VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ŠACHOVÁ WEBOVÁ APLIKACE

CHESS WEB APPLICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MIRKA KOLAŘÍKOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

146122

Ústav:	Ústav informačních systémů (UIFS)
Studentka:	**Kolaříková Mirka**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Šachová webová aplikace**
Kategorie:	Informační systémy
Akademický rok:	2022/23

Zadání:

1. Prostudujte současné technologie pro tvorbu interaktivních webových aplikací.
2. Seznamte se s požadavky na aplikaci pro podporu dětského šachového kroužku, evidenci členů, sledování docházky, pořádání turnajů, záznam šachových partií v rámci klubu a dalších.
3. Na základě analyzovaných požadavků navrhněte architekturu aplikace.
4. Po konzultaci s vedoucím zvolte vhodné technologie a implementujte navrženou aplikaci. Zakomponujte grafickou vizualizaci partií a motivační program pro žáky.
5. Proveďte testování vytvořené aplikace v praxi.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Gutmans, A., Rethans, D., Bakken, S.: Mistrovství v PHP 5, Computer Press, 2012
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015

Při obhajobě semestrální části projektu je požadováno:
Body 1 až 3.

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

Vedoucí práce:	**Burget Radek, doc. Ing., Ph.D.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	18.10.2022

Abstrakt

Cílem této práce bylo navrhnout a implementovat interaktivní webovou aplikaci pro podporu dětského šachového kroužku. Aplikace obsahuje evidenci členů, sledování docházky, pořádání turnajů, motivační program pro studenty a vizualizované přehrání šachové partie ve formátu PGN. Aplikace byla implementována pomocí moderních webových technologií Vue.js, Laravel a databáze MySQL. Výsledná aplikace byla úspěšně vytvořena a otestována.

Abstract

The aim of this thesis was to design and implement an interactive web application for a children's chess club. The application includes member management, attendance tracking, tournament management, a motivational program for students, and a visualized replay of chess games in PGN format. The application was implemented using modern web technologies such as Vue.js, Laravel, and MySQL database. The resulting application was successfully created and tested.

Klíčová slova

Webová aplikace, Backend, Frontend, TypeScript, Přehrávač šachových partií, Laravel, PHP, Vue.js, MySQL, REST API.

Keywords

Web application, Backend, Frontend, TypeScript, Chess game player, Laravel, PHP, Vue.js, MySQL, REST API.

Citace

KOLAŘÍKOVÁ, Mirka. *Šachová webová aplikace*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Radek Burget, Ph.D.

Šachová webová aplikace

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana doc. Ing. Radka Burgeta Ph.D. Další informace mi poskytl Ing. Jan Kolařík. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

.....
Mirka Kolaříková
5. května 2023

Poděkování

Ráda bych poděkovala panu doc. Ing. Radku Burgetovi Ph.D. za skvělé vedení této bakalářské práce, odborné konzultace a cenné rady. Dále bych ráda poděkovala Ing. Janu Kolaříkovi za možnost podílet se na vývoji aplikace pro dětský šachový kroužek a za poskytnutí potřebných informací.

Obsah

Kapitola 1

Úvod

Cílem této bakalářské práce je navrhnout a vytvořit webovou aplikaci pro podporu dětského šachového kroužku. Výsledná aplikace by měla umožnit vedoucímu kroužku spravovat členy a sledovat docházku, organizovat turnaje a zaznamenávat jejich výsledky včetně jednotlivých šachových partií. Uložené šachové partie by měly jít i vizualizovat pro lepší prostudování jednotlivých tahů a možných taktik. Dále by tato aplikace měla nabídnout motivační program pro studenty, aby mohli být v průběhu kroužku odměněni za svou aktivitu.

Kapitola 2 se zabývá moderními webovými aplikacemi a rozebírá jejich výhody, nevýhody a příklady použití tohoto typu aplikací v aktuální době. Nejdříve jsou popsány základní typy architektur v sekci 2.1 a dále se kapitola věnuje současně používaným technologiím pro vývoj webových aplikací. Technologie jsou zde rozděleny na serverové 2.2.1 a klientské 2.2.2. Obě podkapitoly se věnují jednotlivým programovacím jazykům a také jejich frameworkům, které nabízí řešení na časté programátorské problémy a umožňují rychlý a organizovaný vývoj aplikací.

Kapitola 3 se poté blíže věnuje analýze konkrétních požadavků kroužku na výslednou aplikaci a zaměřuje se na používaný způsob záznamu výsledků turnaje a partií. Na tuto kapitolu je navázáno kapitolou 4, která se zaměřuje na aktuální prezentaci šachového kroužku a poté zkoumá další existující řešení.

Dále je představen návrh řešení aplikace a zvolené technologie pro implementaci v kapitole 5. Tato kapitola také rozebírá vytvořený diagram případů užití, který je celý dostupný v příloze B, a ER diagram pro definici jednotlivých entit v systému v příloze C.

Kapitola 6 popisuje způsob implementace aplikace z pohledu serverové i klientské aplikace zvolenými technologiemi, představuje způsob řešení hlavních částí aplikace a použití dalších knihoven, zejména pro implementaci vizuálního přehrávače partií. Způsob a výsledky testování výsledné aplikace jsou představeny v kapitole 7.

Na závěr jsou v kapitole 8 rozebrány možnosti dalšího vývoje, jako je možné využití umělé inteligence pro automatické vyhodnocení zadaných šachových úloh.

Kapitola 2

Webové aplikace

Webová aplikace je software, který na rozdíl od desktopových aplikací běží na webovém serveru a uživatel s ní interaguje přes internetový prohlížeč. Příkladem běžných prohlížečů je Google Chrome[1], Microsoft Edge[2] nebo Safari[3].

Existují statické i dynamické webové stránky. Statické webové stránky slouží hlavně k zobrazování neměnných informací. Jsou sice jednoduché z pohledu vývoje a rychle se načítají, ale zobrazují všem uživatelům stejný obsah. Dynamické webové stránky dokáží reagovat na požadavky uživatele a zobrazit požadovaný obsah podle jeho interakcí. Často využívají databáze k uchování dat a obsahují interaktivní prvky jako jsou formuláře, různá tlačítka, vyskakovací okna a podobně. Příklad dynamické webové aplikace může být internetový obchod, jako je Ebay[4] a Amazon[5]. Uživatelé zde mohou vybírat zboží, přidávat ho do košíku, zadávat kupóny a provádět platby. Uživatelé tak ovlivňují to, co se na stránce zobrazuje. Ovšem i tyto stránky využívají statické prvky. Příkladem může být sekce s kontakty, tedy data, se kterými není potřeba interagovat.

Díky webovým aplikacím není pro uživatele potřeba aplikaci stahovat ani instalovat na svém zařízení a není tak omezen hardwarovými požadavky. S připojením k internetu může uživatel využít aplikaci na jakémkoliv zařízení, což může být výhodné v případě, kdy dojde k poruše zařízení, nebo kdy uživatel chce využít aplikaci na jiném zařízení. Data jsou často uložená na serveru, takže uživatel nemusí řešit zálohu dat. Aktualizace aplikace se také provádí na serveru. Na druhou stranu, pro využívání těchto aplikací je uživatel závislý na internetovém připojení a na poskytovateli služby. Také jsou zde možné problémy se zabezpečením.

Webové aplikace jsou čím dál tím více využívány a mohou mít mnoho různých příkladů použití. Mezi nejpopulárnější patří sociální sítě jako Facebook[6] a Twitter[7], které umožňují uživatelům propojit se s ostatními uživateli, sdílet různá média a komunikovat přes tuto platformu s přáteli. Dalším příkladem jsou aplikace pro konzumaci různých médií jako je Netflix[8] a Spotify[9]. Webové aplikace jsou také často využívány jako informační systémy

[1] https://www.google.com/intl/cs_CZ/chrome/
[2] https://www.microsoft.com/cs-cz/edge
[3] https://www.apple.com/safari
[4] https://www.ebay.com
[5] https://www.amazon.com
[6] https://www.facebook.com
[7] https://twitter.com
[8] https://www.netflix.com
[9] https://open.spotify.com

různých organizací. V neposlední řadě patří mezi populární příklady i emailové aplikace, internetové bankovnictví a mnoho dalších. Tento text vycházel z literatury [19] a [12].

2.1 Architektura

Architektura webových aplikací se zabývá návrhem a organizací aplikace, včetně toho, jak spolu jednotlivé části interagují.

2.1.1 Klient-server

Architektura klient-server se skládá ze dvou částí: klienta a serveru. Klientská aplikace běží na zařízení uživatele, zatímco server běží na jiném zařízení nebo na vzdáleném serveru. Tato architektura je běžná pro webové aplikace, kde je klientem často internetový prohlížeč. Klient je zodpovědný za prezentaci dat a získávání požadavků od uživatele. Posílá požadavky serveru, který je zpracovává a posílá odpověď. Aplikační logika může být implementována klientem, serverem nebo oběma. Síťová komunikace je standardizována pomocí internetových protokolů TCP/IP.

Při prvních implementacích architektur klient/server nebyly používány otevřené protokoly, jako je TCP/IP. To znamenalo, že komunikace mezi klientem a serverem byla omezena na vlastní protokol nebo technologii, což komplikovalo integraci s jinými systémy nebo platformami. Klienti tak museli být navrženi speciálně pro server, se kterým komunikovali, což ztěžovalo opětovné využití nebo sdílení kódu. [24]

2.1.2 Třívrstvá architektura

Třívrstvá architektura je složená ze tří vrstev. Výhodou této architektury je, že jednotlivé vrstvy jsou na sobě nezávislé a mohou být vyvíjeny zvlášť. Popis této architektury a jednotlivých vrstev vychází z literatury [26].

Prezentační vrstva Tato vrstva zobrazuje uživatelské rozhraní, tj. jednotlivé stránky i s prvky jako jsou tlačítka a formuláře se kterými uživatel interaguje. Může být přizpůsobena různým zařízením a platformám. Pro ukládání a získávání dat je závislá na aplikační vrstvě.

Aplikační vrstva Tato vrstva zajišťuje veškerou logiku aplikace a výpočty. Provádí operace mezi uživatelskými požadavky z prezenční vrstvy a daty z vrstvy datové.

Datová vrstva Jedná se o nejnižší vrstvu, je zodpovědná za získávání a ukládání dat, včetně datově-funkčních operací jako je specifický výběr, agregace a předzpracování.

2.2 Technologie

Webové aplikace je možné rozdělit do dvou základních typů – klientské a serverové aplikace, neboli **backend** a **frontend**. Oba typy aplikací jsou běžně používány v moderním webovém vývoji a často se používají v kombinaci, kdy klient požaduje data od serveru pomocí klientské aplikace a serverová aplikace poskytuje tyto data klientovi. Ke komunikaci využívají internetovou síť.

2.2.1 Backend

Serverové aplikace jsou programy, které běží na serveru a slouží ke zpracovaní dat a interagují s databází. Tyto aplikace jsou napsány pomocí serverových programovacích jazyků, jako je například PHP, Python nebo Java.

Serverové aplikace mohou být také vytvořeny pomocí frameworků, které poskytují hotová řešení pro časté problémy, jako je přesměrování na určitou URL, validace vstupních dat, autentizace uživatelů a řízení přístupu. Pomáhají i s obranou proti útokům na zabezpečení. Frameworky tak často usnadňují a zrychlují celkový vývoj.

PHP

PHP je multiplatformní programovací jazyk vyvinutý především pro web. Je to skriptovací jazyk běžící na straně serveru. V rámci HTML souboru lze zakomponovat i PHP kód, který se potom provede vždy při navštívení dané stránky. Je interpretovaný webovým serverem a může generovat HTML a komunikovat s databází. PHP je velmi rychlý a efektivní jazyk, takže se hodí na malé i velké projekty. Je to nejpoužívanější serverový programovací jazyk, k datu 2. května 2023 ho využívalo až 77.4% webových stránek [15]. Je k němu tedy i velká komunita aktivních uživatelů a zdokumentovány nejrůznější případy použití. Jakožto jazyk vytvořený pro web nabízí spoustu knihoven pro usnadnění práce například s posíláním emailů, parsování XML, práce s cookies, generování PDF dokumentů a dalších. PHP má k dispozici přímé připojení k mnoha databázovým systémům. Je proto možné se přímo přijit k MySQL, PostgreSQL, MongoDB a dalším. Nabízí také databázovou abstraktní vrstvu PDOs (z angl. PHP Database Objects), která pomáhá dodržovat nejlepší programovací praktiky a konzistenci kódu. Tento text vycházel z knihy [29].

Pro PHP existuje mnoho frameworků. Tyto frameworky nabízejí předem vytvořená zapouzdřená řešení pro různé programátorské problémy, a programátoři se tak mohou zaměřit na vývoj specifických částí aplikace a urychlit tak celkový vývoj. Každý framework má předdefinovanou strukturu kódu, což umožňuje udržovat kód hezký a konzistentní. Je možné si i nechat vygenerovat kostry určitých částí kódu.

Nejpoužívanější PHP frameworky jsou Laravel[10] a Symfony[11], dále pak například CodeIgniter[12], Slim[13] a PHPCake[14].

Symfony Symfony je open-source PHP framework, který byl vytvořen v roce 2005. Tento framework umožňuje vývoj webových aplikací s využitím návrhového vzoru MVC[15] a nabízí mnoho znovupoužitelných komponent. K instalaci a generování těchto komponent, přidávání knihoven a dalších podobných úkonů využívá Symfony balíčkovací nástroj **Composer**. Pro vytvoření uživatelského rozhraní Symfony nabízí šablonovací systém **Twig**. Symfony se spíše hodí pro větší projekty a zkušenější vývojáře, neboť má složitější syntaxi a strukturu. Framework nabízí ORM vrstvu **Doctrine** pro efektivní práci s databází. [20]

[10] https://laravel.com
[11] https://symfony.com
[12] https://codeigniter.com
[13] https://www.slimframework.com
[14] https://cakephp.org
[15] MVC – Model-View-Controller, aplikace je logicky rozdělena na prezenční vrstvu (View), data (Model) a ovladače (Controller), které řídí zobrazování dat.

Laravel Laravel vznikl v roce 2011 a je nejpoužívanějším PHP frameworkem, který vychází ze Symfony. Využívá některé komponenty ze Symfony, konkrétně `HttpFoundation` pro práci s HTTP, jako jsou cookies a správa session, a `Routing` komponentu pro routování stránek na určité URL. Dále navazuje vlastními stavebními bloky a filozofií. Nabízí ORM vrstvu **Eloquent** a pro psaní HTML nabízí šablonovací systém **Blade**. Laravel je vhodný pro projekty různých velikostí a s jeho jednoduchou syntaxí je ideální pro začátečníky. [20]

Java

Java je významným programovacím jazykem pro webové aplikace. Jde o vysoce úrovňový, objektově orientovaný a interpretovaný jazyk. Java byla vyvinuta společností Sun Microsystems v roce 1995. Znamenala revoluci v programování a nastavila standardy pro návrh programovacích jazyků. Z některých jejích principů vychází třeba .NET[16]. Java se neustále rozvíjí a s každou novou verzí přináší nové možnosti programování. Java přebírá syntaxi z jazyka C a objektový model návrhu z jazyka C++.

Programy v Javě jsou přenositelné a bezpečné. Výsledný kód překladače totiž není spustitelný přímo, ale je to bytecode – optimalizovaný set instrukcí pro prostředí běhu v javě **JVM** (z angl. Java Virtual Machine). JVM je tedy interpret tohoto bytecodu. Program tak stačí přeložit jen jednou a jakmile je na nějakém zařízení nainstalované JVM, může tam běžet i tento program. Java je tak velmi přenosný a multiplatformní jazyk. Díky JVM je také zabezpečený souborový systém. JVM zajišťuje bezpečnost tím, že nedovolí programu vykonávat žádné neoprávněné operace nebo vedlejší efekty a funguje tak jako firewall.

V Javě existuje tzv. **Garbage Collector** – systém, který automaticky vyhledává nepoužívané části paměti, uvolňuje je pro další použití a pomáhá tak zabránit únikům paměti.

Tento text vycházel z knihy [21].

Python

Python je vysokoúrovňový interpretovaný programovací jazyk, jehož první verze byla vytvořena v roce 1991 a nyní je k dispozici verze 3. Původní implementace Pythonu byla napsána v jazyce C. Jazyk Python je open-source a většina distribucí GNU/Linuxu již obsahuje Python jako součást instalace. Tato kapitola vychází z knihy [23].

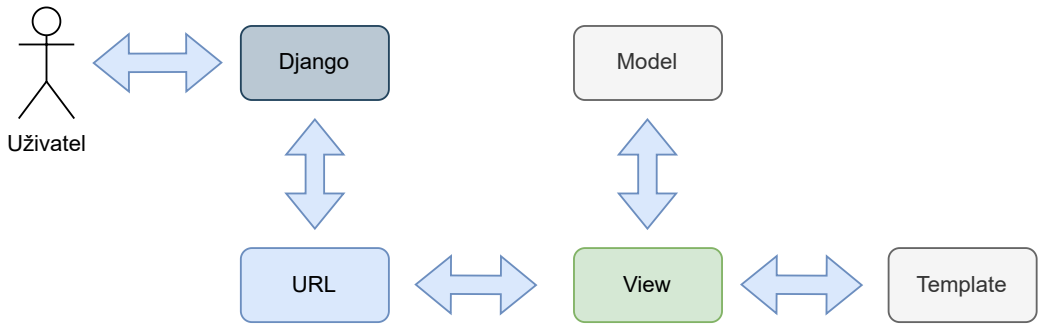
Python má dynamickou kontrolu datových typů, což znamená, že proměnné mohou být definovány programátorem nebo mohou být dynamicky odvozeny a mohou být během průběhu programu změněny. Python podporuje jak objektově orientované, tak funkcionální programování. Kód psaný v Pythonu je obvykle krátký a dobře čitelný, protože pro definici jednotlivých bloků se využívá pouze odsazování namísto závorek.

V porovnání s jazyky jako je Java, PHP nebo C není tak rychlý a efektivní, což ale není pro běžné webové aplikace nijak problematické.

Django Django[17] je populární multiplatformní webový framework napsaný v jazyce Python, který byl poprvé vydán v roce 2005 a stále se aktivně vyvíjí. Každý měsíc jsou vydávány nové balíčky s opravami chyb a vylepšením bezpečnosti. Framework je snadno naučitelný, ale zároveň mocný a flexibilní. Obsahuje všechny potřebné moduly k vývoji webových aplikací, jako je například přístup k databázi nebo renderování šablon, a tak není nutné nic dalšího stahovat.

[16] https://dotnet.microsoft.com/en-us
[17] https://www.djangoproject.com/

Nabízí i ORM vrstvu ve stylu pythonu pro snadnější zacházení s databází. Na rozdíl od výše zmíněných PHP frameworků je Django založen na architektuře **MVT** (Model-View-Template), což je podobné architektuře MVC. Princip je znázorněn na obrázku 2.1. S databází zde neinteraguje ovladač (Controller), nýbrž prezenční vrstva (View), která se zároveň stará i o renderování jednotlivých HTML šablon (template), které jsou prezentovány uživateli.



Obrázek 2.1: Princip Model-View-Template (převzato z literatury [13])

Django automaticky nabízí i vestavěný režim správy databáze s uživatelským rozhraním, nazvaného Django Admin. Toto rozhraní je automaticky generováno z jednotlivých databázových entit definovaných v aplikaci, a nabízí přístup a kompletní správu těchto entit přímo z uživatelského rozhraní.

