FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Aplikace pro sjednocení informací o nadcházejících turnajích v plážovém volejbale
Student:	Bc. Richard Vacenovský
Vedoucí:	Ing. Jan Bliznìčenko
Studijní program:	Informatika
Obor / specializace:	Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Cílem práce je návrh, implementace a otestování webové aplikace sloužící pro vyhledávání nadcházejících turnajů v plážovém volejbale. Aplikace bude obsahovat následující funkce:
· automatizované získávání a ukládání informací o turnajích z různých zdrojů
· možnost přidání turnaje manuálně
· uživatelsky přívětivé zobrazení všech potřebných informací o turnajích
· optimalizované zobrazení pro mobilní zařízení

Úkoly:
1. Vyberte vhodné zdroje informací o turnajích.
2. Nastudujte a analyzujte problematiku web scrapingu.
3. Vyberte a implementujte vhodné způsoby získávání informací z vybraných zdrojů (web scraping, využití API apod.).
4. Ukládejte data v jednotném formátu.
5. Navrhněte rozhraní pro komunikaci mezi webovou aplikací a backendem.
6. Navrhněte a implementujte webovou aplikaci.
7. Aplikaci otestujte a zdokumentujte.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 23. května 2023 v Praze.

Diplomová práce

APLIKACE PRO SJEDNOCENÍ INFORMACÍ O NADCHÁZEJÍCÍCH TURNAJÍCH V PLÁŽOVÉM VOLEJBALE

Bc. Richard Vacenovský

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Jan Blizničenko
10. ledna 2024

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Rád bych poděkoval hlavně svému vedoucímu Ing. Janu Blizničenkovi za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování diplomové práce. Dále bych rád poděkoval rodině za podporu a trpělivost.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 10. ledna 2024

Abstrakt

Tato diplomová práce se zabývá vyhledáváním nadcházejících turnajů v plážovém volejbale a implementací webové aplikace pro jejich zobrazení. Aplikace kromě zobrazování dat nabízí i další funkcionality, mezi které patří manuální vytváření nových turnajů nebo jejich export v podobě kalendáře. V analytické části práce jsou rozebrány a porovnány technologie jak pro získávání dat z webového zdroje, tak pro samotný vývoj webové aplikace. Následuje návrh architektury včetně specifikací požadavků a souvisejících případů užití. Na základě specifikací vznikla samotná implementace, jejíž součástí je i nasazení aplikace. Na závěr je uskutečněno uživatelské a manuální testování, jehož výstupy se odráží na konečných úpravách. Web je již v provozu a aktivně využíván hráči plážového volejbalu.

Klíčová slova volejbal, plážový volejbal, web, webová aplikace, web scraping, turnaj, React, Flask, PostreSQL, nasazení

Abstract

This thesis deals with the search for upcoming beach volleyball tournaments and display them in web application. In addition to displaying data, the application also offers other functionalities, including the manual creation of new tournaments or their export in the form of a calendar. In the analytical part of the work, technologies for obtaining data from a web source and for the development of a web application are analyzed and compared. Next is architecture design including requirements specifications and related use cases. Based on the specifications, the implementation itself was introduced, which also includes its deployment. Finally, user and manual testing is performed. The results of the testing are reflected in the final modifications of the application. The website is already up and running and being used by beach volleyball players.

Keywords volleyball, beach volleyball, web, web application, web scraping, tournament, React, Flask, PostreSQL, deployment

Seznam zkratek

HTTP	Hyper-text Transfer Protocol
API	Application Programming Interface
URL	Uniform Resource Locator
XML	Extensible Markup Language
XPath	XML Path Language
DOM	Document Object Model

Úvod

Plážový volejbal vznikl ve státě Kalifornie ve dvacátých letech minulého století. Od té doby je popularita tohoto sportu stále na vzestupu a díky tomu roste i počet turnajů. Hráči, kteří se rozhodnou poměřit síly s konkurencí, to ale nemají vůbec jednoduché. Pořadatelé své turnaje totiž zveřejňují na různých místech, mezi která patří jejich vlastní webové stránky, stránky sportovních areálů nebo sociální sítě. Taková diverzita může být pro hráče matoucí a demotivující. Jako řešení se nabízí mít všechny informace o turnajích k dispozici přehledně a na jednom místě. Hráčům se zobrazí tabulka nebo kalendář se všemi budoucími turnaji, případně si výběr zúží použitím filtrů (kategorie, místo konání atd.). Pořadatelé, kteří nemají vlastní stránky, mohou službu využít ve svůj prospěch a přímo na ní své turnaje vytvářet a spravovat. Výše zmíněné řešení nabízí webová aplikace, která je předmětem této práce. Určena je především pro aktivní hráče plážového volejbalu, ale i pro pořadatele turnajů oblíbeného sportu na písku.

Téma si autor zvolil, protože se plážovému volejbalu věnuje již přes 10 let a aktivně se účastní turnajů různého druhu. Kromě toho podle rešerše žádné takové řešení na území České republiky zatím neexistuje.

Cíle práce

Cílem analytické části diplomové práce je nejprve vybrat vhodné zdroje informací o turnajích. Dále nastudovat a analyzovat problematiku web scrapingu a následně vybrat vhodné způsoby pro získávání informací z vybraných zdrojů pomocí metod, jako jsou právě web scraping, API a další.

Cílem praktické části je navrhnout architekturu a rozhraní pro komunikaci mezi backendem a webovou aplikací. Na základě návrhu vznikne samotná implementace. Aplikace bude obsahovat funkce pro automatizované získávání a ukládání informací z různých zdrojů, manuální přidání turnaje a uživatelsky přívětivé prostředí včetně optimalizovaného zobrazení pro mobilní zařízení. Na závěr bude aplikace otestována a zdokumentována.

Kapitola 1
Analýza

V této kapitole proběhne seznámení s teoretickou částí diplomové práce a použitými technologiemi. Blíže budou rozebrány jednotlivé metody získávání dat a populární nástroje pro vývoj frontendu, backendu a databází. Na závěr dojde ke zhodnocení rešerše existujících řešení.

1.1 Analýza metod pro získávání informací o turnajích

Problematika, které je nutné hned na začátku věnovat pozornost, je získání dat a s tím spojený výběr vhodných zdrojů. Zdroj je v tomto kontextu možné chápat jako informační kanál nebo jako metodu pro získání informací. Proto budou rozebrány oba případy.

Pro výběr správného zdroje ve smyslu informačního kanálu je nutné brát v potaz několik základních informací, které jsou pro uživatele stěžejní. Kdy a kde se turnaj koná, pro koho je určený a jak se přihlásit. Pokud zdroj obsahuje odpověď na všechny zmíněné otázky, může být označen za vhodný.

Volba vhodných zdrojů jako metod pro zisk informací je o něco komplikovanější. K dispozici jsou různé metody, lišící se svou dostupností, spolehlivostí i složitostí extrakce dat. Pro účely této diplomové práce se nabízejí tři, splňující vždy jen některé z výše zmíněných vlastností. Těmi jsou API, web scraping a ruční vkládání dat.

1.1.1 Stručné seznámení s pojmy

Web scraping Proces extrakce dat z webových stránek, zpravidla automatizovaný.[1]

API (Application programming interface) Rozhraní pro programátory, které umožňuje aplikacím vzájemně spolupracovat a komunikovat, sdílet data a provádět akce.[2]

Bot V kontextu web scrapingu jde o automatizovaný skript nebo program, který automaticky prochází webové stránky a sbírá nebo zpracovává data bez přímého zásahu člověka.[3]

1.1.2 API vs web scraping

Obě metody jsou standardem pro získávání dat z webových zdrojů a výběr jedné z nich bude vždy záviset na konkrétní situaci. Proto se tato sekce věnuje jejich porovnání z různých úhlů pohledu a vychází z článku[4].

1.1.2.1 Přístup

Web Scraping Umožňuje získat data z jakékoli webové stránky. Některé stránky však mohou mít ochranu proti botům nebo scrapingu.

API Nemusí být vždy veřejné nebo může mít omezený přístup. API mohou mít různá omezení, např. rate limits.

1.1.2.2 Struktura Dat

Web Scraping Získává neupravená, nestrukturovaná data z HTML. Vyžaduje zpracování a analýzu pro konverzi na strukturovaná data.

API Vrací strukturovaná data ve formátu, jako jsou JSON nebo XML.

1.1.2.3 Rychlost

Web Scraping Je časově náročný, zejména při zpracování více stránek.

API Je obvykle rychlejší, protože vrací agregovaná data.

1.1.2.4 Stabilita

Web Scraping Je náchylný k chybám a změnám na stránkách.

API Je obecně stabilnější, ale může být ovlivněno vysokou zátěží.

1.1.2.5 Spolehlivost

Web Scraping Může být detekován a blokován.

API Je spolehlivější, protože je vytvořeno vývojáři stránky.

1.1.2.6 Technické znalosti

Web Scraping Vyžaduje znalosti HTML struktury, parsování a anti-bot opatření.

API Vyžaduje porozumění technické dokumentace a práci s požadavky a odpověďmi.

1.1.2.7 Legální Aspekty

Web Scraping Musí dodržovat podmínky stránek a zákony o ochraně dat.

API Je řízeno podmínkami poskytovatele.

1.1.3 Vyhodnocení

API bude vždy první volbou pro účely této práce, a to hlavně díky spolehlivosti, stabilitě a rychlosti. Podmínkou je dobrá dokumentace, dostupnost a obsah požadovaných dat.

Pokud API nenabízí potřebná data nebo ho stránka vůbec neposkytuje, přichází na řadu web scraping. Jeho použití je podmíněno absencí anti-bot opatření.

Poslední možností je ruční vytvoření turnaje přímo v aplikaci. Ta nastává ve chvíli, kdy je data komplikované nebo dokonce nemožné získat jednou z předchozích metod. Typicky se jedná o situace, kdy jsou turnaje prezentovány ve formě událostí na sociálních sítích nebo posterů.

1.2 Web scraping

Tato sekce blíže rozebírá metodu web scrapingu a témata s ní spojená, jako jsou web crawling, užívané technologie nebo etické aspekty.

1.2.1 Web crawler

Hned na úvod je třeba si přiblížit pojem web crawler, jinak označovaný i jako web spider, robot nebo právě web scraper. Jeho úkolem je systematicky sbírat informace z webu za účelem idexace obsahu pro vyhledávače. Stránky jsou kopírovány a zpracovány prohlížečem, který je následně indexuje pro snazší vyhledávání na webu. Uživatel má tak k dispozici rychlejší zobrazení výsledků.

Druhou funkcí, kterou označujeme jako **web scraping**, je automatické načítání obsahu z libovolné webové stránky. To se objevilo ve chvíli, kdy společnosti začaly používat web scraping pro své vlastní účely. Příkladem může být sledování dynamicky se měnících cen produktů u konkurence v oblasti e-commerce. Dále už bude věnována pozornost pouze druhé ze zmíněných funkcí, web scrapingu.[3]

1.2.2 Web scraping

Web scraping, nazývaný také jako webová extrakce nebo harvesting, je technika extrakce dat z World Wide Webu (WWW) a uložení do souborového systému či databáze pro pozdější využití (např. analýzu). Běžně využívá HTTP protokol nebo webový prohlížeč. Proces scrapingu může probíhat jak manuálně, tak automatizovaně pomocí robota nebo web crawlera.[1]

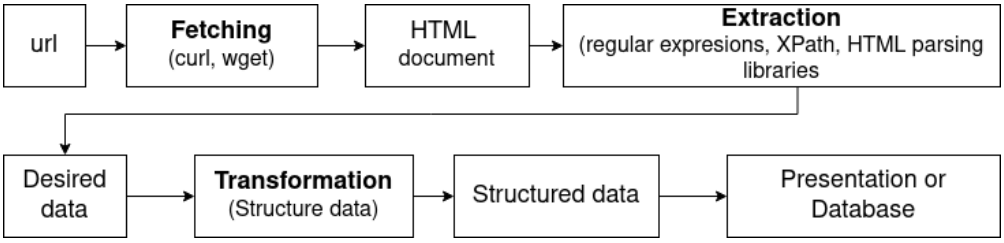
Jak píše Emil Person ve své práci, rozdělujeme proces web scrapingu1.1 na 3 fáze.

Fáze získání dat Získání dat neboli fetching má za úkol přístup k cílové webové stránce pomocí HTTP protokolu, který umožňuje odesílat a přijímat požadavky od webových serverů. Využívají se knihovny, jako jsou curl a wget, pro odeslání HTTP GET požadavku a získání odpovědi ve formě HTML (Hypertext Markup Language) stránky.[5][6]

Fáze extrakce Po získání HTML stránky následuje fáze extrakce, během které jsou pomocí regulárních výrazů, knihoven pro parsování HTML a dotazů XPath identifikovány a získány důležité informace.[5][6]

Fáze transformace Poslední fází je transformace extrahovaných dat do strukturovaného formátu. Data pak slouží k prezentaci nebo ukládání do databáze. Uložená data se mohou dále zpracovávat a mohou pomáhat například při rozhodování v oblasti business intelligence.[5][6]

V dnešní době, kdy lze internet považovat za největší „datové centrum“ na světě, představuje web scraping mocný nástroj, který disponuje opravdu širokou škálou využití. Sledování cen produktů, analýza sentimentu, agregace obsahu nebo akademický výzkum. Svou roli může mít i v oblasti umělé inteligence, při analýze a poskytování výsledků. Jedním z konkrétnějších příkladů může být shromažďování aktuálních zpráv z různých zpravodajských portálů a následné



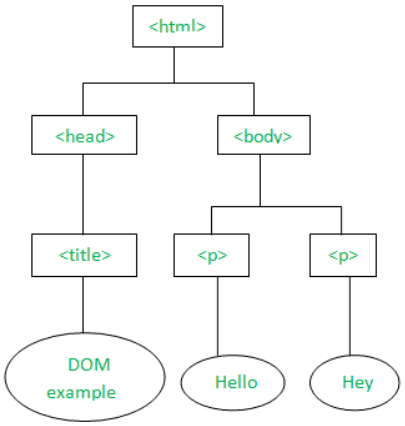
Obrázek 1.1 Proces web scrapingu[5]

zobrazení všech nejnovějších informací na jednotné platformě.[7] Na stejném principu je založena i tato diplomová práce.

1.2.3 Document Object Model

Pro lepší porozumění problematice web scrapingu a jednotlivých technik je potřeba chápat, co je to DOM neboli Document Object Model. Podle konsorcia W3C se jedná o API (1.3) pro dokumenty HTML a XML (Extensible Markup Language). Definuje logickou strukturu dokumentů a způsob, jakým se k dokumentu přistupuje a jak se s ním manipuluje. Ve specifikaci DOM se termín „dokument“ používá v širokém smyslu. XML stále častěji slouží jako způsob reprezentace mnoha různých druhů informací, jejichž velká část by se tradičně považovala spíše za data než za dokumenty. XML však tato data prezentuje jako dokumenty a k jejich správě lze použít právě DOM. Pomocí DOM mohou programátoři vytvářet a sestavovat dokumenty, procházet jejich strukturu a přidávat, upravovat nebo odstraňovat prvky a obsah. Příklad DOM je k dispozici v obrázku1.2[8]

```
<html>
<head>
<title>
 DOM example
</title>
</head>
<body>
<p id = "para1">Hello</p>
<p id =  “para2”>Hey</p>
</body>
</html>
```



Obrázek 1.2 Příklad DOM[9]

1.2.4 Techniky scrapingu

Pro scraping se dají využít různé metody. Každá má své výhody i nevýhody a je užitečná ve specifických případech.

1.2.4.1 Manuální scraping

První a zdaleka nejjednodušší technikou je manuální scraping, pro širokou veřejnost známý také jako copy-pasting. Nevyžaduje žádné znalosti v oblasti programování a každý, kdo někdy pracoval s počítačem, se s touto metodou téměř určitě setkal. Jedná se o ruční kopírování obsahu z webu a následné vkládání do vlastního úložiště, např. databáze. Kromě jednoduchosti může být výhodou i překonání ochrany proti robotům. Jednoduchost si však vybírá svou daň ve formě vysoké časové náročnosti. Od toho se přímo odvíjí velmi nízká efektivita.

Manuální scraping tedy není vhodný pro zpracování většího množství dat a vyžaduje repetetivní práci programátora.[10]

1.2.4.2 Analýza DOM

Analýza Data Object Modelu neboli DOM rozebírá webovou stránku na základě její HTML struktury a umožňuje získávat konkrétní elementy, jako jsou nadpisy, paragrafy nebo obrázky. Je tedy vhodná pro dobře strukturované weby a disponuje velkou flexibilitou. Nevýhodou je nutnost dobrého porozumění HTML struktuře daného webu, rychlost a skutečnost, že některé stránky mohou dokonce tuto techniku blokovat.[7]

1.2.4.3 Regulární výrazy

Regulární výrazy označované také jako regex jsou využívány hlavně pro svou účinnost a úsporu času. Hodí se pro případy, kdy uživatel na stránce hledá data ve specifickém formátu, jako je email nebo telefonní číslo. Pokud jsou ale data nestrukturovaná, není použití regulárních výrazů ideální volbou. Nevýhodou může být také složitá syntaxe.[7]

1.2.4.4 XPath

Poslední rozebranou technikou je XML Path, zkráceně XPath. XML připomíná v mnoha ohledech HTML, ale disponuje odlišnými charakteristikami. Pro lepší pochopení je možné XPath přirovnat k SQL dotazu, který je zaměřený na manipulaci s XML a HTML dokumenty. Zároveň představuje vynikající nástroj pro RPA, neboli robotic process automation. Praktickým příkladem může být situace, kdy je potřeba vyplnit textové pole libovolnými údaji. Velkou výhodou je také fakt, že jsou výrazy XPath podporovány celou řadou programovacích jazyků.[11]

1.2.5 Etické a legální aspekty

Nejprve je důležité zmínit, že web scraping je zcela legální, pokud se nepoužívá k jakýmkoliv škodlivým účelům, přímo nepoškozuje obchod nebo provoz dotyčného webu a zpracovaná data nezahrnují žádné osobní informace. Evropa se v tomto případě řídí hlavně General Data Protection Regulation (GDPR).[12]

Web scraping je tedy legální, pokud splňuje výše zmíněné podmínky. Neméně důležitá je ale i jeho etika. Zde jsou zmíněny některé etické zásady podle Jamese Densmora[13], kterými se snaží řídit i tato diplomová práce:

1. Primárně využít API, pokud je veřejné a poskytuje potřebné informace.
2. Vždy přidat User Agent řetězec, který objasní záměry scrapingu a poskytne kontakt na programátora.
3. Snažit se posílat požadavky s rozumnou frekvencí a vyhnout se tak obvinění z pokusu o DDoS útok.
4. Ukládat pouze nezbytně potřebná data.

5. Respektovat získaný obsah a nevydávat ho za svůj vlastní.
6. Hledat způsob, jak stránce na oplátku pomoci, například ji zviditelnit a zvýšit tak její návštěvnost.
7. Včas odpovídat na případné dotazy a spolupracovat na řešení problémů.
8. Provádění scrapingu za účelem vytvořit z dat novou hodnotu, ne je pouze duplikovat.

Dále by každý uživatel měl respektovat soubor *robots.txt* (příklad zde1.3), který poskytuje informace a pokyny k procházení webové stránky. Obsahuje například odkazy s důvěrnými informacemi nebo data, pro crawlery nepřístupná. Může také definovat intervaly pro zasílání požadavků na danou stránku.[7]

Takový soubor by měla poskytovat každá webová stránka přidáním */robots.txt* za její kořenovou adresu. Tj. na stránce *www.example.com* je soubor na adrese *www.example.com/robots.txt*.

Obrázek 1.3 Příklad robots.txt na google.com[14]

1.2.6 Populární nástroje

V této sekci proběhne krátké seznámení s několika populárními nástroji. Budou zmíněny základní charakteristiky spolu s výhodami a nevýhodami každého z nástrojů, které uvádí Anthony Heath ve svém článku[7]. To usnadní následný výběr vhodné technologie.

