VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM HÁZENKÁŘSKÉHO KLUBU

INFORMATION SYSTEM OF A HANDBALL CLUB

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PATRIK HOMA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Homa Patrik**
Program: Informační technologie
Název: **Informační systém házenkářského klubu**
Information System of a Handball Club
Kategorie: Informační systémy

Zadání:

1. Seznamte se s principy vývoje webových aplikací, dostupnými prostředími a frameworky.
2. Analyzujte požadavky na informační systém házenkářského klubu zahrnující správu jednotlivých týmů a zápasů, přihlašování na tréninky s využitím automatických emailů a evidenci jízd včetně příslušných dokladů.
3. Navrhněte informační systém dle požadavků z bodu 2. Návrh konzultujte s vedoucím.
4. Implementujte navržený informační systém a otestujte jeho funkčnost na vhodném vzorku dat.
5. Zhodnoťte dosažené výsledky a další možné pokračování tohoto projektu.

Literatura:

- Welling, L., Thomson, L.: PHP a MySQL: Kompletní průvodce vývojáře. CPress, 2017.
- Dokumentace k frameworku Laravel. Dostupné na https://laravel.com/docs/5.8.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 24. října 2019

Abstrakt

Tato práce se zabývá vytvořením informačního systému pro házenkářský klub. Informační systém je vytvořen jako webová aplikace implementovaná v jazyce PHP s využitím frameworku Laravel. Práce si klade za cíl vytvořit takový informační systém, který ušetří administrativní práci zástupcům házenkářského klubu a pomůže jim při vedení klubu. Všechny funkcionality plynou z reálných potřeb házenkářského klubu. Hlavními funkcemi systému jsou evidence statistik hráčů v zápasech, přihlašování na klubové akce a evidence výdajových dokladů. Systém byl úspěšně vytvořen a otestován.

Abstract

This thesis deals with the creation of an information system for a handball club. The information system is created as a web application implemented in PHP and using the Laravel framework. The thesis aims to create such an information system that will save administrative work for the representatives of the handball club and help them in leading the club. All functionality resulting from the real needs of the handball club. The main functions of the system are the registration of player statistics in matches, registration for club events and the registration of expenditure documents. The system was successfully created and tested.

Klíčová slova

Webová aplikace; PHP; Laravel; Bootstrap; Sportovní klub, Informační systém, házená

Keywords

Web application; PHP; Laravel; Bootstrap; Sports club, information system, handball

Citace

HOMA, Patrik: *Informační systém házenkářského klubu*, Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Informační systém házenkářského klubu

Prohlášení

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

……
Patrik Homa
15. května 2020

Poděkování

Zde bych chtěl poděkovat svému vedoucímu panu Ing. Vladimíru Bartíkovi Ph.D. za poskytnuté konzultace a rady a také své rodině, přítelkyni a přátelům za podporu.

Obsah

Kapitola 1

Úvod

Tato bakalářská práce představuje tvorbu a finální výsledek informačního systému pro druholigový házenkářský klub TJ Dolní Cerekev. Když tento klub v roce 2016 postoupil do třetí nejvyšší házenkářské soutěže, ukázalo se, že je v souvislosti s tímto postupem nezbytné upravit mimo jiné zpracovávání administrativy organizace a fungování tohoto klubu. Organizace fungování klubu sice samozřejmě vždy na nějaké bázi fungovala, ale jelikož se jedná o činnost nevýdělečnou a dobrovolnou, bylo nutné zejména procesy administrativy zefektivnit tak, aby uživatelé optimalizovali čas, který administrativě věnují. Cílem bakalářské práce je vytvořit zcela nový informační systém, který by vyhovoval potřebám moderního sportovního klubu. Aby autor práce správně identifikoval funkcionality, které má webová aplikace umět, realizoval nejprve dotazníkové šetření, v rámci něhož se dotazoval hráčů a funkcionářů klubu na všechny potřebné funkce a požadavky na nový systém tak, aby komplexně pojmul celou potřebnou oblast. Vzhledem k tomu, že doposud se veškerá administrativa nezbytná k fungování klubu zaznamenávala ručně tužkou na papír, uvítali všichni dotázaní možnost správy administrativy pomocí webové aplikace. Navrhovaná webová aplikace by byla využívána jak trenéry a funkcionáři klubu, tak hráči, přičemž každý z nich bude využívat různé její funkcionality.

Hlavními funkcemi systému jsou evidence statistik hráčů v zápasech, přihlašování na klubové akce a evidence výdajových dokladů. Tyto a mnohé další funkce vzešly jako požadavky z dotazníkového šetření. Autor práce uvádí, že tato práce si klade za cíl nastavit základní rámec pro webovou aplikaci, která bude moci být následně v budoucnu rozšířena o další funkcionality dle potřeby sportovního klubu.

Práce je systematicky členěna do kapitol, které jsou nejprve teoretického charakteru a následně se věnují aplikaci teoretických poznatků a konkrétnímu postupu při tvorbě webové aplikace. Ve druhé kapitole autor popisuje výběr nejvhodnějšího jazyka a technologií. Autor se zabývá principy tvorby webových aplikací a zaměřuje se především na silné a slabé stránky jednotlivých variant, které připadají při jejich tvorbě v úvahu.
Třetí kapitola se věnuje použitým technologiím. Kapitola představuje jeden z nejužívanějších PHP frameworků Laravel, který je v práci využit. Dále je v této kapitole popisován CSS Framework Bootstrap, nástroj pro správu knihoven a balíčků Composer a grafický jazyk pro zobrazení a navrhování programových systémů.

Analýza požadavků zadavatele na systém je obsažena ve čtvrté kapitole této práce, která je jednou z nejvýznamnějších, protože představuje základ vytvářené webové aplikace. Kapitola se věnuje administrativě uživatelů, týmů a zápasů. Dále jsou představeny požadavky zápisu a vyhodnocení zápasových statistik, organizace tréninků, zápasů a klubových akcí, evidence dokladů vydaných na jízdné, rozhodčí a organizaci akcí. Závěr kapitoly bude věnován diagramu případu užití pro role hráče a role vedoucího klubu.

Pátá kapitola této práce je věnována návrhu informačního systému, který v sobě zahrnuje návrh struktury databáze pomocí ER (entity relationship) diagramu.

Následující šestá kapitola vychází z přechozích kapitol a využívá teoretických poznatků, analýzy požadavků a návrhu databázových tabulek pro samotnou implementaci systému.

Sedmá kapitola se zaměřuje na komplexní otestování informačního systému pomocí uživatelského testování. Autor v této fázi zapojil do testování především zástupce házenkářského klubu TJ Dolní Cerekev, jakožto budoucí pravidelné uživatele informačního systému.

V závěru autor hodnotí naplnění cíle práce a představí silné a slabé stránky informačního systému. Dále se autor zabývá možným budoucím vývojem systému a případnými možnostmi jeho rozšíření.

Kapitola 2

Principy tvorby webových aplikací

V následující kapitole bude popsáno, co to vlastně webová aplikace je, základní principy tvorby webových aplikací a používané technologie. Technologie použité na informační systém pro házenkářský klub budou dále popsány v kapitole Kapitola 3.

2.1 Webová aplikace

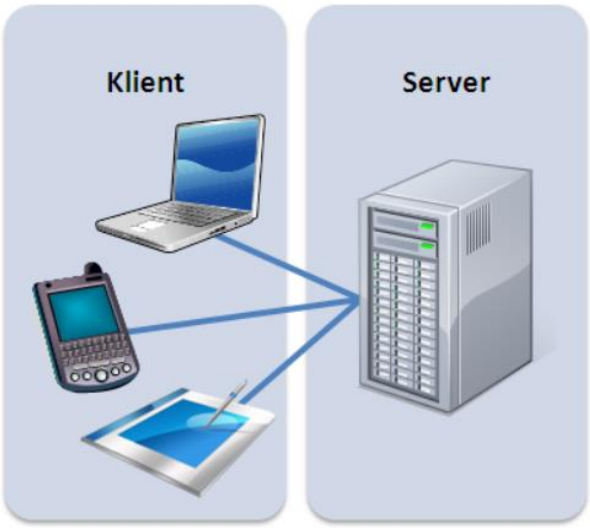
Webová aplikace je uložená na serverech a uživatelům je dostupná skrze internetovou síť. Dříve byla pro aplikace využívána architektura typu klient-server, kde uživatel používal pro přístup do aplikace klientský program, který sloužil jako uživatelské rozhraní aplikace a byl nainstalován na počítači každého zaměstnance/uživatele. Avšak dnes již většina webových aplikací využívá třívrstvou architekturu. [23]

Popularita webových aplikací je daná především všudypřítomností webového prohlížeče, který je využíván jako klient. Tento klient se nazývá tenkým klientem, protože sám o sobě nezná logiku aplikace. Webová aplikace může na první pohled vypadat jako webová stránka, ale obvykle se jedná o složitější aplikaci provádějící složitější úlohy a využívající databázi. [22] Webová aplikace je kolekcí statických a dynamických webových stránek. *Statická webová stránka* se nemění, když o ni návštěvník webového místa požádá: webový server odešle stránku webovému prohlížeči, který o ni požádal, beze změny. Naproti tomu *dynamickou webovou stránku* server modifikuje před odesláním prohlížeči, který o ni požádal. Proměnlivá povaha stránky je důvodem, proč se jí říká dynamická. [9]

2.1.1 Architektura klient-server

Klient-server je dvouvrstvá architektura a jeden z typů architektury, která je využívána informačními systémy.

Model klient-server je forma distribuovaného zpracování výpočetního výkonu mezi koncovým zařízením (klientem) a serverem, kteří mezi sebou komunikují a předávají si vzájemně data. Klient transformuje uživatelův požadavek do podoby (zprávy), které server rozumí a čeká na odpověď od serveru, kterou opět transformuje do podoby, které tentokrát rozumí uživatel a prezentuje jí na obrazovku počítače. Server v této architektuře zpracovává dotazy v databázi a klient je prezentuje, zajišťuje aplikační logiku a zprostředkovává rozhraní pro uživatele. [2]

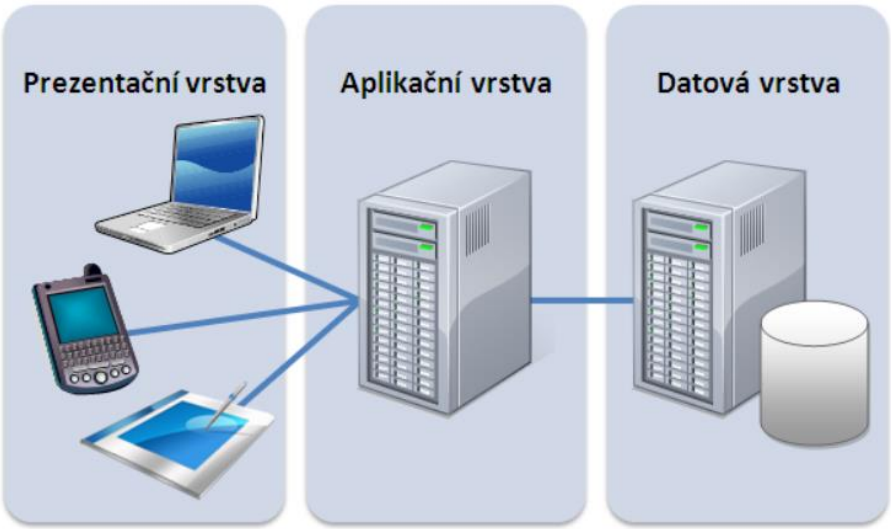


Obrázek 2.1: Architektura klient – server
Zdroj: [2]

2.1.2 Třívrstvá architektura

Třívrstvá architektura je nejčastěji používanou architekturou pro webové aplikace. V té nejběžnější formě je webový prohlížeč první vrstvou (prezentační), nástroje pro dynamické generování stránek (např. PHP) je vrstvou střední (logickou) a databáze je vrstvou třetí (datovou). Webový prohlížeč posílá požadavky střední vrstvě, která je obsluhuje prostřednictvím dotazů do databáze (resp. její aktualizací) a generováním uživatelského rozhraní. [12]

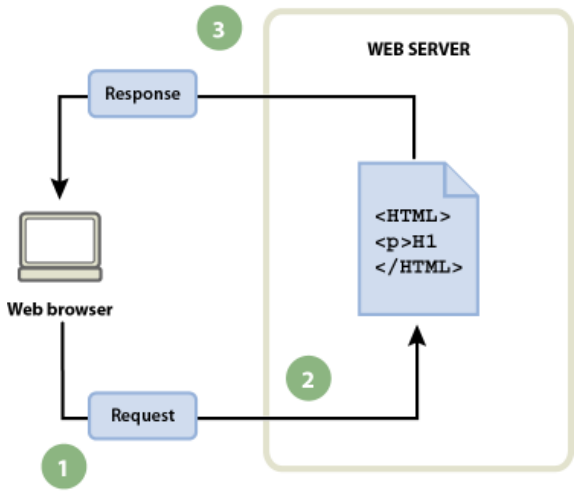
Velké množství webových aplikací je napsáno v čistém programovacím jazyce, jako je například PHP. Dnes ovšem existuje velké množství systémů tzv. frameworků, které jsou nadstavbou čistých programovacích jazyků a umožňují vývojáři popsat program na vyšší úrovni abstrakce a s nižší chybovostí z důvodu jednoduchosti a přehlednosti kódu. Vývojář se tak může více soustředit na vlastní zadání. Některé frameworky budou popsány v kapitole 2.2.4.



