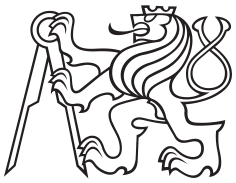
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Aplikace pro tenisové kluby – frontendová část

Vývoj, implementace a testování

Ondřej Mareš

Školitel: RNDr. Žára Ondřej
Obor: Softwarové inženýrství a technologie
Květen 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Mareš** Jméno: **Ondřej** Osobní číslo: **474776**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Aplikace pro tenisové kluby - frontendová část

Název bakalářské práce anglicky:

Tennis application for clubs - frontend component

Pokyny pro vypracování:

Nastudujte metodologii návrhu a tvorby moderních webových aplikací, dělených na logicky oddělenou frontendovou a backendovou část. Tyto komponenty jsou spolu provázány jen pomocí jasně definovaného HTTP rozhraní (REST, RPC, GraphQL a podobně).
Následně navrhněte a vytvořte frontendovou část webové aplikace, poskytující jednoduchý a přehledný způsob správy tenisových klubů přinášející benefity i samotným hráčům. Vycházejte primárně z dat, nabízených oficiálním webem 'cztenis.cz'. Analyzujte nabízená data, uvažte možnost jejich strojového zpracování a vytvořte nástroj, který bude tato data konzumovat.
Pro potřeby komunikace se serverovou stranou aplikace definujte a otestujte vhodné rozhraní, které pokryje všechny uživatelské scénáře.
Hlavní rysy aplikace budou:
1) Registrace tenisových klubů
2) Registrace hráčů
3) Správa kurtů
4) Harmonogramy a výsledky soutěží
Aplikaci navrhněte s ohledem na použití jak na desktopových, tak mobilních zařízeních. Výslednou aplikaci otestujte z uživatelského (kvalitativní testování) i bezpečnostního hlediska. Popište, jaké jsou technologické limity produktu, tj. je-li závislý na přítomnosti konkrétních klientských API a jak.

Seznam doporučené literatury:

Axel Rauschmayer: Deep JavaScript, 2019,
https://exploringjs.com/deep-js/index.html
Addy Osmani: Learning JavaScript Design Patterns, O'Reilly Media 2017,
ISBN 9781449331818
https://developer.mozilla.org/en-US/docs/Web

Jméno a pracoviště vedoucí(ho) bakalářské práce:

RNDr. Ondřej Žára, Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **10.02.2020** Termín odevzdání bakalářské práce: **22.05.2020**

Platnost zadání bakalářské práce: **30.09.2021**

RNDr. Ondřej Žára podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.	
Datum převzetí zadání	Podpis studenta

Poděkování

Chtěl bych poděkovat vedoucímu mé bakalářské práce RNDr. Ondřeji Žárovi za věcné připomínky, cenné rady a vstřícnost při konzultacích.

Dále bych chtěl poděkovat kolegovi Ottovi Vodvářkovi za spolupráci při vývoji webové aplikace.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

V Praze 19. května 2020

Abstrakt

Práce se zabývá metodologií návrhu a tvorby moderní frontendové části webové aplikace, která je rozdělená na backendovou a frontendovou část. Tyto dvě komponenty jsou spolu provázány jen pomocí jasně definovaného HTTP REST a websocketového rozhraní.

Zabývá se návrhem a vytvořením klientské části webové aplikace poskytující jednoduchý a přehledný způsob správy tenisových klubů, která přináší benefity i samotným hráčům.

Aplikace využívá primárně data veřejně přístupná na oficiálním webu Českého tenisového svazu (cztenis.cz). Práce analyzuje nabízená data a zabývá se jejich strojovým zpracováním, následně tyto poznatky aplikuje při vývoji nástroje, který tato data konzumuje.

Klíčová slova: webová aplikace, frontend, informatika, software, JavaScript, React, tenis, UX design, UI design, REST

Školitel: RNDr. Žára Ondřej

Abstract

This work deals with the design methodology and creation of the front-end of a modern web application which is divided into back-end and front-end. These two components are interconnected through a clearly defined HTTP REST and a web socket interface.

It deals with the design and creation of the client components of the web application, providing a simple and clear way to manage tennis clubs, also for the benefit of the players.

The applications uses data publicly available on the official website of the Czech Tennis Association (cztenis.cz). This work analyzes available data and searches for possibilities for its machine processing, then this knowledge is used to develop a tool that consumes this data.

Keywords: web application, front-end, informatics, software, JavaScript, React, tennis, UX design, UI design, REST

Obsah

Obrázky

Tabulky

Seznam použitých zkratek

ČTS	Český tenisový svaz
SPA	Single-page aplikace
UX	User experience
UI	User interface
DOM	Document Object Model
SEO	Search Engine Optimization
HTML	Hypertext Markup Language
CSS	Kaskádové styly
JS	JavaScript
HTTP	Hypertext Transfer Protocol
PWA	Progressive web application
XSS	Cross-site scripting
API	Application Programming Interface
REST	Representational State Transfer
CSRF	Cross-Site Request Forger

Kapitola 1

Úvod

Tento dokument se zabývá problematikou tvorby moderní webové aplikace se zaměřením na frontendovou část (tj. část, kterou vidí návštěvník webu). Poznatky se poté aplikují na návrh, implementaci a testování aplikace TK21.

Jedná se o aplikaci sloužící ke správě a zveřejňování informací tenisových klubů v České republice, využívající primárně data veřejně přístupná na oficiálním webu Českého tenisového svazu (cztenis.cz). Data jsou analyzována a následně aplikací využita. Hlavními rysy jsou:

- Vyhledávání informací o tenisových klubech
- Registrace rekreačního tenisového klubu
- Registrace závodního tenisového klubu
- Registrace hráčů
- Párování hráčů s profilem na tenisovém svazu
- Správa kurtů
- Harmonogramy a výsledky soutěží

Aplikace TK21 (`https://www.tk21.cz`) je navržena s ohledem na použití jak na desktopových, tak mobilních zařízeních. Dokument se zabývá kvalitativním testováním a popisuje technologické limity produktu a bezpečnost.

Backendovou část (tj. serverovou část) aplikace a bakalářskou práci na ni vytváří kolega Otto Vodvářka.

Část I

Problematika

1.1 Současná situace

1.1.1 Tenisové kluby a moderní technologie

Tenis je v České republice velmi oblíbeným sportem, kdy se tenisový klub nachází prakticky v každém menším městě. Jen v severočeském regionu se nachází 117 tenisových klubů, které jsou zapsány u Českého tenisového svazu[1].

Většina tenisových klubů v České republice nevyužívá dostatečně moderní technologie, které by jim usnadnily práci s provozem klubu. Běžně mívají pouze webové stránky, případně facebookovou stránku. Příkladem může být Tenisová akademie Březno u Loun[2], Tenisový klub Most[3], Tenisový klub Písnice[4] nebo OÁZA Říčany[5].

Pokud některý tenisový klub má online rezervační systém, jedná se většinou o velké kluby, které jsou navíc situovány ve velkých městech a okolí (např. Tenis Cibulka[6], Hamr tenis[7], I. ČLTK Praha[8]). Rezervační systémy používané kluby jsou většinou složité, nepřehledné a nejsou plně zaměřené na tenis.

1.1.2 Český tenisový svaz

Oficiálním webem ČTS (Českého tenisového svazu) jsou stránky `http://cztenis.cz`. Tento web je důležitý především pro kluby, které pořádají celostátní turnaje nebo se účástní soutěží smíšených družstev, a hráče, kteří se účastní turnajů a soutěží smíšených družstev, případně jejich blízkých.

Na stránkách svazu se nacházejí informace o klubech, turnajích, soutěžích smíšených družstev, hráčských žebříčcích a jednotlivých hráčích. Jednotlivá data jsou uložená v tabulkách a nedá se v nich vyhledávat, vyjma vyhledávání hráčů v horním panelu, které není moc chytré a vyhledává jména pouze ve tvaru *příjmení jméno*, místo obvyklejšího *jméno příjmení* (nejlépe, kdyby na pořadí nezáleželo). U většího počtu dat je realizováno stránkování, které na přehlednosti dat také nepřidává.

Velkým problémem je ale propojenost dat, kdy data na stránkách nejsou propojená, např. pokud vyhledáte nějakého hráče, kliknutím na jeho jméno se dostanete na jeho stránku, kde je napsán jeho kmenový klub, který ale není propojený se stránkou s informacemi o klubu. Provázány jsou pouze stránky o hráčích, na které vedou odkazy z žebříčků, klubových soutěží a turnajů.

Data ohledně turnajů jsou zapsána pomocí tabulky platící na danou sezónu pro celou republiku, ve které není možné filtrovat, tzn. že pokud máte nějaký oblíbený klub a chtěli byste se přihlásit na všechny turnaje, které pořádá, musíte je hledat v tabulce mezi všemi turnaji v dané sezóně.

[1] Severočeský region - Ústecký a Liberecký kraj [3]
[2] `http://www.tenisbrezno.cz/tenisova-akademie.html`
[3] `http://tkmost.cz`
[4] `http://www.tkpisnice.cz`
[5] `http://www.oazaricany.cz/sport-a-relax/tenis-ricany/`
[6] `http://www.teniscibulka.cz`
[7] `http://www.hamrsport.cz/cs/tenis/`
[8] `https://cltk.cz/cs/`

1.2 Konkurence

Konkurencí k našemu systému by mohly být různé rezervační systémy pro sportovní kluby, jichž existuje nepřeberné množství, například ProKlub[9] (dále Jdeme na to[10]), který nenabízí rozhraní pro členy klubu nebo řešení přímo na míru tenisovým klubům, tzn. že v aplikaci je spousta funkcí navíc, které mohou působit rušivě a dělají celou aplikaci složitější z uživatelského hlediska, to je ve výsledku jeden z důvodů, který odrazuje kluby od používání rezervačních systémů.

Další konkurencí budou pravděpodobně online katalogy sportovních klubů, například Sport v okolí[11], který opět kombinuje více sportů dohromady. Tyto konkurenční systémy data získávají přímo od samotných klubů, což znamená, že neukazují aktuální informace (turnaje, soutěže družstev apod.), nenašel jsem žádný konkurenční systém, který by četl a nějak zpracovával data z webu ČTS.

1.3 Motivace a vize

Já i kolega Otto Vodvářka hrajeme závodně tenis od dětství, díky tomu jsme dobře seznámeni se stavem tenisu a moderních technologií v České republice. Napadlo nás vytvořit systém přímo pro tenis, kde hráči najdou potřebné informace o klubech a rychle uvidí, jaké turnaje a soutěže družstev se v daném klubu a kdy konají a na jaké soutěže jsou přihlášeni.

Další neméně důležitou funkcionalitou je jednoduchý rezervační systém, který bude obsahovat jen funkcionality, které jsou potřeba pro rezervaci tenisových kurtů, klademe si za cíl, aby byl rezervační systém tak jednoduchý, aby jej mohly využívat i ty nejmenší kluby.

Vizí aplikace je také možnost zjistit informace o klubech v místech, kde je více klubů v okolí. Pokud by všechny kluby v okolí používaly náš systém, hráč uvidí přehledně zaplněnost kurtů a aktuální informace v jednotlivých klubech v okolí, na základě toho si může zvolit, kam půjde hrát, a nemusí volat na několik telefonních čísel, aby zjistil zaplněnost kurtů v klubech.

[9] `http://proklub.cz`
[10] `https://jdemenato.cz`
[11] `https://www.sportvokoli.cz`

Část II

Analýza a návrh

Kapitola 2
Sběr požadavků

2.1 Poptávané vlastnosti

Na základě problematiky jsme si sestavili vlastnosti, které budeme požadovat, jako základní pilíře nově budované aplikace (viz tabulka 2.1).

Vlastnost	Popis
Registrace klubu	Možnost, aby přihlášený uživatel mohl zaregistrovat klub na základě dat z ČTS, nebo úplně nový klub.
Informace o klubech	Vyhledávání klubů a zobrazování dat zjištěných u tenisového svazu a zveřejněných administrátorem klubu. U klubů nových (mimo svaz) zobrazovat pouze informace zveřejněné administrátorem klubu.
Registrace uživatele	Možnost registrace uživatele. Možnost provázat přihlášeného uživatele se stránkou na ČTS.
Členi klubu	Přidávání členů do klubu, kterým budou umožněna vyšší práva, oproti uživatelům, kteří členi nejsou (např. rezervace kurtu).
Rezervace kurtu	Možnost zarezervovat si kurt, přičemž administrátor klubu bude moci nastavit, zda kurt může rezervovat kdokoliv nebo člen klubu.
Přehled uživatele	Uživatelská stránka, která bude zobrazovat nadcházející rezervace uživatele, aktuality z klubů, kde je členem a pokud bude hráčem z ČTS, tak turnaje a soutěže smíšených družstev.

Tabulka 2.1: Poptávané vlastnosti

2.2 Data z Českého tenisového svazu

Většina dat zobrazovaných na stránkách ČTS[1] je zveřejňována pomocí tabulek. Cílem této části dokumentu je ukázat, kde a v jakém formátu se jednotlivá data nacházejí a jaké mají limity.

2.2.1 Kluby

Každý klub má svoji stránku, na které je zobrazeno číslo klubu, název klubu, adresa, kontaktní e-mail a někdy telefon.

Získaná data o klubech bychom chtěli využívat především k zobrazování údajů o klubech, aniž bychom museli kontaktovat kluby v republice a o data žádat.