1.2.6.1 Beautiful Soup

Prvním nástrojem je Beautiful Soup. Knihovna jazyku Python, vhodná převážně pro statické webové stránky, které posílají najednou celý HTML dokument. Umí vytvořit strom pro analýzu

a následnou extrakci potřebných dat. Nehodí se pro zpracování stránek s dynamickým obsahem.[15]

Výhody a nevýhody

+ Jednoduché rozhraní
+ Dobré vyhledávání a filtrování
+ Mechanismy pro error handling
+ Open-source
+ Cross-platform
– Pomalejší
– Nevhodné pro komplexní požadavky

1.2.6.2 Scrapy

Dalším nástrojem je Scrapy, framework založený opět na oblíbeném Pythonu. Disponuje obsáhlou sadou funkcí pro crawling, extrakci a zpracování dat. Na rozdíl od Beautiful Soup umí pracovat i s dynamickými weby. Scrapy je využívaný hlavně pro jeho výkon, komplexitu a možnost zpracovat výsledky v několika formátech, včetně CSV, JSON, XML a dalších.[7]

Výhody a nevýhody

+ Účinné a rychlé scrapování
+ Flexibilní extrakce dat
+ Zabudovaná podpora pro webové protokoly
+ Přizpůsobitelnost a rozšiřitelnost
+ Asynchronní zpracování dat
– Nevhodné pro začátečníky
– Limitovaná podpora pro webové stránky založené na JavaScriptu
– Nevhodné pro jednoduché úkoly

1.2.6.3 Selenium

Selenium vniklo už v roce 2004, kdy se využívalo hlavně na testování webových stránek a aplikací napříč prohlížeči. Dnes nabízí kompatibilitu s mnoha programovacími jazyky (Python, Java, C# a další) a poskytuje robustní API pro interakci s weby včetně navigace, klikání nebo textového vstupu. Obsahuje také vestavěné metody pro přístup ke konkrétním prvkům z webové stránky pomocí ID prvků a tříd.[7][16]

Výhody a nevýhody

+ Open-source
+ Kompatibilita napříč vyhledávači
+ Kompatibilita s mnoha programovacími jazyky
+ Automatizace
– Omezená podpora pro jiné než webové aplikace
– Pomalejší

1.2.6.4 Octoparse

Octoparse na rozdíl od předchozích metod nevyžaduje žádnou zkušenost s programováním. Poskytuje jednoduché vizuální rozhraní typu point-and-click a umí exportovat výsledek do různých formátů, jako je Excel, CSV, JSON a další. Poskytuje také cloudové řešení scrapingu, které uživatelům umožňuje spouštět jej na svém serveru. Na rozdíl od předchozích nástrojů je ale placený a omezený, co se týče přizpůsobitelnosti.[17]

Výhody a nevýhody

+ Uživatelsky přívětivé rozhraní
+ Bez programování
+ Pokročilé možnosti scrapování
+ Cloudové řešení
– Přizpůsobitelnost
– Limitovaná automatizace
– Zpoplatnění
– Limitovaná podpora pro webové stránky založené na JavaScriptu

1.3 Application programming interface (API)

Application programming interface neboli API je rozhraní používané vývojáři softwaru k interakci se softwarovými komponentami nebo prostředky mimo jejich kód. Jednodušeji lze tento koncept definovat jako část softwarové komponenty, která je přístupná ostatním komponentám. API má své využití od nástrojů příkazové řádky až po cloud-native architektury a důležitou roli hraje i v oblasti vývoje softwaru.[2]

Pro jeho užívání je důležité znát specifikaci a softwarové rozhraní. Specifikace popisuje proces výměny informací mezi programy a softwarové rozhraní ji dodržuje. Nejpoužívanější architektury pro specifikaci komunikačního protokolu jsou REST (Respresentational State Transfer) a SOAP (Simple Object Access Protocol).[18]

1.3.1 REST

Representational State Transfer neboli REST není protokol ani standard, ale sada architektonických omezení s několika různými způsoby implementace.

Když klient vytvoří požadavek přes RESTful API, přenese se reprezentace stavu zdroje přes HTTP. Tato reprezentace může být v různých formátech, jako JSON, HTML nebo prostý text. Nejčastěji se jedná o JSON, hlavně pro jeho jazykovou agnostiku a čitelnost pro programátory i stroje.

Důležité jsou také hlavičky a parametry v HTTP metodách při posílání RESTful API požadavků. Obsahují totiž důležité informace o metadatech, autorizaci, URI a další.

API může být označena jako RESTful, pokud splňuje několik klíčových kritérií. Patří sem klient-server architektura s řízením požadavků přes HTTP a bezstavová komunikace. To znamená, že mezi GET požadavky není ukládána žádná informace o klientovi a každý požadavek je samostatný a nezávislý.

Další kritérium zahrnuje možnost ukládání dat do mezipaměti (cache), což usnadňuje interakci mezi klientem a serverem. Důležitým aspektem je také uniformní rozhraní, které zajišťuje přenos informací ve standardní formě.

REST je obecně používanější než SOAP, protože není vázán na konkrétní protokol a umožňuje lehčí implementaci a lepší škálovatelnost. Ideální je např. pro IoT a vývoj mobilních aplikací.[19]

1.4 Analýza technologií pro vývoj backendu a frontendu

V této sekci proběhne porovnání jednotlivých technologií pro vývoj backendu i frontendu a seznámení s pojmem framework.

1.4.1 Framework

Nejprve je nutné se seznámit s pojmem framework, který bude v další části této sekce často skloňován. Jedná se o rozsáhlou sadu vývojářských nástrojů, které umožňují programátorům nepsat celý kód úplně od nuly. Pomáhá v rychlém procesu vývoje a funguje jako šablona, kterou lze použít a upravit tak, aby splňovala požadavky projektu.[20][21]

1.4.2 Backend frameworky

Backendové frameworky poskytují vývojářům pevný základ a strukturované prostředí pro rychlejší vývoj serverových komponent softwaru. Dodržováním konvencí a osvědčených postupů se umožňují zaměřit na jádro aplikace, což zrychluje vývoj, zlepšuje údržbu kódu a škálovatelnost. Nejlepší backendové frameworky jsou postaveny na populárních jazycích, jako Python, JavaScript, Ruby atd.[22]

1.4.2.1 Django

Django je open-source framework vytvořený pro vývoj komplexního kódu a aplikací. Široké využití má kromě webových aplikací také v oblasti API a patří mezi nejpopulárnější frameworky v jazyce Python. Svou popularitu si vysloužil hlavně svým bohatým výběrem knihoven a možností znovupoužití komponent.

Mezi další výhody patří schopnost definovat URL vzory, integrovaný autentizační systém a podpora několika mechanismů pro ukládání do cache.

Mezi omezení patří nevhodnost pro menší projekty, protože se jedná o high-level framework, nebo možné zpomalení webových stránek způsobené zpracováním velkého množství požadavků. Pro efektivní využití je navíc nutná jeho pokročilá znalost.[23]

1.4.2.2 CherryPy

CherryPy je lightweight framework disponující rychlostí a stabilitou webového vývoje. Stejně jako Django je open-source a založený na Pythonu. Poskytuje také širokou škálu funkcí včetně sessions, nahrávání souborů, cookies nebo statického obsahu.

Mezi výhody patří spolehlivost, jednoduchá obsluha více HTTP serverů současně a podpora Python 2.7+, Python 3.5+, PyPy, Android i protokolu HTTP/1.1. Navíc disponuje robustním konfiguračním systémem pro vývojáře a administrátory.

Hlavním nedostatkem je absence podrobné dokumentace. Tím se framework stává nedostupným pro kohokoliv, kdo s ním začíná pracovat.[23]

1.4.2.3 Flask

Flask je framework vyvinutý pro flexibilní aplikace, který je dostupný pod BSD licencí (Berkeley Software Distribution – licence pro svobodný software). Své využití nachází spíše u menších a jednodušších projektů.

Výhodou může být možnost posílání RESTful požadavků, plná kompatibilita s WSGI 1.0 (Web Server Gateway Interface) nebo široká škála komunitou vyvinutých rozšíření poskytujících nové funkce.[23]

1.4.2.4 Další frameworky

Mezi další frameworky pro vývoj backendu v Pythonu patří například Pyramid, Grok, TurboGears, Web2Py, Tornado, Bottle nebo BlueBream.[23]

1.4.3 Frontend frameworky

Frontend frameworky jsou sbírkou klíčových nástrojů, knihoven a předem navržených komponent. Využívají převážně opakovaně použitelné komponenty (části kódu) pro Javascript, CSS a HTML, které mohou vývojáři znovu uplatnit v různých projektech.[24]

1.4.3.1 React

Prvním populárním frameworkem pro vývoj frontendu je React.js vyvíjený a spravovaný společností Meta Platforms (dříve Facebook). Oblíbený je pro svou komponentovou architekturu a vykreslování (rendering) pomocí virtuálního DOM. Mimo jiné disponuje bohatým ekosystémem se spoustou knihoven třetích stran a nástrojů. Další výhodou je skvělá dokumentace a pravidelná aktualizace.

Nedostatkem může být kompatibilita verzí díky rychle rostoucímu ekosystému nebo obtížnost pro programátory, kteří s Reactem pracují poprvé.[24]

1.4.3.2 Angular

Druhým frameworkem je Angular, kde za vývojem a správou stojí společnost Google. Angular je vhodný pro implementaci dynamických jednostránkových aplikací s rozmanitým výběrem funkcí a poskytuje efektivní aktualizace v reálném čase díky obousměrné datové vazbě. Výhodou je i jeho CLI (Command Line Interface), který vývojářům značně zjednodušuje vytváření a správu projektů.

Stejně jako React není Angular vhodný pro začátečníky hlavně kvůli komplexní syntaxi. U velkých aplikací navíc hrozí problémy s výkonem.[24]

1.4.3.3 Vue.js

Poslední rozebranou technologií je Vue.js. Nenáročný a vysoce výkonný framework s rychlým vykreslováním stránek a skvělým uživatelským zážitkem. Oproti zmíněným konkurentům je vhodný i pro začátečníky díky detailní dokumentaci a jednoduché syntaxi. Mimo jiné také umožňuje integraci do již existujících projektů.

Na rozdíl od Reactu a Angularu má ale Vue.js daleko menší komunitu uživatelů, což může vést k omezeným zdrojům. Dalším nedostatkem je potenciální absence pokročilých funkcí, dostupných ve vyspělejších technologiích, a stejně jako u Reactu zde hrozí problémy s kompatibilitou verzí díky rychle se vyvíjejícímu ekosystému.[24]

1.4.3.4 Další frameworky

Mezi další frameworky pro vývoj frontendu patří například Svelte a Ember.[24]

1.5 Analýza databázových systémů pro webové aplikace

Při výběru databáze se nabízí mnoho možností. Žádná z nich však není jednoznačně ideální volbou pro webovou aplikaci. Vždy záleží na různých faktorech, zvlášť pro potřeby konkrétního webu. V této sekci proběhne porovnání několika relačních databází.

1.5.1 MySQL

MySQL spadá do kategorie relačních databází a je oblíbeným nástrojem nejen pro účely webových aplikací. Poskytuje kompatibilitu s různými operačními systémy a programovacími jazyky, jako je Java, C++ nebo Python. Často se při webovém vývoji používá ve spojení s dalším open-source softwarem, jako je PHP, Apache nebo Linux. K dispozici je jak komerční, tak komunitní edice v závislosti na potřebách uživatele.[25]

1.5.2 PostgreSQL

PostgreSQL je open-source relační databáze s dobrou škálovatelností, spolehlivostí a rozšiřitelností. Poskytuje pokročilou podporu SQL a díky tomu pomáhá vývojářům pracovat s komplexním zpracováním a analýzou dat. Databáze je navíc na dnešních populárních platformách, jako jsou Windows, MacOS nebo Linux, zcela zdarma.[25]

1.5.3 SQLite

SQLite je databáze napsaná v jazyku C, která na rozdíl od tradičních relačních databází využívá jako úložiště tzv. flat file — *„Flat file je kolekce záznamů, ve které mají data jednotný formát a řídí se určitými pravidly.“*[26]. Vhodná je převážně pro mobilní zařízení, embedded systémy nebo malé aplikace, ale bez problému zvládá ukládání velkého množství dat a poskytuje snadnou manipulaci a maximální výkon.[25]

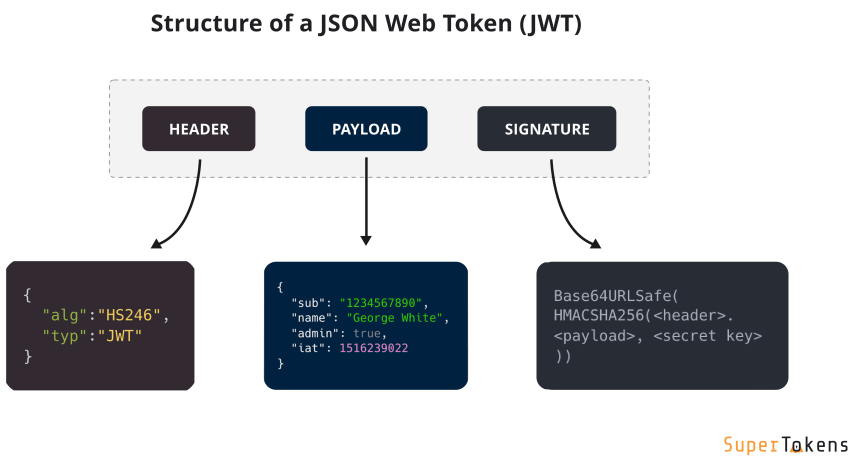
1.6 Autentizace a autorizace

Autentizace je proces ověření totožnosti uživatelů, což bývá prvním krokem v každém bezpečnostním procesu. Proces autentizace může být proveden pomocí hesel, jednorázových PIN kódů, autentizačních aplikací nebo biometrických údajů, jako jsou otisky prstů nebo sken očí. V některých případech je vyžadováno vícefaktorové ověření, což znamená použití více než jednoho způsobu autentizace před udělením přístupu.

Oproti tomu autorizace je proces, při kterém jsou uživateli přidělena přístupová práva k určitému zdroji nebo funkci. Pro dodržení bezpečnosti by autorizace měla vždy následovat po autentizaci. Pokud uživatel prokáže pravost své identity, obdrží oprávnění k požadovaným zdrojům.[27]

1.6.1 JSON Web Token (JWT)

JSON Web Token je standard, který definuje způsob bezpečně předávat informace mezi stranami v podobě JSON objektu. Díky své malé velikosti je možné JWT posílat přes URL jako parametr metody POST nebo jako součást HTTP hlavičky. Díky obsahu všech potřebných informací o dané entitě stačí dotázat databázi pouze jednou a příjemce tokenu navíc ani nepotřebuje validaci tokenu od serveru. Strukturu JWT je možné vidět na obrázku1.4. Důležité je zmínit, že JWT je standard, tj. ne každý token je JWT, ale každý JWT je token.[28]

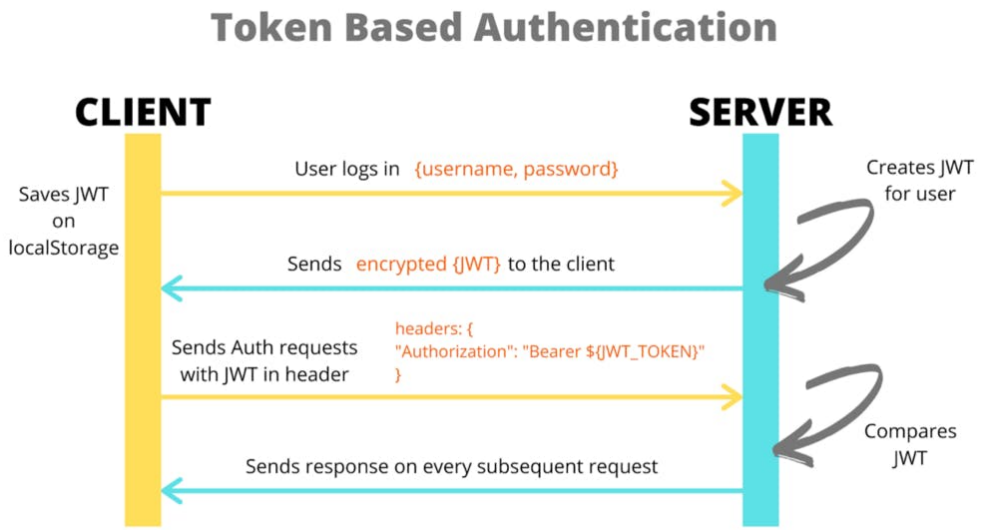


Obrázek 1.4 Struktura JWT[29]

Existuje několik způsobů, jak lze JWT využít. Prvním je autentizace, kdy se uživatel úspěšně přihlásí a je mu vráceno ID tokenu, které je podle specifikace OpenID Connect (OIDC) vždy JWT. Autentizace pomocí tokenu je dostupná na obrázku1.5. Dalším způsobem je autorizace. Jakmile je uživatel úspěšně přihlášen, může aplikace komunikovat např. pomocí již zmíněného API a dotazovat se na konkrétní služby nebo zdroje. Pro úspěšné provedení musí být ke každému požadavku přidán tzv. Access Token, který může být ve formě JWT. Posledním využitím je výměna informací. Díky možnosti token podepsat a jeho struktuře může mít příjemce jistotu, že s daty nebylo nijak manipulováno a odesílatel je opravdu tím, za koho se vydává.[28]

1.7 Analýza existujících řešení

Na téma aplikace provedl autor rešerši, kde hledal nejen řešení v rámci plážového volejbalu, ale jakékoliv aplikace fungující na podobném nebo stejném principu. Hledání ukázalo *díru na trhu* a bylo jednou z hlavních motivací pro vznik této diplomové práce.



Obrázek 1.5 Autentizace pomocí tokenu[30]

1.7.1 Volleybox

Jedním z řešení, které odpovídá této aplikaci hlavně tematicky, je Volleybox1.6[31]. Jedná se o webovou stránku s volejbalovou databází, kterou tvoří komunita více než 46 tisíc uživatelů.

