FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Systém pro správu leteckých soutěží
Student:	Ivan Harašta
Vedoucí:	Ing. Filip Glazar
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat webovou aplikaci, která bude sloužit k evidování leteckých soutěží a to konkrétně závodů větroňů. Zároveň systém bude umožňovat zaznamenání výsledků závodu a jejich vizualizaci. Pro implementaci serverové části aplikace budou použity následující technologie: NodeJS, MongoDB a Express framework. Pro implementaci klientské části React a pro uživatelské rozhraní a vizualizaci Material UI.

1. Proveďte rešerši existujících řešení
2. Navrhněte serverou a klientskou část aplikace
3. Na základě návrhu implementujte obě části aplikace
4. Nasaďte aplikaci do reálného provozu
5. Zhodnoťte použitelnost aplikace na reálných datech a navrhněte možná budoucí vylepšení

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 12. srpna 2020 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Systém pro správu leteckých soutěží

Ivan Harašta

Katedra softwarového inženýrství
Vedoucí práce: Ing. Filip Glazar

12. května 2021

Poděkování

Tímto bych chtěl poděkovat Ing. Filipu Glazarovi, za vedení mé práce, za jeho čas, cenné informace a příjemnou spolupráci. Mé poděkování také patří mé rodině za podporu během studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při 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 předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, 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 zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). 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 zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 12. května 2021 . . . . . .

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2021 Ivan Harašta. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Harašta, Ivan. *Systém pro správu leteckých soutěží.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021. Dostupný také z WWW: ⟨`https://www.tcup.cz`⟩.

Abstrakt

Bakalářská práce se věnuje návrhu a implementaci webové aplikace, která slouží k evidování leteckých soutěží, konkrétně závodů větroňů. Zároveň aplikace umožňuje evidování výsledků a jejich vizualizaci.

Klíčová slova webová aplikace, informační systém, soutěž, plachtění, React, SPA, NodeJS, Express, MongoDB, Naviter

Abstract

Subject of this bachelor thesis deals with design and implementation of web application, which manages flying competitions, gliding competitions in particular. At the same time, application allows results logging and visualisations.

Keywords web application, information system, competition, gliding, React, SPA, NodeJS, Express, MongoDB, Naviter

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Plachtění je krásnou aktivitou, při které se dá prožít mnoho zážitků, ale která je veřejnosti celkem neznámá. Protože se plachtění nedá dobře sledovat, není proto pro diváky moc atraktivní. V České republice létá v 50 aeroklubech asi 3 000 plachtařů. Já jsem jedním z nich, což je jedním z důvodů, proč se tímto tématem zabývám.

Plachtění je ale také i kompetitivním sportem, proto se v tomto letu bez motoru konají soutěže. Snaha plachtařů porovnat své výkony tu byla od samého počátku plachtění a je tu i dnes.

Aeroklub Toužim, ve kterém jsem dlouholetým členem, pořádá od roku 2009 každoročně soutěž *tcup* [1]. Organizátoři soutěže do roku 2019 používali informační systém, který uměl jen základní věci a proto mě oslovili s návrhem vytvoření nového systému – nové webové aplikace.

Tento nový systém by vyřešil mnoho problémů spojené s organizací soutěže, počínaje registracemi soutěžících, přes každodenní komunikaci se závodníky, vyhodnocování výsledků ale i samotné zobrazování výsledků. Nad vyhodnocování výsledků stráví organizátoři mnoho času. Dá se tu tedy velmi ušetřit.

Cílem této práce je zanalyzovat existující řešení, vymezit funkční i nefunkční požadavky, navrhnout a implementovat webovou aplikaci. Systém bude umožňovat zobrazení výsledků včetně vizualizace samotných letů na mapě. To je jedna z věcí, kterou existující systémy postrádají. Nesmí chybět nasazení do reálného provozu.

Práci jsem rozčlenil do 5 kapitol. Kapitolou č. 1 je část práce, kde rozebírám plachtění do většího detailu, analyzuji existující řešení a specifikuji požadavky na systém. V kapitole č. 2 navrhuji architekturu a použité technologie. Kapitoly č. 3 a č. 4 se zabývají samotnou implementací serveru a klienta. Konečně kapitola č. 5 popisuje nasazení aplikace a použití v reálném provozu při soutěži. V závěru práce hodnotím splnění cílů práce a popisuji možnosti rozšíření aplikace.

Kapitola 1

Analýza

1.1 Soutěžní plachtění

Pro pochopení kontextu práce, je potřeba vysvětlit co je soutěžní plachtění. V plachtění spolu soutěží kluzáky. Kluzák je bezmotorové letadlo, které je velmi lehké s velkou štíhlostí křídel. Díky tomu může bez motoru uletět velké vzdálenosti (až 1000 km za jediný den). Toto může zvládnout pouze v kombinaci s dobrými meteorologickými podmínkami a schopným pilotem.

Pilot využívá stoupajícího vzduchu a jakmile nastoupá potřebnou výšku, letí o kus dál a proces se opakuje. V plachtění se pilot může v tomto sportu vydat dvěma směry. Buď létá tzv. přelety, kde se snaží po předem definované trati uletět co největší vzdálenost a nebo se může účastnit soutěží – závodů. Tam se soutěží o to, kdo poletí průměrně nejrychleji opět na předem definované trati, která je pro všechny závodníky stejná.

Nezáleží ale jen na pilotovi, ale také na jeho kluzáku. Každý kluzák je jinak výkonný, proto je mu přiřazen koeficient, aby piloti s horším kluzákem nebyli hendikepováni oproti pilotům s lepším kluzákem. To eliminuje výkonnostní rozdíly a na konci záleží pouze na schopnostech pilota.

Plachtařská soutěž obvykle trvá okolo dvou týdnů. Stává se, že ne každý den se může soutěžně letět, hlavní příčinou jsou nevhodné meteorologické podmínky. Ze dnů, kdy je možné letět, se každý den určí trať, kterou se piloti snaží obletět co nejrychleji. Za každý den dostanou body, které jsou určeny vzdáleností, rychlostí a koeficientem kluzáku a na konci soutěže vyhraje ten, kdo má v součtu nejvíce bodů.

Každý pilot zaznamenává do letového zapisovače svůj let, který pak odevzdá rozhodčím a ty jej vyhodnotí.

1.2 Analýza existujících řešení

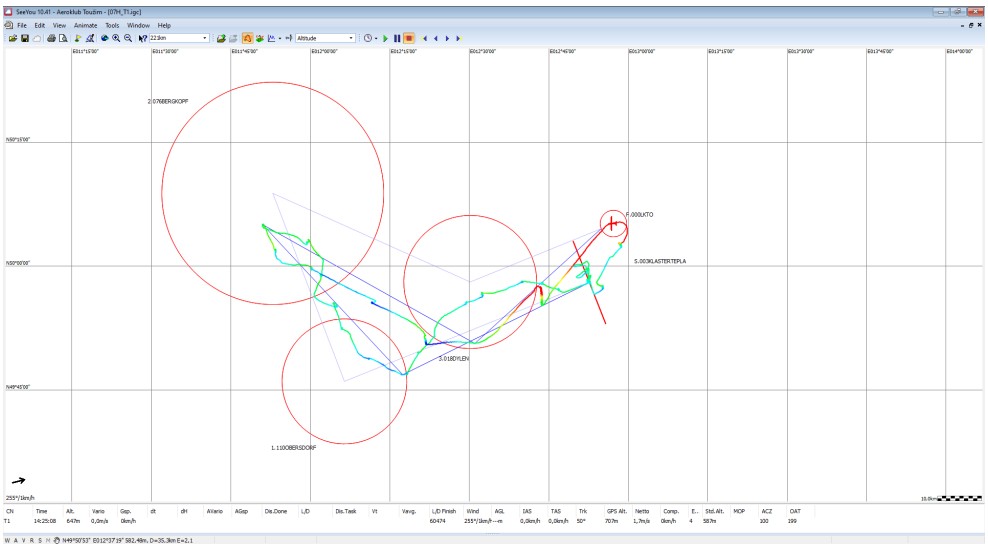
Je nutné vysvětlit, jaké nástroje používají nejen diváci a závodníci, ale i organizátoři. V každé subsekci vysvětlím, na co se programy používají a jakou alternativu nabízím já.

1.2.1 SeeYou

Program SeeYou od společnosti Naviter je hlavním programem, který používá každý závodník. [2] Slouží hlavně k plánování letů a jejich přehrávání a analýze. V letadle mají závodníci letový zapisovač, který zapisuje GPS polohu a výšku do IGC souboru (logu letu).

Tento soubor se pak dá využít nejen k analýze, ale právě i k vyhodnocování výsledků v závodech (více v sekci 1.2.2). Analyzovat svůj let, ale i lety ostatních (po stažení jejich IGC souborů ze SoaringSpotu) mohou piloti právě v tomto programu. Jak program vypadá můžeme vidět na obrázku 1.1.

Problém ovšem nastává v tom, že tento program mají pouze závodníci a nikoliv diváci. Ti pak často neví, že se na let mohou podívat v 1.2.3 Soaring Spotu. Já budu mít vizualizaci letu přímo na stránkách soutěže, tudíž tento problém vyřeším.



Obrázek 1.1: Přehrávání letu v programu SeeYou

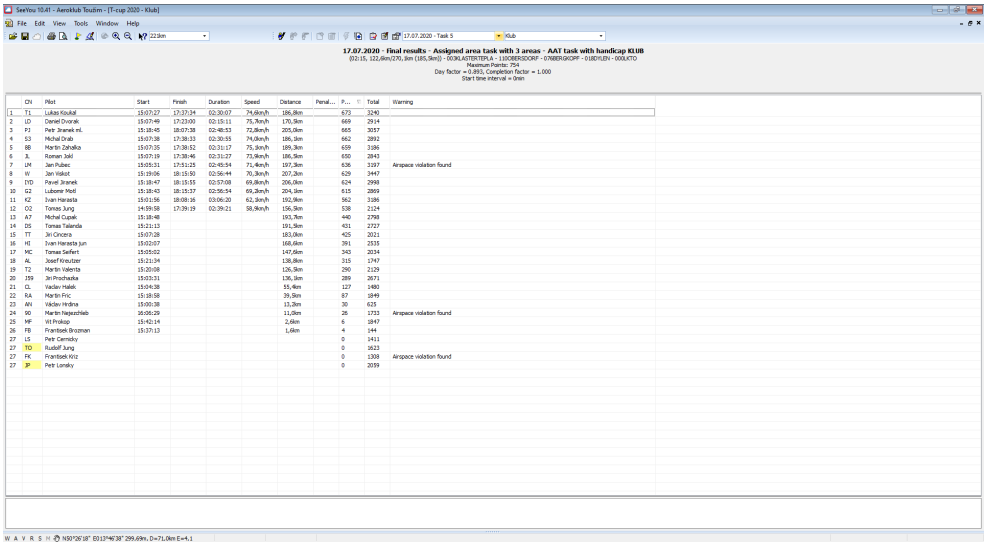
1.2.2 SeeYou Competition

SeeYou Competition je část programu SeeYou určená k administraci závodů. [3] Neexistuje alternativa, tudíž se používá ve všech soutěžích, včetně té naší.

V SeeYou Competition se vytvoří soutěž společně se všemi nastaveními, poté se vloží seznam zaregistrovaných závodníků. Pro každý závodní den, piloti odešlou své IGC soubory.

SeeYou Competition z těchto souborů vyhodnotí pomocí interní logiky výsledky. To můžeme vidět na obrázku 1.2. Výsledky pak aplikace odešle do aplikace určené pro jejich publikování – 1.2.3 Soaring Spot.

Problém s původními stránkami tcupu je ten, že neexistuje integrace se SeeYou Competition. Například seznam zaregistrovaných závodníků se musí vždy přepisovat manuálně, to samé platí u vyhodnocování výsledků, kdy se každý let musí stáhnout ručně a vzniká tím velká nepřehlednost. Oba tyto problémy mám na nových stránkách vyřešeny.

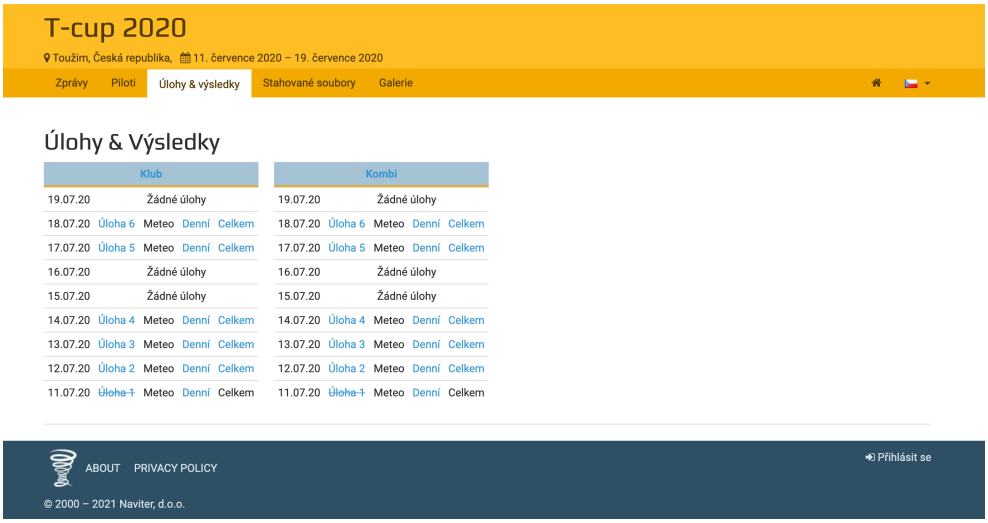


Obrázek 1.2: Vyhodnocování výsledků v SeeYou Competition

1.2.3 Soaring Spot

Soaring Spot je platforma pro správu soutěží opět od Naviteru. [4] Organizátoři zde zakládají soutěže, pro každý soutěžní den publikují úlohy a výsledky. Soutěžící pak ve své třídě letí stejnou úlohu co ostatní. Úlohy i výsledky se sem synchronizují automaticky ze 1.2.2 SeeYou Competition.

Jak webová stránka SoaringSpotu vypadá můžeme vidět na obrázcích 1.3 a 1.4. Většina stránek odkáže soutěžící nebo diváky přímo na Soaring Spot, kde sledují výsledky. Takto to dělaly i původní stránky tcupu. My budeme výsledky zobrazovat přímo na stránkách společně s vizualizací letů. Pro tento účel použijeme veřejnou API, kterou Soaring Spot poskytuje.



