VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INFORMAČNÍ SYSTÉM FOTBALOVÉHO TÝMU

INFORMATION SYSTEM OF A FOOTBALL CLUB

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MICHAL VAŠKO**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. SVETOZÁR NOSKO**
SUPERVISOR

BRNO 2018

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Vaško Michal**
Program: Informační technologie
Název: **Informační systém fotbalového týmu**
Soccer Team Information System
Kategorie: Uživatelská rozhraní

Zadání:

1. Nastudujte literaturu a existující řešení na téma informačních systémů sportovních klubů, zejména fotbalových. Přitom se zaměřte na uživatelská rozhraní takových systémů.
2. Navrhněte funkčnost, prostředí a postup implementace informačního systému pro fotbalový klub.
3. Zhodnoťte dosažitelné vlastnosti a funkce navrženého řešení a případně aktualizujte návrh řešení.
4. Implementujte navržený systém a demonstrujte jeho funkčnost na vhodném příkladu.
5. Vyhodnoťte dosažené výsledky a případně možnosti dalšího pokračování práce.

Literatura:

- Dle pokynů vedoucího práce

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Nosko Svetozár, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 6. listopadu 2018

Abstrakt

Cieľom bakalárskej práce je vytvoriť informačný systém pre futbalový klub. Zameriava sa na užívateľov tohto systému a ich možnosti interakcie. Tréner mužstva má právomoc vytvárať a evidovať zranenia, pokuty, alebo tréningy. Hráčom je potom umožnené sa tréningov zúčastniť. Systém obsahuje podrobné štatistiky hráčov v lige a ligovú tabuľku tímov. Po zápase sa dá prezerať zostava tímu a jednotlivé štatistiky hráča v zápase. Na vývoj práce bol využitý PHP framework Laravel verzia 5.7 a databáza MySQL.

Abstract

The goal of this bachelor thesis is to create information system. Thesis is focused on users and their interaction with system. Manager of the team has rights to create injury, fine or trainings. Players are allowerd to join the trainings. System contains detail statistics about players and information about team positions in the league table. After match you are allowed to check team squad and the player's statistics from the match. PHP framework version 5.7 and MySQL database were used for the development of the system.

Kľúčové slová

Futbal, informačný systém, webová aplikácia, uživateľské rozhranie, Laravel, HTML, CSS, JavaScript, Bootstrap, PHP, SQL

Keywords

Football, Information system, web application, user interface, Laravel, HTML, CSS, JavaScript, Bootstrap, PHP, SQL

Citácia

VAŠKO, Michal. *Informační systém fotbalového týmu.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Svetozár Nosko

Informační systém fotbalového týmu

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Svetozára Noska. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Michal Vaško
15. mája 2019

Poďakovanie

Moje poďakovanie patrí pánovi Ing. Svetozárovi Noskovi za jeho užitočné rady a odborné vedenie mojej práce. Poďakovanie patrí aj klubu FK Furča Haniska za poskytnutie informácií k tvorbe systému.

Obsah

Kapitola 1

Úvod

V dnešnej dobe uľahčuje tvorbu webových aplikácií veľké množstvo voľne dostupných frameworkov. Miesto dlhého písania rovnakých riadkov kódu, frameworky umožňujú opätovné použitie už vytvorených častí kódu na projekty, ktoré vyžadujú podobnú implementáciu. Obsahujú pred pripravené časti kódu, čim urýchľujú vývoj a zároveň šetria čas. Výhodou je aktívna komunita ľudí. Preto je na internete množstvo video návodov, alebo kníh, ktoré pomáhajú pri vývojárskych problémoch. Dnešné informačné systémy môžu byť vytvárané ako webové aplikácie. Od klasických desktopových sa líšia tým, že nemusia byť fyzický nainštalované. Potrebujú iba webový prehliadač. Sú teda dostupné z mnohých zariadení, ako sú mobilné zariadenia, osobné počítače a podobne.

Cieľom tejto bakalárskej práce je vytvoriť informačný systém pre futbalový klub s myšlienkou rozšírenia ho pre celú súťaž. Bude sa jednať o klub z nižšej slovenskej sútaže. Systém by mal umožniť lepšiu organizáciu klubu, čo bude zahŕňať spravovanie tréningov, pokút, alebo evidenciu zranení. Vynechané nebudú ani možnosti vedenia štatistík hráčov a klubu. Myšlienkou tohto projektu je poskytnúť možnosť organizácie klubom z nižších súťaží, bez možností vytvorenia si svojho systému. Štruktúra takýchto klubov je veľmi komplikovaná a evidencia štatistik veľmi slabá. Systém by mal uľahčiť vedenie klubu, zjednodušiť prístup k hráčom a ich povinnostiam.

K vytvoreniu informačného systému sa použije PHP framework Laravel. K jeho zvoleniu sa prišlo na základe jeho kladných vlastností, ktoré budú v tejto práci aj popísané. Vytvorený bude systém ako webová aplikácia. K optimálnemu zobrazeniu na všetkých zariadeniach sa využíva knižnica Bootstrap. Tá zabezpečuje responzívne zobrazenie obrazovky pre všetky zariadenia. Vďaka nej je vzhľad webu prispôsobený používanému vybaveniu. K základnému vzhľadu webovej aplikácie sa použijú zavedené praktiky tvorby stránok ako CSS, HTML a JavaScript.

Práca je systematický rozdelená do kapitol od teórie, až po implementáciu a testovanie. Druhá kapitola pojednáva o základných pojmoch, s ktorými je potrebné zoznámiť pre pochopenie tejto práce. Tretia kapitola sa zameriava na všetky použité technológie. Veľká časť je venovaná najpopulárnejším frameworkom. Zároveň obsahuje odôvodnenie, prečo bol vybraný framework Laravel pre túto pracú. Kapitola sa nevyhne ani popisu backend a frontend technológiám. Kapitola štyri špecifikuje zadanie. Úlohou tejto kapitoly je priblížiť čitateľovi zámer autora a objasniť zadanie. Piata kapitola sa zaoberá požiadavkami na systém. Analyzuje a skúma bližšie jednotlivé časti systému a užívateľov, ktorí sa v ňom vyskytujú. Návrh informačného systému sa objaví v šiestej kapitole. Jeho hlavnou výplňou je ER diagram. Nasleduje implementácia systému kde sú popísané najdôležitejšie implementačné kroky a snahou je detailne priblížiť čitateľovi vývoj projektu. Posledné kapitoly tvoria

testovanie, možnosti rozšírenia a záver. V testovaní sú budúci užívatelia webu pozvaní k jeho odskúšaniu a možnosti kriticky sa vyjadriť. Prebraté sú aj potencionálne rozšírenia. V závere je zhodnotený vývoj práce.

Kapitola 2

Základné pojmy

V tejto kapitole budú prebrané základné a významné pojmy ohľadom informačných systémov, webových aplikácií a ich tvorby. Ich uvedenie a pochopenie je dôležité k ďalšiemu postupu v tejto bakalárskej práci.

2.1 Informácia

Informácie sa využívajú k získavaniu znalostí. Spôsob ako sa meria hodnota informácie, závisí na fakte, ako nová získaná informácia zmení niečo, čo je už známe. Informácia sa vyjadruje jednotkou nazvanou bit. Jeden kúsok informácie o polovicu znižuje nevedomosť. Je to ekvivalentné zodpovedaniu na otázku "áno", alebo "nie" kde je rovnaká pravdepodobnosť správnosti odpovedania [21]. Informácie sú tiež interpretované dáta, ktoré majú sémantiku (význam) [24].

Vlastnosti informácie:

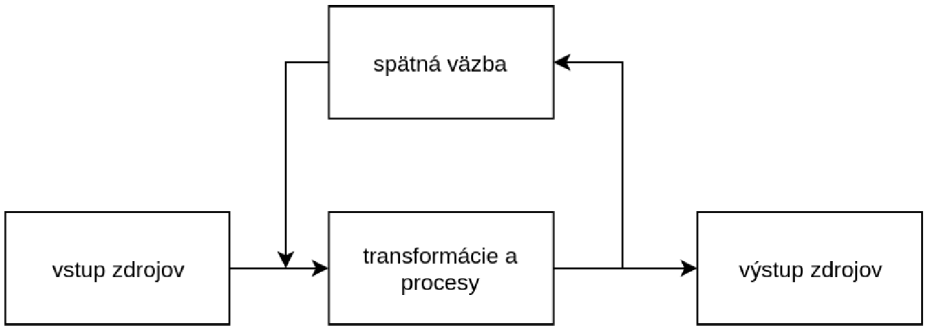
- nehmotná
- popisujúca význam
- vždy v rámci nejakého modelu
- v nejakom stupni abstrakcie

2.2 Dáta

Dáta sa chápu ako zbierka informácií na jednom mieste. Medzi takéto informácie patrí napr. číslo domu, názov mesta, alebo telefónne číslo. Sami o sebe nepredstavujú žiaden úžitok. Predstavujú hodnoty, ktoré sú nehmotné a majú rôzne dátové typy [18]. Hodnoty dát väčšinou udávajú stav systému. Dáta nemajú sémantiku, teda význam, sú to iba vety formálneho jazyka [20].

2.3 Systém

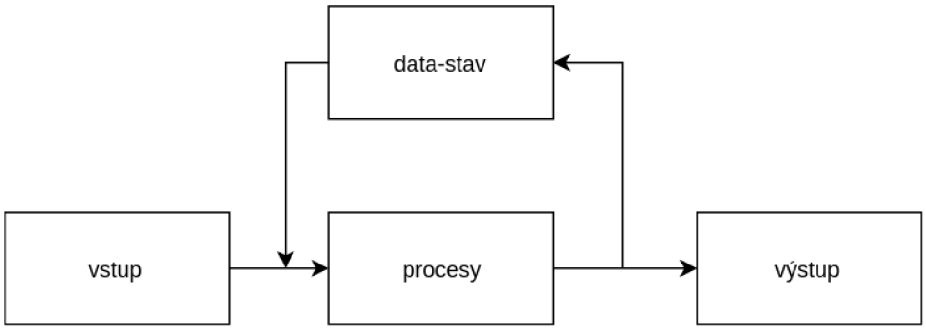
Systém predstavuje množinu prvkov medzi ktorými sú väzby, ktoré sú definované na nosiči. Pod nosičom chápeme množinu prvkov systému vo vzájomných vzťahoch a prvky nosiča nazývame zdroje [20].



Obr. 2.1: Obecná schéma systému.

2.4 Informačný systém

Informačný systém predstavuje obecný systém pracujúci s informáciami. Je zložený z vstupnej a výstupnej časti, odkiaľ sa do systému získavajú informácie. Medzi vstupnou a výstupnou časťou prebiehajú transformácie dát, ktoré prevádzajú algoritmy. Prebiehajú tu procesy, preto je nutné sa zaoberať spôsobom definície procesov, ich vzájomnej interakcií, alebo paralelným prevádzaním [24]. Na obrázku 2.2 je možné vidieť obecnú schému fungovania informačného systému, ktorá bola popísaná vyššie.



Obr. 2.2: Obecná schéma informačného systému.

Informačný systém je otvorený systém. Z hľadiska modelu teda modeluje skutočné zdroje iného, väčšinou fyzického systému. Model, ale nikdy nemôže obsiahnuť všetky vlastnosti, alebo chovanie svojej predlohy, preto je jeho virtuálna kopia vždy na istej úrovni abstrakcie.

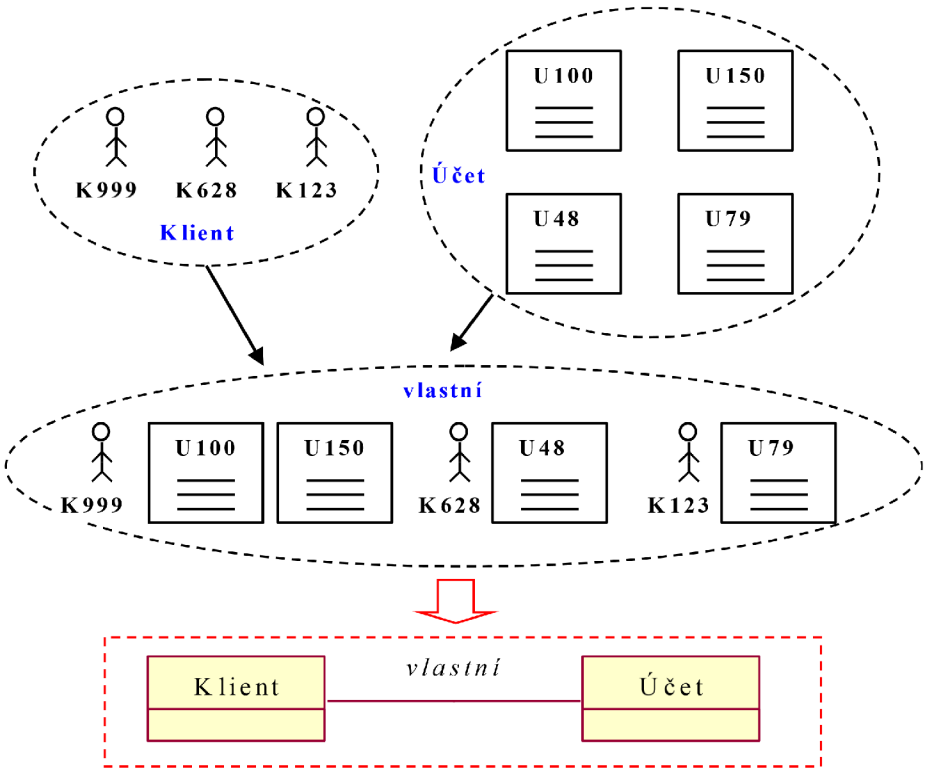
2.5 Entity Relationship Diagram

Entity Relationship Diagram (ER) v preklade entitný vzťahový model sa v softvérovom inžinierstve používa pre abstraktné a konceptuálne znázornenie dát. ER model je založený na chápaní sveta ako množiny základných objektov entít (Entity) a vzťahov (Relationship).

Dáta popisuje v kľude a zároveň neukazuje aké operácie sa budú s dátami robiť. Tretím základným prvkom modelu sú atribúty.

Základní pojmy

- **Entita** - vec, alebo objekt reálneho sveta rozlíšiteľný od iných objektov
- **Entitna množina** - množina entít toho istého typu, ktoré zároveň zdieľajú tie iste vlastnosti, alebo atribúty
- **Vzťah** - asociácia medzi viacerými entitami
- **Atribút** - vlastnosti entity, ktoré nás v kontexte daného problému zaujímajú
- **Vzťahová množina** - množina vzťahov toho istého typu, ktoré zdieľajú tú istú vlastnosť



Obr. 2.3: Proces tvorby ER diagramu.

2.6 Webová aplikácia

Ďalšou významnou časťou pri tvorbe informačných systémov je definovanie webovej aplikácie a jej fungovania.

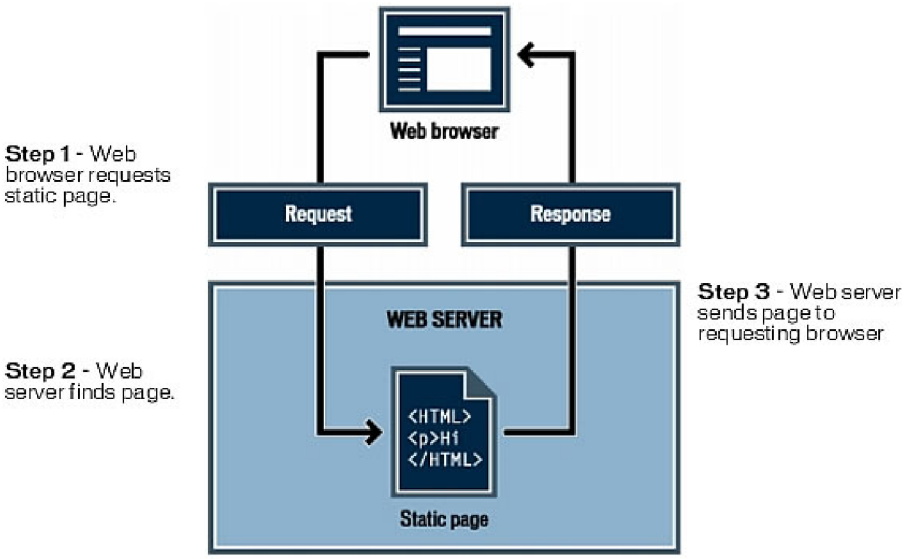
Je definovaná ako aplikácia, ktorú nie je nutné inštalovať na zariadenie užívateľa, teda na strane klienta, ale je možné ju spustiť z ľubovoľného zariadenia pomocou webového prehliadača, pretože už beží na strane serveru. Webovú aplikácia sa dá predstaviť ako miesto, ktoré obsahuje stránky s čiastočným, alebo neurčeným obsahom. Finálny vzhľad a obsah stránky sa určí, až keď užívateľ požiada o stránku z webového serveru [17].

