VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM FOTBALOVÉHO KLUBU

INFORMATION SYSTEM OF A FOOTBALL CLUB

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **VERONIKA GAZDOVÁ**
AUTHOR

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

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

Studentka: **Gazdová Veronika**
Program: Informační technologie
Název: **Informační systém fotbalového klubu**
Information System of a Football Club
Kategorie: Informační systémy

Zadání:

1. Seznamte se s principy tvorby webových aplikací, dostupnými prostředími a frameworky.
2. Analyzujte požadavky na informační systém fotbalového klubu zahrnující grafický editor sestav pro trenéra, podporu pro organizaci tréninků, evidenci cestovních příkazů s možností jejich tisku v PDF a další.
3. Navrhněte informační systém dle požadavků, použijte vhodné modelovací techniky.
4. Navržený informační systém implementujte a ověřte jeho funkčnost na vhodném vzorku dat.
5. Zhodnoťte dosažené výsledky a diskutujte další možné pokračování v tomto projektu.

Literatura:

- Naramore, E., Gerner, J. et al: PHP 6, MySQL, Apache: Vytváříme webové aplikace. Computer Press, 2009. ISBN: 978-8-0251-2767-4.
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

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

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 26. října 2018

Abstrakt

Tato práce se zabývá vytvořením informačního systému pro fotbalový klub. Jedná se o webovou aplikaci implementovanou v jazyce PHP s využitím frameworku Nette. Mezi nejdůležitější funkce systému patří zejména možnost vytvářet grafickou sestavu pro jednotlivé zápasy, evidovat si výkazy jízd s jejich následným vygenerováním v PDF, organizace tréninků a v neposlední řadě evidence výsledků jednotlivých zápasů. Systém byl úspěšně vytvořen a následně otestován skupinou potenciálních uživatelů.

Abstract

This thesis deals with the creating of an information system of a football club. It is a web application implemented in PHP using Nette Framework. In particular, the most important functions of the system include the creation of graphical line up for every single match, the evidence of journey reports with the possibility of generating in PDF, training organization and evidence of results of every single match. The system was successfully implemented and then tested by a group of potential users.

Klíčová slova

PHP 7, Nette, MySQL, Bootstrap, informační systém, web, fotbal

Keywords

PHP 7, Nette, MySQL, Bootstrap, information system, web, football

Citace

GAZDOVÁ, Veronika. *Informační systém fotbalového klubu*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Informační systém fotbalového klubu

Prohlášení

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

.....
Veronika Gazdová
7. května 2019

Poděkování

Ráda bych poděkovala panu Ing. Vladimíru Bartíkovi, Ph.D. za vedení mé bakalářské práce, čas strávený na konzultacích a kontrolou mé práce.

Obsah

Kapitola 1

Úvod

Tento dokument vznikl jako bakalářská práce na Fakultě informačních technologií Vysokého učení technického v Brně. Cílem této bakalářské práce je vytvořit informační systém pro fotbalový klub TJ Sokol Žalkovice, který usnadní organizaci tréninků jednotlivých týmů, umožní graficky vytvářet sestavy pro daný zápas, hráčům a trenérům umožní evidenci cestovních příkazů a v neposlední řadě bude zobrazovat výsledky jednotlivých zápasů.

Aktuální organizace chodu klubu není příliš efektivní, z tohoto důvodu vznikl požadavek na tento informační systém, který bude umožňovat správu některých částí klubu a zaměří se na co největší zjednodušení práce trenérům, vedoucím a hráčům.

Vzhledem k tomu, že je potřeba vytvořit systém, který nebude omezen pouze na specifická zařízení, byl realizován jako webová aplikace, která využívá technologie dostupné u většiny webových prohlížečů. Samozřejmě je potřeba myslet i na to, že uživatelé budou systém využívat i z mobilních zařízení. Z tohoto důvodu byla pro vývoj informačního systému použita knihovna Bootstrap, která optimalizuje vzhled aplikace pro zařízení s různou velikostí displeje.

Obsah práce je rozdělen do několika kapitol, které postupně vysvětlují celý proces při vytváření informačního systému. Kapitola 2 popisuje principy tvorby webových aplikací a seznamuje čtenáře se základními programovacími a skriptovacími jazyky, které se obvykle využívají při tvorbě webových aplikací.

Další kapitolou je kapitola 3, která podrobněji popisuje technologie, které byly pro vývoj tohoto informačního systému použity. V této kapitole může také čtenář vidět srovnání PHP frameworků.

Kapitola 4 na rozdíl od dvou předcházejících kapitol již není teoretická. Je zde uvedena specifikace a analýza požadavků, které by měl vyvíjený systém splňovat. Tato část zahrnuje stanovení konkrétních požadavků na funkčnost a také uživatelských rolí, které budou v systému vystupovat.

Na tuto kapitolu navazuje kapitola 5, ve které je popsán návrh informačního systému, který byl vytvořen na základě požadavků stanovených v předchozí kapitole.

V kapitole 6 je čtenář seznámen se samotnou implementací výsledného systému. Během implementace byly využity poznatky získané při analýze požadavků a návrhu systému. Je zde popsána implementace nejdůležitějších částí systému.

Následující kapitola 7 se zabývá testováním systému, které je nezbytné při vytváření aplikace. Je zde uvedeno, jakým způsobem byl systém prověřen kvůli odhalení případných nedostatků.

Poslední kapitolou je kapitola 8. Zde jsou zanalyzovány a zhodnoceny veškeré dosažené výsledky, které byly při vývoji tohoto informačního systému získány. Tato kapitola obsahuje také část zabývající se možnými budoucími rozšířeními informačního systému.

Kapitola 2

Principy tvorby webových aplikací

Tato kapitola popisuje, co je to webová aplikace a jaká je její architektura. Také je v této kapitole popsána architektura klient-server, která se především dříve využívala pro tvorbu webových aplikací a informačních systémů.

Kapitola *Principy tvorby webových aplikací* se dále zaměřuje na technologie, které se k tvorbě webových aplikací používají. Další vybrané a použité technologie budou popsány v kapitole 3.

2.1 Webová aplikace

Webová aplikace je aplikace, která je určena pro prostředí internetu (WWW - World Wide Web). Běží na serverech, které jsou umístěny na internetu a uživatelé mají k této aplikaci přístup skrze internet. Architektura webových aplikací může být buď typu klient-server (viz kapitola 2.1.1), nebo může být strukturována jako třívrstvá (viz kapitola 2.1.2). Webová aplikace využívá tzv. tenkého klienta, což je internetový prohlížeč. [13]

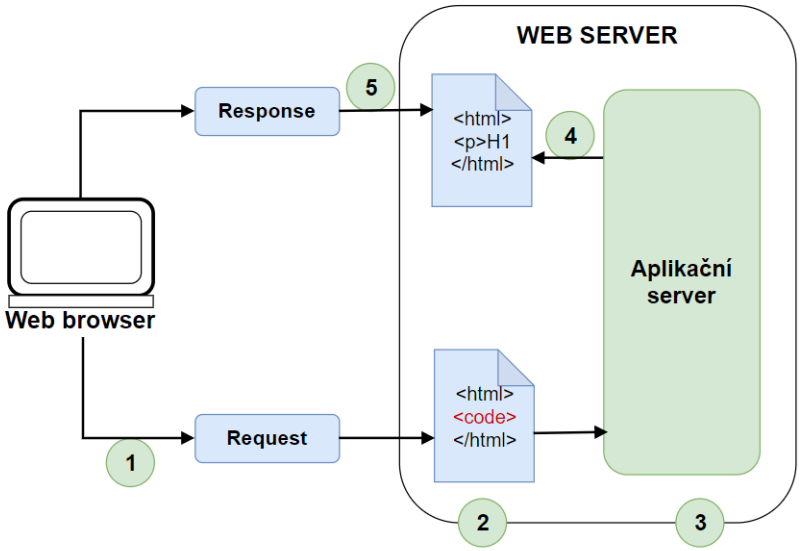
Webová aplikace může na první pohled vypadat jako běžná webová stránka, ale obvykle se však jedná o aplikaci, která provádí složitější úlohy a využívá k tomu databázi. [11] Obsahuje stránky s částečně nebo úplně neurčeným obsahem. Jakmile návštěvník požádá o stránku z webového serveru, určí se konečný obsah stránky. Vzhledem k tomu, že konečný obsah stránky se pro různé požadavky liší a závisí na akci, kterou provede návštěvník, tak se takové stránce říká dynamická. [2] Schéma zpracování dynamické webové stránky je zobrazeno na obrázku 2.1.

Výhody webových aplikací

- Uživatel nemusí instalovat žádný speciální software na svém počítači, stačí mu pouze internetový prohlížeč.
- Uživatel má vždy přístup k aktuální zveřejněné verzi aplikace.
- Data jsou uchovávána na serveru, kde zároveň dochází k jejich aktualizaci.

Nevýhody webových aplikací

- Rychlost aplikace je závislá na kvalitě připojení, z toho důvodu může být někdy pomalejší tok dat a práce s aplikací.
- V případě nekvalitního poskytovatele může dojít k bezpečnostnímu riziku, například k úniku dat.

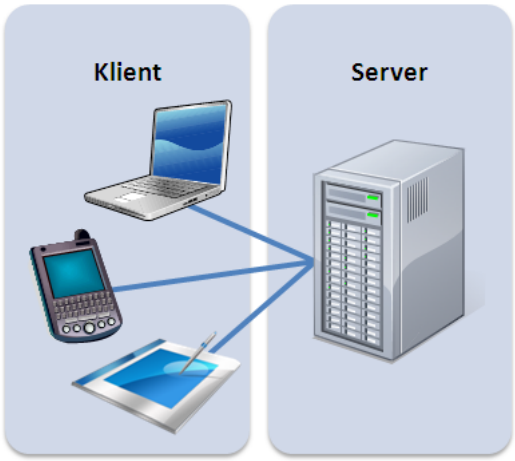


Obrázek 2.1: Schéma zpracování dynamické webové stránky [2]

2.1.1 Architektura klient-server

Architektura klient-server je jeden z typů architektury informačních systémů. Tato architektura je dvouvrstvá, kde klient obsahuje uživatelské rozhraní a aplikační logiku, na serveru běží relační databáze. Schéma architektury klient-server je možné vidět na obrázku 2.2.

Klient a server mezi sebou komunikují a vzájemně si předávají data. Hlavní úlohou klienta je přeložit uživatelský požadavek tak, aby byl srozumitelný serveru a následně přeložit odpověď od serveru tak, aby byla srozumitelná uživateli a mohla být prezentována na obrazovce. Server tedy zpracovává dotazy v databázi a klient je prezentuje, zajišťuje aplikační logiku a zprostředkovává rozhraní pro uživatele. [9]

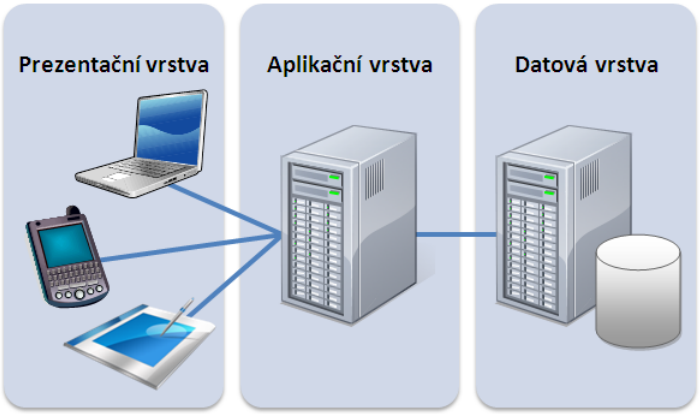


Obrázek 2.2: Schéma architektury klient-server [9]

V dnešní době již není architektura klient-server tolik využívána. Je to zejména proto, že klient obsahuje většinu aplikační logiky a se složitostí aplikace rostou nároky na klientská zařízení. Dalším důvodem je obtížnější aktualizace aplikace na straně klienta a také to, že je problematické zpřístupnit část aplikace pomocí webového prohlížeče. [9] Z dvouvrstvé architektury se tedy později vyvinula architektura třívrstvá, která je více popsána v kapitole 2.1.2.

2.1.2 Třívrstvá architektura

Architektura webových aplikací je nejčastěji strukturována jako třívrstvá (viz obrázek 2.3). První vrstvou (prezentační) je webový prohlížeč, druhá vrstva (aplikační) je tvořena nástroji pro dynamické generování stránek, např. PHP a třetí vrstvu (datovou) tvoří databáze. Webový prohlížeč posílá požadavky střední vrstvě, která je obsluhuje pomocí dotazování se databáze a následně dochází ke generování uživatelského rozhraní [5]. Výhodou třívrstvé architektury je, že odděluje jednotlivé vrstvy tak, aby na sobě nebyly závislé.



Obrázek 2.3: Schéma třívrstvé architektury [10]

Mnoho webových aplikací je napsáno přímo v čistém programovacím jazyku, jako je například PHP (viz kapitola 2.2.3). Naproti tomu dnes existuje řada systémů, tzv. frameworků, které usnadňují vývoj aplikace tak, aby se vývojář mohl soustředit pouze na své zadání. Některé hojně využívané frameworky budou popsány v kapitole 3.1.

2.2 Jazyky a technologie

V této sekci budou popsány základní jazyky a technologie, které se využívají při tvorbě webových aplikací.

2.2.1 HTML

HTML (*Hypertext Markup Language*) je značkovací jazyk používaný pro tvorbu webových stránek. Vývoj HTML byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka. [29]

HTML popisuje strukturu webu pomocí značek, tzv. *tagů* a jejich vlastností. Mezi značky se uzavírají části textu dokumentu a tím se určuje sémantika obsaženého textu. Názvy

jednotlivých tagů a jejich vlastností se uzavírají mezi znaky `<` a `>`. Tagy jsou obvykle párové, koncová značka je shodná se značkou počáteční, ale má navíc před názvem znak `/`. [29]

Každá HTML stránka musí obsahovat přesně definovanou strukturu. Celá stránka je obalena tagem `<html>` a ukončovacím tagem `</html>`. Uvnitř těchto tagů se nacházejí tagy `<head>` (hlavička dokumentu) a `<body>` (tělo dokumentu), které jsou také párové. V hlavičce dokumentu jsou obsaženy důležité metainformace (titulek, CSS styly atd.). Následuje tělo, ve kterém se nachází samotný obsah stránky formátovaný pomocí spousty HTML značek.