Obrázek 1.3: Úlohy a výsledky v Soaring Spotu

1.2.4 tcup 2019

Největším českým portálem na poli plachtařských soutěží je gliding.cz. Najdeme zde seznam všech soutěží v České republice. [5]

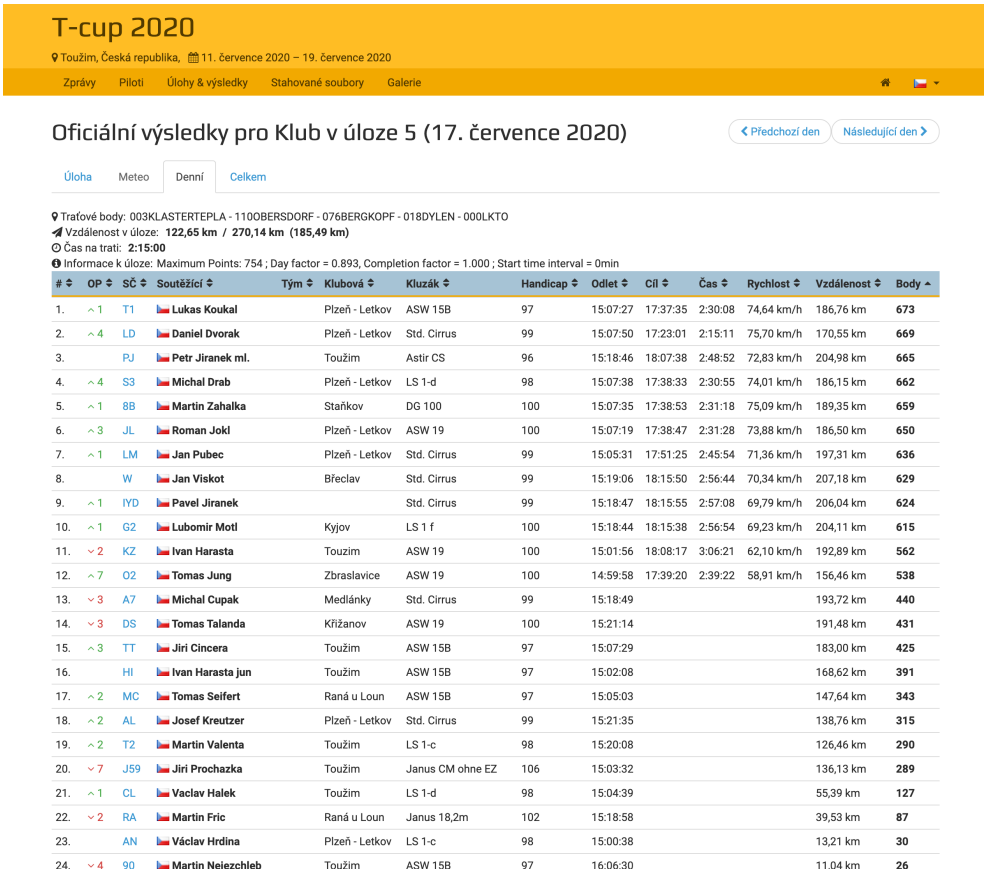
Řada organizátorů používá stránky vygenerované přímo tímto portálem, stejně jako ji používali organizátoři soutěže *tcup* [1] od roku 2009 do 2019. Na obrázku 1.5 můžeme vidět jak vypadala stránka v roce 2019.

Stránka slouží jako vstupní portál pro závodníky, kteří se na závod zaregistrují, vyplní své osobní a další údaje potřebné pro soutěž. Jsou informování pomocí novinek, mohou si stahovat potřebné soubory. V samotném průběhu soutěže se odkazem dostanou na 1.2.3 SoaringSpot pro sledování výsledků. Dále přes sekci „Odeslat IGC“ uploadují své IGC soubory – organizátoři je využijí pro vyhodnocení soutěže.

V dnešní době přechází většina na vlastní řešení zejména z možnosti implementace více funkcionalit. Organizátoři soutěže *tcup* mě s tímto návrhem oslovili a díky tomu mohla tato práce vzniknout. Nejsem sám, kdo se o vlastní řešení snaží, můžeme to vidět u ostatních soutěží, zejména když porovnáme nedávné soutěže s těmi staršími. [5] V mé práci chci všechny existující funkcionality zachovat a přidat další.

1.3 Analýza požadavků

Pro úspěšné dokončení projektu je nutné zanalyzovat požadavky. Tyto požadavky musí nasazená verze aplikace splňovat. Požadavky budu kategorizovat na funkční a nefunkční.[6]



T-cup 2020

Toužim, Česká republika, 11. července 2020 – 19. července 2020

Zprávy Piloti Úlohy & výsledky Stahované soubory Galerie

Oficiální výsledky pro Klub v úloze 5 (17. července 2020)

Předchozí den Následující den

Úloha Meteo Denní Celkem

Traťové body: 003KLASTERTEPLA - 110OBERSDORF - 076BERGKOPF - 018DYLEN - 000LKTO
Vzdálenost v úloze: **122,65 km / 270,14 km (185,49 km)**
Čas na trati: **2:15:00**
Informace k úloze: Maximum Points: 754 ; Day factor = 0.893, Completion factor = 1.000 ; Start time interval = 0min

#	OP	SČ	Soutěžící	Tým	Klubová	Kluzák	Handicap	Odlet	Cíl	Čas	Rychlost	Vzdálenost	Body
1.	^ 1	T1	**Lukas Koukal**		Plzeň - Letkov	ASW 15B	97	15:07:27	17:37:35	2:30:08	74,64 km/h	186,76 km	**673**
2.	^ 4	LD	**Daniel Dvorak**		Plzeň - Letkov	Std. Cirrus	99	15:07:50	17:23:01	2:15:11	75,70 km/h	170,55 km	**669**
3.		PJ	**Petr Jiranek ml.**		Toužim	Astir CS	96	15:18:46	18:07:38	2:48:52	72,83 km/h	204,98 km	**665**
4.	^ 4	S3	**Michal Drab**		Plzeň - Letkov	LS 1-d	98	15:07:38	17:38:33	2:30:55	74,01 km/h	186,15 km	**662**
5.	^ 1	8B	**Martin Zahalka**		Staňkov	DG 100	100	15:07:35	17:38:53	2:31:18	75,09 km/h	189,35 km	**659**
6.	^ 3	JL	**Roman Jokl**		Plzeň - Letkov	ASW 19	100	15:07:19	17:38:47	2:31:28	73,88 km/h	186,50 km	**650**
7.	^ 1	LM	**Jan Pubec**		Plzeň - Letkov	Std. Cirrus	99	15:05:31	17:51:25	2:45:54	71,36 km/h	197,31 km	**636**
8.		W	**Jan Viskot**		Břeclav	Std. Cirrus	99	15:19:06	18:15:50	2:56:44	70,34 km/h	207,18 km	**629**
9.	^ 1	IYD	**Pavel Jiranek**			Std. Cirrus	99	15:18:47	18:15:55	2:57:08	69,79 km/h	206,04 km	**624**
10.	^ 1	G2	**Lubomir Motl**		Kyjov	LS 1 f	100	15:18:44	18:15:38	2:56:54	69,23 km/h	204,11 km	**615**
11.	v 2	KZ	**Ivan Harasta**		Touzim	ASW 19	100	15:01:56	18:08:17	3:06:21	62,10 km/h	192,89 km	**562**
12.	^ 7	O2	**Tomas Jung**		Zbraslavice	ASW 19	100	14:59:58	17:39:20	2:39:22	58,91 km/h	156,46 km	**538**
13.	v 3	A7	**Michal Cupak**		Medlánky	Std. Cirrus	99	15:18:49				193,72 km	**440**
14.	v 3	DS	**Tomas Talanda**		Křižanov	ASW 19	100	15:21:14				191,48 km	**431**
15.	^ 3	TT	**Jiri Cincera**		Toužim	ASW 15B	97	15:07:29				183,00 km	**425**
16.		HI	**Ivan Harasta jun**		Toužim	ASW 15B	97	15:02:08				168,62 km	**391**
17.	^ 2	MC	**Tomas Seifert**		Raná u Loun	ASW 15B	97	15:05:03				147,64 km	**343**
18.	^ 2	AL	**Josef Kreutzer**		Plzeň - Letkov	Std. Cirrus	99	15:21:35				138,76 km	**315**
19.	^ 2	T2	**Martin Valenta**		Toužim	LS 1-c	98	15:20:08				126,46 km	**290**
20.	v 7	J59	**Jiri Prochazka**		Toužim	Janus CM ohne EZ	106	15:03:32				136,13 km	**289**
21.	^ 1	CL	**Vaclav Halek**		Toužim	LS 1-d	98	15:04:39				55,39 km	**127**
22.	v 2	RA	**Martin Fric**		Raná u Loun	Janus 18,2m	102	15:18:58				39,53 km	**87**
23.		AN	**Václav Hrdina**		Plzeň - Letkov	LS 1-c	98	15:00:38				13,21 km	**30**
24.	v 4	90	**Martin Neiezchleb**		Toužim	ASW 15B	97	16:06:30				11.04 km	**26**

Obrázek 1.4: Denní výsledky v Soaring Spotu

1.3.1 Funkční požadavky

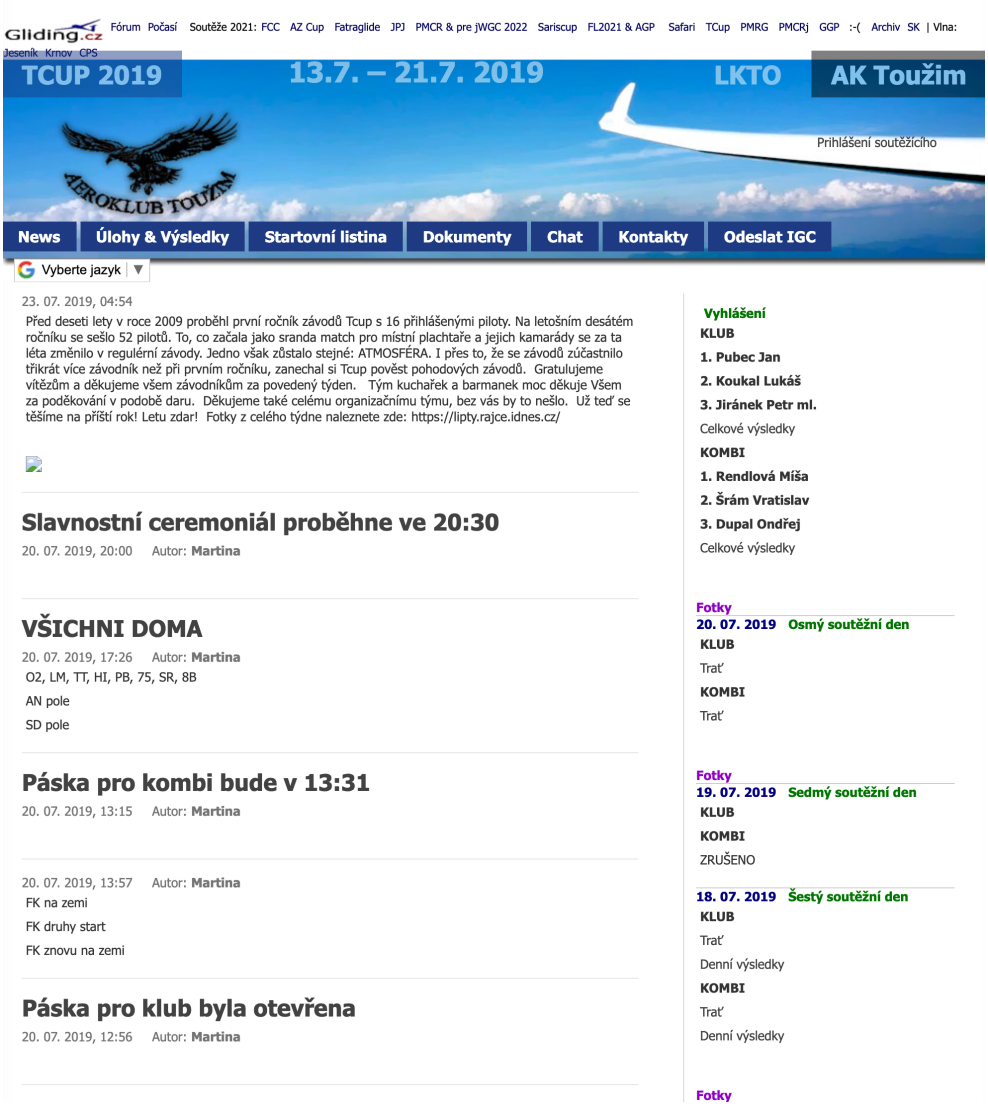
Funkční požadavky popisují chování systému za konkrétních podmínek a zahrnují funkce produktu, které se musí do řešení přidat. Jednotlivé požadavky mohou být manipulace s daty, interakce uživatele, výpočet, business proces nebo jakákoliv jiná specifická funkcionalita.[7]

F1: Registrace a přihlášení

Aplikace bude umožňovat uživatelům si v systému vytvořit účet. Po vytvoření účtu se budou moci do aplikace přihlásit.

F2: Obnova, změna hesla a údajů

Aplikace bude umožňovat uživatelům změnu hesla nebo při zapomenutí hesla jeho obnovu přes email. Přihlášený uživatel si bude moci změnit své osobní údaje, jako je jméno nebo email.



Obrázek 1.5: Existující stránka soutěže tcup z roku 2019

F3: Vytvoření a úprava přihlášky

Uživatelé si budou moci vytvořit přihlášku na soutěž. Po vytvoření si budou moci přihlášku upravovat.

F4: Startovní listina

Uživatelé si budou moci zobrazit startovní listinu. Uvidí tak všechny uživatele co si podali přihlášku do soutěže. Program bude umožňovat administraci startovní listiny. Aplikace bude umožňovat export startovní listiny do programu SeeYou Competition v potřebném formátu.

F5: Novinky

Aplikace bude umožňovat zobrazení publikovaných novinek, jak v sekci „Novinky“, tak na hlavní stránce. Systém bude umožňovat spravovat novinky, jako je přidání, mazání, tak i notifikování emailem.

F6: Dokumenty

Aplikace bude umožňovat uživatelům stahovat nahrané dokumenty, stejně tak nahrávat dokumenty nové.

F7: Statusy soutěžících

Aplikace bude umožňovat uživatelům sledovat soutěžící a jejich status letu. Také umožňuje správu těchto statusů soutěžících pro každý soutěžní den.

F8: IGC soubory

Aplikace umožňuje nahrávání IGC souborů (záznamů letu). Dalé umožňuje spravovat tyto soubory.

F9: Zobrazení výsledků

Uživatelé budou mít možnost si zobrazit výsledky soutěže, jak v průběhu, tak po skončení. Budou moci sledovat jak celkové výsledky podle soutěžních tříd, tak výsledky rozdělené i podle soutěžního dne.

F10: Vizualizace letů

V denních výsledcích bude možno zobrazit let daného závodníka. Celý průběh letu bude být moci přehrán na mapě.

F11: Synchronizace výsledků

V aplikaci budou vždy synchronizovaná aktuální data s daty ze Soaring Spotu. Těmito daty budou celkové výsledky, denní výsledky a identifikátor potřebný pro přehrání letů.

F12: Překlad do cizích jazyků
Aplikace bude přeložena do českého jazyka. Cizinci budou moci využít překladu do angličtiny.

1.3.2 Nefunkční požadavky

N1: Platforma
Aplikace musí být webová, pro snadný a rychlý přístup uživatelů k ní.

N2: Testování
Funkčnost aplikace musí být otestována před nasazením do reálného provozu.

N3: Dostupnost
Aplikace musí být nasazená v reálného provozu.

N4: Spolehlivost
Aplikace musí fungovat spolehlivě a bez fatálních chyb. Větší důraz na spolehlivost se klade při konání soutěže.

N5: Výkon
Aplikace musí být rozumně responzivní, načítání dílčích dat nesmí trvat déle než 10 vteřin.

N6: Uživatelská přívětivost
Aplikace musí být přehledná a pochopitelná nejen pro technicky znalé ale i pro starší generaci.

1.4 Role uživatelů

Uživatele lze rozdělit do 3 skupin. Podle těchto skupin budu popisovat 1.5 Případy užití.

Nepřihlášený uživatel, který je buď divákem nebo soutěžícím, který systém používá nepřihlášen

Přihlášený soutěžící, který systém používá přihlášen

Administrátor uživatel, který má administrátorská práva a spravuje soutěž (může být zároveň soutěžícím)

1.5 Případy užití

Existuje hodně definic pojmu „Use case“. Jednou z nich je, že případ užití znamená popis souboru akcí, které systém provádí a které poskytují uživateli (aktérovi) užitečný a zároveň pozorovatelný výsledek. [8]

Případy užití by měly obsahovat:

- Popis případu užití
- Aktér
- Tok událostí

UC1: Registrace
Popis: Nový uživatel si může vytvořit účet pomocí registračního formuláře.
Aktér: Nepřihlášený
Tok událostí: Po kliknutí na tlačítko *Registrace* v navigaci v pravém horním rohu se uživateli zobrazí registrační formulář. Po vyplnění *Jména*, *Příjmení*, *Emailu* a *Hesla* formulář odešle. Aplikace zvaliduje odeslaná data a uživatele rovnou přihlásí. To můžeme vidět opět v pravém horním rohu v navigaci, kdy zmizela tlačítka *Přihlásit se* a *Registrovat* a objevilo se jméno uživatele. V případě chyby je uživateli zobrazena chybová hláška a uživatel zůstává nezaregistrován a nepřihlášen.

