Masarykova univerzita
Fakulta informatiky

Webová aplikace pro astronomické pozorovací deníky

Bakalářská práce

Ondřej Skýba

Brno, 2016

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Ondřej Skýba

Vedoucí práce: RNDr. Martin Kuba, Ph.D.

Poděkování

Rád bych poděkoval doc. RNDr. Miloslavu Zejdovi, Ph.D. za jeho čas a ochotu odpovídat na všechny mé dotazy z oblasti astrofyziky. Dále děkuji RNDr. Martinu Kubovi, Ph.D. za cenné rady ohledně psaní bakalářské práce.

Shrnutí

Cílem této bakalářské práce je pro Ústav teoretické fyziky a astrofyziky Přírodovědecké fakulty Masarykovy univerzity analyzovat a reimplementovat webovou aplikaci pro vedení pozorovacích deníků. Práce stručně seznámí s Nette Frameworkem, popíše novou iterativně vyvíjenou aplikaci. Podrobněji jsou rozebrány některé vybrané problémy a části aplikace. Nakonec také navrhuje možná řešení do budoucnosti.

Klíčová slova

Webová aplikace, astronomické pozorovací deníky, PHP, Nette

Obsah

Úvod

Hlavním cílem této práce je reimplementovat webovou aplikaci pro astronomické deníky. Astronomické deníky jsou záznamy o pozorování proměnných hvězd, které obsahují hodnoty hvězdné velikosti hvězdy v čase, a k nim další důležité informace jako použitý filtr, formát juliánského datování atp.

Jedno pozorování odpovídá měření hvězdné velikosti jedné proměnné hvězdy během jedné noci. Naměřené hodnoty jsou spolu s příslušnými časy měření uloženy v datovém souboru. Z více pozorování se pak může k jedné periodicky proměnné hvězdě vykreslit tzv. fázová křivka, díky které je možné zobrazit kompletní křivku z několika částečných pozorování. Aby se fázová křivka mohla vykreslit, je potřeba znát periodu světelných změn hvězdy P a okamžik extrému jasnosti hvězdy M_0 zadaný v Juliánském datu.

Nová aplikace byla vyvíjena iterativním způsobem vývoje a pravidelně konzultována se zadavatelem. Přestože v průběhu vývoje došlo ke značným změnám ve fungování aplikace, v této práci budu převážně popisovat aktuální podobu, pouze místy upozorním na předchozí implementaci.

První kapitola této práce se věnuje analýze původního systému, jaké jsou funkce aplikace a možnosti uživatele. Některé funkce jsou podrobněji popsány. Dále popisuje použité technologie a upozorňuje na značné nedostatky a chyby v aplikaci.

Druhá kapitola seznamuje s požadavky, které by měla splňovat nová aplikace. Požadavky jsou rozděleny na funkční a nefunkční.

Ve třetí kapitole se zabývám návrhem aplikace. Jsou zde popsány použité technologie, návrh databáze a tříd.

Čtvrtá kapitola již popisuje hotovou aplikaci, její jednotlivé části a přednosti. Dále se zaměřuje na některé konkrétní problémy či implementace, které podrobně rozebírá.

1 Analýza původní aplikace

Stávající webová aplikace s názvem MECA – Measurements of Eclipsing Binaries – Archive (dále jen „MECA“) byla vytvořena Bc. Lubošem Brátem v roce 2013 a slouží pro archivování fotometrických měření zákrytových dvojhvězd.

Aplikace je online[1], ale přístupná pouze pro přihlášené uživatele. Registrace pro nově příchozí není nabízena, o přidání nového uživatele do systému se musí postarat jeden z administrátorů. Administrátoři byli ručně nastaveni v databázi při vzniku aplikace.

1.1 Funkce aplikace

Hlavní funkcí aplikace je ukládání tzv. pozorování, vykreslování grafu a fázové křivky na základě uložených dat a zobrazování veřejných pozorovacích záznamů v seznamu, který je možné řadit dle sloupců. Editovat a mazat tyto záznamy je možné pouze, je-li uživatel jejich vlastníkem. Dále by mělo být možné přidat nebo upravit svůj pozorovací profil[2], ale tato funkce je v aktuální verzi nedostupná.

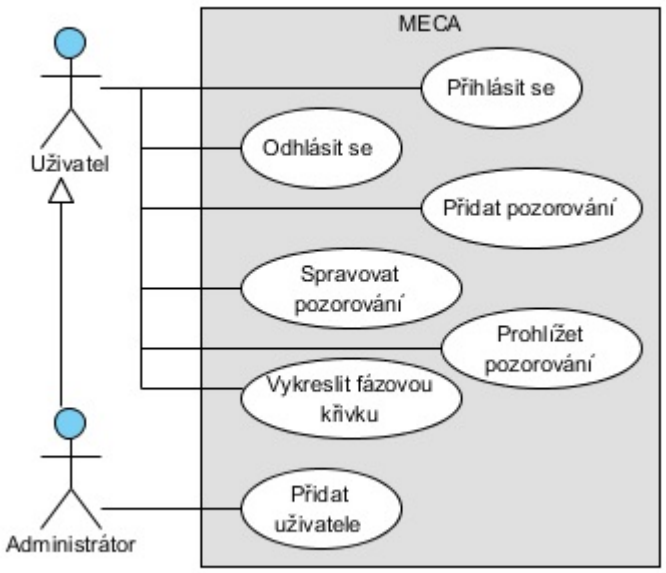
1.1.1 Ukládání pozorování

K této funkci lze přistoupit dvěma cestami. Buď je možné vložit pozorování bez předlohy, nebo pozorování s předvyplněnými údaji na základě již existujícího pozorování. Samotné ukládání se pak skládá ze tří částí:

- Vyplnění povinných údajů pozorování, výběr zdrojového souboru a nepovinně souboru s tzv. mapkou[3] a nastavení viditelnosti záznamu. Vybrání objektu, ke kterému se pozorování uloží, probíhá formou pouhého vepsání názvu objektu do textového pole.

---

1. V době psaní práce dostupná na adrese `<http://xmeca.physics.muni.cz/>`
2. Udává zdroj měření a použitý přístroj.
3. Na mapce jsou vyznačeny kontrolní hvězdy, vůči kterým se počítá jasnost proměnné hvězdy.



Obrázek 1.1: Diagram případů užití původní aplikace

- Kontrola údajů ve formuláři a náhled nahraných souborů. V případě chyby je nutné chybné položky opravit.
- Odeslání formuláře a uložení záznamu do databáze.

1.1.2 Nastavení viditelnosti pozorování

Aplikace řeší viditelnost záznamů velice jednoduše. Vložené pozorování může být buď'to viditelné pouze pro vlastníka, nebo viditelné pro všechny uživatele. Dále je možné specifikovat, které části záznamu zobrazit a které schovat. Na výběr jsou tyto možnosti:

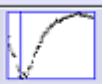
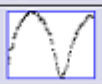
- Zobrazovat mapku – ostatní uživatelé si mohou zobrazit pozorování a mapku (pokud byla nahrána).
- Zobrazovat mapku a křivku – pozorování je veřejně přístupné, ale není uveden odkaz na zdrojový soubor, ze kterého je generována křivka.

- Zobrazovat mapku, křivku a data – kompletně přístupné pozorování.

1.1.3 Procházení dostupných pozorování

Procházet pozorování je možné pouze z hlavní strany, kde je potřeba kliknout na jméno uživatele, nebo na odkaz, který zobrazí všechna dostupná pozorování. Následně se zobrazí tabulka s výsledky, které je možné řadit dle sloupců vzestupně nebo sestupně. Tyto výsledky jsou stránkovány po 30 položkách. Nad tabulkou je také možnost vyfiltrovat pouze pozorování k určitému objektu a tato pozorování následně skládat do fázové křivky (viz podkapitola 1.1.4).

Na obrázku 1.2 je ukázka tabulky s výsledky. Z pozorování jsou v této tabulce zobrazeny pouze nejdůležitější údaje, zbytek údajů a vykreslené grafy jsou vidět v detailu pozorování. Na detail pozorování se ale dá dostat pouze přes kliknutí na odkaz v sloupci *hvězda*, což je velmi neintuitivní.

#	hvězda ^ v projekt	datum ^ v	filtr ^ v	křivka	mapka data	stanice pozorovatel ^ v	Přístroj poznámka	vložena ^ v
768	V784 Aql misca	3.7.2015 20:10 - 4.7.2015 1:34 UT	R		mapka 73	MUO Brno Miloslav Zejda	RL600 info	2015-07-04 add new
767	V373 Dra misca	29.6.2015 19:57 - 30.6.2015 1:39 UT	V		mapka 76	MUO Brno Miloslav Zejda	RL600 info	2015-07-02 add new

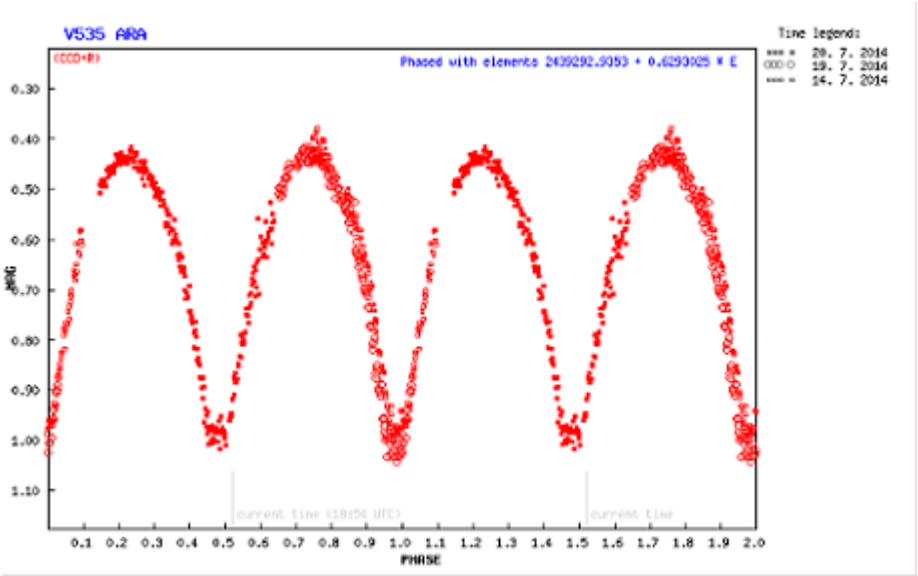
Obrázek 1.2: Ukázka výpisu pozorování v tabulce.

