SLOVENSKÁ TECHNICKÁ UNIVERZITA BRATISLAVA
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Tímový projekt

OBJEKTOVÉ ÚLOŽISKO DÁT

Bc. Juraj Bahno
Bc. Lenka Baková
Bc. Zuzana Jalcová
Bc. Peter Kajan
Bc. Marek Uhlár
Bc. Katarína Valalikova

Vedúci tímového projektu: Ing. Ľubomír Varga
Akademický rok: 2010/11

Obsah

Kapitola 1

Úvod

Autor: Katarína Valaliková

V súčasnosti existuje množstvo spôsobov ukladania rôznych druhov dát. Fotografie si môžeme ukladať napríklad prostredníctvom sociálnych sietí, kde si ich následne môžu zobraziť a prezrieť naši kamaráti, známi, rodičia. Pre bežného a menej náročného používateľa je toto riešenie ukladania fotografií postačujúce. Nevýhodou sociálnych sietí však je, že sú centralizované a tým sa čiastočne stráca kontrola nad informáciami.

Ďalší problém nastáva pri náročnejších používateľoch, ktorí si chcú fotografie nielen uchovávať a prezerať, ale aj kategorizovať ich či vyhľadávať v nich podľa zvolených kritérií. Pre ukladanie dokumentov, video záznamov či hudobných skladieb musíme zasa hľadať iné riešenia ukladania.

Hlavným cieľom nášho tímu je spraviť web ako distribuované úložisko objektov a skúsiť ho pochopiť aj z inej stránky akú poznáme. Mnohí sa už určite stretli s pojmami ako je sémantický web, sociálna sieť či sociálny graf. Vytýčený cieľ by sme chceli naplniť práve využitím sémantického webu, ktorý umožňuje pochopiť informáciám na webe nielen ľuďom, ale aj počítačom a následne potom s nimi pracovať. Informácie, ktoré sú strojovo spracovateľné sú reprezentované formou metadát.

1.1 Účel a prehľad dokumentu

Tento dokument vznikol ako výsledok tímovej práce na Fakulte Informatiky a Informačných technológií Slovenskej Technickej Univerzite v Bratislave na predmete Tímový projekt
v akademickom roku 2010/2011.

V kapitole 2 Analýza problému je rozobratá problematika metadát a sú uvedené niektoré štandardy, ktoré sa používajú v spojení s pojmom metadáta. Ďalej v tejto kapitole nasleduje analýza troch systémov, ktoré slúžia na ukladanie fotografií spolu s metadátami. Kapitola 3 Opis riešeného problému je problém priblížený definovaním cieľov vyvíjaného produktu. Kapitola 4 Špecifikácia požiadaviek je rozdelená na

dve časti. Prvú časť predstavujú funkcionálne požiadavky, kde sú opísané prípady použitia. Druhú časť predstavujú nefunkcionálne požiadavky na systém. Nasleduje kapitola 5 Návrh riešenia, kde je rozobraná architektúra systému, dekompozícia systému na jednotlivé komponenty a ich opis, návrh GUI a návrh CLI. V kapitole 6 Opis riešenia sa nachádza opis prostriedkov použitých na vývoj aplikácie, technológií použitých pri vývoji, porovnanie RDF databáz a testy na Mularu.

1.1.1 Skratky

Autor: Marek Uhlár

BSD permisívna licencia, používaná pre open source softvér

CRUD (Create, Read, Update and Delete) – štyri základné funkcie pri perzistentnom uchovávani

DNG (Digital Negative) - súborový formát pre uchovávanie obrázkov od spoločnosti Adobe

EXIF (Exchangeable image file format) – špecifikácia pre formát obrázkov využívajúci existujúce súborové formáty (JPEG, TIFF) s pridaním metadát

FTP (File Transfer Protocol) - protokol určený na prenos súborov

GPS (Global Positioning System) - satelitný navigačný systém

ITPC IIM (International Press Telecommunications Council Information Interchange Model) – súborová štruktúra a sada metadát, ktorá môže byť aplikovaná na multimediálne súbory

JPEG (akronym pre Joint Photographic Experts Group) – súborový formát pre uchovávanie obrázkov

PSD (Adobe Photoshop) – natívny súborový formát pre Adobe Photoshop

RDF (Resource Description Framework) – metadátový model

RIFF WAV - súborový formát pre uchovávanie obrázkov

sRGB (standartd Red Green Blue) – štandardizovaný farebný priestor

TIFF (Tagged Image File Format) - súborový formát pre uchovávanie obrázkov podporujúci tagy

URI (Uniform Resource Identifier) – reťazec znakov určený na identifikáciu

URL (Uniform Resource Locator) – slúži k presnej špecifikácií umiestnenia zdrojov na internet

WAV (Waveform Audio File Format) - súborový formát pre uchovávanie zvukových stôp

XML (eXtensible Markup Language) – značkovací jazyk vyvinutý konzorciom W3C

Kapitola 2

Analýza problému

V tejto časti je opísané ako sa využívajú metadáta, čo je to Adobe XMP a rámec opisujúci zdroje (RDF). Nachádza sa tu aj analýza vybraných existujúci riešení: Flickr, Adobe LightRoom a Google Picassa.

2.1 Metadáta

Autor: Katarína Valaliková
Metadáta sú často označované ako dáta o dátach alebo informácie o informáciách. Sú to štruktúrované informácie, ktoré opisujú, vysvetľujú, lokalizujú alebo iným spôsobom uľahčujú získavanie, používanie alebo manažovanie zdrojov.

Metadáta sa zvyknú používať ako záznamy o zdrojoch (videá, fotografie, dokumenty) a ako dáta, ktoré sú strojovo spracovateľné. Existuje viacero typov metadát a viacero schém a štandardov na ukladanie metadát. Medzi tri základné typy patria:

- opisujúce metadáta (napr. meno autora, rok vydania knihy, vydavateľstvo)
- štrukturálne metadáta (napr. usporiadanie strán, ktoré vytvárajú kapitolu)
- administratívne metadáta (dva druhy: manažment práv, metadáta potrebné na uchovanie zdroja).

Metadáta sa používajú na opis zdrojov rôznych úrovní (práca, vyjadrenie, manifest) a rôznych druhov prepojení (zdroj, kolekcia zdrojov, časť rozsiahlejšieho zdroja). Môžu byť uložené priamo v zdroji alebo osobitne, vo vlastnej databáze alebo súbore. Ukladanie metadát priamo do zdroja (časť v HTML alebo v obrázkoch) zaručí, že sa metadáta nestratia, pri zmene sa obnovia aj dáta, aj metadáta a rovnako je vyriešený problém s previazaním dát a metadát. Problém je, že nie všetky typy objektov podporujú ukladanie metadát priamo v objekte. Ukladanie medatát mimo objektu môže zjednodušiť ich manažovanie, vyhľadávanie a získavanie.

Metadáta uľahčujú prácu s objektami, ich hľadania, uchovávanie, identifikáciu. Umožňujú:

- vyhľadávanie zdrojov na základe rôznych kritérií,
- identifikovanie zdrojov,
- grupovanie podobných zdrojov,
- rozpoznávanie rôznych zdrojov,
- poskytnutie informácie o lokalizácii,
- porozumieť obsahu nielen ľuďmi, ale aj strojmi – zaručenie interoperability (nezáleží na hardvéri, softvéri, dátovej štruktúre),
- poskytnutie informácie identifikujúcej objekt, ktorý opisujú (URL objektu,...),
- aby dáta boli zachované aj do budúcnosti (archivovanie, uchovávanie metadát).

Existuje viacero štandardov (schém) na vyjadrenie metadát. Každá schéma pozostáva z množiny elementov označovaných ako sémantika schémy a hodnôt elementov označovaných ako obsah. Medzi známe štandardy na uchovávanie metadát patria nasledovné:

- Dublin Core
- Iniciatíva kódovania textu (z angl. Text Encoding Initiative - TEI)
- Štandard kódovania a prenášania metadát (z angl. Metadata Encoding and Transmission Standard - METS)
- Schéma metadát pre opis objektu (z angl. Metadata Object Description Schema - MODS)
- Kódovaný opis archívu (z angl. Encoded Archival Description - EAD)
- Učenie sa metadát objektu (z angl. Learning Object Metadata – LOM)
- Rozšíriteľná platforma metadát (z angl. Extensible Metadata Platform - XMP)

Tieto štandardy sú najčastejšie definované XML Schémou, ktorá slúži na vytváranie XML dokumentov, štandardným všeobecným značkovacím jazykom (z angl. Standard Generalized Mark-up Language - SGML) s využitím definície typu dokumentu (z angl. Documnet Type Definition - DTD) a rámcom opisujúcim zdroje (z angl. Resource Description Framework – RDF). V najväčšej miere sa na ukladanie metadát používa formát rámca opisujúceho zdroje (RDF formát).

2.2 Adobe XMP (eXtensible Metada Platform)

Autor: Marek Uhlár
Štandard XMP od Adobe opisujúci definíciu, vytváranie a spracovanie metadát. Metadáta vo formáte XMP môžu byť pridané do širokého spektra multimediálnych súborov (PDF, JPEG, JPEG 2000, GIF, PNG, HTML, TIFF, Adobe Illustrator, PSD, MP3, MP4, Audio Video Interleave, WAV, PostScript, Encapsulated PostScript atď.). Na trhu je veľké množstvo softvéru podporujúce tento formát (ACDSee, ExifTool, Gwenview), väčšina umožňuje aj editáciu.

Pridanie metadát zachová kompatibilitu s XMP-nekompatibilným softvérom. Podpora je vstavaná aj v niektorých verziách operačných systémov (Windows Vista, Windows 7). Metadáta vo formáte XMP môžu opisovať súbor ako celok, ale aj jeho jednotlivé časti (jednotlivé strany PDF). Formát XMP plne podporuje metadátový model RDF.

XMP Toolkit

Toolkit slúžiaci na prácu s XMP formátom. Pozostáva s dvoch knižníc:

1. XMPCore – na prácu s metadátami v o formáte XMP
2. XMPFiles – na samotné vkladanie a vyberanie metadát zo súborov

Je dostupný pod licenciou BSD aj pre programovacie jazyk Java a C++.

2.3 Rámec opisujúci zdroje

Autor: Lenka Baková
Rámec opisujúci zdroje (Resource Description Framework), ďalej len RDF, je štandard slúžiaci na ukladanie metadát. Bol vytvorený v roku 1999, v roku 2004 prešiel aktualizáciou. RDF slúži nielen na ukladanie metadát, ale aj na vyjadrovanie infomrácií o vzťahoch medzi vecami reálneho sveta.

