Masarykova univerzita
Fakulta informatiky

Nástroj pre vyhľadávanie fotometrických dát stelárnych objektov

Bakalárska práca

Michal Krajčovič

Brno, jar 2016

Prehlásenie

Prehlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Michal Krajčovič

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

Poďakovanie

Ďakujem RNDr. Martinovi Kubovi, Ph.D. za vedenie mojej práce a užitočné rady pri písaní textovej časti práce. Taktiež chcem poďakovať môjmu konzultantovi doc. RNDr. Miloslavovi Zejdovi, Ph.D za odbornú pomoc.

Zhrnutie

Cieľom práce bolo vytvoriť nástroj pre uľahčenie vyhľadávania fotometrických dát. Na implementáciu je použitý jazyk Java a platforma JavaFX na vytvorenie desktopovej aplikácie. Práca obsahuje analýzu, návrh riešenia, implementáciu a návody na používanie.

Kľúčové slová

Java, JavaFX, Spring, fotometria, premenné hviezdy

Obsah

1 Úvod

Premenné hviezdy sú objekty, ktorých jasnosť sa mení v čase. Je to pomerne častý jav, vďaka ktorému sa môžeme o objekte dozvedieť mnoho informácií. [1, kap. 1]

Nájde sa mnoho hviezd, ktorých zmenu jasnosti je možné pozorovať voľným okom, avšak v počiatkoch astronómie sa im neprikladal veľký význam. [1, kap. 2]

Prvý prelom pozorovania premenných hviezd nastal v 16. storočí, keď astronóm Tycho Brahe objavil „novú“ hviezdu v súhvezdí Kasiopeja. Na základe jej meniacej sa jasnosti získal prvý časový rad premennej hviezdy, z ktorej bola vytvorená prvá svetelná krivka. Napriek tomu sa so systematickým výskumom týchto hviezd začalo až koncom 18. storočia. [1, kap. 2]

V dnešnej dobe je zaznamenaných už cez 394 tisíc premenných hviezd [2]. Namerané fotometrické dáta týchto objektov sú prístupné vo viacerých on-line databázach. V prípade záujmu astronóma o dáta zo všetkých dostupných pozorovaní, musí objekt ručne vyhľadať v katalógu VSX[1], z ktorého vedú odkazy do iných databáz. VSX však nie vždy poskytuje obsahy pre všetky existujúce databázy, a vtedy je potrebné chýbajúce databázy ručne dohľadať. Každá databáza poskytuje fotometrické dáta v inom formáte, ktoré potom astronóm musí manuálne previesť na jednotný formát. Tento postup je veľmi pracný a časovo náročný.

Cieľom tejto bakalárskej práce je vytvorenie nástroja, ktorý uľahčí astronómom prácu pri získavaní fotometrických dát. Nástroj má automaticky vyhľadať hviezdu vo všetkých dostupných databázach a poskytnúť užívateľovi fotometrické dáta v jednotnom formáte. Nástroj musí počítať s možnosťou pridania novej databázy, a preto je jeho súčasťou aj systém rozšírení implementujúcich prístup do jednotlivých databáz.

Text práce je rozdelený na osem kapitol. V druhej kapitole sa venujem analýze požiadaviek od zadávateľa práce. Obsahuje ich zoznam a podrobnejší rozbor.

Tretia kapitola popisuje návrh riešenia daného problému.

---

1. Katalóg premenných hviezd. Dostupný na: `https://www.aavso.org/vsx/`

Štvrtá kapitola približuje postup pri implementácii, rozdelenie projektu a popis implementácií jednotlivých častí. Obsahuje taktiež zoznam použitých technológií.

V piatej kapitole navrhujem možné rozšírenia do budúcnosti.

Šiesta a siedma kapitola obsahujú podrobný návod, ako aplikáciu používať a rozširovať o ďalšie databázy.

Posledná kapitola zhrňuje výsledky práce.

2 Analýza požiadaviek

Vytvorený program má slúžiť ako praktická pomôcka pre astronómov pri získavaní fotometrických dát stelárnych objektov. Hlavnou úlohou programu je vyhľadať zadaný stelárny objekt v každej z dostupných databáz, získať z nich fotometrické dáta a konvertovať ich do jednotného formátu.

Určenie požiadaviek a ich analýza je dôležitá pre dosiahnutie spokojnosti budúcich užívateľov. V tejto kapitole popisujem požiadavky zo zadania, ktoré boli podrobnejšie vysvetlené na konzultáciach s doc. RNDr. Miloslavom Zejdom, Ph.D., konzultantom bakalárskej práce.

2.1 Počiatočný stav

V súčasnej dobe neexistuje nástroj, ktorý získa fotometrické dáta automaticky. Astronómovia si môžu vyhľadať objekt v databáze VSX, z ktorej vedú odkazy na ostatné databázy, ale ďalšie spracovanie musia robiť manuálne.

2.2 Funkčné požiadavky

2.2.1 Vyhľadávanie

Stelárny objekt bude možné vyhľadať dvomi spôsobmi, buď pomocou jeho názvu, alebo pomocou súradníc a odchýlky, v ktorej sa ešte môže objekt nachádzať. Zadávanie súradníc bude fungovať v jednotkách stupňov alebo hodín, pričom každý spôsob sa môže zapisovať viacerými formátmi podľa štandardov základnej astronomickej databázy SIMBAD[1].

2.2.2 Jednotný formát dát

Každá z databáz pre fotometrické dáta si ich ukladá v rôznom formáte. Je potrebné všetky previesť na jednotný textový formát. Ten pozostáva z troch stĺpcov, ktoré sú oddelené medzerou: juliánsky dátum, hviezdna magnitúda a chyba merania.

1. `http://simbad.u-strasbg.fr/simbad/`

2.2.3 Exportovanie výsledkov

Získané dáta sa budú dať exportovať z aplikácie podľa každej databázy zvlášť v samostatnom súbore. Dostupná bude aj správa o výsledkoch vyhľadávania a pôvodné súbory najdené v jednotlivých databázach.

2.3 Nefunkčné požiadavky

Na implementáciu tohoto nástroja neboli zadaním kladené iné nefunkčné požiadavky než jednoduché rozšírenie aplikácie o nové databázy. Toto rozšírenie malo byť vykonané pomocou systému rozšírení, pričom jednotlivé rozšírenia mali byť implementované v ľubovoľnom programovacom jazyku.

Pri voľbe ostatných technológií som mal voľnú ruku, ale bolo potrebné dbať na to, aby výber nevytváral prekážky pre užívateľov v budúcnosti.

Počas konzultácie s vedúcim bakalárskej práce mi bolo odporučené vytvoriť desktopovú aplikáciu v programovacom jazyku Java, kvôli podpore všetkých bežne používaných operačných systémov.

3 Návrh riešenia

V tejto kapitole sa budem venovať môjmu návrhu, ako by mala aplikácia fungovať.

3.1 Návrh štruktúry projektu

Projekt bude rozdelený do viacerých menších modulov spravovaných nástrojom Maven. Jeden modul bude obsahovať logiku pre desktopovú aplikáciu, čiže jednotlivé FXML súbory s pohľadmi, modely a ovládače pre interakciu s užívateľom. Druhý modul bude slúžiť ako API[1] s triedami pre získavanie dát o stelárnych objektoch, prístup k súborom a správu rozšírení.

3.2 Návrh systému rozšírení

Aplikácia má byť v budúcnosti rozšíriteľná o nové databázy, preto je potrebné vytvoriť systém rozšírení. Každé rozšírenie bude slúžiť na získavanie dát z niektorej z dostupných databáz fotometrických dát stelárnych objektov. Keďže rozšírenia musia byť podľa požiadavky implementovateľné v ktoromkoľvek programovacom jazyku, bude potrebné, aby každé rozšírenie obsahovalo spustiteľný súbor. Rozšírenia budú rozdelené po zložkách. Každá zložka bude obsahovať minimálne samotný spustiteľný súbor a konfiguračný súbor s názvom spustiteľného súboru a s príkazom, ktorým ho je možné spustiť.

