Masarykova univerzita
Fakulta informatiky

Webová aplikace pro astronomické pozorovací deníky

Bakalářská práce

Robert Havlíček

Brno, jaro 2015

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.

Robert Havlíček

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

Poděkování

Rád bych poděkoval svému vedoucímu bakalářské práce RNDr. Martinu Kubovi, Ph.D. a mému konzultantovi doc. RNDr. Miloslavu Zejdovi, Ph.D. za trpělivost a cenné rady. A dále bych chtěl poděkovat mé rodině a přítelkyni za neustálou podporu.

Shrnutí

Tato bakalářská práce je zaměřena na vývoj webové aplikace pro vedení pozorovacích deníků pro Ústav teoretické fyziky a astrofyziky Přírodovědecké fakulty Masarykovi univerzity. Vývoj aplikace probíhal na základě analýzy a reimplementace staré aplikace. Zároveň byla aplikace rozšířena o požadované funkce pomocí iterativního procesu tvorby aplikací.

Klíčová slova

webová aplikace, pozorovací deník, astrofyzika, ASP .NET, MVC

Obsah

1 Úvod

Astronomický deník je neodmyslitelný pomocník každého hvězdného badatele. Slouží především k chronologickému ukládání prováděných pozorování hvězd a dalších kosmických útvarů. Můžeme zde zaznamenávat aktuální svítivost hvězd, vzdálenost hvězd od Země a další astronomické jevy související s pozorováním. Dále si sem většina pozorovatelů zapisuje i použitý přístroj při pozorování, jeho nastavení, použité filtry aj. Všechny údaje jsou většinou ukládány v juliánském datu.

Prvním cílem této práce bylo analyzovat webovou aplikaci pro vedení pozorovacích deníků, dostupnou na odkazu [1]. Tuto aplikaci používají pracovníci v Ústavu teoretické fyziky a astrofyziky na Přírodovědecké fakultě Masarykovy univerzity. Aplikace umožňuje vytvářet databázové záznamy pro souhvězdí a jejich hvězdy. Následně je možné ke každé hvězdě přidávat záznamy o pozorování svítivosti a dopočítávat periodu hvězdy pro neúplná data.

Druhým cílem této práce bylo vytvořit novou webovou aplikaci pro pozorovací deníky na základě předchozí analýzy doposud používané verze. Nová aplikace měla navíc umožňovat sdílení dat mezi jednotlivými uživateli.

Práce je rozdělena do pěti kapitol. Ve druhé kapitole se věnuji analýze aktuálně používané aplikace. Třetí kapitola popisuje použité technologie a frameworky. Ve čtvrté kapitole se soustředím na vlastní implementaci nové aplikace. V poslední, páté kapitole shrnuji výsledek práce a představuji možnosti rozvoje nové aplikace v budoucnosti.

2 Aplikace MECA

Ačkoliv je v oficiálním zadání uveden požadavek na analýzu aplikace *Sekce proměnných hvězd a exoplanet ČAS*[1], zadavatel si přál reimplementovat aplikaci *MECA*[2] *(Measurements of Eclipsing Binaries – Archive)*. Tato aplikace je upravenou verzí aplikace z původního zadání. Z tohoto důvodu jsem se rozhodl analyzovat aplikaci *MECA*.

Pracovníci Ústavu teoretické fyziky a astrofyziky Přírodovědecké fakulty Masarykovy univerzity aktuálně používají aplikaci *MECA* pro ukládání dat k pozorování proměnných hvězd. Kvůli jeho dnes již nepostačujícím funkcím a velkému pracovnímu vytížení autora systému bylo rozhodnuto, že je třeba tuto aplikaci předělat a rozšířit o požadované funkce.

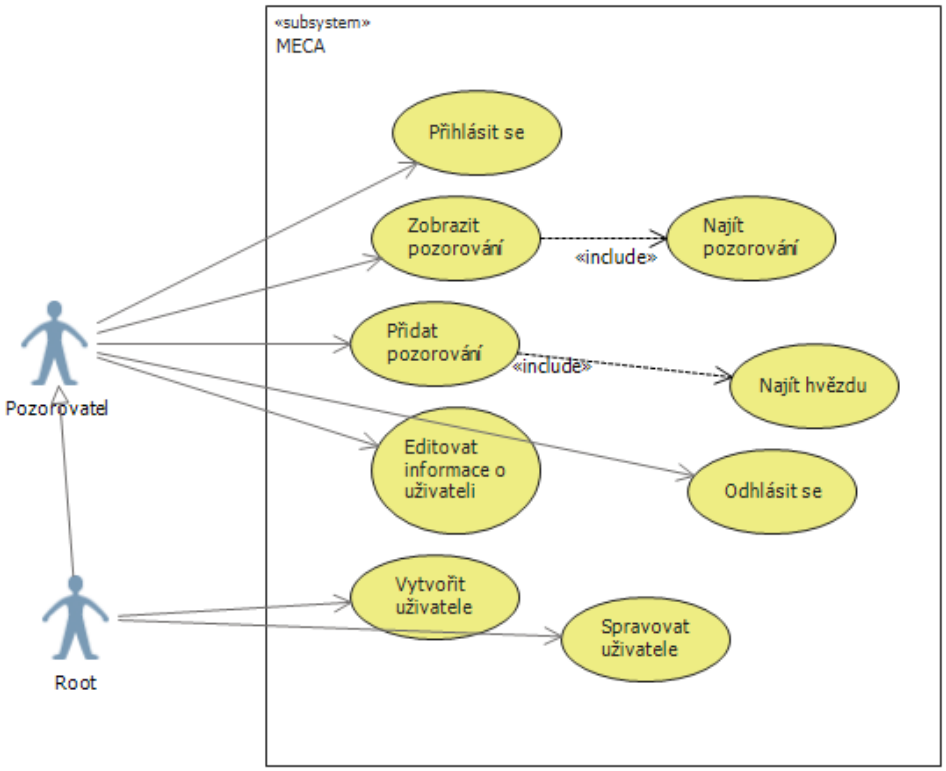
2.1 Analýza aplikace

Aplikace umožňuje přidat pozorování proměnných hvězd a s tím spojená data do databáze a tyto data poté zpětně zobrazit. Při zobrazení pozorování *MECA* vypíše detailní informace o pozorování, zobrazí mapku s polohou hvězd a vykreslí graf svítivosti z přiloženého datového souboru, který byl při vytvoření pozorování nahrán a uložen na server. Další funkcí této aplikace je propojení s databází *GCVS*[3]. Tato databáze obsahuje všeobecné informace, jako je perioda hvězdy a epocha hvězdy, které jsou potřebné pro správné vykreslení grafu. Díku tomuto propojení *MECA* umožňuje automatické vyhledávání a doplňovaní informací o hvězdách.

MECA podporuje dva typy uživatelských účtů. Typ *Root*, který umožňuje přidávat a spravovat uživatele a typ *Pozorovatel*, ten umožňuje přidávat a prohlížet pozorování (viz diagram případů užití na obrázku 2.1). Aplikace je lokalizována do třech jazyků a to češtiny, slovenštiny a angličtiny.

---

1. `http://var2.astro.cz/obslog.php`
2. `http://xmeca.physics.muni.cz`
3. `http://www.sai.msu.su/gcvs/gcvs/`



Obrázek 2.1: Diagram případů užití aplikace MECA

2.1.1 Použité technologie

Jako hlavní programovací jazyk aplikace *MECA* byl použit skriptovací programovací jazyk *PHP*. Aplikace byla také tvořena pomocí *HTML*[4], *CSS*[5] a *LESS*[6]. Autor této aplikace při práci využil i několik knihoven a to:

- *jQuery* - vysvětleno v kapitole 3.4.1
- *Smooth scroll - jQuery* knihovna, která vylepšuje posouvání obsahu stránky
- *Custom UI - jQuery* knihovna, která přidává možnost využít interaktivních prvků jako *DatePicker* (prvek pro výběr data),

4. značkovací jazyk pro tvorbu webových stránek
5. styl elementů v dokumentu v HTML nebo XML
6. rozšíření CSS o dynamické prvky

Dialog (vyskakovací okno), *Draggable* (objekt, který je možné přemístit pomocí myši), *Resizable* (objekt, který je možné zmenšit/zvětšit pomocí myši), *AutoComplete* (automatické doplňování), *Button* (tlačítko) aj.

