichni uživatelé jsou uchováni v tabulce `uzivatele`, kde primárním klíčem je umělé id. O každém z uživatelů si dále budeme uchovávat jeho jméno a příjmení, heslo, email, datum registrace a posledního přihlášení, popis, kterým se budou moct prezentovat ostatním uživatelům a ještě informace o tom, z jakého je kraje a jaký sport dělá. Kraj a sport budou v této tabulce cizí klíče do číselníků, ve kterých jsou kraje a sporty předdefinované. Při registraci si z těchto seznamů bude muset uživatel vybrat.

Rozlišení rolí následně řeší tři další tabulky, a to `admin`, `sverenci` a `treneri`. Každá z nich kromě tabulky `treneri` má pouze jeden atribut, kterým je umělý identifikátor. Trenéři mají navíc pole `nazev_skupiny`, který mohou použít při psaní článků pouze v rámci skupiny. Je to cizí klíč k id z entity `uzivatele`. Při každém vytvoření záznamu v uživatelích se podle výběru v registračním formuláři vytvoří nový záznam do entity `sverenci` nebo `treneri`. To, jestli bude uživatel administrátor nemůže ovlivnit sám, ale konkrétní práva mu musí přidělit hlavní administrátor.

K vytváření skupin slouží tabulka `skupiny`, která má složený primární klíč z id trenéra a id svěřence. Další dvě položky ve skupině jsou `potvrzeno` a `od`. První z jmenovaných atributů říká, jestli je vztah mezi trenérem a svěřencem platný. Předejde to případu, kdy by se chtěli neoprávnění svěřenci dostat do cizích skupin apod. Druhý z atributů slouží k tomu, abychom poznali, kdo odeslal žádost o zařazení do skupiny. Díky této M:N relaci je možné, aby měl trenér více svěřenců ve skupině a svěřenec byl naopak členem více skupin najednou.

3.2.2 Tréninky

Trénink v databázi zastupuje entita `treninky`. Primárním klíčem je opět umělý číselný identifikátor. O každém tréninku držíme informace o datu, místě a jeho textovém popisu. Dále také o tom, který uživatel (ať už trenér nebo svěřenec) trénink vložil a id trenéra. To může být prázdné z toho důvodu, že svěřenec vůbec trenéra mít nemusí. Nebo může nastat situace, že absolvuje trénink, který mu jeho trenér nenaplánoval.

Přiřazení svěřenců k tréninku uchovává tabulka `prirazeni_treninku`. Má primární složený klíč id svěřence a id tréninku. V tabulce je navíc atribut `splneno`, který nabývá pouze hodnot 0 nebo 1 a znamená, jestli svěřenec trénink splnil nebo ne. Je tu také atribut `poznamka`, do které může uživatel zapsat své pocity z tréninku, informaci o tom, jestli trénink absolvoval kompletně nebo jestli šel něco trochu jinak apod.

3.2.3 Tréninkové ukazatele

Tato část si zasloužila největší pozornost. Samotné ukazatele jsou uloženy v tabulce `ukazatele`. Primárním klíčem je zde číselný identifikátor. Každý ukazatel má vlastní název, jednotku a popis. Aby bylo vedení tréninkových ukazatelů naprosto individuální i v rámci jednotlivých členů tréninkové skupiny, tak má také každý dva cizí klíče, které ovšem mohou nabývat hodnot `null`. Jsou jimi id svěřence a id trenéra. Jeden z nich musí být vždy vyplněn o což se ale stará aplikace. Pokud je vyplněno id svěřence, tak se jedná o jeho soukromý ukazatel, který nikdo jiný, kromě jeho trenéra nevidí. Ukazatele, které mají vyplněné id trenéra jsou automaticky k dispozici všem členům tréninkové skupiny.

Každý ukazatel má ještě jeden atribut, který jsem nezmínil, a to cizí klíč s názvem `id_skupiny_ukazatelu`. Odkazuje se na záznam v tabulce `skupiny_ukazatelu`. Tato tabulka má stejné atributy jako entita `ukazatele`. Sdružuje dohromady sady ukazatelů, které se budou moct v aplikaci vyhodnocovat jako jeden. Pravidla pro viditelnost ve skupině

jsou stejná jako v případě předchozí tabulky. Svěřenec má možnost si vybrat mezi tím, jestli ukazatel nebo skupina bude soukromá nebo k dispozici pro celou skupinu.

3.2.4 Závody

Závody a soutěže se ukládají do tabulky `zavody`. Primárním klíčem je číselné id. Každý závod má datum, název, místo konání, propozice a id uživatele, který závod založil. Struktura závodů je velmi podobná tréninkům. Přiřazení uživatelů je zaznamenáváno v entitě `prirazeni_zavodu`. Ta má primární klíč složený ze čtyř atributů – id závodu, id disciplíny, id svěřence a pořadí startu. Obsahuje ještě další dva sloupce – výkon a poznámku. Pořadí startu je pomocná hodnota, která svěřencům umožňuje zapsání více stejných disciplín do jednoho závodu (v atletice např. rozběh a finále na 100 metrů, které běžně bývá v jeden den). Id disciplíny je zároveň cizím klíčem k tabulce `discipliny`. Ta obsahuje název disciplíny, cizí klíč id sportu a id formátu. V tabulce `sporty` je potom seznam všech sportů, které mohou být použity. Tabulka `format_disciplin` obsahuje informace o formátu výkonu, se kterým se můžeme setkat při vyhodnocování disciplíny. Je to číselníková hodnota, která je reprezentována regulárním výrazem. Je zde připravena i hláška, která uživatele při zadávání upozorní, v případě, že regulární výraz nesedí se zadanou hodnotou.

Závody nepotřebují žádné ukazatele z toho důvodu, že každá disciplína je ekvivalentem určitého ukazatele. Máme o každém účastníkovi závodu informaci o jeho výkonu a disciplíně, z čehož se na základě datum dají vytvořit pěkné statistiky.

3.2.5 Práva a nastavení

Databázové rozdělení práv se bude týkat pouze uživatelů v roli administrátor. Vytvořil jsem tabulku `admin_prava`, která je číselníkem, kde primárním klíčem je číselné id a název role. Mezi konkrétní práva potom patří např. `admin`, který funguje bez omezení nebo právo `clankyVeSportu`, umožňující psát články týkající se sportu, ve kterém je uživatel zaregistrován atd. Přiřazení práv jednotlivým uživatelům zajišťuje entita `v_admin_prava` se složeným primárním klíčem id administrátora a id práva.

Uživatelské nastavení je záležitost připravená do budoucna. Tabulka `nastaveni` obsahuje název a popis nastavení a `nastaveni_uzivatelu` je mezi ní a uživatelem vazební tabulkou se složeným primárním klíčem a atributem s hodnotou příslušného nastavení. Zde bude moct být v podstatě libovolné nastavení, které by uživatelům mohlo ještě více ulehčit, zpříjemnit a přizpůsobit práci ke konrétním potřebám.

3.3 Aplikační struktura

Formální struktura aplikace je u Nette frameworku daná, takže z ní budu vycházet a popíši jednotlivé rozložení složek a způsoby uložení souborů. Potom proberu konkrétní návrh tříd, které vychází z toho, že Nette využívá návrhový vzor MVC, jak jsem již psal v kapitole 3.1.1. Podrobněji ho popíši v kapitole následující.

3.3.1 MVC návrhový vzor

Jedná se o způsob stavění aplikace, který ji rozděluje do tři hlavních částí. Tou první je model (M). Ten má na starost práci s databází resp. s obecnými daty, která aplikace potřebuje. Pohled (V – anglicky View) zajišťuje zobrazení dat poskytovaných modelem do formy, která bude srozumitelná uživateli. Tvoří se zde tedy grafické rozhraní aplikace. Třetí částí je Kontroler (C – anglicky Controller), který zajišťuje komunikaci mezi modelem a pohledem. Má na starosti modifikace dat v modelu na základě akcí, které uživatel udělá v pohledu a obráceně pohled zásobuje daty, které poskytuje model.

3.3.2 Adresářová struktura

Adresářová struktura Nette aplikace vypadá následovně:

- app – adresář s aplikační logikou
  - components/ – adresář s komponentami
  - config/ – adresář s konfiguračními soubory
  - models/ – adresář s modely pro komunikaci s databází
  - presenters/ – adresář s presentery pro spojení modelů a šablon
  - templates/ – adresář se šablonami
  - bootstrap.php – zaváděcí soubor aplikace
- libs/ – adresář s externími knihovnami
  - Nette/ – adresář s Nette frameworkem
- log/ – adresář pro logování chyb a výjimek
- temp/ – adresář pro cache a session data
- test/ – adresář s unit testy aplikace
- www/ – adresář, který je veřejně dostupný z webu

Adresář `app` obsahuje nejdůležitější část celé aplikace. V prvé řadě je to zaváděcí soubor `bootstrap.php`, který zajistí nastavení z konfiguračních souborů ve složce `config` a spustí aplikaci. Složka `models` obsahuje konkrétní modelové třídy pro komunikaci s jednotlivými

tabulkami a částmi databáze. V Nette se kontrolerům říká presentery a patří do složky `presenters`. Každý presenter je reprezentován jedním souborem a vlastní složkou v `app/templates`. Zde se jejich složky nazývají stejně a uvnitř obsahují sadu šablon, se kterými daný presenter pracuje.

Do šložky `libs` ukládáme všechny externí php knihovny, které v projektu používáme. Do složky `log` nám Nette ukládá všechny chyby a neošetřené výjimky, pokud máme tuto možnost v konfiguraci zapnutou. Adresář `temp` se využívá pro ukládání cache souborů, které Nette vytváří, aby zrychlovalo fungování aplikace. Stejně tak si sem Nette ukládá session soubory. Složka `test` slouží k uložení tříd pro testování naší aplikace. Poslední složkou v adresářové struktuře je `www`. Sem se ukládají všechny soubory, které mají být dostupné přímo z webu. Jedná se tedy o obrázky, soubory s JavaScriptem, kaskádovými styly a další.

3.3.3 Modely

Jedním z možných přístupů je vytvoření třídy pro každou tabulku. Některé tabulky jsou však vazební a sami o sobě nemají žádný význam. Vždy je potřeba, aby byly přidruženy k jiným tabulkám. Je tedy lepší, když vytvoříme modelovou třídu pro každou skupinu tabulek, které dohromady tvoří logický celek.

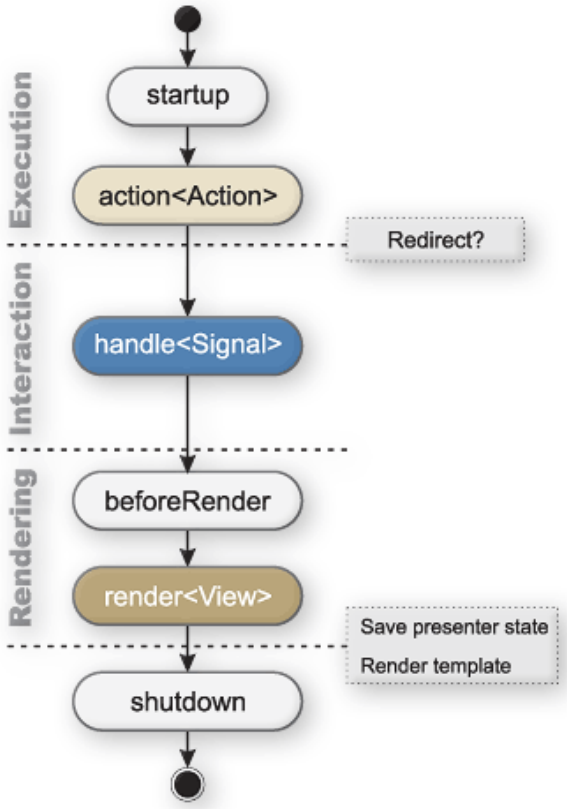
Všechny modely pojmu jako třídy se sadou metod, které vrací požadovaný výsledek. Návrh modelů udělám podle [9], kde je základem rodičovská třída `Repository`. Ta má k dispozici metody pro vrácení celého výběru tabulky nebo podle položek a hodnot předaných v asociativním poli. V konstruktoru třídy se předává parametr typu `\Nette\Database\Connection`, který zprostředkovává spojení s databází. Proměnná tohoto typu využívá sadu metod pro zpracování výsledku. Nejdůležitější metodou pro mě bude `query`, pomocí níž budu moct pracovat s konkrétními SQL dotazy na databázi. Použití ostatních metod očekává předem určené nastavení názvů databáze a čistě umělé klíče u každé tabulky, které já nemám. Jediný model, který nebude mít základ v `Repository` se bude starat o přihlašování a ověřování uživatelů. Ponese název `Authenticator` a bude implementovat rozhraní `Nette\Security\IAuthenticator`, které poskytuje přihlašovací metodu `auhenticate`.