1.1.4 Fázová křivka

Fázová křivka je velice důležitá součást aplikace. Umožňuje vybrat vícero pozorování od jedné periodicky proměnné hvězdy a poskládat z těchto jednotlivých pozorování celistvou křivku vztaženou k jediné periodě vyjádřené v podobě fáze v intervalu (0;1). K výpočtu fáze je využita perioda P a zvolený okamžik extrému jasnosti, tzv. epocha M_0.

Obrázek 1.3 zobrazuje fázovou křivku ze tří pozorování pro objekt V535 Ara. Jednotlivá pozorování jsou na křivce odlišena symboly, které jsou vysvětleny v pravé části grafu.

Obecně vykreslování grafů a fázové křivky v aplikaci funguje, ale není bezchybné. Vstupy pro fázovou křivku nejsou ošetřeny pro neplatné hodnoty, v určitých případech elementy grafu značně přesahují jeho hranice a zasahují do popisků a popisky také nejsou nijak zarovnané.



Obrázek 1.3: Ukázka fázové křivky ze tří pozorování objektu V535 Ara

1.2 Použité technologie

MECA je provozována na serveru Apache a databázi MySQL. Samotná aplikace je psána ve skriptovacím jazyku PHP, nicméně není navržena objektově, přestože je tento způsob v PHP podporován od verze 5.0 (rok 2004)[2]. PHP skripty pak generují HTML stránku ve standardu XHTML 1.0.

Přihlášení uživatele je řešeno tzv. Basic Access Authentication[4] Nevýhodou této metody může být fakt, že přenos není chráněn proti odposlechu.[1]

4. Basic Access Authentication je jednoduchý typ autentizace, kdy server vyzve uživatele pomocí HTTP požadavku, aby zadal jméno a heslo[1].

Odhlášení se pak docílí zavřením prohlížeče, nebo voláním PHP funkcí `destroy_session()`, která odstraní data spojená s aktuální session (kromě globálních proměnných) [3], a `session_unset()`, která uvolní všechny proměnné v session.[4]

Generování obrázků křivek je provedeno pomocí funkcí, které jsou součástí jazyka PHP. Nicméně obrázky se nikam neukládají, tudíž se generují pokaždé, je-li zapotřebí je zobrazit. Tento přístup šetří úložné místo na serveru, ale zároveň server více zatěžuje při každém přístupu k uloženým záznamům.

Pro zvětšení zobrazení obrázku je použit javascriptový plugin LightBox verze 2.51[5], který obrázek zobrazí v překryvném průhledném okně (tzv. modální okno)

1.3 Uživatelské rozhraní

Stránky mají velmi jednoduchý design, ale zacházení s nimi je už méně uživatelsky příjemné.

Při vstupu na stránku je uživatel vyzván k přihlášení. Tato výzva se pak může zobrazit znovu v případě, že během práce na webu skončí platnost přihlášení.

Po odhlášení se zobrazí pouze jednoduchá stránka s textem, který potvrzuje odhlášení a odkazem, který uživateli opět nabídne přihlášení.

Po úspěšném přihlášení má uživatel možnost vložit nový záznam o měření, zobrazit veřejné záznamy ostatních uživatelů, vyhledat záznamy podle souřadnic, nebo objektu, který je nutné ručně zadat. Vzhledem k neomezování výpisu může být zobrazení uživatelů nepřehledné, pokud by se jich v databázi vyskytovalo větší množství.

Filtrování záznamů podle objektů je také velmi nepřehledné a uživatelsky nepříjemné vzhledem k množství souhvězdí a objektů v nich.

Dále je na hlavní straně možnost zobrazit si vlastní uživatelský profil. Tato funkce však není implementována, proto se po kliknutí uživatel setká pouze s HTTP chybou 404.

---

5. Autorem LightBoxu je Lokesh Dhakar. LightBox spadá pod licenci *Creative Commons Attribution 2.5 License[5]*

1.4 Jazyková lokalizace

Aplikace nenabízí žádnou viditelnou a uživatelsky příjemnou změnu jazyka. Výchozí jazyk je čeština, přesto jsou však některé výstupy a možnosti aplikace v angličtině.

Změnu jazyka do angličtiny resp. slovenštiny lze provést dopsáním parametru `lang` s hodnotou `en` resp. `sk` do adresy webu. Tato možnost přeloží pouze některé prvky stránky a při přechodu do jiné části aplikace parametr lang zmizí.

2 Požadavky na novou aplikaci

Po konzultaci se zadavatelem byly ustanoveny požadavky na novou aplikaci, které jsou podrobněji popsány v následujících podkapitolách. Tyto požadavky byly postupem času různě úpravovány popř. doplňovány pro lepší výsledek, nebo z důvodu realizace.

Zároveň bylo rozhodnuto, že nová aplikace ponese stejné jméno, respektive stejnou zkratku, jako původní aplikace – MECA

2.1 Nefunkční požadavky

Cílová aplikace bude spuštěna na serveru Apache s databází MySQL, aplikaci je tedy doporučeno psát stejně jako původní verzi ve skriptovacím jazyce PHP.

Aplikace má být spolehlivá (každý správně zadaný požadavek bude úspěšně vykonán), rychlá a především uživatelsky příjemná. Uživatelské rozhraní musí být přehledné, jednoduché a intuitivní.

Jazykem aplikace bude angličtina, aby ji mohli používat uživatelé nejen z České a Slovenské republiky.

2.2 Funkční požadavky

2.2.1 Přístup k aplikaci a práva

Pro nepřihlášenou veřejnost nemá být umožněn přístup k aplikaci. Každý nepřihlášený má být vyzván k přihlášení nebo registraci.

Uživatelé s přístupem do systému (tedy ti, kteří prošli registrací, nebo byli jiným způsobem přidáni do databáze) se budou rozdělovat na obyčejné uživatele a administrátory. Administrátoři budou mít všechna práva jako uživatelé a navíc budou mít umožněno spravovat uživatele i jakákoliv data.

2.2.2 Nové funkce

Aplikace musí obsahovat funkce z předchozí verze, tzn. ukládání a správa pozorování, vykreslení grafu a fázové křivky, výpis a filtrace

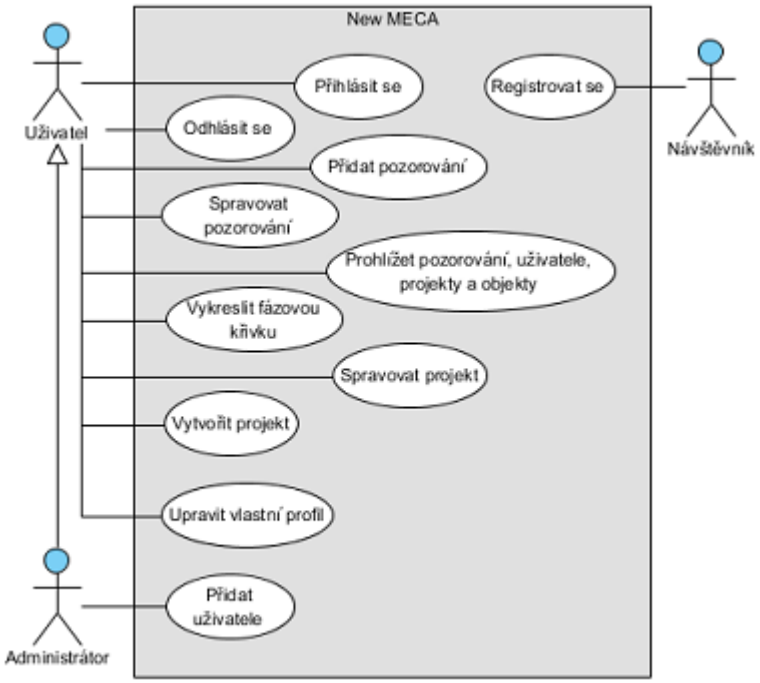
pozorování (podrobněji viz 1.1). Nová aplikace má navíc umožnit sdílení pozorování mezi uživateli a to formou projektů.