- *LightBox - jQuery* knihovna, která umožňuje zobrazení obrázku, po kliknutí na jeho náhled

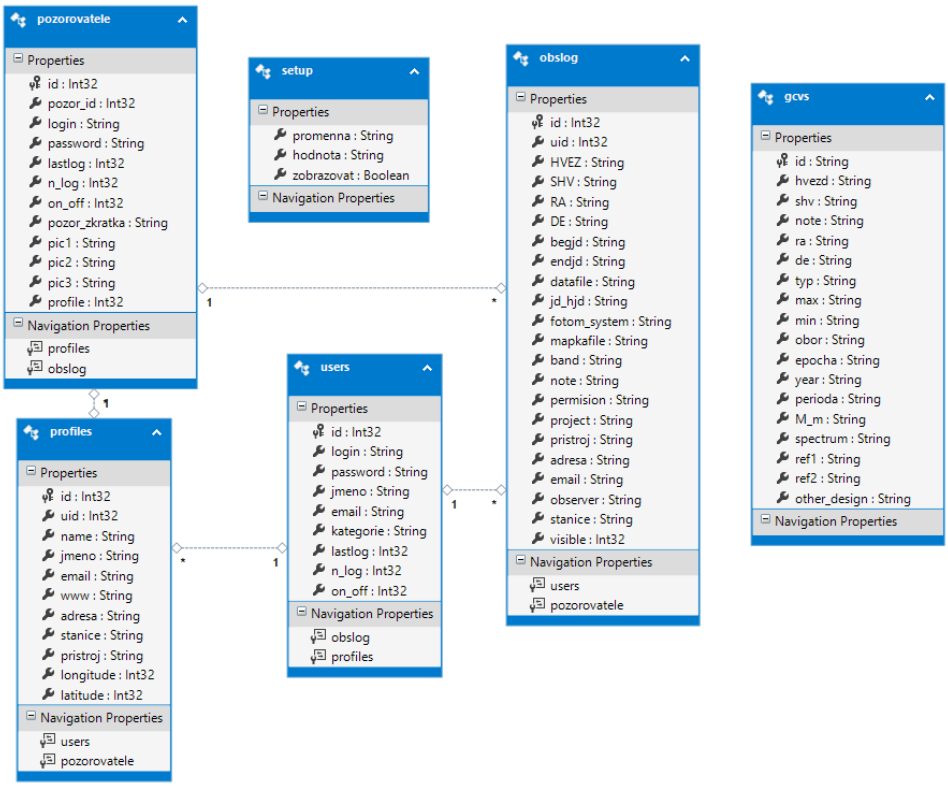
2.1.2 Databáze aplikace MECA

Pro ukládání databázových záznamů byla použita databáze *MySQL*, která obsahuje celkem šest tabulek pro ukládání informací o uživatelích, pozorování a hvězdách (viz obrázek 2.2).

- Tabulka *obslog* slouží k ukládání pozorování. Tato tabulka obsahuje celkem dvaadvacet sloupců. Sloupce *HVEZ*a *SHV* slouží k uložení názvu hvězdy respektive souhvězdí. Do sloupců *RA* a *DE* se ukládá pozice hvězdy. Sloupce *begjd, endjd* a *jd_hjd* slouží k ukládání začátku, konce a času přidání pozorování v juliánském datu. Do sloupců *datafile* a *mapkafile* jsou uloženy adresy souborů. Další sloupce slouží jako doplňující informace k pozorování, jako jsou použité filtry, informace o pozorovateli apod.
- Tabulka *setup* slouží k uložení zkratek z názvu jazyků lokalize.
- Tabulka *gcvs* slouží k uložení záznamů z databáze *GCVS*. Jedná se především o obecné informace o proměnných hvězdách. Nejdůležitější sloupce jsou *perioda* a *epocha*, které jsou využity při tvorbě grafu z datového souboru přiloženého při tvorbě pozorování.
- Tabulka *users* slouží k ukládání informací o uživateli typu *root*.
- Tabulka *profiles* slouží k uložení profilů uživatelů.
- Tabulka *pozorovatele* slouží k uložení informací o uživateli typu *pozorovatel*.

Jelikož jsem neměl možnost osobního kontaktu s autorem této aplikace a jeho kód je dosti nepřehledný a není nikterak okomentovaný, domnívám se, že vztahy mezi jednotlivými tabulkami měli být tak,

jak je na ukázce. Osobně se mi zdá návrh této databáze jako dosti zmatečný. Přijde mi, že některé informace jsou zde ukládány redundantně. Například sloupce *adresa*, *email*, *stanice* v tabulce *obslog* se nachází i v tabulce *profiles*, ze které by bylo možné tyto hodnoty také získat. Zároveň jsou tyto informace ukládány u každého pozorování. Tím dostáváme stejná data v tabulce pro spoustu záznamů. Podobným příkladem jsou i informace o hvězdách, které mohli být získány z tabulky *gsvc*, nebo pro tyto informace mohla být vytvořena tabulka samostatná.



Obrázek 2.2: ERD diagram aplikace MECA

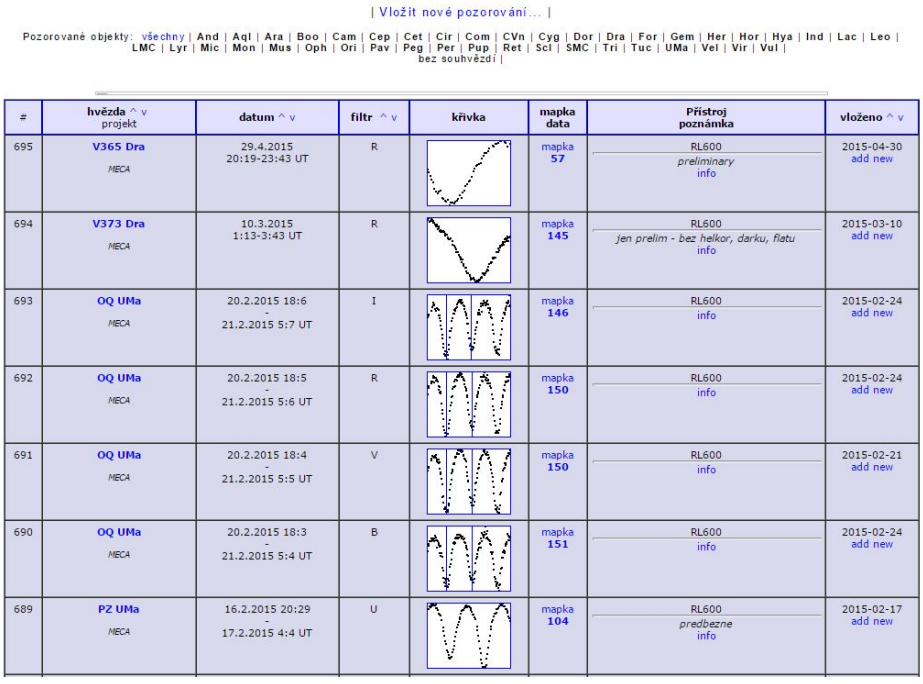
2.1.3 Ukázka uživatelského prostředí

První ukázka z uživatelského prostředí aplikace *MECA* je na obrázku 2.3. Na tomto obrázku se nachází výpis pozorování. Stránka je rozdělena na tři části. V první se nachází hypertextový odkaz na vložení

nového pozorování. Ve druhé se nachází seznam souhvězdí. Po kliknutí na název souhvězdí se vypíší do stejného řádku odkazy s názvy hvězd, které náleží danému souhvězdí.

Po té, co je vybrána některá z hvězd, je aktualizována poslední třetí část. V této části se nachází tabulka s výpisem samotných pozorování pro konkrétní hvězdu. Tato tabulka obsahuje osm sloupců. V prvním se nachází pořadové číslo pro přidání pozorování. Druhý obsahuje odkaz s názvem dané hvězdy, který slouží k zobrazení detailu pozorování. Třetí obsahuje datum pozorování. Čtvrtý ukazuje použitý filtr. V pátém sloupci se nachází miniatura grafu. Šestý sloupec obsahuje odkaz na použitou mapku a její ID. Předposlední sloupec slouží k zobrazení použitého přístroje a poznámky z pozorování. V posledním sloupci je datum, kdy bylo pozorování vloženo do systému, spolu s odkazem na vložení nového pozorování.

Tabulku je možné seřadit vzestupně nebo sestupně podle názvu hvězdy, data pozorování, použitého filtru a data vložení pozorování, po kliknutí na malou šipku v příslušném sloupci v hlavičce pozorování.



Obrázek 2.3: Část stránky s výpisem pozorování

Druhou ukázkou uživatelského prostředí aplikace MECA je obrazovka pro přidání pozorování (viz obr 2.2). Jako první je tu dvojice textových polí pro identifikaci hvězdy. Následuje zaškrtávací tlačítko pro *Příslušnost k projektu*. Toto tlačítko je zde pouze jedno a mělo pravděpodobně sloužit ke sdílení dat mezi uživateli, ovšem v tomto systému nefunguje. Další v pořadí je dvojice tlačítek pro výběr datového souboru a obrázku s mapou. Je zde také několik zaškrtávacích tlačítek pro výběr juliánského data, použitého filtru a fotometrického systému. Na stránce se nachází také textové pole pro poznámku a prvky pro zvolení viditelnosti pozorování. Poslední je tlačítko, které slouží ke zpracování dat a jejich následné kontrole na další obrazovce.

Obrázek 2.4: Část stránky pro přidání pozorování

2.2 Souhrn konečných požadavků

V této podkapitole následuje výpis všech požadovaných funkcí, které má nová aplikace splňovat. Ať už na základě analýzy systému MECA, nebo po konzultaci se zadavatelem.

- Vytvoření souhvězdí
- Vyhledávání informací o hvězdách v systému Sesame
- Vyhledávání informací o hvězdách v databázi VSX
- Vytvoření hvězdy
- Přidání hvězdy k příslušnému souhvězdí
- Přidání pozorování k příslušné hvězdě
- Tvorba grafu ze souboru přiloženého k pozorování
- Podpora uživatelských účtů
- Dva typy uživatelského účtu
- Sdílení dat mezi uživateli

3 Použité technologie při implementaci aplikace

V následující kapitole se věnuji popisu technologií, knihoven a frameworků, které jsem použil pro usnadnění vývoje, efektivitu a větší uživatelský komfort. Obecně zde popisuji technologie jako takové, důvod proč jsem danou technologii zvolil a stručně uvádím příklady použití v této aplikaci.