UC2: Přihlášení
Popis: Uživatel, který se dříve zaregistroval, se může přihlásit.
Aktér: Nepřihlášený
Tok událostí: Po kliknutí na tlačítko *Přihlásit se* v navigaci v pravém horním rohu se uživateli zobrazí přihlašovací formulář. Po vyplnění *Emailu* a *Hesla* formulář odešle. Aplikace ověří platnost těchto údajů a uživatele přihlásí. Opět zmizí tlačítka *Přihlásit se* a *Registrovat* a objeví se jméno uživatele. V případě chyby je uživateli zobrazena chybová hláška a uživatel zůstává nepřihlášený.

UC3: Obnova hesla
Popis: Uživatel se nemůže přihlásit, protože zapomněl své heslo. Obnoví si ho.
Aktér: Nepřihlášený
Tok událostí: Po kliknutí na tlačítko *Přihlásit se* a poté na odkaz *zapomenuté heslo* se uživateli zobrazí formulář. Po vyplnění *Emailu* formulář odešle. Aplikace ověří platnost emailu a uživateli zobrazí, že mu byl odeslán email s obnovovacím odkazem. V případě chyby je uživateli zobrazena chybová hláška a nic se dalšího se nestane.

Po kliknutí na odkaz v zaslaném emailu se uživatel vrátí zpátky do aplikace a zobrazí se mu formulář s obnovením hesla. Po vyplnění *Hesla* a *Hesla znovu* formulář odešle. Uživateli se zobrazí hláška, že se úspěšně podařilo změnit heslo. Uživatel může pokračovat UC2.

UC4: Vytvoření a úprava přihlášky
Popis: Uživatel si může vytvořit přihlášku. Tu si pak může později zobrazit i upravit.
Aktér: Přihlášený
Tok událostí: Po kliknutí na tlačítko *Přihláška* se uživateli zobrazí formulář pro vyplnění přihlášky. Po vyplnění všech polí formulář odešle. Aplikace ověří platnost všech údajů a přihlášku přidá do veřejné startovní listiny. V případě chyby má uživatel možnost údaje opravit a odeslat znovu.

Uživatel svoji vyplněnou přihlášku vidí a při kliknutí na tlačítko *upravit přihlášku* může údaje změnit. Aplikace se chová stejně jako při vytváření.

UC5: Startovní listina
Popis: Uživatel si zobrazí startovní listinu.
Aktér: jakýkoliv
Tok událostí: Po kliknutí na tlačítko *Startovní listina* se uživateli zobrazí stránka. Aplikace najde v databázi všechny přihlášky a seskupí je podle soutěžních tříd. Uživateli se pro každou soutěžní třídu zobrazí tabulka s přihláškami závodníků.

UC6: Administrace startovní listiny
Popis: Administrátor soutěže může měnit přihlášky všem závodníkům.
Aktér: Administrátor
Tok událostí: po zobrazení startovní listiny z UC5 klikne uživatel na přihlášku jakéhokoliv soutěžícího tlačítkem *úprava přihlášky*. To mu zobrazí stránku stejně jako v UC4, kde ji může editovat.

UC7: Export startovní listiny
Popis: Administrátor soutěže může exportovat startovní listinu do programu SeeYou Competition.
Aktér: Administrátor
Tok událostí: Po zobrazení startovní listiny z UC5 klikne uživatel na tlačítko *export přihlášek*. Tato akce mu stáhne soubor se všemi přihláškami soutěže, ale v CSV formátu určeném pro import do programu SeeYou Competition. Následně importuje tento seznam v SeeYou Competition a nahraje piloty do soutěže. V tomto programu poté může kliknout na *Publikovat do SoaringSpotu*, tím se seznam pilotů nahraje na Soaring Spot. Tento krok je nutný pro pozdější vyhodnocování.

UC8: Novinky
Popis: Uživatel si zobrazí novinky soutěže.
Aktér: jakýkoliv
Tok událostí: Na hlavní stránce vidí uživatel novinky soutěže. Ty samé novinky také uvidí po kliknutí na tlačítko *Novinky*.

UC9: Administrace novinek
Popis: Administrátor může přidávat, odebírat novinky a notifikovat o nich e-mailem.
Aktér: Administrátor
Tok událostí: Po kliknutí na stránku *Novinky* se kromě publikovaných novinek zobrazí nad nimi administrátorovi formulář pro přidání novinky. Po vyplnění *Nadpisu* a *Textu*, může ještě zaškrtnout, jestli budou o novince notifikováni zaregistrovaní uživatelé emailem. Po odeslání se přidá nová novinka a zároveň systém (ne)odešle email všem uživatelům. Vedle všech novinek je tlačítko *smazat novinku*. Po kliknutí na něj systém danou novinku smaže.

UC10: Zobrazení a stáhnutí dokumentu
Popis: Uživatel si může stáhnout libovolný dokument.
Aktér: jakýkoliv
Tok událostí: Po kliknutí na tlačítko *Dokumenty* se uživateli načtou všechny nahrané dokumenty. Po vybrání jednoho z nich si může tlačítkem *stáhnout* daný dokument stáhnout.

UC11: Administrace dokumentů
Popis: Administrátor může mazat a nahrávat nové dokumenty.
Aktér: Administrátor
Tok událostí: Po kliknutí na tlačítko *Dokumenty* se uživateli načtou všechny nahrané dokumenty. Po vybrání jednoho z nich ho může tlačítkem *smazat* smazat.

Také může vybrat soubor pomocí tlačítka *Vybrat soubor* a kliknutím *Nahrát* nahraje nový dokument. Systém informace o dokumentu nahraje do databáze a soubor uloží do AWS S3 Storage, pro případné pozdější stahování. Uživatelům si tento soubor společně s ostatními mohou stáhnout UC10.

UC12: Zobrazení statusů soutěžících
Popis: Uživatel se může podívat na statusy soutěžících
Aktér: jakýkoliv
Tok událostí: Po kliknutí na tlačítko *Statusy soutěžících* se uživateli zobrazí tabulka z posledního soutěžního dne. Uživatel si vybere v nabídce soutěžní den. Po kliknutí na daný den se načte tabulka se statusy. Pro každého soutěžího jde vidět, jestli je *na zemi*, *letí* nebo jestli přistál *doma* na domovském letišti nebo *na poli* mimo letiště.

UC13: Administrace statusů soutěžících
Popis: Uživatel může dělat změny ve stavech statusů soutěžících.
Aktér: Administrátor
Tok událostí: Po kliknutí na tlačítko *Statusy soutěžících* se uživateli zobrazí stejná tabulka z posledního soutěžního dne jako v UC12. Uživatel může změnit

soutěžní den v té samé nabídce. S tím rozdílem, pro každý stav je zde tlačítko se změnou stavu a vybarvený je aktuální stav. Po stisku tlačítka se hned změní stav pilota.

UC14: Nahrání IGC souboru
Popis: Uživatel nahraje IGC soubor do systému
Aktér: jakýkoliv
Tok událostí: Po stisku tlačítka *Odeslat IGC* se uživateli zobrazí formulář pro nahrání IGC souboru. Uživatel z nabídky vybere pilota, kterému soubor patří a vybere soubor, který chce nahrát. Po stisku tlačítka *Nahrát* se tyto data odešlou do systému, kde se zpracují. Kontextuální informace o souboru se nahrají do databáze a soubor se uloží do AWS S3 Storage, pro pozdější stahování.

UC15: Administrace IGC souborů
Popis: Uživatel má možnost stahovat IGC soubory a měnit jejich stav.
Aktér: Administátor
Tok událostí: Po stisku tlačítka *Odeslat IGC* se uživateli kromě formuláře pro nahrání souboru zobrazí sekce *Stáhnout IGC*. Zde může vybrat z nabídky soutěžní den. Pro tento soutěžní den se mu v tabulce zobrazí nahrané IGC soubory z UC14. Uživatel stáhne soubor a může k němu dát příznak *staženo* nebo později *zpracováno*. To podle toho, jestli soubory spároval s lety v aplikaci SeeYou Competition.

UC16: Zobrazení výsledků
Popis: Uživatel si zobrazí výsledky soutěže.
Aktér: jakýkoliv
Tok událostí: Po stisku tlačítka *Výsledky* se uživateli zobrazí celkové výsledky pouze prvních tří umístěných závodníků v každé soutěžní třídě. Po stisku tlačítka *Celkové výsledky* u dané třídy se uživateli zobrazí nová stránka s celkovými výsledky všech závodníku pro danou třídu. Po stisku tlačítka *Denní výsledky* se mu zobrazí denní výsledky posledního soutěžního dne první třídy. Uživatel si může vybrat kombinaci kokrétního dne a třídy a po kliknutí na toto tlačítko se mu zobrazí dané denní výsledky.

UC17: Vizualizace letů
Popis: Uživatel si může přehrávat všechny lety závodníků v přehrávači letů na mapě.
Aktér: jakýkoliv
Tok událostí: Na stránce zobrazení *Denních výsledků* z UC16 vidí uživatel tabulku denních výsledků pro konkrétní soutěžní den a třídu. U každého pilota je vpravo tlačítko *přehrát let*. Po kliknutí na toto tlačítko se uživateli maxi-

malizuje přehrávač letů. Může se kouknout na to, co ho zajímá a pak buď přehrávač minimalizovat nebo vybrat let jiného pilota.

UC18: Synchronizace výsledků
Popis: Aplikace periodicky synchronizuje výsledky.
Aktér: Aplikace
Tok událostí: Server automaticky opakovaně synchronizuje výsledky všech soutěžních dnů a soutěžních tříd. Zaručuje to aktuálnost informací na stránce. Systém udělá HTTP dotaz na Soaring Spot API, zpracuje výsledky a uloží do databáze. Z této databáze pak posílá data pro UC16 *Zobrazení výsledků* a pro UC17 *Vizualizaci letů*.

UC19: Překlad stránky
Popis: Uživatel může aplikace přeložit z češtiny do angličtiny.
Aktér: jakýkoliv
Tok událostí: Po kliknutí na *vlajku* v pravém horním rohu se zobrazích seznam podporovaných lokalizací. Po výběru jazyka se celá stránka interaktivně bez obnovení stránky přeloží. Stránka podporuje český a anglický jazyk.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
UC1	X											
UC2	X											
UC3		X										
UC4			X									
UC5				X								
UC6			X	X								
UC7				X								
UC8					X							
UC9					X							
UC10						X						
UC11						X						
UC12							X					
UC13							X					
UC14								X				
UC15								X				
UC16									X			
UC17										X		
UC18									X	X	X	
UC19												X

Tabulka 1.1: Pokrytí funkčních požadavků

Kapitola 2

Návrh

2.1 Architektura klient-server

V návrhu mé aplikace využiji architekturu klient-server. Jedná se o dvouvrstvou, resp. třívrstvou architekturu, ve které první vrstva – klient zajišťuje uživatelské rozhraní se zobrazovací logikou, druhá vrstva – server pak zajišťuje aplikační logiku a poslední vrstva – databáze, poskytuje data serveru. Klient překládá uživatelův požadavek tak, aby byl srozumitelný serveru a reverzně pak klient překládá odpověď od serveru, aby jí rozuměl uživatel. [9]

Klient si vyžádá nějaká data nebo službu a server je klientu vrátí. V této architektuře jsou všechny požadavky a odpovědi doručeny po síti, jak je znázorněno na obrázku 2.1. Můžeme v něm navíc vidět komunikaci serveru a databáze, serveru a úložiště, ale také i komunikaci serveru s externí Soaring Spot API. Ta v návrhu aplikace nesmí chybět.

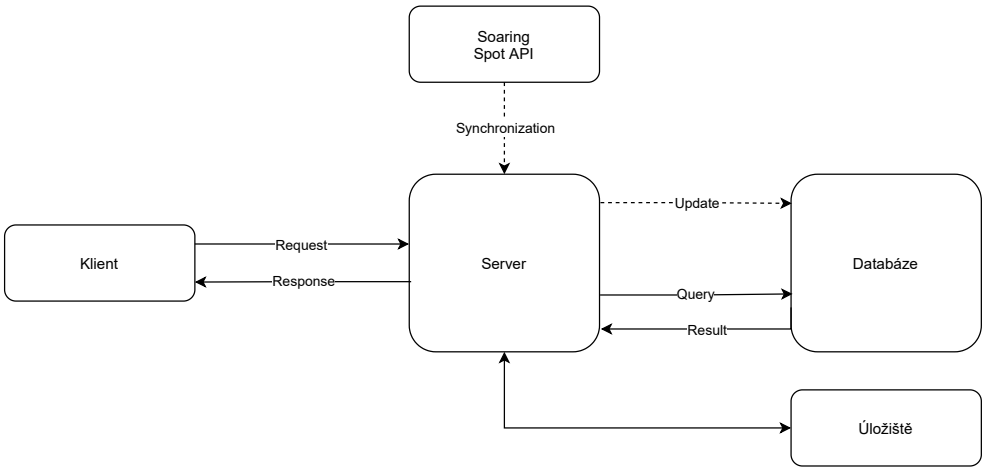
2.1.1 Klient

Jako klienta naší architektury použijeme webovou aplikaci. Se serverem bude komunikovat pomocí HTTP REST API. Bude se starat o veškerou zobrazovací logiku.

2.1.2 Server

Aby mohl klient se serverem komunikovat, musí server poskytovat API. Existují dva nejpopulárnější paradigmata pro návrh API. My použijeme REST API, ale alternativou by mohlo být také novější GraphQL API.

REST je architektura rozhraní, navržená pro přístup ke zdrojům, které jsou identifikované pomocí URI. Nejpoužívanějším protokolem je HTTP, nicméně možnosti u volby protokolu jsou neomezené. Jedním z hlavních principů REST je bezestavovost. To znamená, že dotaz musí obsahovat všechny infor-



Obrázek 2.1: Architektura

mace potřebné k jeho pochopení a zpracování a žádná kontextová informace nesmí být na serveru uložena. O stav „session“ se tedy musí starat klient.

Pro přístup ke zdrojům REST specifikuje pouze a jen CRUD operace, viz Tabulka 2.1. [10]

HTTP metoda	CRUD operace	význam
POST	Create	vytvoření dat
GET	Read	čtení dat (nesmí žádná data měnit)
PUT	Update	update dat
DELETE	Delete	smazání dat

Tabulka 2.1: CRUD operace

2.1.3 Databáze

Databáze je poslední vrstvou architektury. Poskytuje data pro server. Zde si můžeme vybrat mezi SQL (relačními) a No-SQL (ne pouze relačními) databázemi. V No-SQL nemusíme, ale můžeme definovat přesnou strukturu dat. Tyto databáze jsou ale více flexibilní pro nestrukturovaná data, což není náš případ.

Musím přiznat, že na začátku vývoje jsem se rozhodl pro No-SQL databázi, což se zdálo jako dobrý nápad, nicméně časem jsem zjistil, že výhody No-SQL databáze nevyužiji a že není třeba. V implementaci mi to ale zatím nedělalo problém, protože v dotazu nikdy nespojuji více než tři tabulky. Změna typu databáze na SQL databázi je ale určitě dobrým podnětem pro budoucí vylepšení celé aplikace.

2.1.4 Úložiště

Někam budeme muset ukládat uživateli nahrané soubory. Zvolíme nějaké cloudové řešení, viz. sekce 2.3 Použité technologie.

