TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

WEBOVÁ APLIKACE PRO SPORTOVNÍ VÝSLEDKY

DIPLOMOVÁ PRÁCE

Liberec 2017 **Bc. Filip Kulka**

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

WEBOVÁ APLIKACE PRO SPORTOVNÍ VÝSLEDKY

Diplomová práce

Studijní program: N2612 – Elektrotechnika a Informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Filip Kulka**
Vedoucí práce: doc. RNDr. Pavel Satrapa, Ph.D.

Liberec 2017

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

WEB APPLICATION FOR SPORTS RESULTS PROCESSING

Master thesis

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 1802T007 – Information Technology

Author: **Bc. Filip Kulka**
Supervisor: doc. RNDr. Pavel Satrapa, Ph.D.

Liberec 2017

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Filip Kulka**
Osobní číslo:	**M15000177**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Webová aplikace pro sportovní výsledky**
Zadávající katedra:	**Ústav nových technologií a aplikované informatiky**

Zásady pro vypracování:

1. Proveďte rešerši webů orientovaných na sportovní výsledky.
2. Navrhněte webovou aplikaci pro zobrazování výsledků turnajů a statistik ve hře ricochet.
3. Soustřeďte se na možnosti interaktivního zobrazování dostupných dat, analýzu výsledků a vzájemné porovnávání hráčů a automatické vytváření žebříčků.
4. Navrženou aplikaci implementujte a otestujte.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **40 - 60 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] GASSTON, Peter. Moderní web. 1. vydání. Přeložil Ondřej BAŠE. Brno: Computer Press, 2015. ISBN 978-80-251-4345-2.
[2] LECKY-THOMPSON, Ed a Steven D. NOWICKI. PHP 6: programujeme profesionálně. Vyd. 1. Brno: Computer Press, 2010. Programujeme profesionálně. ISBN 978-80-251-3127-5.

Vedoucí diplomové práce: **doc. RNDr. Pavel Satrapa, Ph.D.**
Ústav nových technologií a aplikované informatiky

Datum zadání diplomové práce: **20. října 2016**

Termín odevzdání diplomové práce: **15. května 2017**

Technická univerzita v Liberci - 14 -

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Dr. Ing. Jiří Maryška, CSc.
vedoucí ústavu

V Liberci dne 20. října 2016

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou 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 diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu 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.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15.5.2017

Podpis: 

Poděkování

Mé poděkování patří doc. Ing. Pavlu Satrapovi, Ph.D. za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování diplomové práce věnoval.

Dále bych rád poděkoval Ing. Janu Pulkrábovi za poskytnutí potřebných podkladů pro vypracování praktické části mé diplomové práce.

Abstrakt

Aktuální webová aplikace zobrazující výsledky sportu ricochet je velmi náročná na správu a neobsahuje žádné funkce zobrazující statistiky hráčů. Z toho důvodu vznikl požadavek ze strany České ricochetové asociace na vytvoření nové aplikace. V rámci diplomové práce jsem pomocí PHP frameworku Laravel vytvořil požadovanou webovou aplikaci, která obsahuje rozsáhlé hráčské statistiky a zároveň je jednoduchá na její správu. Tato webová aplikace je dostupná na adrese www.ricostats.cz a od sezóny 2017/2018 nahradí stávající webovou aplikaci.

Klíčová slova: webová aplikace, PHP, Laravel, ricochet

Abstract

The current web application for sport results from ricochet is very demanding and does not contain any statistics feature of players. For this reason, the requirement arose from the Czech Ricochet Association to create a new application. Within the diploma thesis I created a web application using PHP and Laravel framework that contains statistics of each player and is easy to manage. The web application is located at www.ricostats.cz and will be replacing the current application at the start of the 2017/2018 season.

Keywords: web application, PHP, Laravel, ricochet

Obsah

Seznam symbolů, zkratek a termínů

- **MVC** (Model-View-Controller) je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent.
- **Open source** je počítačový software s otevřeným zdrojovým kódem.
- **ORM** (Object-relational mapping) je programovací technika v softwarovém inženýrství zajišťující automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem.
- **ActiveRecord** je architektonický návrhový vzor pro práci s datovými zdroji.
- **MySQL** je relační databázový systém.
- **HTML** (HyperText Markup Language) je název značkovacího jazyka používaného pro tvorbu webových stránek.
- **CRUD** (Create, Read, Update, Delete) je zkratka shrnující čtyři základní operace nad daným záznamem: vytvoření (create), čtení (read), editace (update) a smazání (delete).
- **Markdown** je odlehčený značkovací jazyk, který slouží pro úpravu prostého textu a jeho následný převod na formátovaný text publikovatelný na webu, zejména ve formátu HTML.
- **String** je název datového typu sloužícího k uložení konečné posloupnosti znaků.
- **API** (Application Programming Interface) je termín používající se v softwarovém inženýrství, označující rozhraní pro programování aplikací.

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Úvod

Webových aplikací, které zobrazují sportovní výsledky, je nespočetné množství. Pokud se však zaměříme na konkrétní sporty, například ty, které nejsou tolik rozšířené, počet webových aplikací rapidně klesá. Vezmeme-li v potaz sport ricochet, nalezneme na internetu pouze jednu webovou stránku zobrazující výsledky z turnajů. Jedná se o oficiální webovou stránku České ricochetové asociace. Tato webová aplikace je však z administrátorského hlediska velmi náročná na správu. Data se vkládají do předdefinovaného souboru typu excel, který se následně musí nahrát na server. Další problém je ve finanční náročnosti aplikace, kdy se pouze za chod redakčního systému platí nemalé měsíční poplatky.

Ze strany České ricochetové asociace vznikl požadavek na vytvoření nové webové aplikace, která bude jednodušeji spravovatelná, bude obsahovat větší množství dat a bude obsahovat funkce na zobrazení jednotlivých zápasů a porovnání dvou hráčů. Tento požadavek jsem se rozhodl realizovat v rámci diplomové práce a vytvořit tak novou webovou aplikaci obsahující všechny požadované funkce.

1 Webové aplikace se sportovními výsledky

Každý, kdo se nějakým způsobem zajímá o sport, chce mít přehled o daných sportovních událostech. V dnešní době internetu, sociálních sítí a chytrých telefonů je získání takových dat o mnoho jednodušší než dříve, kdy se dané výsledky daly zjistit pouze z tisku, teletextu, nebo rádia. Nastává však otázka, jakým způsobem uživateli co nejvíce zjednodušit hledání daných informací a jakým způsobem mu je dodat v co možná největším množství. Pokud si vezmeme jako příklad sport fotbal, v dnešní době můžeme velice jednoduše zjistit, kolik rohů, karet, offsidů, vhazování, faulů, přímých kopů a dalších událostí v daném zápase proběhlo. Webové aplikace se sportovními výsledky, mezi které patří například Livesport.cz a Onlajny.com, nám tato data jednoduše poskytují.

Pokud si v těchto aplikacích rozkliknete sekci fotbal, můžete denně sledovat desítky až stovky fotbalových výsledků a statistik, které jsou pomocí těchto webových aplikací v reálném čase přenášeny ke koncovým uživatelům. Hloubka statistik záleží na prestiži dané ligy nebo zápasu. Jedná-li se o velice sledované utkání, můžete kromě výše zmiňovaných statistik sledovat i textový komentář zápasu. Ovšem pokud jde o zápas s velmi nízkou sledovaností, jedinou dostupnou informací je výsledek, který je v některých případech zveřejněn až po skončení utkání.

O tato data je na internetu, ať už z fanouškovského hlediska, nebo ze strany sázkařů, kteří podle statistik vsází své peníze s vidinou výhry, obrovský zájem. Existuje několik velice propracovaných aplikací, kterými jsem se při vypracování této diplomové práci inspiroval. Dále se detailněji zaměřím na některé z nich.

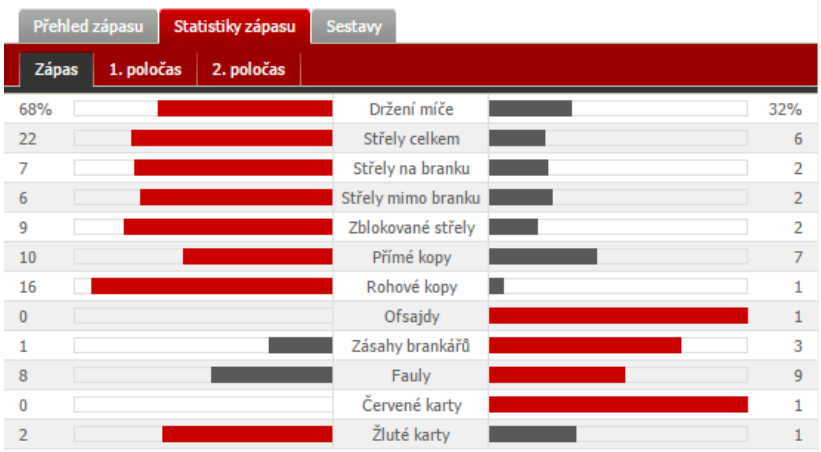
1.1 Livesport.cz

Jedná se o český projekt, který se postupem času rozrostl v mezinárodně uznávaný a využívaný portál se sportovními výsledky. Tento portál obsahuje statistiky z 33 sportů, avšak mezi nejsledovanější sporty patří fotbal (1138 soutěží), hokej (223 soutěží) a tenis (2862 soutěží).

Mezi hlavní výhody této webové aplikace oproti konkurenci patří široká škála zpracovávaných sportů, rychlost zobrazení výsledků, možnost personalizace, live tabulky, zvukové notifikace, live centre, video sestřihy a tzv. H2H porovnání.

Obrázek 1: Domovská stránka Livesport.cz

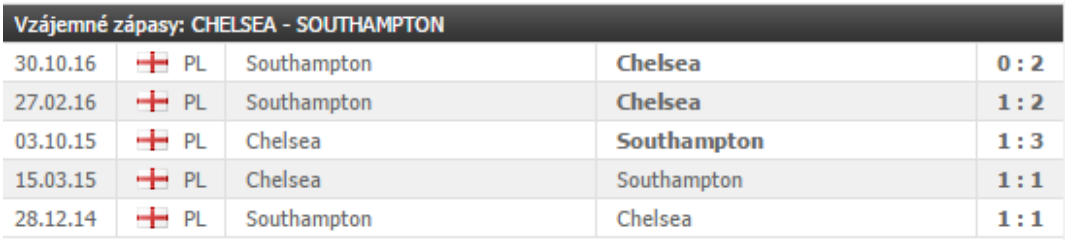
Server Livesport.cz se pyšní statistikou, kdy live výsledky nejsou zobrazeny později, než za 10 vteřin od dané události, například od vstřelení gólu. Každý uživatel si může na serveru vytvořit svůj profil, do něhož lze vložit vlastní vybrané oblíbené ligy, které se mu budou zobrazovat jako první. Dále je možné do profilu vložit oblíbené týmy, u kterých se mu vždy automaticky zobrazí výsledky v reálném čase. Profil dále nabízí možnost zvukových notifikací. Pokud se tedy v zápase, který má uživatel označený jako oblíbený, stala určitá důležitá událost, dostanete o tom ihned informaci.



Obrázek 2: Zobrazení podrobných statistik na Livesport.cz

U vysoce sledovaných zápasů je zde možnost sledovat v reálném čase i velice podrobné statistiky (střely na branku, střely mimo branku, zablokované střely, vhazování, rohové kopy, fauly atd.), live komentář a video sestřihy důležitých momentů zápasu.

Poslední, avšak velice důležitá možnost portálu Livesport.cz je takzvané H2H porovnání, které spočívá v zobrazení historických sportovních událostí (výsledky a statistiky) dvou vybraných týmů.