Každý model dědící od `Repository` se nazývá `NazevTabulkyRepository`. Prvním je tedy model `UzivateleRepository`. Bude obsahovat metody pro práci s uživatelem, ale i svěřencem a trenérem. Model `ClankyRepository` bude mít na starost

získávání článků. Podobně budou vytvořeny menší modely pro získání dat z číselníků jako jsou disciplíny, kraje, nastavení, oddíly nebo sporty. Zajímavější potom budou zbývající modely s tréninky, ukazateli a závody. Vzájemně se totiž mohou kombinovat a budou mít na starost vykonávání složitějších dotazů např. pro generování statistik, kde vrácená data budou záviset jak na ukazatelích, tak na trénincích apod.

3.3.4 Presentery

V Nette se kontrolery (viz 3.3.1) nazývají presentery. Název každého presenteru je vždy `NazevPresenteruPresenter`. Z toho se odvozují názvy složek, ve kterých jsou uloženy šablony (viz 3.3.5). Každý presenter má svůj životní cyklus, v rámci kterého se postupně volají metody podle Ilustrace 5. Jako první se volá metoda `startup`, v které se zajišťuje základní nastavení atributů třídy. Hned po ní se volá příslušná metoda `action<Action>`, kde parametrem je název šablony, která se vykonává. Např. výchozí šablona má `action` metodu `actionDefault`. Zde se řeší přístupová práva a případné přesměrování při nedostatečném oprávnění. Metody `handle<Signal>` (např. `handleLoadData`) se provádí následně. Používají se zejména pro zpracování akcí od uživatele. Jako další se provede metoda `beforeRender`, v které se nastavují proměnné pro více šablon najednou.



Ilustrace 5: Životní cyklus presenteru z [10]

Posledním krokem před ukončením presenteru je provedení metody `render<View>`, kde je parametrem obdobně jako u `action` název šablony a předávají se zde proměnné, se kterými potom šablony pracují. Dále už následuje pouze ukončení cyklu.

Základem všech presenterů bude třída s názvem `BasePresenter`, která je potomkem obecného presenteru `Nette\Application\UI\Presenter` a od kterého budou všechny ostatní dědit. Každý presenter bude zastupovat jednu sekci aplikace. Jednotlivé sekce jsou popsány níže.

- presenters
  - AdminPresenter – administrační rozhraní
  - BasePresenter – základní presenter
  - ClankyPresenter – zobrazení článků
  - ErrorPresenter – chybový presenter
  - FAQPresenter – zobrazení nejčastějších dotazů
  - HomepagePresenter – hlavní strana
  - PrihlaseniPresenter – přihlašovací presenter
  - ProfilPresenter – rozhraní profilu uživatele
  - RegistracePresenter – registrační presenter
  - TreninkyPresenter – rozhraní pro zobrazení a práci s tréninky
  - ZavodyPresenter – rozhraní pro zobrazení a práci se závody

3.3.5 Šablony

Nette využívá šablonovací systém Latte (viz [11]). Každý presenter má své šablony, do kterých posílá data k vykreslení. Složka `templates` bude obsahovat tolik podsložek, kolik je presenterů využívajících zobrazení. V mém případě bude mít složku každý, kromě `BasePresenteru`, protože je abstraktní a ostatní od něj jen dědí. Název složek je stejný jako název presenteru, pouze se vynechá koncové slovo `Presenter`. Kromě složek se zde navíc nachází soubor `@layout.latte`. Jedná se o hlavní šablonu, která obsahuje předpis struktury celé aplikace. Jednotlivé šablony se při vykreslování vkládají do této hlavní a mění se tak pouze její část.

Každý pohled, který bude k dispozici bude reprezentován jednou šablonou. V každé složce bude vždy výchozí šablona `default.latte`, která se otevře, pokud po presenteru nebudeme požadovat konkrétní šablonu.

4 Popis funkcionality aplikace

V této části práce budu popisovat konkrétní funkce jednotlivých částí aplikace. Vychází to tak, že každý presenter zastřešuje jeden celek a skládá se z menších částí, které mají dohromady požadovanou funkcionalitu. Tréninky a závody budou ale např. velmi blízce spojeny, protože je potřeba oboje brát jako záznam v deníku sportovce. Ačkoliv se s nimi pracuje různými způsoby, tak se musí zobrazovat na stejné úrovni. Jednotlivé sekce mají stejný základ pro všechny uživatelské role, ale každému se budou zobrazovat pouze informace příslušné konkrétní roli.

4.1 Konfigurace aplikace

Základní konfigurace celé Nette aplikace se provádí v konfiguračním souboru `config.neon`, který se nachází v adresáři `app/config`, jak jsem již psal v kapitole 3.3.2. Nastavuji zde přístupové údaje k databázi pod názvem `database`, které mají konkrétní parametry v podobě hesla, serveru, apod. Stejně tak nastavuji dobu, po kterou budou platit veškeré session proměnné. Nejdůležitější částí je zde ovšem zaregistrování služeb (anglicky services). První službu, kterou zaregistrujeme je samotná databáze. Přístup k databázi je v Nette zprostředkován třídou `Nette\Database\Connection`, která přejímá všechny informace potřebné k jejímu připojení. Ty jsme si navíc nastavili na začátku konfiguračního souboru, takže je nemusíme psát znovu, ale stačí je zde použít. Dále je potom jako služby potřeba zaregistrovat všechny modely. Výhodou tohoto nastavení je to, že nebudeme muset vytvářet instance jednotlivých modelů, ani třídy connection, ale z presenteru budeme schopni volat pouze `$this->context->serviceName`, což nám vrátí konkrétní model. Nette takto dokáže zaregistrovat všechny modely díky tomu, že umí nalézt již definovanou službu typu, který dané třídy potřebují do konstruktoru a automaticky ji předá za nás (viz [12]).

Po nastavení konfiguračního souboru je ho ještě třeba aplikovat. K tomu slouží soubor `bootstrap.php`, ve kterém se postupně načtou všechna nastavení frameworku. Soubor se nahraje při přístupu na `index.php`, v kořenovém adresáři aplikace. Po načtení samotného Nette nejprve nastavíme, jestli aplikace poběží v produkčním nebo vývojovém prostředí. V závislosti na tom potom zapínáme ladící nástroje. Ve všech případech se následně nastaví složky `temp` a `log`, do kterých se budou zapisovat dočasné soubory a případné chyby. Nakonec se načte `config.neon` a z celého nastavení se vytvoří kontejner aplikace.

Poslední věcí před spuštěním aplikace je nastavení směrování, které nám vytváří strukturu odkazů. Nette díky tomuto nastavení vytvoří tzv. hezké url na základě parametrů, které předáme jednotlivým cestám. Př. cesty: `new Route('treninky/detail/<id_treninku>', "Treninky:detail");` Tento předpis říká, že pokud jsme v presenteru `Treninky` na šabloně `detail`, tak bude mít odkaz formát `'treninky/detail/1'`. Za posledním lomítkem je potom `id_treninku`, které je parametrem a bude místo něj vždy konkrétní hodnota. Pokud bychom zde neměli takovou cestu, tak by se nám vygenerovala adresa `presenter=treninky&action=detail&id_treninku=1`, která je na první pohled mnohem méně přehledná. Ve finále by to ale nebyl zas takový problém, protože se uživatelé na odkazy nedívají a v případě jejich potřeby je snadno zkopírují, ať už jsou hezké nebo ne. Po zaregistrování všech cest se spustí aplikace.

4.2 Registrace a přihlašování

Pro přístup do aplikace je vyžadováno přihlášení. Bez přihlášení se uživatel dostane pouze na výchozí stránku, kde je rozcestník pro registraci a následně přihlášení.

4.2.1 Registrace

Registrace probíhá v presenteru `RegistracePresenter`. Ten má k dispozici pouze výchozí šablonu `default.latte`. Základem je registrační formulář, který je definován v presenteru v metodě `createComponentRegistrace()`. Pro vytvoření formuláře se využívá třída `Nette\Application\UI\Form`. Ta nám poskytuje metody pro vytvoření formulářových polí. Mezi poskytované metody patří i přidávání validačních pravidel. Budeme od uživatele chtít všechny položky, které jsem popisoval v kapitole 3.2.1. Pro každou položku bude ve formuláři kontrolka. Většina položek jsou textové, až na výběr kraje, sportu a oddílu. Ty se načítají z databáze a vytvoří se z nich rozbalovací nabídka.

Poprvé jsem zde také využil Ajax. Oddíly se totiž načítají dynamicky v závislosti na výběru sportu. Může se stát, že v daném sportovním odvětví ještě oddíly nejsou vytvořeny a ne všichni sportovci do nich budou patřit, takže je toto pole samozřejmě nepovinné. Po vybrání sportu je vytvořen signál, který se asynchronně pošle serveru s dotazem na metodu `loadOddily($sport)`. Ta nám naplní kontrolku s oddíly novými daty a invaliduje ji, takže se uživateli otevře nová nabídka.

V následujícím kódu je vidět vytvoření rozbalovacího menu. Prvním parametrem je název kontrolky, přes který k ní můžeme přistupovat. Dalším je popisek a posledním jsou data, kterými

se má menu naplnit. Jedná se o konkrétní volání metody `findSporty` v modelu `SportyRepository`, která vrací pole, kde index je id sportu a hodnota jeho název.

```
$form->addSelect("sport", "Vyberte sport:", $this->context-
>sportyRepository->findSporty())
   ->setPrompt("--Vyberte sport--")
   ->addRule(Form::FILLED, 'Zvolte sport.')
   ->getControlPrototype()->class("ajax singleselect");
```

Další metody se dají jednoduše zřetězit, takže menu rovnou přidáme výchozí hlášku a validační pravidlo, že pole musí být vyplněno včetně hlášky, která se uživateli objeví při nedodržení. Nakonec přidáváme třídy pro kaskádové styly. V poslední řadě musíme formuláři nastavit funkci, která se vykoná po odeslání.

Ta jako parametr přejímá instanci odeslaného formuláře. V této metodě se nastaví asociativní pole s indexy odpovídajícími názvům sloupců v tabulce, který se jako parametr se jménem tabulky předá funkci `insert` a záznam se vloží. Na základě vybrané role se potom vloží ještě záznam do tabulky `treneri` nebo `sverenci`. Pokud vše proběhne v pořádku, tak se uživateli vrátí zpráva o úspěšném uložení zadaných dat. K tomu v Nette slouží flash zprávičky, které má k dispozici presenter. Po zavolání metody `flashMessage('Text zprávy')` se po opětovném načtení vypíše zpráva v poli, které je nastylované podle toho, jestli byl výsledek úspěšný či nikoliv. V metodě je odchytávána výjimka `PDOException`, která nastane při chybě komunikace s databází. Vyvolaná výjimka v sobě má kód chyby, která nastala, podle kterého je můžeme ošetřovat. Kontrolujeme, jestli nenastala chyba s číslem 1062, která znamená, že by v databázi vznikl duplicitní záznam. To může nastat pouze tehdy, když uživatel zadá již zaregistrovaný email. V tu chvíli je o přesném problému informován. Při výskytu jakékoliv jiné chyby je o ní uživatel informován obecně.