2.1.5 Soaring Spot API

Pro zobrazování výsledků budeme používat veřejnou API třetí strany – Soaring Spot API. Naše databáze bude pro klienty vždy sloužit jako jediný zdroj dat. Proto naší databázi musíme udržovat aktuální s daty v Soaring Spotu. Bude probíhat periodická synchronizace dat z této API a našimi daty v databázi.

2.2 Databázový model

I přes to, že používáme No-SQL databázi, můžeme nadefinovat přesnou strukturu dat. Struktura je nadefinována na obrázcích 2.2 a 2.3.

Contest tabulka pouze s jedním záznamem, kde jsou informace o soutěži

User tabulka určená pro všechny uživatele aplikace včetně administrátorů

CompetitionClass tabulka se všemi soutěžními třídami, mají soaringSpotId

CompetitionDay tabulka se všemi soutěžními dny

Region tabulka se všemi regiony, nutná pro přihlášku

GliderType tabulka se všemi typy kluzáků, nutná pro přihlášku

AccomodationType tabulka s typy ubytování, nutná pro přihlášku

Registration tabulka, kde jsou všechny přihlášky soutěžících, je propojená s konkrétním uživatelem, regionem, typem kluzáku a typem ubytování

CompetitorStatus tabulka, kde se uchovává status konkrétního soutěžího v konkrétním soutěžním dnu

DailyResult tabulka s denními výsledky jednoho soutěžícího v konkrétní soutěžní den

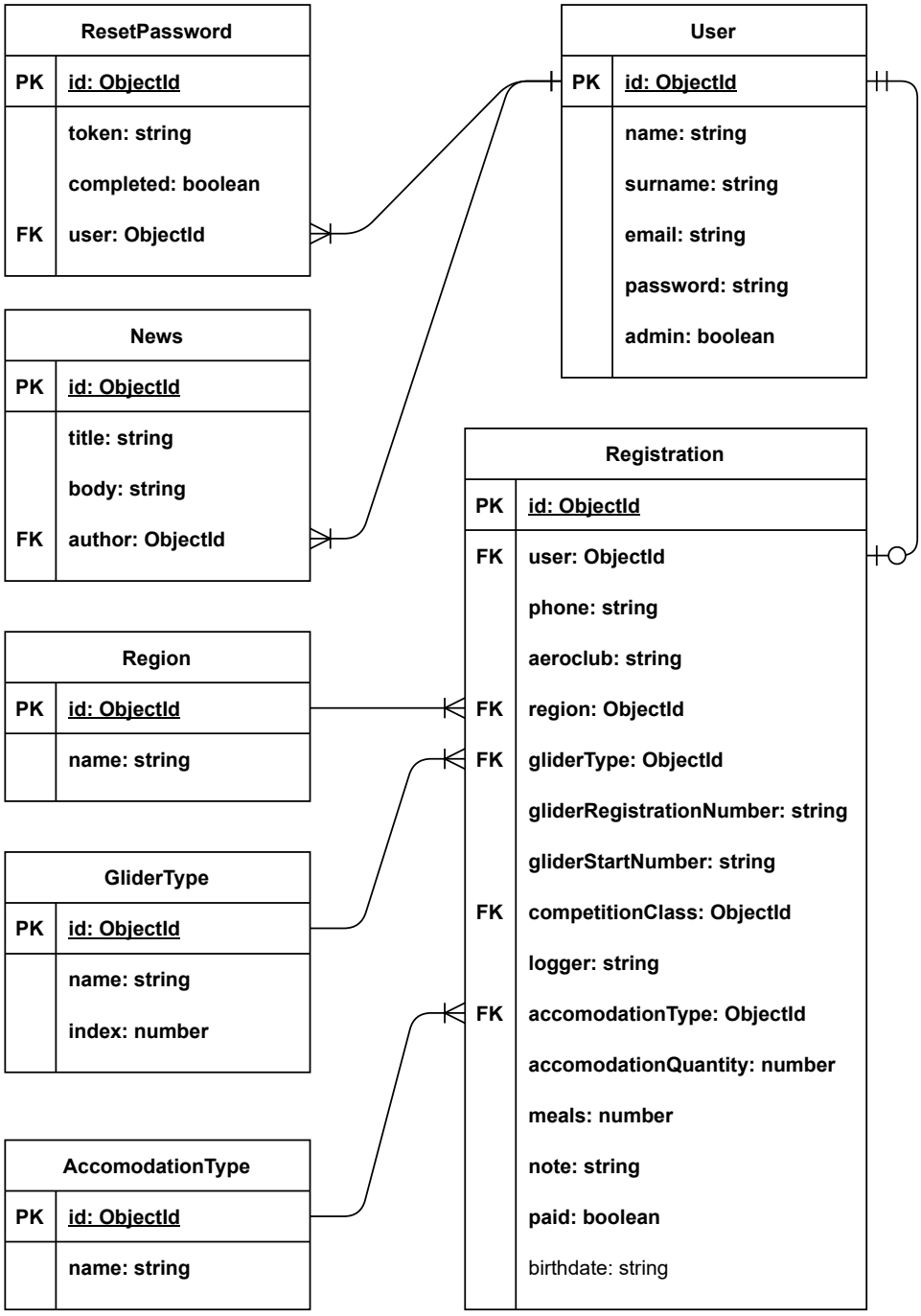
News tabulka s novinkami

Document tabulka metadat nahraných dokumentů

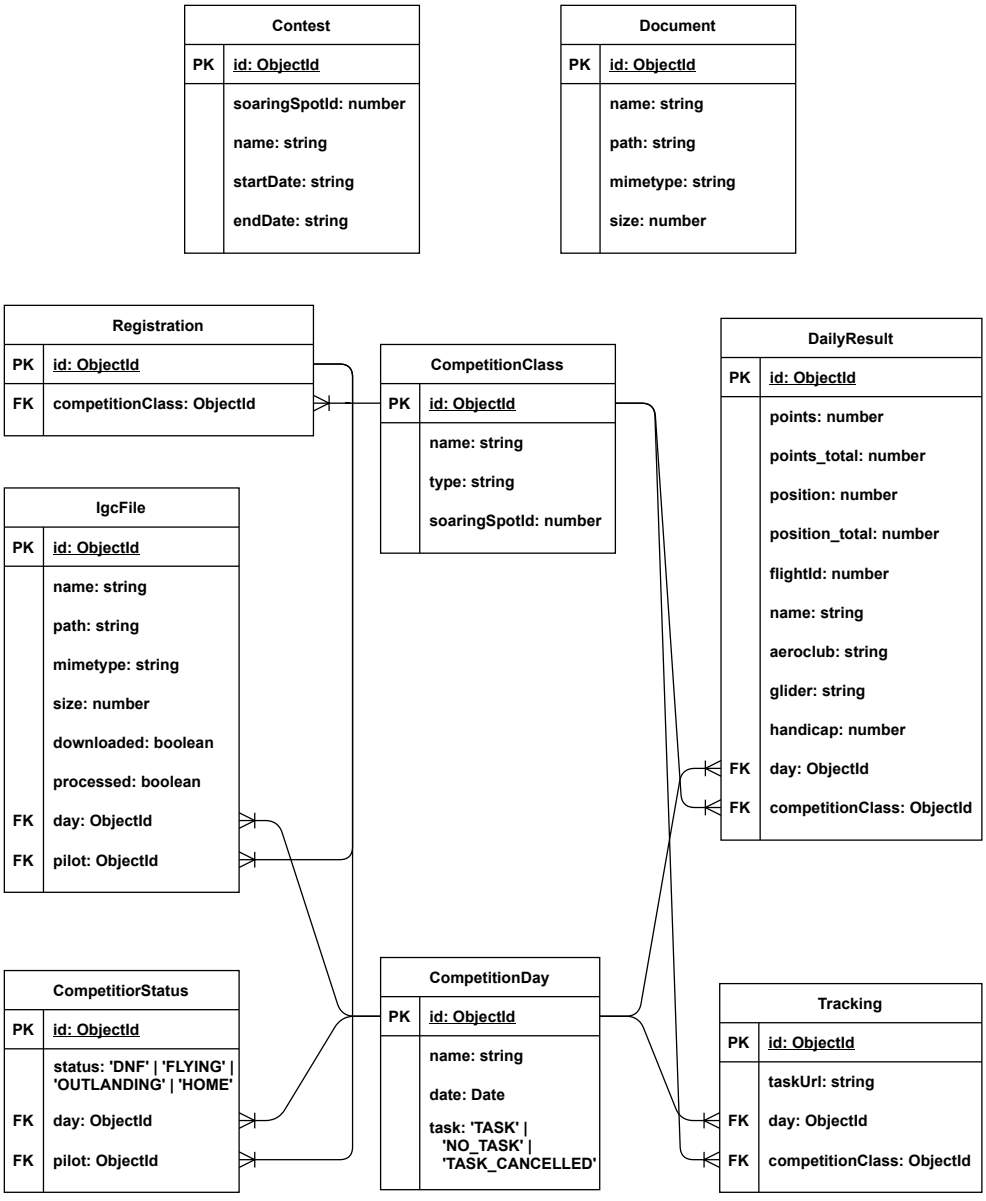
IgcFile tabulka metadat nahraných IGC souborů

ResetPassword tabulka určená pro obnovování hesel, včetně historie

Tracking tabulka pro odkaz na tracking závodní třídy v konkrétní závodní den



Obrázek 2.2: 1. část databázového modelu



Obrázek 2.3: 2. část databázového modelu

2.3 Použité technologie

2.3.1 Klientská část

Dle nefunkčního požadavku o platformě jsem se rozhodl, že klient bude webový, tedy ve formě webové aplikace. Prohlížeč nativně pracuje se zdrojovými kódy jazyka HTML, styly CSS a Javascriptem pro interpretování dynamického obsahu. Nemám tedy moc alternativ pro technologie, ve kterých bude výsledná aplikace sestavena. S jinými technologiemi prohlížeč pracovat nativně neumí. Můžu si ale např. místo JS vybrat TypeScript. Ten lze pak zkompilovat do Javascriptu pro sestavení aplikace.

Webové stránky se dají psát buď tradičním přístupem 2.3.1 MPA (Multi Page Application) nebo i modernějším stylem např. dnes se hojně využívá 2.3.1 SPA (Single Page Application).

MPA

Tradiční volbou by bylo použít MPA. Při tomto řešení se pohybujeme na stránce tak, že kliknutím na odkaz se stránka obnoví a server nám pošle novou HTML stránku. Tato stránka již obsahuje všechna data, protože ji server sestavil a poté odeslal. Zpravidla je na těchto stránkach málo interaktivity a po každém pohybu na stránce musíme dlouho čekat, než se něco stane.

SPA

Tento přístup se vyznačuje tím, že při pohybu na stránce si pokaždé nestahujeme HTML stránku s jejími CSS styly, ale stáhneme si jednu základní stránku (Single Page) a o naší interakci na stránce už se stará Javascript tím, že právě modifikuje data (DOM) na již existující, stažené stránce. [11]

SPA má spoustu výhod ale samozřejmě i nevýhod. [12] Mezi výhody patří:

- rychlost – tím, že stahujeme pouze data, která zrovna potřebujeme zvýšíme rychlost interakcí na stránce,
- spojení se serverem – server je kompletně oddělený od klienta a může se vyvíjet separátně,
- UX – aplikace je více responzivní z hlediska user experience, působí opravdu jako aplikace, spíše než klasická webová stránka.

Mezi nevýhody patří:

- rychlost – tím, že při prvotním načtění musíme stáhnout i Javascript, zvýší se nám velikost celé aplikace a tím pádem sníží rychlost načtění,
- SEO – největší nevýhodou SPA je optimalizace vyhledávacích enginů, tím že obsah jednotlivých stránek není staticky vyhledatelný, přicházíme o možnost být dobře „vidět“ ve vyhledávačích.

Tyto nevýhodou ale jdou vyřešit např. kombinací CSR (Client Side Rendering) a SSR (Server Side Rendering). CSR se vyznačuje tím, že ve stažené HTML stránce již jsou přednačtená data ze serveru, je to optimalizované právě pro vyhledávače. To je ale mimo rozsah mojí práce, takže se tím nebudu zabývat.

Po zanalyzování výhod a nevýhod jsem se rozhodl koncept SPA použít, s tím, že nevýhody SPA nejsou tak podstatné, jako by mohly být např. v jiném projektu. Problém vyhledání stránky je vyřešen, tím, že je projekt specifický a má unikátní jméno, je prvním výsledkem v Google vyhledáváči, když se napíše „tcup".

React

Pro vývoj SPA mám na výběr několik alternativ JS frameworků, zde uvedené 3 nejpoužívanější:

1. React
2. Vue
3. Angular

[13]

Nejoblíbenějším a nejpoužívanějším frameworkem je React [14] používaný např. Facebookem a Airbnb. [15] Tento framework je velmi populární, má velkou podporu, jak ze strany Facebooku tak dalších knihoven, které React obohacují o další feature. Dále je zde třeba velká šance najmout další programátory na udržování aplikace za nejlevnější cenu v porovnání s ostatními frameworky. Jazyk je odzkoušený mnoha uživateli a v poslední době je již velmi stabilní, takže je ideální volbou pro frontendový vývoj.

Vybíral jsem si mezi všemi frameworky, nicméně jsem v minulosti používal pouze React, tudíž z tohoto subjektivního důvodu jsem ho i zvolil pro práci.

React virtualizuje DOM do komponent. Probíhá mapování virtuálního DOMu do reálného DOMu pomocí vnitřních implementačních detailů. Vždy když se změní stav komponenty promítne se to v reálném čase do reálného DOMu přerendrováním. To zajišťuje onu interaktivitu aplikace. [16] [17]

Redux

Pro správu stavu aplikace se hodí použít nějaký state management nástroj. React se často používá spolu s Reduxem. [18] Do Reduxu se dají uložit načtená data (třeba ze serveru) a komponenty je pak mohou číst. Data se ukládají do tzv. store, čtou se pak pomocí selectorů. K ukládání dat se využívají tzv. akce, které se dispatch ují. Reducery se pak podle typu akce rozhodnou, co s daty udělají.

V architektuře Reduxu se využívají tyto principy:

jediný zdroj pravdy pouze jeden globální store

jednosměrnost přečtený stav jde aktualizovat pouze dispatchem akce

Další výhodou je to, že se dají data oddělit od prezentační vrstvy a veškerou logiku změny dat řešíme mimo prezentační vrstvu. [19] [20] [21] [22]

Reactstrap

Rozhodl jsem se použít Reactstrap, který poskytuje React komponenty, které obalují Bootstrap. [23] Bootstrap je framework, který obsahuje nadesignované elementy. [24] Není pak potřeba psát vlastní CSS styly, ale lze použít již naimplementovaný Bootstrap.

Alternativou by mohl být např. Material UI od Google.

Typescript

Typescript je programovací jazyk vytvořený firmou Microsoft. Jeho syntax je nadmnožinou JS, dá se tedy v TS využít stávající znalost JS. Oproti JS umí navíc např. statické typování. Vidíme to i u jiných dynamicky typovaných jazyků jako je např. Python. Přináší mnoho výhod, kterými jsou např. kontrola typů při kompilaci což odhalí mnoho chyb, na které bychom třeba přišli až později. Výstupem kompilace TS je právě JS, takže se dá použít ve všech webových projektech. [25]

Jak jsem se přibližoval konečné fázi projektu, nedalo mi to a začal jsem projekt přepisovat z JS do TS. Bohužel jsem nestihl projekt kompletně zkonvertovat, takže v implementaci jsou přítomné oba jazyky.

2.3.2 Serverová část

Pro vývoj serverové aplikační části již nejsem limitován volbou programovacího jazyka. Mám zde na výběr hodně možností od nejpoužívanější Javy, přes C#, Python nebo dokonce i Javascript. Já jsem se rozhodl pro JS, přesněji tedy NodeJS, je to jeden z nejpoužívanějších jazyků [26] a tuto volbu zvýhodňuje to, že už jej používám na frontendu. Je to velmi rychlý jazyk s obrovskou podporou knihoven a moji volbu také potvrzuje to, že firmy jako Netflix, PayPal, LinkedIn, nebo Uber tento jazyk používají právě pro jeho vynikající rychlost a škálovatelnost.