Je jednoduchý na použitie aj na pochopenie. Základnou myšlienkou je, že akákoľvek informácia či vlastnosť sa dá napísať vo forme malých úsekov, tzv. tripletov. Triplet je trojica Objekt - Predikát - Subjekt. Ak by sme mali napríklad objekt typu Fotografia, to, že fotografia má tag škola môžeme považovať za metadáta k fotografii. Aj samotný fakt, že daný objekt je typu fotografia, môžeme považovať za metadáta k objektu. Vo forme tripletu by tieto metadáta vyzerali nasledovne:
Objekt – je – Fotografia
Fotografia – máTag – Škola

RDF označuje všetky zdroje, ktoré popisuje, jedinečným identifikátorom URI. URI býva na webe priradené nie len objektom, ale aj ich vlastnostiam a objektom reálneho sveta. RDF poskytuje len všeobecný model pre popis zdrojov. Neobsahuje žiadnu slovnú zásobu pre vytváranie samotných metadát. Na to slúžia ontológie.

Pre prácu s RDF existuje viacero knižníc či rámcov. Medzi najznámejšie patria:

- Jena
- JRDF
- Sesame

2.4 Analýza existujúcich riešení

V tejto časti dokumentu sú analyzované systémy, ktoré sa približujú zadanému problému. Sú uvedené dva programy, a to Google Picassa, Adobe LightRoom a sociálna sieť Flickr.

2.4.1 Google Picassa

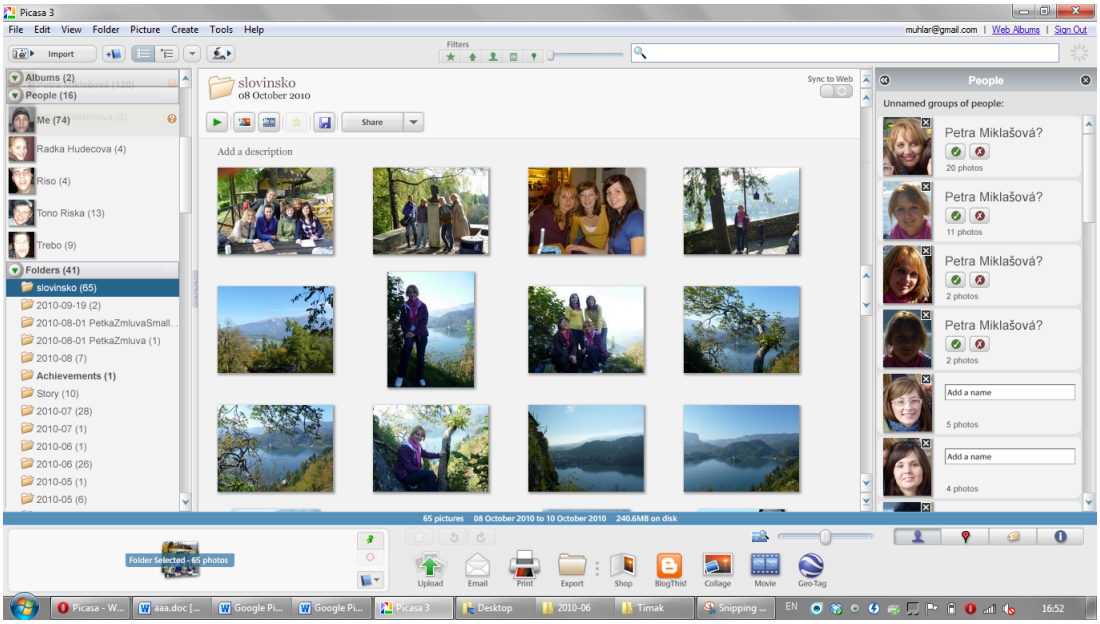
Autor: Marek Uhlár
Program od spoločnosti Google slúžiaci na organizáciu, zdieľanie a základnú úpravu fotiek. Pozostáva z desktopovej a webovej aplikácie, ktoré sú navzájom previazané. Do webovej aplikácie môže používateľ pristupovať pomocou svojho existujúceho konta do poštového klienta od spoločnosti Google. Desktopová aplikácia má príjemné a prívetivé používateľské prostredie (Obr. 2.1).

Import

Je možné importovať fotografie z vybraného adresára. Program má kontrolu na duplikáty a umožňuje pri importe aj zároveň odoslať fotografie do webovej aplikácie.

Export

Program umožňuje exportovanie fotiek z albumu do iného adresára s nastavením zmeny veľkosti a kvality. Do presunutých fotografií je možné pridať vodotlač. Fotky sa dajú tiež exportovať z desktopovej aplikácie do webovej, pričom sa zachová organizácia do albumov.



Obr. 2.1: Desktopová aplikácia

Ovládanie a možnosti

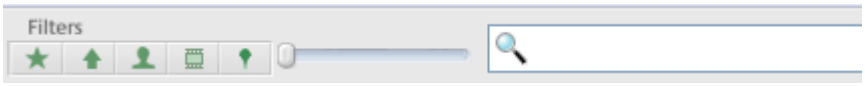
Po nainštalovaní s potvrdením používateľa prehľadá celý diskový priestor alebo len adresáre určené pre obrázky. Snaží sa vybrať len plnohodnotné fotografie (malé obrázky dá do Other staff). Používateľ vidí na ľavo zoznam priečinkov s obrázkami, tie čo obsahujú metadáta o čase, usporiada podľa roku vyhotovenia. Podporuje vytváranie albumov (virtuálnych priečinkov) a ich upload na web.

Aplikácia automaticky vyhľadá tváre na zaradených fotografiách. Algoritmus je veľmi účinný. Otagovanie ľudí na 2000 fotiek trvá priemernému používateľovi cca. 15 minút. Je podporované manuálne pridávanie tagov. Program podporuje Extensible Metadata Platform s Exchangeable image file format (JPEG, TIFF Rev. 6.0, and RIFF WAV dohromady s metadátami) vrátane zmenšených obrázkov (thumbnails) a ITPC. Geotaging je podporovaný nielen z EXIF, ale aj manuálnym vyhľadaním na mape (previazané s Google Maps). Veľkým nedostatkom je, že nie je možné jednoduchou cestou preniesť fotografie na iné PC vrátane tagov.

Vyhľadávanie

Vyhľadávanie prebieha pomocou zadania reťazca. Program tento reťazec vyhľadáva v názvoch súborov, adresárov, albumov, manuálne pridaných tagoch, EXIF (nie plná podpora). Program nepodporuje vyhľadanie reťazca iba v jednej z kategórií (napr. iba v manuálnych tagoch).

Vyhľadávanie sa dá kombinovať s prepínačmi. Pričom je možne vyhľadávať len fotky označené hviezdičkou, nasadené, obsahujúce tvár, video, obsahujúce geotag (Obr. 2.2). Tieto prepínače je možné navzájom kombinovať. Podporuje operátor AND pomocou medzery medzi dvoma výrazmi a operátor NOT v tvare –výraz. Vyhľadávaný výraz Marek Peter –Juro teda vyhľadá všetky fotky obsahujúce Mareka a Petra, ale neobsahujúce Jura. Z vyhľadaných výsledkov sa vytvorí dočasný album, s ktorým sa dá manipulovať ako s plnohodnotným albumom.

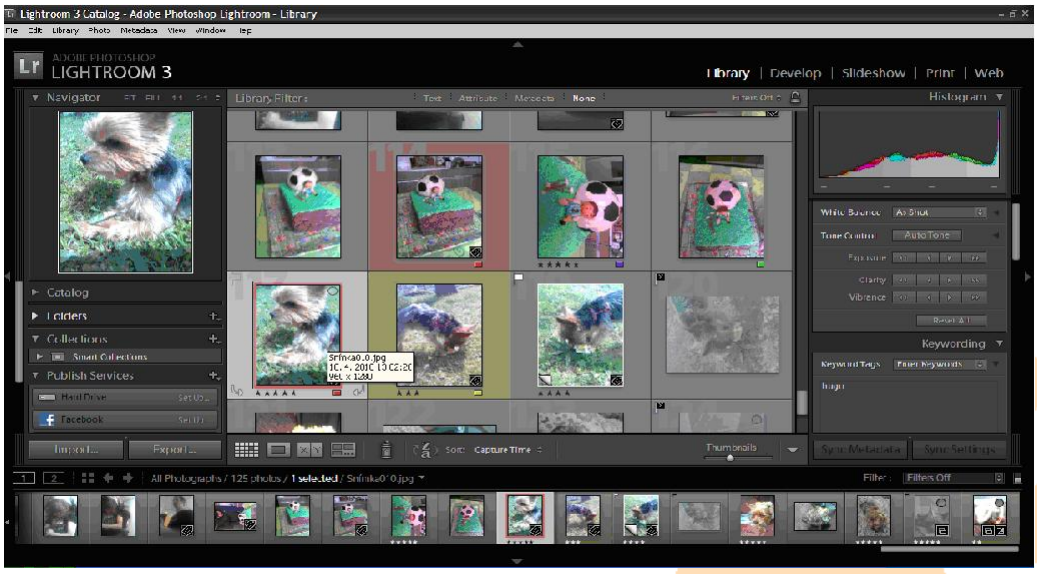


Obr. 2.2: Vyhľadávanie

2.4.2 Adobe LightRoom

Autor: Lenka Baková

Adobe LightRoom (Obr. 2.3) je program slúžiaci na importovanie, upravovanie, zdieľanie a uchovávanie fotografií a videí. V nasledujúcom texte budú analyzované len tie funkcie programu, ktoré súvisia s témou projektu.



Obr. 2.3: Adobe LightRoom

Import

Podporované formáty sú RAW, DNG, TIFF, JPEG, PSD. LightRoom nepodporuje súbory typu PNG, NEF, súbory väčšie ako 65.000 pixelov na stranu, súbory väčšie ako 512 megapixelov. Program umožňuje importovať dáta z disku alebo z iného zariadenia. Má funkciu, ktorá po pripojení fotoaparátu alebo pamäťovej karty začne dáta automaticky importovať. Táto funkcia môže byť niekedy nežiadaná, preto je možnosť ju deaktivovať. Takisto je umožnené importovať celý vytvorený katalóg.

Export

Je možné exportovať fotografie v nasledovných formátoch:

- JPEG – označené ako sRGB s maximálnou kvalitou a rozlíšením 240 ppi,
- DNG a
- JPEG e-mail formát – s maximálnou veľkosť 640 pixlov(na šírku alebo výšku), rozlíšenie 72 ppi.

Je možné exportovať celé katalógy alebo kolekcie. LightRoom podporuje doplnky, ktoré rozširujú funkcionalitu exportu, napríklad nahrávanie fotografií priamo na sociálne siete, ako je Facebook alebo Flickr.