Grafická stránka formuláře je zpracována v šabloně `default.latte`. K vykreslení jsou využítá makra `form`, `label` a `input`. Makro `form` je párové a jeho parametrem je název komponenty, která se do kontejneru zaregistrovala metodou `createComponent<Name>`. Makra `label` a `input` jsou nepárová a parametr, který přejímají, je název formulářového pole zadávaný při definici formuláře. Makra jsou zasazena do HTML kódu, konkrétně do tabulky. Poslední makro v šabloně je `snippet`. Je párové a označují se jím části kódu, které je možné invalidovat po odeslání signálu (uživatelské akce). V tomto případě je jimi obalena kontrolka, která zprostředkovává výběr oddílu a blok JavaScriptu, který po invalidaci inicializuje styly rozbalovacích menu.

4.2.2 Přihlašování

`PrihlaseniPresenter` se stará o přihlašování. Jeho základem je přihlašovací formulář definovaný v metodě `createComponentSignInForm()`. Uživatel se přihlašuje pomocí zaregistrovaného emailu a hesla. Třetím políčkem, které se dá vyplnit je zaškrtávací pole, které rozhoduje, zda-li se má uživatel odhlásit po zavření prohlížeče či nikoliv.

Každý presenter má k dispozici volání metody `getUser()`, která vrací informaci o aktuálně přihlášeném uživateli. Pokud není uživatel přihlášen, tak se tak může jednoduše stát zavoláním `login($login, $heslo)` na objekt uživatele. Při úspěšném přihlášení se vytvoří objekt identity uživatele, s kterým je možno dále pracovat a my navíc zaznamenáme do databáze datum posledního přihlášení. Pokud uživatel zadá špatné heslo nebo přihlašovací údaj, který je v našem případě email, tak se vyvolá výjimka `AuthenticationException`. Tu lze zachytit a požádat uživatele o opakované přihlášení.

Vnitřně metoda `login` volá metodu `authenticate` z modelu `Authenticator`. Zde probíhá samotná kontrola zadaných dat s těmi, které jsou uloženy v databázi a případně se zde vyvolává výše zmíněná výjimka. Dále se při úspěšném ověření zadaných údajů nastaví uživatelské role, které budou po celou dobu přihlášení k dispozici. Každý uživatel má buď roli `trener` nebo `sverenec`. Další role se uživateli nastaví, pokud má záznam v tabulce `admin` a přidělená konkrétní práva ve vazební tabulce s tabulkou `admin_prava`. Metoda nakonec vrací objekt třídy `Nette\Security\Identity` s id uživatele, jeho rolemi a s případnými dalšími daty, po kterých chceme, aby byly snadno k dispozici. V našem případě je v těchto datech všechno, co o uživateli v tabulce uchováváme kromě hesla, dat registrace a posledního přihlášení a popisu.

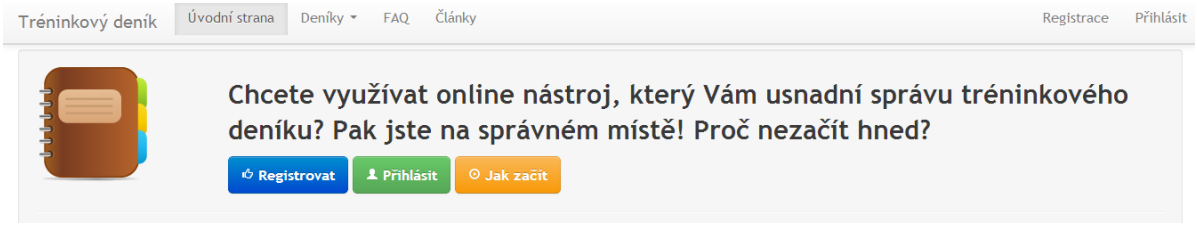
Šablona s přihlašovacím formulářem se nazývá `in.latte`. Je v ní jednoduše vykreslen stejným způsobem jako registrační formulář. Ve složce šablon přihlášení je ještě `out.latte`, která je sice prázdná, ale slouží nám k odhlášení uživatele. V presenteru to zajišťuje metoda `actionOut`. Učiní se to jednoduše podobně jako při přihlašování, a to zavoláním metody `logout` na objekt uživatele.

4.3 Úvodní strana

4.3.1 Nepřihlášení

Úvodní strana aplikace má dvě hlavní, rozdílné tváře. Obě závisí na tom, je-li uživatel přihlášen do systému nebo ne. Pokud není, tak se mu zobrazí uvítací stránka se stručným představením webu (viz Ilustrace 6). Uvidí zde několik obrázků z vnitřku aplikace, které

by ho mohly motivovat k registraci a dalšímu prozkoumání. Pod uvítací zprávou ho čeká seznam posledních devíti článků týkajících se samotného webu s možností přepnutí na seznam všech existujících článků. Pokud se uživatel rozhodne zaregistrovat, tak má k dispozici výrazné tlačítko, které ho přesměruje na registrační formulář, případně méně výrazný text se stejnou funkcí v pravé části horní lišty. To samé platí pro přihlašovací formulář, pokud se uživatel na stránky už vrací. Další možností, která není podmíněná registrací je prohlédnutí návodu, který uživatele krok po kroku provede od registrace až po založení tréninkové skupiny. Poslední možností je prohlížení často kladených otázek, které mu mohou pomoct v lepší pochopení případného problému.



Ilustrace 6: Horní menu a hlavní stránka nepřihlášeného uživatele

4.3.2 Přihlášení

Po přihlášení se uživateli zobrazí čtyři různě barevné dlaždice (viz Ilustrace 7). Každá z nich má informativní charakter a dohromady slouží jako rozcestník do aplikace. Po prvním přihlášení budou tři z nich určitě prázdné. Jediná naplněná dlaždice zobrazuje posledních deset vložených článků, ve kterých se dá kolečkem rolovat. O článcích bude více řečeno v kapitole 4.4.

Ilustrace 7: Barevně odlišené sekce na hlavní straně přihlášeného uživatele

První z ostatních tří má na starost zobrazení posledních deseti tréninků. K zobrazení je využit JavaScriptový nástroj z Twitter Bootstrap, kterému se říká „carousel“. Na pravé i levé straně dlaždice jsou šipky, kterými se uživatel dostane na předchozí či další záznam.

Vždy je tedy vidět pouze jeden. Aby nemusel pokaždé klikat na šipky, tak má k dispozici i seznam čísel, kde vidí aktivní záznam a po kliknutí na číslo se dostane na záznam odpovídající. Při absenci záznamů je uživateli doporučeno založení nového tréninku. V ostatních případech se poslední založené tréninky zobrazí s informací o konkrétní náplni tréninku, datu absolvování, seznamem účastníků a jménem a ikonou uživatele, který trénink vložil. O ukládání uživatelských ikon se sám nestarám a využívám k tomu službu Gravatar [13], kde si uživatel zaregistruje emailovou adresu a nahraje k ní obrázek, který se potom následně zobrazí na jakékoliv stránce, která tuto službu podporuje. Z každého přehledu tréninku se uživatel dostane do jeho detailu ke správě po kliknutí na datum.

Další z dlaždic funguje obdobným způsobem s tím rozdílem, že zobrazuje posledních deset záznamů vložených závodů. Všechny informace jsou stejné až na náplň tréninku, místo které je rozšířený seznam účastníků o disciplíny, které jednotliví uživatelé v daném závodě absolvovali. Navíc je zde ještě místo konání závodu.

Poslední z dlaždic má nejjednodušší, ale možná nejpřínosnější charakter. Jejím úkolem zobrazení náplně aktuálního dne. Pokud na den není žádný plán, tak se uživateli zobrazí, že má volno. V opačném případě se mu vypíše seznam naplánovaných záznamů s časy, které vedou na detailnější informace a konkrétní náplň. Pro jednoduchý přehled tedy nemusíme rovnou zobrazovat detail.

4.3.3 Realizace dlaždic

Každou dlaždici reprezentuje komponenta uložená v adresáři `app/components`. Šablony k jednotlivým komponentám jsou potom uložené v další podsložce, která se jmenuje stejně jako presenter. Každá komponenta má konstantu, ve které je uložená cesta k šabloně. Můžeme tak jednoduše vytvořit více šablon, které budou zpracovávat stejná data. Všechny dědí od třídy `Nette\Application\UI\Control`, mají konstruktor a metodu `render()`. V konstruktoru se předává objekt třídy `Connection` pro spojení s databází. Pomocí něj se nastaví atributy třídy. Ve všech případech potřebujeme pracovat s modelem uživatele a následně s modelem tréninků nebo závodů. V metodě `render()`, která se vykonává při vykreslování šablony se nejdřív metodou `createTemplate()` vytvoří objekt šablony a nastaví se jí konkrétní soubor. Vybereme tedy nastavenou konstantu. Zbývá pouze předání dat do šablony.

Data se však budou lišit podle toho, jestli je uživatel v tréninkové skupině nebo je jednotlivec a v případě skupiny, jestli je to svěřenec nebo trenér. Pokud je jednotlivec

(nemá trenéra), tak se všechna data budou vztahovat pouze k němu. Člen skupiny ale uvidí aktivitu všech ostatních členů včetně trenéra. Tato rozhodnutí učiníme v metodě `render()`.

V tuto chvíli jsou vytvořeny všechny komponenty a zbývá je pouze vykreslit v hlavní šabloně `HomepagePresenteru`. Slouží k tomu makro `widget`. Jeho použití je následující: `{widget treninkyCarouselControl}` – vykreslí seznam posledních tréninků. Parametrem je název komponenty. Poslední věcí je vytvoření tovární metody pro každou komponentu v presenteru. Tvoří se stejně jako formulář a to metodou `createComponent<Name>`. Bude mít jediný řádek a bude vracet instanci konkrétní komponenty. Od teď můžeme komponenty používat, protože jsou zaregistrovány v kontejneru aplikace. Příklad tovární metody pro vytvoření komponenty `treninkyCarouselControl` v presenteru:

```
protected function createComponentTreninkyCarouselControl() {
    return new TreninkyCarouselControl($this->db);
}
```

4.4 Články

Aby mohli být uživatelé informováni o novinkách a změnách týkajících se systému, tak jsem se rozhodl vytvořit jednoduchý modul pro vkládání článků. Lehká zmínka o článcích padla v minulé kapitole, kde se vypisovaly v komponentě na hlavní straně. V tuto chvíli bude řeč o tom, jak jsou reprezentovány v presenteru `ClankyPresenter`.

Využívají se zde pouze dvě šablony, a to výchozí `default.latte` a šablona `detail.latte`. Výchozí šablona se stará o výpis náhledů článků s odkazy na konkrétní detaily s informacemi o autorovi, datu, kategorii a počtu zobrazení. V detailu se potom dozvíme veškeré informace, které článek obsahuje. Na konci každého článku je několik nástrojů. Jedním z nich je tisk článků, který po kliknutí na ikonku tiskárny otevře v prohlížeči možnost tisku. Pro tisk se využije tisková šablona, která nezobrazí žádné obrázky ani formuláře. Další prvky mají na starost sdílení na sociálních sítích jako je Facebook nebo Google+, k čemuž se využívají volně dostupné nástroje těchto společností.

Vkládání článků není zahrnuto do tohoto presenteru, ale do administrace, kde vkládání probíhá. Pro tento případ je potřeba jednoduchý formulář s poli pro titulek, kategorii a samotný text článku. Pro kategorie článků jsem nevytvořil žádnou samostatnou tabulku z toho důvodu, že se budou týkat pouze konkrétních sportů a obecných záležitostí, jako jsou návody nebo obecné informace týkající se změn na serveru apod.