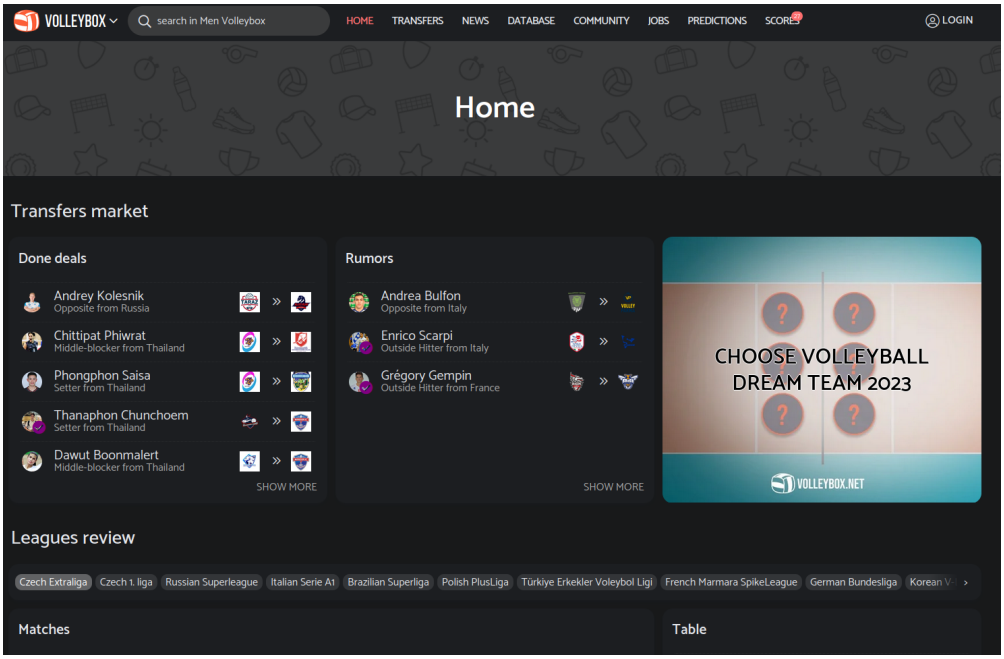
Volleybox se věnuje jak klasickému, tak plážovému volejbalu a nabízí prakticky cokoliv, co by si mohl volejbalový uživatel přát. Stránka je určená nejen pro hráče a organizátory, ale i pro fanoušky, trenéry nebo fyzioterapeuty. Volleybox nabízí možnosti:

- Napsat něco o sobě, přidat fotografii a zmínit například svůj oblíbený tým.
- Potkat nové lidi. Stránka eviduje k roku 2023
  - 15 037 hráčů
  - 3 213 zaměstnanců
  - 819 členů vedení týmu

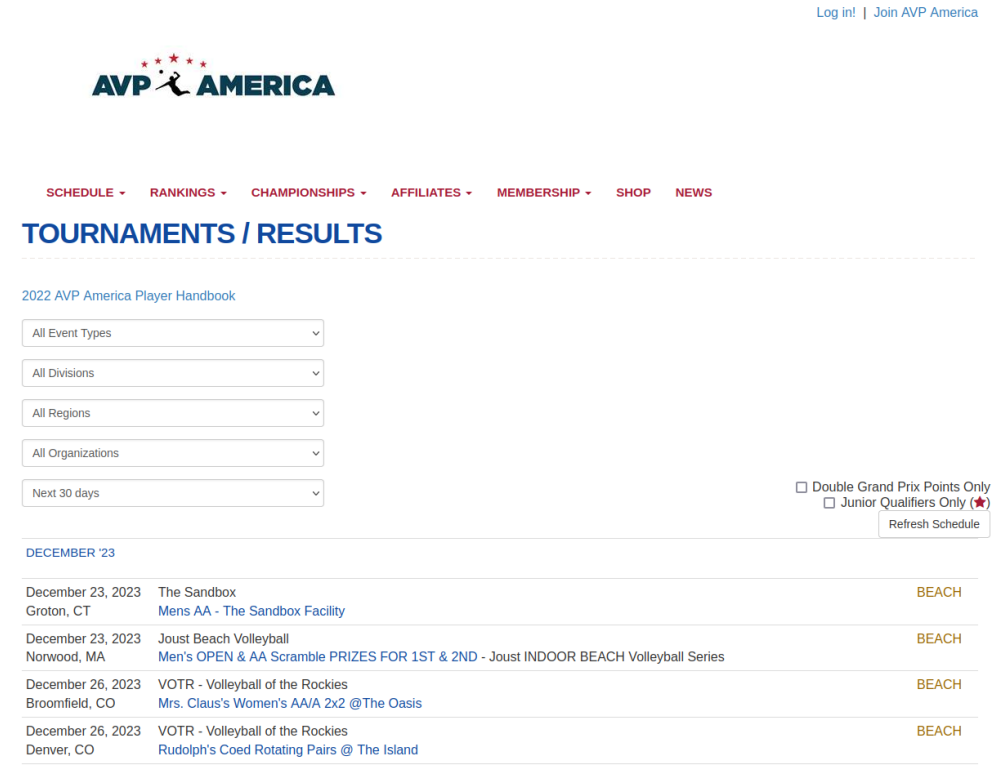
  a umožňuje posílání soukromých zpráv.
- Přidat areál, hráče, hráčský přestup, klub nebo turnaj do databáze.
- Okomentovat novinky nebo videa v rámci veřejného fóra.
- Sledovat výsledky zápasů.
- Hledat práci v oblasti volejbalu. Členové týmového vedení pravidelně přidávají nové pracovní pozice.

1.7.2 AVP America

AVP America je největší venkovní volejbalová organizace ve Spojených státech s cílem podporovat spolupráci mezi promotéry plážového a travnatého volejbalu po celých Spojených státech. Stránka nabízí souhrn turnajů na písku, v hale i na trávě a k tomu poskytuje uživatelům novinky z AVP a vlastní eshop. Jak lze vidět na obrázku1.7, turnaje je možné filtrovat na základě několika kritérií, jako je typ (tráva, písek, hala), region nebo divize.[32]



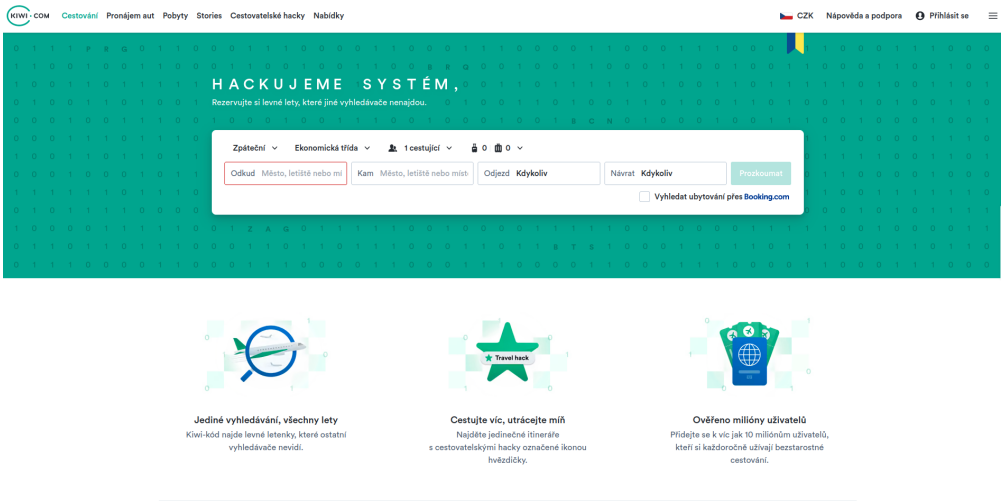
Obrázek 1.6 Volleybox — hlavní strana[31]



Obrázek 1.7 AVP — rozvrh turnajů[32]

1.7.3 Kiwi

Jedním z existujících řešení je i populární portál kiwi.com1.8, dříve známý jako Skypicker. Společnost, která sbírá informace o letenkách z různých zdrojů a poskytuje jejich přehled s propracovaným vyhledáváním a filtry. Přestože se nejedná o plážový volejbal, je princip fungování aplikace prakticky totožný s touto prací.[33]



Obrázek 1.8 Kiwi — hlavní strana[33]

Kapitola 2
Návrh

V této kapitole je rozebrán návrh architektury včetně volby technologií, specifikace požadavků a případů užití, ze kterých vychází samotná implementace.

2.1 Návrh architektury

Architektura aplikace je od začátku koncipována jako model client-server s databází pro ukládání dat, což umožňuje jednoduše oddělit odpovědnost. Server má na starost komunikaci s databází a celou logiku získání a zpracování dat pomocí metod rozebraných v analýze. Zároveň poskytuje REST API pro komunikaci s klientem. Klient je zodpovědný za přehledné zobrazení dat a zpracování informací vložených uživatelem, které následně posílá na server.

2.1.1 Výběr technologií

Důvody pro výběr konkrétních technologií se u každé části aplikace lišily.

Pro účely serveru byl zvolen Flask na základě jeho flexibility a možnosti odesílání RESTful požadavků. Roli hrála i autorova dobrá znalost jazyku Python a předchozí zkušenost z Flaskem z bakalářské práce.

React, jakožto populární knihovna JavaScriptu, byl zvolen pro implementaci klienta hlavně díky své současné popularitě a s tím spojené bohaté nabídce knihoven třetích stran a často aktualizované podrobné dokumentaci.

Jako databáze slouží relační PostgreSQL. U výběru hrála roli jeho dobrá škálovatelnost, spolehlivost a rozšiřitelnost. PostgreSQL je navíc zcela zdarma na platformách, jako jsou Windows, MacOS nebo Linux, a nevyžaduje žádné licenční poplatky nebo omezení.

2.1.2 Server

Server se stará o veškerou logiku získání a zpracování dat. K tomu komunikuje s databází, kam zpracovaná data ukládá. Pomocí REST API potom komunikuje i s klientem, který může posílat požadavky např. na zaslání dat nebo autentizaci uživatele. Jak už zmiňuje výběr technologií výše, server je kompletně implementován ve Flasku s využitím knihoven, jako je například SQLAlchemy pro integraci databáze nebo flask_jwt_extended pro práci s JWT a s tím spojenou autentizací.

2.1.3 Klient

Klient slouží především pro zobrazení informací a interakci s koncovým uživatelem. Hlavní důraz je kladen na přehlednost, intuitivnost, rychlou odezvu a celkově dobrý uživatelský zážitek. Klient neobsahuje žádnou složitější logiku a pro získání dat k zobrazení se dotazuje na server pomocí REST API. Kromě zobrazení informací má na starost i interakci s uživatelem v podobě navigačních tlačítek a (nejen) textových vstupů. S použitím Reactu má programátor možnost spoustu komponent implementovat univerzálně a používat je na více různých místech. To předchází zbytečné duplikaci kódu a pomáhá k lepší přehlednosti celého projektu.

2.1.4 Databáze

Databáze ukládá veškerá data o turnajích, týmech a jednotlivých uživatelích. V prvotním návrhu aplikace se v databázi nacházely pouze dvě tabulky. Jedna s informacemi o turnajích a druhá o uživatelích. Tento návrh se dočkal své realizace a až po jeho implementaci se autor rozhodl do aplikace přidat funkcionalitu pro přihlašování týmů. Tím přibyly do databáze další 2 tabulky pro informace o hráčích a přihlášených týmech. V tu chvíli se ukázala volba názvu *user* pro původní tabulku s uživateli poněkud nešťastná, a to hned z několika důvodů. Tabulka se začala využívat pro ukládání informací pouze o organizátorech, takže je název moc obecný a může být matoucí. Slovo *user* může být navíc v kontextu databázového užití klíčové a tím působit problémy při implementaci. V případě této práce naštěstí ke konfliktu s klíčovým slovem nedochází, a proto se autor rozhodl vzhledem k netriviálnosti změny názvu tabulky tento název ponechat (změna názvu proběhne v navazující práci). Konečný stav databáze tedy obsahuje čtyři tabulky: *user*, *tournament*, *signed_team* a *player*. Ty mají mezi sebou různé vztahy a jsou propojeny pomocí cizích klíčů.

Díky knihovně SQLAlchemy ve Flasku může server jednoduše komunikovat s databází a provádět celou řadu operací, včetně úpravy stávajících dat nebo vytváření nových tabulek. Databáze je tedy propojena pouze se serverem. Klient s databází komunikuje jen prostřednictvím serveru a k databázi nemá přímý přístup.

2.1.5 Datový model

Jak už je zmíněno výše, pro ukládání dat slouží relační databáze PostreSQL. Ta se skládá ze čtyř tabulek. Díky hlavnímu cíli aplikace je nejdůležitější tabulka *tournament*, která uchovává veškerá data o turnajích a obsahuje následující atributy:

- id
- název turnaje
- datum konání
- město
- hostující areál
- kapacita
- počet přihlášených týmů
- cena turnaje
- čas začátku turnaje
- jméno organizátora

- kategorie
- úroveň
- odkaz na zdrojovou stránku (pro více informací a přihlášení)
- id uživatele, který turnaj vytvořil
- čas poslední aktualizace
- možnost přihlášení na turnaj přes aplikaci (ano/ne)

Druhou tabulkou je *user*, která slouží pro uchování informací o organizátorech. *Tournament* a *user* mají nepovinnou vazbu díky cizímu klíči *user_id*. Ta slouží k přidání podrobnějších informací o uživateli, který turnaj vytvořil. Platí to však pouze pro turnaje vytvořené v aplikaci.

Třetí tabulkou, určenou pro ukládání informací o hráčích, je *player*. Ta obsahuje pouze email, heslo a celé jméno hráče.

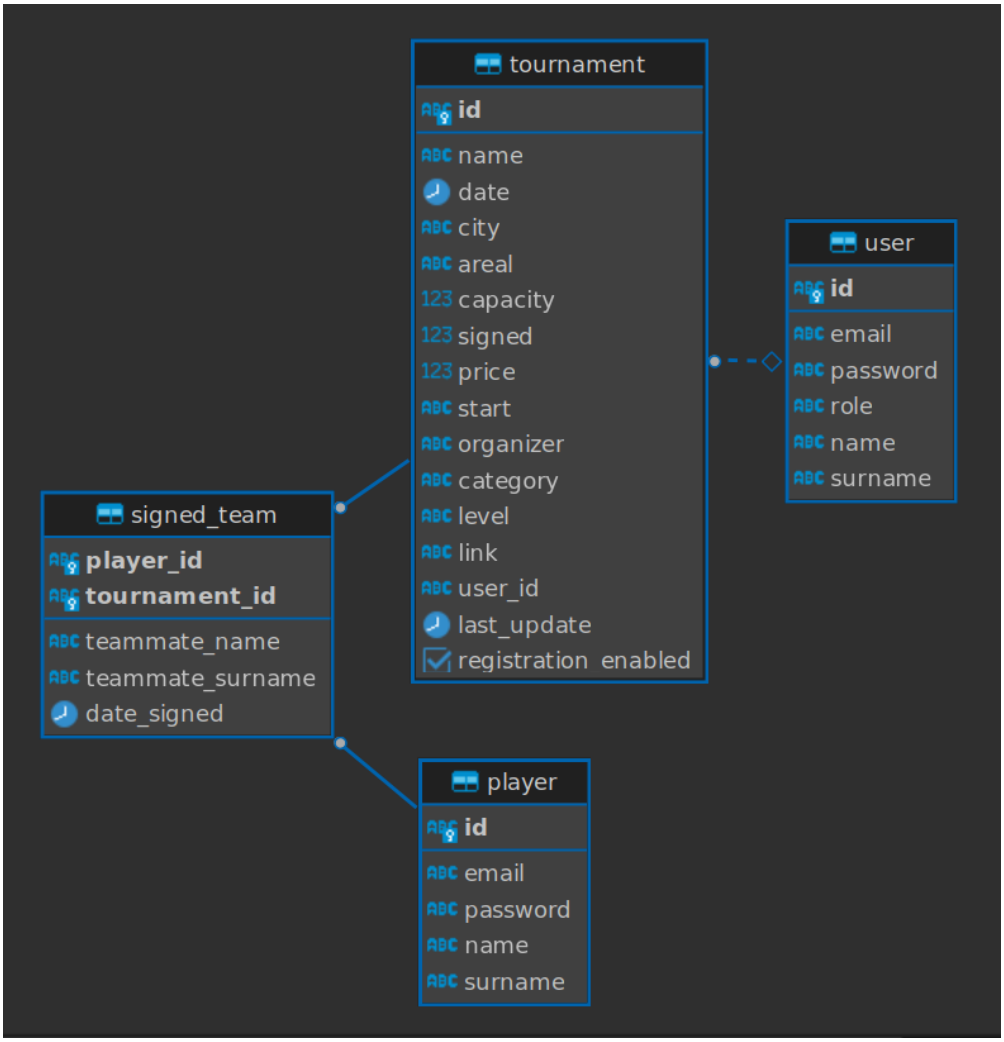
Poslední je pomocná tabulka *signed_team*, která slouží pro evidenci přihlášených týmů a propojuje turnaj s hráčem pomocí cizích klíčů *tournament_id* a *player_id*, jejichž kombinace tvoří unikátní identifikátor tabulky. K propojení přidává další informace, jako je celé jméno spoluhráče a datum přihlášení.

Celé databázové schéma je znázorněno na obrázku2.1.

2.2 Specifikace požadavků

V této sekci proběhne seznámení s požadavky, které budou rozděleny na funkční a nefunkční. Pro zobrazení priority u funkčních požadavků bude složit tzv. MoSCoW metoda[34] s rozdělením na následující skupiny:

- **M: Must have**
  - Požadavky, které jsou nezbytné pro úspěšné dokončení projektu.
- **S: Should have**
  - Požadavky přínosné pro budoucí vydání, ale ne nezbytné.
  - Jejich absence neohrozí funkčnost produktu, ale zvýší jeho hodnotu.
- **C: Could have**
  - Požadavky, které mají značně menší dopad, pokud jsou vynechány.
  - Často deprioritizované na úkor must-have a should-have.
- **W: Will not have**
  - Neprioritní požadavky v daném časovém rámci.
  - Mohou být přesunuty do budoucích projektů nebo zcela ignorovány.



Obrázek 2.1 Schéma databáze

2.2.1 Funkční požadavky

Funkční požadavky slouží pro specifikaci funkčních aspektů aplikace.[35]

1. (M): Automatizované vyhledávání a následné zobrazení dat o nadcházejících turnajích.
2. (M): Vytváření, úprava a mazání turnajů.
3. (S): Zobrazení dat jak v tabulce, tak v kalendáři.
4. (S): Filtrování nalezených turnajů na základě důležitých atributů (město, areál, kategorie a úroveň).
5. (S): Vytvoření celého user managementu (včetně možnosti změnit heslo nebo zaslat nové na email v případě jeho ztráty).
6. (C): Přihlášení přes Google.
7. (C): Možnost přihlášení pro hráče.

8. (C): Možnost přihlášení na turnaj pro přihlášeného hráče.
9. (C): Vygenerování iCal linku pro import událostí do vlastního kalendáře.

2.2.2 Nefunkční požadavky

Nefunkční požadavky definují očekávané vlastnosti softwaru.[35]

1. Optimalizované požadavky na data (žádné dlouhé načítání).
2. Optimalizované zobrazení pro mobilní zařízení.
3. Intuitivní uživatelské rozhraní (jednoduchá orientace v datech a práce s turnaji).
4. Vzestupné řazení dat dle data konání.

2.3 Případy užití (use cases)

Případy užití jsou odvozeny od požadavků definovaných výše. Detailně popisují hlavní funkcionality a aktéry zapojené do procesů. Jsou rozděleny do tří skupin:

- Případy specifické pro práci s turnaji
  - UC-T-01: Vytvoření turnaje
  - UC-T-02: Úprava turnaje
  - UC-T-03: Smazání turnaje
  - UC-T-04: Zobrazení všech turnajů vytvořených daným uživatelem
  - UC-T-05: Zobrazení všech turnajů, na které je uživatel přihlášen
  - UC-T-06: Přihlášení na turnaj
  - UC-T-07: Zobrazení podrobností o turnaji
  - UC-T-08: Filtrování zobrazených turnajů
- Případy specifické pro uživatele
  - UC-U-01: Registrace
  - UC-U-02: Přihlášení
  - UC-U-03: Zapomenuté heslo
  - UC-U-04: Změna hesla
  - UC-U-05: Zobrazení údajů o uživateli
  - UC-U-06: Žádost o přidělení práv organizátora
- Další případy
  - UC-D-01: Vygenerování iCal linku
  - UC-D-02: Automatické mazání starých turnajů

Aktéři

Aktéři pomáhají odlišit různé typy uživatelů a externích systémů. Umožňují definovat případy užití, které se týkají jen některých aktérů nebo které mají odlišné chování na základě konkrétního aktéra.

Organizátor

Uživatel, registrovaný jako organizátor s rolí *basic*.

Organizátor s právy

Uživatel, registrovaný jako organizátor s přidělenou rolí *organizer*.

Hráč

Uživatel, registrovaný jako hráč (role *player*).

Administrátor

Uživatel se všemi právy (role *admin*).

Nepřihlášený uživatel

Kdokoliv, kdo přistoupí na webovou stránku a není přihlášen.

Systém

Server, který provádí periodické operace.

2.3.1 UC-T-01: Vytvoření turnaje

Vytvoření turnaje přímo ve webové aplikaci bylo jedním ze základních požadavků. Vytvoření turnaje by mělo být jednoduché a přímočaré. Probíhá pomocí vyplnění jednoduchého formuláře se dvěma typy uživatelského vstupu. Prvním je prostý textový vstup s přidanou kontrolou formátu a hodnot. Pro případy, kde je předem známá množina všech možností, se používá druhý typ, a tím je jednoduchý výběr jedné z připravených hodnot.

Aktéři Organizátor s právy, Administrátor

Předpoklady (Pre-conditions)

- Uživatel je přihlášený.
- Uživatel má práva *organizer* nebo *admin*.
- Na stránce *Profil* se zobrazuje možnost *Přidat turnaj*.
- Uživatel vyplní všechny povinné údaje.