Nejnovější verzí HTML je v současné době HTML5 [26], která byla vytvořena aktualizací předchozí verze, tedy HTML4. Tato verze přinesla některé nové značky, elementy či atributy. Chceme-li používat HTML5, musíme na začátek dokumentu vložit značku `<!DOCTYPE html>`.

2.2.2 CSS

CSS (*Cascading Style Sheets*) je jazyk pro popis způsobu zobrazení elementů na stránkách napsaných v jazycích HTML. Kaskádové styly nabízejí rozsáhlejší možnosti formátování než samotné HTML. Umožňují jednoduše změnit nějaký detail webové stránky, například při změně barvy nadpisu stačí pouze změnit jednu vlastnost v CSS souboru, který je přilinkovaný ke všem stránkám a není tak potřeba procházet celý HTML kód. [8]

CSS může být do HTML přidáno třemi způsoby:

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

CSS je zpravidla definován jako seznam *pravidel*. Každé pravidlo je složeno ze *selektoru* a *bloku deklarací*. Selektor určuje, na který element dokumentu se aplikuje vlastnost nastavená v deklaraci. Blok deklarací je tvořen seznamem dvojic *identifikátor vlastnosti*-*hodnota vlastnosti*. [8]

2.2.3 PHP

PHP (*Hypertext Preprocessor*) je skriptovací programovací jazyk, který je určen zejména pro vytváření dynamických a interaktivních webových stránek. [27]

Typický PHP skript obsahuje jednak část s HTML kódem a jednak kusy programového kódu. Skripty PHP jsou prováděny na straně serveru, kde se vygeneruje HTML dokument a ten je následně odeslán uživateli. Uživatel tedy obdrží výsledek spuštěného skriptu, ale nezjistí, jaký byl původní PHP kód. Jazyk PHP není závislý na platformě a skripty lze většinou mezi operačními systémy přenášet bez jakýchkoliv úprav. [20]

Pro implementaci informačního systému bylo využito PHP 7.2 spolu s Nette Frameworkem, který bude více popsán v kapitole 3.2.

2.2.4 MySQL

Pro ukládání dat informačního systému je využita relační databáze MySQL. MySQL (*My Structured Query Language*) je systém řízení báze dat uplatňující relační databázový model. Jedná se o multiplatformní databázi a komunikace s ní probíhá prostřednictvím jazyka SQL. Tento jazyk umožňuje jednoduché získání dat z databáze. [12]

Data v MySQL databázi jsou uložena v tabulkách, které slučují záznamy stejného významu. Každá tabulka má sloupce a řádky, v každém řádku jsou záznamy předem určeného typu. Záznamy by měly být v tabulce jednoznačně identifikovány, k tomu se využívá primární klíč. Primárním klíčem může být jediný sloupec nebo kombinace více sloupců tak, aby byla zaručena jeho jednoznačnost. Zároveň také musí platit, že primární klíč nebude obsahovat hodnotu NULL a každá tabulka bude mít definovaný právě jeden primární klíč. [28]

Na obrázku 2.4 je možné vidět tabulku tým, tabulka obsahuje 2 záznamy a každý je jednoznačně identifikovaný primárním klíčem, kterým je sloupec *id*.

id	nazev	je_v_klubu
1	TJ Sokol Žálkovice A	1
2	TJ Sokol Žálkovice B	1

Obrázek 2.4: Tabulka tým

Kapitola 3

Použité technologie

Tato kapitola popisuje technologie, které byly použity pro vývoj informačního systému. Bude zde uvedeno srovnání nejpoužívanějších PHP frameworků pro tvorbu webových aplikací. Dále je představen Nette Framework, který byl vybrán jako nejvhodnější pro vývoj této bakalářské práce. Následující sekce se zabývají ostatními technologiemi a nástroji, které byly při implementaci využity.

3.1 Srovnání frameworků

V této části bude uvedeno porovnání některých hojně využívaných PHP frameworků spolu s Nette Frameworkem, kterému se následně bude více věnovat kapitola 3.2.

3.1.1 Laravel

Laravel je open-source PHP framework, poskytující velmi přehledné a rychlé řešení pro vývoj webových aplikací. Je založen na principu architektury model-view-controller (MVC). Mezi hlavní výhody, které Laravel poskytuje, patří jednoduchá integrace s různými knihovnami, jednoduché nastavení cest, snaha dodržovat čitelnost a jednoduchost kódu. Laravel také poskytuje jednoduchou práci s databází. Nezbytnou součástí tohoto frameworku je i Composer, který bude více popsán v kapitole 3.3. [7]

3.1.2 CodeIgniter

CodeIgniter je také open-source PHP framework, který však není striktně založený na principu architektury MVC. Je nutné používat třídy komponenty řadiče (Controller), ale modely a pohledy povinné nejsou, je možné používat své vlastní jmenné a kódovací konvence. Díky tomu poskytuje CodeIgniter vývojářům značnou volnost. Nabízí také sady nástrojů a knihoven s jednoduchým rozhraním, které řeší časté a opakující se úlohy. Vývojář se tedy může zaměřit na řešení zadaného problému, kterého dosáhne s menším množstvím potřebného kódu. [3]

3.1.3 CakePHP

Stejně jako výše zmíněný Laravel a CodeIgniter patří i CakePHP mezi open-source frameworky a je založen na architektuře MVC. CakePHP využívají weby proslulých značek, jako je např. BMW a Hyundai. Jedná se o velmi dobrý nástroj pro vytváření webových

aplikací, které potřebují vysokou úroveň zabezpečení, protože má mnoho zabudovaných bezpečnostních funkcí. Další výhodou CakePHP je, že má srozumitelně uspořádanou strukturu, což zajišťuje snadnou orientaci v celém kódu. [24]

3.1.4 Nette

Nette patří také mezi open-source frameworky pro tvorbu webových aplikací, zaměřuje se na eliminaci bezpečnostních rizik, podporuje AJAX a znovupoužitelnost kódu. Nette Framework využívají významné společnosti jako je např. GE Money, Mladá fronta, Slevomat a další. [17] Jedná se o český framework, který není tolik rozšířený v zahraničí jako například Laravel nebo CakePHP, nicméně byl zvolen jako nejlepší volba pro implementaci, proto bude dále více popsán v kapitole 3.2.

3.2 Nette

Jak bylo zmíněno výše, Nette Framework byl zvolen pro implementaci této bakalářské práce. Tato kapitola popisuje hlavní výhody tohoto frameworku, zabývá se architekturou Nette, šablonovacím systémem Latte, ladícím nástrojem Tracy a dalším.

Velkou výhodou Nette je velmi aktivní česká komunita, která si vzájemně radí, pomáhá a také vytváří vlastní doplňky a rozšíření. [17] Při implementaci bylo použito rozšíření PDFResponse (kapitola 3.2.4) a DateInput.

Nette Framework nabízí řadu dalších výhod:

- excelentní šablonovací systém Latte (viz kapitola 3.2.2),
- důmyslné zabezpečení před zranitelnostmi,
- vyzrálý a čistý objektový návrh,
- ladící nástroj Tracy (viz kapitola 3.2.3).

3.2.1 MVC aplikace, Presentery a Router

Model-View-Controller je softwarová architektura, která vznikla za účelem oddělit u aplikací s grafickým rozhraním kód obsluhy (controller) od kódu aplikační logiky (model) a od kódu zobrazujícího data (view). Díky tomu je aplikace přehlednější, je usnadněn budoucí vývoj a je možné testovat jednotlivé části zvlášť. [16]

Model

Model je datový a funkční základ celé aplikace. Model nemá informace o existenci controlleru nebo view. Model obsahuje veškerou aplikační logiku. Pod tím si lze představit komunikaci s databází nebo různé výpočty. Například přihlášení uživatele do systému představuje akci modelu. [16]

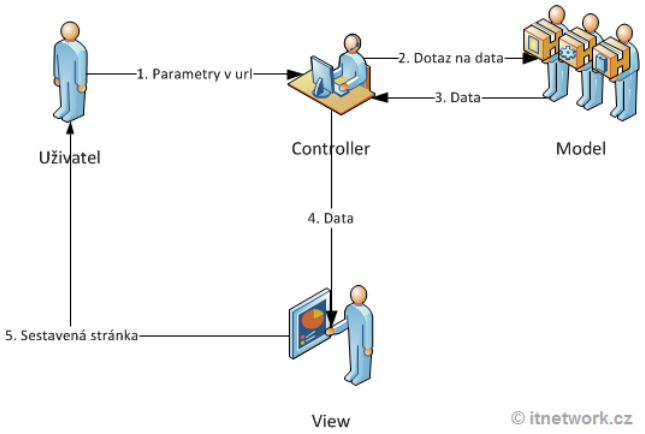
View

View je vrstva aplikace, která má na starost zobrazení výsledku požadavku. Většinou využívá šablonovací systém a ví, jak se má zobrazit výsledek získaný z modelu. View je závislý pouze na datech, která jsou mu poskytnuta a neměl by mít možnost si je sám brát. [16]

Controller

Jedná se o propojující prvek obou komponent, který zpracovává požadavky uživatele. Na jejich základě volá patřičnou aplikační logiku (model) a poté požádá view o vykreslení dat. V Nette Frameworku je controller nahrazen *presenterem*, který bude dále více popsán. [16]

Na obrázku 3.1 je možné vidět architekturu MVC.



Obrázek 3.1: Architektura MVC [21]

Presenter

Jedná se o objekt, který vezme požadavek přeložený routerem z HTTP požadavku a vygeneruje odpověď. Odpovědí může být HTML stránka, obrázek, XML dokument a další. [18]

Presenter je v Nette potomek třídy `Nette\Application\UI\Presenter` a podle příchozích požadavků spouští odpovídající akce a vykresluje šablony. Akce presenteru je logická část presenteru, která vykonává jednu akci, např. odhlásí uživatele. [18]

Router

Router je obousměrný překladač mezi HTTP požadavkem/URL a akcí presenteru. Z HTTP lze odvodit akci presenteru, ale i obráceně lze k akci vygenerovat odpovídající URL. [18]

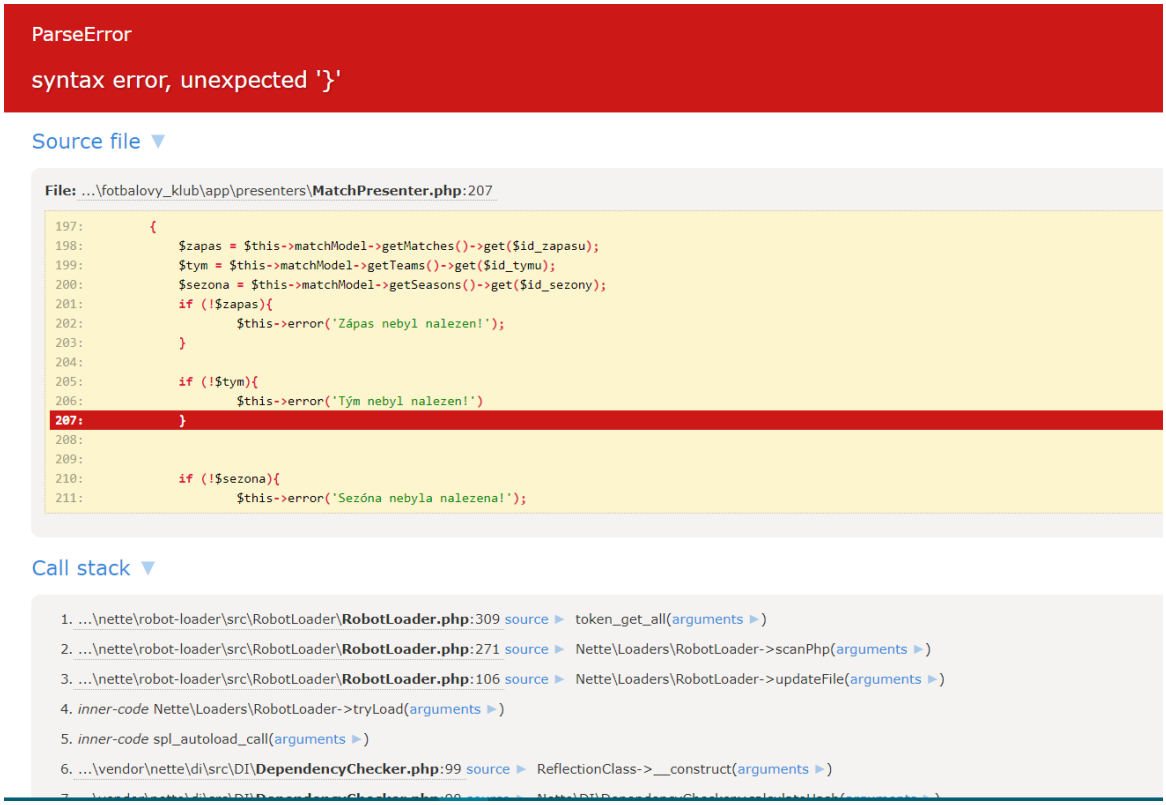
3.2.2 Šablonovací systém Latte

Latte je šablonovací systém pro PHP, který zpříjemňuje práci a zabezpečuje výstup před zranitelnostmi. Překládá šablony na jednoduchý a optimalizovaný PHP kód. Důležitou vlastností Latte je, že automaticky převádí znaky, které mají v HTML speciální význam na jiné odpovídající sekvence.

Latte poskytuje velké množství maker a filtrů, které zpříjemňují práci v šabloně a dělají ji přehlednější. [15]

3.2.3 Tracy

Tracy je ladící nástroj Nette, který pomáhá rychle odhalit a opravit chyby, logovat chyby, měřit čas a paměťové nároky. Pokud je ve zdrojovém kódu nějaká chyba, Tracy ji umí přehledněji a konkrétněji zobrazit než chyba, která by byla zobrazena interpretem jazyka PHP. Zobrazení výpisu chyby pomocí Tracy lze vidět na obrázku 3.2. [19]



Obrázek 3.2: Ladící nástroj Tracy

3.2.4 PDFResponse

PDFResponse je doplněk pro Nette, který umožňuje vytvořit dokument ve formátu PDF z HTML šablony a využívá k tomu knihovnu mPdf. Instalace tohoto rozšíření se provádí jednoduše pomocí composeru (viz kapitola 3.3):

```
composer require jkuchar/pdfresponse:dev-master
```

Tento doplněk byl využit při implementaci výkazů jízd a jejich zobrazení v PDF. Použitím tohoto doplňku se bude více zabývat kapitola 6.8.