Vzájemné zápasy: CHELSEA - SOUTHAMPTON				
30.10.16	PL	Southampton	**Chelsea**	**0 : 2**
27.02.16	PL	Southampton	**Chelsea**	**1 : 2**
03.10.15	PL	Chelsea	**Southampton**	**1 : 3**
15.03.15	PL	Chelsea	Southampton	**1 : 1**
28.12.14	PL	Southampton	Chelsea	**1 : 1**

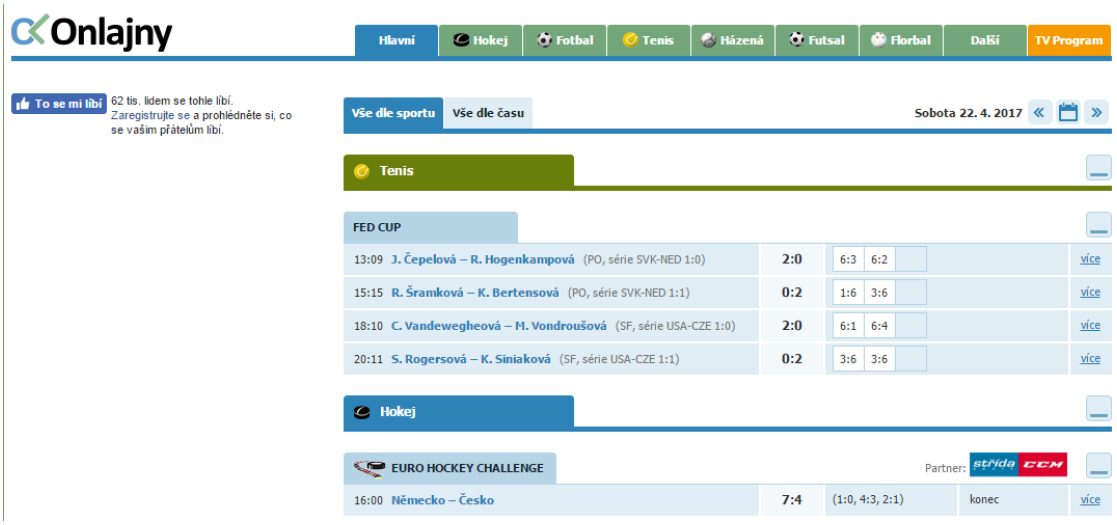
Obrázek 3: Zobrazení H2H porovnání na Livesport.cz

Webovou aplikací Livesport.cz jsem se inspiroval v mnoha aspektech zobrazování výsledků. Dle mého názoru je tato aplikace nejvíce propracovaná ze všech dostupných aplikací a zároveň obsahuje největší množství dat.

1.2 Onlajny.com

Jedná se o další český projekt, který je svým obsahem velmi podobný serveru Livesport.cz. Bohužel však zdaleka neposkytuje tak širokou škálu sportů a především sportovních lig, jako výše popisovaná aplikace Livesport.cz. Na tomto serveru lze nalézt pouze nejvíce populární sporty, mezi které patří fotbal, lední hokej, tenis, basketbal, házená, futsal a florbal. V těchto sportech jsou navíc dostupné pouze nejvyšší ligy a v nich jen udělené karty, vyloučení, góly a střídání hráčů.

Jako jedinou pozitivní věc na serveru Onlajny.com shledávám zobrazení důležitých událostí, jako jsou góly, ale i gólové šance, formou animace ve formátu GIF.



Obrázek 4: Domovská stránka Onlajny.com

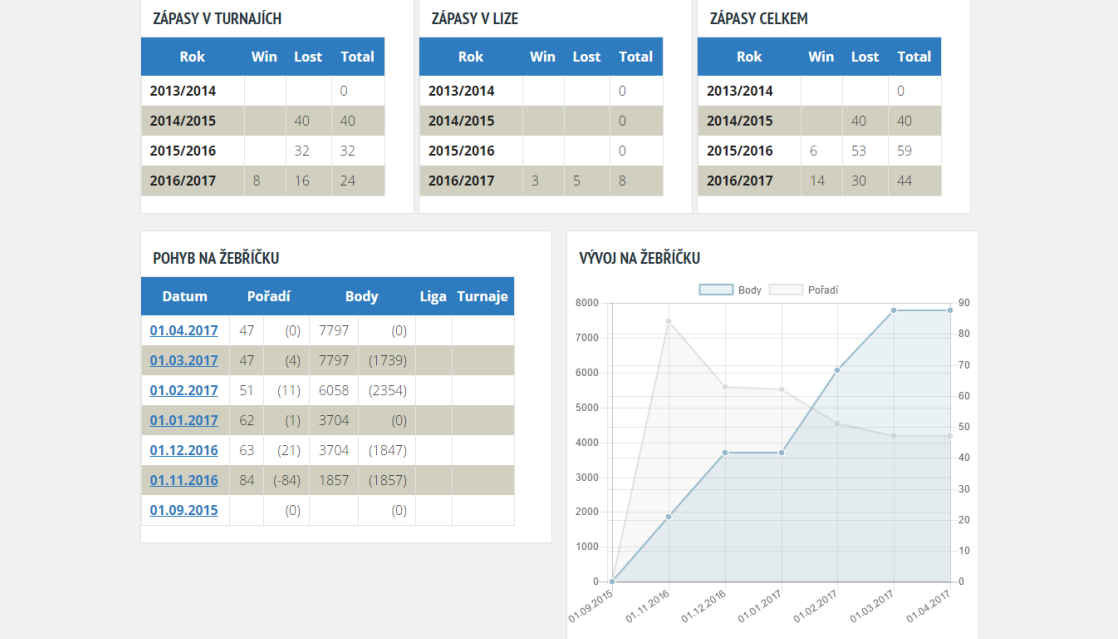
Jelikož se na tomto portálu vyskytují i tipy na dané zápasy od ostatních uživatelů, analýzy utkání a aktuální kurzy od sázkové společnosti Tipsport, je tento portál spíše zaměřený na uživatele – sázkaře.

1.3 Czechsquash.cz

Jak již název napovídá, webová aplikace Czechsquash.cz pracuje pouze s výsledky sportu squash v České republice. Jde o oficiální webové stránky České squashové asociace – svazu klubů a hráčů v České republice.

Jedná se o velice podobný sport, jakým je ricochet. Mezi hlavní rozdíly mezi squashem a ricochetem patří jiné technické parametry hracích kurtů a vybavení (raket a míčků) a několik dalších rozdílů v pravidlech. Dalším velkým rozdílem je komunita lidí, která se squashi a ricochetu věnuje. Pokud mluvíme pouze o České republice, je zde k 28. 4. 2017 registrováno 763 mužů, hrajících squash [8] a kolem 60 mužů, hrajících ricochet. V celosvětovém měřítku je tento rozdíl mnohem větší. Juniory a ženy do tohoto porovnání nezařazuji, jelikož jich je v ricochetu zanedbatelné množství.

Můžeme zde nalézt novinky z daných turnajů, detailní pavouky (rozlosování jednotlivých zápasů turnaje) a výsledky všech odehraných zápasů. Po rozkliknutí profilu jednotlivého hráče se zobrazí jeho detailní statistiky, jako je historický pohyb na žebříčku, odehrané zápasy a získané body.



Obrázek 5: Zobrazení profilu hráče na Czechsquash.cz

Historická data na webu Czechsquash.cz sahají až na začátek sezóny 2013/2014. Pokud někdo začal hrát později než v této sezóně, má ve svém profilu defaultně zobrazené nuly, což dle mého názoru není ideální způsob zobrazení historických výsledků jednotlivých hráčů. Pokud tedy nějaký hráč začal hrát squash až v sezóně 2016/2017,

má ve svém profilu tři řádky zaplněné nulami, a až poslední řádek, který označuje aktuální sezónu, má vyplněný konkrétními daty (viz Obrázek 5: Zobrazení profilu hráče na Czechsquash.cz).

Jelikož se každý měsíc hraje několik squashových turnajů, jsou stránky Czechsquash.cz naplněny velkým množstvím dat, která se zobrazují pomocí tabulek a grafů. Avšak chybí mi zde porovnání dvou vybraných hráčů a jejich aktuální forma. Na to, s jak rozsáhlými daty tato webová aplikace pracuje, by mohla mít více funkcí, než jen zobrazení profilu hráčů, turnajů a žebříčků.

1.4 E-ricochet.cz

E-ricochet.cz je jediná webová stránka, zabývající se sportem ricochet v České republice. Obsahuje seznam hráčů a turnajů. Jedná se o oficiální stránku České ricochetové asociace.

Po zobrazení vybraného turnaje se objeví volba otevření výsledků v souboru PDF, který musí správce webové aplikace vždy ručně nahrát na server.

Po zobrazení hráčského profilu je k vidění stručný popis hráče, rok narození, klubová příslušnost a aktuální pozice v žebříčku.

Tato aplikace není příliš rozsáhlá, a kromě výsledků v souborech PDF a republikového žebříčku neobsahuje žádné statistické údaje.

2 Ricochet

Jak již bylo zmíněno, ricochet (anglicky znamenající odraz) je sport velice podobný squashi. Jedná se o hru pro dva hráče, při které musí být vždy jednoznačně určen vítěz. Hraje se na tři vítězné sety a hráči se střídají na podání podle toho, kdo vyhrál předchozí výměnu. Podává ten, kdo směnu vyhrál. První podání zápasu náhodně určuje los techniky (podobné např. hodu mincí u fotbalu), která zároveň počítá skóre zápasu. Set vyhrává hráč, jenž jako první dosáhne 15 bodů s tím, že poslední bod musí získat ze svého podání. Aby set vyhrál, musí tedy pokořit hranici 15 bodů a vyhrát poslední dvě výměny. Dále hráč musí zvítězit rozdílem 2 bodů. Pokud se nedaří set dohrát, vyhrává hráč, který první dosáhne 21 bodů (bez ohledu na pravidlo o dvoubodovém rozdílu – je tedy možný výsledek 21:20).

Ricochet se hraje v uzavřené místnosti, kde tři stěny jsou betonové a pouze zadní zeď je skleněná (kvůli rozhodčímu a divákům). Kurt je 8 m dlouhý, 5,5 m široký a 2,7 m vysoký [14]. Každý hráč má vlastní raketu, jež je rozměrově větší než ping-pongová pálka a menší než squashová raketa. Hraje se s gumovým míčkem, který se dle zdatnosti hráčů dělí na dvě kategorie – červený a šedivý. S červeným míčkem hrají junioři a začátečníci, protože více skáče a není potřeba mít tolik razantní údery. Se šedivým míčkem se hrají republikové i mezinárodní turnaje, jedná se o oficiální míček mezinárodní ricochetové asociace.

Obrázek 6: Ricochetový kurt [14]

Smyslem hry je donutit protihráče k chybě. Chybou se počítá neodehrání míčku nebo porušení pravidel. Základní pravidla jsou taková, že míček může před odehráním spadnout maximálně jednou na zem. Hráč může míček odehrát na libovolnou zeď (boční i strop), ale vždy se musí odrazit od hlavní (přední) stěny. Na spodní části přední stěny je červený pruh, pod který se nesmí hrát (podobně jako u tenisu síť). Pokud hráč zahraje pod červenou čáru, snímací technika, zabudovaná v kurtu, zapípá.

Ricochetové turnaje se dělí na 4 kategorie:

- Mezinárodní turnaje, kterých se mohou zúčastnit hráči z celého světa, se konají v těchto státech: Německu, Slovensku, Maďarsku, Nizozemsku, České republice a Norsku.
- Mistrovství České republiky konající se jednou za rok a pouze pro hráče s českou národností.
- Turnaje typu Classic, které se organizují vždy jednou měsíčně.
- Turnaje typu Gold, které se konají dvakrát za sezónu a nahrazují turnaje typu Classic.

Jediný rozdíl mezi turnaji typu Gold a Classic je ten, že vítěz zápasu na turnaji typu Gold získá navíc 10 % bodů z republikového žebříčku svého soupeře.

Získané body z mezinárodních turnajů se nezapočítávají do republikového žebříčku, ale do žebříčku mezinárodního.