Bežné využitie webových aplikácii

Webové aplikácie majú mnoho možnosti využitia pre vývojárov a návštevníkov webu. Medzi niektoré z nich patrí:

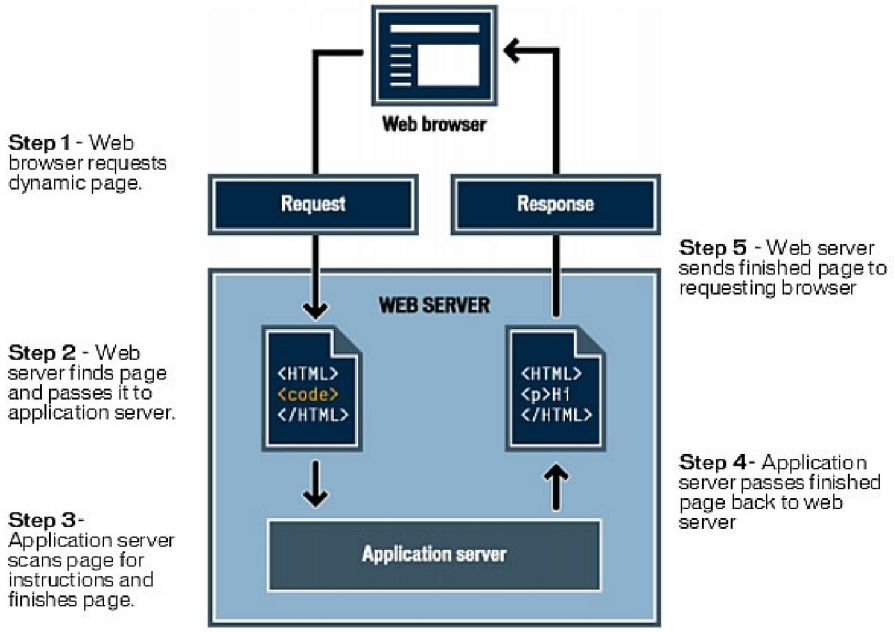
- Umožniť užívateľom rýchle a ľahké nájdenie informácií vo webovom mieste s bohatým obsahom
- Zhromažďovať, ukladať a analyzovať dáta, ktoré poskytli návštevnici webového miesta
- Aktualizovať webové miesta s nestále sa meniacim obsahom

Webová aplikácia sa skladá z kolekcií statických a dynamických webových stránok. Statická webová stránka sa nemení. Keď o ňu požiada užívateľ, webový server pošle stránku vo webovom prehliadači bez zmeny. Takéto statické webové miesto sa skladá z množiny súvislých stránok HTML a súborov hosťujúcich v počítači na ktorom beží webový server. Na základe požiadavok od webových prehliadačov, webový server posiela stránky. Takáto požiadavka sa vytvára, keď užívateľ na webovom prehliadači klikne na odkaz, prípadne vyberie záložku. Obsah stránky, ktorý dostane užívateľ je totožný s tým, čo navrhol návrhár stránky [5]. Fungovanie statickej webovej stránky je zobrazené na obrázku 2.4.



Obr. 2.4: Statická webová stránka [12].

V prípade dynamickej stránky je proces zložitejší. V momente ako webový server prijme požiadavok na dynamickú stránku, tak predá stránku softwaru, ktorý zodpovedá za dokončenie stránky. Obecne tento software nazývame aplikačný server. Princíp spočíva v tom, že aplikačný server si prečíta kód na stránke, dokončí ju podľa inštrukcií v kóde a potom ju odstráni zo stránky. Výsledkom je statická stránka, ktorú aplikačný server pošle späť webovému serveru, ktorý ju prepošle prehliadaču. Na konci prehliadač dostane už len čisté HTML [5]. Fungovanie dynamickej webovej stránky je zobrazené na obrázku 2.5.



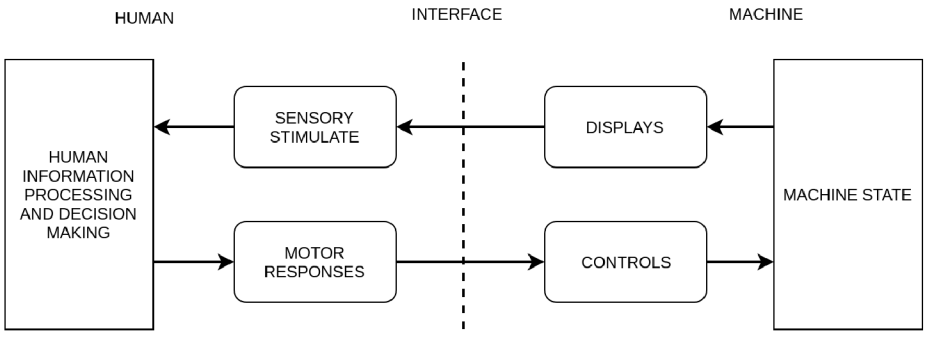
Obr. 2.5: Dynamická webová stránka [11].

2.7 Užívateľské rozhranie

Užívateľské rozhranie, alebo user interface, tiež označovaný ako HMI (human-machine interface), alebo MMI (man-machine interface) je jedným z prvkov ICT (informačnej a komunikačnej technológie), ktorý sa dá popísať ako komunikačný kanál medzi užívateľom a systémom. Zjednodušene povedané, je to súhrn spôsobov akým užívatelia ovplyvňujú chovanie systému [19].

Interakcia človek-počítač

Užívateľské rozhranie patrí do zložiek interakcie medzi človekom a počítačom, kde človek predstavuje koncového užívateľa stroja a stroj je zariadenie, na ktorom beží užívateľské rozhranie. Interakcia medzi týmito dvoma bodmi znamená zadávanie požiadaviek človekom, ktoré stroj následne vyhodnocuje a prezentuje späť užívateľovi cez užívateľské rozhranie [15]. Ilustrácia tohto procesu je na obrázku 2.6.

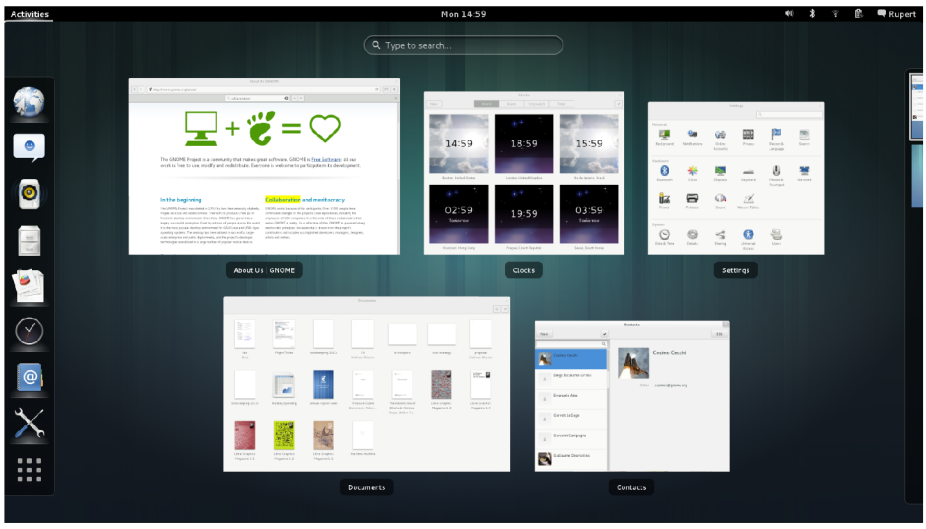


Obr. 2.6: Interakcia človek-počítač [15].

Na obrázku je vidieť HCI teda interakciu človek-počítač zjednodušenú na tri komponenty. Užívateľa, ktorý zadáva dotazy do systému, stroj, ktorý na základe týchto dotazov vyhodnocuje, zobrazuje dané odpovede a udalosti, ktoré vznikajú prebiehajúcou komunikáciou medzi nimi. Spôsobov interakcie medzi užívateľom a systémom je mnoho, a každý jeden je vhodný pre iný typ komunikácie. Interakcie môžu prebiehať pomocou gest, zvukov, príkazového riadku či prirodzeného jazyka. V dnešnej dobe je najrozšírenejšou technikou rozhranie s priamou manipuláciou. Užívateľ tu komunikuje so strojom pomocou hardwardového zariadenia, alebo dotykom prostredníctvom grafických prvkov v užívateľskom rozhraní, ako sú ikony, menu, okná a ovládacie prvky v kombinácii s textom. Takéto rozhranie sa nazýva grafické užívateľské rozhranie [15].

Grafické užívateľské rozhranie

Grafické užívateľské rozhranie ďalej už len ako GUI, je forma rozhrania, ktorá dovoľuje užívateľovi komunikovať s elektrickým zariadením pomocou grafických ikoniek, alebo vizuálnych indikátorov. Ako už bolo vyššie spomenuté, medzi tieto interaktívne grafické ovládacie prvky patria ikony, menu, okná, tlačítka, formuláre a podobné grafické prvky. Akcia v GUI je väčšinou prevádzaná pomocou priamej manipulácie s grafickými prvkami. Okrem počítačov sa GUI samozrejme využíva aj v mnohých iných zariadeniach, ako sú mobilné zariadenia, smartfóny či hracie zariadenia [14][3].



Obr. 2.7: Grafické užívateľské rozhranie.

2.8 MVC architektúra

Model-view-controller (MVC) je softwarová architektúra, ktorej základná myšlienka je oddeliť logiku od výstupu. Ide o návrhový vzor, ktorý pomáha navrhnúť a udržať štruktúru kódu pre projekt, zvýšiť jeho prehľadnosť, uľahčiť písanie a umožniť jednoduchšie udržiavanie kódu, prípadne aj jeho rozširovanie. Zámerom MVC je, aby zdrojový kód s logikou vyzeral ako zdrojový kód (napr. PHP kód) a výstup ako čistá HTML stránka bez čo najmenšieho zásahu ďalších kódov [35]. Celá architektúra je rozdelená do troch komponentoch. View predstavuje HTML stránku, ktorá je posielaná klientovi. Model sú doménové dáta

a Controller predstavuje časť aplikácie, ktorá spracováva požiadavky od užívateľa. Požiadavky sú aplikácií predávané pomocou HTTP requestov [26].

Model

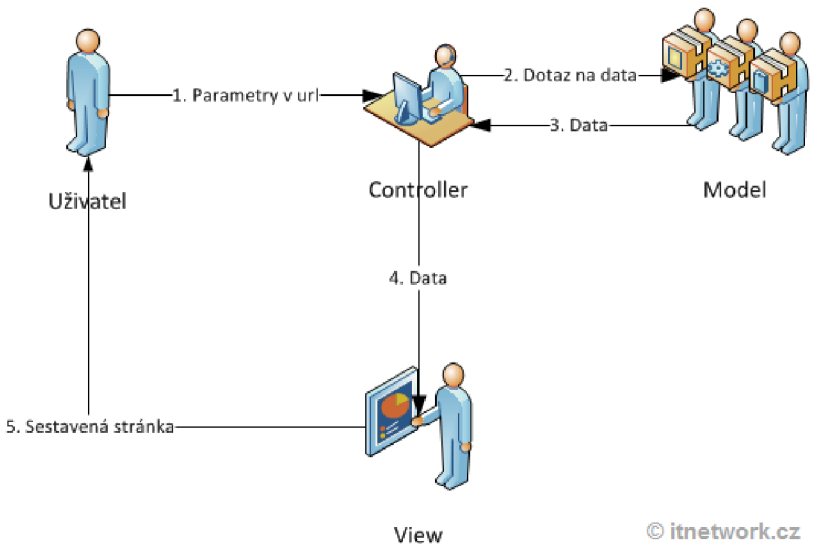
Model zahŕňa logiku a všetko s ňou spojené. Môžu ísť o databázové dotazy, výpočty, alebo validácie. O tom čo je na výstupe, model vôbec nevie. Jeho jedinou úlohou je prijať parametre zvonku a vydať tie dáta von. Model však nevie odkiaľ dáta v parametroch prišli, ani ako budú tieto dáta na výstupe reprezentované. Modely sa zo všetkých troch častí najviac blížia štandardným funkciám či metódam. Usporiadané sú do tried a používajú sa ich metódy pre manipuláciu s dátami [35][25].

View

Úlohou view (resp. pohľadu) je zobrazenie výstupu užívateľovi. Zaisťuje vykreslenie webu v HTML tak, aby sa užívateľovi všetko zobrazilo správne a so správnymi fontmi. Pohľadov môžme mať mnoho. Môže sa jednať o funkcionalitu s entitou administrátora napr. admin-login, admin-úvod a podobne. Všetko závisí od vývojára, čo si ako vytvorí. Napr. admin-úvod je potom spoločný pre všetkých administrátorov a menia sa tam len dáta podľa toho, kto je práve zobrazený. Pohľad podobne ako Model nemá informácie odkiaľ dostal dáta ktoré prišli, jeho funkcia je ich len zobraziť užívateľovi [35][25].

Controller

Posledná časť architektúry, ktorá spája predchádzajúce časti. Controller niekedy nazývaný aj prezenter, slúži ako prostredník s ktorým komunikuje užívateľ, model a view. Drží celý systém pohromade a komponenty prepojuje. Môže byť predstavený ako nejaký manažér v ktorom je popísaný proces, a čo všetko sa musí stať pre finálny výsledok [35][25].



Obr. 2.8: Diagram fungovania MVC architektúry [35].

Kapitola 3

Použité technológie

Nasledujúca kapitola bude zameraná na technológie, ktoré boli použité v bakalárskej práci. Keďže oblasť vývoja webových aplikácií a informačných systémov podlieha rýchlemu vývoju, cieľom bolo využiť a implementovať čo najnovšie možnosti v oblastí technológií. PHP bolo použité na naprogramovanie backendu informačného systému. Na vizuálnu časť stránky a jej vzhľad, teda frontend, bol použitý značkovací jazyk HTML, ktorý je určený na vytváranie webových stránok. K detailnejšiemu úpravu vzhľadu boli použité kaskádové štýly (CSS). Pre prácu s databázou bolo využité MySQL. Celá práca bola vyvíjaná pomocou PHP frameworku Laravel, konkrétne sa používala najnovšia dostupná verzia 5.7.

3.1 Frontend

Frontend predstavuje tú časť webovej stránky, ktorá je viditeľná a je možné s ňou pracovať [28]. Často je označovaná aj pojmom User Interface, teda používateľské rozhranie. Ako príklad môže byť uvedený napr. internetový obchod, kde je objednávkový formulár a všetko čo je tam viditeľné si predstavíme ako frontend. Na realizáciu frontendu sa najčastejšie využívajú HTML a CSS [23].

HTML

Hypertext Markup Language, alebo skratka HTML pomenúva názov značkovacieho jazyka, ktorý sa používa na tvorbu webových stránok, ktoré sú prepojené hypertextovými odkazmi. Navrhnutý bol v roku 1990 pánom Berners-Lee, ktorý okrem jazyka HTML navrhol aj protokol HTTP (Hypertext Transfer Protocol - protokol pre prenos hypertextu). HTML je hlavným jazykom pre vytváranie stránok v systéme World Wide Web, vďaka ktorému je možné publikovať dokumenty na internete. Je pomerne ľahký na naučenie a silný v možnostiach tvorby [4].

Význam Hypertext Markup Language

- Hypertext je metóda pomocou ktorej sa pohybujeme na webe. Klikaním na špeciálny text pomenovaný hyperlinks, ktorý privedie uživateľa na ďalšiu stránku.
- Markup, alebo značkovanie znamená, čo robia HTML tagy s textom medzi nimi. Označia to ako špeciálny typ textu a prevedú operáciu podla danej značky.
- HTML je jazyk, ako každý iný a má svoju syntax.

HTML pozostáva zo série krátkych kódov napísaných v textovom súbore. Ukladá sa ako html súbor a zobrazuje sa cez webový prehliadač, napr. cez Google Chrome. Daný prehliadač prečíta súbor a preloží ho do viditeľnej formy pre návštevníka stránky. Na to, aby však k tomu došlo sa využívajú značky (tags). Oddeľujú normálny text od HTML kódu. Sú to slová, ktoré sa nachádzajú medzi zátvorkami tzv. <angle-brackets>. Umožňujú nám vkladať text aký potrebujeme, pridávať obrázky, alebo prípadne tabuľky. Všetko zaleží na type značky akú použijeme. Pre príklad si uvedieme značku, ktorá nám zmení text [32].

```
<b>These words will be bold</b>, and these will not.
```

Vývoj HTML ide stále dopredu a najnovšou aktuálnou verziou je HTML 5.3. Táto verzia bola použitá aj v tejto bakalárskej práci.

CSS

Kaskádové štýly, skrátene CSS (skratka z anglického výrazu Cascadind Style Sheets) je všeobecné rozšírenie HTML. Zatiaľ čo HTML sa používa na základné formovanie webovej stránky, CSS ide viac do detailov a špecifikuje štýl dokumentu. Príkladom môže byť farba, fonty, alebo umiestnenie prvkov na stránke. CSS prináša štýl a detaily na webovú stránku pomocou interakcie s HTML elementami. Klasický paragraf v HTML vyzerá takto.

```
<p>This is my paragraph!</p>.
```

Pomocou CSS však môžme zmeniť napríklad farbu a hrúbku textu paragrafu.

```
p { color:pink; font-weight:bold; }.
```

Syntax kaskádových štýlov pozostáva z niekoľkých pravidiel. Každé pravidlo obsahuje selektor a blok deklarácií. Každý blok deklarácií obsahuje tieto deklarácie oddelené bodkočiarkou **;** a každá deklarácia pozostáva z identifikátoru vlastnosti. Na konci nasleduje dvojbodka **:** a hodnota vlastnosti.