Ovládanie a možnosti

LightRoom ukladá fotografie do zložiek, medzi ktorými je možné ich presúvať. Zložky môžeme premenovávať, pridávať alebo vymazávať v počítači priamo, bez toho aby sme mali LightRoom spustený. Fotografie môžeme zoskupovať do kolekcií a katalógov. Presunom sa fotografia fyzicky nepresunie zo zložky, preto môže byť zaradená aj do viacerých kolekcií a katalógov.

Modul Library umožňuje aj tagovanie, pridávanie hodnotenia vybraným dátam, pridávanie popisov a iných metadát. Podporuje metadáta vo formáte IPTC a EXIF so zmenšnými obrázkami (thumbnails) vrátane editácie. Ak fotoaparát podporuje zaznamenávanie GPS údajov, importujú sa spolu s fotografiami a objavia ako EXIF metadáta. Ak fotoaparát nepodporuje GPS, môžeme použiť zariadenia zabezpečujúce geotagovanie na zachytenie GPS súradníc, a potom pomocou iného softvéru zlúčiť tieto údaje s fotografiou. Je možné si vytvárať šablóny nastavení a použiť ich pri exporte a importe. A tak isto je možné si vytvárať aj šablóny metadát a použiť ich pri importe. Metadáta sa uložia ako XMP súbory.

Zaujala nás skutočnosť, že pri úprave fotografií sa úpravy priraďujú k jednotlivým fotografiám, alebo sa upravujú virtuálne kópie fotografií. To znamená, že máme stále zachovanú aj pôvodnú podobu fotografie a nemusíme si jednotlivé kúsky ukladať

duplicitne. Teda šetríme aj miesto na disku. Ďalšia funkcia, ktorá je nápomocná a urýchľuje čas je sprej. Je to niečo ako „metlička“ v MSword. Umožňuje nám preberať formátovanie z jednej fotografie na inú jedným kliknutím. Pohodlne sa ovláda aj vďaka klávesovým skratkám. LightRoom dáta automaticky zálohuje.

LightRoom umožňuje tlačenie celých katalógov, kolekcií, alebo iným spôsobom vybraných fotiek. Fotografie môžeme vytláčať pomocou preddefinovaných mriežkových šablón. Mriežky je možné vytvárať si podľa vlastnej potreby. Môžeme vytlačiť informácie o fotkách, ako je názov súboru, názov, popis a kľúčové slová. Tieto informácie sú prevzaté z metadát, ktoré zadáme v module Library. Informácie sa vytlačia pod každú fotografiu.

Pomocou FTP dokáže LightRoom nahrať fotografie priamo na webový server po vytvorení webovej fotogalérie.

Vyhľadávanie

Triediť a hľadať fotografie je možné podľa rôznych metadát. Napríklad podľa názvu kolekcií, katalógov, hodnotenia, dátumu importu, dátumu zmeny, kľúčových slov. Môžeme vyselektovať tie bez kľúčových slov.

2.4.3 Flickr

Autor: Zuzana Jalcová

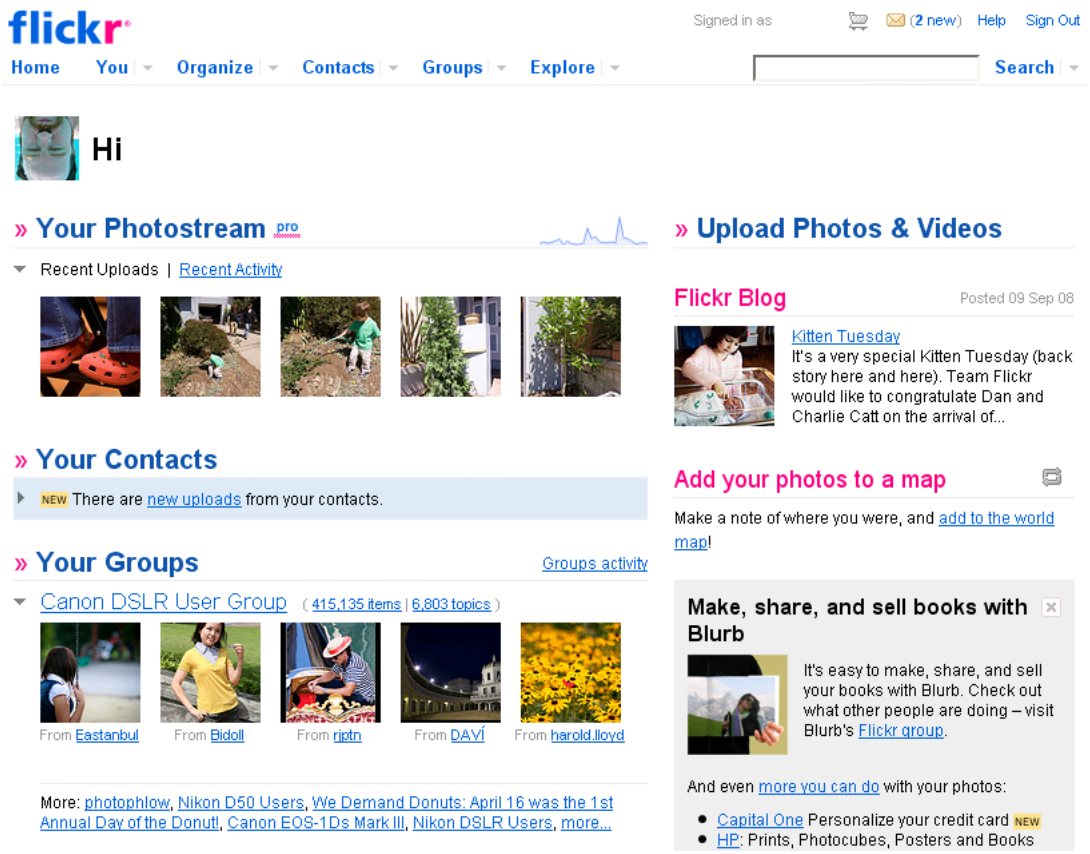
Flickr (Obr. 2.4) je sociálna sieť, ktorá slúži ako úložisko fotografií a videí. Umožňuje zdieľanie fotografií s inými používateľmi, ich komentovanie, označovanie osôb, miest a vecí na fotkách a usporiadanie fotografií do albumov s rôznym tématickým zameraním. Pre používanie Flickr je potrebné, aby si používateľ vytvoril konto na Yahoo!, pretože pre registráciu na Flickr je nutné zadať Yahoo! ID a heslo.

Import

Nahrávanie fotografií na túto sociálnu sieť môže používateľ vykonať rôznymi spôsobmi: cez Flickr Uploadr, iPhoto, Aperture, Windows XP plugins, cez e-mail, cez Flickr upload stránku alebo cez programy tretích strán.

Export

Flickr umožňuje ukladať obrázky na Facebook, Twitter, Yahoo! Updates a Blogging. Taktiež si môže používateľ uložiť na jeho osobnú stránku prezentáciu alebo koláž fotiek.



Obr. 2.4: Flickr

Ovládanie a možnosti

Fotky je potrebné manuálne nahrať na sociálnu sieť Flickr, otagovať ich, prideliť k nim rozširujúce informácie (napr. miesto, kde bola fotka vytvorená). Používateľ si môže vytvárať albumy, zaraďovať svoje fotky do rôznych skupín a aj sám používateľ môže byť členom diskusnej skupiny na Flickr.

Flickr udržuje partnerstvo s Picnick. Jedná sa o aplikáciu na úpravu (odstrániť červené oči, otočiť fotku, orezať a pridávať rôzne efekty) fotografií priamo na Flickr. Aplikácia Maps zaznamená pozíciu na mape kde bola fotka vytvorená a používateľ si môže vyhľadať kliknutím na miesto na mape, aké fotky boli vytvorené v jeho okolí alebo v lokalite, ktorá ho zaujíma.

Ďalšia vymoženosť je Snapfish, ktorá umožňuje používateľovi dať si vytlačiť fotografie vo vysokej kvalite a rôznych veľkostiach až do rozmerov plagátu, vytvárať pohľadnice, knihy s fotografiami alebo tlačenie na plátno. Všetko toto používateľovi pohodlne doručia domov.

Registrácia na Flickr je bezplatná, je však obmedzená veľkosť dát a ak je používateľ neaktívny 90 dní nasledujúcich po sebe, jeho konto je zrušené. Za nadštandardné služby sa platí. Títo používatelia majú okrem neobmedzeného počtu vkladaných fo-

tiek a videí na mesiac k dispozícii aj štatistiky o návštevnosti ich účtu. Flickr podporuje JPEG formát, neanimovaný GIF a PNG. Taktiež je možné nahrať súbor TIFFs a iné typy súborov, tie však budú automaticky skonvertované a uložené ako JPEG.

Vyhľadávanie

Vyhľadávanie prebieha pomocou zadania reťazca. Dá sa nastaviť, kde sa má hľadať - fotky, skupiny alebo používatelia. Používateľ si môže nastaviť, či si želá, aby sa jeho fotka dala vyhľadať.

Kapitola 3

Opis riešeného problému

V tejto časti je riešený problém priblížený definovaním cieľov vyvíjaného produktu.

3.1 Ciele produktu

Autor: Peter Kajan
Vytváraný systém ma slúžiť na uchovávanie objektov a ich metadát. Dôraz bude kladený na pokročilé vyhľadávanie vo veľkom množstve objektov na základe metadát. Cieľom je aby systém podporoval vyhľadávanie na základe nasledovných vlastností metadát:

- hodnotovosť – pri vyhľadávaní bude možne zadať hodnoty metadát ako intervaly alebo množiny (viď Hodnoty),
- typovosť – podpora vyhľadávania na základe typu metadát (viď Vzťahy),
- hierarchickosť – podpora vytvárania hierarchie medzi metadátami a vyhľadávania, ktoré podporuje túto hierarchiu (viď Zahrnutie vzťahov medzi hodnotami do vyhľadávania)

Cieľom projektu je vytvoriť úložisko objektov poskytujúce spomínanú funkcionalitu. K úložisku sa bude pripájať klientská knižnica, ktorá bude poskytovať verejné rozhranie pre klientské aplikácie. Bezpečnosť bude riešená prihlasovaním sa do systému a šifrovaním dát na strane klienta.

Ciele produktu sú:

- vytvorenie úložiska objektov
- vytvorenie klientskej knižnice s verejným rozhraním
- vyhľadávanie objektov na základe hodnotovosti metadát

- vyhľadávanie objektov na základe typovosti metadát
- vyhľadávanie objektov na základe hierarchickosti metadát
- zabezpečenie prihlasovaním do systému
- zabezpečenie dát šifrovaním na strane klienta

Kapitola 4

Špecifikácia

Nasledujúca špecifikácia opisuje jednoduchú GUI aplikáciu, ktorá prezentuje funkcionalitu knižnice spolupracujúcej s úložiskom objektov. Hlavnými výstupmi projektu sú úložisko objektov a knižnica, ktorú budú používať klientské aplikácie. Okrem GUI aplikácie na strane klienta bude implementovaná CLI aplikácia, ktorá bude slúžiť predovšetkým na testovacie účely.