2.2.3 Možnosti uživatele

Každý uživatel bude mít možnost přidávat a spravovat nová pozorování, vytvářet a spravovat projekty. Do projektu nadále může přidat pozorování, nebo další uživatele (je-li tvůrcem projektu). Každý člen projektu má přístup k pozorováním uvnitř projektu, ale naopak kdokoliv jiný mimo projekt k nim přístup nemá.

Projekty a uživatelské profily jsou přístupné a viditelné všem, pouze pozorování mají omezenou přístupnost.

Nakonec je samozřejmostí, že si uživatel bude moci upravit svůj vlastní uživatelský profil.



Obrázek 2.1: Diagram případů užití nové aplikace

3 Návrh aplikace

Na základě požadavků jsem se rozhodl aplikaci vyvíjet v jazyce PHP a využívat databáze typu MySQL. Zároveň jsem zvolil verzi PHP 5.4, přestože jsou již v současné době dostupné verze novější, protože právě tato verze běží na produkčním serveru.

Jelikož by byl vývoj aplikace čistě v PHP velmi neefektivní, rozhodl jsem se využít Nette Frameworku, který popisuji v následující podkapitole 3.1.

3.1 Nette Framework

Nette Framework (dále jen „Nette“) je framework pro tvorbu webových aplikací v jazyce PHP, který velice usnadňuje vývoj, zajišťuje bezpečnost aplikace (např. eliminací bezpečnostních mezer ve formulářích) a podporuje kvalitní objektový návrh. V mé aplikaci budu využívat Nette verze 2.2.9, které je určeno pro PHP verze 5.3 a vyšší.

Nette je šířeno jako svobodný software a je licencováno pod New BSD nebo GNU General Public License [6].

3.1.1 Návrhový vzor MVP

V Nette je využíván návrhový vzor Model-View-Presenter (MVP), který vychází ze známějšího návrhového vzoru Model-View-Controller (MVC) [7].

Princip fungování MVP je následující: [8]

- Model – měl by zajišťovat přístup k datům a manipulaci s nimi. Zároveň by neměl vědět o tom, že nějaký presenter nebo view existuje.

- View – funkcí view je zobrazovat data získaná z modelu v podobě, kterou uvidí uživatel (např. stránka v HTML). View o modelu může a nemusí vědět.

- Presenter – vrstva, která propojuje view s modelem a provádí uživatelské akce (např. změna view, příkaz pro model, atd.).

3.1.2 Ladění chyb

Velkou předností Nette je jeho knihovna Tracy (původně Debugger, v českém překladu Laděnka). Tracy je nejen odchytává PHP chyby a výjimky a vypisovat je v přijatelné podobě, ale i chyby loguje a poskytuje diagnostické nástroje jako měření času načtení stránky (provedení požadavku v aplikaci), výpis dotazů do databáze a měření času těchto dotazů, informace o aktuálním uživateli.

Pokud aplikace běží ve vývojářském prostředí, je Tracy automaticky zapnuta, zobrazuje se diagnostická lišta na stránce a všechny ladící informace se při chybě rovnou vypíší na obrazovku. Pokud však aplikace běží v produkčním prostředí, pak je Tracy vypnuta. V případě chyby je uživatelům vypsána příslušná HTTP chyba (např. 404, 500) a detaily chyby jsou logovány do souborů pro vývojáře. [9]

3.1.3 Další přednosti Nette

Za zmínku stojí některé další přednosti Nette, kterými jsou: [10]

- Podpora AJAX[1], SEO[2].
- Router[3] pro tvoření hezkých URL.
- Česká komunita a dokumentace.
- Možnost stažení a instalace dalších doplňků od komunity.

3.2 Další použité technologie

3.2.1 jQuery v1.11.1

Knihovna jQuery je JavaScriptová knihovna, která poskytuje jednoduché API[4] pro práci s HTML dokumentem, událostmi, animacemi a AJAX [12]. jQuery je základem pro většinu JavaScriptových rozšíření.

---

1. Asynchronous JavaScript and XML
2. Search engine optimization
3. Obousměrný překládač mezi HTTP požadavkem / URL a akcí presenteru. [11]
4. Application Programming Interface

3.2.2 Bootstrap 3

Bootstrap je JavaScriptový a CSS[5] framework, který se používá pro snadné vytváření responzivních webových stránek vhodných i pro mobilní zařízení. Software je licencován pod MIT licencí [13].

Bootstrap také zajišťuje kompatibilitu skrze různé prohlížeče a poskytuje spoustu předdefinovaných komponent, např. ikony, rozbalovací menu, interaktivní upozornění a další.

3.2.3 Bootstrap datetime picker

Bootstrap datetime picker je JavaScriptové rozšíření založené na jQuery. Autorem tohoto rozšíření je Sebastien Malot, který vylepšil již existující kód [14].

Rozšíření slouží ke snadnému vkládání data a času v různých formátech s podporou několika jazyků.

3.3 Návrh databáze

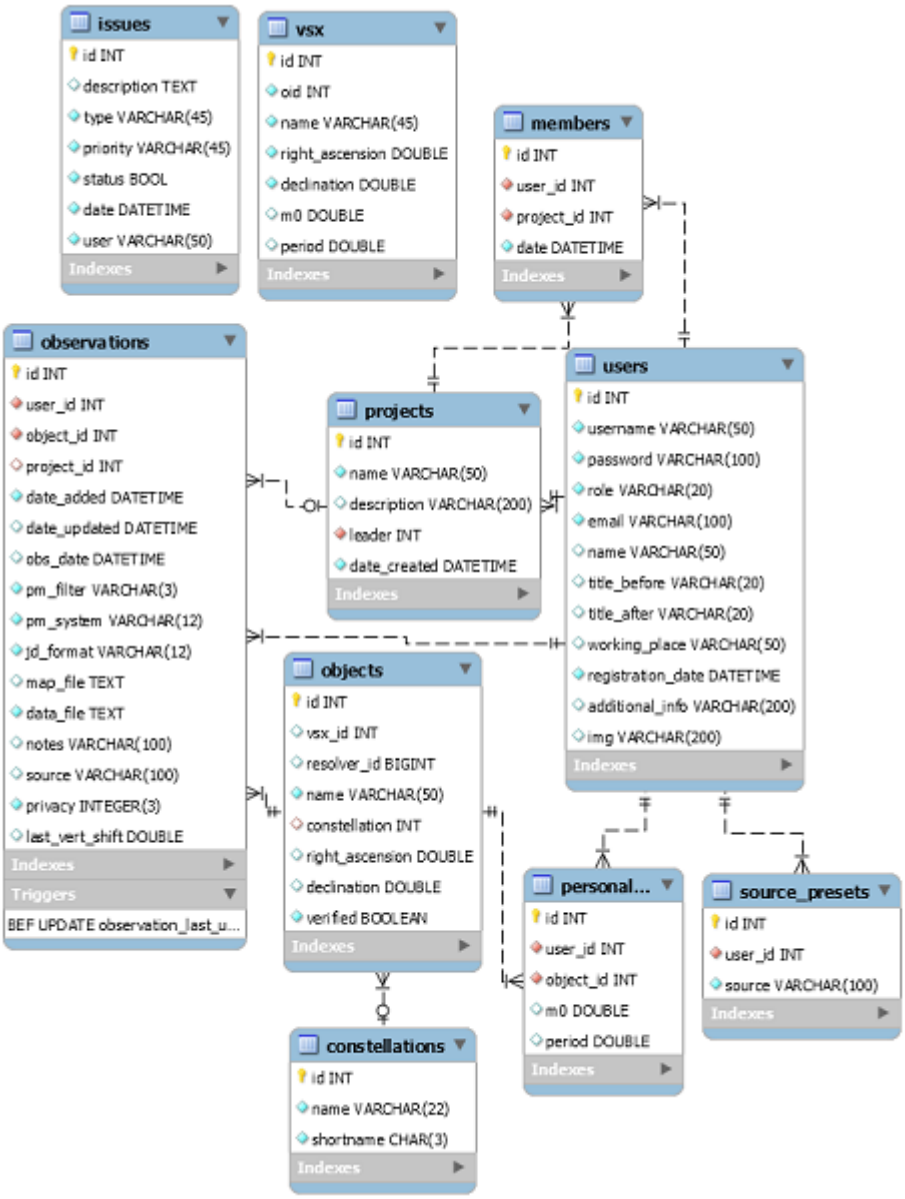
Správně navržená databáze je velmi důležitou součástí každého systému. Nemělo by docházet k duplikaci dat, ani by tabulky neměly obsahovat zbytečné sloupce. Na obrázku 3.1 je ERD[6] aktuální podoby databáze.

Návrh se v průběhu vývoje velmi měnil. Tabulky *issues*, *vsx*, *personal_elements* a *source_presets* se v původním návrhu nevyskytovaly a byly přidány až s přibývajícími požadavky od zadavatele. Tabulka *issues* je navíc tabulkou pouze dočasnou, která slouží pro zaznamenání hlášení chyb od uživatelů. *VSX* je kopií externí ASCII databáze. Proč tabulka existuje je podrobněji popsáno v kapitole 4.3.

Tabulka *users* představuje jednoho uživatele a informace o něm. Na *users* se váže tabulka *source_presets*, která pro uživatele ukládá jejich přednastavení zdrojů pro pozorování.