Podobné ako HTML, CSS je napísané do jednoduchého textového súboru, Sú však tri spôsoby, ako môže byť CSS kód pridaný do HTML stránky. Pri prvom spôsobe sa ukladá napísaný textový súbor pomocou CSS do súboru s koncovkou css. Na to, aby to kooperovalo s HTML je treba zahrnúť v html súbore do hlavičky odkaz k nášmu vytvorenému súboru. Ďalšou možnosťou je napísanie CSS inštrukcií priamo do hlavičky html súboru. Toto je vhodné najmä v prípade, že existuje stránka s jedinečným vzhľadom. Treťou možnosťou je vpísanie CSS inštrukcii priamo do html kódu. Aplikuje sa to však len na dotyčný element. Obecne najpoužívanejšou formou je však prvá vďaka jej efektivite a priehľadnosti [30].

Bootstrap

Bootstrap [1] je jednoduchý open-source framework pre tvorbu webov a webových aplikácií. Obsahuje HTML a CSS návrhárske šablóny pre úpravu typografie, tlačidiel, formulárov a iných komponentov rozhrania. Medzi hlavné výhody patrí jeho responzívnosť. To znamená, že sa rozloženie stránky dynamicky prispôsobuje s ohľadom na používané zariadenie. Je to výhoda v prípade, ak programátor vytvára webovú stránku napr. na mobilné zariadenie, alebo iné zariadenia s rôznymi veľkosťami obrazovky. Je vhodný pre začiatočníkov, alebo programátorov, ktorí nemajú skúsenosti s tvorbou stránok. Stačí si stiahnuť vybranú šablónu a pomocou HTML ju zakomponovať do svojho projektu.

3.2 JavaScript

JavaScript [29] je vysoko úrovňový, dynamický, bez typový, skriptovací programovací jazyk. Vysoko úrovňový definuje jazyk vytvorený bez potreby vedieť detaily o chode počítača. Dynamický predstavuje možnosť rozšíriť kód, alebo pridať objekt za chodu programu. Je objektovo orientovaný a využíva sa hlavne pri tvorbe webových stránok. Beží na strane klienta. Spolu s HTML a CSS, JavaScript predstavuje jednu z troch hlavných technológií na tvorbu webov.

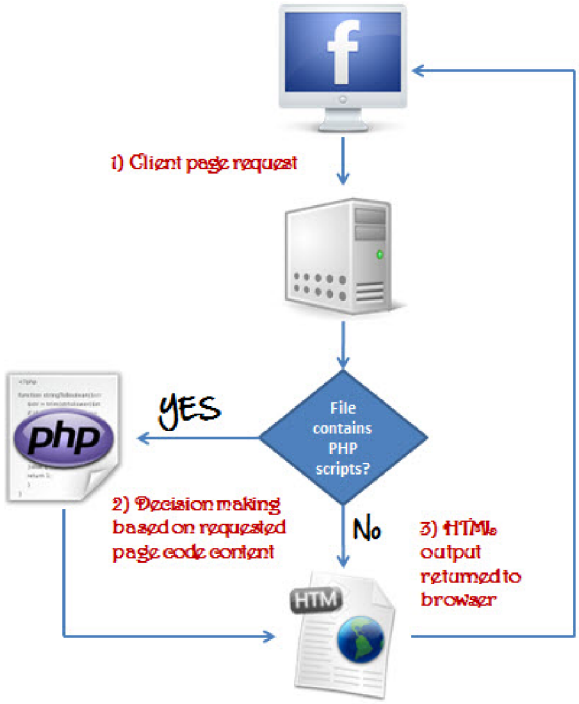
3.3 Backend

Backend tvorí pri tvorbe webov základ pre aplikáciu. Pre užívateľov je táto časť aplikácie skrytá. Jeho úlohou je úzko spolupracovať so serverom a databázou. Backend by sa pri objednávacom formuláre staral o to, ako bude web reagovať pri vyplnení a odoslaní formuláru. Určuje nám čo bude zobrazené pomocou frontendu [23].

PHP

Jedná sa o populárny open source jazyk, ktorý radíme k skupine skriptovacích. Využíva sa pri vývoji dynamických webových stránok a pri programovaní klient-server aplikácií, tie opisujú vzťah medzi dvoma počítačovými programami, kde jeden z nich je klient, ktorý posiela požiadavku na službu z druhého programu, ktorý sa nazýva server a ten ak môže, danú požiadavku splní. Ďalšie využitie pri tvorbe konzolových, alebo desktopových aplikácií. Pri aplikovaní PHP [9] pre dynamické stránky sú skripty prevádzané na strane serveru. Užívateľ produktu vidí až finálny výsledok činnosti. Syntax jazyka je inšpirovaná niekoľkými programovacími jazykmi ako Perl, C, Pascal alebo Java. Výhodou je, že nie je závislý na platforme, preto nie je problém používať hocijaký operačný systém. Je však veľmi populárny v kombinácií s operačným systémom Linux, databázovým systémom MySQL a webovým serverom Apache, kde táto kombinácia dostala prezývku LAMP a využíva sa k tvorbe webových aplikácii. Táto kombinácia bola použitá aj pri tvorbe tejto záverečnej práce [33].

Podobne ako iné programovacie jazyky, PHP sa stále posúva dopredu a napreduje. Najnovšou verziou je 7.2 ktorá bola použitá aj v tejto práci.



Obr. 3.1: Reagovanie serveru na požiadavky klienta v prítomnosti PHP.[16]

3.4 Databáza

Databáza je množina dát uložená v počítači. Tieto dáta sú väčšinou štrukturované tak, aby prístup k nim bol čo najjednoduchší. Jedným z typov databázy je aj relačná databáza. Relácia je primárna jednotka pre uloženie dát v relačnej databáze. Relačná databáza sa vyznačuje údajmi, ktoré sú uložené v tabuľkách. Každá tabuľka pozostáva z jedinečnej množiny riadkov a stĺpcov. Záznam je uložený ako riadok a atribúty dát v tabuľke definujeme do stĺpcov. Tieto tabuľky sú navzájom prepojené a musia byť medzi nimi vzťahy [24].

Pri využívaní databázy budeme používať SQL. Strutured Query Language je štandardizovaný štrukturovaný dotazovaní jazyk, ktorý sa používa na prácu s dátami v relačných databázach. Syntax SQL je rovnaká tej anglickej, preto je jeho používanie relatívne jednoduché. Medzi hlavné výhody môže byť zaradená rýchlosť, jednoduchosť a početná komunitu developerov, ktorí pomôžu pri vyskytnutých problémoch. Predstavené budú tri populárne databázy, z ktorých jedna bude využívaná pre túto prácu.

Oracle 12c

Populárna databáza od Oracle [8]. Môže byť nasadená na jednom, alebo viacerých serveroch. Umožňuje spravovať databázu obsahujúcu až miliardu záznamov. Medzi hlavné výhody sa radí obrovské množstvo nástrojov, ktoré umožňujú jednoduchšiu prácu s databázou. Ďalšou prednosťou je pravidelná aktualizácia od tvorcov z Oracle. Nevýhodou môže byť vysoká cena. Oracle databáza je preto určená skôr veľkým firmám s obrovskou databázou vyžadujúcou veľké množstvo nástrojov.

PostgreSQL

PostgreSQL [10] patrí medzi populárne voľne dostupné databázy. Je frekventovane využívané pri webových stránkach. Databáza sa dokáže prispôsobiť, preto zvládne terabajty dát bez výrazného poklesu výkonu. Podporuje JSON a má rozmanitý počet preddefinovaných funkcií. Medzi slabiny databázy je radená dokumentácia slabšej kvality, preto je potrebné si pohľadať veci ohľadom tvorby na iných stránkach. Pre nováčikov je problém aj konfigurácia, ktorá je mätúca.

MySQL

MySQL [7] je open source, SQL relačný databázový server. Podporovaný je na viacerých platformách. V prípade tejto práce sa bude využívať na Linuxe. Poskytuje veľké množstvo funkcií napriek tomu, že je voľne dostupná. Podporuje rozličné užívateľské rozhrania, ktoré môžu byť implementované. Výhodou je možnosť spolupráce s ďalšími databázami ako je DB2, alebo Oracle. Najčastejšie sa používa pri kombinácii s PHP na tvorbu webových aplikácií. Je súčasťou a aplikácie LAMP, ktorá už bola spomenutá a preto sa bude využívať v tejto práci.

3.5 Backend frameworks

Webový framework (v skratke WF) je softwarový framework ktorého cieľom je podporovať vývoj webových aplikácií. Cieľom frameworkov je zamerať sa na automatizáciu tvorby webu a na bežné aktivity pri jeho tvorbe. Poskytuje knižnice pre prístup do databázy, šablóny, alebo opätovné použitie existujúceho kódu(v angličtine pojem code reuse), teda vygenerovanie už šablóny kódu, ktorá tvorí základ aplikácie. Frameworky používame primárne pre dynamické webové stránky, ale sú použiteľné aj pre statické stránky.

Ich vývoj stále napreduje a každým rokom prichádzajú s novými funkciami a vymoženosťami. Predstavené budú štyri najpopulárnejšie webové frameworky a dôvod, prečo bol vybraný práve jeden z nich.

Flask

Flask [31] je populárny mikroframework napísaný v Python. Mikro znamená, že nepoužíva knižnice ani iné nástroje. V podstate neobsahuje validáciu formulárov, databázovú vrstvu, alebo iné komponenty, ktoré využívajú existujúce knižnice. Má to svoje výhody aj nevýhody. Za výhodu sa považuje jeho minimálna závislosť na aktualizáciách, či stráženie bezpečnostných chýb, ktoré môžu nastať. Nevýhody spočívajú v tom, že programátor niekedy musí spraviť viac práce v dôsledku chýbajúcich knižníc. Flask podporuje rozšírenie ako napr. pridanie funkcionality do aplikácie ako keby bola implementovaná priamo vo Flasku.

Najznámejšie aplikácie využívajúce Flask sú Pinterest a Linkedln.

Django

Django [2] patrí tiež do rodiny frameworkov napísaných v Python. Je založený na štruktúre Model-View-Template (MVT) čo je variácia známejšej štruktúry MVC, ktorá bola predstavená vyššie. Pomáha vývojárom napísať čistý a efektívny kód. Používajú ho spoločnosti ako Instagram, Youtube, Google, alebo NASA pre ich webovú stránku. Je definovaný ako framework, ktorý obsahuje batérie, čo je zaužívaný pojem, ktorý znamená, že môžeme použiť veci

z "krabice". To znamená importovať balíčky, ktoré budeme potrebovať k vývoju programu. Medzi balíčky sa zaraďuje autorizácia užívateľa, alebo balíček, ktorý obsahuje veci potrebné pre prácu administrátora, napr. rozhranie webu, alebo prihlasovanie administrátora. Toto všetko uľahčuje programátorom hodiny písania vlastného kódu.

Symfony

Symfony [13] je framework napísaný v PHP a sám seba popisuje ako framework pre webové projekty, ale súčasne ako sadu komponentov, ktoré môžme opätovne využiť. V skratke to znamená, že aj keď jeho framework nie je používaný pre smerovanie požiadavky, alebo pre iné úlohy súvisiace s vytváraním webu, stále je možné využívať jeho komponenty pre jednotlivé úlohy ako je šablónovanie, správa konfiguračných súborov, či ladenie. Symfony je klasický Model-View-Controller (MVC) framework a teda využíva jeho architektúru. Podobne ako Laravel aj Sympfony využíva príkazový riadok, ktorý sa používa pre vytváranie a správu projektov. Obsahuje funkcionalitu, ktorá uľahčuje spúšťanie projektov vo vstavanom webovom servery PHP. Medzi výhody Symfony sa považuje skutočnosť, že je komerčne sponzorovaná. SensioLabs, tvorca spoločnosti a sponzor, pravidelne poskytujú oficiálne návody a certifikáty. Zároveň veľa významných platforiem ako phpBB, Drupal, Magento, alebo eZ Publish používajú jej komponenty. V neposlednej rade má Symfony obrovskú komunitu fanúšikov, ktorí prispievajú k tvorbe tohto frameworku [33].

Laravel

Laravel [6] je open source PHP framework pre webové aplikácie. Vyvinul ho Taylor Otwell. Charakterizuje ho rýchlosť a priehľadné riešenie webových aplikácií. Využíva softwarovú architektúru Model-View-Controller (MVC) popísanú v predchádzajúcej časti. Opiera sa o frameworkové komponenty a podporné systémy ako sú Sympfony, Composer, Eloquent ORM a Blade. Taktiež si zobral inšpiráciu od technológií, ako Ruby on Rails a Sinatra [33].

Artisan

Artisan je súčasťou Laravelu. Je to príkazový riadok, ktorý obsahuje príkazy pre zrýchlenie a uľahčenie opakujúcich činností vykonávaných pri tvorbe webových aplikácií. Pomocou neho dokážeme generovať jednotlivé časti kódu, ktoré tvoria kostry aplikácie, používať migráciu databázy, či prepínať stavy aplikácie.

Blade

Ide o jednoduchý šablónovací nástroj z dielne Laravelu. Jeho hlavná úloha pozostáva zo zabudovania PHP kódu priamo do pohľadov, ktoré tvorí HTML kód. Všetky Blade šablóny majú *.blade.php* rozšírenie v názve súboru.

Composer

Hlavným princípom composeru je spravovať závislosti jednotlivých knižníc či balíčkov na projekte. Primárnym cieľom je ich inštalovať, alebo aktualizovať. V angličtine sa pomenúva ako "package manager", kde v preklade by sa dal zjednodušene pomenovať ako manažér balíčkov. Je neoddeliteľnou súčasťou PHP frameworkov, vrátane Laravelu. Spúšťa sa pomocou príkazového riadku.

Výhody Laravelu

- Laravel aplikuje moderné postupy vývoja webových aplikácií. Ako už bolo spomenuté, využíva Composer pre správu závislosti, výborne pracuje s migráciou potrebnou pre chod databázy a svoju činnosť ešte zefektívňuje a urýchľuje pomocou príkazového riadku Artisan.
- Má prehľadnú a jednoduchú syntax.
- Obsahuje MVC architektúru. Vyhýba sa starým tradičným architektúram kde vývojári písali HTML a PHP kód do jedného súbora.
- Každý vývojár ma k dispozícií veľmi priehľadne spracovanú dokumentáciu.
- Pre prístup do databázy sa využíva Laravel Query Builder. Poskytuje priamy prístup do databázy. Dokáže vytvárať špecifické SQL dotazy.
- Oproti ostatným frameworkom má Laravel zabudovaný nástroj pomenovaný Artisan. Jedná sa o príkazový riadok, ktorý poskytuje množstvo užitočných príkazov. Medzi takéto príkazy patrí napr. migrácia tabuliek do databázy.
- Popularita Laravelu stále narastá a jeho obľúbenosť je veľmi vysoká. Z toto vyplýva, že má nespočetne veľa materiálov pre začiatočníkov či pokročilých. Medzi najznámejšie vzdelávacie weby patrí Laracast. Obsahuje cez štyristo video návodov, vďaka čomu je učenie Laravelu o dosť jednoduchšie.

Na prácu bude použitý Laravel. K rozhodnutiu prišlo na základe všetkých jeho kladných vlastností. Zo všetkých možných frameworkov sa javí ako najlepšia voľba pre tvorbu webových aplikácií. Je najrozšírenejší framework na svete, preto poskytuje množstvo materiálov a návodov k tvorbe. Ma prehľadnú a jednoduchú syntax, vďaka čomu sa s ním ľahšie pracuje. Využíva Laravel Query Buider, používajúci SQL jazyk. Ako programátor s ním mám dlhodobejšie skúsenosti oproti ostatným frameworkom. Obsahuje teda všetky prvky potrebné k vytvoreniu požadovaného informačného nástroju.

Kapitola 4

Špecifikácia zadania

Pred analýzou a návrhom bude najprv bakalárska práca špecifikovaná v tejto kapitole. Cieľom je ozrejmiť čitateľovi jej zámer a vyžitie v reálnom svete.

Témou tejto práce je Informačný systém futbalového klubu. Pod týmto názvom sa dá predstaviť veľa variácii takéhoto systému. Úlohou tohto systému je poskytnúť užívateľovi priehľadný fungujúci systém, ktorý môže využívať amatérsky futbalový klub z nižších súťaží. Keďže takéto kluby nemajú silnú infraštruktúru a dostatočný počet zamestnancov, ktorí by sledovali a informovali o dianí v klube. Snahou je rozšíriť informačné kluby do amatérskych klubov a zlepšiť tak ich technické vybavenie, prípadne zrýchliť beh klubu. Podstatou je odlíšiť sa už od existujúcich portálov, ktoré sa viac zameriavajú na štatistiky tímov a mužstiev a nejdú do hĺbky v rámci užívateľských možností personálu. Plán systému je sprístupniť ho najprv jednému klubu, ktorý si ho bude sám zriaďovať, no po skúšobnej dobe by sa mohol spustiť pre celú ligu a nasadiť do používania pre všetky dostupné tímy.

Zameriava sa na dvoch konkrétnych užívateľov systému, hráča a trénera. V rámci trénera sa upriami na tri aspekty jeho práce, a to tréningu, zranení a pokút. Bude umožňovať trénerovi vytvoriť tréning a nastaviť jeho parametre. Na druhej strane, hráčovi bude umožnené sa naňho prihlásiť, prípadne odhlásiť. Tréner tak bude mať prehľad o tom, kto sa zúčastní tréningu. Zamedzí sa tak krkolomnému dohadovaniu na sociálnych sieťach, prípadne osobnému dohadovaniu. Všetky tieto informácie budú uložené aj v systéme a databáze. Tréner si tak bude môcť kedykoľvek overiť kto bol na tréningu konkrétny deň.

