

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ  
V PRAZE

FAKULTA ELEKTROTECHNICKÁ  
KATEDRA POČÍTAČŮ



Návrh a implementace uživatelského  
rozhraní sítě robotických dalekohledů  
řízených systémem RTS2  
Diplomová práce

Autor: Bc. Radek Mečiar

Studijní obor: Otevřená informatika  
Zaměření: Počítačové inženýrství



Czech Technical University in Prague  
Faculty of Electrical Engineering  
Department of Control Engineering

# DIPLOMA THESIS ASSIGNMENT

Student: **Mečiar Radek**

Study programme: Open Informatics  
Specialisation: Computer Engineering

Title of Diploma Thesis: **Design and implement user interface for robotics network of telesocpes with RTS2**

## Guidelines:

1. Get knowledge about RTS2
2. Design and implement modular user interface according these conditions
  - (a) Distinguish access level for every telescope and user
  - (b) Allow real-time controll telescope
  - (c) Planning of observations in whole telescope network
  - (d) Implement data access interface considering data flow optimalization
  - (e) Notifications for client (WebSocekt)
3. Implement missing functionalities in RTS2
4. Evaluate capability and usability of application

## Bibliography/Sources:

- [1] Petr Kubánek, "RTS2: The Remote Telescope System," Advances in Astronomy, vol. 2010, Article ID 902484, 9 pages, 2010. doi:10.1155/2010/902484
- [2] Kubánek, Petr, et al. "RTS2: meta-queues scheduling and its realisation for FLWO 1.2 m telescope." SPIE Astronomical Telescopes+ Instrumentation. International Society for Optics and Photonics, 2012.
- [3] Pérez-del-Pulgar, Carlos Jesús, et al. "GLORIA: The First Free Access e-Infrastructure of Robotic Telescopes for Citizen Science." Advances in Information Systems and Technologies. Springer Berlin Heidelberg, 2013. 293-304.

Diploma Thesis Supervisor: Ing. Stanislav Vitek, Ph.D.

Valid until the summer semester 2017/2018

prof. Ing. Michael Šebek, DrSc.  
Head of Department

prof. Ing. Pavel Ripka, CSc.  
Dean

Prague, November 30, 2016



# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 20.5.2017

Podpis: .....



# Poděkování

Zde bych rád poděkoval svému vedoucímu práce Ing. Stanislavu Vítkovi, Ph.D především za zajímavé téma a za pomoc s řešením, hlavně při hledání informací.





# Abstrakt

Práce se zabývá návrhem a implementací softwaru pro sjednocení přístupu k observatořím využívajících systém RTS2. Cílem práce je zpřehlednit a ulehčit ovládání observatoří z jednoho místa, společně s náhradou konzolového přístupu ovládání. Hlavním výsledkem práce je software. Konkrétně se jedná o webovou aplikaci, která se skládá z JavaScriptové klientské části a serveru napsaného v jazyce Java. Práce dále shrnuje základní informace o systému RTS2, které jsou potřebné k implementaci. Hlavní analýza RTS2 systému se týká HTTP metod, které umožňují vzájemnou komunikaci mezi servery. Výsledkem práce je serverové řešení, které je schopné komunikovat s více systémy RTS2. Vytvořený server používá protokol HTTP, jako klient, tak jako server pro přístup. Pro serverovou část byl implementován WebSocket server a zprovozněna ukázková funkce. Klientská část je schopna se připojit a komunikovat s vytvořeným serverem. Je určena k přehlednému podání informací uživateli. Vytvořená aplikace umožňuje správu uživatelů a systémů RTS2, jejich cílů pozorování, zařízení a jednotlivých parametrů, front pro plánování a umožňuje přístup k napozorovaným datům.

## Key words

rts2, SPA, ovládání, dalekohled, observatoř, teleskop, web, server, klient, remote telescope system, JavaScript, AngularJS, Bootstrap, Java, PostgreSQL



# Abstract

The thesis deals with design and implementation of a software for a control unification with observatories using the RTS2 system. The objective of the thesis is to facilitate and simplify the control of an observatory from one web application together with a console access control replacement. The main result of the thesis is the software. Specifically, the software is a web solution that consists of a JavaScript client and a Java server. The thesis also analyzes basic informations about the RTS2 system in volume needed for the implementation. The analysis of the RTS2 system concerns HTTP methods that allow communication between servers. The created server uses HTTP as a client as well as an server and it is able to communicate with multiple RTS2 systems. Basic WebSocket protocol was implemented for the server part and also with a demonstration of use. The client part is able to communicate with server part and provide informations for users. Created solution enables an administration of users, observatories, targets, devices and their properties, queues management for planing of observations and pictures access considering data flow optimalization.

## Key words

rts2, SPA, control, telescope, observatory, web, server, client, remote telescope system, single page application, JavaScript, AngularJS, Bootstrap, Java, PostgreSQL



# Obsah

<b>1</b>	<b>Úvod</b>	<b>17</b>
1.1	Popis požadavků práce . . . . .	17
1.2	Motivace . . . . .	19
1.3	Zdroje pro práci . . . . .	19
<b>2</b>	<b>Analýza</b>	<b>21</b>
2.1	Remote Telescope System 2 . . . . .	21
2.1.1	JSON API . . . . .	21
2.1.2	Přístup k napozorovaným datům . . . . .	21
2.2	Analýza současných projektů . . . . .	22
2.3	Výběr technického pozadí . . . . .	23
2.3.1	Typ komunikace server - klient . . . . .	23
2.3.2	Výběr konkrétních technologií . . . . .	24
2.3.3	Databáze . . . . .	25
2.3.4	Administrace projektu . . . . .	25
<b>3</b>	<b>Popis zvoleného řešení</b>	<b>27</b>
3.1	Klientská část RTS2 UI . . . . .	27
3.1.1	Hlavní soubor klienta - index.html . . . . .	27
3.1.2	Hlavní soubor JS - app.js . . . . .	28
3.1.3	Kontrolery . . . . .	30
3.1.4	Rozložení . . . . .	31
3.1.5	Servisy . . . . .	31
3.2	Serverová část RTS2 UI . . . . .	32
3.2.1	Konfigurace serveru . . . . .	33
3.2.2	Hlavní logika aplikace . . . . .	33
3.2.3	HTTP rozhraní serverové části . . . . .	35
3.3	Databáze RTS2 UI . . . . .	39
<b>4</b>	<b>Popis funkce</b>	<b>41</b>
4.1	Správa observatoří . . . . .	41
4.2	Ovládaní zařízení . . . . .	43
4.3	Správa cílů pozorování . . . . .	48
4.3.1	Seznam cílů . . . . .	48
4.3.2	Editace a vytvoření cíle . . . . .	49
4.4	Plánování . . . . .	50
4.4.1	Přidání cíle do fronty . . . . .	51
4.4.2	Zobrazení fronty . . . . .	52
4.4.3	Zobrazení plánu . . . . .	53
4.5	Přístup k obrázkům . . . . .	53

4.5.1	Podle data pozorování . . . . .	53
4.5.2	Podle cíle pozorování . . . . .	54
4.6	Správa uživatelů a oprávnění . . . . .	55
<b>5</b>	<b>Zhodnocení a závěr</b>	<b>57</b>
<b>A</b>	<b>CD</b>	<b>60</b>
<b>B</b>	<b>Seznam použitých zkratk</b>	<b>61</b>
<b>C</b>	<b>Návod na použití projektu</b>	<b>62</b>

---

# Seznam obrázků

1.1	Diagram projektu RTS2 UI. . . . .	18
3.1	Diagram vrstev navrženého serveru. . . . .	28
3.2	Schéma databáze RTS2 UI. . . . .	39
4.1	Rozhraní RTS2 UI - seznam teleskopů. . . . .	41
4.2	Rozhraní RTS2 UI - seznam teleskopů pro veřejnost. . . . .	41
4.3	Rozhraní RTS2 UI - formulář pro správu teleskopu. . . . .	42
4.4	Rozhraní aplikace <i>rts2-mon</i> . . . . .	43
4.5	Rozhraní RTS2 UI. . . . .	44
4.6	Rozhraní RTS2 UI - zobrazení času. . . . .	45
4.7	Rozhraní RTS2 UI - zobrazení běžných datových typů. . . . .	45
4.8	Rozhraní RTS2 UI - zobrazení logického typu. . . . .	45
4.9	Rozhraní RTS2 UI - zobrazení parametru typu pole. . . . .	46
4.10	Rozhraní RTS2 UI - zobrazení parametru typu obdélník. . . . .	46
4.11	Rozhraní RTS2 UI - zobrazení parametru typu objekt Alt Az souřadnic. . . . .	46
4.12	Rozhraní RTS2 UI - zobrazení aktuálního snímku. . . . .	47
4.13	Rozhraní RTS2 UI - seznam pozorovacích cílů. . . . .	49
4.14	Rozhraní RTS2 UI - editace pozorovacího cíle. . . . .	50
4.15	Rozhraní RTS2 UI - formulář pro přidání cíle do fronty. . . . .	52
4.16	Rozhraní RTS2 UI - tabulka pro zobrazení front. . . . .	53
4.17	Rozhraní RTS2 UI - tabulka pro zobrazení plánu. . . . .	53
4.18	Rozhraní RTS2 UI - zobrazení obrázků podle z určitého dne. . . . .	54
4.19	Rozhraní RTS2 UI - zobrazení snímků podle pozorovacího cíle. . . . .	55
4.20	Rozhraní RTS2 UI - seznam uživatelů. . . . .	55
4.21	Rozhraní RTS2 UI - formulář uživatele. . . . .	56
4.22	Rozhraní RTS2 UI - přiřazení oprávnění k teleskopu. . . . .	56

## Seznam Kódů

3.1	Oblast pro zobrazení layoutů. . . . .	28
3.2	Definice modulu 'app'. . . . .	28
3.3	Centrální řešení požadavků a odpovědí klientské části. . . . .	29
3.4	Směrování v klientské části. . . . .	30
3.5	Definice modulu. . . . .	30
3.6	Rozšíření modulu. . . . .	30
3.7	Použití kontroleru. . . . .	31
3.8	Ukázka metody servisu. . . . .	32
3.9	Zapouzdření obsluhy HTTP žádosti. . . . .	36
3.10	Metoda pro obsluhu požadavku na uložení uživatele. . . . .	36
3.11	Hlavní metoda filtru pro tokenovou autorizaci požadavku. . . . .	37
4.1	Příkaz pro přidání pozorovacího cíle do fronty. . . . .	51



## Seznam tabulek

3.1	Seznam závislosti na ostatních knihovnách. . . . .	33
4.1	Metoda pro získání seznamu zařízení systému RTS2. . . . .	44
4.2	Metoda pro získání parametrů zařízení. . . . .	44
4.3	Metoda pro uložení parametru zařízení. . . . .	47
4.4	Metoda pro získání současného snímku. . . . .	48
4.5	Metoda pro spuštění expozice u zařízení typu kamera. . . . .	48
4.6	Metoda pro získání seznamu všech pozorovacích cílů. . . . .	49
4.7	Metoda pro získání seznamu specifických pozorovacích cílů. . . . .	49
4.8	Metoda pro získání informací o konkrétním pozorovacím cíli. . . . .	50
4.9	Metoda pro vytvoření pozorovacího cíle. . . . .	51
4.10	Metoda pro úpravu pozorovacího cíle. . . . .	51
4.11	Metoda pro vykonání příkazu nad zařízením. . . . .	51



# 1. Úvod

Již několik let pracuji pro komerční sféru jako webový vývojář, i se sám o webové technologie zajímám. Jsem tedy rád za možnost uplatnění těchto znalostí v závěrečné práci, která se týká tohoto směru a ještě zajímavějšího prvku, a tím je pozorování vesmíru.

Účelem této práce je vytvoření webového rozhraní, které zpřístupní možnosti observatoře. Observatoři bude možnost do systému přidat více a centrálně je spravovat. Bude možné provádět základní úkony, které potřebují vědci pro obsluhu observatoře, jako správa zařízení a nastavování parametrů, správa cílů pozorování, možnost pozorovat, přístup k napozorovaným datům a další. Observatoře, které půjdou do centrálního systému přidat, musí používat systém RTS2 [6].

Protože název práce je trochu delší, tak jsem tuto práci, či softwarový projekt, nazval RTS2 UI, jako navázání na projekt RTS2, který podrobněji popisují v sekci 2.1 a zkratka UI znamená uživatelské rozhraní z anglického *User Interface*. Konkrétně se bude jednat o webové uživatelské rozhraní, o které se RTS2 UI snaží původní projekt RTS2 rozšířit.

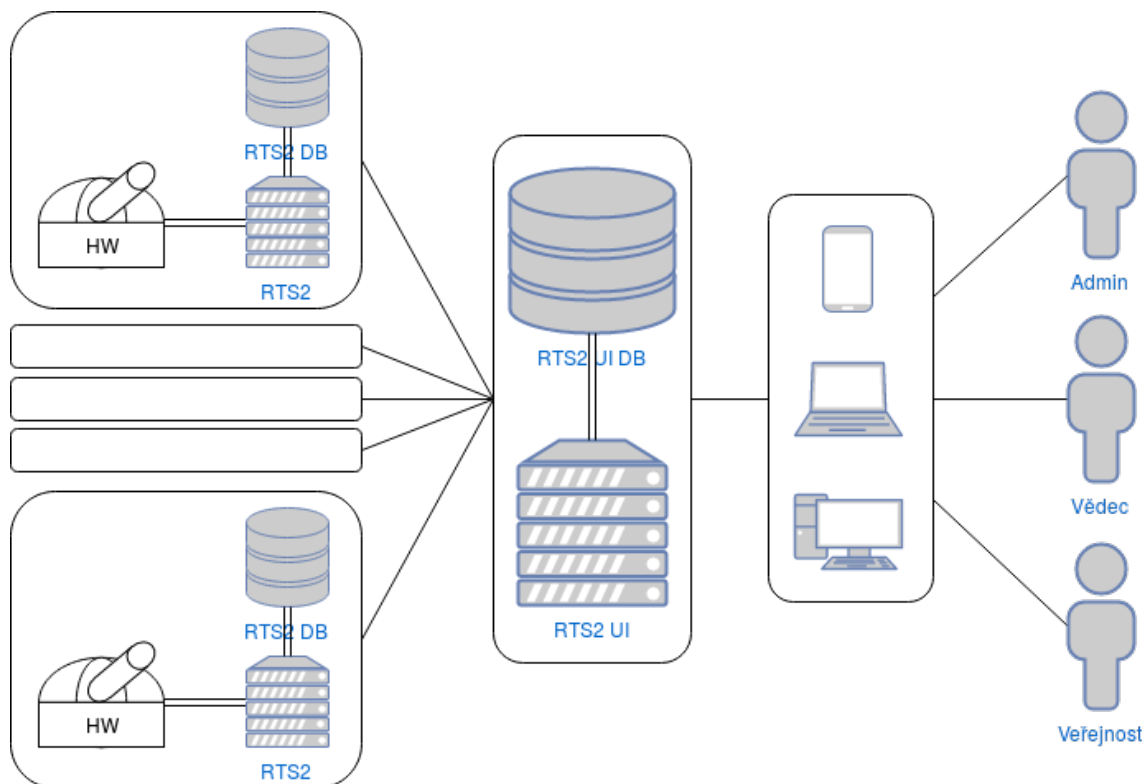
Práce má za úkol praktickou implementaci rozhraní pro uživatele, které bude umožňovat snadnou správu a ovládání, také nabídne vyšší nadhled. Pro začátek bych uvedl, že projekt RTS2 je zjednodušeně řešeno software, který zajišťuje chod observatoře jak z pohledu hardwaru, tak s možností plánovat a vzdáleně pozorovat chování observatoře. Nechybí mu ani možnost autonomního pozorování. Projekt RTS2 umožňuje mnoho věcí, ale chybí mu centrální jednoduchý nadhled nad více observatořemi.

## 1.1 Popis požadavků práce

Samotný projekt RTS2 je vedený pod licencí LGPL. Ovladače, které již tento projekt zahrnuje jsou vedeny pod GPL licencí. I softwarový projekt RTS UI musí mít otevřený zdrojový kód.