Projects představuje jeden projekt v systému, který má jméno, popis, vedoucího projektu jako odkaz na řádek v tabulce *users* a datum vytvoření projektu. Do projektu může vedoucí zařadit různé uživa-

5. Cascade Style Sheet - využívá se pro stylování HTML elementů
6. Entity relationship diagram



Obrázek 3.1: ERD schéma databáze.

tele jako členy, na to slouží tabulka *members*. Ta obsahuje sloupce s odkazem na tabulky *projects* a *users*, což představuje vztah „Uživatel ve sloupci *user_id* je součástí projektu *project_id*“.

V tabulce *objects* se nachází všechny používané astronomické objekty. Nový záznam se tedy přidá tehdy, je-li potřeba přidat nové pozorování k objektu, který zde ještě není uložen. Uloženy jsou název a souřadnice objektu, ID k externím databázím (pokud v nich objekt existuje, více k tomuto tématu v kapitole 4.3), příznak *verified*, který slouží k odlišení objektů přidaných z externí databáze a přidaných manuálně. Posledním sloupcem je souhvězdí, které nemusí být vyplněno a odkazuje na tabulku *constellations*. Tabulka *constellations* má předem vyplněné a neměnné hodnoty – obsahuje názvy všech 88 moderních souhvězdí a k nim Velký a Malý Magellanův oblak.

Ve vztahu s tabulkami *objects* a *users* je tabulka s názvem *personal_elements*, která umožňuje uživateli uložit s k danému objektu vlastní hodnoty světelných elementů M_0 a periody P.

Poslední nezmíněná tabulka je *observations*. Ta obsahuje záznam o pozorování, tzn. všechny informace zadané uživatelem při odesílání formuláře, a dále si pamatuje vlastníka, objekt, ke kterému patří, a popř. projekt, do kterého patří. K této tabulce je zhotoven trigger[7], který mění hodnotu ve sloupci *date_updated* na aktuální datum ve chvíli, kdy se řádek se záznamem pozmění.

3.4 Jmenné prostory a třídy

Jmenné prostory (angl. namespace) slouží k tomu, aby logicky sdružovaly třídy (nebo další jmenné prostory), které spolu souvisejí, a zároveň předcházely konfliktům ve jménech tříd. Třídy Nette jsou definovány pod jmennými prostory Nette*, kde * reprezentuje další jeden, nebo více jmenných prostorů.

Vlastní aplikace do Nette nijak nezasahuje a proto má i vlastní jmenný prostor App. Jmenné prostory aplikace jsem navrhl takto:

- App\Components – zahrnuje třídy představující znovupoužitelné komponenty
- App\Entities – zde jsou třídy, které reprezentují jednotlivé entity (např. Observation)

7. Činnost automaticky spuštěná v případě definované událost (např. při přidání nového řádku)

- App\Exceptions – jmenný prostor pro vlastní výjimky
- App\Model – obsahuje třídy představující model aplikace z návrhového vzoru MVP
- App\Presenters – obdobně jako App\Model, akorát obsahuje třídy presenterů
- App\Utils – pomocné třídy pro výpočty, vykreslování, parsování atp.

Zároveň názvy jmenných prostorů odpovídají adresářové struktuře. To znamená, že třídy ve jmenném prostoru App\Model budou v adresáři /app/model.

3.4.1 Modelové třídy

Modelové třídy jsou jediné třídy, které mají přístup k databázi z celé aplikace. Jejich společným rodičem je třída Database, která v konstruktoru bere jeden parametr a to databázový kontext (instance třídy *Nette\Database\Context*). Zároveň jsou všechny modelové třídy zaregistrovány do konfiguračního souboru aplikace jako služby, díky tomu je možné jednoduše používat model v presenterech.

Jedna třída spravuje jednu entitu, názvy tříd proto mají na konci slovo „manager“ a na začátku název entity (např. UserManager). Typickými funkcemi managerů je přidávání entity do databáze, úprava, mazání a získávání dat z databáze na základě předaných parametrů. Tato data jsou pak typicky vracena v podobě entitní třídy, tzn. při nalezení objektu v databázi přes ObservationManager je vytvořen objekt Observation, naplněn získanými daty a vrácen. Výhodou je možnost nastavení různých omezení při nastavování objektu Observation, nemůže se pak stát, že by se do databáze dostala neplatná data. Další podstatnou výhodou je, že vracíme jasně danou datovou strukturu, pokud bychom vraceli výsledek z databáze, byl by závislý na názvech sloupců a to by představovalo velké problémy, pokud bychom se rozhodli v budoucnu sloupec přejmenovat.

3.4.2 Presentery

Presentery jsou třídy, které zpracují HTTP požadavek přeložený routerem a poté vygenerují odpověď (nejčastěji HTML stránku).

Všechny presentery jsou potomkem třídy *Nette\Application\UI\Presenter* a musí na konci svého názvu obsahovat slovo „Presenter“.

Jeden presenter obsluhuje stránky se stejným tématickým zaměřením. Tedy stránku zobrazující uživatele a jejich seznam bude obsluhovat jiný presenter než stránku, na které se zobrazují projekty.

V presenteru je možné mimo jiné definovat i metody zobrazené na obrázku 3.2. *Startup* je volána ihned po vytvoření presenteru. Je vhodná pro inicializaci proměnných a ověření uživatelského oprávnění (v této aplikaci je využita pro zjištění, zda je uživatel přihlášen a případné přesměrování na přihlašovací stránku).

V metodě *render<View>*[8] se většinou předávají proměnné do šablony, ale je možné zde definovat cokoliv, co se má stát při renderování daného view. Metoda *action<Action>* je podobná, ale *action<Action>* zpravidla nic nevykresluje a volá se dříve než metoda *render<View>* [15].

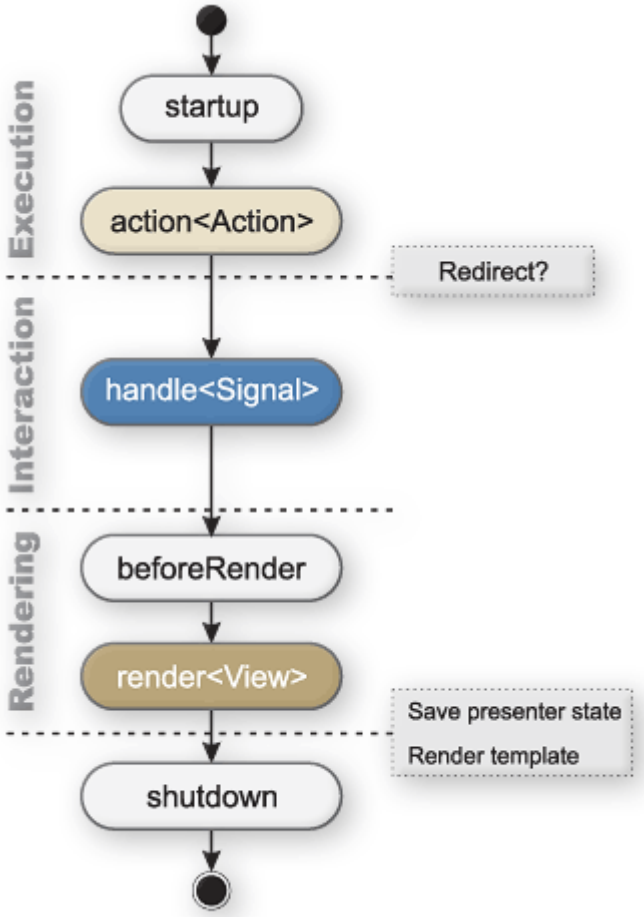
Další metodou může být *handle<Signal>*, která zpracovává signály, například AJAX požadavky.

3.4.3 Komponenty

Komponenta obecně je třída, která je zpravidla potomkem třídy *Nette\Application\UI\Control* a je vykreslitelná. Má vlastní šablonu a metodu *render()*, která předá data do šablony a nakonec šablonu vykreslí.

Komponenty jsou základ pro znovupoužitelnost kódu a zároveň umožňují využívat práce komunity.

8. <View> představuje název view, který chceme zobrazit. Podobně u dalších příkladů



Obrázek 3.2: Životní cyklus presenteru [15]

4 Popis implementace nové aplikace

V této kapitole popisuji konkrétní implementaci nové aplikace MECA. Aplikace je již nasazena na produkčním serveru, ale jelikož je stále ve vývoji, budu zde popisovat verzi aplikace 0.9.6. Změny v jednotlivých verzích aplikace je možné shlédnout v changelogu po přihlášení.

4.1 Aplikace z pohledu uživatele

Pro přístup do aplikace je potřeba být zaregistrován a přihlášen. Pokud tomu tak není, je uživatel vždy přesměrován na přihlašovací stránku, kde je i nabídka registrace. Samotná registrace vyžaduje unikátní uživatelské jméno, unikátní email a heslo. Pokud registrace proběhla úspěšně je uživatel přesměrován na stránku přihlášení, kde již může vstoupit do aplikace, pokud se vyskytla chyba, uživatel je vyzván, aby ji opravil.

Po přihlášení se uživatel ocitne na domovské stránce, která zatím obsahuje pouze informaci o verzi aplikace a varování, že aplikace je stále ve vývoji. V budoucnu se zde mohou vyskytovat např. novinky v aplikaci, nová pozorování a zajímavé statistiky.