Databáze

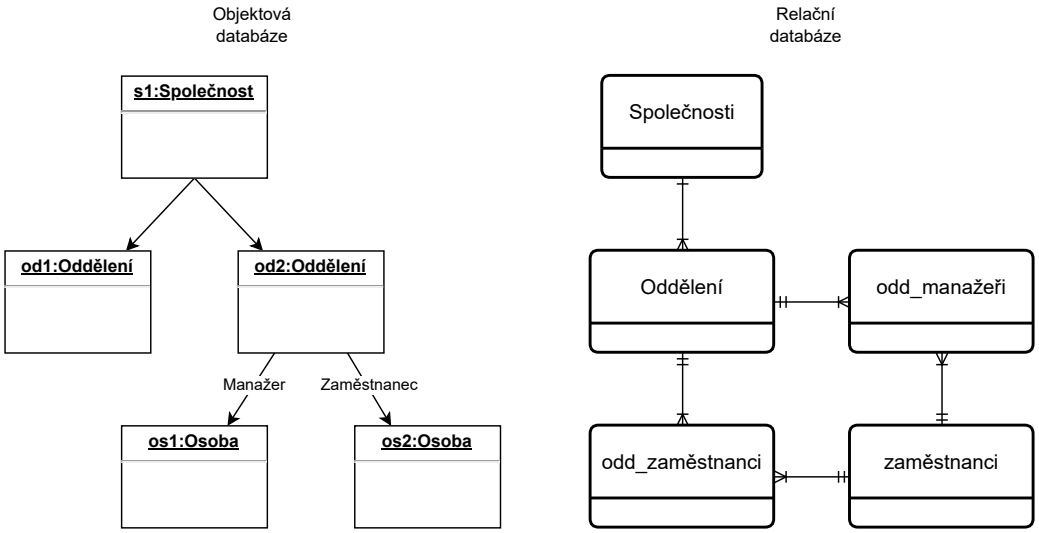
Pro ukládání dat na serveru se nejčastěji používá systém řízení báze dat, označovaný jako **SŘBZ**. Tento systém umožňuje efektivní práci s daty, jako je ukládání, čtení, úpravy a mazání, také označováno jako **CRUD** (create, read, update, delete). Databáze bývá obvykle na vzdáleném serveru. Existují dvě hlavní kategorie databází: relační a objektové, viz obrázek 2.2. Popis jednotlivých typů vychází z literatury [7].

Relační databáze Tento typ databáze je založen na konceptu tabulek, kde každá tabulka reprezentuje určitou entitu, a jednotlivé sloupce jsou atributy, zatímco řádky představují jednotlivé záznamy. Tabulky jsou mezi sebou propojeny na základě odkazů na externí primární klíče, což se nazývá relace. Mezi nejpoužívanější relační databáze patří MySQL[18], PostgreSQL[19] a MS SQL[20].

Objektové databáze Tento typ databáze reprezentuje data v podobě objektů, což je lepší přístup pro ukládání dat při programování pomocí objektově orientovaném principu, protože datový model je tak konzistentnější s modelem v programu. Objektové databáze jsou také přímo propojené s programovacím jazykem. Příkladem objektové databáze je Realm[21].

[18] https://www.mysql.com
[19] https://www.postgresql.org
[20] https://www.mssql.cz
[21] https://realm.io

NoSQL databáze Kromě relačních a objektových databází existují i NoSQL databáze, které nepoužívají tabulky a relace mezi nimi, ale jiný přístup, především dokumenty. MongoDB[22] například využívá dokumenty se strukturou dat podobnou formátu JSON. Je podobná objektové databázi.



Obrázek 2.2: Porovnání objektové a relační databáze (převzato z literatury [7])

ORM Většina webových frameworků nabízí **ORM** (anglicky Object-Relational Mapping) k zjednodušení práce s relační databází. Tento přístup mapuje objekty v programu na tabulky v databázi a umožňuje pracovat s daty jako s objekty, což je pohodlnější než psát dotazy v čistém SQL. [1]

Aplikační rozhraní

Ke komunikaci serverové aplikace s klientskou aplikací se používá definice koncových bodů API (z angl. application programming interface). Klientská aplikace požaduje nějaké data pomocí požadavku na konkrétní koncový bod a server požadavek zpracuje a data odešle. API je vlastně soubor pravidel definující způsob vzájemné komunikace. Tato podkapitola vychází z literatury [5].

SOAP Simple Object Access Protocol, jedná se o protokol založený na komunikaci přes zprávy ve formátu XML. Klient vytvoří zprávu zapsanou v XML s potřebnými informacemi a pošle ji HTTP POST požadavkem na server. Odpověď ze serveru je poté také ve formátu XML.

REST API Tento protokol implementuje čtyři typické metody pro odesílání dat. Konkrétně je to vytvoření nových dat, získání dat, aktualizace dat a mazání dat, běžně označované jako CRUD operace (create, retrieve, update, delete). Každá metoda se odlišuje typem použitého HTTP požadavku, a to **GET**, **POST**, **PUT** a **DELETE**.

[22] https://www.mongodb.com

V následujícím příkladě je uvažována entita článek (article). Pro zacházení s konkrétním článkem je třeba do koncového bodu specifikovat i jeho ID. Koncové body pro správu článku pomocí principů REST API vypadají následovně:

- POST api/article – vytvoření nového článku
- GET api/article/:id – získání článku
- PUT api/article/:id – úprava článku
- DELETE api/article/:id – smazání článku

GraphQL API Jedná se o způsob komunikace, kdy klientská aplikace požádá server o konkrétní data pomocí dotazu a server vrátí přesně ta požadovaná data. Klient v dotazu i definuje, jakou strukturu dat od serveru očekává. Na rozdíl od REST API a SOAP API tak server nevrací pevnou sadu, ale klient přímo specifikuje o jaká data má zájem.

2.2.2 Frontend

Klientské webové aplikace běží přímo v prohlížeči. Nabízí grafické uživatelské rozhraní a jsou to s čím uživatel přímo interaguje. Zpracovávají interakce jako je kliknutí myši, vstup z klávesnice a odesílání formulářů. Obsahují vizuální elementy jako jsou tlačítka, tabulky a formuláře.

Je zde kladen důraz na uživatelskou přívětivost a nepochybně i na design. Klientské webové aplikace by měly prezentovat data uživateli jasně a srozumitelně. Také by měly být použity vhodné barvy, fonty a grafické prvky, aby byla aplikace příjemná a lákavá.

Využívané technologie jsou především HTML, CSS a JavaScript.

HTML

HTML (z angl. HyperText Markup Language) je technologie, která umožňuje definovat strukturu vizuálních elementů webové stránky.

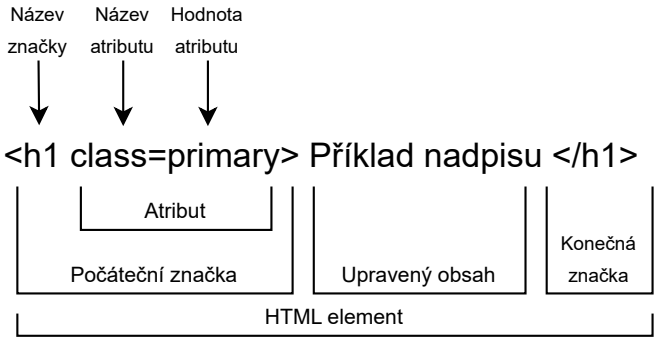
HTML dokument sestává z běžného textu, jako je „Ahoj světe!“, který je ohraničen značkami (tagy), například `<body>` a `<button>`. Značky fungují jako metadata, díky kterým dokážeme odlišit jednotlivé typy vizuálních elementů a definovat strukturu webové stránky. [17]

HTML element je tedy úsek dokumentu vymezený značkami. Koncová značka obsahuje lomítko. Například `<p>Ahoj světe</p>`. Každý element má název, atributy a obsah. Pomocí atributů lze definovat elementu další vlastnosti. Syntax HTML elementu je také rozebrána na obrázku 2.3. Názvy značek vychází z anglických zkratek, například paragraph `<p>` jako odstavec, či image `<img>` jakožto obrázek. Na velikosti písmen přitom nezáleží.

Dobré je také zmínit, že elementy se dále rozdělují na blokové a řádkové. Blokové elementy automaticky začínají na novém řádku a mají okolo sebe určité odsazení od ostatních elementů. Mezi ně se řadí elementy jako `<div>`, `<table>` a `<article>`. Řádkové elementy naopak nezačínají na novém řádku a lze je tedy skládat vedle sebe. Příkladem jsou zejména textové elementy, jako jsou `<p>`, `<span>` a `<cite>`. Tento princip není vždy pro design webové stránky úplně ideální, a proto je možné využít CSS 2.2.2 pro nejrůznější stylizaci.

Celý výčet elementů lze nalézt v aktuálním HTML standardu[23].

[23] https://html.spec.whatwg.org/multipage

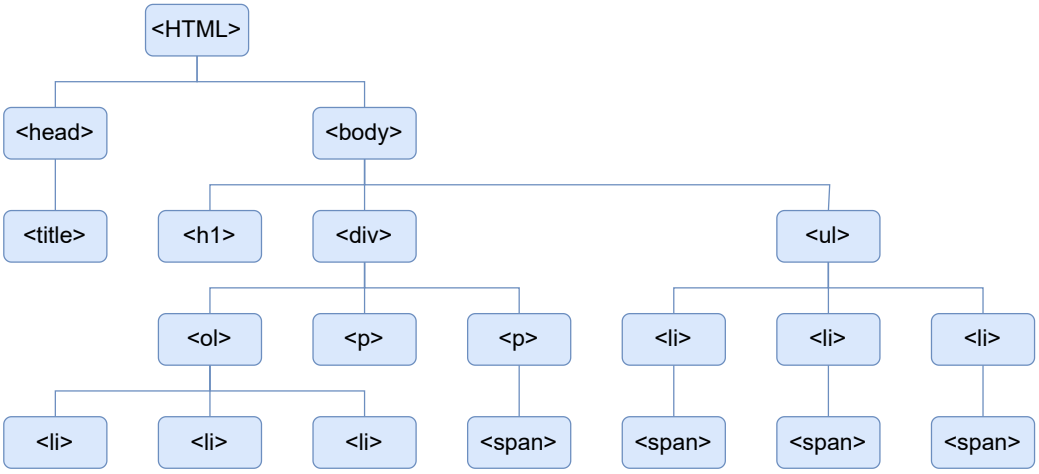


Obrázek 2.3: Syntax HTML elementu (převzato z literatury [16])

Strom elementů

Jednotlivé HTML elementy jdou do sebe různě zakomponovávat a vzniká tak hierarchická stromová struktura. Této struktuře se říká Objektový model dokumentu, zkráceně **DOM** (z angl. Document Object Model). DOM je způsob reprezentace objektů pomocí HTML elementů, s kterými se dá potom dále interagovat pomocí skriptovacího jazyka jako je JavaScript. [24]

Kořenový element je tvořen elementem `<html>`. Další běžné rozdělení je dále na `<head>` a `<body>`, tedy hlavičku a tělo dokumentu. Tělo dokumentu pak může obsahovat nejrůznější texty, obrázky, další multimédia a podobně, a dohromady tak utvoří celou webovou stránku. Hierarchii elementů lze vidět na obrázku 2.4.



Obrázek 2.4: Strom elementů DOM (převzato z literatury [17])

CSS

Kaskádové styly CSS (zkratka z anglického Cascading Style Sheets) jsou technologií, která umožňuje přiřadit jednotlivým HTML elementům určitý design a formátování, jako jsou různé barvy, okraje, odsazení od ostatních elementů a další vlastnosti.

K výběru požadovaného elementu z DOMu se používají selektory, které umožňují vybrat elementy na základě jejich názvu značky, atributu a jeho hodnoty, třídy (class), nebo jedinečného ID atributu. Je důležité vybírat elementy co nejpřesněji, aby se styl neaplikoval

i na nechtěné elementy. Selektory také umožňují vybrat více elementů najednou a umožňují definovat různé podmínky výběru. Po výběru elementu je možné definovat konkrétní styly v podobě dvojice *vlastnost*: *hodnota*, viz následující ukázka kódu 2.1.

```
1 body {
2   background-color: lime;
3 }
4 p {
5   color: red;
6   font-size: 23px;
7 }
```

Výpis 2.1: Definice stylů pomocí CSS. Pozadí stránky bude zelené, text v odstavci bude mít velikost 23 pixelů a červenou barvu.

Princip kaskádových stylů znamená, že pokud pro daný element existují dva různé styly, pak platí styl, který byl definován později. V případě, že se v kaskádových stylech používají selektory s různou váhou, platí styl s vyšší vahou. V případě shody vah platí styl, který byl definován později.

Pro CSS existují i různé frameworky, které umožňují kratší zápis stylů, nabízejí předdefinované třídy a předem navržené komponenty jako jsou tlačítka, formuláře a navigace. Nabízejí také systém pro usnadnění tvorby responzivního designu. Tím umožňují vytvářet pěkné webové stránky snadněji a rychleji. Mezi populární frameworky patří například Bootstrap[24], TailwindCSS[25] a Bulma[26]. Každý framework má svůj vlastní přístup k tvorbě webu. [2]

JavaScript

JavaScript je programovací jazyk, který se používá na webových stránkách, a je součástí většiny moderních webových prohlížečů. Je to nejrozšířenější programovací jazyk na světě. Může být vložen přímo do stránky zapsané v HTML.

JavaScript je vysokoúrovňový, dynamický, interpretovaný jazyk. Proměnné v tomto jazyce mohou obsahovat jakoukoliv hodnotu, protože je netypovaný. Tento jazyk se také dobře hodí pro objektové a funkcionální programování. Syntaxe JavaScriptu je odvozena z jazyka Java a dědičnost založená na prototypech vychází z jazyka Self. [8]

Zapsaný JavaScript kód se spouští až v prohlížeči, kde dochází k jeho interpretaci. JavaScript umožňuje definovat a implementovat dynamické chování webových stránek, jako například reagování na interakce uživatele jako kliknutí nebo pohyb myši, dynamické generování určitých prvků na stránce a ovládání otevírání dalších oken.

Je možné využít knihovnu jQuery [27], která je zaměřena na práci s HTML dokumentem. Jedním z hlavních přínosů jQuery je, že poskytuje jednoduché a efektivní způsoby, jak získat přístup k HTML elementům a manipulovat s nimi pomocí CSS selektorů. Díky tomu je práce s HTML dokumentem snadnější a efektivnější než pomocí čistého JavaScriptu (viz ukázka 2.2). [9]

[24] https://getbootstrap.com
[25] https://tailwindcss.com
[26] https://bulma.io
[27] https://jquery.com

```
1 document.getElementById("#button").addEventListener("click", function(){
2    // ...
3 });
4
5 $("#button").click(function(){
6    // ...
7 });
```

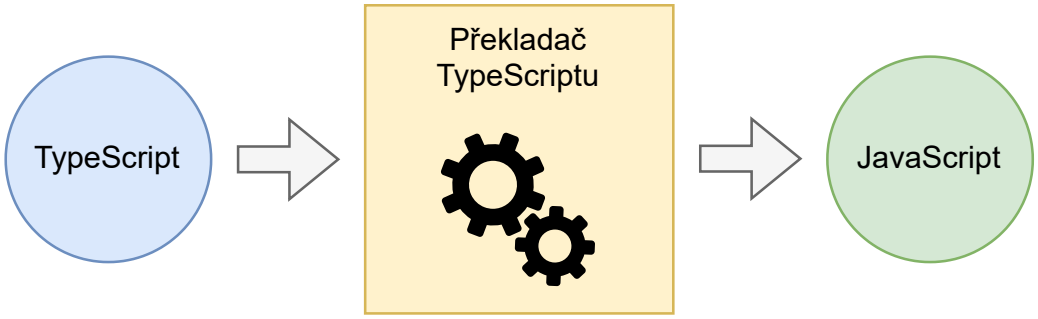
Výpis 2.2: Obsluha při kliknutí na tlačítko, nejprve v samotném JavaScriptu a poté s knihovnou `jQuery`.

TypeScript

TypeScript[28] je programovací jazyk, který poskytuje JavaScriptu typovou kontrolu. Díky této kontrole nedochází k provádění nepovolených operací, což vede k eliminaci mnoha chyb v průběhu vývoje a nasazení aplikace. V JavaScriptu mohou být chyby objeveny až při běhu programu, což může způsobit zbytečné zpoždění při vývoji aplikace a riziko chyb v produkčním prostředí. Příklady nepovolených operací, při kterých vyhodí v JavaScriptu výjimku až při interpretaci jsou například násobení čísla a listu, volání funkce s listem stringů jako vstupním parametrem, když s ním poté funkce zachází jako s listem objektů, nebo import modulu, který byl nedávno přesunut. [3]

Díky silné typové kontrole TypeScriptu můžeme mít jistotu, že jsou všechny proměnné a datové struktury správně definovány. Další výhodou používání TypeScriptu je také lepší dokumentace kódu. S typovou kontrolou je také snadno vidět, jaké typy dat očekává funkce a jaký bude její výstup. TypeScript také umožňuje definovat vlastní typy a rozhraní, což zvyšuje celkovou přehlednost programu.

Kód napsaný v TypeScriptu se vždy překládá a výsledný kód bude v prohlížeči běžet jako běžný JavaScript, znázorněno na obrázku 2.5.



Obrázek 2.5: TypeScript se překládá na JavaScript (převzato z literatury [27]).

Angular

Angular[29] je populární JavaScriptový framework vyvinutý společností Google v roce 2010. Jedná se o jeden z frameworků pro vývoj moderních webových aplikací s vysokou interaktivitou. Angular nabízí připravenou strukturu webové aplikace a je vhodný pro i velké projekty, na kterých pracuje více vývojářů současně.

[28] https://www.typescriptlang.org
[29] https://angular.io

Angular nabízí mnoho funkcí, které usnadňují vývoj aplikací. Například umožňuje vytváření jednostránkových webových aplikací (SPA), kde se načítají pouze potřebné prvky do prohlížeče, místo toho, aby se z serveru načetla celá HTML šablona, jak bylo běžné dříve. Umožňuje vytváření vlastních HTML elementů – komponent, které obsahují vlastní logiku a mohou být opakovaně použity. Celý Angular je napsán v jazyce TypeScript a také se v Typescriptu píše kód pro jednotlivé komponenty. Každá komponenta se skládá ze tří souborů: HTML šablony, TypeScript kódu a souboru s CSS styly. Tyto soubory jsou obvykle uloženy ve stejné složce, která je pojmenována podle názvu komponenty.

Jednou z podstatných funkcí Angularu je tzv. „data binding“ neboli „vazba dat“, což umožňuje snadné propojení kódu s šablonou. Tento mechanismus umožňuje psát kratší kód přímo do HTML a není potřeba ho definovat zvlášť. Další důležitou funkcí Angularu je jeho schopnost zacházet s daty reaktivně, což usnadňuje asynchronní programování.

Nové verze Angularu jsou pravidelně vydávány každých šest měsíců a je důležité ho aktualizovat, aby bylo možné využívat nejnovější funkce a opravy chyb. [22]

Vue.js

Dalším frontendovým frameworkem je Vue.js[30]. Byl vytvořen vývojářem Evanem You v roce 2014. Evan You pracoval na projektu pro společnost Google, kde získal zkušenost s vývojem webových aplikací v Angularu. Rozhodl se vytvořit vlastní framework, který bude o něco jednodušší a flexibilnější než Angular a bude tak více odpovídat jeho vlastím představám[6].

Vue.js podporuje podobně jako Angular vývoj jednostránkových aplikací. Při založení nového projektu se stáhne pouze jádro aplikace a čistý základ Vue.js. Další balíčky a vylepšení k němu jdou přidat postupně dle potřeby. Příkladem je `Vuelidate` pro frontendovou validaci či `router` pro přechod mezi jednotlivými stránkami na jiných URL adresách.

Typický soubor ve Vue.js obsahuje tři hlavní bloky kódu, `<template>` pro HTML šablonu, `<script>` pro JavaScriptový kód a `<style>` pro CSS styly. Tyto tři části jsou tedy od sebe odděleny, ale pořád jsou součástí jednoho souboru. Tento koncept se označuje jako SFC (z angl. single file component).

Vue.js nabízí deklarativní renderování, tedy lze jednotlivé části HTML šablony renderovat na základě řídících direktiv, jako například `v-if`, `v-else`, `v-for` a dynamicky tak měnit obsah webu pro uživatele. Další klíčovou vlastností je reaktivita. Vue.js sleduje stavy jednotlivých zobrazovaných proměnných, a v případě změny hodnoty ihned zareaguje a adekvátně upraví obsah DOMu.

Další výhodou je, že Vue.js má plnou podporu pro TypeScript a projekt tak lze psát v tomto jazyku. Vývoj ve Vue.js tak získá typovou bezpečnost a všechny další výhody TypeScriptu popsané v sekci 2.2.2.

Vue.js nabízí dva způsoby zápisu kódu. Options API a Composition API. Popis a ukázky kódu jsou převzaty z oficiální dokumentace [28].

Options API Logika Option API spočívá ve vytvoření objektu, který obsahuje jednotlivé třídní atributy, jako jsou data, metody a metody životního cyklu komponenty. Příkladem metody životního cyklu může být `mounted`, metoda, která se provede v momentě načtení komponenty do prohlížeče. Typicky v ní proběhne inicializace třídních proměnných a načtení dat ze serveru. K jednotlivým třídním atributům lze přistupovat pomocí klíčového slova `this`, které odkazuje na aktuální instanci komponenty. Zápis komponenty tímto přístupem je na ukázce 2.3.