4.1 Identifikácia používateľov

Autor: Zuzana Jalcová
V tejto časti sú identifikované role používateľov systému.

Bežný používateľ

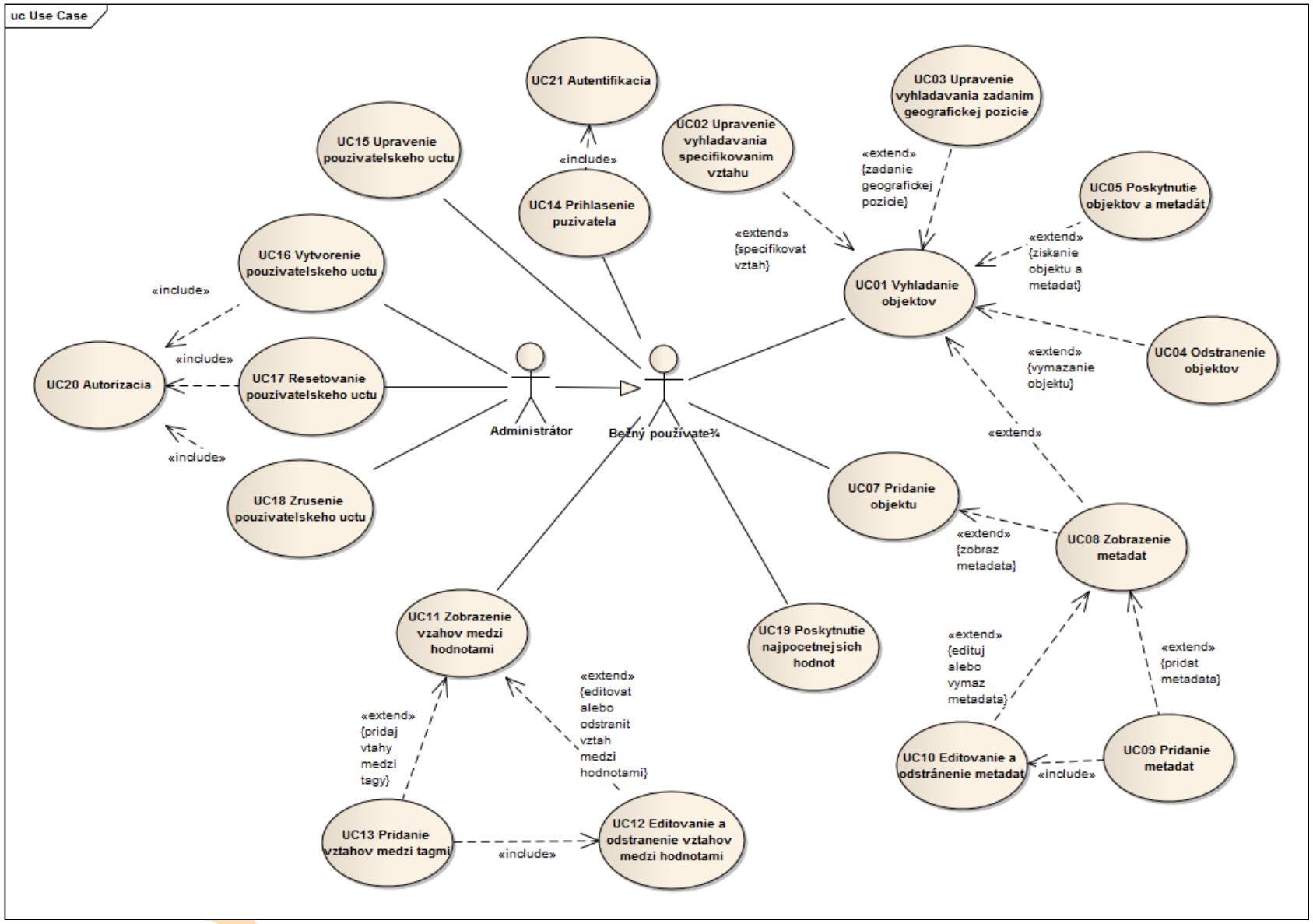
Rola predstavuje bežného používateľa, ktorý má v systéme vytvorený používateľský účet. Po prihlásení pomocou prihlasovacieho mena a hesla môže využívať funkcie ako ukladanie, vyhľadávanie objektov a môže pracovať s metadátami.

Administrátor

Rola administrátor môže využívať všetky funkcie aplikácie ako bežný používateľ. Okrem toho má možnosť spravovať používateľské účty (vytvorenie, odstránenie a resetovanie).

4.2 Diagram prípadov použitia

Autor: Peter Kajan
Na obrázku 4.2 sa nachádza diagram prípadov použitia.



Obr. 4.1: Diagram prípadov použitia

4.3 Opis základných dátových entít

Autor: Peter Kajan
V tejto časti sú definované základné dátové entity systému, ktoré sú nasledovné:

- objekt,
- metadáta, ktoré pozostávajú z:
  - hodnôt,
  - vzťahov.

4.3.1 Objekt

Objekt (alebo Dátový objekt) reprezentuje entitu vloženú používateľom do systému, po ktorej sa bude neskôr dopytovať. Príkladmi objektov sú:

- fotografia,
- dokument,
- film,
- hudobná skladba.

Typ objektu

Objekt môže byť nasledovných typov:

- *Jednoduchý*– jednoduchý objekt uložený v systéme.
- *Revízia* – objekt uchovávajúci si referenciu na objekt, ktorého úpravou vznikol. Vzťah medzi objektom a jeho revíziou je špecifikovaný v tabuľke 4.3.
- *Zložený* – objekt pozostávajúci z viacerých objektov (napríklad panoráma zložená z viacerých fotiek). Objekty, z ktorých sa zložený objekt skladá, sú uložené v systéme samostatne a k zloženému objektu majú vzťah špecifikovaný v tabuľke 4.3.

4.3.2 Metadáta

Metadáta sú dáta o objekte, napríklad dátum vzniku objektu, meno autora alebo klasické tagy používané na označenie objektu. V systéme budú pozostávať z hodnôt a vzťahov.

Tabuľka 4.1: Typy hodnôt a spôsoby popisu množiny hodnôt

Typ hodnoty	**Spôsob popisu množiny hodnôt**	**Upresnenie**
Text	Regulárny výraz	
Formát fotografie	Regulárny výraz	Formát fotografie môže byť JPEG, JPG, GIF, PNG, TIFF, RAW, BMP, SVG
Formát filmu	Regulárny výraz	Formát filmu môže byť avi, mpeg, divx, mpg, div, mov, wmv, 3gp
Formát hudobnej skladby	Regulárny výraz	Formát hudobnej skladby môže byť mp3, wma, flac
Formát dokumentu	Regulárny výraz	Formát dokumentu môže byť doc, docx, dot, pdf, xls, pps, ppt, odf, odt, txt, tex
Číslo	Interval	
Súradnice	Interval (vymedzenie obĺžnikom)	Zemepisná šírka a dĺžka
Geografický názov	-	
Meno	Regulárny výraz	Meno, priezvisko, tituly, stredné mená.
Dátum a čas	interval	
ID objektu	Regulárny výraz	Jedinečný identifikátor objektu v systéme
ID hodnoty	Regulárny výraz	Jedinečný identifikátor hodnoty v systéme

Hodnota

Spolu so vzťahom popisujú vlastnosť objektu alebo inej hodnoty. Hodnoty môžu byť rôznych typov. Pre každý typ je určený spôsob popisu množiny hodnôt (resp. spôsob vymedzenia rozsahu hodnôt), čo bude využívané pri vyhľadávaní (viď Ciele produktu). Typy hodnôt a spôsoby popisu množiny hodnôt sú špecifikované v tabuľke 4.1.

Vzťah

Entita vyjadruje vzťah medzi objektami navzájom, objektami a hodnotami a medzi hodnotami navzájom. Základné typy vzťahov, ktoré bude systém podporovať, sú uve-

dené v tabuľkách 4.3, 4.4 a F.1. Hodnota, ktorej nie je explicitne priradený jej vzťah k objektu, je implicitne priradený vzťah tag. Tieto hodnoty ďalej nazývame tagy.

4.3.3 Zahrnutie vzťahov medzi hodnotami do vyhľadávania

Systém bude podporovať vzťahy medzi hodnotami špecifikované v tabuľke 4.4. Ďalej je popísaný ich vplyv na vyhľadávanie.

Vzťah synonymum

Vzťah zabezpečuje, že do vyhľadávania budú zahrnuté aj synonymá zadaných hodnôt. Vzťah je tranzitívny a komutatívny. Napríklad, ak:

- X má vzťah synonymum k hodnote Y,
- Y má vzťah synonymum k hodnote Z.

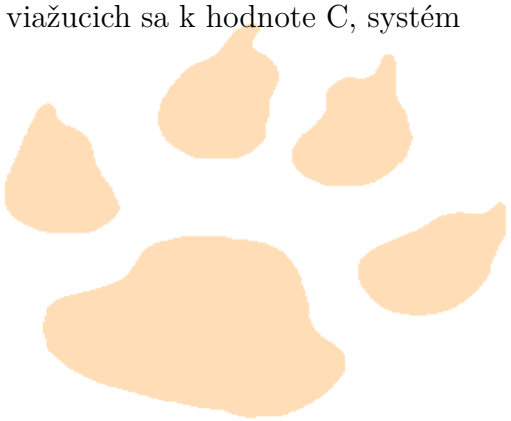
Potom, ak používateľ zadal vyhľadávanie objektov viažucich sa k hodnote x, systém rozšíri vyhľadávanie aj o hodnoty y a z.

Vzťah podhodnota

Vzťah zabezpečuje, že do vyhľadávania budú zahrnuté aj hodnoty nachádzajúce sa v hierarchii pod zadanou hodnotou. Vzťah je tranzitívny. Napríklad ak:

- A má vzťah podhodnota (je podhodnota) k hodnote B
- B má vzťah podhodnota (je podhodnota) k hodnote C
- D má vzťah podhodnota (je podhodnota) k hodnote C

Potom, ak používateľ zadal vyhľadávanie objektov viažucich sa k hodnote C, systém rozšíri vyhľadávanie aj o hodnoty A, B a D.