Obrázek 2.2: Třívrstvá architektura
Zdroj: [18]

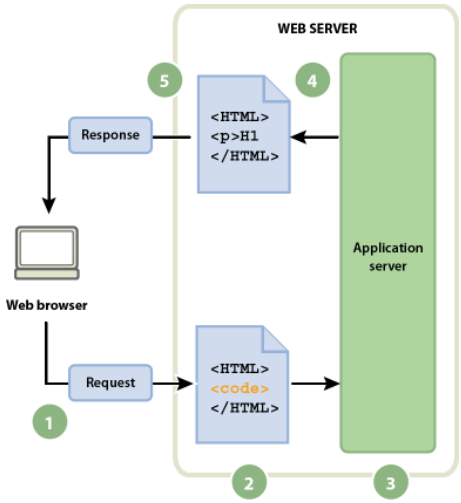
2.1.3 Statické a dynamické webové stránky

Statické webové místo se skládá z množiny souvisejících stránek HTML a souborů hostovaných v počítači, na kterém běží webový server. [9] Webový server je počítač, ale i též software, který je odpovědný za vyřizování HTTP požadavků od klientů (nejčastěji webových prohlížečů). Požadavek je uživatelem vytvořen, když uživatel zadá URL nebo v případě, že klikne na nějaký odkaz. Vyřízením požadavků se rozumí odeslání cíle specifikovaného URL. Statické webové stránky jsou neměnné a uživateli je webový server zobrazí přesně tak, jak je návrhář určil a jak jsou na webovém serveru uloženy. Tento princip je znázorněn na obrázku Obrázek 2.*3* níže.



Obrázek 2.3: Zpracování statické webové stránky
Zdroj: [9]

Namísto toho u dynamické webové stránky se proces zobrazování stránky webovým serverem liší. Jak již bylo řečeno, webový server odesílá uživateli beze změny, kdežto při požádání o dynamickou webovou stránku uživatelem webový server nejprve požadavek předá aplikačnímu serveru, který podle kódových instrukcí stránku modifikuje. Tato stránka se po modifikaci stala stránkou statickou a aplikační server ji předá webovému serveru a až poté ji odesílá uživateli. Tento princip zpracování je znázorněn na obrázku Obrázek 2.*4* níže.



Obrázek 2.4: Zpracování dynamické webové stránky
Zdroj: [9]

2.2 Běžně využívané technologie

V této kapitole bude popsáno, jaké technologie a programovací jazyky se využívají při tvorbě webových aplikací. Programovacích jazyků je celá řada. V textu jsou uvedeny pouze nejčastěji používané programovací jazyky a v některých případech i jejich frameworky.

2.2.1 HTML

HTML (Hypertext Markup Language) je značkovací jazyk používaný pro tvorbu webových stránek. Vývoj HTML byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka. [8] Tento jazyk popisuje webovou stránku pomocí množiny značek, tzv. tagů a jejich vlastností, tzv. atributů. Název tagu a jeho vlastnosti jsou obaleny znaky většítka a menšítka. Tyto tagy jsou většinou párové, kde oba jsou skoro stejné, pouze koncový tag má před názvem ještě navíc znak lomítka.

2.2.2 CSS

Kaskádové styly (Cascading style sheets) je jazyk, který popisuje způsob zobrazení elementů na stránkách napsaných v jazycích HTML, XHTML nebo XML. Zpřehledňuje HTML kód, protože nemusíme styl psát „inline“ tj. přímo do atributu HTML značky. Pokud chceme nějaké značce nastavit styl, můžeme k tomu využít buď přímo název značky nebo její třídu či id. Když nějaký element nacházející se v dokumentu nemá definovaný styl, dědí ho většinou od svých rodičovských elementů. Pokud tedy chceme definovat, aby stránka využívala jeden font o jedné velikosti a jedné barvě, nadefinujeme styl nějakému hlavnímu elementu, nejčastěji je to `<body>`.
CSS může být do HTML přidáno třemi způsoby:

- použitím atributu style v HTML elementu,
- definicí v hlavičce dokumentu,
- nebo v externím CSS souboru (nejpoužívanější)

CSS je zpravidla definován jako seznam pravidel. Každé pravidlo je složeno ze selektoru a bloku deklarací. Selektor určuje, na který element dokumentu se aplikuje vlastnost nastavená v deklaraci. Blok deklarací je tvořen seznamem dvojic identifikátor vlastnosti – hodnota vlastnosti. [11]

2.2.3 MySQL

MySQL (My Structured Query Language) je systém řízení báze dat uplatňující relační databázový model. Jedná se o multiplatformní databázi a komunikace s ní probíhá prostřednictvím jazyka SQL. Tento jazyk umožňuje jednoduché získání dat z databáze. [25] Je to relační databáze, která běží na straně serveru. Využívá více vláken současně, což vede ke zvýšení rychlosti databáze [17]. Každému připojení je přiřazeno nové vlákno. MySQL umožňuje také správu uživatelů. Autentizace uživatelů je prováděna buď pomocí uživatelského jména, hesla a hostitele nebo pomocí certifikátu přes SSL připojení.

Data jsou uložena na disku na serveru. Každá databáze má svou složku, ve které každý soubor reprezentuje jednu tabulku. Z důvodu uložení souborů na serveru je dobré dbát na pojmenování tabulek a databází. Na unixových serverech záleží na velkých a malých písmenech, ale na serverech s operačním systémem Windows na velikosti nezáleží.

Pro správu dat databáze a databáze můžeme využít mnoho nástrojů. Příklady nástrojů, uložených na serveru jsou phpMyAdmin nebo Adminer.

2.2.4 Programovací jazyky

Webové aplikace se dají vyvíjet v několika různých programovacích jazycích a jejich kombinaci. Mezi nejoblíbenější kombinaci patří PHP a JavaScript, ale existují i jiné hojně používané programovací jazyky, které jsou uvedeny níže a u některých i jejich frameworky.

2.2.4.1 Python

Jazyk Python je interpretovaný, objektově orientovaný, vysokoúrovňový jazyk s dynamickou sémantikou. [26] Kromě objektové orientace podporuje i jiná programovací paradigma včetně imperativního, procedurálního nebo funkcionálního. Velkou výhodou Pythonu je, open-source vývoj a nezávislost na platformě, z čehož plyne velká základna a díky tomu má je k dispozici tisíce knihoven třetích stran, které skvěle doplňují velkou standardní knihovnu. [16]
Pro vývoj webových aplikací jsou nejpoužívanější tyto frameworky: [13]

- Django
- Flask
- Tornado

2.2.4.2 JavaScript

JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk. Je většinou vkládaný přímo do HTML souborů, a tudíž interpretaci provádí přímo webový prohlížeč oproti jiným interpretovaným jazykům, mezi které patří např. PHP, ty jsou interpretovány přímo na serveru. Z tohoto plynou bezpečnostní omezení. Nicméně JavaScript lze používat i na serveru pomocí platformy Node.js. [29]

2.2.4.3 PHP

PHP je skriptovací, netypovaný, objektově orientovaný programovací jazyk, který se především používá k vytváření webových aplikací a dynamických internetových stránek. Veškeré výpočty jsou prováděny na straně serveru a v současné době je to nejpoužívanější skriptovací jazyk na straně serveru. Umožňuje programovat jak procedurálně, tak i objektově a je multiplatformní. PHP je poměrně starý programovací jazyk, který byl vytvořen v roce 1995 [24]. To, že se jedná o poměrně starý programovací jazyk, přináší řadu výhod nicméně i nevýhod. Mezi výhody patří:

- rozsáhlá a přehledná dokumentace
- vysoká podpora u webhostingů
- množství funkcí ve standardní knihovně

Jako nevýhody se uvádí [30]:

- nekonzistentní logika pořadí parametrů funkcí
- nekonzistentní pojmenování funkcí
- `array()` není objekt
- malé množství standardizovaných funkcí pro zpracování chyb [21]

Mezi nejpoužívanější PHP frameworky patří: [1]

- Laravel
- Symfony
- CodeIgniter
- CakePHP

Pro implementaci informačního systému v této práci bylo použito PHP 7.2 spolu s frameworkem Laravel verze 6, o kterém se autor zmiňuje v další části této práce.

Kapitola 3

Použité technologie

Kapitola popisuje technologie použité pro vývoj informačního systému pro házenkářský klub. Je zde představen PHP framework Laravel, který byl vybrán jako nejvhodnější, ať už z důvodu obliby a velké komunity, ale i velké přehlednosti. Dále jsou popsány další technologie a nástroje, které byli využity při vývoji.

3.1 Laravel

Laravel je nejoblíbenější a nejpoužívanější PHP framework na světě, vytvořený Taylorem Otwellem. [20] Oblíbený je zejména pro jednoduchou spolupráci s různými knihovnami či různé způsoby pro přístup k relačním databázím. Pro vývoj informačního systému házenkářského klubu byla využita verze 6, která byla vydaná 3. září 2019. Webové aplikace implementované pomocí Laravelu jsou založeny na Model – View – Controller (MVC) architektuře. Laravel se snaží o udržování jednoduchosti, přehlednosti a čitelnosti kódu a také proto byl pro tuto práci vybrán. Pro práci s databází využívá Laravel prostředky jako Query Builder, či vlastní ORM Eloquent nebo migrace společně se seedy, které napomáhají k udržování databáze v konzistenci bez nutnosti složitých mechanismů. [4] Pro tvorbu frontendu je využíván šablonovací systém Blade, který umožňuje vkládat HTML i PHP kód do jednoho souboru. Tento soubor je následně zkompilován do PHP kódu a je kešován do doby, než je změněn. [19] Pro vzhled stránky je využíván SASS. SASS rozšiřuje CSS syntaxi o proměnné, podmínky, cykly, mixiny, funkce aj. SASS je potřeba vždy zkompilovat do CSS souboru, protože samotný SCSS (Sassy CSS) soubor nelze spustit, protože prohlížeč nezná jeho syntaxi.

Mezi základní výhody Laravelu a důvody, proč si autor tento PHP framework pro vývoj informačního systému vybral, patří [32]:

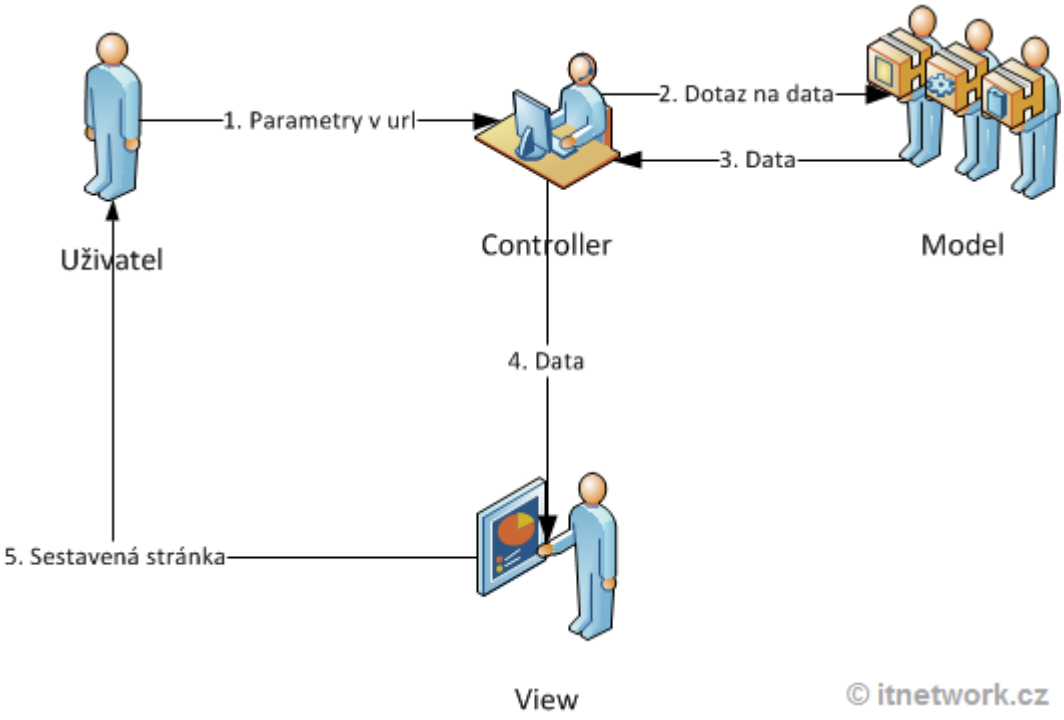
- podpora MVC architektury a OO přístup,
- obsahuje autorizační a autentifikační balíček,
- podpora nástroje composer,
- podpora různých uložišť a souborových systémů,
- podpora nástroje Artisan,
- Eloquent ORM.