3.3 Composer

Composer je nástroj na správu závislostí v PHP. Umožňuje nám snadnou a automatickou instalaci knihoven a závislostí do našeho projektu. Parametry pro Composer jsou uvedeny v souboru `composer.json` v kořenovém adresáři projektu. V tomto souboru se také nachází

meta informace o projektu nebo knihovně a jejich závislostech. Composer se ovládá z příkazové řádky, stačí pouze uvést příkaz na stáhnutí rozšíření či balíčku a Composer se sám postará o jeho instalaci. [14]

3.4 Bootstrap

Bootstrap je nejoblíbenější open-source framework pro HTML, CSS a JavaScript, díky němu je možné vytvářet responzivní webové stránky optimalizované i pro mobilní zařízení. [25] Bootstrap nabízí responzivní šablony CSS a HTML pro tlačítka, tabulky, navigační menu, rozbalovací nabídky a další prvky, které jsou použitelné na webové stránce. [4]

Bootstrap není jen rámec CSS, ale využívá také JavaScript, díky kterému je možné vytvářet dynamické prvky na stránce, například výsuvné postranní menu. Jedná se tedy o velmi užitečný nástroj, hlavně díky tomu, že umožňuje zobrazovat stránky ve vysoké kvalitě na obrazovkách různých úhlopříček. [4]

3.5 JavaScript

JavaScript je objektově orientovaný jazyk, který je navržen pro skriptování v prostředí webového prohlížeče. Důležitou vlastností JavaScriptu je, že běží na straně klienta a ne na serveru, na kterém jsou stránky uloženy. Díky tomu je možné měnit obsah webové stránky u uživatele až po jejím načtení. [23]

JavaScript se využívá zejména pro vytváření interaktivních webových stránek. Typickým příkladem použití může být kontrola správného vyplnění formuláře, obrázky měnící se po přejetí myši nebo rozbalovací menu. [1]

3.6 FullCalendar

FullCalendar [6] je JavaScriptový kalendář událostí. Kalendář zobrazuje události, které je možné získat z databáze, a proto byl zvolen pro implementaci správy tréninků v této bakalářské práci. FullCalendar umožňuje pro každou událost nastavit několik parametrů. Pro potřebu evidence tréninků byly využity parametry: `id, start, end, title, url` a `editable`. Parametr `id` jednoznačně identifikuje událost, parametry `start` a `end` popisují časový interval události, `title` je název zapsané události, `url` označuje adresu stránky, na kterou se po kliknutí na danou událost aktuální stránka přesměruje a parametr `editable` určuje, zda je možné událost editovat.

Více bude implementace evidence tréninků a použití doplňku FullCalendar popsáno v kapitole 6.7.

Kapitola 4

Analýza a specifikace požadavků

Analýza a specifikace požadavků patří při vývoji informačního systému mezi jednu z nejdůležitějších a klíčových částí. Cílem analýzy je specifikovat požadavky na daný systém a popsat jeho funkčnost tak, aby přesně odpovídal požadavkům zadavatele.

V případě tohoto informačního systému byl po konzultaci s hráči, trenéry a vedoucím klubu TJ Sokol Žalkovice vytvořen seznam požadavků. Bylo potřeba zjistit, základní funkcionalitu, kterou by měl systém splňovat a určit zásadní vlastnosti, které by ulehčily zejména hráčům a trenérům práci. Tyto požadavky byly následně konzultovány a schváleny vedoucím bakalářské práce panem Ing. Vladimírem Bartíkem, Ph.D..

Mezi nejdůležitější požadavky, které by systém měl splňovat, patří:

- správa sezón, soutěží, týmů a hráčů,
- evidence výkazů jízd na zápasy, případně tréninky a jejich následná možnost tisku v PDF,
- organizace tréninků zahrnující zobrazení účasti/neúčasti hráčů,
- grafický editor pro zobrazení sestavy konkrétního zápasu
- a v neposlední řadě také výsledky zápasů týmů v klubu, včetně výpisů událostí (např. červené karty, góly) u daného zápasu.

Tyto a další požadavky budou popsány v několika dalších kapitolách.

Také je potřeba zmínit, že v systému budou vystupovat následující role uživatelů:

- běžný nepřihlášený uživatel,
- hráč klubu,
- trenér
- a vedoucí klubu.

Tyto role jsou potřeba pro odlišení oprávnění a akcí, které může uživatel v systému vykonávat.

4.1 Správa uživatelů, přihlášení

Velmi důležitou a nezbytnou částí systému bude správa uživatelů, která zahrnuje vytvoření uživatelských účtů, možnost aktivace/deaktivace a změnu hesla uživatelů. Toto oprávnění bude mít pouze jediná role - vedoucí klubu.

Pro pokročilé funkce systému bude potřeba být přihlášen, aby se zamezilo vstupu do systému osobám, které nebudou mít do některých sekcí přístup. Pokud bude uživatelský účet vedoucím klubu aktivován, bude se moci daný uživatel přihlásit. Přihlášený uživatel bude mít možnost změnit si přihlašovací heslo a případně upravit osobní údaje, jako je např. email. Taktéž bude mít uživatel další oprávnění v závislosti na své roli, což bude popsáno v kapitole 4.7.

4.2 Správa sezón, soutěží, týmů a hráčů

Tato část bude taktéž klíčová pro správný chod systému. Vzhledem k tomu, že systém má evidovat výsledky zápasů klubových týmů, bude tedy potřeba přiřadit hráče do týmů a týmy registrovat do soutěže v dané sezóně. Oprávnění vytvářet sezóny, soutěže, týmy a hráče bude mít pouze uživatel v roli vedoucí klubu. Stejně tak bude vedoucí klubu jediným, kdo může registrovat týmy a přidávat hráče do týmů.

Uživatel, který bude v roli hráč nebo trenér, případně nepřihlášený uživatel, bude mít pouze možnost zobrazovat si informace o tom, jaké soutěže se hrají v dané sezóně, týmy v jednotlivých soutěžích a samozřejmě také hráče, kteří jsou v konkrétním týmu.

4.3 Výkazy jízd

Výkazy jízd jsou část systému, která bude sloužit především pro hráče a trenéry. Stávající situace je taková, že pokud hráč nebo trenér dojíždí na tréninky či zápasy, má možnost si vykázat jízdy a jsou mu následně proplaceny. Hlavní nevýhoda spočívá v tom, že pokud si dotyčný vykazuje např. 10 jízd, musí vypisovat ručně odkud-kam jel, následně spočítat kilometry a částku, která mu má být proplacena. Z tohoto důvodu tedy vznikl tento požadavek, který by ulehčil evidenci jízd.

Jak plyne z předchozího odstavce, uživatel, který bude v roli hráč nebo trenér, bude mít možnost si evidovat jízdy v informačním systému. Následně bude možné si jízdy vygenerovat v PDF, vytisknout a vykázat. Vedoucí klubu bude oprávněn zobrazit si evidované jízdy všech uživatelů. Naopak uživatel, který není přihlášen, do této sekce nebude mít přístup.

4.4 Kalendář tréninků

Tato část bude vytvořena především za účelem přehledného zobrazení tréninků v kalendáři. Taktéž bude velmi důležitá pro trenéra, který bude mít možnost zobrazit si, jaká je účast, případně neúčast na tréninku. Tato informace bude poskytnuta i hráčům, jen s tím rozdílem, že hráč si bude moci zobrazit účast pouze u tréninku, který se týká jeho týmu. U ostatních tréninků mu bude zpřístupněno pouze místo konání a čas konání tréninku.

Uživatel v roli trenéra bude oprávněn tréninky vytvářet, měnit čas konání a v neposlední řadě také trénink zrušit. Vedoucí klubu bude mít možnost zobrazit si kalendář tréninků včetně účasti/neúčasti hráčů. Nepřihlášený uživatel nebude mít do této části systému přístup povolen.

4.5 Grafický editor sestav

Grafický editor sestav bude sloužit opět především pro hráče a trenéry. Uživatel v roli trenéra bude mít možnost vytvořit u každého zápasu klubového týmu grafickou sestavu. Nejprve bude potřeba vytvořit soupisku 11 hráčů a následně bude možné hráče graficky rozmístit v editoru, v závislosti na tom, jakou formaci si trenér zvolí, např. 4-4-2.

Hráč bude mít možnost si trenérem vytvořenou grafickou sestavu zobrazit a bude tedy před zápasem vědět, na jakém postu bude hrát. Vedoucí klubu bude taktéž oprávněn zobrazit si sestavu. Nepřihlášenému uživateli nebude tato funkcionalita přístupná.

4.6 Zápasy

Část informačního systému, která bude zobrazovat zápasy, bude přístupná všem uživatelům. Nepřihlášený uživatel si bude moci zobrazovat skóre u daného zápasu a také podrobnosti zápasu, jako je např. informace o tom, jaké byly soupisky týmů, kdo z hráčů dával gól apod.

Přihlášený uživatel v roli vedoucího klubu nebo trenéra bude moci vytvářet zápasy, upravovat zápas a zadávat výsledné skóre. Dalším oprávněním pro tyto uživatele bude možnost vytvářet soupisky týmů a přidávat události k zápasu. Jak bylo zmíněno v kapitole 4.5, trenér bude mít na rozdíl od vedoucího klubu možnost vytvářet sestavu graficky.

4.7 Diagram případů užití

Diagram případů užití (Use Case Diagram) zobrazuje chování systému a také zobrazuje funkcionalitu z pohledu uživatele. Účelem diagramu je zobrazit to, co se od výsledného systému očekává. Ukazuje tedy to, co má systém umět, ale neříká, jak to bude dělat. [22] Diagram případů užití je výsledkem analýzy a specifikace požadavků. Většinou je to první diagram, který se při návrhu informačního systému vytváří. Skládá se z případů užití (use case) a aktérů (actors) a vztahů mezi nimi. [22]

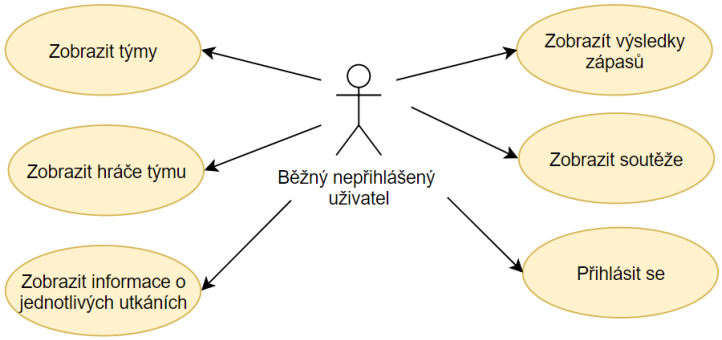
V následujících kapitolách bude zobrazen diagram případů užití pro jednotlivé role systému.

4.7.1 Nepřihlášený uživatel

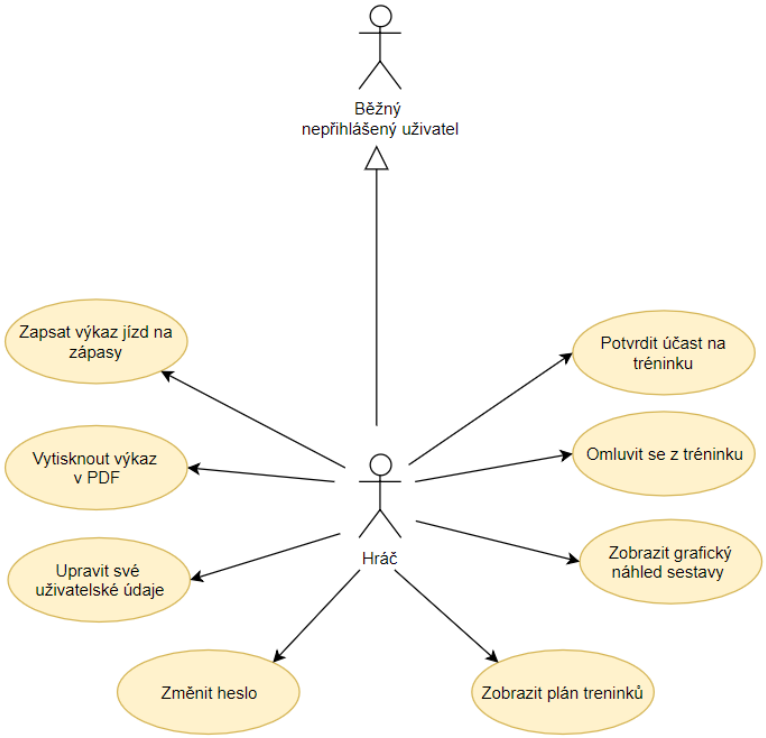
Jak bylo zmíněno v předchozích odstavcích nepřihlášený uživatel příliš mnoho oprávnění mít nebude. Tento uživatel si bude moci například zobrazovat výsledky zápasů včetně podrobností o konkrétním zápasu. Pokud mu budou poskytnuty přihlašovací údaje, bude se moci přihlásit do systému. Všechny případy užití nepřihlášeného uživatele jsou zobrazeny na obrázku 4.1.

4.7.2 Role hráč

Uživatel v roli hráč bude moci vykonávat stejné akce jako nepřihlášený uživatel. Navíc bude mít oprávnění evidovat si výkazy jízd, potvrzovat účast na tréninku, případně se z tréninku omlouvat. Hráč si také bude moci zobrazit grafickou sestavu konkrétního zápasu. Veškerá oprávnění hráče jsou zobrazena na obrázku 4.2.