Tabuľka 4.3: Základné vzťahy medzi objektami a hodnotami a medzi objektami navzájom

Názov vzťahu	**Prináležiaci typ hodnoty**	**Popis**	**Kedy je hodnota platná**
Vzťahy medzi objektami a hodnotami			
tag	text	označuje tagy objektu. Ak nie je uvedený typ vzťahu, vzťah je typu tag	Ak obsahuje aspoň jeden znak (nie biely)
Geografická pozícia	súradnice	Označuje miesto viažuce sa k objektu	Ak označuje miesto na zemskom povrchu
Geografická pozícia	Geografický názov	Označuje miesto viažuce sa k objektu	Ak systém názov rozpozná
autor	meno	označuje autora objektu	Ak obsahuje aspoň jeden znak (nie biely)
názov	text	Označuje názov samotného objektu	Ak obsahuje aspoň jeden znak (nie biely)
zmenený	dátum a čas	Vzťah označuje dátum a čas zmeny objektu	
vytvorený	dátum a čas	Vzťah označuje dátum a čas vzniku objektu	
Vzťahy medzi objektami navzájom			
upravený	ID objektu	Označuje vzťah medzi objektom a jeho revíziou (viď Objekt)	Ak ID objektu je identifikátor objektu uloženého v systéme
zložený	ID objektu	Označuje vzťah medzi zloženým objektom a objektom, z ktorého sa skladá (viď Objekt).	Ak ID objektu je identifikátor objektu uloženého v systéme

Tabuľka 4.4: Vzťahy medzi hodnotami navzájom

Názov vzťahu	Typ hodnoty1	Typ hodnoty2	Popis	Kedy je hodnota platná
Vzťahy medzi hodnotami navzájom				
podhodnota	ID hodnoty	ID hodnoty	Označuje, že hodnota1 je podhodnota hodnoty2 (viď Zahrnutie vzťahov medzi hodnotami do vyhľadávania)	Ak ID hodnoty je identifikátor hodnoty uloženej v systéme
synonymum	ID hodnoty	ID hodnoty	Označuje, že hodnota1 je synonymum hodnoty2 (viď Zahrnutie vzťahov medzi hodnotami do vyhľadávania)	Ak ID hodnoty je identifikátor hodnoty uloženej v systéme

4.4 Opis prípadov použitia

Autor: Peter Kajan
V nasledujúcej časti sú opísané všetky identifikované prípady použitia.

4.4.1 Vyhľadanie objektov

Autor: Peter Kajan
Identifikátor: UC01
Krátky opis: Používateľ vyhľadá objekty na základe metadát.
Predpoklady: žiadne
Dôsledky: žiadne
Účastníci: bežný používateľ
Priorita: 1 = vysoká
Hlavný tok:

1. Používateľ zadá Vyhľadanie objektov
2. Systém zobrazí formulár Vyhľadanie objektov
3. Používateľ zadá tagy podľa, ktorých sa objekty vyhľadajú. Používateľ môže zadať tag ako regulárny výraz.

4. Systém rozšíri hľadanie zahrnutím vzťahov medzi hodnotami (viď tabuľka 4.4) na zadané hodnoty (viď Zahrnutie vzťahov medzi hodnotami do vyhľadávania) a vyhľadá objekty podľa zadaných kritérií

5. Systém zobrazí zoznam nájdených objektov, ktorý obsahuje:

   (a) identifikátory objektov

   (b) metadáta k objektom

   (c) ukážky (thumbnail previews) objektov

Alternatívny tok: Upresnené vyhľadanie objektov
Aktivuje sa po kroku 5 hlavného toku, ak našiel viac ako 100 objektov

1. Systém zobrazí počet nájdených položiek a kritéria hľadania. Ponúkne používateľovi možnosti:

   (a) Upresniť hľadanie

   (b) Zobraziť prvých 100 nájdených objektov

   (c) Ukončiť hľadanie

2. Ak používateľ zadá Upresniť hľadanie, systém uloží kritéria hľadania a prípad použitia pokračuje krokom 3 hlavného toku

3. Ak používateľ zadá Zobraziť prvých 100 nájdených objektov, systém zobrazí zoznam prvých 100 nájdených objektov, ktorý obsahuje:

   (a) identifikátory objektov

   (b) metadáta k objektom

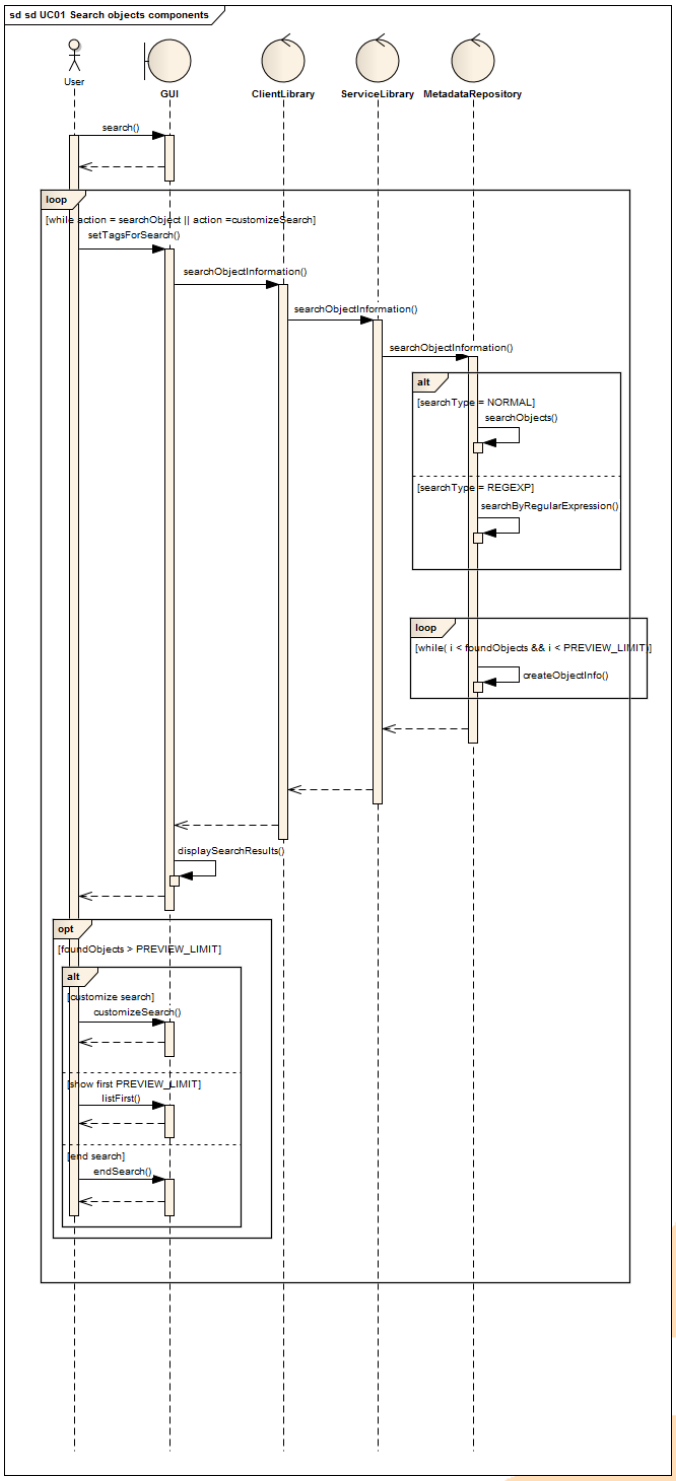
   (c) ukážky (thumbnail previews) objektov

4. Ak používateľ zadá Ukončiť hľadanie, prípad použitia končí.

5. Prípad použitia končí

Body rozšírenia:
Upravenie vyhľadávanie špecifikovaním vzťahu: krok 2 hlavného toku
Upravenie vyhľadávanie zadaním geografickej pozície: krok 2 hlavného toku
Odstránenie objektov: krok 5 hlavného toku
Poskytnutie objektov a metadát: krok 5 hlavného toku

Na obrázku 4.2 je prípad použitia opísaný sekvenčným diagramom.



Obr. 4.2: Sekvenčný diagram UC01 Vyhľadanie objektov

4.4.2 Upravenie vyhľadávania špecifikovaním vzťahu

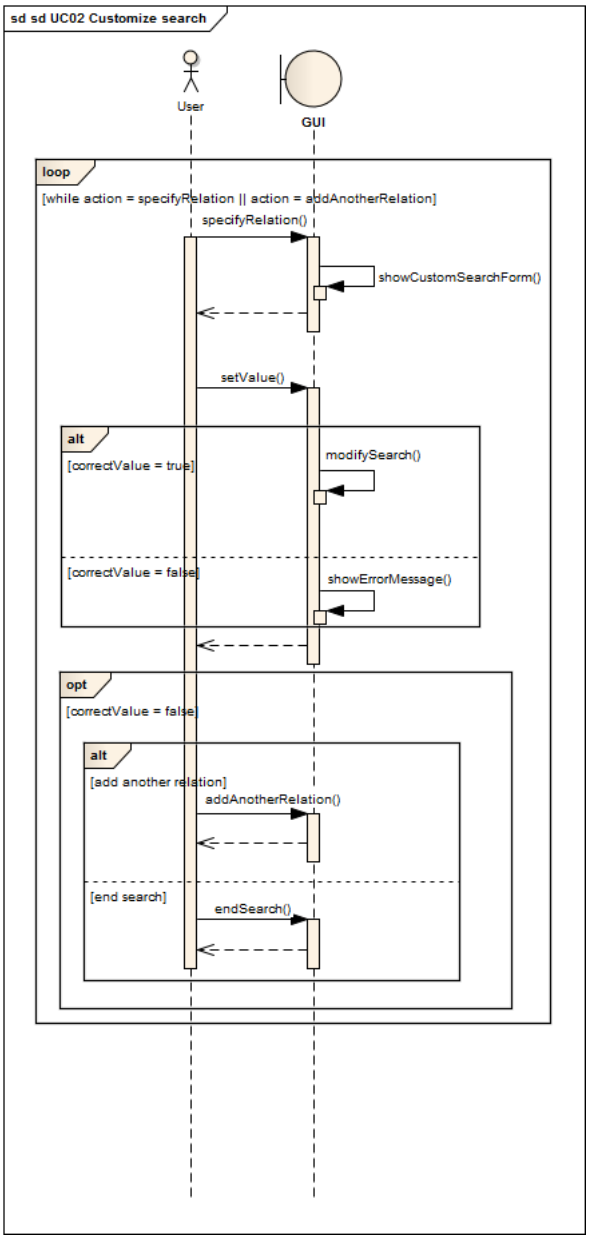
Autor: Peter Kajan
Identifikátor: UC02
Krátky opis: Používateľ upresní vyhľadávanie zadaním vzťahu hodnôt k objektu (viď tabuľka 4.3,F.1)
Predpoklady: žiadne
Dôsledky: žiadne
Účastníci: bežný používateľ
Priorita: 1 = vysoká
Hlavný tok:

1. Používateľ určí vzťah (viď tabuľky 4.3,F.1) , iný ako geografická pozícia
2. Systém zobrazí pole, do ktorého používateľ zadá hodnotu. Pre rôzne typy hodnôt (viď tabuľka 4.1) budú polia špecifické.
3. Používateľ zadá hodnotu alebo vymedzí jej rozsah (viď tabuľka 4.1)
   (a) intervalom v prípade, že hodnota je číselného typu
   (b) regulárnym výrazom v inom prípade
4. Systém overí:
   (a) typ zadanej hodnoty (viď tabuľka 4.3),
   (b) korektnosť zadanej hodnoty sa skontroluje podľa tabuľky (viď tabuľka 4.3)
5. Systém upresní vyhľadávanie podľa zadaných kritérií

Alternatívny tok: Zmena zadanej hodnoty
Aktivuje sa po kroku 4 hlavného toku, ak používateľ zadal hodnotu nesprávneho typu alebo neplatnú hodnotu (viď tabuľky 4.3,F.1).