Druhým podstatným účelom budú pokuty. Veľa krát sa v klube stáva, že hráč poruší pravidlá klubu. Medzi takéto prípady patrí zbytočná červena karta za surové správanie na ihrisku, alebo neskorý príchod na tréning, či zápas. Trénerovi hlavného tímu tak bude umožnené zadať pokutu konkrétnemu hráčovi so špecifikovaním prečo pokutu dostal, a aká je suma, ktorá musí byť vyplatená. Manažér má opäť možnosť to spätne pohľadať. Zaisťuje mu to možnosť porovnávať si kto, a kedy dostal pokutu a sledovať disciplinárne prehrešky tímu. Respektíve upozorniť často pokutovaného hráča na disciplínu.

Tretím aspektom sú zranenia. Hráči prichádzajú k zraneniam na tréningu, alebo v zápase. Zranenia hráčov sa v amatérskych kluboch historicky neeviduje a nevie sa s presnosťou určiť jeho dĺžka, keďže kluby nemajú profesionálnych lekárov. Zámerom je preto možnosť zaevidovania zranenia hráča, typ zranenia a určiť približnú dĺžku zranenia. Tréner tak bude vedieť, kedy je možný návrat hráča. Taktiež si bude môcť pohľadať históriu zranení.

Najpodstatnejšou špecifikáciou pre hráčov bude možnosť potvrdenia účasti na tréningu. V nižších súťažiach nie sú tréningy usporiadané na pravidelnej báze, a ani účasť na nich nie je vždy stopercentná. Cieľom systému je umožniť takýmto hráčom vstúpiť na web

a skontrolovať si termíny tréningov a jednotlivo sa na ne prihlásiť. Ak sa naskytne problém, je možné sa odhlásiť a tréner tak má prehľad o účasti na tréningu.

Zameriavať sa bude aj na súpisky. Prvotným plánom bolo vytváranie súpisiek pred zápasom, čo sa však zamietlo po konzultácií s trénerom mužstva. Bolo by to veľmi nepraktické, keďže tréner zostavu určuje až tesne pred zápasom. Preto sa bude klásť dôraz na po-zápasovú súpisku. Určí sa kto hral, koľko gólov dal a koľko kariet dostal. Jednotlivý tréneri si tak budú môcť nájsť ich odohráte zápasy, porovnať, ktorá zostava bola úspešná a analyzovať.

V špecifikáciu zadania bol vysvetlený podrobnejší úmysel tejto práce. Priniesť systém pre kluby z nižších súťaží a tried, a zároveň sa zamerať na organizačnú aj užívateľskú časť.

Kapitola 5

Analýza požiadaviek

V nasledujúcej kapitole bude rozobratý existujúci informačný futbalový portál na Slovensku. Budú sa analyzovať nedostatky a požiadavky na zlepšenie v mojom riešení. Analýza patrí medzi najdôležitejšie veci pri tvorbe informačného systému. Môj návrh som robil na základe konzultácií s trénermi futbalových klubov. Keďže sa už dlho pohybujem vo futbalovom prostredí, mal som možnosť pozorovať veci, ktoré by sa hodili a bolo by ich možné implementovať v rámci informačných systémov. Mojím zameraním je spraviť systém pre nižšie ligy a kluby, kde to prvotne bude testovať jeden klub. Na základe testovaní bude ďalšou možnosťou rozšírenie systému pre kluby z celej ligy.

5.1 Existujúci futbalový portál

Na slovenskej scéne momentálne nefiguruje veľa informačných portálov. Nájsť môžeme len jeden oficiálny web, ktorý združuje všetky ligy na Slovensku pod hlavičkou Slovenského futbalového zväzu. Týmto portálom je futbalnet[1]. Ako už bolo spomenuté, zaoberá sa všetkými ligami na Slovensku od najvyššej po najnižšiu. Jeho obrovskou výhodou je, že na jednom mieste sa môže návštevník dostať k rôznym štatistikám, či tabuľkám z rôznych líg. Dalo by sa povedať, že sa jedná skôr o štatistický portál v poslednej dobe aj s komerčným zameraním, ktorý poskytuje užívateľom štatistiky ohľadom futbalových klubov, ale chýba mu väčšia hĺbka a využitie v rámci služieb pre hráčov a trénerov.

5.2 Analýza futbalnetu

V nasledujúcej podkapitole bude analyzovaný portál futbalnet. Cieľom analýzy je priblížiť čitateľovi možnosti podobného systému. Futbalnet sa síce orientuje viac na štatistiky, no budú zdielať podobné vlastnosti v niektorých oblastiach.

Možnosti informačného systému Futbalnet

- Na jednom mieste je možné nájsť všetky sútaže slovenského zväzu. Od najvyššej sútaže až po najnižšiu okresnú ligu.
- Keďže sa v prvom rade jedná o štatistický portál, obsahuje množstvo štatistických tabuliek ako je počet strelených, či inkasovaných gólov.

[1] Dostupné na https://futbalnet.sk/

- Futbalnet sa nespráva len ako portál, ktorý informuje o štatistikách, ale vydáva aj novinové články v ktorom sa dajú nájsť aj rôzne rozhovory s hráčmi, alebo inými členmi klubu.
- Medzi jeho najnovšie rozšírenia patrí aj online obchod. Kde je možné si zakúpiť športové potreby.
- Informuje aj o iných súťažiach nespojených s futbalom. Najnovšie sa rozrástol o súťaž zaoberajúcou sa futsalom.
- Možné je navštíviť aj sieň slávy slovenského futbalu.

V analýze boli ukázané možnosti podobného systému zaoberajúceho sa futbalovou tematikou.

V nasledujúcej sekcií sa budú analyzovať užívatelia systému, ktorí budú vytváraní v rámci tejto práce. Cieľom bude uviesť si jednotlivé požiadavky, ktoré majú títo užívatelia spĺňať. Analyzovaní budu všetci užívatelia, ktorí budú v systéme pôsobiť. Sekcia sa bude zaoberať ich rolami a jednotlivými právomocami v informačnom systéme. Cieľom bude ustanoviť čo najpresnejšiu analýzu, aby sa následne mohol vytvoriť návrh systému.

5.3 Užívatelia systému

Každý informačný systém musí obsahovať užívateľov, ktorí v ňom vykonávajú činnosť. Jadrom celého sýtemu je administrátor, ktorý sa stará o chod systému. Jeho hlavnou úlohou je vytvárať ďalších užívateľov sýtemu, vytvárať ligy, alebo sezóny v závislosti na to, v akom systéme administrátor pracuje. Tento systém okrem administrátora bude obsahovať ďalších troch užívateľov. Dvaja z nich budú mať právomoci zasahovať do diania a tretí z nich, neprihlásený užívateľ bude akýkoľvek návštevník webu, ktorý si bude môcť prezerať štatistiky hráčov, líg a pod., ale nebude mu umožnená možnosť sa registrovať a následne sa prihlásiť a zasahovať do portálu.

Role užívateľa

- Administrátor
- Tréner
- Hráč
- Neprihlásený užívateľ

5.4 Administrátor

Pod pojmom administrátora sa rozumie osoba s najvyššími právomocami v rámci systému. Stoji nad všetkými užívateľmi portálu. Má možnosť meniť a pridávať nové veci v informačnom systéme, bez jeho práce by portál nemohol fungovať. V následujúcom texte budu popísané požiadavky na jeho prácu.

Vytváranie nových užívateľov, registrácia a prihlasovanie

Administrátor bude v systéme už vytvorený. Na základe požiadavky od príjemcu webu môže informačný systém obsahovať aj viac administrátorov. Hráčov a trénerov bude vytvárať administrátor informačného systému. Vytvorí ich meno, priezvisko, vek, pozíciu na akej hrajú a príslušnosť ku klubu. Cieľom je vytvárať hráčov najprv ručne pomocou administrátora, no postupne s nárastom využívania systému a zvyšovaním počtu hráčov by sa prešlo na automatické vytváranie účtov pre hráčov.

Pri vytváraní účtu administrátor zadá aj heslo. Všetky tieto informácie platia aj pre trénera. Pri jeho vytvorení mu zadá meno, priezvisko, vek a klub, ktorý trénuje. Jeho vytváranie bude stále ručné, keďže počet trénerov v systéme nikdy nepresiahne vysoké množstvo.

Sezóny a ligy

Sezóny a ligy bude vytvárať administrátor. Jedným zo zámerov bude možnosť prezrieť si minulé ročníky futbalovej sezóny, teda do hľadať si, ako si viedol daný tím v danej sezóne. Jej účel bude uchovávať dáta o lige z predošlých rokov. Každá sezóna bude trvať jeden rok, teda dĺžku trvania futbalovej sezóny. Sezóna sa vždy vytvorí pred začatím ročníka. V nej sa vytvoria ligy, ktoré sa budú v daný rok odohrávať. Každá liga bude obsahovať počet tímov, ktoré sa do nej prihlásili.

Zápasy

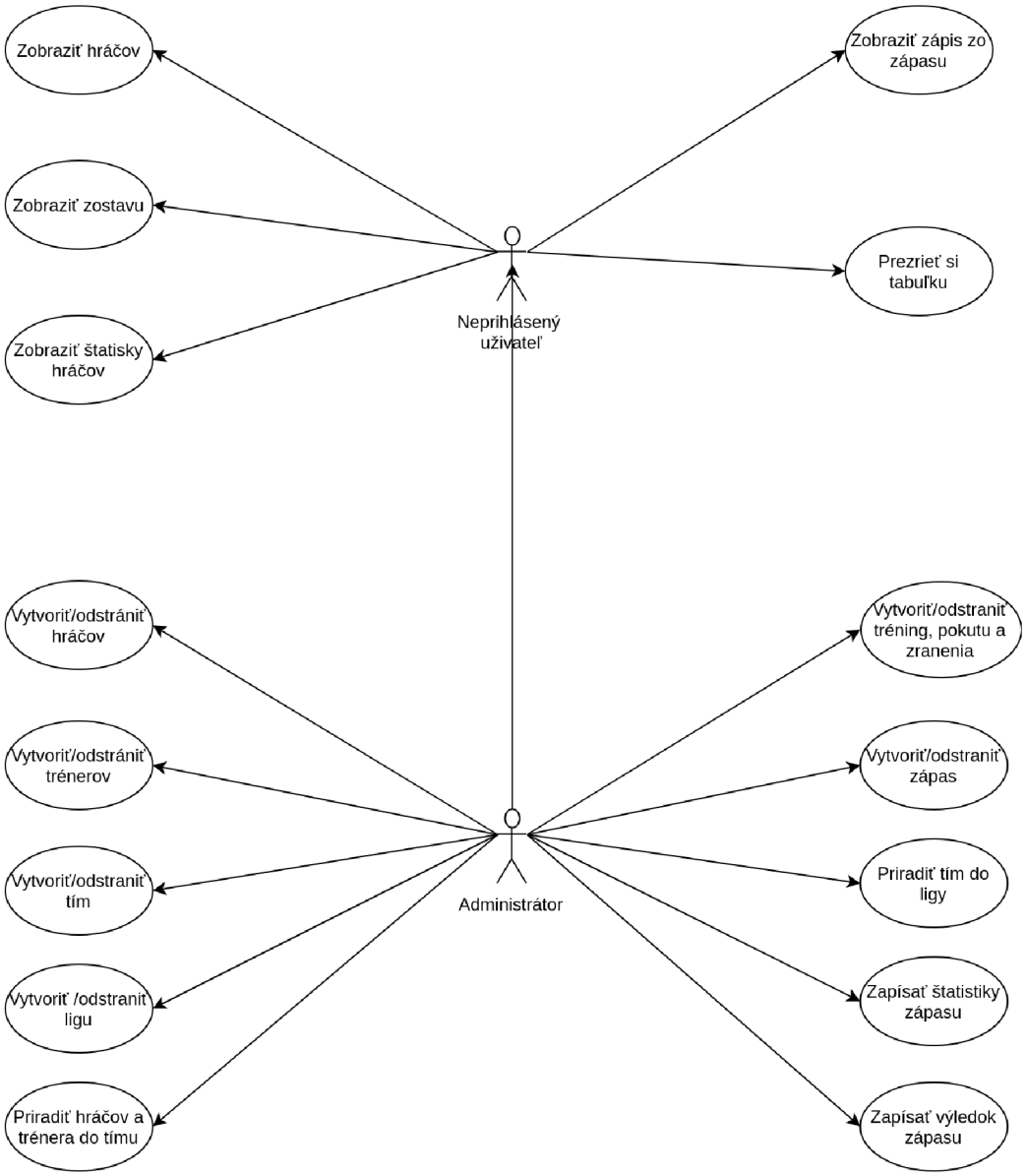
Ďalšou významnou úlohou administrátora bude tvorba zápasov. Výsledky jednotlivých zápasov bude zapisovať administrátor, alebo poverený zapisovateľ v danej lige. V položke zápas bude môcť administrátor zapisovať strelcov gólov a striedanie oboch tímov či náhradníkov, ktorí boli na zápase.

Tímy

Neodmysliteľnou súčasťou futbalového portálu sú tímy. Vytvorené budú pred začiatkom prvej sezóny. Následne sa už budú len pridávať konkrétni hráči, ktorí prestúpili do daného tímu. Počet hráčov v tíme bude závisieť od počtu registrovaných hráčov v tíme. V nižších ligách býva počet hráčov nižší a nepresahuje číslo 20.

Hráči a tréneri

Právomoc vytvoriť, pridať hráčov a trénera bude znova priradená administrátorovi. Ako sa už spomínalo v predchádzajúcej podkapitole, každý hráč bude mať informácie o tom ako sa volá, aký ma vek, na akej pozícii hraje, prípadne na viacerých, a v akom klube momentálne pôsobí. Prístupné budú aj informácie o jeho váhe, výške a emaily ak ho bude chcieť návštevník kontaktovať. Všetky tieto informácie budú prístupné na kartičke hráča. Tréner sa po vytvorení okamžite priradí k tímu, ktorý trénuje.



Obr. 5.1: Diagram prípadu použitia administrátora.

5.5 Tréner

Ďalším užívateľom portálu bude tréner. Právomoc prihlasovať sa ako tréner bude mať hlavný manažér, prípadne asistent tímu. Po administrátorovi bude mať druhé najväčšie kompetencie. Jeho možnosti budú ohľadom tvorby zostavy, formácie, riadenia tréningov, zadávania pokút a zapisovania zranení.

Tréning

Systém bude umožňovať vytvárať tréningy. Ich vytváranie bude mať na starosti tréner. Tvorba tréningu bude pozostávať zo dňa a času konania. Taktiež bude potrebné zadať dĺžku tréningu. Trénerovi bude poskytnutá možnosť špecifikovať tréning. Bude mať možnosť

zadefinovať či sa jedná o kondičný, taktický, alebo klasický tréning. Okrem špecifikácie tréningu bude môcť zadať, aká bola náplň tréningu. Ak bol kondičný, ako dlho behali, aké vzdialenosti a podobne. Tréningy sa budú vytvárať každý týždeň počas sezóny. Tréner bude mať právomoc po tréningu zadať počet účastníkov na tréningu.

Tvorba zostavy a formácie

Po každom zápase tímu bude potrebné vyplniť informácie o tom, kto hral v zápase a koľko bodov nazbieral. Táto právomoc bude udelená administrátorovi ako aj trénerovi mužstva. Tréner určí jedenásť hráčov, ktorí hrali v základnej zostave a piatich náhradníkov, ktorí prípadne naskočili až do neskoršej časti zápasu. Okrem výberu zostavy, ktorá nastúpila mu bude umožnené vyplniť štatistiky. Taktiež určí, ktorí hráči striedali.

Pokuty a zranenia

V prípade výnimočnej situácie bude trénerovi udelená právomoc zadať pokutu hráčovi. Takéto situácie nastanú, ak hráč hrubo poruší pravidlá tímu, vo väčšine prípadov pôjde o vynechanie tréningov bez dopredu avizovanej neúčasti. Systém umožní trénerovi zadať akúkoľvek sumu. V prípade dokázania neviny, bude možne pokutu odstrániť.

Ak sa počas tréningu, alebo zápasu zraní hráč, tréner bude povinný to zapísať. Určovať sa bude typ zranenia, teda oblasť a dĺžka očakávanej absencie.

Obr. 5.2: Diagram prípadu použitia trénera.

5.6 Hráč a neprihlásený užívateľ

Poslednými dvoma užívateľmi systému budú hráč a neprihlásený užívateľ. Hráč nebude mať veľa možnosti zasahovania do systému. Zmeny bude môcť prevádzať iba pri potvrdení účasti, alebo zrušení účasti na tréningu. Po potvrdení účasti sa jeho meno objaví v kolónke hráčov, ktorí prídu na tréning. Ak sa na tréningu objaví aj fyzicky, tréner to v systéme potvrdí.

Pod pojmom neprihlásený užívateľ, sa myslí hocijaký užívateľ, ktorý navštívi futbalový portál, no nemôže sa do portálu prihlásiť. Jeho právomoci sú veľmi obmedzené a pozostávajú iba z možnosti prehliadania portálu a zobrazovania si jednotlivých hráčov, alebo jednotlivých štatistík tímov a hráčov. Všetky právomoci neprihláseného užívateľa preberajú aj ďalší užívatelia systému.