Pokusy o vytvoření obdobného softwaru v minulosti již byly, například ve zdroji [5]. Podle dostupných informací a provedené analýzy jim chybí flexibilní generování požadavků. Umožňují uživatelům měnit jisté zpřístupněné parametry, ale obvykle jen pomocí specifických metod, což není moc praktické. Pokud by se například změnil název proměnné zařízení nebo celé zařízení, tak by již takto ručně napsaný kód nemohl fungovat a musel by se opět přizpůsobit nové změně dalším ručním přepsáním. Z toho vyplývá, že jedním z hlavních požadavků softwarového projektu RTS2 UI je, aby uměl generovat požadavky podle dostupných dat o zařízení přizpůsobivě, dle aktuálních informací.

Dalším bodem je oprávnění uživatelů. Projekt RTS2 samotný poskytuje autorizace k požadavkům pomocí protokolu HTTP. Ověření funguje pouze na lokální observatoř a může nastavit uživateli oprávnění zacházet se zařízeními. Pro potřeby



Obrázek 1.1: Diagram projektu RTS2 UI.

centralizovaného přístupu jako u projektu RTS2 UI bude potřeba vytvořit různá oprávnění a správu uživatelů. Uvažovaná oprávnění jsou administrátor, vědec a veřejný účet. Kdy administrátor má mít možnost provádět všechny akce. Oprávnění vědce by umožňovalo provádět úkony týkající se jednotlivých dalekohledů. A veřejný účet by umožnil plánovat pozorování pouze v časech, kdy si observatoř vyhlásí veřejně přístupné hodiny.

Aby uživatelé mohli plánovat pozorování, bude nutné zpřístupnit správu pozorovacích cílů. Pozorovací cíl je bod na obloze, jehož souřadnice se uloží společně s pojmenováním a dalšími parametry do databáze. Pozorovací cíle jsou na každé observatoři vedeny jinak, tj. stejné souřadnice na obloze mohou mít jinou identifikaci v databázi, či jiný název.

Observatoře se systémem RTS2 umožňují automatické vykonávání plánu. Správa plánu bude přístupná i z projektu RTS2 UI. Plán je realizován za pomoci přístupu front, z kterých se podle priority a vhodnosti cíle vybírá. Centrální systém musí umět tyto fronty spravovat.

Nakonec je potřeba umět k uživateli dostat výsledek pozorování. Snímky jsou uchovávány v každé observatoři zvlášť. Některé observatoře, vzhledem k jejich poloze, mají pomalejší připojení k internetu. Bude tedy vhodné vytvořit centrální přístup k napozorovaným snímkům s ohledem na odlehčení toku dat mezi observatořmi a centrálním serverem.

## 1.2 Motivace

Systém RTS2 nabízí správu observatoře pomocí HTTP protokolu, konzolových aplikací a jednoduchého webu, který spíše ukazuje informace o stavu. Vzhledem k asociativnímu myšlení lidí je lepší, když vidí možnosti, než ruční psaní příkazů z paměti. Doufám v praktické použití projektu vědci a dalšími lidmi, kterým takovýto interface může ulehčit práci.

Dále pak centralizovaný přístup může nabídnout lepší přehled nad již napozorovanými daty nebo nad celkovým plánem sítě observatoří. Což může otevřít cestu dalším modulům projektu, které mohou provádět například statistické analýzy snímků nebo vylepšovat metody plánování pozorování centrálně v síti observatoří.

V neposlední řadě je zajímavou motivací zpřístupnění observatoře veřejnosti. Protože jsou některé observatoře financovány z veřejných peněz, mají povolený přístup pro veřejnost v určitých hodinách.

## 1.3 Zdroje pro práci

Hlavním zdrojem pro práci je dokumentace a zdrojový kód projektu RTS2. Vše je dostupné ve zdrojích [6] a [1]. Zdroj [6] je webová stránka projektu, kde je k nalezení popis a zdrojový kód. Zdroj [1] je dokument popisující různou funkcionalitu a hlavně HTTP rozhraní.

Pro tvorbu softwaru využiji své znalosti. Protože jsou hlavně z komerční sféry, budu muset najít vhodné alternativy, které mi umožní vytvořit podobné řešení.



## 2. Analýza

### 2.1 Remote Telescope System 2

Remote Telescope System 2 (RTS2) je projekt, na který navazují. Zajišťuje standardní rozhraní pro komunikaci s observatoří. Je velmi rozsáhlý. Zahrnuje abstraktní třídy pro zařízení, modulárním přístupem implementuje různé servisní služby jako plánovač, vykonávač pozorování, služba BigBrother zajišťuje plánování mezi více teleskopy zároveň a další služby. A hlavně je navržen pro samostatný běh observatoře.

Tato práce nejvíce využije HTTP server, který poskytuje komunikační rozhraní mezi observatoří a standardním HTTP klientem. Projekt RTS2 má otevřený zdrojový kód, tedy je dostupný a dá se volně rozšiřovat a upravovat. Jazyk tohoto projektu je hlavně C/C++. Dále jsou různé doplňkové skripty napsány v Pythonu. A používaná databáze je PostgreSQL. Projekt je velice rozsáhlý, a proto dále uvedu pouze informace, které budu využívat. Více informací o projektu se nachází na webové stránce ze zdroje [6].

#### 2.1.1 JSON API

Projekt RTS2 UI bude komunikovat s jednotlivými observatořemi, které používají RTS2 systém pomocí HTTP protokolu. RTS2 zajišťuje komunikaci pomocí HTTPD servisu. Servis očekává požadavky využívající metodu GET, neboli snadno formulovatelné pomocí URL adresy. Každý požadavek musí být autorizován HTTP ověřením uživatele, a to pomocí jména a hesla. Pokud je požadavek úspěšný, server odpovídá kódem 2XX, v opačném případě se použije kód 4XX. Server vrací data ve formátu JSON v těle odpovědi, kde může být výsledek dotazu nebo chybová hláška.

Informace, jak správně formulovat požadavky pro RTS2 observatoř, jsou z části ve zdroji [1]. Dokument je však neúplný, obsahuje řadu chyb a neaktuálností. Dalším zdrojem je pak samotný zdrojový kód, který je dostupný na serveru GitHub a odkaz je ve zdroji [10]. Ten není obvykle opatřen komentáři a ani intuitivními názvy vstupních proměnných do požadavků, což způsobuje problém při napojení dalších projektů jako RTS2 UI. Konkrétní metody použité k realizaci budou popsány v kapitole 4, kde je i vysvětlení použití.

#### 2.1.2 Přístup k napozorovaným datům

Napozorovaná data, jsou dostupná pomocí protokolu HTTP. Soubory jsou uloženy na disku a RTS2 systém umožňuje procházet úložiště observatoře, na kterém software běží.

Způsoby vyčtení dat jsou různé. Je možnost zobrazit přímo data fyzicky v adresářové struktuře, která je tvořena podsložkami pojmenovanými podle dat, kdy jsou snímky, které se ve složce nachází, pořízeny. Další způsoby jsou zobrazení snímku konkrétního pozorovacího cíle nebo snímků konkrétního již proběhlého pozorování.

Systém RTS2 disponuje rozšířením, které podle souřadnic snímku dokáže určit, jestli je snímek dobrý. Automaticky pak třídí snímky, které jsou přexponované nebo rozmazané.

Systém RTS2 poskytuje několik formátů snímku. Při dotazu na současný snímek je formát binární a struktura byla nalezena ve zdrojovém kódu. Dále standardní formát JPEG, jak pro náhledy, tak pro snímky plného rozlišení. A hlavním astronomickým formátem je FITS [9], který obsahuje i různá metadata o pozorování. Snímky jsou černobílé s větší bitovou hloubkou na jeden pixel, konkrétně 12 bit, což se převede do 16 bit proměnné *short*.

## 2.2 Analýza současných projektů

Podobné softwarové řešení již vzniklo v rámci diplomové práce. Práce byla více zaměřena na ovládání jedné observatoře se systémem RTS2 pomocí mobilní aplikace. Název práce je „Mobilní aplikace pro řízení v reálném čase“ a autorem je Jan Šára. Lze ji najít v referenci [5].

Práce volí jako hlavní jazyk JavaScript. Ten je využit jak pro klientskou část, tak pro serverovou část. Pro projekt RTS2 UI nepovažuji tuto volbu pro serverovou část za vhodnou. Nevím, jestli se v JavaScriptu dají řešit různé problémy, například více vláknová obsluha nebo různé servery. Jazyk Java toto určitě umí a není problém jej využít. Disponuje množstvím knihoven, díky kterým si programátor může ulehčit práci. JavaScript na serverové části určitě není tak rozšířený. To znamená, že i množství řešených problémů bude menší a v případě nějakého specifického požadavku to může být neřešitelné za pomoci standardního postupu. V neposlední řadě bych odhadoval, že Java bude i efektivnější než JavaScript z hlediska výkonu.

Dále práce používá MongoDB databázi. Je to zřejmě výhodná volba při použití právě s programovacím jazykem JavaScript. Databáze MongoDB ukládá data ve formátu JSON v binární podobě. JavaScript nepotřebuje přetypovávat nebo vytvářet dotazy, ale stačí data jen načítat a ukládat. Zřejmě práce nevyužívá ani referenční integrity dat, protože MongoDB není relační databází. Projekt RTS2 UI bude mít určitě pevnou strukturu dat s potřebou zachování referenční integrity. Proto byla raději vybrána relační databáze PostgreSQL.

Při analýze použitých HTTP metod od systému RTS2, které využívá práce, bych odhadoval, že není realizována s ohledem na generování formulářů a požadavků podle aktuálních informací. Zpřístupněné kontrolní prvky výsledné aplikace také napovídají, že umožňuje jen provedení některých úkonů. V případě změny parametrů se pravděpodobně bude muset přepsat znovu. Proto jsem se nemohl inspirovat ani v klientské části aplikace pro generování formulářů. Robustnost řešení je jedním z hlavních cílů RTS2 UI.

Práce mě inspirovala okrajově v tom, že i RTS2 UI může být mobilní aplikací. Vzhledem ke zvolené vzhledové knihovně Bootstrap je toto realizováno snadno, ale není na to brán primární ohled. Je možné, že něco nebude správně zarovnané. Hlavní myšlenka mobilní verze je, že se prvky, které jsou v řádku vedle sebe, mají schopnost



při zúžení obrazovky přeskupit pod sebe. To ulehčí práci uživateli, který pak nebude muset posouvat okno v horizontálním směru, ale pouze ve vertikálním. Rovněž se tímto prvek nepatrně zvětší a je pak snadněji ovladatelný na menší obrazovce.

## 2.3 Výběr technického pozadí

### 2.3.1 Typ komunikace server - klient

Aktuálních dostupných webových technologií je dnes spousta. Dají se použít různé architektury a styl komunikace mezi klientem a serverem. Nejznámějšími technikami jsou pravděpodobně MVC nebo *SPA + BackEnd*, které jsem také uvažoval a dále je rozeberu.

#### MVC

MVC je zkratka z anglického *Model View Controller* a znamená styl, kdy je propojen pohled uživatele přímo s kódem serveru. Hlavními zástupci tohoto přístupu jsou skriptovací jazyk PHP, pro jazyk Java je to pak technologie JSP a například pro C# či VisualBasic je to technologie ASP.NET. Nevýhodou tohoto přístupu jsou hlavně v minulosti naprogramované aplikace, které v jednom zdrojovém kódu kombinovaly více jazyků. Například PHP vygeneruje HTML stránku s JavaScriptem a CSS stylem. Dnešní názor je, že je to nepřehledné a zastaralé. Existují různé připravené knihovny či rámce (*frameworks*), které práci usnadňují a snaží se tento přístup co nejvíce zjednodušit. Pro jazyk PHP je to například projekt Laravel.

#### SPA + BackEnd

Druhý uvažovaný přístup je *SPA + BackEnd*. *SPA* značí klientskou část aplikace a *BackEnd* je serverová část. Tyto dvě části mezi sebou vzájemně komunikují. Na první pohled možná zdánlivě můžou naznačovat složitější přístup, přeci jen vývoj dvou aplikací místo jedné by měl být složitější. Není to ale tak moc znatelné a nabízí to více výhod než MVC přístup.

*SPA* je z anglického *Single Page Application*. Překlad z angličtiny, tj. aplikace v samostatné stránce, je výstižný. Vztahuje se to pouze na klientskou část webové aplikace. Ta funguje tak, že při požadavku od uživatele na server se jako první krok provede poslání právě této klientské *SPA* části aplikace. Klientskou část má uživatel načtenou pouze jednou a pak už může přistupovat k dalším informacím pouze prostřednictvím klientské části. Přístup *SPA* má několik hlavních výhod. První je, že se nemusejí neustále posílat zbytečné informace jako HTML formátování, CSS styly či JavaScriptové části, ale požadují se po serveru pouze čistá data. Což je výhodné z hlediska rychlosti načítání a objemu přenášených dat. Druhá hlavní výhoda je, že se tato *SPA* část napojuje pomocí HTTP protokolu a například REST API, což umožní napojení jiných klientů v budoucnu. Pokud se projekt rozšíří a bude potřeba vytvořit například aplikaci pro mobilní platformu, nemusí se celá logika serveru programovat znovu, ale využije se stávající. Stačí pouze schopnost klientské

části komunikovat pomocí protokolu HTTP, což zvládá prakticky každý jazyk. Další výhodou je zaměnitelnost částí aplikace. Jak již bylo zmíněno, klientská část může být zpřístupněna pro různé jazyky, a tím i pro jiné platformy, ať už web, Android, Windows, Mac a další.

*BackEndová* neboli serverová část aplikace je potom realizovaná samostatně, a to jednoduchým přístupem, kdy se definuje pouze způsob autorizace, vstupní data pro požadavek a pak výstupní data požadavku. Serverová část může být opět v jakémkoliv jazyce, který umí poskytovat komunikaci přes HTTP protokol pro server. Dobrý výběr jazyka pro tuto část je vcelku podstatný. Bylo by sice hezké mít co nejmenší nároky na server a zvolit pokud možno jazyk C, ale bohužel to značně znepráhlední kód a neposkytuje takové výhody jako ostatní, například interpretované jazyky.

### 2.3.2 Výběr konkrétních technologií

#### Klient

Jak pro klientskou část, tak i pro serverovou existují komerční i nekomerční řešení. Vzhledem k tomu, že projekt RTS2 UI by měl být otevřený, využijí pouze otevřené knihovny. Pro výběr webové klientské části jsem zvolil framework *AngularJS*. Je to JavaScriptový framework, který poskytuje snadnou a srozumitelnou architekturu a na vývoji se podílí komunita vývojářů společně s firmou Google. JavaScript samotný poskytuje HTTP rozhraní pro komunikaci se serverem. A dnešní webové prohlížeče podporující HTML 5 standard umožňují mnoho dalšího. Podpora HTML 5 bude určitě hlavní předpoklad pro korektní funkci aplikace. Použijí styl knihovny Bootstrap, který zajistí, že aplikace bude mít estetický a standardní vzhled. Pro uzpůsobení vzhledu jsou dostupné různé styly v případě potřeby. Tato stylová knihovna není pouze o jednoduchosti vzhledu, ale při správném použití umí zajistit korektní zobrazení na různých zařízeních jako tablety a mobilní telefony.

#### Server

U serverové části webových aplikací se dá s výhodou použít přístup aplikačních serverů, které implementují kontejnerovou logiku. Tento přístup má několik výhod a nevýhod. Protože se aplikace publikuje po částech, může na jednom serveru a stejném portu běžet více různých aplikací. Různé části budou mít pak pouze jiné směrování na začátku URL adresy například jako

```
http://app1.server/ nebo http://server/app1/  
http://app2.server/ nebo http://server/app2/.
```

Také u produkčního prostředí jsou zřejmé výhody v tom, že není nutné vypnout další části či celé aplikace, pokud potřebujete nahrát novou verzi aplikace. V neposlední řadě tento přístup umožňuje centralizovanou správu konfigurace serveru pro všechny aplikace. Správa SSL certifikátů a různé další nastavení se řeší pomocí konfigurace serveru a nemusí se o to starat programátor aplikace.