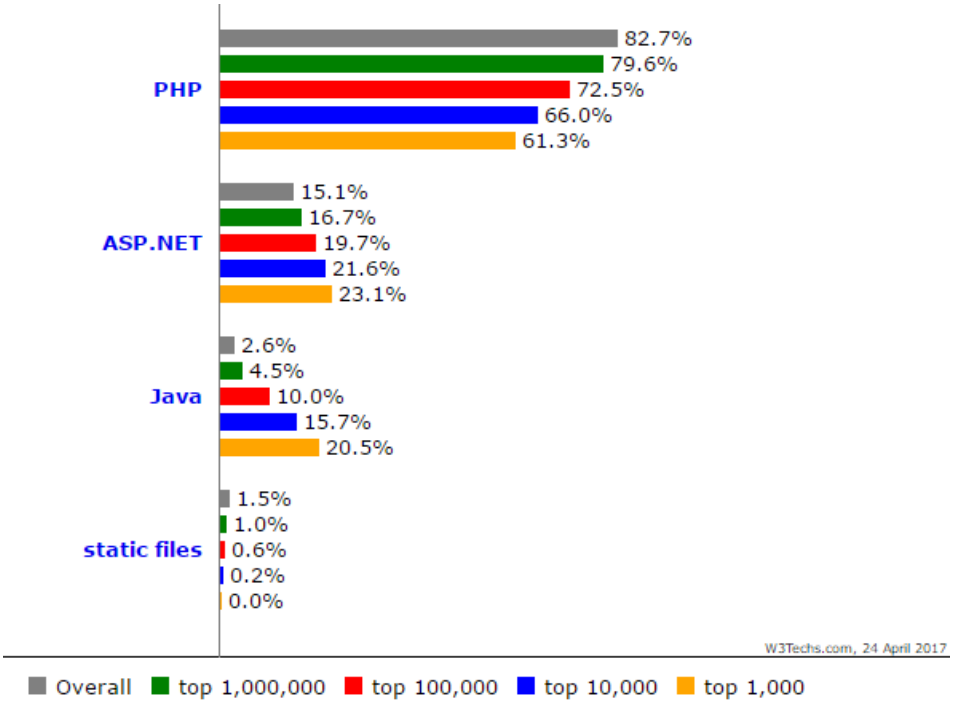
3 Použité technologie

Webové aplikace je možné psát v několika programovacích jazycích, kdy každý z nich má nespočet frameworků. Pro mou diplomovou práci jsem si vybral jazyk PHP, který společně s frameworky Nette, Symfony a Laravel popisuji v následujících kapitolách.

3.1 PHP

PHP, neboli Hypertext Preprocessor (původně Personal Home Page), je skriptovací programovací jazyk, určený pro programování dynamických internetových stránek a webových, konzolových i desktopových aplikací.

Syntaxe jazyka PHP je nejvíce inspirována jazyky C, Java a Perl. Jazyk podporuje mnoho knihoven pro různé účely, například přístup ke většině databázových systémů, práci se soubory, zpracování grafiky, textu atd. Dále podporuje celé řady internetových protokolů. Při použití PHP pro dynamické stránky se skripty provádí na serverové straně, k uživateli je přenesen až výsledek jejich činnosti [1].



Obrázek 7: Procentuální využití skriptovacích jazyků pro web [2]

Dle údajů ze serveru W3Techs – Web Technology Surveys [2], je jazyk PHP nejpoužívanějším skriptovacím jazykem současnosti (duben 2017) s podílem 82,7 %. Na druhém místě se nachází jazyk ASP.NET s 15,1 % a třetím v pořadí je programovací jazyk Java s 2,6 %.

Jazyk PHP nejčastěji nalezneme v kombinaci s webovým serverem Apache, databázovým systémem (MySQL nebo PostgreSQL) a operačním systémem Linux. Tato kombinace se vžila jako zkratka LAMP.

Mezi nejznámější webové stránky používající jazyk PHP patří např. Facebook, Wikipedia, Twitter aj. Aktuální stabilní verze 7.1.2 byla vydána 17. února 2017. Avšak nejpoužívanější verzí dle serveru [2] zůstává verze 5, která je používána na 94,7 % webových stránkách využívajících jazyk PHP.

Tabulka 1: Procentuální využití verzí PHP (k 24. dubnu 2017), dle [2]

verze 5	94,7 %
verze 7	4,3 %
verze 4	1,0 %
verze 3	méně než 0,1 %

Pro vypracování aplikace jsem použil PHP verze 5, jelikož jsem žádnou z novinek verze 7 při tvorbě webové aplikace nepotřeboval. K volbě také velmi napomohl fakt, že PHP 5 je nejrozšířenější verzí PHP a zároveň nejlépe dosažitelnou na hostingových serverech.

3.2 Frameworky PHP

Frameworky jsou knihovny, vytvořené k ulehčení práce při programování dané aplikace. V praxi to znamená méně psaní, přehlednější kód a rychlejší vývoj. Nejlépe slovo framework vystihl Jeff Croft, který ho popsal jako sadu nástrojů, knihoven, konvencí a osvědčených postupů, jež vytváří abstrakci nad rutinními úkoly do obecných modulů, které mohou být lehce znovu využity [3].

Jako hlavní výhody při používání frameworků bych uvedl podporu MVC, která zajišťuje oddělení prezentace od logiky, podporu moderních postupů vývoje webových aplikací, jako jsou například objektově orientované programovací nástroje, zvýšení úrovně zabezpečení webu anebo používání organizovaného, opakovaně použitelného a udržovaného kódu. Mezi další výhody u populárnějších frameworků patří rozsáhle komunity, které dopomáhají vývoji, vytváří návody k daným problémům a jsou připraveny pomoci.

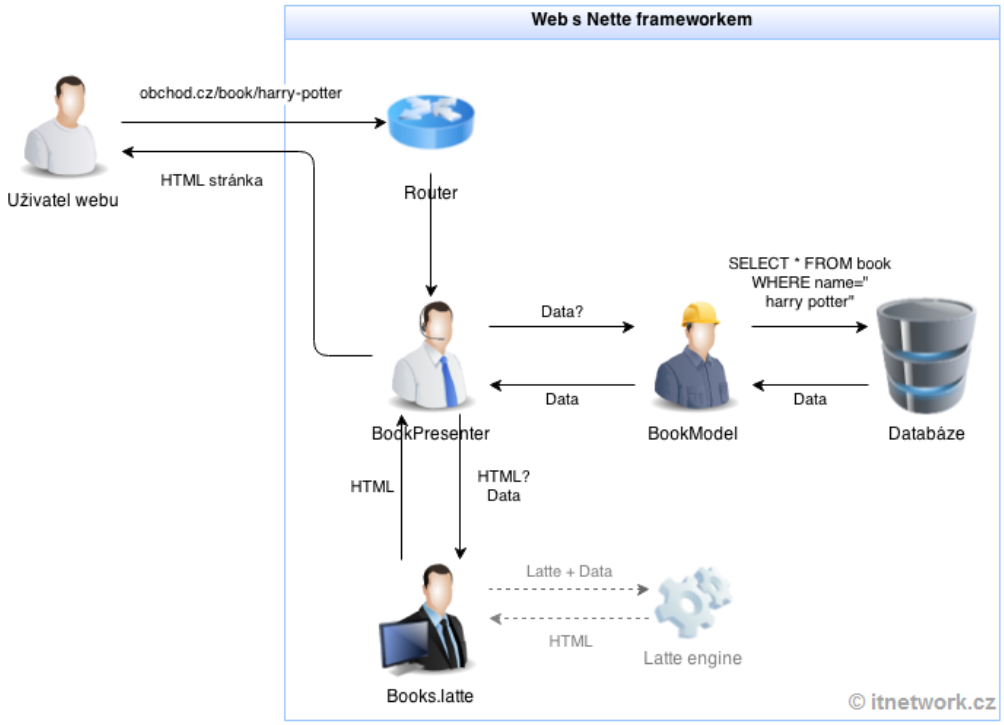
Tabulka 2: Nejoblíbenější PHP frameworky dle [4] z roku 2015, upraveno

Země	Celkem hlasů	V zaměstnání	Hlasů	Osobní	Hlasů
USA	819	Laravel	219	Laravel	293
Česká republika	770	Nette	611	Nette	639
UK	496	Laravel	138	Laravel	166
Německo	428	Symfony2	76	Laravel	100
Francie	343	Symfony2	149	Symfony2	136

PHP frameworků existuje velké množství, každý má své specifikace, výhody i nevýhody. Je tedy důležité vybrat si ten, který danému programátorovi sedí nejvíce. Na některé známé frameworky se nyní podíváme detailněji.

3.2.1 Nette

Nette je český open source framework pro tvorbu webových aplikací v PHP 5 a PHP 7, založený na MVC architektuře. Jedná se o nejvyžadovanější PHP framework v České republice vůbec [4]. Své oblíbenosti u českých a slovenských programátorů se mu dostalo především díky široké komunitě a podrobné dokumentaci v českém jazyce. V mezinárodním měřítku se nejedná o tolik rozšířený framework.



Obrázek 8: Životní cyklus stránky v Nette frameworku [5]

Pro práci s Nette frameworkem je potřebné stáhnout si tzv. sandbox. Jedná se o kostru webové aplikace, do které se postupně přidávají stránky.

MVC neboli u Nette MVP (Modely, Views, Presentery), stojí na komponentách třech typů. Tyto tři komponenty se v aplikaci dělí o řízení, logiku a výstup. Jen takto rozdělená aplikace je totiž přehledná a lehce rozšiřitelná.

Modely zajišťují práci s databází, obstarávají výpočty, starají se o logiku aplikace. Každá entita má většinou svůj model.

Views, neboli šablony, jsou výstupem pro uživatele a obsahují Latte šablony s HTML kódem. Latte je šablonovací jazyk, který do HTML šablon umožňuje vkládat data z PHP pomocí speciálních značek.

Presenter je komponenta, s níž komunikuje přímo uživatel. Jedná se o jakéhosi prostředníka. Uživatel předá presenteru parametry, ten je předá modelům, od kterých získá data. Tato data předá pohledům, jež je následně začlení do HTML kódu, který presenter pošle zpět uživateli jako HTML celek.

Adresářová struktura frameworku Nette je zobrazena v příloze A.

3.2.2 Symfony

Symfony je framework inspirovaný jinými webovými aplikačními frameworky, jako jsou Ruby on Rails, Spring a Django. Jedná se o open source projekt, na kterém stojí například diskuzní systém phpBB, CMS Drupal, nebo i jiné PHP frameworky, jako například Laravel. Vývoj Symfony je dále sponzorován francouzskou firmou Sensio Labs.

Komponenty frameworku Symfony jsou opakovaně použitelné knihovny PHP, které se používají k provádění nejrůznějších úloh, jako je tvorba formulářů, konfigurace objektů, routing, ověřování, šablonování a další. Libovolnou komponentu lze nainstalovat pomocí dependency managera zvaného Composer, jenž se automaticky stará o závislosti daných knihoven.

Struktura frameworku Symfony je popsána v příloze B.

3.2.3 Laravel

Přestože je Laravel relativně nový PHP framework (první verze byla vydána v roce 2011), podle nejnovějšího průzkumu serveru Sitepoint [4] jeho popularita rychle roste a překonal řadu zavedených frameworků. Opomineme-li Českou republiku, kde je jednoznačně nejoblíbenější český framework Nette, je Laravel nejpoužívanějším frameworkem mezi vývojáři jak v zaměstnání, tak k osobním účelům (viz Tabulka 2). Oficiální webové stránky nabízejí celou řadu tzv. screencast tutoriálů, zvaných Laracasty. Laravel, jakožto nejoblíbenější framework, má velmi rozsáhlou komunitu lidí, kteří se podílí na vývoji. Autorem je softwarový inženýr Taylor Otwell, jenž měl vývoj frameworku až do konce roku 2014 pouze jako vedlejší činnost.

Framework Laravel jsem vybral pro diplomovou práci z důvodu, že porozumění jemu a všem jeho komponentám bylo mnohem snadnější a intuitivnější než u výše zmi-

ňovaných frameworků. Dalším důvodem pro volbu Laravelu bylo rozsáhlé množství návodů (Laracastů), podle kterých jsem při práci postupoval.

Laravel a jeho komponenty jsou podrobněji popsány v kapitole 3.3 Vlastnosti frameworku Laravel a dále v jejích podkapitolách.

Framework Laravel používá ke správě závislostí takzvaný Composer, což je nástroj pro správu závislostí v jazyku PHP. Před samotnou instalací frameworku je tedy nutná instalace Composer.

Laravel lze nainstalovat třemi různými způsoby (doporučená instalace je právě přes Composer):

1. Instalátorem Laravel

Nejdříve je potřeba nainstalovat daný instalátor, to lze provést pomocí: *composer global require "laravel/installer"*. Následně stačí příkazem *laravel new blog* framework nainstalovat. Vytvoří se složka blog v daném adresáři.