Následky (Post-conditions)

- Vytvořený turnaj je přidán do databáze a zobrazuje se v přehledu turnajů.

Hlavní scénář

1. Uživatel se přesune na stránku *Profil* pomocí hlavního menu v pravém horním rohu.
2. Uživatel klikne na tlačítko *Přidat turnaj* v sekci *Moje turnaje*.
3. Aplikace přesměruje uživatele na stránku pro vytvoření nového turnaje.
4. Uživatel vyplní požadovaná data a klikne na tlačítko *Přidat turnaj*.
5. Aplikace ověří vstupní hodnoty. Pokud jsou některé z nich chybné, vyzve uživatele, aby je opravil výpisem chybové hlášky. V opačném případě odešle data na server.
6. Server odešle odpověď zpět na klienta. Pokud se objeví nějaká chyba, dojde k vypsání chybové hlášky. V opačném případě aplikace přesměruje uživatele zpět na stránku profilu.

2.3.2 UC-T-02: Úprava turnaje

Zásadní funkcionalitou je možnost data o turnaji libovolně upravovat. Organizátor může upravovat pouze své ručně vytvořené turnaje, administrátor všechny.

Aktéři Organizátor s právy, Administrátor

Předpoklady (Pre-conditions)

- Uživatel je přihlášený.
- Uživatel má práva *organizer* nebo *admin*.
- Na stránce *Profil* v sekci *Moje turnaje* je alespoň jeden turnaj.
- Při rozkliknutí detailu turnaje se zobrazuje tlačítko *Upravit turnaj*.

Následky (Post-conditions)

- Provedené úpravy jsou propsány do databáze a zobrazí se v přehledu turnajů.

Hlavní scénář

1. Uživatel se přesune na stránku *Profil* pomocí hlavního menu v pravém horním rohu.
2. Uživatel vybere turnaj v sekci *Moje turnaje* a po rozkliknutí detailu turnaje vybere možnost *Upravit turnaj*.
3. Aplikace přesměruje uživatele na stránku pro úpravu existujícího turnaje.
4. Na stránce se zobrazí všechny dostupné informace k úpravě, každá s vlastním tlačítkem *editovat*.
5. Po stisknutí tlačítka *editovat* u konkrétní informace se objeví pole pro textový vstup/výčet možností a tlačítko *uložit*.
6. Stisknutím tlačítka *uložit* dojde k ověření zadané hodnoty a formátu. Pokud je vstup chybný, vyzve aplikace uživatele, aby ho opravil výpisem chybové hlášky. V opačném případě odešle data na server.
7. Server odešle odpověď zpět na klienta. Pokud se objeví nějaká chyba, dojde k vypsání chybové hlášky. V opačném případě se aktualizuje hodnota zvoleného atributu.

8. Ve chvíli, kdy uživatel tímto způsobem upraví všechny potřebné údaje, klikne na konci formuláře na tlačítko *Dokončit úpravy*, které ho přesměruje zpět na stránku profilu.

2.3.3 UC-T-03: Smazání turnaje

Stejně jako je důležitá možnost turnaj upravit, tak musí mít uživatel možnost turnaj i smazat. Opět administrátor může mazat libovolné záznamy, zatímco organizátoři mají přístup jen ke svým turnajům, které vytvořili ručně přímo v aplikaci.

Aktéři Organizátor s právy, Administrátor

Předpoklady (Pre-conditions)

- Uživatel je přihlášený.
- Uživatel má práva *organizer* nebo *admin*.
- Na stránce *Profil* v sekci *Moje turnaje* je alespoň jeden turnaj.
- Při rozkliknutí detailu turnaje se zobrazuje tlačítko *Smazat turnaj*.

Následky (Post-conditions)

- Vybraný turnaj je smazán z databáze a dále se již nezobrazuje v přehledu turnajů.

Hlavní scénář

1. Uživatel se přesune na stránku *Profil* pomocí hlavního menu v pravém horním rohu.
2. Uživatel vybere turnaj v sekci *Moje turnaje* a po rozkliknutí detailu turnaje vybere možnost *Smazat turnaj*.
3. Aplikace vyzve uživatele k potvrzení, že vybraný turnaj chce opravdu smazat.
4. Po potvrzení je informace o odstranění turnaje odeslána na server.
5. Server odešle odpověď zpět na klienta. Pokud se objeví nějaká chyba, dojde k vypsání chybové hlášky a turnaj není smazán. V opačném případě je turnaj úspěšně odstraněn z databáze.

2.3.4 UC-T-04: Zobrazení všech turnajů vytvořených daným uživatelem

Každý organizátor by měl mít možnost vidět všechny turnaje, které vytvořil (v aplikaci) přehledně na jednom místě. Administrátor využívá stejné zobrazení, ale k dispozici má všechny existující turnaje (vytvořené ručně i získané z webu).

Aktéři Organizátor s právy, Administrátor

Předpoklady (Pre-conditions)

- Uživatel je přihlášený.
- Uživatel má práva *organizer* nebo *admin*.
- Uživatel má vytvořený alespoň jeden turnaj v případě, že se jedná o organizátora.
- V databázi existuje alespoň jeden záznam v případě, že se jedná o administrátora.

Následky (Post-conditions)

- Každý turnaj umožňuje zobrazení detailu po rozkliknutí, včetně možností *Upravit turnaj* a *Smazat turnaj*.

Hlavní scénář

1. Uživatel se přesune na stránku *Profil* pomocí hlavního menu v pravém horním rohu.
2. Aplikace zobrazí odpovídající záznamy. V případě administrátora všechny existující.
3. Aplikace u každého záznamu zobrazuje (i bez otevření detailu) informace dostatečné k identifikaci konkrétního turnaje (datum, název, kapacita).

2.3.5 UC-T-05: Zobrazení všech turnajů, na které je uživatel přihlášen

Stejně jako je pro organizátora důležité vidět, jaké turnaje vytvořil, tak hráč potřebuje vidět turnaje, na které je přihlášen. Tato funkcionalita se opět vztahuje pouze na turnaje, které byly vytvořeny ručně v aplikaci.

Aktéři Hráč

Předpoklady (Pre-conditions)

- Uživatel je přihlášený.
- Uživatel má práva *player*.
- Uživatel je přihlášený alespoň na jeden turnaj vytvořený v aplikaci.

Následky (Post-conditions)

- V sekci *Moje turnaje* jsou zobrazeny všechny turnaje, na které se uživatel přihlásil přes aplikaci.
- Každý turnaj umožňuje zobrazení detailu po rozkliknutí, včetně přihlášených týmů a tlačítka *Odhlásit*, jehož funkce odpovídá názvu.

Hlavní scénář

1. Uživatel se přesune na stránku *Profil* pomocí hlavního menu v pravém horním rohu.
2. Aplikace zobrazí sekci *Moje turnaje*, kde jsou zobrazeny všechny turnaje, na které se uživatel přihlásil přes aplikaci.
3. Aplikace u každého záznamu zobrazuje (i bez otevření detailu) informace dostatečné k identifikaci konkrétního turnaje (datum, název, kapacita).

2.3.6 UC-T-06: Přihlášení na turnaj

Při ručním vytváření nebo úpravě turnaje v aplikaci má organizátor právo zpřístupnit hráčům přihlašování na turnaje. Administrátor má tuto možnost u kteréhokoliv turnaje, ale u těch, které jsou získány pomocí API nebo scrapingu, skoro vždy funguje přihlašování přímo ve zdroji. Tento use case má tedy své uplatnění primárně u uživatelů, kteří nemají vlastní webové stránky a pro správu a prezentaci turnajů využívají právě tuto platformu.

Aktéři Hráč

Předpoklady (Pre-conditions)

- Uživatel je přihlášený.
- Uživatel má práva *player*.
- Mezi dostupnými turnaji existuje alespoň jeden, který umožňuje přihlášení přes tuto aplikaci.

Následky (Post-conditions)

- Data o přihlášení včetně jména spoluhráče jsou uložena do databáze.
- Přihlášený turnaj se objeví na profilu uživatele v sekci *Moje turnaje*.
- Přihlášený tým se objeví v sekci *Přihlášené týmy* (případně *Náhradníci*) v detailu turnaje.

Hlavní scénář

1. Uživatel vyhledá na hlavní straně turnaj, na který se chce přihlásit.
2. Uživatel otevře detail turnaje a zkontroluje, že se na něj může přihlásit (vidí tlačítko *Přihlásit* a sekci *Přihlášené týmy*).
3. Uživatel klikne na tlačítko *Přihlásit*, čímž se zobrazí formulář pro vyplnění jména spoluhráče.
4. Uživatel klikne na tlačítko *Přihlásit*, čímž se zobrazí formulář pro vyplnění jména a příjmení spoluhráče.
5. Uživatel vyplní obě kolonky a potvrdí stiskem *Přihlásit*.
6. Aplikace zkontroluje, že oba vstupy jsou text a nejsou prázdné. Pokud je vše v pořádku, jsou data odeslána na server. V opačném případě je uživatel vyzván opravit/doplnit vstupní data.
7. Server odešle odpověď zpět na klienta. Pokud se objeví nějaká chyba, dojde k vypsání chybové hlášky. V opačném případě se tým objeví mezi přihlášenými týmy a spolu s tím se místo *Přihlásit* objeví tlačítko *Odhlásit*.

2.3.7 UC-T-07: Zobrazení podrobností o turnaji

Informace o turnajích se zobrazují na hlavní straně aplikace i v profilu uživatele, jak je již zmíněno v předchozích případech užití. V obou případech jsou ale na první pohled vidět pouze vybrané informace. Hlavním důvodem je přehlednost a úspora místa, protože atributů pro zobrazení je celkem 11. Proto se na hlavní straně zobrazují v základu pouze datum, název, kategorie a kapacita, které jsou pro uživatele nejdůležitější, a pro více podrobností je nutné vybrat konkrétní turnaj.

Aktéři Hráč, Organizátor, Organizátor s právy, Administrátor, Nepřihlášený uživatel

Předpoklady (Pre-conditions)

- Existuje alespoň jeden turnaj v databázi.

Následky (Post-conditions)

- Zobrazení všech dostupných informací o konkrétním turnaji.

Hlavní scénář

1. Uživatel se nachází buď v profilu, nebo na hlavní straně, kde je tabulka s alespoň jedním turnajem.
2. Uživatel zvolí libovolný turnaj a klikne kamkoliv v řádku, kde se turnaj nachází.
3. Aplikace zobrazí dodatečné informace o zvoleném turnaji, jako je úroveň, cena startovného nebo jméno organizátora.

2.3.8 UC-T-08: Filtrování zobrazených turnajů

Jen s obtížemi se najde hráč, kterého by zajímaly všechny turnaje. Například ženy nebudou zajímat mužské kategorie a stejně tak výkonnostní hráči pravděpodobně nebudou vyhledávat turnaje pro amatéry. Proto je k dispozici několik filtrů, které slouží k výběru záznamů pomocí několika základních kritérií. Konkrétně se jedná o město, areál, úroveň a kategorii. Každý z filtrů nabízí variantu multichoice, tj. uživatel si může například zvolit současně kategorii muži i mixy. Kombinace více filtrů najednou je samozřejmostí. Konkrétní požadavek může tedy vypadat následovně: Uživatel hledá amatérské turnaje v Praze v kategorii muži nebo mixy.

Aktéři Každý uživatel kromě systému

Předpoklady (Pre-conditions)

- V databázi se nachází alespoň jeden záznam o turnaji.

Následky (Post-conditions)

- Zobrazení všech turnajů na základě zvolených požadavků.

Hlavní scénář

1. Uživatel se nachází na hlavní straně, kde je tabulka s alespoň jedním turnajem.
2. Uživatel v sekci filtrů nacházející se přímo nad tabulkou/kalendářem s turnaji zvolí požadavky na filtrování.
3. Uživatel stiskne tlačítko *Filtrovat*.
4. Aplikace odešle informace na server.
5. Server vrátí odpověď s turnaji, které odpovídají požadavkům uživatele.

2.3.9 UC-U-01: Registrace

Většina akcí prováděných v aplikaci vyžaduje přihlášení uživatele a tomu předchází registrace. Registrovat se uživatel může jako organizátor i jako hráč. Tyto dvě role jsou záměrně odděleny, protože každá umožňuje jiné akce a poskytuje odlišné uživatelské rozhraní. Znamená to tedy, že obě varianty jsou zcela odděleny a každá vyžaduje samostatnou registraci (uživatelské jméno/email i heslo mohou být stejná pro oba profily).

Aktéři Nepřihlášený uživatel

Předpoklady (Pre-conditions)

- Uživatel se stejnou emailovou adresou ještě není pro danou roli registrován.

Následky (Post-conditions)

- Uživatel je uložen do databáze a má možnost se přihlásit.
- Uživatel má unikátní email pro vybranou roli.

Hlavní scénář

1. Uživatel na hlavní straně vybere v pravém horním rohu možnost *Přihlásit*.
2. Aplikace otevře stránku s formulářem pro přihlášení a dalšími možnostmi.
3. Uživatel vybere možnost *Registrovat* ve spodní části stránky.
4. Aplikace přesměruje uživatele na stránku s formulářem pro registraci.
5. Uživatel vyplní všechny požadované údaje a pro potvrzení vybere možnost *Registrovat*.
6. Aplikace ověří vstupní hodnoty. Pokud jsou některé z nich chybné (špatný formát nebo se hesla neshodují), vyzve uživatele, aby je opravil výpisem chybové hlášky. V opačném případě odešle data na server.
7. Server odešle odpověď zpět na klienta. Pokud se objeví nějaká chyba (například uživatel již existuje), dojde k vypsání chybové hlášky a uživatel není registrován. V opačném případě aplikace přesměruje uživatele zpět na stránku Přihlášení.

Alternativní scénář — registrace přes Google

1. Uživatel na hlavní straně vybere v pravém horním rohu možnost *Přihlásit*.
2. Aplikace otevře stránku s formulářem pro přihlášení a dalšími možnostmi.
3. Uživatel zvolí přihlášení přes Google výběrem možnosti *Google*.
4. Aplikace zobrazí okno pro výběr účtu Google.
5. Uživatel zvolí Google účet a aplikace uživatele registruje a automaticky přihlásí.
6. Aplikace přesměruje přihlášeného uživatele na hlavní stranu.

2.3.10 UC-U-02: Přihlášení

Jak je již zmíněno u registrace, většina akcí vyžaduje přihlášení. Jedná se tedy o stěžejní funkcionalitu pro maximální využití aplikace. U přihlášení uživatel stejně jako u registrace volí, zda si přeje přihlášení jako hráč, či jako organizátor.

Aktéři Nepřihlášený uživatel

Předpoklady (Pre-conditions)

- Uživatel je pro zvolenou roli registrován.
- Uživatel není přihlášen.

Následky (Post-conditions)

- Uživatel je přihlášen.
- Uživatel má možnosti na základě svých oprávnění.

Hlavní scénář

1. Uživatel na hlavní straně vybere v pravém horním rohu možnost *Přihlásit*.
2. Aplikace otevře stránku s formulářem pro přihlášení a dalšími možnostmi.
3. Uživatel vyplní všechny požadované údaje a pro potvrzení vybere možnost *Přihlásit*.
4. Aplikace ověří vstupní hodnoty. Pokud jsou některé z nich chybné, vyzve uživatele, aby je opravil výpisem chybové hlášky. V opačném případě odešle data na server.
5. Server odešle odpověď zpět na klienta. Pokud se objeví nějaká chyba (například uživatel se zadaným emailem neexistuje), dojde k vypsání chybové hlášky a uživatel není přihlášen. V opačném případě aplikace uživatele přihlásí a přesměruje zpět na hlavní stranu.

Alternativní scénář — přihlášení přes Google

1. Uživatel na hlavní straně vybere v pravém horním rohu možnost *Přihlásit*.
2. Aplikace otevře stránku s formulářem pro přihlášení a dalšími možnostmi.
3. Uživatel zvolí přihlášení přes Google výběrem možnosti *Google*.
4. Aplikace zobrazí okno pro výběr účtu Google.
5. Uživatel zvolí Google účet a aplikace uživatele přihlásí.
6. Aplikace přesměruje přihlášeného uživatele na hlavní stranu.

2.3.11 UC-U-03: Zapomenuté heslo

Jedním ze základních požadavků na správu uživatelů je možnost vygenerování nového hesla v případě ztráty toho stávajícího. Probíhá to většinou prostřednictvím zaslání nového hesla na email uživatel, a tady tomu není jinak. I tady funguje rozlišení role hráče a organizátora.

Aktéři Nepřihlášený uživatel

Předpoklady (Pre-conditions)

- Uživatel existuje v databázi pro vybranou roli.

Následky (Post-conditions)

- Uživatel má vygenerované nové heslo.

Hlavní scénář

1. Uživatel na hlavní straně vybere v pravém horním rohu možnost *Přihlásit*.
2. Aplikace otevře stránku s formulářem pro přihlášení a dalšími možnostmi.
3. Uživatel zvolí možnost *Zapomněli jste?*, která se nachází u pole pro vyplnění hesla.
4. Aplikace přesměruje uživatele na stránku zapomenutého hesla s formulářem pro vyplnění emailu.
5. Uživatel zadá email, který sloužil jako uživatelské jméno pro zapomenuté heslo.
6. Aplikace ověří formát emailu. Pokud je formát chybný, vyzve uživatele, aby email opravil výpisem chybové hlášky. V opačném případě odešle data na server.
7. Server odešle odpověď zpět na klienta. Pokud email existuje a nevyskytne se žádná chyba, je uživateli zasláno nové heslo na uvedený email. Aplikace zároveň vypíše oznámení *Nové heslo bylo zasláno na uvedený email*. V opačném případě dojde k vypsání chybové hlášky.

2.3.12 UC-U-04: Změna hesla

Změna hesla může být užitečná v různých situacích. V každém případě je ale nedílnou součástí správy uživatelů, hlavně v souvislosti s generováním nového hesla.

Aktéři Hráč, Organizátor, Organizátor s právy, Administrátor

Předpoklady (Pre-conditions)

- Uživatel je přihlášený.
- Uživatel zná své současné heslo.

Následky (Post-conditions)

- Nové heslo je propsáno do databáze.

Hlavní scénář

1. Uživatel se přesune na stránku *Profil* pomocí hlavního menu v pravém horním rohu.
2. Uživatel zvolí možnost *Změnit heslo* v sekci *Osobní údaje*.
3. Aplikace přesměruje uživatele na stránku s formulářem pro změnu hesla.
4. Uživatel zadá své původní a dvakrát nové heslo. Změnu hesla potvrdí tlačítkem *Změnit heslo*.
5. Aplikace ověří, že se nová hesla shodují, a odešle data na server. V opačném případě vyzve uživatele, aby hesla zadal znovu, výpisem chybové hlášky.
6. Server odešle odpověď zpět na klienta. Pokud je původní heslo zadáno správně a nevyskytne se žádná chyba, je uživateli heslo úspěšně změněno. V opačném případě dojde k vypsání chybové hlášky a je nutné akci zopakovat.

2.3.13 UC-U-05: Zobrazení údajů o uživateli

Každý přihlášený uživatel by měl mít možnost zkontrolovat své osobní údaje. K tomu slouží tento use case.

Aktéři Hráč, Organizátor, Organizátor s právy, Administrátor

Předpoklady (Pre-conditions)

- Uživatel je přihlášený.

Následky (Post-conditions)

- Údaje o uživateli obsahují jeho celé jméno, email a příslušnou roli (player, organizer nebo admin).

Hlavní scénář

1. Uživatel se přesune na stránku *Profil* pomocí hlavního menu v pravém horním rohu.
2. Aplikace zobrazí sekci *Osobní údaje* v horní části stránky s celým jménem, emailem a rolí uživatele.

2.3.14 UC-U-06: Žádost o přidělení práv organizátora

K získání práv pro správu turnajů v aplikaci je třeba poslat žádost. Tato možnost se nachází v profilu každého organizátora, který taková práva ještě nemá.

Aktéři Organizátor

Předpoklady (Pre-conditions)

- Uživatel je přihlášený jako organizátor.
- Uživatel má roli *basic*, tj. nemá přidělenou roli *organizer*.