Pri takomto prístupe nebude možné rozšírenia integrovať priamo do hlavnej aplikácie, čo nie je ideálne riešenie, ale umožní to podporu akéhokoľvek programovacieho jazyka. Výmenu dát medzi rozšírením a hlavnou aplikáciou je možné vyriešiť tak, že rozšírenie bude získané dáta vypisovať na štandardný výstup v jednotnom formáte. Hlavná aplikácia ich odtiaľ môže čítať a ďalej spracovávať.

Ďalším riešením by bolo obmedziť počet jazykov na tie, ktoré sa dajú integrovať do jazyka Java. Príkladom je samotný jazyk Java alebo jazyk Python pomocou knižnice Jython [3].

---

1. Application Program Interface (rozhranie pre programovanie aplikácií)

3.3 Návrh aplikačnej logiky

Aplikačná logika aplikácie musí mať funkcie na vyhľadanie dodatočných informácií o hľadanom objekte. Tieto informácie sa dajú získať pomocou webovej služby Sesame[2]. Sesame poskytuje jednoduché REST[3] API a slúži na získanie aliasov a súradníc objektu z rôznych databáz.

Ďalšia využiteľná služba je Vizier[4]. Obsahuje veľké množstvo katalógov hviezd, medzi ktorými je aj VSX. Z katalógu VSX sa dajú získať informácie, ako názov, súradnice, epocha a perióda stelárneho objektu. Služba neposkytuje žiadne jednoduché API, ale je možné dotazovať sa na ňu pomocou metódy POST HTTP[5] protokolu. Údaje poskytuje vo viacerých formátoch, ktoré sa dajú následne upraviť a vybrať z nich požadované dáta.

Dáta získané pomocou týchto služieb budú pripravené na odovzdanie ako parametre pre jednotlivé rozšírenia. Tie sa následne postarajú o získanie fotometrických dát.

3.4 Návrh užívateľského rozhrania

Užívateľské rozhranie by malo byť čo najintuitívnejšie a jednoduché na používanie aj pre menej technicky zdatných užívateľov. Treba dbať na to, aby ovládanie nebolo mätúce a nebolo v ňom priveľa zbytočných prvkov ako napríklad tlačidlá alebo textové polia.

Ideálne riešenie pre vyhľadávanie bude mať len jedno textové pole, do ktorého bude možné vložiť názov alebo súradnice požadovaného objektu. Program sa postará o to, aby rozpoznal o ktorý typ dotazu ide. Pre jednoznačnosť by mala existovať možnosť vnútiť vyhľadávanie vybraným spôsobom. Jednoduché riešenie môže byť napríklad zadať na začiatok dotazu prefix, ktorý vynúti vyhľadávanie jeho priradeným spôsobom.

---

2. Dostupná na: `http://cds.u-strasbg.fr/cgi-bin/Sesame`
3. Representational State Transfer
4. Dostupná na: `http://vizier.u-strasbg.fr/viz-bin/VizieR`
5. Hypertext Transfer Protocol

Aplikácia musí bežať v minimálne dvoch vláknach, jedno pre grafické rozhranie a druhé pre dlhšie výpočty. Zabraňuje to zasekávaniu sa grafického rozhrania pri čakaní na výpočty.

4 Implementácia

V tejto kapitole sa budem podrobnejšie venovať tomu, ako som postupoval pri implementácii jednotlivých častí aplikácie.

4.1 Použité technológie

4.1.1 Java

Ako hlavný programovací jazyk som zvolil objektovo orientovaný jazyk Java. Jeho hlavnou výhodou je, že ho podporujú všetky bežne používané operačné systémy. [4]

Vývojové prostredie som zvolil program IntelliJ Idea od spoločnosti JetBrains.

4.1.2 JavaFX

Pre grafické rozhranie som sa rozhodol použiť knižnicu JavaFX. Táto knižnica je najnovší framework grafického rozhrania pre jazyk Java a od verzie 8 sa stala jeho súčasťou. [5] Pri tvorbe užívateľského rozhrania v JavaFX je potrebné dbať na architektonický vzor MVC.

MVC umožňuje lepšie organizovať a udržiavať kód. Skladá sa z troch hlavných komponentov:

- Model
- View (pohľad)
- Controller (ovládač)

Model slúži na udržiavanie stavu objektov grafického rozhrania a oznámenia o jeho zmene posiela do komponentu Controller. [6]

View slúži na reprezentáciu grafických prvkov aplikácie. Je to jediná časť, ktorú užívateľ vidí a interaguje s ňou. V JavaFX sú pohľady tvorené FXML[1] súbormi a dá sa v nich priamo vytvoriť odkaz na ich ovládač. Tvorba FXML súborov sa dá zjednodušiť pomocou programu Scene Builder, v ktorom sa dá jednoducho vytvoriť celý pohľad bez písania akéhokoľvek kódu. [6]

1. Značkovací jazyk, ktorý slúži na vytváranie užívatelského rozhrania v JavaFX

Controller slúži na premostenie komponentov Model a View. Ovládač zabezpečuje funkcionalitu pre interaktívne prvky užívateľského rozhrania a pri zmene modelu upraví pohľad. [6]

4.1.3 Spring

Ďalší užitočný framework, ktorý som využil, je Spring. Jedným z najdôležitejších dôvodov prečo som ho zvolil je, že zjednodušuje použitie návrhového vzoru Dependency injection. Tento návrhový vzor slúži na oddelenie závislostí medzi triedami. [7]

4.1.4 Log4j 2

Na logovanie aplikácie som využil knižnicu Log4j 2. Je to následník jednej z najpoužívanejších knižníc Log4j. Umožnuje jednoduchú konfiguráciu v XML súbore. [8]

4.1.5 Maven

Na preklad zdrojových kódov som použil nástroj Maven, ktorý zároveň slúži na jednoduché spravovanie závislostí projektu. [9]

4.1.6 Git

Ako verzovací systém pre celý projekt som zvolil Git. Na hosting vzdialeného repozitára som využil webovú službu GitHub[2].

4.1.7 Inno Setup 5

Program Inno Setup 5 slúži na jednoduché vytvorenie inštalačného súboru pre platformu Windows. [10]

4.2 Postup pri implementácii

Počas celého vývoja som postupoval inkrementálne. Začal som spustiteľnou aplikáciou s jednoduchým grafickým rozhraním. Postupne som

2. Repozitár projektu je dostupný na: `https://github.com/m-krajcovic/photometric-data-retriever`

jej pridával nové funkcie. Týmto spôsobom som bol schopný ukázať reálne výsledky v krátkom čase. Tento postup viedol však aj k tomu, že bolo niekoľkokrát potrebné prerobiť veľké kusy implementácie.

Pre uľahčenie zmeny implementácie v jednotlivých moduloch bolo dôležité klásť dôraz na používanie rozhraní. Jednotlivé implementácie bolo možné vďaka tomu jednoducho vymeniť podľa potreby nakonfigurovním frameworku Spring.

4.3 Štruktúra aplikácie

Celý projekt je rozdelený do troch modulov:

- *backend*
- *app*
- *plugins*

Prvý modul s názvom *backend* má na starosti aplikačnú logiku. Obsahuje rozhrania pre vyhľadávanie stelárnych objektov podľa názvu a súradníc, načítanie a spúšťanie jednotlivých rozšírení na získavanie fotometrických dát, prístup ku konfiguračnému súboru a základné operácie nad ním.

Druhý modul *app* predstavuje prezenčnú vrstvu aplikácie. Obsahuje Java triedy na ovládanie (vrstva Controller), udržiavanie stavu pohľadov (vrstva Model), niekoľko vlastných rozšírení pre knižnicu JavaFX a nastavenie frameworku Spring. Obsahuje aj FXML súbory s pohľadmi a CSS[3] pre úpravu prezentácie prvkov v pohľadoch. Ďalšou dôležitou časťou je súbor s textami, ktoré sa zobrazujú v aplikácii. Umožňuje jednoducho lokalizovať celú aplikáciu do viacerých jazykov.