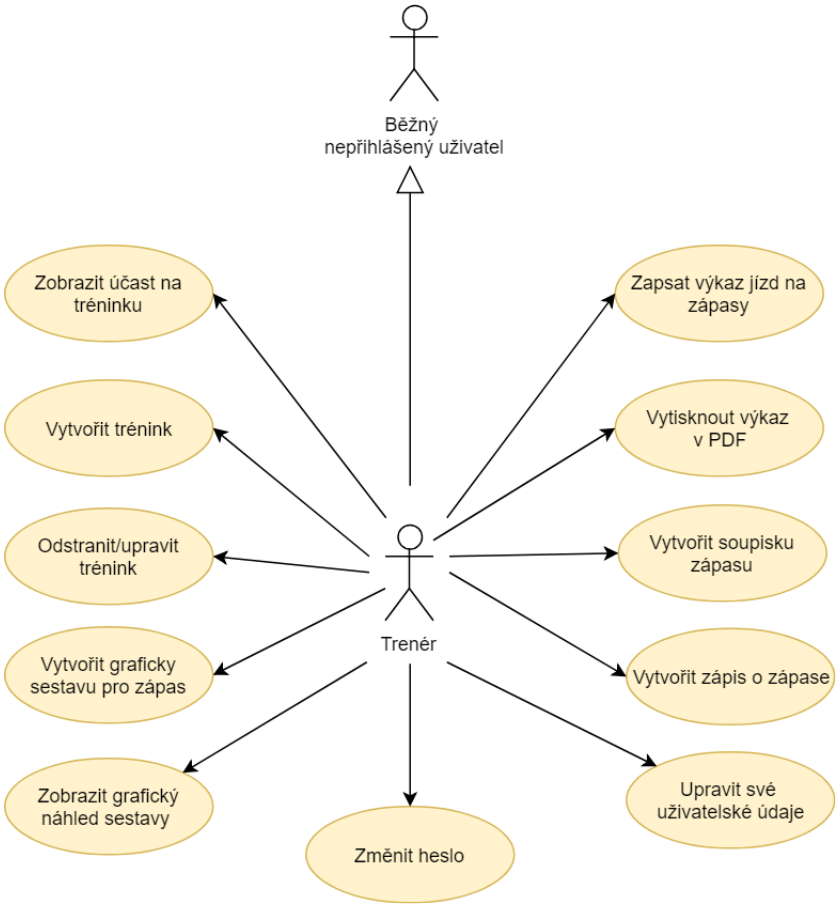
Obrázek 4.1: Diagram případů užití - nepřihlášený uživatel



Obrázek 4.2: Diagram případů užití - role hráč

4.7.3 Role trenér

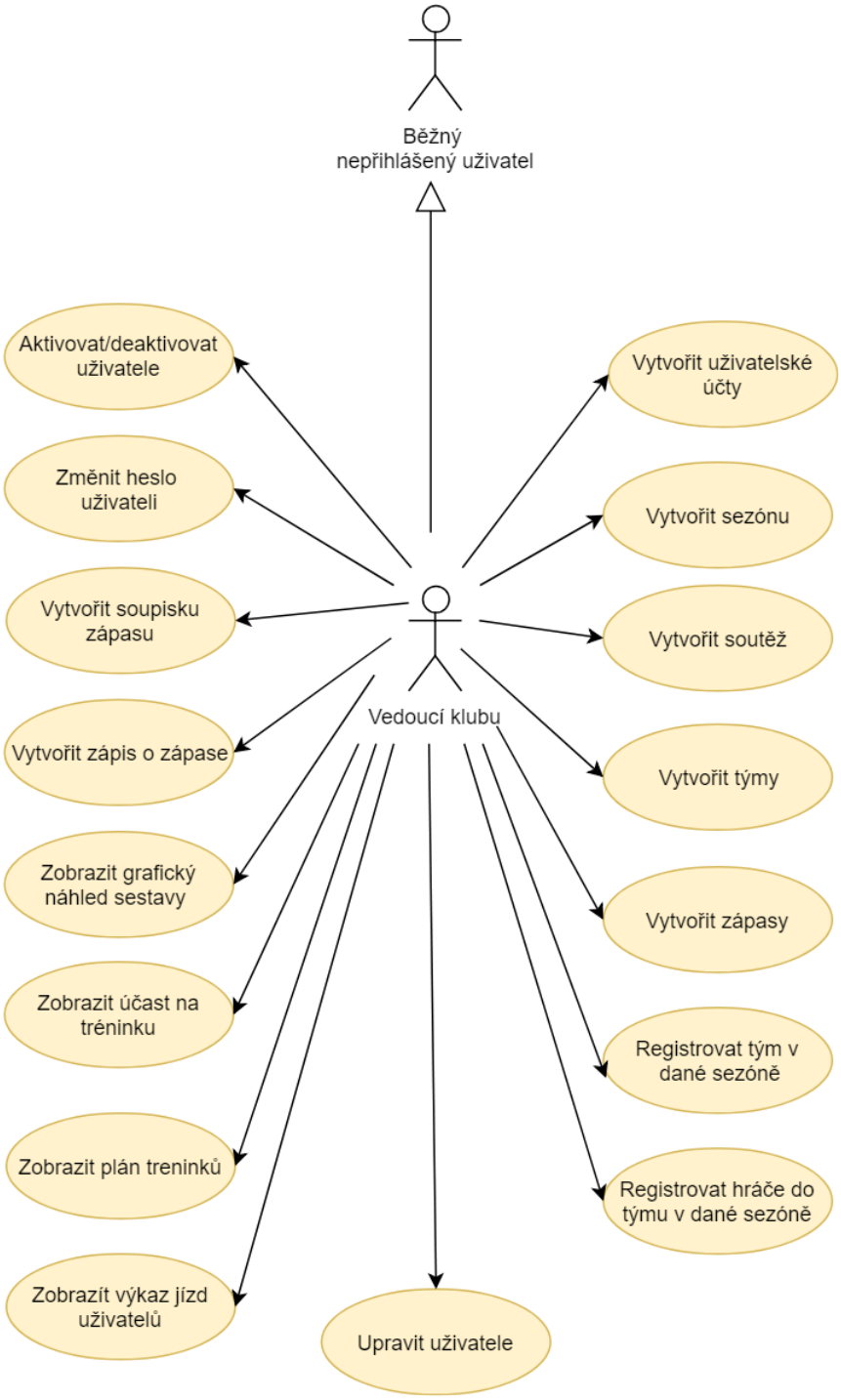
Jak je možné vidět na obrázku 4.3, role trenér bude mít v systému již více oprávnění. Stejně jako hráč, bude moci i trenér provádět v systému to, co běžný nepřihlášený uživatel. Dalším oprávněním, které bude mít trenér na rozdíl od hráče, je možnost vytvářet tréninky, případně je editovat. Navíc bude mít trenér i možnost zapisovat výsledky u zápasů a vytvářet soupisky (sestavy 11 hráčů). Velmi důležitým oprávněním trenéra bude také vytváření grafické sestavy u jednotlivých zápasů.



Obrázek 4.3: Diagram případů užití - role trenér

4.7.4 Role vedoucí klubu

Vedoucí klubu, jak už název této role napovídá, bude mít v systému nejvíce oprávnění. Můžeme říci, že vedoucí klubu bude vykonávat roli administrátora systému. Z toho vyplývá, že bude mít na starost veškeré uživatelské účty, bude vytvářet sezóny, týmy, hráče apod. Seznam všech akcí, které bude moci vedoucí klubu v systému vykonávat je zobrazen na obrázku 4.4.



Obrázek 4.4: Diagram případů užití - role vedoucí klubu

Kapitola 5

Návrh systému

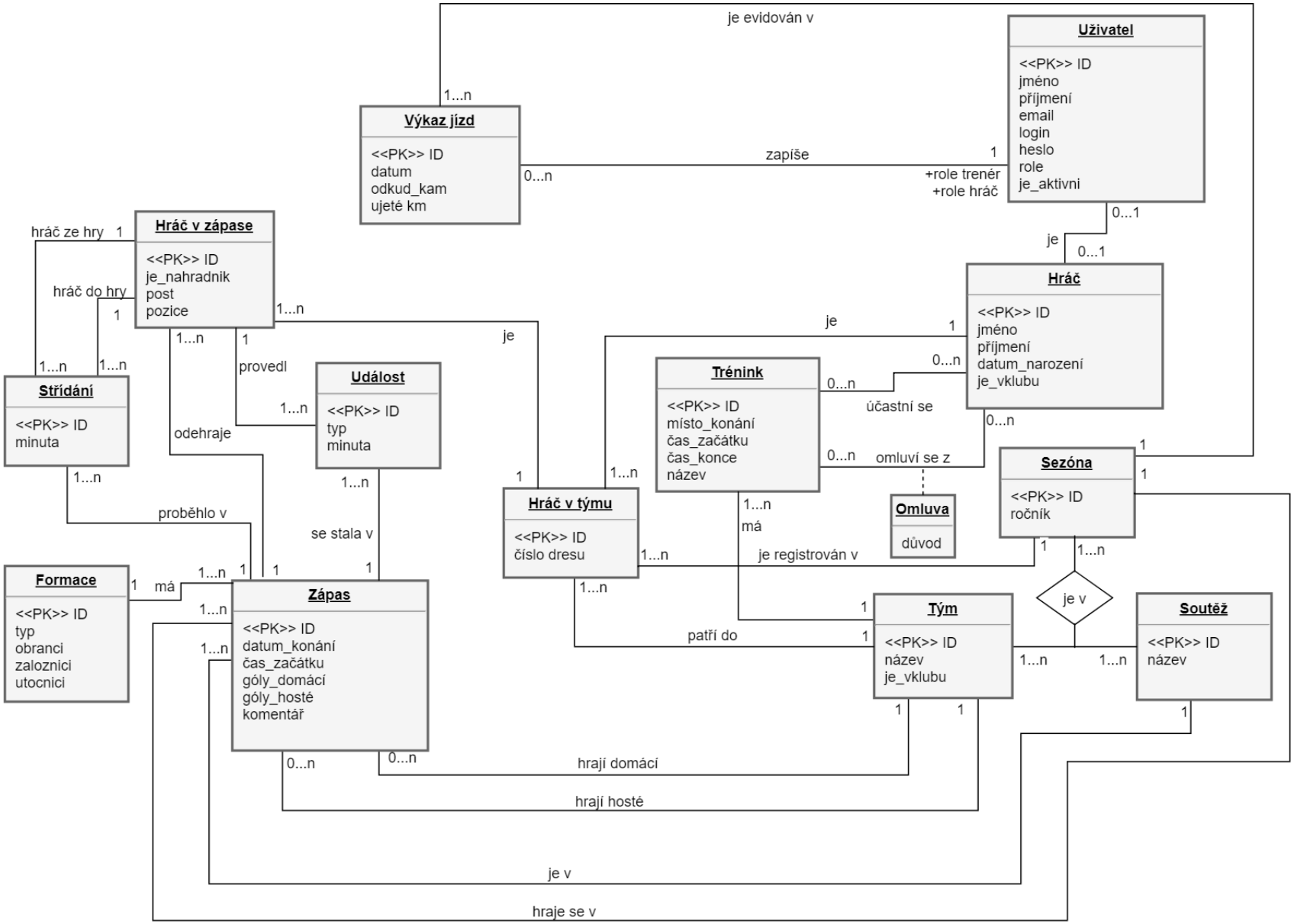
Tato část se zaměřuje na datový návrh aplikace, který byl vytvořen na základě požadavků definovaných v přechozí kapitole. Pro datový návrh aplikace byl využit ER diagram, který bude v této části popsán spolu s některými zajímavými entitními množinami. Dále zde bude popsán i návrh uživatelského rozhraní, zejména rozložení prvků na obrazovce, položky menu apod.

5.1 ER diagram

ER diagram (*Entity-Relationship diagram*) je konceptuální model, který se využívá pro návrh a znázornění dat v systému a vztahů mezi nimi. Mezi základní prvky ER diagramu patří [30]:

- **Entita** - jedná se o objekt reálného světa, o kterém chceme uchovávat data v databázi
- **Entitní množina** - množina entit stejného typu mající stejné vlastnosti neboli atributy
- **Atributy** - vlastnosti entit
- **Vztahy** - vyjadřují asociace mezi několika entitami
- **Vztahové množina** - množina vztahů téhož typu

Na obrázku 5.1 bude zobrazen ER diagram vytvářeného systému a v následujících sekcích budou popsány jednotlivé entitní množiny a vztahy mezi nimi.



Obrázek 5.1: ER diagram

5.1.1 Uživatel

Entitní množina uživatel slouží k ukládání dat o uživatelích. Jak bylo zmíněno v kapitole 4, v systému vystupují 3 role uživatelů - vedoucí klubu, trenér a hráč. Z toho vyplývá, že jedním z atributů této entitní množiny je právě *role*. V závislosti na roli pak mají uživatelé různá oprávnění.

Dále je u uživatele potřeba ukládat *jméno*, *příjmení*, *email*, *login*, který musí být pro každého uživatele unikátní a *heslo*. Posledním atributem je atribut *je_aktivni*, který slouží ke změně aktivity uživatele. Pokud je nastaven na hodnotu 0, uživatel se nemůže do systému přihlásit. Využití tohoto atributu je zejména v případě, kdy klubový hráč opustí klub a tímto se mu zamezí přístup do systému.

5.1.2 Sezóna - Soutěž - Tým

Entitní množina sezóna má pouze jeden atribut, kterým je *ročník* a je ve tvaru např. 2018-2019. Soutěž má taktéž jen jeden atribut - *název*, který musí být unikátní. Třetí entitní množinou, kterou je potřeba v této části zmínit je tým. U týmu je podobně jako u soutěže potřeba uchovávat atribut *název* a také to, zda se jedná o klubový tým nebo ne. K tomu slouží atribut *je_vklubu*. U klubových týmů jsou potřeba pokročilé funkce, jako je např. evidence tréninků, proto bylo nutné přidat tento atribut. Neklubové týmy slouží především pro evidenci dat u zápasů. Vzhledem k tomu, že každý zápas má jeden hostující a jeden domácí tým, je potřeba ukládat data o neklubových týmech, jinak by nebylo zápasy vůbec možné vytvářet.

Tyto 3 entitní množiny bylo potřeba vzájemně propojit, protože spolu souvisí. Tým může být ve více sezónách a v každé sezóně může být v jiné soutěži, například pokud postoupí z prvního místa do vyšší soutěže. Soutěž může hrát více týmů a zároveň je možné zaregistrovat soutěž ve více sezónách. Z toho vyplývá, že entitní množiny sezóna, soutěž a tým mezi sebou mají ternární vztah s kardinalitami M. V kapitole 5.2 bude popsáno, jak se tato situace řeší.

5.1.3 Hráč - Hráč v týmu - Hráč v zápase

Hlavní podstatou entitní množiny hráč je evidence jednotlivých hráčů. U hráče je potřeba uchovávat několik atributů - *jméno*, *příjmení*, nepovinný atribut *datum narození*. Podobně jako u týmu je potřeba rozlišit, zda se jedná o klubového hráče, což zajišťuje atribut *je_vklubu*. Význam tohoto atributu bude více popsán v kapitole 6.3.

Entitní množina hráč v týmu má pouze jeden atribut - *číslo dresu*, které má hráč v týmu vždy stejné pro danou sezónu. Hlavním záměrem pro vznik této entitní množiny bylo zamezit duplicitám, které by vznikaly, pokud by zůstala pouze množina hráč. Vzhledem k tomu, že hráč může být v různých sezónách v jiném týmu, bylo potřeba tuto množinu zavést. Hráč je tedy zaregistrován v sezóně do týmu, má příslušné číslo dresu a tato data jsou uchována v entitní množině hráč v týmu.