Tyto výhody ovlivnily práci autora natolik, že se rozhodl v budoucnu věnovat se mu více a využít jej pro další projekty.

3.1.1 Architektura Model – View – Controller

Model – View – Controller se často označuje za návrhový vzor, nicméně se jedná o softwarovou architekturu nejčastěji používanou při vývoji webových aplikací. Rozděluje datový model aplikace, uživatelské rozhraní a řídící logiku do tří nezávislých komponent tak, že modifikace některé komponenty má pouze minimální vliv na ostatní komponenty. Těmi komponentami jsou:

- Model – definuje data aplikace a jejich chování, ale není zodpovědný za prezentování nebo formátování těchto dat.
- View – zobrazuje data, která jsme dostali od modelu, uživateli. Nejčastěji se jedná o phtml šablonu, obsahující HTML stránku a tagy nějakého značkovacího jazyku, který umožňuje provádět iterace a podmínky nebo vkládat proměnné.
- Controller – tzv. „prostředník“, který komunikuje s uživatelem, modelem, také view a propojuje všechny komponenty.



Obrázek 3.1: Model-View-Controller
Zdroj: [5]

Na obrázku 3.1 výše je zobrazen životní cyklus stránky s principem MVC architektury. *View* komponenta zobrazuje data uživateli, mezitím akce uživatele zpracovává komponenta *controller*. *Controller* následně posílá požadavky na data (uložení, smazání, úprava nebo zobrazení). Komponenta *model* poté požadavky zpracuje a data vrátí.

3.1.2 Práce s databází

Pro práci s databází využívá Laravel nástroj Query Builder, který poskytuje velmi jednoduché a pochopitelné rozhraní. Z hlediska bezpečnosti Query Builder využívá vazbu parametrů PDO (PHP Data Objects), která chrání aplikaci před SQL injection útoky. [14]

Pro provádění běžných databázových operací jako je zápis, čtení úprava a mazání záznamů, které označujeme jako CRUD operace (podle anglických ekvivalentů create, read, update a delete), můžeme využít ORM (Object-relation mapping) Laravelu s názvem Eloquent. Eloquent ORM používá implementaci Active Record. Jedná se o způsob přístupu k datům v databázi způsobem, kdy každá třída modelu představuje jednu tabulku v databázi a každá instance této třídy představuje jeden záznam (řádek) z tabulky v databázi. [7] Velkou výhodou Active Record je jednoduchost používání vztahů mezi modely. Volání metody vztahu vrací instanci třídy Query Builder a proto můžeme zřetězit další metody Query Builderu (např. `where()`) pro omezení výsledku a samotný SQL dotaz vykonáme na

konci zavoláním metody `get()`. Na obrázku 3.2 můžeme vidět volání metody vztahu (Eloquent ORM) a filtrování pomocí `where()` a výsledek je instance třídy Query Builder typu *HasMany*. Dále vidíme stejné volání vztahu s použitím filtru, ale zakončen voláním metody `get()`, která vrací kolekci instancí třídy modelu.

```
dd($user->posts()->where('subject', 'like', '%hello%')->latest());


HasMany {#209 ▼
  #foreignKey: "posts.user_id"
  #localKey: "id"
  #query: Builder {#206 ▶}
  #parent: User {#212 ▶}
  #related: Post {#201 ▶}
}


dd($user->posts()->where('subject', 'like', '%hello%')->latest()->get());


Collection {#213 ▼
  #items: array:3 [▼
    0 => Post {#214 ▶}
    1 => Post {#218 ▶}
    2 => Post {#219 ▶}
  ]
}
```

Obrázek 3.2: Příklad použití Eloquent ORM
Zdroj: Vlastní zpracování

3.1.3 Homestead

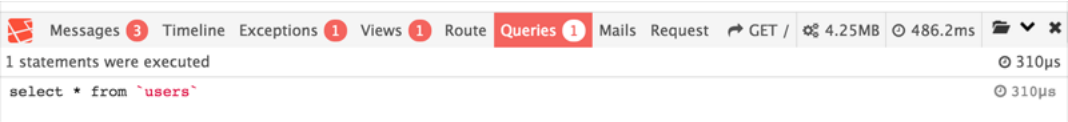
Pro rychlý a jednoduchý začátek ve vytváření webových aplikací ve frameworku je možno využít vlastní vývojové prostředí s názvem Homested, které je součástí Laravelu. Jedná se o oficiální multiplatformní předpřipravený Vagrant balíček, který poskytuje vývojové prostředí bez nutnosti instalování PHP, webového serveru a dalších jiných serverových softwarů na váš lokální počítač. Nemusíte si tak na svůj operační systém instalovat všechny potřebné technologie pro vývoj, jen abyste si rozběhli svůj projekt. Vagrant je zcela nezávislý a pokud někde vznikne chyba, můžete balíček smazat a vytvořit znovu téměř během minuty. [10]

3.1.4 Debugbar

Laravel Debugbar je balíček, který vývojáři umožní během vývoje velmi snadno a rychle vidět:

- zprávy, které si vývojář nechá vypsat;
- časovou osu, na které vývojář může vidět úzká místa;
- odchycené výjimky;
- jaké byly použity šablony;
- jaké dotazy do databáze proběhly a jejich časová délka; a
- odeslané emaily z aplikace.

Díky těmto funkcím je to skvělý a velmi užitečný nástroj pro vývoj a pro detekci míst, která lze v aplikaci zrychlit a vylepšit.



Obrázek 3.3: Debugbar
Zdroj: Vlastní zpracování

3.1.5 PHP Artisan

Artisan je rozhraní příkazové řádky, které je vloženo společně s Laravelem. Poskytuje řadu zajímavých a užitečných příkazů, které programátorovi velmi ulehčí práci. Jako například příkaz `php artisan make`, který dokáže vytvořit model, controller, migraci a další. Všechny dostupné příkazy si lze zobrazit pomocí příkazu `php artisan list`. V artisanu si můžeme vytvořit i vlastní příkazy nebo budou přidány při instalaci některých balíčků

3.2 Bootstrap

Bootstrap je open source CSS framework. Tento framework je jednoduchý a výhodný zejména pro svoji dokumentaci, která velmi pomáhá vývojářům. O jeho jednoduchosti vypovídá fakt, že ho používají i programátoři, kteří frontendu a obecně kódování příliš neholdují, a i tak vypadá výsledné designové zpracování profesionálně. Bootstrap je obsažen už v základu frameworku Laravel. Jeho velká komunita v důsledku značí i velké množství šablon, které jsou volně k použití. Framework Bootstrap je plně responzivní a využívá ‚Mobile-first' architekturu. To je velmi výhodné, pokud předpokládáte, že vaše aplikace bude využívaná na zařízeních, které mají různou velikost obrazovky. Bootstrap nevyužívá pouze CSS, ale ve frameworku je využíván i JavaScript, díky kterému je možné využívat výsuvná menu, nebo dynamicky zobrazované prvky formuláře. [3]

3.3 Composer

Composer je nástroj určený pro správu knihoven a jejich závislostí v PHP. Díky Composeru je velmi jednoduché starat se o složité návaznosti knihoven v PHP. Stačí si v souboru s parametry **`composer.json`** zadat knihovny, které chceme v aplikaci používat a o navazující závislosti těchto knihoven se postará sám Composer. Ovládá se pomocí terminálu (či příkazové řádky), kde stačí napsat příkaz `composer install` a Composer se postará o nainstalování nebo aktualizaci všech knihoven a balíčků, které jsou uvedeny v souboru s parametry. Je možné si, ale nainstalovat i samostatné balíčky, a to pomocí příkazu, který se většinou nachází v dokumentaci každého balíčku v části instalace. Po instalaci se tento balíček automaticky přidá do souboru s parametry pro Composer. Přes Composer lze spravovat stovky projektů najednou, lze také distribuovat jednou napsaný kód do všech aplikací současně.

Kapitola 4

Analýza požadavků

Následující kapitola se bude věnovat analýze a specifikaci požadavků, která je při vývoji informačního systému považována za klíčovou a nejdůležitější část. Můžeme totiž udělat skvělý systém s krásným designem, ale pokud nebude splňovat požadavky, které od systému požadoval a očekával zadavatel, tak je tento systém pro zadavatele bezcenný, protože nesplňuje, co zadavatel požadoval, aby systém uměl.

Pro informační systém házenkářského klubu byly požadavky specifikovány trenéry, hráči a hlavně vedoucími klubu, kteří budou do systému přistupovat nejčastěji, formou dotazníkového šetření. Bylo potřeba od nich zjistit, základní funkcionality, které musí aplikace splňovat, aby byla užitečná a usnadnila těmto osobám práci. Většina osob podílejících se na chodu házenkářského klubu za to nepobírá žádnou finanční kompenzaci, je tedy potřeba, aby se alespoň jejich volný čas věnovaný administrativě snížil na minimum. Právě to je hlavním cílem autora této práce, který sám působí jako hráč a rozhodčí házené.
Mezi hlavní požadavky, které by měla aplikace splňovat, patří:

- správa sezón, zápasů
- zápis a vyhodnocení zápasových statistik
- organizace tréninků, zápasů a akcí klubu zahrnující přihlašování na tyto akce
- evidence dokladů vydaných na jízdné, rozhodčí a organizaci akcí

Je také potřeba zmínit, že do systému budou přistupovat uživatelé v následujících rolích:

- hráč
- vedoucí klubu

Tyto role jsou důležité pro rozlišení oprávnění k akcím, které aplikace pro uživatele nabízí a pro omezení zobrazovaného obsahu.

4.1 Správa uživatelů

Pro každý informační systém je klíčová správa uživatelů. Je to první krok k zabezpečí systému. Správa uživatelů zahrnuje vytvoření a úpravu uživatelských účtů, změna hesla uživatelům a aktivace/deaktivace uživatelů. Do této sekce informačního systému bude mít přístup pouze role vedoucí klubu.

Protože se jedná o interní systém, bude pro přístup do informačního systému nutné být přihlášen. Přihlásit se budou moci pouze uživatele, které vedoucí klubu aktivuje. Uživatel s rolí hráče může upravovat své osobní údaje, jako např. email nebo bydliště.

4.2 Správa týmů a zápasů

Nejdůležitější částí systému je celková správa klubu a jedna z částí správy klubu je správa týmů a jejich zápasů. Tato správa obsahuje evidenci výsledků v zápasech klubu pro daný soutěžní ročník a soutěž.

Bude nutné také přiřadit hráče k týmu. Je také velmi důležité hlídat dobu platnosti registrace a datum platnosti lékařských potvrzení hráčů. Oprávnění k vytvoření soutěžního ročníku, týmu, zápasu a soutěže a přiřazení hráčů k týmu a zápasu bude mít pouze role vedoucí týmu. Role hráče bude mít pouze možnost si zobrazit, jací hráči se účastnili daného zápasu a k výsledku zápasu.

4.3 Zápis a vyhodnocení zápasových statistik

Zápasové statistiky jsou další důležitý článek správy klubu. V polovině a na konci sezóny jsou vyhodnocovány statistiky hráčů. Před vytvořením tohoto informačního systému statistiky v průběhu zápasu zapisoval na papír druhý brankář, pokud byl k dispozici. Pokud druhý brankář k dispozici nebyl, tak statistiky zapisoval některý z hráčů na střídačce. Následně někdo musel tyto statistiky vyhodnotit. Tento postup zabíral hodně času a už se to nechtělo nikomu dělat.