Jako můj hlavní programovací jazyk jsem zvolil vysokoúrovňový, objektově orientovaný jazyk C#. Tento jazyk jsem si vybral z důvodu, že se jedná o moderní, univerzální programovací jazyk, se kterým mám zkušenosti z mobilní platformy Windows Phone 8. Dalším důvodem, proč jsem tento jazyk zvolil, byla kvalitní dokumentace a množství dostupných knihoven.

Jelikož jsem si zvolil C#, hlavním vývojovým prostředím, které jsem během této práce použil bylo Visual Studio 2013 od firmy Microsoft. K tomuto vývojovému prostředí jsem si pouze doinstaloval ReSharper [2]. Tento doplněk přidává lepší našeptávání, upozorňuje na chyby už v době psaní kódu a celkově udržuje kód čistější.

3.1 Návrhový vzor MVC pro ASP .NET

Bez návrhového vzoru by webová aplikace vypadala tak, že máme v jedné třídě dohromady logiku programu a výstup programu. To ovšem sebou nese spoustu obtíží. Jednak zvýraznění syntaxe pro více programovacích jazyku současně dělá většině IDE problémy. Ale také vzniká nepřehledná stromová struktura, promíchaná databázovými dotazy apod. Takový kód se špatně udržuje. Abychom tomu předešli, můžeme použít některý z návrhových vzorů. Základní myšlenkou MVC bylo oddělení výstupu aplikace od logiky aplikace. MVC rozděluje aplikaci na tři základní komponenty, z kterých vznikl i název tohoto vzoru.

- Model
- View (pohled)
- Controller (řadič)

První komponentou je *Model*. Ten obsahuje logiku aplikace. Mohou se zde nacházet databázové dotazy nebo výpočty. Ve většině případů přímo odpovídá databázovým tabulkám. Hlavním účelem modelu je přijetí parametru a následné vydání dat. Tím jeho funkce končí, to znamená, že se nestará o to, jakým způsobem byl daný parametr získán, nebo jak bylo posléze naloženo s daty na výstupu.

Další komponentou je *View* neboli pohled. Ten se stará o zobrazení výstupních dat uživateli. Jedná se o HTML stránku s minimálním množstvím obslužného kódu. Výjimku tvoří validace vstupu na straně uživatele, procházení dat cykly nebo různé podmínky apod. Pro jeden model můžeme mít více pohledů. To znamená, že můžeme mít například pohled pro editaci, pro přidání další položky, vypsání detailu nebo vypsání všech dat z *Modelu*. Tyto šablony je možné různě kombinovat, vkládat do sebe a používat pro různá data. Tím se vyvarujeme zbytečné duplikaci kódu a výsledný vzhled aplikace se snáze udržuje a edituje. Stejně jako *Model*, ani *Pohled* neví, kde se daná data vzala.

O distribuci dat se stará *Controller* neboli řadič. Jedná se o prostředníka, se kterým komunikuje uživatel, model i pohled. Je to entita, která drží celý systém pohromadě a propojuje *Model* a *View*. Většinou máme pro jeden *Model* právě jeden řadič. Řadič převezme požadavek od uživatele, ten zpracuje, a pokud je potřeba, zavolá model, ze kterého získá potřebná data. Tyto data si uloží a zobrazí *View* a následně jsou sem tato data předána. *View* přijme data a zobrazí uživateli výslednou stránku v HTML.

Tímto způsobem je oddělena logika od výstupu, což usnadňuje další údržbu. Navíc to ulehčuje i vývoj aplikace, protože pro každý problém existuje přesně určená komponenta, se kterou budeme pracovat.

3.2 Entity framework

Entity Framework, dále jen EF, je ORM (Object/Relational Mapping, vysvětleno dále v textu) framework, který umožňuje vývojářům pracovat s relačními daty jako s objekty. Tím odstraňuje potřebu psaní většiny kódu pro manipulaci s daty. To znamená, že pokud vývo-

jář pošle dotaz psaný v LINQ[1], EF vrátí data jako silně typované objekty. Mimo jiné EF poskytuje podporu pro logování změn, rozlišování identit, líné načítání a transformaci dotazů a podobně, takže se vývojáři mohou více soustředit na specifické požadavky své aplikace, spíše než na základy pro přistup k datům.

ORM je nástroj, který automaticky ukládá data z objektů do relační databáze bez velkého programování. Skládá se z tří základních věcí: objektových tříd, relačních databázových objektů a mapování mezi nimi. ORM umožňuje držet odděleně schéma databáze a dizajn tříd. Tím se stává aplikace lépe udržovatelnou a rozšiřitelnou. ORM také automaticky implementuje standartní CRUD operace[2], takže je vývojář nemusí psát ručně.

V EF existují tři základní přístupy, jak pracovat s databází, a to: Database First (databáze nejprve), Model First (model nejprve) nebo Code First (kód nejprve).

- *Database First* se používá, pokud je již hotová databáze. EF designer, který je součástí Visual Studia umí automaticky vytvořit modely pro data, která se nachází v databázi. Jednotlivé modely odpovídají tabulkám v databázi a jejich atributy zase jejich sloupcům. Všechny informace o struktuře databáze, modelu a mapovaní mezi nimi jsou poté uloženy do XML souboru s příponou edmx. EF designer poskytuje grafické uživatelské prostředí, ve kterém je možno posléze tento soubor otevřít a případně upravit tak, aby vztahy mezi entitami odpovídaly požadavkům.
- *Model First* se používá, pokud není ještě hotová databáze a to tak, že se v EF designeru vytvoří model. Jakmile bude model vytvořen, EF automaticky vygeneruje databázi. Stejně jako v Database First, jsou poté jednotlivé vztahy mezi entitami spolu s modelem uloženy do edmx souboru.
- *Code First* se používá v obou případech, tedy když je databáze vytvořena, ale i když není. V tomto přístupu se nepoužívá EF designer, ale vytváří se vlastní třídy a atributy, které budou odpovídat tabulkám a sloupcům v databázi. Pokud už data-

1. `dotazovací jazyk, integrovaný v jazyku C# od verze 3.0`
2. `operace na vytvoření, čtení, aktualizaci a mazání objektů`

báze existuje, EF umí vygenerovat třídy automaticky tak, aby odpovídaly hotové databázi. Vztahy mezi databází a třídami jsou pak spravovány pomocí speciálního API (rozhraní pro programování aplikací). Pokud EF vytvoří databázi a posléze se změní model, podle kterého byla databáze vytvořena, EF automaticky tyto změny promítne do databáze.

Jelikož byla toto moje první zkušenost s EF, zvolil jsem v této práci poslední přístup, tedy *Code First*, protože jsem nepotřeboval využít grafického editoru pro tvorbu mého modelu a *Code First* umožňuje jednoduchou aktualizaci databáze.

3.3 ASP.NET Identity

ASP.NET Identity je framework, který řeší podporu uživatelů a rolí v aplikacích. Je navržen tak, že je použitelný pro webové, mobilní, store (desktopové aplikace pro Windows 8) nebo hybridní aplikace. V Identity je možno vytvořit vlastní přihlašovací systém, který umožňuje jednoduchou správu informací (datum narození, bydliště apod.) o uživatelských účtech. Ovšem můžete použít i přihlašování uživatelů pomocí účtů na sociálních sítí. Další výhodou tohoto frameworku je, že používá *Code First* přístup, známý z EF, takže je možné měnit názvy tabulek nebo datový typ primárního klíče uživatelů.

S nejnovější verzí byl Identity framework rozšířen o možnost použití dvoufázového ověření. To znamená, že uživatel musí dokázat, že je ten, za koho se vydává, pomocí dvou nezávislých ověření. Například pomocí správné kombinace uživatelského jména a hesla a zadáním kódu, který získal z SMS zprávy. Další novinkou je, možnost zablokování uživatelského učtu v případě, že se uživatel pokusí několikrát nesprávně přihlásit. Dále umožňuje ověření uživatelského účtu pomocí emailové adresy, nebo vygenerování nového hesla v případě, že uživatel svoje heslo zapomněl.

Pro potřeby této aplikace jsem využil základní funkcionalitu tohoto frameworku, tedy základní přihlašovací systém, kde se uživatel musí nejprve registrovat. Dále jsem upravil předpřipravený uživatelský model tak, aby obsahoval informace, které si zadavatel vyžádal. Z rozšiřujících funkcí jsem použil pouze resetování zapomenutého hesla.

3.4 Další využité knihovny a frameworky

3.4.1 jQuery

Jedná se o mezi-platformní JavaScriptovou knihovnu, jejíž hlavním účelem je propojení HTML a JavaScriptu. Knihovna jQuery je vytvořena tak, aby ulehčovala práci s DOM[3] prvky, usnadňovala vytváření animací, obsluhu událostí a tvorbu AJAX[4] aplikací. Dále podporuje manipulaci s CSS a její funkčnost lze jednoduše rozšířit pomocí zásuvných modulů. Knihovnu poprvé představil John Resig na BarCamp NYC roku 2006 a od té doby se stala nejpoužívanější knihovnou vůbec. Tento balíček je rovněž součástí ASP. NET MVC Frameworku.