Hráč v zápase slouží především k vytvoření soupisky a k evidenci událostí a střídání, které byly provedeny v konkrétním zápase. U této entitní množiny je důležitý atribut *je_nahradnik*, který slouží k rozlišení 11 hráčů hrajících v základní sestavě a hráčů, kteří budou připraveni na střídání. Dalšími atributy jsou *post* a *pozice*. *Pozice* je zavedena kvůli grafickému editoru a slouží pro zapamatování pozice hráče na hřišti (viz kapitola 6.9).

5.1.4 Zápas

Tato entitní množina slouží k evidenci výsledků zápasů. Zápas je vždy vytvořen v dané sezóně a pro konkrétní soutěž. O zápase je potřeba evidovat *datum konání*, *čas začátku*, skóre domácího týmu, které udává atribut *góly domácí*, hostujícího týmu atribut *góly hosté* a posledním atributem je *komentář*, který je nepovinný.

Pro každý zápas jsou také potřeba 2 týmy - domácí a hostující. Proto z entitní množiny zápas vedou 2 vazby k množině tým. Jde o vazby 1:N, tým se může zúčastnit více zápasů a zápas má vždy jen jeden domácí a hostující tým. Dále se zápasem také souvisí entitní množiny formace, událost a střídání, které budou popsány v dalších kapitolách.

5.1.5 Událost a Střídání

Vzhledem k tomu, že u zápasu je potřeba evidovat události, které se staly, byla zavedena entitní množina událost. Jsou pro ni důležité 2 atributy: *typ* události, kterým se myslí např. červená karta a *minuta*, v které tato událost nastala.

Událost vždy provede konkrétní hráč, proto je tato entitní množina spojena s množinou hráč v zápase a to ve vztahu 1:N. Hráč může provést několik událostí během zápasu a událost je vždy vztažena k jednomu hráči.

Entitní množina střídání, jak už název napovídá, slouží k uchování údajů o proběhnutých střídáních v zápase. Podobně jako událost má atribut *minuta*, který udává, kdy střídání proběhlo.

Pro střídání jsou potřeba 2 hráči, jeden vstupuje do hry a druhý z ní odchází. Z tohoto důvodu vedou 2 vazby k entitní množině hráč v zápase a stejně jako u události se jedná o vazby 1:N.

5.1.6 Formace

Jak bylo zmíněno v sekci 5.1.4, je u zápasu potřeba evidovat údaje o formaci. Tato entitní množina vznikla z požadavku na grafický editor sestav u zápasu. Klubové hráče, kteří jsou na soupisce, je možné umístit do grafické sestavy v závislosti na zvolené formaci, např. 4-4-2. U zápasu se uloží typ vybrané formace a následně je možné si tuto formaci zobrazit.

Atributy, které je potřeba uchovávat u této entitní množiny jsou: *typ* formace a následně počet *obránců*, *záložníků* a *útočníků* této formace.

5.1.7 Trénink

Tato entitní množina slouží k ukládání dat o trénincích. U tréninku je potřeba evidovat *název* (např. zda se jedná o trénink či přípravné utkání), *místo konání*, dále *čas začátku* a *čas konce*, aby hráči věděli, jak dlouho bude trénink trvat.

Trénink se vždy týká konkrétního týmu a hráči mají možnost se z tréninku omluvit nebo potvrdit účast. Z tohoto důvodu jsou mezi entitní množinou hráč a trénink 2 vazby M:N.

5.1.8 Výkaz jízd

Poslední entitní množinou, kterou je potřeba zmínit, je výkaz jízd. Vzhledem k požadavku uvedeném v kapitole 4, je nutné ukládat informace o výkazech jízd hráčů a trenérů. Každý výkaz jízd má následující atributy - *datum*, udávající datum cesty, *odkud-kam* popisující, odkud a kam se jelo a *ujeté km*, který ukládá celkový počet ujetých kilometrů u dané jízdy.

Tato entitní množina je spojena vazbou 1:N s množinou uživatel a pouze uživatel v roli trenéra nebo hráče si může výkazy jízd evidovat. Protože je také potřeba odlišit výkazy jízd v jednotlivých sezónách, je tato množina spojena vazbou 1:N i s množinou sezóna.

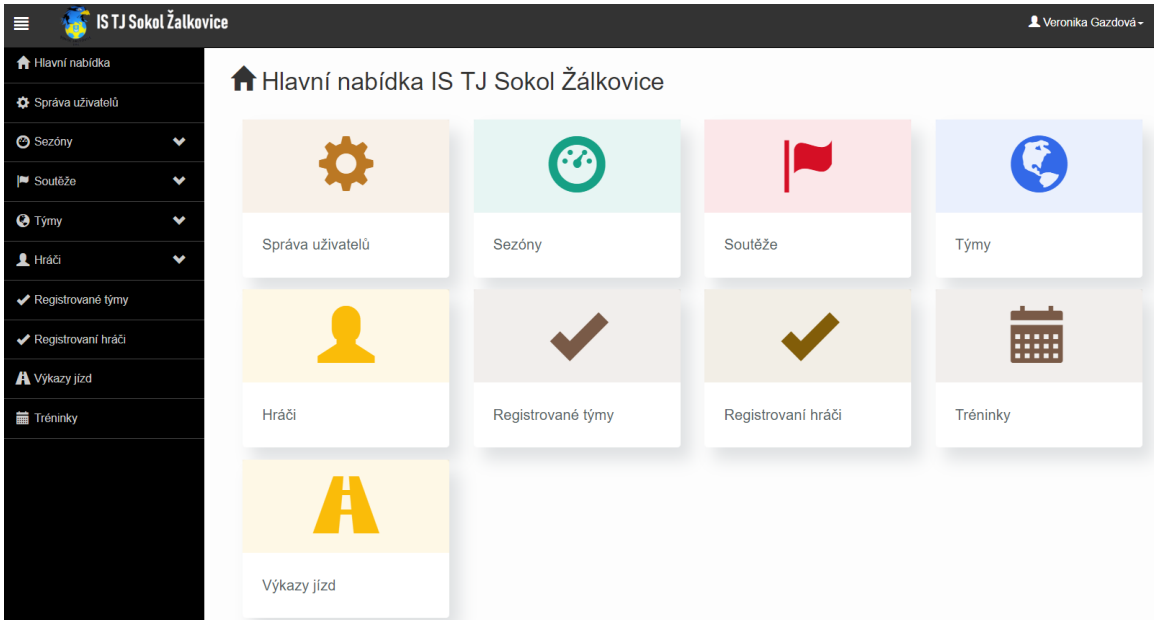
5.2 Transformace ER diagramu na tabulky relační databáze

Při transformaci ER diagramu na relační databázi dochází k převedení všech entitních množin i s primárními klíči na tabulky relační databáze. Vztah 1:1, který je v tomto případě pouze u množin hráč-uživatel se převede tak, že do jedné z tabulek se přidá cizí klič odkazující se na primární klíč druhé tabulky. Podobným způsobem se převede i vztah 1:N, jen s tím rozdílem, že cizí klíč se přidá do tabulky, u které je kardinalita N.

Vztahy M:N vyžadují při převodu vznik tzv. vazební tabulky. Tyto tabulky budou obsahovat ve většině případech 2 atributy. Pro spojení tabulek sezóna-soutěž-tým bude obsahovat vazební tabulka atributy 3. Tyto atributy budou dohromady tvořit jeden složený primární klíč, aby se zamezilo vytvoření duplicitních kombinací těchto atributů. Každý z klíčů bude zároveň cizím klíčem, který se odkazuje na jednu z tabulek původního M:N vztahu.

5.3 Návrh uživatelského rozhraní

Při návrhu uživatelského rozhraní je potřeba myslet na to, aby bylo pro uživatele co nejjednodušší a přehledné. V tomto případě byl využit čistý styl s použitím několika málo barev. Na obrázku 5.2 můžete vidět náhled uživatelského rozhraní zobrazující hlavní nabídku pro vedoucího klubu.



Obrázek 5.2: Náhled hlavní nabídky pro vedoucího klubu

Z obrázku je také patrné, jak vypadá rozložení stránky. V horní části se nachází navigační menu, v levé části je postranní panel a v prostřední části můžeme vidět obsah (tělo) stránky. Tyto části budou v následujících kapitolách popsány.

5.3.1 Navigační menu

Navigační menu je umístěno, jak bylo zmíněno výše, v horní části stránky a je zde fixované. Na levé straně je umístěno tlačítko, které slouží pro zobrazení nebo skrytí postranního panelu. Vedle tohoto tlačítka se nachází logo webové stránky. Pokud uživatel klikne na toto logo, dojde k přesměrování stránky do hlavní nabídky.

V pravé části navigačního menu se přihlášeným uživatelům zobrazuje jejich jméno a po kliknutí se rozbalí nabídka s dalšími možnostmi, např. změnit heslo či odhlásit se. Nepřihlášený uživatel má v této části navigačního menu pouze možnost na přihlášení se do systému. Zároveň pro nepřihlášené uživatele není zobrazen postranní panel a ve střední části navigačního menu mají zobrazeny pouze 2 odkazy - domů a sezóny (viz obrázek 5.3).

Obrázek 5.3: Náhled hlavní nabídky pro nepřihlášeného uživatele

5.3.2 Postranní panel

Postranní panel je umístěn v levé části webové stránky, hned pod navigačním menu. Jak bylo zmíněno výše, tento panel je zobrazen pouze pro přihlášené uživatele. Pokud chce uživatel využívat celou šířku stránky, může si tento panel skrýt pomocí tlačítka nacházejícího se v navigačním menu. Na zařízeních s malým displejem je tento panel automaticky skryt.

Kapitola 6

Implementace

Tato kapitola bude popisovat nejdůležitější části implementace informačního systému fotbalového klubu. Implementace vychází z analýzy a specifikace požadavků (viz kapitola 4) a z následného návrhu systému (viz kapitola 5).

Nejprve bude popsána adresářová struktura tohoto informačního systému, dále bude vysvětlen přístup k databázi. Následně je v této kapitole popsáno vytváření uživatelů spolu s přihlašováním a správou uživatelů. Také je zde zmíněna implementace nejdůležitějších částí systému jako je organizace tréninků, grafický editor sestav, evidence výkazů jízd a další.

6.1 Adresářová struktura

Adresářová struktura vychází z doporučené adresářové struktury Nette Frameworku, známé jako sandbox.

V následujícím textu budou rozebrány jednotlivé adresáře a některé důležité soubory. Kořenový adresář obsahuje 5 složek, které budou popsány níže a 4 soubory, nejdůležitější z nich jsou:

- **.htaccess** - tento soubor slouží pro úpravu vlastností serveru, je zde nastaveno, aby požadavek na zobrazení webu směřoval do složky `www/`
- **composer.json, composer.lock** - jedná se o konfigurační soubory pro nástroj Composer

app

Tato složka obsahuje veškeré zdrojové kódy aplikace, z tohoto důvodu je důležité, aby se do této složky nedostala žádná třetí strana. K tomu slouží soubor `.htaccess`, ve kterém se nastaví úplný zákaz jakéhokoliv zobrazení této složky z prohlížeče.

Důležitým adresářem je adresář `config`, který obsahuje 2 konfigurační soubory:

- **.config.local.neon** - tento soubor obsahuje nastavení přístupu k databázi (název host serveru, název databáze, uživatelské jméno a heslo)
- **composer.json, composer.lock** - obsahuje nastavení projektu

Dále se v adresáři `app` nachází složka `model`, jedná se o modelovou vrstvu a její třídy. Následuje složka `presenters`, která, jak už název napovídá, obsahuje, presentery a také

složku se šablonami. Poslední složkou je složka `router`, která obsahuje třídu routerů - tedy třídu, která zajišťuje překlad url adres.

Je potřeba zmínit i zaváděcí soubor `bootstrap.php`, který zajišťuje spuštění celého frameworku.

log

Tato složka slouží k uložení chybových stavů a errorů, které mohou při běhu aplikace nastat.

temp

Zde Nette ukládá dočasné soubory.

vendor

Tento adresář obsahuje veškeré knihovny Nette Frameworku včetně knihoven nainstalovaných pomocí Composeru.

www

Jedná se o veřejnou složku, ve které jsou soubory dostupné přes prohlížeč. Obsahuje adresář s css styly (`css`), s obrázky (`images`) a s JavaScriptovými soubory (`js`).

Důležitým souborem je soubor `index.php`, který se stará o spuštění celé aplikace. Soubor `favicon.ico` obsahuje ikonu aplikace.

6.2 Přístup k databázi

Pro správný chod tohoto informačního systému je nezbytný přístup k databázi a provádění dotazů nad daty v databázi. Pokud si uživatel chce zobrazit výsledky zápasu nebo například vytvořit uživatele, je k tomu potřeba právě databáze.

Nette poskytuje metody, které komunikaci s databází velmi zjednodušují. Nejprve je však potřeba nakonfigurovat přístup k databázi. O toto se stará soubor `config.local.neon`, který byl zmíněn již výše v kapitole 6.1. Následně je potřeba pro připojení k databázi vytvořit instanci třídy `Nette\Database\Connection` a v konstruktoru ji uložit do privátní proměnné `$database`. Toto se nachází ve složce `model` v souboru `Model.php`, kde je navíc implementován i přístup k jednotlivým databázovým tabulkám.

Všechny presentery pak využívají tuto třídu pro přístup k databázi. Následně je možné nad tabulkami provádět operace SELECT, INSERT, UPDATE či DELETE. Pokud bychom například chtěli zobrazit všechny klubové týmy, bude dotaz vypadat následovně:

```
$this->template->tym = $this->teamModel->getTeams()
                              ->where('je_vklubu', 1);
```

6.3 Vytváření uživatelů

Vzhledem k tomu, že tento informační systém je uzavřený, není potřeba mít registrační formulář. Uživatelské účty může vytvářet pouze vedoucí klubu v části systému *Správa uživatelů*. O samotné vytváření uživatelů se stará formulář `createComponentUserForm()` v souboru `UserPresenter.php`. V tomto formuláři zadá vedoucí klubu jméno, příjmení, email,

heslo a roli uživatele. Po stisknutí tlačítka uložit se zavolá metoda `userFormSucceeded(Form $form)`, která provede vytvoření uživatele a uložení dat do databáze.