[30] https://vuejs.org

```
<script>
export default {
  data() {
    return {
      count: 0
    }
  },
  methods: {
    increment() {
      this.count++
    }
  },
  mounted() {
    console.log(‘The initial count is ${this.count}.‘)
  }
}
</script>

<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

Výpis 2.3: Komponenta Vue.js definovaná pomocí Options API. Atributy v `data()` jsou reaktivní a lze k nim přistupovat pomocí klíčového slova `this`.

Composition API V Composition API se logika komponenty definuje pomocí importovaných funkcí API. Zároveň se změní element `<script>` na `<script setup>`, čímž je Vue poskytnuta nápověda, že se používá Composition API. Proměnné a funkce deklarované ve skriptové části jsou poté přímo použitelné v HTML šabloně. Reaktivita proměnných se zajišťuje pomocí klíčového slova `ref` pro jednoduché typy, případně `reactive` pro objekty. Definice komponenty tímto způsobem je na příkladu 2.4.

```
<script setup>
import { ref, onMounted } from ’vue’
const count = ref(0)
function increment() {
  count.value++
}
onMounted(() => {
  console.log(‘The initial count is ${count.value}.‘)
})
</script>

<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

Výpis 2.4: Komponenta Vue.js definovaná pomocí Composition API. Klíčové slovo `ref` zajišťujě reaktivitu proměnné.

React

React[31] je dalším populárním JavaScriptovým frameworkem. Byl vytvořen společností Facebook v roce 2013. React se zaměřuje na tvorbu jednostránkových aplikací a je optimální pro vývoj stránek s rychle se měnícími daty. Místo změny dat přímo v DOM, React nejdříve mění data ve virtuálním DOM, což je mnohem rychlejší. Změny v DOM se provedou až poté, co React uzná za podstatné.

Na rozdíl od kompletních frameworků jako je Angular a Vue.js, se React soustředí pouze na vrstvu View z architektury MVC, tedy vrstvu, která prezentuje data uživateli. Tato vrstva je popsána čistě pomocí JavaScriptu s HTML syntaxí zvanou **JSX** zakomponovanou jako součást kódu. React se proto často využívá s dalšími knihovnami a a existuje kolem něj celý ekosystém s nabídkou nejrůznějších nástrojů, knihoven a pluginů.

Nabízí podobně jako další frontendové frameworky tvorbu vlastních komponent. Komponentu lze definovat buď třídou nebo funkcí, jako je na ukázce 2.5. Tyto komponenty jsou definovány svými vstupy, které poté komponenta už nemůže změnit a vlastním stavem pro data, která se mohou měnit v čase a která komponenta může změnit. React má již zabudovaný systém správy stavu (angl. state management system). [4]

```
1 function Welcome(props) {
2     return <h1>Hello {props.name}!</h1>;
3 }
4 ReactDOM.render(<Welcome name="John"/>, document.getElementById('root'));
```

Výpis 2.5: Ukázka kódu v Reactu, komponenta `Welcome` je definována pomocí funkce (převzato z literatury [18]).

React se dá využít i pro vykreslení stránky na straně serveru, typicky v kombinaci s Node.js[32]. Existuje i projekt React Native[33], který umožňuje vytvoření mobilních aplikací pomocí Reactu a JavaScriptu. [14]

[31] https://react.dev
[32] https://nodejs.org
[33] https://reactnative.dev

Kapitola 3

Analýza požadavků aplikace pro šachový kroužek

V této kapitole představím jednotlivé požadavky na aplikaci. Jedná se o aplikaci pro dětský šachový kroužek, takže by i z hlediska designu uživatelského rozhraní mělo jít o hravou, barevnou a jednoduchou aplikaci k používání.

Nemělo by se jednat o konkurenci školním profesionálním webům nebo profesionálním webům šachových klubů a komerčním systémům. Cílová skupina této aplikace je amatérský šachový kroužek, vedený dobrovolníky a nijak nepropojený se závodní šachovou sférou. Většina takovýchto šachových kroužků řeší veškerou komunikaci a administraci přes uzavřené skupiny na sociálních sítích.

Hlavním cílem této aplikace je zjednodušit komunikaci, poskytnout všechny relevantní informace na jednom místě a nabídnout další zajímavé funkce pro dětský amatérský šachový kroužek, jako je digitální záznam šachových partií a možnost jejich vizualizace.

Požadavky na aplikaci byly získány prostřednictvím osobních konzultací s vedoucím kroužku. Návrh implementace byl průběžně konzultován a připomínky byly zohledněny v návrhu aplikace.

3.1 Evidence členů

Šachový kroužek sestává z jednoho vedoucího a několika studentů. Vedoucí by měl být schopen vést přehled o svých jednotlivých studentech a o jejich informacích, jako je jméno a příjmení, emailová adresa a přezdívka na webu. Je potřeba poskytnout přidávání nových členů, jejich editaci a mazání. Vedoucí kroužku vytvoří nového člena i s jeho přihlašovacími údaji (přezdívka a heslo). Student si své heslo po přihlášení může změnit. Web by měl splňovat zásady GDPR[1], takže pro nepřihlášené uživatele by měla být viditelná pouze přezdívka studenta a žádné soukromé informace.

3.2 Sledování docházky

Šachový kroužek je rozdělený na několik semestrů, pro který jsou definovány jednotlivé dny, kdy kroužek probíhá. Vedoucí i studenti by měli být schopni sledovat docházku. Správu této docházky bude mít na starosti vedoucí, konkrétně bude moci definovat jednotlivé dny

[1] GDPR – General Data Protection Regulation (obecné nařízení o ochraně osobních údajů)

kroužku a poté pro každého studenta zaznamenat, zda se v daný den účastnil či nikoliv. Studentům bude docházka dostupná k nahlédnutí.

3.3 Motivační program

Studenti by měli být v průběhu svého docházení do kroužku i nějak motivováni a odměňováni. Během kroužku se kromě hraní klasických partií šachu dějí i jiné aktivity, jako jsou domácí úkoly, vyplňování učebních sešitů, různé hry k procvičování nejlepších tahů a podobně.

Vedoucí kroužku by měl umět tedy umět definovat jednotlivé aktivity a nastavit pro každou z nich určitou hodnotu, váhu. U každého studenta by poté označil, zda danou aktivitu splnil či nikoliv. Studenti tak mohou během kroužku plnit různé úkoly a získávat za ně body, které se budou sčítat a bude možné i vidět žebříček vítězů, který se bude s každou nově splněnou aktivitou postupně měnit.

Získávání bodů mělo by být implementováno způsobem, který je pro studenty příjemný a motivující k plnění dalších aktivit. Ohodnocení studentů za jejich aktivity by nemělo být příliš kompetitivní nebo stresující a určitě by nemělo být hlavní funkcionalitou této aplikace.

3.4 Turnaje

V rámci kroužku probíhají i turnaje mezi studenty. Každý turnaj je definován datem konání a jeho účastníky. Přihlášení studenti poté hrají hry každý s každým a na konci jsou sečteny výsledky jednotlivých partií a určeno pořadí vítězů. Vedoucí kroužku by měl být schopen vytvářet a upravovat tyto turnaje, a poté zapisovat bodové výsledky jednotlivých partií. Studenti by si potom měli umět výsledky zobrazit.

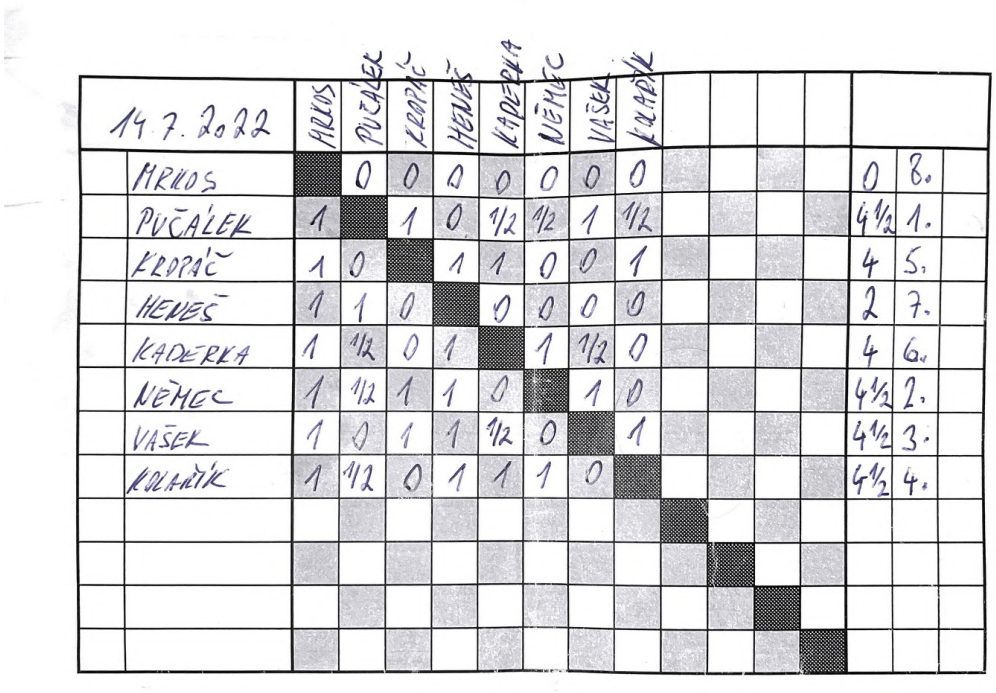
Každá šachová partie má tři možné výsledky, výhra hráče s bílými figurkami, výhra hráče s černými figurkami a remíza, neboli nerozhodný výsledek, kdy není vítězem ani jeden z hráčů. Vítězný hráč získá jeden bod, poražený nula bodů a v případě remízy dostane každý po půlce bodu.

Aktuálně se k zápisu výsledků využívají tabulky vytisknuté na papíře, jako je na obrázku 3.1. Jednotlivá partie je vždy z pohledu hráče v levém sloupci. Na odpovídajícím políčku střetnutí s dalším hráčem lze vyčíst, zda hrál s černou barvou figurek či bílou, zda prohrál, zvítězil, nebo jestli hra skončila remízou. Například hráč Němec (levý sloupec, šestý hráč) při hře s Mrkosem hrál za černé figurky a zvítězil. Markos při hře s Pulčákem naopak hrál za bílé a prohrál.

Tato tabulka je symetrická a je tak potřeba vyplnit výsledek partie z pohledu obou hráčů, což může vést k chybám z nepozornosti při zápise a k chybám ve finálních výsledcích. Tento způsob záznamu by tedy bylo dobré převést do aplikace a umožnit, aby výsledek partie stačilo vyplnit pro jednoho hráče, a aplikace ho automaticky vyplní i pro pohled druhého hráče.

Ve sloupcích napravo potom postupně vidíme sečtené body pro hráče ze všech partií a celkové pořadí podle počtu jeho získaných bodů. V případě, že dva nebo více hráčů dosáhnou stejného počtu bodů, nejsou použita pomocná hodnocení ze závodního šachu[2]. Pro šachový kroužek je jako pomocné hodnocení dostačující pořadí zápisu účastníků v tabulce.

[2] například Buchholz a Sonnenborn-Berger, postupy výpočtu pomocného hodnocení při rovnosti bodů



Obrázek 3.1: Způsob zápisu výsledků šachového turnaje v kroužku.

3.4.1 Záznam partií

Průběh a jednotlivé tahy obou hráčů by bylo dobré mít možnost i nějak zaznamenat. K záznamu se běžně používá formát PGN. Možnost mít tento záznam může pomoci k analýze partie, zaměření se na svá slabá místa a poučení se z nich pro další hry. Tento záznam by k partiím přidával vždy vedoucí a studenti by si ho poté mohli zobrazit. Dále by bylo vhodné jednotlivé tahy figurkami vizualizovat přímo na šachovnici a mít možnost si celou partii vizuálně přehrát, včetně možnosti se ve hře postupně pohybovat po jednotlivých tazích. Přístup k tomuto přehrávači partií by mohl mít i nepřihlášený uživatel.

PGN

K záznamu šachových partií se běžně využívá formát **PGN** (z angl. Portable Game Notation). Je navržený tak, aby byl zpracovatelný počítačem ale zároveň ho dokázal přečíst i člověk. Záznam je v ASCII, a může tak být jednoduše sdílen například jako součást textové zprávy.

Sestává ze dvou hlavních částí, hlavičky a jednotlivých tahů. V hlavičce jsou informace jako název partie, datum, jména bílého a černého hráče a výsledky. Tahy jsou potom číslovány po dvou, vždy je první bílý hráč a poté černý. První písmeno označuje typ figurky, zkratka pochází z anglického názvu figurky. Další písmeno a číslice definují určité políčko na šachovnici. Součástí tahu může být i komentář, je ohraničen složenými závorkami `{komentář}`. [25]

```
[Date "1994.09.10"]
[White "Seirawan, Yasser"]
[Black "Smyslov, Vassily"]
[Event "Interpolis International Tournament"]
1. d4 Nf6 2. c4 e6 3. Nf3 Bb4 4. Nbd2 c5 5. a3 Bxd2 6. Bxd2 d6 7. dxc5 dxc5
8. Qc2 Nbd7 9. Qc7 10. g3 Ng4 11. Bf4 e5 12. Bh3 h5 13. Bxg4 hxg4 14.
Nxe5 Nxe5 15. Rd5 Rh5 16. Rxe5 Rxe5 17. Qh7 f6 18. Rd1 Kf7 19. Qh8 Qc6 20. Rd8
Re8 21. Qh5 Ke7 22. Rd6 Qe4 23. Rd3 Be6 24. Bd6+ Kd8 25. Be5 Bd7 26. Qf7 Qf5
27. Bf4 g5 28. Be3 Qe6 29. Qg7 b6 30. Rd5 Qe7 31. Rxg5 Qxg7 32. Rxg7 Kc7 33. b3
Rxe3 34. fxe3 Rh8 35. e4 Rxh2 36. Kd2 Rg2 37. Rf7 Rxg3 38. e3 Rf3 39. Ke2
{On time} 0-1
```

Výpis 3.1: Příklad záznamu šachové partie ve formátu PGN (převzato z literatury [25])

3.5 Psaní příspěvků

Dále by vedoucí kroužku měl mít možnost psát veřejné příspěvky. Příspěvek by obsahoval název, datum a text. Studenti by potom měli možnost k jednotlivým příspěvkům psát komentáře. Vedoucí by mohl příspěvek i upravit a odstranit.

Dále by bylo dobré rozdělit obecné příspěvky či novinky od pořádaných akcí, které se vážou na konkrétní datum, a vytvořit tak jakýsi kalendář akcí.

Tyto příspěvky, nebo také novinky, by měly býti viditelné hned na hlavní stránce webu.

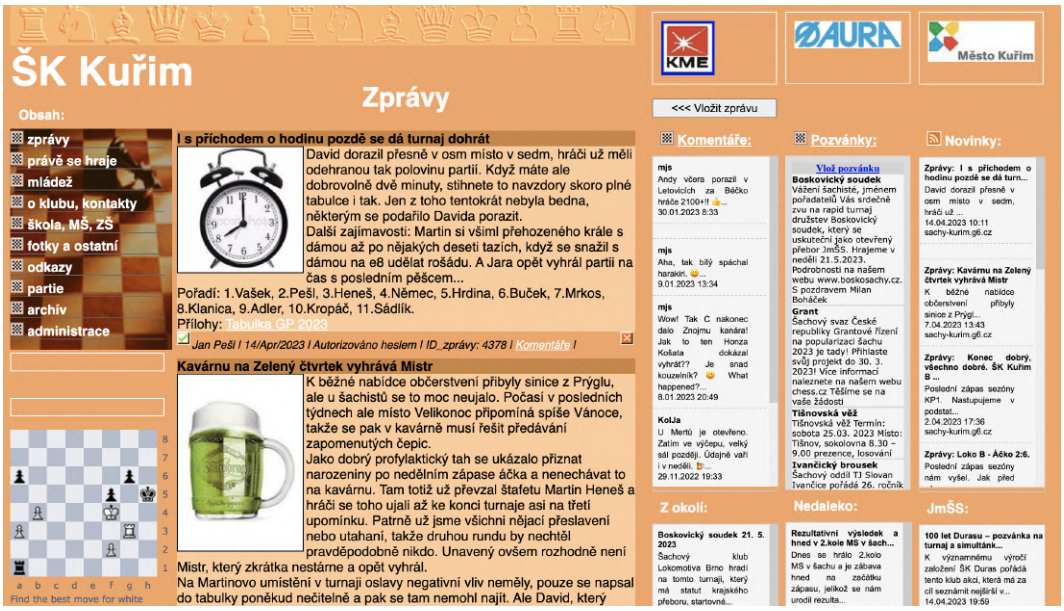
Kapitola 4

Existující řešení

Neprofesionální dětské šachové kroužky většinou ke komunikaci využívají email nebo uzavřené skupiny na sociálních sítích. U školních kroužků jsou to poté školní interní systémy pro komunikaci, jako je například Bakaláři[1] a EduPage [2]. V této sekci se tedy zaměřím spíše na aktuální způsob prezentace šachového klubu a poté na jednotlivé prvky potřebné pro cílenou aplikaci, jako jsou přehrávače partií a záznam výsledků turnaje.

4.1 Web ŠK Kuřim

Aktuálně se pro Kuřimský šachový klub používají webové stránky ŠK Kuřim[3]. Na těchto stránkách jsou informace pro dětský šachový klub společně s informacemi pro klasický šachový klub Kuřim.



Obrázek 4.1: Hlavní stránka webu šachy Kuřim. Na levé stránce je menu, uprostřed sekce zprávy a napravo jsou komentáře a novinky z externích zdrojů.

[1] https://www.bakalari.cz
[2] https://portal.edupage.org
[3] http://www.sachy-kurim.g6.cz

Tyto webové stránky kombinují vlastní obsah s různými odkazy, externími zdroji a widgety třetích stran, a na první pohled tak mohou působit mírně nepřehledně až přehlceně velkým množstvím informací v různých podobách. Na hlavní stránce jsou to hned všechny zprávy a je nutné podotknout, že web nevyužívá stránkování, takže se dá skrolovat opravdu hodně hluboko do historie. Dále jsou zde agregovány a zobrazeny nejnovější komentáře k příspěvkům na webu, pozvánky na různé akce v okolí, kde novou pozvánku může vložit i nepřihlášený uživatel, čímž se web nabízí k možnému zaspamování nevyžádaným obsahem. Dále jsou tu novinky z okolí definované pomocí externích RSS[4] zdrojů hned ve třech sloupečcích. Nechybí i sloupeček s novinkami z vlastních zpráv na webu, takže na hlavní stránce je možné nakonec novou zprávu vidět dokonce dvakrát. Na levé straně stránky pod menu jsou interaktivní widgety rozehrané šachové partie, kde je uživatel vyzván k nalezení nejlepšího možného tahu. V podobném stylu je vytvořen veškerý obsah webu. Hlavní strana webu je na obrázku 4.1.

Registrace na web je zde řešena tak, že se může registrovat kdokoliv a admin poté musí registraci potvrdit. I nepřihlášený uživatel tu vidí možnost „změnit heslo“ a „potvrdit registraci“, zachyceno na obrázku 4.2. Tyto akce je sice potřeba potvrdit heslem, ale lepší by bylo, kdyby byly primárně skryté a viditelné až pro uživatele s potřebným oprávněním.

Obrázek 4.2: Nepřihlášený uživatel vidí akci potvrzení registrace, kterou může vykonat jen admin.

Sekce pro dětský šachový kroužek zde není kompletně oddělena. Záložka v menu „mládež“ obsahuje pouze vlastní sekci zpráv. Případné další informace o dětském šachovém kroužku je třeba hledat mezi ostatními sekcemi na webu kde jsou informace spojené s klasickým šachovým klubem, jako jsou sekce „partie“, „o klubu“ či „fotky a ostatní“.

4.2 Turnaje

Webová stránka Chess Results[5] slouží k uchovávání informací o šachových turnajích a jejich výsledcích z celého světa. Databáze byla vytvořena s pomocí slosovacího programu pro šachové turnaje Swiss Manager[6]. V současné době databáze obsahuje téměř 590 000 oficiálních i neoficiálních šachových turnajů, včetně evropských šampionátů a šachových olympiád. Na stránce lze vyhledávat turnaje podle různých parametrů, jako je název turnaje, orga-

[4] RSS – Really Simple Syndication, XML formát pro syndikaci obsahu
[5] https://chess-results.com
[6] https://swiss-manager.at

nizátor, stát, místo a datum konání. Je také možné vyhledat konkrétního hráče a zobrazit statistiky z turnajů, kterých se účastnil. Stránka umožňuje sledovat aktuálně probíhající turnaje i s postupně aktualizovanými výsledky. Turnaj v aplikace lze vidět na obrázku 4.3.