Klub, který je registrovaný s pomocí těchto získaných dat, se stává klubem ověřeným, u kterého může aplikace garantovat správnost údajů.

Administrátor data, která jsou získaná z ČTS, nemůže upravovat přímo v aplikaci, ale přes ČTS.

2.2.2 Hráči

Seznam hráčů se na webu ČTS dá zjistit dvěma způsoby. Jeden způsob je hráče přímo vyhledat pomocí vyhledávacího pole, kdy musíte napsat vyhledávací text ve tvaru *příjmení jméno* (místo běžnějšího *jméno příjmení*). Druhý způsob jsou seznamy hráčů v žebříčcích pro každou věkovou kategorii.

Na stránce hráče je k dispozici, jméno, příjmení, klubová příslušnost, datum narození a jeho výsledky na turnajích a v soutěžích družstev.

U hráčských dat budeme využívat pouze k ověření, zda se jedná o daného hráče, datum narození, jméno a klubovou příslušnost. Dále chceme zaznamenávat výsledky hráče, které mu poté můžeme zobrazit.

2.2.3 Celostátní turnaje a soutěže smíšených družstev

Celostátní turnaje jsou dělené na základě věkových kategorií (Dospělí, Dorost, Starší žactvo, Mladší žactvo), stejně tak i soutěže smíšených družstev.

Turnajová stránka obsahuje název turnaje, datum konání, klub a adresu, kde se pořádá. Pokud se již dá na turnaj přihlásit, je zde také zobrazený seznam hráčů, kteří jsou aktuálně na turnaj přihlášeni, tento seznam je prolinkován se stránkami hráčů (viz 2.2.2).

U soutěží smíšených družstev jsou k dispozici ligy a divize jednotlivých oblastních svazů. Na stránkách těchto soutěží se nacházejí informace o pořadí jednotlivých týmu, datech, kdy spolu zápasí, a výsledcích předešlých zápasů.

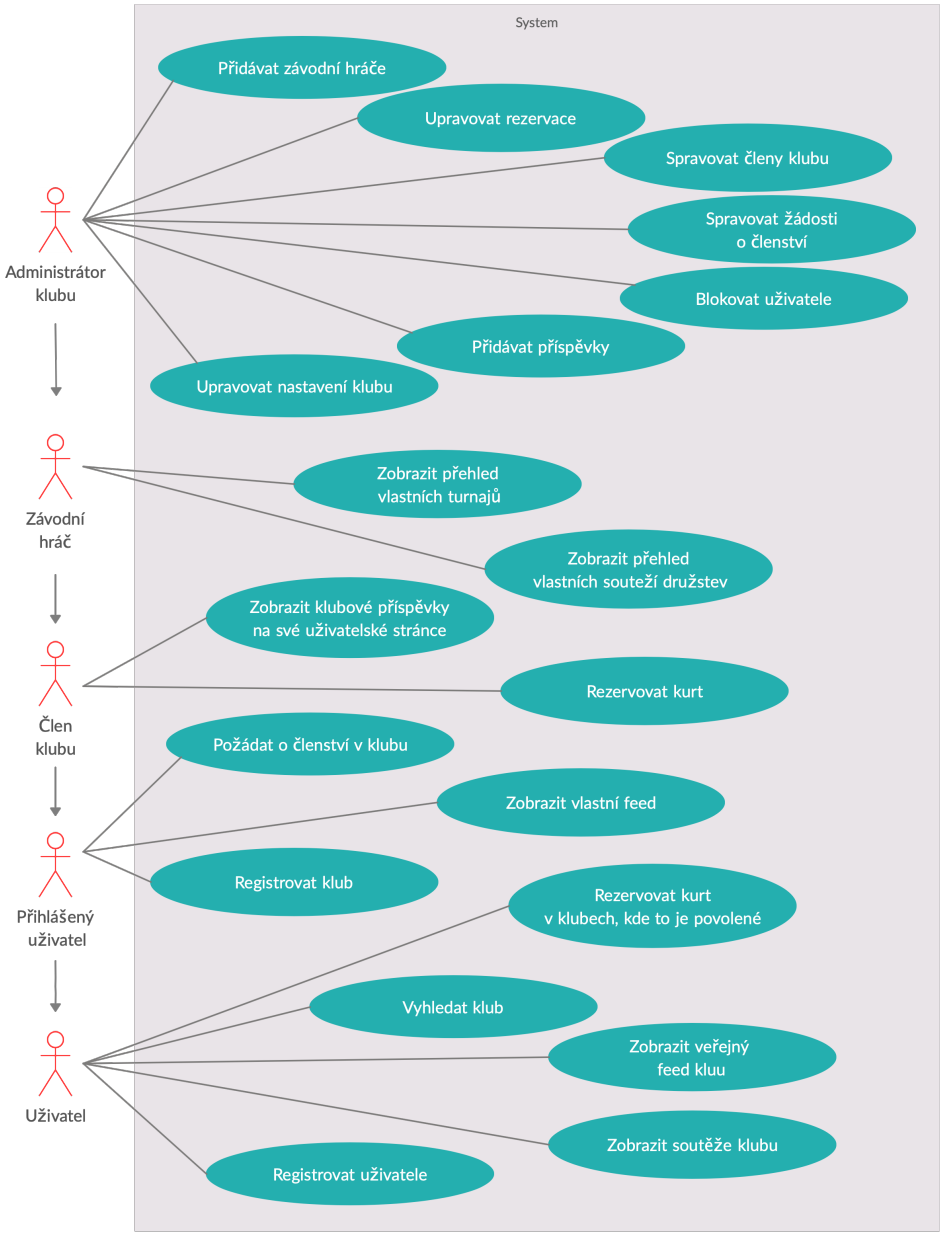
Získaná data o přihlášených hráčích chceme využít k notifikování uživatele o jeho budoucích turnajích a soutěžích družstev a jeho výsledcích. Turnaje v jednotlivých klubech nám zase poslouží k zobrazení dalších informacích o klubu.

[1] `http://cztenis.cz`

2.3 Případy užití (use cases)

Pro zmapování funkcionalit systému jsme použili use case diagram (česky diagram případů užití), který zobrazuje chování systému z uživatelského pohledu. Diagram vypovídá o tom, co má systém umět, ale neukazuje, jak to bude dělat. Skládá se z případů užití (use case), aktérů a vztahů mezi nimi. Aktér je role, nebo systém (např. uživatel, administrátor apod.), který dané funkcionality (případy užití) bude smět v našem systému vykonávat. [5]

Rozhodli jsme se implementovat prozatím pět rolí *uživatel*, *přihlášený uživatel*, *člen klubu*, *závodní hráč* a *administrátor klubu*. V budoucnu bychom mohli implementovat ještě *trenéra* a *zaměstnance*, avšak prozatím pro implementaci základních funkcionalit postačí zmíněných pět rolí (viz obrázek 2.1).



Obrázek 2.1: Diagram případů užití

Kapitola 3
Řešení vlastností aplikace

3.1 Registrace nového uživatele

Při registraci budeme po uživateli požadovat jméno a příjmení z důvodů rezervací kurtů, budoucího propojení se stránkou na ČTS a celkového identifikování uživatele v klubu. K přihlašování do aplikace potřebujeme heslo a jedinečný identifikátor uživatele, kterým jsme zvolili e-mail, který zároveň potřebujeme k ověření účtu a notifikování uživatele, např. při rezervaci kurtu apod. Místo e-mailu, jakožto identifikátoru, jsme uvažovali také nad použitím jedinečného uživatelského jména, které jsme nakonec nezvolili, jelikož se ukázalo, že by bylo v aplikaci nadbytečné a e-mail bychom stejně potřebovali.

K zprovoznění uživatelského účtu je také potřeba e-mail ověřit. Po registraci přijde uživateli na jím vyplněný e-mail odkaz, který bude v parametru obsahovat klíč, díky čemuž se po kliknutí na takový odkaz odešle požadavek na server, kterým se ověří konkrétní e-mailová adresa ke konkrétnímu účtu. Tímto způsobem chceme zabránit možnosti operovat v aplikaci na cizí e-mail nebo nekontrolovatelně vznikajícím uživatelským účtům bez ověřeného komunikačního prostředku.

3.2 Registrace klubu

Aby tenisový klub nemohl vytvářet kdokoliv, kdo přijde na webovou stránku (nebo robot), zaregistrování klubu jsme podmínili uživatelským účtem. Klub může tedy zaregistrovat pouze přihlášený uživatel v aplikaci, který ověřil svou e-mailovou adresu.

3.2.1 Rekreační

První věcí, co musíme vyřešit, je registrace rekreačního klubu, tzn. klubu, který není zapsán u ČTS. Od takového klubu potřebujeme při registraci název klubu a adresu klubu (obec, PSČ, ulice a číslo popisné), u které jsme původně plánovali, že ji budeme používat jako identifikátor klubu, ale později se ukázalo, že na webu ČTS, mají kluby unikátní názvy, tak jsme naše požadavky upravili a jako identifikátor používáme název klubu.

Z důvodu využívání unikátního názvu nemůže být zaregistrován jako rekreační takový klub, který je na webu ČTS, alespoň ne pod stejným názvem.

3.2.2 Závodní

Jelikož je registrace klubu podmíněna přihlášením uživatele, rozhodli jsme se, že klub, který je zapsán u ČTS, zaregistruje pouze účet, jehož e-mail bude shodný s kontaktním e-mailem klubu uvedeným na webu ČTS. Je-li kontaktních e-mailů víc, klub může zaregistrovat každý účet, jehož e-mail je uveden u klubu na ČTS.

Díky tomu, že e-mail je pro dokončení registrace uživatele potřeba ověřit (viz 3.1), nemůže se stát, že by klub z ČTS zaregistrovala neoprávněná osoba.

3.3 Informace o klubech

U klubů budeme zobrazovat pro všechny uživatele (i nepřihlášeného) data, která jsou volně dostupná na stránkách ČTS, jedná-li se o klub, který je zveřejněný na webu ČTS. U takového klubu budeme zobrazovat turnaje, které budou seřazené podle toho, zda se již hrály nebo ne. Chceme také zobrazit aktuální soutěže družstev – týmy, jejich skóre a pořadí mezi ostatními.

Nebude-li klub zaregistrovaný v aplikaci, budeme jej zobrazovat i přes tuto skutečnost, avšak uživatele upozorníme na skutečnost, že klub není v aplikaci zaregistrovaný, a proto nemůže využívat všechny funkcionality klubu, který zaregistrovaný je.

U registrovaného klubu bude moci administrátor klubu přidávat příspěvky, které jsou plánované rozdílně od příspěvků například na Facebooku. Tyto příspěvky by měly být dlouhodobého charakteru, např. informace o tenisovém soustředění, která by jinak v příspěvcích na Facebooku zapadla za jinými příspěvky. Rovněž bude administrátor moci nastavit otevírací dobu, dny, kdy je otevírací doba změněná, kurty v klubu, data začátku a konce sezón (zimní a letní), kontakt a popis.

3.4 Provázání klubů se závodními hráči

3.4.1 Zamýšlené varianty

Řešení provázání závodních hráčů s kluby bylo složitější a bylo to potřeba více promyslet, jelikož hráči na stránkách ČTS nemají žádný identifikátor, z důvodu ochrany osobních údajů. Proto jsme si sepsali dvě varianty (viz Varianta A a Varianta B) a jejich podvarianty možných řešení a následně se rozhodli, proč a pro kterou se přikloníme.

Obě varianty počítají s tím, že status závodního hráče vznikne uživateli aplikace jedině tím způsobem, že jej ověří administrátor klubu, ve kterém by měl být závodním hráčem. Administrátor klubu je totiž ověřený uživatel

pomocí e-mailové adresy, tudíž on je jediný v aplikaci, který může nějakého uživatele provázat s hráčskou stránkou na webu ČTS.

Varianta A

Varianta A spočívá v tom, že se do databáze aplikace nebudou ukládat všichni hráči, kteří jsou na webu ČTS, ale pouze hráči, kteří budou ověřeni administrátorem klubu. Ověření tímto způsobem by probíhalo přes WebSocket protokol[1], kdy by administrátor odeslal jméno a příjmení, na základě čehož by mu přišel seznam hráčů nalezených na webu ČTS (pomocí vyhledávání hráčů, viz 2.2.2), kde by následně administrátor vybral hráče, kterého by spároval s e-mailovou adresou uživatele aplikace. Následně se bude každou noc aktualizovat do databáze aplikace pouze data spárovaných hráčů z webu ČTS.

Administrátor by měl možnost vyplnit údaje přes formulář, včetně e-mailové adresy, načež by mu přišel z backendové strany seznam odpovídajících hráčů – výběrem hráče by poté přišel na řadu samotný hráč, který by pomocí e-mailu měl potvrdit, zda to je skutečně on. Existoval-li by již v aplikaci uživatel s takovým e-mailem, tak po potvrzení by se uživatel stal členem klubu (pokud jím již nebyl) a rovnou by byl i závodním hráčem. Jestli uživatelský účet s takovým e-mailem v aplikaci není, tak se účet vytvoří, přičemž v e-mailu, který by uživatel obdržel, by byl odkaz na nastavení hesla, čímž by se zároveň ověřila i e-mailová adresa, po ověření bude uživatel rovněž členem a závodním hráčem klubu.