Základní struktura stránky je hlavička, obsah a zápatí, kde hlavička obsahuje menu připnuté k horní části obrazovky, tělo je dynamický obsah dle vybraného view a zápatí obsahuje technické informace jako je copyright a verzi PHP a Nette. Menu obsahuje následující položky:

- Logo a název aplikace – odkáže na hlavní stránku.
- Observations – dropdown, který umožňuje prohlížet pozorování, zobrazit vlastní pozorování a nebo přidat nové.
- Users – zobrazí seznam registrovaných uživatelů.
- Projects – obdobně jako u Observations, umožňuje zobrazit všechny projekty, projekty, které uživatel vlastní a nebo je členem a také možnost vytvořit nový projekt.

- Objects – možnost zobrazit všechny objekty přidané do aplikace nebo manuálně přidat nový objekt.
- Help – tato položka byla přidána kvůli spolupráci s uživateli, je zde možnost zobrazit seznam změn, které eviduji od spuštění aplikace na produkčním serveru (od 11.5.2015), a možnost nahlásit chybu nebo návrh na zlepšení.
- My account – poslední položka menu, přes kterou se může uživatel odhlásit, zobrazit svůj uživatelský profil, nebo se dostat do nastavení, kde si tento profil může upravit.

4.1.1 Nastavení

Nastavení uživatele zatím obsahuje čtyři části – Úprava profilu, změna profilového obrázku, změna hesla a nastavení tzv. Source preset. V úpravě profilu lze změnit informace jako jsou jméno, tituly před a za jménem, pracoviště, popř. další informace, který je uživatel ochoten o sobě napsat. Všechny tyto údaje jsou veřejné a zobrazeny v uživatelově profilu. Nastavení Source preset je uživatelův osobní seznam přednastavených zdrojů, které může mazat a přidávat. Tento seznam slouží pouze k nastavení zdroje, který je možné uvést v pozorování.

4.1.2 Objekty

Na stránce Objects se vypisuje seznam všech souhvězdí, přičemž u každé položky se vypisuje číslo reprezentující počet objektů přidaných k danému souhvězdí. Při kliknutí na souhvězdí se zobrazí podseznam se všemi objekty, které obsahuje. Objekty bez souhvězdí jsou vypisovány zvlášť na konci stránky.

Při kliknutí na objekt je pak uživatel přesměrován na stránku s detailem objektu. Zde jsou zobrazeny důležité informace o objektu jako souřadnice, světelné elementy M_0 a P, a pokud objekt existuje v databázi VSX[1], pak je zde uveden i odkaz na detail objektu přímo na webu VSX. Pod informacemi je pak omezený seznam posledních po-

1. <ftp://cdsarc.u-strasbg.fr/pub/cats/B/vsx/vsx.dat.gz>

zorování přidaných k danému objektu s možností vykreslení fázové křivky a s odkazem na kompletní seznam pozorování k objektu.

Mimo jiné může uživatel přidat nový objekt do databáze, ale pouze v případě, že nebyl nalezen v žádné jiné externí databázi. Více o tomto problému ja napsáno v kapitole 4.3

4.1.3 Uživatelé

V seznamu uživatelů se zobrazuje jejich uživatelské jméno, jméno, které si vyplnili v nastavení profilu, počet pozorování a pracoviště.

Je-li to možné, pak jednotlivé položky seznamu tvoří odkaz na detail zmíněné položky. Toto chování je obecné pro celou aplikaci, aby byla navigace pro uživatele co nejjednodušší. V případě seznamu uživatelů je při kliknutí na počet pozorování uživatel odkázán na seznam pozorování vyfiltrovaný dle vybraného uživatele.

Na detail uživatele v seznamu je možné se dostat kliknutím na uživatelské jméno. Tento detail slouží jako veřejný profil, který je možné upravit vlastníkem v Nastavení (viz kapitola 4.1.1). Tento profil mimo jiné obsahuje také informace o projektech, jichž je uživatel součástí nebo jež vytvořil, nebo seznam posledních pozorování.

4.1.4 Projekty

Každý uživatel má možnost vytvořit projekt. Tyto projekty slouží ke shromáždění určitých pozorování na jedno místo a jejich zpřístupnění vybraným uživatelům.

Po vytvoření projektu, ke kterému je potřeba pouze zadat název a popis, má vedoucí projektu možnost přidávat či odebírat uživatele jako členy. Všichni členové pak mohou přispívat svými pozorováními do projektu a tato pozorování jsou viditelná pouze členům projektu.

V seznamu projektů se zobrazuje název projektu a jeho popis, dále jméno vedoucího projektu (s odkazem na jeho profil) a počet členů projektu, podle těchto jednotlivých sloupců je možné výsledky vzestupně nebo sestupně třídit. Seznam je stránkovaný po dvaceti položkách, aby nedocházelo k zahlcení stránky.

Pokud je projekt smazán, všechna pozorování v něm uložená se z projektu vyjmou a zůstanou bez projektu.

4.1.5 Pozorování

Seznam pozorování zobrazuje všechna pozorování v databázi. U veřejně nepřístupných pozorování lze pouze zobrazit jeho detail a prohlédnout data nebo křivky. Uvedeny jsou základní informace o pozorování, což jsou: objekt, vlastník, projekt, filtr, data pozorování a přidání, popř. poznámka. Pokud je pozorování přístupné, lze si kliknutím na ikonu oka zobrazit celý detail pozorování. Pokud je navíc přístupný i graf pozorování, lze si hned v seznamu zobrazit zmenšený náhled grafu kliknutím na ikonu přiblížení.

Protože pozorování budou tisíce, nelze zobrazovat všechna data najednou, ale je potřeba zavést určitý druh stránkování. Klasické stránkování zde ovšem není vhodné kvůli vykreslování fázové křivky, proto jsou data načítána po částech. Jakmile uživatel dojde na konec seznamu, má možnost si seznam prodloužit načtením dalších položek až dokud nebudou vypsány všechny výsledky. Komponentu pro seznam pozorování detailněji popisuji v kapitole 4.5.

Výsledky lze samozřejmě filtrovat a řadit podle všech dostupných kritérií, filtr je zobrazen hned nad seznamem pozorování. Jakmile jsou navíc výsledky filtrovány dle určitého objektu, zobrazí se v seznamu možnost vykreslení fázové křivky. Informace o technickém provedení filtru se nachází v kapitole 4.6.

V detailu pozorování se zobrazují kompletní informace uložené k pozorování a v závislosti na nastavení soukromí se zobrazují i data nebo křivky. Všechny vyplnitelné záznamy lze vlastníkem editovat za předpokladu, že se pozorování nenachází v projektu. Pokud se pozorování nachází v projektu, je potřeba jej nejprve vyjmout z projektu, upravit potřebné informace a poté znovu vložit do projektu.

4.2 Proces přidávání pozorování

Aby bylo možné přidat pozorování k určitému objektu, je nutné, aby objekt existoval v lokální databázi. Uživatel má možnost buď'to objekt vybrat z nabídky objektů, a nebo vyhledat (viz kapitola 4.3).

Jakmile je vybrán objekt, uživatel vyplní formulář s údaji o pozorování, kde povinnými údaji jsou filtr, formát juliánského datování, fotometrický systém a minimálně jeden datový soubor. Nepovinně je možné vyplnit zdroj pozorování, poznámku, nahrát soubor s map-

kou, přiřadit pozorování k projektu, nebo určit viditelnost pozorování (výchozí nastavení je soukromé).

Po odeslání formuláře jsou zkontrolovány údaje a soubory a pokud je vše pořádku, údaje jsou uloženy do databáze, soubory nahrány a uživatel je přesměrován na stránku, kde se zobrazí seznam právě nahraných pozorování. Pokud se v průběhu nahrávání objeví chyba, např. z důvodu chybného souboru, již nahraná pozorování a soubory jsou smazány a uživatel je vyzván chybu opravit.

4.3 Problém jednoznačnosti objektů

Astronomické objekty mívají zpravidla více názvů dle jejich výskytů v různých katalozích, pozorovacích projektech apod. Příkladem může být proměnná hvězda V535 Ara, která má dle serveru Simbad až 19 názvů [16]. Přestože je jméno V535 Ara označeno jako jméno hlavní, odkazovat na tuto hvězdu pod jejím jiným jménem není špatně. Zde může nastat situace, že nějaký uživatel může znát danou hvězdu pod jiným jménem, než jaké je uvedeno v naší aplikaci. Otázkou tedy je, jak zabránit duplikátům objektů a jak sjednotit všechny názvy pod jeden záznam v naší databázi? A který název má být použit v naší databázi?

Částečným řešením tohoto problému je v použití externích databází, které tyto názvy obsahují. Použity jsou databáze VSX (Variable star index) a služba Sesame Name Resolver[2], ze které využívám pouze databázi Simbad. Sesame poskytuje veřejnou službu, do které je možné zadat jméno hvězdy a server vrátí výsledek ve formátu XML. Tento výsledek obsahuje všechny aliasy dané hvězdy. VSX bohužel žádnou podobnou službu neposkytuje, ale poskytuje celou databázi v ASCII formátu ke stažení. Databáze VSX je proto stažena, rozparsována a poté jsou důležité údaje vloženy do lokální tabulky VSX, ve které už je možné snadno vyhledávat.

Bohužel ani jedna databáze není stoprocentně kompletní a aktuální, některé objekty, které se vyskytují v databázi VSX se nevyskytují v Simbadu a naopak. Některé nové objekty navíc nemusí být ani v jedné databázi.