Nový turnaj se dá založit přes program Swiss Manager, ke kterému je dostupná demo verze. Tato verze je ale omezená a nabízí například pouze vytvoření turnaje se čtyřmi koly. K plné verzi je již potřeba platná licence. Licence pro Českou republiku je dostupná pro členy šachového svazu České republiky s platnou třídou rozhodčího.[10]

Celkově se jedná o užitečnou stránku pro amatéry i profesionály, ale funkce pro amatéry jsou omezené. Losování dvojic pro každé kolo, které tento program nabízí, není pro požadavky kroužku podstatné, jelikož hraje každý s každým a hráči si vybírají další soupeře podle toho, kdo je právě volný.

Liga por equipos de Gran Canaria 2023 - Segunda A

Organizátor	Federación Insular de Ajedrez de Gran Canaria
Federace	Španělsko (ESP)
Ředitel turnaje	Almeida Sánchez, Ciriaco (ID 2236737)
Hlavní rozhodčí	AAC Pujales Llanos, José Manuel (ID 22286284)
Bedenkzeit (Standard)	60 minutos + 30 segundos por jugada
Místo	Gran Canaria - Islas Canarias - España
Number of rounds	10
Tournament type	Každý s každým pro družstva
Výpočet ratingu	Elo národní, ELO mezinárodní
Datum od	2023/02/11 Datum do 2023/05/20
Ø ELO turnaje	1165
Losovací program	Swiss-Manager Heinz Herzog, Turnajový soubor Swiss-Manageru

Poslední aktualizace15.04.2023 21:31:00 / Page cached 15.04.2023 21:40:04 77min., Creator/Last Upload: Spanish Federation (Arbiter Comitee)

Výběr turnaje	Preferente, Primera, ***Segunda A***, Segunda B, Segunda C, Sub 12-16, Sub 8-10
Odkazy	Oficiální stránka organizátora, Odkaz na turnaj v turnajovém kalendáři
Výběr parametru	nezobrazovat detaily o turnaji, bez vlajek
Seznamy	***Křížová tabulka podle pořadí***, Tabulka podle startovní listiny, Startovní listina hráčů, Abecední seznam hráčů, Alphabetical list all groups, Herní plán Sestava družstev s výsledky jednotlivců, Sestava družstva bez výsledků, Los družstev pro všechna kola
Nasazení hráčů	Kolo.1, Kolo.2, Kolo.3, Kolo.4, Kolo.5, Kolo.6
Nejlepší hráči	Board list
Nejlepší hráči	Player performance list
Excel a tisk	Export do Excelu (.xlsx), Export do souboru PDF

Search for team Hledej

Křížová tabulka podle pořadí

Poř.	Družstvo	1a	1b	2a	2b	3a	3b	4a	4b	5a	5b	PH 1	PH 2	PH 3
1	CanariasChess	*	*	2		4		3½		3½	3½	16,5	9	0
2	Caja Vecindario E	2		*	*	3	2½	3½		1½		12,5	7	0
3	Fundación La Caja D	0		1	1½	*	*	2		3		7,5	3	0
4	Club Cadetra	½		½		2		*	*	3		6	3	0
5	ArucasChess B	½	½	2½		1		1		*	*	5,5	2	0

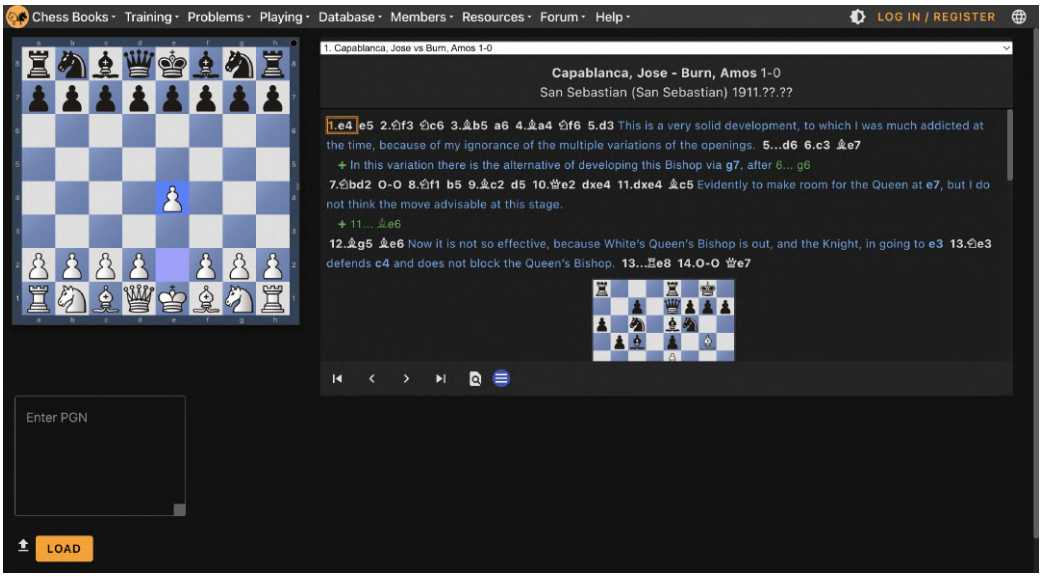
Obrázek 4.3: Detail turnaje na stránce Chess Results. Obsahuje mnoho informací o turnaji jako je datum a místo konání, organizátora a finální výsledky.

4.3 Přehrávače partií

Na internetu je několik možností, kde lze přehrát šachové partie. K záznamu se využívá formát PGN. Aplikace jsou schopné jej zpracovat na jednotlivé tahy a poté je vizuálně znázornit na šachovnici i s animací pohybu.

Chess Tempo

ChessTempo[7] je webová stránka zaměřená na trénování hráčů na všech úrovních, za účelem hrát šachy lépe a dělat lepší tahy. Mezi funkcemi webu patří například analýza otevíracích a koncových her, výuka taktických postupů a různá cvičení. Pro tento účel je na webu k dispozici přehrávač partií ve formátu PGN, zachycen na obrázku 4.4.



Obrázek 4.4: Vizuální přehrávač partií na stránce Chess Tempo

Na levé straně stránky je šachovnice s jednotlivými figurkami, ve které se potom figurky pohybují a uživatelé tak můžou hru sledovat. Na spodním levém rohu je textové pole pro vložení požadované partie hry v PGN notaci. Na hlavním bloku stránky je zobrazena partie, která je rozdělena na jednotlivé tahy. Každý tah je zobrazen s ikonkou reprezentující typ figurku a také její umístění na hrací desce. Komentáře jsou modře zvýrazněny. Ve spodní části tohoto bloku jsou ovládací ikonky, umožnující se posouvat ve hře po jednotlivých tazích tam a zpět, ale i přeskočit celou hru na první a poslední tah. Při pohybu na další tah se také aktuální tah zvýrazní oranžovým ohraničením.

Jedná se o velmi dobré a praktické řešení. Jediné, co se mi nezdá je, že figurky na šachovnici jsou uchopitelné myší a lze s nimi pohnout, ovšem po upuštění figurky se figurka vrátí zpět na své původní místo. Uchopitelné figurky jsou zde tedy zbytečné.

Ingram Braun

Jedná se o přehrávač partií, který je součástí osobní stránky vývojáře Ingrama Brauna[8]. Tento web obsahuje i možnost komentářů, podle kterých jsou uživatelé s funkčností spokojení. Přehrávač je zachycen na obrázku 4.5.

Jedná se o podobné řešení jako na stránce Chess Tempo, ale nachází se tu o dost více tlačítek. U některých tlačítek není poznat, co daná akce znamená, protože po najetí neobsahují žádnou nápovědu a z ikonky to není jednoznačné. Také jsou tu dva druhy ikonek naznačující pohyby ve hře vpřed a vzad, jedny pod šachovnicí a druhé pod notací hry,

[7] https://chesstempo.com
[8] https://ingram-braun.net/erga/online-pgn-viewer

Obrázek 4.5: Vizuální přehrávač partií na stránce Ingrama Brauna

nicméně ty druhé po stisknutí nic nedělají. Oproti Chess Tempo je tu i možnost obrátit klávesnici a vidět tak hru z pohledu hráče druhé barvy, což může být pro analýzu hry určitě užitečné. Vidíme zde i fotku hráče, takže na základě hlavičky souboru a jména hráčů si aplikace pravděpodobně i vyhledá člověka v nějaké databázi a zobrazí jeho fotku.

ŠK Kuřim

Na webu šachového klubu Kuřim se při kliknutí na nahranou partii otevře nové okno s načtenou hrou do přehrávače. Tento přehrávač, zachycený na obrázku 4.6, nabízí šachovnici s figurkami, pod ní ovládání s tlačítky, obsahující akce vpřed, vzad, dopředu a dozadu celé hry. Stisknutím tlačítka plus se automaticky celá hra začne pomalu přehrávat sama, což jde poté i zastavit. Vpravo od šachovnice je PGN s hlavičkou a v notaci jsou zkratky pro figurky nahrazené příslušnou ikonkou. Jedná se o jednoduché a elegantní řešení.

Obrázek 4.6: Vizuální přehrávač partií na stránce šachového klubu Kuřim

Kapitola 5

Návrh řešení

Na základě získaných požadavků na aplikaci od uživatele jsem vytvořila návrh na řešení aplikace.

5.1 Diagram případů užití

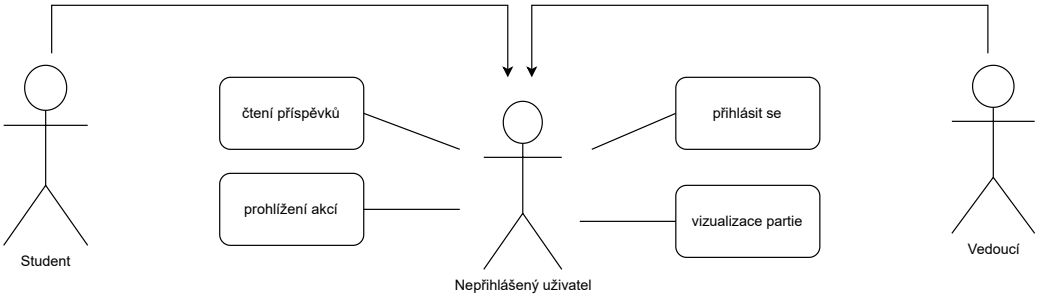
Tento diagram (anglicky use-case diagram) ukazuje různé typy uživatelů systému a akce, které mohou vykonávat. Tato aplikace bude mít tři typy uživatelů: nepřihlášený uživatel, student a vedoucí. Diagram bude dále rozebrán podle těchto typů uživatelů a kompletní verze diagramu je v příloze B.

Nepřihlášený uživatel

Nepřihlášený uživatel je uživatel, který se buď ještě nepřihlásil, nebo nemá v aplikaci účet. Obrázek 5.1 znázorňuje akce tohoto uživatele. Tento uživatel má přístup k hlavní stránce s příspěvky a plánem akcí klubu. Jednotlivé příspěvky si může prohlížet, otevřít a přečíst si i případné komentáře. Nepřihlášený uživatel má také přístup k části aplikace pro vizualizaci partií, kde si po nahrání vlastního záznamu může přehrát danou hru po jednotlivých tazích.

V patičce hlavní stránky budou k dispozici informace o klubu, včetně adresy a kontaktních údajů, což uživatelům umožní snadno kontaktovat klub v případě jakéhokoli požadavku, včetně zájmu o registraci.

Samozřejmě bude mít také možnost se do systému přihlásit. Úspěšně přihlášený uživatel se následně dělí na **vedoucího** kroužku a **studenta**.



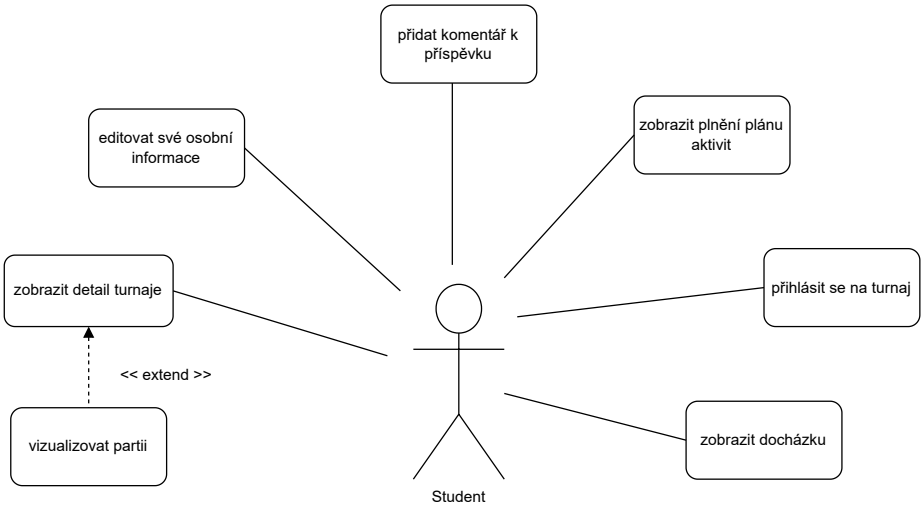
Obrázek 5.1: Akce nepřihlášeného uživatele, vychází z něj vedoucí a student.

Student

Student je typ přihlášeného uživatele s možnostmi akcí nepřihlášeného uživatele, ale také s dalšími specifickými akcemi. Kromě prohlížení článků na webu a přehrávání partií, může přidávat vlastní komentáře a upravovat či mazat své komentáře. Student také vidí nadcházející a proběhlé turnaje v kroužku a může se k nadcházejícím turnajům přihlásit. Pokud se chce odhlásit, může tak učinit před začátkem turnaje. V turnaji má přístup k detailům, jako je název, datum konání a popis, a může si prohlížet finální výsledky jednotlivých partií a žebříček vítězů. Pokud k dané partii existuje nahraný záznam, může si ho přehrát v přehrávači partií.

Student vidí aktuální stav plánu aktivit a počet získaných bodů. Student může sledovat svou docházku a aktualizovat své osobní údaje v systému. Dále má možnost změnit své heslo.

Všechny akce studenta jsou znázorněny na obrázku 5.2.

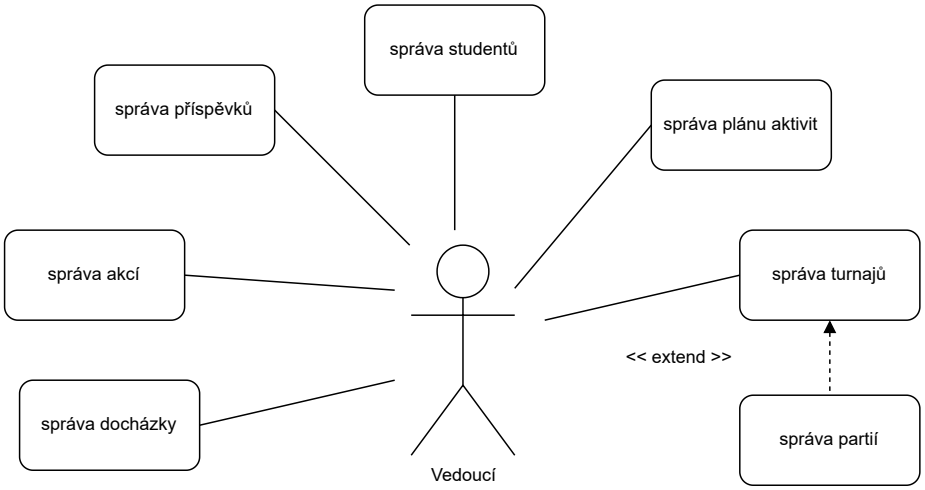


Obrázek 5.2: Akce studenta

Vedoucí

Vedoucí je další typ přihlášeného uživatele. Má na starosti organizaci celého kroužku, čemuž odpovídají jeho možnosti v systému. Všechny akce lze vidět na obrázku 5.3.

Může psát a upravovat příspěvky, vkládat komentáře a definovat kalendář akcí. Dalším úkolem vedoucího je správa studentů, a proto má právo přidávat a odebírat uživatele a upravovat jejich údaje. Dále má možnost definovat dny kroužku a následně označit docházku studentů v daný den. Vedoucí rovněž stanovuje plán aktivit pro kroužek a má tak možnost přidávat a upravovat jednotlivé aktivity. K aktivitě lze přiřadit název a hodnotu, a poté ji pro studenty označit jako splněnou nebo nesplněnou. Vedoucí má také právo vytvářet a upravovat turnaje. V turnaji poté může pro účastníky zaznamenávat výsledky jednotlivých partií. Kromě toho má možnost nahrávat záznamy jednotlivých partií.



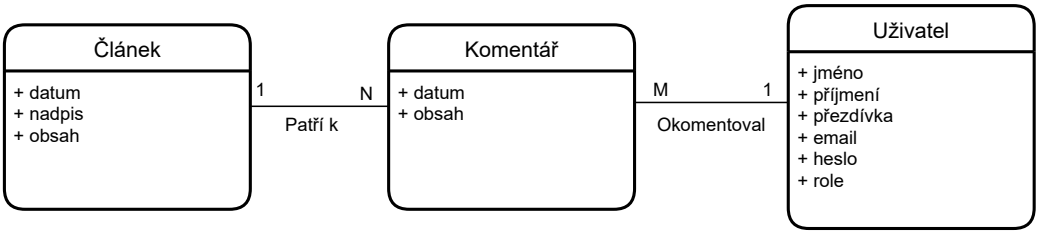
Obrázek 5.3: Akce vedoucího kroužku

5.2 ER diagram

Vytvořila jsem ER diagram (z angl. Entity Relation diagram), který slouží k specifikaci a vymezení jednotlivých konceptů a entit v systému a popisuje jejich vzájemné vztahy. Tento způsob návrhu je velmi užitečný pro pozdější modelování databáze. Každá entita v diagramu má svůj název a parametry, které ji charakterizují. Vztahy mezi entitami jsou reprezentovány relacemi, které mají na každém konci určenou kardinalitu, což znamená maximální počet vztahů, které mohou být navázány mezi instancemi této entity. Celý diagram je v příloze C.

Články a komentáře

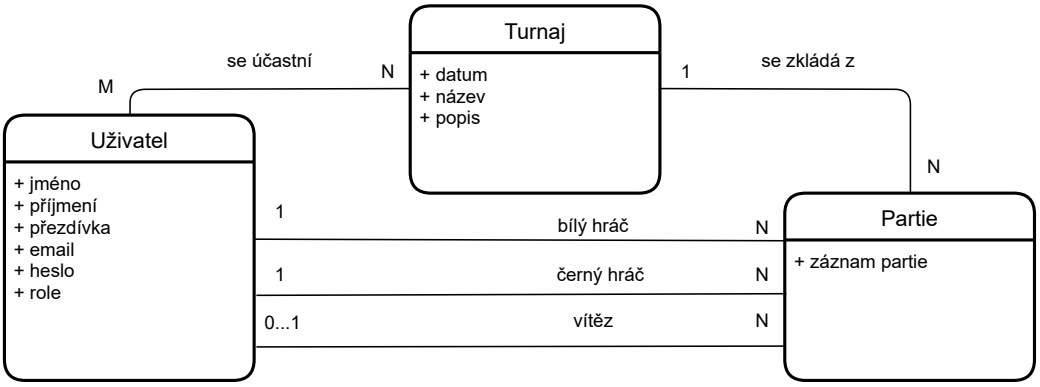
V diagramu je hlavní entitou uživatel, který má své parametry, jako je jméno, příjmení, email, heslo a roli. Uživatel může být buď vedoucí nebo student. Kromě uživatelů jsou zde i články, které budou vytvářet vedoucí. Vzhledem k tomu, že v systému bude pouze jeden vedoucí, není nutné přímo vázat články na vedoucího. V případě, že by v budoucnu bylo možné mít více vedoucích v systému, by bylo nutné tuto vazbu zavést. Kromě toho může být každý článek okomentován několika komentáři, přičemž každý komentář bude navázán na konkrétního uživatele. Uživatel bude má možnost napsat kolik komentářů, kolik bude chtít. Tento vztah je znázorněn na obrázku 5.4.