1. Systém notifikuje používateľa a ponúkne možnosti
   (a) Zadať hodnotu znova
   (b) Ukončiť
2. Ak používateľ zvolí Zadať pozíciu znova, prípad použitia pokračuje krokom 3 hlavného toku
3. Ak používateľ zvolí Ukončiť, prípad použitia končí.

Na obrázku 4.3 je prípad použitia opísaný sekvenčným diagramom.



Obr. 4.3: Sekvenčný diagram pre UC02 Upravenie vyhľadávania špecifikovaním vzťahu

4.4.3 Upravenie vyhľadávania zadaním geografickej pozície

Autor: Peter Kajan
Identifikátor: UC03
Krátky opis: Používateľ upresní vyhľadávanie zadaním geografickej pozície objektu (viď tabuľka 4.3)
Predpoklady: žiadne
Dôsledky: žiadne

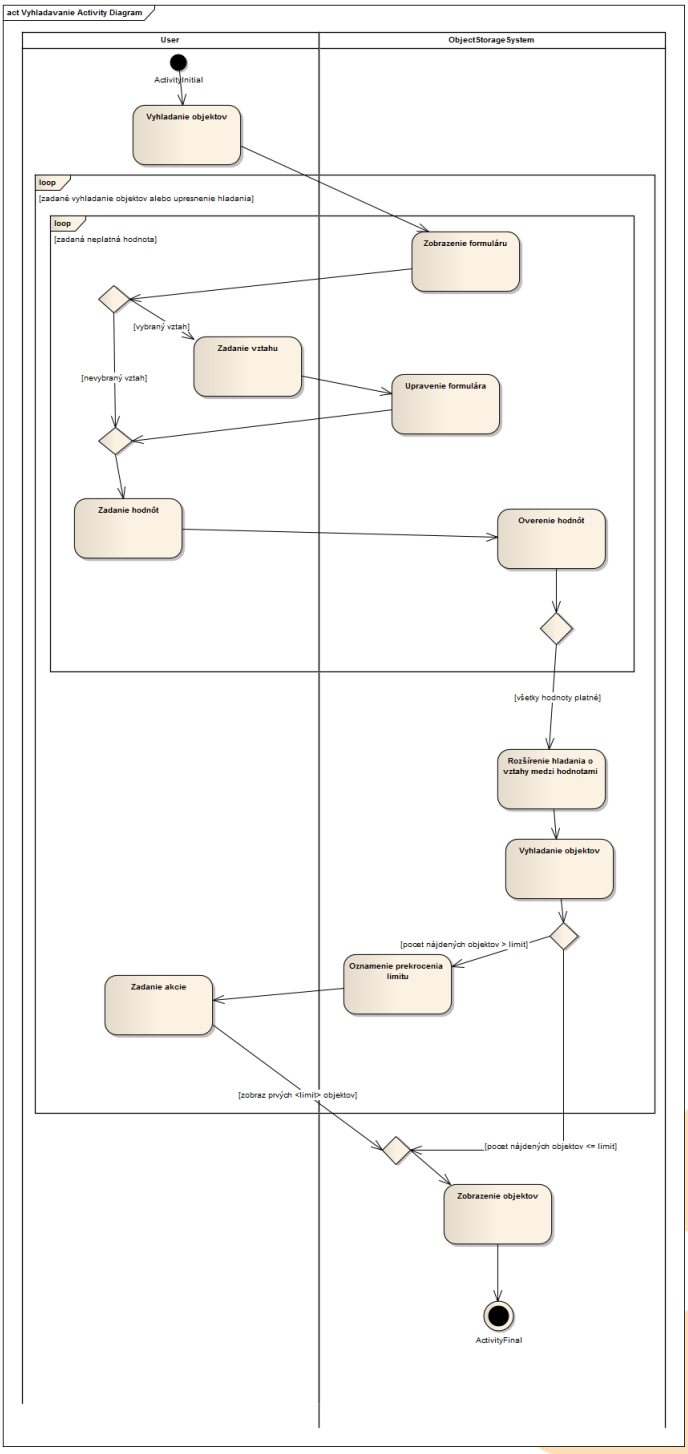
Účastníci: bežný používateľ
Priorita: 3 = nízka
Hlavný tok:

1. Používateľ zadá vzťah geografická pozícia objektu (viď tabuľka 4.3)
2. Systém zobrazí príslušné pole na zadanie geografickej pozície objektu
3. Používateľ môže zadať pozíciu nasledovne:
   (a) uvedením súradníc
   (b) zadaním zemepisného názvu
4. V prípade, že používateľ zadáva pozíciu uvedením súradníc, môže vymedziť hodnoty intervalom (viď tabuľka 4.1)
5. Systém overí:
   (a) typ zadanej hodnoty (viď tabuľka 4.3),
   (b) korektnosť zadanej hodnoty sa skontroluje podľa tabuľky (viď tabuľka 4.3)
6. Systém upresní vyhľadávanie podľa zadaných kritérií

Alternatívny tok: Zmena zadanaj pozície
Aktivuje sa po kroku 5 hlavného toku, ak používateľ zadal nekorektné súradnice alebo zemepisný názov, ktorý systém nerozpoznal

1. Systém notifikuje používateľa a ponúkne možnosti
   (a) Zadať pozíciu znova
   (b) Ukončiť
2. Ak používateľ zvolí Zadať pozíciu znova, prípad použitia pokračuje krokom 3 hlavného toku
3. Ak používateľ zvolí Ukončiť, prípad použitia končí.

Na obrázku 4.4 je zobrazený proces vyhľadávanie objektov pomocou diagramu aktivít. Proces sa týka prípadov použitia UC01 Vyhľadanie objektov, UC02 Upravenie vyhľadávania špecifikovaním vzťahu a UC03 Upravenie vyhľadávania zadaním geografickej pozície.



Obr. 4.4: Diagram aktivít pre vyhľadanie objektu

4.4.4 Odstránenie objektov

Autor: Peter Kajan
Identifikátor: UC04
Krátky opis: Používateľ odstráni vybrané objekty zo systému
Predpoklady: žiadne
Dôsledky: žiadne
Účastníci: bežný používateľ
Priorita: 1 = vysoká
Hlavný tok:

1. Používateľ zvolí objekty z výberu a zadá odstránenie objektov
2. Systém si vyžiada od používateľa potvrdenie odstránenia objektov
3. Ak používateľ potvrdí odstránenie objektov, systém odstráni objekty a k nim prislúchajúce metadáta
4. Inak prípad použitia končí

Alternatívny tok: Pokročilé odstránenie objektov
Aktivuje sa po kroku 1, ak aspoň k jednému odstraňovanému objektu má iný objekt vzťah (viď tabuľka 4.3).

1. Systém notifikuje používateľa a vyžiada si od neho potvrdenie
2. Ak používateľ potvrdí odstránenie, systém odstráni objekty a k nim prislúchajúce metadáta.
3. Ak používateľ nepotvrdí odstránenie, prípad použitia končí
4. Systém ponechá objekty spolu s metadátami, ktoré boli vo vzťahu k odstránenému objektu
5. Prípad použitia končí

4.4.5 Poskytnutie objektov a metadát

Autor: Peter Kajan
Identifikátor: UC05
Krátky opis: Systém poskytne používateľovi metadáta uložené v systéme
Predpoklady: žiadne
Dôsledky: žiadne
Účastníci: bežný používateľ
Priorita: 1 = vysoká
Hlavný tok:

1. Používateľ zvolí objekty z výberu a zadá poskytnutie objektov
2. Systém zobrazí formulár Poskytnutie objektov
3. Používateľ môže zadať priečinok pomocou štandardného dialógu, kam majú byť objekty uložené
4. Ak používateľ zadá priečinok, systém bude ukladať objekty do daného priečinku
5. Inak systém bude ukladať objekty do prednastaveného priečinku
6. Systém ponúkne používateľovi možnosť poskytnutia metadát objektov
7. Ak používateľ zadá poskytnutie metadát, systém bude ukladať metadáta spolu s objektami
8. Používateľ potvrdí poskytnutie objektov
9. Systém uloží objekty do zadaného priečinku, pričom informuje o priebehu ukladania pomocou Dialógu ukladania

Alternatívny tok: Úprava konfliktných názvov
Tok sa aktivuje po kroku 8, ak aspoň dva objekty majú rovnaký názov súboru alebo v priečinku sa nachádza súbor zhodný so súborom objektu.