Vzhledem k tomu, že každý uživatel musí mít jedinečné přihlašovací jméno, dochází k tvorbě loginu automaticky. Přihlašovací jméno je vždy vytvořeno ve tvaru:

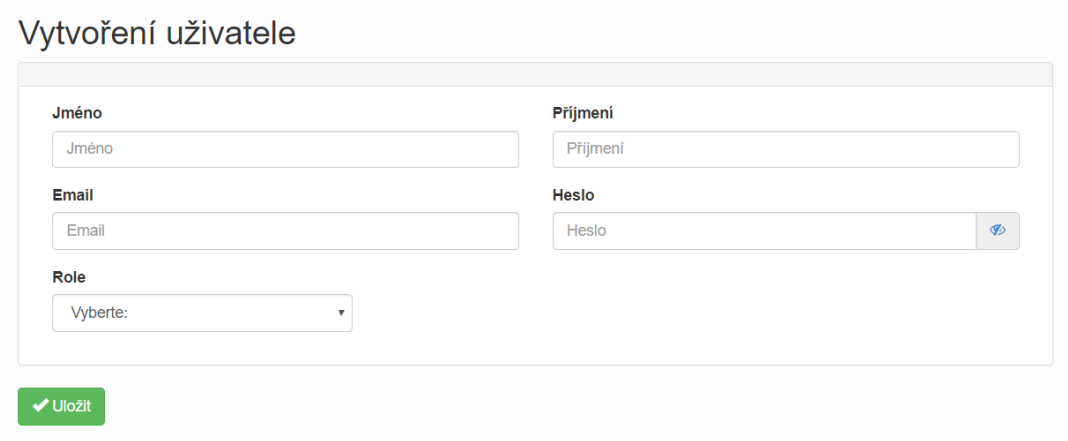
```
$login = Strings::firstLower(Strings::toAscii($values->prijmeni)) .
Strings::firstLower(Strings::toAscii($values->jmeno)) .
Random::generate(2, '0-9');
```

tzn. příjmení, jméno a následně jsou přidány 2 číselné znaky. Tímto se zamezí vzniku duplicitních uživatelských jmen.

Z důvodu bezpečnosti také není vhodné uchovávat hesla v databázi v otevřené textové podobě. Proto je heslo do databáze uloženo jako hash. O tuto funkcionalitu se stará statická třída Nette `Nette\Security\Passwords`. Hash hesla je pak následně vytvořen metodou `hash($heslo)`, která vygeneruje otisk hesla pomocí algoritmu bcrypt.

Pokud je při vytváření uživatele zvolena role hráč, je také automaticky vytvořen hráč se jménem a příjmením zadaným ve formuláři a hodnotou ve sloupci `je_vklubu` nastavenou na 1.

Na obrázku 6.1 je možné vidět formulář na vytváření uživatelů.



Obrázek 6.1: Formulář na vytvoření uživatele

6.4 Přihlašování, autentizace a autorizace

Pro pokročilé funkce systému je potřeba se přihlásit. Při přihlašování zadává uživatel uživatelské jméno (login) a heslo. O samotný proces přihlašování se pak stará třída `Authenticator`, která obsahuje metodu `authenticate(array $credentials)`, kde `$credentials` obsahuje jméno a heslo. Tato metoda nejprve zjistí, zda zadaný uživatel existuje, následně dochází k porovnání zadaného hesla s otiskem pomocí metody `verify($password, $row->heslo)`. Pokud se hesla neshodují, vypíše se uživateli chybová hláška, že zadal špatné heslo.

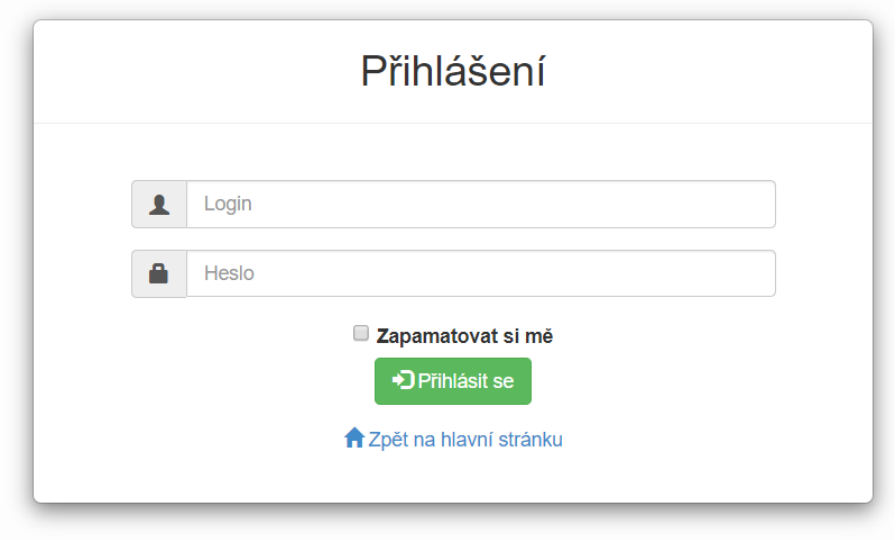
Pokud proběhlo přihlášení úspěšně, volá se ještě metoda `needsRehash($row->heslo)`, která kontroluje, zda je potřeba heslu znovu spočítat hash. V neposlední řadě také ještě dochází ke kontrole, zda má daný uživatel aktivovaný uživatelský účet. Pokud uživatel

není aktivní, nepodaří se mu do systému přihlásit. Více bude využití aktivace a deaktivace uživatelských účtů popsáno v kapitole 6.5.

Návratovou hodnotou metody `authenticate` je identita uživatele, která nese soubor informací o uživateli (id uživatele, login, email, role atd.)

Jakmile je uživatel autentizován dochází k autorizaci, tzn. ověření, že má uživatel oprávnění k určité operaci. V tomto informačním systému se autorizace provádí pomocí podmínek, kde se kontroluje, zda je daný uživatel přihlášen a zda má příslušnost k roli, která má oprávnění danou operaci provádět. V těchto podmínkách se využívá proměnná `$user`, která je objektem třídy `Nette\Security\User`.

Na obrázku 6.2 je možné vidět přihlašovací formulář.



Obrázek 6.2: Přihlašovací formulář

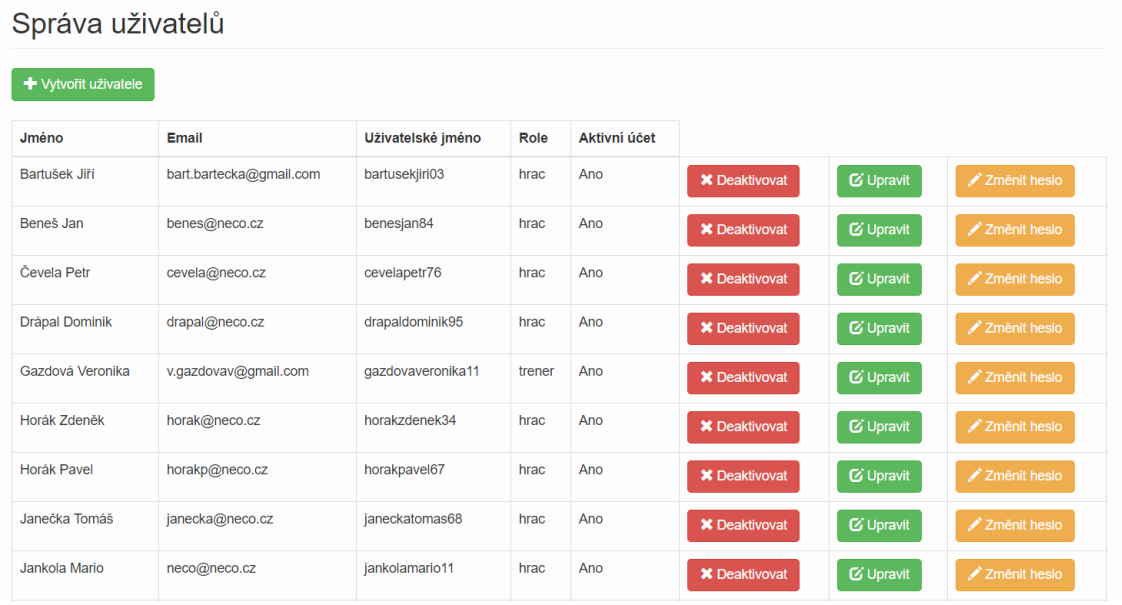
6.5 Správa uživatelů

Tato část systému je přístupná pouze uživateli v roli vedoucí klubu - administrátorovi systému. Je zde vypsán seznam všech uživatelů včetně jejich rolí a informace o tom, zda mají aktivní či neaktivní účet (viz obrázek 6.3). O zobrazení těchto dat se stará metoda `renderDefault()` v souboru `UserPresenter.php`.

V této sekci je samozřejmě možné vytvářet uživatele, což bylo popsáno výše v kapitole 6.3. Zároveň je také možné u každého uživatele upravit údaje. Při stisku tlačítka na úpravu uživatelských údajů se vyvolá akce `actionEdit($id)`, která nejprve ověří, že uživatel s daným `id` existuje a následně dojde k zobrazení formuláře `editUserForm` na úpravu údajů.

Obdobným způsobem lze také každému uživateli změnit heslo, například v situaci, že by uživatel své stávající heslo zapomněl.

Důležitou roli zde také hraje možnost aktivovat či deaktivovat uživatele. Tato funkcionalita bude využita hlavně v případě, pokud by nějaký z klubových hráčů nebo trenérů opustil klub. V této situaci je potřeba tomuto uživateli zamezit přístup do systému, což lze právě tlačítkem deaktivovat. Po stisknutí tohoto tlačítka se pošle AJAXový požadavek do presen-



Obrázek 6.3: Správa uživatelů

teru `UserPresenter.php` a vykoná se metoda `handleChangeActivation($id_uzivatele, $jeaktivni)`. Tato metoda změní hodnotu ve sloupci `je_aktivni`.

6.6 Formuláře

O formulářích již byla řeč výše, z toho vyplývá, že jsou důležitou součástí systému. Používají se na vytváření a editování dat, která je potřeba pro správný běh systému ukládat.

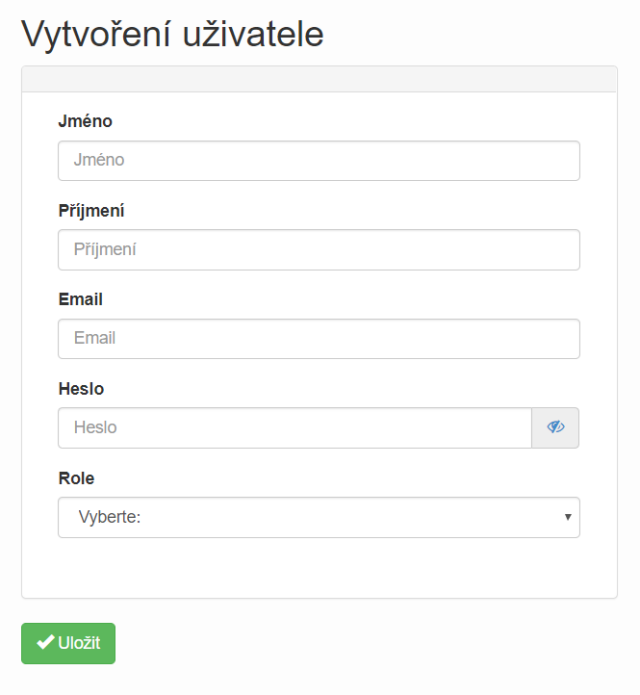
Podstatnou vlastností formulářů je, aby jejich vzhled byl pro uživatele přehledný a také to, aby byly responzivní, tj. přizpůsobitelné zařízením s různou velikostí displeje. Obrázek 6.4 znázorňuje formulář na vytvoření uživatele, který byl již zobrazen v kapitole 6.3, přizpůsobený na zařízení s menším displejem. Pro zajištění této vlastnosti byly formuláře vytvářeny s použitím knihovny Bootstrap (viz kapitola 3.4).

Při vyplnění formuláře uživatelem je také potřeba kontrolovat správnost zadaných dat. Pokud bude chtít uživatel formulář odeslat a nebude mít data správně vyplněna, bude na tuto skutečnost upozorněn chybovou hláškou. Validace se u formulářů v Nette provádí poměrně jednoduchým způsobem, viz následující ukázka:

```
$form->addText('rocnik', 'Rocnik souteze:')
        ->setRequired('Vlozte prosim rocnik souteze!')
        ->addRule(Nette\Forms\Form::PATTERN, 'Toto neni validni sezona,
        zadejte sezonu ve tvaru napr.: 2018-2019.',
        ' *([0-9]{4} *)\-([0-9]{4} *)');
```

Pokud by uživatel odeslal formulář a nezadal ročník soutěže, bude na tuto skutečnost upozorněn, navíc je potřeba vyplnit ročník ve správném tvaru uvedeném v pravidle `addRule`.

V neposlední řadě je také potřeba zmínit doplněk, který byl pro tvorbu formulářů využit. Jedná se o knihovnu `DateInput`. Díky této knihovně je možné ve formulářích zadávat



Obrázek 6.4: Ukázka responzivního formuláře

jednoduše údaje jako je datum a čas, což je využito například při tvorbě zápasu nebo tréninku. Vzhled tohoto doplňku je možné vidět na obrázku 6.5

Obrázek 6.5: Doplněk DateInput

6.7 Tréninky

Důležitou součástí systému je možnost evidovat tréninky. Pro přehledné zobrazení tréninků byl použit plugin FullCalendar, který byl představen v kaptiole 3.6.

Vytvořit trénink může pouze uživatel v roli trenéra. Pro vytvoření tréninku je zobrazen formulář, kde je potřeba vyplnit název, tým, kterého se trénink týká, a kdy a kde se trénink

koná. Po odeslání formuláře kliknutím na tlačítko uložit dojde k vytvoření tréninku metodou `createTrainingFormSucceeded(Form $form)`. Tato metoda také zajistí odeslání emailu všem hráčům týmu s informací o datumu a času konání tréninku. O samotné vytvoření a odeslání emailu se starají třídy `Nette\Mail\Message` a `Nette\Mail\SendmailMailer`.