Obrázek 5.4: Vztah uživatele k článku a komentáři

Turnaje a partie

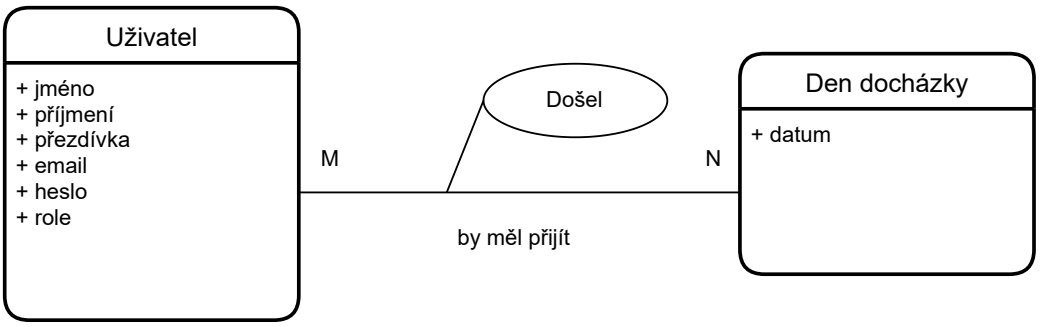
Další částí návrhu je zahrnutí turnajů. Každý uživatel bude mít možnost účastnit se několika turnajů, které budou mít určitý počet přihlášených účastníků. Každý turnaj se bude skládat z jednotlivých partií, ve kterých hráči budou hrát proti sobě. Každá partie bude patřit právě k jednomu turnaji. V každé partii bude třeba určit hráče, který bude hrát s bílými figurkami a který s černými figurkami. Dále bude potřeba určit vítězného hráče, tedy kromě případu nerozhodného výsledku. Tento vztah je znázorněn na obrázku 5.5.



Obrázek 5.5: Vztah uživatele k turnaji a partii

Docházka

Kroužek se obvykle koná jednou denně v určitých dnech v týdnu. Pro sledování docházky bude existovat entita, která definuje konkrétní den, kdy se lekce koná. Pro každého studenta bude zaznamenáno, zda se daného dne zúčastnil či nikoliv. V jeden den se může zúčastnit více studentů a každý student se během kurzu účastní několika hodin, jako je znázorněno na obrázku 5.6.

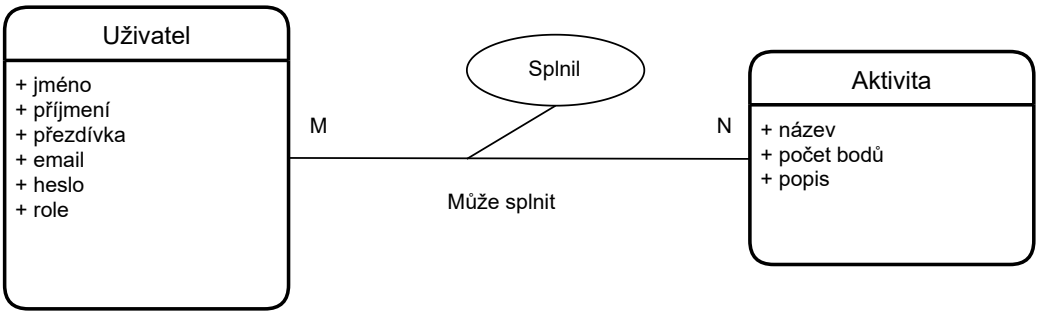


Obrázek 5.6: Vztah uživatele ke dni docházky

Aktivity

Pro definování aktivit během kroužku je zde vytvořena entita Aktivita. Aktivita je charakterizována názvem, počtem bodů udělených za její splnění a doplňujícím popiskem. Během průběhu kroužku mohou studenti postupně plnit jednotlivé aktivity. Každá aktivita může

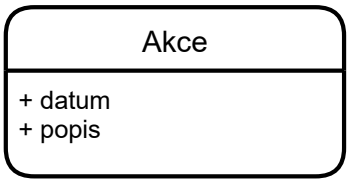
být splněna více studenty. Vztah mezi studentem a aktivitou také určuje, zda je aktivita splněna či nikoliv. Tento vztah je na obrázku 5.7.



Obrázek 5.7: Vztah uživatele k aktivitě

Akce

Poslední entitou je akce, viz obrázek 5.8. Akce je definována dnem konání, názvem a dalším popisem. Účel akce je hlavně informativní a není potřeba ji vázat na konkrétního uživatele. Ovšem v případě rozdělení kroužku na více družstev, by byly potřebné vazby přidány.



Obrázek 5.8: Entita akce v ER diagramu

5.3 Architektura aplikace a zvolené technologie

Systém bude fungovat na základě třívrstvé architektury, která oddělí prezenční vrstvu, aplikační vrstvu a databázovou vrstvu. Uživatel bude interagovat s aplikací pomocí prezenční vrstvy, která poběží přímo v prohlížeči.

5.3.1 Backend

Pro implementaci serverové aplikace jsem zvažovala použití frameworků Django, Laravel a Symfony. Jakýkoliv z těchto výběrů by z mnoha hledisek nebyl špatný. Po prozkoumání několika programovacích tutoriálů jsem zjistila, že se mi více líbí styl psaní backendu v PHP než v Pythonu, takže jsem Django vyloučila. Nicméně, se Symfony již mám zkušenosti z jednoho školního projektu, a proto jsem se rozhodla naučit se něco nového a vyzkoušet pro mě nový framework – Laravel. Jako databázi jsem poté zvolila MySQL, kvůli předchozí zkušenosti a snadné integraci s Laravel aplikací.

5.3.2 Frontend

Původně jsem plánovala realizovat celý projekt pouze pomocí backendového frameworku s využitím MVC architektury a šablonovacího systému. Nicméně, když jsem prozkoumávala

knihovny pro práci se šachovnicí, zjistila jsem, že jsou většinou napsané v JavaScriptu. To mě vedlo k rozhodnutí se dozvědět více o frontendových frameworcích. Mezi nimi mě nejvíce zaujal Vue.js, jelikož je vhodný i pro menší projekty a má podstatně jednodušší syntaxi než Angular. Dále se mi líbí, že Vue.js rozděluje kód komponenty do tří částí, a připadá mi tak přehlednější než kód v Reactu.

Nakonec jsem z vlastní zkušenosti zjistila, že programování v čistém JavaScriptu není moc příjemné. Například pro práci s většími datovými strukturami, které chodí z backendu, nedokáže vývojové prostředí našeptávat jednotlivé atributy, a na příčinu mnoha chyb musím dlouho přicházet až při běhu aplikace, i když byly způsobených třeba pouhým přepsáním se v názvu proměnné. Proto jsem se nakonec rozhodla projekt psát v TypeScriptu, který jde ve Vue.js nastavit již při vytváření projektu.

Aplikační rozhraní

Komunikace mezi backendovou a frontendovou aplikací bude probíhat pomocí aplikačního rozhraní s využitím principů **REST API**. Backend bude fungovat jako API, které komunikuje s frontendovou aplikací pomocí HTTP požadavků. Namísto odesílání celých HTML stránek bude server odesílat pouze konkrétní data, která jsou požadována frontendovou aplikací.

Návrh základních koncových bodů API s typem HTTP požadavku:

- [POST] /login – přihlášení
- [POST] /logout – odhlášení
- [GET, POST, DELETE, PUT] /user/:id – uživatelé
- [GET, POST, DELETE, PUT] /news/:id – novinky
- [GET, POST, DELETE, PUT] /activity/:id – aktivity
- [GET, POST, DELETE, PUT] /tournament/:id – turnaje
- [GET, POST, DELETE, PUT] /event/:id – akce

5.4 Návrh UI

Další důležitou součástí návrhu je uživatelské rozhraní. Vzhledem k tomu, že se jedná o aplikaci primárně určenou pro děti, je důležité, aby uživatelské rozhraní bylo co nejjednodušší a přehledné. Pro dosažení tohoto cíle je vhodné použít zábavné barvy a kulaté tvary HTML prvků. Návrh jednotlivých stránek uživatelského rozhraní byl průběžně konzultován s vedoucím kroužku.

Pro tvorbu uživatelského rozhraní jsem se rozhodla dále využít Bootstrap pro předdefinované CSS třídy a knihovnu s komponenty PrimeVue[1] pro Vue.js, která nabízí design komponent vhodný i pro dětskou aplikaci.

[1] https://primevue.org

Kapitola 6

Implementace

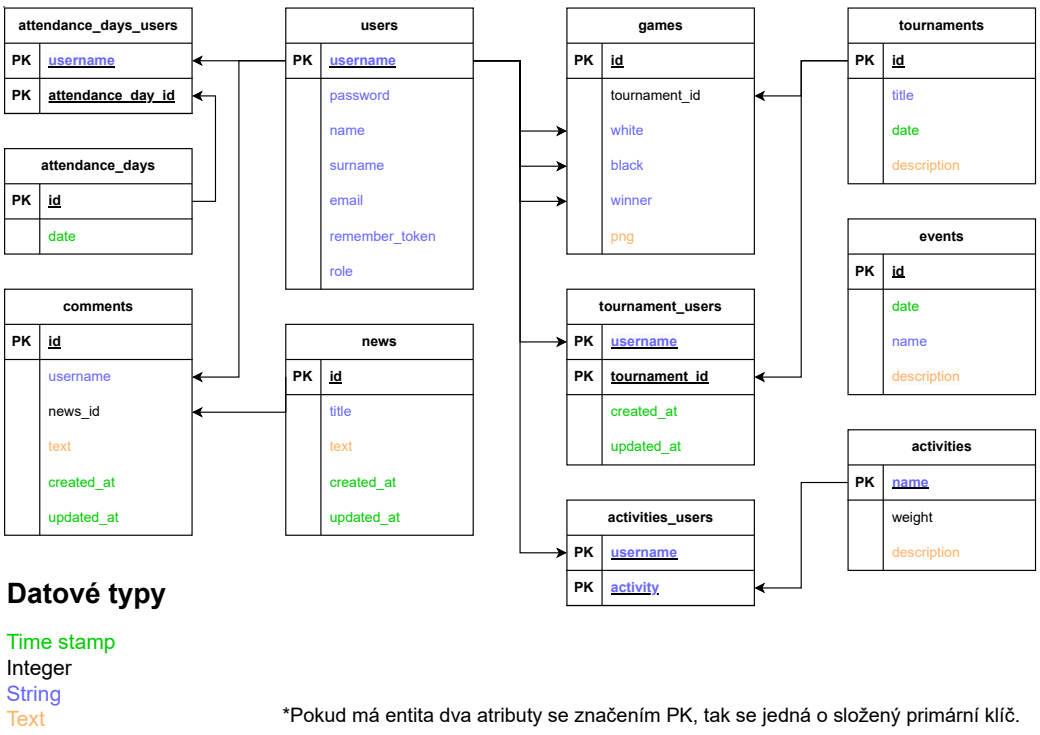
V této kapitole popíši implementaci výsledné aplikace pomocí vybraných technologií. Kapitola je rozdělena na backend a frontend, a dále jsou pak popsány zajímavé části aplikace a jejich implementace.

6.1 Backend

Backend tvoří MySQL databáze a serverová aplikace napsaná v PHP frameworku Laravel.

6.1.1 MySQL – databázová vrstva

Podle navrhnutí systému ER diagramem (příloha C), jsem vytvořila schéma databáze pomocí migrací v Laravelu. Výsledné schéma databáze je na obrázku 6.1.



Obrázek 6.1: Schéma databáze

Na schématu jsou barevně odlišeny datové typy atributů, modře je značený datový typ `string`, zeleně je datum s časem, textový typ je žlutě a modře jsou celá čísla. V tabulce jsou i primární a cizí klíče. Při primárním klíči je zkratka `PK`. V případě, že se jedná o složený primární klíč, je tato zkratka u všech atributů, ze kterého se primární klíč skládá. Z cizích klíčů vede šipka k odkazovanému klíči pro označení relace mezi entitami.

6.1.2 Laravel – aplikační vrstva

Na implementaci jsem využila PHP framework Laravel. Hlavní část implementace tvoří migrace, modely, ovladače a definice koncových bodů API.

Migrace

Migrace slouží k definování změn v databázi jako je vytvoření či smazání tabulky, přidání nových sloupců do již existující tabulky a podobně. Pro vytvoření tabulky pomocí migrace se definuje struktura tabulky, tedy jaké sloupce jakého typu obsahuje, včetně primárních a cizích klíčů, jako je na ukázce kódu 6.1. Všechny migrace jsou uloženy v adresáři **database/migrations** a provedou se příkazem `php artisan migrate`.

```
1  class CreateTournamentsTable extends Migration{
2      public function up(){
3          Schema::create('tournaments', function (Blueprint $table) {
4              $table->id();
5              $table->string('title');
6              $table->timestamp('date');
7              $table->text('description')->nullable();
8          });
9      }
10     //...
11 }
```

Výpis 6.1: Migrace databáze implementující vytvoření nové tabulky turnaje. Obsahuje definici jednotlivých atributů tabulky a jejich datových typů.

Modely

V Laravelu jsou modely základním prvkem pro interakci s databází a reprezentují entity v databázi. Pomocí modelů můžeme komunikovat s databází prostřednictvím ORM, kde jednotlivé třídy reprezentují jednotlivé tabulky a atributy třídy reprezentují atributy tabulky v databázi.

Důležitými atributy modelů jsou **fillable** a **hidden**. Fillable určuje, které atributy v tabulce mohou být naplněny nebo změněny pomocí Laravelu. Hidden atribut určuje, které atributy by měly být skryté při serializaci. Typicky se skrývá heslo uživatele, nebo přidružená pivot tabulka, aby nebyly vždy posílány zároveň s entitou na frontendovou část aplikace.

V Laravelu je potřeba explicitně definovat některé atributy, které nejsou automaticky pro model předpokládány. Například, pokud tabulka v databázi neobsahuje časové známky, musí to být v modelu definováno pomocí `public $timestamps = false`, aby bylo možné správně sestavit SQL dotaz. Dalším příkladem může být primární klíč. Předpokládaným primárním klíčem je totiž atribut ID typu `integer`. Například v modelu entity uživatele

`User` používám jako primární klíč uživatelské jméno, a ve třídě je to explicitně definováno pomocí atributů `$primaryKey = 'username'` a `$keyType = 'string'`.

V modelu jsou také definovány metody pro získání vazeb mezi entitami, jako je například metoda `users()` v modelu `Tournament`, která získává všechny uživatele, kteří se tohoto turnaje účastní (viz kód 6.2). Pro definování těchto vztahů se v Laravelu používají metody `belongsToMany()` pro vztah N:1 a `hasMany()` pro opačný vztah 1:N. Vztah N:N je možné definovat pomocí `belongsToMany()` z pohledu obou potřebných modelů. Kromě toho jsou k dispozici metody `hasOne()` a `belongsTo()` pro definování vztahu 1:1.

```
1  class Tournament extends Model{
2      public $timestamps = false;
3      protected $fillable = ['title', 'date', 'description'];
4      protected $hidden = ['pivot'];
5
6      public function users(){
7          return $this->belongsToMany(User::class,
8          'tournaments_users', 'tournament_id', 'username');
9      }
10
11     public function games(){
12         return $this->hasMany(Game::class);
13     }
14 }
```

Výpis 6.2: Implementace třídy turnaje v Laravel. Obsahuje definici atributů a vazeb pro práci s modelem v databázi.

Ovladače

Ovladač (anglicky Controller) implementuje jednotlivé metody pro konkrétní model a provádí různé dotazy do databáze. Ovladače jsou umístěny v adresáři **app/Http/Controllers**. V mém případě server funguje jako aplikační rozhraní, takže ovladač implementuje metody, které se vykonají po odeslání požadavku na konkrétní body API. Vstupním parametrem je typicky požadavek `request`, a výstupem je odpověď v podobě dat, hlášky nebo chybový kód. Příklad ovladače je na ukázce kódu 6.3.

```
1  class EventController extends Controller{
2      public function index(){
3          $upcoming = Event::where('date', '>=', now())
4              ->orderBy('date')->get();
5          $passed = Event::where('date', '<', now())
6              ->orderByDesc('date')->get();
7          return [ 'upcoming' => $upcoming, 'passed' => $passed];
8      }
9  }
```

Výpis 6.3: Ovladač akcí – implementuje metodu `index()` pro zobrazení všech akcí s rozdělením na nadcházející a již proběhlé.

Koncové body API

Jednotlivé API endopointy jsou definované v souboru **app/routes/api**. Každý endpoint definuje cestu, zda vyžadují přihlášeného uživatele, typ HTTP požadavku a následně metodu v příslušném ovladači, která se má vykonat po přijetí požadavku na tento endpoint. Endpointy jsou tvořeny pomocí přístupu REST API. Pro tuto aplikaci je zde vytvořeno celkem 46 endpointů. Například v následující ukázce kódu 6.4, pro získání všech akcí je endpoint `/event`, a po poslání GET požadavku na něj se zavolá metoda `index()` z ovladače `EventController`.

```
1 Route::get('/event', [EventController::class, 'index']);
2 Route::get('/event/{event}', [EventController::class, 'show']);
3 Route::middleware('auth:sanctum')->post('/event',
4     [EventController::class, 'store']);
5 Route::middleware('auth:sanctum')->delete('/event/{event}',
6     [EventController::class, 'destroy']);
7 Route::middleware('auth:sanctum')->put('/event/{event}',
8     [EventController::class, 'update']);
```

Výpis 6.4: Definice API endpointů pro entitu `akce` v Laravel podle přístupu REST API.

6.2 Frontend

Klientská část aplikace je implementována pomocí frameworku Vue.js.

6.2.1 Vue.js

Vue.js je v projektu ve verzi tři a využívám přístupu Composition API. Projekt je celý místo JavaScriptu napsaný v TypeScriptu, pro předání toho faktu do composition API se ve skriptové části přidá navíc atribut definující jazyk `<script setup lang="ts" >`.

Projekt je logicky členěn do složkové struktury.

- **src/main.ts** – jedná se o jeden z hlavních souborů celé aplikace. Importují se zde jednotlivé komponenty, aby byly globálně přístupné a inicializuje se zde kořenová komponenta aplikace App.vue.
- **src/App.vue** – jedná se o kořenový soubor hierarchie všech komponent. V tomto souboru mimo jiné definuji hlavní menu a patičku aplikace, aby byla viditelná na všech stránkách. Také jsou zde implementovány metody pro přihlášení a odhlášení uživatele, protože tyto funkce jsou pro uživatele dostupné právě z hlavního menu.
- **src/views** – tato složka obsahuje jednotlivé stránky aplikace rozdělené do vlastních složek (rozdělení podle stránek na rozdílných URL). Každá z těchto podsložek obsahuje primární `.vue` soubor a další vlastní případné komponenty využité touto stránkou.
- **src/views/other** – tato složka obsahuje vedlejší stránky, jako je „Přístup odepřen“ a „Stránka nenalezena“.
- **src/shared/interface.ts** – soubor obsahující definici všech datových rozhraní. Využíváno především na definici struktury dat, které chodí ze serveru (na ukázce 6.5 je

to definice struktury dat pro akce). Jsou vytvořeny pomocí `export interface` a dají se tak využít jako datový typ v celém projektu.

```
export interface IEvent {
  id: number;
  name: string;
  date: string;
  description: string;
}

export interface IEventResponse {
  upcoming: IEvent[];
  passed: IEvent[];
}
```

Výpis 6.5: Definice typového rozhraní v TypeScriptu

- **src/components** – v této složce jsou menší znovu využitelné komponenty. Je zde například Footer.vue pro definici patičky aplikace a TableLoading.vue, která definuje kostru tabulky, ve které zrovna probíhá načítání dat ze serveru.
- **src/router/index.ts** – soubor pro definici jednotlivých URL a komponent, které se mají při návštěvě daného URL inicializovat. URL mohou být definovány i regulérním výrazem a obsahovat dynamickou část, jako je na ukázce 6.6.

```
{
    path: "/tournaments/:id",
    name: "tournament",
    component:()=> import("../views/tournaments/TournamentView.vue"),
    props: true,
},
{
    path: "/:pathMatch(.*)*",
    component:()=> import("../views/other/PageNotFound.vue"),
},
```

Výpis 6.6: Směrování komponent na URL ve Vue.js

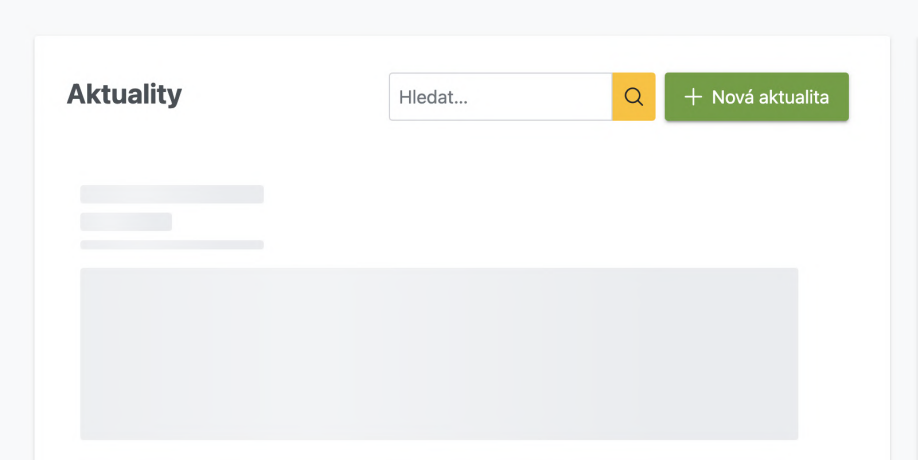
6.2.2 PrimeVue

Z knihovny komponent PrimeVue jsem využívala velké množství komponent, protože ulehčují vývoj a poskytují jednotný styl designu pro celou aplikaci. Komponenty lze po importování do příslušného souboru a využívat podobně jako HTML tagy.