Nevýhodou tohoto přístupu je, že musíte mít aplikační server a musíte jej správně nakonfigurovat. Aplikace musí být kompatibilní s verzí serveru. Jsou zde jak komerční řešení jako je ISS server od Microsoftu, tak otevřené řešení jako je TomCat server od Apache. Určitý problém je také v tom, že pokud je server otevřený, může být lehce zkoumán útočníky. Na starší a neaktualizované verze serveru se objevují běžné útoky, které dokážou zpřístupnit útočnickovi data a kontrolu nad serverem. Pro použití aplikačního serveru je nutné jej udržovat neustále aktuální.

Pro projekt RTS2 UI jsem zvolil jednoduchý HTTP server bez kontejnerové logiky. Důvody jsou takové, že před použitím projektu nemusíte instalovat a konfigurovat jiný software. Obvykle nebývá u takovýchto projektů potřeba, aby na jednom počítači a jednom portu fungovalo několik aplikací najednou a tudíž není nutné mít výhody aplikačního serveru. Projekt RTS2 implementuje server stejným způsobem, tj. využívá jen základního objektu pro HTTP komunikaci a směrování, načítání vstupních dat a další řeší kód aplikace.

Pro serverovou část projektu RTS2 UI jsem se snažil najít jednoduché řešení v jazyce C/C++. V tomto jazyce je napsaný projekt RTS2. Vyhledal jsem několik řešení v podobně knihoven, protože například směrování požadavků se dá vyřešit elegantním způsobem a programátor to nemusí sám řešit pomocí analýzy textového parametru cesty požadavku. Nejlepší byla knihovna POCO [12], která uměla základ protokolu HTTP a mnoho dalšího, ale nezdálo se mi to z hlediska přehlednosti dobré.

Nakonec jsem se rozhodl zvolit jazyk Java. Projekt RTS2 UI je odděleným systémem a není tak podstatné, jaký bude mít jazyk, hlavně když bude kód přehledný. Přehlednost kódu zajistí i vývojové prostředí NetBeans, které ulehčuje pohyb v hierarchii metod a další programátorské úkony.

### 2.3.3 Databáze

Mezi databázovými systémy jsou aktuálně dvě hlavní volby, a to NoSQL databáze, které se používají pro velká data, a pak klasické relační databáze. Protože neočekávám od projektu RTS2 UI zpracování velkého objemu dat a je zřejmý databázový model, využiji raději klasického relačního přístupu databází. Relační přístup nabízí i ošetření vazeb mezi entitami, které bude u projektu RTS2 UI potřeba. Konkrétně byla vybrána databáze PostgreSQL, kterou využívá i projekt RTS2. Je volně použitelná za dodržení podmínek licence MIT.

### 2.3.4 Administrace projektu

Pro jednoduchý vývoj projektu budou použity dvoje vývojová prostředí, a to NetBeans a Visual Studio Code. NetBeans umožňuje přehlednou tvorbu aplikací v jazyce Java, umí i HTML 5 a JavaScript, ale pro tyto dva webové jazyky jsem raději zvolil Visual Studio Code.

Každý rozsáhlejší projekt se neobejde bez systému na správu verzí. Pro tento účel jsem použil program *git*. Aby byl projekt dostupný, využiji služby GitHub, který umožní vedení repozitáře projektu zdarma.

Pro správu a využití ostatních knihoven potřebných k projektu využijí pro serverovou část *Maven* [16] a pro klientskou *bower* [17]. Obě aplikace jsou podobné. Umožňují jednoduchá přidání knihovny z repozitáře aplikace do projektu, také snadnou změnu na novější verzi a řeší i závislosti jednotlivých knihoven.

## 3. Popis zvoleného řešení

Celý softwarový projekt je rozdělen do mnoha vrstev. Každá z vrstev zajišťuje určitou specifickou funkcionalitu, kterou dále podrobně vysvětlím. Základní představu o rozvržení projektu lze získat z diagramu 3.1. První dva řádky diagramu jsou pro vytvoření nadhledu. Poslední řádek obsahuje jednotlivé podbloky, které jsou podrobněji rozkresleny. Ale každý řádek diagramu představuje to samé.

Celý softwarový projekt je dostupný na službě portálu GitHub a konkrétní odkaz naleznete v referenci [11] nebo lze využít tento příkaz:

```
git clone https://github.com/Ramese/rts2_ui.git
```

Repozitář je rozdělen na složku *Database*, *Documentation* a *Project*. Složka *Database* obsahuje SQL skript pro vytvoření nové databáze k softwarovému projektu RTS2 UI. SQL skript by měl být kompatibilní s PostgreSQL a byl vygenerován pomocí programu *pgAdmin*. Složka *Documentation* obsahuje hlavně text této práce a dále například dokumentaci k RTS2 JSON API ve formátu PDF, která ale není plně korektní, nicméně je to nejlepší dostupná verze a byla použita pro vytvoření některé funkcionality. Složka *Project* obsahuje zdrojový kód softwarového projektu RTS2 UI. Složku je možno otevřít ve vývojovém prostředí Netbeans s nainstalovaným rozšířením *Maven*. Pro zkompilování je nutné stáhnout závislosti projektu. Adresář *Web*, který se nachází v adresáři *Project*, obsahuje soubory klientské části projektu.

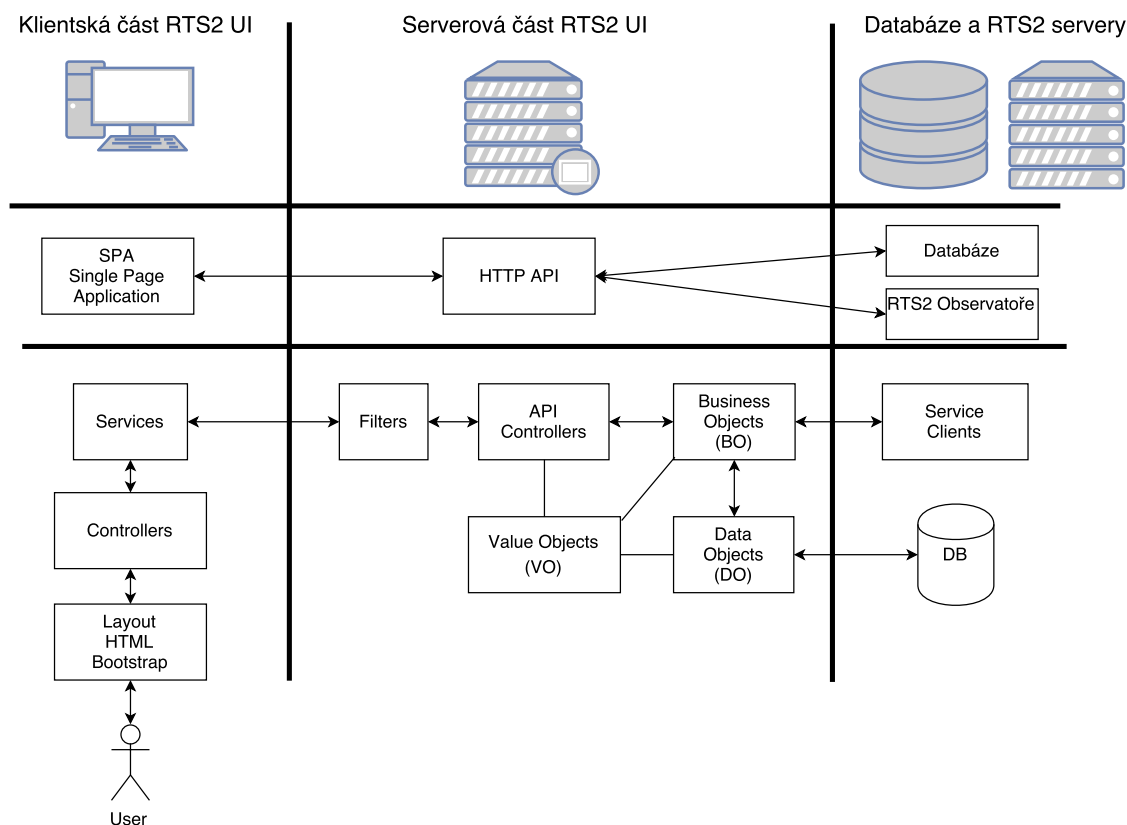
### 3.1 Klientská část RTS2 UI

Jak bylo zmíněno v kapitole týkající se analýzy práce, pro klientskou část projektu byl vybrán JavaScriptový framework *AngularJS*, který je dostupný ze zdroje [13]. Dále klientská část využívá různé přidané knihovny jako projekt *UI Bootstrap*, který doplňuje framework *AngularJS* o mnoho komponent, jako kalendář, posuvník atd. a je dostupný ze zdroje [15]. A další přidané knihovny jako *Font-Awesome*, *Angular-Cookies*, *Angular-Route*, atd.

Adresář *Project/Web* není adresářem, který by obsahoval metadata pro konkrétní vývojové prostředí, lze použít jakékoliv. Jediná metadata, která klientská část obsahuje, jsou závislosti knihoven pro správu aplikací *bower* a *JSHint* [18] pro udržitelnost JavaScriptového kódu. Všechny použité knihovny jsou pak k nalezení v adresáři *Web/bower\_components*.

#### 3.1.1 Hlavní soubor klienta - index.html

Hlavním souborem, který načte celou SPA aplikaci je soubor *index.html*. Ten načte ze serveru všechny skripty a styly, které pak využívá a nepotřebuje je stahovat znovu. Dále obsahuje základní rozvržení stránky tj. vrchní menu a kontejner náhledů pro



Obrázek 3.1: Diagram vrstev navrženého serveru.

vložení obsahu podle požadavku. Obsah se vkládá do elementu `div` s definovaným atributem „`ng-view`“ ve výpisu kódu 3.1.

```
1 <div data-ng-view class="container-fluid"></div>
```

Kód 3.1: Oblast pro zobrazení layoutů.

### 3.1.2 Hlavní soubor JS - `app.js`

Začátek JavaScriptové aplikace lze najít v souboru `Scripts/app.js`. Ten definuje nový modul Angularu, a to modul nazvaný `'app'`, pod kterým celá klientská část běží a navazuje další části aplikace. Specifikuje závislosti přidávaných knihoven a dalších vytvořených částí, jak je vidět v úseku kódu 3.2. Použití tohoto modulu jako hlavního modulu klientské části aplikace, je deklarováno v souboru `index.html` v tagu „`<html data-ng-app='app'>`“.

```
1 var app = angular.module('app', [
2     // ANGULARS MODULES:
3     'ngRoute',
4     ...
5     // 3TH PART MODULES:
6     'ui.bootstrap',
7     ...
8     // OURS MODULES:
```

```

9     'services',
10    'controllers',
11    ...
12  });

```

Kód 3.2: Definice modulu 'app'.

Aby JavaScriptový klient mohl být obslužen serverem, musí se správně identifikovat. Identifikace probíhá pomocí tokenu. Každý jeden požadavek na server nejprve projde klientská část, která je vidět ve výpisu kódu 3.3. Tato část je zodpovědná za přidání hlavičkové informace do HTTP požadavku, a to konkrétně „Token“ a „UserName“. Tyto dvě informace jsou používány místo neustálého posílání uživatelského jména a hesla. Klient je získá při prvotním přihlášení do aplikace. Token je serverovou částí vygenerovaný řetězec, který se nachází v databázi a jeho platnost je osm hodin od posledního provedeného úkonu, aneb při každém požadavku, který je platný, se jeho platnost prodlužuje o dalších osm hodin.

Tato technika autorizace je lepší z důvodu možnosti odchyčení zprávy mezi klientem a serverem. Při použití tokenové autorizace bude mít útočník pouze token, který je platný po určitou dobu. Samozřejmě hrozí odchyčení samotného hesla, a proto by se měl použít protokol HTTPS, který používá asymptotické šifrování.

Token je společně s dalšími informacemi uložen v cookies souborech prohlížeče. Výhodou uložení pouze tokenu a nikoliv jména a hesla je, že tento mechanismus zabraňuje i případným útokům s analýzou dat v cookies, kdy útočník přečte soubor cookies z lokální stanice. Může získat omezený přístup na konkrétní dobu, ale například při odhlášení se token znehodnotí a útočník přístup ztratí.

```

1  $httpProvider.interceptors.push(function ($q, $rootScope, $window,
2    $translate, CONSTANTS) {
3    return {
4      'request': function (config) {
5        if ($rootScope.user !== undefined) {
6          config.headers.Token = $rootScope.user.Token;
7          config.headers.UserName = $rootScope.user.UserName;
8        }
9        return config;
10     },
11     'responseError': function (response) {
12       ...
13       // response error central handler
14       ...
15     }
16   };
17   });

```

Kód 3.3: Centrální řešení požadavků a odpovědí klientské části.

A další část souboru *app.js* řeší směrování pro klienta. Ve výpisu kódu 3.4 je vidět jedna definice směrování. Každá taková definice je pro jednu akci respektive jeden pohled do aplikace a určuje cestu („/interface/“), HTML rozvržení („Views/inter-

face.html“), kontrolér („InterfaceCtrl“), přístup („freeAccess: false“) a nadpis („Interface RTS2 Web“). Specifikace cesty může obsahovat parametry jako je vidět zde „:id“, kdy se předá parametr do kontroléru „InterfaceCtrl“ pomocí servisu *\$routeParams*.

Přístup řeší toto směrování pouze základně, a to na úrovni jestli je uživatel přihlášen. Pokud není, odfiltruje rovnou zbytečný požadavek na server a požádá uživatele o přihlášení. Směrování v klientské části řeší komponenta *Angular-route* dostupná ze zdroje [14].

```

1 $routeProvider
2   .when('/interface/:id', {
3     templateUrl: 'Views/interface.html',
4     controller: 'InterfaceCtrl',
5     freeAccess: false,
6     title: "Interface RTS2 Web"
7   })

```

Kód 3.4: Směrování v klientské části.

Klientská část obsahuje ještě dva další hlavní soubory a to *Scripts/controllers.js* a *Scripts/services.js*. Závislost na těchto modulech je deklarována v části kódu 3.2.

Každý modul může být rozšířen v separátních souborech. Například modul „controllers“ je definován v souboru *Scripts/controllers.js* (viz kód 3.5), ale jeho rozšíření se nachází v každém ze souborů nacházejících se ve složce *Scripts/Controllers* (viz kód 3.6), tj. *InterfaceCtrl.js*, *TelescopeCtrl.js*, *UserCtrl.js* apod.

```

1 angular.module('controllers', [])

```

Kód 3.5: Definice modulu.

```

1 angular.module('controllers')

```

Kód 3.6: Rozšíření modulu.

Hranaté závorky značí definici modulu a zároveň deklaraci závislostí, pokud by byly použity na více místech nebo v různých souborech, budou neustále modul definovat znovu a ztratí předchozí funkcionalitu. V projektu RTS2 UI jsou takto vytvořeny pouze dva moduly, a to moduly „services“ a „controllers“.

### 3.1.3 Kontrolery

Klientské kontrolery představují v diagramu 3.1 blok s názvem „Controllers“. Tato část obsahuje logiku, která je ukrytá na pozadí každého náhledu, tj. při zobrazení formuláře pro přidání teleskopu, logika pro obsluhu událostí formuláře je právě v jednom z kontrolerů. V tomto případě je to kontroler *Controllers/TelescopeCtrl.js*. Kontrolery jsou v projektové složce webové části obsaženy ve složce *Scripts/Controllers* a v souboru *Scripts/controllers.js*.

Každý kontroler obsahuje své paměťové místo v proměnné *\$scope*, které je obousměrně provázáno s pohledem uživatele. Kontrolery umožňují hierarchické uskupení



pod sebe do stromové struktury. Nejvyšší kontroler v této aplikaci je *NavCtrl*, který je definovaný pro celý dokument HTML v tagu *body*, jak je vidět v úseku kódu 3.7.

*NavCtrl* zajišťuje společné funkce pro aplikaci, jako určení role uživatele, a jeho prostor je dostupný ze všech pod kontrolerů. Přístup do kontroleru, který je v hierarchii výše, je realizován pomocí poskytovaného servisu *\$scope*. Ten při zavolání na neexistující proměnnou nebo funkci v místním paměťovém prostoru automaticky zkusí zjistit, jestli nadřazený kontroler implementuje tuto funkci, a pokud ano, rovnou ji vykoná.

```
1 <body data-ng-controller="NavCtrl">
```

Kód 3.7: Použití kontroleru.