V tomto jazyku budu psát HTTP REST API, na kterou použiji framework Express. Jde o nejpoužívanější framework [27] hlavně pro jeho jednoduchost, ale zároveň flexibilitu.

Pro správu databáze používám knihovnu Mongoose pro MongoDB. Databáze je v cloudu, tedy je přístupná přes internet. [28]

Pro správu úložiště používám knihovnu Multer s rozšířením Multer-S3 pro Amazon S3 Storage. Stejně jako databáze je v cloudu. [29]

2.3.3 Databáze

Jako databázi jsem zvolil MongoDB. [30] Jde o NoSQL databázi, která se často používá v tzv. MERN stacku (kombinace MongoDB, Express, React a Node). [31] [32] Tato volba nám přijde vhod nejen při samotném vývoji, ale i později kdy budeme aplikaci nasazovat do reálného provozu.

2.3.4 Úložiště

Jako úložiště nahraných souborů jsem zvolil AWS Storage S3 od Amazonu. [33] Jde opět o jedno z nejpoužívanějších řešeních.

2.4 Soaring Spot API

Soaring Spot poskytuje data pro vývojáře formou REST API. Data poskytuje v hal+json formátu. Pro přístup do API potřebujeme vytvořit API klíče v administraci soutěže v Soaring Spotu. Po této autorizaci se můžeme doptat na informace o soutěži již přes API. Toto proběhne jednorázově. Dále budeme periodicky synchronizovat výsledky soutěže do naší databáze, viz. obrázek 2.1. [34] [35]

Kapitola 3

Implementace serveru

3.1 REST API

V této podkapitole popíšu, jak jsem implementoval REST API s ukázkami kódu.

3.1.1 Express router

Pro implementaci REST API jsem použil framework Express pro NodeJS. Když provede klient dotaz na server, router přebere zodpovědnost za správné handlování požadavku. Router se chová zároveň jako middleware, takže ho můžeme použít jako druhý argument do `app.use`, společně s prvním argumentem *cestou*. Do metod routeru jako jsou např. `get` nebo `post` a další předáváme první argument *cestu* a jako další argumenty tzv. *middlewary*. Middleware je funkce, která bere 3 argumenty:

req argument s požadavkem

res argument s odpovědí

next argument pro volání dalšího middlewaru v pořadí

[36]

Nejprve začnu ukázkou nastavení a spuštění serveru v souboru server.js. Pomocí `app.use` se definuje router pro daný prefix endpointu.

```
import express from 'express';
import config from '../config';
import version from './routes/api/version';

const app = express();

app.use('/', version);
```

```
const port = config.PORT || 5000;

app.listen(port, () => console.log(`Server started on port ${port}`));

export default app;
```

Následuje soubor, ve kterém je nadefinován samotný router, zde se už specifikuje daná HTTP metoda s danou URI a následuje jeden až mnoho middlewarů pro handlování požadavků. V tomto příkladě je zde pouze jeden middleware. Vybral jsem na úvod jen takhle jednoduchý soubor, pro objasnění jak vypadá routing v Expressu.

```
import express from 'express';
import { version } from '../../../package.json';

const router = express.Router();

// pro priklad dalsi middleware
const middleware = (req, res, next) => {...}

// @route GET /
// @desc  Gets npm package version
// @access Public
router.get('/', /* middleware , */async (req, res) => {
    return res.status(200).json({
        version
    });
});

export default router;
```

3.1.2 Routes

V této sekci popíšu část nadefinovaných rout v celé API. Neuvedu všechny, protože je jich celkem 50. V tabulce 3.1 lze vidět HTTP metodu, endpoint a popis endpointu. Jako příklad jsem vybral ty nejzajímavější, v příloze C je kompletní seznam endpointů.

3.2 Autentifikace

3.2.1 Client-server

Aby měl klient možnost využít všechny funkcionality aplikace, musí mít možnost se autentifikovat. Vstupním bodem pro autentifikaci je endpoint `POST api/auth`. Ten najde podle emailu v databázi uživatele a ověří heslo oproti

HTTP metoda	URI	popis
GET	/	vrátí verzi aplikace
POST	api/users	zaregistruje nového uživatele
POST	api/auth	přihlásí uživatele
GET	api/auth/user	získá informace o uživateli
GET	api/classes	vrátí soutěžní třídy
GET	api/days	vrátí soutěžní dny
GET	api/documents	vrátí dokumenty
POST	api/registration	vytvoří novou přihlášku
PUT	api/registration	upraví přihlášku
GET	api/starting-list	vrátí startovní listinu
POST	api/igc	odešle IGC soubor
GET	api/results/total	vrátí celkové výsledky

Tabulka 3.1: API endpointy

hashi v databázi. Tato hash vznikla pomocí funkce `hashPassword`. Po ověření hesla ve funkci `checkPassword` pokračuje vygenerováním JWT tokenu ve funkci `getToken`, který obsahuje `id` uživatele. V dalších requestech client používá tento token pro autentifikaci. Tento token se ověřuje pomocí funkce `verifyToken`.

```
import bcrypt from 'bcryptjs';
import jwt from 'jsonwebtoken';
import { v4 as uuidV4 } from 'uuid';

import config from '../../config';

export const hashPassword = (password) => {
    return new Promise((resolve, reject) => {
        bcrypt.genSalt(10, (err, salt) => {
            if (err) reject(err);
            bcrypt.hash(password, salt, async (err, hash) => {
                if (err) reject(err);
                resolve(hash);
            });
        });
    });
};

export const checkPassword = (password, hash) => {
    return bcrypt.compare(password, hash);
};

export const getToken = (id) => {
    return new Promise((resolve, reject) => {
        jwt.sign({ id }, config.JWT_SECRET, { expiresIn: 3600 }, (err,
```

```
            token) => {
            if (err) reject(err);
            resolve(token);
        });
    });
};

export const verifyToken = (token) => {
    return jwt.verify(token, config.JWT_SECRET);
};
```

3.2.2 Server-Soaring Spot

Ještě je potřeba popsat jak funguje autentifikace směrem od serveru k Soaring Spot API. V administraci Soaring Spotu získáme údaje nutné pro vypočtení podpisu pro každý požadavek. Těmi jsou `SOARING_SPOT_CLIENT_ID` a `SOARING_SPOT_SECRET`. Podpis se vypočítává takto:

```
import crypto from 'crypto';
import config from '../../../config';

const { SOARING_SPOT_CLIENT_ID, SOARING_SPOT_SECRET } = config;

const nonce = 'abcdefghijklmnopqrstvwxyz';
const created = new Date().toISOString();

const signature = crypto
    .createHmac('sha256', SOARING_SPOT_SECRET)
    .update(nonce + created + SOARING_SPOT_CLIENT_ID)
    .digest('base64');

const authorization = `http://api.soaringspot.com/v1/hmac/v1
    ClientID="${SOARING_SPOT_CLIENT_ID}", Signature="${signature}",
    Nonce="${nonce}", Created="${created}"`;

const headers = {
    Authorization: authorization
};
```

Tuto vypočtenou `Authorization` hlavičku vkladám ke každému requestu.

3.3 Přístup do databáze

Na přístup do databáze používám knihovnu Mongoose. Je to knihovna pro objektové modelování a dotazování do MongoDB pro NodeJS. Do databáze se připojuje pomocí URI takto:

```
mongoose.connect(config.MONGO_URI, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useCreateIndex: true,
    reconnectTries: Number.MAX_VALUE,
    reconnectInterval: 1000
});
```

Pro nadefinování *mongoose modelu* tabulky je potřeba vytvořit schéma:

```
import mongoose from 'mongoose';
import timestamp from 'mongoose-timestamp';

const Schema = mongoose.Schema;

const UserSchema = new Schema({
    name: {
        type: String,
        required: true,
        trim: true
    },
    surname: {
        type: String,
        required: true,
        trim: true
    },
    email: {
        type: String,
        required: true,
        unique: true,
        trim: true
    },
    password: {
        type: String,
        required: true
    },
    admin: {
        type: Boolean,
        required: true,
        default: false
    }
});

UserSchema.plugin(timestamp);

const User = mongoose.model('user', UserSchema);

export default User;
```

Příkladem dotazu do databáze pak je např. toto:

```
import User from '../../models/User';
import Registration from '../../models/Registration';

const user = await User.findById(userId);

const registrations = await Registration.find({})
    .populate('user', '-password')
```

3.4 Upload souborů

Pro upload souborů používám další middleware. Využívá Amazon S3 úložiště přes knihovnu multer-S3 s knihovnou aws-sdk.

```
import S3 from 'aws-sdk/clients/s3';
import multer from 'multer';
import multerS3 from 'multer-s3';

import config from '../../config';

const s3 = new S3({
    secretAccessKey: config.AWS_SECRET,
    accessKeyId: config.AWS_ACCESS_KEY_ID,
    region: 'eu-central-1'
});

const storage = multerS3({
    s3: s3,
    bucket: 'tcup-documents',
    acl: 'public-read',
    metadata: (req, file, cb) => {
        cb(null, { fieldname: file.fieldname });
    },
    key: (req, file, cb) => {
        cb(null, file.originalname);
    }
});

const limits = {
    fileSize: 1024 * 1024 * 20
};

const upload = multer({ storage, limits });

export default upload;
```

3.5 Synchronizace výsledků

Na synchronizaci výsledků používám skript, který se autentifikuje, z databáze si zjistí soaringSpotId každé třídy a poté se dotáže na výsledky jednotlivé třídy ze SoaringSpotu.

```
const getSoaringSpotResult = async (competitionClass) => {
    const config = {
        headers
    };

    try {
        const id = competitionClass.soaringSpotId;
        const res = await
            axios.get(`https://api.soaringspot.com/v1/classes/${id}/
            results`, config);
        const classResults =
            res.data['_embedded']['http://api.soaringspot.com/rel/
            class_results'];

        const days = classResults.map(mapDay);

        return { ...competitionClass, days };
    } catch (err) {
        catchError(err);
    }
};

const getSoaringSpotResults = async () => {
    const classes = await CompetitionClass.find({}).lean();

    const results = await Promise.all(classes.map((competitionClass)
        => getSoaringSpotResult(competitionClass)));

    return results;
};
```

Výsledky pak nahraje do databáze pomocí funkce syncResults. Tento skript se pouští opakovaně během soutěže každých 30 minut.

```
const syncResults = async () => {
    // ...
    for (const result of results) {
        const dailyResult = new DailyResult({
            ...result,
            competitionClass,
            day: competitionDay
        });
```

```
        await dailyResult.save();
    }
    // ...
};
```

3.6 Odesílání emailů

Pro odesílání emailů uživatelům používám knihovnu nodemailer. [37] Zde uvádím příklad odesílání novinek.

```
const sendEmail = (user, pass, from, to, subject, text, html) => {
    const transporter = nodemailer.createTransport({
        // ...
    });

    transporter.sendMail(mailOptions, (error, info) => {
        if (error) {
            throw error;
        }
        console.log('Email sent:', info.messageId, 'to', to);
    });
};

export const sendNewsEmail = async (title, body, author) => {
    const SEND_EMAILS_TO_ALL = config.SEND_EMAILS_TO_ALL;
    const user = config.SMTP_USER;
    const password = config.SMTP_PASSWORD;
    const from = '"tcup novinky" <noreply@tcup.cz>';
    const to = await getNewsReceivers(SEND_EMAILS_TO_ALL);
    const subject = getSubject(SEND_EMAILS_TO_ALL, title);

    const text = getNewsText(title, body, author);
    const html = getNewsHtml(title, body, author);

    sendEmail(user, password, from, to, subject, text, html);
};
```

3.7 Obnovení hesla

Pro implementaci obnovení hesla bylo potřeba udělat následující. Nejdříve se pošle email s vygenerovaným tokenem danému uživateli. Po kliknutí na odkaz v emailu uživatel vyplní nové heslo, kde server ověří platnost tokenu a heslo změní. První krok vidíme ve funkci `reset-password`. Druhý krok vidíme naopak v `reset-password/reset`.

```
// @route  POST api/auth/reset-password
// @desc   Initial password reset
// @access Public
router.post('/reset-password', async (req, res) => {
    const { email } = req.body;
    // ...

    const token = await generateToken();

    const newResetPassword = new ResetPassword({
        user,
        token
    });

    await newResetPassword.save();

    await sendResetPasswordEmail(user.email, token);

    return res.status(200).json({ msg: 'Email sent' });
});

// @route  POST api/auth/reset-password/reset
// @desc   Set a new password
// @access Private
router.post('/reset-password/reset', resetPassword, async (req, res)
    => {
    const { password } = req.body;

    // ...

    const resetPw = await ResetPassword.findOne({ token:
        req.resetPasswordToken });

    const user = await User.findById(resetPw.user);

    user.password = await hashPassword(password);
    resetPw.completed = true;

    user.save();
    resetPw.save();

    sendPasswordResetCompleteEmail(user.email);

    return res.status(200).json({ msg: 'Password was successfully
        changed' });
});
```

Kapitola 4

Implementace klienta

4.1 Redux

Redux je centrálním úložištěm dat pro celou aplikaci. Úložiště se skládá z tzv. reducerů, dílčích úložišť. Na handlování side efektů (např. HTTP requestů) používám rozšíření redux-thunk. [38] To mi umožňuje mít jako redux akce asynchronní funkce.

```
import { createStore, applyMiddleware } from 'redux'
import thunk from 'redux-thunk'
import { composeWithDevTools } from 'redux-devtools-extension'

import rootReducer from './store/rootReducer'

const initialState = {}

const middleware = [thunk]

const store = createStore(
    rootReducer,
    initialState,
    composeWithDevTools(
        applyMiddleware(...middleware)
    )
)

export default store
```

Na ukázku zde můžeme vidět authReducer určený pro uložení přihlášeného uživatele. Reducer vždy bere předchozí stav a akci, a podle typu akce vrátí nový stav immutabilně. Nemění existující stav, ale vždy vrací nový.

```
const initialState: AuthState = {
```

```
    token: sessionStorage.getItem('token'),
    isAuthenticated: false,
    isAdmin: false,
    isLoading: false,
    user: null,
}

const authReducer = (state = initialState, action: AuthAction) => {
    switch (action.type) {
        // ...

        case authActionTypes.LOGIN_SUCCESS:
        case authActionTypes.REGISTER_SUCCESS:
            sessionStorage.setItem('token', action.payload.token)
            return {
                ...state,
                ...action.payload,
                isAdmin: action.payload.user.admin,
                isAuthenticated: true,
                isLoading: false,
            }
        // ...
    }
}

export default authReducer
```

Vytvořením redux-thunk akce můžeme tuto funkci dispatchnout v React komponentě a změna dat se propíše do storu. Pro příklad uvedu funkci pro přihlášení.

```
export const login = ({ email, password }: LoginCredentials):
    ThunkAction<void, AppState, null, LoginAction> => async (
    dispatch
) => {
    const config = {
        headers: {
            'Content-Type': 'application/json',
        },
    }

    const body = JSON.stringify({ email, password })

    try {
        const res = await axios.post(`${API_ENDPOINT}/api/auth`, body,
            config)
        dispatch({ type: authActionTypes.LOGIN_SUCCESS, payload:
            res.data })
    } catch (err) {
```

```
        dispatch({
            type: authActionTypes.LOGIN_FAIL,
        })
        dispatch(returnErrors(err, authActionTypes.LOGIN_FAIL))
    }
}
```