Uvažovali jsme i nad možností párování se stránkou na ČTS samotným hráčům/uživatelům, kterým by se websocketové spojení otevřelo při registraci uživatele (viz 3.1) a backendová strana by jim odeslala seznam hráčů, kde by vybrali, kým z webu ČTS jsou. V takovém případě by o této skutečnosti byl informován administrátor klubu a ověřil by identitu takového uživatele.

Varianta B

Tato varianta počítá s tím, že backendová strana aplikace by každou noc aktualizovala databázi podle hráčských žebříčků (viz 2.2.2), tzn. že budou uloženi v databázi i hráči, kteří nemají spárovaný účet v aplikaci.

Zde by byla možnost ukládat do databáze jen hráče, kteří mají klubovou příslušnost shodnou s nějakým klubem, který je v naší aplikaci zaregistrovaný, nebo by aplikace ukládala všechny hráče, které podle žebříčků najde.

3.4.2 Závěr

Po diskuzi s kolegou Ottou Vodvářkou jsme se rozhodli přiklonit k variantě **A** s tím, že prozatím nebudeme implementovat možnost párování samotným hráčům/uživatelům při jejich registraci, ale necháme to prozatím v gesci administrátora s tím, že v budoucnu můžeme tuto funkcionalitu doimplementovat.

[1] WebSockets – protokol umožňující perzistentní spojení mezi klientem a serverem, takže si data můžou vyměňovat kdykoliv [2]

Rozhodnutí se opíralo především o to, že seznamy hráčských žebříčků, z kterých by probíhalo ukládání dat při variantách **B** nejsou kompletní a obsahují záznamy pouze hráčů, kteří v předchozí sezóně získali nějaké body (tj. hráli soutěže a byli úspěšní).

3.5 Rezervace kurtů

Rozhodli jsme se, že necháme vybrat administrátora klubu, zda umožní rezervace komukoliv (i nepřihlášenému uživateli), pouze členům klubu, nebo uživatelům, kteří mají účet v aplikaci, ať si každý klub rozhodne, zda chce nést riziko zneužití rezervací náhodným uživatelem.

Při vytváření rezervace budeme požadovat e-mail, pro zaslání rekapitulace (případně odkazu na smazání rezervace pro nepřihlášené uživatele), jméno a příjmení, přičemž přihlášenému uživateli by se tyto údaje předvyplnily automaticky. Administrátor klubu musí mít také možnost vytvářet rezervace opakující se (např. kvůli tréninkům).

Musí zde být možnost nějaký kurt udělat nedostupným, např. z důvodu rekonstrukce. Některé kurty jsou zase dostupné pouze v letní/zimní sezóně a některé v obou sezónách, aplikace musí umožnit nastavení kurtu pro jednu nebo obě sezóny. Sezónu je třeba připravit dopředu (tréninky apod.), proto musí mít administrátor možnost vypnutí rezervací pro ostatní na určitou sezónu. Měla by taky existovat možnost nastavit maximální dobu rezervace na jeden den, aby se nestalo, že si jeden uživatel zarezervuje všechny kurty na celý den jen pro sebe.

Otevírací dobu budeme aplikovat na všechny kurty najednou, nepůjde nastavit jinou otevírací dobu pro různé kurty, protože nastavování otevírací doby pro jednotlivé kurty by bylo příliš složité a my chceme, aby byla aplikace jednoduchá a dostupná i menším klubům. Omezení otevírací doby nějakého kurtu se dá vyřešit zarezervováním nadbytečných hodin, ale aplikace musí umožňovat změnu otevírací doby na nějaký konkrétní den, případně úplné uzavření klubu (např. z důvodu rekonstrukce).

Samozřejmostí musí být možnost správy vlastní rezervace, u nepřihlášeného uživatele alespoň její smazání. Administrátor by zase měl mít, na rozdíl od ostatních, možnost vidět jména a e-maily všech rezervací v klubu s možností jejich úprav.

3.6 Přehled uživatele

Uživatelský přehled by měl sloužit ke sdělení rychlých informací uživateli aplikace, a to přehled klubů, v nichž je členem a příspěvků z takovýchto klubů, seřazených podle data. Jedná-li se o uživatele propojeného se stránkou na webu ČTS, měl by ve svém přehledu vidět i následující turnaje, na které je přihlášený a aktuální stav soutěží družstev (pořadí jeho týmu, bodové ohodnocení týmu a následující utkání). V přehledu by také měly být zobrazeny následující rezervace, jako připomínka uživateli aplikace.

Kapitola 4

Nástroje

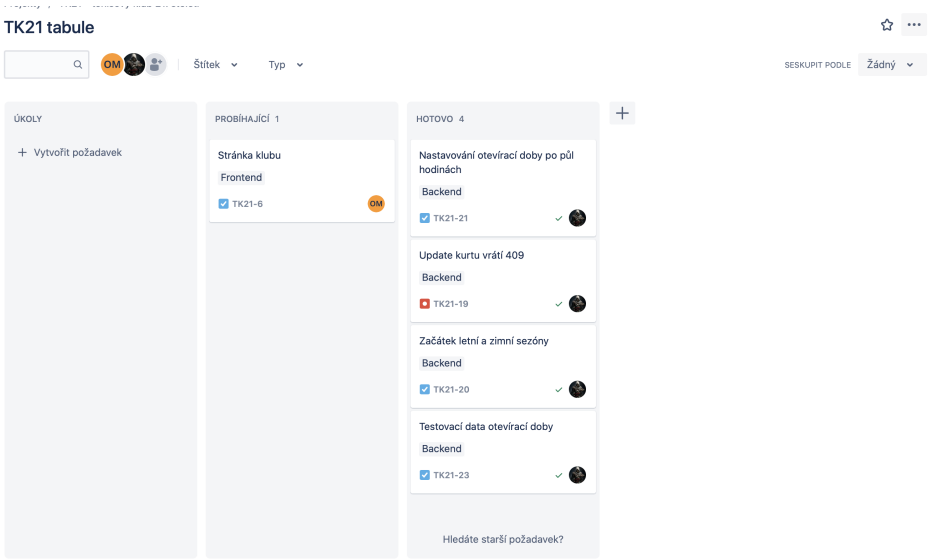
4.1 Komunikace při vývoji

Jelikož aplikaci vyvíjíme ve dvou, bylo nejprve potřeba určit si komunikační nástroje mezi mnou a kolegou.

Z počátku vývoje jsme používali ke komunikaci obyčejné chatovací aplikace jako Facebook Messenger a iMessage od Applu, mysleli jsme si totiž, že pro vývoj ve dvou to bude stačit.

Postupem času jsme zjistili, že se nám spousta úkolů jak pro backend, tak frontend nahromadí a poté v chatu zapadnou někde v předchozí konverzaci, proto jsme přešli na využívání Jira Software[1], ve kterém jsme si zvolili způsob vývoje kaban, kdy máme tabuli se třemi sloupci - úkoly, probíhající a hotovo. Jeden z nás vždy vytvoří úkol a přiřadí si ho sobě nebo kolegovi, který po vykonání přesune do sloupce hotovo (viz obrázek 4.1).

[1]Od společnosti Atlassian – `https://www.atlassian.com/cs/software/jira`



Obrázek 4.1: Ukázka kaban projektu v Jira Software

4.2 Tvorba případů užití

Pro tvorbu případů užití (viz 2.1), jsme použili webový nástroj Creately[2], jedná se o poměrně nový nástroj, který nebyl na začátku našich studií příliš intuitivní, avšak nyní poslouží jako bezproblémový rychlý nástroj na tvorbu diagramu případů užití.

4.3 Tvorba UX designu

Na tvorbu UX[3] designu (viz 5) jsem použil nástroj Adobe Comp[4]. Nástroj je dostupný zdarma na operačních systémech iOS a ipadOS. Umožňuje rychle vytvářet UX a UI[5] design s pomocí elektronické tužky. Nicméně osvědčil se mi více při tvorbě UX designu.

[2] `https://creately.com/diagram-type/use-case`

[3] User experience – návrh aplikace tak, aby splnil požadavky, avšak nezahrnuje grafický design [6]

[4] `https://www.adobe.com/cz/products/comp.html`

[5] User interface – grafický design stránek aplikace včetně návrhu animací [6]

Kapitola 5

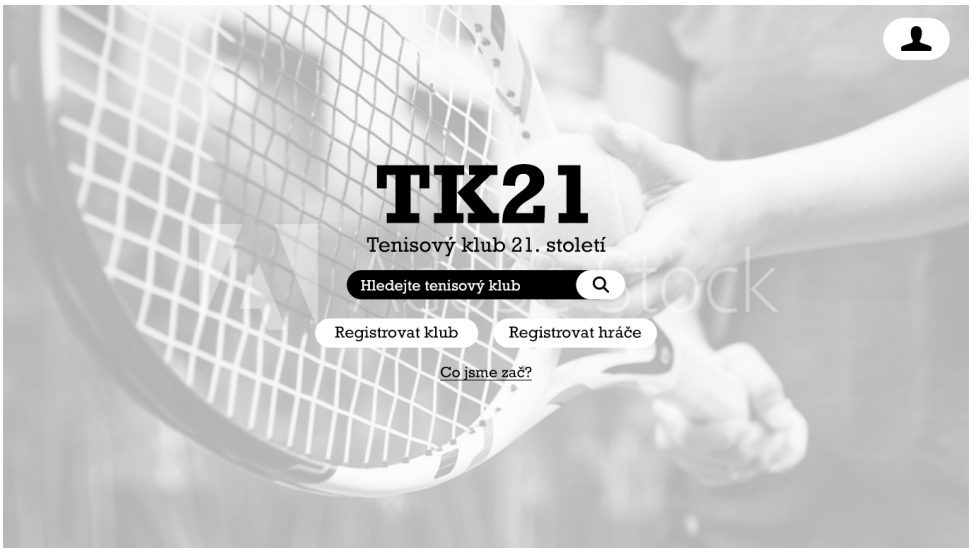
UX design

V této kapitole jsem si vytvořil seznam všech obrazovek (s přibližným obsahem, viz 5.1), které bude potřeba vytvořit a návrh rozložení pěti obrazovek, ačkoliv tyto návrhy obsahují některé grafické prvky, nejedná se o finální řešení. Jejich cílem je si přibližně ujasnit, co budou jednotlivé obrazovky obsahovat a jak by měla probíhat interakce aplikace s uživatelem, aby uživatel nenarazil na žádnou překážku a dojem z používání aplikace byl pro něj co nejlepší [6].

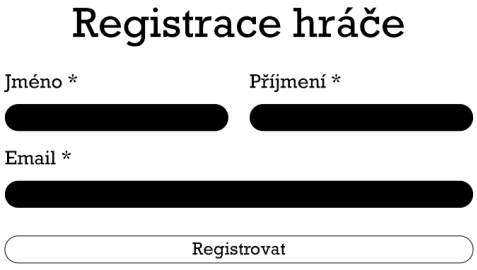
Příklad rozdílu mezi UX a UI designem dle [6]: *„Dejme tomu, že jste na klasickém e-shopu s nábytkem. To, co vidíte na stránce daného e-shopu - obrázky, texty, navigaci, barvy, rozložení prvků atd. - je uživatelské rozhraní, neboli UI. To, jak daný e-shop funguje - jakým způsobem se produkt vloží do košíku, jak se přesměrujete na platební bránu, jak se produkt vyhledává - je zkušenost uživatele na webu, neboli UX.“*

Název	Obsah
Úvodní strana (5.1)	Povídání o aplikaci, odkaz na registraci uživatele, vyhledávací pole klubů, možnost přihlásit se do aplikace.
Registrace uživatele (5.2)	Jméno, příjmení, e-mail, možná heslo, odkaz na stránku s přihlášením.
Registrace klubu (5.3)	Název klubu, obec, PSČ, ulice a č. p., možná údaje na kontaktní osobu – jméno, příjmení a e-mail.
Vyhledávání klubů (5.4)	Vyhledávací pole, seznam nalezených klubů s odkazem na klubové stránky.
Klubová stránka (5.5)	Kontakt na klub, odkaz na rezervace kurtů, aktuální příspěvky, otevírací doba, změny v otevírací době, klub z ČTS navíc – odkaz na stránku turnajů a soutěží družstev.
Rezervace kurtů	Časová osa se znázorněnými rezervacemi, pole pro změnu data, formulář pro vytvoření rezervace
Turnaje	Seznam turnajů, které se budou konat, anebo se konaly v klubu.
Soutěže družstev	Seznam týmů klubu, jejich věkových kategorií, pořadí, bodového hodnocení, soupeřů se soupisem jednotlivých utkání.
Uživatelský přehled	Místo úvodní strany pro přihlášeného – vyhledávací pole klubů, seznam členských klubů, aktuální příspěvky z členských klubů, následující rezervace, jedná-li se o závodní hráče, pak – následující turnaje a soutěže družstev.
Žádosti o členství	Seznam žádostí o členství v klubu, s možností uživatele přijmout, zamítnout nebo zablokovat.
Správa členů	Seznam členů klubu, s možností člena vyhodit, případně z něj udělat administrátora.
Blokace	Seznam blokovaných uživatelů, s možností odblokování.
Nastavení klubu	Formuláře pro nastavování klubových vlastností.
Nastavení uživatelského profilu	Formuláře s možností změny hesla, e-mailu, a pokud se nejedná o závodního hráče, tak – jména a příjmení.