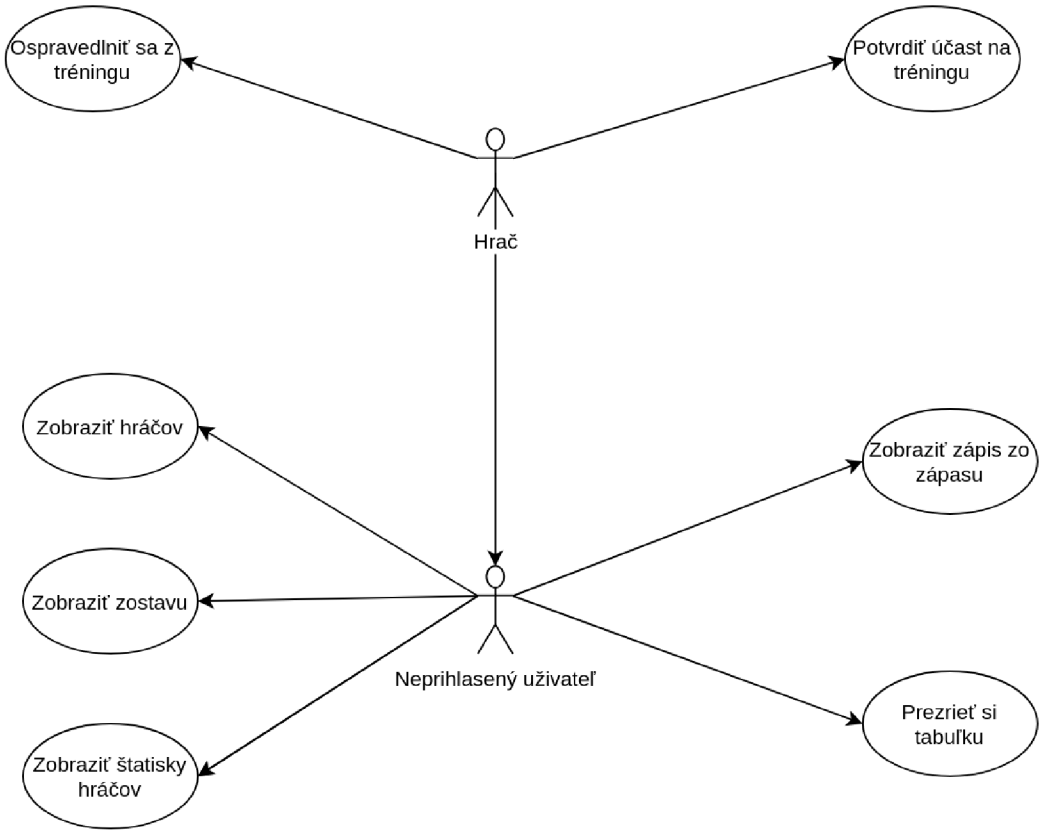
Zobraziť hráčov a štatistiky

Každý návštevník ma možnosť kliknúť na vybraného hráča. Po kliknutí na jeho profil sa mu objavia všetky dostupné informácie o hráčovi. Či už pôjde o jeho pozíciu, alebo vek. Dostupná bude aj tabuľka celkových štatistik hráčov v sezóne. Štatistiky budú hovoriť

o najväčšom počte strelených gólov, najviac vytvorených šancí, alebo o najväčšom počte odohraných minút.

Zobraziť zostavy, formácie a tabuľky

Okrem prezerania konkrétnych hráčov, bude možnosť prezrieť si aj konkrétne zostavy z vybraného zápasu. Užívateľ tam uvidí kto dal koľko gólov, kto kedy striedal, a ako vyzerala celá formácia družstva. Ako stojí jeho obľúbené mužstvo bude možné ľahko vyhľadať kliknutím na aktuálnu tabuľku sezóny. Tabuľka bude obsahovať všetky štatistiky doposiaľ prebiehajúcej sezóny. Počet gólov mužstva, počet inkasovaných gólov a celkový počet nazbieraných bodov.



Obr. 5.3: Diagram prípadu použitia hráča.

Zvolené prípady použitia

Niektoré prípady použitia sú pochopiteľné už z diagramu. Tie zložitejšie budú však podrobnejšie vysvetlené v nasledujúcej časti.

Prvým prípadom použitia bude vysvetlená trénerská právomoc vytvoriť tréning. V ďalšom prípade použitia bude predvedený proces vytvárania zranenia zo strany trénera. Na podobnom princípe funguje aj možnosť vytvorenia pokuty. V treťom prípade bude predstavená možnosť vytvárania súpisiek tímov a s tým spojené zapisovanie štatistík. V štvrtom prípade bude predvedená časť, ktorú obsluhuje hráč. Konkrétne pôjde o prípad potvrdzovania účasti na tréningu.

Identifikátor	UC01	
Prípad použitia	Vytvorenie tréningu	
Vstupná podmienka	Užívateľ je prihlásený ako tréner.	
Výstupný stav	Do databázy bude pridaný tréning.	
Užívatelia	Tréner	
Postupnosť krokov	**Krok**	**Činnosť**
	1	Prípad začne voľbou “Vytvoriť Tréning”.
	2	Tréner určí začiatok tréningu.
	3	Tréner určí dátum konania tréningu.
	4	Tréner určí zameranie tréningu.
	5	Tréner určí dĺžku konania tréningu.
	6	Systém uloží informácie o vytvorenom tréningu do databázy.
Výnimky	**Krok**	**Činnosť**
	1.a	Tréner môže kedykoľvek prerušiť činnosť vytvárania tréningu.
	3.a	V daný dátum už je vytvorený tréning.
	6.a	Tréner nevyplnil všetky položky formulára. Systém upozorní trénera.

Tabuľka 5.1: Vytvorenie tréningu pre mužstvo.

Identifikátor	UC02	
Prípad použitia	Vytvorenie zranenia	
Vstupná podmienka	Užívateľ je prihlásený ako tréner.	
Výstupný stav	Do databázy bude pridaný zranený hráč.	
Užívatelia	Tréner	
Postupnosť krokov	**Krok**	**Činnosť**
	1	Prípad začne voľbou “Vytvoriť Zranenie”.
	2	Tréner vyberie z tímu zraneného hráča.
	3	Tréner určí typ zranenia.
	4	Tréner určí dĺžku absencie.
	5	Systém uloží informácie o zranenom hráčovi do databázy.
Výnimky	**Krok**	**Činnosť**
	1.a	Tréner môže kedykoľvek prerušiť činnosť vytvárania zranenia.

Tabuľka 5.2: Zaevidovanie zraneného hráča.

Identifikátor	UC03	
Prípad použitia	Vytvorenie zostavy a zápis štatistík	
Vstupná podmienka	Užívateľ je prihlásený ako tréner.	
Výstupný stav	Do databázy bude priradené štatistiky hráčov.	
Užívatelia	Tréner	
Postupnosť krokov	**Krok**	**Činnosť**
	1	Prípad začne voľbou “Editácia súpisky”.
	2	Tréner vyberie svoj tím.
	3	Tréner určí jedenásť hráčov a päť náhradníkov.
	4	Tréner zapíše jednotlivé štatistiky každému hráčovi.
	5	Systém uloží informácie o štatistikách a zostave do databázy.
Výnimky	**Krok**	**Činnosť**
	1.a	Tréner môže kedykoľvek prerušiť činnosť vytvárania súpisky.
	3.a	Tréner vyberie toho istého hráča dvakrát. Systém ho upozorní.
	4.a	Tréner nevyplní všetky štatistické formuláre. Systém ho upozorní.

Tabuľka 5.3: Vytvorenie súpisky tímu a zapísania štatistík hráčov.

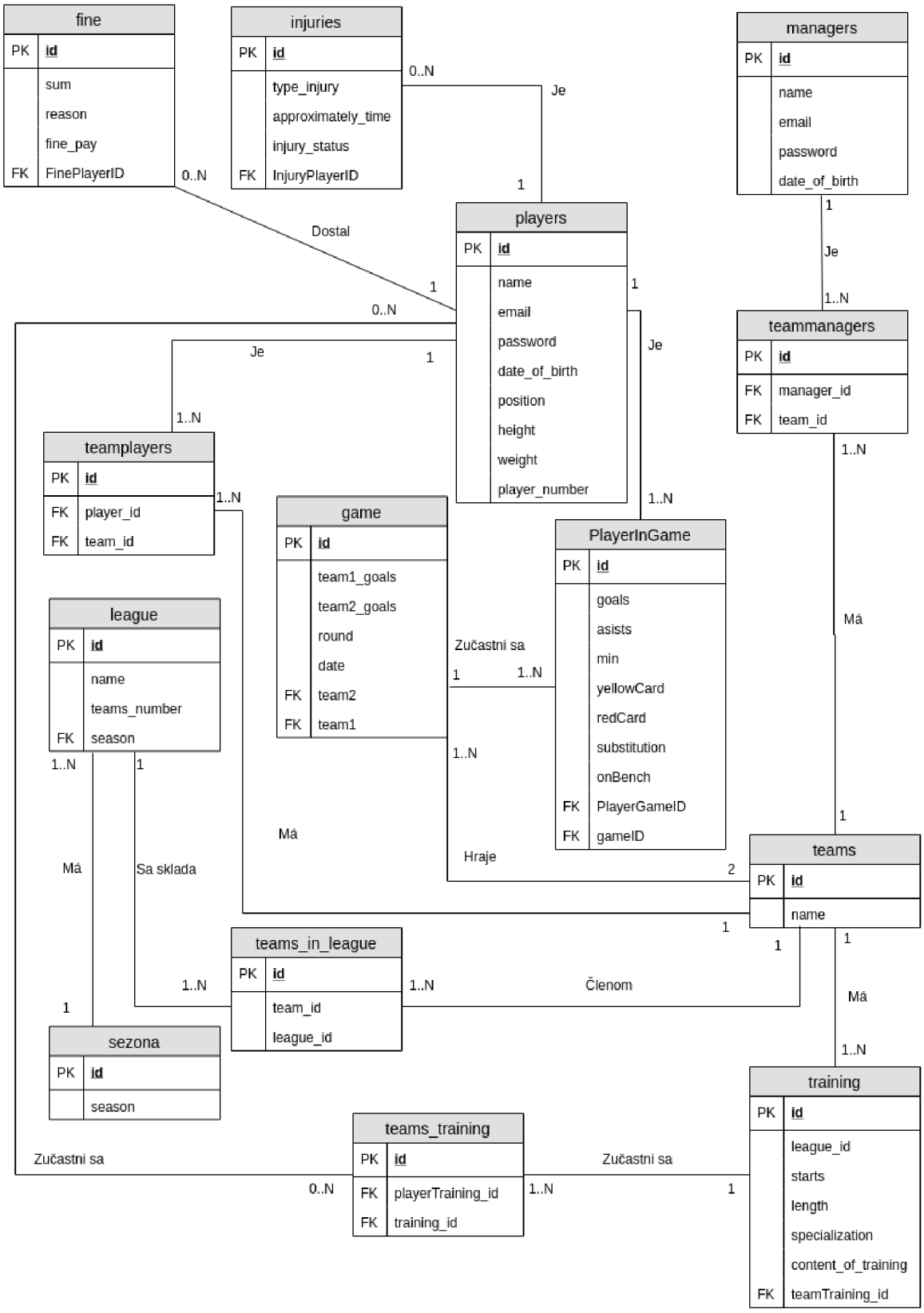
Identifikátor	UC04	
Prípad použitia	Potvrdenie účasti na tréningu	
Vstupná podmienka	Užívateľ je prihlásený ako hráč.	
Výstupný stav	Hráč je v databáze priradený k vybranému tréningu.	
Užívatelia	Hráč	
Postupnosť krokov	**Krok**	**Činnosť**
	1	Prípad začne voľbou "Tréning tímu”.
	2	Hráč klikne na tlačidlo "Zúčastnený".
	3	Hráč je priradený k tréningu. Čas tréningu zmení farbu na zelenú.
Výnimky	**Krok**	**Činnosť**
	1.a	Hráč môže kedykoľvek prerušiť činnosť prihlasovania na tréning.
	2.a	Hráč už je prihlásený. Systém mu umožňuje kliknúť iba na tlačidlo Ödmietnuť".

Tabuľka 5.4: Prihlásenie na tréning.

Kapitola 6

Návrh

Táto kapitola sa bude venovať návrhu informačného systému. V prechádzajúcej kapitole boli ustanovené požiadavky na vytváraný systém na základe ktorých môžeme vytvoriť ER diagram, ktorý je nepostrádateľný pre návrh. Ten uľahčí ďalšiu prácu a pomôže pri implementácií zadania. Po ER diagrame budú následne popísané a vysvetlené všetky entity, ktoré tento diagram obsahuje.



Obr. 6.1: ER diagram.

6.1 Entity

V nasledujúcich podkapitolách budú vypísané všetky entity a popis ich činnosti. Názov entít odpovedá presným názvom v databáze, aby neprišlo k nesprávnemu pochopeniu modelu.

players

Entita players popisuje hráča systému a klubu, ktorý je zároveň užívateľom systému. Primárny kľúčom je jeho id. Ďalej popisuje informácie ako je jeho meno, email s ktorým sa bude prihlasovať, zašifrované heslo, jeho vek, váhu, výšku a pozícia v klube. Nebude dochádzať k žiadnej zmene, až vo veci týkajúcej sa veku. Ten sa bude automaticky meniť každý rok.

teamplayers

Teamplayers predstavuje spojovaciu entitu pre tabuľky players a teams. Obsahuje ich primárne kľúče. Je potrebná k uloženiu informácií o tom, ktorý hráč v akom klube hraje. Tabuľka je nutná z dôvodu, že hráči môžu meniť kluby. Bolo by veľmi nepraktické priraďovať hráčovi klub priamo v jeho tabuľke.

teams

Entita predstavujúca tímy. Obsahuje len jeden podstatný atribút a to názov klubu. Predstavuje reálne tímy v lige.

managers

Podobná entita ako players. Narozdiel od nej, ale predstavuje všetkých reálnych manažérov v klube a v lige. Zahŕňa dôležité informácie ako je meno manažéra, heslo, email s ktorým sa dostane do systému a jeho skutočný vek.

teammanagers

Spojovacia tabuľka pre manažérov a tímy. Uchováva ich kľúče vďaka ktorým je umožnené v databáze spojiť manažéra s jeho tímom. Dôvod pomocnej tabuľky je rovnaký ako v prípade hráča.

fine

Hráči môžu za porušenie pravidiel klubu dostávať pokuty. K tomu aby sa vedelo, ktorý hráč dostal pokutu, je nutné ukladať jeho id ako cudzí kľúč. Taktiež je potrebné vedieť sumu, ktorú musí klubu vyplatiť a dôvod jeho prehrešku, aby sa spätne vedelo čo hráč porušil.

injuries

Hráči sa bežne môžu zraniť a vypadnúť z klubu na dlhšiu dobu. Táto entita teda predstavuje zranenia, kde uchováva informácie o tom, aký je to typ zranení, a na ako dlho s ním manažér a mužstvo nemôže rátať. V neposlednej rade musí ukladať aj jeho id ako cudzí kľúč, aby bolo možne priradiť hráča konkrétnemu zraneniu.

league a sezona

Entita league ukazuje na tabuľku liga. Tá bude obsahovať svoj názov ligy, keďže každá liga môže mať rozdielne meno. Taktiež je dôležité určiť koľko tímov môže obsahovať. Cudzí kľúč season sa bude vzťahovať na tabuľku sezóna.

Entita sezóna obsahuje len jeden atribút, ktorý rozhoduje o akú sezónu sa jedna. Teda určuje ročník vo formáte 2018/2019, alebo menej.

teams_in_league

Tímy potrebujú priradiť v ktorej lige hrajú. Keďže účasť v jednotlivej lige nemusí byť stála, je nutná pomocná tabuľka. Na to slúži pomocná tabuľka. Teamid a leagueid sú cudzie kľúče, ktoré priraďujú jednotlivé tímy k ligám.

game

Zápas obsahuje veľké množstvo atribútov, ktoré je potrebné ukladať do databázy. Je spojený s tabuľkou teams, pretože je potrebné vedieť, ktoré mužstvá tie zápasy hrali. To je riešené cez dva cudzie kľúče team1 a team2. Aby však bolo jasné, ktoré mužstvo vyhralo, sú nevyhnutné informácie o počte gólov. Atribúty team1goals a team2goals predstavujú nie len strelené góly, ale je z nich aj určený počet inkasovaných gólov. Každá sezóna má niekoľko kôl, kde tímy hrajú dva zápasy jeden doma a jeden vonku. Na to slúži atribút round. Pomocou neho sa vie užívateľ zorientovať o ktoré kolo sa hraje, koľko ešte kôl ostáva, prípadne, koľko už bolo odohraných.