4.2 Překlady

Celý klient podporuje překlady pomocí knihovny i18next. Překlady jsou nadefinované v JSON souboru formou klíč-hodnota. Podle zvoleného jazyka se vybere daný soubor s překlady, typicky `cs.json` nebo `en.json` a podle klíče ve funkci `t()` se daný řetězec přeloží.

```
    const { t } = useTranslation()

    return (
        <h2 className="sub-header">
            {t('Letosni tcup se bude konat')} <br />
            {t('11. 7. -19. 7. 2020 v Touzimi')}.
        </h2>
    )
```

4.3 Vizualizace letů

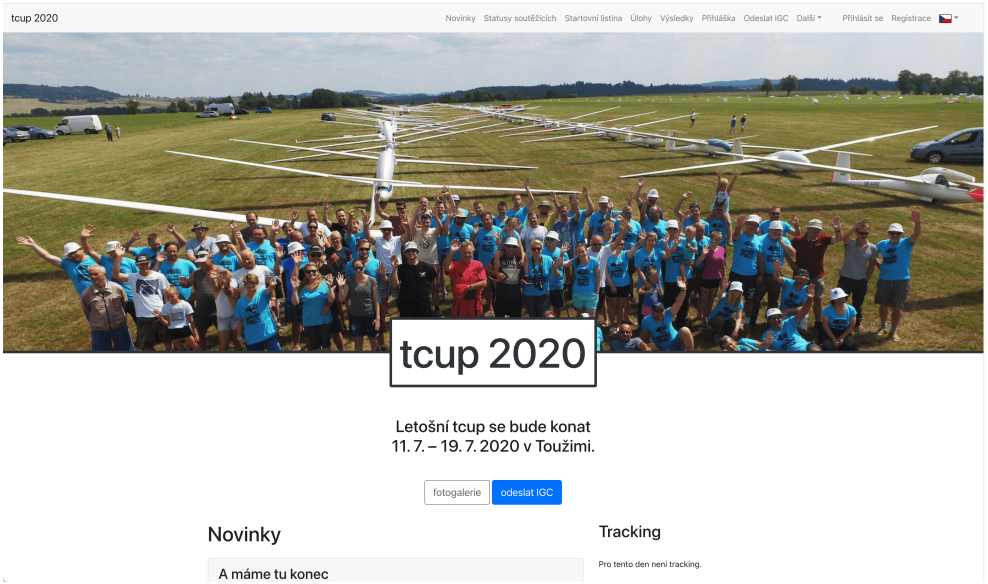
Pro vizualizaci letů používám `iframe`, ve kterém je zobrazen přehrávač letů ze stránky `seeyou.cloud/flight/public`, která je propojená se Soaring Spotem. Přehrávač se dá minimalizovat/maximalizovat a dynamicky měnit jednotlivé lety. `contestId` mám synchronizované ze SoaringSpotu a tím pádem uložené v databázi v tabulce Contest. `flightId` je u denního výsledku jednotlivého pilota opět z databáze, do které jsou tyto výsledky sychronizovány ze SoaringSpotu.

```
const SeeYouCloudVisualisation: React.FC<Props> = ({ contestId,
    flightId, maximised, setMaximised }) => {
    // ...
    <iframe
        title="seeyou.cloud"
        src={`https://www.soaringspot.com/cs/
            download-contest-flight/${contestId}-${flightId}`}
        // ...
    />
    // ...
}
```

4.4 UI

Pro zobrazení UI prvků používám Bootstrap, resp. jeho reactí knihovnu reactstrap. Pro příklad postačí pár obrázků z aplikace, zbytek snímků můžete najít v příloze D.

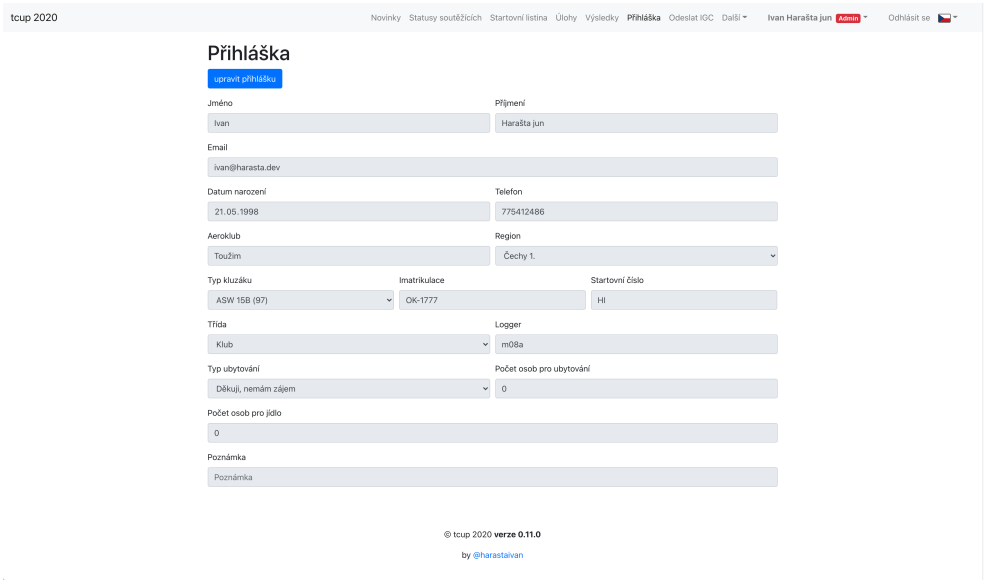
Na následujícím obrázku můžete vidět hlavní stranu aplikace. Je to jediná stránka, kde jsem napsal vlastní CSS styly, pro nastylování této velké fotografie. Nahoře je použitý Navbar z reactstrap knihovny, dále na stránce pak nadpisy a novinky ve formě karty.



Obrázek 4.1: Hlavní stránka

Na obrázku 4.2 lze vidět formulář pro úpravu přihlášky. Zde už používám čistě komponenty od reactstrap, jako ve zbytku aplikace. Po kliknutí na tlačítko upravit přihlášku lze jakékoliv pole upravit a uložit.

Na poslední obrázku 4.3 můžeme vidět stránku s denními výsledky. Nad nimi je vizualizace zrovna vybraného letu. Mapu nad výsledky lze minimalizovat nebo maximalizovat, tím se zvětší nebo zmenší prostor dole pro výsledky. Na mapě můžeme vidět trajektorii vybraného letu společně s vertikálním profilem. Let se dá přehrávat, pozastavit, přiblížit či oddálit a řadu dalších věcí, které byste očekávali při pohybu po 2D mapě.



Obrázek 4.2: Úprava přihlášky

Obrázek 4.3: Vizualizace letu

Kapitola 5

Nasazení a reálný provoz

5.1 Nasazení serveru

Pro nasazení serveru je potřeba několik věcí. Je potřeba se rozhodnout, kde se bude aplikace hostovat, jestli např. na vlastním serveru nebo se zvolí služba, která poskytuje hosting. Pro testovací účely jsem nejdříve zvolil vlastní server, nicméně nebylo udržitelné jej spravovat, tak jsem se rozhodl pro hosting v cloudu. Zvolil jsem populární službu *Heroku*, kde hostuji jak server, tak klienta. [39]

V Heroku se vytvoří tzv. app, tato jednotka se bude o naši aplikaci starat. Propojil jsem ji s GitHubem, kde hostuji repozitář s kódem. V Heroku se zvolí tzv. Buildpack, který naší aplikaci sestaví, já jsem použil heroku/nodejs buildpack přímo od Heroku. Pro zprovoznění aplikace je důležité správně nastavit tyto tzv. environment variables.

```
NODE_ENV=production
MONGO_URI_PROD=mongodb+srv://user:password...
MONGO_URI_DEV=mongodb+srv://user:password...
MONGO_URI_TEST=mongodb+srv://user:password...
JWT_SECRET=myjwtsecretforencryption
PORT=5000
AWS_ACCESS_KEY_ID=AWSAccessKeyId
AWS_SECRET=AWSSecretKey
SEND_EMAILS_TO_ALL=false
SMTP_USER=SMTP_USER
SMTP_PASSWORD=SMTP_PASSWORD
SOARING_SPOT_CLIENT_ID=client_id
SOARING_SPOT_SECRET=secret
```

Po jejich nastavení se server spustí a běží. Při aktualizace master branche v gitu se automaticky nasadí nová verze aplikace.

Nakonec se dá v Heroku nastavit vlastní doména, jediné co pro to potřebu-

jeme je přidat *CNAME* záznam v administrace naší domény. Pro administraci domény používám českou službu *WEDOS*. [40] Heroku i za malý poplatek automaticky spravuje naše SSL certifikáty, takže jsem schopni používat HTTPS velmi jednoduše.

Produkční server běží na adrese `https://api.tcup.cz`. Paralelně mám nasazený ještě jeden server, určený pro účely prezentace bakalářské práce, ten běží na adrese `https://api.thesis.tcup.cz`.

5.2 Nasazení klienta

Nasazení klienta probíhá velmi obdobně jako nasazení serveru. Jako u serveru používám pro hosting Heroku. Postup nasazení probíhá obdobně, jen s tím rozdílem, že pro sestavení aplikace používám
`github.com/mars/create-react-app-buildpack` buildpack, protože se jedná o React aplikaci, nikoliv o NodeJS a Heroku pro React aplikace nemá vestavěný buildpack.

Stejně jako u serveru mám nastavenou vlastní doménu. Produkční aplikace tentokrát běží na adrese `https://www.tcup.cz`. Pro účely prezentace je práce nasazená zde `https://thesis.tcup.cz`.

5.3 Soutěž v roce 2020

V roce 2020 proběhla soutěž tcup 2020, která používala moji vytvořenou aplikaci, ve stavu, ve kterém se v tom roce nacházela. Měla ovšem omezenou funkcionalitu, ale pomohla mi získat reálná data, pro vytvoření mé bakalářské práce. Lze se na ni historicky podívat, běží na adrese `http://2020.tcup.cz`. Jak lze vidět, aplikace nebyla propojená se Soaring Spotem, pouze na něj odkazovala, ale tyto data jsem mohl využít právě pro prezentaci v aplikaci `https://thesis.tcup.cz`.

Dokončením mé bakalářské práce bude aplikace v roce 2021 již využívat všechny nová vylepšení.

Aplikace byla otestována reálnými uživateli, kde jsem se setkal vesměs s kladnou zpětnou vazbou, jak od soutěžících, tak od organizátorů soutěže. Neobsahovala žádné závažné chyby. Bylo mi ovšem doporučeno několik návrhů změn, které zmiňuji v závěru práce.

V seznamu níže popisuji kvantitativní použivatelnost aplikace zanalyzováním databáze.

- do aplikace se přihlásilo 55 unikátních uživatelů
- bylo vytvořeno 53 přihlášek, což odpovídá 53 soutěžícím
- aplikaci používalo 5 administrátorů

- bylo publikováno 40 novinek
- proběhlo 5 letových dnů, z celkových 9
- bylo odesláno 272 IGC souborů pro vyhodnocení soutěže

5.4 Návrhy vylepšení aplikace

Po získání zpětné vazby od uživatelů aplikace *tcup 2020*, bych chtěl navrhnout možné změny aplikace do budoucna. Týkají se jak použitých technologií, tak uživatelské přívětivosti jak pro soutěžící, tak pro organizátory.

5.4.1 Použité technologie

TS klient

Pro odhalení více chyb při dalším vylepšování aplikace je třeba dokončit již probíhající migraci jazyka kódu z Javascriptu do Typescriptu. Tuto část jsem ve své práci nestihl dokončit.

TS server

Stejnou migraci jako v klientovi je potřeba udělat i v kódu serveru. Je to možná i důležitější tady, než na serveru, protože typová chyba může i dokonce shodit server.

PostgreSQL

V mé práci jsem zmiňoval, že jsem použil jako databázi No-SQL MongoDB. Tato databáze funguje dobře, nicméně nedokážu využít výhody No-SQL databází, jako je např. možnost nestrukturovaných dat. Pro budoucí vylepšování aplikace bude potřeba více entit, tudíž více komplexity z hlediska propojování tabulek. To se v MongoDB dělá složitě a není na to vyloženě určená.

5.4.2 Uživatelská přívětivost

Formulář s přihláškou

Po registraci soutěžící vyplňuje přihlášku. Tam je zatím nutné minimum dat, které uživatel vyplňuje. Organizátorům by se ale hodila další data, která by jim ušetřila další čas.

Při vyhodnocování letu je nutné zjistit, jestli kluzák, který má motor, neporušil pravidla a motor při soutěžním letu nezapnul. V registračním formuláři by se hodilo mít pole jestli je kluzák motorový či ne, tím pádem by se tato kontrola vyhnula kluzákům bez motoru.

Organizátoři na začátku soutěže musí propojit přihlášené soutěžící s jejich záznamem v IGC databázi, kvůli propojení jejich výsledků s jejich předchozími výsledky. Podle nich se pak nominují např. na mistrovství ČR nebo světa. Své `igcId` by si tedy soutěžící musel vybrat již ve formuláři s přihláškou.

Redesign

Stránce by prospěl celkový redesign, možná s nějakými vhodnými leteckými prvky. Na hlavní stránce bude nutné udělat stránkování nebo seskupování novinek podle soutěžních dnů. Možná i napsat vlastní CSS styly s pomocí grafika.

Administrace

Pro mnoho administračních detailů je nutné buď udělat ruční změnu v databázi a nebo poslat REST API požadavek. Tyto opakující se administrační změny by bylo vhodné zahrnout přímo do UI klienta.

Podpora více ročníků soutěže

Systém nemá podporu pro více ročníků soutěže, tudíž každý rok se musí nasadit nová aplikace se správnými nastaveními. Toto nasazení nějakou dobu trvá, navíc roste cena hostingu. Kdyby systém více ročníků podporoval, dále by to proto ulehčilo mnoho času organizátorům. Tato změna by byla reálná právě po změně použitých technologií.

Závěr

Cílem práce bylo analyzovat, navrhnout a implementovat systém pro správu leteckých soutěží. Všechny tyto dílčí cíle se mi podařilo splnit.

První kapitola se věnovala analýze existujících řešení a následným nadefinováním funkčních a nefunkčních požadavků. Dále jsem zohlednil případy užití, kde bylo popsáno, jak uživatelé aplikaci používají. To mi pomohlo při návrhu architektury a použitých technologií v druhé kapitole. Díky robustnímu datovému modelu se mi podařilo úspěšně a velmi rychle naimplementovat server. S implementací klienta jsem také neměl jediný problém, díky využití předchozích pracovních zkušeností a zkušeností ze školy.

Po úspěšné implementaci se mi podařilo nasadit aplikaci do produkce, kde se použila při soutěži *tcup 2020*. Aplikace fungovala bez problémů a já jsem se proto mohl soutěže zúčastnit i jako soutěžící i jako organizátor. Díky používání aplikace reálnými uživateli jsem získal reálná data pro dokončení této práce. Členům aeroklubu i závodníkům se líbila interaktivita aplikace, design hlavní stránky, tracking i statusy soutěžících. Dále jsem z vesměs kladné zpětné vazby uživatelů dostal návrhy na vylepšení, které se mi podařilo v této práci také naimplementovat.

Po nasazení těchto změn je aplikace připravena pro použití v dalším ročníku soutěže *tcup 2021*. Vývoj softwaru nikdy nekončí, existují tedy další možnosti rozšíření aplikace, které jsem zmínil v předchozí kapitole. Tyto požadavky bych chtěl určitě implementovat v nejbližší budoucnosti. Jedním z nejbližších rozšíření určitě bude redesign UI aplikace a změny ve formuláři s přihláškou.

Ambiciózním cílem by mohlo být aplikaci zobecnit a poskytovat ji i dalším plachtařským soutěžím. Po srovnání mého systému se stránkami dalších soutěží si myslím, že by ostatní organizátoři tuto možnost zvážili.