3.4.2 Bootstrap

Bootstrap je HTML, CSS a JavaScriptový framework pro tvorbu interaktivních a mobilních webových aplikací. Tento balíček byl původně vytvořen vývojáři a návrháři společnosti Twitter. Bootstrap se ovšem postupně stal jedním z nejrozšířenějších frameworků vůbec. Mimo jiné je součástí základní šablony pro webové MVC 5 aplikace. Bootstrap obsahuje šablony usnadňující úpravu typografie, tlačítek, navigačního menu a dalších prvků. Dále také podporuje responzivní dizajn, to znamená, že se přizpůsobí šířce displeje, na kterém je daná stránka zobrazována, ať už se jedná o mobilní telefon, nebo o stolní počítač. Mezi pokročilejší funkce Bootstrap frameworku patří například drop-down menu, generování náhledu, stránkování, varovná oznámení, panel průběhu, modální okna, automatické našeptávání při vyplňování formulářů a další.

3.4.3 Image Picker

Jak už název napovídá, Image Picker [4] slouží k přehlednému vybírání ze skupiny obrázků. Jedná se o jednoduché rozšíření jQuery, které upravuje grafický vzhled tagu *select*. Tím se vybírání obrázků

3. `objektový model dokumentu`
4. `webové aplikace skládájící se z XML, HTML, CSS a jazyka JavaScript`

stává více uživatelsky přívětivé. Samotná knihovna umožňuje vybírání v jednoduchém nebo rozšířeném režimu. Jednoduchý režim povoluje vybrat pouze jeden obrázek ze skupiny. Rozšířený režim podporuje vybírání i více obrázků najednou, přičemž jejich maximální počet se dá omezit pomocí kódu. Dále je možné pomocí této knihovny zobrazit popisy obrázků, nebo je rozdělit do skupin. Image Picker také umožnuje dynamicky měnit zobrazovaný obsah.

Jedním z důvodů, proč jsem tuto knihovnu zvolil, je také její propojenost s knihovnou Bootstrap, což dává ve výsledku jednotnou grafiku uživatelského prostředí.

3.4.4 Flot

Flot [3] je JavaScriptová knihovna pro jQuery, která slouží k vykreslování nejrůznějších typů grafů, jako například základní grafy, koláčové grafy, grafy s více osami, ale i grafy měnící se v reálném čase. Knihovna dále podporuje tvorbu grafů s možností interakce. Mezi základní podporované vlastnosti patří přibližování, vybírání výřezu grafu, vybíraní osy, automatické překreslování při změně velikosti okna, popis dat a mnoho dalšího. Flot podporuje všechny prohlížeče s podporou tagu *canvas* z jazyka HTML 5. Veliká všestrannost této knihovny spolu s dobře zdokumentovaným API vedla k tomu, že jsem tento balíček vybral pro tuto aplikaci. Knihovnu jsem využil při vykreslování grafu z dat získaných při pozorování.

4 Popis implementace

V této kapitole se věnuji detailnímu popisu implementace specifických částí pro tento projekt, řeším zde návrh databáze, propojení s externími systémy, zpracování dat a uživatelské prostředí. A také popisuji problémy, se kterými jsem se setkal během své práce a způsob jakým jsem je vyřešil. Dále tato kapitola obsahuje i ukázky zdrojových kódů.

Prvním krokem při tvorbě této aplikace bylo její pojmenování. Rozhodl jsem se aplikaci pojmenovat WAfAOD, což je zkratka z anglického názvu této práce „Web application for astronomical observation diaries."

4.1 Popis databáze

Aplikace WAfAOD obsahuje celkem dvě oddělené databáze. Jedná se o hlavní databázi, která slouží k ukládání uživatelů a ukládání dat k pozorování, a databázi z externího systému, kterou více přiblížím v podkapitole „Propojení aplikace s databází AAVSO."

4.1.1 Hlavní databáze

Na začátku vývoje aplikace WAfAOD jsem musel vhodně navrhnout databázi tak, aby splňovala požadavky kladené zadavatelem a zároveň byla aplikace lehce rozšiřitelná o další funkce. Jelikož jsem využil Entity Framework s Code First přístupem, vytvořil jsem několik modelových tříd, ze kterých vznikla databáze uložená do souboru *usersDB.mdf*. Výsledné schéma této tabulky se nachází na obrázku (obr. 4.1). Tato databáze obsahuje deset tabulek, devět hlavních a jednu k ukládání migrací (změny v modelu promítnuté do databáze).

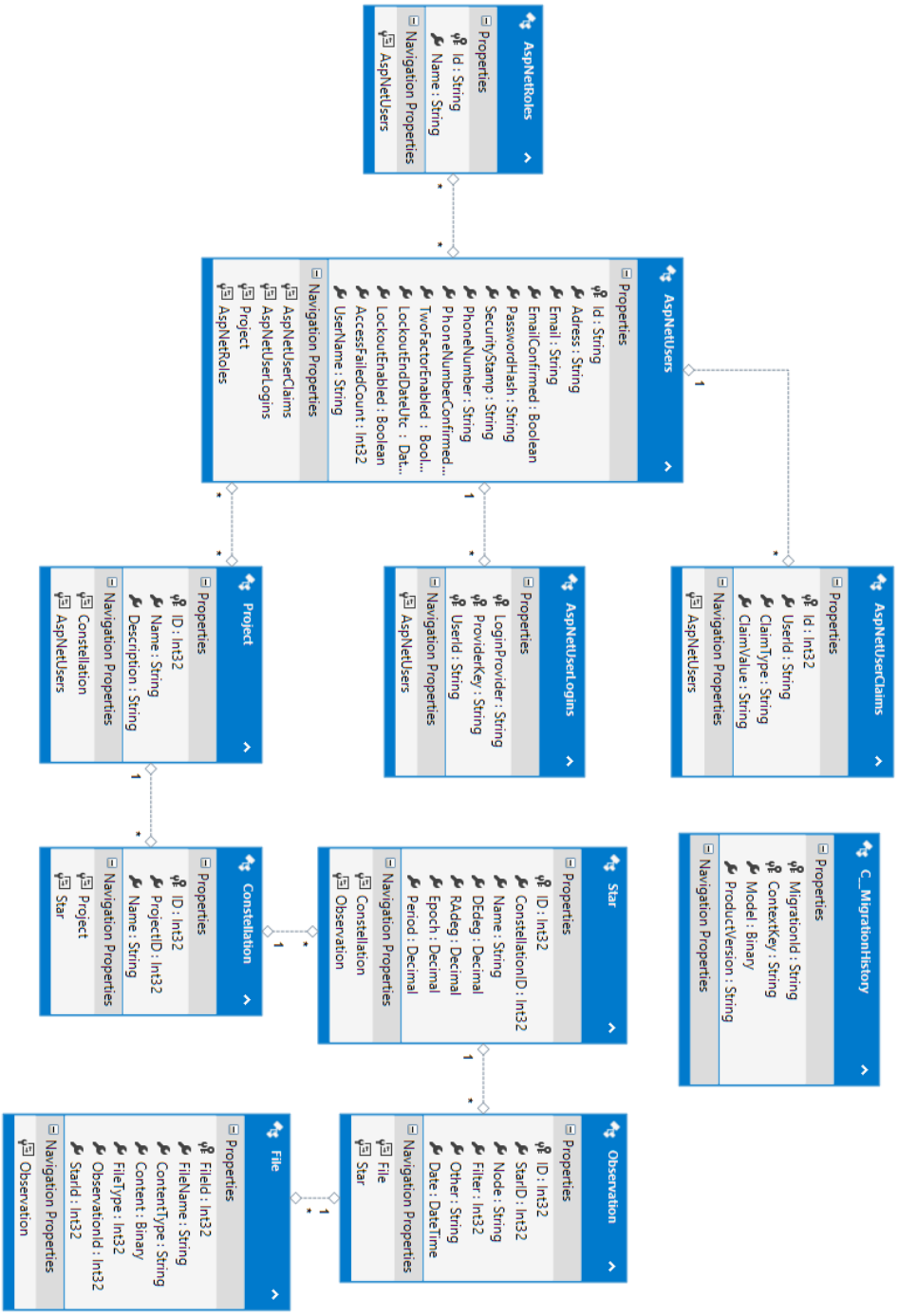
- Tabulka *Project* slouží k ukládání projektů. Tabulka obsahuje tři sloupce. První sloupec je *ID*, který je primárním klíčem tabulky. Druhý sloupec *Name* pro ukládání názvu projektu a poslední sloupec *Description* k uložení popisu projektu.
- Tabulka *Constellation* slouží k ukládání souhvězdí. Tabulka obsahuje tři sloupce. První sloupec je *ID*, který je primárním klí-

čem tabulky. Druhý sloupec obsahuje informaci o cizím klíči do tabulky *Project*. Poslední sloupec *Name* slouží pro ukládání názvu souhvězdí.