1. Systém notifikuje používateľa a ponúkne mu nasledovné možnosti:
   (a) Premenovať konfliktné súbory objektov
   (b) Ukončiť
2. Ak používateľ zvolí Premenovať konfliktné súbory objektov, systém si vyžiada nové názvy konfliktných súbor.
3. Ak používateľ zvolí Ukončiť, prípad použitia končí
4. Používateľ zadá názvy
5. Prípad použitia pokračuje krokom 7 hlavného toku

4.4.6 Pridanie objektu

Autor: Peter Kajan
Identifikátor: UC07
Krátky opis: Používateľ pridá objekt do systému
Predpoklady: žiadne
Dôsledky: žiadne

Účastníci: bežný používateľ
Priorita: 1 = vysoká
Hlavný tok:

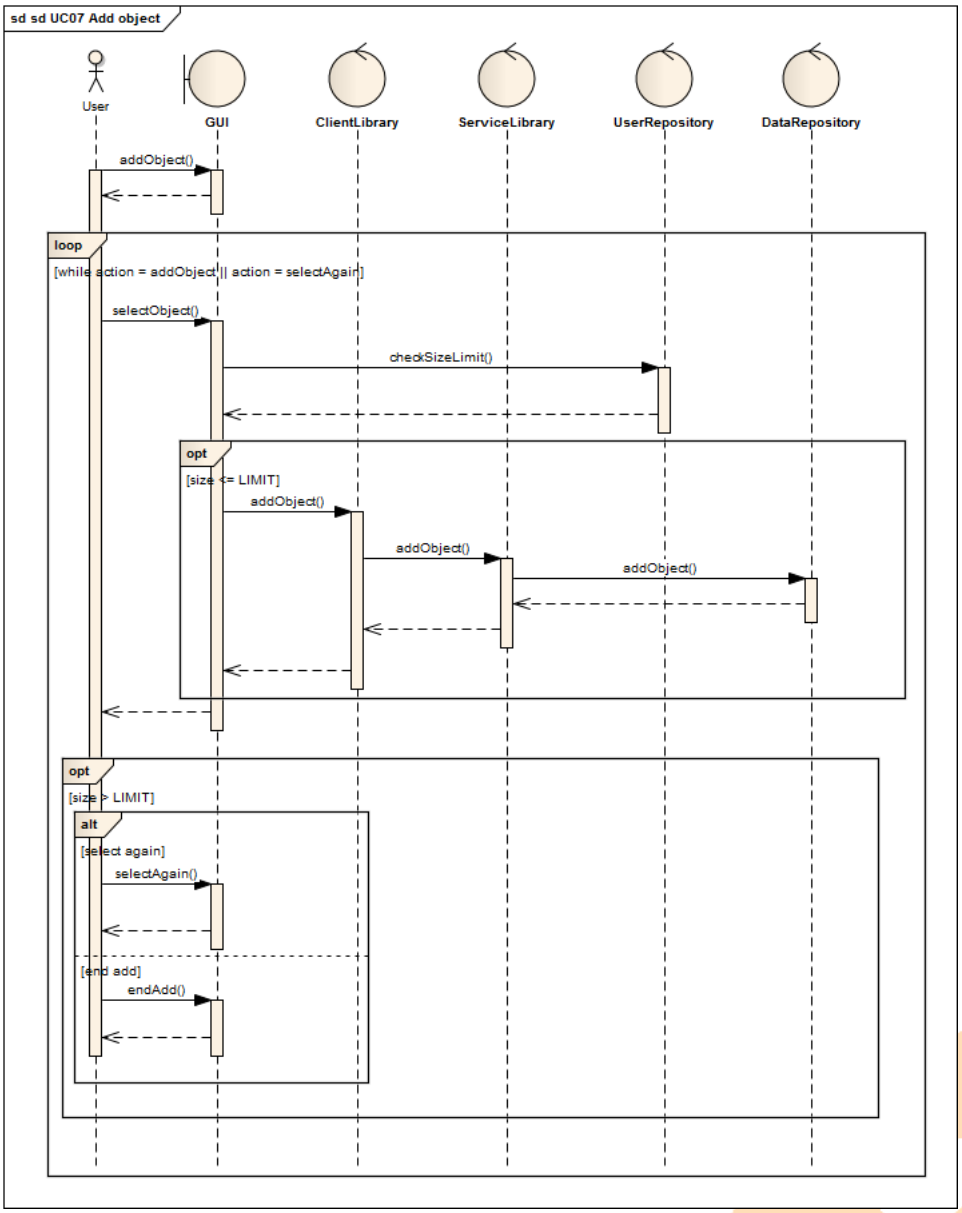
1. Používateľ zvolí pridanie objektu
2. Systém zobrazí formulár Pridanie objektu
3. Používateľ pomocou štandardného dialógu na výber súborov zadá objekt, ktorý chce pridať do systému
4. Systém uloží objekt

Alternatívny tok: Pridanie objektu pri prekročení dátového limitu
Aktivuje sa po kroku 3 hlavného toku, ak by uložením objektu v systéme by používateľ prekročil dátový limit jeho používateľského účtu

1. Systém notifikuje používateľa a ponúkne možnosti
   (a) Zadať objekt znova
   (b) Ukončiť
2. Ak používateľ zvolí Zadať objekt znova, prípad použitia pokračuje krokom 3 hlavného toku
3. Ak používateľ zvolí Ukončiť, prípad použitia končí.

Body rozšírenia:
Zobrazenie metadát: krok 4 hlavného toku

Na obrázku 4.5 je prípad použitia opísaný sekvenčným diagramom.



Obr. 4.5: Sekvenčný diagram UC07 Pridanie objektu

4.4.7 Zobrazenie metadát

Autor: Peter Kajan
Identifikátor: UC08
Krátky opis: Používateľovi sa zobrazia metadáta vybraného objektu v editovateľnom formulári
Predpoklady: vybraný objekt je uložený v systéme
Dôsledky: žiadne
Účastníci: bežný používateľ
Priorita: 1 = vysoká
Hlavný tok:

1. Používateľ zadá zobrazenie metadát vybraného objektu
2. Systém zobrazí editovateľný formulár so zoznamom dvojíc vzťah (viď tabuľka 4.3,F.1) a hodnota
3. Používateľ zadá ukončiť
4. Ak používateľ zmení aspoň jeden vzťah alebo hodnotu produktu, systém si vyžiada potvrdenie uloženia zmien
5. Ak používateľ potvrdí uloženie zmien, systém zmení vlastnosti produktu
6. Ak používateľ nepotvrdí uloženie zmien, systém neurobí žiadne zmeny

Body rozšírenia:
Pridanie metadát: krok 2 hlavného toku
Editovanie a odstránenie metadát: krok 2 hlavného toku

4.4.8 Pridanie metadát

Autor: Peter Kajan
Identifikátor: UC09
Krátky opis: Používateľovi pridá k vybranému objektu metadáta
Predpoklady: žiadne
Dôsledky: žiadne
Účastníci: bežný používateľ
Priorita: 1 = vysoká
Hlavný tok: Pridanie hodnoty

1. Používateľ pridá hodnotu
2. Systém určí typ vzťahu ako tag, a overí:

(a) typ zadanej hodnoty (viď tabuľka 4.3),

(b) korektnosť zadanej hodnoty sa skontroluje podľa tabuľky (viď tabuľka 4.3)

3. Systém upraví metadáta

Hlavný tok: Pridanie vzťahu

1. Používateľ pridá vzťah
2. Systém zobrazí pole, do ktorého používateľ zadá hodnotu. Pre rôzne typy hodnôt (viď tabuľka 4.1) budú polia špecifické.
3. Používateľ zadá hodnotu
4. Systém overí:

   (a) typ zadanej hodnoty (viď tabuľka 4.3,F.1),

   (b) korektnosť zadanej hodnoty sa skontroluje podľa tabuľky (viď tabuľka 4.3,F.1)

5. Systém upraví metadáta

Alternatívny tok: Oprava nesprávneho vstupu
Aktivuje sa po kroku 2 hlavného toku Pridanie hodnoty alebo po kroku 4 hlavného toku Pridanie vzťahu, ak používateľ zadal hodnotu nesprávneho typu alebo neplatnú hodnotu (viď tabuľka 4.3,F.1)

1. Systém notifikuje používateľa a ponúkne mu nasledovné možnosti:

   (a) Zadať novú hodnotu

   (b) Zmeniť vzťah

   (c) Ukončiť

2. Ak používateľ zvolí Zadať novú hodnotu, prípad použitia pokračuje krokom 1 hlavného toku Pridanie hodnoty.
3. Ak používateľ zvolí Zmeniť vzťah, aktivuje sa prípad použitia UC07 Editovanie a odstránenie metadát
4. Inak systém vráti vzťah a hodnotu do pôvodného stavu
5. Prípad použitia končí

4.4.9 Editovanie a odstránenie metadát

Autor: Peter Kajan
Identifikátor: UC10
Krátky opis: Používateľovi edituje alebo odstráni metadáta vybraného objektu
Predpoklady: žiadne
Dôsledky: žiadne
Účastníci: bežný používateľ
Priorita: 1 = vysoká
Hlavný tok: Editovanie hodnoty

1. Používateľ edituje hodnotu
2. Systém overí:
   (a) typ zadanej hodnoty (viď tabuľka 4.3,F.1),
   (b) korektnosť zadanej hodnoty sa skontroluje podľa tabuľky (viď tabuľka 4.3,F.1)
3. Systém upraví metadáta

Hlavný tok: Editovanie vzťahu

1. Používateľ edituje vzťah
2. Systém overí:
   (a) typ hodnoty prislúchajúcej editovanému vzťahu (viď tabuľka 4.3,F.1)
   (b) korektnosť zadanej hodnoty sa skontroluje podľa tabuľky (viď tabuľka 4.3,F.1)
3. Systém upraví metadáta

Hlavný tok: Odstránenie metadát

1. Používateľ odstráni hodnotu a vzťah
2. Systém odstráni vybranú hodnotu a vzťah z metadát objektu

Alternatívny tok: Oprava nesprávneho vstupu
Aktivuje sa po kroku 2 hlavného toku Editovanie hodnoty alebo po kroku 2 hlavného toku Editovanie vzťahu, ak používateľ zadal hodnotu nesprávneho typu alebo neplatnú hodnotu (viď tabuľka 4.3,F.1)