V zápase jsou statistiky sbírány společné pro všechny hráče, brankáře a hráče v poli. Mezi statistiky, které jsou pro všechny hráče stejné, patří:

- plus bod (získání míče, trest pro hráče, který fauloval daného hráče, zablokování střely)
- mínus bod (ztráta míče, technická chyba, faul, po kterém je zahráván 7m hod)
- tresty (žlutá karta, dvouminutový trest, červená karta a modrá karta)
- asistence (poslední hráč, který přihrál skórujícímu hráči)

Statistiky, které jsou sledovány u brankářů jsou

- chycená střela z 6metrového hodu
- chycená střela ze 7metrového hodu
- chycená střela z 9metrového hod
- obdržený gól z 6metrového hodu
- obdržený gól ze 7metrového hodu
- obdržený gól z 9metrového hodu

Statistiky, které jsou sledovány pouze u hráčů v poli jsou:

- úspěšná střela z 6metrového hodu
- úspěšná střela ze 7metrového hodu
- úspěšná střela z 9metrového hodu
- neúspěšná střela z 6metrového hodu
- neúspěšná střela ze 7metrového hodu
- neúspěšná střela z 9metrového hodu

4.4 Organizace tréninků, zápasů a klubových akcí

Pro tak malý klub jako je TJ Dolní Cerekev, je důležité přehledné zobrazení účasti a neúčasti na trénincích, z důvodu správného rozvržení tréninkových dávek, které jsou hlavně důležité zejména před

zahájením sezony. Klíčovou je také evidence a účasti a neúčasti na zápasech. Hráči nejsou profesionálové, a tak se často stane, že jim jejich pracovní povinnosti zamezí účastnit se tréninku či zápasu. V případě zápasu je pak dopředné zajištění dostatečného počtu hráčů klíčové, protože možnost střídání významně ovlivní výsledek zápasu. Historicky se ukázalo, že komunikace ve věci účasti na zápasu není vždy přehledná. Z toho důvodu má organizace tréninků a zápasů ve webové aplikaci významné postavení.

Důležitá také je organizace klubových akcí, které jsou pro chod klubu významné, z důvodu zajištění finančních prostředků, bez kterých by nebylo možné zajistit chod klubu a jeho budoucnost. Organizace klubových akcích je zajišťovaná z řad hráčů a členů oddílu a jejich rodinných příslušníků, a proto je nutné vědět, kolik těchto lidí se na dané akci může podílet na organizaci. Jedná se například o klubem řízené sběry železného odpadu, klubem organizované taneční zábavy a turnaje. U posledních dvou zmíněných jsou finanční prostředky získávány především z prodeje nápojů a občerstvení.

Do sekce trénink a klubové akce budou mít přístup všechny role. Role vedoucí týmu bude moci vytvářet nové tréninky a klubové akce, upravovat již exitující tréninky a klubové akce a zvát na ně hráče a organizátory. Roli hráče se zobrazí pouze události, na které byl pozván, jejich detail a popis jako je například datum konání události, její název a kdo se dané události zúčastní.

4.5 Evidence dokladů vydaných na jízdné, rozhodčí a organizaci akcí

Pro zlepšení přehledu výdajů na sezonu bude informační systém obsahovat i sekci s evidencí výdajových dokladů. Tato funkce je potřeba pro zlepšení hospodaření klubu. Vedoucí klubu bude moci vystavit výdajový doklad na různé účely, jako například na platbu rozhodčímu za zápas nebo za refundaci nákladů hráče, který poskytl své auto na dopravu týmu k zápasu. Bude možnost vystavit doklady na zajištění organizace na událostech pořádaných klubem.

Hráč bude mít v této sekci pouze oprávnění na zobrazení výdajových dokladů, na kterých je uveden jako adresát.

4.6 Diagram případů užití

Výstupem analýzy a specifikace požadavků je nejen slovní soupis požadavků a jejich specifikace od zadavatelů a uživatelů aplikace, ale také Diagram případů užití, který tyto požadavky zobrazuje přehledně jako jeden nebo více diagramů. Při návrhu a tvorbě systému je mnohem jednoduší převést požadavky zadavatelů do diagramu než celý systém vyvíjet pomocí psaného popisu. Po zjištění a pochopení požadavku na systém je tedy vytvořeno několik diagramů, které obsahují jistou míru abstrakce pro snadnější pochopení problému a jednodušší vývoj a samotná implementace daných požadavků na funkčnost je skryta.

Use case diagram se skládá z případů užití (use case), účastníků (actors) a vztahů mezi nimi [6] V následujících odstavcích budou stručně popsány jednotlivé části Use Case diagramu, kde účastníky budou vedoucí klubu a hráč. Nabízel by se ještě účastník trenér, nicméně v prostředí takto malého klubu je trenér i vedoucím klubu, takže je účastník vedoucí klubu spojením trenéra a vedoucího klubu.

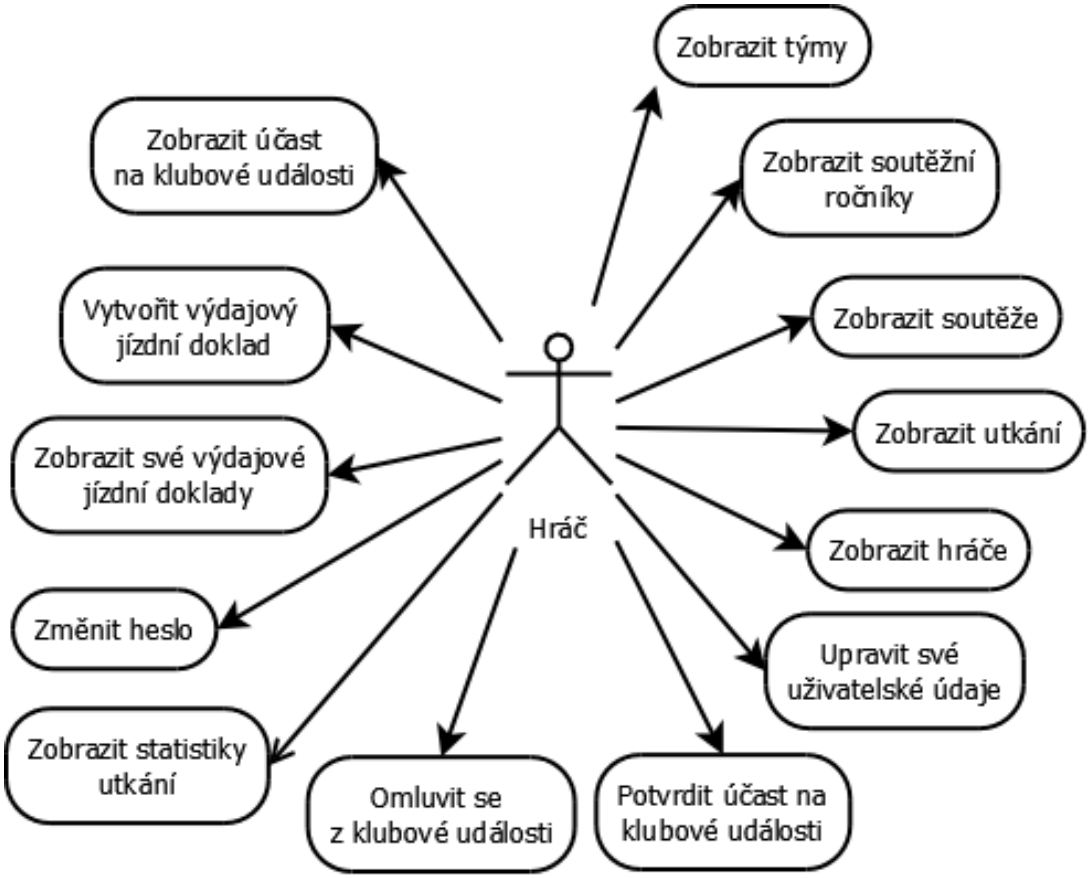
4.6.1 Role hráč

Role hráče nebude mít oprávnění vykonávat příliš mnoho akcí. Půjde převážně o zobrazování povolených dat jako je zobrazení utkání, zobrazení týmů, zobrazení statistik utkání nebo si bude moci zobrazit, kdo se zúčastní klubové akce. Na druhou stranu bude role hráč moci vytvářet a modifikovat

data jako například změnit si své uživatelské údaje, mezi které patří email, telefon nebo bydliště, změnit si heslo pro přístup do systému. Bude mu povoleno potvrzování a omlouvání se na klubových akcích a trénincích. Tato na první pohled omezení role hráč, vychází pouze z požadavků uživatelů na webovou aplikaci, se naopak ukazují jako zcela praktické.

Hráč bude také moci vytvořit nové výdajové jízdní doklady, když hráč použil k cestě na zápas svůj automobil. Tato varianta se v praxi stává velice často, protože v rámci úspor cestovních nákladů zajistí hráči přepravu na zápas pro sebe a spoluhráče vlastními automobily. Při plně obsazeném automobilu je pak tato varianta výhodnější než cesta hromadnou dopravou.

Veškeré činnosti, které bude moci dělat uživatel s rolí hráče, jsou znázorněny v obrázku 4.1



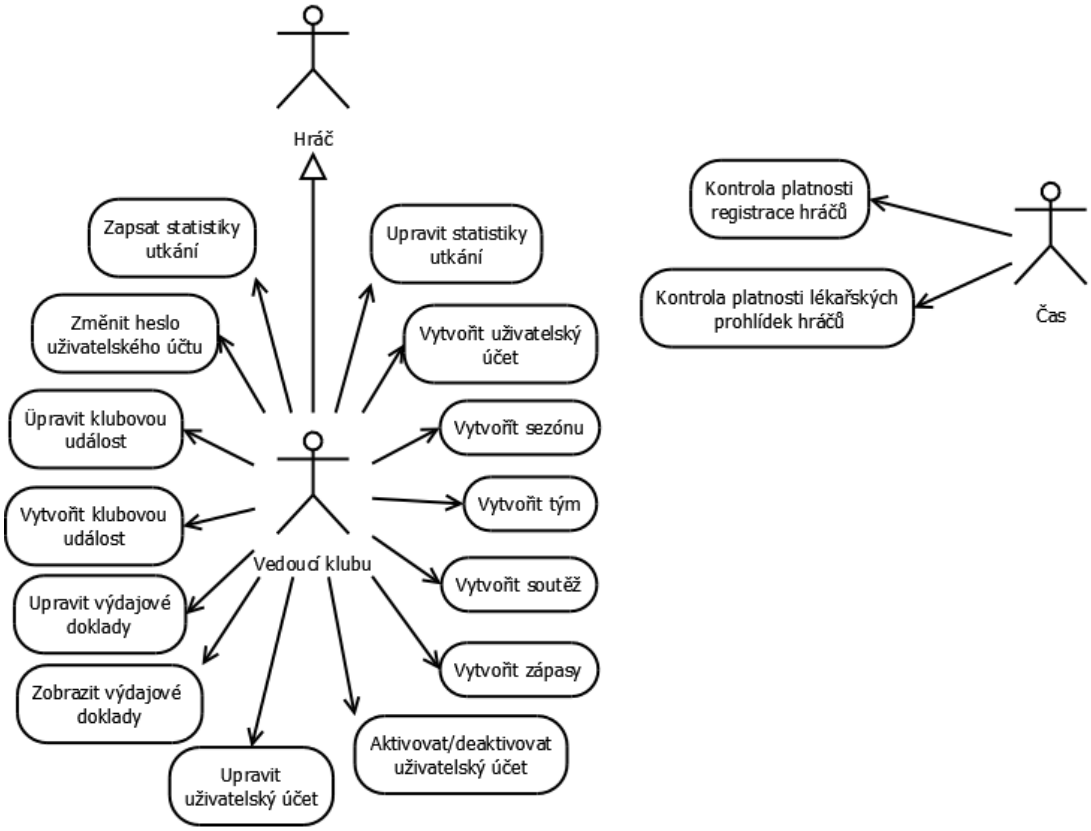
Obrázek 4.1: Diagram případů užití pro roli hráč
Zdroj: Vlastní zpracování

4.6.2 Role vedoucí klubu

Jak již název role napovídá, tak vedoucí klubu bude mít nejvíce oprávnění k akcím v informačním systému. Můžeme říci, že vedoucí klubu je v podstatě administrátor systému. Vedoucí klubu bude mít povoleno vše co uživatel s rolí hráče a ještě další akce, které jsou znázorněny na obrázku 4.2. Vedoucím klubu pak v praxi budou trenéři a funkcionáři.