- Tabulka *Star* obsahuje informace týkající se jednotlivých hvězd. Tato tabulka obsahuje celkem sedm sloupců a to *ID*, *ConstellationID* což je cizí klíč do tabulky *Constellation*, dále sloupce *DEdeg* a *RAdeg* pro pozici hvězdy, sloupec *Period* a *Epoch* pro ukládání periody, respektive epochy.
- Tabulka *File* slouží k ukládání jednotlivých souborů z pozorování. Účel souboru je uložen ve výčtovém typu. Momentálně existují 2 typy souborů, se kterými tato aplikace pracuje a to s textovým souborem a s obrázky.
- Tabulka *Observation* slouží pro ukládání samotného pozorování. V této tabulce se nachází celkem šest sloupců a to: *ID*, *StarID* jako cizí klíč do tabulky *Star*, *Node* k ukládání poznámky, *Date* pro zapamatování data pořízení pozorování a poslední dva sloupce obsahují informace k použitému filtru.
- Zbývající tabulky slouží k ukládání informací o uživateli. Tyto tabulky byly vygenerovány poté, co jsem využil Identity 2.0. Po přidání základní šablony tohoto frameworku do projektu bylo vygenerováno celkem pět tabulek. Jedinou změnu, kterou jsem provedl na modelu základní šablony, bylo přidání adresy do informací o uživateli.

Při prvním spuštění projektu jsou do tabulky *AspNetUsers* přidány dva typy uživatelských účtu a to typ *Admin* a typ *User*. Zároveň se přidává iniciální uživatel do tabulky *AspNetUsers*, tento uživatel je typu *Admin* a musí být v aplikaci při prvním spuštění aplikace. Zároveň tento uživatel slouží k vytvoření dalších uživatelů typu *Admin*, jelikož normální registrací je novému uživateli automaticky přidána role *User*.

Při inicializaci databáze jsem také musel přepsat metodu OnModelCreating(DbModelBuilder modelBuilder) tak, abych mohl v databázi ukládat a poté z ní vypsat čísla, která mají více desetinných míst. Kdybych tuto metodu neupravil, aplikace by zobrazovala desetinná čísla pouze se dvěma desetinnými místy.



Obrázek 4.1: Entity Relationship Diagram pro databázi uživatelů

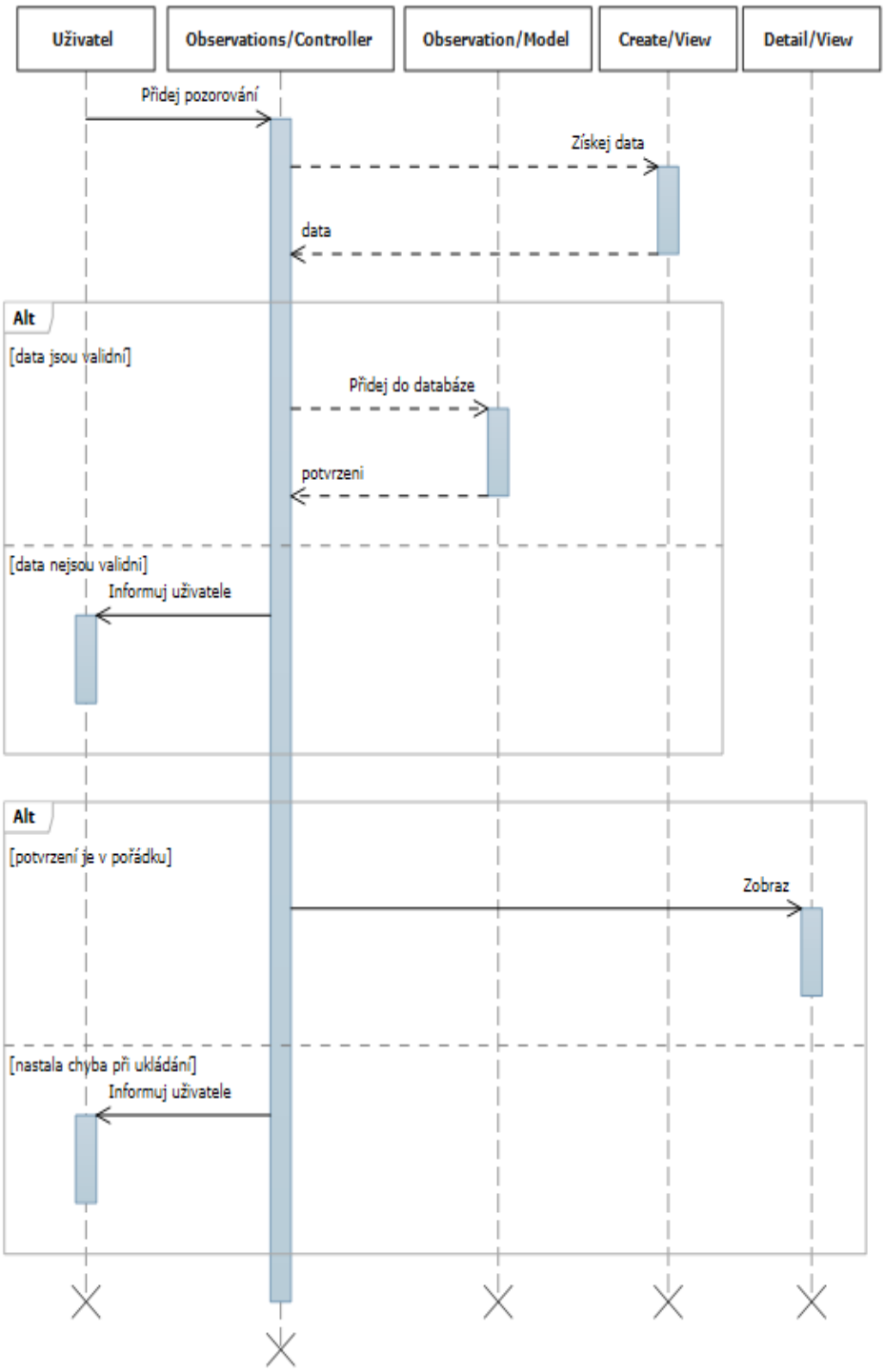
Na obrázku (obr. 4.1) je vidět, že tabulky také obsahují *Navigation Properties*[1](dále *NP*). Nejzajímavější vztahy souvisí s tabulkou *Project*. Tato tabulka propojuje uživatelský systém (tabulka *AspNetUsers*) s daty o pozorování. Za normálních okolností tento vztah není praktický, avšak EF automaticky vytvoří další tabulku *AspNetUsersProject*. Tato tabulka slouží k rozbití *M:N* vztahu na *M:1* a *1:N*. Informace obsažené v *AspNetUsersProject* se upravují nepřímo a to právě pomocí *NP* tabulek *AspNetUsers* a *Project*. Tímto způsobem může mít každý uživatel svoje vlastní data a zároveň svoje data sdílet i s ostatními uživateli. Druhou výhodou tohoto přístupu je ulehčení práce programátora, jelikož objekty, které mají nějakou *NP*, obsahují instanci objektu, na kterou tyto *NP* ukazují, tudíž je programátor nemusí vyhledávat zvlášť.

Ukázka SQL scriptu pro vytvoření Observation vygenerovaného EF

```
CREATE TABLE [dbo].[Observation] (
    [ID]      INT             IDENTITY (1, 1),
    [StarID] INT              NOT NULL,
    [Node]    NVARCHAR (MAX) NULL,
    [Filter] INT              NOT NULL,
    [Other]  NVARCHAR (50)   NULL,
    [Date]    DATETIME        NOT NULL,
    CONSTRAINT [PK_dbo.Observation] PRIMARY KEY
   CLUSTERED ([ID] ASC),
    CONSTRAINT [FK_dbo.Observation_dbo.Star_StarID
   ] FOREIGN KEY ([StarID]) REFERENCES [dbo]. [
   Star] ([ID]) ON DELETE CASCADE
);
```

Na obrázku 4.2 se nachází sequence diagram pro přidání pozorování do aplikace WAfAOD. Jelikož je aplikace postavena na architektonickém vzoru *MVC*, uživatel musí nejprve poslat akci na vytvoření pozorování na *Controller*. Ten poté zkontroluje vypsaná data ve *View* a pokud jsou v pořádku, přidá záznam do databáze a zobrazí uživateli detail přidaného pozorování. Pokud nastane chyba při validaci, nebo při vkládání dat do databáze, bude uživatel informován.

1. poskytuje způsob, jakým se v EF popisují vztahy mezi jednotlivými tabulkami



Obrázek 4.2: Sequence diagram pro přidání pozorování

4.2 Propojení aplikace se systémem Sesame

4.2.1 Aplikace Sesame

Každá proměnná hvězda má obvykle více jmen. Záleží na tom, kdy byla daná hvězda objevena a případně do jakého byla zařazena katalogu. Každý katalog má svoje zvyklosti, podle kterých se hvězda pojmenovává, například v jejím názvu může být název souhvězdí, případně nějaká variace s písmenem řecké abecedy. Z důvodu zjednodušení a sjednocení jmen byl WAfAOD propojen se systémem Sesame.

Sesame je webová aplikace dostupná na odkazu [6]. Je to pojmenovávací služba, která pro řetězec, který představuje název astronomického objektu, vrátí jeho pozici, seznam jmen a několik detailů týkající se dané hvězdy. Přístup k této aplikaci je možný přes webový prohlížeč, HTTP-GET[2] metodu a přes Web Service[3]. V této aplikaci jsem využil druhou možnost. Výsledná data z této aplikace se dají získat upravením URI[4] adresy. Pomocí změny několika parametrů, můžeme dostat data v určitém formátu s daným množstvím informací o daném objektu. Je zde i možnost vybrat si zdroj, ze kterých Sesame čerpá.