Následky (Post-conditions)

- Uživatel má přidělenou roli *organizer*.
- Uživatel může vytvářet, mazat a editovat vlastní turnaje.

Hlavní scénář

1. Uživatel se přesune na stránku *Profil* pomocí hlavního menu v pravém horním rohu.
2. Aplikace zobrazí sekci *Moje turnaje* ve spodní části stránky s návrhem na zaslání žádosti o práva spravovat turnaje a tlačítkem *Poslat žádost*.
3. Uživatel zasláním žádosti odešle požadavek na server, odkud je odeslán email administrátorovi s údaji o uživateli a žádostí o přidělení oprávnění.
4. Administrátor změní roli uživatele z *basic* na *organizer* a od té chvíle má uživatel všechna potřebná oprávnění pro kompletní správu svých turnajů.

2.3.15 UC-D-01: Vygenerování iCal linku

Pro lepší přehled a integraci událostí do svého kalendáře slouží iCal link, který je možné si vygenerovat a upravit dle vlastních preferencí. Výchozím výstupem jsou všechny turnaje, ale odkaz je možné upravit volbou filtrů (např. pouze město Praha nebo kategorie mixy).

Aktéři Každý uživatel kromě systému

Předpoklady (Pre-conditions)

- Tento use case nemá žádné povinné předpoklady.

Následky (Post-conditions)

- Uživatel má k dispozici odkaz na míru, který může použít pro import událostí do svého kalendáře.

Hlavní scénář

1. Uživatel na hlavní straně přepne formát zobrazení dat na *Kalendář*.
2. Uživatel zvolí filtry dle vlastního zájmu. Nemusí použít žádné.
3. Uživatel zvolí možnost *Vygenerovat URL*, která se nachází pod kalendářem s turnaji.
4. Aplikace zobrazí URL pro export kalendáře upravené na základě zvolených filtrů.

2.3.16 UC-D-02: Automatické mazání starých turnajů

Aplikace slouží pouze k přehledu o turnajích nadcházejících. Proto jsou turnaje, které již proběhly, automaticky mazány.

Aktéři Systém

Předpoklady (Pre-conditions)

- Tento use case nemá žádné povinné předpoklady.

Následky (Post-conditions)

- V databázi se nacházejí pouze turnaje s datem dnes nebo později.

Hlavní scénář

1. Systém periodicky (každou hodinu) provádí stahování a aktualizaci dat o turnajích. V rámci aktualizace je kontrolován každý turnaj v databázi.
2. Systém kontroluje datum konání turnaje, a pokud je datum starší od současného data o více než den, je turnaj smazán.

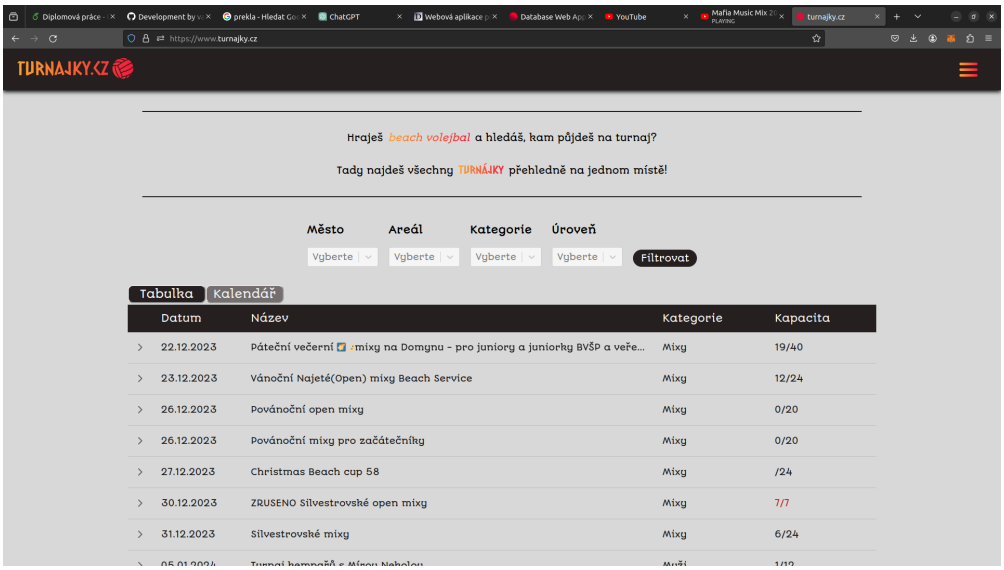
2.4 Návrh uživatelského rozhraní

V uživatelském rozhraní hraje největší roli umístění jednotlivých komponent, intuitivní navigace po webové stránce a predikovatelné chování. Práce má za úkol příchozímu uživateli prezentovat nadcházející turnaje v plážovém volejbale. Je tedy důležité, aby tyto informace byly co nejpřehlednější a co nejlépe dostupné. Zároveň je úkolem umožnit vytváření a správu turnajů, což by měl zvládnout prakticky kdokoliv. Návrh uživatelského rozhraní se tedy soustředí hlavně na tyto dvě stěžejní funkcionality. Kromě toho ale nabízí i spoustu dalších aditivních možností, včetně vygenerování linku pro import kalendáře s turnaji nebo možnost se přes aplikaci na turnaje

přihlašovat. V této sekci bude čtenář seznámen s návrhem uživatelského rozhraní různých částí aplikace a s jejich chováním.

Upozornění

Velikost jednotlivých prvků v přiložených snímcích obrazovky nemusí zcela přesně odpovídat velikosti při reálném zobrazení. Důvodem je snaha o zobrazení větší části stránky v jednom snímku. Pro porovnání je v obrázku2.2 zobrazena domovská strana v originální velikosti.



Obrázek 2.2 Hlavní strana v originální velikosti

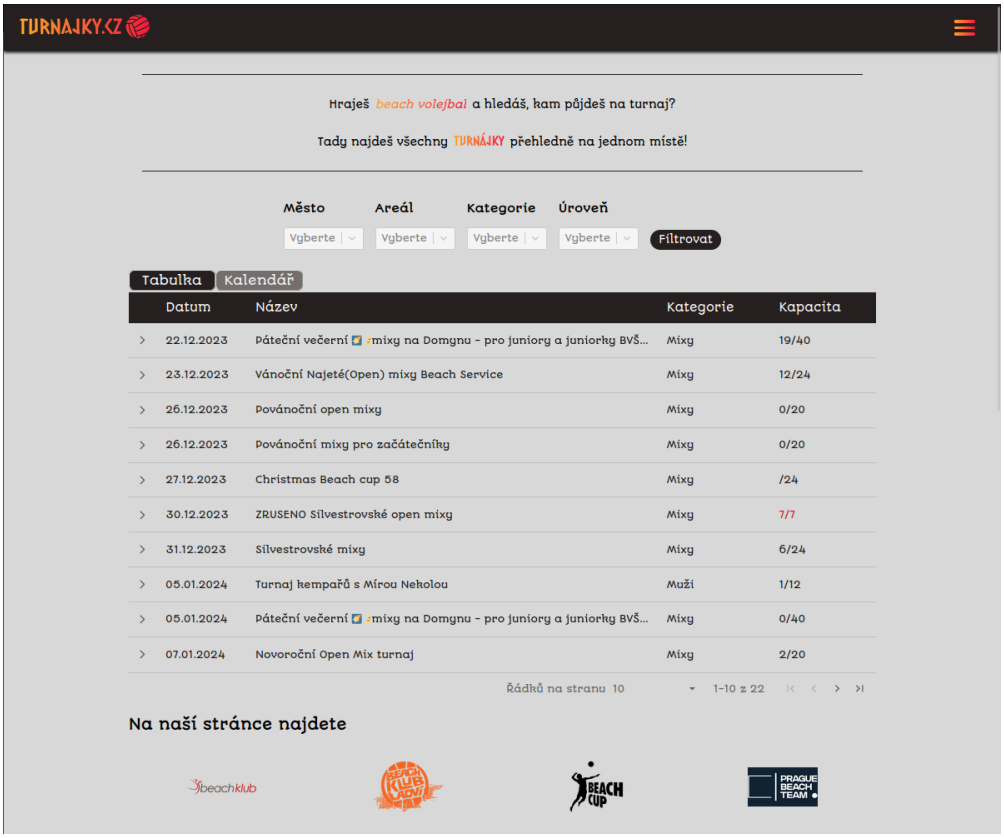
2.4.1 Hlavní strana

Libovolný uživatel, který přistoupí na webovou stránku, by měl mít prakticky ihned přístup k seznamu turnajů. K tomu slouží hlavní strana aplikace. Skládá se z navigačního panelu, který je součástí většiny stránek v aplikaci, seznamu filtrů pro selekci turnajů, tabulky/kalendáře s turnaji (v závislosti na stavu přepínače), seznamu areálů a footeru. Kopie obrazovky hlavní strany se nacházejí na obrázcích2.3 a2.4.

2.4.2 Tabulka s turnaji

Součástí hlavní strany je tabulka s turnaji2.5, která plní hlavní roli celé aplikace, zobrazit uživateli nadcházející turnaje v plážovém volejbale. K tabulce má přístup libovolný uživatel (není třeba přihlášení). Tabulka obsahuje na první pohled pouze základní informace o turnaji, kterými jsou datum, název, kategorie a kapacita. V případě zájmu o více informací může uživatel otevřít detail kliknutím kamkoliv do příslušného řádku v tabulce. Tím se mu zobrazí aditivní informace včetně ceny startovného, areálu a jména pořadatele.

V případě, že to turnaj umožňuje, může se uživatel, který je v aplikaci přihlášený (jako hráč) na turnaj registrovat2.6. Jedinou podmínkou pro přihlášení je zadání jména spoluhráče/spoluhráčky.



Obrázek 2.3 Hlavní strana 1. část

2.4.3 Kalendář s turnaji

K tabulce je v horní části připojený přepínač, který umožňuje změnit výchozí zobrazení ve formě tabulky na kalendář2.7. Pokud tedy uživatel preferuje tento způsob zobrazení, stačí překliknout z možnosti *tabulka* na *kalendář*. Kalendář nabízí výchozí zobrazení současného měsíce, ale dovoluje uživateli změnit formát na týden, den nebo agendu. Pro zobrazení detailů2.8 opět stačí kliknout na vybraný turnaj. Ve zobrazení kalendáře není možné se přihlásit na turnaj.

Jednou z již zmíněných přidaných funkcionalit je tlačítko *Vygenerovat URL* hned pod kalendářem, které vytvoří iCal link pro import turnajů do vlastního kalendáře. Exportované turnaje jsou omezeny na základě zvolených filtrů při generování odkazu.

2.4.4 Navigační panel

Navigační panel slouží, jak už název napovídá, k navigaci. V případě této práce má za úkol zajistit navigaci na hlavní stranu, profil a přihlášení/registraci uživatele. V levé části panelu je velké logo aplikace *turnajky.cz*, které po kliknutí odkazuje odkudkoliv zpět na hlavní stranu. Logo stránky odkazující na hlavní stranu je už určitým zažitým standardem, takže by mělo být zajištěno jeho intuitivní používání. V pravé části se nachází tlačítko *Přihlásit*2.9. Není asi velkým překvapením, že odkazuje na stránku s přihlášením. Z pohledu přihlášeného uživatele se ale na stejném místě nachází tzv. *hamburger* menu2.10. To poskytuje odkazy na profil, o nás a možnost odhlášení2.11. Chování menu by opět mělo být intuitivní. Jedním kliknutím na 3 vodorovné čáry se menu otevře a dalším kliknutím zase zavře.

Obrázek 2.4 Hlavní strana 2. část

2.4.5 Profil

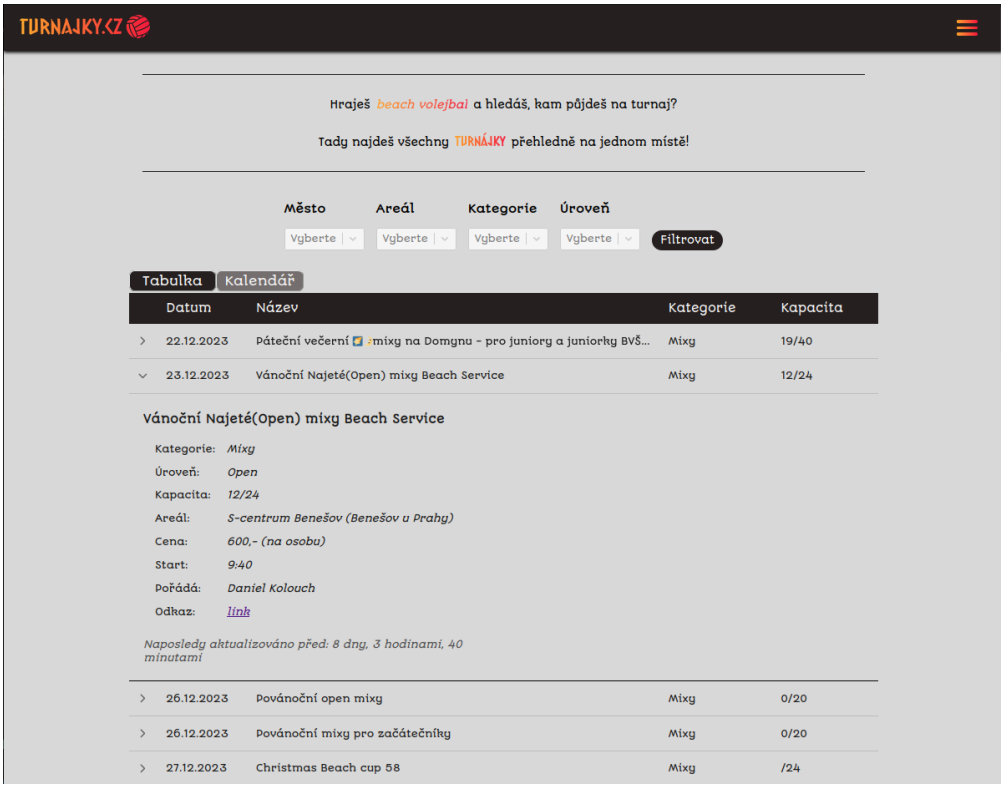
Profil může mít hned několik podob v závislosti na roli přihlášeného uživatele. Část, která se nemění a na straně profilu se očekává, jsou informace o přihlášeném uživateli (celé jméno, email a role) a hned pod nimi tlačítko pro změnu hesla. Proměnnou je sekce *Moje turnaje*. Název zůstává sice neměnný, avšak obsah už závisí na roli uživatele.

Role basic

Pokud se jedná o čerstvě registrovaného organizátora, který disponuje rolí **basic** (tento název bude v návaznosti na tuto diplomovou práci změněn, protože může být matoucí), nachází se v této sekci pouze krátký text a tlačítko pro zaslání žádosti o práva pro správu turnajů. Snímek obrazovky je dostupný na obrázku2.12.

Role organizer a admin

Pokud se jedná o organizátora, kterému byla již práva přidělena (s rolí **organizer**), nachází se v této sekci tabulka s turnaji, které uživatel v aplikaci vytvořil. V případě administrátora jsou to všechny turnaje. Pod tabulkou je navíc tlačítko *Přidat turnaj*, jehož funcionalitu snad není třeba vysvětlovat. Snímek obrazovky je dostupný na obrázku2.13.



Obrázek 2.5 Tabulka s otevřeným detailem turnaje

Role player

Poslední možností je pozice hráče s odpovídající rolí *player*, který v sekci vidí turnaje, na které se pomocí aplikace přihlásil. Snímek obrazovky je dostupný na obrázku2.14.

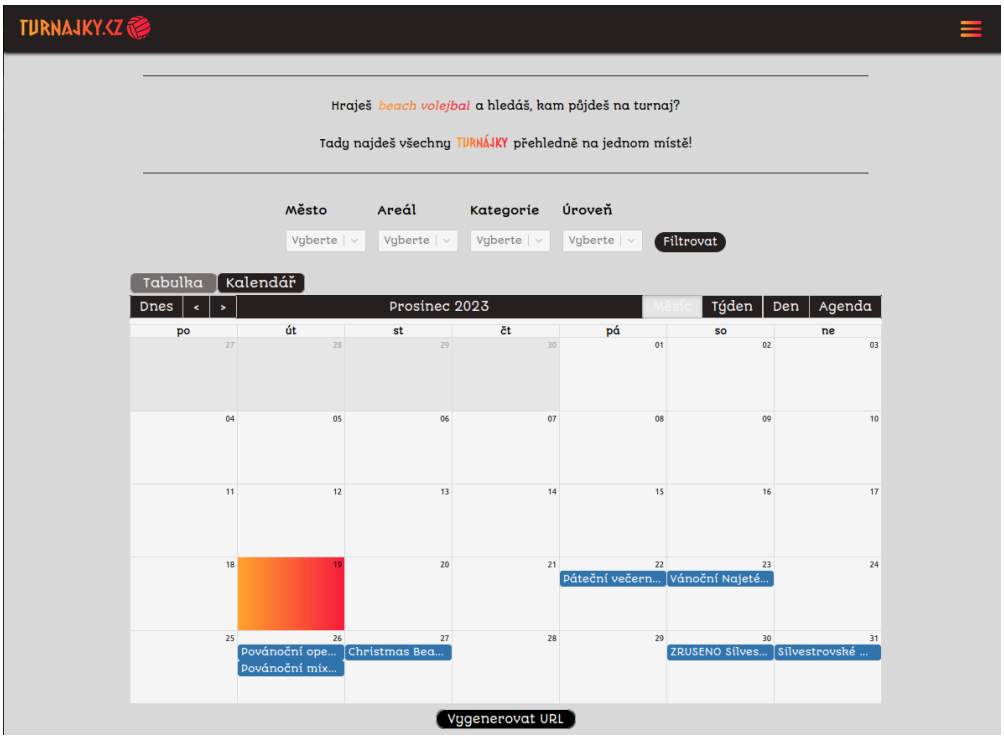
2.4.6 Vytvoření (úprava a smazání) turnaje

Vytvoření turnaje je druhým hlavním požadavkem na tuto práci, a mělo by proto být opět co nejvíce intuitivní. Tato možnost se nachází v Profilu uživatele a týká se pouze rolí *organizer* a *admin*. Tito uživatelé v sekci *Moje turnaje* mají k dispozici výrazné tlačítko *Přidat turnaj*.

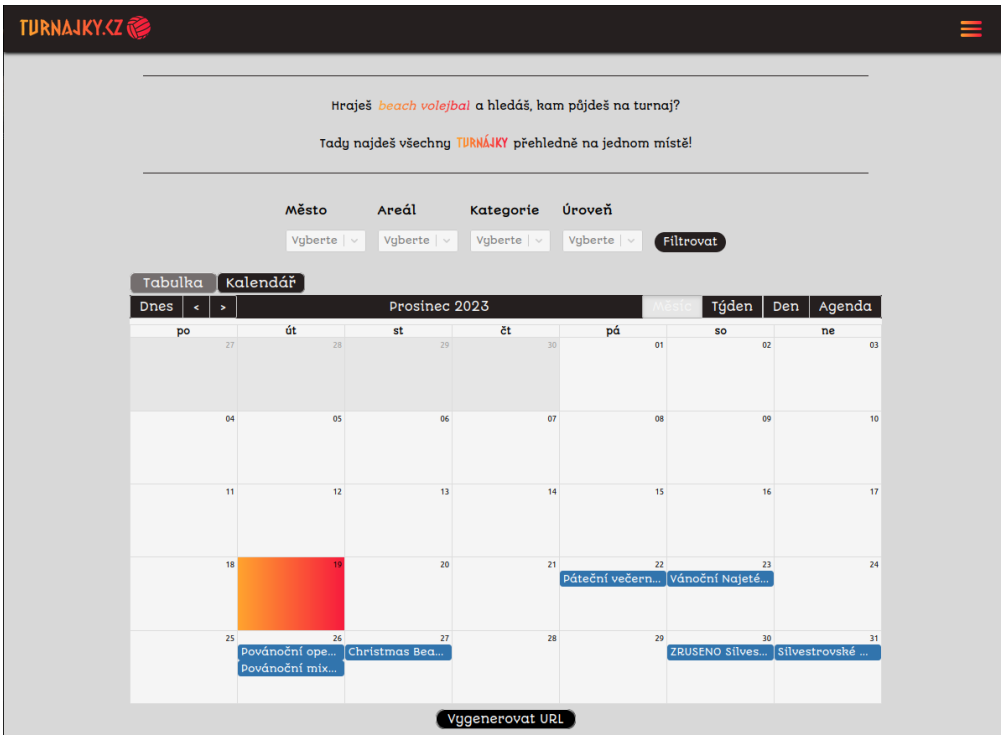
Po zvolení této možnosti se otevře nová stránka pro vytvoření nového turnaje2.15, která by měla být opět uživatelsky přívětivá. Všechny textové vstupy, u kterých není na první pohled jasný formát, mají definovaný placeholder, který navádí uživatele ke správnému formátu vstupu. U vstupů s předem známými variantami se nachází místo textového vstupu pouze výběr jedné z možností. Odeslat formulář a tím vytvořit turnaj je možné pouze ve chvíli, kdy jsou vyplněny všechny informace ve správném formátu. Uživatel tak učiní stiskem tlačítka *Přidat turnaj*. To by mělo vyvolat očekávané chování a odkázat uživatele zpět na profil.