Úkolem kontroleru je zpracování dat od uživatelů. Jsou v něm definovány funkce pro ošetření špatných vstupů nebo případně transformace dat do vhodného objektu JSON. Následně komunikuje s některým z jiných kontrolerů nebo použije servis pro komunikaci se serverovou částí, kde opět může získaná data zpracovat do podoby pro uživatele, jako například převedení počtu sekund od data 1.1.1970 na klasický datový formát.

Jedním z kontrolerů je *Error404Ctrl* obsažený v souboru *Scripts/controllers.js*. Tento zajišťuje odpověď na špatné volání neexistujících cest aplikace, tedy server nemusí obsluhovat ani takovéto volání.

### 3.1.4 Rozložení

Rozložení aktuálního pohledu je definováno v klasických souborech HTML. Adresář *Views* obsahuje všechny rozvržení aplikace. Podle cesty v adresovém řádku se použije vhodné rozvržení dle definice v souboru *app.js* jako je ve výpisu kódu 3.4. Požadované rozvržení se umístí do tagu v kódu 3.1 nebo případně do konkrétní specifické části aplikace jako může být definovaná direktiva nebo atribut *ng-include*.

Každé rozložení definuje vzhled dané části a použití proměnných z prostoru kontroleru. Rozložení je s kontrolerem provázáno pomocí servisu *\$scope*. Také definuje omezení vstupu uživatele a případné varování, že by vstup měl být opraven. K těmto definicím se využívají atributy frameworku *AngularJS* začínající na „ng-...“ nebo „data-ng-...“ a také vzhledové knihovny Bootstrap. Je i možné definovat a použít své atributy nebo třídy.

### 3.1.5 Servisy

Servis v této aplikaci zajišťuje převážně komunikaci se serverovou částí, ale může zajišťovat komunikaci mezi kontrolery nebo například jako zde použitý servis *AuthService* správu cookies souborů. Definice modulu *services* se nachází v souboru *Scripts/services.js* a ostatní části jsou pak umístěny ve složce *Scripts/Services*.

Účelem servisu je definovat jednoduché obalové metody pro získání nebo uložení dat. Metoda přijme objekt, který obsahuje parametry požadavku, a vytvoří požadavek podle vzoru. Následně vrátí takzvaný objekt *promise*, který je návrhovým vzorem

pro asynchronní zpracování událostí. Tento objekt může definovat událost vyvolanou, pokud se podaří požadavek dokončit úspěšně, anebo událost, která se vyvolá při neúspěchu.

Ukázka takovéto metody je vidět ve výpisu kódu 3.8. Metoda definuje klasický HTTP požadavek, konkrétně metodou POST, cestu požadavku, hlavičkové informace a případná data, poslaná v obsahu požadavku jako je to specifikováno v metodě POST. Obdobně se dají využít i jiné druhy HTTP metod jako GET, PUT, DELETE atd. Objekt *\$http* je opět servis definovaný frameworkem *AngularJS* a jeho výsledkem je objekt *promise*.

```
1 this.getTelescopes = function (pagination) {
2   return $http({
3     method: "POST",
4     url: CONSTANTS.API_PREFIX + "/telescopes",
5     headers: { 'Content-Type': 'application/json' },
6     data: pagination
7   }).error(function (data) {
8     console.log(data);
9     console.log("TelescopeService error");
10  });
11  };
```

Kód 3.8: Ukázka metody servisu.

Neúspěšné dokončení požadavku je obsluhováno centrálně v souboru *app.js*. Může být však obsluženo na více místech. Ale snahou tohoto navrženého řešení je obsluha špatné události na jednom místě. Podle příznaku, proč došlo k neúspěchu, se vykoná akce. Například server může odpovědět, že je identifikační token neplatný a uživatel musí získat nový, tedy se provede akce odhlášení uživatele společně s vymazáním souborů cookies a zobrazením rozvržení stránky pro přihlášení. Nebo v případě chyby se vypíše uživateli, proč k chybě došlo.

Úspěšné volání požadavku pak skončí obsluhou v kontroleru a vhodným zpracováním dat s následnou prezentací výsledků.

## 3.2 Serverová část RTS2 UI

Repozitář obsahuje projekt vývojového prostředí Netbeans s pluginem Maven ve složce *Project*. Zdrojový kód je pak obsažen ve složce *Project/src*. Projekt má několik závislostí na ostatních knihovnách. Využívá se platforma Javy verze 1.8. To je specifikováno v nastavení projektu v kompilaci pomocí zvolení JDK 1.8. Další závislosti jsou na různých knihovnách, a to pomocí nástroje Maven. Tyto závislosti jsou specifikovány v souboru *Project/pom.xml* a jejich seznam je vidět v tabulce 3.1.

Serverová část softwaru je koncipována tak, aby nebylo třeba instalovat a konfigurovat žádné jiné doplňky a servery pro funkci. Jediné, co je třeba, je databáze PostgreSQL. Stačí spustit program s dostatečným oprávněním.

Knihovna	Verze	Autor	Popis
gson	2.8.0	Google	Pro práci s JSON formátem.
httpClient	4.5.2	Apache	Pro HTTP klienta.
postgresql	9.4.1212.jre7	PostgreSQL	Pro komunikaci s DB.
commons-io	2.5	Apache	Pro standardní knihovní funkce.

Tabulka 3.1: Seznam závislosti na ostatních knihovnách.

### 3.2.1 Konfigurace serveru

Konfigurační část serveru je obsažena v souboru *rts2ui/Config.java*. Je zde možné upravit základní nastavení serveru.

Parametr *GLOBAL\_WEB\_API\_PATH* určuje virtuální cestu na serveru k dotazům pro HTTP klienta a její přednastavená hodnota je „/API“. Pokud například bude server poskytovat službu na cestě „/getUsers“ pro získání uživatelů, bude mít tato služba, stejně jako všechny ostatní, obsaženu předponu ve svém směrování a to „/API“, tedy celá cesta bude „/API/getUser“. Tento princip má výhodu jednoduchého přesunu celé aplikační vrstvy v případě nutnosti, která může nastat například pokud se bude překrývat s nějakým statickým kontextem nebo jinou komponentou s podobným směrováním.

Parametr *GLOBAL\_WEB\_PATH* je obdobný parametr, který ale nastavuje virtuální složku na serveru pro požadavky na statický kontext. Všechny statické texty, ikony, styly atd. jsou dotazovány s touto předponou, aby se opět dalo jednoduše přesměrovat v případě kolize. Přednastavená hodnota tohoto parametru je „rts2“. Tedy hlavní webový interface se zobrazí po zadání požadavku „http://server/rts2“.

Parametr *GLOBAL\_WEB\_FOLDER\_PATH* určuje, kde se fyzicky na serveru nachází složka se statickým kontextem webu, tj. klientská část. Uživatel je pak odstíněn i od informace, kde se tato část reálně na serveru nachází. Obdobně je specifikován parametr „GLOBAL\_WEB\_INDEX\_PATH“ pro určení hlavního souboru klientské části. Tímto souborem se potom odpovídá i na neexistující požadavky. Přednastavená hodnota pro hlavní soubor je „index.html“ a složka klientské části je „/Web“.

Dále jsou ještě vytaženy parametry pro změnu portů. Jak pro klasický HTTP protokol, který využívá hlavní server, tak i pro WebSocket server.

Vše je nastaveno podle současného rozložení projektu. Pokud je potřeba změnit některou část, je nutné, aby nastavení odpovídalo. Klient se musí dozvědět o změně. Například pokud se změní cesta pro požadavky, která je nyní „/API“, musí se nastavit tato skutečnost i u klienta, a to v souboru *Web/Scripts/app.js* v části *CONSTANTS*.

### 3.2.2 Hlavní logika aplikace

Logika je rozdělena na několik vrstev. Základně jsou to *BusinessObjects*, *DataObjects*, *ValueObjects* a *Clients*. Každá vrstva má za úkol konkrétní činnost, kterou dále

vysvětlím. Tento způsob usnadňuje orientaci v projektu a dále umožňuje snadnější nahraditelnost jedné části v případě nutnosti.

### ValueObjects

Je nejjednodušší vrstvou. Jsou to paměťové objekty, které přenáší data v rámci aplikace. Jsou také známy jako DTO neboli z anglického *Data Transfer Object*. Objekt je pak z hlavní části obrazem databázové tabulky, ale může obsahovat i jiné přidané informace. Neměl by však obsahovat žádnou logiku. Zdrojové kódy jsou obsaženy v balíku „VO“ a mají tuto zkratku obsaženu i v názvu objektu, jako například „TelescopeVO“ je objekt uchovávající základní informace o teleskopu.

Obvykle se tato část generuje pomocí objektově relačního databázového nástroje (viz zdroj [7]) jako projekt Hibernate [20]. Ale rozsah databáze projektu není tak velký a byl raději zvolen způsob ročního napsání.

### DataObjects

Tato vrstva objektů zajišťuje ukládání a získávání objektů z databáze pomocí *ValueObjects* vrstvy. Vrstva disponuje připojením na databázi. Mapování objektů z vrstvy *ValueObjects* zajišťuje mapovacími a plnicími funkcemi. Mapovací funkce (*Map*) jsou použity ke konverzi mezi Java objektem a databázovým dotazem. Plnicí funkce (*Fill*) jsou použity pro získání dat z dotazu na Java objekt.

Předpokladem této vrstvy je, že data jsou již ošetřena na jiné úrovni a vrstva je pouze uloží, jak je dostane. Vrstva přidává pouze informaci o posledním uložení entity, která se nachází ve sloupci *UpdateDate*, a době vzniku entity uložené ve sloupci *CreateDate*. V případě neúspěchu uložení nebo načtení vrací vrstva výjimku, nesnaží se jakkoliv napravovat situaci nebo nevrací nelogicky *null*.

Použití vrstvy by mělo být omezeno na nadřazenou vrstvu, která se stará o logickou část. Využití v jiném projektu nebo rozšíření aplikace je možné, ale mělo by být pamatováno na to, že vrstva pouze ukládá a vrací data. Osobně bych zvolil přístup, že by jiná rozšíření aplikace neměla možnost k vrstvě přistoupit a měla by používat nadřazenou aplikační vrstvu i pro ucelení logiky v jednom místě. Tuto skutečnost lze dosáhnout metodou připojení datové vrstvy jako privátní k vrstvě logické. Využití privátní dostupnosti ale zakáže přístup k databázi pomocí základních metod i v případě vygenerování knihovny. Raději jsem zamezování přístupu nepoužil a vrstva je dosažitelná z celé části aplikace jako statická knihovna.

Zdrojový kód této vrstvy se pak nachází v balíku „DO“ a jednotlivé objekty obsahují zkratku i v názvu. Příkladem objektu z této vrstvy je objekt *UserDO*, který se stará o práce s daty ohledně databázové tabulky *User*.

Pro připojení do databáze se používá PostgreSQL komponenty ze standardní knihovny.

## Clients

Vrstva *Clients* zajišťuje komunikační rozhraní mezi serverem RTS2 UI a ostatními službami. V případě projektu RTS2 UI je to připojení na další observatoře se systémem RTS2. Obecně to pak může být například veřejně dostupný insolvenční registr ČR, či jiné služby. Komunikace je pak dostupná přes standardní protokol.

U většiny těchto webových servisů se využívá mechanismus WSDL, který jednoduše zajistí klientské části vygenerování datových typů, objektů a metod pro komunikaci. Projekt RTS2 používá HTTP protokol pro poskytování a získávání dat a z toho se nedá jednoduše vygenerovat klientský kód. Klientský kód pro komunikaci s RTS2 observatoří byl realizován ručním přepsáním, a to konkrétně v souboru *client/RTS2Client.java*.

Účelem klientského objektu je poskytnutí standardních metod pro volání služby. Objekt pak následně podle požadovaných informací zformuluje dotaz podle protokolu HTTP a dokumentace [1] a vrátí odpověď. Všechny realizované metody fungují synchronním přístupem. Aby se nemusel implementovat protokol HTTP pro komunikaci, byla využita knihovna *httpClient* od Apache (viz zdroj [19]).

## BusinessObjects

Vstupní bod logické části aplikace je vrstva *BusinessObjects*. Vrstvu lze pak připojit k jednotlivým úkonům uživatele, ať už z grafického rozhraní nebo HTTP požadavku. Účelem této vrstvy je převzetí údajů a následného logického zpracování složeného třeba i z několika menších úkonů z nižších vrstev. K realizaci úkonů může vrstva využít databázovou vrstvu *DataObjects* nebo *Clients* za pomoci vrstvy hodnotových objektů *ValueObjects*.

Vrstva by měla ošetřovat výjimky na úrovni vrácení změn, pokud něco nefunguje, jak má. Případně nahlásí stav, ve kterém byla činnost neočekávaně ukončena. Výjimky pak předává vyšším vrstvám pro prezentaci uživateli a případný záznam pro opravu chyb. Zdrojový kód vrstvy se nachází v balíku „BO“ a objekty obsahují v názvu stejnou zkratku.

Příkladem objektu vrstvy je *ImageBO*. Tento objekt se stará o práci z obrázky a je na něm vidět využití logické vrstvy. Při požadavku od uživatele na konkrétní obrázek se provede nejprve dotaz do databáze RTS2 UI. Podle výsledku se provede další akce. Pokud obrázek již v databázi existuje, předá se uživateli bez nutnosti kontaktovat RTS2 observatoř. Pokud obrázek není v databázi ještě obsažen, požádá se o něj u observatoře. Data nově získaného obrázku se pak ukládají do RTS2 UI databáze pro budoucí požadavky a předají se uživateli.

### 3.2.3 HTTP rozhraní serverové části

Protože je aplikace webová a byla zvolena architektura *SPA + BackEnd*, nebude serverová aplikace připojena na grafické rozhraní přímo, ale pomocí definovaných metod HTTP protokolu. Server je jednoduchý, využívá pouze objektu ze standardní knihovny Javy. Konkrétně se jedná o *HttpServer*.

Hlavní třída, která řídí a inicializuje serverovou část, se nazývá *MainServer* a nachází se v balíku *webcontroller*. Je složena z HTTP a WebSocket serveru. Samotná třída *MainServer* provádí inicializaci a nezbytné úkony pro spuštění části serveru.

## HTTP server

HTTP server je část kódu, která využívá objekt *HttpServer*. Ten je definován jako seznam obsluh událostí na jednotlivá volání. Každá obsluha je pak samostatně obsažena v objektu *Controller*, který implementuje rozhraní *HttpHandler*.

Rozhraní *HttpHandler* obsahuje metodu *handle()*, kterou je nutno nadefinovat společně s cestou požadavku. Objekt *Controller* pak přepisuje tuto metodu *handle()* na konečnou (*final*), kvůli zapouzdření chytání výjimek, které nečekaně nastanou. Objekt *Controller* umožňuje nadefinovat obsluhu požadavku v metodě *inerHandle()*, kterou interně volá metoda rozhraní *handle()*. Tento mechanismus je zvolen z důvodu stability serveru a definice záznamu chyb na jednom místě. Přepsání metody objektu *Controller* je vidět ve výpisu kódu 3.9.

```

1 @Override
2 public final void handle(HttpExchange he) throws IOException {
3     try {
4         inerHandle(he);
5     } catch (Exception ex) {
6         // log ...
7     }
8 }
```

Kód 3.9: Zapouzdření obsluhy HTTP žádosti.

Pro definice jednotlivých obsluh události jsem pak zvolil objekty, které pouze vyrobí objekt typu *Controller*, a jsou to objekty obsažené v balíku *webcontroller.controllers*. Jejich struktura je volná, ale snahou je, aby společné téma, jako je například uložení nebo načtení stejného typu objektu, bylo obsaženo v jednom souboru. Příkladem je společná definice obsluh událostí pro databázový objekt uživatele, která je definována v souboru *UserController*. Obsahuje obsluhu pro požadavek k dostání dat všech uživatelů „/getUsers“, jednoho uživatele „/getUser“ a uložení dat uživatele „/saveUser“. Definice obsluhy pro uložení uživatele je vidět ve výpisu kódu 3.10.

```

1 private Controller SaveUserCtrl() {
2     return new Controller("/saveUser", true) {
3         @Override
4         public void inerHandle(HttpExchange t) throws Exception {
5             if(GLOBAL_DEBUG)
6                 System.out.println("SaveUserCtrl:");
7
8             UserVO userFromJSON = (UserVO)GetObjectFromBody(t.
9                 getRequestBody(), UserVO.class);
10
11             UserVO user = (UserVO)t.getAttribute("user");
```

```

11         UserBO.UpdateUser(userFromJSON);
12
13         Controller.SendGoodResponse(true, t);
14     }
15 };
16 };
17 }

```

Kód 3.10: Metoda pro obsluhu požadavku na uložení uživatele.