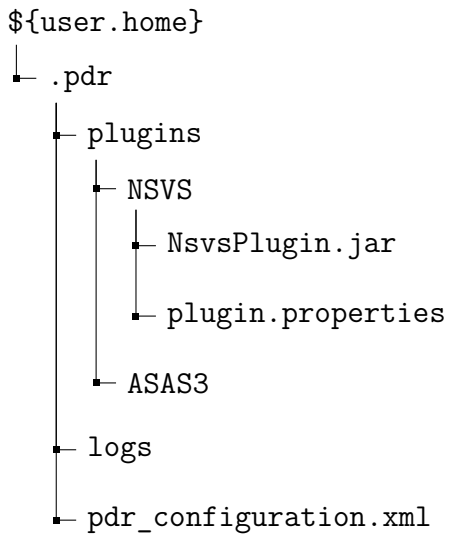
Posledný modul *plugins* obsahuje jednotlivé implementácie rozšírení na získavanie fotometrických dát z rôznych databáz.

3. Cascading Style Sheets (kaskádové štýly)

4.4 Hlavné časti implementácie

Hlavnou triedou aplikácie je trieda *MainApp* z balíka *app*, ktorá je podtriedou *Application* z knižnice JavaFX. Slúži na inicializáciu a následné zobrazenie hlavného okna s vyhľadávaním.

Počas inicializácie sa vytvorí kontext pre framework Spring a skontroluje sa, či existujú potrebné súbory pre fungovanie aplikácie. V prípade, že zložka s aplikačnými dátami neexistuje, vytvorí sa s prednastavenými súbormi.



Obr. 4.1: Ukážka štruktúry aplikačných dát

4.4.1 Získavanie informácií o stelárnych objektoch

Aby bolo možné stelárny objekt nájsť v čo najväčšom množstve rôznych databáz, je potrebné najprv zistiť niekoľko dodatočných informácií. Na získavanie týchto dát používa aplikácia dve webové služby.

Sesame Prístup k službe Sesame zabezpečuje rozhranie *SesameNameResolver*. Na vstupe metódy pre získanie dát berie názov hľadaného stelárneho objektu a vracia objekt typu *StellarObject*.

Implementácia rozhrania na získanie dát zo služby používa HTTP protokol a dotazovaciu metódu GET. Odkaz dotazu vyzerá nasledovne:

http://cdsweb.u-strasbg.fr/cgi-bin/nph-sesame/-oIX/A?$\underbrace{\${názov}}_{\text{názov objektu}}$

Ako parameter sa nastaví názov vyhľadávaného objektu. Služba vráti informácie z niekoľkých zdrojov vo formáte XML[4]. Súradnice a aliasy z výsledku sa uložia do objektu *StellarObject*, ktorý je výstupom metódy rozhrania.

Vizier Získavanie dát zo služby Vizier zabezpečuje rozhranie *VizierResolver*. Na vstupe metódy pre získanie dát berie objekt typu *VizierQuery*, ktorý obsahuje textový reťazec dotazu (súradnice alebo názov) a nepovinný parameter odchýlky. Výstupnou hodnotou je zoznam s objektami *VizierResult*.

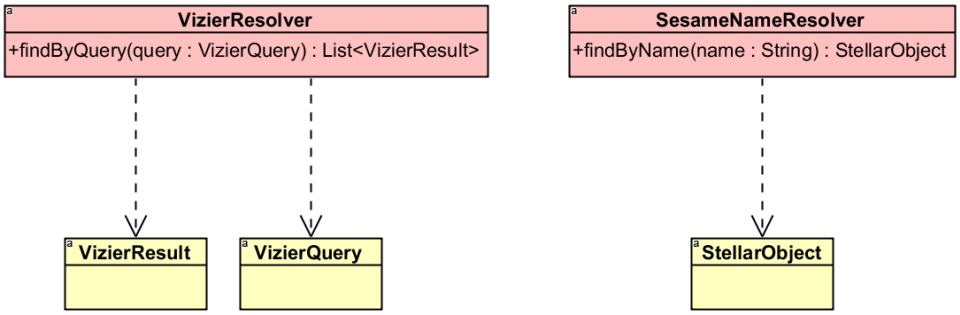
Služba Vizier neposkytuje zdokumentované API. Pri implementácii som musel najprv odsledovať, aké parametre a na akú adresu sa odosielajú pri vyhľadávaní cez formulár na ich webovej stránke. Implementácia v aplikácii využíva tieto parametre, pričom k nim pridáva dotaz a odchýlku a posiela ich službe metódou POST. Služba poskytuje údaje v niekoľkých formátoch. Zvolil som formát so stĺpcami oddelenými bodkočiarkou, z ktorého sa dajú jednoducho získavať hodnoty.

4.4.2 Vyhľadávanie

Aplikácia pri vyhľadávaní objektu postupuje dvomi rôznymi spôsobmi. Záleží, či užívateľ zadal názov alebo súradnice. Oba prípady vedú k naplneniu objektu *StellarObject*, ktorý sa použije ako vstupný parameter pre rozhranie *PhotometricDataRetrieverManager*. Ten má na starosti spustenie jednotlivých rozšírení a získanie dát z nich.

Rozhodnutie o spôsobe vyhľadávania sa určuje podľa regulárnych výrazov, ktoré odpovedajú možným formátom zápisu súradníc. Ak zadaný text neodpovedá žiadnemu z formátov, použije sa vyhľadávanie podľa názvu.

4. Extensible Markup Language (rozšíriteľný značkovací jazyk)



Obr. 4.2: Zjednodušený diagram rozhraní pre získavanie dát o stelárnych objektoch

Pri vyhľadávaní podľa názvu sa pomocou oboch rozhraní *SesameNameResolver* a *VizierResolver* získajú dodatočné informácie, ktoré sa spoja do jedného objektu *StellarObject*.

Pri vyhľadávaní podľa súradníc je možné zadať aj odchýlku súradníc. Podľa súradníc a odchýlky sa pomocou rozhrania *VizierResolver* vyberie z najbližších objektov maximálne 50 a aplikácia dá užívateľovi možnosť vybrať ten, o ktorý má záujem. Po výbere sa využije rozhranie *SesameNameResolver* na získanie ďalších informácií, ktoré sa spoja do objektu *StellarObject*.

4.4.3 Systém rozšírení

Aplikácia musí byť jednoducho rozšíriteľná o nové databázy, a to bez zmeny základného programu. Preto bolo potrebné vytvoriť systém rozšírení. Každé rozšírenie má za úlohu získať fotometrické dáta z určitej databázy. Takéto rozšírenie musí byť implementovateľné v akomkoľvek programovacom jazyku, a preto nebola možná úplná integrácia rozšírení do aplikácie.

Rozšírenie v tejto aplikácii musí obsahovať spustiteľný súbor. Na zariadení, z ktorého je spúšťaná aplikácia, musí byť súbor spustiteľný z príkazového riadku. Na vstupe môže brať niekoľko parametrov, ako napríklad súradnice, jedinečný identifikátor pre databázu alebo URL[5] odkaz na stránku so získateľnými dátami.

5. Uniform Resource Locator (jednotný vyhľadávač zdrojov)

Aby aplikácia rozpoznala nové rozšírenie, je potrebné spustiteľný súbor vložiť spolu s konfiguračným súborom do jednej zložky. Tá musí byť ďalej umiestnená v zložke pre rozšírenia (obr. 4.1).