---

2. `<http://cds.u-strasbg.fr/cgi-bin/Sesame>`

4.3.1 Postup hledání objektu

Po vložení názvu a odeslání formuláře je nejprve prohledána lokální databáze, zda-li neobsahuje hledaný objekt. Pokud ano, je uživatel přesměrován na stránku, kde může přidat pozorování k nalezenému objektu (viz 4.2).

Pokud objekt není nalezen, je prohledána databáze VSX a poté Simbad. Pokud není objekt nalezen ani v jednom případě, je uživatel upozorněn, že objekt v žádné databázi neexistuje. V opačném případě se nalezený objekt vždy porovná s lokální databází dle vsx_id nebo resolver_id, což jsou jednoznačné identifikátory objektů z databází VSX a Simbad, které jsou spolu s objektem ukládány do lokální databáze. V případě, že byla nalezena shoda v lokální databázi, postupuje se stejně jako v prvním případě. Nebyla-li nalezena shoda, pak je uživatel přesměrován na stránku, kde má možnost přidat nově nalezený objekt do databáze a nastavit mu správné souhvězdí (je-li souhvězdí obsaženo v názvu hvězdy, pak je automaticky zparsováno a není potřeba jej zadávat). Po přidání je pak přesměrován na stránku s formulářem pro nové pozorování.

Samotné hledání má několik variací. Pokud je objekt nalezen přes databázi VSX, je zároveň prohledán i Simbad. Pokud je tomu naopak, pak je VSX prohledána pod každým aliasem ze Simbadu, dokud není nalezena shoda. Jméno objektu je pak vždy určeno podle databáze VSX (která obsahuje pouze jedno jméno). Pokud se však objekt ve VSX nevyskytuje, je uživateli ponechána možnost vybrat název z aliasů, které poskytl Simbad.

4.3.2 Možné problémy

Vypracované řešení bohužel není bezchybné, což vychází ze závislosti na externích službách. XML dokument, který je poskytován službou Sesame, podle XML Schematu na adrese `<http://vizier.u-strasbg.fr/xml/sesame_4x.xsd>` nezaručuje zobrazení aliasů. V některých vzácných případech tedy může nastat (a již se mi několikrát v průběhu vývoje stalo), že se aliasy nezobrazí. Pak se nemusí výsledek najít v databázi VSX a tím objekt nebude mít přístup ke světelným elementům M_0 a P.

Dalším problémem mohou být zastaralá data, tedy případ, že re-

lativně nově objevená hvězda dostane po čase unifikovaný název, nebo je přidána do databáze, kde dřív nebyla.

V neposlední řadě je tu problém s objekty, které byly do systému manuálně přidány a tedy nejsou propojeny ani se Simbadem ani s VSX. Takové objekty však mohou být časem do těchto databází přidány.

4.3.3 Automaticky spouštěné skripty

Řešením problému zastaralých dat mohou být automaticky spouštěné skripty. To jsou PHP skripty, které periodicky spouští server vždy v určitý čas.

Momentálně je součástí aplikace skript, který automaticky stáhne databázi VSX, zkontroluje, zda-li je novější než předchozí verze a popř. ji extrahuje, rozparsuje a nahradí lokální tabulku VSX za novou.

V budoucnu by mohly existovat skripty, které by kontrolovaly objekty, jež nejsou přiřazeny k některé z externích databází, zda-li nebyla nalezena shoda v externí databázi, poté přidaly chybějící ID a případně upravily název objektu.

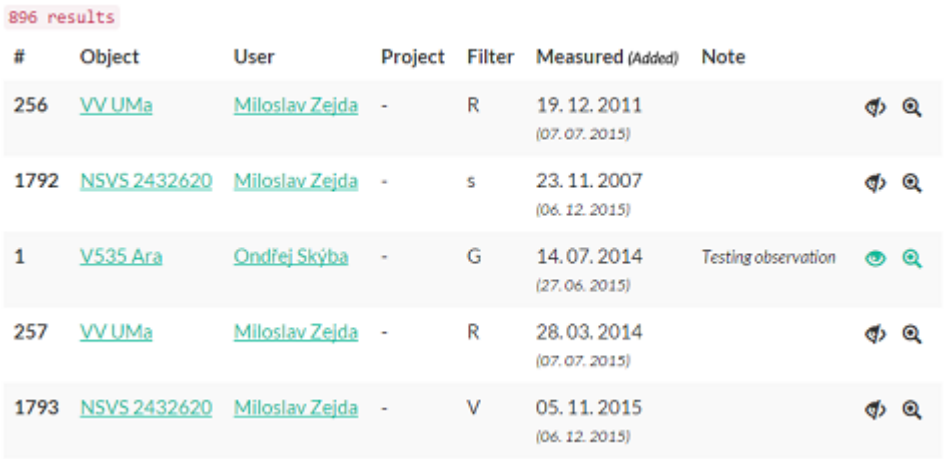
4.4 Heliocentrická korekce

Astronomická pozorování používají pro zaznamenání času Juliánského data. Toto datum může být navíc v geocentrickém, nebo heliocentrickém formátu. Geocentrická verze juliánského data je světový čas UTC přepočtený na dny a zlomky dne pro dané místo pozorování. Heliocentrické juliánské datum navíc obsahuje tzv. heliocentrickou korekci, která odstraňuje vliv oběhu Země kolem Slunce; pozorovatel jako by se nacházel uprostřed Slunce. Pro skládání časových řad v astronomii je nutné používat alespoň heliocentrické juliánské datování.

Funkce pro vypočítání této korekce se nachází jako veřejná statická metoda ve třídě *App\Utils\Conversions*. Před vykreslením grafu nebo fázové křivky je vždy zkontrolováno, zda-li není pozorování v geocentrickém formátu a případně je přičtena vypočtená korekce.

4.5 Komponenta pro seznam pozorování

Seznam pozorování je komponenta s názvem *ObservationList*, která má na starosti zobrazit vybraná pozorování v pevně daném formátu s možností vykreslení fázové křivky. Účelem této komponenty je mít možnost jednoduše vkládat jednotný seznam pozorování do různých částí aplikace a zabránit mnohonásobné duplikaci kódu.



Obrázek 4.1: Ukázka seznamu pozorování

Komponentě se při vytváření předává databázový kontext, aby mohla pracovat s modelovými třídami. Konkrétně využívá *ObservationManager* pro získání vybraných pozorování a třídy *VSX* a *PersonalElementsManager* pro zjištění elementů objektu a jejich použití ve fázové křivce.

Při použití komponenty je nutné v presenteru nastavit, jaká pozorování se budou zobrazovat a jaký je jejich celkový počet a po kolika výsledcích se budou pozorování načítat. Ve třídě *ObservationManager* je definována funkce *getObservationSelection*, která jako parametry bere podmínky pro vyfiltrování výsledků. Funkce vrátí instanci třídy *Nette\Database\Table\Selection*, kterou je možno dále upravovat, nebo použít pro získání výsledků z databáze. Právě objekt typu *Selection* se předává komponentě, která jej dále použije předáním třídě *ObservationManager* a získáním výsledků z databáze.

Důvodem, proč se komponentě nepředávají rovnou objekty *Ob-*

servation, ale *Selection*, je nutnost výsledky načítat postupně. Dejme tomu, že výsledků s použitým filtrem je 43, pak komponenta získá z databáze pouze počet výsledků na stránku, například dvacet. Jakmile uživatel klikne na tlačítko „Load more“, je z databáze vytaženo dalších dvacet výsledků a ty jsou připojeny k již zobrazeným výsledkům. Při této činnosti se však překreslí pouze část komponenty a zbytek si ponechává svůj stav. Presenter nic o stránkování komponenty neví a ani nic ve své šabloně nepřekresluje, bylo by proto nemožné předávat výsledky rovnou z presenteru. Při dalším načtení pozorování by komponenta již připojila pouze zbývající tři výsledky a skryla tlačítko „Load more“.

Volitelně je možné komponentě nastavit dva příznaky. Prvním je *expandable*, nastavením na *true*, což je také výchozí možnost v případě, že příznak není nastaven, zajistíme, že výsledky mohou být rozšiřovány tlačítkem „Load more“, v opačném případě nebude možnost zobrazena. Druhým příznakem je *phase*, ten v případě nastavení na *true* zobrazí možnost vykreslení fázové křivky, která je ve výchozím nastavení skryta. Zobrazit fázovou křivku je možné pouze tehdy, je-li filtr nastaven podle nějakého objektu, to se ale nedá zjistit přímo v komponentě, proto je nutné tento příznak nastavit již v presenteru.

ObservationList je využit v rozšiřitelné podobě na stránce s kompletním výpisem pozorování. Dále je však také využit omezený výpis pozorování na stránkách s detailem uživatele, projektu a objektu.

4.6 Komponenta pro filtrování pozorování

ObservationFilter je důležitá komponenta, která omezuje výsledky pozorování zadanými podmínkami. Je zatím využita pouze v *ObservationsPresenter*, tedy na stránce s výpisem všech pozorování, a slouží spíše pro oddělení a přehlednost kódu.

Filtrovat je možné podle objektu, projektu, uživatele, filtru a podle data, před a po kterým bylo pozorování zaznamenáno. Součástí filtru je také možnost řazení výsledků vzestupně, nebo sestupně, podle jednotlivých položek filtru.