Všechny požadavky na server jsou realizovány pomocí HTTP metody POST. V těle HTTP požadavku je obsažen JSON objekt, který obsahuje potřebné data pro realizaci požadavku na straně serveru. Data z formátu JSON jsou převedena na klasický objekt pomocí knihovny *json*. To je zapouzdřeno do metody *GetObjectFromBody()*, kde je převod definován.

Obsluha tedy načte data, pokusí se provést požadovaný úkon, a pokud bylo vše úspěšné, vrátí kladnou odpověď. Odpověď obsahuje opět data ve formátu JSON a o konverzi z Java objektu do JSON formátu se také stará knihovna *json*, konkrétně v metodě *SendGoodResponse()*.

Pokud požadavek není z nějakého důvodu úspěšně zpracován, jsou definované standardní odpovědi přímo jako metody obsluhy, na které patřičně reaguje klientská strana. Například při zavolání metody *NotLoggedIn()* z objektu *Controller*, má klientská strana definováno v souboru *Scripts/app.js*, že se musí odhlásit a smazat soubory cookies, protože uživatel není přihlášen nebo nemá platný token pro identifikaci.

Z objektu *Controller* se tvoří objekt *HttpContext*, který je podobjekt objektu *HttpRequest*. *HttpContext* může mít definovaný řetěz obsluh požadavku, který je provázaný a následuje v řadě za sebou. Jakýkoliv článek v řetězu může přerušit vykonávání požadavku a vrátit chybu nebo odpověď. Tento řetěz využívám pro získání dat o uživateli. Je definován filtr, který se předradí před obsluhy, které to vyžadují. Název objektu obsahující filtrovací metodu je *TokenFilter* a je obsažen v balíku *webcontroller.filter*. Hlavní metodu je možné vidět ve výpisu kódu 3.11.

```

1  @Override
2  public void doFilter(HttpContext he, Chain chain) {
3      try {
4          //...
5          if(!(headers.containsKey("Token") && headers.containsKey("
6              UserName"))){
7              Controller.TokenUnparsable("No token info in header.", he);
8              return;
9          }
10         String token = headers.get("Token").get(0);
11         String userName = headers.get("UserName").get(0);
12         UserVO user = UserBO.IsLoggedIn(token, userName);
13
14         if(user == null) {
15             Controller.NotLoggedIn("Token expired.", he);
16             return;

```

```
16     } else {
17         he.setAttribute(USER_OBJECT_STRING, user);
18     }
19     chain.doFilter(he);
20 } catch (Exception ex) {
21     // ...
22     // return server failed
23 }
24 }
25 }
```

Kód 3.11: Hlavní metoda filtru pro tokenovou autorizaci požadavku.

Účelem tohoto filtru je přečíst hlavičku požadavku, podívat se jestli obsahuje token ve správné podobě. Pokud neobsahuje, rovnou odpovídá klientovi chybou, a ten následně smaže cookies a zobrazí uživateli možnost se znovu přihlásit. Pokud obsahuje požadavek token, provede se poté dotaz do databáze a v případě platnosti tokenu se jeho platnost opět prodlouží o nějaký konstantní čas (konkrétně 8 hodin). Pokud token již není platný, neumožní se uživateli přístup k další části požadavku a odpoví se klientovi tak, aby uživatele identifikoval znovu.

V případě platného tokenu se vytažený záznam dat uživatele z databáze rovnou použije jako atribut v objektu *HttpExchange*, který si jednotlivé služby mezi sebou vzájemně předávají a mohou do něj ukládat jiné podobjekty. Konkrétně se tento objekt uloží pod názvem „user“ a ostatní služby požadavků jej mohou přečíst bez volání z databáze. Služby za tokenovým filtrem mohou předpokládat, že je uživatel ověřen, a pracovat se získanými daty bez nutnosti znovuověření. Tímto způsobem je realizováno ověření uživatele na straně HTTP serveru.

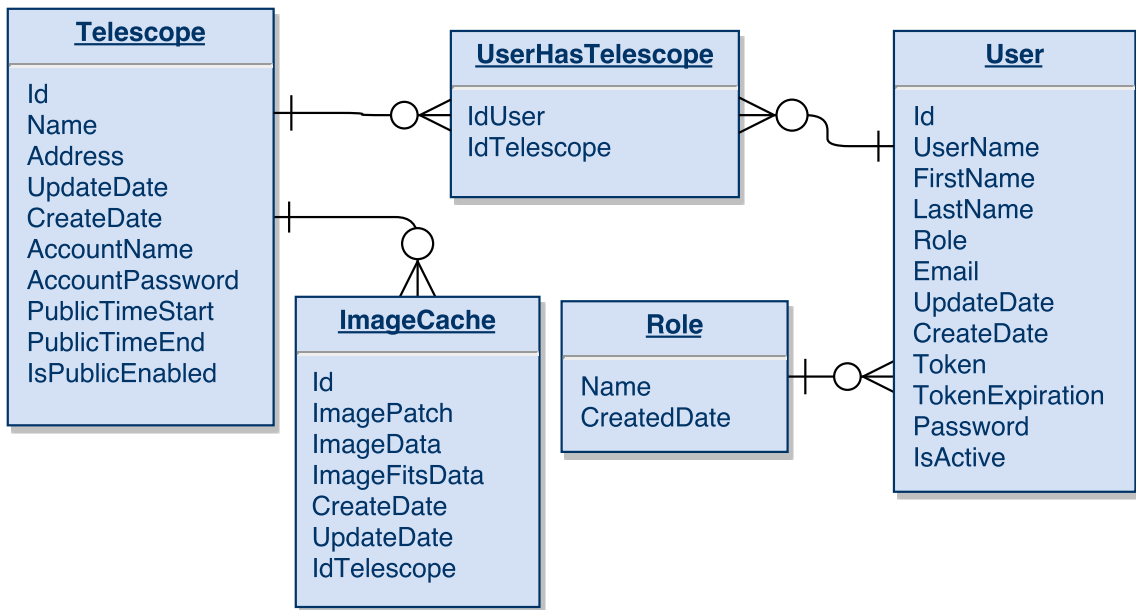
HTTP server je obsluhován vícevláknově. Konkrétně byl zvolen přístup vytvoření počtu vláken, které se starají o obsluhu klienta, podle aktuální potřeby. Pokud je vlákno ukončeno, zůstává po dobu jedné minuty v paměti. Pokud do té doby není využito, tak zaniká, aby uvolnilo systémové prostředky.

## WebSocket server

Druhou serverovou částí je WebSocket server. Ten je implementován základně podle dokumentace ve zdroji [8]. Základem je klasická socketová komunikace, které eviduje jednotlivé klienty. Každý klient nejprve naváže komunikaci pomocí HTTP protokolu a požádá v něm o změnu komunikace na základní socketovou komunikaci. Tato změna tzv. *handshake* je implementován podle dokumentace a vyzkoušen v souboru *Web/websocket.html*, který obsahuje JavaScriptového klienta.

Výhoda WebSocketové komunikace oproti HTTP komunikaci je možnost kontaktovat klienta přímo ze serveru. Server pak nemusí být přetěžován požadavky, ať už periodickými nebo pozastavenými, které mají odpověď v době změny.





Obrázek 3.2: Schéma databáze RTS2 UI.

### 3.3 Databáze RTS2 UI

Databáze projektu RTS2 UI je navržena, aby zbytečně netvořila duplicity. Pokud je možné získat informaci z observatoře, získá se z observatoře. Ukládají se pouze data, která souvisí přímo s aplikací.

Potřeba je uložit informace o uživateli, kteří budou systémem využívat, jejich identifikace a případně další údaje. A každý uživatel bude mít evidovanou roli, pod kterou v systému vystupuje.

Dále se musí uložit informace o observatořích, které systémem RTS2 UI může využít. Je potřeba uložit jejich veřejně dostupná internetová adresa a údaje pro přihlášení. Údaje pro přihlášení se ukládají pouze v textové podobě, protože je potřeba znát heslo. A další atributy, jako například veřejné časy observatoře nebo pojmenování.

Protože oprávnění *scientist* má mít možnost přistupovat pouze k přiděleným observatořím, je tato informace uchována v databázi RTS2 UI. Informace se ukládá v jednoduché tabulce, které nahrazuje vazbu N:M a skládá se pouze z identifikace uživatele a teleskopu.

Data obrázků se uchovávají v databázi RTS2 UI. Je to z důvodu odlehčení toku dat mezi observatořemi a serverem RTS2 UI. Pro jednoduchost je uložen formát JPEG i FITS. Formát JPEG by se dal získat z FITS formátu, ale systémem zatím nepracuje s knihovnou, která by tuto konverzi umožnila.

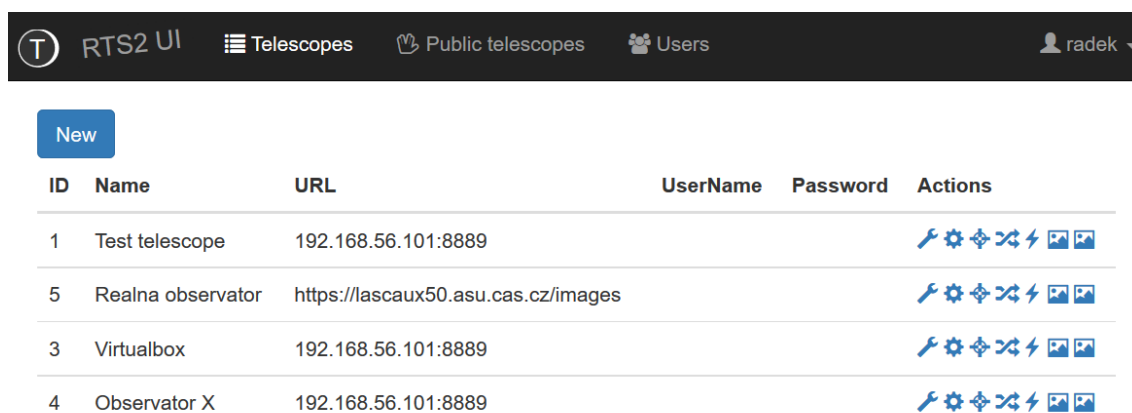
Databáze je navržena s ohledem na relační vazby. Software nemusí nutně hlídat, jestli je integrita dat v pořádku.



## 4. Popis funkce

### 4.1 Správa observatoří

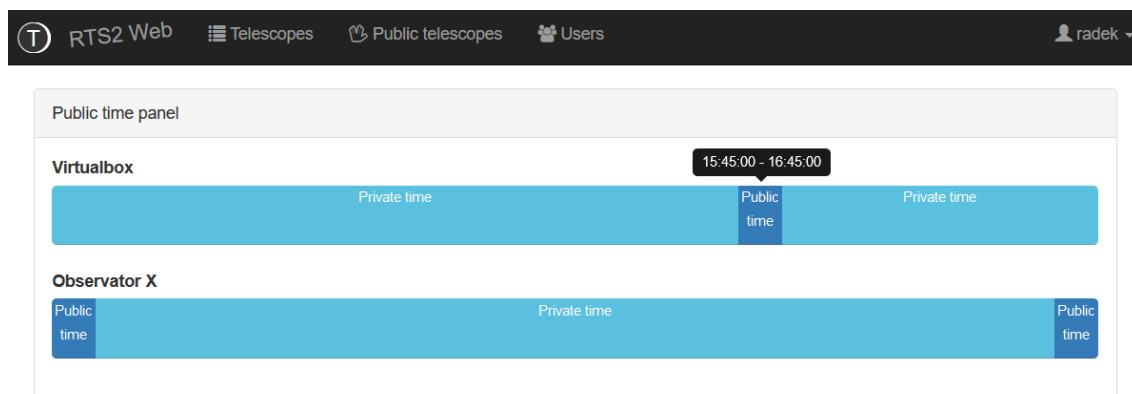
Projekt RTS2 UI umožňuje připojení na více observatoří se systémem RTS2. Seznam těchto připojených systémů je uložený v lokální databázi RTS2 UI serveru. Každý přípojný bod se skládá z názvu, informace o veřejně dostupné adrese, identifikaci a dalších vlastností. Seznam je vidět na obrázku 4.1. Přístup je možný pro oprávnění administrátora a vědce z hlavního menu. Nahoře vlevo je umístěno tlačítko „New“ pro přidání dalšího teleskopu.



ID	Name	URL	UserName	Password	Actions
1	Test telescope	192.168.56.101:8889			
5	Realna observator	https://lascaux50.asu.cas.cz/images			
3	Virtualbox	192.168.56.101:8889			
4	Observator X	192.168.56.101:8889			

Obrázek 4.1: Rozhraní RTS2 UI - seznam teleskopů.

Pro veřejné účty a nezaregistrované je možnost prohlížet veřejné observatoře. Jedinými zobrazenými informacemi jsou název observatoře a rozsah veřejného času. Obrázek 4.2 ukazuje seznam s grafickým zobrazením času pro veřejnost.



Obrázek 4.2: Rozhraní RTS2 UI - seznam teleskopů pro veřejnost.

Každý řádek v seznamu na obrázku 4.1 umožňuje otevření dalších akcí. První akcí s ikonkou klíče se uživatel dostane k formuláři pro správu vybraného teleskopu. Formulář je zobrazen na obrázku 4.3. Je zde zpřístupněno nastavení databázového

záznamu projektu RTS2 UI s možností záznam změnit pomocí tlačítka „Save“. Dále je pro jednoduché ověření správnosti veřejné internetové adresy a zadaných údajů implementováno načtení obrázku z cesty „/current/“. Obrázek je vidět v pravé části a zobrazuje informace v grafické podobě o aktuálním nasměrování teleskopu a jeho další cíl.

The screenshot shows the 'RTS2 Web' interface. At the top, there is a navigation bar with 'Telescopes', 'Public telescopes', and 'Users' menus, and a user profile 'radek'. The main content area is divided into two sections. On the left is a form for editing telescope properties:

- Id:** 3
- Name:** Virtualbox (with a green checkmark)
- URL:** 192.168.56.101:8889 (with a green checkmark)
- Account Name:** test (with a green checkmark)
- Account Password:** test (with a green checkmark)
- Public hours:** A checked checkbox.
- Start - End:** Four empty input fields separated by colons.
- Save:** A blue button at the bottom of the form.

On the right is the 'Other properties' section, which features a star chart. The chart is a circular representation of the sky with a grid of right ascension and declination. It shows a 'Default model' in green, a red circle labeled 'TEL', and a blue circle labeled '107720'. A blue 'Regresh' button is located below the chart.

Obrázek 4.3: Rozhraní RTS2 UI - formulář pro správu teleskopu.

Druhá ikonka v seznamu 4.1, a to ikonka ozubeného kola, umožňuje uživateli zobrazení a správu zařízení, kterými teleskop disponuje. Po kliknutí se zobrazí rozhraní z obrázku 4.5 a podrobnější informace jsou popsány v sekci 4.2.

Třetí ikonka v seznamu 4.1 se zaměřovačem umožňuje uživateli zobrazit správu pozorovacích cílů na konkrétní observatoři. Po kliknutí se zobrazí rozhraní z obrázku 4.13 a více informací o administraci cílů pozorování se nachází v sekci 4.3.

Čtvrtá ikonka slouží pro zobrazení front, které jsou dostupné v zařízení SEL („selector“). Po kliknutí se zobrazí okno 4.16 a více informací ohledně práce s frontami, se nachází v sekci 4.4.

Pátá ikonka s bleskem slouží k zobrazení informací o aktuálním pozorovacím plánu. Informace se načítají ze zařízení s názvem EXEC („executor“). Po kliknutí se zobrazí seznam z obrázku 4.17 a více informací je dostupných v sekci 4.4.3.