Mezi povolené akce, které nejsou povoleny roli hráče, ale roli vedoucího klubu patří například celková správa uživatelských účtů jako přidání a aktivace nových uživatelů nebo deaktivace již vytvořených uživatelských účtů. Dále bude mít možnost vytvářet nové sezony a soutěže a přiřazovat do nich týmy. Bude mít také povoleno zapisovat nové statistiky k jednotlivým utkáním případně opravovat chyby v již vytvořených statistikách. Další úlohou, o kterou se bude starat je evidence výdajových dokladů a vytváření klubových událostí a tréninků na které bude zvát hráče.

U role vedoucího vystupuje také další aktér a to čas. Čas v systému kontroluje, zda některému aktivnímu uživateli nevyprší platnost registrace nebo platnost lékařské prohlídky. Na případné vypršení platnosti některého z dokladu bude uživatel s rolí vedoucího klubu upozorněn při přihlášení do systému, aby se mohl postarat o prodloužení platnosti.



Obrázek 4.2: Diagram případů užití pro roli vedoucího klubu
Zdroj: Vlastní zpracování

Kapitola 5

Návrh systému

Kapitola 5 se zabývá návrhem systému. Tak jak je důležitá správná analýza a specifikace požadavků, tak je i neméně důležité podle těchto specifikovaných požadavků vytvořit kvalitní návrh systému. V případě kvalitního návrhu systému je pak daleko snazší následná implementace a není zapotřebí tolika oprav, jako u zanedbaného a nepromyšleného návrhu.

V této kapitole bude popsán návrh architektury, která vychází z MVC architektury. Bude zde také popsán návrh modelu dat, který je vizualizován jako ER diagram. Na závěr kapitoly bude rozebrán návrh uživatelského rozhraní

5.1 Architektura systému MVC

Jak již bylo zmíněno v kapitole Kapitola 3 Laravel pracuje na architektuře Model-View-Controller. Tato architektura byla základem pro informační systém. Pokud je vytvořen kvalitní návrh architektury, je poté velmi jednoduché jakékoli, ať už malé či velké, rozšíření implementovat. Při dobrém architektonickém návrhu je totiž zřejmé, kam do zdrojového kódu musí programátor rozšíření doprogramovat. Blokové schéma navržené architektury je zobrazeno na obrázku 3.1.

5.1.1 Model

Modelem v implementovaném informačním systému reprezentuje datovou vrstvu. Pro každou tabulku uloženou v databázi je v aplikaci vytvořena třída, která rozšiřuje třídu **Model**. Tato vytvořená třída je rozhraním pro komunikaci s databází a každý atribut třídy reprezentuje sloupec v dané tabulce. V této třídě jsou také uloženy metody, které zajišťují vazbu mezi tabulkami se společnými vlastnostmi.

5.1.2 Controller

Jedná se o logickou vrstvu, která žádá model o data a následně je uživateli zobrazí pomocí view vrstvy uživateli. V systému je controller implementován jako třída rozšiřující třídu **Controller** a pro každý model je vytvořen controller, který zajišťuje operace nad modelem. Nově vytvořená třída obsahuje metody pro základní CRUD operace, nicméně ve vytvořeném systému metody pro mazání dat většinou není podporovaná. Dále tam jsou obsaženy i další metody. Na všechny tyto metody odkazujeme pomocí cest nadefinovaných v souboru **web.php**.

5.1.3 View

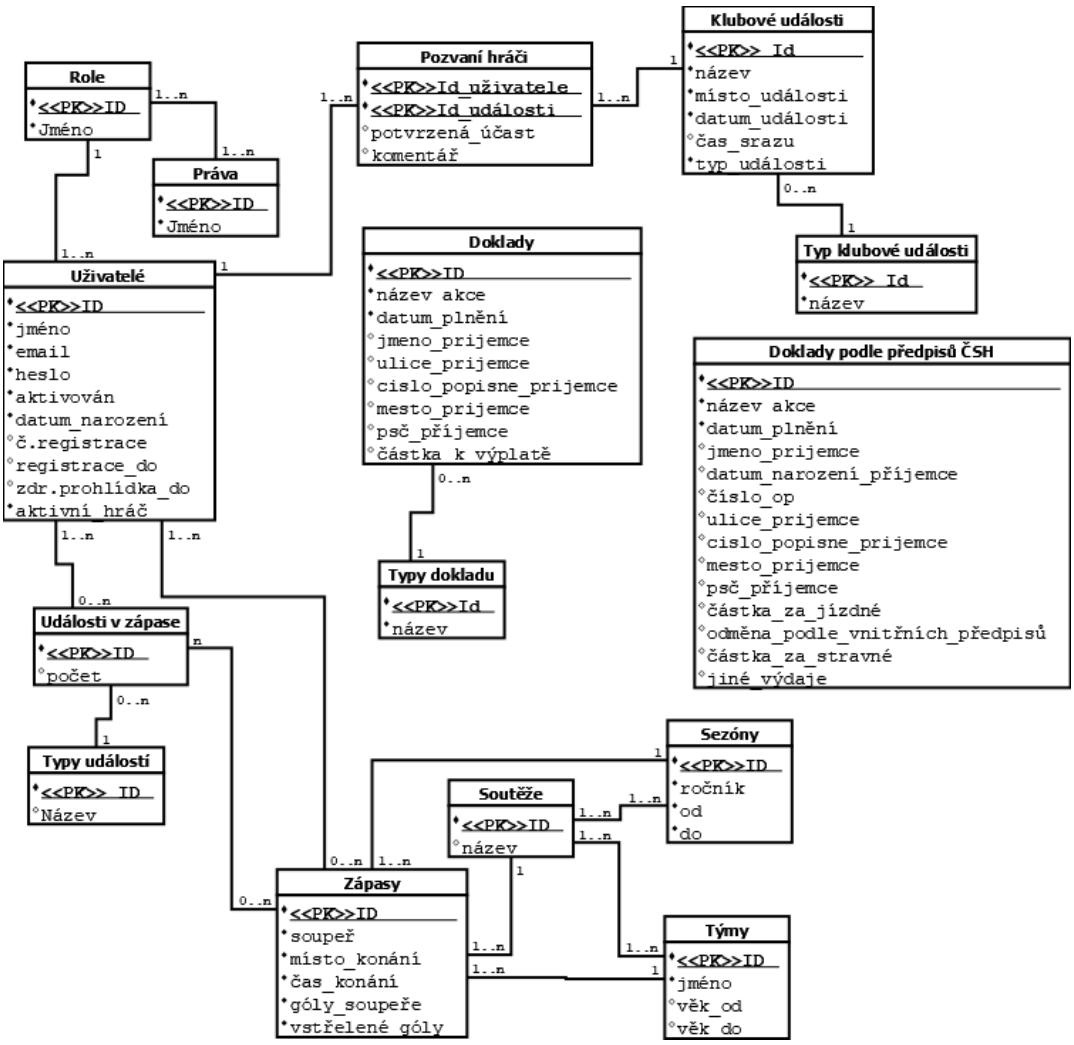
Views neboli pohledy reprezentují prezentační vrstvu, která zobrazuje požadovaná data uživateli. Pohledy obsahují HTML a PHP a jsou to šablony typu Blade. Veškeré pohledy jsou uloženy v adresáři *resouces/views*. Hlavní pohledy, které jsou načítány vždy, jsou uloženy ve složce *layouts* a obsahují základní zobrazení stránky. Pohledy zobrazující data v těle stránky mají zvláštní adresáře, které jsou pojmenovány stejně jako modely, jejichž data tyto pohledy zobrazují.

5.2 ER diagram

ER (*Entity-Relationship*) diagram je konceptuální návrh vycházející z datové analýzy požadavků. Je využíván pro zobrazení systémových dat a vztahů mezi nimi. Z ER diagramu vycházíme, při vytváření logického návrhu databáze. [28] Základními prvky, které jsou využívány při tvorbě ER diagramu, jsou:

- **Entita** – reálný objekt, o kterém chceme uchovávat vlastnosti v databázi, který si můžeme představit jako objekt v diagramu tříd;
- **Entitní množina** – množina entit stejného typu, které mají stejné atributy, je možné ji chápat jako třídu v diagramu tříd;
- **Vztahy** – definují asociace mezi dvěma či více entitami;
- **Vztahová množina** – množina vztahů téhož typu a
- **Atributy** – vlastnosti entit.

Na obrázku Obrázek 5.1 je zobrazen ER diagram pro vytvářený informační systém. V podsekcích této kapitoly budou popsány jednotlivé entity a vztahy mezi těmito entitami.



Obrázek 5.1: ER diagram
Zdroj: Vlastní zpracování

5.2.1 Uživatelé – Role – Práva

Entitní množina uživatelé zobrazuje všechny uživatelé, kteří mohou vstoupit do systému. Jak již bylo zmíněno v kapitole Kapitola 4, v systému jsou dvě role (vedoucí klubu a hráč). Ke každé roli je přiřazeno více práv, kde vedoucí klubu, jak z názvu vyplívá, má mnohem více práv než hráč. Každý uživatel může disponovat pouze jednou rolí.

U uživatelů je nutné, kromě role a k ní příslušných práv, také ukládat *jméno a příjmení*, *email*, který slouží jako přihlašovací jméno, takže musí být unikátní. Mezi povinné atributy ještě patří, zda je uživatel *aktivní hráč, heslo* a jestli se jedná o *aktivovaného* uživatele, kde tento atribut je využíván např. při odchodu hráče z klubu, z důvodu zachování konzistence historických dat.

U uživatele mohu využít ještě nepovinné atributy *datum narození, číslo a platnost registrace* a *platnost lékařské prohlídky*

5.2.2 Sezóny – Soutěže – Týmy

Entitní množina soutěže má kromě primárního klíče pouze jeden další atribut a tím je *název* soutěže. U entitní množiny sezóny je potřeba ukládat atributy *ročník, od* a *do*, kde od a do udávají začátek a konec dané sezóny. U týmu ukládáme *název* a *věk od* a *věk do,* který definuje, jak staří hráči mohou nastoupit do zápasů za tento tým.

Entitní množiny sezóny, soutěže a týmy spolu vzájemně souvisí a je třeba je propojit. Jsou v ternárním vztahu s kardinalitami N, protože tým může být ve více sezónách i ve více soutěžích (například když postoupí nebo sestoupí do vyšší, respektive do nižší soutěže, nebo když je svazem přesunut do jiné soutěže, ale na stejné výkonnostní úrovni), nicméně v jedné sezóně se může zúčastnit pouze jedné soutěže.

Tyto entitní množiny s ternárním vztahem propojíme pomocí vazební tabulky, stejně jako vztah binární. Tato vazební tabulka musí obsahovat primární klíč každé z těchto tabulek, které budou ve vazební tabulce cizími klíči [27], kde jejich kombinace musí být unikátní. Primárním klíčem ale nejsou tyto cizí klíče společně, nýbrž z důvodu využívání vazební tabulky v entitní množině zápasy je primárním klíčem inkrementované přirozené číslo.

5.2.3 Zápasy a události v zápase

Hlavním účelem entitní množiny zápasy je evidence jednotlivých zápasů každého týmu házenkářského klubu. Každý zápas je přiřazen do jednotlivé soutěže, sezóny a k danému týmu a je k tomu využíván primární klíč vazební tabulky těchto entitních množin. U zápasu uchováváme *protihráče, počet obdržených a vstřelených branek, místo a termín konání zápasu.*

Do zápasu jsou přes vazební tabulku přiřazováni hráči, kteří v daném zápase nastoupili. U každého hráče pro každý zápas jsou dále evidovány statistiky popsané v kapitole Zápis a vyhodnocení zápasových statistik. Tyto statistiky jsou ukládány v entitní množině události v zápase, která má atributy *id_zápasu, id_uživatele, id_typ_události,* tyto atributy jsou cizími klíči a zároveň jejich kombinace musí být unikátní. V entitní množině události v zápase je pak vlastní atribut pouze počet, který udává, kolikrát se daná událost v zápase danému hráči stala.

5.2.4 Klubové události

Klub potřebuje vytvářet klubové události a následně na ně zvát členy klubu. Hlavní entitní množinou je klubová událost a následně její vztah s entitní množinou uživatelů. U klubové události ukládáme *název a místo události*, dále *datum události* a *čas srazu*, který je důležitý pro pozvané členy. Je zde

i atribut *typ klubové události*, který je cizím klíčem v entitní množině klubové události. Mezi typy klubové události budou řazeny i tréninky, pro zjišťování účasti na nich nebo pomoc členů na turnajích, či událostech, které mají za cíl získání peněz pro chod klubu.