Pro jakoukoliv manipulaci s turnajem stačí ho kliknutím zvolit a tím otevřít detail. V detailu se nachází mimo jiné i tlačítka pro úpravu a smazání turnaje.

Obrázek 2.6 Tabulka s otevřeným detailem turnaje s možným přihlášením



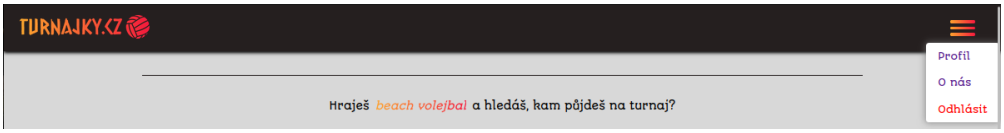
Obrázek 2.7 Hlavní strana — kalendář



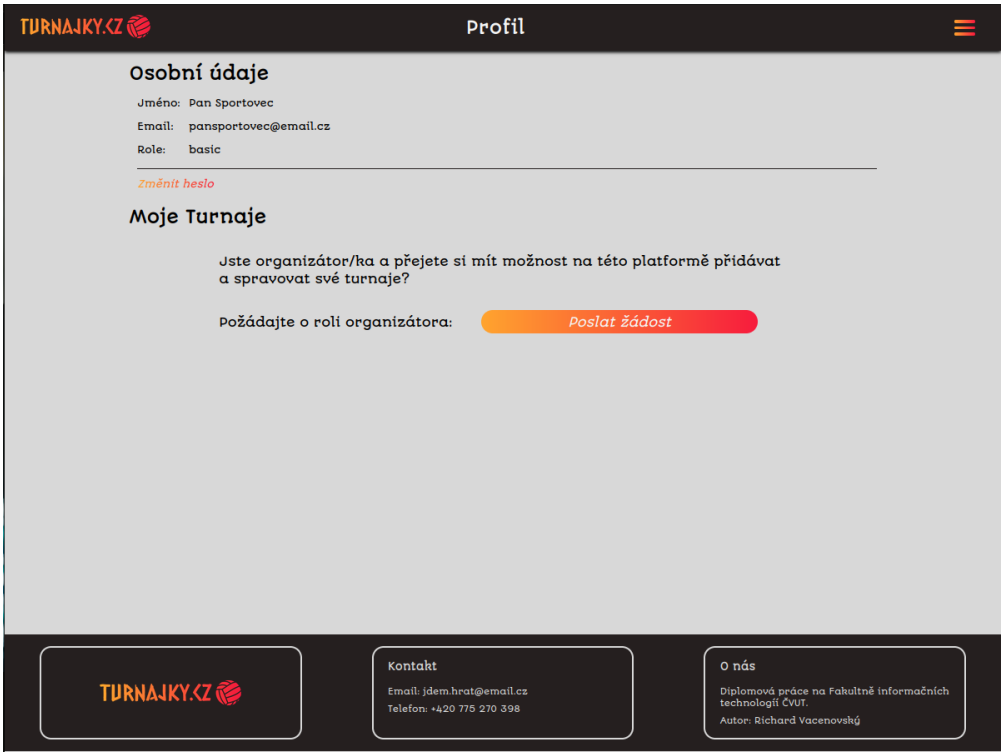
Obrázek 2.8 Detail turnaje v kalendáři

Obrázek 2.9 Navigační panel — odhlášený uživatel

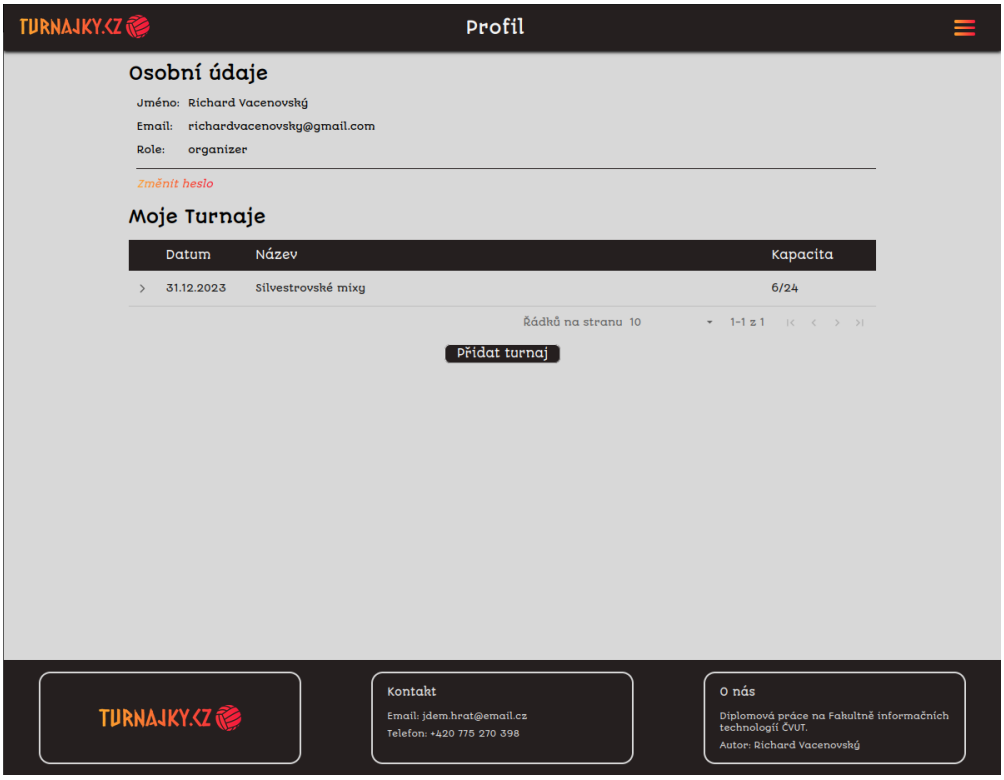
Obrázek 2.10 Navigační panel — přihlášený uživatel



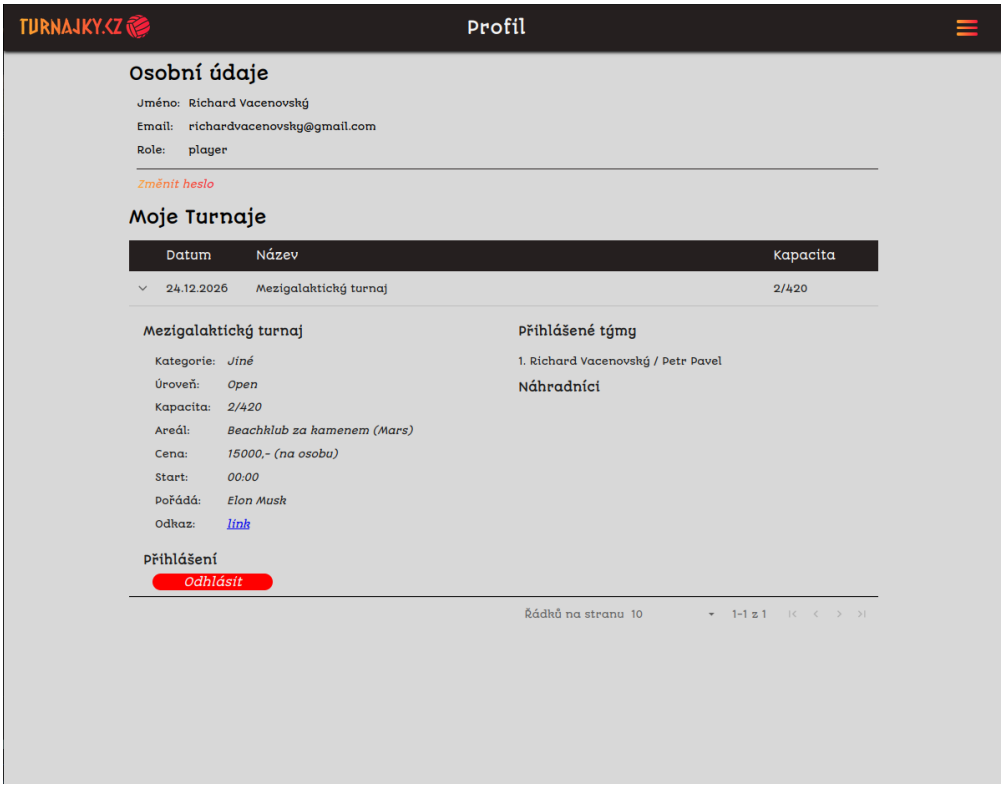
Obrázek 2.11 Navigační panel — otevřené menu



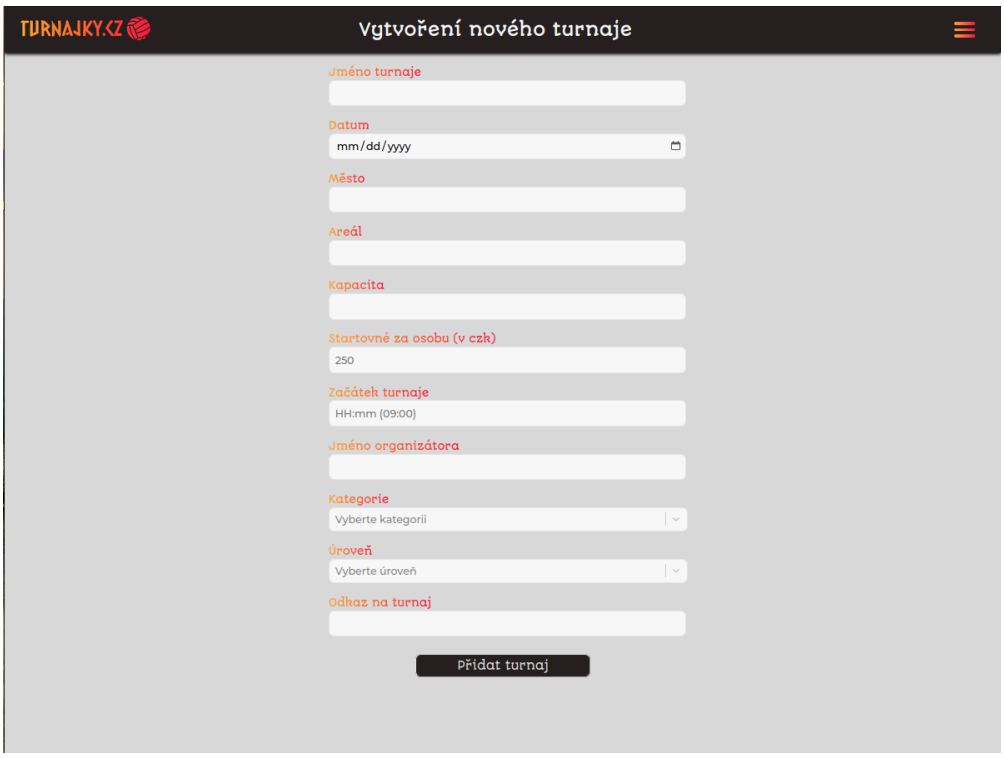
Obrázek 2.12 Profil — role basic



Obrázek 2.13 Profil — role organizátor



Obrázek 2.14 Profil — role player



Obrázek 2.15 Vytvoření nového turnaje

Kapitola 3
Implementace

V této kapitole bude přiblížena implementace jednotlivých částí aplikace. Součástí popisu budou i části kódu a případné problémy, se kterými se autor v průběhu vývoje setkal. Na konci bude rozebráno nasazení aplikace.

3.1 Databáze

Jelikož u databáze hraje hlavní roli její návrh, podrobný popis včetně všech tabulek a jejich vztahů se nachází už v sekci návrhu2.1.4. Vytvoření tabulek probíhalo pomocí knihovny SQLAlchemy v rámci implementace serverové části. V souboru *models.py* byly vytvořeny všechny třídy pro definici odpovídajících tabulek v databázi, včetně jejich vztahů. Samotné vytvoření proběhlo v příkazové řádce Flasku pomocí metody *create_all*. Příklad jedné z tříd, použitých pro vytvoření tabulky, je k dispozici zde3.3.

K dalším pomocným operacím, jednoduchému zobrazení nebo atomickým úpravám dat slouží nástroj Dbeaver — multiplatformní databázový nástroj pro vývojáře.

Samotné úložiště poskytuje ElephantSQL, služba pro hostování PostgreSQL databází. Připojení k databázi řeší už zmíněné SQLAlchemy pomocí URI s příslušnými přístupovými údaji.

3.2 Server

Implementace serveru se skládá z několika částí. Jedná se o automatizované získávání dat o turnajích, komunikaci s databází, komunikaci s klientem a logování.

3.2.1 Získání dat

Teorie různých způsobů, jak sbírat data z internetu, proběhla už v analýze. Serverové části se přímo týkají dvě ze tří zmíněných metod, a to web scraping a API.

Pro účely získání a zpracování dat o turnajích vznikly třídy *TournamentInfo* a *TournamentManagement*.

3.2.1.1 Třída TournamentInfo

Jak je možné vidět ve výpisu kódu3.1, třída *TournamentInfo* svou strukturou odpovídá tabulce *tournament* v databázi. Slouží pouze pro přehlednější ukládání dat o turnaji.

Výpis kódu 3.1 Třída TournamentInfo

```
class TournamentInfo:
    name: str
    tournament_date: date
    city: str
    areal: str
    capacity: int
    signed: int
    price: int
    start: str
    organizer: str
    category: str
    level: str
    link: str
    last_update: date
```

3.2.1.2 Třída TournamentManagement

Třída *TournamentManagement* se stará o samotnou extrakci dat z webových zdrojů a jejich následné zpracování. Obsahuje hlavně metody pro získání dat z jednotlivých zdrojů. Větší část z nich využívá web scraping a s tím knihovnu Selenium. Zjednodušený příklad postupu u takové metody vypadá následovně:

1. Otevření odkazu zdroje pomocí chrome driveru.
2. Použití metody find_element (využitím jména třídy, id nebo regulárního výrazu) pro nalezení tabulky s turnaji (příklad útržku kódu zde3.2).
3. Kontrola, zda byly nalezeny nějaké turnaje.
4. Uložení elementů, které reprezentují jednotlivé turnaje.
5. Iterace nad nalezenými turnaji.
6. Pro každý nález se informace extrahují a ukládají do kontejneru (konkrétně set).

   Většina operací využívá *try* a *except* pro případné odchycení jakékoliv chyby.

Výpis kódu 3.2 Útržek lokalizace tabulky s turnaji

```
if not self.open_chrome_with_url(url):
    return False

try:
    tournament_table = driver.find_element(By.ID, "event_list")
except:
    self.logger.error("Table␣with␣tournaments␣was␣not␣found.")
    return False
```

Důležitou roli hraje metoda *get_all_data()*, která spouští všechny ostatní metody a vrací Python list typu *TournamentInfo* s nalezenými turnaji.

3.2.1.3 Třída CzechMonths

Jak už název napovídá, jedná se o pomocnou třídu, která pracuje s českými měsíci. Konkrétně se inicializuje s textovým řetězcem reprezentujícím název měsíce v českém jazyce a poskytuje metodu *to_number()*. Ta vrací číselnou reprezentaci daného měsíce (leden — 1, únor — 2, ..., prosinec — 12).

3.2.2 Komunikace s databází

Pro komunikaci s databází poskytuje Flask knihovnu SQLAlchemy, sadu nástrojů a objektově relační mapovač, který umožňuje plnou kontrolu nad SQL.

Pro vytvoření jednotlivých tabulek a jejich závislostí sloužily stejnojmenné třídy (Tournament, User, Player a SignedTeam) v souboru *models.py*. Příklad jedné z nich je k dispozici zde3.3.

Výpis kódu 3.3 Model pro definici tabulky tournaments

```
class Tournament(db.Model):
    __tablename__ = "tournament"
    id = db.Column(db.String(32), primary_key=True, unique=True,
    default=get_uuid)
    name = db.Column(db.String(100), nullable=False)
    date = db.Column(db.Date, nullable=False)
    city = db.Column(db.String(100), nullable=False)
    areal = db.Column(db.String(100), nullable=False)
    capacity = db.Column(db.Integer)
    signed = db.Column(db.Integer)
    price = db.Column(db.Integer)
    start = db.Column(db.String(20))
    organizer = db.Column(db.String(100))
    category = db.Column(db.String(100))
    level = db.Column(db.String(100))
    link = db.Column(db.String(100))
    last_update = db.Column(db.DateTime)
    registration_enabled = db.Column(db.Boolean, default=False)

    # Add a~foreign key column refering the 'id' of the 'User' table
    user_id = db.Column(db.String(32), db.ForeignKey('user.id'))

    # Define a~relationship to the 'User' table
    creator = db.relationship('User', back_populates='tournaments')

     # Define a~relationship to the SignedTeam table
    signed_teams = db.relationship('SignedTeam',
    back_populates='tournament')

    def __repr__(self):
        return f"{self.name}\nAreal:␣{self.areal}\n\n"
```

Samotná komunikace s databází probíhá na dvou různých místech. Prvním je soubor *update.py*, který byl vytvořen hlavně pro účely nasazení a slouží jako skript pro aktualizaci dat. Využívá právě již zmíněnou třídu *TournamentManagement* a její metodu *get_all_data()* pro získání aktuálních informací o všech turnajích, které následně zapisuje do databáze. Kromě aktualizace a přidávání nových dat probíhá i kontrola, zda se v databázi nenachází staré informace a dochází k mazání již uskutečněných turnajů.

Druhým místem je jádro celé aplikace, soubor *app.py*. Stará se o všechny ostatní operace včetně manuálního přidání turnaje, registrace uživatele nebo přihlášení týmu na turnaj. Jak taková operace vypadá, je možné vidět v příkladu3.4

3.2.3 Komunikace s klientem

Komunikace s klientem probíhá pouze v již zmíněném *app.py* pomocí REST API. Příklad takové funkce je zobrazen v příkladu3.4. Jak je možné vidět v uvedeném kódu, Flask poskytuje tzv.

routing, což znamená namapování URL na specifickou funkci. Díky této funkcionalitě bylo možné jednoduše vytvořit API pro komunikaci s klientem.

Výpis kódu 3.4 Příklad funkce pro registraci

```
@app.route("/register", methods=["POST"])
def register_user():
    email = request.json.get("email", None)
    password = request.json.get("password", None)
    name = request.json.get("name", None)
    surname = request.json.get("surname", None)
    isPlayer = request.json.get("isPlayer", None)


    if isPlayer:
        user_exists = Player.query.filter_by(email=email).first()
        is not None
    else:
        user_exists = User.query.filter_by(email=email).first()
        is not None

    if user_exists:
        return jsonify({"error": "User␣already␣exists"}), 409
    hashed_password = bcrypt.generate_password_hash(password)
    .decode("utf-8")

    if isPlayer:
        new_user = Player(email=email, password=hashed_password,
        name=name,surname=surname)
    else:
        new_user = User(email=email, password=hashed_password,
        name=name,surname=surname)

    db.session.add(new_user)
    db.session.commit()

    return email, 200
```

3.2.4 Logování

O logování na straně serveru se stará třída *CustomLogger* v *Logger.py*. S využitím handlerů z knihovny *logging* jsou informace zapisovány jak do příkazové řádky, tak do souboru. Třída nabízí mimo jiné i metody pro změny ve formátování, což umožňuje srozumitelné a přehledné výpisy z běhu serveru3.5.