Informace z databáze se získávají skrze `ClankyRepository` model. Stačí nám zde pouze pár metod pro vrácení všech záznamů, všech záznamů v kategorii a jednoho článku podle jeho id pro zobrazení detailu. Protože vedeme informaci o tom, kolik lidí si článek zobrazilo, tak máme další metodu, která navyšuje počet zobrazení o jedna. Poslední metoda vrací pole kategorií, které může uživatel při psaní článků využít. Závisí to především na jeho roli, přidělených právech a sportovním odvětví, v kterém je zaregistrován. Administrátor se všemi právy tak může psát články ve všech obecných kategorií, které jsou v modelu fixně definované a uživatel s oprávněními pro psaní článků ve sportu pouze ve svém sportu.

4.5 Tréninky

Tréninky patří mezi jednu ze tří největších částí aplikace. Tato sekce obsahuje všechny nástroje pro vkládání, editaci a mazání všech záznamů. Samozřejmostí je také přehledně zpracovaný výpis pro různé situace. Ke všem položkám se ve webovém rozhraní dostaneme přes položku „Deníky“ v horním menu. Hlavní akce probíhají v presenteru `TreninkyPresenter`.

4.5.1 Vložení tréninku

Šablona, která vykresluje vložení tréninku se nazývá `novy.latte`. Dostaneme se do ní přes možnost „Deníky – Nový trénink“ nebo v seznamu tréninků kliknutím na tlačítko „nový trénink“ v příslušném dni. Její obsah se liší na základě uživatelské role. Svěřenec i trenér mají rozdílné formuláře. Pokud trénink zakládá svěřenec, tak může zadat všechny informace (viz 3.2.2) v příslušných polích, které se zapisují do entity `treninky`. Navíc si může rovnou vyplnit poznámku a označit trénink jako splněný, pokud už ho absolvoval a teprve ho neplánuje. Další možností je výběr ukazatelů, které ovšem ještě nemůže vyplnit. K tomu má k dispozici nástroje v detailu tréninku (viz 4.5.3). Nakonec si ještě vybere trenéra, se kterým trénink absolvoval a nebo nechá pole nevyplněné, což znamená, že nebude navázaný na trenéra a ve skutečnosti šel např. něco nad plán nebo je to jednotlivec, který trenéra nemá.

Trenér vyplňuje pouze obecné údaje k tréninku. K nim patří i výběr ukazatelů a nesmí zde chybět výběr svěřenců. Nelze vytvořit trénink, ke kterému by nebyl nikdo přiřazen, takže pokud na to trenér zapomene, bude mu oznámeno, že musí učinit výběr. Stejně tak se nesmí stát, že trenér přiřadí trénink někomu na datum a hodinu, kde už má jiný záznam. Formulář zabírá levou polovinu stránky a na té pravé je nástroj pro zobrazení tréninků svěřenců v aktuálním týdnu. Nad tímto seznamem je navigační lišta s názvy dnů pro jejich přepínání a šipkami pro přepínání mezi týdny. Trenér tak při zakládání vidí, co svěřenci dělali minulý týden a dokáže tak lépe a snadněji vymyslet nový trénink, aniž by musel opouštět editační formulář. Pokud

se stane, že některý z přiřazovaných svěřenců má již v daný čas trénink, vypíše se trenérovi jeho jméno ve varovné hlášce (pokud jich je víc, vypíšou se všechna jména). V tuto chvíli se může rozhodnout pro změnu data, pokud se spletl a nebo může vypsaným svěřencům okamžitě zrušit přiřazení předchozího tréninku a vložit ho znovu na stejné datum. Ve výpisu účastníků tréninku na pravé straně je vždy jeho jméno s křížkem, který automaticky maže přiřazení.

O vložení tréninku se stará metoda v modelu `TreninkyRepository` `insertTrenink($treninkInfo)`. Předávají se jí informace z formuláře. Celá metoda běží v transakci, takže pokud nastane chyba v kterémkoliv dotazu na databázi, všechny provedené změny před chybou se vrátí zpět. Nemůže se tak stát, že by se vytvořil trénink bez přiřazených svěřenců. Před uložením přiřazení se zjišťuje, jestli nemá příslušný uživatel v daném datu jiný záznam. Pokud ano, tak se přiřazení nevykoná a zaznamenáme si do pole jeho id a jméno. Metoda vrací asociativní pole s id nově vytvořeného tréninku a seznamem svěřenců, kteří mají obsazený čas. Pokud ale nastane odchytávaná chyba `PDOException`, tak se ve výsledku vrátí její objekt. Po vložení tréninku a všech přiřazení se už jen kontroluje nastavení uživatelů (viz 4.7.1), jestli mají zapnutou emailovou notifikaci. Pokud ano, odešle se jim email s informacemi o novém tréninku.

4.5.2 Seznam tréninků

Seznam tréninků má několik možností zobrazení. Opět je rozdíl mezi svěřencem a trenérem, ale navíc ještě mezi týdenním a měsíčním zobrazením. Kromě měsíčního zobrazení trenéra se na všechny typy zobrazení dostaneme přes položku „Můj deník“ v sekci „Deníky“. Šablonou pro seznam je `seznam.latte`.

V roli svěřence se kliknutím na tuto položku automaticky dostaneme na aktuální týdenní výpis (viz Ilustrace 8). Každý den představuje jednu položku, která je označena datem a názvem dne. Svěřenec má možnost založit nový trénink v každém dni kliknutím na tlačítko „Nový trénink“, které je hned vedle data. Příslušné datum bude při vytváření tréninku automaticky předvyplněné. V ní je potom seznam jednotlivých záznamů v deníku, což mohou být tréninky nebo závody. Každý box s tréninkem je buď označen jako splněný nebo nesplněný. Rozdílem je barva jejich pozadí a ikony, která uvozuje řádek. Mimo to je zde výpis tréninku, u vyplněného i svěřencova poznámka a dále tři tlačítka. Jedno z nich uživatele přesměruje na vyplnění do detailu tréninku, druhé po kliknutí označí trénink jako splněný nebo nesplněný a to třetí vymaže uživatelovo přiřazení tréninku z databáze. Na vrchu stránky je vždy filtrační formulář, který umožňuje přepínat mezi jednotlivými týdny v roce nebo měsíčním a týdenním

zobrazením. Je tu také ikona tiskárny, která stejně jako u článků přepne stránku do tiskového stylu a otevře možnost tisku prohlížeče.

Ilustrace 8: Ukázka dvou dnů z týdenního výpisu svěřence

Měsíční zobrazení je kalendář se všemi dny, které mohou nabývat tří barev. Zelený den je takový, ve kterém je alespoň jeden záznam, žlutý ukazuje, že v daném dni žádný záznam není a červená dlaždice značí jiný měsíc. Každý záznam ve dni je označen ikonou totožnou z týdenního výpisu a časem. Po kliknutí na čas se uživatel dostane na detail tréninku.

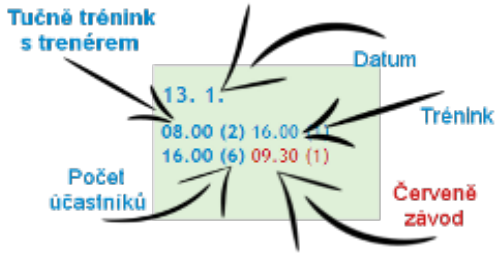
Ilustrace 9: Ukázka z týdenního přehledu trenéra

V roli trenéra má týdenní přehled stejný formát jako u svěřence (viz Ilustrace 9). Rozdílem je neutrální barevné pozadí všech záznamů. Tréninky také nejsou označeny žádnou ikonou a trenér také samozřejmě nemá co na žádném z tréninků nastavovat. V jednom dni je tedy seznam tréninků všech svěřenců, aby měl o všem jednotný přehled. O každém tréninku na první pohled vidí jeho čas, náplň a účastníky včetně informace o tom, jestli ho daný svěřenec splnil

nebo ne. Hodina tréninku ho vždy přesměruje na jeho detail, stejně tak jako jméno účastníků na jejich veřejný profil.

Měsíční zobrazení trenéra je realizováno v šabloně `prehledTren.latte`. Je to z toho důvodu, že tuto šablonu mohou využít i svěřenci pro zobrazení měsíce kteréhokoliv ze svých trenérů. Filtrační formulář zde má tedy navíc kolonku pro výběr trenéra.

Dlaždice se zabarvují stejným principem jako u svěřence. Rozdílem je, že z každého data je odkaz na přehled celého dne, protože se může stát, že je v některém dni tolik záznamů, že to uživateli nic kromě této informace nepřinese. Přehled dne je potom to samé jako jeden konkrétní den v přehledu týdne trenéra. Každá hodina, na kterou je nějaký plán má v závorce počet účastníků. Tučně modré hodiny jsou tréninky naplánované trenérem, modré bez ztučnění jsou tréninky svěřence bez trenéra a červeně jsou označeny závody, jak je vidět z Ilustrace 10.



Ilustrace 10: Popis dlaždice v přehledu měsíce trenéra

Data týdenní přehledu se vykreslují z asociativního pole, které v případě svěřence sestavuje funkce `getSverenecWeek` a trenéra `getTrenerWeek`. Formát pole je následující:

```
datum => array(
        hodina => array(
             [0] => Nette\Database\Row,
             [1] => array(
                  "zavod" => Nette\Database\Row,
                  "discipliny" => Nette\Database\Statement
             )
        )
)
```

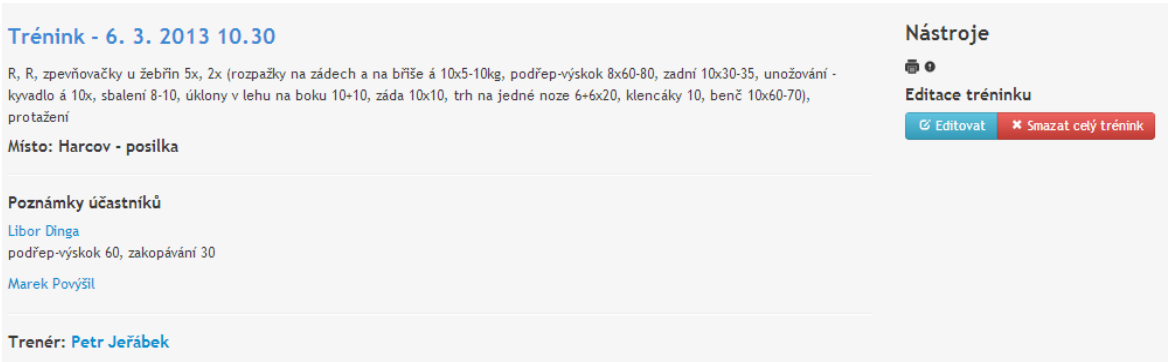
Každý datum tedy obsahuje pole časů, které obsahuje pole záznamů. V tomto příkladě je na indexu 0 objekt `Nette\Database\Row`, což znamená, že se jedná o trénink a můžeme přes něj rovnou vypisovat všechny atributy tréninku. Na indexu 1 je závod, který je vždy fixní pole s položkami `zavod` a `discipliny`. `zavod` je potom stejný případ jako trénink

a „discipliny“ jsou objekt `Nette\Database\Statement`, v kterém je každá položka při průchodu cyklem `foreach` typu `Nette\Database\Row`.

V presenteru jsou obdobné i metody pro zobrazení měsíce, a to `getTrenerMonth` a `getSverenerMonth`. Struktura je vždy obdobná a obsahuje pouze jiné informace.

4.5.3 Detail tréninku

Detail tréninku je v šabloně `detail.latte`. Rozdíly mezi rolemi zde spočívají pouze v možnosti použití různých nástrojů. Přehled detailu vypadá tak, že jsou zobrazeny veškeré informace o tréninku jako je datum, náplň a místo konání. Pod těmito informacemi je potom seznam všech účastníků, kteří pod svým jménem mají poznámka k tréninku, pokud ji vyplnili (viz Ilustrace 11). Poslední údajem je jméno trenéra, který trénink naplánoval. Trénink je označen jako bez trenéra, pokud k němu žádný přiřazen není.