Vazební tabulka pozvaní hráči není tvořena pouze primárními klíči spojovanými tabulkami, ale má i vlastními atributy *potvrzená účast* a případný komentář k účasti či neúčasti na dané klubové události.

5.2.5 Doklady

Doklady jsou důležitou součástí tohoto informačního systému, ale i života klubu jako takového. Doklady jsou rozděleny do dvou tabulek, a to na doklady které jsou vyhotovovány podle vnitřních předpisů Českého svazu házené (ČSH) a na jednoduché výdajové doklady. Atributy entitní množiny doklady jsou *název akce, datum plnění, místo plnění, jméno příjemce, adresa příjemce*, která je složený atribut, který se skládá z *ulice, čísla popisného, města a poštovního směrovacího čísla.*

Mezi nepovinné atributy v této entitní množině patří *datum narození* a *číslo občanského průkazu*, které není povinné, ale většinou je při výplatě rozhodčího potřeba, jelikož je tento údaj vyžadován Českým svazem házené.

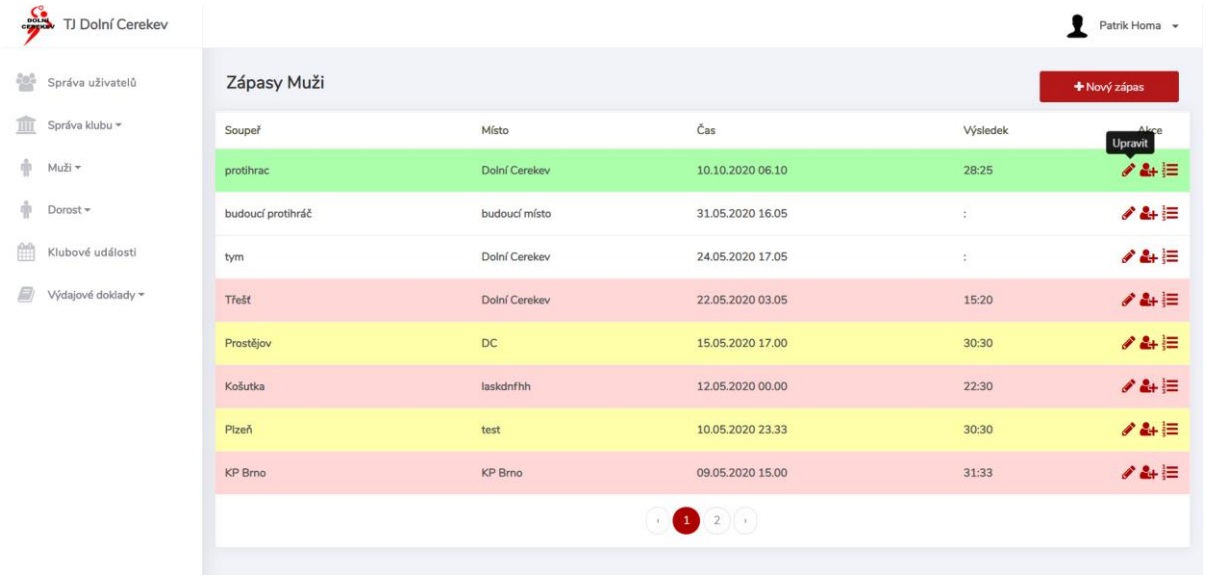
V informačním systému budou také rozlišovány různé *typy dokladů*, kde bude ještě ukládán *název typu*. Ve vztahu M:N budou pomocí vazební tabulky přiřazovány položky dokladu jako například odměna podle vnitřních předpisů Českého svazu házené, příspěvek na cestovní výdaje ať už na hromadnou dopravu nebo na dopravu vlastním vozidlem a případně výdaje na stravné nebo jiné výdaje. Mezi typy dokladů patří mimo výplaty rozhodčích také refundace nákladů za využití automobilu hráče využitého na dopravu hráčů na zápas.

5.3 Návrh uživatelského rozhraní

U návrhu uživatelského rozhraní je důležité myslet na to, aby byl systém pro uživatele co nejjednodušší a nejpřehlednější. Výsledný návrh byl také konzultován s uživateli, kteří budou informační systém využívat. Uživatelské rozhraní bylo navrženo v barvách klubu, které jsou červená, bílá a černá. Při vytváření návrhu byla využita šablona s názvem Ready Bootstrap Dashboard[1], která byla upravena pro požadavky Informačního systému pro házenkářský klub.

V horním panelu je bílý pruh, kde vpravo je název přihlášeného uživatele s rozbalovacím menu, ve kterém se nachází položky, týkající se přihlášeného uživatele. V levé části je postranní panel, ve kterém se nachází hlavní navigační menu. V největší prostřední části můžeme vidět tělo stránky, které se dynamicky mění, podle toho, jaká data chce uživatel zobrazit. Na obrázku 5.2 vidíme celkové rozložení stránky a v těle je výpis zápasů daného týmu v aktuální sezóně.

[1] Dostupné z: https://www.themekita.com/ready-bootstrap-dashboard.html



Obrázek 5.2: Návrh uživatelského rozhraní
Zdroj: Vlastní zpracování

5.3.1 Postranní panel s navigačním menu

Navigační menu je zafixováno v levém panelu, ve vrchní části je logo klubu, které uživatele po kliknutí přesměruje na domovskou stránku. Menu je rozložené do částí Správa uživatelů, které přesměruje uživatele na přehled všech uživatelů, dále správa klubu, která se po kliknutí uživatelem rozbalí a tato rozbalené nabídka obsahuje soutěže, týmy, sezóny a typy klubových akcí a dokladů.

Dále jsou v menu rozbalitelné položky týmů, které se dynamicky zobrazují podle toho, které týmy jsou uložené v databázi. Po rozbalení položky týmu se uživateli zobrazí statistiky, zápasy, tréninky a archiv zápasů a statistik podle sezón. Pod položkami týmů se nachází klubové události a výdajové doklady.

Kapitola 6

Implementace

Tato kapitola bude popisovat nejzajímavější části implementace informačního systému házenkářského klubu. Celá implementace vychází z analýzy a specifikace požadavků v kapitole Kapitola 4 a z návrhu systému, který vychází ze specifikace a je popsán v kapitole Kapitola 5

6.1 Adresářová struktura

Při tvorbě informačního systému byla dodržována adresářová struktura doporučená vývojáři Laravelu. Nachází se zde 9 hlavních adresářů a několik souborů, kde nejdůležitější budou popsány níže.

Mezi nejdůležitější soubory patří:

- **composer.json** a **composer.lock** – jsou konfigurační soubory, které používají nástroj Composer
- **package.json** a **package.lock** – jedná se o konfigurační soubory pro nástroj npm
- **.env** – nachází se zde základní nastavení aplikace jako například jméno aplikace a její url, dále se zde nachází nastavení přihlašovacích údajů do mail serveru a do databáze. Můžeme si zde přidat nastavení pro další nástroje, které budeme v aplikaci využívat.

app

Adresář **app** obsahuje hlavní zdrojové soubory aplikace. Je automaticky načten pomocí Composeru se standardem PSR-4. Mnoho podadresářů ve výchozím stavu neexistuje a je nutné je vytvořit například pomocí příkazu `make` Artisanu. Obsahuje dále další podadresáře. [15]

- **Console** – obsahuje veškeré příkazy příkazy Artisan
- **Exceptions** – nachází se zde vyjímky, které jsou vyvolávány v aplikaci. Pro úpravu logování a vykreslování výjimek je potřeba upravit třídu *Handler*, která se nachází v tomto adresáři.
- **Http** – v tomto adresáři je umístěna téměř veškerá logika pro zpracování požadavků, vytvářející se v aplikaci. Patří mezi ně controllery, middlewary a požadavky na formuláře.
- **Models** – v adresáři model jsou uloženy veškeré třídy modelů, které ve velké míře kopírují databázovou strukturu

bootstrap

Hlavní soubor adresáře je soubor **app.php**, který načítá framework a konfiguruje autoloading. Adresář také obsahuje složku cache, která obsahuje soubory vytvořené frameworkem za účelem optimalizace výkonu.

config

Zde jsou uloženy veškeré konfigurační soubory aplikace. Vývojáři Laravelu doporučují si celý adresář projít a pročíst soubory, pro zjištění, jaké má vývojář možnosti. [31]

database

V adresáři database, jsou uloženy veškeré migrační soubory a soubory se seedy

public

V tomto adresáři je uložen důležitý soubor **index.php**, který se jako první načítá při každém požadavku do aplikace. Dále obsahuje css a javascript sobory a obrázky.

resources

Jsou zde shromážděny veškeré pohledy (views) a nezkompilované javascriptové soubory nebo SASS či LESS. Nachází se zde také adresář lang s multijazyčnými soubory pro snadný překlad

routes

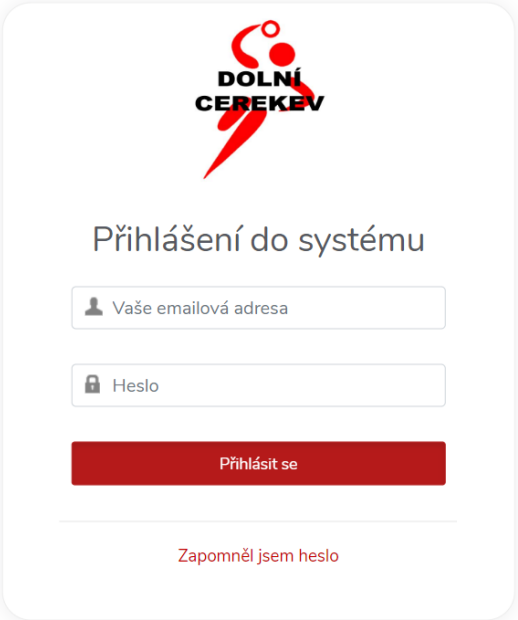
Toto je důležitý adresář, který obsahuje veškeré definované cesty (routes) v aplikaci. Nachází se v něm čtyři soubory, kterými jsou **Api.php, console.php, channels.php a web.php**.

Cesty uložené v souboru web.php umístí RouteServiceProvider do middlewaru skupiny web, který obsahuje vše, co uživatel webu obyčejně potřebuje, jako jsou sessions, cookies nebo CSRF ochrana.

6.2 Autentizace a autorizace

Jelikož se jedná o interní systém je nutné se vždy pro přístup přihlásit. Uživatel se přihlašuje pomocí e-mailu, který je používán jako uživatelské jméno, a hesla.

Pro přihlašování je využíván základní autentizační balíček, který se už nachází v Laravelu. Pro použití toho balíčku stačí spustit příkaz `php artisan ui vue auth`. Při nainstalování balíčku se nám vytvořily i autorizační cesty, nicméně hlavní soubory obsahující logiku autentizace jsou v adresáři **Auth**, který se nachází v adresáři Controllers. Pro nás jsou nejdůležitější soubory **LoginController.php** a **RegisterController.php**. Tyto soubory jsou primitivní a není v nich žádná důležitá logika, nicméně používají rysy (traity) **AuthenticatesUsers** respektive **RegistersUsers**, nacházející se v jádru Laravelu. Tyto rysy obsahují metody, které jsou při autentizaci využívány.



Obrázek 6.1: Přihlašovací obrazovka

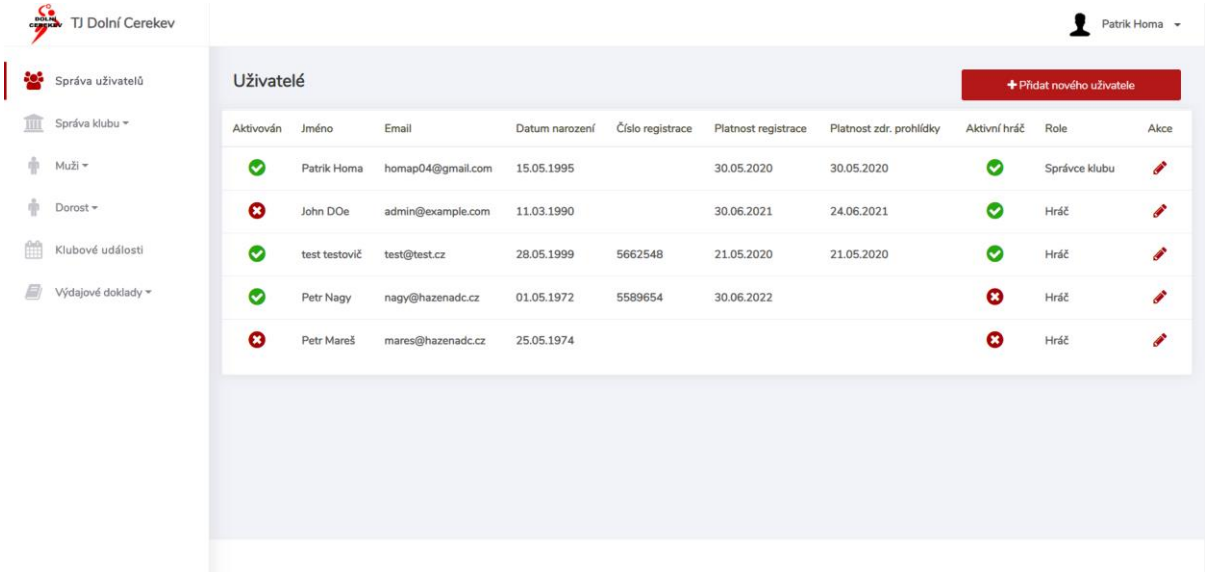
Pro autorizaci byly vytvořeny tabulky *Role* a *Práva*. Role jsou přiřazovány uživateli a práva jsou přiřazovaná roli pomocí spojové tabulky. Ke každé cestě je přiřazeno právo, pomocí middlewaru, které musí být přiřazeno roli uživatele. Při každém požadavku, který je vytvořen v systému, je porovnáváno, zda aktuálně přihlášený uživatel má k této akci právo.