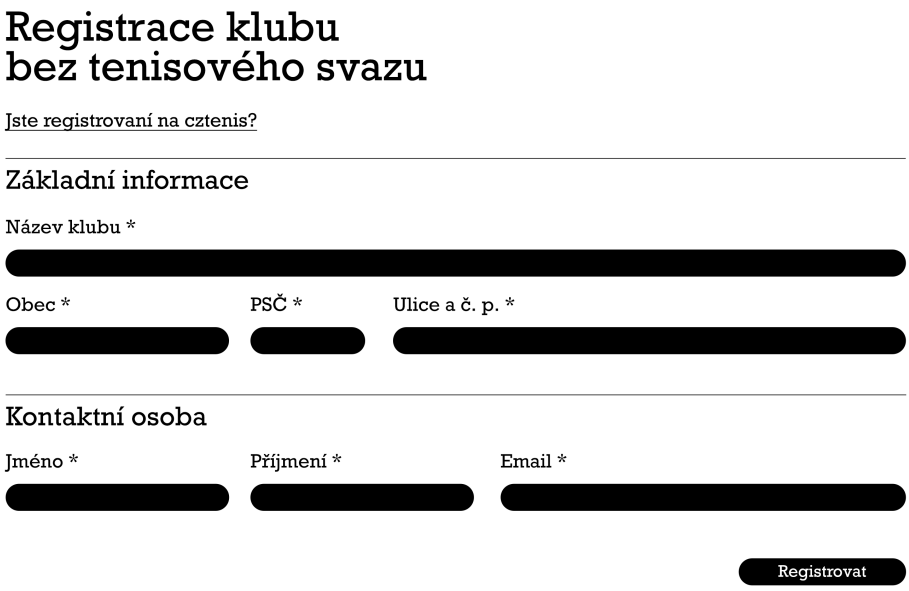
Tabulka 5.1: Obrazovky aplikace



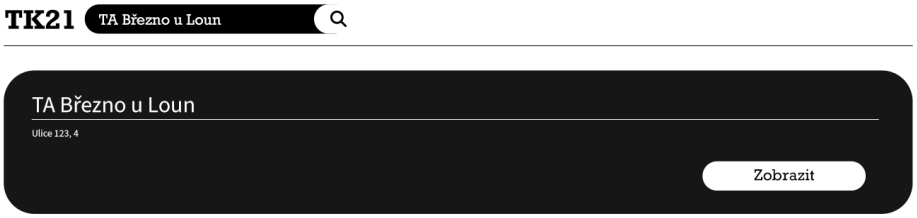
Obrázek 5.1: UX design: úvodní strana aplikace



Obrázek 5.2: UX design: registrace uživatele



Obrázek 5.3: UX design: registrace klubu



Obrázek 5.4: UX design: vyhledávání klubů

Obrázek 5.5: UX design: stránka klubu

Část III

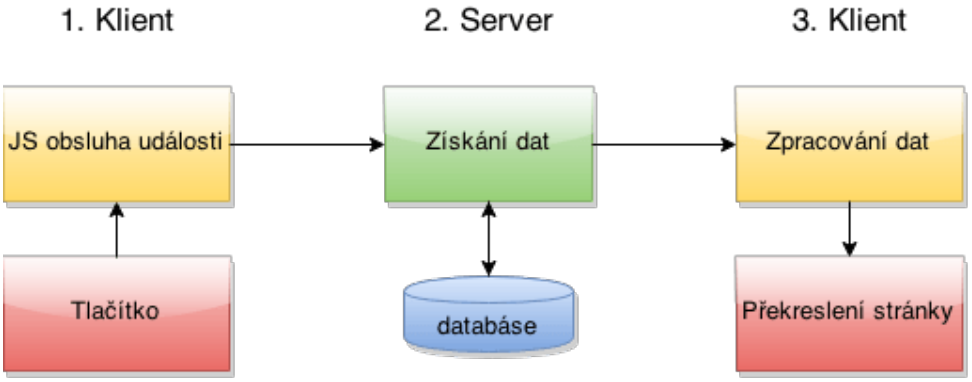
Implementace

Kapitola 6
Architektura

6.1 Běžný × single-page frontend

6.1.1 Co je single-page webová aplikace

„Single page application (někdy také one page application) je typ webové aplikace, která většinou používá server pouze jako zdroj a úložiště dat. Data jsou potom kompletně vykreslována JavaScriptem.“ [7]



Obrázek 6.1: Ukázka akce v single-page webové aplikaci (převzato z [7])

6.1.2 Výhody

Hlavní výhodou, oproti běžnému webu je, že u SPA (single-page aplikace) se stáhne celý HTML kód (a dodatečné soubory, například JavaScriptové a kaskádové styly) na začátku a při změně stránky se již nestahuje celý kód, včetně potřebných souborů, ale pouze se pomocí dotazu na server změní určitá část stránky. Kromě datové úspory nemusí prohlížeč překreslovat celou stránku, ale jen část, díky tomu je odezva webu výrazně rychlejší a uživatelský pocit při procházení webu je lepší. [7]

6.1.3 Nevýhody

SPA je přímo závislá na JavaScriptu, který ale může být u uživatele vypnutý, v nějakých případech to může být zásadní překážka, proč nevytvářet web jako SPA [7]. Podle [8], využívá celosvětově k 28. 4. 2020 JavaScript 95,43 % uživatelů, proto si nemyslím, že by to byl pro naši aplikaci zásadní problém.

Další nevýhodou může být i obvykle delší čas prvního načtení, kdy se musí stáhnout celá aplikace do prohlížeče uživatele [7], tento problém se dá částečně vyřešit, viz 7.4.1.

6.1.4 Závěr

Aplikaci jsem se rozhodl vytvářet jako SPA, jelikož se jedná o moderní způsob tvorby webových aplikací, kde vzhledem k povaze vyvíjené aplikace (především rezervace kurtů), je vítané, že se bude překreslovat jen část webu, a ne celá stránka při změnách, pro uživatele to je pohodlnější, i vzhledem k tomu, že JavaScript je celosvětově široce využívaný (viz 6.1.3).

6.2 Single-page nástroje

Existuje řada nástrojů na tvorbu SPA, které mají již vyřešené základní vlastnosti SPA, jako je vykreslování DOM[1] a struktura projektu. V této sekci popíši nejběžnější nástroje na tvorbu SPA a mé rozhodnutí.

6.2.1 Angular

Populární JavaScriptový framework vyvíjený Googlem. Je založen na komponentách a využívá dvoucestnou vazbu dat a Shadow DOM[2]. [9]

I přes velkou komunitu a dokumentaci jakou Angular má, je naučit se pracovat v něm velmi náročné a dokumentace je velmi matoucí, zejména pro začátečníky. [9]

6.2.2 Vue.js

JavaScriptový framework, který na rozdíl od Angularu (viz 6.2.1) a Reactu (viz 6.2.3) nemá podporu od nějaké velké společnosti. Vyvinul ho Evan You, který se předtím podílel na vývoji AngularuJS. Je založen na komponentách, stejně jako Angular a React. Vykreslování DOMu je velmi rychlé. Aplikace vytvořená ve Vue.js má významně menší velikost v porovnání s Angularem a Reactem. [9]

Nevýhodou tohoto frameworku je to, že je nejmladší ze zde uváděných, tzn. že vývojářská komunita oproti zbylým dvěma není tak rozsáhlá. Další

[1] Document Object Model – reprezentuje HTML dokument jako strom, kde každý uzel reperezentuje část dokumentu (např. element) [2]

[2] Shadow DOM – dovoluje skrytému DOM stromu připojit se k regulernímu DOM elementu – strom Shadow DOM začíná v shadow root, na který může být navázán jakýkoliv DOM element [2]

nevýhodou může být také to, že není vedený dobře známou velkou společností. [9]

6.2.3 React

Nejpopulárnější z úváděných nástrojů [10]. Oproti Angularu (viz 6.2.1) a Vue.js (viz 6.2.2) se jedná o JavaScriptovou knihovnu. Založen je na komponentách, využívající JSX syntaxi a vývoj je řízen Facebookem. Hlavní charakteristikou je virtualizace DOM, přičemž vazba dat je jednocestná, díky tomu je pro React charakteristický výjimečný výkon, v porovnání s Angularem je React na učení jednodušší. [9]

Jelikož se jedná o knihovnu, při tvorbě složitější aplikace je potřeba stáhnou další knihovny (např. React Router pro routování), z čehož vychází i další nevýhoda, že React není příliš SEO[3] přátelský. [9]

JSX syntaxe

JSX se odklání od klasického JavaScriptu, rozšiřuje syntaxi o kód podobající se HTML. Například [2]:

```
1 const heading = <h1>Mozilla Developer Network</h1>;
```

Tato konstanta nadpisu je JSX výraz. Stejně jako v HTML můžeme vytvářet stromovou strukturu, například [2]:

```
1 const header = (
2   <header>
3     <h1>TK21</h1>
4   </header>
5 );
```

Prohlížeč takovouto syntaxi nepřečte, musí se zkompilovat, předchozí kód se zkompiluje do následující podoby [2]:

```
1 const header = React.createElement("header", null,
2     React.createElement(
3         "h1",
4         null,
5         "TK21"
6     )
7 );
```

6.2.4 Závěr

Jednotlivé nástroje na tvorbu SPA, uváděné v předchozích sekcích, si jsou velmi podobné v tom, že se všechny zakládají na komponentách. Jelikož z žádným z nástrojů nemám zkušenost, v použití pro mě odpadá Angular, jelikož je nejsložitější na naučení, což by mohlo být velkým problémem při vývoji aplikace. Rozhodl jsem se zvolit React, a to především v porovnání

[3]Search Engine Optimization – optimalizace dělající weby viditelnější v internetových vyhledávačích [2]

s Vue.js kvůli velké vývojářské komunitě a podpoře dobře známé a velké společnosti.

6.3 Struktura a komponenty

Jak bylo zmíněno v 6.2.3, React je založen na komponentách, proto jsem si rozdělil strukturu aplikace na jednotlivé adresáře a podadresáře obsahující určitý typ komponent a souborů.

- authentication – obsahuje komponenty potřebné k přihlášení
- files – obsahuje externí soubory jako obrázky, písmo, dokumenty apod.
- forms – obsahuje komponenty jednotlivých formulářů
  - fields – obsahuje komponenty formulářových prvků (button, checkbox, input, select ...)
- pages – obsahuje komponenty stránek
  - clubPages – obsahuje komponenty stránek související se stránkou tenisového klubu
    - settings – obsahuje komponenty stránek sloužící pro nastavování klubových informací
  - header – obsahuje komponenty hlaviček používaných napříč celou aplikací
  - parts – obsahuje nejmenší komponenty, například nadpis (Heading.js), dialogové okno apod.
  - patterns – obsahuje vzory určitého rozložení stránky
  - reservationPages – komponenty sloužící pro rezervace kurtů
- props – obsahuje JS soubory nesoucí validační funkce, konstanty využívané napříč aplikací a pomocné funkce

Důležitá je znovupoužitelnost komponent, které se poskládají kompozičně do sebe a vznikne celá uživatelská stránka, například přihlašovací formulář může být v hlavičce, ale zároveň můžu stejnou komponentu použít i v dialogovém okně, samotný přihlašovací formulář se skládá z dalších komponent – textového pole, tlačítka, nadpisu apod.

6.4 Komunikace s backendem

6.4.1 Rozhraní

Existuje spousta rozhraní pro komunikaci mezi frontendovou a backendovou stranou. S kolegou jsme se rozhodli pro HTTP REST z jediného důvodu, máme s ním oba největší zkušenost. REST (Representational State Transfer) je architektura rozhraní pro distribuované systémy, která určuje, jakým způsobem se přistupuje k datům [12].

REST

„Rozhraní REST je použitelné pro jednotný a snadný přístup ke zdrojům (resources). Zdrojem mohou být data, stejně jako stavy aplikace (pokud je lze popsat konkrétními daty). Všechny zdroje mají vlastní identifikátor URI a REST definuje čtyři základní metody pro přístup k nim.“ [12]

Implementují se čtyři základní metody, které jsou známy pod zkratkou CRUD – Create (vytvoření dat), Retrieve (získání dat), Update (změna dat), Delete (smazání dat). Tyto metody jsou implementovány pomocí odpovídajících metod HTTP protokolu[4], v našem případě – POST (vytvoření), GET (získání), PUT (změna) a DELETE (smazání). [12]

GET. Provedu-li touto metodou požadavek na server na nějakou konkrétní URL adresu, očekávám, že v odpovědi dostanu data v JSON[5] formátu.

POST. Touto metodou zasílám nová data na server v JSON formátu.

PUT. Použiju-li tuto metodu, v těle požadavku zašlu data v JSON formátu, které si přeji změnit.

DELETE. Metoda, která na základě URL adresy smaže data.

Kód 6.1: Ukázka JSON notace

```
{
    "page": {
        "current":1,
        "last":1
    },
    "clubs": [
        {
            "id":855,
            "name":"TK Louny",
            "address": {
                "street":"Masarykovy sady 2790",
                "city":"Louny",
                "zip":"44001"
            },
            "registered":false,
            "scraped":true
        }
    ]
}
```

WebSockets

Jedná se o protokol, umožňující perzistentní spojení mezi serverem a klientem, takže si mohou vyměňovat data kdykoliv oběma směry [2].