2. Pomocí Composer create-project

Druhým způsobem je instalace příkazem: *composer create-project --prefer-dist laravel/laravel blog.*

3. Stažením aktuálního repositáře ze serveru GitHub

Po stažení aktuální verze frameworku a jejím rozbalení stačí spustit příkaz: *composer install*, který doinstaluje potřebné závislosti.

Po instalaci frameworku není potřeba žádná konfigurace. Zobrazí se uvítací stránka a framework lze používat. Jediné, co je potřeba nakonfigurovat, jsou případně údaje k databázi nebo k e-mailovému klientovi. Konfiguraci lze nalézt v adresáři config.

3.3 Vlastnosti frameworku Laravel

Laravel využívá architekturu MVC (Model, View, Conroller), která rozděluje kód do tří nezávislých vrstev. Tyto vrstvy mezi sebou následně komunikují a tvoří jednotný celek. Model obsahuje logiku aplikace, stará se o výpočty, obsahuje data. View se stará o vykreslování výstupu uživateli. Controller zpracovává požadavky, mění stav modelu, je to řídící vrstva aplikace.

Framework obsahuje několik velmi užitečných rozhraní a nástrojů, které zjednoduší programátorovi práci. Nejdůležitější z nich jsem popsal v následujících kapitolách.

3.3.1 Artisan

Artisan je rozhraní příkazového řádku, které je součástí frameworku Laravel. Umí generovat kostry jednotlivých částí aplikace, pracovat s migracemi (*php artisan make:migration*), vytvářet modely (*php artisan make:model*), kontrolery (*php artisan make:conroller*) atd. Všechny možné příkazy lze zjistit pomocí *php artisan list*.

3.3.2 Blade Templates

Webové frameworky obvykle obsahují nástroje pro práci se šablonami. Účelem těchto nástrojů je oddělení HTML od kódu programovacího jazyka.

Blade je jednoduchý, avšak výkonný šablonovací systém, obsažený ve frameworku Laravel. Na rozdíl od ostatních PHP šablonových systémů Blade neomezuje v používání jednoduchého PHP kódu v pohledech. Ve skutečnosti jsou všechny Blade pohledy kompilovány do jednoduchého PHP kódu a ukládány do mezipaměti, dokud nebudou modifikovány. Což znamená, že Blade pohledy nepřidávají do aplikace žádnou náročnost [6]. Blade pohledy používají příponu souboru .blade.php a jsou obvykle uloženy v adresáři resources/views.

3.3.3 Eloquent ORM

Eloquent ORM, jakožto součást Laravelu, poskytuje jednoduchou implementaci návrhového vzoru pro práci s datovými zdroji ActiveRecord. Pomocí ActiveRecord lze způsobem zobrazeným ve Zdrojový kód 1 vytvořit například uživatele.

Nejdříve se vytvoří objekt User, do kterého se v následujícím kroku vloží na řádek username hodnota Filip. Příkaz save volaný na objekt User uloží hodnoty do databáze.

```
1.  $user = new User;
2.  $user->username = 'Filip';
3.  $user->save();
```

Zdrojový kód 1: Práce s Eloquent ORM, vytvoření objektu

Jedná se o vrstvu, která se nachází mezi databází a aplikací. Každá tabulka z databáze má svůj vlastní Model, který se používá pro interakci s danou tabulkou. Modely umožňují získávání dat z databázových tabulek, ale i vkládání nových záznamů. Eloquent ORM podporuje databáze typu MySQL, Postgres, SQLite a SQL Server.

Pro správnou komunikaci mezi databází a Eloquent ORM by měla mít tabulka v databázi stejný název, akorát v množném čísle, jako Model. Např. tabulka users, model User.

3.3.4 Query Builder

Jedná se o rozhraní pro práci s databází, které je abstraktnější než třída DB, navíc automaticky chrání před SQL injection. Není tedy nutné ošetřovat předávané řetězce.

3.3.5 Schema Builder

Třída Schema obsahuje metody pro definici a úpravy struktur databázových tabulek. Pracuje se všemi podporovanými typy databází a ve všech těchto systémech má stejné API.

Vytváření tabulek probíhá následujícím způsobem:

```
1.  Schema::create('users', function($table)
2.  {
3.      $table->increments('id');
4.      $table->string('title');
5.  });
```

Zdrojový kód 2: Schema Builder, vytvoření tabulky v databázi

Přejmenování nebo smazání tabulky se provádí:

```
1.  Schema::rename($from, $to);
2.  Schema::drop('users');
```

Zdrojový kód 3: Schema Builder, přejmenování a smazání tabulky

3.3.6 Migrations

Migrace ve frameworku Laravel se dají popsat jako verzovací systém nad databází. Jsou to třídy, které se ukládají do adresáře app/database/migrations a obsahují dvě metody, up a down. Pomocí metody up se vytváří nebo upravují tabulky. V těle metody up se používá Schema Builder (popsaný v kapitole 3.3.5 Schema Builder). Metoda down zajišťuje smazání dané tabulky.

3.3.7 Seeding

Seeding, neboli naplnění databáze daty (a to nejen testovacími), se v Laravelu provádí pomocí třídy Seed. Všechny třídy typu Seed jsou uloženy v adresáři database/seed. Pojmenování těchto tříd nemá žádná pravidla, avšak je doporučeno držet se názvů podle toho, do které tabulky daná třída vkládá data. Vytvoření seederu probíhá v příkazové řádce pomocí:

php artisan make:seeder UsersTableSeeder

Tělo seederu pak může vypadat takto:

```
1.  class UsersTableSeeder extends Seeder
2.  {
3.      public function run()
4.      {
5.          DB::table('users')->insert([
6.              'name' => str_random(10),
7.              'email' => str_random(10).'@gmail.com',
8.              'password' => bcrypt('secret'),
9.          ]);
10.      }
11.  }
```

Zdrojový kód 4: Seedování dat do databáze

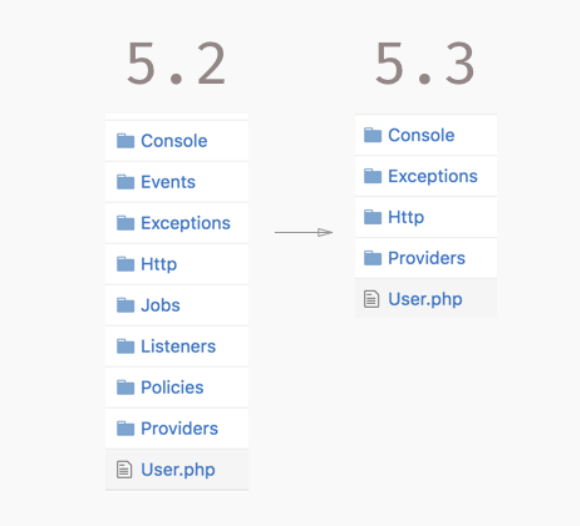
3.3.8 Adresářová struktura

Uživatele začínající s frameworkem Laravel čeká při pohledu na adresářovou strukturu aplikace překvapení. Na rozdíl od jiných frameworků (Nette zobrazeno v příloze A, Symfony v příloze B) u Laravelu nenaleznete adresář s modely. Vývojáři frameworku toto odůvodňují rozdílným výkladem slova model. Někteří ho definují jako třídy pracující s relační databází, jiní jako veškerou logiku aplikace. Z tohoto důvodu se všechny modely nachází v adresáři app.

Adresářová struktura frameworku Laravel se v průběhu jeho vývoje velmi radikálně mění. Novější verze přinášejí méně adresářů a jiné uspořádání souborů. Největší změny přinesly verze 5.0 a 5.3, kdy se velkou měrou změnil adresář app. Jak lze vidět na Obrázek 9: Porovnání adresáře app ve verzích 4.2 a 5.0 [12]od verze 5.0 se zredukoval počet adresářů, které nově začínají velkým písmenem. Například soubor routes.php a adresář controllers jsou od verze 5.0 ve složce Http.

Obrázek 9: Porovnání adresáře app ve verzích 4.2 a 5.0 [12]

Další velká změna nastala při zveřejnění verze 5.3 (23. 8. 2016), kdy se adresář app zredukoval pouze na čtyři složky (viz Obrázek 10: Porovnání adresáře app ve verzích 5.2 a 5.3 [13]). Adresáře Events, Jobs, Listeners a Policies byly odstraněny, jedná se pouze o stylistickou změnu zajišťující větší přehlednost, jež nemá žádný vliv na chod aplikace. I přes odstranění zmiňovaných složek zůstává framework zpětně kompatibilní.



Obrázek 10: Porovnání adresáře app ve verzích 5.2 a 5.3 [13]

V aktuální verzi 5.4 je adresář app rozšířen o volitelné složky, konkrétně o Events, Jobs, Listeners, Mail, Notification a Policies, které lze vytvořit příkazem *make*.

Konečná adresářová struktura frameworku Laravel verze 5.4 vypadá takto:

```
názevProjektu/
├── app/                ← adresář s aplikací
│   ├── Console         ← obsahuje veškeré příkazy Artisan
│   ├── Exceptions      ← obsahuje veškeré výjimky aplikace
│   ├── Http            ← zde se nachází logika: controllers, middleware a requests
│   ├── Providers       ← zahrnuje tzv. service providers, registrující události, routy…
├── bootstrap           ← spouštěcí soubor, nastavuje autoloading, obsahuje cache
├── config              ← adresář s konfiguračními soubory
├── database            ← zahrnuje veškeré migrations a seeds databáze
├── public              ← obsahuje soubor index.php, CSS soubory, obrázky…
├── resources           ← zahrnuje views aplikace
├── routes              ← obsahuje definice všech cest (routes) v aplikaci
├── storage   ← obsahuje předkompilované části aplikace např. Blade templates
├── tests     ← zahrnuje jednotkové testy
├── vendor    ← adresář obsahující veškeré závislosti
```

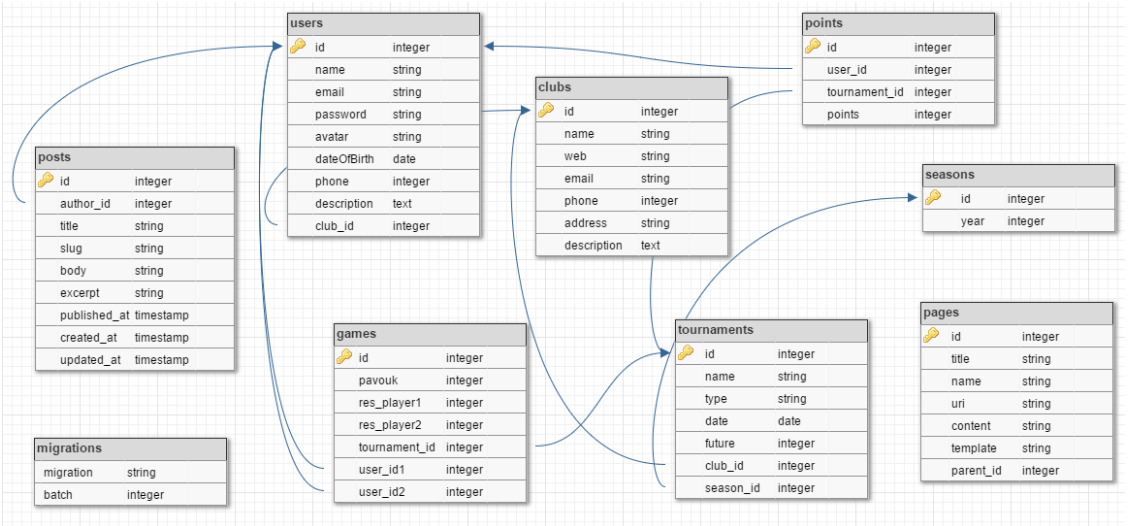
4 Webová aplikace

Pro tvorbu webové aplikace, sloužící k zobrazování výsledků turnajů a statistik ve hře ricochet, jsem si vybral PHP framework Laravel (popsán v kapitolách 3.2.3 Laravel a 3.3 Vlastnosti frameworku Laravel). K němu MySQL databázi, zajišťující datovou stránku projektu. Pro vývoj na vlastním počítači jsem použil prostředí pro vývoj webových aplikací na operačním systému Windows, zvané WampServer. Jedná se o prostředí podporující Apache2, PHP a MySQL spolu s phpMyAdmin rozšířením, pomocí kterého lze jednoduše spravovat databázi. Aplikaci jsem vyvíjel na operačním systému Windows 10.