Spúšťanie rozšírení

Aplikácia pred spustením rozšírení získa dodatočné informácie o objekte ako jeho súradnice a rôzne názvy. Tie sa pošlú rozhraniu *PhotometricDataRetrieverManager*, ktorý slúži na spustenie rozšírení. V ďalšom kroku sa pomocou triedy *ParameterUtils* vytvorí zoznam dostupných parametrov pre rozšírenia z niekoľkých zdrojov:

1. súradnice
2. regulárne výrazy spustené na dodatočných informáciach objektu
3. nastaviteľné spojené parametre
4. vytvorený URL odkaz z predošlých parametrov

Súradnice v stupňoch sa pridajú do zoznamu parametrov pod názvami *radeg* a *decdeg*.

Zo získaných aliasov objektu sa pomocou predom nastavených regulárnych výrazov vyberú identifikátory. Názvy parametrov sú určené pomenovanými skupinami v regulárnom výraze. Príklad regulárneho výrazu je:

$$\underbrace{(?<nsvs>NSVS\backslash s\overbrace{(?<nsvsID>\backslash d*)}^{\text{identifikátor}})}_{\text{celý alias}}$$

Ak regulárny výraz vyhovuje jednému z aliasov objektu, do zoznamu parametrov sa pridajú dva parametre *nsvs* a *nsvsID*.

Spojené parametre obsahujú názov paramera a reťazec obsahujúci iné názvy parametrov z predošlých dvoch krokov, napríklad *${radeg}:${decdeg}* spojí obe súradnice znakom *':'*.

Podobne ako pri spojených parametroch je možné vytvoriť URL odkazy pre každú databázu zvlášť. Týchto odkazov môže byť niekoľko a sú usporiadané podľa preferencie. Môžeme tak mať odkaz, ktorý odkazuje priamo na stránku s dátami alebo odkaz odkazujúci na

vyhľadávanú oblasť, kde sa stelárny objekt nachádza. Prvý odkaz, v ktorom sa dajú zameniť všetky parametre za hodnoty, sa vloží do zoznamu parametrov pod názvom *url*.

V ďalšom kroku sa skontrolujú možné príkazy, s ktorými sa dá rozšírenie spustiť. Použije sa prvý príkaz, v ktorom je možné zameniť všetky parametre.

Príkaz sa spolu s ostatnými parametrami odovzdá objektu triedy *PhotometricDataProcessStarter*. Ten vykoná kontrolu a v prípade, že je príkaz možné spustiť so všetkými potrebnými parametrami, spustí ho v novom procese. Potom číta jeho štandardný výstup, na ktorý rozšírenie vypisuje získané dáta. Prečítané informácie spája do objektov typu *PhotometricData* a vracia ich zoznam.

Získavanie dát z databázy môže byť pomerne časovo náročné, preto som sa rozhodol spúšťať rozšírenia vo viacerých vláknach naraz.

Tvorba rozšírenia

Na vytvorenie rozšírenia netreba implementovať žiadne rozhranie. Jediné podmienky na rozšírenie sú, že musí byť spustiteľné z príkazového riadku, brať dostupné parametre z aplikácie a na štandardný výstup vypisovať získané dáta. Nezáleží na tom, ako rozšírenie dáta získava a väčšina rozšírení, ktoré som vytvoril, obsahuje len jednu triedu s metódou *main*.

V dobe implementácie rozšírení žiadna z databáz neposkytovala jednoduché API na dotazovanie. Pri implementácii som využil knižnicu Jsoup, ktorá slúži na syntaktickú analýzu HTML[6]. Pomocou nej sa dajú jednoducho vyhľadávať elementy v dokumente.

Podrobnejší postup ako vytvoriť a nakonfigurovať rozšírenie popisujem v kapitole 7.

4.4.4 Prístup k dátam

Aplikačné dáta

Keďže aplikácia nepotrebuje ukladať veľké množstvo entít, zvolil som pre ukladanie nastavení XML súbor. Je uložený v aplikačných dátach s názvom *pdr_configuration.xml* (obr. 4.1). Je v ňom uložený zoznam

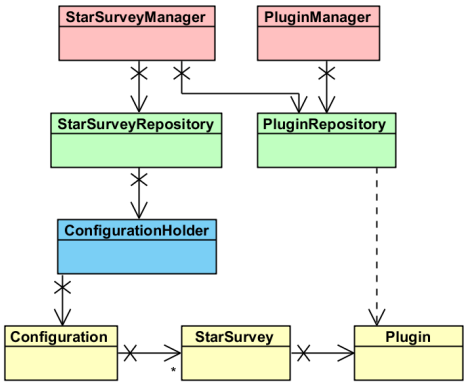
6. HyperText Markup Language (Hypertextový značkový jazyk)

dostupných databáz fotometrických dát, regulárne výrazy na vyhodnotenie identifikátorov objektu a spojené parametre.

Po spustení aplikácie sa celý konfiguračný súbor načíta do pamäte a roztriedi sa do niekoľkých zoznamov, ktore slúžia ako cache. V prípade, že je konfiguračný súbor zmenený ručne počas behu aplikácie, načíta sa súbor znova. Pre prístup ku všetkým dátam z konfigurácie slúži v aplikácii rozhranie *StarSurveyManager*.

Na získanie dát z konfiguráčného súboru slúži rozhranie *StarSurveyRepository*. V konfigurácii je uložený len názov rozšírenia priradeného k databáze a ostatné informácie o rozšírení je potrebné získať pomocou rozhrania *PluginRepository*. V implementácii metód, ktoré vracajú objekty typu *StarSurvey* v rozhraní *StarSurveyManager*, sa využívajú obe rozhrania a výsledok z nich sa spája dokopy.

Na priamu manipuláciu so súborom slúži rozhranie *ConfigurationHolder*. Má na starosti čítanie a ukladanie XML súboru. Na zjednodušenie prekladu XML do objektov a naopak využíva knižnicu *XStream*.



Obr. 4.3: Zjednodušený diagram rozhraní pre prístup k dátam

Rozšírenia

Na uloženie rozšírení slúži zložka s názvom *plugins*, prednastavene umiestnená v zložke s aplikačnými dátami (obr. 4.1). Podobne ako pri aplikačných dátach sa údaje o rozšíreniach načítaju najprv do pa-

mäte. V prípade, že užívateľ pridá počas behu aplikácie do zložky nové rozšírenie, aplikácia to zaregistruje a načíta všetky údaje o rozšíreniach znova.

Pre prístup k rozšíreniam slúži rozhranie *PluginManager*, ktoré deleguje metódy rozhrania *PluginRepository*.

Ostatné nastavenia

Niektoré údaje, ako napríklad cestu k aplikačným dátam alebo k zložke s rozšíreniami, sa ukladajú pomocou knižnice *Preferences*, ktorá je súčasťou jazyku Java. Slúži na jednoduché ukladanie užívateľských nastavení. Miesto, kde sa ukladajú, záleží od operačného systému [11].

4.4.5 Grafické rozhranie

Pri implementácii grafického rozhrania som postupoval podľa princípov vzoru MVC. Implementácia rozhrania sa nachádza v module *app* a skladá sa z niekoľkých hlavných balíkov:

- controller
- model
- javafx
- view

Balík *controller* obsahuje triedy pre ovládanie pohľadov a interakciu s užívateľom. V jednotlivých triedach sú atribúty prepojené s prvkami z pohľadu pomocou anotácie *@FXML*. V pohľade je možné zadefinovať, ktorá metóda sa má zavolať pri určitej udalosti. Takáto metóda je v triede ovládača taktiež označená anotáciou *@FXML*.

V balíku *controller* je aj balík *service*, v ktorom sú triedy na spúšťanie dlhšie trvajúcich funkcií. Sú to podtriedy triedy *Service* z knižnice JavaFX. Poskytuje asynchrónne spúšťanie funkcií, ktoré sa často opakujú. Po skončení operácie sa zavolá metóda z vlákna grafického rozhrania, v ktorej je možné upraviť pohľad.