Literatura

[1] Gliding.cz: TCUP 2019. [online], [cit. 2020-12-6]. Dostupné z: http://www.gliding.cz/souteze/2019/tcup/

[2] Naviter: SeeYou. [software], [cit. 2021-03-11]. Dostupné z: https://naviter.com/seeyou-makes-you-a-better-pilot/

[3] Naviter: SeeYou Competition. [software], [cit. 2021-03-11]. Dostupné z: https://naviter.com/products/seeyou-competition/

[4] Naviter: Soaring Spot. [software], [cit. 2021-03-11]. Dostupné z: https://www.soaringspot.com/

[5] Gliding.cz: Gliding.cz diskusní fórum. [online], [cit. 2020-12-6]. Dostupné z: https://www.gliding.cz/souteze/

[6] Tkachenko, I.: Functional vs Non-functional Requirements: Main Differences & Examples. [online], [cit. 2020-4-27]. Dostupné z: https://theappsolutions.com/blog/development/functional-vs-non-functional-requirements

[7] What is a Functional Requirement? Specification, Types, EXAMPLES. [online], [cit. 2020-4-27]. Dostupné z: https://www.guru99.com/functional-requirement-specification-example.html

[8] Cockburn, A.: *Writing Effective Use Cases*. Addison-Wesley Professional, 2000, ISBN 978-0201702255.

[9] Architektura klient-server. [online], listopad 2016, [cit. 2021-03-11]. Dostupné z: https://managementmania.com/cs/architektura-klient-server

[10] Fielding, R. T.: *Architectural Styles and the Design of Network-based Software Architectures.* Dizertační práce, University of California, Irvine, 2000, [cit. 2021-03-11].

[11] BLAKIT - IT Solutions: Single-page applications vs. multiple-page applications: pros, cons, pitfalls. [online], srpen 2017, [cit. 2020-12-6]. Dostupné z: https://blak-it.com/blog/spa-advantages/

[12] Asper Brothers: Single Page Application (SPA) vs Multi Page Application (MPA) – Two Development Approaches. [online], listopad 2019, [cit. 2020-12-6]. Dostupné z: https://asperbrothers.com/blog/spa-vs-mpa/

[13] Daityari, S.: Angular vs React vs Vue: Which Framework to Choose in 2020. [online], srpen 2020, [cit. 2020-12-6]. Dostupné z: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

[14] Oza, K.: Angular v/s React v/s Vue: The Complete Comparison. [online], září 2020, [cit. 2020-12-6]. Dostupné z: https://dev.to/ozakaran/angular-v-s-react-v-s-vue-the-complete-comparison-2b7k

[15] Łukasz Kubok: Top 10 Companies Using React.js. [online], září 2020, [cit. 2020-12-6]. Dostupné z: https://selleo.com/blog/top-10-companies-using-reactjs

[16] Gackenheimer, C.: *Introduction to React*. Apress, 2015.

[17] Hamedani, M.: React Virtual DOM Explained. [online], prosinec 2018, [cit. 2021-03-11]. Dostupné z: https://programmingwithmosh.com/react/react-virtual-dom-explained/

[18] Banks, A.; Porcello, E.: *Learning React: functional web development with React and Redux*. Beijing: O'Reilly Media, 2017, ISBN 978-1-491-95462-1.

[19] Bugl, D.: *Learning Redux*. United Kingdom: Packt, 2017.

[20] Abramov, D.; Clark, A.: Redux. [software], [cit. 2021-03-11]. Dostupné z: https://redux.js.org/

[21] Abramov, D.; Clark, A.: React-Redux. [software], [cit. 2021-03-11]. Dostupné z: https://react-redux.js.org/

[22] Redux. [online], [cit. 2021-03-11]. Dostupné z: https://viptrust.com/technologie/ostatni/redux

[23] Reactstrap. [software], [cit. 2021-03-11]. Dostupné z: https://reactstrap.github.io/

[24] Bootstrap. [software], [cit. 2021-03-11]. Dostupné z: https://getbootstrap.com/

[25] Microsoft: Typescript. [software], říjen 2012, [cit. 2021-03-11]. Dostupné z: https://www.typescriptlang.org/

[26] Gajda, M.: Why use Node.js web development? Scalability, performance and other benefits of Node based on famous web applications. [online], červenec 2020, [cit. 2020-12-6]. Dostupné z: https://tsh.io/blog/why-use-nodejs/

[27] Labay, G.: 10 Most Popular NodeJS Frameworks of 2020. [online], září 2020, [cit. 2020-12-6]. Dostupné z: https://wiredelta.com/10-most-popular-nodejs-frameworks-of-2020/

[28] Automattic: Mongoose. [software], [cit. 2021-03-11]. Dostupné z: https://mongoosejs.com/

[29] Wong, D.: Multer-s3. [software], [cit. 2021-03-11]. Dostupné z: https://www.npmjs.com/package/multer-s3

[30] MongoDB. [software], [cit. 2021-03-11]. Dostupné z: https://www.mongodb.com/

[31] Tyagi, D.: Why choose MERN stack? [online], červenec 2020, [cit. 2020-12-6]. Dostupné z: https://medium.com/aeologic/why-choose-mern-stack-323b4d95e4ea

[32] Hoque, S.: *Full-Stack React Projects*. Packt, 2018, ISBN 978-1788835534.

[33] Amazon: Amazon S3. [software], [cit. 2021-03-13]. Dostupné z: https://aws.amazon.com/s3/

[34] Naviter: Public API for Soaring Spot. [online], [cit. 2021-03-13]. Dostupné z: https://kb.naviter.com/en/kb/public-api-for-soaring-spot/

[35] Naviter: *Soaring Spot API documentation*. [cit. 2021-03-13]. Dostupné z: http://download.naviter.com/soaringspot/api/index.html

[36] StrongLoop, Inc.: *Express documentation*. [cit. 2021-03-13]. Dostupné z: http://expressjs.com/en/api.html

[37] Nodemailer. [software], [cit. 2021-03-11]. Dostupné z: https://nodemailer.com/about/

[38] Abramov, D.; Clark, A.: Redux-thunk. [software], [cit. 2021-03-11]. Dostupné z: https://github.com/reduxjs/redux-thunk

[39] Salesforce: Heroku. [software], [cit. 2021-03-11]. Dostupné z: https://www.heroku.com/

[40] Wedos. [software], [cit. 2021-03-11]. Dostupné z: https://www.wedos.cz/

Příloha A

Seznam použitých zkratek

API Application Programming Interface

AWS Amazon Web Services

CRUD Create, Read, Update, Delete

CSR Client Side Rendering

CSS Cascading Style Sheets

DOM Document Object Model

GPS Global Positioning System

HAL Hypertext Application Language

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IGC International Gliding Commission

JS JavaScript

JSON JavaScript Object Notation

MERN MongoDB, Express, React, Node

MPA Multi Page Application

No-SQL not only SQL

REST Representational State Transfer

SDK Software Development Kit

SPA Single Page Application

SQL Structured Query Language

SSR Server Side Rendering

tcup Toužim Cup

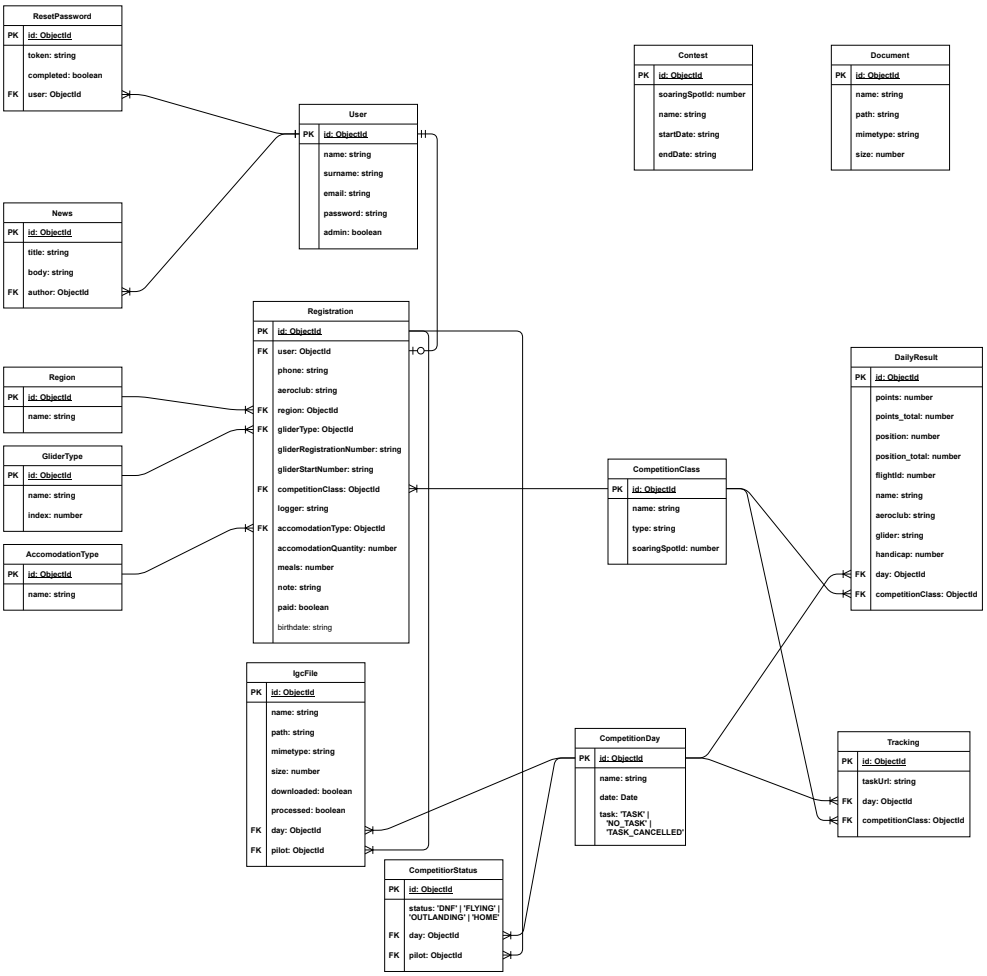
TS TypeScript

URI Uniform Resource Identifier

UX User Experience

Příloha B

Databázový model



Obrázek B.1: Kompletní databázový model

Příloha C

API endpointy

HTTP metoda	URI	popis
POST	api/auth	přihlásí uživatele
POST	api/auth/change-password	změní heslo uživateli
GET	api/auth/user	získá informace o uživateli
POST	api/auth/reset-password	první krok obnovy hesla
POST	api/auth/reset-password/valid	ověření validního tokeny pro obnovu hesla
POST	api/auth/reset-password/reset	obnoví heslo
GET	api/starting-list	vrátí startovní listinu
GET	api/starting-list/export	CSV export startovní listiny
GET	api/starting-list/export/seeyou/:cls	export startovní listiny do SeeYou
POST	api/competitiorstatuses	vytvoří status soutěžícího
PUT	api/competitiorstatuses/:id	upraví status soutěžícího
GET	api/competitiorstatuses/:day	vrátí statusy soutěžících

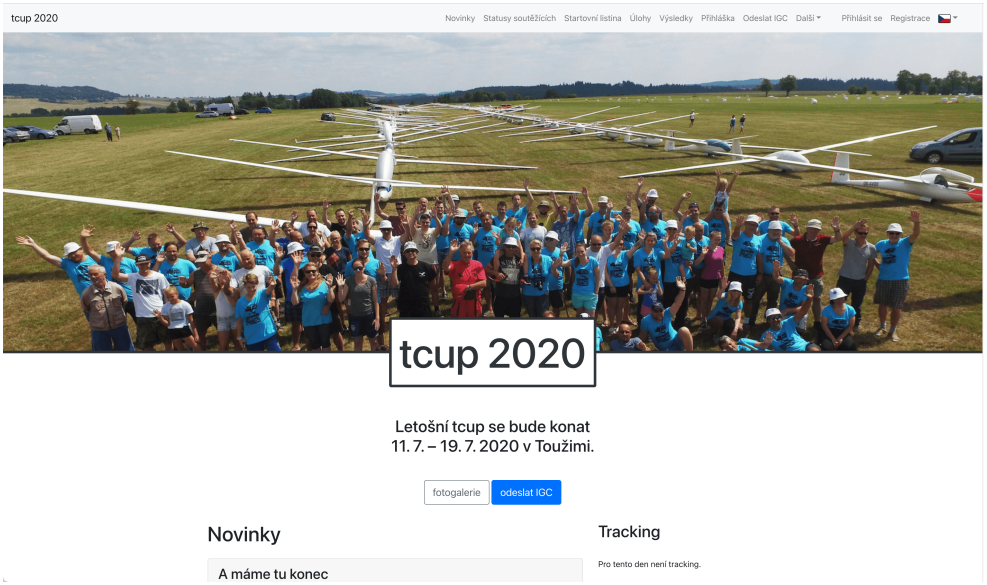
Tabulka C.1: 1. část endpointů

HTTP metoda	URI	popis
POST	api/accomodationtypes	vytvoří typ ubytování
GET	api/accomodationtypes	vrátí typy ubytování
POST	api/classes	vytvoří soutěžní třídu
GET	api/classes	vrátí soutěžní třídy
POST	api/days	vytvoří soutěžní den
GET	api/days	vrátí soutěžní dny
PUT	api/days/:id	upraví soutěžní den
POST	api/documents	vytvoří metadata dokumentu
GET	api/documents	vrátí dokumenty
DELETE	api/documents/:id	smaže dokument
POST	api/glidertypes	vytvoří typ kluzáku
GET	api/glidertypes	vrátí typy kluzáků
POST	api/igc	odešle IGC soubor
GET	api/igc/form	vrátí data pro formulář
GET	api/igc/:day	vrátí IGC soubory pro daný den
PUT	api/igc/:id	upraví informace o IGC souboru
POST	api/news	vytvoří novinku
GET	api/news	vrátí novinky
DELETE	api/news/:id	smaže novinku
POST	api/regions	vytvoří region
GET	api/regions	vrátí regiony
POST	api/registration	vytvoří novou přihlášku
GET	api/registration	vrátí vytvořenou přihlášku
GET	api/registration/form	vrátí data pro formulář
PUT	api/registration	upraví přihlášku
GET	api/registration/:id	vrátí přihlášku podle id
PUT	api/registration/:id	upraví přihlášku podle id
GET	api/results/top	vrátí výsledky
GET	api/results/total	vrátí celkové výsledky
GET	api/results/daily/filters	vrátí filtry pro výsledky
GET	api/results/daily/:cls/:day	vrátí denní výsledky
POST	api/tracking/:day/:cls	vytvoří tracking třídy
GET	api/tracking/:day	vrátí tracking třídy
POST	api/users	zaregistruje nového uživatele
GET	api/users	vrátí uživatele
PUT	api/users	upraví uživatele
GET	/	vrátí verzi aplikace