[4] HyperText Transfer Protocol – základní síťový protokol umožňující zasílání hypermédií na webu [2]

[5] Javascript Object Notation – datový formát, lehce čitelný a zapisovatelný lidmi, lehce generovatelný a parsovatelný strojem, vycházející z JS objektů [2]

Rozhodli jsme se tento protokol využít v rezervacích, jelikož uživatel poté uvidí změny v rezervacích ihned, stránka s rezervacemi může být také zobrazená v klubovně a zobrazovat aktuální zaplnění kurtů. Další využití se nám hodilo při párování závodních hráčů (viz 3.4.1), především z důvodu udržení stavu při komunikaci. Data předáváme taktéž v JSON formátu.

6.4.2 Apiary

Webový nástroj pro tvorbu dokumentace a testování REST rozhraní.

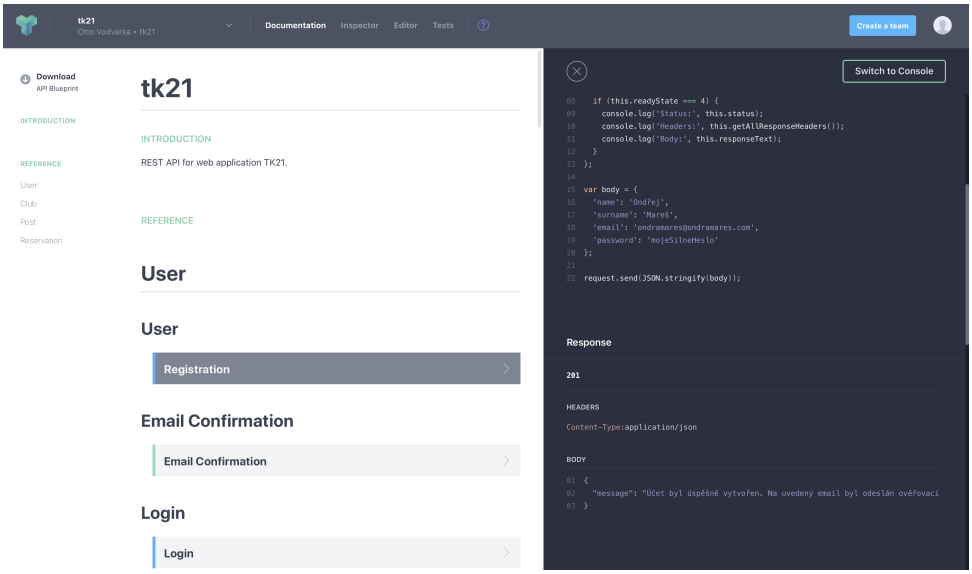
Dokumentaci je možno psát ve dvou typech značkovacích jazyků, jeden je Blueprint a druhý je Swagger. Zvolili jsme jazyk Blueprint, který nám přišel přehlednější a jednodušší než Swagger. Nástroj poskytuje přehlednou a interaktivní dokumentaci (viz 6.3), kterou se můžete proklikat k jednotlivým HTTP dotazům, u nichž vidíte, jakou HTTP metodou (GET, POST, PUT, DELETE) máte vznést požadavek na server, abyste dostali validní odpověď. Nástroj dokonce zobrazuje možnost vygenerování si kódu v určitém jazyce, který zdokumentovaný požadavek vykoná. Přehledně zobrazuje i hlavičky jak požadavku, tak odpovědi, včetně stavového kódu, který má klientská strana očekávat po validním, či nevalidním zpracování požadavku.

S psaním dokumentace v Blueprint (viz 6.2) jsem se setkal s několika problémy, které trochu ztěžovaly práci s jinak velmi užitečným nástrojem. Především při větším množství dotazů se kód stává velice nepřehledným, ocenil bych, kdyby byl rozdělen do více souborů. Psaní také není úplně ideální, pokud chcete vložit nějaký kus kódu jinam než na konec souboru, tak se vám ten kód již barevně neoznačí, což ještě více zhoršuje nepřehlednost. Po uzavření a otevření problém přetrvává, a dokonce neobarvenou syntaxi vidí i kolega, který soubor také upravuje. Ocenil bych také automatické ukládání, pokud něco napíšete a zapomenete kliknout na tlačítko *uložit*, vaše práce se neuloží a po uzavření prohlížeče o ni přijdete.

Apiary kromě dokumentace poskytuje také testovací rozhraní. Vygeneruje adresu, na kterou pokud se dotážete nějakým dotazem z dokumentace, tak vám odešle validní odpověď, pokud jsou hlavičky požadavku správné. Zde musím konstatovat, po zkušenosti, že to ne vždy funguje, jak bych předpokládal. Nepovedlo se mi zprovoznit, abych při určitém požadavku dostal jinou odpověď než tu, která je v dokumentaci uvedena jako první, tzn. že požadavky, které jsem na počátku implementace zkoušel přes Apiary, jsem musel vyzkoušet stejně znovu i přes hotovou serverovou stranu, což zbytečně ubírá čas.

```
# Group User

## User [/api/user]

### Registration [POST]

+ Request (application/json)

        {
            "name": "Ondřej",
            "surname": "Mareš",
            "email": "ondramares@ondramares.com",
            "password": "mojeSilneHeslo"
        }

+ Response 201 (application/json)

        {
            "message": "Účet byl úspěšně vytvořen. Na uvedený email byl odeslán ověřovací link"
        }
```

Obrázek 6.2: Ukázka zápisu REST dokumentace v Blueprint



Obrázek 6.3: Ukázka vygenerované dokumentace v Apiary

Kapitola 7
Technická řešení

7.1 Přihlašování a odhlašování

7.1.1 Princip

React zajišťuje skládání komponent do sebe a šíření parametrů skrz komponenty. Parametry do komponent se předávají vždy z vyšší komponenty do nižší (ve stromové struktuře), pokud bych takto chtěl předávat informaci o přihlášeném/odhlášeném uživateli, musel bych skoro do každé komponenty na každém místě v aplikaci tuto informaci předávat.

Globální data mohou být předávána pomocí React Context. Tato funkcionalita umožňuje předávání dat (proměnné, zda je uživatel přihlášen) a funkcí (přihlášení a odhlášení) napříč komponentami ve stromové struktuře bez manuálního předávání (předávání dat a funkcí každé komponentě zvlášť). Vzhledem k povaze Reactu se změna těchto kontextových proměnných projeví překreslením komponent.

Ověření přihlášeného uživatele má v gesci především backendová strana, která generuje token pro přihlášeného uživatele a klientovi jej odesílá jako HTTP-Only Cookie[1]. K obraně proti CSRF útoku ještě přidáváme CSRF-token, více v 7.5.2. Při inicializaci aplikace se dotáži serveru, zda je uživatel přihlášen, a získám odpověď i s parametry přihlášeného uživatele.

7.1.2 Observer

Při implementaci jednotlivých obrazovek (viz 5), se objevila nechtěná vlastnost v rezervacích a klubových příspěvcích, kdy pokud se přihlásíte jako administrátor a přejdete například na klubovou stránku, kde jsou klubové příspěvky, máte možnost příspěvky upravovat, pokud se odhlásíte, možnost úprav zmizí. Problém nastane, pokud se nyní přihlásíte s jiným účtem, který není administrátorský v onom klubu – z předchozího přihlášení uvidíte znovu možnost úprav příspěvků.

[1] Cookie – parametr, který je ukládán do prohlížeče uživatele, který jej následně odesílá v požadavcích na server – HTTP-Only příznak znamená, že ona Cookie není čitelná JavaScriptem [2]

Je to způsobeno tím, že při žádosti o příspěvky na backendovou stranu přijde v odpovědi, včetně seznamu příspěvků, také právo na úpravy. Administrátor klubu tedy přijde na stránku s příspěvky, backendová strana pošle, že může příspěvky upravovat. Poté se odhlásí, informace o úpravách v příspěvcích zůstává, ale jelikož je uživatel odhlášen, úpravy se nezobrazují. Dojde-li nyní k přihlášení neadministrátorského účtu, komponenty se na základě přihlášení překreslí, včetně příspěvků, v nich je ale stále informace, že uživatel příspěvky může upravovat.

Potřeboval jsem tedy najít způsob, který mi umožní při přihlášení/odhlášení provést znovu dotaz na příspěvky, případně navázat nové websocketové spojení u rezervací, kde se ověřuje identita na začátku spojení, jenže spojení by bylo trvalé i při odhlašování/přihlašování, tudíž je zde podobný problém.

Řešení jsem našel v podobě návrhového vzoru Observer. Jedná se o návrhový vzor, kde je objekt, který si drží seznam objektů, které upozorní při změně nějakého stavu, konkrétně v mém případě při změně stavu přihlášen/odhlášen [4]. Objekt držící si seznam objektů, jež musí upozornit, je u mě třída App, jedná se o kořenovou React komponentu. Pomocí React Context (viz 7.1.1) si předám do celé stromové struktury komponent funkci, která jako parametr přijímá další funkci, která má být zavolána, při změně přihlašovacího stavu (viz 7.1). Dojde-li k přihlášení nebo odhlášení, zavolám každou funkci ze seznam funkcí, které o tom mám upozornit (viz 7.3). Existuje zde také funkce (viz 7.2), která odebere funkci ze seznamu, to je potřeba předtím, než dojde k odmontování komponenty z obrazovky (viz 7.5).

Kód 7.1: Přidání funkce, která se bude upozorňovat na změnu stavu

```
addLoginObserver(fce) {
    const key = this.loginObserversSize;
    this.loginObserversSize++;

    this.loginObservers[key] = fce;
    return key;
}
```

Kód 7.2: Odebrání funkce, která se měla upozorňovat při změně stavu

```
removeLoginObserver(key) {
    delete this.loginObservers[key];
}
```

Kód 7.3: Upozornění funkcí na změnu stavu

```
notifyLoginObservers() {
    for (let fce of Object.values(this.loginObservers))
        fce();
}
```

Kód 7.4: Přihlášení funkce k upozorňování změn

```
componentDidMount() {
    this.loginObserverKey = this.context.addLoginObserver(
    this.loginChange);
}
```

Kód 7.5: Odhlášení funkce k upozorňování změn

```
componentWillUnmount() {
    this.context.removeLoginObserver(this.loginObserverKey);
}
```

7.1.3 Automatické odhlášení v nevhodnou chvíli

Jelikož to, zda je uživatel přihlášen, nebo ne, záleží na backendové straně, může se stát, že dojde k odhlášení v nevhodnou chvíli, například při psaní dlouhého klubového příspěvku administrátorem. V takové chvíli by u běžné webové aplikace došlo při odeslání příspěvku k odhlášení uživatele, načež by mohl přijít o rozpracovaný příspěvek.

Takovéto situaci jsem se rozhodl zabránit. V SPA se neaktualizuje celá stránka po odeslání příspěvku, ale pouze se vytváří samotný příspěvek, poskytuje mi to možnost se této situaci vyhnout. Přijde-li ze serveru odpověď 401 (přístup zamítnut), zobrazím přihlašovací formulář v dialogovém okně s možností přesměrování na hlavní stranu (resp. odhlášení). Uživatel má tedy rozepsaný dlouhý příspěvek, klikne na tlačítko pro odeslání příspěvku, přijde odpověď, že je uživatel odhlášen, zobrazím přihlašovací formulář, uživatel se znovu přihlásí a jeho příspěvek vidí stále rozpracovaný, tzn. může ho znovu odeslat.

7.2 Požadavky a Websocket API

7.2.1 REST požadavky

Z počátku implementace jsem si připravil pouze funkci, která vykonávala *fetch* funkci z Fetch API[2], ve které byly přednastavené některé vlastnosti požadavku, jako adresa serveru, formát odesílání dat apod.

Později se to neukázalo jako šťastné řešení, jelikož jsem musel u každého požadavku zpracovávat odpovědi serveru pro jednotlivé HTTP stavové kódy, jenže u spousty požadavků jsem vykonával stejné věci, tudíž se množily duplicity kódu.

Vytvořil jsem si třídu *PreparedRequest* využívající návrhového vzoru Singleton – instance třídy *PreparedRequest* je pouze jedna na celý běh aplikace [4]. Tato třída má metodu *run*, která spustí požadavek na server, přijímá parametr JavaScriptového objektu, ve kterém je zapsáno, co se má stát při konkrétní chybové odpovědi ze serveru, především obsahuje název metody třídy *PreparedRequest*, která se má vykonat při takové odpovědi.

Následující ukázka ukazuje, jak takový JavaScriptový objekt vypadá v případě registrace klubu, při HTTP odpovědi 401 (přístup zamítnut) ze serveru se provede metoda *isLogout401* ze třídy *PreparedRequest*, která vyvolá dialogové okno pro přihlášení uživatele. Při odpovědi 409 se zobrazí hláška, přes metodu vypisující chybovou hlášku u formuláře registrace klubu.

[2]Fetch API – poskytuje generickou definici požadavku a odpovědi serveru (a další věci spojené s požadavkem na server) [2]

```
{
    401: { method: "isLogout401" },
    409: {
            method: "message",
            args: {
                executor: this.message,
                content: notice(201)
            }
    }
}
```

Vzhledem k tomu, že využívám Fetch API, které je podporované ve všech moderních prohlížečích (v ČR funkční přibližně u 94,42 % uživatelů, viz 7.6), u starších prohlížečů, typu Internet Explorer 11 toto API není podporováno, z tohoto důvodu nebude aplikace v tomto prohlížeči funkční. [8]

7.2.2 Websocket API

Z analýzy vyplynulo, že spojení přes Websocket API[3] budeme potřebovat dvakrát (viz 3.4.1). Pro rezervace kurtů a párování závodních hráčů s uživateli aplikace.