Obecné URI pro Sesame s využitím HTTP-GET

```
http://cdsweb.u-strasbg.fr/cgi-bin/nph-sesame/
-opt/~SNVA?object_name
```

Tato metoda umožňuje nastavení několika parametrů. *SNVA* je jeden z těchto parametrů. Ovlivňuje použité databáze, které Sesame prohledává. Písmeno *S* přidává do prohledávání databází Simbad[5], *N* pro NED[6], *V* VizieR[7] a *A* které prohlédá všechny zadané databáze. Bez tohoto parametru se vrací pouze výsledek z databáze s prvním pozitivním výsledkem.

2. dotazovací metoda protokolu HTTP
3. metoda komunikace dvou elektronických zařízení po síti
4. Uniform Resource Identifier - slouží k přesné specifikaci zdroje požadovaného dokumentu
5. http://simbad.u-strasbg.fr/simbad
6. http://ned.ipac.caltech.edu/index.html
7. http://vizier.u-strasbg.fr/vizier/

Dalším parametrem, kterým je možné měnit formát výstupu, je *opt*. Pomocí nějaké z následujících možností (všechny lze libovolně kombinovat):

- *-ox* s tímto parametrem je výsledný dokument ve formátu XML, který je vytvořen podle XSD[8] dokumentu. Další varianty této proměnné jsou *-ox4* a *-ox2*, se kterými bude vrácen XML dokument podle staršího XSD dokumentu a *-oxp*, pro který bude výstup v textové podobě.
- *-oI* vyplní výsledný dokument všemi dostupnými názvy hvězd. Bez tohoto nastavení je vrácen pouze základní název z použité databáze.
- *-oF* přidává do dokumentu magnitudy hvězd. Tato možnost funguje pouze s databází Simbad.

Posledním parametrem, který je třeba změnit je *-object_name*. Místo tohoto parametru se vkládá název hvězdy, který má Sesame vyhledat.

Ukázka konkrétního URI pro Sesame

```
http://cdsweb.u-strasbg.fr/cgi-bin/nph-sesame/
-oIFx/NSV?NGC+4321
```

4.2.2 Použití v aplikaci WAfAOD

Pro použití Sesame v této aplikaci jsem zvolil parametry *SNVA* a *oxI*. Tedy, že výstup bude dokument ve formátu XML formátováný podle XSD dokumentu, který je možné stáhnout na odkazu[9].

Prvním krokem k úspěšnému přečtení výstupních dat bylo vytvoření modelové třídy na základě XSD dokumentu. K tomu jsem využil nástroj XML Schema Definition Tool [5]. Tento nástroj umožňuje z přiloženého XML nebo XSD dokumentu vytvořit modelovou třídu pro několik programovacích jazyků. Mezi těmito jazyky se nachází i C#. Tímto postupem jsem získal třídu *Sesame*, uloženou do souboru *sesame_4x.cs*, kterou jsem přidal do modelů této aplikace.

Dalším krokem bylo získání a načtení XML dokumentu, který poskytuje Sesame. K tomu jsem využil třídu *XmlReader*. Ta poskytuje

8. schéma XML dokumentu
9. http://vizier.u-strasbg.fr/xml/sesame_4x.xsd

přetíženou metodu *Create*, která jako jeden z parametrů bere URI. Poté provede *WebRequest*[10] a stáhne soubor poskytnutý na adrese z parametru a ten uloží jako objekt typu *XmlReader*. Tento objekt jsem pojmenoval *readeras*.

Posledním krokem je deserializace objektu *readeras*. Deserializace je proces, při kterém dochází k převodu dokumentu, určeného pro přenos dat (například XML), do datových struktur. Abych byl tohoto schopen, musel jsem nejprve vytvořit novou instanci třídy *XmlSerializer*, která v konstruktoru bere cílovou datovou strukturu. V tomto případě třídu *Sesame* z prvního kroku. Poté jsem tuto instanci použil na objekt *readeras* z druhého kroku. Tímto vznikla instance třídy Sesame, ve které jsou uloženy všechny dostupné informace z poskytnutého XML dokumentu.

Získání výsledku pro dotaz a jeho následná deserializace

```
XmlReader readeras = XmlReader.Create(
@"http://cdsweb.u-strasbg.fr/cgi-bin/nph-sesame/
-oxI/~SNVA?"+Name);
XmlSerializer ser = new XmlSerializer(typeof(Sesame));
Sesame ses = ser.Deserialize(readeras) as Sesame;
```

4.3 Propojení aplikace s databází VSX

Protože Sesame neposkytuje údaje o epoše a periodě hvězdy, které jsou potřebné pro tuto aplikaci, bylo nutné WAfAOD propojit s další databází. Zadavatelem byla vybrána databáze VSX[11]. Tato databáze obsahuje všechny hvězdy, které mají záznam v AAVSO International Variable Star Index[12], a navíc jsou v ní obsaženy všechny zbývající potřebné informace.

Jelikož jsem pro tuto databázi nenašel žádné vhodné veřejné API takové, abych mohl získávat tyto informace přímo z této databáze, musel jsem stáhnout databázi VSX jako datový soubor, který je volně ke stažení.

10. požadavek na vzdálený zdroj na základě URI
11. http://cdsarc.u-strasbg.fr/viz-bin/Cat?B/vsx
12. http://www.aavso.org/vsx

Po prozkoumání přiloženého bytového rozdělení datového souboru jsem se rozhodl vytvořit novou databázi, kterou jsem připojil k aplikaci WAfAOD. Ta obsahuje pouze jednu tabulku, a to *StarInfo*, která má 4 sloupce: *ID*, *Name*, *Epoch*, *Period*. Tuto databázi jsem uložil do souboru *starInfo.mdf*. Abych naplnil tabulku *StarInfo*, vytvořil jsem jednoduchou konzolovou aplikaci, která načte datový soubor. Tento soubor rozdělí podle dokumentace, vybere pro WAfAOD důležitá data a ta uloží do pomocné třídy. Tato třída má čtyři atributy, které odpovídají sloupcům tabulky *StarInfo*. Z této třídy jsou poté data uloženy do tabulky.

Výběr periody z datového souboru, převedení na vhodný typ a uložení do atributu pomocné třídy

```
ord.Period = float.Parse(line.Substring(140, 15)
    .Trim(),CultureInfo.InvariantCulture);
```

Část příkazu pro vložení záznamu do tabulky StarInfo

```
cmd = new SqlCommand("Insert into StarInfo
    (Name,Period,RAdeg,DEdeg,Epoch) values
    (@name,@period,@radeg,@dedeg,@epoch)", sc);
cmd.Parameters.AddWithValue("@name", ord.Name);
```

4.4 Zpracování dat z pozorování

Při každém přidání nového pozorování je potřeba přiložit textový soubor, který obsahuje data z přístroje, kterým bylo pozorování uskutečněno. Textový soubor může obsahovat 2 řádky textu s informacemi o použitém přístroji, použitých filtrech apod. Jelikož tyto informace ale neposkytuje každý přístroj, jsou tyto 2 řádky, pokud se nachází v souboru, ignorovány a uživatel musí tyto informace nastavit ručně. Další částí tohoto souboru jsou samotná data. Ta jsou rozdělena minimálně do třech sloupců oddělených mezerou. Jedná se o reálná čísla reprezentující: juliánské datum, magnitudu a chybu při měření. Poslední dva sloupce se v dokumentu mohou nacházet vícekrát, avšak pro potřeby této aplikace je možné je ignorovat. Z těchto dat poté systém WAfAOD vykreslí příslušný graf.

4.4.1 Uložení souboru do databáze

Pokud má aplikace, která stojí na architektuře MVC, podporovat nahrávání a stahování souboru, musí se ve *View* využít přetížená metoda *Html.BeginForm*. Konkrétně se musí nastavit atribut *enctype* na hodnotu *multipart/form-data*. V aplikaci WAfAOD jsem nastavil *View Create* pro *Controller Observation* následujícím způsobem.

Využití přetížené metody Html.BeginForm

```
@using (Html.BeginForm("Create", "Observations",
    null, FormMethod.Post, new { enctype =
    "multipart/form-data" }))
```

Přidání tlačítka na vybrání souboru do View

```
<input type="file" id="Data" class="btn btn-default"/>
```

Jakmile je nastaven atribut *enctype*, je možné ve *View* využít tlačítka pro výběr souboru. Pokud uživatel přes toto tlačítko vybere soubor a odešle data v příslušném *View*, *Controller* může daný soubor načíst a uložit do databáze v binární podobě. Soubor je předán *Controlleru* jako *HttpPostedFileBase*[13].

Nahraní textového souboru do databáze v Controlleru

```
 var dataCon = new File {
  FileName =
   System.IO.Path.GetFileName(data.FileName),
  FileType = FileType.Data,
  ContentType = data.ContentType,
  StarId =  col["ddlstar"].AsInt()
};
 using (var reader = new
   System.IO.BinaryReader(data.InputStream)){
  dataCon.Content =
   reader.ReadBytes(data.ContentLength);
  }
 observation.Files.Add(dataCon);
```

13. třída, která poskytuje přístup k souborům odeslaných klientem

4.4.2 Převod juliánského data

Základní časová jednotka, která se používá v astronomii, je juliánské datum. Toto datum udává počet dní, které uplynuly od poledne 1. 1. 4713 před naším letopočtem. Zapisuje se jako desetinné číslo, kde desetinná část odpovídá příslušné části dne.