Výpis kódu 3.5 Příklad logu ze serveru

```
---------------------------- New update started at 2023-12-24 22:01:17
Areal: Beachklub Ladvi.
   Number of found tournaments: 1
   Processing tournament 1/1
   Scraping...
   Data scraped succesfuly!
--------------------------------------------------------------------
Areal: PBT Stresovice - Ondra Michalek
   Number of found tournaments: 2
   Processing tournament 1/2
```

```
   Scraping...
   Data scraped succesfuly!
   Processing tournament 2/2
   Scraping...
   Data scraped succesfuly!
----------------------------------------------------------------------
Areal: Beachklub Pankrac
   Number of found tournaments: 3
   Processing tournament 1/3
   Scraping...
   Data scraped succesfuly!
   Processing tournament 2/3
   Scraping...
   Data scraped succesfuly!
   Processing tournament 3/3
   Scraping...
   Data scraped succesfuly!
----------------------------------------------------------------------
Arealy: BVSP
   Number of found tournaments: 3
   Data fetched succesfuly!
----------------------------------------------------------------------
PBT Stresovice:
   Number of found tournaments: 1
   Data fetched succesfuly!
---------------------------------------- SUMMARY -----------------
Total of found tournaments: 10
Succesfully processed: 10
```

3.3 Klient

Implementace klienta je komplexní a značně rozsáhlá. Proto jsou v této sekci popsány hlavně nejdůležitější části, principy a problémy.

3.3.1 Definice pojmů spojených s Reactem

V této sekci jsou stručně rozebrány pojmy spojené s knihovnou React pro uvedení do problematiky a lepší porozumění následujícímu textu.

3.3.1.1 Single-page application

Single-page application, zkráceně SPA, je implementace webové aplikace, která načítá pouze jeden webový dokument. Aktualizuje se pouze obsah daného dokumentu s použitím API, jako je např. fetch. Taková implementace umožňuje uživateli rychlejší a efektivnější zobrazení obsahu bez nutnosti načítání celé nové stránky ze serveru.[36]

3.3.1.2 React hooks

React hook je funkce, která umožňuje využívat stav a další funkcionality v komponentách Reactu. Hook byl představen ve verzi 16.8 jako způsob, jak psát komponenty s vnitřním stavem a funkčním přístupem, místo používání tříd.[37]

UseState je funkce (hook), která umožňuje komponentám uchovávat a dynamicky aktualizovat svůj vnitřní stav. Tento hook přijímá počáteční hodnotu stavu a vrací dvojici hodnot: aktuální stav a funkci pro jeho aktualizaci. Použitím useState lze implementovat reaktivní chování komponenty a umožnit jí dynamicky reagovat na uživatelské interakce či změny dat, což přispívá k flexibilitě, znovupoužitelnosti a čitelnosti kódu.[38]

Výpis kódu 3.6 Příklad použití useState hooku

```
const [state, setState] = useState(initialState);
```

[38]

UseEffect je funkce (hook) umožňující provádění efektů v reakci na změny stavu, životního cyklu komponenty nebo jiné události. Používá se pro integraci vedlejších efektů, jako jsou načítání dat, odběr událostí nebo jiné asynchronní operace. Definuje se pomocí funkce, která určuje samotný efekt a pole závislostí. To určuje, kdy dochází ke spuštění efektu.[39]

Výpis kódu 3.7 Příklad použití useEffect hooku

```
import { useEffect } from 'react';
import { createConnection } from './chat.js';

function ChatRoom({ roomId }) {
  const [serverUrl, setServerUrl] = useState('initialUrl');

  useEffect(() => {
    const connection = createConnection(serverUrl, roomId);
    connection.connect();
    return () => {
      connection.disconnect();
    };
  }, [serverUrl, roomId]);
  // ...
}
```

[39]

3.3.1.3 React props

React prop, nebo také jednoduše prop, je zkratka pro property, neboli vlastnost. Props jsou způsobem, jak předávat data z nadřazené komponenty (rodiče) do podřízené komponenty (dítě). Jsou to parametry, na základě kterých příslušná komponenta renderuje svůj obsah.[40]

3.3.2 Komunikace se serverem

Pro komunikaci se serverem využívá klient funkci *fetch()*, která slouží k posílání HTTP požadavků na server a získávání odpovědí. Konkrétně zprostředkovává komunikaci s API a získává data ze serveru. Fetch je postaven na konceptu *promise*, což znamená, že umožňuje odeslání požadavku a reakci na odpověď, až když je operace dokončena.

V metodě dochází k deklaraci typu, hlavičky a případně těla požadavku. Mezi základní typy požadavku patří:

- GET: Nalezení požadovaných dat a odeslání zpět na klienta.
- PUT: Aktualizace záznamu v databázi.
- POST: Vytvoření nové položky v databázi.

- DELETE: Vymazání položky z databáze.

Hlavička obsahuje atribut *Content-type*, který určuje, v jakém formátu budou data odesílána. Většinou se jedná o JSON, což je i případ této práce. V případech, kdy je k odeslání dotazu potřeba určité oprávnění, odesílá se v rámci hlavičky ještě autorizační token. V tomto případě se jedná o access token, který je součástí JWT.

V metodě POST se posílá i tělo požadavku s libovolnými daty. Zde jsou to data ve formátu JSON, jak je již uvedeno v hlavičce.

Jelikož je metoda asynchronní, používá pro práci s příchozími daty metodu *then()*, která se volá až po obdržení dat ze serveru. V první řadě dochází ke kontrole status kódu (200 — OK) a v případě očekávané hodnoty jsou data dále zpracována podle konkrétní situace. Na konci se vždy nachází *catch()* pro odchycení případných chyb.

Konkrétní příklad použití metody fetch je v přiloženém kódu3.8.

Výpis kódu 3.8 Příklad volání metody fetch

```
const saveData = (form_data) => {
    setFilterResults(form_data);
    console.log("form_data", form_data);
    fetch(`${apiUrl}/filter`, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(form_data),
    })
      .then((resp) => resp.json())
      .then((resp) => setTournamentsData(resp));
  }.catch((error) => {
        if (error.response) {
          console.log(error.response);
          console.log(error.response.status);
          console.log(error.response.headers);
        }
    }
```

3.3.3 Správa stavu (state management)

React funguje na bázi komponent — částí kódu, které se dají znovu využívat na různých místech a společně tvoří celou aplikaci. Ačkoliv se jedná o praktickou architekturu, správa stavu může být v Reactu výzvou, a to hlavně u větších a komplexnějších aplikací.

Důležitou se stává hlavně proto, že umožňuje komponentám udržovat a aktualizovat informace o svém stavu během života aplikace. Existuje několik klíčových aspektů, které lze považovat za problémy, a několik různých přístupů k jejich řešení.

Nejzákladnějším řešením, které na mnoha místech používá i tato aplikace, je lokální stav udržovaný hookem useState. Tento stav je ale omezen jen na konkrétní komponentu, což může být nevýhodné v případech, kdy je třeba sdílet stav mezi různými částmi aplikace.

Pro sdílení stavu mezi komponentami se používá prop, který vždy předává rodičovská komponenta potomkovi. Někdy však nastává situace, kdy je potřeba předávat stav (nebo funkce na jeho aktualizaci) skrze mnoho vrstev komponent, což může způsobovat složitost a nečitelnost kódu. Takový problém se nazývá prop drilling.

Pro prevenci těchto problémů využívá aplikace knihovnu Recoil. Ta poskytuje centrální úložiště (store), kde může být stav komponenty uchováván a aktualizován bez prop drillingu.

Výpis kódu 3.9 Příklad řešení stavu pomocí knihovny Recoil

```
import { atom } from "recoil";

export const screenSize = atom({
  key: "screenSize", // unique ID (in context of atoms/selectors)
  default: "desktop", // default value (aka initial value)
});
```

3.3.4 Směrování (routing)

Směrování, neboli routing, je v kontextu Reactu proces, ve kterém je uživatel směrován na různé stránky na základě své akce nebo požadavku. Využití nachází převážně u jednostránkových webových aplikací (SPA).[41] Obecně funguje směrování tak, že je definována cesta ve tvaru */path*, pomocí které může uživatel přistoupit na příslušnou stránku v rámci aplikace. Konkrétní příklad je k dispozici v útržku kódu3.10, kde je definována cesta ke stránce pro vytvoření nového turnaje *add_tournament*. Pro přístup k dané stránce je tedy třeba zadat základní URL aplikace + danou cestu. V tomto případě je kompletní URL *https://www.turnajky.cz/add_tournament*.

Výpis kódu 3.10 Příklad definice routingu

```
<Route path="/add\_tournament" element={<AddTournament />} />
```

3.3.5 Hlavní tabulka s turnaji

Nejdůležitější komponentou celé aplikace z pohledu uživatele, je tabulka s turnaji. Pro její implementaci byla použita knihovna *react-data-table-component*. Naplnění daty proběhlo bez větších problémů, tabulka nabízí i možnost rozbalení řádku, tj. zobrazení dalších informací spojených s daným záznamem. Jak tabulka vypadá, je možné vidět v přiložených kopiích obrazovky v sekci2.4.

3.3.5.1 Komplikace

Problémy nastaly při implementaci vizuální stránky tabulky. Pro změnu barvy pozadí neexistuje v knihovně žádný příkaz ani transparentní název třídy. Název třídy celé tabulky se navíc mění při různých situacích (tabulka s daty, tabulka bez dat, data se načítají). Není tak možné nastavit barvu pozadí jednotně pro všechny situace. Lišící se názvy třídy musely být vyhledány ručně ve vývojovém prostředí prohlížeče a barva pozadí nastavena pro každou situaci zvlášť.

3.3.6 Kalendář

Pro implementaci kalendáře byla využita knihovna *react-big-calendar*. Postup byl analogický s vytvářením tabulky, kdy plnění daty proběhlo bez problémů a komplikace nastaly znovu u tvorby vizuální stránky kalendáře. Problém byl opět ve složitém dohledávání názvů jednotlivých komponent pro jejich stylizaci.

Knihovna *react-big-calendar* na rozdíl od *react-data-table-component* nenabízí rozbalení položky pro více informací. Pro tento účel slouží v tomto případě tzv. modal — okno, které se zobrazí v popředí stránky. Modal poskytuje knihovna *react-responsive-modal*.

3.3.7 Formuláře

Nedílnou součástí celé aplikace jsou i formuláře používané jak při vytváření a úpravě turnaje, tak při přihlašování nebo registraci. Použita je knihovna *react-hook-form*, která práci s nimi značně zjednodušuje.

3.3.8 Mobilní zobrazení

Jak je uvedeno v zadání, aplikace musí být responzivní a dobře se zobrazovat i v mobilních zařízeních. K tomu slouží knihovna *react-responsive*. Práce využívá metodu *useMediaQuery()*, která umí vrátit typ zařízení, na kterém se aplikace právě zobrazuje.

Výpis kódu 3.11 Příklad použití funkce useMediaQuery

```
const deviceType = useMediaQuery({ query: "(max-width:␣1224px)" })
? "mobile" : "desktop"
```

3.4 Nasazení

Jedním z hlavních důvodů vzniku této aplikace bylo umožnit hráčům plážového volejbalu přístup k portálu, kde najdou všechny turnaje na jednom místě. Proto bylo třeba vyřešit nasazení aplikace a tím ji zpřístupnit veřejnosti. Výsledný produkt je dostupný na adrese turnajky.cz.

3.4.1 Klient a Vercel

Nasazení klienta proběhlo vcelku jednoduše díky Vercelu. Vercel je cloudová platforma optimalizovaná pro rychlé nasazování moderních webových aplikací. Podporuje statické i dynamické stránky, využívá globální CDN (Content Delivery Network) pro rychlé načítání obsahu a nabízí automatizované nástroje pro efektivní vývoj, testování a nasazování. Díky serverless architektuře umožňuje vytvářet a nasazovat funkce bez správy serverů. Je navíc kompatibilní s různými programovacími jazyky a frameworky, usnadňuje týmovou spolupráci a podporuje open source komponenty. Poskytuje bezplatné i placené plány, přičemž se hodí pro vývojáře vytvářející webové aplikace postavené na moderních technologiích.[42]

Díky propojení s nástrojem GitHub je nasazení aplikace ve Vercelu opravdu jednoduché. Při vytváření projektu stačí povolit oprávnění pro přístup ke GitHub repozitáři a o vše ostatní už se postará samotná platforma. Kdykoliv dojde ke změně v příslušné větvi gitu (většinou se jedná o main), spustí se ve Vercelu automaticky nový build a změny jsou během chvíle promítnuty do produkce. Základní verze, kterou používá tato práce, je navíc zcela zdarma.

Vercel mimo jiné nabízí i možnost přidání vlastní domény. Proto byla přes Webglobe.cz zakoupena česká doména *turnajky.cz*, kterou aplikace v produkci využívá.

3.4.2 Server a Heroku

Nasazení serveru proběhlo pomocí Heroku, kontejnerově založené cloudové PaaS (Platform as a Service). Heroku slouží k nasazování, správě a škálování moderních aplikací.[43]

Nasazení probíhalo obdobně, jako u klienta, díky možnosti propojit službu s nástrojem GitHub. Do repozitáře stačilo přidat *Procfile*, což je soubor specifický pro Heroku, určující, jaké příkazy mají být zavolány pro spuštění aplikace. Pro správný běh aplikace muselo být pozměněno několik drobností včetně inicializace knihovny Selenium.

Heroku nabízí i add-ons, jako je *Heroku Scheduler* pro plánování pravidelných procesů nebo Papertrail pro lepší přehled logování.

Platforma je oproti Vercelu placená s cenou zhruba 0.010$ za hodinu, ale oproti jiným bezplatným platformám poskytuje jednoduché nasazení a mnoho přídavných modulů.

3.4.3 Možné problémy

Vzhledem k tomu, že obě části aplikace (klient a server) běží na zcela odlišných platformách, může vznikat několik potenciálních komplikací. Jednou z nich je horší latence. Ačkoliv se oba servery nacházejí na území Evropy, čas během něhož musí data cestovat z jednoho serveru na druhý, nemusí být zanedbatelný. Ačkoliv se tato komplikace nijak viditelně neprojevuje na odezvě aplikace, bylo by určitě vhodnějším řešením mít všechny zdroje na jednom místě.

Dalším problémem je již zmíněná cena. Ačkoliv se nejedná o vysokou sumu, aplikace v tuto chvíli negeneruje žádný zisk, a proto jsou jakékoliv náklady na její provoz nežádoucí. S tím je spojené je i riziko, že se bude cena služby postupem času zvyšovat.

3.5 Dokumentace

Jako dokumentace kódu slouží komentáře, které jsou součástí celé aplikace a poté soubor README, kde jsou popsány postupy, jak instalovat potřebné závislosti a lokálně spustit aplikaci. Celý projekt je k dispozici na platformě GitHub, která sloužila po celý čas jako verzovací nástroj pro tuto práci.

- Frontend — Odkaz na GitHub
- Backend — Odkaz na GitHub

3.6 Struktura složek

```
root ..... Hlavní složka
├── tournament_info_frontend ..... Frontend aplikace
│   ├── node_modules ..... Všechny instalované moduly
│   ├── public ..... Všechna média
│   │   ├── index.html ..... Hlavní HTML soubor
│   │   ├── resource_logos ..... Loga areálu a organizátorů
│   │   └── všechny další obrázky použité ve frontendu
│   ├── src ..... Zdrojové soubory
│   │   ├── components ..... React komponenty
│   │   ├── config ..... Konfigurační soubory
│   │   ├── icons ..... Používané ikony
│   │   ├── pages ..... React komponenty tvořící celé stránky
│   │   ├── state ..... Soubory pro správu stavu aplikace
│   │   ├── styles ..... Soubory kaskádových stylů(.css)
│   │   ├── index.js ..... Základní zdrojový soubor celé aplikace - obsahuje App.js
│   │   ├── index.css ..... Styl souboru index.js
│   │   ├── App.js ..... Zdrojový soubor se směrováním všech stránek
│   │   ├── App.css ..... Styl souboru App.js
│   │   ├── repotWebVitals.js ..... Automaticky generovaný soubor
│   │   ├── setupTests.js ..... Automaticky generovaný soubor
│   │   └── variables.css ..... Globální proměnné pro všechny soubory (.css)
│   ├── package.json ..... Balíčky Reactu
│   ├── package-lock.json ..... Balíčky Reactu
│   └── README.md ..... Obecné instrukce pro spuštění
└── tournament_info_backend ..... Backend aplikace
    ├── logging ..... Logy z běhu serveru
    │   └── main.log ..... Hlavní soubor s logy
    ├── migrations ..... Migrace databáze
    ├── app.py ..... Hlavní zdrojový soubor serveru
    ├── config.py ..... Konfigurace Flasku
    ├── CzechMonths.py ..... Třída pro převod na české měsíce
    ├── Logger.py ..... Třída pro logování
    ├── models.py ..... Třídy pro databázové modely
    ├── requirements.txt ..... Požadavky pro spuštění serveru
    ├── Tournaments.py ..... Třída pro správu turnajů
    ├── update.py ..... Script pro periodické získávání dat
    └── Procfile ..... Konfigurační soubor pro nasazení aplikace
```

Kapitola 4
Testování

Testování by mělo předcházet nasazení každé webové aplikace, aby došlo k odhalení chyb a možných problémů před uvedením do provozu.

V softwarovém inženýrství mohou být použity různé typy a techniky testování ke splnění konkrétních požadavků. V této kapitole proběhne nejprve seznámení s výsledky uživatelského testování a poté popis průběhu manuálního testování serverové části.

4.1 Uživatelské testování (Usability testing)

Uživatelské testování využívá skupinu reprezentativních uživatelů k interakci s webovou aplikací. Má za úkol zjistit, zda je používání aplikace intuitivní a poskytuje dobrý uživatelský zážitek. Tento proces testování pomáhá odhalit problémy, které by mohly zůstat bez povšimnutí reálnými uživateli. Úkolem je vytvořit produkt, který pomáhá dosáhnout cílů a řeší problémy uživatele s dobrým uživatelským zážitkem.

4.1.1 Scénáře

Pro lepší přehlednost bylo testování rozděleno do několika scénářů. Uživatel u každého scénáře obdrží pouze obecné instrukce, bez detailního popisu. Příklad instrukce: Přesuňte se na stránku pro registraci. Už ale není řečeno, kde má hledat odkaz. To by mělo otestovat jednoduchost a intuitivnost celé aplikace. Každý uživatel tedy obdržel dokument ve formátu *docx* s krátkým úvodem do problematiky a scénáři níže. U každého scénáře byl testovací subjekt vyzván k přidání komentáře, zda bylo vše v pořádku, nebo zda narazil na nějaké problémy.

4.1.1.1 Registrace, přihlášení a žádost o přidělení práv organizátora

Tento scénář se týká uživatelů, kteří budou chtít spravovat turnaje v aplikaci. Měl by otestovat, že registrace a žádost o přidělení práv jsou jednoduché a intuitivní.

1. Chcete se registrovat jako organizátor. Přesuňte se na stránku pro registraci.
2. Vyplňte osobní údaje (zvolte reálný email) a přejděte na stránku s přihlášením.
3. Přihlaste se pomocí údajů, kterými jste se registrovali.
4. Najděte tlačítko pro vyžádání organizátorských práv a zažádejte o ně.
5. Odhlaste se a chovejte se, jako byste zapomněli své stávající heslo. Pokuste se získat nové.