Aby nedocházelo k duplicitnímu kódu, vytvořil jsem si třídu *PreparedWebsocket*, která obsahuje veškerou logiku websocketového spojení. Na rozdíl od třídy *PreparedRequest* (viz 7.2.1) není jedna instance pro celou aplikaci, mohlo by se totiž stát, že bude zapotřebí dvou spojení najednou, v současném návrhu aplikace to nicméně není zamýšleno.

Třída *PreparedWebsocket* obstarává opětovné připojení, dojde-li k výpadku spojení (Websocket API tuto možnost nenabízí). Umožňuje také restart připojení, např. z důvodu odhlášení nebo přihlášení uživatele, kdy je potřeba znovu ověřit identitu. A zajišťuje trvalé spojení, pomocí odesílání domluvené zprávy (textové, s obsahem: *-h-*) v určitém časovém intervalu na server, server by měl neprodleně odpovědět stejnou zprávou, dojde-li 3x v časovém intervalu k nečinnosti serveru, spojení se ukončí, na tuto skutečnost je upozorněn uživatel a dochází k opětovným pokusům o připojení.

Websocket API má k dispozici přibližně 96,57% uživatelů v ČR, viz 7.6, proto se může stát, že u některého uživatele k dispozici nebudou, bez podpory této technologie nebudou funkční rezervace a ani párování závodních hráčů s uživateli. Z nějakých důvodů nemusí být na zařízeních podporována ani samotná komunikace pomocí Web Sockets.

7.3 Ztráta připojení k síti

Jelikož u single-page aplikace nedochází k aktualizacím stránky, bylo potřeba vymyslet, co se stane v případě ztráty připojení k internetu. K identifikaci *online/offline* stavu využívám *window.navigator.online* proměnnou, nabývající hodnot *true*, pokud je prohlížeč připojen k síti a *false*, pokud není. Pro

[3] Websocket API – pokročilá technologie, umožňující dvou-cestné interaktivní spojení mezi klientem a serverem [2]

upozornění na změnu připojení používám události *Document.online/offline*, které zavolají moje funkce v případech připojení/odpojení od sítě. Tyto funkcionality má v ČR k dispozici přibližně 96,57% uživatelů, viz 7.6.

Jelikož SPA posílá neměnná data a soubory rovnou na začátku inicializace do prohlížeče uživatele, přesměrování mezi stránkami funguje i bez připojení k síti, pouze obrázky a videa se nenačtou. Informuji tedy uživatele, že ztratil připojení k síti a funkce aplikace jsou částečně omezené, tzn. že může například i bez připojení k síti psát nějaký delší popis tenisového klubu. Při opětovném připojení k internetu zobrazí aplikace hlášku, že spojení s internetem bylo opět navázáno, v tento moment může uživatel odeslat svoji rozpracovanou práci na server. Pokud by byl offline a pokusil se odeslat formulář, uživatel by získal hlášku, že se něco nepovedlo, pravděpodobně není připojen k internetu, nepřišel by však o žádná rozepsaná data a může to zkusit znovu, až bude znovu připojen k síti.

Bude-li chtít uživatel v offline stavu navštívit stránku, která potřebuje pro své zobrazení získat data ze serveru (například stránka konkrétního klubu nebo vyhledávání klubů), opět dostane hlášku *„něco se nepovedlo, pravděpodobně nejste připojen k internetu“*. Pokud znovu dojde k navázání spojení s internetem, bez interakce uživatele dojde například u vyhledávání k získání výsledků vyhledávání ze serveru.

7.4 Progressive web application

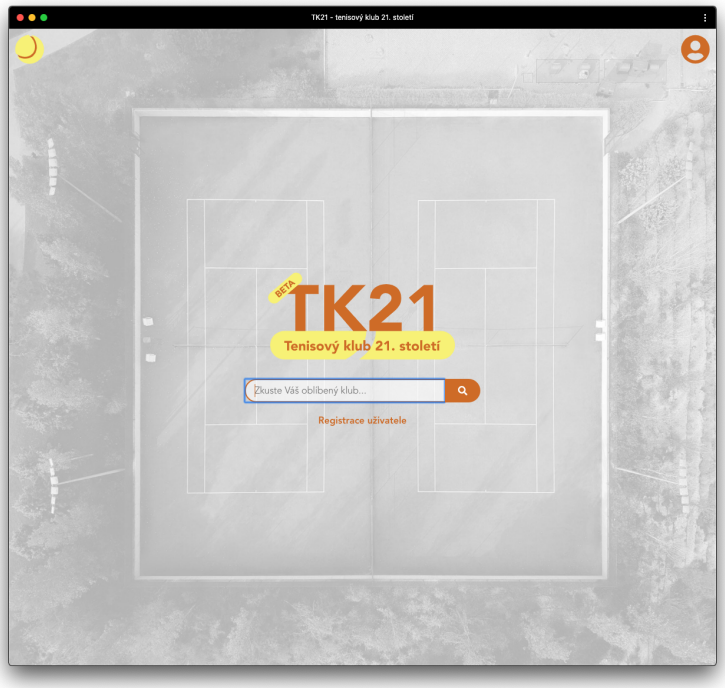
PWA (Progressive web application) je webová aplikace sestavená a vylepšená moderním API, přinášející vlastnosti nativních aplikací, jako je spolehlivost a instalovatelnost, zatímco je spustitelná na prakticky jakémkoliv zařízení. [11]

7.4.1 Ukládání do mezipaměti

React poskytuje všechny nástroje pro vytvoření PWA defaultně při generování produkční verze kromě možnosti aplikaci spustit bez připojení k internetu. K tomu, aby se aplikace načetla i bez připojení k internetu, je zapotřebí zaregistrovat Service Worker (podpora viz 7.6), který umožňuje zastavovat a kontrolovat, jak webový prohlížeč zachází s požadavky na server (z naší webové aplikace) a umožňuje také ukládání do mezipaměti [2].

Díky ukládání do mezipaměti se aplikace stáhne do prohlížeče při prvním načtení webu, následně pokud přijdete o připojení k internetu, stránka aplikace se vám stejně načte (jen bude fungovat omezeně). To řeší jeden z problémů SPA, kterým je pomalé počáteční načtení aplikace, jelikož webová aplikace se bude načítat z mezipaměti, i pokud připojení k internetu bude (požadavky na server se do mezipaměti neukládají), tzn. že aplikace se poté bude načítat prakticky okamžitě.

Dojde-li ke změně dat uložených do paměti, Service Worker je stáhne na pozadí a změní je po restartování aplikace (vypnutí všech oken s aplikací a opětovným spuštěním), vše probíhá tak, že uživatel to nepostřehne.



Obrázek 7.1: PWA: ukázka nainstalované aplikace na desktopu (macOS Catalina)

Obrázek 7.2: PWA: ukázka nainstalované aplikace na smartphonu (iOS 13)

7.5 Zabezpečení

7.5.1 XSS

React sám zajišťuje zabezpečení proti XSS[4], a to tak, že překresluje do HTML pouze JSX elementy (viz 6.2.3), pokud bych se pokusil vložit obyčejný HTML kód, React ho vykreslí jako text (znaky rezervované pro HTML nahradí entitami[5]), to platí i pro uživatelský vstup, který bych obdržel od serveru jako textový řetězec. Tato vlastnost se dá obejít přes atribut JSX elementu *dangerouslySetInnerHTML*, který využívám v následujícím případě.

Rozhodl jsem se implementovat možnost formátovat klubové příspěvky. Formátování příspěvků ale vyžaduje vkládání HTML kódu od uživatele, to přináší zranitelnost na XSS útok. Našel jsem řešení v podobě Quill JS textového editoru (`https://quilljs.com`), který uživatelský vstup transformuje do JSON kódu, takový kód obsahuje informace o stylech a samotný text. JSON odešlu na server a při výpisu příspěvků mi je navrácen, následně ho s pomocí Quill JS převedu na HTML kód, který dosadím do příspěvku. Formátování jsem omezil pouze na tučný text, kurzívu, odstavce a seznamy – při transformaci z JSONu do HTML se převede pouze to formátování, které je povolené.

Kód 7.6: Ukázka Quill JSONu s formátovaným obsahem

```
{
  ops: [
    { insert: 'TK21', attributes: { bold: true } },
    { insert: ' - aplikace pro kluby 21. století', attributes: { color:
     '#cccccc' } }
  ]
}
```

7.5.2 CSRF

Jako obranu proti CSRF[6] mi při přihlášení backendová strana zašle CSRF token, který uložím a následně ho posílám v hlavičce při každém požadavku serveru.

[4]Cross-site scripting – útočník vloží do HTML kódu stránky škodlivý kód, kterým může získat nebo ovlivnit klientská data [2]

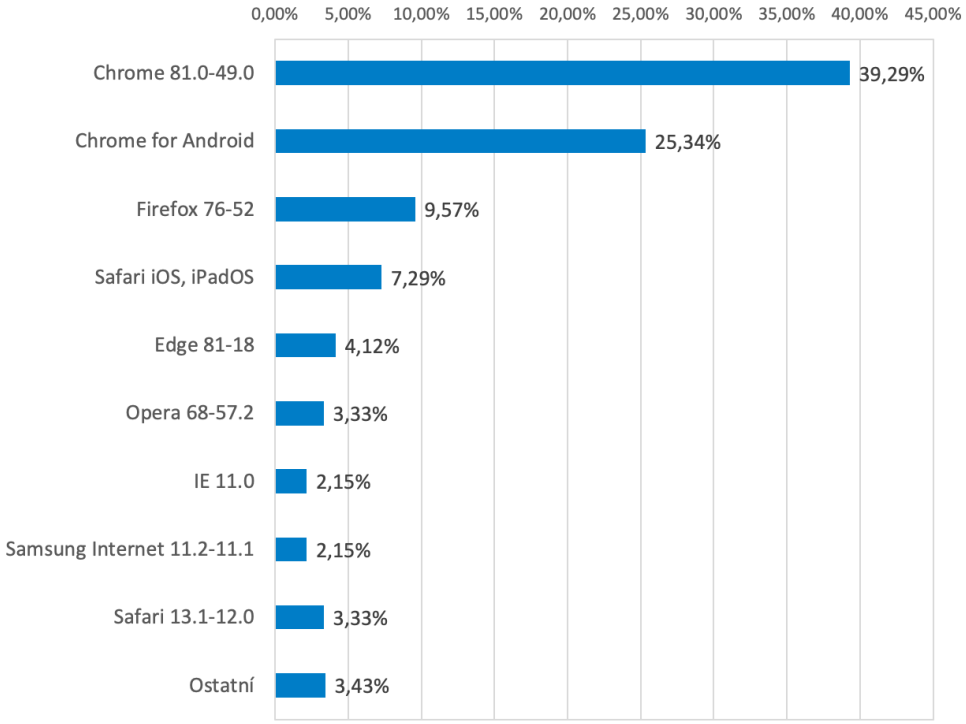
[5]HTML entita – kus textu začínající znakem *&* a končící *;*, který prohlížeč vyobrazí jako některý ze znaků, rezervovaných pro HTML syntaxi nebo jako některý speciální znak (např. znak pro copyright – *©*) [2]

[6]Cross-Site Request Forgery – útok, při kterém útočník nabídne uživateli odkaz/formulář, který vykoná nějaké činnosti na jiné stránce, jelikož uživatel může být na takové stránce ověřený, kliknutím na takový odkaz (odesláním formuláře) může vykonat nechtěné akce (např. změnu e-mailu) [2]

7.6 Metodika odhadu podpory funkcionalit v ČR

V předchozích částech jsem uváděl procentuální podpory jednotlivých funkcionalit v prohlížečích uživatelů v České republice. Jelikož se tato data obtížně shánějí a nenalezl jsem statistiky podpory funkcionalit pouze pro ČR (jen globálně, viz [8]), musel jsem tato data odhadnout.

K dispozici jsou data o využívání verzí prohlížečů v ČR [13]. Dále je k dispozici web Can I use, kde se dá zjistit, zda konkrétní verze prohlížeče podporuje onu danou funkcionalitu [8]. Tudíž na základě využívání předních prohlížečů v ČR (zobrazené viz Obrázek 7.3), které podporují danou funkcionalitu, si odvodím procentuální podporu dané funkcionality v ČR. Tato data jsou platná k 9. květnu 2020.



Obrázek 7.3: Používání verzí prohlížečů v ČR k 9. květnu 2020 (zdroj dat [13])

Nevýhodou takto zjištěných dat je, že některé funkcionality se dají v prohlížečích vypnout, avšak dá se předpokládat, že většina uživatelů naší aplikace budou uživatelé, které neprovádí takto zásadní změny v nastavení prohlížeče.

Pod položkou *ostatní* se většinou nacházejí nižší verze prohlížečů, které jsou v grafu přímo jmenované. Tyto prohlížeče mohou, ale také nemusí danou funkcionalitu podporovat, proto je nezapočítávám do statistik mezi podporované a nepodporované. Výsledná procenta tedy udávají přibližnou dolní hranici podpory funkcionality v ČR.