Mezi nejvyužívanější komponenty patří například:

- **InputText** – textové vstupní pole, velmi využívané ve formulářích.
- **Button** – klasické tlačítko, umožňuje i snadné vložení ikonky.
- **Dialog** – dialogové okno, které se otevře přes obsah stránky. Obsahuje i zavírací ikonku v pravém horním rohu. Dále umožňuje snadno definovat nadpis a tlačítka akcí.

- **Calendar** – vstupní pole pro výběr data v podobě kalendáře. Umožňuje i výběr konkrétního času.
- **Paginator** – komponenta pro stránkování, například příspěvků. Této komponentě se dají vstupní hodnoty jako je celkový počet příspěvků a požadovaný počet příspěvků, a komponenta poté reaguje na interakce uživatele při přechodu mezi stránkami.
- **Toast** – komponenta zobrazující vyskakující notifikace. Notifikace je pro celý projekt, takže zůstane viditelná i poté, co uživatel přejde na jinou URL.
- **Icons** – PrimeVue nabízí i vlastní knihovnu ikonek. Ikonky jsem se na UI snažila používat co nejvíce, protože tvoří aplikaci přívětivější pro uživatele.
- **Skeleton** – komponenta umožnující vytvořit kostru nějakého elementu, který slouží jako dočasný obsah (anglicky placeholder). Díky Skeletonu lze vytvořit dočasný obsah nejrůznějších tvarů a velikostí. Tento element jsem využívala například při načítání dat ze serveru do tabulek, či příspěvků jako je na obrázku 6.2.



Obrázek 6.2: Využití komponenty Skeleton z PrimeVue pro vytvoření dočasného obsahu při načítání příspěvku ze serveru

6.2.3 Bootstrap

Protože mi všechny komponenty z PrimeVue ne vždy vyhovovaly, je pro některé části využit i Bootstrap. Například při návrhu tabulek PrimeVue umožňuje pouze definici hlavičky tabulky buď jako první sloupec, nebo jako první řádek. V některých případech jsem potřebovala mít jako hlavičku obojí a v PrimeVue to jednoduše nevypadalo dobře. Také jsem Bootstrap používala pro snadné nastavené okrajů, odsazení a snadné zacházení s prvky s flex vlastnostmi.

6.2.4 Použité knihovny

Mezi použité externí knihovny se řadí zejména ty využívané na frontendové aplikaci. K realizaci vizuálního přehrávače partií jsem použila knihovny chess.js[1] a vue3-chessboard[2]. Pro posílání HTTP požadavků na server jsem použila knihovnu Axios[3].

Chess.js

Jedná se o šachovou knihovnu v TypeScriptu, která umí vytvořit abstraktní šachovou hru a je schopná validovat a generovat možné tahy. Umí také pracovat s PGN notací hry a abstraktní hru lze načtením PGN inicializovat. Pro nevalidní PGN vrátí knihovna chybu. Po načtení hry tímto způsobem lze použít metodu `history()` a získat hru rozloženou na jednotlivé tahy, včetně políček definující **odkud** a **kam** se figurkou táhlo. Knihovna umí také získat komentáře z PGN pomocí metody `getComments()`.

Vue3-chessboard

Tato knihovna určená přímo pro Vue.js ve třetí verzi nabízí grafickou vizualizaci šachovnice a figurek. Šachovnice se vloží do HTML šablony pomocí `<TheChessboard>` elementu, a inicializuje se pomocí `@board-created="(api) => (boardAPI = api)"`, kde `boardAPI` je reference na třídu implementující jednotlivé metody pro zacházení se šachovnicí. Pomocí `board-config` jdou nastavit další vlastnostmi klávesnice, já jsem například nastavila šachovnici jen na prohlížení, aby uživatel nemohl provádět tahy přímo pomocí uchycení a posunutí figurkou a vypnula jsem zobrazování souřadnic šachovnice. Použití této šachovnice je také na následující ukázce kódu 6.7.

```
1  const boardAPI = ref<ChessboardAPI>();
2  const boardConfig: BoardConfig = {
3    coordinates: false,
4    viewOnly: true,
5  };
6  //...
7  <div class="chessboard">
8      <TheChessboard
9          :board-config="boardConfig"
10         @board-created="(api) => (boardAPI = api)"
11     />
12 </div>
```

Výpis 6.7: Vložení a inicializace vizuální šachovnice pomocí knihovny `Vue3-chessboard`

Třída `ChessboardAPI` poté nabízí metody pro vizuální alternaci šachovnice jako je `resetBoard()` pro navrácení figurek na počáteční pozici, `undoLastMove()` pro vrácení posledního provedeného tahu, `move()` pro provedení dalšího tahu, kde se jako parametry metody vloží atributy z jakého políčka na jaké políčko, a šachovnice už provede tah figurkou ležící na odpovídajícím políčku. Mimo jiné dále také nabízí metodu `toggleOrientation()` pro prohození bílých a černých figurek, aby se uživatel mohl podívat na partii z pohledu druhého hráče.

[1] https://github.com/jhlywa/chess.js
[2] https://github.com/qwerty084/vue3-chessboard
[3] https://axios-http.com/docs/intro

Axios

Jedná se o knihovnu pro posílání HTTP požadavků. Axios oproti výchozí `fetch()` metodě například rovnou dokáže transformovat a vložit přijatá data ze serveru přímo do objektů v JavaScriptu, kdežto u `fetch()` se musí data explicitně transformovat pomocí metody `response.json`. Podobně je to i naopak, při posílání dat na server se při `fetch()` musí data z objektů explicitně přetypovat na textový řetězec pomocí `JSON.stringify`. Axios nabízí také další možnosti, které umožňují zkrátit kód potřebný k poslání požadavku a zpracování odpovědi. [11]

6.3 Vizuální přehrávač partií

Vizuální přehrávač partií je implementován pomocí knihoven **chess.js** a **Vue3-chessboard** popsaných v sekci 6.2.4. Výsledná forma implementace v aplikaci je na obrázku 6.3.

Soubor PGN se načte do hry v `chess.js` a rozloží se na jednotlivé tahy. Jedná se o strukturu typu pole, takže každý tah je na určitém indexu. Pomocí indexů je možné se v poli pohybovat vpřed a vzad, na čemž je založen princip ovládání partie pomocí tlačítek. V proměnné `index` si uchovávám aktuální pozici v seznamu tahů.

Pro pohyb vpřed po kliknutí na příslušné tlačítko se zavolá funkce `nextMove()`. V té se podle aktuálního indexu vybere políčko `Square` [4] **odkud** a políčko **kam**. Tyto dvě hodnoty se potom dají jako parametry funkce z knihovny Vue3-chessboard `makeMove()` a figurka se animovaně posune na šachovnici, viz ukázka kódu 6.8. Při zmáčknutí tlačítka zpět se zavolá funkce `lastMove()`, která funguje na podobném principu a zavolá metodu šachovnice `undoLastMove()`. Při každém pohybu se kontroluje, zda se jedná o první či poslední možný index a nastaví se proměnné `isNextDisabled` a `isPrevDisabled` typu `boolean`. Pokud jsou tyto proměnné nabývají pravdivé hodnoty `true`, tak se vypne příslušné tlačítko. Tedy podle toho, zda už nejsou další možné pohyby vpřed nebo vzad.

```
1 var fromSquare = <Square>history.value[index.value].from;
2 var toSquare = <Square>history.value[index.value].to;
3
4 boardAPI.value?.makeMove(fromSquare, toSquare);
```

Výpis 6.8: Pohyb na vizuální šachovnici podle historie tahů

Nevýhoda metody `history()` je to, že rozložená hra na tahy bohužel neobsahuje komentáře. Komentáře tedy bylo potřeba získat jinak. Použila jsem metodu `getComments()` z knihovny`chess.js`, která vrátí strukturu typu seznam obsahující komentáře a příslušnou pozici zapsanou ve formátu FEN[5]. Pro namapování zápisu pozice touto formou na aktuální pozici ve hře tak bylo nutné se kromě vizuálního pohybu na šachovnici ještě pohybovat v abstraktní hře z `chess.js`, protože ta dokáže pro aktuální stav hry vygenerovat i příslušnou FEN notaci. Po vytvoření pozice v této notaci se kontroluje, zda seznam komentářů obsahuje právě tento FEN řetězec, a pokud ano, tak je komentář zobrazen uživateli.

Pro zvýraznění aktuálního tahu jsem využila dynamické renderování elementů. Pomocí `v-if` a `v-else` se zobrazuje element s třídou `higlighted` či element bez této třídy. Pokud

[4] Datový typ Square je alias definující možné typy polí v šachovnici, jako je třeba `a8` či `g6`

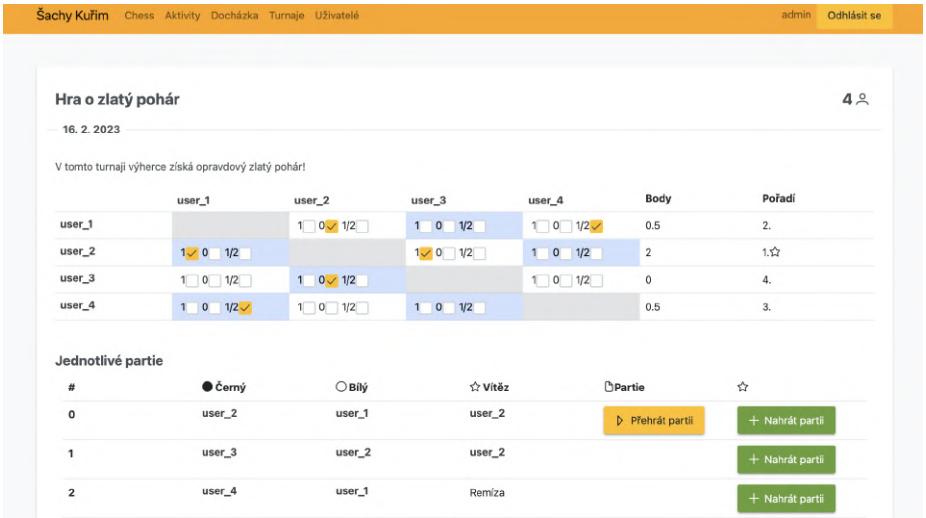
[5] Notace pro zapsání aktuální pozice figurek na šachovnici, na rozdíl od PGN notace není pro člověka lehce čitelná

element splňuje podmínku, že odpovídá aktuálnímu tahu, aplikuje se na třídu `highlighted` definovaný CSS styl a aktuální pozice je tak pro uživatele zvýrazněna.

Obrázek 6.3: Implementace přehrávače partií, aktuální tah je zvýrazněn.

6.4 Turnaje

Turnaje jsou rozděleny na dvě obrazovky. První obrazovka slouží jako seznam všech turnajů, dají se filtrovat podle nadcházejících a již proběhlých. V náhledu každého turnaje lze vidět název, datum konání, počet přihlášených a tlačítka s jednotlivými akcemi (přihlásit, odhlásit, smazat, upravit). Přihlásit se lze pouze na nadcházející turnaje.



Obrázek 6.4: Detail turnaje

Další obrazovka se poté zobrazí po kliknutí na konkrétní turnaj, tato obrazovka je na obrázku 6.4. Obsahuje výsledky turnaje v tabulce, která je vytvořená podle šablony popsané v kapitole 3.4. V tabulce vedoucí zadá výsledek partie pro dva hráče výběrem jedné hodnoty ze tří `checkboxů`: značící výhru, prohru a remízu. Jelikož je tabulka symetrická, po zakliknutí jednoho výsledku se v ní i automaticky vyplní výsledek pro pohled druhého hráče. Nakonec jsou v tabulce sečteny body a určeno pořadí vítězů. V případě shodného počtu bodů je pořadí určené podle pořadí zápisu uživatele v tabulce.

Tato tabulka se generuje dynamicky v HTML šabloně podle počtu účastníků. Zajímavé je zde generování barev jednotlivých buněk. Ty jsou implementované pomocí dynamického přiřazení CSS třídy na základě splnění podmínky. Tato třída pak určuje, zda bude daná buňka obarvena šedě (jedná se o průnik hráče se sebou samým – partie neprobíhá), nebo modře (naznačuje hráče s černými figurkami). Pokud buňka nemá přidělenou žádnou třídu, zůstává neobarvená a značí hráče s bílými figurkami. Pravidlo pro generování třídy kromě střídání barev také závisí na tom, zda se jedná o buňky nad nebo pod diagonálou, kde se barvy navíc posunují o jedno políčko. Tento princip je znázorněn v následující ukázce kódu 6.9.

```
1 <tr v-for="(user, iRow) in tournament.users">
2 <td v-for="(userColumn, iCol) in tournament.users"
3     :class="{
4         'table-secondary':
5             user.username === userColumn.username,
6         'table-primary':
7             (iCol < iRow && (iCol + iRow) % 2 === 1) ||
8             (iCol > iRow && (iCol + iRow + 1) % 2 === 1)}">
9 </tr>
```

Výpis 6.9: Dynamické generování třídy pro obarvení buňky v tabulce výsledků

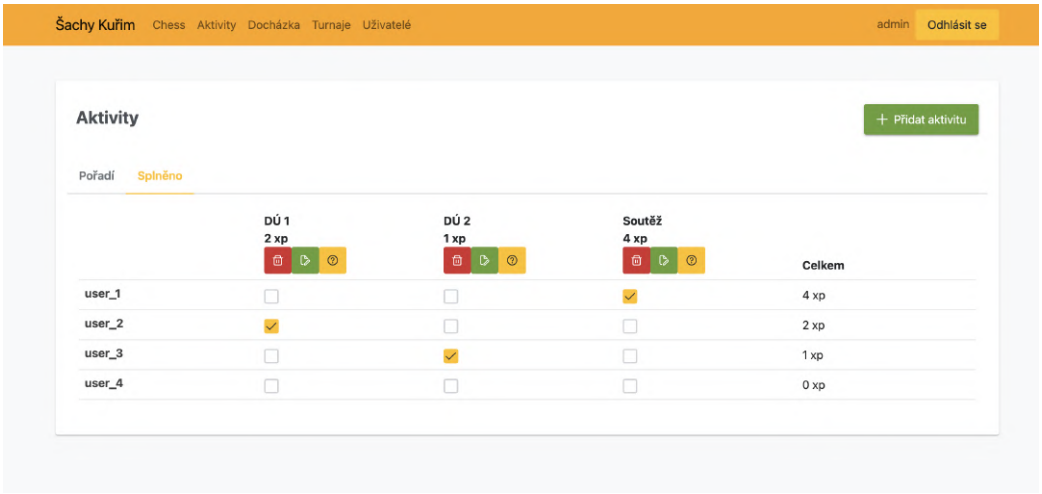
Pod tabulkou s výsledky se nachází seznam jednotlivých partií, do kterého může vedoucí turnaje dodat záznam v PGN notaci (viz 3.4.1). Tento záznam je před zasláním na server pro uložení validován pomocí knihovny `chess.js`. Uložený záznam si poté mohou přihlášení uživatelé přehrát. Po kliknutí na tlačítko se přenesou na stránku s přehrávačem partií, kde bude záznam již předvyplněn.

6.5 Aktivity

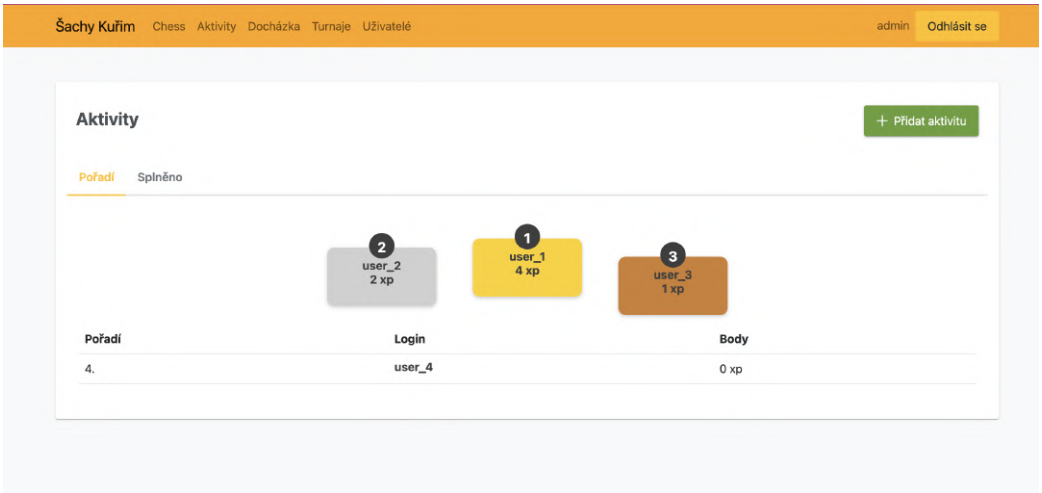
Aktivity slouží i k motivování studentů plnit různé aktivity.

K zápisu splněných aktivit pro vedoucího je použita tabulka, ve které může měnit hodnotu `checkboxu` odpovídající konkrétnímu studentovi a aktivitě. Provedené změny v této tabulce se ihned odesílají na server. U jednotlivých aktivit je i tlačítko s otazníkem, které po zmáčknutí zobrazí doplňující popisek aktivity. Je zde zobrazen i součet získaných bodů pro každého studenta. Výsledná podoba zápisu aktivit je na obrázku 6.5.

Dalším náhledem pro větší motivaci studentů je i forma v podobě vizuálních stupínků vítězů. Student na prvním místě podle získaných bodů je zobrazen zlatou barvou, student na druhém místě je barvou stříbrnou a student na třetím místě je barvou bronzovou. Další studenti jsou poté seřazeni v tabulce pod stupínky. Barva vítězů se poté i odráží na profilu studenta. Tato forma zobrazení lze vidět na obrázku 6.6.



Obrázek 6.5: Tabulka pro zápis a zobrazení splněných aktivit



Obrázek 6.6: Pořadí studentů ve splněných aktivitách v podobě stupní vítězů

6.6 Autentifikace a autorizace

Pro přihlášení a ověření identity uživatele používám `bearer tokeny`. Po zadání uživatelského jména a hesla se ověří, zda se shodují s údaji uloženými v databázi. Pokud je ověření úspěšné, server vygeneruje `bearer token`, což je náhodný řetězec znaků, který si uloží a zároveň pošle v odpovědi zpět na frontendovou aplikaci. Tento token se poté uloží do lokální paměti prohlížeče (anglicky local storage), jako je na ukázce kódu 6.10. Díky uloženému tokenu tak zůstane uživatel přihlášený a nemusí při každé návštěvě aplikace opakovaně zadávat své přihlašovací údaje.

```
1 localStorage.setItem("token", response.data.token);
2 token.value = localStorage.getItem("token");
```

Výpis 6.10: Uložení a opětovné získání autorizačního tokenu z lokální paměti prohlížeče

Pro autorizaci, neboli oprávnění uživatele k provedení konkrétních akcí se také používají `bearer tokeny`. Pokud uživatel požaduje vykonání určité akce, například mazání turnaje nebo psaní komentáře, přidá se jeho token jako součást autorizační hlavičky k HTTP požadavku. Server si poté ověří, zda má uživatel oprávnění k vykonání dané akce na základě spojení tokenu s konkrétním uživatelským účtem.