Ilustrace 11: Detail tréninku z pohledu trenéra

Pravá strana stránky je nazvána nástroje. Zde je opět možnost tisku, která funguje stejně jako předchozí možnosti tisku. Pokud se na detail tréninku dostane svěřenec, který ho nemá v deníku, ale absolvoval ho, tak má v nástrojích dvě možnosti. Jednou z nich je přidání tréninku a tou druhou přidání a zároveň splnění, což využije ve chvíli, kdy už trénink absolvoval. Pokud je k němu tedy svěřenec přiřazen, tak se mu otevřou nové možnosti. Z možností vyplnění je to tlačítko pro zrušení toho, že je trénink splněný, smazání přiřazení a otevření formuláře pro vyplnění. Je tu také editační nástroj, který svěřenci dovoluje přiřadit k tréninku tréninkové ukazatele. Trenér a svěřenec, který trénink založil má navíc možnost editace všech vypsaných informací a navíc ještě kompletní smazání celého tréninku z databáze včetně přiřazení všech svěřenců.

Všechny formuláře jsou zpočátku skryté a zobrazí se v překryvném okně teprve po kliknutí na příslušné tlačítko. Využívá se k tomu jQuery a HTML5. Tlačítko má nastavenou css třídu `openDialog` a atribut `data-dialog`. HTML tag `fieldset`, který obsahuje samotný formulář musí mít css třídu `jqueryDialog` a `id` shodné s atributem `data-`

`dialog` tlačítka, které ho má otevírat. Inicializace překryvného dialogového okna se v jQuery provede zavoláním `$(".jqueryDialog").dialog(dialogDefault)`, kde parametr `dialogDefault` je naše výchozí nastavení pro tvorbu dialogů. Každý element s třídou `jqueryDialog` je nyní inicializován jako dialogové okno. Následuj kousek kódu, který inicializuje všechna tlačítka s třídou `openDialog`.

```
$(".openDialog").click(function() {
    $("#" + $(this).attr("data-dialog")).dialog("open")
    .find("input[type='submit']").click(function() {
         $(this).closest(".jqueryDialog").dialog("close");
    });
});
```

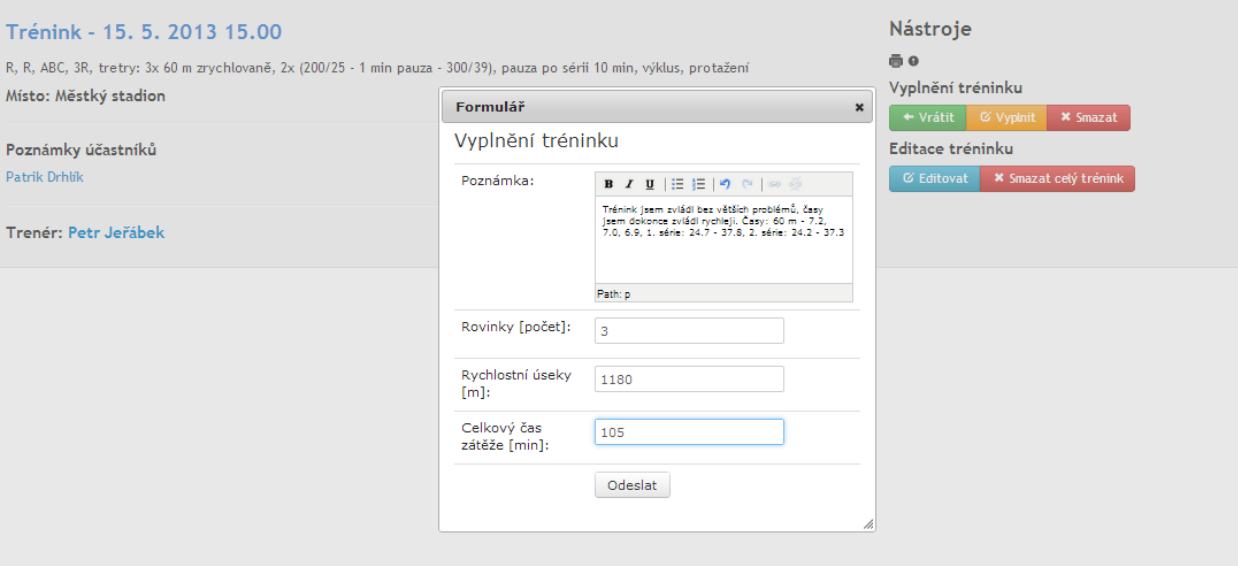
Po kliknutí na tlačítko se zavolá metoda `.dialog("open")` na objekt s `id` podle `data-dialog` atributu tlačítka. V nalezeném formuláři se ještě nalezne tlačítko pro odesílání a nastaví se mu, že se dialogové okno po kliknutí zavře metodou `.dialog("close")`.

4.5.4 Editace a vyplnění tréninku

V předchozí kapitole jsem popisoval, jaké jsou nástroje pro práci s tréninkem a jaký typ uživatele má ke kterým přístup. Splňování a mazání tréninku funguje stejně jako v přehledu týdne. Pokud svěřenec zruší přiřazení tréninku a je jeho posledním členem, tak se z databáze smaže i záznam o samotném tréninku.

Vyplnění tréninku už zastává jinou funkci. Otevře dialogové okno s formulářem pro jeho vyplnění (viz Ilustrace 12). Tento formulář se od jiných liší především tím, že se jeho pole vytvářejí dynamicky. Pouze poznámka a odesílací tlačítko jsou statické. Vyplňují se zde totiž ukazatele, které svěřenec nemusí mít vůbec přiřazené a nebo jich může mít přiřazeno jen několik, což je nejčastější případ. Aby se zbytečně nevytvářela pole pro všechny ukazatele, tak se při sestavování formuláře v presenteru v továrničce `createComponentVyplnitTrenink()` nejdřív získá seznam ukazatelů, které má uživatel k tréninku přiřazené a dynamicky se vytvoří pole pro každý z nich.

```
$ukazatele = $this->treninky->findUserTrainingMarks($idTr, $id);
foreach ($ukazatele as $uk) {
    $form->addText("uk" . $uk->id, $uk->nazev . " [" . $uk-
>jednotka . "]:");
    $defaults += array("uk" . $uk->id => $uk->hodnota);
}
```



Ilustrace 12: Ukázka vyplnění tréninku včetně ukazatelů

Jako první se naplní proměnná `$ukazatele` metodou z modelu `TreninkyPresenter` a vrátí požadovaná data. Foreach cyklem se projdou všechny záznamy a pro každý z nich se přidá textové pole s popiskem ve formátu „název ukazatele [jednotka]:“. Do pole `$defaults` se následně přidá hodnota z databáze a pole se potom předá formuláři, aby příslušné hodnoty vypsal.

Jedinou editaci tréninku, kterou může provést svěřenec, který trénink nezaložil, je nastavení tréninkových ukazatelů a slouží k tomu jednoduchý formulář `editaceUkazatelu`. Trenéři a zakladatelé tréninků využívají formulář `editaceTreninku`, v kterém lze upravit všechno od tréninkové náplně, přes datum až po tréninkové ukazatele pro všechny jeho účastníky.

4.6 Závody

Další velkou a důležitou částí aplikace jsou závody. Umožňují si vézt záznamy o absolvovaných soutěžích, výkonech v jednotlivých disciplínách apod. Všechny vyplněné údaje se následně dají zúročit při generování statistik dostupných z profilu uživatele (viz 4.7.4).

4.6.1 Vložení závodu

Obdobně jako při vkládání tréninků je formulář vykreslen v šabloně `novy.latte`. Přidávat závody mohou jak svěřenci, tak trenéři a všechny položky formuláře jsou pro obě role naprosto totožné. O samotném závodu se drží informace pouze o názvu, datu, místě a propozicích. Abychom předešli vytváření duplicitních záznamů, tak se v pravé části stránky vždy po výběru data objeví informace o tom, zda-li už v daném dni není uložený jiný závod. Pokud není, objeví se hláška s výzvou v pokračování vytváření závodu. Pokud však je,

tak se nám vypíše seznam závodů, které jsou ten den k dispozici a pokud je v seznamu ten, který chce uživatel založit, tak už nemusí a rovnou se kliknutím na název závodu dostane na jeho detail. Jednou ze zajímavostí je, že se pod výpisem vytvořených závodů objeví odkaz u sportu, v kterém je uživatel zaregistrován, který vede na stránku daného sportu s kalendářem závodů. Tato funkce zatím funguje pouze u sportu atletika, která má na svých stránkách atletika.cz kalendář, do jehož url můžeme předat datum, které nám bude zobrazeno. Další sporty se musí nastavit na žádost uživatelů.

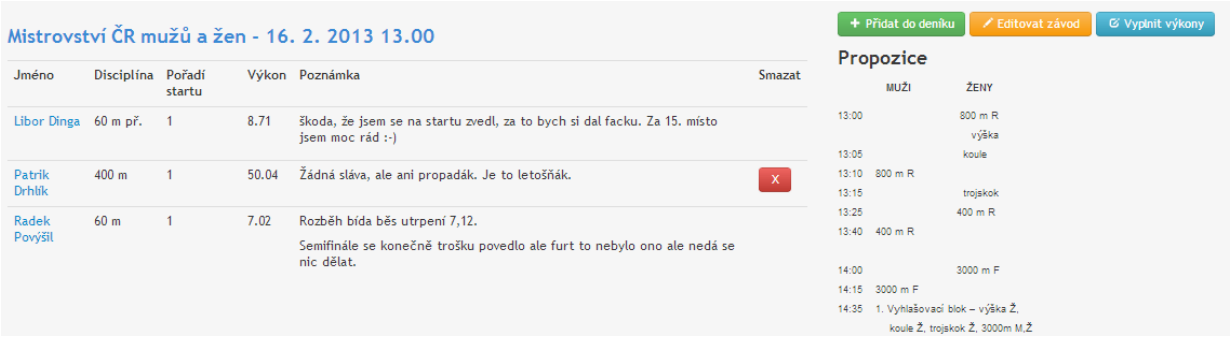
Když je závod přiřazen, stačí si ho už jen přidat do deníku s konkrétní disciplínou. Tato editace se dělá v detailu závodu. Svěřenec má k dispozici formulář, který nabízí výběr disciplín. Má také možnost zvolit více stejných disciplín v jednom závodě, u nichž se bude lišit pouze pořadí startu. Jedná se o případ, kdy např. v atletice sportovec ve stejný den absolvuje kvalifikaci a finále. Trenér má ve formuláři navíc přítomnou položku pro výběr svěřence, aby mu mohl disciplíny naplánovat.

4.6.2 Kalendář závodů

Kalendář závodů je obdobou měsíčního přehledu tréninků. Grafické zobrazení je naprosto totožné. Rozdílem je však to, že se v jednotlivých dnech se záznamy závodů zobrazují jejich názvy. To jsou zároveň odkazy, které vedou na jeho detail. Struktura, která se vykresluje je sestavena funkcí `getZavodyMonth`. Zobrazení je zde stejné pro obě uživatelské role. V kalendáři se tedy všem zobrazují závody, které založil kdokoliv z tréninkové skupiny včetně trenéra.

4.6.3 Detail závodu

Detail závodu se vykresluje v šabloně `detail.latte`. Na první pohled je vidět název závodu s datem a časem konání. Levá strana obsahuje tabulku s jednotlivými přiřazeními. Každé přiřazení nese informaci o jméně závodníka s odkazem na jeho veřejný profil. Další důležitou



Ilustrace 13: Ukázka detailu závodu s přihlášenými závodníky

informací je disciplína, které se zúčastnil, pořadí startu disciplíny, dosažený výkon a poznámka obsahující pocity a jiné doplňující informace o startu. Pokud je uživatel přihlášen na detailu

závodu, který absolvoval, tak má navíc na konci řádku možnost přiřazení smazat pro případ, že se spletl při vkládání apod. Pravá strana obsahuje uložené propozice s bližšími informacemi o závodu (viz Ilustrace 13).