Fetch API

- **Podporují prohlížeče:** Chrome 81.0-49.0; Chrome for Android; Firefox 76-52; Safari iOS/iPadOS; Edge 81-18; Opera 68.0-57.2; Samsung Internet 11.2-11.1; Safari 13.1-12.0 ... **94,42%**
- **Nepodporují prohlížeče:** IE 11.0 ... **2,15%**
- **Ostatní: 3,43%**

Websocket API

- **Podporují prohlížeče:** Chrome 81.0-49.0, Chrome for Android, Firefox 76-52, Safari iOS/iPadOS, Edge 81-18, Opera 68.0-57.2, IE 11.0, Samsung Internet 11.2-11.1, Safari 13.1-12.0 ... **96,57%**
- **Nepodporují prohlížeče:** *žádný z uváděných* ... **0%**
- **Ostatní: 3,43%**

Online/offline stav

- **Podporují prohlížeče:** Chrome 81.0-49.0, Chrome for Android, Firefox 76-52, Safari iOS/iPadOS, Edge 81-18, Opera 68.0-57.2, IE 11.0, Samsung Internet 11.2-11.1, Safari 13.1-12.0 ... **96,57%**
- **Nepodporují prohlížeče:** *žádný z uváděných* ... **0%**
- **Ostatní: 3,43%**

Service Workers

- **Podporují prohlížeče:** Chrome 81.0-49.0, Chrome for Android, Firefox 76-52, Safari iOS/iPadOS, Edge 81-18, Opera 68.0-57.2, Samsung Internet 11.2-11.1, Safari 13.1-12.0 ... **94,42%**
- **Nepodporují prohlížeče:** IE 11.0 ... **2,15%**
- **Ostatní: 3,43%**

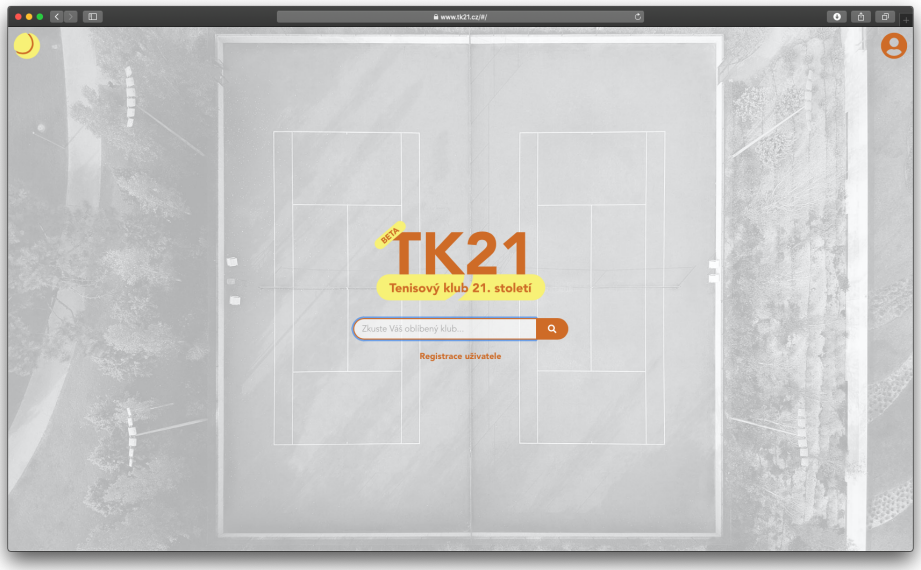
Kapitola 8

UI Design

Aplikace TK21 je přizpůsobená pro zařízení s rozlišením desktopu, až po mobilní zařízení s minimální šířkou rozlišení 280 pixelů. Ve všech verzích designu jsou zachovány veškeré funkcionality aplikace.

Aby bylo prostředí aplikace uživatelsky přívětivě, zvolil jsem pouze 3 barvy, které se v celé aplikaci opakují (kódy barev #CE6B27, #F7F176 a #F2DCCA).

Pro zpestření aplikace využívám ikony z Font Awesome[1], jedná se o službu poskytující širokou škálu vektorových ikon zdarma.



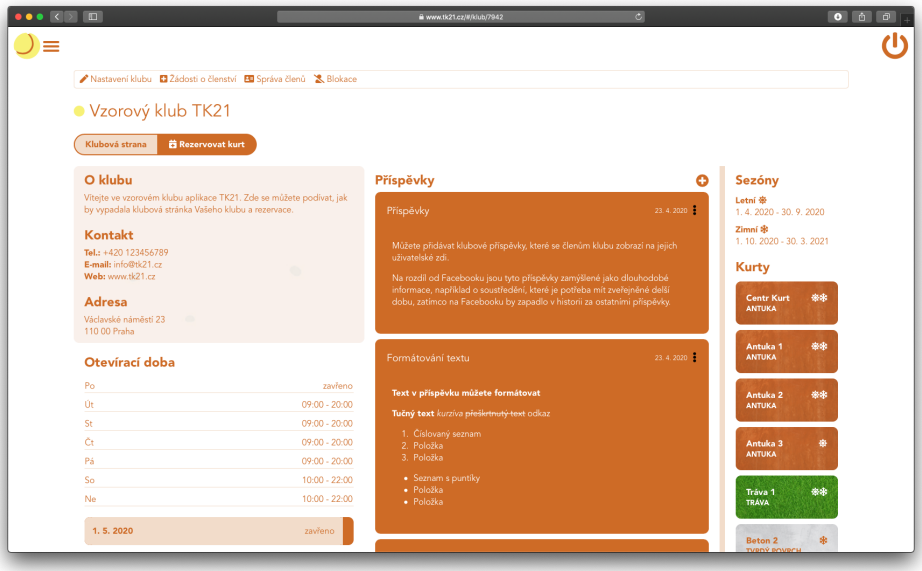
Obrázek 8.1: UI design: úvodní strana aplikace

[1] `https://fontawesome.com`

8.1 Klubová stránka

Stránka je rozdělena do třech sekcí (na desktopovém zařízení se zobrazí jako 3 sloupce, viz obrázek 8.2). První sekce obsahuje základní informace o klubu – popis klubu, kontakt, adresu (kliknutím na adresu se vyhledá na Google mapách) a otevírací dobu se změnami v otevírací době. Druhá sekce (na desktopu prostřední sloupec) obsahuje seznam příspěvků klubu, seřazených od nejnovějších po nejstarší. V poslední sekci jsou informace o sezónách a kurtech. V horní části klubové stránky se nachází název klubu, pod kterým je navigace, která zobrazuje odkaz na rezervaci kurtu, pokud se jedná o klub z ČTS, nachází se zde taky odkazy na stránky se seznamem turnajů a soutěží družstev. Nachází-li se na stránce klubu administrátor, v horní části se mu zobrazí ještě odkazy na nastavování klubu a správu členů.

Příspěvků je na stránce zobrazeno maximálně 20, má-li klub vícero příspěvků zobrazí se na konci tlačítko, které provede požadavek a načte dalších 20 příspěvků.



Obrázek 8.2: UI design: klubová stránka

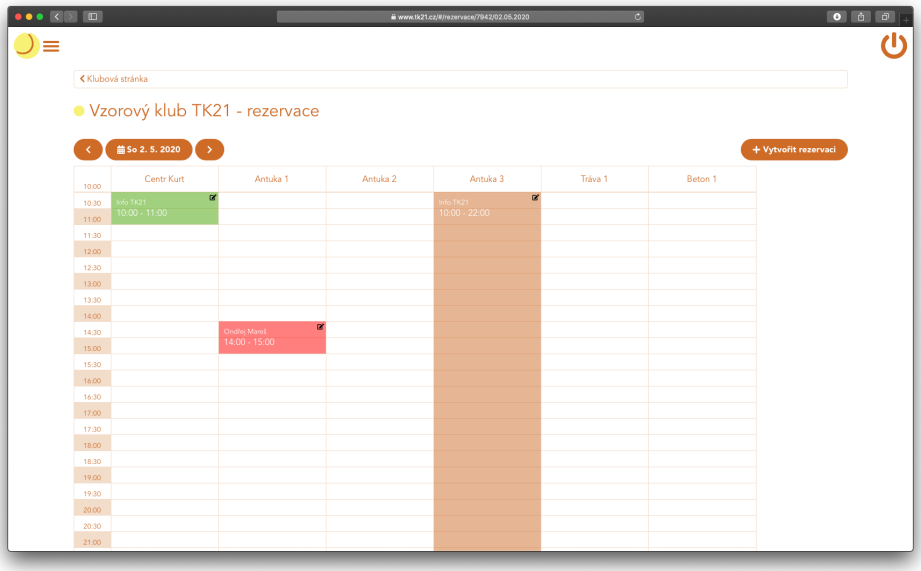
8.2 Rezervace kurtu

Jednotlivé rezervace zobrazuji na časové ose, kde v horizontálním směru jsou sloupce, které znázorňují jednotlivé kurty (pokud je kurtů víc, než je šířka zařízení, sloupce se dají posouvat, viz obrázek 8.3). Ve vertikálním směru je znázorněn čas po 30 minutách. V prostřední části osy se poté vykreslují jednotlivé rezervace, které na sobě zobrazují text s časem, pokud rezervaci může uživatel upravovat, tak navíc jméno a příjmení, na koho je rezervace rezervována a ikonka znázorňující editaci – po kliknutí na ikonku se zobrazí formulář s možností editace rezervace, jednotlivé rezervace jsou podbarvené (viz tabulka 8.1).

V horní části stránky s rezervacemi se nachází tlačítko, které vyvolá kalendář, ve kterém uživatel může vybrat, jaký den si přeje zobrazit, po stranách tohoto tlačítka se nacházejí šipky, kterými se dá rychle přepínat mezi dny – dá se předpokládat, že se uživatel bude chtít často podívat na následující dny, hledání v kalendáři by bylo v takovém případě méně uživatelsky přívětivé.

Barva	Význam
Červená	Rezervace cizího uživatele
Oranžová	Opakující se rezervace (např. tréninky)
Zelená	Moje rezervace

Tabulka 8.1: Podbarvení rezervací

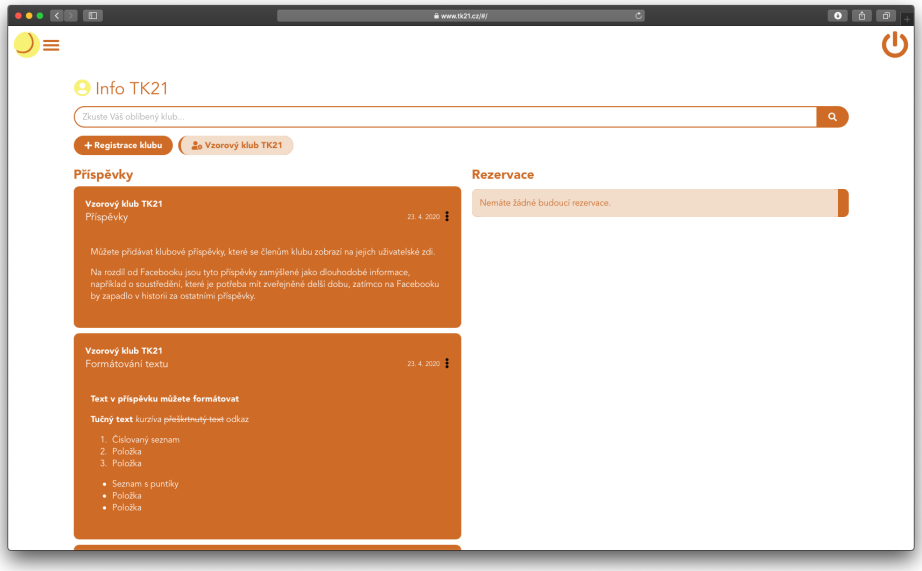


Obrázek 8.3: UI design: rezervace

8.3 Přehled uživatele

Přehled přihlášeného uživatele je stránka, která se zobrazuje místo úvodní strany, pokud je uživatel přihlášen (viz obrázek 8.4). V horní části této strany se nachází pole pro vyhledávání klubů, tlačítko pro registraci nového klubu a seznam klubů, v nichž je uživatel členem.

Spodní část stránky je rozdělena na dvě sekce (na desktopu 2 sloupce), v jedné sekci nalezne uživatel aktuální příspěvky z klubů, ve kterých je členem, v druhé sekci jeho následující rezervace, a pokud se jedná o hráče z ČTS, tak také následující turnaje a soutěže smíšených družstev.