V balíku *model* sa nachádzajú triedy pre objekty držiace stav pohľadov. Atribúty týchto tried implementujú rozhranie *Property<T>*

z knižnice JavaFX, čo umožňuje ich prepojenie s grafickými prvkami. Pri zmene v pohľade sa automaticky zmení jeho model.

Pri implementovaní bolo potrebné vytvoriť niekoľko vlastných prvkov do JavaFX, ktoré sa spolu s niektorými pomocnými triedami nachádzajú v balíku *javafx*. Vlastné prvky sa dajú jednoducho vkladať aj do FXML šablón, ale aplikácia Scene Builder ich nerozozná, takže ostatné úpravy je potrebné robiť manuálne.

Balík *view* obsahuje FXML šablóny pre pohľady aplikácie. V šablónach je definované, kde sa zobrazia grafické prvky, aká trieda ich riadi a ktorá metóda sa zavolá pri vyvolaní niektorej z udalostí užívateľom.

4.5 Inštalácia a spustenie

Z projektu je možné vytvoriť archív JAR[7] alebo aplikačné balíky pomocou nástroja Maven a jeho rozšírenia pre JavaFX. Aplikačný balík je zložka, ktorá obsahuje všetky potrebné súbory na beh aplikácie spolu s JRE[8]. [12]

Pred vytvorením aplikačných balíkov je nutné nainštalovať všetky potrebné závislosti projektu príkazom

```
mvn clean install
```

Tento príkaz taktiež vytvorí rozšírenia z modulu *star-survey-plugins*, ktoré sa pridajú do zložky `${user.home}‣.pdr‣plugins`.

Hlavnú triedu projektu je možné spustiť bez vytvárania balíka príkazom

```
mvn exec:java -pl app
```

Na vytvorenie samotných aplikačných balíkov slúži príkaz

```
mvn jfx:native -pl app
```

Po spustení sa v zložke `${project.build.directory}‣jfx` objavia dve nové zložky, `app` obsahujúca JAR archív a `native` obsahujúca aplikačné balíky pre aktuálny operačný systém.

7. Java Archive
8. Java Runtime Environment

5 Podobné a budúce práce

Podľa vyjadrenia konzultanta práce, doc. RNDr. Miloslava Zejdu, Ph.D, neexistuje žiadny podobný nástroj na získavanie fotometrických dát.

Aplikácia je momentálne funkčná, obsahuje niekoľko databázových rozšírení a pravdepodobne sa bude rozširovať o ďalšie. Po tom, ako sa zistí v akom jazyku sa najviac implementujú nové rozšírenia, by bolo dobré rozšíriť aplikáciu o priamu integráciu rozšírení.

Vyhľadávanie v aplikácii je možné rozšíriť o ďalšie zdroje dodatočných informácií o stelárnych objektoch, čím by sa dosiahla vyššia úspešnosť pri získavaní fotometrických dát.

Väčšina znovupoužiteľných tried sa nachádza v module *backend*. Triedy sú založené na rozhraniach, takže je jednoduché zmeniť ich implementáciu, ak to bude potrebné.

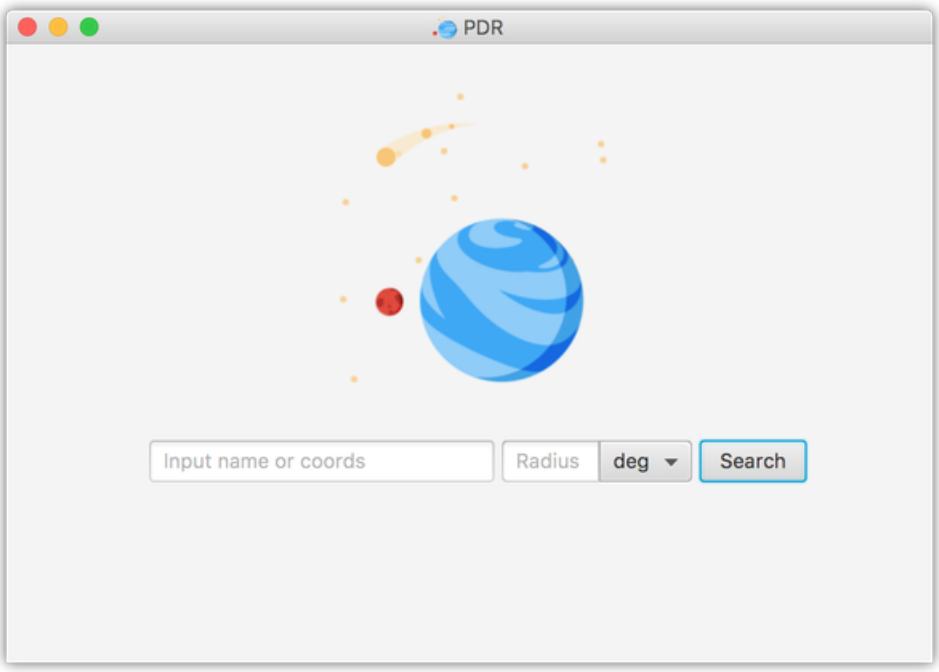
Pre jednoduchší prístup k nástroju by bolo možné ho implementovať ako webovú aplikáciu s využitím modulu *backend*. V tomto prípade by stačilo dovytvoriť užívateľské prostredie.

6 Návod na použitie

Aj keď je ovládanie aplikácie intuitívne, je dobré mať návod na použitie pre podrobnejší popis, ako správne využiť potenciál všetkých funkcií.

6.1 Vyhľadávanie

Po zapnutí a inicializácii aplikácie sa zobrazí hlavné okno s dvomi textovými poliami a tlačidlom na spustenie vyhľadávania, tak ako na obrázku 6.1.



Obr. 6.1: Pohľad na rozhranie s vyhľadávaním

Do prvého textového poľa je možné zadať vyhľadávaný objekt buď podľa jeho názvu, napríklad „*RW Com*“, alebo podľa súradníc v niekoľkých tvaroch:

- v uhlových jednotkách (stupňoch) „*188.25117 +26.71622*“

- v časovo-uhlových jednotkách
  - HH MM SS DD MM SS „*12 33 00.28 +26 42 58.4*“
  - HH:MM:SS DD:MM:SS „*12:33:00.28 +26:42:58.4*“

Pri zadávaní súradníc je možné využiť aj druhé textové pole na zadanie odchýlky a vybrať jednotku odchýlky z rozbaľovacieho menu (deg, arcmin alebo arcsec). Ak nie je zadaná žiadna odchýlka, použije sa hodnota 2 *arcmin*.

Spôsob vyhľadávania sa dá v aplikácii vynútiť zadaním prefixu „*name:*“ alebo „*coords:*“ do prvého textového poľa, čím sa zmení jeho vzhľad ako na obrázku 6.2.