PlayerInGame

Zatiaľ sú uchovávané informácie o hráčovi iba obecne. Každý hráč však odohráva zápasy za klub v ktorom nazbiera určité informácie. Tieto informácie sú potom potrebné pre štatistické tabuľky o hráčovi. Preto je nutná tabuľka popisujúca hráča v zápase. K tomu sú nutné dva cudzie kľúče. Hráčove id, aby sa vedelo o ktorého hráča ide a zápasová id, aby bolo jasné v ktorom zápase hral. Hráč si môže v zápase pripísať gól, asistenciu, žltú, alebo červenú kartu. V zápase odohraje aj určitý počet minút. Všetky tieto údaje sú zbierané a na základe toho vytvorená tabuľka štatistík o jednotlivých hráčoch.

training

Tréning je vytváraný pre každý tím samostatne. Preto je potrebné vedieť id tímu, aby ho bolo možné priradiť tréningu. Aby hráči vedeli kedy sa majú tréningu zúčastniť, je nutné uvádzať začiatok tréningu. Povinnosťou je udať aj jeho dĺžku, ktorá však nemusí byt finálna, keďže dĺžka tréningu sa môže z nejakých dôvodov predĺžiť. Súčasťou entity je aj konkrétna špecializácia tréningu a jeho náplň.

teams_training

Pomocná tabuľka pre tréning. Hráči majú možnosť sa tréningu zúčastniť, alebo nezúčastniť. Ak potvrdí účasť ich id je priradené tréningu a pomocou SQL príkazov tak tréner vidí, kto mu na tréning príde. Pomocný stĺpec training_parcipitation, bude slúžiť pre trénera. Ak sa hráč dostaví na tréning, tréner to potvrdí a do stĺpca sa priradí hodnota jeden, označujúca hráčovu fyzickú prítomnosť na tréningu.

Kapitola 7

Implementácia

V tejto kapitole bude popísaná konkrétna implementácia informačného systému. K jej realizácií sa pristupuje na základe informácií získaných z predchádzajúcich kapitol. Pri tvorbe sa využili všetky technológie popísane v kapitole popisujúcej použité technológie. Konkrétne sa jedná o Laravel, PHP, MySQL, Javascript, HTML alebo CSS. To ako bude fungovať systém, a aké role majú jednotlivý užívatelia systému nám popísali diagramy prípadov použitia z kapitoly 5. Databáza bola vytvorená na základe Entity Relationship Diagramu.

7.1 XAMPP server

Prvú vec, ktorú bolo potrebné urobiť, bola nutnosť inštalácie lokálneho serveru na ktorom by bežala webová stránka a na ktorej by sa dalo pracovať na jeho implementácií. XAMPP je voľne dostupný softvér vytvorený spoločnosťou Apache friends. XAMPP aplikácia obsahuje Apache server, MariaDB, PHP a Perl. Ide teda o aplikáciu, ktorá vytvára lokálny server a poskytuje všetky potrebné komponenty pre túto prácu. Jej inštalácia je veľmi jednoduchá. Najprv bol stiahnutý súbor z oficiálnej stránky Apache friends [22] a následne nainštalovaný na lokálny počítač. Po spustení programu sa počítač správa ako lokálny server a je možné na ňom vytvárať webovú stránku.

7.2 Inštalácia Laravelu

Jednou z ďalších vecí, ktorá bola vykonaná, bola inštalácia Laravelu, keďže sa jednalo o primárny framework a nástroj s ktorým bola vytvorená táto bakalárska práca. Spôsobov ako si zaobstarať Laravel framework je viacero [27]. Prvým spôsobom je pomocou Laravel inštalátora zadaním príkazu `composer global require laravel/installer`. Ďalšou alternatívnou možnosťou jej využitia composeru. Príkaz `composer create-project -prefer-dist laravel/laravel blog` nám vytvorí Laravel do danej zložky, ktorá bola určená. Ak je však PHP nainštalované lokálne stačí použiť zabudovaný PHP server, ktorý poslúži aplikácií pomocou príkazu `php artisan serve`. Tento príkaz spustí server na lokálnej adrese `http://localhost:8000:`. V našom prípade sa využila druhá možnosť, kde bol použitý Composer o ktorom bolo povedané viac už v tretej kapitole tejto práce. Projekt však musel byť nasmerovaný do zložky obsahujúcej nainštalovaný XAMPP, aby s ním mohol Laravel spolupracovať.

7.3 Databáza

Ako prvé bolo potrebné analyzovať celý systém a na základe toho si definovať, ako má vyzerať databáza. Preto bol vytvorený ER diagram, ktorý nám definoval jednotlivé tabuľky, atribúty a ich jednotlivé vzťahy. K práci s databázou sa využíval aj voľne dostupný softvér phpMyAdmin napísaný v php jazyku. Keďže poskytuje užívateľovi priehľadné užívateľské rozhranie, bol vhodný na malé úpravy ako mazanie nechcených dát v databáze, či kontrolu tabuliek.

Pre prácu s databázou priamo v kóde slúži Query Builder, ktorý je zabudovaný priamo v Laravel frameworku. Poskytuje možnosť tvorby rôznych SQL dotazov ako sú SELECT, JOIN, alebo UNIONS. Obrázok 7.1 znázorňuje SQL dotaz typu SELECT a JOIN spojením dvoch tabuliek, kde sú vybrané atribúty z tabuľky teams na základe zhody id užívateľa, ktorý je prihlásený v systéme a id hráča z tabuľky teamplayers.

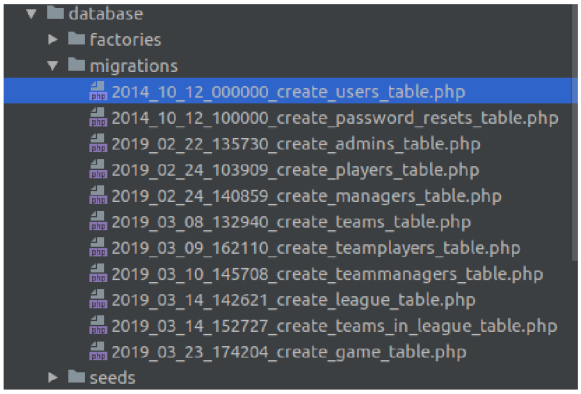
```
$teams = DB::table( table: 'teams')
    ->select( columns: 'teams.*')
    ->join( table: 'teamplayers', first: 'teams.id', operator: '=', second: 'teamplayers.team_id')
    ->where( column: 'teamplayers.player_id', operator: '=', $user)
    ->first();
```

Obr. 7.1: SQl dotazy typu SELECT a JOIN.

Migrácia

Migrácia slúži na ovládanie databázy. Umožňuje tvorbu nových tabuliek a ich modifikáciu. Príkaz `php artisan make:migration create_users_table` nám vytvorí tabuľku so zadaným názvom. Následne sa výplní údajmi ako primárny kľúč a atribúty. Spustením php artisan migrate začneme migráciu, kde sa všetky vytvorené tabuľky načítajú do databázy a následne ich môžeme upravovať pomocou vyššie spomínaného phpMyAdmin.

Na nasledujúcom obrázku 7.2 je možné vidieť jednotlivé tabuľky vytvorené pre potrebu databázy.



Obr. 7.2: Príklad všetkých tabuliek vytvorených pre databázu.

7.4 Routing

Routy v Laraveli slúžia k nasmerovaniu všetkých požiadaviek aplikácie na požadujúci controller. Routy môžu mať viacero metód, no v tejto práci sa využívajú iba GET a POST. GET v prípade, ak sa majú zobraziť vybrané informácie o hráčovi na danom pohľade. Ak

sa však administrátor vytvára nového hráča, použije sa metóda POST, aby sa vložili požadované informácie do databázy. Súbor s routami je na začiatku automaticky vygenerovaný a uložený v súbore `web.php`. Ako je možné vidieť na obrázku 7.3, každý rout sa presmeruje pomocou URL na daný controller a jeho funkciu. V práci je využívaná aj možnosť pomenovania routu. Ak je potrebné vo frontend časti zavolať rout pre vykonanie POST metódy, tak sa použije iba jeho skrátene pomenovanie.

```
Route::prefix('admin')->group(function() {
    Route::get('/login', 'Auth\AdminLoginController@showLoginForm')->name('admin.login'); // name sme si urcili prezyvku
    Route::post('/login', 'Auth\AdminLoginController@login')->name('admin.login.submit');
    Route::get('/', 'AdminController@index')->name('admin.dashboard');
    //HRAC
    Route::get('/admin_update_player/{id}', 'AdminController@editPlayer')->name('admin.player.update');
    Route::post('/admin_update_player/{id}', 'AdminController@updatePlayer')->name('admin.player.updatePlayer');
    Route::get('/admin_list_player', 'AdminController@playersList')->name('admin.players.list');
    Route::get('/admin_insert_player', 'AdminController@newPlayer')->name('admin.player.insert');
    Route::post('/admin_insert_player', 'AdminController@insertPlayer')->name('admin.player.insertPlayer');
```

Obr. 7.3: Názorná ukážka routov pre admina.

7.5 Controllers

V MVC architektúre písmeno C stoji za významom controller. Chovanie controlleru sa definuje ako prostredník medzi pohľadmi a modelmi. Úlohou controlleru je riadiť celý systém a prepojiť komponenty jeho systému. Miesto definovania si metód a funkcii priamo v route, je možné vytvorením controllerov pre každú skupinu požiadaviek. Daný controller obsahuje triedu a v nej funkcie, ktoré sú volané v prípade ich potreby. Volané sú na základe URL, ktorá je zadaná užívateľom, musí však byť rovnaká ako URL príslušnej route. Funkcie môžu byť však aj volané na základe skratky funkcie uvedenej pri vytvorenej rout. Najviac práce má v tomto systéme AdminController, ktorý obstaráva tzv. CRUD (Create, Read, Update, Delete). Jeho úlohou je vytváranie, mazania či aktualizovanie administratívnych vecí. Na ďalšom obrázku 7.4, je vidieť úryvok z kódu AdminControlleru, ktorý sa volá v prípade, ak chce administrátor vytvoriť nového hráča.

```
public function newPlayer()
{
    $teams = DB::table( table: 'teams')->get();
    $data = [
        'teams' => $teams,
    ];
    return view( view: 'admin.admin_insert_player', $data);
}
```

Obr. 7.4: Funkcia vytvárajúca nového hráča.

7.6 Autentifikácia užívateľa

Autentifikácia je proces, v ktorom sa identifikujú užívateľské prihlasovacie údaje, ako je prihlasovacie meno a heslo. Ak sa tieto údaje zhodujú, užívateľ je identifikovaný a prihlásený do systému. Laravel umožňuje programátorovi vytvoriť autentifikáciu veľmi jednoducho. Pomocou príkazu `php artisan make:auth` sa nainštalujú a vytvoria prihlasovacie a registračné pohľady a routy pre autentifikáciu.

Vytváranie a prihlasovanie užívateľa

Užívateľské účty vytvára administrátor. Ručná registrácia a neskoršie prihlásenie nebude v systéme umožnené. Návštevníci, ktorí nie sú súčasťou klubu majú možnosť prezerať si web a štatistiky jednotlivých klubov, no nie je im umožnený prístup do systému. Administrátor vytvára hráčov a trénerov kde im vyplňuje informácie potrebné k ich fungovaniu. Tieto informáciu sú ukladané do databázy. Heslo sa však neukladá ako jednoduchý text. Ide o typ hash, aby v prípade úniku dát z databázy nedošlo k ich zneužitiu.

Pri prihlasovaní sa jednotlivo zadávajú informácie. Užívateľský email a heslo. Ak sa obidva zadané informácie zhodujú s informáciami v databáze, užívateľ ma povolený prístup do systému. V inom prípade je vyzvaný systémom k zadaniu správneho mena, alebo hesla.

Obr. 7.5: Prihlasovacie okno pre hráča.

7.7 Administrátorský panel

Panel pre administrátora bola jednou z prvých vecí, ktorá bola implementovaná z dôvodu jej dôležitosti v rámci projektu. Administrátor predstavuje hlavný bod od ktorého sa odvíjajú ďalšie časti projektu. Pri tvorbe bol použitý pojem, ktorý sa pri programovaní volá CRUD. Jeho úlohou je poskytnúť administrátorovi štyri základné funkcionality produktu a to možnosť vytvoriť, čítať, aktualizovať a vymazať. Administrátor má kontrolu nad celým systémom. Keďže sa tu využíva CRUD, jeho úlohou je vytvárať všetky entity, ktoré sa zobrazia na hlavnej stránke ku ktorej ma prístup tréner a hráč. Môže vytvoriť hráčov, manažérov, tímy ligy, zápasy a tréningy. Taktiež ich môže vymazať, alebo aktualizovať

ak nastane napr. zmena v hráčovej pozícií, alebo klube. Tréner preberá právomoc tvorby tréningov.

Pri tvorbe frontendu sa použil voľne dostupný layout pomenovaný Matrix Admin Template. Urýchlilo to implementáciu a poskytoval všetky dôležité prvky pre panel, taktiež sa dal veľmi ľahko implementovať do projektu.

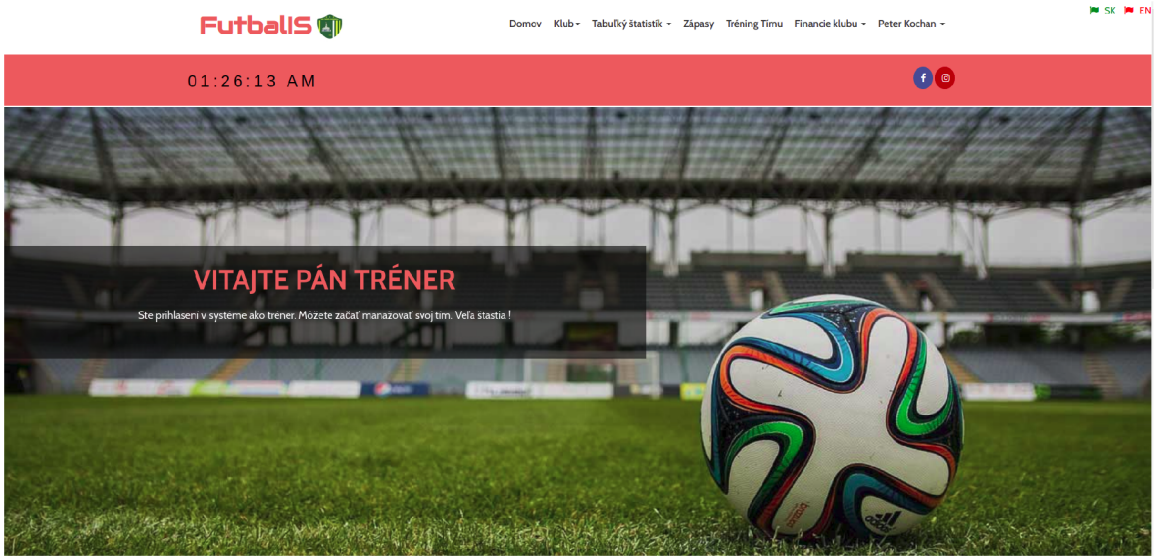
Ako už bolo spomenuté v sekcii Controllers, najviac práce vykonáva AdminController. Obsluhuje celú časť sekcie pre administrátora. Každá časť ako je hráč, alebo tím, pozostáva vždy zo štyroch funkcií, ktoré obsahujú SQL dotazy na základe ktorých sú vyťahované potrebné dáta. Pred načítaním pohľadu s dátami je potrebné najprv pomocou SQL dotazu vytiahnuť dáta z tabuľky, ktorú potrebujeme pre danú činnosť. Takéto funkcie sú väčšinou v kóde pomenované ako napr. `NewPlayer`. Insert (vkladanie) nových dát pomocou formulára do databázy pracuje na opačnom princípe. Po načítaní stránky sa administrátorovi zobrazia formuláre, ktoré sú vyplnené administrátorom. Informácie z formulára sú predané do databázy pomocou funkcie, ktorá ich priraďuje náležitým stĺpcom tabuľky. Pri vyplnení formulára je kontrolované či pole formulára nezostalo prázdne, teda NULL, ináč vyskočí chybová hláška a musí sa vyplniť. K všetkým vytvorením záznamom je prístup cez zoznam, kde je možnosť editovania. Pri editovaní sa predáva id získane z pohľadu, aby sa do databázy mohol poslať dotaz v rámci ktorého sa budú hľadať také záznamy, ktoré spĺňajú požiadavku vybraného id. Ak sa edituje jednotlivý hráč, jeho id sa predá funkcii, ktorá obsahuje SQL dotaz na funkciu, kde sa pomocou where klauzuly porovnávajú id a to, ktoré sa zhoduje s predaním id sa vyberie.

7.8 Užívateľská strana a formuláre

Po vytvorení administračnej časti bola vytvorená hlavná užívateľská časť, ktorú využívajú neprihlásení užívatelia, manažér a hráč. K lepšiemu vzhľadu bol podobne ako u administrátora využitý dostupný layout zo stránky w3layouts[1]. Využitá bola iba hlavná kostra, inač bol prerobený k potrebám vytváraného systému.

Informačný systém je lokalizovaný do dvoch jazykov. Základný jazyk je slovenský, keďže systém bude využívať slovenský tím. Pridaná je aj možnosť využívania systému v anglickom jazyku. Prepnúť si na konkrétny jazyk je možné v pravom hornom rohu. Na následujúcom obrázku 7.6, je možné vidieť časť úvodnej obrazovky trénera po načítaní stránky. Vzhľad celej stránky je organizovaný pomocou už vyššie spomínanej knižnice Bootstrap.

[1]Dostupné na https://w3layouts.com/



Obr. 7.6: Úvodná stránka trénera po prihlásení do systému.

Zadávanie informácií

Hlavným princípom činnosti trénera v systéme je vytvárať a organizovať veci v klube. K tomu mu slúžia formuláre do ktorých môže vypĺňať relevantné informácie, ktoré sa následne ukladajú do tabuľky.