Nejdříve jsem navrhl schéma databáze s potřebnými entitami a pomocí Migrations (popsané v kapitole 3.3.6 Migrations) frameworku Laravel vytvořil jak objekty ve frameworku, tak i dané entity v MySQL databázi.

4.1 Databáze

Hlavní stavební jednotky databáze, které jsem potřeboval vložit, byly jednotlivé turnaje, místa konání turnajů, hráči, zápasy a body do žebříčku.



Obrázek 11: Schéma databáze

Aby byla MySQL databáze správně propojená s frameworkem Laravel, musí názvy jednotlivých entit v databázi končit na písmeno s, jako indikátor množného čísla v anglickém jazyce. Další podmínkou je pojmenování cizích klíčů ve tvaru název tabulky (v jednotném čísle), na kterou odkazuje podtržítko id, takže například club_id.

Vytvořil jsem si schéma (viz Obrázek 11: Schéma databáze), obsahující všechny potřebné entity. Laravel si sám přidal tabulku migrations, kde uchovává záznamy o všech provedených migracích. Dále jsem k potřebným entitám přidal další dvě: posts a pages. V tabulce pages si ukládám navigační menu stránky a jeho volby. V tabulce posts jsou všechny vložené články.

Pracoval jsem s následujícími vztahy.

Hráč:

- Musí být členem právě jednoho klubu
- Může se zúčastnit libovolného počtu turnajů
- Může odehrát libovolný počet zápasů
- Může mít žádný, nebo více bodů z různých turnajů

Turnaj:

- Je odehraný právě v jedné sezóně
- Je pořádaný právě jedním klubem
- Má několik hráčů
- Má několik odehraných zápasů

Klub:

- Může mít žádného, nebo více registrovaných hráčů
- Může pořádat žádný, nebo více turnajů

Zápas:

- Musí být odehraný právě dvěma hráči
- Musí být odehraný na právě jednom turnaji

Následně jsem pomocí Migrations vytvořil entity v MySQL databázi a objekty ve frameworku. Každá tabulka nyní potřebuje pro práci v Laravelu svůj model. Modely se dají vytvořit ručně, ale jednodušší je vytvoření pomocí příkazové řádky a příkazu:

php artisan make:model NazevModelu

Dle pravidel musí název modelu začínat velkým písmenem a jmenovat se stejně, jako tabulka v databázi, akorát v jednotném čísle. Po vytvoření databáze pomocí Migrations a jednotlivých modelů zbývá naplnit databázi daty, aby měla s čím pracovat a nezobrazovala chybové kódy. K tomu jsem pro vložení náhodných dat použil třídu Seed (popsáno v kapitole 3.3.7 Seeding).

4.2 Backend

Tvorbu aplikace jsem začal vytvořením backendu – administrací celé webové aplikace. Přístup do administrace má pouze administrátor, který dále může spravovat všechny sekce webové aplikace. Pokud se do administrace přihlásí běžný uživatel, zobrazí se mu pouze jeho osobní profil s možností změny profilové fotografie – avataru.

Po přihlášení se administrátorovi zobrazí seznam naposledy zveřejněných článků a pět naposledy přihlášených uživatelů. Tento souhrn jsem nazval jako Dashboard. Dále administrace obsahuje sekci stránky, kde vytváří dynamické menu pro frontend aplikace a blog, kde lze vkládat články. Následují sekce hráči, turnaje a kluby, kde má administrátor možnost spravovat veškerá data týkající se sportu ricochet. Zmiňované sekce popisuji podrobněji v následujících kapitolách.

4.2.1 Články a stránky

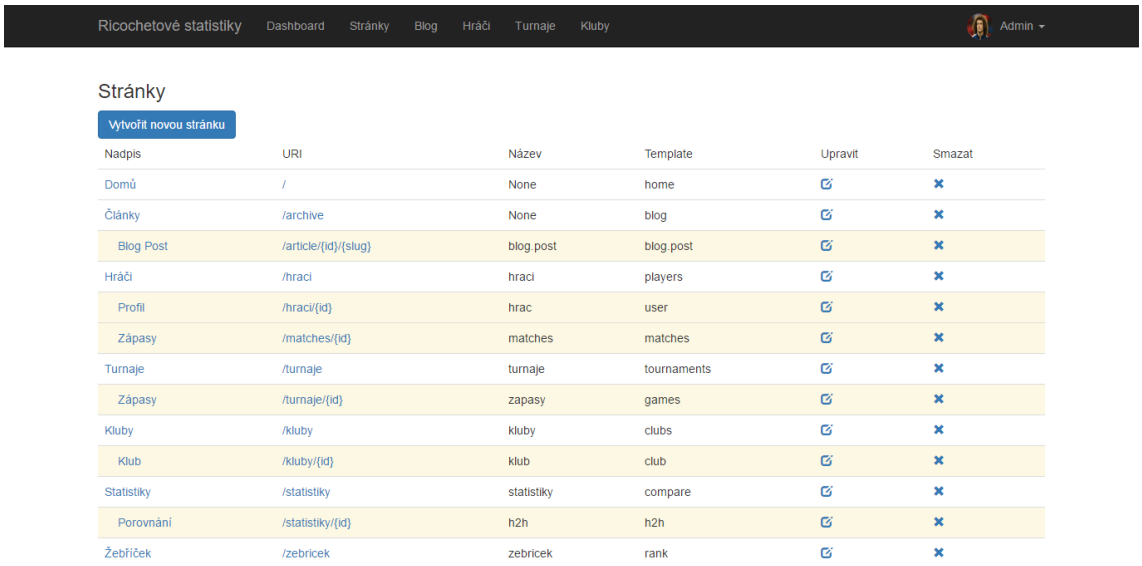
Nejdříve jsem se zaměřil na stránku jako na redakční systém. Bylo potřeba určit způsob, jakým vytvářet, upravovat a mazat články.

Model Post jsem si vytvořil již při vytvoření databáze. Pomocí

php artisan make:controller BlogController

jsem v adresáři app/Http/Controllers/Backend vytvořil controller s názvem BlogController, rozšiřující třídu Controller.

V něm jsem vytvořil funkci index, která se stará o zobrazení všech článků v backendu. Následně funkce create a store – pro vytvoření a uložení nového článku, edit a update – pro úpravu a uložení upraveného článku, a nakonec confirm a destroy, starající se o smazání článku. Funkce pracují pomocí Eloquent objektů (popsáno v kapitole 3.3.3 Eloquent ORM) přímo s MySQL databází.



Obrázek 12: Backend webové aplikace, zobrazení stránek

Poté jsem v adresáři app/Presenters vytvořil PostPresenter.php, který mimo jiné převádí pomocí třídy CommonMarkConverter markdown do HTML. Nyní bylo potřeba přidat routy do souboru routes.php, který se nachází v adresáři app/Http.

```
1. Route::get('backend/blog/{blog}/confirm',
2. ['as' => 'backend.blog.confirm', 'uses' => 'Bac-
   kend\BlogController@confirm']);
3. Route::resource('backend/blog', 'Backend\BlogController');
```

Zdrojový kód 5: Ukázka routování v aplikaci

Dále bylo potřeba v adresáři app/Http/Request vytvořit dva Request soubory pro uložení a update článku. Konkrétně StorePostRequest.php a UpdatePostRequest.php, oba popisují podmínky pro vložení záznamů do databáze pomocí funkce:

```
1. public function rules()
2. {
3.     return [
4.         'title' => ['required'],
5.         'slug' => ['required'],
6.         'published_at' => ['date_format:Y-m-d H:i:s'],
7.         'body' => ['required']
8.     ];
9. }
```

Zdrojový kód 6: Podmínky pro vložení dat do databáze

Jako poslední věc, týkající se článků, zbývalo vytvořit samotné Views pro zobrazení, ale i vytvoření a úpravu. V adresáři resources/views/backend/blog jsem využil nástroj Blade Templates (popsaný v kapitole 3.3.2 Blade Templates) a vytvořil tři sou-

bory: index.blade.php pro zobrazení všech článků, form.blade.php pro vytvoření a úpravu článků a confirm.blade.php pro smazání daného článku. Při vkládání/úpravě článku používám SimpleMDE markdown editor, který umožňuje základní funkce textového editoru, jako kurzívu, podtržení, tučné písmo, nadpisy, vkládání odkazů a obrázků atd.

Dále jsem se zaměřil na vytvoření stránek. Databázová tabulka pages, model Page, presenter PagePresenter.php a views v adresáři resources/views/backend/pages. Dá se říci, že jde o to samé, akorát neukládám články, ale stránky z navigačního menu. Mohu si vybrat, jestli je daná stránka v pořadí před, nebo až po jiné stránce. Dále mohu určit, jestli je „dítětem“ jiné stránky, to způsobí, že bude pod danou stránkou v rozbalovacím menu. A nakonec lze určit, jestli bude stránka zobrazena v navigaci nebo bude skrytá. Jedná se o dynamické navigační menu, uložené v databázi a následně vykreslované ve frontendu aplikace.

4.2.2 Hráči

Pro práci s hráči jsem vytvořil model User, implementující třídy AuthenticatableContract, AuthorizableContract a CanResetPasswordContract, které se starají o autentizaci a autorizaci hráče.

Dále používám dvě funkce, definující Eloquent vztahy (popsáno v kapitole 3.3.3 Eloquent ORM) mezi danými entitami v databázi.

```
1.  public function games()
2.  {
3.          return $this->hasMany("Proj\Game", "id");
4.  }
```

Zdrojový kód 7: Ukázka Eloquent ORM vztahu 1:M

Funkce games určuje, že se jedná o vazbu 1:M, tedy, že jeden uživatel může mít několik zápasů. První parametr funkce je cesta k danému modelu, druhý parametr je název cizího klíče. Pokud se cizí klíč jmenuje podle pravidel nazevTabulky_id, nemusí se zmiňovat a framework ho sám rozpozná. Zato funkce hrajeZa je inverzí vazby 1:M. V tomto případě definuje, že jeden hráč může hrát za právě jeden klub.

```
1.  public function hrajeZa()
2.  {
3.        return $this->belongsTo("Proj\Club", "club_id" );
4.  }
```

Zdrojový kód 8: Ukázka Eloquent ORM inverzního vztahu 1:M

Funkci hrajeZa používám například při zobrazení seznamu hráčů (viz Obrázek 13: Backend, zobrazení a správa hráčů), kdy se dotazuji na tabulku users a zároveň na tabulku clubs. Kód na zobrazení názvu klubu hráče vypadá takto:

```
1.  {{ $user->hrajeZa->name }}
```

Zdrojový kód 9: PHP kód pro zobrazení názvu klubu hráče

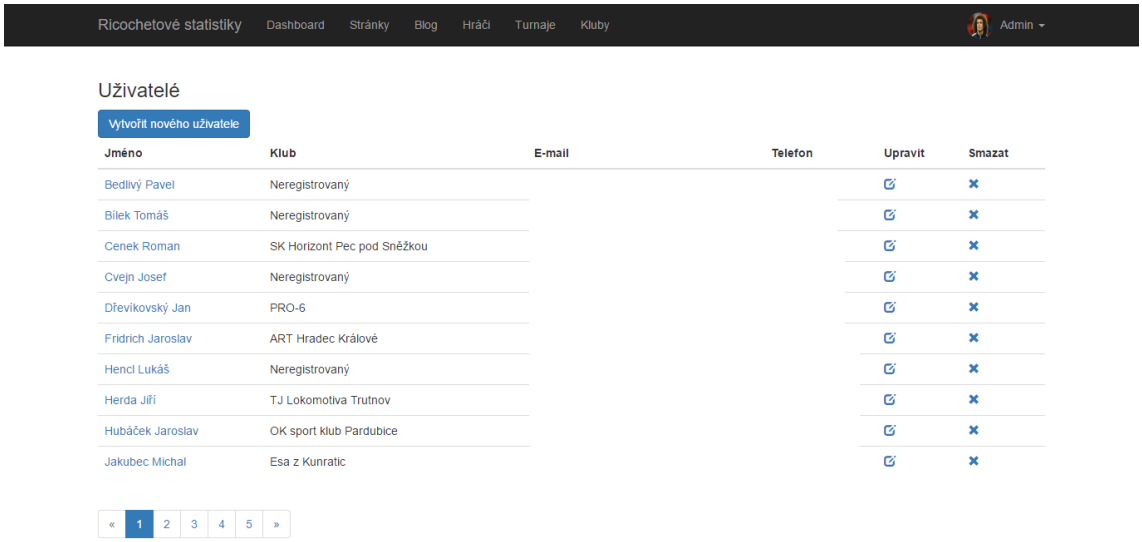
Výše zmiňovaný kód je ve foreach smyčce, nevyužívá se tedy indexů. Na objekt user se zavolá metoda hrajeZa a podle cizího klíče nalezne hodnotu name (název klubu) v tabulce clubs.