V této tabulce se vypíší všichni závodníci, kteří závod absolvovali a patří do stejné tréninkové skupiny jako uživatel, který závody prohlíží. Do šablony se předávají proměnné `$zavod`, `$canEdit`, `$pocet_zavodniku` a `$zavodnici`. `$zavod` obsahuje informace o závodu na základě jeho `id`. `$canEdit` je proměnná typu `boolean`, která je `true` v případě, že je aktivní uživatel přihlášen v závodě nebo má přihlášeného svěřence (pokud je trenér). Pokud je `true`, tak uživatel může upravovat detaily tréninku.

Obdobně jako u tréninků jsou zde na pravé straně stránky nástroje. Nepřihlášený uživatel má jedinou možnost, a to přidání závodu do deníku. Po přidání mu tato možnost zůstává, aby své přihlášené disciplíny mohl později doplnit. Přibude mu navíc možnost editace závodu a vyplnění výsledků.

4.6.4 Editace a vyplnění závodů

O tom, kde se nachází editační nástroje a komu jsou zpřístupněné jsem psal v předchozí kapitole. Patří mezi ně přidávání disciplín do závodu, editace závodu a vyplňování, v případě svěřence.

O přidávání se stará formulář v továrničce `createComponentPridatZavod()`. Ten má dvě různé podoby. Obě role mají možnost přidání disciplíny a trenér má navíc možnost výběru svěřence, kterému chce disciplínu přiřadit. Při odeslání formuláře se kontroluje vloží záznam a odchytává se výjimka `PDOException`. Pokud je odchycena s kódem 1062, tak to znamená, že má svěřenec uložený záznam se stejnou disciplínou. Aby nemusel sám vyplňovat pořadí startu, tak se to vyřeší v `catch` bloku odchytávání. Z databáze se vybere uložené pořadí startu disciplíny uživatele v závodě a inkrementuje se o jedničku.

Závod se edituje prostým formulářem s editačními poli pro každou editovatelnou položku a je stejný pro obě role. Továrnička pro formulář je v metodě `createComponentEditaceZavodu()`.

Poslední formulář je k dispozici pouze pro svěřence, protože obstarává funkci vyplňování přihlášených disciplín. Ze tří formulářů je tento rozhodně nejzajímavější z toho hlediska, že se jeho pole generují dynamicky. V tovární metodě `createComponentVyplnitZavody()` se po vytvoření instance formuláře načtou z databáze všechny disciplíny, na které je svěřenec přihlášen. Potřebujeme získat poznámku

z přiřazení závodu včetně výkonu, názvu disciplíny a pořadí startu. Ke každé disciplíně navíc z tabulky `format_disciplin` vybereme položky `hlaska` a `format`. `format` obsahuje regulární výraz, který omezuje formát výkonu a `hlaska` obsahuje text, který uživatele upozornění na to, v jaké formátu se má výkon v dané disciplíně ukládat, pokud se splete nebo se ho pokusí uložit jinak.

Po získání dat z databáze se v cyklu `foreach` projdou jednotlivé záznamy a do formuláře se pro každou z nich přidají dvě položky. První z nich je pole pro zadání výkonu a tou druhou je poznámka k výkonu. Názvy formulářových polí musí být unikátní, takže se dynamicky generují do formátu: „disc_idDiscipliny_poradiStartuDiscipliny“ a popisek pole je složen z názvu disciplíny a pořadí jejího startu. Stejným způsobem je složen název pole pro zadání poznámky, akorát se místo uvozujícího slovíčka „disc“ použije „pozn“. Obě pole mají vždy nastavenou výchozí hodnotu jako data, která byla uložena dříve. Pole pro zadávání výkonu má navíc validační pravidlo, které odeslanou hodnotu porovnává s regulárním výrazem, uloženým v databázi.

Odeslání takto dynamicky sestaveného formuláře není prosté jako u těch statických. Předem totiž nevíme, jaké hodnoty nám z formuláře přijdou.

```
foreach ($values as $key => $val) {
   $i++;
   if ($i % 2 == 0) {
        $id = explode("_", $key);
$query .= " poznamka='" . $val . "' WHERE idSv='" . $this->id .
"' AND idDisc='" . $id[1] . "' AND idZav='" . $this->idZav . "'
AND poradi_startu='" . $id[2] . "'";
        $this->db->query($query);
   } else {
        $query = "UPDATE prirazeni_zavodu SET vykon='" . $val .
"', ";
   }
}
```

Proměnná `$values` je převzatá z odeslaných dat formuláře. Je to asociativní pole, jehož klíčem je název formulářového pole a hodnotou vyplněná hodnota. Díky tomu, že víme, jaký je formát názvů, tak můžeme postupně konstruovat SQL příkazy a posílat je databázi ke zpracování. Sestavení příkazu je rozděleno do dvou kroků, kdy se v prvním vždy nastaví výkon a ve druhém poznámka s podmínkami, které specifikují, jaký řádek se upraví a nakonec následuje samotné odeslání.

4.7 Profil

Poslední z hlavních sekcí je uživatelský profil. V horní navigační liště se uživatel dostane do nabídky profilu. Obsahuje několik podsekcí – soukromý profil s možnostmi nastavení účtu, veřejný profil s informacemi dostupnými pro ostatní uživatele, statistiky tréninků a závodů a možnosti, jak vytvořit tréninkovou skupinu.

4.7.1 Soukromý profil a nastavení

Soukromý profil využívá systém záložek, kde každá z nich má přiřazený obsah a ten se zobrazí pouze pokud je příslušná záložka aktivní. Využívá se k tomu plugin z Twitter Bootstrap zvaný `tabs`. První z nich je editace údajů. Obsahuje jednoduchý formulář na změnu jména, příjmení, emailu, kraje a oddílu. Formulář je statický a funguje stejným způsobem jako dříve popsané statické formuláře. Další záložkou v pořadí je změna hesla. Patří také k editaci údajů, ale je pro ní oddělený jednoduchý formulář. Pro změnu hesla musí uživatel zadat své správné staré heslo a nové heslo, které musí ještě potvrdit v kolonce ověření nového hesla. Všechny položky jsou samozřejmě povinné. Zajímavostí ověření hesla je použití validačního pravidla, které kontroluje obsah předchozího pole s novým heslem a v případě, že se nerovnají, tak se formulář neodešle. Třetí záložka obsahuje další formulář, který má jediné velké textové pole pro editaci osobního popisu, který si může kdokoliv přečíst na uživatelově veřejném profilu.

Na záložce ukazatele jsou dvě velké interaktivní tabulky pro individuální správu ukazatelů a jejich skupin. Jedna tabulka je seznamem ukazatelů (viz Ilustrace 14) a druhá seznamem skupin ukazatelů. Obě mají svůj předpis v samostatné šabloně v adresáři `app/templates/Profil/default`. Každá má možnost přidání nového záznamu po kliknutí na tlačítko, kdy se následně objeví formulář v dialogovém okně stejným principem jako např. formuláře v detailu tréninku. Tabulka ukazatelů má položky název, skupina, jednotka, popis, skupinový a smazat. Položka skupina je rozbalovací menu s vyznačenou skupinou ukazatele. Změnou položky se okamžitě změní záznam v databázi Ajaxovým voláním. Editace ostatních položek probíhá po kliknutí na buňku tabulky. Změní se v textové pole předvyplněné původní hodnotou, kterou můžeme změnit a stisknutím klávesy enter nebo kliknutím mimo textové pole se záznam uloží. Toto nefunguje pouze u položky „skupinový“, která pouze říká, jestli je ukazatel skupinový nebo ne. Pokud není, tak to znamená, že je soukromý a může ho používat pouze svěřenec, který ho vlastní. V opačném případě může ukazatele využívat celá skupina. Položka smazat obsahuje tlačítko, které se uživatele po stisknutí zeptá, jestli

si je opravdu jistý odebráním ukazatele. Po potvrzení se záznam smaže a provede se aktualizace obsahu tabulky.

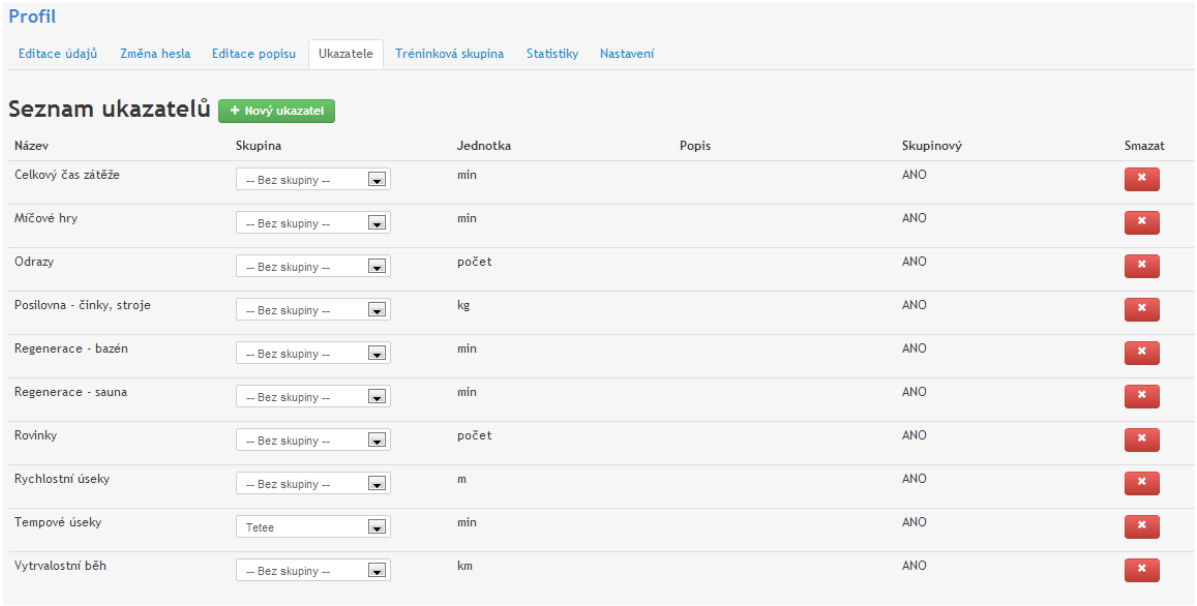
Seznam skupin ukazatelů funguje naprosto totožným způsobem. Jediné, co zde oproti ukazatelům není, je logicky skupina ukazatelů. K editaci polí využívám vlastní jQuery plugin zvaný `makeInput()`. Aby editace fungovala, je potřeba, aby každá editovatelná buňka měla příslušné HTML data atributy. Jedním z nich je atribut `data-input`, jehož hodnota je HTML tag elementu, na který se má pole proměnit po kliknutí. Tou druhou je atribut `data-column`, který obsahuje název sloupce v tabulce, který se má editovat. Každý řádek tabulky potom musí mít data atribut se jménem primárního klíče tabulky a jeho hodnotou, abychom byli schopni záznam upravit. Ukazatele mají atribut `data-id-ukazatele` a skupiny `data-id-skupiny-ukazatelu`. Parametrem pro plugin je objekt v JSON formátu s položkou `classes`, která obsahuje css třídy, které se předají vytvořenému formulářovému poli a položkou `callback`, což je funkce, která se vykoná po ukončení editace položky.

```
$("tr td[data-column]").click(function(e) {
   var markId = $(this).closest("tr").attr("data-id-ukazatele");
   var groupMarkId = $(this).closest("tr").attr("data-id-
skupiny-ukazatelu");
   if (groupMarkId == null) {
        var callback = function(column, value) {
             $.get("?do=editTrainingMark", {
                  "markId" : markId,
                  "column" : column,
                  "value"    : value
             });
        }
   }
   if (markId == null) {
        callback = function(column, value) {
             $.get("?do=editTrainingMarkGroup", {
                  "markGroupId" : groupMarkId,
                  "column" : column,
                  "value"    : value
             });
        }
   }
   $(this).makeInput({
        "classes"  : "ajax",
        "callback" : callback
   });
});
```

V předešlém kusu kódu je vidět přiřazení editovatelnosti všech buňkám tabulky, které obsahují atribut `data-column`. Abychom pokryli editovatelnost obou tabulek,

tak si zaznamenáme id ukazatele a id skupiny ukazatelů. Podle toho, které bude mít hodnotu poznáme, v jaké jsme tabulce. Na základě toho také nastavíme `callback` funkci do proměnné. Jedná se o jQuery funkci `$.get`, která volá skript na serveru a znovu načte části kódu, které jsou v šabloně obalené makrem `{snippet nazevSnippetu}`. Volané metody na serveru pouze vykonávají SQL dotaz update na záznamu s předaným id.