Vyplnenie tabuliek pomocou formulárov sa deje pomocou dvoch možných variácií formulárov. Buď je to formou dropdown formy, kde je možné si rozkliknúť formulár a vybrať požadovanú možnosť, alebo prostredníctvom klasického formulára. K vyberú slúžia len dáta, ktoré boli predom vybrané, napr. ak je zadávaný tréning pre hráčov daného klubu, je možné vybrať iba ich. Všetko je to riešené pomocou už veľa krát spomínaných SELECTOV a SQL. Všetky dáta sú vyberané z tabuľky na základe požiadavky. Pri dropdowne, ale dochádza k vybratiu vybraného ID z inej tabuľky, táto hodnota je následne priradená do požadovanej tabuľky kde sú zapísané hodnoty. Ako príklad je možné uviesť zápis zranenia. Zranenia sú uložené v tabuľke `injuries`. Zapísané má byť zranenie hráča z Hanisky. Do dropdown formuláru sú načítané všetky ID hráčov z tabuľky `players`. Manažér si vyberie hráča a jeho ID, potom sú vyplnené klasické formuláre z ktorých idú informácie priamo do tabuľky `injuries`. Vytiahnuté ID hráčov sa vo funkcii priradí ekvivalentnému atribútu v tabuľke `injuries` a následne vyplní. Formuláre sú zobrazované pod sebou kvôli praktickosti. Jediná výnimka nastáva pri administrátorovi a tvorbe zostavy. Uplatňuje sa tam varianta formulárov pod sebou, ale aj vedľa seba z dôvodu zadávania veľkého kvanta informácií. Všetky formuláre musia byť vyplnené a žiaden nesmie byť NULL. V opačnom prípade skočí chybová hláška upozorňujúca na nevyplnené políčko. Ošetrené je to pomocou IF, ktorý sa pýta či daná požiadavka na požadovaný input je NULL. Ak áno, bude vypísaná chyba.

Štatistiky hráčov

Dôležitou implementáciou sú jednotlivé štatistiky hráčov v danej lige. V zápase je možné nazbierať štatistiky, ktoré sa následne ukladajú do tabuľky. Slúžia ako výkonnostný prehľad pre trénerov a hráčov, aby vedeli zistiť koľko daný hráč dal gólov, alebo nazbieral kariet. Zbierané budu štatistiky o góloch, asistenciách, odohraných minút, červených, alebo žltých

kariet. Ide o najdôležitejšie údaje, ktoré je možné vo futbalovom zápase získať. K výberu štatistík bude užívateľ vyzvaný po pre kliknutí na stránku štatistiku hráčov. Tá bude obsahovať výber už vyššie spomenutých štatistík. Každá štatistika sa bude vyhodnocovať na základe počtu. V prípade gólov sa budú spočítavať všetky góly v zápasoch a následne z toho vznikne tabuľka, kde hráč na prvom mieste, bude hráč s najväčším počtom gólov. Ak sa nájdu hráči s rovnakým počtom gólov tak k radeniu v tabuľke sa pristúpi podľa abecedného poradia ich mien. V tabuľke štatistík sa budú objavovať hráči aj v prípade, že nestrelili žiaden gól. Podmienkou na zaradenie hráča bude odohranie aspoň jedného zápasu.

Vyťahovanie štatistických údajov sa rieši pomocou SQL príkazov. Používa sa už vyššie spomínaný Query Builder, vďaka ktorému je tvorba dotazov intuitívnejšia a rýchlejšia. Implementácia je definovaná pomocou piatich funkcií starajúcich sa o jednotlivú štatistiku. Pri každej dochádza ku komplikovanému dotazu obsahujúci nanajvýš štyri príkazy join. Veľký počet joinov je spôsobený obsiahlou databázou a nutnosťou prepojenia všetkých cudzích kľúčov. Hráči sa nachádzajú v tabuľke databázy `players`. Z nej sa vyťahujú informácie ako je meno, pozícia, alebo vek. K vytiahnutiu informácii o počte gólov a ďalších štatistík je však potrebné prepojiť tabuľku s ďalšou tabuľkou databázy a to s `PlayerInGame`. K prepojeniu dochádza pomocou porovnania id hráčov. Keď však systém obsahuje viac líg je potrebné riešiť vypísanie štatistik správnej ligy. Vtedy sa využívajú ďalšie joiny k prepojeniu ešte tabuliek `teamplayers` a `teams_in_league`. Následne sa použije príkaz where, kde sa porovná id požadovanej ligy s id, ktoré bolo predané ako parameter. Hráč však hraje rozličné zápasy v ktorých môže dať rôzny počet gólov. Preto je dôležité použiť klauzulu GROUP BY ktorá zoskupi vybrané záznamy tak, aby nedochádzalo k duplikovaniu záznamov. Góly a ďalšie štatistiky sú spočítané pomocou agregačnej funkcie SUM. Na vizuálnu časť tabuliek je využívaný framework Bootstrap a jeho časť zaoberajúca sa štylizáciou tabuliek. Výpis poradia hráčov v tabuľkách rieši krátky JavaScript kód.

Najlepší strelci v lige

0.Hráč	Vek	Pozícia	Góly
1.Štefan Lamy	17	Utocnik	2
2.Michal Vaško	23	Zaloznik	1
3.Kristián Krakovsky	16	Utocnik	1
4.Matúš Adamkovič	20	Zaloznik	1
5.Peter Hirko	30	Utocnik	0
6.Martin Basár	17	Zaloznik	0
7.Matúš Šingľar	19	Obranca	0
8.Tomas Brezansky	23	Brankar	0
9.Jakub Jozef Krakovský	24	Obranca	0
10.Juraj Kula	23	Zaloznik	0

Obr. 7.7: Tabuľka zobrazujúca štatistiky strelených gólov.

Poradie tímov v lige

Významnou časťou systému je implementácia jednotlivého poradie tímov v lige. Tímy v lige odohrávajú určitý počet zápasov na základe ktorých získavajú body. Za výhru sú tri body, za remízu jeden a za prehru nezískajú žiaden. Tieto body sú potom sčítané a prezentované do tabuľky. Tabuľka bude obsahovať tímy, ktoré hrajú v lige, počet získaných, strelených

a inkasovaných gólov. Poradie tímov zavisí na počte získaných bodov. V prípade rovnosti bodov sa poradie v tabuľke bude určovať na základe strelených gólov.

Implementácia ligovej tabuľky sa však líšila od implementácie štatistík hráčov. Keďže ukladať získane body z jednotlivých zápasov do databázy by bolo veľmi nepraktické, nedalo sa spoľahnúť len na SQL dotazy. K SQL sa sa pristúpilo v rámci vytiahnutia tímov z databázy. V prípade viacerých líg by to znova prišlo k porovnávaniu predaného parametru id ligy. Keďže zápas obsahuje dva tímy pomenované ako team1 a team2, nie je možné im pomocou jednoduchého dotazu priradiť príslušný tím a spočítať ich body. Riešené je to pomocou cyklusu foreach. Na začiatku sú nastavené počiatočne premenné hodnôt góly a body na nula. Cieľom je prejsť databázu game a postupne priraďovať jednotlivým tímom príslušne id. Následne sa tieto id porovnajú na základe príslušných gólov. Ak ma jeden tím viac gólov ako druhý sú mu priradené tri body. Ak majú rovnaký počet obidvaja dostanú po bode. V tomto cykle sú zároveň priraďované aj údaje odkazujúce na počet výhier, prehier a remíz.

Ligová tabuľka

O.Tím	Zápasy	Strelené góly	Obdržané góly	Výhry	Remízi	Prehry	Body
1.FK Furča Haniska	2	4	1	2	0	0	6
2.FK Kojatice	1	3	1	1	0	0	3
3.FK Javorina Krásna Lúka	2	1	2	0	1	1	1
4.FK Krivany	1	1	1	0	1	0	1
5.Ruská Nova Ves	1	1	3	0	0	1	0
6.ŠK Svinia	1	1	3	0	0	1	0

Obr. 7.8: Štatistika tímov v lige.

Súpiska tímu a zápis výsledku

Jednou z najdôležitejších vecí v systéme je zápis zápasu. Na to, aby mohla byť vytvorená tabuľka tímov je nutné mať výsledky zápasov. Zapisovanie zápasov je umožnené cez administrátorský panel. Jeho úlohou je vyplniť všetky podstatné informácie ohľadom zápasu. Či už ide o dátum, ligové kolo, strelené góly a o tímy, ktoré ten zápas hrali. Po vytvorení je všetko uložené do databázy a pripravené k spracovaniu.

Vytváranie súpisiek zo zápasu funguje na inom princípe. Je to prepojené s možnosťou určenia jednotlivých štatistík hráča v zápase. Administrátor si v paneli zobrazí zápasy, ktoré sa odohrali. Umožnená je možnosť kliknutia na id zápasu a následne na jednotlivý tím. Parameter id sa tu predáva, aby sa administrátor mohol dostať na panel tímu, ktorý má byť editovaný. Pri vytváraní súpisy je nutné vybrať jedenásť hráčov z daného klubu a piatich náhradníkov. Pri tvorbe súpisky je zakomponovaná aj nutnosť udať aké štatistiky hráč nazbieral. Náhradník a striedania sú vyriešené pomocou flagu. Pri hráčoch, ktorí striedali, alebo boli náhradníci sa určia atribúty substitution a OnBench hodnotou 1. Vďaka nim je potom možné pomocou SQL dotazu vybrať rozdielne hráčov základnej jedenástky a náhradníkov.

Výsledok zápasu a súpiska sú kvôli lepšej priehľadnosti rozdelené na dva samostatné stránky. Prvá stránka obsahuje výsledok zápasu, kolo, čas a strelené góly. Slúži ako rýchly sumár zápasového kola pre návštevníka stránky. Súpiska zápasu sa nachádza na ďalšej stránke na ktorú sa dá dostať kliknutím na id zápasu. Výpis hráčov je riešený cez štyri tabuľky. Tabuľky na ľavej strane obsahujú hráčov z domáceho tímu. Spodná tabuľka zo-

brazuje náhradníkov, ktorí sa zúčastnili zápasu. Jej výpis je riešený cez SQL dotaz kde sa pomocou príkazu where určuje, že aj prítomný atribút OnBech je rovný nula, tak sú tí hráči považovaní za náhradníkov. Tabuľky zobrazujú mena hráčov, ich pozíciu a ostatné štatistiky. Výpis zápasu sa je dostupný každému používateľovi systému.

Základna zostava Tím 1.

Hráč	Pozícia	Góly	Minúty	Striedal
Stanislav Durovčík	Brankar	0	90	
Jakub Vardžík	Obranca	0	90	
Vincent Korinok	Obranca	0	90	
Daniel Sedlák	Obranca	0	70	Áno
Ľubomír Vaňo	Zaloznik	0	90	
Matúš Adamkovič	Zaloznik	1	90	
Tomáš Vysocký	Zaloznik	0	90	
Vojtech Schwartz	Zaloznik	0	75	Áno
Martin Basár	Zaloznik	0	90	
Štefan Lamy	Utocnik	2	90	
Štefan Kočiš	Utocnik	0	60	Áno

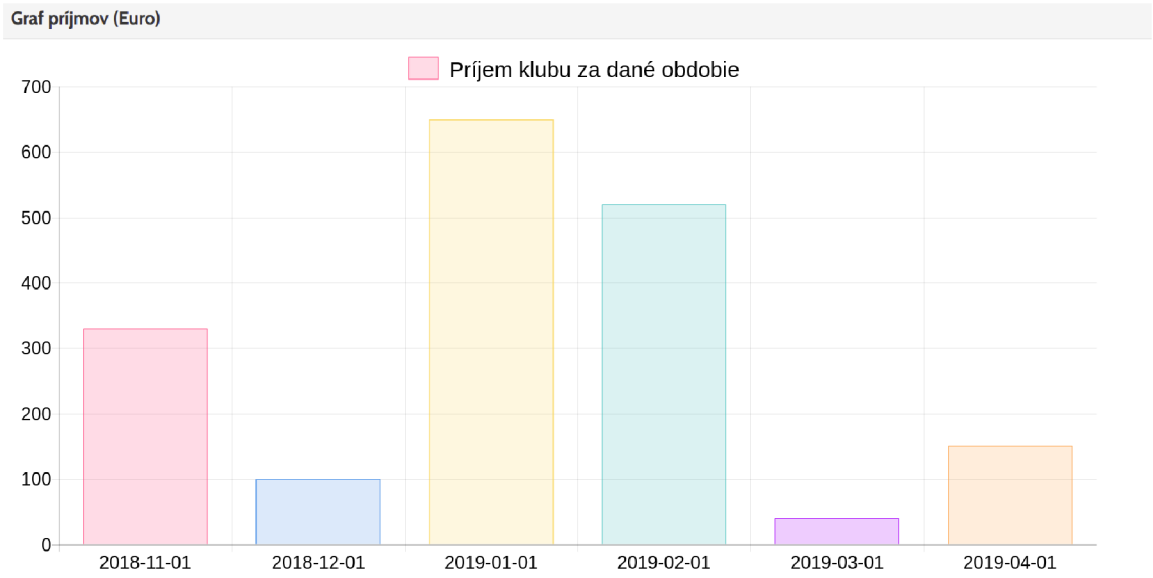
Nahradníci 1.

Hráč	Pozícia	Góly	Minúty	Striedal
Peter Fečo	Obranca	0	20	Áno
Marek Herkeľ	Obranca	0	0	
Lukáš Magda	Zaloznik	0	0	
Michal Vaško	Zaloznik	0	15	Áno
Jozef Vaško	Utocnik	0	30	Áno

Obr. 7.9: Zápis zápasu. Obrázok zobrazuje súpisku domáceho tímu.

Financie, tréning, pokuty a zranenia

Klub má počas roka príjmy a výdaje. Všetko zavisí od veľkého počtu faktorov ako je predaj vstupeniek, alebo platy hráčov. Finančná situácia sa mení každým mesiacom. Preto bola v systéme implementovaná možnosť vykresľovanie grafov z nadobudnutých dát. Viditeľné sú grafy príjmov a výdavkov. Na každom sa dá na určitom období porovnať narasť, alebo pokles financií. Implementované boli pomocou knižnice Chart JS a webovej techniky Ajax. Názorný príklad je na obrázku 7.10.



Obr. 7.10: Grafy zobrazujúce príjmy klubu za obdobie šiestich mesiacov.

Tréner má v systéme možnosť zadávať pokuty, zapisovať zranenia a vytvárať tréningy. Zranenia a pokuty sú implementované iba v trénerskom paneli. Tréningy sú viditeľne aj pre hráčov. Úlohou týchto častí je evidencia o hráčoch pre trénera. Ak dôjde k zraneniu hráča, tréner má možnosť zapísať jeho zranenie. Všetko funguje na princípe SQL dotazov. Tréner vyberie hráča, typ zranenia a dĺžku jeho absencie. Dáta sú následne uchované v databáze. Výhodou je, že dáta nie sú mazané a je možné ich nájsť aj po dlhšom čase. Klub si tak môže evidovať ako často bol hráč zranený. Nevýhodou by však bolo zobrazovať aj rok staré zranenia trénerovi. Problém je vyriešený pomocou pomocného stĺpca injury_status v databáze. Ten určuje či zranenie pretrváva, alebo sa jeho zranenie vyliečilo. Počiatočná hodnota stĺpca je nula. To znamená, že hráč je stále zranený. Trénerovi bude zobrazená tabuľka so zraneniami. Každé meno hráča je najprv vyznačené červenou farbou. Červená farba bude označovať, že hráč sa ešte nevyliečil. Každý hráč má v riadku dve tlačidlá. Pomocou kliknutia na zelené tlačidlo sa jeho hodnota v stĺpci zmení na jedna. To bude určovať, že hráč je vyliečený a dôjde aj k zmene farby jeho mena kvôli priehľadnosti pre užívateľa. Druhé tlačidlo slúži ako pomoc, ak by prišlo k nechcenému kliknutie na zelené tlačidlo. Umožňuje tak trénerovi vrátiť hodnotu pomocného stĺpca spať na pôvodnú hodnotu zranenia. Zároveň príde aj k zmene farby na pôvodnú červenú farbu. Ako to vyzerá v systéme je možné vidieť na obrázku 7.11. Na podobnom princípe fungujú aj pokuty a tréning. V prípade zaplatenia pokuty, je trénerom stisnuté tlačidlo, ktoré určí, že hráč si splnil svoje záväzky voči klubu. Pri tréningoch tréner môže vidieť kto potvrdil účasť. Vo viacerých prípadoch sa však stáva, že sa hráč tréningu nezúčastní. Tréner tak znova pomocou tlačidiel určí kto tam bol a kto nie.

Obr. 7.11: Na obrázku je viditeľná tabuľka zranených hráčov.

Kapitola 8

Testovanie a potencionálny vývoj

Táto kapitola sa bude zaoberať testovaním popisu jednotlivých testovacích subjektov a možnosťami ďalšieho vývoja. Dôležitou súčasťou vývoja softwerových aplikácií je testovanie. Po vytvorení aplikácie je cieľom vývojára otestovať jej funkčnosť, ktorá má vplyv na bezproblémové fungovanie aplikácie. Všetky možné chyby je potrebné pred predaním projektu zákazníkovi odstrániť. Testovanie je definované ako proces spúšťania programu s cieľom nájsť chybu [34].

Testovanie informačného systému pozostávalo z dvoch častí. Prvé bolo testované vývojárom projektu a po dokončení systému aj užívateľmi, ktorým je tento systém určený. Testovanie vývojárom prebiehalo po celú dobu vývoja systému. Užívatelia a vývojár boli testovateľmi projektu, čo znamená že testovanie bolo výhradne manuálne a neprebehli žiadne automatické testy.