V tomto systému jsou pro kontrolu přístupu využívány middlewary, které kontrolují, zda je uživatel přihlášen, respektive jestli má práva k provedení dané akce. Na obrázku Obrázek 6.1 je vidět přihlašovací formulář na který je uživatel vždy přesměrován, pokud není přihlášen.

Uživateli s rolí hráče se v hlavním menu zobrazují pouze všechny týmy, které jsou vytvořené, klubové události. Načítání týmů pro menu je řešeno v souboru **AppServiceProvider.php**, kde při každém načítání pohledu s menu, jsou z databáze vybrány všechny týmy a ty jsou poslány do pohledu.

6.3 Správa uživatelů a správa klubu

Správa uživatelů je první položkou v levém kontextovém menu. Přístup ke správě má pouze uživatel s rolí správce klubu. Uživatel s rolí hráče si může pouze zobrazit uživatele, kteří jsou v systému zaregistrovaní. Na hlavní stránce správy vidíme přehled všech uživatelů, kde jsou všechny důležité informace viz obrázek 6.2. Uživatelsky přívětivé je také zjednodušení, kdy pro aktivaci uživatele nebo pro změnu aktivního hráče stačí kliknout na zelenou, respektive červenou ikonu, a dojde k negaci aktuální hodnoty, bez nutnosti zobrazování formuláře pro editaci. Rozdíl mezi aktivním hráčem a aktivovaným účtem je zřejmý na první pohled. Aktivního hráče je možné přiřazovat na zápasy, zatímco položka aktivovaný účet slouží k informaci o možnosti přihlásit se přes autorizační údaje do aplikace. Správa uživatelů je uložena v souborech **UserController.php**, **User.php** a pohledy jsou uloženy ve složce **users**.



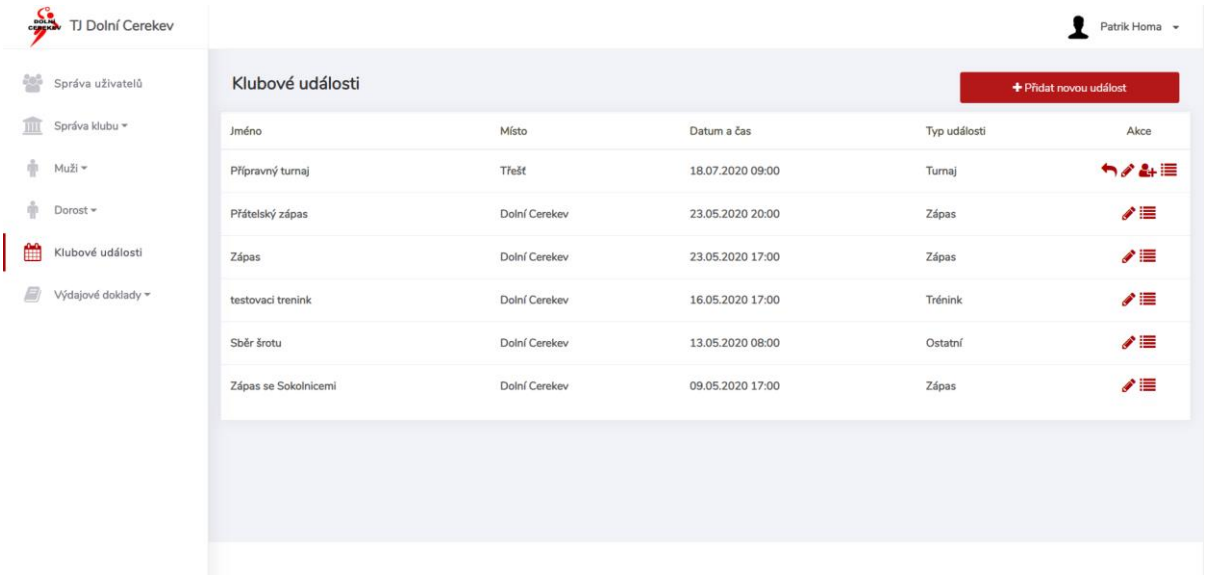
Obrázek 6.2: Hlavní stránka správy uživatelů

Ve správě klubu se nachází evidence soutěží, týmů, sezón a typů klubových akcí. Jsou zobrazovány ve stejném designu jako přehled uživatelů.

6.4 Klubové události

Důležitou součástí systému jsou klubové události. Z hlediska organizace ať již turnajů mládežnických kategorií nebo akcí, které jsou pořádány k vydělání prostředků na chod klubu, je nutné, aby se těchto událostí zúčastnili členové klubu.

V přehledové tabulce jsou zobrazovány všechny události seřazené sestupně podle data konání. V tabulce jsou uvedeny všechny důležité informace. Pokud jsem přihlášen pod rolí hráče, mohu si zobrazit přehled pozvaných uživatelů a jejich odpověď s případným komentářem na pozvání. Pokud uživatel na tuto pozvánku k události ještě neodpověděl, může odpovědět a události se zúčastnit, případně se omluvit. Vzhledem k tomu, že odhlašování z událostí není časté a probíhá vždy na osobní bázi s odůvodněním správci klubu, nepovažoval autor za vhodné zavést možnost změnit již jednou potvrzenou účast. V případě požadavku na změnu, autor přidá podmínku na zobrazení, nová metoda controlleru a nový pohled.



Obrázek 6.3: Klubové události

Uživatel s rolí správce klubu může navíc oproti roli hráče vytvořit novou událost a pokud ještě neproběhla, tak na ní může pozvat členy. Pokud je událost přiřazena k nějakému týmu, mohu na ní pozvat pouze hráče, které k tomuto týmu patří, pokud k týmu přiřazena není, lze na ní pozvat všechny aktivní uživatele. Přiřazení hráče k týmu je řízeno na základě věku uživatele a toho, zda je uživatel aktivním hráčem. Po pozvání je uživatelům odeslán email s informací, že byli pozváni na novou událost a s tlačítkem, který přesměruje uživatele na stránku s odpovědí.

6.5 Výdajové doklady

Výdajové doklady jsou děleny na doklady, které jsou nutné vydávat na základě vnitřních předpisů Českého svazu házené o odměně. Druhým typem dokladu jsou klasické jednoduché výdajové doklady, které využívají neplátci DPH.

Tyto jednoduché výdajové doklady jsou zároveň využívány pro evidenci jízd. Rozšíření na jiné evidence je velmi jednoduché, stačí přidat nový typ dokladu a ten se následně objeví i v menu v položce výdajové doklady. Doklady mohou vytvářet všichni uživatelé, avšak upravovat je mohou, dokud je správce nepotvrdí. Všechny typy dokladů je také možně si vyexportovat do PDF a následně vytisknout. To je důležité, protože doklady jsou potřeba potvrdit podepsáním a následně se předají příjemci. Pro

generování výkazů byl použit nástroj MPDF. Pomocí toho nástroje je načtena šablona, do které jsou následně doplňovány vyplněné údaje do formuláře.

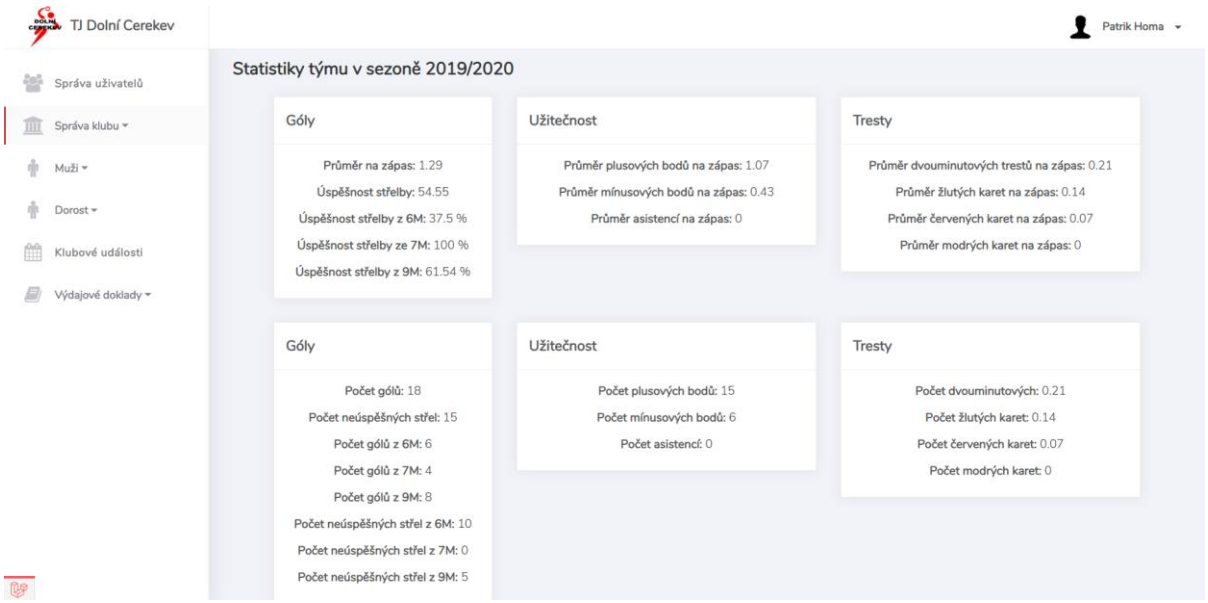
6.6 Týmy

Základní obrazovka u týmů je přehled zápasů. Zobrazují se zde všechny zápasy daného týmu v aktuální sezóně a jak je patrné z obrázku 5.2, lze velmi přehledně vidět výsledek utkání, a to díky podbarvení daného řádku. Nachází se zde také archiv zápasů a statistik týmu podle sezony.

Uživatel si může také zobrazit všechny tréninky a účast či neúčast hráčů na daném tréninky.

6.7 Statistiky

V systému se nachází také evidence statistik v zápase a následně jejich vyhodnocení a zobrazení. Jak je vidět na obrázku 6.4, vyhodnocení je jednoduché a je možné si zobrazit statistiky celého týmu v dané sezoně. Přihlášený uživatel si také může zobrazit své statistiky v aktuální sezoně, a to pomocí položky v menu přihlášeného uživatele.



Obrázek 6.4: Statistiky týmu

Kapitola 7

Testování

V této kapitole bude popsána další důležitá a nezbytná část ve vývoji informačního systému a tím je testování. Hlavním úkolem testování je zkoumání aplikace, odhalování chyb a následné zhodnocení kvality softwaru. Testování informačního systému pro házenkářský klub bylo prováděno ve dvou etapách na základě stadia vývoje systému.

7.1 Testování programátorem

První etapou testování bylo testování programátorem. Testování programátorem probíhá po celý životní cyklus vývoje informačního systému. Každá nově přidaná funkcionalita je programátorem testována, dokud nebudou všechny nalezené chyby odstraněny. V případě přidání a otestování většího celku funkcionalit, které dohromady tvořili část systému, byla tato část konzultována s budoucími uživateli. Na základě této konzultace byla výsledná část systému upravena a znovu otestována.