Myšlenka při tvorbě uživatelů byla taková, že pouze administrátor může přidávat nové hráče do webové aplikace, ať už nově registrované nebo stávající. Po vložení hráče do aplikace dostává hráč přístup do určitých částí backendu pomocí emailové adresy a hesla. Jediná sekce, do které má klasický uživatel přístup, je změna avataru hráče. Do ostatních sekcí – vkládání turnajů, hráčů, klubů atd. má přístup pouze administrátor. V controlleru UsersController.php získávám pomocí:

```
1.  $users = $this->users->where('id','>','1')->orderBy('name', 'asc')
2.          ->paginate(10);
```

Zdrojový kód 10: Načtení hráčů z databáze

všechny uživatele kromě administrátora (podmínka id > 1) seřazené podle jména a stránkované po 10 uživatelích (viz Obrázek 13: Backend, zobrazení a správa hráčů).



Obrázek 13: Backend, zobrazení a správa hráčů

Pole e-mail a telefon jsem kvůli zobrazení osobních údajů zakryl. K těmto údajům má přístup pouze administrátor webu. Libovolného uživatele může buď editovat, nebo smazat.

Při vytvoření nového hráče musí administrátor vyplnit následující políčka: jméno, národnost, email, datum narození, telefon, popis a vybrat ze seznamu klubů ten, ve kterém je daný hráč registrovaný.

4.2.3 Kluby

Kluby potřebuji v této webové aplikaci hned ze dvou důvodů. Prvním důvodem je uvedení organizátorů daných turnajů. Pokud se chce hráč zúčastnit turnaje, potřebuje vědět, kde se daný turnaj koná. Druhým důvodem je klubová příslušnost hráčů, kdy hráči zpravidla hrají za svůj konkrétní klub. V případě, že hráč nehraje za žádný klub, nachází se v sekci neregistrovaných.

Důležité informace klubu, které se vkládají při jeho vytvoření, jsou název, adresa, email a telefonní číslo. Hráči se ke svým konkrétním klubům přidávají při vkládání hráčů, ne při vkládání klubů.

V modelu Club používám metodu maHrace, jež propojuje daný klub s několika hráči (viz Zdrojový kód 7: Ukázka Eloquent ORM vztahu 1:M).

4.2.4 Turnaje, zápasy a body

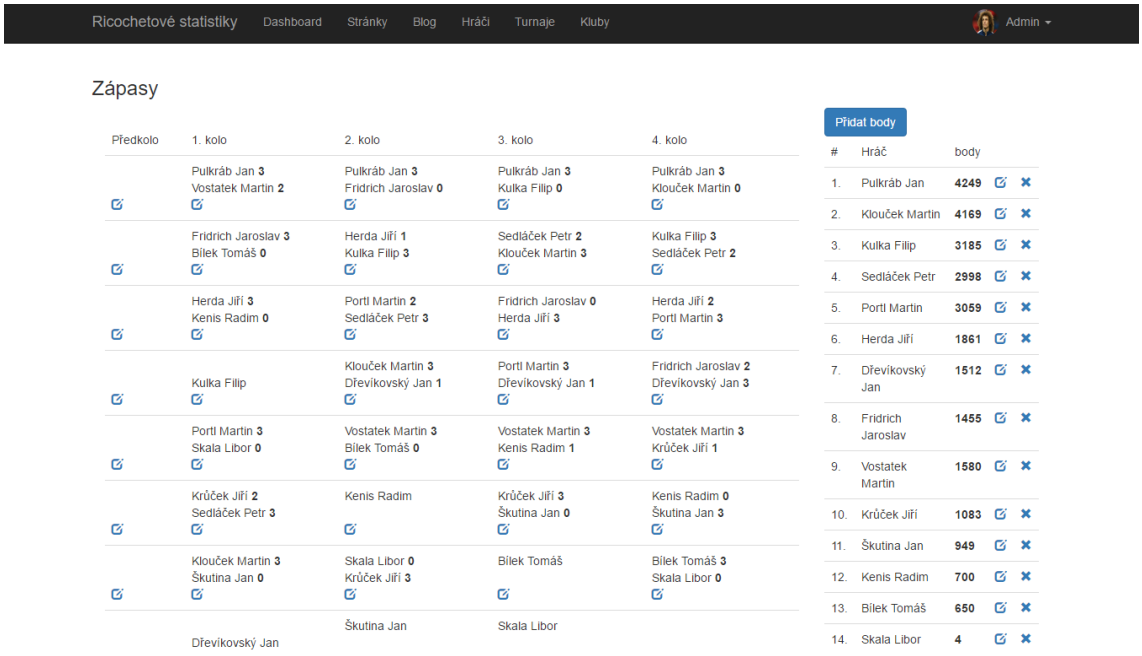
Tabulky turnajů, zápasů a bodů spolu velmi úzce souvisí. Každý turnaj musí mít nějaké zápasy, každý zápas musí mít právě jeden turnaj, stejně jako body musí být získány právě z jednoho turnaje.

Daný turnaj je pořádaný právě jedním klubem, v právě jedné sezóně. Model Tournament tedy obsahuje dvě funkce:

```
1.  public function season()
2.  {
3.          return $this->belongsTo("Proj\Season");
4.  }
5.  public function klub()
6.  {
7.          return $this->belongsTo("Proj\Club", "club_id" );
8.  }
```

Zdrojový kód 11: Model Tournament, práce s Eloquent ORM

Při vytváření nového zápasu používám funkce create a store. Funkce create načte z databáze všechny hráče ve formátu klíč a hodnota – jméno a id, seřazené podle jména. Turnaje načte ve formátu klíč a hodnota – název a id, seřazené podle id turnaje (viz Zdrojový kód 12: Vytvoření nového zápasu).



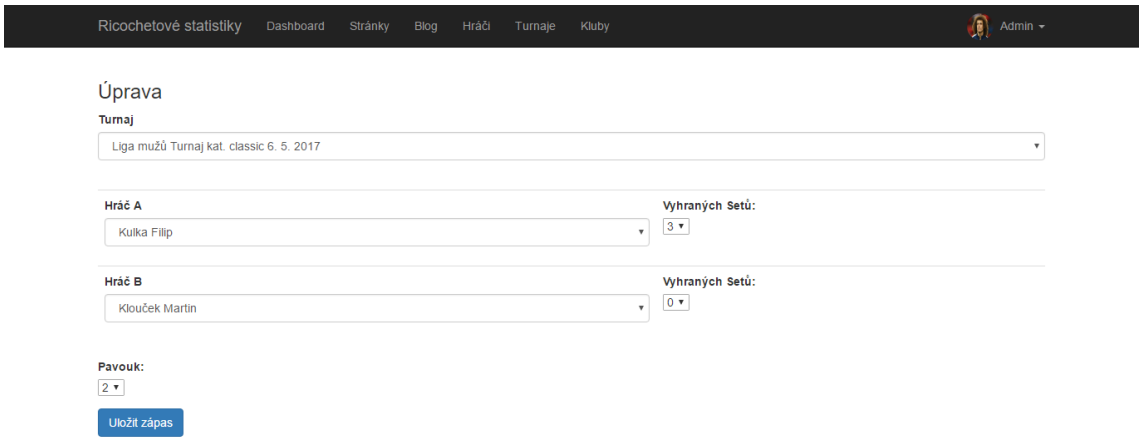
Obrázek 14: Backend, zobrazení pavouka turnaje se zápasy

Funkce store ukládá data do databáze, konkrétně id a počet vyhraných setů obou hráčů, id turnaje a hodnotu o tom, v jaké části turnaje (turnajového pavouka) byl zápas odehrán.

```
1.  public function create(Game $games)
2.  {
3.  $players = User::where('id','>','1')->orderBy('name','asc')
4.              ->lists('name', 'id');
5.  $tournaments = Tournament::orderBy('id', 'asc')->lists('name', 'id');
6.  return view('backend.games.form', compact('games', 'players', 'tournaments'));
7.  }
```

Zdrojový kód 12: Vytvoření nového zápasu

Body se hráčům přidávají na konci turnaje, kdy je známé konečné umístění všech zúčastněných. Tuto možnost jsem implementoval přímo k zobrazení všech zápasů v daném turnaji (viz Obrázek 14: Backend, zobrazení pavouka turnaje se zápasy), kde může administrátor jednoduše vložit daným hráčům body z turnaje. Body se ukládají do databáze ve formátu id turnaje, id uživatele, počet bodů a umístění hráče. Pokud tedy chci zjistit aktuální počet bodů daného hráče, sečtu všechny hodnoty počet bodů, kde se id hráče rovná id požadovaného hráče. Tímto způsobem později implementuji žebříček.



Obrázek 15: Backend, vložení zápasu

Doposud se turnaje, zápasy i body řešily na oficiální webové aplikaci České ricochetové asociace (popsáno v kapitole 1.4 E-ricochet.cz) jiným způsobem.

Webová aplikace E-ricochet.cz obsahuje pouze databázi uživatelů, klubů a žebříček. O vše ostatní se stará administrátor pomocí předem definovaných souborů typu excel. Do těchto souborů zadává jednotlivé výsledky zápasů a následně vygeneruje soubor typu PDF, který nahraje na server a v sekci daného turnaje na něj umístí odkaz. Dále administrátor ručně přepočítá body ze žebříčku, aktuálně získané body u každého hráče a zapíše je do excelového souboru. Ten následně exportuje do formátu CSV, nahraje na server a aplikace z něj následně čerpá data (jména hráčů a počet bodů). Výstup z turnaje je pouze PDF soubor s výsledky zápasů a CSV soubor s žebříčkem.

4.3 Frontend

K zobrazení webové aplikace jsem zvolil tři různé Templates, jeden pro domovskou stránku, druhý pro články a třetí pro zbytek stránek. Všechny tři mají společný prvek, kterým je dynamické navigační menu v horní části stránky. Na domovské stránce se zobrazují nejnovější články a boční menu obsahující žebříček s pěti nejlepšími hráči, výpis sekcí a uživatelský profil. Pokud je uživatel přihlášen, vidí v uživatelském profilu svůj avatar, odkaz na profil a možnost odhlášení se. Pokud uživatel přihlášený není, je zde tlačítko pro přihlášení se. Zbytek stránek má stejné rozložení jako stránka domovská, jen místo nejnovějších článků obsahuje data dané sekce. Výpis článků a jednotlivé články neobsahují boční menu.

Aplikační logiku frontendu řeším v adresáři app/Templates, kde má každá třída svůj Template soubor implementující AbstractTemplate.php. V něm mám předdefino-

vanou funkci prepare, která vždy pošle daná data přímo do pohledu. Jednotlivé pohledy ukládám do složky resources/views/templates, kde má každý Template soubor svůj pohled typu Blade (popsáno v kapitole 3.3.2 Blade Templates). Routy pro frontend řeším v souboru config/cms.php, kde zvlášť definuji každou třídu, například uživatele tímto způsobem:

```
1.  'user' => Proj\Templates\UserTemplate::class,
```

Zdrojový kód 13: Frontend, ukázka routování

4.3.1 Hráči

Po otevření sekce hráči se uživateli zobrazí výpis všech hráčů v tabulce, spolu s klubem, za který hrají.

V souboru PlayersTemplate.php načítám hráče z databáze, seřazené dle jména, které následně posílám pohledu typu Blade. Po rozkliknutí hráče se zobrazí jeho profil, na který se odkazuje pomocí id hráče (viz Obrázek 16: Frontend, profil hráče).