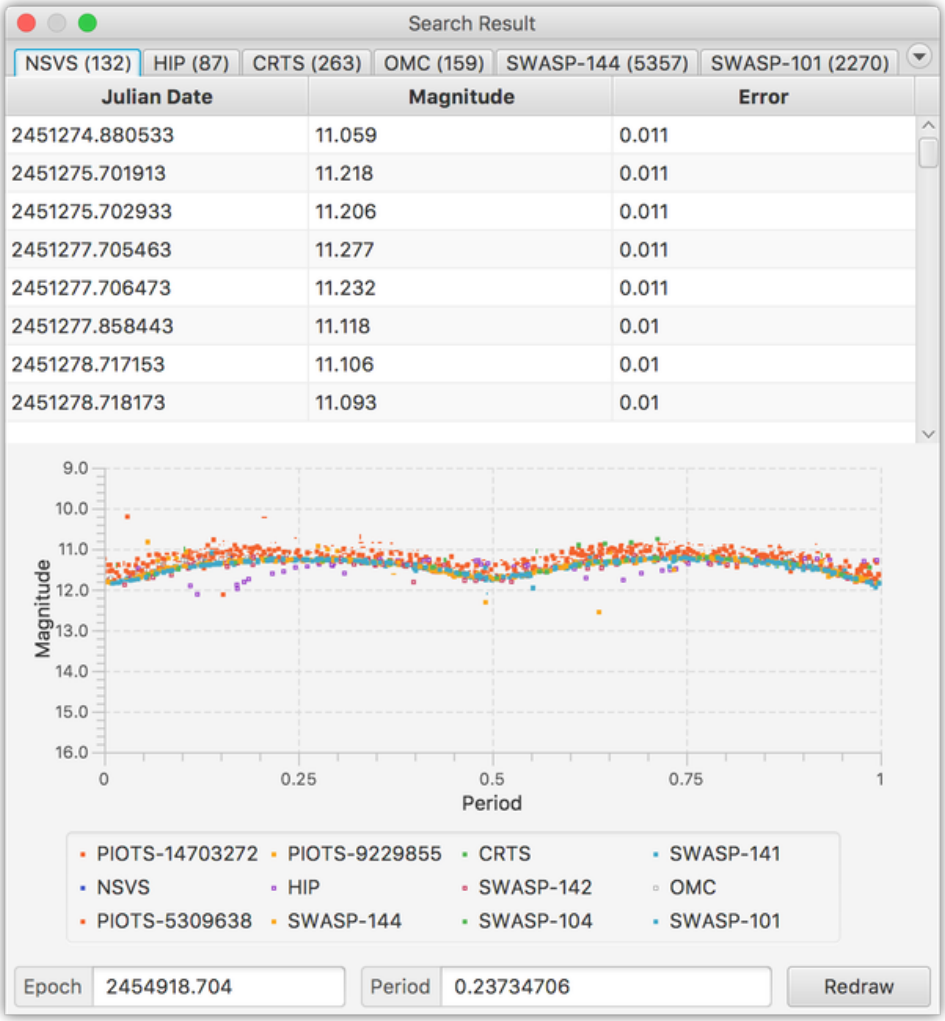
Obr. 6.2: Zmena textového poľa podľa spôsobu vyhľadávania

Po zadaní všetkých údajov stačí stlačiť tlačidlo Search a počkať na výsledok vyhľadávania.

6.2 Zobrazenie výsledku vyhľadávania

Po získaní dát zo všetkých dostupných rozšírení sa zobrazí okno s výsledkom vyhľadávania, podobne ako na obrázku 6.3.

V tomto okne je zoznam všetkých získaných dát, rozdelených podľa jednotlivých databáz do záložiek. V prípade, že sa v databáze nachádza objekt pod viacerými identifikátormi, vznikne pre každý identifikátor samostatná záložka. Spodná časť obsahuje graf zo získaných výsledkov. Ak sa nenájde príslušná epocha a perióda pre stelárny objekt, zobrazí sa v grafe na osi x juliánsky dátum. V grafe je možné zobraziť dáta každej záložky zvlášť, dvojklikom na záložku alebo cez menu Graph › Show... . Na návrat k zobrazeniu všetkých dát v grafe slúži menu Graph › Show all. Pod grafom sa nachádzajú dve textové polia, v ktorých možno manuálne nastaviť epochu a periódu pre graf. Po nastavení hodnôt sa graf prekreslí stlačením tlačidla Redraw.



Obr. 6.3: Okno s výsledkom vyhľadávania

Exportovať dáta je možné dvomi spôsobmi. Pre každú databázu zvlášť cez menu File ⟩ Export one alebo všetky spolu v súbore ZIP cez menu File ⟩ Export All..., poprípade klávesovou skratkou Ctrl + S. Pri oboch spôsoboch je možné vybrať si ako formát exportu CSV[1] alebo textový súbor, kde sú stĺpce hodnôt oddelené tabulátorom.

1. Comma-separated values

V prípade, že sú dostupné originálne súbory z databáz, je možné dostať sa k nim cez menu Original files » Open original....

6.3 Nastavenia

V aplikácii je možné nastaviť, kde sa majú ukladadať aplikačné dáta a rozširenia. Dostať sa do týchto nastavení je možné z hlavného okna cez File » Preferences... alebo klávesovou skratkou Ctrl + Alt + S. Po uložení nových nastavení je potrebné aplikáciu reštartovať, aby sa zmeny aplikovali.

7 Návod na vytvorenie rozšírenia

Aplikáciu je možné rozšíriť o nové databázy. V tejto kapitole sa venujem tomu, ako rozšírenie vytvoriť a pridať ho do aplikácie.

Rozšírenie je samostatný program, ktorý je spustiteľný z príkazového riadku. Zabezpečuje prístup do databázy a získanie fotometrických dát. Pre posielanie správ z rozšírenia do aplikácie sa využíva štandardný výstup. Jeden riadok výstupu vyzerá nasledovne:

$$\underbrace{2453470.79466}_{\text{juliánsky dátum}}, \overbrace{10.91}^{\text{magnitúda}}, \underbrace{0.05}_{\text{chyba}} \overbrace{[, 141]}^{\text{dobrovoľné id}}$$

Rozšírenie môže taktiež ukladať originálne súbory z databáz. Aby bolo možné súbory rozlíšiť v aplikácii, je potrebné aby sa ukladali do zložky *output*, ktorá je v zložke rozšírenia a aby sa názov súboru začínal na názov rozšírenia.

Rozšírenie je možné vytvoriť v akomkoľvek programovacom jazyku. V tomto návode popisujem, ako vytvoriť rozšírenie pre databázu Hipparcos[1] v jazyku Python 2.7.

7.1 Analýza databázy

Ešte pred implementáciou je potrebné získať prístup k fotometrickým dátam z databázy. Dáta objektu z databázy Hipparcos sa dajú získať na odkaze `http://cdsarc.u-strasbg.fr/viz-bin/vizExec/Vgraph?1239/{id}`, kde {id} je parameter pre jedinečný identifikátor objektu databázy Hipparcos. Tento identifikátor bude musieť rozšírenie dostať ako parameter pri spustení. Na tejto stránke je ďalej odkaz *Data as a Table*, cez ktorý sa dá dostať priamo ku textovému súboru s dátami.

1. `http://www.cosmos.esa.int/web/hipparcos/hipparcos-2`

7.2 Implementácia rozšírenia

V tejto sekcii popisujem presný postup ako implementovať rozšírenie pre databázu Hipparcos v jazyku Python 2.7. Celý kód je dostupný v prílohe B.

7.2.1 Získanie fotometrických dát

Prvý krok je vytvoriť súbor *HipPlugin.py*, v ktorom bude implementácia rozšírenia. Celý program využíva len dva moduly, ktoré treba pridať v hlavičke súboru riadkom `import sys, urllib2`. Ďalej treba skontrolovať s koľkými parametrami bol program spustený. Parametre obsahuje zoznam `sys.argv` a prvý je vždy názov súboru. V tomto príklade je potrebný jeden parameter, a preto ak je dĺžka zoznamu menšia ako dva, možeme ho hneď ukončiť.

```
if len(sys.argv) < 2:
    sys.exit(2)
```

Ďalším krokom je vytvorenie odkazu pre dáta z databázy pomocou paramera a získanie jeho obsahu. Dáta na stránke sú uložené po riadkoch a stĺpce sú oddelené medzerou.

```
hid = sys.argv[1]
url = "http://cdsarc.u-strasbg.fr/viz-bin/nph-Plot/Vgraph
   /txt?I/239/./" + hid + "&0&P=0&-Y&mag&-y&-&-&-"
data = urllib2.urlopen(url)
```

Funkcia `urllib2.urlopen(url)` vracia zoznam riadkov na danej adrese, ktorý uložíme do premennej `data`. Následne budeme v cykle prechádzať všetky riadky. Pre každý riadok treba skontrolovať, či ide o záznam dát, komentár alebo len prázdny riadok. S tým nám pomôže funkcia `line_is_valid(string)`.

```
def line_is_valid(string):
    return string and string.strip() \
           and not str.startswith(string, '#')
```

V prípade, že riadok je záznam dát, je potrebné rozdeliť ho po medzerách metódou `str.split()`. K prvému stĺpcu, v ktorom je juliánsky dátum, je potrebné ešte pripočítať hodnotu `2440000`, o ktorú sú dáta posunuté. Upravené dáta potom vypíšeme na štandardný výstup so stĺpcami oddelenými čiarkou.

```
for line in data:
    if line_is_valid(line):
        split = line.split()
        if len(split) >= 3:
            jd = float(split[0]) + 2440000
            mag = split[1]
            err = split[2]
            print ','.join([str(jd), mag, err])
```

7.2.2 Uloženie originálneho súboru

Ak chceme, aby rozšírenie ukladalo aj originálny súbor dát z databázy, treba do implementácie pridať niekoľko riadkov.