Pokud je přiložen soubor s daty z pozorování, WAfAOD automaticky přepočítá juliánské datum na gregoriánské, které se běžně používá v naší společnosti. Hodnotu bere z prvního záznamu v souboru. Výsledné gregoriánské datum je poté uloženo s pozorováním, pro usnadnění orientace v pozorování.

K převodu jsem použil následující metodu. Od juliánského data jsem odečetl počet dní, které uběhly od počátku unixového času[14] tj. 2440587,5. Výslednou hodnotu jsem poté vynásobil číslem 86400 (počet vteřin v jednom dni) a tuto hodnotu jsem poté přičetl k proměnné typu *DateTime*, kterou jsem předtím nastavil na počátek unixového času. Tímto způsobem WAfAOD získá datum v gregoriánském datu.

Převod juliánského data na gregoriánské

```
double unixTime = (julianDate - 2440587.5) * 86400;
DateTime dtDateTime = new DateTime(1970,1,1,0,0,0,0,
   DateTimeKind.Utc);
dtDateTime =
   dtDateTime.AddSeconds(unixTime).ToLocalTime();
```

4.4.3 Tvorba grafu pomocí knihony FLOT

Poslední věcí, na kterou je možné využit přiložený soubor, je tvorba grafu s periodou proměnné hvězdy. Tento graf má osu x v juliánském datu, osa y představuje hodnotu magnitudy. K vykreslení tohoto grafu jsem využil knihovnu *FLOT*. Tato knihovna podporuje mnoho typů grafů. Z nich jsem nakonec zvolil graf, označený jako *Navigation* (viz obr. 4.3).

Tento graf umožňuje přibližování nebo oddalování dat pomocí kolečka myši, nebo dvojklikem. Dále je možné celý graf libovolně posouvat tažením myši. Výhodou tohoto grafu také je, že se automaticky přepočítává měřítko os. Jako jednu z hlavních nevýhod, se

14. počet vteřin od (UTC) 00:00:00 1. 1. 1970

kterou jsem se setkal, při testování tohoto typu grafu bylo překrývání popisu dat na ose *x* z důvodu velkého počtu cifer juliánského data. Tento problém jsem nakonec vyřešil natočením těchto popisů o 25%, pomocí úpravy kaskádových stylů tak, aby byl výsledný graf přehledně zobrazen.

Data jsou do tohoto grafu předávána pomocí ajaxové metody. Tato metoda se automaticky spouští při načítání *Detail View* pro pozorování. Poté, co je spuštěna, zavolá metodu *GetPoints(int ID)* z *Observation Controlleru* a předá jí *ID* právě zobrazovaného pozorování. *Controller* vyhledá v databázi datový soubor, který náleží danému pozorování. Tyto data se musí nejprve převést z binární podoby na textovou. K tomu jsem využil třídu *Enconding*. Jakmile jsou data převedena, přeskočí se první dva řádky textu, ale pouze v případě, že obsahují informace o filtrech apod. Posledním krokem je rozdělení souboru na sloupce a vybrání prvních dvou, které jsou předány jako *JSON*[15] objekt zpátky do *View*, kde jsou tyto data posléze použity knihovnou FLOT, která vykreslí graf.

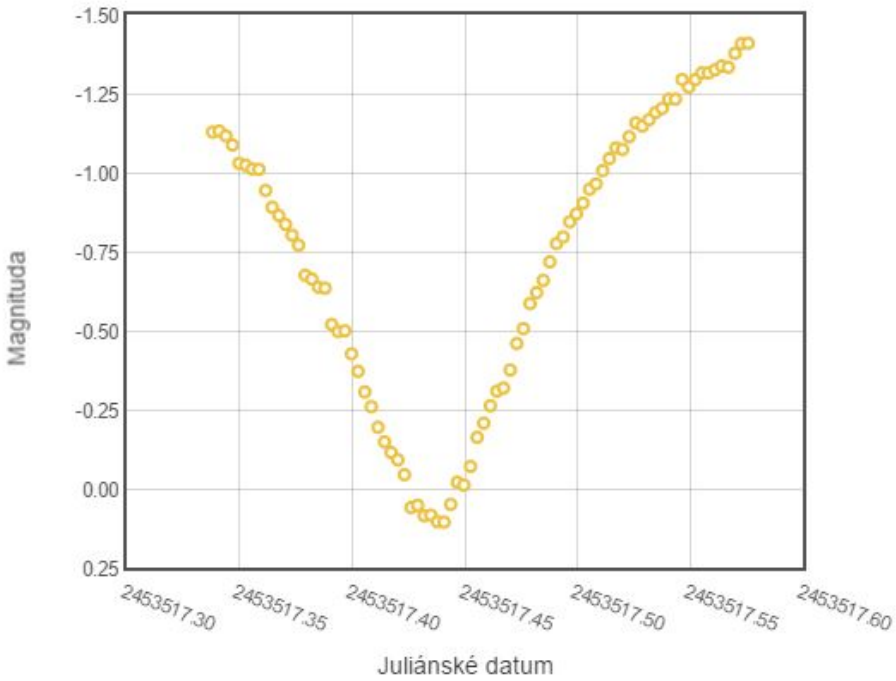
4.5 Popis implementace uživatelského prostředí

Při návrhu grafického uživatelského prostředí jsem se snažil držet jednoduchého stylu, který je zároveň intuitivní na ovládání. Za tímto účelem jsem zvolil standartní styl tlačítek a jiných prvků, které poskytuje knihovna Bootstrap. Tohoto jsem docílil nastavením atributu *class*. Abych získal základní šedé tlačítko se zaoblenými hranami, musel jsem tento atribut nastavit na hodnotu *btn*.

Pro použití některé z funkcí aplikace WAfAOD se musí uživatel nejprve přihlásit. Nepřihlášený uživatel se dostane pouze na úvodní obrazovku s krátkým popisem aplikace a s možností přihlášení, případně registrace. Aplikace je momentálně lokalizována pouze do českého jazyka.

Na obrázku 4.4 se nachází obrazovka pro přidávání pozorování. Jelikož každé pozorování musí být přiřazeno právě k jedné hvězdě, jsou zde umístěny dva *DropDownList*[16] prvky pro výběr souhvězdí

15. způsob zápisu dat nezávislý na počítačové platformě
16. ovládací prvek, který umožňuje uživateli vybrat jednu položku z rozevíracího seznamu



Obrázek 4.3: Ukázka grafu z aplikace WAfAOD

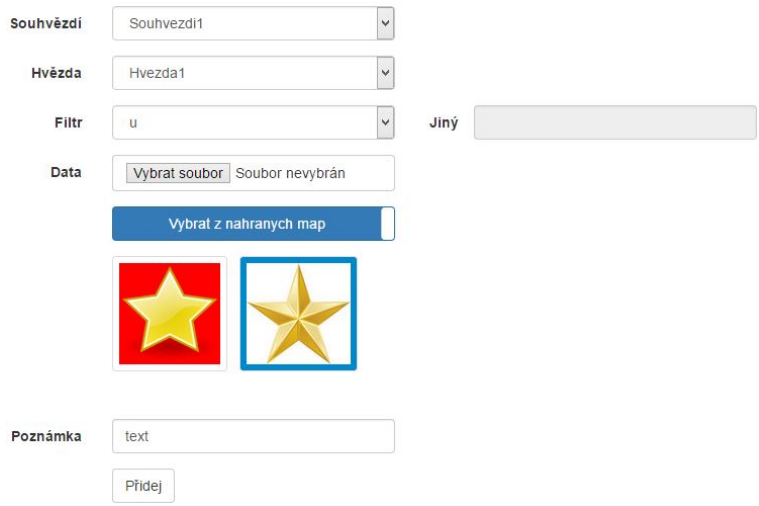
a hvězdy. *DropDownList* pro hvězdu se dynamicky mění podle toho, jaké souhvězdí je momentálně vybrané, což ulehčuje vyhledávání. Pokud není vybrané žádné souhvězdí, je tento prvek neaktivní. Další částí této obrazovky je *DropDownList* pro filtr, který obsahuje název devíti nejpoužívanějších filtrů v astronomii. Avšak pokud byl při pozorování použit jiný filtr, stačí vybrat položku *Jiný*. Poté se aktivuje textové pole, do kterého je možné uložit nový filtr. Pod filtrem následuje tlačítko na vybrání souboru. Pokud je soubor vybrán, je u něho zobrazen jeho název.

Jako jeden z požadavků zadavatele bylo, aby bylo možné vybrat mapu (obrázek, který je přiložen při vytvoření pozorování) buď z lokálního disku, nebo z dříve nahraných pozorování pro danou hvězdu. Z tohoto důvodu jsem se rozhodl přidat *toggle*[17] tlačítko, které dynamicky mění vzhled této obrazovky. Jeho přepnutím se zobrazí tlačítko pro výběr souboru, který chceme nahrát, nebo Image

17. `ovládací prvek, který může měnit stav`

Picker, jak je možné vidět na této ukázce. Ten zobrazuje náhled map, které jsou přiložené k hvězdě, a umožňuje jednu z nich vybrat. Kolem vybrané hvězdy se zobrazí modrý rámeček.

Poslední textové pole na této obrazovce slouží k uložení případné poznámky. Za tímto polem následuje tlačítko, které následně vytvoří nové pozorování a to uloží do databáze.