Koncové body API mohou být v Laravelu zabezpečeny pomocí `middleware Sanctum`. Sanctum ověří platnost tokenu při přístupu k danému koncovému bodu a v případě neúspěchu vrátí v odpovědi chybový kód, viz ukázka 6.11.

```
1 Route::get('/news', [NewsController::class, 'index']);
2 Route::middleware('auth:sanctum')->post('/news',
3 [NewsController::class, 'store']);
```

Výpis 6.11: Zabezpečení API endpointů. Pro získání novinek nemusí být uživatel přihlášený, ale pro vytvoření novinky již ano.

Práva uživatele jsou definována pomocí jeho role. Momentálně existují dva typy rolí: **student** pro studenta a **admin** pro vedoucího. Vedoucí má přístup ke všem akcím, které může provést student, a také k dalším specifickým akcím, které jsou určené pouze pro něj. Po ověření přístupu přes autentizační bránu se v jednotlivých metodách již pouze ověřuje, zda se v těchto specifických akcích jedná o admina, jak je uvedeno v ukázce kódu 6.12.

```
1 public function store(Request $request){
2     if($request->user()->role != 'admin'){
3         return response()->json(['message' => 'Unauthorized'], 401);
4     }
5     //...
6 }
```

Výpis 6.12: Autorizace uživatele. Tuto akci (vytvoření nového článku) může provést pouze uživatel s rolí admin. V jiném případě je vrácen standardní chybový kód 401.

6.7 Další použité technologie

Při implementaci jsem využívala i další technologie a aplikace, konkrétně **Github**[6] jako zálohovací a verzovací systém, vývojové prostředí **Visual Studio Code**[7], **Prettier**[8] pro konzistentně formátovaný kód a **Trello** [9] pro lepší plánování a organizaci jednotlivých úkolů v průběhu implementace aplikace.

[6] https://github.com
[7] https://code.visualstudio.com
[8] https://prettier.io/
[9] https://trello.com

Kapitola 7

Testování

Aplikaci jsem nejdříve testovala v průběhu vývoje a později i s uživatelem. Jednotlivé poznatky z testování jsem poté zapracovala do aplikace.

7.1 Testování během vývoje

Během vývoje jsem aplikaci manuálně testovala. Snažila jsem se ji testovat co nejdůkladněji vždy po dokončení nové funkcionality, protože odhalení a okamžité řešení chyb je rychlejší a efektivnější než je opravovat až po testování s reálnými uživateli nebo v produkční verzi. Dále jsem aplikaci testovala jako celek a sestavila možné scénáře užití. Při testování aplikace jako celku jsem například zjistila, že definice CSS stylů se občas přenášejí z jedné komponenty nebo stránky na druhou, a bylo proto nutné vytvářet specifičtější třídy pro CSS selektory. Díky manuálnímu testování jsem odhalila menší i větší chyby již během vývoje.

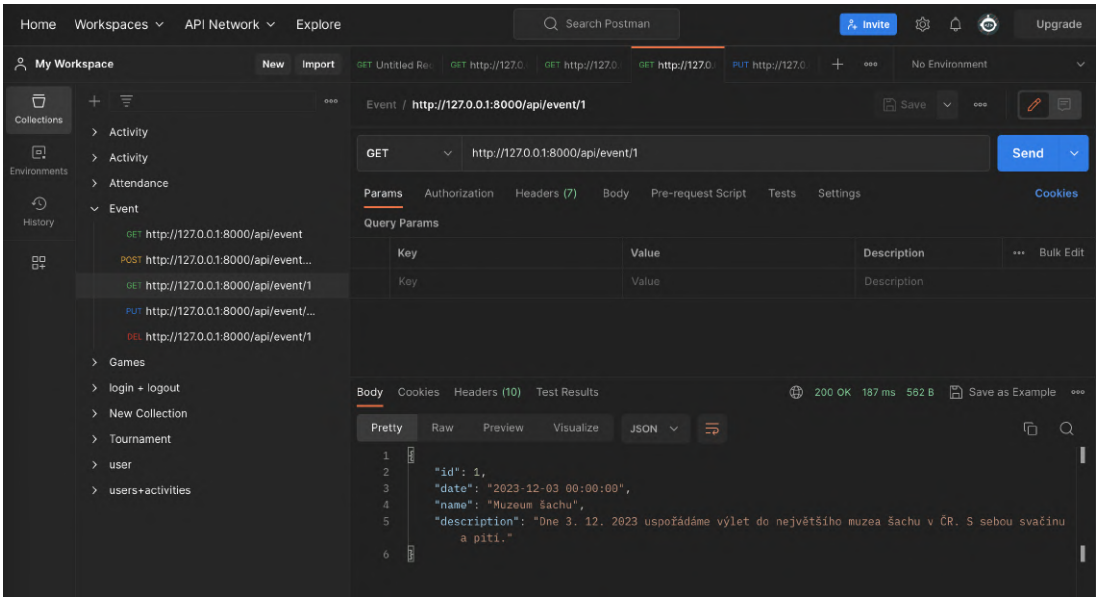
7.1.1 Testovací data v databázi

Během vývoje jsem využila možnost definovat počátečních data databáze pomocí Laravelu. Tento proces (anglicky database seeding) umožňuje naplnit prázdnou databázi předem definovanými počátečními daty, což mi umožnilo v případě potřeby snadno vrátit databázi do počátečního stavu. Ušetřilo mi to spoustu času, protože jsem se při manuálním testování vyhnula ručnímu opakovanému vkládání dat do databáze. Když jsem například smazala článek, všechny jeho komentáře byly také automaticky smazány. Bez této možnosti bych musela pro účely dalšího testování opět ručně vytvořit článek spolu s všemi jeho komentáři, což by zabralo hodně času.

7.1.2 Testování koncových bodů API

Díky oddělení frontendové a backendové části aplikace bylo možné testovat jednotlivé koncové body API a ověřit tak jejich správnou funkčnost, návratové hodnoty a typy chyb ještě před propojením s uživatelským rozhraním. Pro testování těchto endpointů jsem použila aplikaci **Postman**[1], která umožňuje definovat a uložit si jednotlivé požadavky včetně parametrů a následně je odeslat na server. Pro každý endpoint se také volí typ HTTP požadavku a hlavička, do které lze případně vložit autentizační token pro testování zabezpečení a přístupů. Odpověď ze serveru se následně také zobrazí v aplikaci.

[1] https://www.postman.com



Obrázek 7.1: Testování koncových bodů API v aplikaci **Postman**

Díky testování API endpointů odděleně jsem mohla vždy otestovat serverovou část zvlášť před propojením s frontendem a zajistit tak, že požadovaná data jsou uživateli správně doručena. Nebo při zjištění chyby s daty jsem mohla určit, zda se jedná o chybu frontendu nebo backendu. V aplikaci jsem si vytvořila složky podle typu požadavku a v nich uložila jednotlivé koncové body. Na obrázku 7.1 je zachyceno testování bodu pro získání dat o akci s `ID 1` ve zmíněné aplikaci.

7.2 Uživatelské testování

Aplikace jsem testovala i s uživatelem, konkrétně s vedoucím šachového kroužku. Tento typ testování probíhal průběžně během vývoje aplikace, aby se možné chyby či doplnění specifikace požadavků vyřešilo co nejdříve. Vedoucí například poznamenal, že bylo vhodné na hlavní stránce aplikace rozdělit akce na nadcházející a již proběhlé, a aby se dalo vyhledávat ve článcích. Také bylo dodatečně omezena maximální váha aktivity ze 100 na 10. Vedoucí měl také dodatečný požadavek na automatické odesílaní emailu s přístupovými údaji na email nově registrovaného studenta. Dále jsme také vyřešili různé drobnější chyby, jako jsou překlepy v názvech tlačítek.

Kapitola 8

Možnosti dalšího vývoje

V této kapitole se zaměřím na analýzu možností rozšíření systému v budoucnosti. Tyto části nejsou pro hlavní účel systému aktuálně nezbytné, avšak dlouhodobě by určitě přidání několika funkcionalit neuškodilo, naopak by přineslo více možností jak pro studenty, tak i pro vedoucí.

8.1 Profil pro rodiče

V systému by bylo možné vytvořit nový typ uživatele – rodič nebo zástupce studenta. Tento uživatel by měl přístup do systému podobně jako student, ale s několika odlišnostmi v jeho akcích. Například by mohl posílat zprávy přímo vedoucímu, jako jsou omluvenky. Dále by mohl kontrolovat docházku studenta a plnění aktivit. V současné době by se tento profil od studentského příliš nelišil, kromě toho, že by rodiče psali komentáře pod svým vlastním loginem a nikoliv pod loginem svého dítěte. Tento požadavek by se mohl více uplatnit v budoucnu, pokud by bylo nutné oddělit některé akce pro rodiče a pro studenty. V současnosti by bylo možné náhled do systému pro rodiče řešit jednoduchým sdílením hesla.

8.2 Domácí úlohy

V rámci systému by mohl být zaveden modul pro zadávání domácích úkolů nebo úloh, za něž by studenti získávali bonusové body. Knihovna pro vizualizaci šachovnice by mohla být využita pro vytvoření šachových úloh. Například by se zobrazila šachovnice v určitém stavu a úkolem by bylo zahrát nejlepší možný tah. Tento tah by mohl být buď předem definován, nebo by mohla umělá inteligence dynamicky vyhodnotit výsledek.

Dále by bylo možné do systému integrovat šachovou hru, kde by se hrálo proti počítači a bylo by možné nastavit různé stupně obtížnosti. Nicméně, existuje mnoho volně dostupných řešení pro hraní šachu na internetu, a proto by bylo nutné zvážit, zda by taková integrace měla pro studenty opravdovou hodnotu. Výhodou by mohlo být, že by se studentům v případě výhry automaticky přičítaly nějaké bonusové body.

8.3 Více vedoucích

V současnosti má systém pro účely klubu pouze jednoho vedoucího, ale v budoucnu by bylo vhodné umožnit definování více vedoucích. Tím by bylo možné, aby každý vedoucí mohl spravovat a vést svůj tým studentů nezávisle na ostatních.

8.4 Přizpůsobení na mobilní zařízení

Dalším plánem je přizpůsobit uživatelské rozhraní aplikace na mobilní zařízení. V současné době se samozřejmě dá aplikace i na mobilním zařízení používat, ale není to úplně optimální a zobrazení některých elementů by bylo dobré upravit a zpříjemnit pro uživatele. Aktuálně je problém zejména při zobrazování tabulek s mnoha sloupci.

8.5 Pomocné hodnocení

Aktuálně je pomocné hodnocení pro seřazení výsledků turnaje v případě shody počtu bodů prováděno na základě pořadí zápisu v tabulce turnaje. Do budoucnosti musí být jako hlavní pomocné kritérium výsledek vzájemného zápasu, a pokud ani ten nerozhodne, tak teprve poté pořadí v tabulce.

Kapitola 9

Závěr

Cílem této bakalářské práce bylo prostudovat současné technologie pro tvorbu interaktivních webových aplikací, provést analýzu požadavků pro dětský šachový kroužek, navrhnout možné řešení a následně aplikaci implementovat a otestovat.

Aplikace na základě požadavků uživatele umožňuje mnoho interakcí. Pro vedoucího kroužku je to správa studentů, kontrola docházky, přidávání aktualit a vytváření turnajů. Studenti mohou komentovat aktuality a přihlašovat se na turnaje. Výsledky turnaje a jeho jednotlivé partie je možné uložit do systému, kde si studenti mohou výsledky prohlédnout a také si mohou partie z turnaje vizualizovat v interaktivním přehrávači partií. Je zde zakomponován i motivační program pro studenty, kde je možné získávat v průběhu kroužku body za plnění bodovaných aktivit a umístit se na stupni vítězů.

Při průzkumu existujících řešení jsem zjistila, že amatérské šachové kluby využívají ke komunikaci hlavně email či uzavřené skupiny na sociálních sítích. Tento konkrétní kroužek vlastní webové stránky nemá a využívá stránky klubu šachy Kuřim, kde jejich funkce není úplně oddělena od klubu pro dospělé. Dále jsem prozkoumala existující řešení pro zaznamenávání turnajů a vizuální přehrávače partií.

Pro implementaci klientské aplikace jsem zvolila framework Vue.js, kde jsem použila programovací jazyk TypeScript a knihovnu komponent PrimeVue. Na serverové části aplikace jsem pracovala s PHP frameworkem Laravel a databází MySQL.

Aplikace byla testována během průběhu vývoje i s vedoucím kroužku, a připomínky byly implementovány do řešení, jako je například automatické odeslání přístupových údajů na emailovou adresu uživatele při vytvoření jeho profilu. Během testování jsem používala aplikaci Postman pro testování koncových bodů API a využívala možnosti naplnit databázi počátečními testovacími daty.

Na závěr jsem provedla analýzu možností budoucího vývoje aplikace. Dal by se například vytvořit profil pro rodiče, či implementovat možnost mít kroužek rozdělený na více vedoucích, kde by každý spravoval svůj vlastní tým studentů.

V rámci této práce byla vytvořena demo aplikace, která splňuje požadavky zadání a je plánované plné nasazení.

Naučila jsem se pracovat s novými technologiemi a vytvářet koncové body API pomocí přístupu REST. Celkově se mi práce s použitými technologiemi zamlouvala a ráda je znovu použiji i v dalších projektech.

Literatura

[1] Abba, I. V. *What is an ORM – The Meaning of Object Relational Mapping Database Tools* [online]. 2022 [cit. 2023-04-26]. Dostupné z: https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools.

[2] Bose, S. *Top 5 CSS Frameworks for Developers and Designers* [online]. 2023 [cit. 2023-02-04]. Dostupné z: https://www.browserstack.com/guide/top-css-frameworks.

[3] Cherny, B. *Programming TypeScript: Making Your JavaScript Applications Scale*. 1. vyd. O'Reilly Media, 2019. ISBN 1492037656.

[4] Chinnathambi, K. *Learning React*. 1. vyd. Addison-Wesley Professional, 2016. ISBN 0134546318.

[5] Cocca, G. *Different Types of APIs – SOAP vs REST vs GraphQL* [online]. 2023 [cit. 2023-05-03]. Dostupné z: https://www.freecodecamp.org/news/rest-vs-graphql-apis/.

[6] Cromwell, V. *Between the Wires: An interview with Vue.js creator Evan You* [online]. 2017 [cit. 2023-04-04]. Dostupné z: https://www.freecodecamp.org/news/between-the-wires-an-interview-with-vue-js-creator-evan-you-e383cbf57cc4/.

[7] *What Is an Object-Oriented Database?* [online]. [cit. 2023-04-13]. Dostupné z: https://www.mongodb.com/databases/what-is-an-object-oriented-database.

[8] Flanagan, D. *JavaScript: The Definitive Guide: Activate Your Web Pages*. 6. vyd. O'Reilly Media, 2011. ISBN 0596805527.

[9] *JavaScript vs jQuery: Key Differences You Need to Know* [online]. 2022 [cit. 2023-02-04]. Dostupné z: https://www.edureka.co/blog/javascript-vs-jquery/.

[10] Karali, T. *Swiss-Manager Uživatelský návod* [online]. Uživatelský manuál. 2020 [cit. 2023-04-16]. 53 s. Dostupné z: http://swiss-manager.at/unload/swiss_manager_user_guide_CZE.pdf.

[11] Kelhini, F. *Axios vs. fetch(): Which is best for making HTTP requests?* [online]. 2022 [cit. 2023-04-26]. Dostupné z: https://blog.logrocket.com/axios-vs-fetch-best-http-requests.

[12] Mendes, A. *10 types of web applications and how you can use them* [online]. 2022 [cit. 2023-04-05]. Dostupné z: https://www.imaginarycloud.com/blog/10-types-of-web-applications-and-how-you-can-use-them.

[13] *Django MVT* [online]. [cit. 2023-04-05]. Dostupné z: https://www.javatpoint.com/django-mvt.

[14] Máca, J. *Lekce 1 - Úvod do Reactu* [online]. [cit. 2023-04-10]. Dostupné z: https://www.itnetwork.cz/javascript/react/zaklady/uvod-do-react/.

[15] *Usage statistics of server-side programming languages for websites* [online]. 2023 [cit. 2023-05-02]. Dostupné z: https://w3techs.com/technologies/overview/programming_language.

[16] Powell, T. A. *HTML & CSS: The Complete Reference*. 5. vyd. McGraw Hill, 2010. ISBN 0071496297.

[17] Purewal, S. *Learning Web App Development: Build Quickly with Proven JavaScript Techniques*. 1. vyd. O'Reilly Media, 2014. ISBN 9781449370190.

[18] *What is React?* [online]. [cit. 2023-04-10]. Dostupné z: https://www.w3schools.com/whatis/whatis_react.asp.

[19] Roomi, M. *5 Advantages and Disadvantages of Web Application, Drawbacks & Benefits of Web Application* [online]. 2021 [cit. 2023-04-04]. Dostupné z: https://www.hitechwhizz.com/2021/04/5-advantages-and-disadvantages-drawbacks-benefits-of-web-application.html.

[20] Sakhibgareev, K. *PHP Frameworks: Choosing Between Symfony and Laravel* [online]. [cit. 2023-04-12]. Dostupné z: https://www.toptal.com/php/choosing-between-symfony-and-laravel-frameworks.

[21] Schildt, H. *Java: A Beginner's Guide*. 3. vyd. McGraw-Hill Osborne Media, 2005. ISBN 0072231890.

[22] Seshadri, S. *Angular: Up and Running: Learning Angular, Step by Step*. 1. vyd. O'Reilly Media, 2018. ISBN 1491999837.

[23] Shaw, B., Badhwar, S., Bird, A., Chandra, B. a Guest, C. *Web Development with Django*. 1. vyd. Packt Publishing, 2021. ISBN 978-1839212505.

[24] Shklar, L. a Rosen, R. *Web Application Architecture: Principles, Protocols and Practices*. 1. vyd. Wiley, 2009. ISBN 9780470518601.

[25] Tekeyev, Z. *Portable Chess Game Notation (PGN): Complete Guide* [online]. 2022 [cit. 2023-05-04]. Dostupné z: https://thechessworld.com/articles/general-information/portable-chess-game-notation-pgn-complete-guide/.

[26] *Třívrstvá architektura (Three-tier architecture)* [online]. 2015 [cit. 2023-01-04]. Dostupné z: https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture.

[27] *What is TypeScript* [online]. [cit. 2023-04-02]. Dostupné z: https://www.typescripttutorial.net/typescript-tutorial/what-is-typescript/.

[28] *Introduction - API Styles* [online]. [cit. 2023-04-04]. Dostupné z: https://vuejs.org/guide/introduction.html#api-styles.

[29] Welling, L. a Thomson, L. *PHP and MySQL Web Development*. 5. vyd. Addison-Wesley Professional, 2016. ISBN 978-0275967598.