6. Po získání nového hesla si heslo změňte na vlastní, odhlaste se a následně se přihlaste pomocí nového hesla.

4.1.1.2 Vytvoření, úprava a smazání turnaje

1. Chcete vytvořit nový turnaj. Ke správě turnajů jsou zapotřebí organizátorská práva.
2. Přesuňte se na stránku pro přihlášení a přihlaste se jako organizátor s přihlašovacími údaji:
   - **email:** testuser@email.cz
   - **heslo:** TestUserOrganizer2023
3. Vytvořte nový turnaj. Jako jméno turnaje zvolte ideálně cokoliv kromě „Testovací turnaj" a jako město zvolte „Praha". Ostatní údaje o turnaji mohou být libovolné. Při nedostatku fantazie lze do textových vstupů vyplnit např. řetězec „test".
4. Po přidání turnaje zkontrolujte, že se turnaj objevil jak mezi vašimi turnaji, tak v přehledu všech turnajů na hlavní straně.
5. Upravte město konání turnaje na Brno a zkontrolujte, že se informace změnila v přehledu turnajů.
6. Pokud vše proběhlo v pořádku, turnaj odstraňte a zkontrolujte, že již není v přehledu turnajů.

4.1.1.3 Přihlášení a odhlášení z turnaje

1. Chcete se přihlásit na turnaj. Přihlašte se jako hráč přihlašovacími údaji:
2. - **email:** testuserplayer@email.cz
   - **heslo:** TestUserPlayer2023
3. Přihlaste se na „Mezigalaktický turnaj", který se koná 24. 12. 2026. Jméno spoluhráče může být libovolné.
4. Zkontrolujte, že se turnaj objevil ve vašem profilu.
5. Odhlaste se z turnaje a zkontrolujte, že již nejste mezi přihlášenými týmy.

4.1.1.4 Vygenerování linku pro import kalendáře

1. Chcete si importovat turnaje do vlastního kalendáře.
2. Vyfiltrujte si turnaje tak, aby se zobrazovaly pouze ty v kategorii mixy a úrovně open.
3. Nechte si vygenerovat URL pro import kalendáře s vybranými filtry.

4.1.2 Uživatelé

Bc. Adam Pončák

- student FEL ČVUT
- s aplikací se již v minulosti setkal
- hraje šestkový volejbal

Bc. Erich Winkler

- student FIT ČVUT
- s aplikací se již v minulosti setkal
- hraje plážový volejbal

Ing. Filip Hladej

- absolvent FIT ČVUT
- nikdy dříve s aplikací nepracoval
- nehraje volejbal

Bc. Petra Mihálová

- studentka FD ČVUT
- s aplikací se již v minulosti setkala
- hraje plážový volejbal

Ing. Kristýna Pancová

- absolventka Policejní akademie České republiky v Praze
- aplikaci aktivně využívá z pozice běžného uživatele (ne organizátora)
- hraje plážový volejbal

4.1.3 Výsledky testování

Výsledky testování byly rozděleny do tří kategorií: chyby v aplikaci, neočekávané chování a další připomínky. Ke každé připomínce byl přidán komentář s vysvětlením a aktuálním stavem.

4.1.3.1 Chyby v aplikaci (bugy)

- Je možné vytvořit turnaj, který se odehrál v minulosti.
  - Tato možnost není vnímána jako zásadní nedostatek, jelikož dochází k pravidelnému mazání starých turnajů. Turnaj tedy není na stránce dostupný nikdy déle než 60 minut.
- Při vytváření turnaje je nutné zadat odkaz na turnaj s *https://*, jinak přesměrování nefunguje.
  - Opraveno. Už funguje libovolný validní odkaz.
- Tlačítko pro *Požádejte o roli organizátora* se resetuje s novým načtením stránky, což může způsobit nechtěné opakování žádostí.
  - Uživatel je v momentě odeslání žádosti obeznámen, že byla úspěšně odeslána a její opakované odeslání ničemu nevadí. Zamezení opakovanému zobrazování žádosti by vyžadovalo netriviální řešení v podobě nové role nebo atributu v záznamu uživatele. Nejedná se o zásadní nedostatek, ale je v plánu jeho oprava v navazující práci.
- Při změně hesla stránka nabízí možnost *Zpět na přihlášení*, což může vést k omylnému odhlášení.

- Možnost byla změněna na odkaz zpět na profil uživatele.

- Občasné chyby při přihlašování s hláškou, že je nutné zadat email nebo heslo, i když jsou již zadané.
  - Problém nastával v situacích, kdy byly hodnoty předvyplněny prohlížečem. Hláška se již neobjevuje.

- Občas nefunguje tlačítko *Přihlásit se* po zadání emailu a hesla (neúčinnost), funguje až po obnovení stránky.
  - Zatím se nepodařilo najít příčinu, ale problém bude opraven v navazující práci.

- V některých případech nefunguje více filtrů najednou.
  - Více filtrů nefungovalo v případech, kdy nebyl nalezen žádný výsledek, protože aplikace správně nekontrolovala návratovou hodnotu. Filtry jsou již plně funkční.

4.1.3.2 (Ne)očekávané chování

- Po odhlášení z turnaje zůstává turnaj v seznamu přihlášených turnajů až do obnovení stránky.
  - Jedná se o záměrné chování pro jednoduchou možnost se znovu přihlásit s jiným spoluhráčem, jelikož aplikace nenabízí přímo možnost změnit spoluhráče.

- Po vytvoření se turnaj objeví v profilu až po obnovení stránky.
  - Tento nedostatek bude opraven v navazující práci.

- Tlačítko *Enter* nefunguje pro potvrzení formuláře.
  - Problém souvisel s kontrolou vyplnění vstupu a je již vyřešen. Tlačítko *Enter* tedy už funguje pro potvrzení.

- V každém formuláři (pro přihlášení, registraci, zapomenuté heslo) je třeba překlikávat mezi hráčem a organizátorem. Nepřehlednost registrace: Při přepnutí na organizátora při přihlášení a následně kliknutí na registraci se otevře okno pro přihlášení jako hráč a musím znovu překliknout.
  - Pro vyřešení tohoto chování by bylo nutné zcela přepracovat strukturu komponent pro přihlašování. Jako dočasné řešení slouží přidaná informace o zvolené možnosti v tlačítku pro potvrzení (místo *Přihlásit* je *Přihlásit jako organizátor*).

- Při úspěšné registraci není uživatel rovnou přesměrován na přihlášení, ale musí kliknout na *Zpět na přihlášení.*
  - Toto chování je záměrné, protože se na dané stránce objeví kolonka s potvrzením o úspěšné registraci nebo naopak chybová hláška. Změna chování bude zvážena v navazující práci.

- Zadávání jména a příjmení spoluhráče do dvou vstupních polí je nepřehledné. Lepší by bylo zadávat celé jméno najednou.
  - Aktuální způsob zadání považuje autor za přehledný. Změna bude zvážena v navazující práci.

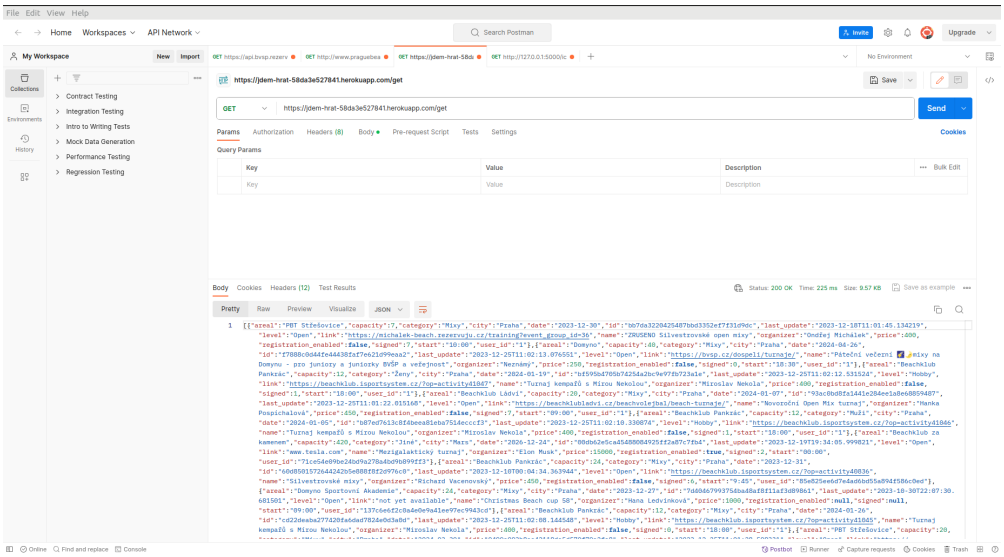
- Pro registraci nepřijde potvrzení na e-mail, což by mohlo umožnit použít libovolný e-mail.
  - Tento nedostatek bude opraven v rámci navazující práce.

4.1.3.3 Další připomínky

- Americké formátování data při vytváření turnaje.
  - Knihovna, kterou autor využívá pro implementaci formulářů, bohužel neposkytuje jinou možnost, ale je dostupný ruční výběr data z kalendáře, který je dostatečně přehledný.
- Formát času turnaje a ceny startovného při vytváření není jednoznačný.
  - Tento nedostatek byl vyřešen přidáním placeholderu v podobě požadovaného formátu.
- Role *basic* nemá výstižný název.
  - Tento nedostatek je autorovi znám a bude vyřešen přejmenováním obou organizátorských rolí.
- Při generování nového hesla by mohla být přidána možnost zadání vlastního hesla ihned.
  - V tuto chvíli je nutné se přihlásit pomocí automaticky vygenerovaného hesla, které bylo zasláno na email, a následně je možné heslo změnit. Možnost zadání vlastního hesla hned při jeho generování bude zvážena v navazující implementaci.
- Navigace k vytvořeným turnajům může být přehlednější.
  - K navigaci k vytvořeným turnajům, se uživatel dostane otevřením menu a volby možnosti *Profil*. Tento postup považuje autor za dostatečně jednoduchý a není v plánu ho měnit.
- Bylo by vhodné přidat tlačítko s odkazem na hlavní stranu.
  - K přesměrování na hlavní stranu slouží logo v levé části navigačního panelu.

4.2 Manuální testování

Na straně serveru proběhlo pouze manuální testování. Funkce, které se volají přes REST API ze strany klienta, byly jednotlivě testovány pomocí aplikace Postman4.1, platformou pro API vývojáře.



Obrázek 4.1 Příklad volání metody get v aplikaci Postman

Metody pro získání dat byly testovány nejprve samostatně v příkazové řádce a poté jako celek součástí skriptu *update.py*. Systém logování poskytl jednoduché odhalení případných problémů.

Kapitola 5
Závěr

Cílem práce byl návrh, implementace a otestování webové aplikace sloužící pro vyhledávání nadcházejících turnajů v plážovém volejbale. Teoretická část měla za úkol vybrat vhodné zdroje a způsoby získání informací, čehož se týkal i rozbor problematiky web scrapingu. Cílem praktické části bylo aplikaci navrhnout a implementovat. To zahrnovalo návrh komunikace mezi backendem a webovou aplikací a implementaci všech požadovaných funkcionalit. Těmi byly automatizované získávání informací, ukládání do databáze a manuální vytváření turnajů. Důraz byl kladen i na uživatelsky přívětivé prostředí a optimalizované zobrazení pro mobilní zařízení. Závěrečným úkolem bylo aplikaci otestovat a zdokumentovat.

Všechny cíle, stanovené na začátku práce, byly splněny a aplikace nabízí i mnoho dalších funkcionalit nad rámec zadání.

V analytické části práce nejprve proběhlo seznámení s metodami pro získávání dat o turnajích. Následně byla podrobněji rozebrána problematika web scrapingu a API. Nakonec proběhlo porovnání populárních nástrojů a technologií pro vývoj backendu i frontentu, včetně databází, vhodných pro účely webových aplikací.

V návrhu jsou rozebrány jednotlivé části aplikace (klient, server, databáze) včetně způsobu komunikace mezi nimi pomocí REST API a SQLAlchemy. Zbylá část analýzy se věnuje specifikaci požadavků a případům užití, ze kterých vychází samotná implementace. Ta podrobně rozebírá server i klienta, včetně použitých významných knihoven a metod. Součástí implementace je i nasazení obou částí aplikace.

Uživatelské testování ukázalo na některé nedostatky v podobě chyb v implementaci a neočekávaného chování. Některé z problémů byly již opraveny, zbylé z nich budou opraveny v navazující práci. Kromě některých výtek mělo ale testování převážně kladné ohlasy. Na základě informací z nástroje Analytics od společnosti Google aplikaci v tuto chvíli (konec roku 2023) aktivně využívá již zhruba 200 uživatelů a jejich počet pomalu roste.

Práce nabízí i mnoho potenciálních rozšíření a úprav. Aplikace může v budoucnu zobrazovat data z širšího okruhu, než je Praha a blízké okolí, s finálním pokrytím celé České republiky. Nabízí se i ukládání dat o výsledcích z turnajů a následné vytváření statistik a žebříčků jednotlivých hráčů.

Pro verzování byl používán nástroj GitHub, kde se nachází celá práce včetně zdrojového kódu a dokumentace.

Bibliografie

1. ZHAO, Bo. Web Scraping. 2017. Dostupné z DOI: 10.1007/978-3-319-32001-4_483-1.
2. TYSON, Matthew. What is an API? Application programming interfaces explained. 2022. Dostupné také z: https://www.infoworld.com/article/3269878/what-is-an-api-application-programming-interfaces-explained.html.
3. DILMEGANI, Cem. Web Crawler: What It Is, How It Works & Applications in 2023. 2023. Dostupné také z: https://research.aimultiple.com/web-crawler/.
4. NEZNÁMÝ, Autor. Web Scraping vs API: Best Way to Get Data in 2024. 2023. Dostupné také z: https://www.zenrows.com/blog/web-scraping-vs-api#differences-and-similarities.
5. PERSSON, Emil. Evaluating tools and techniques for web scraping. 2019. Dostupné také z: https://www.diva-portal.org/smash/get/diva2:1415998/FULLTEXT01.pdf.
6. KHDER, Moaiad Ahmad. Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application. 2021, roč. 13, č. 3. ISSN 2710-1274. Dostupné z DOI: 10.15849/IJASCA.211128.11.
7. HEATH, Anthony. Web Scraping 101: Tools, Techniques and Best Practices. 2023. Dostupné také z: https://medium.com/geekculture/web-scraping-101-tools-techniques-and-best-practices-417e377fbeaf.
8. SAKETH, Kowtha. What is the Document Object Model? [B.r.]. Dostupné také z: https://dev.to/sakethkowtha/about-dom-document-object-model-in-html-4bji.
9. JONATHAN ROBIE, Texcel Research. About DOM (Document object model) in HTML. 2021. Dostupné také z: https://www.w3.org/TR/REC-DOM-Level-1/introduction.html.
10. ROBOT, Scraping. Top 5 Scraping Techniques (And Best Practices) In 2021. 2021. Dostupné také z: https://scrapingrobot.com/blog/scraping-technique/.
11. TONY56024. Mastering XPath for Web Scraping: A Step-by-Step Tutorial. 2023. Dostupné také z: https://www.blog.datahut.co/post/xpath-for-web-scraping-step-by-step-tutorial.
12. DILMEGANI, Cem. Is Web Scraping Legal? Ethical Web Scraping Guide in 2023. 2023. Dostupné také z: https://research.aimultiple.com/web-scraping-ethics/.
13. DENSMORE, James. Ethics in Web Scraping. 2017. Dostupné také z: https://towardsdatascience.com/ethics-in-web-scraping-b96b18136f01.
14. TILLISON, Mark. Robots.txt for eCommerce SEO: A Complete Guide. 2023. Dostupné také z: https://tillison.co.uk/blog/robots-txt-for-ecommerce/.

15. NEZNÁMÝ, Autor. Beautiful Soup (HTML parser). 2023. Dostupné také z: https://en.wikipedia.org/wiki/Beautiful_Soup_(HTML_parser).
16. DHANASHREE. Web Scraping with Selenium: The complete guide. 2023. Dostupné také z: https://nanonets.com/blog/web-scraping-with-selenium/.
17. PANDIAN, Shanthababu. Exploring Octoparse Web Scraping Tool for Data Preparations. 2022. Dostupné také z: https://www.analyticsvidhya.com/blog/2022/04/exploring-octoparse-web-scraping-tool-for-data-preparations/.
18. LUTKEVICH, Ben. API design and management. 2022. Dostupné také z: https://www.techtarget.com/searchapparchitecture/definition/application-program-interface-API.
19. HAT, Red. What is a REST API? 2020. Dostupné také z: https://www.redhat.com/en/topics/api/what-is-a-rest-api.
20. S.R.O., DAMI development. Framework. [B.r.]. Dostupné také z: https://www.damidev.com/slovnik/framework.
21. MATHUR, Apreksha. What is a Framework? 2023. Dostupné také z: https://www.geeksforgeeks.org/what-is-a-framework/.
22. CHAWRE, Huzefa. Top 10 Backend Frameworks in 2024. 2023. Dostupné také z: https://www.turing.com/resources/backend-frameworks.
23. DAS, Sourojit. Top 10 Python Web Development Frameworks in 2023. 2023. Dostupné také z: https://www.browserstack.com/guide/top-python-web-development-frameworks.
24. DESAI, Jemin. 5 Best Frontend Frameworks for Web Development in 2023. 2023. Dostupné také z: https://positiwise.com/blog/best-front-end-frameworks.
25. LE, Vy. Database Web Applications: 7 Best Databases for Web Applications. 2023. Dostupné také z: https://www.orientsoftware.com/blog/database-web-applications/.
26. ROUSE, Margaret. Flat File. [B.r.]. Dostupné také z: https://www.techopedia.com/definition/25956/flat-file.
27. OKTA. Authentication vs. Authorization. 2023. Dostupné také z: https://www.okta.com/identity-101/authentication-vs-authorization/.
28. OKTA, Auth0 by. JSON Web Tokens. [B.r.]. Dostupné také z: https://auth0.com/docs/secure/tokens/json-web-tokens.
29. PODDAR, Rishabh. What is a JWT? Understanding JSON Web Tokens. 2022. Dostupné také z: https://supertokens.com/blog/what-is-jwt.
30. GHATE, Shreya. Using Session Cookies Vs. JWT for Authentication. 2020. Dostupné také z: https://hackernoon.com/using-session-cookies-vs-jwt-for-authentication-sd2v3vci.
31. VOLLEYBOX. Volleybox. [B.r.]. Dostupné také z: https://volleybox.net/.
32. AMERICA, AVP. AVP. [B.r.]. Dostupné také z: https://www.volleyamerica.com/VA-Beach-Volleyball-Schedule.aspx.
33. KIWI.COM. Kiwi. [B.r.]. Dostupné také z: https://www.kiwi.com/cz.
34. BRUSH, Kate. MoSCoW method. 2023. Dostupné také z: https://www.techtarget.com/searchsoftwarequality/definition/MoSCoW-method.
35. CHITRASINGLA2001. Functional vs Non Functional Requirements. 2023. Dostupné také z: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/.
36. MOHAMED HIROLE Chris Mills, Masahiro Fujimoto. SPA (Single-page application). 2023. Dostupné také z: https://developer.mozilla.org/en-US/docs/Glossary/SPA.

37. PLATFORMS, Meta. Hooks at a Glance. [B.r.]. Dostupné také z: https://legacy.reactjs.org/docs/hooks-overview.html.
38. PLATFORMS, Meta. useState. [B.r.]. Dostupné také z: https://react.dev/reference/react/useState.
39. PLATFORMS, Meta. useEffect. [B.r.]. Dostupné také z: https://react.dev/reference/react/useEffect.
40. PLATFORMS, Meta. Components and Props. [B.r.]. Dostupné také z: https://legacy.reactjs.org/docs/components-and-props.html#gatsby-focus-wrapper.
41. JAISWAL, Sonoo. React Router. [B.r.]. Dostupné také z: https://www.javatpoint.com/react-router.
42. GAGE, Justin. What does Vercel do? 2020. Dostupné také z: https://vercel.com/blog/what-is-vercel.
43. COMPANY, Salesforce. Heroku. [B.r.]. Dostupné také z: https://www.heroku.com/what.