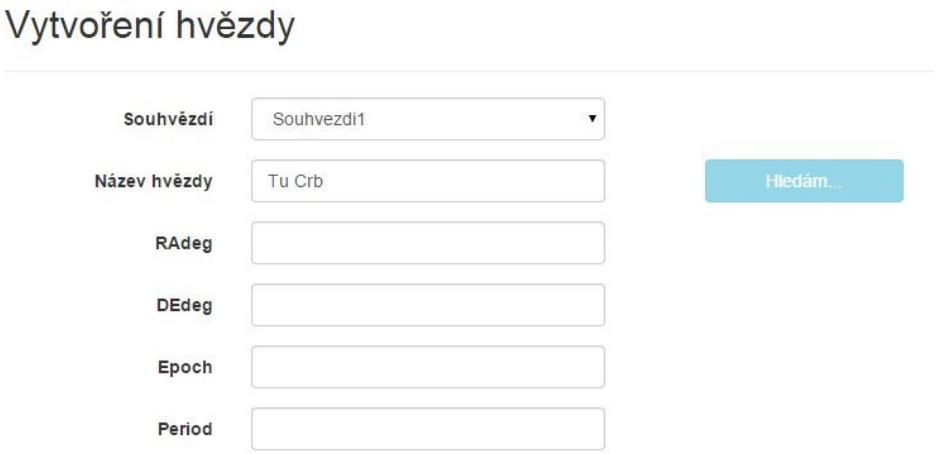


Obrázek 4.4: Část stránky pro přidání pozorování

Další ukázkou uživatelského prostředí je obrázek 4.5. Ten představuje část obrazovky pro přidání hvězd do aplikace WAfAOD. Prvním prvkem je znovu *DropDownList*, který obsahuje doposud přidané souhvězdí, ze kterých je třeba jedno vybrat.

Následuje textové pole pro název hvězdy. Pokud je toto pole vyplněné, je možné nechat aplikaci WAfAOD vyhledat zbývající informace o dané hvězdě pomocí tlačítka *Vyhledat* umístěného vedle tohoto pole. Po zmáčknutí tohoto tlačítka WAfAOD vyhledává v systému *Sesame* alternativní názvy hvězdy. Ty pak použije na vyhledání v databázi *VSX* pro získání dalších údajů. Celý proces může zabrat několik vteřin. Z tohoto důvodu se popis tohoto tlačítka mění na *Hledám*, pro upozornění uživatele, že je WAfAOD aktivní. Po vyhledání příslušných dat je popisek změněn zpátky na *Vyhledat* a jsou automaticky doplněna zbývající textová pole této stránky. Tyto pole je možné vyplnit nebo upravit jejich hodnoty ručně i poté, co je WAfAOD automaticky doplnil.

Při návrhu této obrazovky jsem se setkal se dvěma problémy s validací dat na straně uživatele. První byl, že při použití tlačítka *Vyhledat*, se automaticky kontrolovalo vyplnění všech polí tak, jak to požaduje model. Ovšem v tomto případě byla validace nechtěná. Abych tomuto předešel, musel jsem nastavit tlačítko *Vyhledat* jako *cancel* tlačítko. To jsem udělal nastavením atributu *class* na *cancel*. Po této akci se validace u tlačítka *Vyhledat* už neprovádí. Druhým problémem byla validace reálných čísel. Konkrétně se jednalo o problém, kdy aplikace odmítala přijmout reálné číslo s desetinou čárkou a požadovala desetinou tečku. Toto jsem vyřešil přidáním lokalizačního pluginu a přetížením *DecimalModelBinder* třídy tak, aby akceptovala desetinou čárku i tečku.



Obrázek 4.5: Část stránky pro přidání hvězdy

4.6 Aplikace WAfAOD

Výsledkem mé práce je aplikace WAfAOD, která slouží jako astronomický deník k ukládání informací o pozorování. Aplikace podporuje dva typy uživatelských účtů a to *Admin* a *User*. Typ *User* slouží k přidávání vlastních dat a jejich prohlížení. Typ *Admin* navíc umožňuje spravovat uživatele. Tímto byla splněna část požadavků z kapitoly 2.2, konkrétně:

- Podpora uživatelských účtů

- Dva typy uživatelského účtu

Po přihlášení do aplikace je umožněno uživateli vytvářet *Projekt*. Uživatel může být členem několika projektů. Do projektu se mimo jiné ukládá i seznam členů projektu. Další členy je možné přidat pomocí jejich uživatelského jména, případně emailu. Přidávat nové uživatele do projektu může pouze člen projektu. Z projektu může být odstraněn člen pouze poté, co ho sám opustí. Pokud projekt opustí i poslední člen, tak se automaticky maže. Tímto jsem splnil požadavek zadavatele na umožnění sdílení dat mezi uživateli.

Pokud si uživatel vytvořil *Projekt*, musí si do tohoto projektu před přidáním *Pozorování* přidat alespoň jedno *Souhvězdí* a *Hvězdu*. Při vytváření *Souhvězdí* je třeba zadat jeho název a také, do jakého *Projektu* náleží. Při vytváření *Hvězdy* je potřeba vybrat *Projekt* i *Souhvězdí*. Aplikace také umožňuje při tvorbě *Hvězdy* vyhledat informace o dané hvězdě. Tyto informace jsou vyhledávány v systému *Sesame* a v databázi *VSX*. Tímto byla splněna další část požadavků na aplikaci WAfAOD, konkrétně:

- Vytvoření souhvězdí
- Vyhledávání informací o hvězdách v systému Sesame
- Vyhledávání informací o hvězdách v databázi VSX
- Vytvoření hvězdy
- Přidání hvězdy k příslušnému souhvězdí

Jakmile má uživatel vytvořený *Projekt*, v něm *Souhvězdí* a *Hvězdu*, může přidat *Pozorování*. K přidání *Pozorování* je nutné vybrat k tomu náležící *Hvězdu*. Další věcí potřebnou k úspěšnému přidání *Pozorování* je přiložení datového souboru s vlastními daty z pozorování a vybrání mapy hvězd z dosud nahraných *Pozorování* k dané *Hvězdě*, nebo nahrání nové. Po úspěšném uložení pozorování, je možné zobrazit jeho detail, který mimo jiné obsahuje i interaktivní graf vytvořený na základě datového dokumentu. Tímto jsem splnil poslední dva požadavky na aplikaci WAfAOD, konkétně:

- Přidání pozorování k příslušné hvězdě
- Tvorba grafu ze souboru přiloženého k pozorování

5 Závěr

Cílem této bakalářské práce bylo analyzovat aplikaci *MECA*, kterou používá Ústav teoretické fyziky a astrofyziky na Přírodovědecké fakultě Masarykovy univerzity pro ukládání dat z pozorování proměnných hvězd. Určit její funkce, rozšířit je o další podle požadavků zadavatele a vytvořit novou aplikaci na základě této analýzy.

Úvodní kapitola popisuje aplikaci *MECA*, která slouží k ukládání a následné prezentaci dat z pozorování proměnných hvězd. V této kapitole jsou UML diagramy popisující tuto aplikaci. Kapitola také obsahuje použité technologie v této aplikaci a ukázku jejího uživatelského prostředí i s popisem.

V další kapitole se věnuji popisu technologií, které jsem použil při implementaci nové aplikace. Popisuji zde použité frameworky a knihovny, stejně jako důvod proč a jaký jsem použil hlavní programovací jazyk a vývojové prostředí.

Čtvrtá kapitola obsahuje popis implementace nové aplikace. Zabývám se zde specifickými částmi pro tuto aplikaci, mezi kterými se nachází popis databáze využívaný novou aplikací. Popisuji také její propojení s externí aplikací Sesame a s databází VSX. Dále uvádím způsob jakým ukládám datové soubory v aplikaci a jejich následné zpracování včetně tvorby grafu. Také se zde věnuji převodu juliánského data na gregoriánské. Na závěr této kapitoly jsem přidal ukázku uživatelského prostředí.

V této době provádím na aplikaci poslední úpravy, avšak aplikace jako taková je již plně funkční. Aplikace není momentálně nasazena na žádném serveru. S tímto krokem čekám na vyjádření zadavatele, jelikož zároveň s mojí prací vzniká na toto téma i práce mého kolegy. V budoucnosti bych rád přidal lokalizaci alespoň do třech světových jazyků a přidal možnost zasílání zpráv mezi uživateli.

Literatura

[1] ReSharper :: The Most Intelligent Extension for Visual Studio. *JetBrains s.r.o.* [online]. © 2015 [cit. 2015-04-25]. Dostupné z: `https://www.jetbrains.com/resharper/`.

[2] MECA - Measurements of Eclipsing Binaries - Archive. *Web Services* [online]. © 2015 [cit. 2015-05-12]. Dostupné z: `http://xmeca.physics.muni.cz`.

[3] Flot. *Ole Laursen* [online]. © 2015 [cit. 2015-04-25]. Dostupné z: `http://www.flotcharts.org/`.

[4] Image Picker. *Rodrigo Vera* [online]. © 2015 [cit. 2015-04-25]. Dostupné z: `http://rvera.github.io/image-picker/`.

[5] XML Schema Definition Tool (Xsd.exe). *Microsoft* [online]. © 2015 [cit. 2015-04-25]. Dostupné z: `https://msdn.microsoft.com/en-us/library/x6c1kb0s%28v=vs.110%29.aspx`.

[6] Sesame. *UDS/CNRS* [online]. © 2015 [cit. 2015-04-25]. Dostupné z: `http://cdsweb.u-strasbg.fr/doc/sesame.htx`.