Položka tréninková skupina vypisuje v případě skupinové příslušnosti všechny ostatní kolegy s jejich ikonami a jmény vedoucími na jejich veřejný profil. Položka statistiky potom zobrazuje několik zajímavých čísel jako je počet tréninků, počet splněných a nesplněných a počet závodů a startů v závodech.



Ilustrace 14: Ukázka editačního rozhraní ukazatelů v soukromém profilu

Poslední záložkou v profilu je nastavení. Je zde tabulka zabírající polovinu stránky se seznamem nastavení, které jsou uloženy v tabulce „nastaveni“. Uživatel může nastavení aplikovat označením zaškrtávacího pole a následným odesláním pomocí tlačítka. Každé nastavení má možnost zobrazení detailnějších informací o tom, co které nastavení dělá po kliknutí na otazník vedle zaškrtávadla.

4.7.2 Veřejný profil

Šablona veřejného profilu je `public.latte`. Mimo články a hlavní stranu je toto další sekce, kterou může navštívit libovolný návštěvník aplikace. Stránka je rozdělena do dvou částí. V levé části je tabulka se všemi základními informacemi o uživateli včetně data registrace, jeho role, ikony, přiřazených trenérů a odkazu na jeho tréninkový plán. Ten jsou ovšem schopni vidět pouze uživatelovi kolegové ve skupině nebo jeho trenéři. Na pravé straně je potom zobrazen uživatelův osobní popis a představení, které si může vyplnit ve svém soukromém profilu.

4.7.3 Vytvoření skupiny

Sdružování uživatelů do skupin patří také do sekce profil. Obě role pro tuto funkci mají stejný postup zaměřený na opačnou roli. Aby se nestalo, že si svěřenec přidá spoustu trenérů, kteří ho ani nemusí znát, tak jsem do aplikace zavedl systém žádostí. To je také jedna z položek v profilu. Nazývá se žádosti a má za názvem v závorce číslo počtu aktuálních žádostí, které nebyly potvrzeny nebo odmítnuty. V šabloně `zadosti.latte` je dlaždicový výpis všech takových žádostí se jménem žadatele, který je rovněž odkazem na uživatelův profil, kde si můžeme zkontrolovat, zda-li se opravdu jedná o člověka, se kterým chceme vytvořit skupinu. Mimo jména jsou zde dvě tlačítka, kde první z nich potvrdí členství ve skupině nastavením sloupce „potvrzeno“ v tabulce skupiny a druhé dosud nepotvrzený záznam z tabulky vymaže. V obou případech žádost ze seznamu zmizí a pokud již v seznamu žádná není, vypíše se o tom informace.

Svěřenec posílá žádosti přes položku „Najít trenéra“. Ta je v šabloně `naleztTrenera.latte`, je rozdělena na dvě části a obsahuje dva formuláře. Na levé straně je filtrační formulář pro nalezení trenéra v případě, že je už zaregistrovaný v aplikaci. Můžeme trenéry filtrovat podle sportu a kraje. Ty se budou podle zadaných kritérií automaticky filtrovat ve třetí položce se seznamem trenérů. Pokud je přítomen požadovaný trenér, stačí odeslat kliknutím na potvrzovací tlačítko. Pokud jsme ale hledanou osobu nenašli, máme k dispozici druhý formulář, do kterého zadáme jeden, či více emailů oddělených čárkami a zprávu, kterou chceme poslat. Vygeneruje se email a na zadané adresy se odešle pozvánka s odkazy na registraci do aplikace, po níž bude uživatel k nalezení v předchozím formuláři.

Stejnou možnost má samozřejmě i trenér, a to v sekci „Najít svěřence“. Formuláře vypadají úplně stejně, jen se filtruje v seznamu svěřenců. Při odesílání emailu je použita jiná šablona.

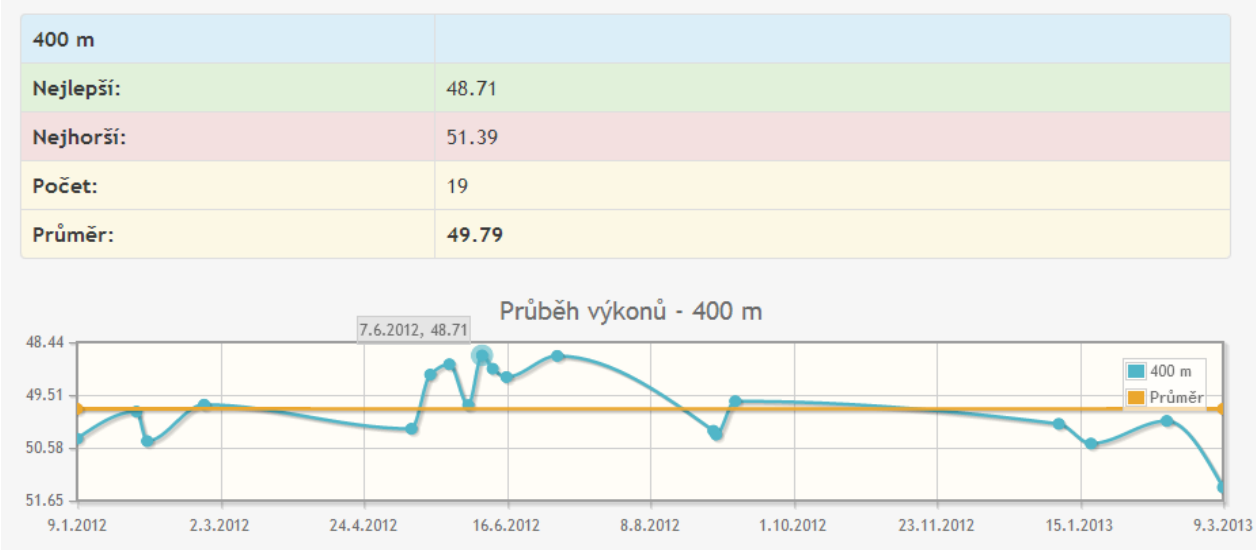
4.7.4 Statistiky závodů

Statistiky jsou pro mnohé sportovce velmi zajímavou věcí, obzvlášť, pokud to někdo nebo něco udělá za ně a ještě jim to hezky graficky zobrazí. To přesně bylo mou snahou při tvorbě této sekce. Rozdíl mezi trenérem a svěřencem je pouze ten, že trenér si ve filtraci závodů může zadat i svěřence, jehož statistiky chce vidět. Svěřenec může sledovat pouze svoje statistiky. Mimo to je zde výběr rozmezí data, v kterém se mají výsledky ze závodů hodnotit. Po vybrání proběhne Ajaxové načtení dat ze serveru, kde se sestaví asociativní pole s potřebnými informacemi. Základem je vykonání následujícího SQL dotazu:

```
SELECT discipliny.nazev_discipliny, discipliny.id_discipliny,
format_disciplin.format, format_disciplin.id_formatu,
MAX(prirazeni_zavodu.vykon) AS max, MIN(prirazeni_zavodu.vykon)
AS min, COUNT(prirazeni_zavodu.id_zavodu) AS pocet,
AVG(prirazeni_zavodu.vykon) AS prumer FROM prirazeni_zavodu
   JOIN discipliny USING(id_discipliny)
   JOIN format_disciplin USING (id_formatu)
   JOIN zavody USING(id_zavodu)
   WHERE prirazeni_zavodu.id_sverence=? AND zavody.datum BETWEEN
? AND ? AND prirazeni_zavodu.vykon IS NOT NULL
   GROUP BY discipliny.nazev_discipliny
```

Využitím příkazu `GROUP BY` a agregačních funkcí jednoduše získáme většinu informací potřebných do statistik. `GROUP BY` nám všechny nalezené záznamy seskupí podle názvu disciplíny a pro každou z nich vypočítá minimum, maximum, počet a průměr. Kromě toho také získáme formát disciplíny, podle kterého následně upravíme některé nepřesné hodnoty. Agregační funkce se na výsledek automaticky dívají jako na čísla a nedokáží např. spočítat průměr závodů na 800 m, které mají výsledky ve formátu 1:57.23. Stejně tak vyhodnotí špatně maximum v běhu na 100 m, protože bude automaticky jako maximum brát nejvyšší číslo. Formát disciplín dopředu známe a tak využijeme konstrukci `switch case`,v které se podle příslušného formátu dopočítají a upraví hodnoty podle tak, aby odpovídaly skutečným výsledkům. Pro vykreslení grafu je potřeba získat pole výsledků v každé disciplíně za celé vybrané období s daty, ve kterých se konaly závody.

Po spuštění filtrace se uživateli vykreslí tolik řádků, kolik v daném časovém úseku absolvoval disciplín. Levá strana vždy obsahuje tabulku s názvem disciplíny a čtyř statistických údajů – nejlepšího a nejhoršího výkonu, počtu startů v disciplíně a průměrný výkon. V pravé



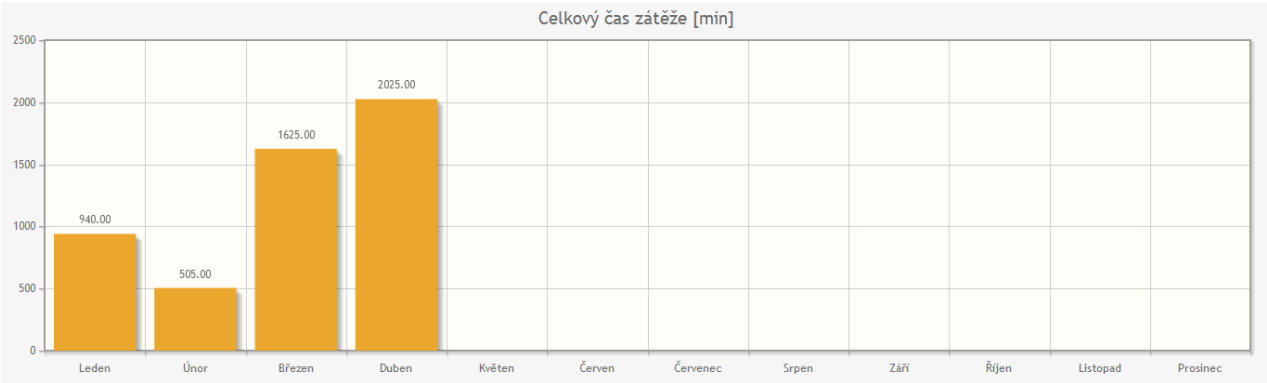
Ilustrace 15: Ukázka statistik na 400 m (v aplikaci je tabulka vedle grafu)

části se navíc vykreslí čárový graf, kde osa x znázorňuje datum závodu a na ose y je zaznamenán konkrétní výkon (viz Ilustrace 15). Graf je navíc interaktivní a po najetí na uzel v čáře se zobrazí přesný datum včetně výkonu. Pro vykreslení grafu je využit plugin jqPlot (viz 3.1.4). Inicializace grafu probíhá po nastavení parametrů zavoláním této funkce: `$.jqplot(containerId, [vykony], jqplotOptions)`, kde `containerId` je id elementu, v kterém se má graf vykreslit, `vykony` obsahují pole výkonů a `jqplotOptions` všechna nastavení týkajících se os, barev a dalších vlastností grafu.