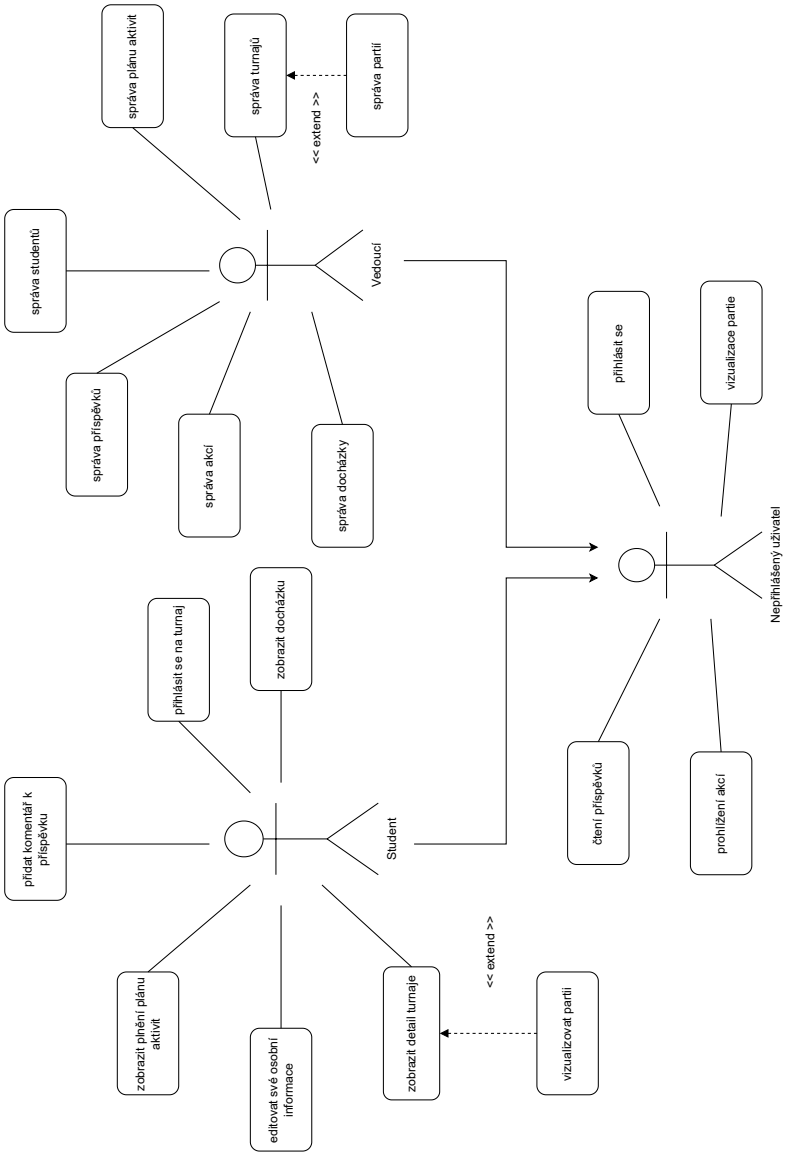
Příloha A

Obsah přiloženého paměťového média

- **xkolar76.pdf** – technická zpráva ve formátu PDF
- **xkolar76.zip** – zdrojové kódy zprávy (LaTeX)
- **xkolar76-src-fe.zip** – zdrojové kódy klientské aplikace
- **xkolar76-src-be.zip** – zdrojové kódy serverové aplikace
- **readme.txt** – obsah paměťového média a návod ke spuštění aplikace

Příloha B

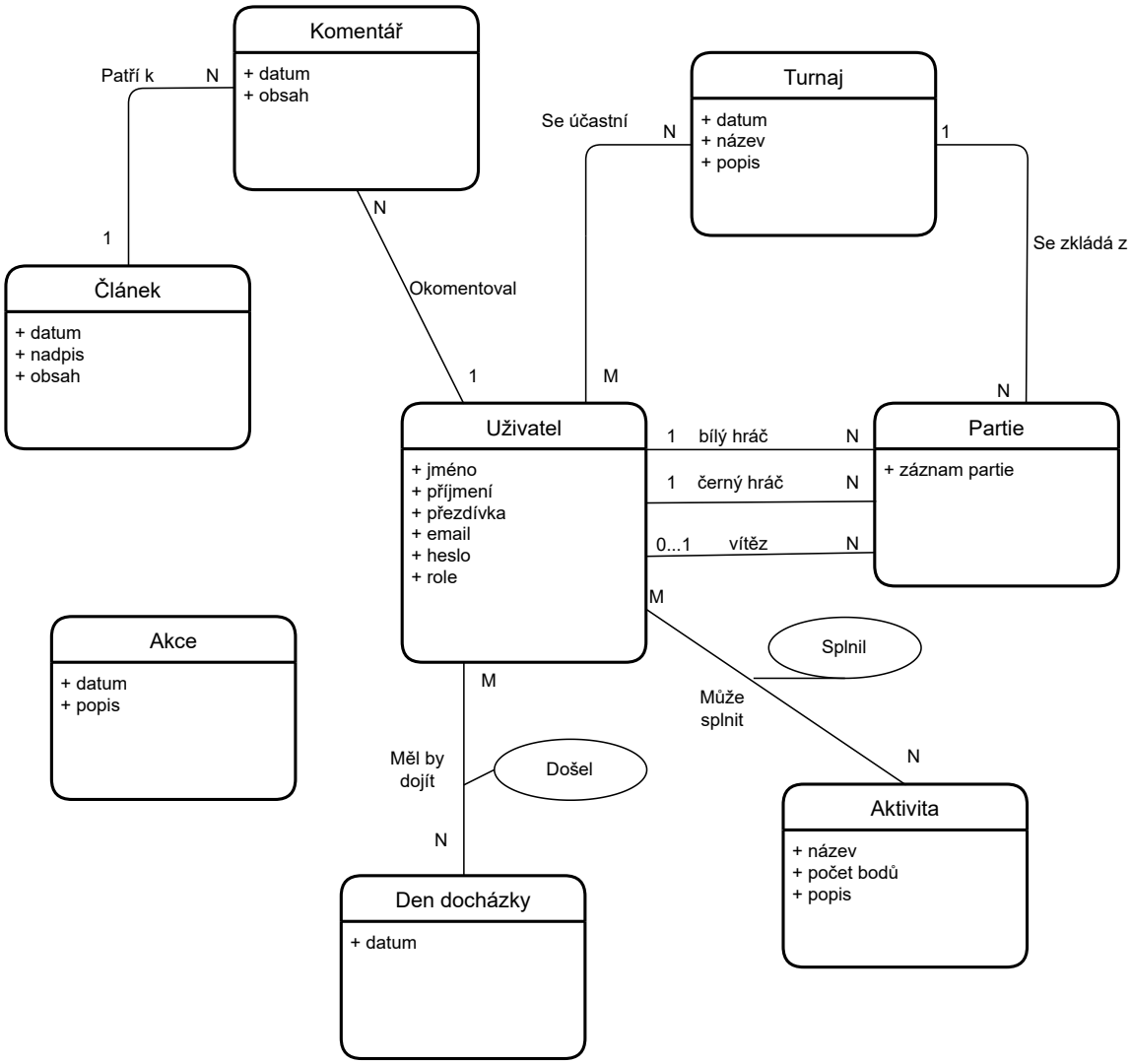
Diagram případů užití



Obrázek B.1: Diagram případů užití

Příloha C

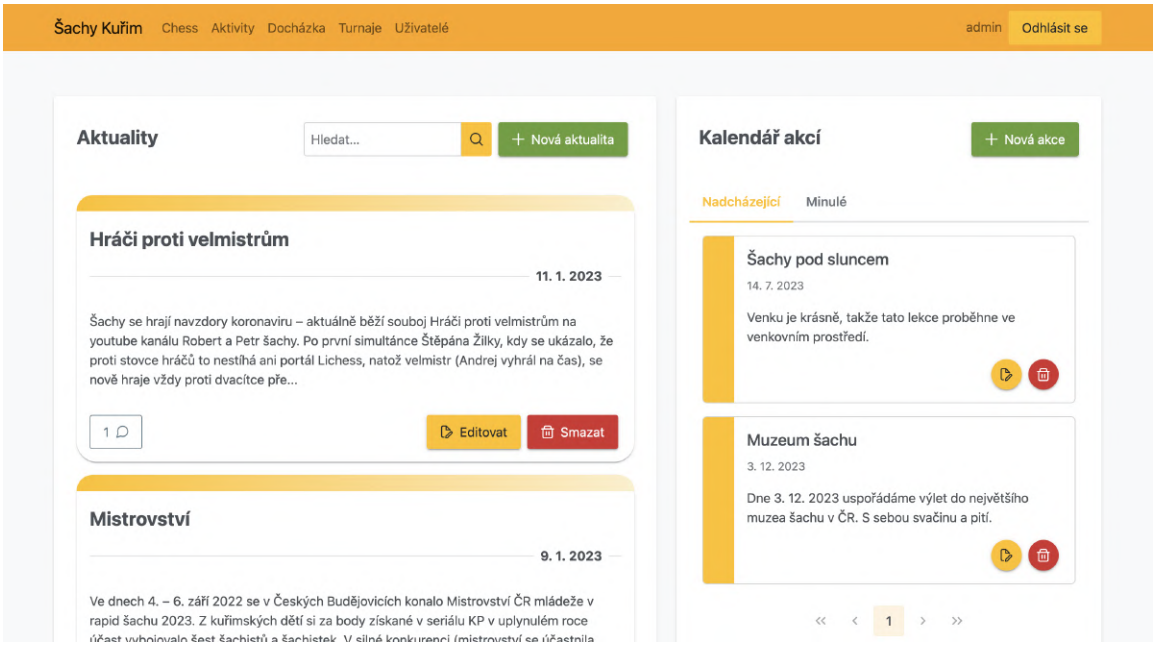
ER diagram



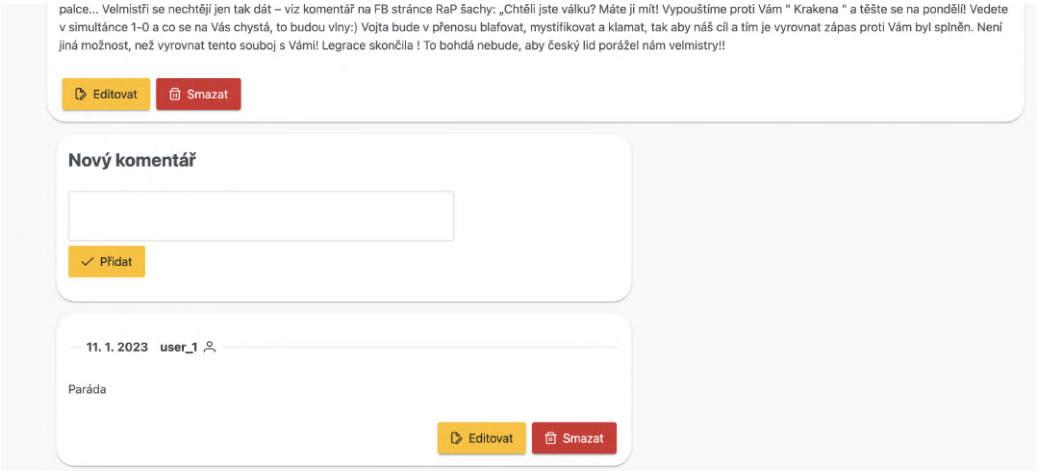
Obrázek C.1: ER diagram

Příloha D

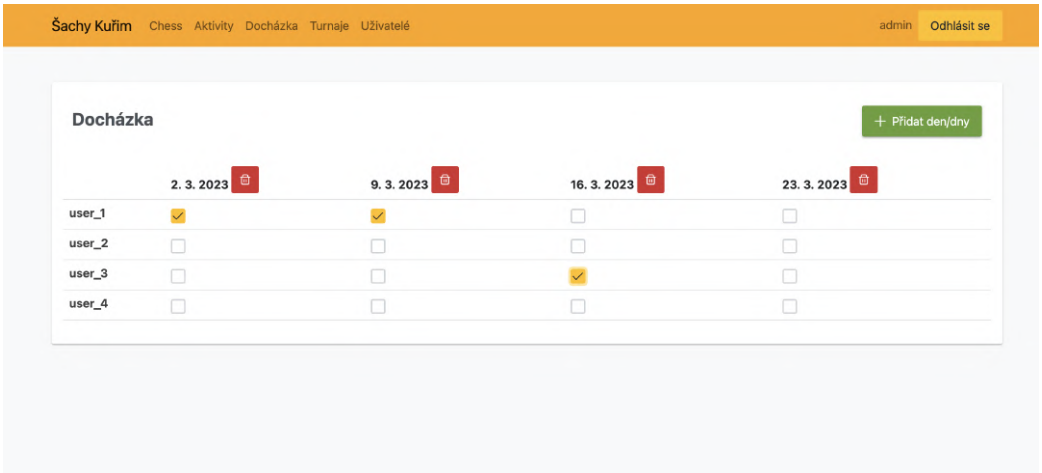
Další použití aplikace



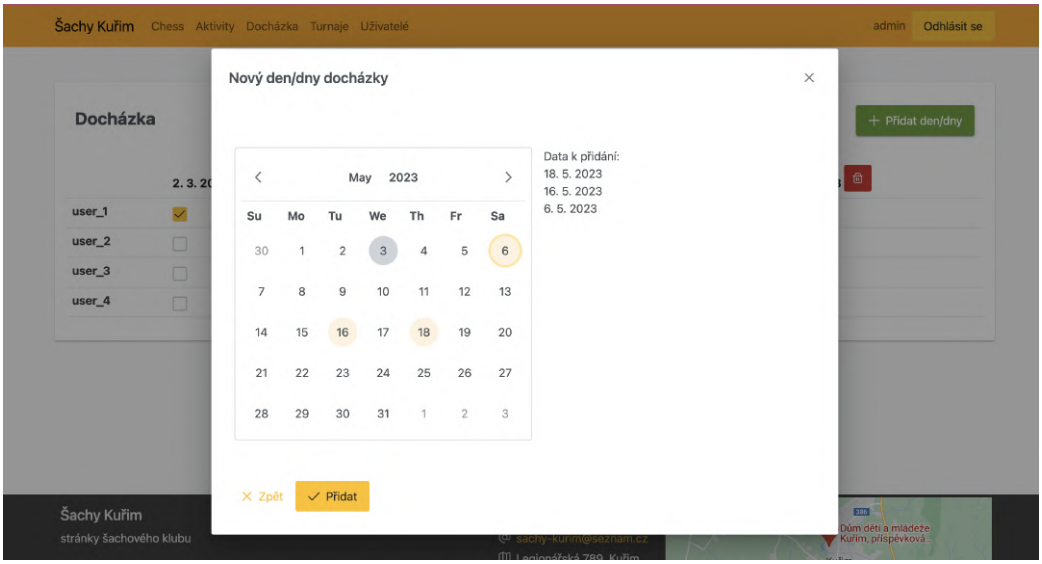
Obrázek D.1: Hlavní stránka aplikace z pohledu vedoucího



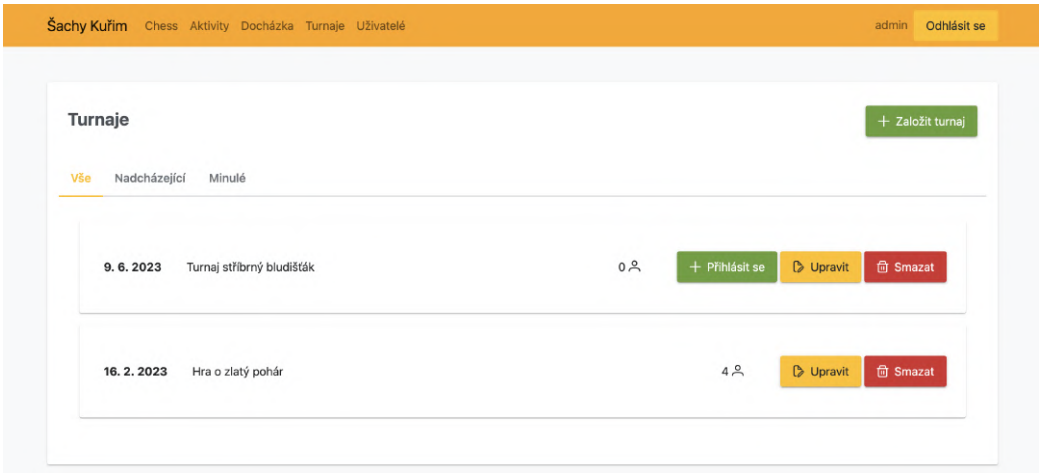
Obrázek D.2: Komentáře k příspěvku



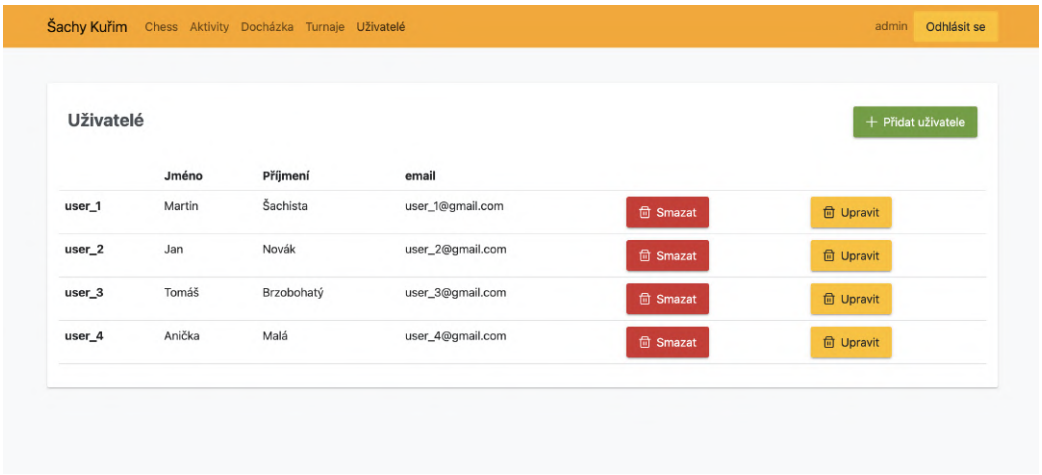
Obrázek D.3: Značení docházky



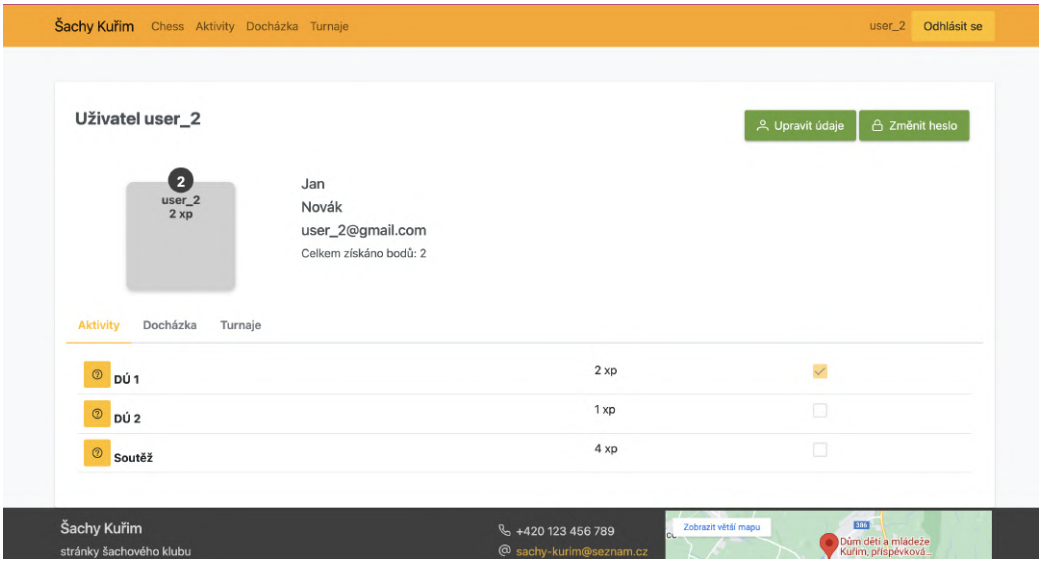
Obrázek D.4: Definice dní docházky výběrem z kalendáře



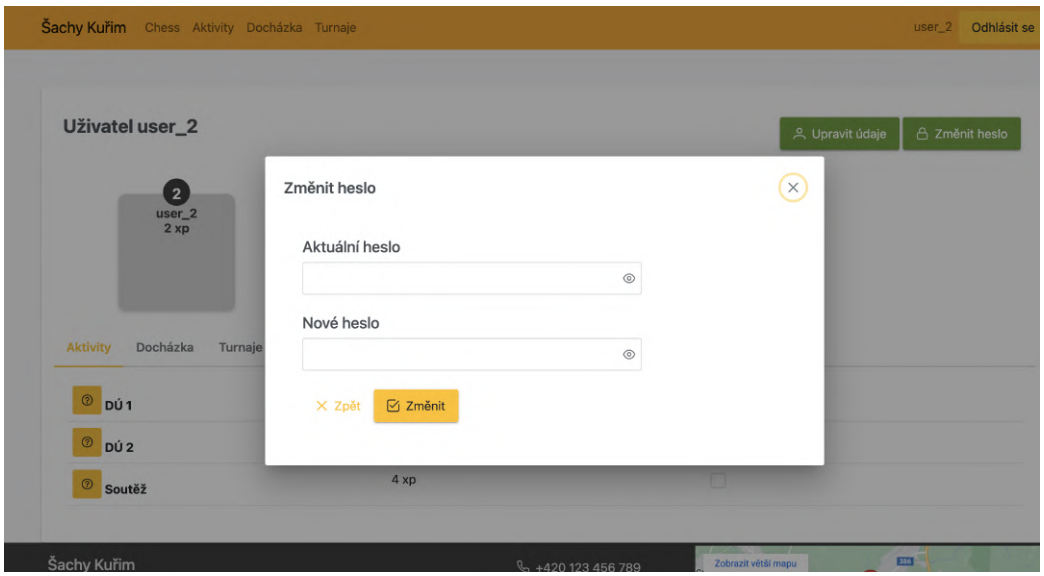
Obrázek D.5: Seznam turnajů



Obrázek D.6: Správa uživatelů



Obrázek D.7: Profil uživatele, odráží se zde barva ze stupně vítězů



Obrázek D.8: Změna hesla, většina akcí na úprovu a přidání dat jsou v modálních oknech