Po kliknutí na tlačítko „Apply filter“, se hodnoty ve filtru uloží do session a stránka se aktualizuje. Nastavení filtru se do seznamu

pozorování propaguje tak, že při každém načtení stránky se přečtou hodnoty uložené v session, ty se zpracují a na jejich základě se vytvoří *Selection*, který je předán komponentě *ObservationList*, která již zobrazí vyfiltrované výsledky.

Díky použití session lze snadno nastavit filtr i mimo komponentu. *ObservationsPresenter* obsahuje akci *filter*, která z get parametrů získá nastavení filtru, např. *object=1*. Akce *filter* má za úkol tyto parametry zpracovat, uložit do session a přesměrovat na defaultní view, které zobrazí pozorování. Na tuto akci lze snadno odkazovat, využití proto najde např. v detailu projektu (a jiných entit), kde můžeme odkazem přesměrovat na již vyfiltrované výsledky podle daného projektu, aniž bychom museli cokoliv nastavovat, nebo mít nepřehlednou URL plnou *get* parametrů.

Aby však filtr i vizuálně nabýval hodnot, které jsou v session uloženy i po aktualizaci stránky nebo po využití akce *filter*, tak i komponenta při svém načtení zpracovává hodnoty session. Podle nich pak nastaví výchozí hodnoty v šabloně filtru tak, aby vizuálně odpovídaly. Pokud by se ve filtru při načtení objevila neplatná hodnota, pak je tato hodnota ze session vymazána.

4.7 Vykreslování fázové křivky a grafů

Generování obrázků je stejně jako ve staré aplikaci řešeno vestavěnými PHP funkcemi. O vykreslování grafu pozorování se stará třída *GraphDrawer*, v případě fázové křivky je to třída *PhaseDrawer*. Jejich společným rodičem je abstraktní třída *ImageDrawer*. Tato třída má konstruktor, který bere jako parametry výšku a šířku vykreslovaného obrázku. Dále definuje společné konstanty (velikost bodu, odsazení atd.), kreslící plátno, barvy a také metody (vykreslení bodu, chybové úsečky, rozhraní grafu atd.).

Zobrazení těchto grafů je řešeno podobně jako zobrazení klasického view. V presenteru existuje místo renderovací metody metoda *actionDataImage*, resp. *actionPhase*, která nemá definovanou šablonu a typ obsahu je nastaven na *image/jpeg* nebo *image/png*. Akce, stejně jako kterékoliv jiné view se šablonou, berou *get* parametry určující zobrazený graf/křivku. Poté je vytvořen patřičný objekt, který je naplněn získanými hodnotami a zavolána metoda *drawJPG()*. V pří-

padě chyby je chyba vypsána do obrázku a zobrazena.

4.7.1 Graf pozorování

Třídě *GraphDrawer* je kromě výšky a šířky potřeba předat objekt typu *Observation*, který představuje vykreslované pozorování, poté číslo vykreslovaného sloupce a příznak, zda-li se mají vykreslovat chybové úsečky.

Prvním krokem je načtení datového souboru pozorování a získání potřebných dat. O to se stará třída *DataParser*, která přečte datový soubor, vynechá řádky, které obsahují chybové hodnoty a vrátí pole všech bodů, kde jeden bod se skládá z hodnot x (Juliánské datum), y (hvězdná velikost v magnitudách) a error (chyba určení hvězdné velikosti).

Následující text je ukázka datového souboru. První dva řádky představují hlavičku souboru, poté již následují data, kde první sloupec je Juliánské datum, druhý sloupec je hvězdná velikost vyjádřená v magnitudách a následuje sloupec s hodnotou chyby v určení hvězdné velikosti. Sloupců s hvězdnou velikostí a chybou může být více a každá tato dvojice představuje jiný graf.

```
JD V-C s1
Aperture: 4, Filter: Green, JD: geocentric
2456858.66732 0.90609 0.01497
2456858.66617 0.90161 0.01477
2456858.66503 0.92919 0.01465
2456858.66388 0.93976 0.01515
2456858.66273 0.95839 0.01495
2456858.66158 0.93731 0.01430
2456858.66043 0.95589 0.01505
2456858.65928 0.99467 0.01536
2456858.65813 0.96212 0.01412
2456858.65698 0.96350 0.01477
2456858.65584 1.00291 0.01493
2456858.65469 1.00682 0.01518
2456858.65354 1.02397 0.01507
2456858.65239 0.98794 0.01475
```

Po převedení datového souboru do pole následuje vykreslování

jednotlivých bodů a v závislosti na příznaku předaném v konstruktoru také chybových úseček.

O zobrazení grafů uživateli se stará komponenta *GraphBrowser* (obrázek 4.3), která umožňuje snadno přepínat mezi dostupnými grafy, vypínat a zapínat chybové úsečky, nebo měnit velikost obrázku.

Mezi velká vylepšení oproti grafům vykreslovaným v předchozí aplikaci patří dynamické metriky grafu. A také jsou opraveny případy, kdy bod nebo chybová úsečka mohly přesahovat hranice grafu.

4.7.2 Fázová křivka

Při vytváření třídy *PhaseDrawer* se kromě výšky a šířky grafu předává pole objektů *Observation* představující všechna pozorování zahrnutá ve křivce, dále pole vertikální posunů a nakonec objekt typu *PhaseCurve*. Třída *PhaseCurve* obsahuje všechny parametry fázové křivky, kterými jsou: světelné elementy M_0 a P (povinné parametry), vymezení os mezi zadané hodnoty, zvýraznění os na určité hodnotě, zobrazení aktuálního času na ose X, zobrazení libovolného data a možnost zobrazit chybové úsečky. Poslední možností je jednotlivé body v grafu barevně odlišovat na základě jejich filtru, nikoliv pozorování. Tato poslední možnost byla přidána až v pozdější fázi vývoje na základě požadavků od uživatelů.

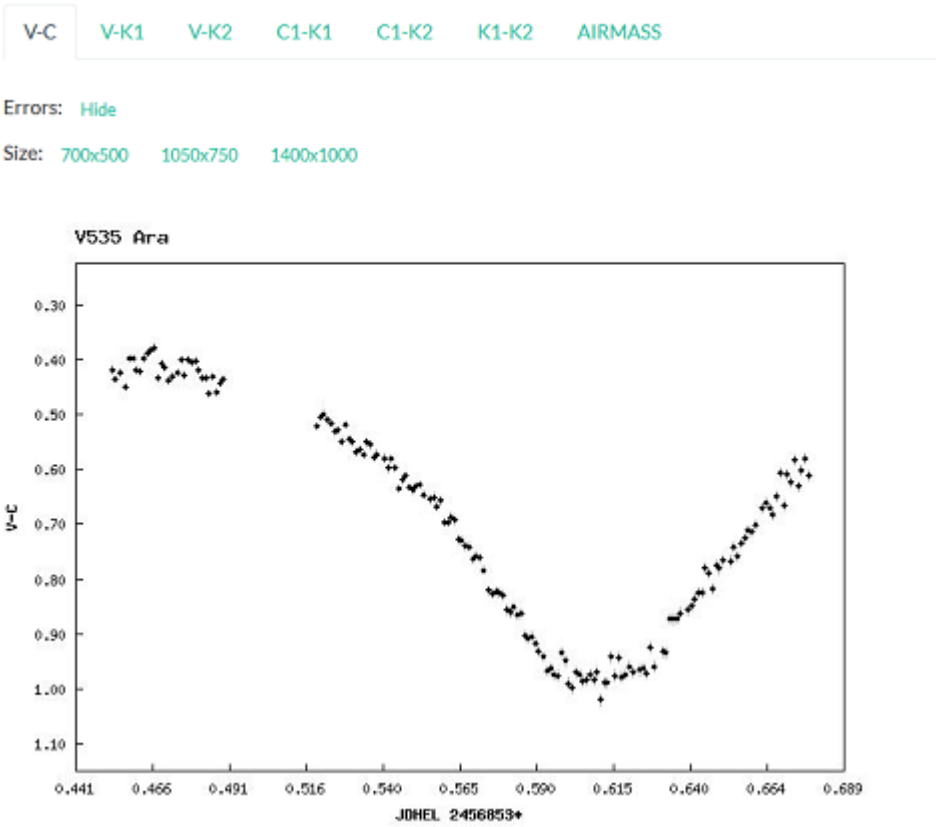
Výhodou použití objektu *PhaseCurve* je ošetřování neplatných hodnot hned při vytváření objektu, tudíž se kontrolou chyb nemusí zabývat třída *PhaseDrawer*.

Obdobně jako u grafu jednoho pozorování, i u fázové křivky jsou nejprve převedena data do polí a následně vykreslena. Zajímavou částí je zde výpočet fáze jednotlivých bodů. Ta se spočítá odečtením M_0 od zaznamenaného času (v Juliánském datu), výsledek je vydělen periodou a z celkového výsledku je odebrána pouze desetinná část, která představuje chtěnou fázi.