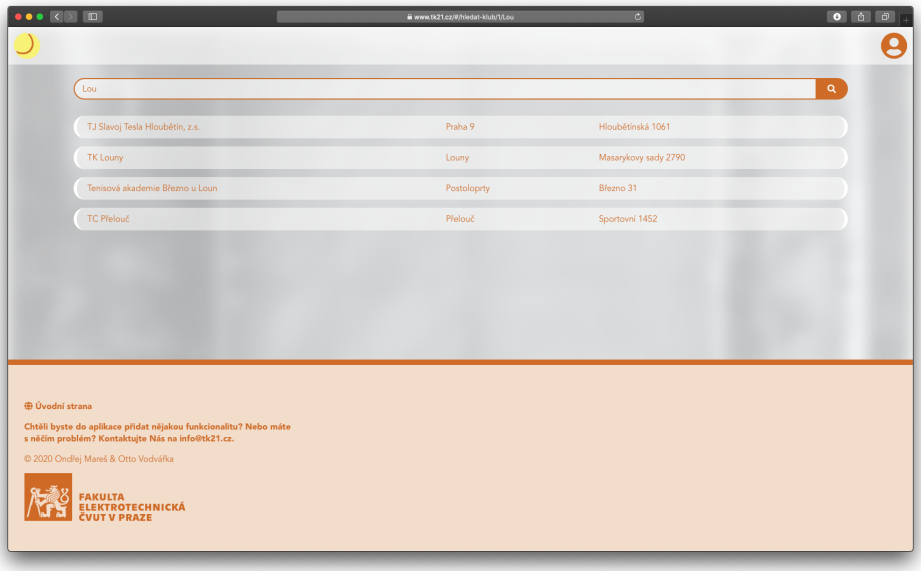


Obrázek 8.4: UI design: přehled uživatele

8.4 Vyhledávání klubů

Při vyhledávání klubů se upřednostňují kluby zaregistrované v aplikaci, zobrazují se ale i kluby z ČTS, které nejsou zaregistrované, odlišuje je od ostatních bílé podbarvení (viz obrázek 8.5), na stránce takového klubu se nezobrazují veškeré informace a je zde hláška o tom, že klub není v aplikaci zaregistrován. Klub, který je zaregistrovaný do aplikace a zároveň je z ČTS, má při vyhledávání před názvem ikonu „fajfky“, která značí, že se jedná o ověřený klub z ČTS.

Vyhledávání probíhá dynamicky, po 500 milisekundách bez psaní textu se začne vyhledávat automaticky.



Obrázek 8.5: UI design: vyhledávání klubů

Část IV

Závěr

Kapitola 9
Kvalitativní testování

Aplikace TK21 je testována v Tenisové akademii Březno u Loun[1]. Jedná se o malý „rodinný“ tenisový klub se třemi venkovními a jedním krytým kurtem, hrající severočeské soutěže družstev, ve kterých hraji za tento klub i já, tudíž se i já přímo podílím na testování.

Z důvodu šíření nemoci COVID-19 mohlo testování v reálném provozu začít až začátkem května 2020 s rozvolňováním opatření. Bohužel severočeský oblastní tenisový svaz zrušil soutěže družstev, tzn. že funkcionality související s touto oblastí nemohou být do odevzdávání práce otestovány. Závěry z testování se tedy soustředí především na rezervace kurtů a sdílení klubových informací.

9.1 Testovací scénář

Nejprve byl Tenisové akademii Březno u Loun předložen testovací scénář, viz příloha A, kdy měli postupovat podle bodů a plnit za sebou jednotlivé úkoly, výsledky zapisovat k jednotlivým bodům, které jsou sestaveny tak, aby se prošla většina funkcí aplikace, především těch hlavních.

Z průchodu testovacího scénáře vyplynuly drobné změny v aplikaci, jako změna názvu tlačítka při vytváření příspěvků. Zjistila se také významná chyba, kdy při změně otevírací doby docházelo k následným pádům mé frontendové části aplikace.

Po vyplnění testovacího scénáře následovala osobní schůzka, na které se řešily další funkční změny v aplikaci (viz 9.3). Následně při dlouhodobějším testování v reálném provozu se objevily další chyby (viz 9.2).

[1]Web: `http://www.tenisbrezno.cz/`, ČTS: `http://cztenis.cz/adresar-klubu/40103`

9.2 Zjištěné chyby

Popis:	Při změně názvu přijde ze serveru odpověď 500 (internal server error) a nedojde ke změně názvu.
Kde:	Backendová strana
Závažnost:	3/5*
Řešení:	Oprava.
Popis:	Po změně otevírací doby přestane server zasílat otevírací dobu v požadavcích, což způsobí pád frontendové části aplikace.
Kde:	Backendová a frontendová strana
Závažnost:	5/5*
Řešení:	Oprava na backendové straně a kontrola otevírací doby na frontendové straně, zda je posílána.
Popis:	Není-li k dispozici datum konání utkání v soutěžích družstev, dojde k pádu frontendové části aplikace.
Kde:	Frontendová strana
Závažnost:	4/5*
Řešení:	Přidání vlastnosti data *„není zveřejněno“* a kontrola, zda datum utkání přijde.
Popis:	Při šířce rozlišení menší jak 510 pixelů dojde v případě klubu z ČTS k přetečení menu v horní části (soutěže družstev, turnaje, rezervace, klubová stránka), to způsobí větší šířku obsahu stránky, než je zmíněných 510 pixelů.
Kde:	Frontendová strana
Závažnost:	1/5*
Řešení:	Přizpůsobení menu pomocí kaskádových stylů pro šířku menší jak 510 pixelů.

** 5 nejzávažnější, 1 nejméně závažná chyba*

9.3 Funkční změny

9.3.1 Telefon v rezervacích

Ukázalo se, že vyžadovat při rezervaci kurtu pouze e-mailovou adresu je nedostačující a klub potřebuje spíše při rezervacích telefonní číslo pro rychlé oznámení nečekaných událostí.

E-mail musí být zachován z důvodu zasílání rekapitulace (a odkazu pro

zrušení rezervace nepřihlášenému uživateli), tudíž jej nebudeme nahrazovat telefonním číslem, ale pouze telefonní číslo doplníme jako další prvek – administrátorovi umožníme zvolit si, zda bude telefonní číslo povinné, či nikoliv. Registrovaný uživatel bude mít možnost si telefonní číslo uložit a bude mu předvyplněno automaticky.

9.3.2 Cizí rezervace přes administrátora

V Březně chtěli ze začátku, aby rezervace sloužily pouze k informování o zaplněnosti kurtů, resp. aby nebylo možné, aby se hráči rezervovali sami. To je umožněno vypnutím rezervací jiných uživatelů než administrátora pro aktuální sezónu. Avšak neexistuje možnost jak zarezervovat kurt administrátorem na jiného uživatele – vyplněním e-mailové adresy a telefonu, jména a příjmení. Primárně by tyto informace měly sloužit k určení, kdo je na danou hodinu zarezervován, avšak pokud bude rezervace provádět pouze administrátor, budou všechny rezervace uložené na administrátora a nebude přehled o tom, kdo má zarezervovanou konkrétní hodinu.

Umožníme tedy administrátorovi rezervovat na cizí jméno, příjmení, telefon a e-mail, přičemž zde bude existovat možnost odeslat, či neodeslat rekapitulaci na uživatelský e-mail (případně odkaz na smazání rezervace).

9.3.3 Detail příspěvku s vlastní adresou

Klubové příspěvky neobsahují detail, resp. stránku, na které by byl celý příspěvek vypsaný, to se ukázalo jako nedostačující řešení, kdy je potřeba někdy sdílet odkaz na samotný příspěvek, aby jej uživatel nemusel hledat v seznamu příspěvků na klubové stránce.

Na základě této skutečnosti v aplikaci přibude stránka s detailem příspěvku s vlastní URL adresou.

Kapitola 10

Shrnutí

Mým hlavním cílem nebylo vytvářet aplikaci jen jako podklad k bakalářské práci, hlavním cílem bylo vytvořit nástroj, který bude praktický a funkční i po skončení bakalářské práce. Vzhledem k počátečním reakcím z Tenisové akademie Březno u Loun si myslím, že se nám to povedlo a tato aplikace má skutečně potenciál změnit využívání moderních technologií v tenisových klubech v České republice.

Naimplementovali jsme vše, co jsme si dali za cíl na začátku práce. Navíc jsme přidali i nějaké funkcionality navíc, které původně nebyly zamýšleny (z počátku se například nepočítalo s přehledem uživatele, viz 3.6).

Předpokládám, že se časem, zvláště při využívání aplikace v reálném provozu, objeví další vlastnosti, které je potřeba doplnit, případně nějaké chyby. S kolegou jsme připraveni na takové věci reagovat, a pokud budou smysluplné, implementovat je do aplikace i v budoucnu.

Aplikace TK21 je dostupná na webové adrese `https://www.tk21.cz` a jakýkoliv klub v České republice se do ní může zaregistrovat a využívat ji.

Přílohy

Příloha A

Testovací scénář – Tenisová akademie Březno u Loun

Vyzkoušejte postupně každý z následujících bodů a připište k němu, zda a jak bylo složité splnit úkol. Nebudete-li si vůbec vědět rady, obraťte se na Ondřeje Mareše (mareson3@fel.cvut.cz).

1. **Registrujte Váš klub.**
   „Registrace klubu jednouchá, srozumitelná.“

2. **Nastavte tenisové kurty ve Vašem klubu – název např. Centr, 1, 2, V lese apod. (nepište slovo kurt do názvu, používejte výhradně číslovky).**
   „Založeni tenisových kurtů srozumitelné.“

3. **Nastavte otevírací dobu Vašeho klubu.**
   „Běžná otevírací doba nastavena a plně vyhovuje.“

4. **Nastavte sezóny v klubu, v následující sezóně neumožněte rezervace.**
   „Výhoda pro antukové kurty.“

5. **Zarezervujte si jakoukoliv hodinu na jakémkoliv kurtě.**
   „Reservace jednoduchá a jasná.“

6. **Vytvořte opakující se rezervaci – opakovat se bude jednou za 7 dní, po dobu alespoň 28 dní, tj. čtyřikrát.**
   „Při prvním pokusu provedení trochu nesrozumitelné, následně opakující se reservace proběhla v pořádku.“

7. **Přidejte závodního hráče – Ondřej Mareš, email: ondramares@ondramares.com, 22. 5. 1997.**
   „Líbilo se mi, že je možné přetáhnout informace z tenisového svazu a je to vlastně jednoduché zadat závodního hráče.“

8. **Vytvořte v aplikaci další uživatelský účet – na libovolné údaje. Následně se na něj přihlaste do aplikace, vyhledejte Váš klub a požádejte o členství v klubu.**
   „Uživatelský účet vytvořen, klub vyhledán a členství bylo přijato.“

9. **Odhlaste se a přihlaste se znovu na administrátorský účet. Nyní potvrďte žádost o členství z předchozího bodu.**
   „V pořádku.“

10. **Nového člena klubu udělejte také administrátorem.**
    „Jednoduché, přehledné.“

11. **Přidejte příspěvek s obrázkem – zkuste obsah příspěvku naformátovat, tučně, kurzívou, vytvořte seznam – následně přidejte libovolný obrázek a příspěvek vytvořte.**
    „Zadání příspěvku jednoduché, tlačítko Přidat, spíše bych volila uložit. Pochopila jsem jako přidání nového příspěvku, to jsem nechtěla, tedy nepřidala a zároveň neuložila a příspěvek se nezobrazil.“

12. **Přidaný příspěvek odstraňte.**
    „Odstraněno.“

Scénář byl vyplněn Tenisovou akademií Březno u Loun.

Příloha B

Literatura

[1] RAUSCHMAYER, Axel. Deep JavaScript: Theory and techniques [online]. Mnichov, Německo, 2019 [cit. 2020-04-28]. Dostupné z: https://exploringjs.com/deep-js/index.html

[2] Web technology for developers [online]. Mozilla and individual contributors, c2005-2020 [cit. 2020-04-22]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web

[3] Adresář tenisových klubů. Český tenisový svaz [online]. Praha, 2020 [cit. 2020-04-22]. Dostupné z: http://cztenis.cz/adresar-klubu

[4] OSMANI, Addy. Learning JavaScript design patterns: [a JavaScript and jQuery developer's guide]. Sebastopol: O'Reilly, 2012. ISBN 978-144-9331-818.

[5] ČÁPKA, David. Lekce 2 - UML - Use Case Diagram. ITnetwork.cz [online]. c2020 [cit. 2020-04-28]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram

[6] ČOŘIČ, Oskar. Rozdíl mezi UI a UX. Oskar Čořič [online]. c2020, 20/12/2017 [cit. 2020-04-28]. Dostupné z: https://oskarcoric.cz/rozdil-mezi-ui-a-ux/

[7] JAHODA, Bohumil. Single page application. JE ČAS [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://jecas.cz/spa

[8] Can I use [online]. Alexis Deveria, c2020 [cit. 2020-04-28]. Dostupné z: https://caniuse.com

[9] FABISIAK, Radek. Which front-end framework is the best in 2019. In: Duomly [online]. c2020, March 10, 2019 [cit. 2020-04-29]. Dostupné z: https://www.blog.duomly.com/which-front-end-framework-is-the-best-in-2019/

[10] ELIOTT, Eric. Top JavaScript Frameworks and Topics to Learn in 2019. In: Medium [online]. Jan 1, 2019 [cit. 2020-04-29]. Dostupné z: https://medium.com/javascript-scene/top-javascript-frameworks-and-topics-to-learn-in-2019-b4142f38df20

[11] Progressive Web Apps. Web dev [online]. Mountain View, Kalifornie, USA: Google developers, 2020 [cit. 2020-05-01]. Dostupné z: https://web.dev/progressive-web-apps/

[12] MALÝ, Martin. REST: architektura pro webové API. In: Zdroják.cz [online]. Budějovická 1550/15a, Praha 4: Devel.cz Lab, 3. 8. 2009 [cit. 2020-05-01]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/

[13] Browser Version Market Share Czech Republic. Statcounter – Global Stats [online]. c1999-2020, May 2020 [cit. 2020-05-09]. Dostupné z: https://gs.statcounter.com/browser-version-market-share/all/czech-republic/#monthly-202005-202005-bar