Poslední dvě ikonky slouží k přístupu k napozorovaným datům. První je podle data pořízení snímku a po kliknutí se zobrazí přehled z obrázku 4.18 a více informací se nachází v sekci 4.5.1. Druhá ikonka slouží pro zobrazení snímků ke konkrétnímu cíli pozorování. Po kliknutí se zobrazí přehled z obrázku 4.19 a více informací se nachází v sekci 4.5.2.

Aby byl umožněn rychlejší a intuitivnější pohyb uživatele mezi jednotlivými pohledy, byl přidán do pravého horního rohu panel s těmito výše popsány ikonkami. Většina snímků aplikace byla pořízena před tímto přidáním a menu se tam ještě nenachází, ale například snímek 4.15 již toto menu vpravo nahoře má.

## 4.2 Ovládaní zařízení

Jedním z bodů práce je zpřístupnění možnosti ovládat zařízení observatoře. Aktuálně je tato možnost realizována přes konzolové aplikace, jako například *rts2-mon*. Zpřístupnění ovládaní by mělo být robustní z pohledu změn. Nemá moc smysl napsat statický kód a při změně zařízení, ať už názvu nebo nějakého z parametrů, kód opět přepisovat.

K umožnění ovládaní jednotlivých zařízení observatoře jsem využil metody HTTP serveru RTS2, které následně v textu popíšu a uvedu v tabulkách. Více informací o metodách a dalších parametrech se nachází ve zdroji [1]. V tomto dokumentu lze nalézt mnoho informací, ale ne všechny. Například struktura binárních dat obrázku byla nalezena ve zdrojovém kódu.

Příkladem typického zařízení observatoře je senzor pro pořizování fotografií oblohy. Ten je u RTS2 pojmenován Cn, kde n je celé číslo a může jich existovat více v jednom systému. Aktuální možnost nastavení zařízení senzoru je vidět na obrázku 4.4, který zobrazuje konzolovou aplikaci *rts2-mon*.

```

System      States      Debug      Help      radek-VirtualBox
C0 0 idle | SHUTTER CLEARED
centr      infotime    2017-05-13T19:28:25.968 CET (-32s)
C0         focuser     F0
CUP        wheel       W0
EXEC       wheelA      W0
F0         chips       1
HTTPD     W SCRIPREP  x
IMG        SCRIPT
SEL        W SCR_COMM
T0         W COMM_NUM  0
W0         W script_status 0 run
monit     W scriptPosition 0
statu     W scriptLen  0
          W elementPosition
          W CCD_TYPE   Dummy
          W CCD_SER    1
          W IMAGETYP  0 object
          W OBJECT
          W calculate stat 0 yes
D: "script loop count"

HARD OFF evening 31 | F9 menu F10 exit | 0 0|LST 09:55:36|2017-05-13 17:28:58

```

Obrázek 4.4: Rozhraní aplikace *rts2-mon*.

Obdobnou logiku nastavování parametrů jednotlivých zařízení jsem se snažil zachovat i ve webovém rozhraní. Uživateli se zobrazí seznam zařízení, jako je vidět v levém sloupci *rts2-mon*, a po následném výběru se vygeneruje formulář pro jednotlivé vlastnosti. Pro získání seznamu zařízení jsem využil metodu „/api/devices“, která vrátí seznam všech zařízení až na zařízení zvané *centrald*. Více informací o me-

Cesta	/api/devices	Pro získání seznamu zařízení dostupných na konkrétním RTS2 serveru.
Parametr	-	
Výsledek	[”C0”, ...]	Pole s textovými názvy zařízení, které observatoř používá.

Tabulka 4.1: Metoda pro získání seznamu zařízení systému RTS2.

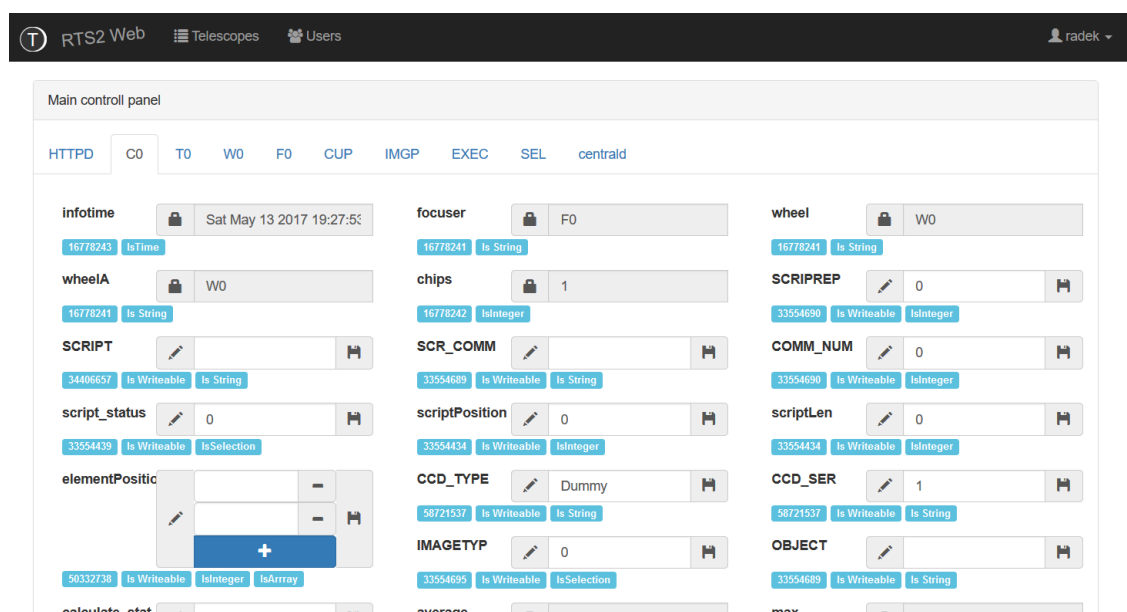
toď se nachází v tabulce 4.1.

Zařízení *centrald* umožňuje nastavit den a noc, také vyčíst pozici slunce, měsíce a další. Předpokládám, že je toto zařízení dostupné u všech RTS2 systémů, a tedy jsem jej pevně přidal do seznamu, aby uživatelé měli možnost pracovat i s tímto zařízením.

Po zvolení zařízení se požádá o načtení jednotlivých parametrů pomocí metody „/api/get“. Informace o této metodě jsou vidět v tabulce 4.2. Z obdržených dat se pak dynamicky generuje formulář. Obdoba rozhraní z obrázku 4.4 je vidět v nové verzi na obrázku 4.5.

Cesta	/api/get	Pro získání seznamu vlastností a jejich hodnot u jednotlivých zařízení.
Parametr	<b>d</b>	Specifikuje vybrané zařízení.
Parametr	e	0 nebo 1. Pokud je 1, hodnoty jsou obsáhlejší.
Výsledek	JSON objekt.	Seznam parametrů daného zařízení.

Tabulka 4.2: Metoda pro získání parametrů zařízení.

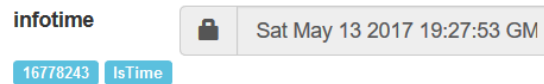


Obrázek 4.5: Rozhraní RTS2 UI.

Formulář je generován na základě informací obsažených v jednom z parametrů dané vlastnosti. Podle hodnoty lze poznat, jestli umožňuje zápis nebo pouze čtení.

Dále lze rozeznat jednotlivé datové typy a jestli je proměnná speciální jako třeba pole, čtverec atd.

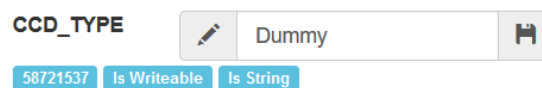
Pokud je parametr typu čas, jeho hodnota se změní na uživatelsky čitelnou hodnotu při načtení. RTS2 systém využívá počet sekund od data 1.1.1970, což není uživatelsky přívětivé. Příkladem je hodnota „infotime“ a je vidět na obrázku 4.6.



Obrázek 4.6: Rozhraní RTS2 UI - zobrazení času.

U parametru „infotime“ z obrázku 4.6 je vidět, že je vlastnost zařízení neměnná. To je pak naznačeno zámekem po levé straně a nemožností uživatelského zápisu do formulářového pole.

Pokud je parametr datového typu *string*, *integer*, *long*, *double* nebo *float*, je načteno bez jakékoliv konverze hodnoty. Případný zápis je povolen klasickým textovým vstupem. Příkladem je *string* hodnota „CCD\_TYPE“, která je vidět na obrázku 4.7.



Obrázek 4.7: Rozhraní RTS2 UI - zobrazení běžných datových typů.

Na obrázku 4.7 je také vidět zobrazení parametrů, které umožňují zápis. Je povolen uživatelský vstup do formulářového prvku a v pravé části se nachází tlačítko pro uložení. Uložení se provede pouze u dané hodnoty, nikoliv celého formuláře.

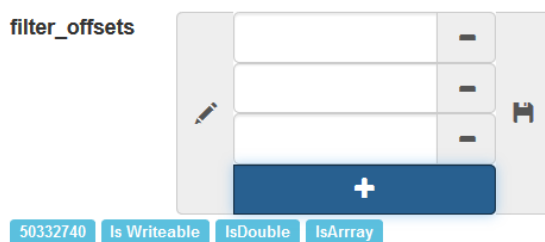
Zobrazení formulářového prvku pro specifický datový typ, jako je logická proměnná typu *boolean* nazvaná „center\_cal“, je vidět na obrázku 4.8. Prvek je realizován pouze pomocí zaškrtačacího formulářového prvku a případného tlačítka pro uložení.



Obrázek 4.8: Rozhraní RTS2 UI - zobrazení logického typu.

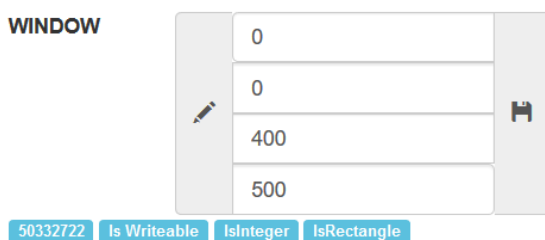
Pokud uživatel neví, co znamená konkrétní proměnná jako například „center\_cal“, může využít nápovědy, která se zobrazí při najetí ukazatele na konkrétní formulářový prvek nebo na název parametru. Ukázka je vidět na obrázku 4.8, kde je zobrazen text „calculate center box statistics“.

Pokud je parametr složitější, jako například datový typ pole, je jeho zobrazení realizováno pomocí několika vstupních polí. Zároveň umožňuje odebrání konkrétního prvku tlačítkem se symbolem mínus a přidání nového prvku tlačítkem se symbolem plus. Příkladem je parametru s názvem „filter\_offsets“ a ukázka formuláře je vidět na obrázku 4.9.



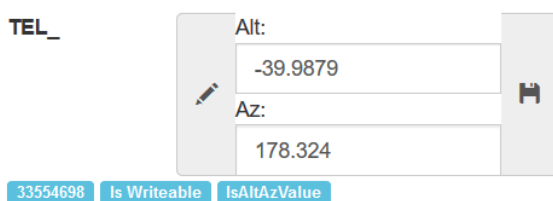
Obrázek 4.9: Rozhraní RTS2 UI - zobrazení parametru typu pole.

Specifický parametr je datový typ obdélník, který obsahuje 4 hodnoty v poli. Od klasického objektu pole se liší pouze tím, že je zakázáno přidávání nebo odebrání hodnot a je umožněna editace pouze 4 hodnot. Příkladem je parametr s názvem „WINDOW“ a jeho zobrazení je vidět na obrázku 4.10.



Obrázek 4.10: Rozhraní RTS2 UI - zobrazení parametru typu obdélník.

Posledním typem parametru je objekt, který obsahuje konkrétní hodnoty v asociativním poli. Příkladem je objekt obsahující souřadnice RA a DEC nebo Alt a Az. Parametr s názvem „TEL\_“ je takovýmto objektem a zobrazení je vidět na obrázku 4.11.



Obrázek 4.11: Rozhraní RTS2 UI - zobrazení parametru typu objekt Alt Az souřadnic.

Ukládání jednotlivých typů parametrů se liší tím, jaký je to datový typ. Například pro pole se nemůže použít formát JSON vypsáný do textového řetězce, protože systém RTS2 to takto neumožňuje. Pole se musí převést na hodnoty oddělené mezerou a následně poslat požadavek k uložení. Obdobně jsem měl problém s datovým typem *boolean*, který jsem raději převedl na hodnotu 0 nebo 1. A objekt se souřadnicemi se musí zkonvertovat na výpis hodnot za sebou v obdržném pořadí oddělených mezerou, obdobně jako pole.

Metoda pro ukládání je „/api/set“ a umožňuje ukládat jednotlivé parametry. Nutno je specifikovat zařízení, kterého se změna týká, následně název parametru

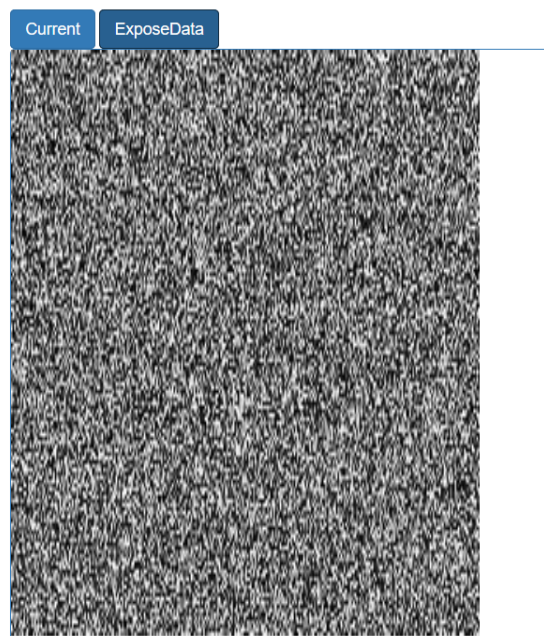


a novou hodnotu. Popis metody je vidět v tabulce 4.3. Komplikace u této metody je, že vždy odpoví kladně, i pokud uživatel zadá neplatnou hodnotu, vrací metoda návratový kód číslo 200 a zprávu „OK“.

Cesta	/api/set	Pro změnu konkrétní hodnoty.
Parametr	<b>d</b>	Specifikuje vybrané zařízení.
Parametr	<b>n</b>	Specifikuje název vybraného parametru.
Parametr	<b>v</b>	Specifikuje novou hodnotu vybraného parametru.
Výsledek	JSON objekt.	Vrací zpět všechny parametry zařízení.

Tabulka 4.3: Metoda pro uložení parametru zařízení.

U zařízení, která jsou pojmenována jako kamera tj. C0, C1, atd., je umožněno získat i aktuální obrázek. Dole pod všemi parametry je přidána část formuláře, která obsahuje dvě tlačítka a to „Current“ a „ExposeDate“. Pod tlačítka je pak objekt, který zobrazuje získaný obrázek. Protože jsou data binární, server RTS2 UI je převede do JSON formátu a následně předá klientské části. Data jsou v binárním poli. Obsahují pouze jednu informaci o pixelu, ale v 16 bitovém rozlišení. Pro vykreslení se používá prvek standardu HTML 5 a to *canvas*. Data jsou převedena do nižšího rozlišení tj. 8 bitů a je zajištěno černobílé zobrazení pomocí přepsání hodnoty do všech barevných kanálu RGB. Část formuláře je vidět na obrázku 4.12, který zobrazuje pouze náhodný šum generovaný pokusným serverem.



Obrázek 4.12: Rozhraní RTS2 UI - zobrazení aktuálního snímku.

Tlačítkem s názvem „Current“ se získá současný obrázek ze zvoleného zařízení. Pokud žádný obrázek není, nic se nezobrazí. Při kladné odpovědi se automaticky přizpůsobí objekt *canvas* rozměry a provede se vykreslení dat. Použitá metoda je „/api/currentimage“ a lepší popis je v tabulce 4.4.

Cesta	/api/currentimage	Pro získání současného obrázku pořízeného senzorem. Nemusí být žádný.
Parametr	<b>ccd</b>	Specifikuje, který senzor bude vybrán.
Parametr	chan	Specifikuje, který kanál senzoru bude vybrán. Přednastavená hodnota je 0.
Výsledek	Binární data.	42 byte hlavička s popisem dat. Zbytek dat je obsah obrázku.

Tabulka 4.4: Metoda pro získání současného snímku.