4.7.5 Statistiky tréninků

Podobně jako u statistik závodů je zde hlavním kamenem jednoduchý filtrační formulář. Statistiky se vždy počítají za celý rok, který se musí zvolit, včetně ukazatele, který chceme vyhodnocovat. Trenér má navíc na výběr seznam svých svěřenců. Oba mají jako filtrační pomůcku ukazatelů k dispozici výběr skupiny ukazatelů. Při výběru položky ukazatelů se automaticky načte rozbalovací menu s ukazateli a aktualizuje se. Principem je opět Ajaxová komunikace se serverem, kdy máme menu obalené v makru `{snippet}` a na základě vybrané skupiny nastavíme formulářovému poli novou sadu údajů. Stejně takto funguje i celé další zpracování grafů.

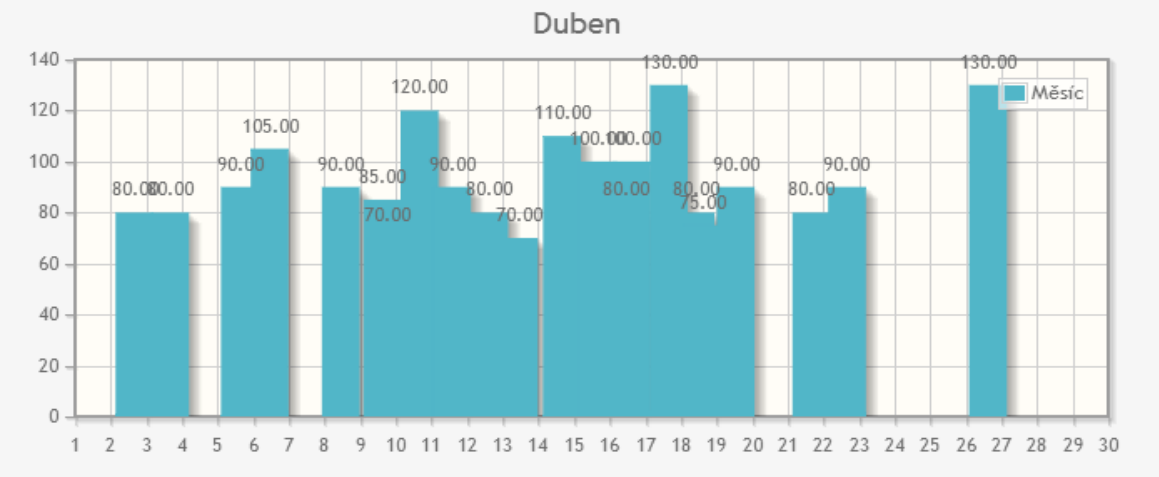
Po výběru sledovaného ukazatele a odeslání formuláře se vykreslí první graf, který je sloupcový, pokud má v daném roce a ukazateli uživatel zaznamenané údaje. Na ose x je dvanáct měsíců a osa y nese hodnotu součtu všech hodnot zadaného ukazatele v měsíci. Titulkem grafu je název ukazatele s jeho jednotkou v hranatých závorkách (viz Ilustrace 16).



Ilustrace 16: Ukázka vykresleného grafu ukazatele „Celkový čas zátěže“ za rok 2013

Graf je interaktivní a každý sloupec odchytává událost kliknutí. Pokud uživatel např. klikne na sloupec v měsíci duben, odešle se další požadavek na server, který na základě ukazatele, roku a uživatele vrátí součty v jednotlivých dnech. Tato data se převedou do formátu

JSON, který graf akceptuje a následně vykreslí. Osa x obdobně nese informaci o dni v měsíci a na ose y se dozvíme hodnotu ukazatele (viz Ilustrace 17). Výsledný graf je také interaktivní a po kliknutí na určitý den se nám vpravo od něj zobrazí všechny informace. Mezi ně patří náplň tréninku, případně více tréninků, pokud jich svěřenec absolvoval víc. Dále jeho poznámka, abychom mohli rychle zhodnotit, proč výsledek nebyl podle očekávání a seznam všech ostatních ukazatelů, které byly v daném tréninku zaznamenány.



Ilustrace 17: Ukázka hodnot ukazatelů v měsíci duben po kliknutí na sloupec v předchozím grafu

4.8 Administrace

V administraci aplikace jsou dostupné nástroje pro tvorbu článků, správu uživatelů a pro přidávání sportů a disciplín. Princip formuláře pro vkládání článků jsem popisoval v kapitole 4.4. Hlavní administrátor má práva pro přístup do všech sekcí. Do sekce články mají přístup ještě administrátoři s oprávněními `clanky`, `clankyVeSkupine` a `clankyVeSportu`. Oprávnění `clanky` může psát pouze v obecných kategoriích jako jsou návody nebo obecné informace týkající se chodu serveru. Uživatel s oprávněním `clankyVeSkupine` může psát články, které budou mít kategorii stejnou jako název tréninkové skupiny a uvidí je pouze její členové. Poslední oprávnění článků umožňuje uživateli psát články v kategorii sportu, ve kterém je zaregistrovaný.

Administrace uživatelů je nástroj, do kterého má přístup pouze hlavní administrátor a slouží k rozdělování práv mezi uživatele. Má formu jednoduché tabulky se seznamem uživatelů, informací o tom, jestli se jedná o svěřence nebo trenéra a rozbalovací menu s možností výběru oprávnění z tabulky `admin_prava`. Při každém zvolení jakéhokoliv oprávnění se asynchronně pošle dotaz na server, konkrétně na metodu `handleSetUserRights($userId, array $rightsIds)`, která má první parametr

id uživatele a druhý je pole id vybraných oprávnění. Algoritmus v metodě na serveru nastaví a smaže oprávnění podle toho, jak byly zvoleny. Aby mohl mít uživatel práva, tak musí mít záznam v tabulce `admin`. Pokud nemá, tak se vytvoří. Následně se z modelu `UzivateleRepository` načtou všechna dosavadní práva z tabulky `v_admin_prava`. Ta se postupně projdou v cyklu. Pokud je aktuální vybrané právo mezi dříve uloženými, tak ho pouze odstraníme z pole a pokračujeme. Pokud vybrané není, tak to znamená, že musí být smazáno z databáze a následně i z pole. Pokud po celém průchodu v poli vybraných práv zbyly záznamy, tak to znamená, že jsou nově přidané a v databázi ještě nejsou, proto se uloží.

Do administrace sportů má přístup administrátor s právem `sporty`. Jeho možnosti spočívají v tom, že dokáže zakládat sporty, které v databázi chybí. Tomuto jednoduchému formuláři tak stačí pouze pole pro zadání názvu sportu a tlačítko k odeslání. Oprávnění `disciplinyVeSportu` dovoluje uživateli přidávat disciplíny do sportu, ve kterém je zaregistrovaný. Formulář pro tuto akci obsahuje pole pro zadání názvu disciplíny a formát toho, jak se výkon v disciplíně zapisuje. Tato data se berou z číselníku `format_disciplin` a uživatel je povinen vybrat jeden z předepsaných formátů, aby bylo možné data zpracovat a použít ve statistikách.

5 Závěr

Po analýze celé problematiky jsem prohlédl několik již existujících řešení. Každá z nalezených aplikací měla specifické zaměření, které se neudávalo směrem, který jsem požadoval. Žádná z nich navíc neměla možnost vytvoření hierarchické struktury tréninkové skupiny nebo týmu a stejně tak mi nestačilo zpracování měřitelných tréninkových jednotek (ukazatelů).

Následně jsem se rozhodl pro tvorbu webové aplikace v PHP frameworku Nette. Při tvorbě aplikace jsem postupem času pocítil jedinou nevýhodu zvoleného frameworku. Jeho vývoj je tak rychlý, že tvorba dokumentace a návody k jeho používání jsou několik kroků pozadu. Jinak se mi osvědčil jako skvělý nástroj zejména v oblasti tvorby formulářů a zpracování Ajaxových požadavků. Kdybych měl udělat stejnou aplikaci znovu, tak se více zaměřím na zpracování datové vrstvy (modelů), které bych pojal jako třídy s atributy odpovídajícími atributům příslušné tabulky v databázi a ne jako třídy se sadou metod.

Testování aplikace probíhalo v průběhu jejího vývoje na mé tréninkové skupině. Zpočátku k aplikaci uživatelé přistupovali přes adresu na doméně mých osobních stránek patrikdrhlik.cz/tdeniky a po úspěšném odladění jsem aplikaci přesunul na vlastní doménu tdeniky.cz, kde bude běžet i nadále. První verzi aplikace testovalo 10 uživatelů a dnes už jich je na oficiálním serveru 45. Pomoc mých kolegů zároveň ověřila funkčnost všech částí aplikace a jejich kritika mi umožnila pohled na aplikaci ze strany uživatele, takže jsme společnými silami odhalili chyby, které jsem sám na první pohled neviděl.

Posledním bodem práce bylo sepsání stručného návodu k nastavení aplikace na lokálním počítači, ke kterému jsou k dispozici všechny zdrojové kódy a SQL skript, který vytvoří databázi a sadu testovacích dat pro vyzkoušení aplikace. Pro bližší seznámení s aplikací po uživatelské stránce jsem napsal jednoduchý návod, který stručně popisuje ovládání účtu od zaregistrování práci s jednotlivými sekcemi. Návod je přístupný na stránce aplikace na adrese tdeniky.cz/clanky/detail/4-jak-na-to.

Seznam použité literatury

[1] TVRZNÍK, Aleš a Vít RUS. *Tréninkový deník*. Praha: Grada, 2002. ISBN 978-80-247-0348-0.

[2] NETTE FOUNDATION. *Nette Framework* [online]. © 2008, 2013 [cit. 2013-04-10]. Dostupné z: http://nette.org/

[3] Scaffolding. MDO a FAT. *Bootstrap* [online]. [2010] [cit. 2013-04-11]. Dostupné z: http://twitter.github.io/bootstrap/scaffolding.html

[4] THE JQUERY FOUNDATION. *JQuery* [online]. [2006] [cit. 2013-04-11]. Dostupné z: http://jquery.com/

[5] LEONELLO, Chris. *JqPlot Charts and Graphs for jQuery* [online]. [2009] [cit. 2013-04-13]. Dostupné z: http://www.jqplot.com/

[6] MOXIECODE SYSTEMS AB. TinyMCE [online]. © 2003-2013 [cit. 2013-04-13]. Dostupné z: http://www.tinymce.com/

[7] THE JQUERY FOUNDATION. JQuery UI [online]. [2007] [cit. 2013-04-13]. Dostupné z: http://jqueryui.com/

[8] RICHARDSON, Trent. Adding a Timepicker to jQuery UI Datepicker. Trent Richardson | Practical Web Development[online]. [2013] [cit. 2013-04-13]. Dostupné z: http://trentrichardson.com/examples/timepicker/

[9] Píšeme první aplikaci. NETTE FOUNDATION. Nette Framework [online]. © 2008, 2013 [cit. 2013-04-16]. Dostupné z: http://doc.nette.org/cs/quickstart

[10] MVC aplikace & presentery. NETTE FOUNDATION. Nette Framework [online]. © 2008, 2013 [cit. 2013-05-15]. Dostupné z: http://doc.nette.org/cs/presenters

[11] Šablony. NETTE FOUNDATION. Nette Framework [online]. © 2008, 2013 [cit. 2013-04-22]. Dostupné z: http://doc.nette.org/cs/templating

[12] Konfigurace. NETTE FOUNDATION. Nette Framework [online]. © 2008, 2013 [cit. 2013-04-23]. Dostupné z: http://doc.nette.org/cs/configuring

[13] MULLENWEG, Matt. AUTOMATTIC. Gravatar – Celosvětově uznávané Avatary [online]. [2005] [cit. 2013-04-24]. Dostupné z: http://cs.gravatar.com/

Přiložené CD

Na přiloženém CD jsou všechny zdrojové kódy aplikace včetně stručného návodu k nastavení aplikace na lokálním počítači a SQL skriptu se strukturou databáze a sadou testovacích dat. Obsahuje také zprávu k bakalářské práci v elektronické formě.