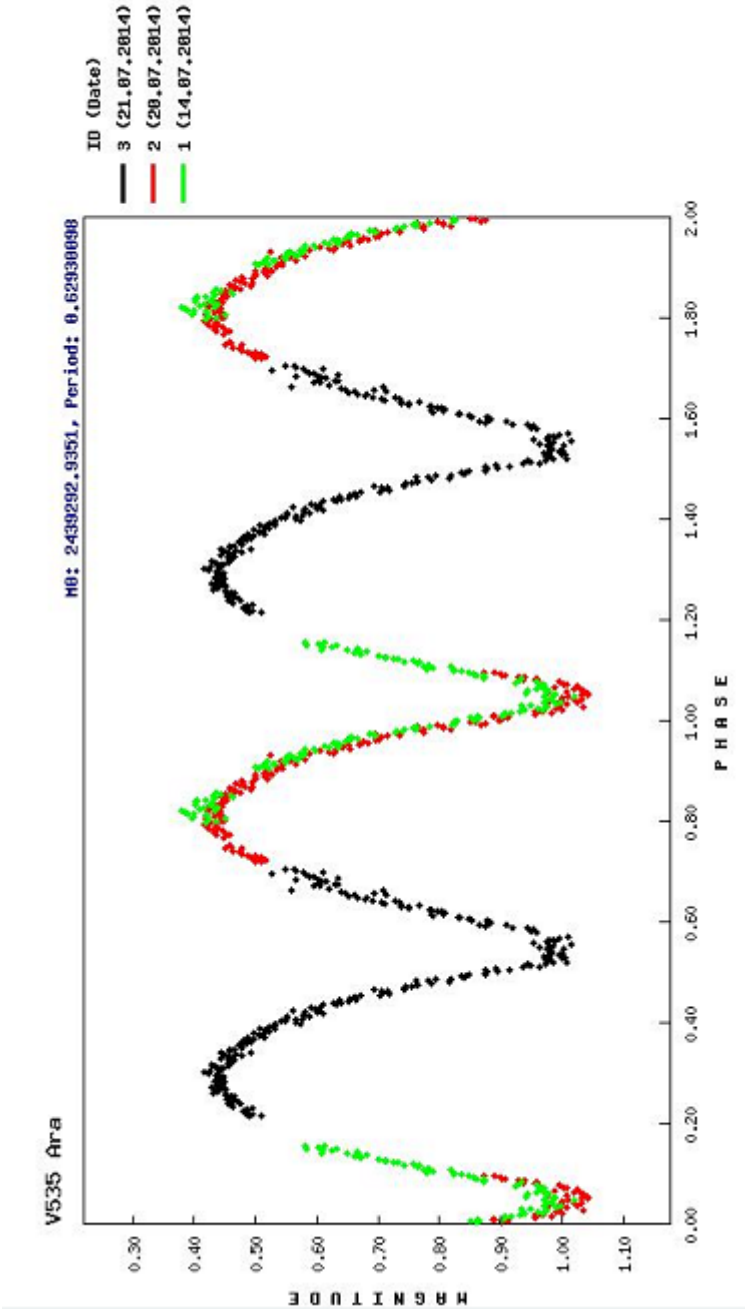
Fázová křivka vypadá velmi podobně jako ve staré verzi, ale bylo provedeno velké množství oprav v zobrazování, popisky jsou dynamické a tedy je možné libovolně měnit velikost grafu, či měnit popisky samotné. Nedojde k přesáhnutí žádné hranice grafu, což byl v předchozí verzi velký problém, který nastával především při zvět-

šení grafu omezením os. Také oproti staré verzi přibyly uživatelské možnosti, kterými je možné křivku přizpůsobit. Náhled na ovládací panel fázové křivky je na obrázku 4.2 spolu se seznamem pozorování.

Obrázek 4.2: Ukázka seznamu pozorování s možností zobrazení fázové křivky



Obrázek 4.3: Ukázka komponenty GraphBrowser a grafu pozorování proměnné hvězdy V535 Ara



Obrázek 4.4: Fázová křivka nové aplikace skládající se ze tří pozorování proměnné hvězdy V535 Ara

5 Záměry do budoucnosti

I přes širokou škálu funkcí, které aplikace nabízí, není ještě plně hotová a plánuji přidávat další funkčnost a upravovat stávající.

Největší plánovanou změnou je určování typu datového souboru při přidávání pozorování. Datové soubory mohou mít různý formát, který se lehce liší například v hlavičce souboru, která navíc i může chybět. Pokud by se zavedlo určování typu souboru při nahrání, dalo by se hned snadno zjistit, zda-li je soubor validní podle zadaného typu. Pro parsování dat by pak existovalo více tříd, jejichž funkce by určovalo rozhraní *IDataParser*, ale implementace by byla rozdílná dle typu souboru. Výhodou tohoto přístupu v budoucnu je snadné přidání nového typu souboru.

Mezi další plány patří možnost exportu dat vybraných ve fázové křivce, korekce pro barycentrické Juliánské datum[1], rozšíření administrátorských možností (funkce administrátora byly v rámci priorit vývoje přesunuty na nejnižší pozici) a některé další drobné úpravy kódu.

1. Čas měřený vůči středu sluneční soustavy

6 Závěr

Cílem bakalářské práce bylo nejprve analyzovat webovou aplikaci pro astronomické deníky využívanou Ústavem teoretické fyziky a astrofyziky na Přírodovědecké fakultě Masarykovy univerzity. Tuto aplikaci následně reimplementovat a přidat požadovanou funkčnost.

První část práce se zabývá analýzou původní aplikace, je popsána funkčnost a jsou jí vytknuty nedostatky ve funkčnosti i v uživatelském rozhraní. Následují požadavky na novou aplikaci, které se dají rozdělit na funkční a nefunkční.

Zbytek práce se již věnuje nové aplikaci, která je psaná v jazyce PHP. Pro vývoj aplikace bylo využito Nette Frameworku a JavaScriptových knihoven, které jsou podrobněji popsány v kapitole třetí. Následuje návrh databáze a tříd, které bude aplikace využívat.

Čtvrtá kapitola nejprve popisuje aplikaci z pohledu uživatele a poté se zaměřuje na jednotlivé implementační problémy a části aplikace, které si zaslouží speciální pozornost. V kapitolách jsou místy popsána možná implementační řešení a vylepšení do budoucnosti a ta, která nebyla sdělena v kontextu, se nachází v kapitole páté.

Aplikace byla vyvíjena iterativním procesem tvorby a pravidelně konzultována se zadavatelem. Díky tomu byla také v průběhu vývoje několikrát pozměněna. Aplikace je v provozu od 11.05.2015 na serverech Masarykovy univerzity, splňuje všechny požadavky v rámci této bakalářské práce a je aktivně využívána uživateli (k 1. lednu 2016 obsahuje téměř 2000 záznamů). Přesto je aplikace stále vyvíjena a zlepšována.

Literatura

[1] The Internet Society *RFC 2617 - HTTP Authentication: Basic and Digest Access Authentication*. [online]. 1999 [cit. 21.10.2014]. Dostupné z: <http://tools.ietf.org/html/rfc2617>

[2] PHP Group. *History of PHP*. [online]. 2001 [cit. 28.10.2014]. Dostupné z: <http://php.net/manual/en/history.php.php>

[3] PHP Group. *session_destroy*. [online]. [cit. 31.10.2014]. Dostupné z: <http://php.net/manual/en/function.session-destroy.php>

[4] PHP Group. *session_unset*. [online]. [cit. 31.10.2014]. Dostupné z: <http://php.net/manual/en/function.session-unset.php>

[5] Dhakar, L. *Lightbox*. [online]. 1999 [rev. 14.4.2014] [cit. 20.10.2014]. Dostupné z: <http://lokeshdhakar.com/projects/lightbox2>

[6] Nette Foundation. *Licenční politika*. [online]. 2014 [cit. 10.12.2015]. Dostupné z: <https://nette.org/cs/license>

[7] Nette Foundation. *Model-View-Presenter*. [online]. 2015 [cit. 10.12.2015]. Dostupné z: <https://doc.nette.org/cs/2.3/glossary#toc-model-view-presenter>

[8] Nette Foundation. *Model-View-Presenter (MVP)*. [online]. 2015 [cit. 10.12.2015]. Dostupné z: <https://doc.nette.org/cs/0.9/model-view-presenter>

[9] Nette Foundation. *Debugování a zpracování chyb*. [online]. 2015 [cit. 10.12.2015]. Dostupné z: <https://doc.nette.org/cs/2.2/debugging>

[10] Nette Foundation. *Přednosti Nette*. [online]. 2015 [cit. 10.12.2015]. Dostupné z: <https://nette.org/cs/#toc-features>

[11] Nette Foundation. *Router*. [online]. 2015 [cit. 13.12.2015]. Dostupné z: <https://doc.nette.org/cs/2.2/glossary#toc-router>

[12] jQuery Foundation. *jQuery*. [online]. 2015 [cit. 10.12.2015]. Dostupné z: <https://jquery.com/>

[13] Otto, M., Thorton, J. *Bootstrap*. [online]. 2015 [cit. 10.12.2015]. Dostupné z: <https://github.com/twbs/bootstrap>

[14] Malot, S. *Bootstrap datetime picker*. [online]. 2012 [cit. 10.12.2015]. Dostupné z: <https://github.com/smalot/bootstrap-datetimepicker>

[15] Nette Foundation. *Životní cyklus presenteru*. [online]. 2015 [cit. 14.12.2015]. Dostupné z: <https://doc.nette.org/cs/2.3/presenters#toc-zivotni-cyklus-presenteru>

[16] *V535 Ara*. [online]. 2016 [cit. 3.1.2016]. Dostupné z: <http://simbad.u-strasbg.fr/simbad/sim-id?Ident=V535Ara>