Obrázek 16: Frontend, profil hráče

U každého hráče je vypsána klubová příslušnost, rok narození, aktuální forma a počet bodů v žebříčku. U aktuální formy jsem se inspiroval serverem Livesport.cz (popsaný v kapitole 1.1 Livesport.cz), kde se zobrazí 5 posledních zápasů hráče ve formátu zelené W, pokud hráč zápas vyhrál a červené L v případě, že hráč zápas prohrál.

Ve vedlejší části je zobrazen avatar hráče, pokud si hráč žádnou osobní fotografii nenahrál, je defaultně zobrazen šedobílý obrys muže.

Následně jsou zobrazeny statistiky, jako počet odehraných zápasů, počet vyhraných zápasů a procentuální úspěšnost. Po rozkliknutí odkazů z těchto statistik se uživateli zobrazí výpis všech odehraných zápasů daného hráče.

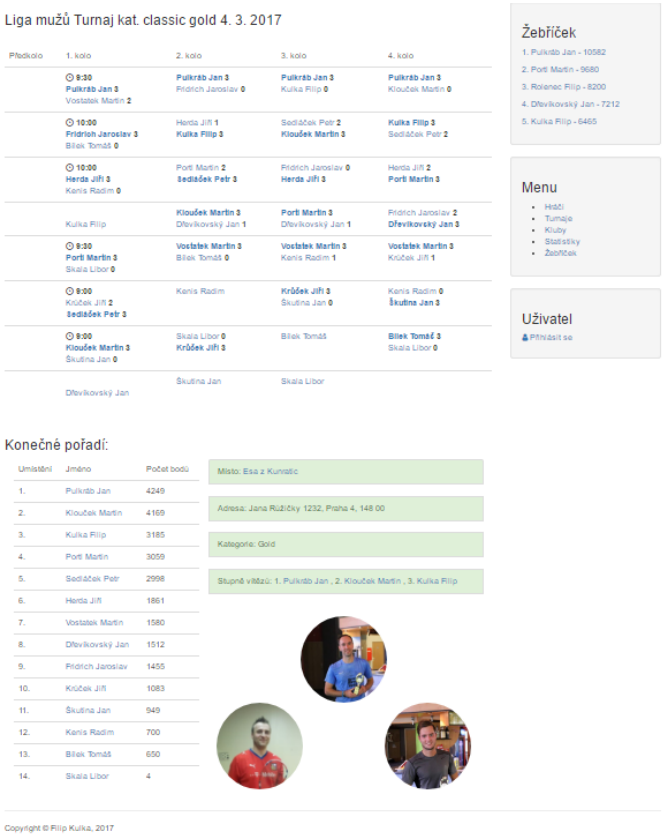
Jako další je zobrazen stručný popis hráče a pod ním výpis jeho 10 posledních utkání. Pokud hráč neodehrál žádné zápasy, zobrazí se hláška, že hráč zatím žádný zápas neodehrál.

4.3.2 Turnaje

Sekce turnaje je rozdělena na tři části. Každá z částí je zobrazena tabulkou s hodnotami název turnaje, typ turnaje a místo konání. V první části se nachází zatím neodehrané turnaje. Druhá část obsahuje již odehrané turnaje z aktuální sezóny. Ve třetí části je výpis všech odehraných turnajů z předešlých sezón.

Neodehrané turnaje se dále dělí na dva typy. Již rozlosované (případně aktuálně hrané) nebo neodehrané. Po rozkliknutí neodehraného turnaje se zobrazí hláška, že turnaj zatím nebyl odehrán. Pokud uživatel rozklikne turnaj, který je rozlosován nebo se aktuálně hraje, zobrazí se mu pavouk turnaje s jednotlivým nasazením hráčů do zápasů. Pro první kolo zde lze nalézt i informaci o časech začátků prvních zápasů. Pod turnajovým pavoukem se nachází místo konání a typ turnaje.

Odehrané turnaje jsou zobrazeny stejným způsobem, jako turnaje rozlosované nebo aktuálně hrané. Navíc obsahují informace o konečném pořadí všech hráčů se získanými body a graficky znázorněné stupně vítězů v podobě zobrazení hráčských avatarů (viz Obrázek 17: Frontend, zobrazení turnaje a odehraných zápasů). Vítěz zápasu je vždy pro lepší přehlednost zobrazen tučným písmem.



Obrázek 17: Frontend, zobrazení turnaje a odehraných zápasů

Data zpracovávám v souborech TournamentsTemplate.php pro výpis všech turnajů a GamesTemplate.php pro zobrazení jednotlivého turnaje s odehranými zápasy.

4.3.3 Kluby

V sekci kluby vypisuji seznam všech ricochetových klubů. Po rozkliknutí jednotlivého klubu se zobrazí detailní profil klubu s následujícími informacemi: adresa, telefonní číslo, email, webové stránky a popis.

Ve vedlejší sekci jsem implementoval mapu, na které je dané ricochetové centrum zobrazeno. Použil jsem knihovnu pro framework Laravel, zvanou GooglMapper [9], jež využívá API Google Maps. Pro správný chod bylo mimo jiné potřeba vygenerovat Google API key a vložit ho do konfiguračního souboru. V souboru ClubTemplate.php generuji mapu pomocí:

```
1.  Mapper::location($club->adress)->map(['zoom' => 15]);
```

Zdrojový kód 14: Generování mapy zobrazující adresu daného klubu

kde do lokace vkládám hodnotu String z databáze a přibližuji cílový bod na mapě na hodnotu 15. Mapu následně v pohledu vykresluji:

```
1.  <div style="height: 250px" class="col-md-6"> {!! Mapper::render() !!} </div>
```

Zdrojový kód 15: HTML kód pro zobrazení mapy

Přednastavená výška na 250px je v tomto případě velmi důležitý atribut, jelikož daná knihovna nedokáže mapu vykreslit, pokud se nevytvoří v divu s předdefinovanou výškou větší než 200px.

Ve spodní části profilu klubu je výpis všech registrovaných hráčů daného klubu. Hráče získávám z databáze následujícím způsobem:

```
1.  $user = $this->user->where('club_id', '=', $parameters['id'])
2.          ->where('id', '>', '1')->orderBy('name', 'asc')->get();
```

Zdrojový kód 16: Načítání hráčů daného klubu

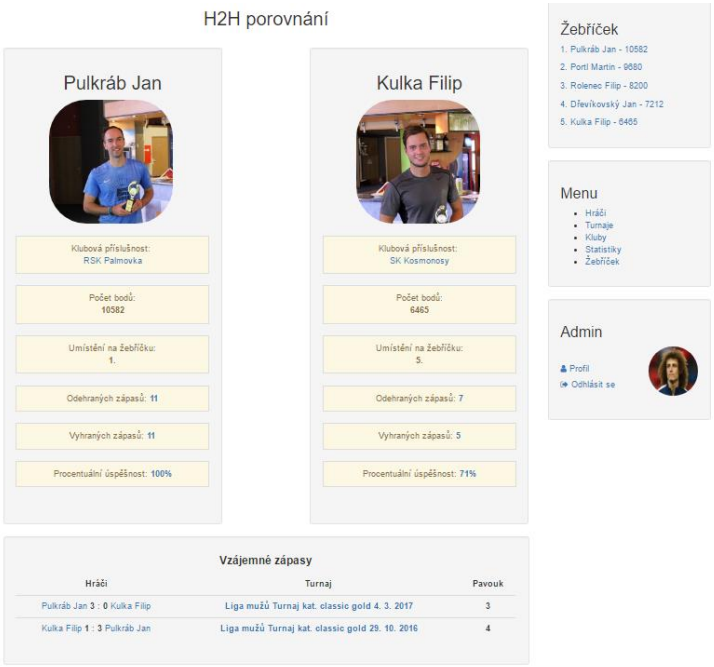
Načítám všechny uživatele kromě administrátora, kteří mají hodnotu club_id stejnou, jako je id klubu, seřazené podle jména.

4.3.4 Statistiky

V sekci statistiky může uživatel porovnat výkony dvou vybraných hráčů a jejich vzájemné zápasy. Tomuto porovnání se říká H2H neboli head to head. V tomto aspektu webové aplikace jsem se opět inspiroval serverem Livesport.cz (viz Obrázek 3: Zobrazení H2H porovnání na Livesport.cz).

Poté, co si uživatel vybere dva hráče ze seznamu všech hráčů, se mu zobrazí detailní porovnání.

Jak je vidět na Obrázek 18, v první části je zobrazena klubová příslušnost, počet bodů, umístění v žebříčku, počet odehraných a vyhraných zápasů a procentuální úspěšnost hráče. Ve spodní části jsou vypsány všechny společné zápasy vybraných hráčů s konečnými výsledky, turnajem a označením kola, ve kterém byl zápas odehrán.



Obrázek 18: Frontend, H2H porovnání dvou hráčů

Všechny společné zápasy dvou hráčů získávám z databáze v souboru H2HTemplate.php příkazem:

```
1.  $matches = $this->game->where('user_id', '=', Input::get('player1'))
2.           ->where('vysl_hrac1', '!=' , ' ')
3.           ->where('user_id2', '=', Input::get('player2'))
4.           ->where('vysl_hrac1', '!=' , ' ')
5.           ->orWhere('user_id', '=' , Input::get('player2'))
6.           ->where('vysl_hrac1', '!=' , ' ')
7.           ->where('user_id2', '=' , Input::get('player1'))
8.           ->where('vysl_hrac2', '!=' , ' ')->orderBy('id_turnaje', 'desc')
9.           ->orderBy('pavouk', 'desc')->paginate(100);
```

Zdrojový kód 17: Načítání společných zápasů dvou hráčů

Nejdříve načítám všechny zápasy, které odehrál hráč 1 a které byly odehrány, následně již z vybraných zápasů vybírám zápasy, které odehrál hráč 2. Avšak jelikož může nastat situace, kdy hráč 2 byl vložen na první místo v zápisu utkání a hráč 1 na druhé, tento příkaz by daný zápas nevybral. Proto pokračuji dalším dělením orWhere, které definuje, že se má provést jak první část příkazu, tak i část druhá. V druhé části je pouze přehozen hráč 1 za hráče 2. Následně jsou všechny zápasy (jak z první, tak z druhé části) seřazeny podle data odehrání a turnajového pavouka.

4.3.5 Žebříček

Žebříček slouží k nasazení hráčů na turnaje. Podle aktuálního postavení se rozdělují hráči do turnajového pavouka. Dále slouží ke každoročnímu ocenění, předávajícímu se na konci sezóny, které získává hráč s nejvíce body.

V žebříčku se sčítají všechny získané body za celou sezónu. To provádím v souboru RankTemplate.php:

```
1.  for ($i = 0; $i < ($number_of_users); $i++)
2.  {
3.      $point[$i] = ['id' => $i,'name' => (DB::table('users')
4.                  ->where('id', '=', $i)->value('name')),
5.                  'points' => (DB::table('points')
6.                  ->where('user_id', '=', $i)
7.                  ->sum('points') ), 'club' => (DB::table('users')
8.                  ->where('id', '=', $i)->value('club_id')) ];
9.  }
```

Zdrojový kód 18: Počítání bodů jednotlivých uživatelů, tvorba žebříčku

Tento for cyklus proběhne tolikrát, kolik je počet hráčů a každému hráči sečte všechny body, které obdržel. Vznikne pole s hodnotami id, jméno uživatele, počet bodů a klub. Nyní potřebuji pole setřídit podle počtu bodů, to provedu následujícím příkazem:

```
1.  $point = collect($point)->sortBy('points')->reverse()->toArray();
```

5 Získání dat

Po vytvoření webové aplikace s testovacími daty jsem potřeboval obstarat reálná data. Informace o klubech byly lehce dohledatelné, náročnější bylo získat osobní informace hráčů. Neexistoval totiž žádný seznam registrovaných hráčů. Na serveru E-ricochet.cz se nachází pouze výpis hráčů s hodnotami jméno, klub a občas rok narození. V mé aplikaci však každý uživatel musí mít vyplněny ještě navíc položky jako je email a telefonní číslo. Poprosil jsem tedy o spolupráci Ing. Jana Pulkrába, jakožto hlavního správce serveru E-ricochet.cz a místopředsedu České ricochetové asociace. Každý hráč se totiž musí nějakým způsobem přihlásit na jednotlivý turnaj. Přihlášky fungují ve stylu vyplnění formuláře, který po potvrzení odešle data na emailovou schránku asociace. Jedná se tedy o stovky emailových zpráv, většinou částečně nevyplněných. Formulář totiž není dostatečně ošetřený, a pokud uživatel například do pole datum narození vloží cokoliv jiného, například nějaké slovo, i tak data odešle.