V prvom rade je potrebné získať cestu k súboru, do ktorého program uloží dáta. Vytvoríme na to funkciu `get_output_file(file_name)`.

```
def get_output_file(file_name):
    cd = os.path.dirname(
                os.path.abspath(inspect.stack()[0][1]))
    output_dir = os.path.join(cd, "output")
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    file_path = os.path.join(output_dir, file_name)
    return file_path
```

Táto funkcia zistí, v ktorej zložke sa program nachádza pomocou funkcie `os.path.dirname(os.path.abspath(inspect.stack()[0][1]))`. V nej vytvorí zložku *output*, ak ešte neexistuje. Návratovou hodnotou funkcie je cesta k súboru, do ktorého je možné zapisovať originálne dáta.

Bez ručného vymazávania súborov zo zložky *output* môže nastať situácia, že jej veľkosť po dlhodobom používaní aplikácie príliš narastie. Vymazávanie starých súborov môžeme pridať priamo do kódu funkcie `get_output_file`. Pridáme jej parameter `amount_to_keep`, ktorý určuje, koľko súborov sa má v zložke ponechať. Ďalej pridáme do funkcie kód na vymazanie najstarších súborov.

```
    files = os.listdir(output_dir)
    os.chdir(output_dir)
    while len(files) >= amount_to_keep:
        oldest = min(files, key=os.path.getctime)
        os.remove(oldest)
        files = os.listdir(output_dir)
```

Po načítaní dát zo stránky otvoríme súbor na zapisovanie.

```
output_file = open(
        get_output_file("HIP-" + hid + ".txt", 3), 'w+')
```

Potom uložíme každý riadok ešte pred úpravou do súboru funkciou `output_file.write(line)`.

Kód celého rozšírenia aj s ukladaním originálneho súboru je v prílohe B.

7.2.3 Ostatné funkcie rozšírenia

Niekedy sa stane, že je ten istý objekt uložený v jednej databáze pod rôznymi identifikátormi a každý obsahuje fotometrické dáta namerané inou kamerou. Ak chceme, aby sa v aplikácií tieto dáta rozlíšili, je potrebné pridať pri výpise na štandardný výstup ďalší stĺpec obsahujúci identifikátor, podľa ktorého sa dáta rozdelia do skupín. Originálny súbor pre jednotlivé identifikátory musí mať potom názov začínajúci na *${názov rozšírenia}-${id}*.

7.3 Konfigurácia rozšírenia

Keď máme vytvorený program pre rozšírenie, pridáme ho do zložky pomenovanej podľa názvu rozšírenia. Pri tomto rozšírení je to zložka *HIP*. Do tejto zložky musíme pridať ešte textový súbor *plugin.properties* s nastaveniami rozšírenia. Bez tohto súboru aplikácia rozšírenie nerozozná. Obsah tohto súboru je zobrazený na obrázku 7.1

```
mainFile=HipPlugin.py
command=python ${mainFile} ${hipID}
```

Obr. 7.1: Príklad obsahu plugin.properties

Hodnota *mainFile* je názov spustiteľného súboru a hodnota *command* predstavuje príkaz, ktorým sa dá spustiť. Ak sa dá súbor spúšťať s viacerými príkazmi, je možné napísať ich za sebou oddelené bodkočiarkou. Parametre v príkaze sa zamenia v aplikácii, ${mainFile} na celú cestu k súboru a ${hipID} sa zamení identifikátorom pre databázu Hipparcos, ktorý sa získa pomocou regulárneho výrazu spusteného na názvoch objektu.

Získanie identifikátorov je závislé od služby Sesame[2]. Nie vždy sú dostupné názvy vo všetkých databázach. Okrem identifikátorov je možné ako parametre použiť aj súradnice objektu v stupňoch (*radeg*, *decdeg*) alebo hodinách (*rah*, *dech*). Čiastočný zoznam parametrov, ktoré je možné použiť v súčasnej verzii aplikácie, sa nachádza v prílohe A.

7.3.1 Pridanie nového parametra

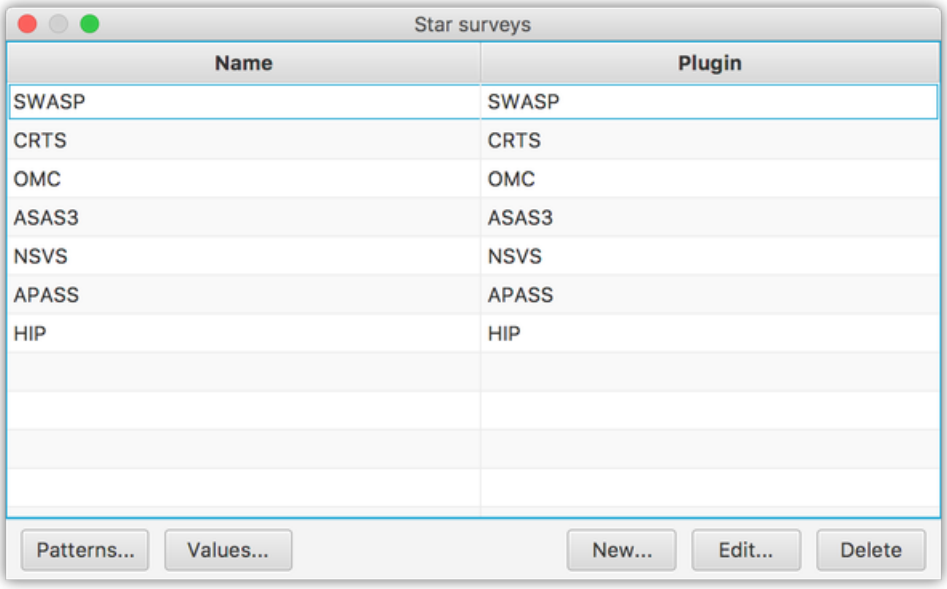
V prípade, že aplikácia neobsahuje potrebný identifikátor pre databázu, je možné pridať ho. V prvom rade je potrebné zistiť, v akom tvare sa nachádza názov objektu v službe Sesame. Pre databázu Hipparcos je to tvar *HIP <ID>*, napríklad *HIP 56088*. Regulárny výraz na získanie identifikátora z názvu je *(?<hip>HIP\s(?<hipID>\d*))*. Je rozdelený do dvoch skupín, *hip*, ktorá obsahuje celý názov a *hipID*, ktorá obsahuje len identifikátor.V prípade, že sa názov nachádza vo výsledku zo služby Sesame a vyhovuje regulárnemu výrazu, pridajú sa parametre *hip* a *hipID* do zoznamu dostupných parametrov pre spúšťanie rozšírení.

Do zoznamu regulárnych výrazov sa v aplikácii dá dostať z hlavného okna cez menu File ⟩ Star Surveys... alebo klávesovou skratkou Ctrl + S . Na spodku je tlačidlo Patterns..., ktoré otvorí okno so zoznamom výrazov. Toto okno zobrazuje všetky už pridané výrazy a tlačidlá pre upravovanie, mazanie a pridávanie nových výrazov.