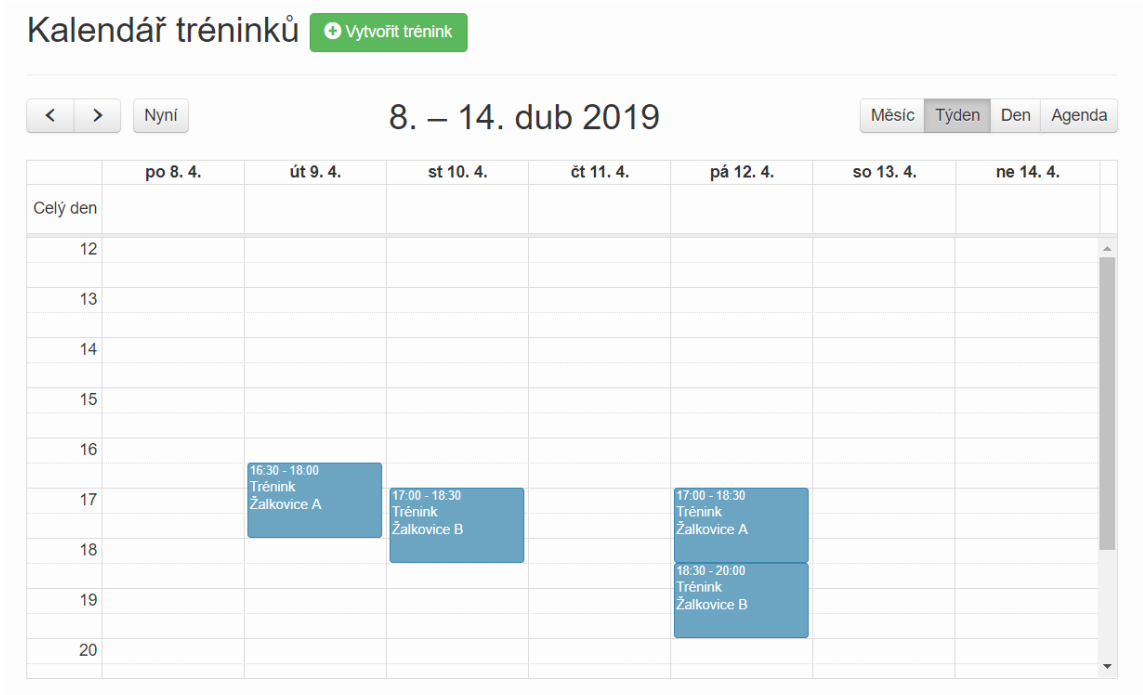
Aby mohl být následně trénink v kalendáři zobrazen, je potřeba nejprve získat data o tréninku, jako je například *id tréninku, název, čas začátku* a *čas konce*. O toto se stará metoda `getTrenink()`, která vrací tato data uložená v poli. Tato metoda se následně zavolá v metodě `handleLoadTreninky()`, která je zobrazí v kalendáři.

Trenér a vedoucí klubu mají navíc možnost trénink editovat pomocí metody *drag and drop*. Díky tomu mohou událost přesunout na jiný čas. Před uložením změny času se nejprve zobrazí hláška, kde je uživatel tázán, zda chce opravdu provést tuto změnu.

Po kliknutí na konkrétní trénink dojde k přesměrování URL na detail tréninku. Zde má trenér možnost zobrazit si seznam hráčů, kteří potvrdili účast na tréninku, případně hráčů, kteří se omluvili. V detailu tréninku je pro trenéra možnost provést i zrušení tréninku. Rušení tréninku provádí metoda `handleSmazatTrenink($id)`. Obdobně jako při vytvoření tréninku dojde i při jeho zrušení k odeslání emailu hráčům týmu s informací o tom, že trénink byl zrušen.

Pokud klikne na konkrétní trénink hráč, stejně jako trenérovi se mu zobrazí detail tréninku. Jestliže se jedná o trénink týmu, ve kterém je hráč registrován, má zde možnost potvrdit účast nebo se z tréninku omluvit a také může vidět seznam hráčů, kteří se tréninku budou účastnit, případně omluvených hráčů. Při omluvě je hráči zobrazen formulář, ve kterém je také nutné uvést důvod neúčasti. V případě, že si hráč zobrazí detail tréninku jiného týmu, má možnost vidět pouze, kde se trénink koná.

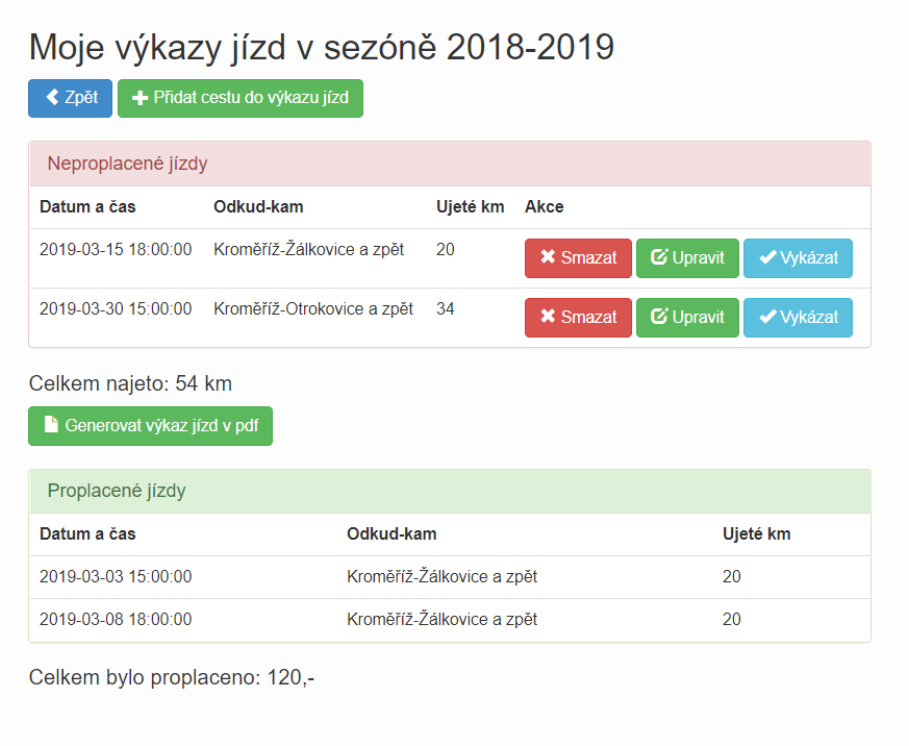
Na obrázku 6.6 je možné vidět, jak kalendář tréninků vypadá. Kalendář se ve výchozím stavu zobrazuje jako týdenní.



Obrázek 6.6: Kalendář tréninků

6.8 Evidence výkazů jízd

Další neméně důležitá část tohoto informačního systému je možnost evidovat si výkazy jízd na zápasy, případně tréninky. Systém umožňuje nejen evidenci jednotlivých cest, ale také vygenerování výkazu jízd v PDF podle vzoru šablony na vykázání jízd, kterou TJ Sokol Žalkovice používá. Jak bylo zmíněno v kapitole 3.2.4, byl pro tvorbu PDF použit doplněk PDFResponse. Evidence cest slouží především pro hráče a trenéry, vedoucí klubu má pouze možnost zobrazit si výkazy jízd jednotlivých uživatelů.



Obrázek 6.7: Zobrazení výkazu jízd pro uživatele v roli hráč nebo trenér

Pokud si chce uživatel přidat cestu do výkazu jízd, zobrazí se mu formulář, kde je potřeba vyplnit datum a čas cesty, odkud a kam jel a počet ujetých kilometrů. Po stisknutí tlačítka uložit dojde k odeslání formuláře a metoda `createVykazFormSucceeded(Form $form)` provede uložení vyplněných dat do databáze. Následně je uživateli jízda zobrazena v tabulce *Neproplacené jízdy*. Každou jízdu je možné upravit, vykázat a případně smazat (viz obrázek 6.7).

Záleží na každém uživateli, kolik cest si bude chtít vykázat a tedy i nechat proplatit, je to zcela individuální. Před samotným vykázáním jízdy je potřeba vygenerovat si PDF s výslednou žádostí na proplacení jízd. Po stisknutí tlačítka generovat výkaz jízd v PDF se uživateli zobrazí výsledné PDF. O tuto funkcionalitu se stará metoda: `handleGenerujPDF($id_sezony)`.

Tato metoda obsahuje posloupnost kroků, které jsou nutné k vytvoření PDF. Nejprve je potřeba vytvořit šablonu `$template` pomocí funkce `createTemplate()` a nastavit cestu k souboru, ve kterém se nachází obsah výsledného PDF. Následně je potřeba do této šablony dostat i všechny jednotlivé cesty výkazu jízd, aby se mohl spočítat celkový počet

kilometrů a částka, která má uživateli být proplacena. Klub TJ Sokol Žalkovice má částku na proplacení pevně stanovenou, jedná se konkrétně o 3 Kč/km a tato částka se nemění.

Dalším krokem je vytvoření PDFResponse objektu a také je možné nastavit ostatní konfigurace výsledného PDF, jako je například název, formát strany atd. Poslední fází je poslání výsledného PDF na výstup a zobrazení uživateli: `$this->sendResponse($pdf)`.

Takto vytvořený výkaz jízd si uživatel může stáhnout nebo vytisknout. Posledním krokem, který musí uživatel vykonat je, vykázat si jednotlivé jízdy, aby se zařadily mezi proplacené. Jakmile uživatel stiskne tlačítko vykázat, zobrazí se nejprve hláška, aby si zkontroloval, zda má vygenerované PDF a následně se metodou `handlevykazat($id_vykazu, $id_sezony)` provede vykázání jízdy. Tato metoda nastaví v tabulce `vykaz_jizd` hodnotu sloupce `odevzdano` na 1.

6.9 Grafický editor sestav

Grafický editor sestav slouží především pro trenéry a hráče. Trenér má u každého zápasu možnost vytvořit grafickou sestavu klubového týmu v závislosti na zvolené formaci, např. 4-3-2-1. Hráč si pak takto vykreslenou sestavu může zobrazit, tím pádem má před zápasem informaci o tom, na jakém postu bude hrát. Stejně tak má i vedoucí klubu možnost podívat se na grafickou sestavu zápasu.

Aby trenér mohl sestavu vytvořit graficky, je nejdříve potřeba sestavit soupisku 11 hráčů, kteří budou hrát v základní sestavě. Kliknutím na tlačítko přidat hráče do sestavy dojde k zobrazení formuláře `createComponentAddPlayerForm()`, ve kterém trenér zvolí hráče a případně vybere i post, na kterém daný hráč bude hrát. Post však není povinné zvolit, protože je každému hráči následně automaticky přiřazen v grafickém editoru, v závislosti na pozici umístění hráče na hřišti, viz další odstavce. Následně dojde k odeslání formuláře stisknutím tlačítka uložit a k zapsání hráče do sestavy metodou `addPlayerFormSucceeded(Form $form)`.

Jakmile je vytvořena soupiska, je možné přejít k grafickému editoru sestav. Kliknutím na tlačítko sestava graficky dojde k zobrazení stránky s grafickým editorem. Zde má trenér možnost zvolit formaci, kterou chce pro daný zápas použít. Má na výběr ze 7 formací (jedná se o celosvětově nejpoužívanější formace). V závislosti na zvolené formaci je následně do hřiště vykresleno 11 dresů s odpovídajícím rozmístěním. Po kliknutí na konkrétní pozici (dres) vybere hráče, kterého zde chce umístit. Na této pozici dojde k vypsání jména hráče a čísla dresu. Pokud na tuto pozici s umístěným hráčem opětovně klikne, dojde k odstranění hráče z této pozice. Jestliže jsou všichni hráči rozmístěni, může vytvořenou grafickou sestavu uložit. Následně také dojde u každého hráče k uložení pozice, na které se nachází, aby bylo možné vytvořenou sestavu opětovně zobrazit, případně upravit.

Grafický editor byl vytvořen s použitím JavaScriptu. Nejprve je vždy vykresleno hřiště s jednou automaticky zvolenou formací (rozmístěním dresů). Šablona `editFormation.latte` obsahuje skript, který vykonává veškeré důležité funkce. Pro správnou funkcionalitu jsou také důležité 2 metody v presenteru `MatchPresenter.php`.

První z nich je `handleGetPlayers($id_zapasu, $id_tymu)`, která získá pole hráčů rozmístěných na hřišti. U každého hráče je potřeba získat jeho *id, post, pozici, číslo dresu* a *jméno*. Tato metoda je následně použita ve skriptu v šabloně, aby bylo možné hráče v hřišti správně vykreslit. Druhou důležitou metodou je `handleSaveFormation($id_zapasu, $id_formace, array $players)`, která provádí uložení formace. Konkrétně provede u každého hráče uložení pozice a postu. Pozice nabývá hodnot od 1 do 11 a v závislosti na číslu pozice je zjištěn odpovídající post pomocí metody `positionToPost($id_formace, $pozice)`.

Post je následně zobrazen u každého hráče v soupisce konkrétního zápasu. Pokud byl hráči post přiřazen již ve formuláři (zmíněno výše), dojde k jeho přepsání v závislosti na pozici hráče v hřišti.

Na obrázku 6.8 je možné vidět grafické zobrazení sestavy zápasu. Z obrázku tedy vyplývá, že je zvolena formace 4-2-4 a v hřišti můžeme vidět rozmístění hráčů. Vzhledem k tomu, že uživatelé budou informační systém využívat i z mobilních telefonů, bylo potřeba přizpůsobit zobrazení grafické sestavy i na zařízení s menšími displeji.

Obrázek 6.8: Příklad grafického vytvoření sestavy

Kapitola 7

Testování

Testování je podstatnou částí vývoje informačního systému, která pomáhá vývojáři implementovanou aplikaci zlepšovat a zdokonalovat. Důležitým úkolem testování je odhalování chyb a nedostatků, které je nutné před uvedením systému do provozu odstranit. Testování probíhalo ve dvou fázích:

- první fází bylo programátorské testování, které probíhalo od počátku až do konce vývoje,
- druhou fází pak bylo testování výsledného informačního systému uživatelem.

Tato kapitola se bude těmto dvěma typům testování dále věnovat.

7.1 Testování programátorem

Programátor by měl vyvíjený systém testovat neustále. Jakmile byla do systému přidána nějaká nová funkcionalita, došlo k jejímu testování a zdokonalování programátorem, dokud nebyla naprosto bezchybná. Vždy, když bylo do systému zavedeno větší množství funkcí, které tvořily nějakou část systému, docházelo i ke konzultacím s členy klubu, zda může daná funkcionalita vypadat tímto způsobem. Vznikl tedy prostor na připomínky a případné zdokonalení dané části. Samozřejmě byla nejprve tato skupina funkcí řádně otestována, aby bylo zajištěno, že se v nich nebudou vyskytovat žádné chyby.