Na jednom z turnajů mi byl zapůjčen počítač s přístupem na email asociace, díky čemuž jsem mohl z přijatých emailů sestavit databázi hráčů.

Po vytvoření hráčských účtů jsem byl schopen zadávat reálné výsledky z odehraných turnajů. K tomu mi posloužily PDF soubory ze serveru E-ricochet.cz, kde byly rozepsány všechny odehrané zápasy i s výsledky. Vytvořil jsem všechny turnaje ze sezóny 2016/2017 se všemi odehranými zápasy. Jedná se o 8 turnajů a 256 odehraných zápasů.

6 Implementace

Posledním úkolem bylo aplikaci implementovat na libovolný webový hosting a následně otestovat. Zvolil jsem adresu www.ricostats.cz, kde se webová aplikace nachází.

Vybral jsem si webhosting FORPSI, který mne zaujal především z hlediska ceny. Oproti konkurenci nabízí velmi levný balíček s .cz doménou prvního řádu, neomezeným prostorem pro web, neomezenou MySQL databází a podporou PHP verze 7.0.

Pro implementaci webové aplikace na webhostingu od firmy FORPSI jsem použil návod [10], který zmiňuje nutnost umístění aplikační logiky a adresáře /public na rozdílná místa. Inicializační soubor index.php nesmí být ve stejném adresáři, jako zbytek souborů, mezi něž patří například soubor .env, ve kterém se nachází důležitá konfigurační data webové aplikace, například k databázi a e-mailu.

Dále bylo potřeba exportovat vytvořenou databázi z mého localhostu na server. Pomocí funkce export v prostředí phpMyAdmin jsem vyexportoval soubor s koncovkou .sql. Po importování tohoto souboru na server se však zobrazila chybová hláška a import se neprovedl. Později jsem zjistil, že použité kódování z mého localhostu nebylo dostupné na hostingu od firmy FORPSI. To však nebyla jediná chyba. Druhá, o něco důležitější, byla chybějící podpora triggerů. Ve webové aplikaci jsem používal trigger na událost AFTER INSERT do tabulky tournaments, kdy se měly vytvořit záznamy v tabulce games. Kvůli chybějící podpoře triggerů jsem byl nucen tuto akci realizovat v kódu aplikace, a ne pomocí SQL.

Po nahrání souborů pomocí FTP na hosting a změně souborů dle výše zmiňovaného návodu aplikace stále nefungovala. Hosting vracel chybovou hlášku 505. Později jsem zjistil, že FORPSI nepodporuje volbu -MultiViews v souboru .htaccess.

Další atribut, potřebný pro možnost nahrání frameworku Laravel na sdílený hosting, je vygenerování aplikačního klíče. Ten se vygeneruje pomocí příkazového řádku a příkazu: *php artisan key:generate*. Klíč je potřeba vložit do výše zmiňovaného konfiguračního souboru .env, který se nachází v kořenovém adresáři frameworku.

Všechny problémy, které se vyskytly při nahrání aplikace na hosting, jsem úspěšně vyřešil. Jediná část aplikace, o kterou jsem po nahrání přišel, byl již zmíněný trigger. Tento problém jsem byl nucen řešit přímo v kódu aplikace.

7 Testování

Testování aplikace proběhlo dne 6. 5. 2017 na republikovém turnaji v Trutnově. Nejdříve bylo potřeba vytvořit daný turnaj a vložit do něj hráče, kteří se na turnaj přihlásili. Informace jsem čerpal z aplikace E-ricochet.cz, na níž vždy před turnajem administrátor nahraje PDF soubor s rozlosováním. Před samotným turnajem, hned při vkládání hráčů, jsem narazil na dva nedostatky aplikace.

Pokud jsem u turnaje chtěl zpřístupnit zobrazení turnajového pavouka, přesunul se do sekce odehrané turnaje. Turnaj však odehraný nebyl, pouze bylo zpřístupněno rozlosování. Přesun mezi sekcemi, ale i zobrazení pavouka, zajišťovala proměnná future, podle které se rozhodovalo, co zobrazit a do jaké sekce turnaj zařadit. Tento problém jsem vyřešil přidáním proměnné los, určující, zda byl turnaj rozlosován nebo ne. Pokud ano, zobrazí se turnajový pavouk. Pokud ne, zobrazí se hláška, že turnaj nebyl odehrán. Jelikož proměnná future nebyla změněna, zůstane turnaj i po rozlosování mezi neodehranými turnaji.

Druhým nedostatkem bylo chybějící zobrazení hracího času u zápasů prvního kola. Čas se zobrazuje pouze u prvních zápasů, aby hráči věděli, v kolik hodin mají na turnaj přijet. Přidal jsem do tabulky zápasů další pole, do kterého může administrátor zadat časy. Ty se následně zobrazí přímo vedle konkrétních zápasů.

Při vkládání výsledků jednotlivých zápasů jsem neshledal žádné nedostatky. Po vložení zápasů do aplikace se výsledky zobrazily jak v daném turnaji, tak v hráčských profilech a porovnáních.

Od uživatelů sledujících výsledky online jsem dostal pouze pozitivní zpětnou vazbu. Jediná poznámka ke zlepšení padla na malou propagaci aktuálně hraného turnaje na domovské stránce aplikace. Tuto věc lze řešit vložením článku o aktuálním turnaji.

Na výše zmiňované chyby se přišlo až za běhu aplikace, kdy se s reálnými daty dostává uživatel do situací, které při tvorbě aplikace nepředvídal. Všechny nalezené nedostatky byly opraveny a aplikace je nyní plně funkční v rozsahu, který byl před její tvorbou navržen.

Závěr

Provedl jsem rešerši webových stránek orientovaných na zobrazování sportovních výsledků, při které jsem se inspiroval určitými aspekty daných webových aplikací. Následně jsem v jazyce PHP, konkrétně ve frameworku Laravel, vytvořil webovou aplikaci pro zobrazování výsledků ve hře ricochet. Vycházel jsem z aktuálního webu České ricochetové asociace E-ricochet.cz, kde se nachází výsledky z turnajů a ručně zpracovaný republikový žebříček. Ve své aplikaci jsem implementoval zobrazení jednotlivých zápasů, turnajů, automatickou tvorbu republikového žebříčku, zobrazení aktuální formy hráčů a porovnání dvou hráčů s jejich společnými zápasy.

Webovou aplikaci jsem naplnil reálnými daty ze sezóny 2016/2017 a nahrál na adresu www.ricostats.cz. Po domluvě s místopředsedou České ricochetové asociace, Ing. Janem Pulkrábem, by měla tato webová aplikace od sezóny 2017/2018 nahradit stávající aplikaci, jež je velice náročná na správu a financování.

Do budoucna plánuji rozšířit webovou aplikaci o další funkce, ke kterým jsem se v rozsahu této diplomové práce nedostal. Mezi nimi by mohlo být např. přihlašování se na turnaje přímo z webové aplikace, nebo správa mezinárodních turnajů a žebříčku.

Použitá literatura

[1] PHP - Getting Started - Manual. *PHP: Hypertext Preprocessor* [online]. [cit. 2017-04-24]. Dostupné z: http://php.net/manual/en/getting-started.php.

[2] Usage of server-side programming languages broken down by ranking. *W3techs - Web Technology Surveys* [online]. [cit. 2017-04-24]. Dostupné z: https://w3techs.com/technologies/cross/programming_language/ranking.

[3] CROFT, Jeff. Frameworks for Designers [online]. 2007 [cit. 2017-04-25]. Dostupné z: hhttp://alistapart.com/article/frameworksfordesignersi.

[4] The Best PHP Framework for 2015: SitePoint Survey Results. *SitePoint* [online]. [cit. 2017-04-24]. Dostupné z: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/.

[5] Úvod do Nette frameworku pro PHP. *Itnetwork.cz* [online]. [cit. 2017-04-25]. Dostupné z: https://www.itnetwork.cz/php/nette/zaklady/uvod-do-php-frameworku-nette/.

[6] Blade Templates. *Laravel* [online]. [cit. 2017-04-24]. Dostupné z: https://laravel.com/docs/5.4/blade.

[7] STAUFFER, Matt. *Laravel: Up and Running: A Framework for Building Modern PHP Apps*. O'Reilly Media, 2016. ISBN 9781491936061.

[8] Česká asociace squashe – žebříček muži. *Česká asociace squashe* [online]. [cit. 2017-04-25]. Dostupné z: https://www.czechsquash.cz/zebricek/dospeli/muzi/#zebricek.

[9] *GooglMapper* [online]. [cit. 2017-04-20]. Dostupné z: https://github.com/bradcornford/Googlmapper.

[10] *How to deploy laravel project in shared hosting* [online]. [cit. 2017-04-20]. Dostupné z: http://justlaravel.com/how-to-deploy-laravel-project-shared-hosting/.

[11] SINHA, Sanjib. Beginning Laravel: A beginner's guide to application development with Laravel 5.3. 1. Apress, 2017. ISBN 978-1-4842-2537-0.

[12] MCCREARY, Jason. All Aboard for Laravel 5.1 [online]. s. 64 [cit. 2017-05-08].

[13] Laravel 5.3 changes the “app” folder. Laravel News [online]. [cit. 2017-05-07]. Dostupné z: https://laravel-news.com/laravel-5-3-changes-app-folder.

[14] E-ricochet.cz. *Oficiální web České ricochetové asociace* [online]. [cit. 2017-05-08]. Dostupné z: http://www.e-ricochet.cz/.

Přílohy

A Adresářová struktura frameworku Nette

```
sandbox/
├── app/                        ← adresář s aplikací
│   ├── config/                 ← konfigurační soubory
│   │   ├── config.neon         ← konfigurační soubor
│   │   └── config.local.neon
│   │
│   ├── forms/                  ← třídy formulářů
│   ├── model/                  ← modelová vrstva a její třídy
│   ├── presenters/             ← třídy presenterů
│   │   ├── HomepagePresenter.php       ← třída presenteru Homepage
│   │   └── templates/          ← adresář se šablonami
│   │       ├── @layout.latte           ← šablona společného layoutu
│   │       └── Homepage/               ← šablony presenteru Homepage
│   │           └── default.latte       ← šablona akce default
│   ├── router/                 ← třídy routerů
│   │
│   └── bootstrap.php           ← zaváděcí soubor aplikace
│
├── log/                        ← obsahuje logy, error logy atd.
├── temp/                       ← pro dočasné soubory, cache, ...
│
├── vendor/                     ← adresář na knihovny (např. třetích stran)
│   ├── nette/                  ← všechny knihovny Nette Frameworku
│   │   └── nette/Nette         ← oblíbený framework nainstalovaný Composerem
│   ├── ...
│   │
│   └── autoload.php    ← soubor starající se o načítání tříd nainstalovaných balíčků
│
└── www/                        ← veřejný adresář, document root projektu
    ├── .htaccess               ← pravidla pro mod_rewrite
    ├── index.php               ← který spouští aplikaci
    └── images/                 ← další adresáře, třeba pro obrázky
```

B Adresářová struktura frameworku Symfony

├── app ← konfigurace, šablony a případné překlady
│ ├── cache ← cache zkompilovaných šablon
│ │ ├── dev ← vývojové prostředí
│ │ └── prod ← produkční prostředí
│ ├── config ← nastavení, většinou v YAML
│ ├── logs ← logy
│ ├── Resources ← zdroje
│ │ └── views ← šablony
├── src ← většina kódu MVC
│ ├── bundles ← bundles jsou do jisté míry samostatné části webu zajišťující jeho funkčnost. Obsahuje podadresáře pojmenované podle jednotlivých bundlů, každý z nich může mít vlastní podadresář pro controllery, adresář pro testy atd.
├── vendor ← závislosti dodavatelů třetích stran
├── web ← adresář přístupný z internetu
│ ├── bundles ← v tomto adresáři jsou bundles implementované v src, zde již s vlastními zdroji (obrázky, CSS, …).