7.2 Testování uživateli

Druhá etapa testování probíhala po vytvoření funkční části systému a následně po implementaci celého systému. I když se jedná o interní systém, tak testování probíhalo nejenom na vybraném vzorku budoucích uživatelů výsledného informačního systému, ale předtím byli k testování pozváni rodinní příslušníci autora. Cílem této fáze testování bylo otestovat přehlednost navrženého uživatelského rozhraní a odhalit další chyby, které se objeví i přesto, že byl systém před touto etapou testován programátorem. Tyto chyby mohou vzniknout například jiným způsobem práce s informačním systémem.

V první vlně byl systém testován rodinnými příslušníky. Z této části testování se očekával výstup především ve formě zpětné vazby na uživatelské rozhraní. Tyto testovací osoby neznají, jak informace o vnitřní struktuře klubu, tak nejsou seznámeny s náležitostmi, které jsou potřebné k organizaci klubových událostí. Toto očekávání bylo naplněno a zpětná vazba od těchto testovacích osob se z větší části týkala rozložení jednotlivých komponent na stránce, zobrazovaných informací nebo rozložení polí ve formuláři, či formát formulářů samotných.

V druhé vlně byl systém testován vybranou skupinou uživatelů z řad členů klubu. U těchto členů se předpokládalo, že pro ně bude jednodušší a rychlejší provádět jednotlivé akce v informačním systému, a to jak z důvodu znalosti prostředí, tak díky zkušenostem nabytým organizací různých klubových událostí (tréninků, turnajů, zápasů nebo tanečních zábav). Tato predikce se při testování ihned potvrdila, kdy bylo od těchto uživatelů nahlášeno mnohem více chyb a případných návrhů na vylepšení než od rodinných příslušníků. Od členů klubu, byla obdržena zpětná vazba především na postupy a funkcionality systému, jako například oprava postupu, při vytváření klubových událostí a následné zvaní uživatelů. Na základě této vlny testování proběhla také diskuze na téma celkové funkcionality systému a možných vylepšeních, která se promítla do dalšího vývoje systému.

Díky testování posbíral autor této práce mnoho podnětů pro budoucí rozšíření webové aplikace.

Kapitola 8

Závěr

Tato práce si kladla za cíl tvorbu zcela nového informačního systému pro házenkářský klub TJ Dolní Cerekev. Požadavek na vznik obdobného systému přišel již před několika lety od členů klubu, kteří přemýšleli nad možnostmi, jak zefektivnit a usnadnit správu klubu. Vzhledem k tomu, že autor je dlouholetým členem klubu, a to jak v pozici hráče, rozhodčího, tak od května 2020 správce klubu pro informační systém svazu, sám vidí, jak velký význam webová aplikace pro klub má.

V první etapě práce na bakalářské práci bylo nezbytné stanovit si funkcionality, které má webová aplikace mít. Autor této práce sesbíral podněty pomocí dotazníkového šetření mezi funkcionáři a hráči klubu. Zároveň bylo s hráči a funkcionáři ujednáno, že v průběhu a především v závěru vývoje, webovou aplikaci otestují. Z testování vzešlo mnoho zajímavých podnětů, které autor práce následně implementoval.

Podařilo se tak naplnit cíl a vyvinout novou webovou aplikaci, která vyhovuje potřebám klubu. Mimo původně zamýšlených funkcionalit, které byly komplexně implementovány otestovány a fungují, rozšířil autor práce webovou aplikaci o statistiky zápasů a evidenci dokladů (v zadání práce se počítalo pouze s evidencí jízd a dokladů k jízdám).

Vzhledem k tomu, že doposud se veškerá administrativa nezbytná k fungování klubu zaznamenávala ručně tužkou na papír a následně se přepisovala do jednoduchých tabulek v MS Word či MS Excel, přivítali členové klubu možnost správy administrativy pomocí webové aplikace s nadšením.

Hlavními funkcemi systému jsou evidence statistik hráčů v zápasech, přihlašování na klubové akce a evidence výdajových dokladů. Tyto a mnohé další funkce vzešly jako požadavky z dotazníkového šetření nebo následujícího testování. Tyto funkcionality představují pouze základ webové aplikace, který je ale plně funkční. V následující podkapitole budou rozvedeny možnosti dalšího rozšíření webové aplikace. Jako nejdůležitější a nejvíce časově náročnou část práce bylo navrhnout strukturu databáze a sestavení ER diagramu. Autor práce má za to, že při budoucích projektech se na tuto část vždy zaměří, protože správně navržená struktura ušetří hodně času a zefektivní samotnou implementaci webové aplikace.

Webová aplikace bude zpřístupněna přes webové stránky obce Dolní Cerekev, která házenou žije. Webová aplikace bude z důvodu zrušení a anulování výsledků sezóny 2019/2020 zavedena na sezonu 2020/2021. Jelikož prozatím nedošlo k rozlosování termínů zápasů a ani k oznámení Českého svazu házené o tom, jaké bude složení jednotlivých soutěží, bude konkrétní termín komunikován členům klubu, až to bude aktuální.

8.1 Budoucí vývoj a navrhovaná rozšíření

Autor práce má v úmyslu na webové aplikaci v budoucnu dále pracovat, tak aby držela krok s požadavky jejích uživatelů. Případné podněty bude sbírat na následujících schůzích klubu. V ideální situaci by pak webovou aplikaci autor nabídl sportovním klubům v okolí Dolní Cerekve, jelikož žádný z nich doposud této možnosti nevyužívá. Autor práce má za to, že vzhledem k tomu, že klubová práce je věc dobrovolná a bez nároku na odměnu zaslouží si tito průkopníci sportu alespoň nástroje pro zefektivnění jejich práce. Dolní Cerekev je obcí s dlouhou tradicí házené a stejně jako svět prochází digitalizací, je třeba i správu klubu digitalizovat.

Autor práce se dále zamýšlel nad možnými rozšířeními webové aplikace, která plánuje v nejbližších letech implementovat. Jednalo by se například o:

- Napojení na API systému Českého svazu házené, s oboustranným propisem dat.
- Grafické zobrazení statistik a lepší možnost sledování svých výsledků, pro jednotlivé hráče.
- Rozšíření o informování emailem o blížící se lékařské prohlídce nebo končící platnosti registračního průkazu.
- Rozšíření o možnost posílání zpráv mezi uživateli.
- Napojení klubových událostí určené pro veřejnost na službu Hlášenírozhlasu.cz
- Vylepšení responsivity aplikace.

Je zřejmé, že tento seznam není konečný a praxe ukáže další potřeby budoucích rozšíření. Autor práce počítá s tím, že práce na webové aplikace bude dlouhodobá a cílem je udržovat v chodu moderní webovou aplikaci.

Literatura

1. 8 Popular PHP Frameworks For Web Development in 2020. *Hackernoon* [online]. 22.2.2020 [cit. 2020-05-18]. Dostupné z: https://hackernoon.com/8-popular-php-frameworks-for-web-development-in-2020-od3f38ez
2. Architektura klient-server. *Management mania* [online]. 4.11.2016 [cit. 2020-05-18]. Dostupné z: https://managementmania.com/cs/architektura-klient-server–
3. Bootstrap: Co je to, odkud se začít učit a jak ji používat. *Flipper World* [online]. [cit. 2020-05-18]. Dostupné z: https://cs.flipperworld.org/pc/bootstrap-co-je-to-odkud-se-zacit-ucit-a-jak-ji-pouzivat
4. Co je Laravel. *Laravel Blog* [online]. [cit. 2020-05-18]. Dostupné z: https://laravelblog.cz/co-je-laravel
5. ČÁPKA, David. MVC architektura. *IT Network* [online]. [cit. 2020-05-18]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor
6. ČÁPKA, David. UML - Use Case Diagram. *IT Network* [online]. [cit. 2020-05-18]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram
7. Eloquent ORM: Vytváříme třídy modelu. *ZOOM* [online]. [cit. 2020-05-18]. Dostupné z: https://zooom.cz/eloquent-orm-vytvarime-tridy-modelu/
8. HTML introduction. *W3School* [online]. [cit. 2020-05-18]. Dostupné z: https://www.w3schools.com/html/html_intro.asp
9. Informace o webových aplikacích. *Adobe* [online]. [cit. 2020-05-18]. Dostupné z: https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html
10. Laravel Homestead pro Windows 10. *Laravel Blog* [online]. [cit. 2020-05-18]. Dostupné z: https://laravelblog.cz/clanek/laravel-homestead-windows-10
11. LAZARIS, Louis, 2014. *CSS okamžitě*. Brno: Computer Press. ISBN 978-80-251-4176-2.
12. NARAMORE, Elizabeth, 2006. *Vytváříme webové aplikace v PHP5, MySQL a Apache*. Brno: Computer Press. ISBN 80-251-1073-7.
13. Python. *JetBrains* [online]. [cit. 2020-05-18]. Dostupné z: https://www.jetbrains.com/lp/devecosystem-2019/python/
14. Query builder. *Laravel documentation* [online]. [cit. 2020-05-18]. Dostupné z: https://laravel.com/docs/6.x/queries#introduction
15. Structure. *Laravel Documentation* [online]. [cit. 2020-05-18]. Dostupné z: https://laravel.com/docs/6.x/structure#the-app-directory
16. SUMMERFIELD, Mark, 2010. *Python 3: výukový kurz*. Brno: Computer Press. ISBN 978-80-251-2737-7.
17. The Main Features of MySQL. *MySQL* [online]. [cit. 2020-05-18]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/features.html
18. Třívrstvá architektura. *Management mania* [online]. 5.12.2015 [cit. 2020-05-18]. Dostupné z: https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture

19. Vlastní blog v Laravel: Layout. *Laravel Blog* [online]. [cit. 2020-05-18]. Dostupné z: https://laravelblog.cz/clanek/vlastni-blog-laravel-layout

20. WALLEY, Kenneth. Top 6 Most Used PHP Frameworks for Web Development 2020. *Dev.to* [online]. 12.11.2019 [cit. 2020-05-18]. Dostupné z: https://dev.to/websitedesignnj/top-6-most-used-php-frameworks-for-web-development-2020-46o5

21. Web programming languages: the best languages for web development. *Ionos* [online]. 11.3.2019 [cit. 2020-05-18]. Dostupné z: https://www.ionos.com/digitalguide/websites/web-development/web-programming-languages/

22. Webová aplikace. *Management mania* [online]. 18.10.2018 [cit. 2020-05-18]. Dostupné z: https://managementmania.com/cs/webova-aplikace-web-application

23. Webové aplikace. *NetDIRECT* [online]. 12.4.2007 [cit. 2020-05-18]. Dostupné z: https://www.netdirect.cz/slovnik-pojmu/670/webove-aplikace

24. WELLING, Luke a Laura THOMSON, 2017. *Mistrovství PHP a MySQL: programátorské techniky a webové technologie*. Brno: Computer Press. ISBN 978-80-251-4892-1.

25. What is MySQL? *MySQL Tutorial* [online]. [cit. 2020-05-18]. Dostupné z: http://www.mysqltutorial.org/what-is-mysql/

26. What is Python? *Python* [online]. [cit. 2020-05-18]. Dostupné z: https://www.python.org/doc/essays/blurb/

27. ZENDULKA, Jaroslav a Ivana RUDOLFOVÁ. Databázové systémy. *FIT VUT* [online]. [cit. 2020-05-18]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIDS-IT%2Ftexts%2FDatabazove_systemy.pdf&cid=11434

28. ZENDULKA, Jaroslav. Databázové systémy a návrh databází: Konceptuální modelování a návrh databáze. *FIT VUT* [online]. [cit. 2020-05-18]. Dostupné z: https://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/2_2.pdf

29. ŽÁRA, Ondřej, 2015. *JavaScript: programátorské techniky a webové technologie*. Brno: Computer Press. ISBN 978-80-251-4573-9.

30. ŽIŽKA, Ondřej. Nevýhody PHP v praktických ukázkách. *Ondřej Žižka* [online]. 31.7.2018 [cit. 2020-05-18]. Dostupné z: https://www.zizka.ch/pages/programming/php/nevyhody_php_v_praktickych_prikladech.html

31. Structure. *Laravel Documentation* [online]. [cit. 2020-05-18]. Dostupné z: https://laravel.com/docs/6.x/structure#the-config-directory

32. 10 Reasons Why Laravel Is The Best PHP Framework For 2019. *Clariontech* [online]. [cit. 2020-05-24]. Dostupné z: https://www.clariontech.com/blog/10-reasons-why-laravel-is-the-best-php-framework-for-2019

Příloha A

Obsah CD

- **/xhomap00/src** – zdrojové kódy informačního systému
- **/xhomap00/bp.pdf** – písemná zpráva ve formátu pdf
- **/xhomap00/instalace.pdf** návod na instalaci ve formátu pdf