Druhým tlačítkem s názvem „ExposeData“ se provede expozice pomocí zvoleného senzoru. Použitá metoda je „/api/exposedata“ a lepší popis je v tabulce 4.5. Funguje obdobně jako metoda pro získání současného snímku.

Cesta	/api/exposedata	Začne expozici na senzoru a vrátí data pořízeného snímku.
Parametr	<b>ccd</b>	Specifikuje, který senzor bude vybrán.
Parametr	chan	Specifikuje, který kanál senzoru bude vybrán. Přednastavená hodnota je 0.
Výsledek	Binární data.	42 byte hlavička s popisem dat. Zbytek dat je obsah obrázku.

Tabulka 4.5: Metoda pro spuštění expozice u zařízení typu kamera.

## 4.3 Správa cílů pozorování

Pozorovací cíl je místo na obloze určené souřadnicemi a názvem. Pro různé observatoře se může stejné místo lišit názvem, protože si každá observatoř vede seznam ve své vlastní databázi. Zobrazení pozorovacích cílů je přichystáno několika způsoby.

### 4.3.1 Seznam cílů

První způsob je požadavek na všechny cíle, který pro získání dat využívá metodu „/targets/api“. Podrobnější informace o metodě lze najít v tabulce 4.6. Cesta metody je správně a dokumentace [1] má cestu metody špatně ve tvaru „/api/targets“, který by vzhledem k ostatním metodám byl očekávanější. Metoda vrací navíc například sloupec „Enabled“. Ostatní metody nevrací tuto hodnotu. Problém je v rychlosti. Při pokusu o získání dat z reálné observatoře trval požadavek celkem dlouho.

Využil jsem i jiné metody pro přístup k pozorovacím cílům a to „/api/tbyname“. Metoda vrací seznam pozorovacích cílů podle zadaného jména nebo jeho části. Metoda neodlišuje velká a malá písmena. Umožňuje částečné shody a další. Její tvar je vidět v tabulce 4.7.

Seznam pozorovacích cílů, realizovaný aplikací RTS2 UI, je vidět na obrázku 4.13. Při otevření formuláře je automaticky volena volba hledání. Uživatel může

Cesta	/targets/api	Získání seznamu všech pozorovacích cílů.
Parametr	-	
Výsledek	JSON objekt.	Struktura obsahující hlavičku tabulky a data.

Tabulka 4.6: Metoda pro získání seznamu všech pozorovacích cílů.

Cesta	/api/tbyname	Získání seznamu specifických pozorovacích cílů.
Parametr	<b>n</b>	Specifikuje hledaný text.
Parametr	<b>e</b>	Specifikuje, že metoda vrací podrobný výpis.
Výsledek	JSON objekt.	Struktura obsahující hlavičku tabulky a data.

Tabulka 4.7: Metoda pro získání seznamu specifických pozorovacích cílů.

přepnout na tabulku všech cílů, pomocí tlačítka „All“ nebo zpět na hledání pomocí tlačítka „Search by name“. Zobrazený rozsah sloupců se odlišuje pro oba módy seznamů. Odlišnost je dána návratovým objektem ze serveru RTS2, který se liší pro oba požadavky. Tabulky se opět generuje automaticky, aby kvůli změně na serveru, nebyl problém se zobrazením. Při psaní do textového pole se automaticky obnovují data z observatoře.

Actions	Target ID	Target Name	RA	DEC	Description	Duration	Scripts	Satisfied
<a href="#">New target</a> <a href="#">All</a> <a href="#">Search by name</a>	590							
	568	NGC5904	229.62272702058	2.09652023315		143.99000072479248	<pre> [["C0":["filter=N for 5 { E 10 } E 5 filter=I E 5 filter=R E 3 filter=V E 5 filter=B E 10 E 10 filter=V E 5 filter=R E 3 filter=I E 5",127,14], ["C1":["for 5 { E 10 } E 60",143.99,6]],["C2":["E 20",20,1]]] </pre>	["tunarDistan
	1504	NGC 5907	228.973375	56.3286667		259.5	<pre> [["C0":["filter=N for 4 { E 20 } filter=I E 20 E 20 filter=B E 20 E 20 filter=V E 20 E 20 filter=R E 20 E 20 }",259.5,12]],["C1":["for 6 { E 10 20 }",93.99,6]],["C2":["E 20",20,1]]] </pre>	["airmass", "l

Obrázek 4.13: Rozhraní RTS2 UI - seznam pozorovacích cílů.

### 4.3.2 Editace a vytvoření cíle

Každý řádek má možnost editace a zobrazení obrázků pozorovacího cíle. Dále je dostupné povolení a zakázání cíle pro autonomní výběr. Tato hodnota je pak vidět v módu zobrazení všech pozorovacích cílů, takže pokud chce uživatel zjistit aktuálně nastavenou hodnotu, musí si zobrazit všechny pozorovací cíle. Metoda pro povolení a zakázání vrací úspěch nebo neúspěch, čili je možno uživateli dát alespoň vědět, jestli se akce provedla pomocí dialogu. Využívá se metody „/api/udate\_target“, která je vidět v tabulce 4.10.

Pro vytvoření nového pozorovacího cíle je v levém horním rohu umístěno tlačítko „New target“, které otevře formulář pro přidání. Obdobný formulář otevře tlačítko pro úpravu, a ten je i automaticky generovaný. Formulář pro úpravu je vidět na obrázku 4.14.

The screenshot shows the 'Target edit' interface in the RTS2 Web application. The header includes navigation links for Telescopes, Public telescopes, and Users, along with a user profile 'radek'. The main content area is titled 'Target edit' and contains a form with the following fields and values:

- Target ID:** 1
- Target Name:** Dark frames
- RA:** (empty)
- DEC:** (empty)
- Description:** (empty)
- Duration:** 60.5
- Scripts:** {"C0":["filter=DF binning=0 D 1 D 2 D 4 D 5 D 10 D 20 binning=1 D 10",60.5,7]}{"C1":["D 1 D 2 D 10 D 20",55.66,4]} {"C2":["W0.filter=DF D 1 D 2 D 4 D 5 D 10 D 20",42,6]}
- Satisfied:** (checkbox)
- Violated:** (checkbox)
- Transit:** (checkbox)
- Observable:** true

At the bottom left of the form is a blue button labeled 'Update target'.

Obrázek 4.14: Rozhraní RTS2 UI - editace pozorovacího cíle.

Pro získání dat a generaci formuláře na obrázku 4.14 je použita metoda „/api/tbyid“. Podrobnější informace jsou vidět v tabulce 4.8. Pro vytvoření nového pozorovacího cíle je použita metoda „/api/create\_target“ z tabulky 4.9.

Cesta	/api/tbyid	Získání vlastností konkrétního pozorovacího cíle.
Parametr	<b>id</b>	Specifikuje identifikátor pozorovacího cíle.
Parametr	e	Specifikuje, že metoda vrací podrobný výpis.
Výsledek	JSON objekt.	Struktura obsahující hlavičku tabulky a data s jedním záznamem.

Tabulka 4.8: Metoda pro získání informací o konkrétním pozorovacím cíli.

Mezi metodami pro správu dat je určitý logický problém. Některé metody umožňují změnu dat, které jiné neumí vyčíst. Například editace „/api/update\_target“ umožňuje modifikace parametru „pm\_ra“, ale již jsem neobjevil metodu, která by mi tento parametr umožnila vyčíst. Raději jsem takovéto parametry zakázal pro editaci.

## 4.4 Plánování

Plánování pozorování je v systému RTS2 realizováno pomocí front. Zařízení, které je pojmenováno SEL (*selector*), je servis běžící na pozadí serveru observatoře. Zajišťuje

Cesta	/api/create_target	Vytvoření nového pozorovacího cíle.
Parametr	<b>tn</b>	Specifikuje název.
Parametr	<b>ra</b>	Specifikuje RA souřadnici.
Parametr	<b>dec</b>	Specifikuje DEC souřadnici.
Parametr	<b>type</b>	Specifikuje typ.
Parametr	<b>info</b>	Specifikuje další informaci.
Výsledek	Id nového objektu.	Id nebo chybu.

Tabulka 4.9: Metoda pro vytvoření pozorovacího cíle.

Cesta	/api/update_target	Editace pozorovacího cíle.
Parametr	<b>id</b>	Specifikuje identifikátor.
Parametr	<b>tn</b>	Specifikuje název.
Parametr	<b>ra</b>	Specifikuje RA souřadnici.
Parametr	<b>dec</b>	Specifikuje DEC souřadnici.
Výsledek	Id nového objektu.	Id nebo chybu.

Tabulka 4.10: Metoda pro úpravu pozorovacího cíle.

správu front a výběr dalšího cíle, který předá druhému zařízení, a to je EXEC (*executor*). Zařízení EXEC je servis běžící na pozadí a stará se o vykonání pozorování.

Cesta	/api/cmd	Vykonání příkazu nad zařízením.
Parametr	<b>d</b>	Specifikuje zařízení.
Parametr	<b>c</b>	Specifikuje příkaz.
Výsledek	JSON Objekt.	Informaci o zařízením.

Tabulka 4.11: Metoda pro vykonání příkazu nad zařízením.

#### 4.4.1 Přidání cíle do fronty

Pro plánování pozorování nejsou připraveny přímé HTTP metody. Musí se využít metoda „/api/cmd“, která je blíže popsána v tabulce 4.11. Metoda je určena pro obecné vykonávání příkazů nad zařízením.

Jednotlivé příkazy, které lze využít, jsou dostupné v klasické terminálu v manuálových stránkách. Například příkazem *man rts2-selector* lze získat nápovědu k zařízením SEL.

Příkaz, který jsem využil pro přidávání cíle do seznamu fronty, je *queue\_at\_nrep*. Ten má následující tvar vstupních parametrů:

```
1 queue_at_nrep navezFronty identifikatorCile start konec opakovani
   pauzy
```

Kód 4.1: Příkaz pro přidání pozorovacího cíle do fronty.

Prvním parametrem je název fronty. Již vytvořené fronty se dají získat v detailu zařízení SEL. Musí se využít metody „/api/get“ z tabulky 4.2. Metoda vrátí objekt, ve kterém je nutné vyhledat proměnnou s názvem „queue\_names“. Obsah proměnné je pole textových hodnot, každá je názvem jedné fronty.

Druhým parametrem je identifikátor cíle pozorování. Ten lze zvolit libovolně z existujících cílů v databázi.

Třetím parametrem je začátek pozorování. Pozorování nemusí být naplánováno přímo v čase začátku, ale nejdříve v čase začátku. Parametr je číslo, které specifikuje počet vteřin od data 1.1.1970.

Čtvrtým parametrem je konec pozorování. Parametr znamená, že po tomto čase, již není o pozorování zájem a cíl se z fronty odebere. Je specifikovaný stejně jako parametr začátku pozorování, tj. počtem vteřin od 1.1.1970.

Pátým parametrem je počet opakování pozorování. Určuje maximální počet opakování a po dosažení je cíl odebrán z fronty. Pokud je nastaven na -1, pozoruje se nekonečně dlouho.

Šestáým parametrem je minimální pauza mezi pozorováním cíle. Může být 0. Specifikuje se ve vteřinách.

Jsou dostupné i jiné příkazy, ale vybral jsem pro realizaci tento, protože umí vše. Ostatní jsou pouze zjednodušené verze, například pokud nechcete specifikovat počet opakování. I tento příkaz umožňuje vynechání specifikace nějakého parametru, respektive nahrazením za „nan“ se parametr vynechá.

V rozhraní RTS2 UI byl připraven pro přidání pozorovacího cíle do fronty formulář z obrázku 4.15. Je zde vidět možnost zvolení fronty, vyhledání cíle a možnost specifikovat ostatní parametry. Pokud se ostatní parametry nespecifikují, budou použity hodnoty „nan“. K vyhledávání cíle je použita metoda „/api/tbyname“ z tabulky 4.7.

Queue	Target	Start	End	Rep	Separation
manual	Mars	17.5.2017	18.5.2017	10	10

Add

Obrázek 4.15: Rozhraní RTS2 UI - formulář pro přidání cíle do fronty.

#### 4.4.2 Zobrazení fronty

Vyčtení seznamu front je možné provést HTTP metodou „/api/get“ z tabulky 4.2. Fronty obsahuje zařízení s názvem SEL. Front může být v jednom RTS2 systému několik. Vyčtení názvu front bylo již popsáno v sekci 4.4.1. Názvy front se pak použijí k vyčtení konkrétních zařazených cílů. Například seznam identifikací cílů pozorování fronty s názvem „manual“ je dostupný v proměnné s názvem „manual\_ids“. Pomocí názvu fronty mohou vyčíst i ostatní údaje jako začátek, konec, počet opakování

a pauzy mezi pozorováním. Neobsahuje však žádné další validní informace o cílech, jako například název. Ten je pak získán asynchronním voláním HTTP metody „/api/tbyid“, které je popsána v tabulce 4.8.

Ze získaných dat, které se postupně načítají, je vytvořena tabulka. Tabulka je vidět na obrázku 4.16. Obsahuje jednoduchý přehled zmíněných hodnot.

Queue	Traget	Q ID	Start	End	Rep	Separation	Actions
manual							▲
plan							▼
	Dark frames	3			-1	-1	🗑️
	Venus	10	2017-05-31 15:47:23 +0200	2017-06-02 15:47:27 +0200	-1		🗑️
	Sun	11			-1		🗑️
simul							▼

Obrázek 4.16: Rozhraní RTS2 UI - tabulka pro zobrazení front.

### 4.4.3 Zobrazení plánu

Plán pozorování je v podobě seznamu identifikací cílů dostupný v zařízení EXEC. Konkrétně v proměnné „next\_executed\_ids“. Použitím metody „/api/get“ se načte seznam identifikací cílů a poté se postupně načtou jednotlivé informace o cíli pomocí metody „/api/tbyid“.

Obrázek 4.17 ukazuje tabulku s aktuálním seznamem pro pozorování.

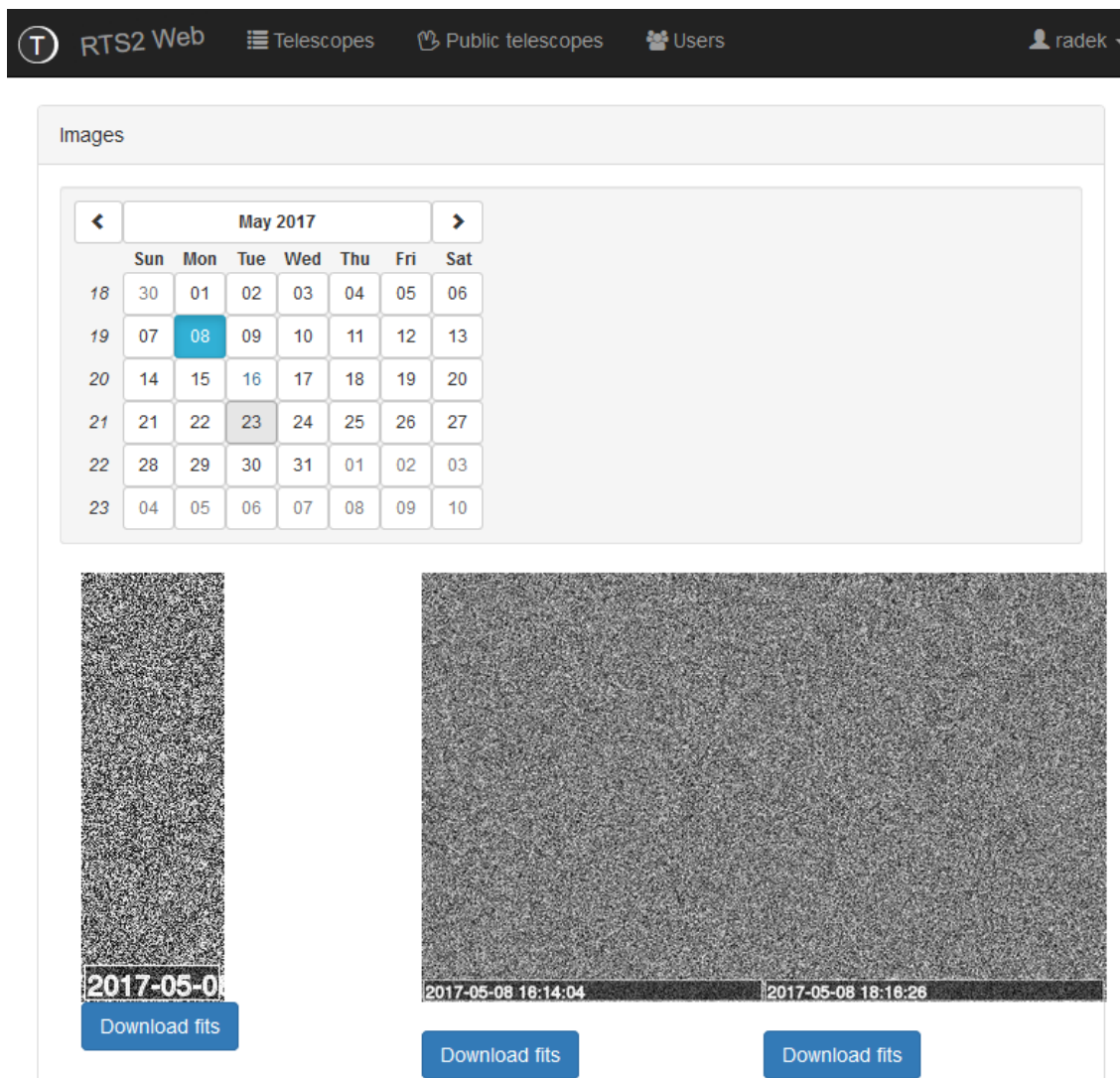
Target	Time	Actions
OJ 287	2017-05-09 22:33:14 +0200	
V* V426 Oph - longscript	2017-05-10 03:06:24 +0200	
3C 454.3	2017-05-10 04:08:51 +0200	
Dark frames	2017-05-10 04:16:56 +0200	
Dark frames	2017-05-10 20:31:55 +0200	
Dark frames	2017-05-10 21:36:44 +0200	
Dark frames	2017-05-11 21:40:12 +0200	

Obrázek 4.17: Rozhraní RTS2 UI - tabulka pro zobrazení plánu.