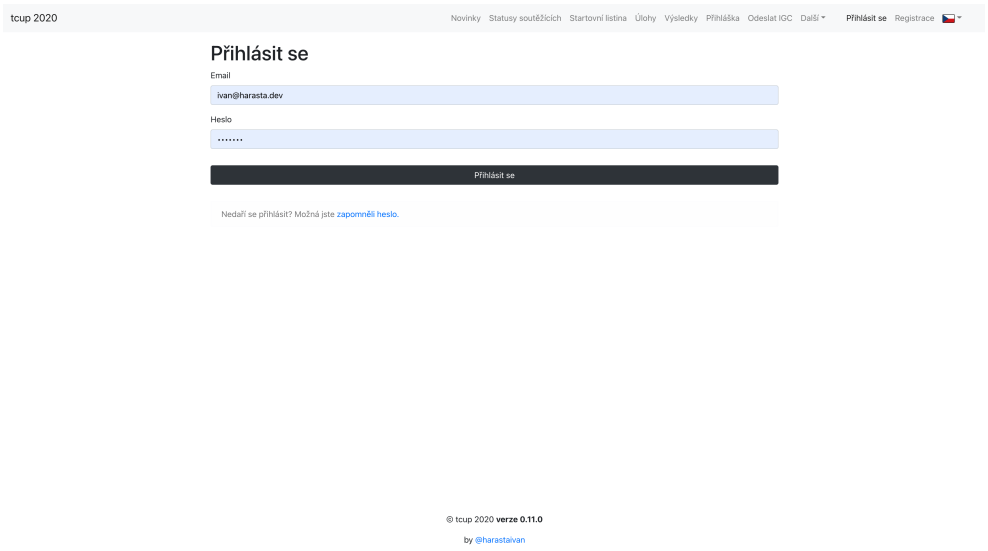
Tabulka C.2: 2. část endpointů

Příloha D

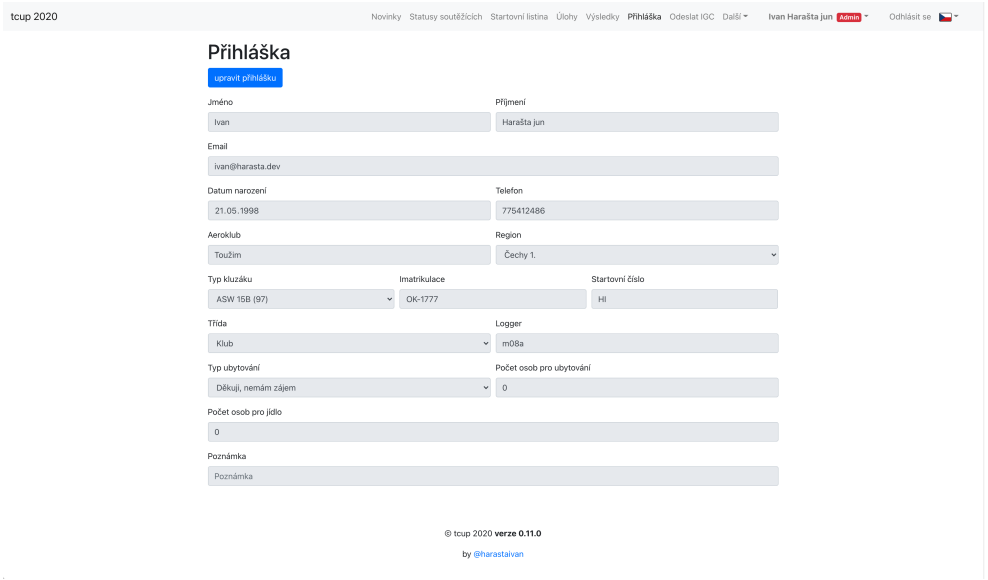
Výsledná aplikace



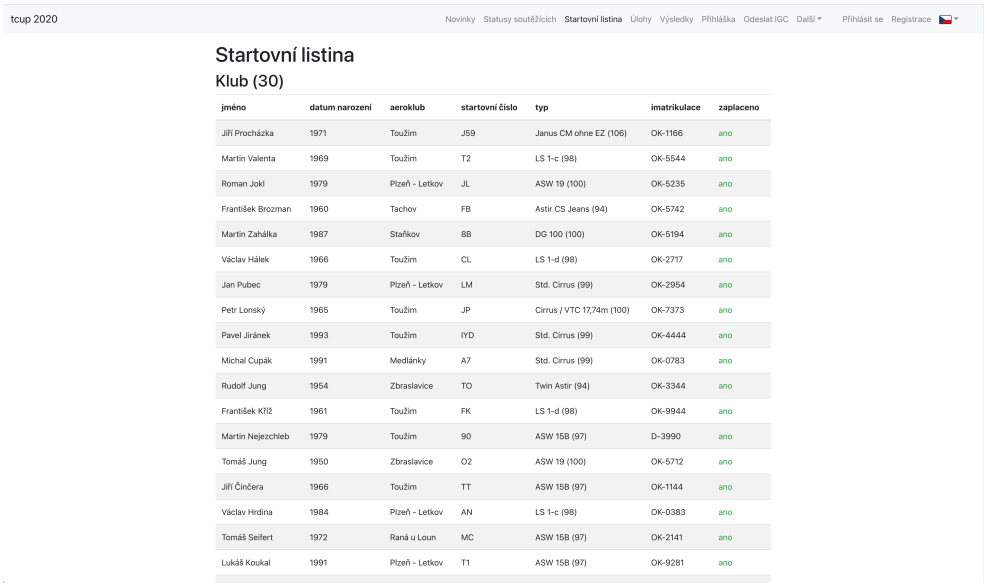
Obrázek D.1: Hlavní stránka



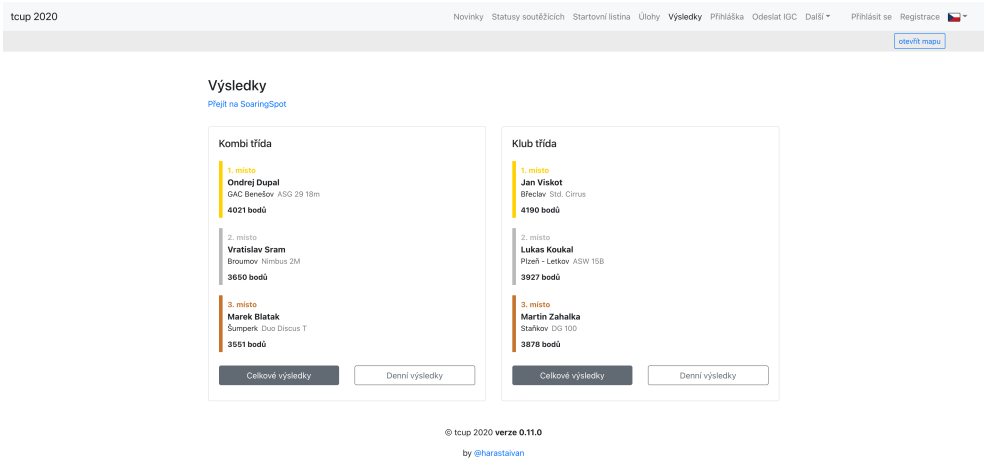
Obrázek D.2: Stránka pro přihlášení



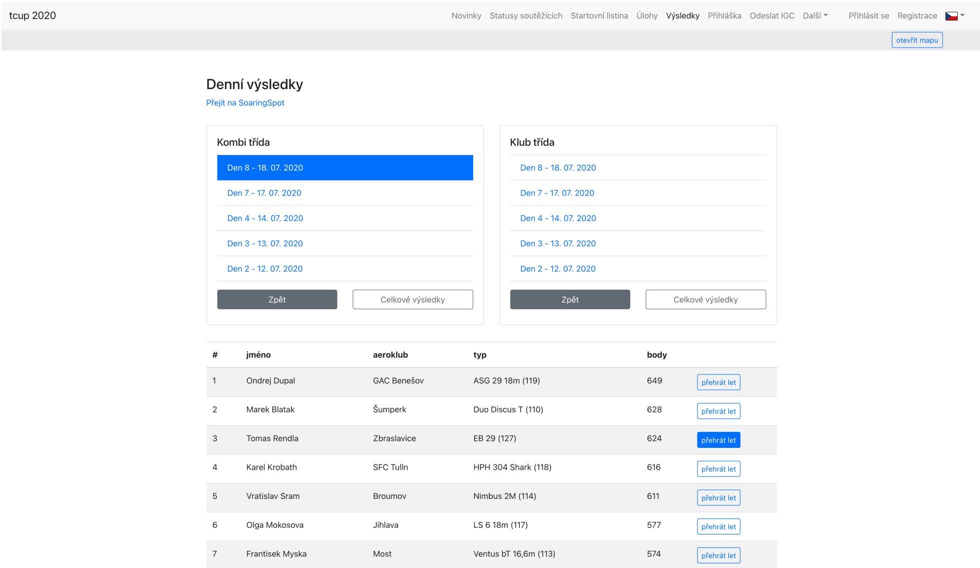
Obrázek D.3: Stránka pro úpravu přihlášky přihlášeného uživatele



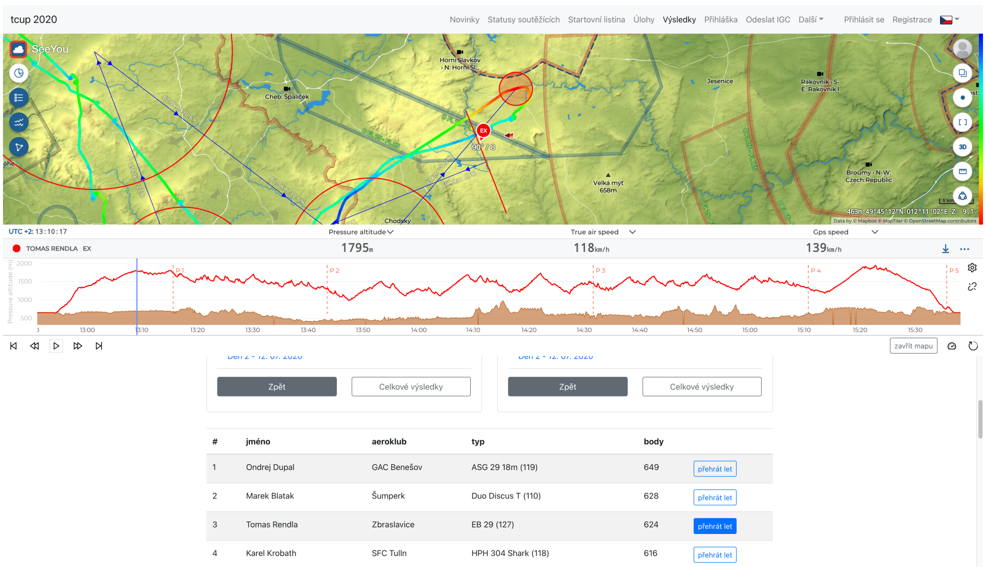
Obrázek D.4: Startovní listina



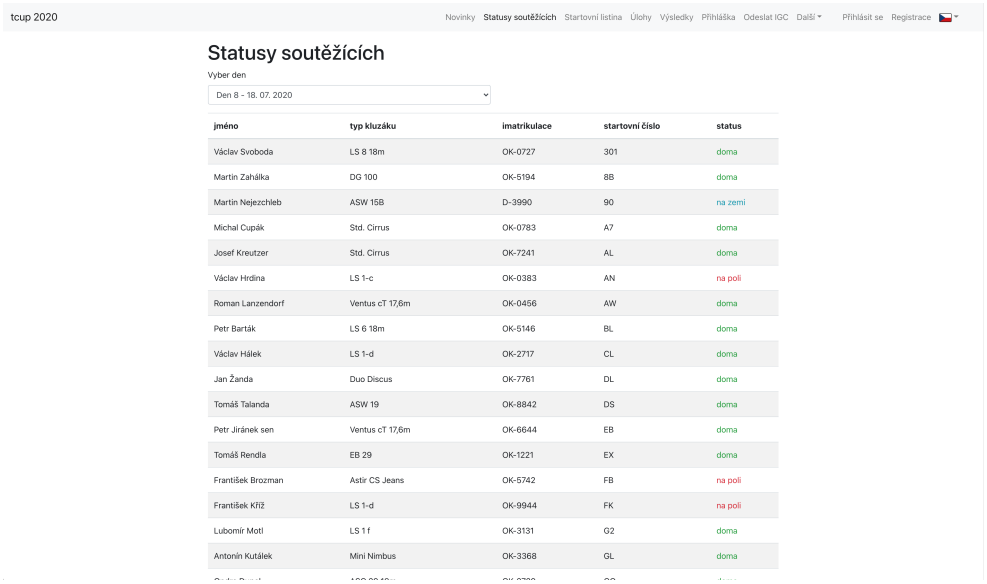
Obrázek D.5: Stránka s konečnými výsledky



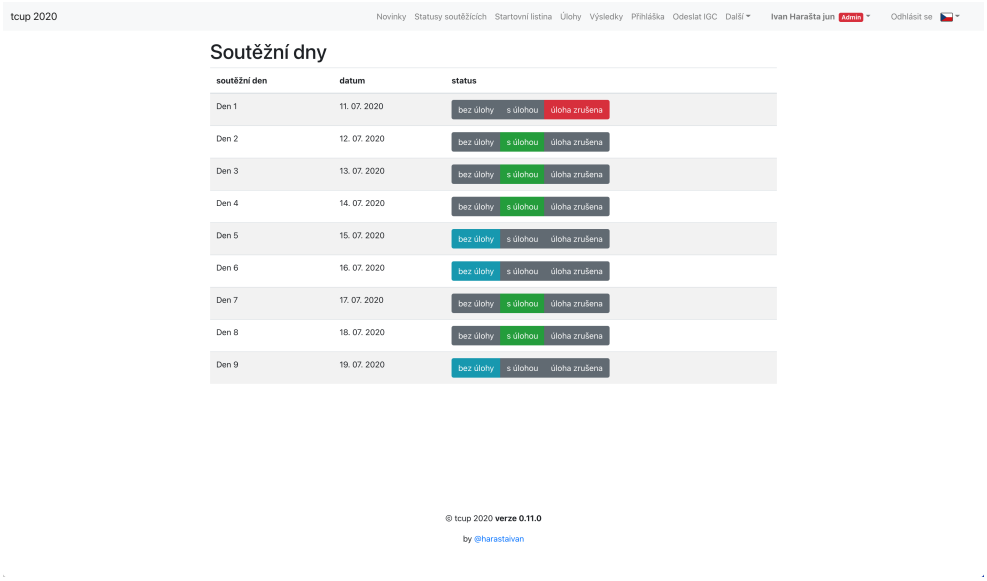
Obrázek D.6: Stránka s denními výsledky s minimalizovanou vizualizací



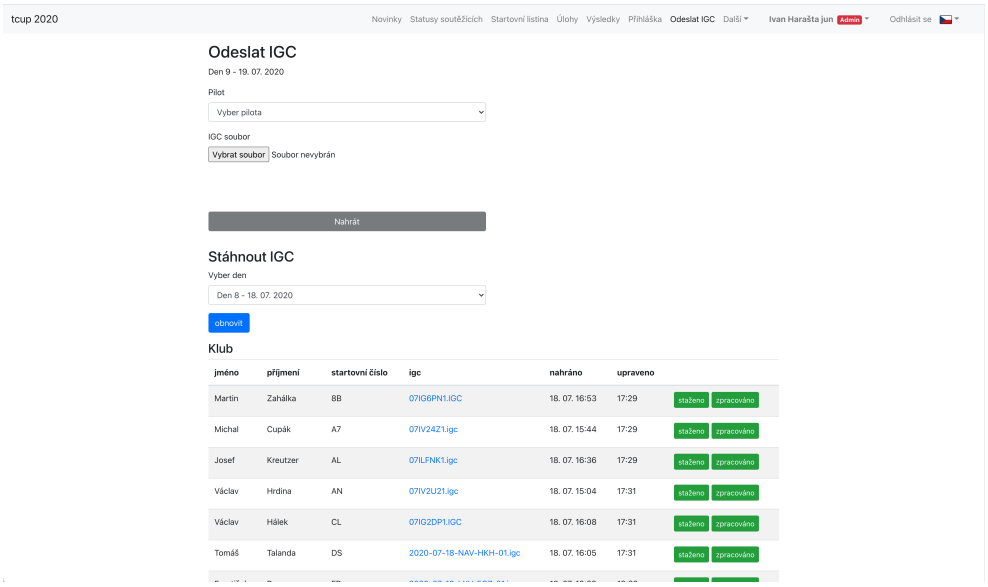
Obrázek D.7: Stránka s denními výsledky s maximalizovanou vizualizací



Obrázek D.8: Statusy soutěžících



Obrázek D.9: Administrace soutěžních dnů



Obrázek D.10: Administrace IGC souborů

Příloha E

Obsah přiloženého flash disku

- `README.md` ..... stručný popis obsahu flash disku
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `client` ..... zdrojové kódy implementace klienta
    - `server` ..... zdrojové kódy implementace serveru
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.txt` ..... text práce ve formátu txt