7.2 Uživatelské testování

Uživatelské testování probíhalo po dokončení informačního systému a po jeho řádném otestování programátorem. Testování se účastnili někteří z hráčů a trenérů fotbalového klubu, ale také lidé, kteří nejsou členy klubu. Účelem tohoto testování bylo zjistit, zda je uživatelské rozhraní dostatečně přehledné a především i to, že se v systému nenachází žádné chyby. Další odstavec popisuje, jakým způsobem toto testování probíhalo.

Pro účely testování systému uživatelem byl vytvořen testovací formulář, který obsahoval následující úkoly:

1. Přihlaste se do informačního systému.
2. Zobrazte si Váš profil a proveďte změnu hesla.

3. Pokud jste přihlášen v roli vedoucí klubu, vytvořte novou sezónu a pokračujte dále, jinak jděte na úkol 10.
4. Vytvořte soutěž.
5. Vytvořte 1 klubový tým a 1 neklubový.
6. Zaregistrujte oba týmy do Vámi vytvořené sezóny a soutěže.
7. Přidejte zápas do Vámi vytvořené soutěže a sezóny, kde domácí a hostující týmy budou ty Vámi zaregistrované.
8. Vytvořte klubového hráče.
9. Zaregistrujte tohoto hráče do Vámi vytvořené sezóny, soutěže a klubového týmu.
10. Pokud jste přihlášen v roli trenéra, vytvořte trénink a pokračujte dále, jinak jděte na úkol 15.
11. Zobrazte si zápasy v sezóně *2018-2019* a v soutěži *1. A třída skupina B*.
12. Zápas s datem konání *14. 04. 2019* nastavte na *odehráno* a zadejte skóre.
13. U stejného zápasu vytvořte sestavu klubového týmu, tj. přidejte 11 hráčů do sestavy.
14. Sestavu vytvořte i graficky.
15. Pokud jste přihlášen jako hráč nebo trenér, vytvořte si 3 cesty ve výkazu jízd, jinak jděte na úkol 19.
16. Vygenerujte si PDF s vytvořenými cestami a následně vykažte tyto 3 jízdy.
17. Pokud jste přihlášen jako hráč, potvrďte účast na tréninku týkajícího se Vašeho týmu, který se koná dne 19. 04. 2019, jinak jděte na bod 19.
18. Zobrazte si grafickou sestavu u zápasu s datem konání *14. 04. 2019*.
19. Odhlaste se.

Každý uživatel byl před samotným testováním seznámen s prostředím informačního systému, aby se v něm alespoň základně orientoval. Následně uživatel obdržel jednotlivé úkoly a měl vyhrazený čas na jejich splnění. Při řešení těchto úkolů nebylo uživateli nijak napomáháno. Každý uživatel obdržel testovací přihlašovací údaje nutné pro vykonání tohoto testu. Čas, který trval na splnění úkolu, byl jeden z důležitých faktorů, které bylo při tomto testování potřeba sledovat. Z časového rozsahu je totiž možné určit, zda je rozhraní aplikace pro uživatele dostatečně přehledné a intuitivní.

Celkem se tohoto testování účastnili 4 uživatelé - 1 v roli vedoucího klubu, 1 v roli trenéra a 2 v roli hráče. Na konci testování proběhla zpětná vazba od uživatelů. Výsledky uživatelského testování můžete vidět v tabulkách 7.1 a 7.2.

Hodnota `ano` v tabulce znamená, že úloha byla vyřešena rychle bez větších obtíží. Řádky tabulky, na kterých se nachází znak „`-`“ označují úkoly, které pro daného uživatele nebyly určeny.

Z výsledků tabulek je zřejmé, že aplikace je pro uživatele víceméně přehledná a srozumitelná. Uživatel, který byl přihlášen jako vedoucí, měl největší problémy s úlohami 6, 7 a 9.

Úlohy/Uživatelé	Uživatel 1 role vedoucí	Uživatel 2 role trenér
1	ano	ano
2	ano	ano
3	ano	-
4	ano	-
5	ano	-
6	2 minuty	-
7	3 minuty	-
8	ano	-
9	2 minuty	-
10	-	ano
11	-	1-2 minuty
12	-	1-2 minuty
13	-	4 minuty
14	-	ano
15	-	1-2 minuty
16	-	ano
17	-	-
18	-	-
19	ano	ano
Zpětná vazba	+- aplikace je přehledná, ale je potřeba si na ni zvyknout, aby v ní byl člověk dostatečně zorientován	+ vše se mi zdálo přehledné - nevím

Tabulka 7.1: Tabulka naměřených hodnot získaných při testování (role vedoucí a trenér)

Úlohy/Uživatelé	Uživatel 3 role hráč	Uživatel 4 role hráč
1	ano	ano
2	ano	ano
15	1-2 minuty	2 minuty
16	ano	ano
17	do 1 minuty	do 1 minuty
18	1-2 minuty	do 1 minuty
19	ano	ano
Zpětná vazba	+ velice oceňuji možnost generování výkazu jízd v PDF a celkově hodnotím aplikaci jako přehlednou - nic mě nenapadá	+ jedná se o aplikaci, která je na první pohled příjemná a myslím, že se v ní každý lehce vyzná

Tabulka 7.2: Tabulka naměřených hodnot získaných při testování (role hráč)

Vyřešení těchto úloh mu trvalo trochu déle (čas je možný vidět v tabulce), nicméně doba, kterou nad těmito úkoly strávil, nebyla nijak kriticky dlouhá a uživatel to dával za vinu tomu, že by na seznámení se systémem potřeboval delší čas. U role trenér můžeme vidět nejdelší čas u úkolu 13. Uživateli jeho vyřešení trvalo déle z důvodu, že bylo potřeba přidat 11 hráčů do sestavy a tím pádem se tato úloha stala časově náročnější. Pokud by měl uživatel přidat v testování pouze jednoho hráče, byl by tento čas podstatně kratší. Uživatelé, kteří byli přihlášeni jako hráči, neměli s žádnou úlohou nějaký větší problém.

Všechny ostatní úlohy zvládli uživatelé bez problémů a testování tedy skončilo s poměrně dobrými výsledky. Při uživatelském testování nebyly odhaleny ani žádné chybové stavy systému.

Kapitola 8

Závěr

Cílem této bakalářské práce bylo vytvořit informační systém pro fotbalový klub TJ Sokol Žalkovice za účelem zjednodušení organizace chodu klubu. Důraz byl kladen zejména na jednoduchou organizaci tréninků, evidenci výkazů jízd, možnost vytvářet grafické sestavy u zápasů a v neposlední řadě také na zobrazování výsledků jednotlivých zápasů. Všechny tyto požadavky byly splněny a informační systém je plně funkční.

V této technické zprávě byly nejprve popsány principy tvorby webových aplikací spolu se základními technologiemi, které se většinou při jejich tvorbě využívají. Následně byl čtenář seznámen s použitými technologiemi při vývoji tohoto informačního systému.

Následující kapitoly byly již více prakticky zaměřené. Bylo potřeba provést analýzu a specifikaci požadavků, která stanovila funkcionalitu celého systému. Analýza byla úspěšně provedena a jejím výsledkem byl diagram případů užití, který zobrazuje veškeré funkční požadavky na výsledný systém. Zároveň v této fázi byly stanoveny i uživatelské role, které v systému vystupují.

Další kapitola se věnuje návrhu informačního systému, který byl vytvořen na základě požadavků získaných při analýze. Nejdůležitější částí návrhu byl ER diagram, který znázorňuje ukládání dat do databáze pomocí entitních množin a vztahů mezi nimi. Byl popsán význam jednotlivých entitních množin a následně byl vytvořen i návrh uživatelského rozhraní.

Jakmile byl návrh systému úspěšně dokončen, následovala implementace, které se věnuje další kapitola. Zde je čtenář seznámen s adresářovou strukturou projektu a následně je detailněji popsána funkcionalita nejdůležitějších částí systému.

Následovalo testování, které bylo rozděleno do dvou fází. Nejprve byla aplikace průběžně testována programátorem a jednotlivé části byly konzultovány se členy klubu. Následně bylo provedeno testování samotnými uživateli. Chyby, které byly nalezeny zejména při testování programátorem, byly vždy odstraněny.

Vzhledem k tomu, že implementace informačního systému proběhla úspěšně, došlo ke splnění hlavních cílů práce. Aplikace je nyní dostupná na adrese `www.is-fotbalovyklub.eu`, nicméně její plné nasazení je plánováno až na podzim roku 2019.

8.1 Budoucí vývoj

Zajímavou možností budoucího vývoje systému by bylo vytvoření mobilní aplikace, která by sloužila jako ekvivalent informačního systému a pravděpodobně by ještě více usnadnila uživatelům jeho používání.

Dalším možným rozšířením by bylo zavedení informačního systému i mezi ostatní kluby v jednotlivých soutěžích. Usnadnilo by to práci především trenérům nebo vedoucím, protože ti musí do systému zadávat soupisky zápasů nejen u svých týmů, ale i u neklubových týmů. Pokud by tedy tento systém využívaly všechny týmy v soutěži, došlo by k vyřešení této situace.

Literatura

[1] Adaptic: *JavaScript* . [Online; navštíveno 27. 03. 2019].
URL http://www.adaptic.cz/znalosti/slovnicek/javascript/

[2] Adobe: *Informace o webových aplikacích*. [Online; navštíveno 17. 01. 2019].
URL https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html#about_web_applications

[3] CodeIgniter: *Welcome to CodeIgniter* . [Online; navštíveno 26. 03. 2019].
URL https://www.codeigniter.com/user_guide/general/welcome.html

[4] FlipperWorld: *Bootstrap: Co je to, odkud se začít učit a jak ji používat* . [Online; navštíveno 27. 03. 2019].
URL https://cs.flipperworld.org/pc/bootstrap-co-je-to-odkud-se-zacit-ucit-a-jak-ji-pouzivat

[5] Foral, J.: *Webová aplikace*. [Online; navštíveno 21. 01. 2019].
URL http://wiki.knihovna.cz/index.php/Webov%C3%A1_aplikace#V.C3.BDhody_webov.C3.BDch_aplikac.C3.AD

[6] FullCalendar: *FullCalendar* . [Online; navštíveno 27. 03. 2019].
URL https://fullcalendar.io

[7] Laravelblog: *Co je Laravel?* . [Online; navštíveno 26. 03. 2019].
URL https://laravelblog.cz/co-je-laravel

[8] Lazaris, L.: *CSS Okamžitě*. Computer Press, 2014, ISBN 978-80-251-4176-2.

[9] ManagementMania: *Architektura klient-server (Client–server model)*. [Online; navštíveno 21. 01. 2019].
URL https://managementmania.com/cs/architektura-klient-server

[10] ManagementMania: *Třívrstvá architektura (Three-tier architecture)*. [Online; navštíveno 21. 01. 2019].
URL https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture

[11] ManagementMania: *Webová aplikace (Web Application)*. [Online; navštíveno 17. 01. 2019].
URL https://managementmania.com/cs/webova-aplikace-web-application

[12] MySQLTUTORIAL: What Is MySQL and Why It Is the World's Most Popular Open Source Database. [Online; navštíveno 22. 01. 2019].
URL http://www.mysqltutorial.org/what-is-mysql/

[13] NetDIRECT: *Webové aplikace*. [Online; navštíveno 17. 01. 2019].
URL https://www.netdirect.cz/slovnik-pojmu/670/webove-aplikace

[14] Nette: *Composer* . [Online; navštíveno 26. 03. 2019].
URL https://doc.nette.org/cs/2.4/composer

[15] Nette: *Latte* . [Online; navštíveno 26. 03. 2019].
URL https://latte.nette.org/cs/guide

[16] Nette: *MVC aplikace a presentery* . [Online; navštíveno 26. 03. 2019].
URL
https://doc.nette.org/cs/2.4/presenters#toc-zpracovani-akce-presenteru

[17] Nette: *Seznámení s Nette Frameworkem* . [Online; navštíveno 26. 03. 2019].
URL https://doc.nette.org/cs/2.4/getting-started

[18] Nette: *Slovníček pojmů* . [Online; navštíveno 26. 03. 2019].
URL https://doc.nette.org/cs/2.4/glossary

[19] Nette: *Tracy* . [Online; navštíveno 26. 03. 2019].
URL https://tracy.nette.org/cs/guide

[20] php.net: *What is PHP?* [Online; navštíveno 22. 01. 2019].
URL http://php.net/manual/en/intro-whatis.php

[21] Čápka, D.: *Architektura MVC* . [Online; navštíveno 26. 03. 2019].
URL https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor

[22] Čápka, D.: *Lekce 2 - UML - Use Case Diagram* . [Online; navštíveno 27. 03. 2019].
URL https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram

[23] Čápka, D.: *Úvod do JavaScriptu* . [Online; navštíveno 27. 03. 2019].
URL https://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk//palcovat/31721/1

[24] Tutorialspoint: *CakePHP Tutorial* . [Online; navštíveno 26. 03. 2019].
URL https://www.tutorialspoint.com/cakephp/

[25] W3schools: *Bootstrap 4 Tutorial* . [Online; navštíveno 27. 03. 2019].
URL https://www.w3schools.com/bootstrap4/

[26] W3schools: *HTML5 Introduction*. [Online; navštíveno 21. 01. 2019].
URL https://www.w3schools.com/html/html5_intro.asp

[27] W3schools: *PHP 5 Tutorial*. [Online; navštíveno 22. 01. 2019].
URL https://www.w3schools.com/php/

[28] W3schools: *PHP MySQL Database* . [Online; navštíveno 22. 01. 2019].
URL https://www.w3schools.com/php/php_mysql_intro.asp

[29] W3schools: *What is HTML?* [Online; navštíveno 21. 01. 2019].
URL https://www.w3schools.com/html/html_intro.asp

[30] Zendulka, J.: *Konceptuální modelování a návrh databáze* . [Online; navštíveno 02. 04. 2019].
URL https://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/2_2.pdf

Příloha A

Obsah CD

- **/xgazdo02/BP/** - technická zpráva ve formátu PDF
- **/xgazdo02/sablona/** - zdrojové soubory technické zprávy
- **/xgazdo02/db/** - skript pro inicializaci databáze
- **/xgazdo02/fotbalovy_klub/** - implementovaný informační systém
- **/xgazdo02/instalace/** - instalační pokyny