## 4.5 Přístup k obrázkům

### 4.5.1 Podle data pozorování

Projekt RTS2 UI umožňuje dvojí možnost zobrazení napozorovaných dat. První možností je výběr snímků z daného dne. Na tu se uživatel dostane pomocí ikony hvězdy ve výpisu všech teleskopů z obrázku 4.1. Stránka ukazující obrázky je pak



Obrázek 4.18: Rozhraní RTS2 UI - zobrazení obrázků podle z určitého dne.

vidět obrázku 4.18, kde je opět zobrazen náhodně vygenerovaný šum. Pod obrázkem je zpřístupněna možnost stažení formátu FITS souboru.

Tato možnost zobrazení obrázků, využívá cesty na RTS2 serveru „/preview/“. Při použití tohoto požadavku na RTS2 server se zobrazí vygenerovaná webová stránka, která ukáže obrázky. Webovou stránku přečtu a uložím si cesty k obrázkům do seznamu. Ten se pošle klientovi. Následně klient žádá o konkrétní obrázky pomocí metody „/jpeg/“ nebo „/fits/“ a známé cesty ze seznamu.

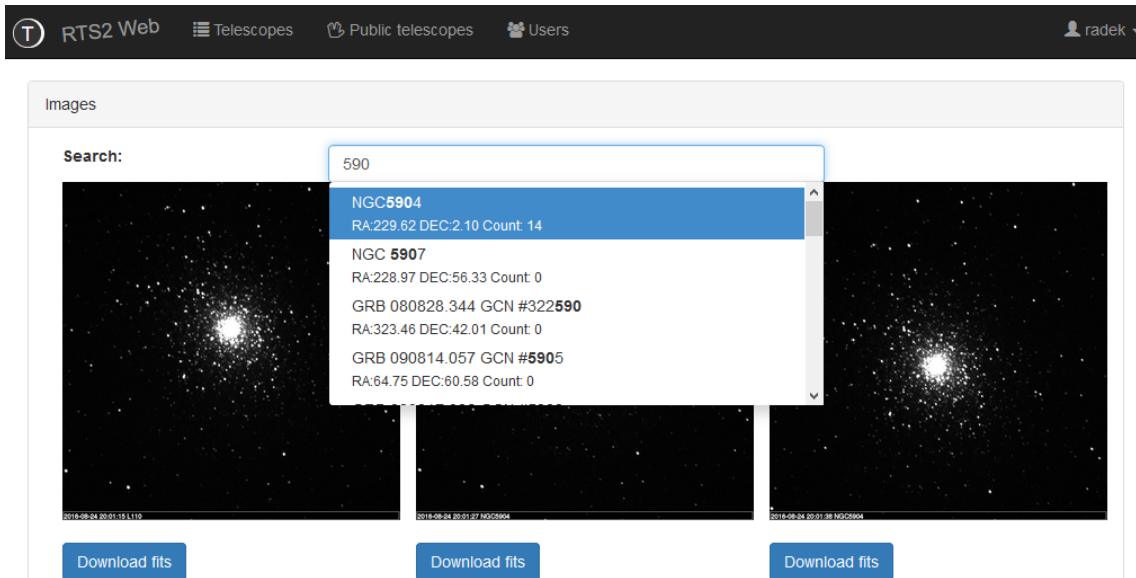
Při každém požadavku na data snímku, ať už fits nebo jpg formátu, je nejprve zkontrolován záznam v databázi RTS2 UI a až poté se žádá samotná observatoř. Při načtení dat snímku z observatoře se zároveň provede uložení do lokální databáze RTS2 UI.

#### 4.5.2 Podle cíle pozorování

Druhým typem zobrazení napozorovaných dat je podle pozorovacího cíle. K této možnosti se uživatel dostane pomocí ikonky obrázku, která se nachází u každého



pozorovacího cíle v seznamu na obrázku 4.13.



Obrázek 4.19: Rozhraní RTS2 UI - zobrazení snímků podle pozorovacího cíle.

Obrázek 4.19 ukazuje možnost výběru snímků podle pozorovacího cíle. Uživateli je umožněno zvolit cíl pomocí vyhledávání. Při výběru cíle se získá identifikátor, který se následně použije v cestě „/targets/id/images/“. Požadavek vrací HTML stránku, kterou je opět nutné zanalyzovat a vytvořit seznam obrázků. Potom je funkce stejná jako u zobrazení podle zvoleného data.

## 4.6 Správa uživatelů a oprávnění

Projekt RTS2 UI má svou vlastní správu uživatelů, která není spojena se samotným systémem RTS2. Uživatelé jsou ověřeni mezi klientskou a serverovou částí. Mezi serverem a observatoří se používá jiná identifikace, který nemá stejné údaje jako uživatel požadující data.

ID	First Name	Last Name	Email	Role	User Name	Active	Actions
36	scientist	scientist	s@s.cc	Public	scientist	true	<a href="#">✎</a>
33	Radek	Meciar	r.meciar@gmail.com	Admin	radek	true	<a href="#">✎</a>
37	nevím	vedec	kkk@kkk.cc	Scientist	nevím	false	<a href="#">✎</a>
35	public	public	nevím@nevím.cz	Public	public	true	<a href="#">✎</a>

Obrázek 4.20: Rozhraní RTS2 UI - seznam uživatelů.

Seznam uživatelů a jejich správa je přístupná pouze pro uživatele s oprávněním administrátor. Seznam je dostupný z menu a je zobrazen na obrázku 4.20. Nad

seznamem se nachází tlačítko pro přidání nového uživatele nebo je možnost editovat stávajícího uživatele zvolením ikonky pro editaci u konkrétního řádku tabulky.

Formulář pro editaci uživatele je vidět na obrázku 4.21. Umožňuje změnu uživatelských vlastností. Nastavitelné role jsou tři a to „Admin“, „Public“ a „Scientist“. Je zde možnost uživatele zakázat. Zakázaný uživatel nebude smazán, ale bude mu pouze při pokusu o přihlášení odmítnut přístup a ten může být časem znovu povolen.

Obrázek 4.21: Rozhraní RTS2 UI - formulář uživatele.

Pro uživatele s oprávněním „Scientist“ se pod editačním formulářem zobrazí ještě panel pro zvolení, na které teleskopy má přístup. Panel je vidět na obrázku 4.22. Uložení panelu je nezávislé na uložení vlastností uživatele.

Obrázek 4.22: Rozhraní RTS2 UI - přiřazení oprávnění k teleskopu.

## 5. Zhodnocení a závěr

Práce měla za úkol vytvořit uživatelské rozhraní pro síť teleskopů, které používají systém RTS2. Úkol byl splněn v každém bodě zadání. Software umožňuje připojení k síti teleskopů. Je dostupná správa uživatelů s oprávněními, které jim umožňují konkrétní přístup do aplikace. Byla zpřístupněna možnost plánovat pozorování v síti a přístup k napozorovaným datům. Napozorovaná data se ukládají v lokální databázi serverové části aplikace, aby bylo odlehčeno síťovému provozu observatoře. A pro budoucí použití je implementována funkce WebSocket serveru.

Celé softwarové řešení bylo vyvinuto jedním člověkem za poměrně krátkou dobu. Vývoj komplikovalo několik špatných informací, hlavně ohledně plánování. Systém RTS2 již nevyužívá službu zvanou BigBrother, která zajišťovala možnost plánovat mezi více teleskopy. Při vývoji práce nebyla informace o změně plánování dostupná, pravděpodobně vlivem staré verze dokumentace. Pokusy o implementaci s napojením ke službě BigBrother byly neúspěšné a zabraly mnoho času. Obdobně zabraly čas chyby samotného systému, jako například vytvoření cíle pozorování s typem „E“. Typ „E“ by měl značit objekty, které obíhají blízko Země. Ale po vložení do systému se ovlivní funkce ostatních metod, tj. nelze vyčíst seznam všech cílů pozorování, a nefunguje vyhledávání cílů až do ručního odstranění špatného záznamu z databáze. Řešení těchto a dalších problémů stálo mnoho času při vývoji. Systém RTS2 je velice rozsáhlý software vyvíjený nekomerčně s otevřeným kódem, také jedním člověkem. Je pochopitelné, že může mít nějaké chyby. A předpokládám, že chyby se objeví i v softwaru RTS2 UI.

Funkce softwaru byla testována na virtuální observatoři, která neměla fyzicky připojený teleskop. V pozdější fázi vývoje byla funkce rovněž otestována na reálné observatoři, ale s nepoužíváním funkcí určených k zápisu dat. Z reálné observatoře se pouze stahovala data, aby se neovlivnil její chod. Funkce softwaru se jevila správná.

# Literatura

## Knihy a publikace

- [1] KUBÁNEK, Petr. *RTS2 JSON API documentation*. 113 stran. 2013.
- [2] KUBÁNEK, Petr. *RTS2: The Remote Telescope System*. Advances in Astronomy, vol. 2010, Article ID 902484, 9 pages, 2010. doi:10.1155/2010/902484
- [3] KUBÁNEK, Petr, et al. *RTS2: meta-queues scheduling and its realisation for FLWO 1.2 m telescope*. SPIE Astronomical Telescopes+ Instrumentation. International Society for Optics and Photonics, 2012.
- [4] Perez-del-Pulgar, Carlos Jesus, et al. *GLORIA: The First Free Access e-Infrastructure of Robotic Telescopes for Citizen Science*. Advances in Information Systems and Technologies. Springer Berlin Heidelberg, 2013. 293-304.
- [5] ŠÁRA, Jan. *Mobilní aplikace pro řízení v reálném čase*. 2014. České vysoké učení technické v Praze. Vypočetní a informační centrum.

## Internetové zdroje

- [6] *RTS2 | Remote Telescope System 2* [online]. [cit. 2017-05-05].  
Dostupné z: <http://www.rts2.org/>
- [7] *Objektově relační mapování* [online]. [cit. 2017-05-05].  
Dostupné z: [https://cs.wikipedia.org/wiki/Objektov%C4%9B\\_rela%C4%8Dn%C3%AD\\_mapov%C3%A1n%C3%AD](https://cs.wikipedia.org/wiki/Objektov%C4%9B_rela%C4%8Dn%C3%AD_mapov%C3%A1n%C3%AD)
- [8] *The WebSocket Protocol - RFC 6455* [online]. [cit. 2017-05-05].  
Dostupné z: <https://datatracker.ietf.org/doc/rfc6455/>
- [9] *FITS Standard Document* [online]. [cit. 2017-05-05].  
Dostupné z: [https://fits.gsfc.nasa.gov/fits\\_standard.html](https://fits.gsfc.nasa.gov/fits_standard.html)

## Knihovny a zdrojové kódy

- [10] *RTS2 - Remote Telescope System, 2nd Version* [online]. [cit. 2017-05-01].  
Dostupné z: <https://github.com/RTS2/rts2>
- [11] *RTS2 UI* [online]. [cit. 2017-05-01].  
Dostupné z: [https://github.com/Ramese/rts2\\_ui.git](https://github.com/Ramese/rts2_ui.git)

- [12] *POCO C++ Libraries* [online]. [cit. 2017-05-01].  
Dostupné z: <https://pocoproject.org>
- [13] *AngularJS* [online]. [cit. 2017-05-01].  
Dostupné z <https://angularjs.org/>
- [14] *AngularJS* [online]. [cit. 2017-05-01].  
Dostupné z <https://docs.angularjs.org/api/ngRoute>
- [15] *UI Bootstrap* [online]. [cit. 2017-05-01].  
Dostupné z <https://angular-ui.github.io/bootstrap/>
- [16] *Apache Maven Project* [online]. [cit. 2017-05-01].  
Dostupné z <https://maven.apache.org/>
- [17] *Bower - A package manager for the web* [online]. [cit. 2017-05-01].  
Dostupné z <https://bower.io/>
- [18] *JSHint* [online]. [cit. 2017-05-01].  
Dostupné z <http://jshint.com/>
- [19] *HttpClient Overview* [online]. [cit. 2017-05-01].  
Dostupné z <https://hc.apache.org/httpcomponents-client-ga/index.html>
- [20] *Hibernate* [online]. [cit. 2017-05-01].  
Dostupné z: <http://hibernate.org/orm/>

## A. CD

Popis hlavních přiložených souborů:

<b>Database</b>	.....	Obsahuje skripty týkající se databáze
· createDB.sql	.....	Skript pro vytvoření databáze
<b>Documentation</b>	.....	Dokumenty týkající se práce
<b>text prace</b>	.....	Text práce
· rts2ui.pdf	.....	Text práce
· jsondoc.pdf	.....	Dokumentace JSON API RTS2
<b>Project</b>	.....	Projektová složka pro IDE Netbeans
<b>src</b>	.....	Zdrojový kód serverové části v jazyce Java
<b>Web</b>	.....	Zdrojový kód klientské části v jazyce JS
· pom.xml	.....	Nastavení závislostí Maven

## B. Seznam použitých zkratek

Zkratka	Význam
FITS	Flexible Image Transport System
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transport Protocol
JDK	Java Development Kit
JPEG	Joint Photographic Experts Group
JS	JavaScript
RGB	Red Green Blue
RTS2	Remote Telescope System 2nd Version
SPA	Single Page Application
UI	User Interface

## C. Návod na použití projektu

### Získání zdrojových kódů

Zdrojový kód je dostupný na přiloženém CD. Lepší varianta je získání kódu ze služby GitHub, kde by měla být aktuální verze:

```
1 git clone https://github.com/Ramese/rts2_ui.git
```

### Vytvoření databáze

Použitá databáze je PostgreSQL. Skript pro vytvoření prázdné databáze je dostupný ve složce *Database*.

### Připojení k databázi

Konfigurace připojení k databázi se nachází v souboru *Config.java* v balíku *rts2ui*. Je zde nutné změnit cestu k databázovému serveru, uživatelské jméno a heslo.

### Konfigurace

Konfigurace serveru je popsána v části 3.2.1. Při ponechání přednastavených hodnot by měl projekt fungovat.

### Spuštění aplikace

Projekt je koncipovaný tak, aby nebylo nutné instalovat jiné aplikace, knihovny nebo aplikační servery. Lze jej přímo spustit.

### Nový uživatel

Nový uživatel se dá zaregistrovat v menu nebo vytvořit přímo v databázi.