8.1 Testovanie programátorom

Programátor testuje svoj produkt neustále. Vždy, keď je pridaná nová funkcionalita je najprv otestovaná programátorom, ktorý ju nie len otestuje, ale v prípade nefunkčnosti aj okamžite opraví. Cieľom je doviesť vyvíjaný projekt k dokonalosti a bezchybnosti. K testovaniu dochádzalo pravidelne aj po konzultácií s trénermi mužstva, ktorí mohli mať pripomienky k systému a tie boli následne upravené v systéme. Po dokončení všetkých dôležitých funkcionalít a prevedení informačného systému do používateľného stavu bolo potrebné dať systém otestovať aj jeho budúcim používateľom. Testovali ho dve skupiny, hráči a tréneri.

8.2 Testovanie členmi klubu a užívateľmi

Toto testovanie je pre celú prácu prakticky najdôležitejšie. Systém je určený pre užívateľov a ich prácu, preto musí spôsob zaobchádzania s ním sedieť ich štýlu. Spôsob spravovania ako si predstavuje programátor sa môže veľmi líšiť od toho, ako s ním chcú, alebo vedia pracovať hráči a tréneri. Priebeh a výsledky testovania užívateľov napovedia programátorovi či bola jeho práca dobrá, alebo treba spraviť vylepšenia.

Tréneri

Testovanie pozostávalo z pripravených pokynov, ktorých sa testovacie subjekty mali držať. Prvé boli testovaní tréneri klubov. Testovalo sa, ako sú schopní vytvárať nové akcie v klube,

prihlasovanie do systému a podobne. Cieľom bolo zistiť intuitívnosť systému. Trénerovi boli zadané pokyny, no bez presnejšieho určenia kde hľadať ich riešenie. Zadané pokyny boli nasledovné:

1. - spustiť webovú stránku
2. - prihlásiť sa ako tréner
3. - vytvoriť zranenie hráča “Peter Fečo”
4. - vyhľadať hráča “Lukáš Magda” v databáze zranených hráčov
5. - zadať pokutu hráčovi “Lukáš Magda” v hodnote 10 eur
6. - vytvoriť tréning s dátumom 15.06.2019
7. - potvrdiť účasť hráča “Stanislav Durovčik” na tréningu konaného 13.05.2019
8. - aktualizovať financie za mesiac júl
9. - odhlásiť sa zo systému

U trénerov sa očakávala kostrbatejšia práca vzhľadom k ich veku a neskúseností s informačnými systémami. Predpoklady sa úplne nenaplnili, a až na pár výnimiek ovládanie sýtemu nebolo veľkým problémom. U trénerov sa požadovalo hlavne otestovania ich právomoci v rámci zadávania tréningov, pokút, alebo zapisovania zranení. Prvotným problémom bol bod tri. Keďže nebol zadaný presný pokyn kde to majú hľadať, niektorí tréneri sa najprv prehrabávali horným navigačným panelom, pokiaľ nášli správnu cestu. Ostatným to však nerobilo problém. Po absolvovaní tretieho bodu, body štyri a päť neboli problém. To isté nastalo pri bode sedem. Tu bolo potrebné sa dostať až na tréning tímu, čo niektorým trénerom robilo problém. Spôsobovala to najmä zle určená farba id tímu. Tá splývala s ostatnými hodnotami v tabuľke. Tréneri tak nevedeli, že sa na to dá kliknúť. Po testovaní prišlo k zmene farby id.

Hráči

Ďalšími testovacími subjektami boli hráči. Keďže ich vekový priemer je menší ako u trénerov dalo sa aj očakávať, že so systémom budú vedieť pracovať lepšie. Očakávania boli aj potvrdené. Ich intuitívne správanie docielilo k tomu, že práca so systémom bola ľahká a rýchla. Všetky zadané pokyny boli zvládnuté skoro bez problémov.Pokyny pre hráčov boli nasledovné:

1. - spustiť webovú stránku
2. - prihlásiť sa ako hráč
3. - vyhľadať štatistiky o najlepších strelcoch
4. - zobraziť si zostavu zo zápasu FK Furča Haniska - Ruská Nova Ves
5. - zobraziť účasť na tréningu 13.05.2019
6. - potvrdiť účasť na tréningu 13.05.2019

7. - zobraziť graf výdajov klubu za posledné mesiace

8. - zmeniť lokalizáciu stránky

Počas testovania však prišlo k pár pripomienkam ohľadom dizajnu a používania systému. Jednou z nich bola nevyhovujúca farba grafu. Všetky grafy mali rovnakú farbu, tým pádom neboli veľmi priehľadné. Tu prišlo k zmene. K zmene došlo aj v prípade farby id, ktoré presmerováva užívateľov. Prišlo aj k odhaleniu pár chýb v systéme. Jedným z nich bola napríklad možnosť vytvárať viac tréningov v rovnaký dátum. Ďalšou chybou bola možnosť nevyplnenie všetkých políčok formulára pri vytváraní zranenia. Tým pádom došlo k chybe a pádu systému. Všetky chyby boli po testovaní opravené.

Užívatelia

Skupinu užívateľov predstavovali rodinní príslušníci, alebo kamaráti. Nikdy neboli členmi futbalového klubu a nikdy nepoužívali podobný systém. Očakávania o intuitívnom využívaní sa nepredpokladali. Zámerom ich testovania boli skôr pripomienky ohľadom dizajnu a rozloženia stránky. Programátor nie je aj dizajnérom zároveň, preto bolo nutné vypočuť si pripomienky bežných užívateľov stránok ohľadom štylizácie stránky. Na základe ich komentárov boli uskutočnené zmeny na frontende, čo sa dotklo aj výmeny layoutu a jeho úpravy.

8.3 Potencionálny vývoj informačného systému

Po zhodnotení a konzultáciách bolo rozhodnuté, že informačný systém bude vytváraný pre jeden konkrétny klub. Možností na rozšírenie systému zostalo však mnoho. V nasledujúcom texte budú prebraté spôsoby ako rozšíriť tento systém.

Rozšírenie systému pre viac klubov

Jedným z hlavných možností ako posunúť systém ďalej, je rozšíriť ho pre viac klubov. Aktuálne slúži ako organizačný nástroj pre jeden klub z nižšej ligy. Jednou z možností je zaviesť ho pre celú ligu. Uľahčila by sa tak práca nie len pre jeden klub, ale pre celú ligu. Potrebné by bolo však bolo vytvoriť organizátora pre ligu, ktorý by zapisoval výsledky. Alternatívou by bola dohoda na jednom administrátorovi, ktorý by po každom zápase zapisoval výsledky všetkých klubov. Taktiež by mal právo pridávať hráčov či vytvárať zápasové formácie.

Informačné štatistické portály pre slovenské ligy už existujú, ale zameriavajú sa skôr na štatistickú stránku, ako je zápis výsledkov a stav ligy. Cieľom by bolo spojiť užívateľskú časť tohto systému, pridať viac štatistických variácií a zaviesť ho do prevádzky.

Pridanie ďalších rolí do klubu

Kluby väčšinou zamestnávajú viac zamestnancov. V nižších súťažiach sa síce stáva, že tréner zastáva praktický všetky role, no v kluboch s lepším zázemím sú pozície ako lekár, športový riaditeľ atď. Preto myšlienkou ďalšieho vývoja by bolo pridanie nových rolí. Človek, ktorý rieši platy, prestupy, alebo hosťovania hráčov sa volá športový riaditeľ. Mal by možnosť prihlásenia ako ostatní užívatelia, rozdiel by bol však vo funkciách. Jedna z prvých by bola možnosť zapisovania a riešenia prestupov. Ak by prestúpil hráč, jeho úlohou by bolo to vyriešiť a zapísať aj v systéme, kde by bola viditeľná čiastka aj čas kedy k tomu došlo.

S tým by prišla aj funkcionalita zmlúv. Na vyššej úrovni sú zmluvy samozrejmosťou. Každý hráč, ktorý by prestúpil do klubu, by podpísal kontrakt s dĺžkou pôsobenia a určenou výškou platu. Športový riaditeľ by zmluvy pridával do systému, kde by boli dostupné ako preňho, tak aj pre trénera. Hráči by k zmluvám prístup nemali.

Kapitola 9

Záver

Cieľom tejto bakalárskej práce bolo vytvoriť informačný systém futbalového klubu. Zámerom práce bolo urobiť systém pre futbalový klub a uľahčiť mu tak spravovanie klubu a organizačné veci. Tvorba sa zameriavala aj na užívateľské rozhranie.

V prvej kapitole bola rozobraná teória ohľadom informačných systémov. Ďalšia kapitola sa venovala technológiám, ktoré boli v tomto systéme použité. Dôležitá bola kapitola rozoberajúca analýzu. Na kapitole sa spolupracovalo spolu s trénermi klubu. Na základe ich názorov a pripomienok boli zanalyzované všetky dôležité funkcie systému. Rozoberali sa všetci užívatelia systému a ich úlohy v ňom. K analýze boli vytvorené aj diagramy prípadov užitia, ktoré graficky znázorňovali chovanie užívateľov v systéme.

Po dôkladnej analýze sa pristúpilo k návrhu systému. Navrhnutý bol čo najpresnejšie, keďže o neho sa opierala následná implementácia zadania. Návrh systému pozostával z vytvoreného ER diagramu. V ňom boli vykreslené všetky entity, ktoré mali v systéme fungovať. Popísané boli jednotlivé atribúty a vzťahy aké budu medzi entitami prevládať.

Pre čitateľov práce bola vysvetlená implementácia. Tá popisovala počiatočné práce, ako bola inštalácia a nastavenie lokálneho serveru. Neskôr sa venovala popisu jednotlivých častí, konkrétne ako funguje prihlasovanie do informačného systému a fungovanie dvoch oddelených časti, a to administrátorskému a užívateľskému panelu. V nich sa do podrobnejšia rozoberala ich implementácia a logika fungovania.

Po vytvorení práce ju bolo potrebné otestovať. Testovanie prebiehalo aj počas vývoja, no jednalo sa iba o testovanie programátorom. K nezávislému testovaniu boli pozvaní členovia klubu ako aj ľudia, ktorí nie sú členmi klubu. Ich úlohou bolo odhaliť chyby fungovania systému ale taktiež pridať aj pripomienky ohľadom dizajnu webovej aplikácie. Na základe ich testovania boli potom odhalené chyby v funkcionalite systému a taktiež aj nedostatky vo vzhľade. Všetky nedostatky boli opravené a systém spĺňa všetky požiadavky zadania.

Informačný systém poskytuje ešte veľa možností na rozšírenie. Jedným z nich je rozšírenie systému pre viac klubov, alebo pridanie ďalších užívateľských rolí do systému. Pri rozšírení systému by boli pridané ďalšie ligy a kluby. Systém by bol tak sprístupnený viacerým hráčom a trénerom. Ak by sa pristúpilo na rozšírenie roli, do klubu by boli implementovaný užívatelia ako je lekár, alebo športový riaditeľ. Tieto dve možnosti rozšírenia sú podrobnejšie rozpísané v predošlej kapitole.

Literatúra

[1] *Bootstrap*. [Online; navštíveno 20.03.2019].
URL https://getbootstrap.com/

[2] *Django-DjangoThe web framework for perfectionists with deadlines.* . [Online; navštíveno 23.03.2019].
URL https://www.djangoproject.com/

[3] *Graphical user interface*. [Online; navštíveno 28.02.2019].
URL https://en.wikipedia.org/wiki/Graphical_user_interface

[4] *Hypertext Markup Language*. [Online; navštíveno 17.03.2019].
URL https://cs.wikipedia.org/wiki/Hypertext_Markup_Language

[5] *Informace o webových aplikacích*. [Online; navštíveno 25.02.2019].
URL https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html

[6] *Laravel - The PHP Framework For Web Artisans*. [Online; navštíveno 24.03.2019].
URL https://laravel.com/

[7] *MySQL*. [Online; navštíveno 10.05.2019].
URL https://www.mysql.com/

[8] *Oracle Database 12c*. [Online; navštíveno 10.05.2019].
URL https://www.oracle.com/database/12c-database/

[9] *PHP*. [Online; navštíveno 20.03.2019].
URL https://www.php.net/

[10] *PostgreSQL: The World's Most Advanced Open Source Relational Database*. [Online; navštíveno 05.05.2019].
URL https://www.postgresql.org/

[11] *Processing dynamic pages*. [Online; navštíveno 19.04.2019].
URL http://etutorials.org/Macromedia/Dream+Weaver+Online+Help/Getting+Started+with+Dreamweaver/Understanding+Web+Applications/How+a+web+application+works/Processing+dynamic+pages/

[12] *Processing static web pages*. [Online; navštíveno 19.04.2019].
URL http://etutorials.org/Macromedia/Dream+Weaver+Online+Help/Getting+Started+with+Dreamweaver/Understanding+Web+Applications/How+a+web+application+works/Processing+static+web+pages/

[13] *Symfony*. [Online; navštíveno 23.03.2019].
URL https://symfony.com/

[14] *Typy uživatelských rozhraní a jejich specifika*. [Online; navštíveno 28.02.2019].
URL https://wikisofia.cz/wiki/Typy_u%C5%BEivatelsk%C3%BDch_rozhran%C3%AD_a_jejich_specifika

[15] *Uživatelské rozhraní (user interface)*. [Online; navštíveno 28.02.2019].
URL https://wikisofia.cz/wiki/U%C5%BEivatelsk%C3%A9_rozhran%C3%AD_(user_interface)

[16] *What is PHP? Write your first PHP Program*. [Online; navštíveno 01.05.2019].
URL https://www.guru99.com/what-is-php-first-php-program.htmld

[17] Boronczyk, T.; Naramore, E.; Gerner, J.; aj.: *PHP 6, MySQL, Apache*. Computer Press, 2009, ISBN ISBN 978-80-251-2767-4.

[18] Bourgeois, D. T.: *Information Systems for Business and Beyond*. The Saylor Foundation, 2014, ISBN ISBN 9781533064165.
URL https://bus206.pressbooks.com/chapter/chapter-1/

[19] Brejcha, J.: *Co skrývá uživatelské rozhraní*. Horava a Associates, 2009, ISBN ISBN 978-80-254-5295-0.

[20] Burget, R.: *Informační systémy Pojem informačního systému, data, informace TEXu*. [Online; navštíveno 25.02.2019].
URL https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIIS-IT%2Flectures%2Fp01_Informacni_systemy.pdf&cid=12157

[21] Evans, D.: *Introduction to Computing*. CreateSpace Independent Publishing Platform, 2011, ISBN ISBN 978-1463687472.
URL https://computingbook.org/FullText.pdf

[22] Friends, A.: *Apache friends-downloads*. [Online; navštíveno 1.10.2018].
URL https://www.apachefriends.org/download.html

[23] Herko, L.: *Frontend vs Backend: v čom je rozdiel?* [Online; navštíveno 01.05.2019].
URL https://www.learn2code.sk/blog/front-end-vs-back-end

[24] Hruška, T.; Krivka, Z.: *Informační systémy (IIS,PIS) Studijní opora – Fakulta informačních technologií VUT v Brně*. . 2012, [Online; navštíveno 25.02.2019].
URL https://www.fit.vutbr.cz/study/courses/WAP/private/podklady/Opory/OIISPojem.pdf

[25] Klíma, T.: *Architektura MVC*. [Online; navštíveno 15.03.2019].
URL http://jakpsatphp.cz/MVC/

[26] Kubíček, B. L.: *Využití přístupů založených na modelu MVC pro tvorbu aplikac*. Diplomová práce, Mendelova univerzita v Brně Provozně ekonomická fakulta, 2011, vedúca práce: Ing. David Procházka Ph.D.

[27] Laravel: *Installation-Laravel*. [Online; navštíveno 1.10.2018].
URL https://laravel.com/docs/5.8/installation

[28] Long, J.: *I Don't Speak Your Language: Frontend vs. Backend.* [Online; navštíveno 01.05.2019].
URL https://blog.teamtreehouse.com/i-dont-speak-your-language-frontend-vs-backend

[29] McFarlin, T.: *JavaScript.* [Online; navštíveno 15.04.2019].
URL https://code.tutsplus.com/tutorials/what-is-javascript--cms-26177

[30] Morriss, S.: *The Ultimate Guide to CSS.* [Online; navštíveno 17.03.2019].
URL https://skillcrush.com/2012/04/03/css/

[31] Ronacher, A.: *Flask-Foreword.* [Online; navštíveno 21.03.2019].
URL http://flask.pocoo.org/docs/0.12/foreword/#

[32] Shannon, R.: *What is HTML?* [Online; navštíveno 17.03.2019].
URL https://www.yourhtmlsource.com/starthere/whatishtml.html

[33] Sklar, D.: *PHP 7 – Praktický průvodce nejrozšířenějším skriptovacím jazykem pro web.* Zoner Press, 2017, ISBN ISBN 978-80-7413-363-3.

[34] Vallušová, V.: *Testovanie webových aplikácii.* [Online; navštíveno 19.04.2019].
URL https://is.muni.cz/th/uxw0p/Diplomova_praca_Vladimira_Vallusova.pdf

[35] Čápka, D.: *MVC architektura.* [Online; navštíveno 15.03.2019].
URL https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor/

Príloha A

Obsah priloženého pamäťového média

- **src/** - zložka obsahujúca zdrojové kódy programu
- **doc/** - zložka obsahujúca bakalársku prácu vo formáte pdf
- **install/** - zložka obsahujúca návod k inštalácií