7.4 Pridanie rozšírenia do aplikácie

Keď už máme zložku s rozšírením, treba ju prekopírovať do zložky rozšírení pre aplikáciu. Zložka rozšírení je prednastavene uložená v `${user.home}` ▸ `.pdr` ▸ `plugins`. Dostať sa k nej je možné aj z aplikácie, cez menu File ⟩ Open plugins folder alebo klávesovou skratkou Ctrl + P . Pridanie rozšírenia do zložky nestačí na to, aby ho aplikácia spúšťala. Je potrebné priradiť ho jednej z databáz v aplikácii. Do prehľadu databáz, ktorý je na obrázku 7.2, sa dá dostať pomocou menu File ⟩ Star surveys... alebo klávesovou skratkou Ctrl + S z hlavného okna.

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



Obr. 7.2: Prehľad databáz v aplikácii

Po stlačení tlačidla New... sa otvorí okno, v ktorom zadáme názov databázy a vyberieme rozšírenie z rozbaľovacieho menu. Po potvrdení tlačidlom Ok sa pridá do zoznamu nová databáza s rozšírením. Pri nasledujúcom vyhľadávaní už bude aplikácia využívať aj nové rozšírenie.

8 Záver

Táto bakalárska práca sa venovala vytvoreniu nástroja pre získavanie fotometrických dát stelárnych objektov. Dôležitou súčasťou bolo vytvorenie systému pre budúce rozšírenia o nové databázy.

V úvodnej časti sme sa zoznámili s pojmom premennej hviezdy a spôsobom spracovania fotometrických dát.

Druhá kapitola sa venuje popisu požiadaviek, funkčných a nefunkčných, ktoré boli kladené pri vytváraní tejto práce.

Nasledujúca kapitola sa zaoberá mojím návrhom ako správne vyriešiť jednotlivé funkcie aplikácie.

Štvrtá kapitola obsahuje popis postupu a samotnej implementácie vytvorenej aplikácie. Je tu vysvetlené, ako je projekt štrukturovaný do viacerých balíkov, ktoré možno využiť samostatne. Ďalej je tu popísané, aké technológie a prečo som ich použil.

V piatej kapitole sa venujem spôsobom, ako je možné aplikáciu vylepšiť alebo ju doplniť o novú funkcionalitu v budúcnosti.

V nasledujúcich dvoch kapitolách popisujem pre obyčajného užívateľa, ako ju správne používať, alebo rozšíriť o novú databázu fotometrických dát.

Výsledok bakalárskej práce má slúžiť ako praktická pomôcka pre astronómov, ktorá im má uľahčiť zber dát z viacerých on-line databáz. Doteraz bolo potrebné sťahovať dáta z každej databázy zvlášť, čo je časovo náročné.

Literatúra

[1] Zdeněk MIKULÁŠEK a Miloslav ZEJDA. *Úvod do studia proměnných hvězd*. Brno: Masarykova univerzita, 2013.

[2] AAVSO. *The International Variable Star Index (VSX)*. <https://www.aavso.org/vsx/> (cit. 7.5.2016).

[3] Jython. *JythonFaq/GeneralInfo - JythonWiki*. <https://wiki.python.org/jython/JythonFaq/GeneralInfo> (cit. 8.5.2016).

[4] IBM Knowledge Center. *Advantages of Java*. <https://www.ibm.com/support/knowledgecenter/ssw_aix_71/com.ibm.aix.performance/advantages_java.htm> (cit. 3.5.2016).

[5] JavaFX Documentation. *1 JavaFX Overview (Release 8)*. <http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm> (cit. 3.5.2016).

[6] Google Chrome. *MVC Architecture - Google Chrome*. <https://developer.chrome.com/apps/app_frameworks> (cit. 3.5.2016).

[7] Spring Documentation. *5. The IoC container*. <http://docs.spring.io/autorepo/docs/spring/3.2.x/spring-framework-reference/html/beans.html> (cit. 3.5.2016).

[8] Apache Software Foundation. *Apache Log4j 2*. <http://logging.apache.org/log4j/2.x//> (cit. 15.5.2016).

[9] Apache Software Foundation. *Welcome to Apache Maven*. <https://maven.apache.org/> (cit. 16.5.2016).

[10] Jordan Russell. *Inno Setup*. <http://www.jrsoftware.org/isinfo.php> (cit. 15.5.2016).

[11] Oracle Java Documentation. *Preferences (Java Platform SE 7)*. <http://docs.oracle.com/javase/7/docs/api/java/util/prefs/Preferences.html> (cit. 3.5.2016).

[12] JavaFX Documentation. *6 Self-Contained Application Packaging*. <https://docs.oracle.com/javafx/2/deployment/self-contained-packaging.htm> (cit. 15.5.2016).

A Zoznam vybraných parametrov

Parameter	Popis	Príklad
mainFile	úplná cesta k rozšíreniu	`C:\HipPlugin.py`
radeg	rektascenzia v stupňoch	118.27
decdeg	deklinácia v stupňoch	27.15
rah	rektascenzia v hodinách	7.88
dech	deklinácia v hodinách	1.81
vs, vsID	názov objektu v GCVS	V* RW Com
gcvs, gcvsID	názov objektu v GCVS	V* RW Com
iomc, iomcID	názov objektu v IOMC	IOMC 2677000065
aavso, aavsoID	názov objektu v AAVSO	AAVSO 2138+43
crts, crtsID	názov objektu v CRTS	CRTS J123300.1+264257
swasp, swaspID	názov objektu v SWASP	1SWASP J123300.28+264258.3
tmass, tmassID	názov objektu v 2MASS	2MASS J12330028+2642582
wolf, wolfID	názov objektu v WOLF	Wolf 423
tyc, tycID	názov objektu v TYC	TYC 1991-1724-1
hic, hicID	názov objektu v HIC	HIC 61243
nsvs, nsvsID	názov objektu v NSVS	NSVS 7622769
hip, hipID	názov objektu v HIP	HIP 61243
asas, asasID	názov objektu v ASAS	ASAS J123300+2642.9

B Zdrojový kód ukážkového rozšírenia

```
import sys, urllib2, os, inspect

if len(sys.argv) < 2:
    sys.exit(2)


def line_is_valid(string):
    return string and string.strip() \
            and not str.startswith(string, '#')


def get_output_file(file_name, amount_to_keep):
    cd = os.path.dirname(
                os.path.abspath(inspect.stack()[0][1]))
    output_dir = os.path.join(cd, "output")
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    file_path = os.path.join(output_dir, file_name)
    files = os.listdir(output_dir)
    os.chdir(output_dir)
    while len(files) >= amount_to_keep:
        oldest = min(files, key=os.path.getctime)
        os.remove(oldest)
        files = os.listdir(output_dir)
    return file_path


hid = sys.argv[1]
output_file = open(
        get_output_file("HIP-" + hid + ".txt", 3), 'w+')
url = "http://cdsarc.u-strasbg.fr/viz-bin/nph-Plot/Vgraph
   /txt?I/239/./" + hid + "&0&P=0&-Y&mag&-y&-&-&-"
data = urllib2.urlopen(url)
for line in data:
    output_file.write(line)
    if line_is_valid(line):
        split = line.split()
        if len(split) >= 3:
            jd = float(split[0]) + 2440000
            mag = split[1]
            err = split[2]
            print ','.join([str(jd), mag, err])
```

C Elektronické prílohy

- *photometric-data-retriever.zip* - obsahuje Maven projekt so zdrojovými kódmi aplikácie
- *PDR.jar* - spustiteľný program v súbore JAR
- *plugins.zip* - obsahuje všetky vytvorené rozšírenia pre aplikáciu, je potrebné prekopírovať ich do zložky `${user.home}‣.pdr`
- *HIP.zip* - obsahuje ukážkové rozšírenie z kapitoly 7