1. Systém notifikuje používateľa a ponúkne mu nasledovné možnosti:

(a) Zadať novú hodnotu

(b) Zmeniť vzťah

(c) Ukončiť

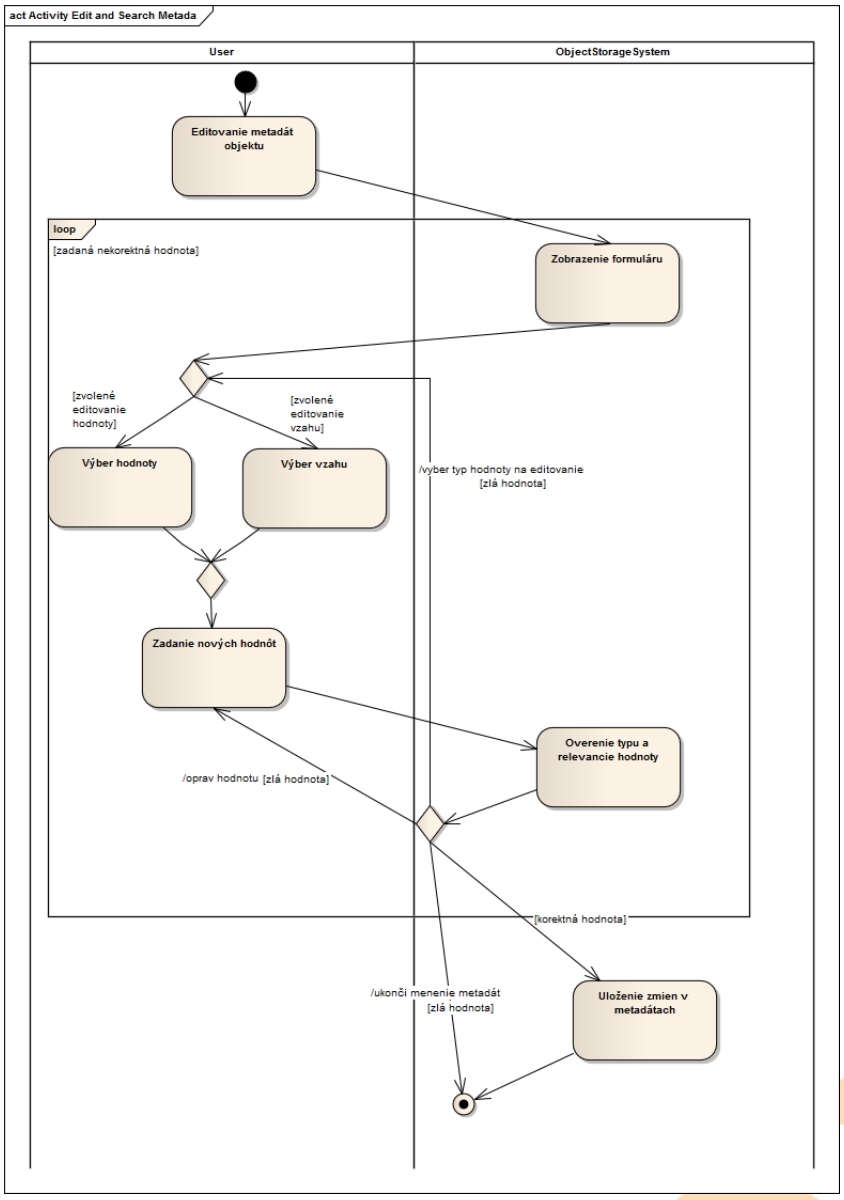
2. Ak používateľ zvolí Zadať novú hodnotu, prípad použitia pokračuje krokom 1 hlavného toku Editovanie hodnoty

3. Ak používateľ zvolí Zmeniť vzťah, prípad použitia pokračuje krokom 1 hlavného toku Editovanie vzťahu.

4. Inak systém vráti vzťah a hodnotu do pôvodného stavy

5. Prípad použitia končí

Na obr. 4.6 je zobrazený diagram aktivít opisujúci proces editácie metadát aj s alternatívnym tokom opravy vstupu.



Obr. 4.6: Diagram aktivít pre editáciu metadát

4.4.10 Zobrazenie vzťahov medzi hodnotami

Autor: Peter Kajan
Identifikátor: UC11
Krátky opis: Používateľovi sa zobrazí editovateľný formulár so vzťahmi medzi hodnotami, ktoré sú uložené v systéme
Predpoklady: žiadne
Dôsledky: žiadne
Účastníci: bežný používateľ
Priorita: 1 = vysoká
Hlavný tok:

1. Používateľ zadá zobrazenie vzťahov medzi hodnotami
2. Systém zobrazí editovateľný formulár so zoznamom trojíc hodnota, vzťah, hodnota
3. Používateľ zadá ukončiť
4. Ak používateľ zmení aspoň jednu vlastnosť produktu alebo pridá trojcu – hodnota, vzťah, hodnota, systém si vyžiada potvrdenie uloženia zmien
5. Ak používateľ potvrdí uloženie zmien, systém zmení metadáta objektu
6. Ak používateľ nepotvrdí uloženie zmien, systém neurobí žiadne zmeny

Body rozšírenia:
Editovanie a odstránenie vzťahov medzi hodnotami: krok 2 hlavného toku
Pridanie vzťahov medzi hodnotami: krok 2 hlavného toku

4.4.11 Editovanie a odstránenie vzťahov medzi hodnotami

Autor: Peter Kajan
Identifikátor: UC12
Krátky opis: Používateľovi edituje alebo odstráni vzťahy medzi hodnotami, ktoré sú uložené v systéme
Predpoklady: žiadne
Dôsledky: žiadne
Účastníci: bežný používateľ
Priorita: 1 = vysoká **Hlavný tok:** Editovanie hodnoty

1. Používateľ edituje hodnotu
2. Systém overí:

   (a) typ zadanej hodnoty (viď tabuľka 4.4),
   (b) korektnosť zadanej hodnoty sa skontroluje podľa tabuľky (viď tabuľka 4.4)
3. Systém upraví vzťahy medzi hodnotami

Hlavný tok: Editovanie vzťahu

1. Používateľ edituje vzťah
2. Systém overí:
   (a) typy hodnôt prislúchajúcich editovanému vzťahu (viď tabuľka 4.4)
   (b) korektnosť zadaných hodnôt sa skontroluje podľa tabuľky (viď tabuľka 4.4)
3. Systém upraví metadáta

Hlavný tok: Odstránenie metadát

1. Používateľ odstráni hodnoty a vzťah
2. Systém odstráni hodnoty a vzťah

Alternatívny tok: Oprava nesprávneho vstupu
Aktivuje sa po kroku 2 hlavného toku Editovanie hodnoty alebo po kroku 2 hlavného toku Editovanie vzťahu, ak používateľ zadal hodnotu nesprávneho typu alebo neplatnú hodnotu (viď tabuľka 4.4)

1. Systém notifikuje používateľa a ponúkne mu nasledovné možnosti:
   (a) Zadať novú hodnotu
   (b) Zmeniť vzťah
   (c) Ukončiť
2. Ak používateľ zvolí Zadať novú hodnotu, prípad použitia pokračuje krokom 1 hlavného toku Editovanie hodnoty
3. Ak používateľ zvolí Zmeniť vzťah, prípad použitia pokračuje krokom 1 hlavného toku Editovanie vzťahu.
4. Inak systém vráti vzťah a hodnotu do pôvodného stavu
5. Prípad použitia končí

4.4.12 Pridanie vzťahov medzi hodnotami

Autor: Lenka Baková
Identifikátor: UC13
Krátky opis: Používateľ pridá trojicu hodnota, vzťah, hodnota do formulára existujúcich trojíc
Predpoklady: žiadne
Dôsledky: žiadne
Účastníci: bežný používateľ
Priorita: 1 = vysoká
Hlavný tok:

1. Používateľ pridá trojicu – hodnota, vzťah,hodnota
2. Systém overí:
   (a) typ zadaných hodnôt (viď tabuľka 4.4),
   (b) korektnosť zadaných hodnôt sa skontroluje podľa tabuľky (viď tabuľka 4.4)
   (c) typy hodnôt prislúchajúcich pridanému vzťahu (viď tabuľka 4.4)
   (d) korektnosť vzťahu medzi hodnotami sa skontroluje podľa tabuľky(viď tabuľka 4.4)

Alternatívny tok: Oprava nesprávneho vstupu
Aktivuje sa po kroku 2 hlavného toku ak používateľ zadal hodnotu nesprávneho typu alebo neplatnú hodnotu (viď tabuľka 4.4)

1. Systém notifikuje používateľa a ponúkne mu nasledovné možnosti:
   (a) Zadať novú hodnotu
   (b) Zmeniť vzťah
   (c) Ukončiť
2. Ak používateľ zvolí Zadať novú hodnotu, prípad použitia pokračuje krokom 1 hlavného toku
3. Ak používateľ zvolí Zmeniť vzťah, prípad použitia pokračuje krokom 1 hlavného toku.
4. Inak systém vráti vzťah a hodnotu do pôvodného stavu
5. Prípad použitia končí

4.4.13 Prihlásenie používateľa

Autor: Zuzana Jalcová
Identifikátor: UC14
Krátky opis: Používateľ sa prihlási do systému.
Predpoklady: používateľ má vytvorený účet
Dôsledky: používateľ prihlásený do systému
Účastníci: bežný používateľ
Priorita: 1 = vysoká
Hlavný tok:

1. Systém zobrazí prihlasovací formulár
2. Používateľ zadá používateľské meno a heslo
3. Systém overí zadané údaje
4. Ak boli zadané údaje správne, systém prihlási používateľa do systému

Alternatívny tok: Nesprávne prihlasovacie meno alebo heslo
Tok sa aktivuje v kroku 3, ak používateľ zadá zlé používateľské meno alebo heslo

1. Systém informuje používateľa, že zadal nesprávne prihlasovacie údaje a ponúkne používateľovi možnosti:
   (a) Zmeniť používateľské meno alebo heslo
   (b) Ukončiť
2. Ak používateľ zvolí Zadať používateľské meno a heslo znova, prípad použitia pokračuje krokom 3 hlavného toku
3. Ak používateľ zvolí možnosť Ukončiť, prípad použitia končí

4.4.14 Upravenie účtu

Autor: Zuzana Jalcová
Identifikátor: UC15
Krátky opis: Používateľ pridá alebo upraví informácie(meno, heslo, e-mail) o používateľskom účte.
Predpoklady: žiadne
Dôsledky: žiadne
Účastníci: bežný používateľ
Priorita: 3 = nízka
Hlavný tok:

1. Používateľ zadá Upravenie účtu
2. Systém zobrazí predvyplnený editovateľný formulár Upravenie účtu
3. Používateľ môže zadať:
   (a) meno
   (b) heslo
   (c) e-mail
4. Ak používateľ zmení aspoň jeden údaj, systém si vyžiada potvrdenie zmien
5. Prípad použitia končí

Alternatívny tok: Potvrdenie hesla
Tok sa aktivuje ak používateľ zadal heslo

1. Systém si vyžiada napísanie heslo druhý krát a overí, či sa zhodujú
2. Ak sa heslá zhodujú, prípad použitia pokračuje krokom 4 hlavného toku
3. Ak sa heslá nezhodujú, systém vráti heslo do predchádzajúceho stavu
4. Prípad použitia pokračuje krokom 3 hlavného toku

4.4.15 Vytvorenie používateľského účtu

Autor: Zuzana Jalcová
Identifikátor: UC16
Krátky opis: Administrátor vytvorí používateľský účet
Predpoklady: žiadne
Dôsledky: vytvorený používateľský účet
Účastníci: administrátor
Priorita: 2 = stredná
Hlavný tok:

1. Administrátor zadá Vytvorenie používateľského účtu
2. Systém zobrazí formulár Vytvorenie používateľského účtu
3. Administrátor zadá meno a heslo používateľa
4. Systém overí zadané údaje
5. Ak boli zadané údaje správne, systém vytvorí používateľský účet

6. Prípad použitia končí

Alternatívny tok: Existujúce používateľské meno
Tok sa aktivuje v kroku 4, ak administrátor zadá už existujúce používateľské meno

1. Systém informuje administrátora, že zadal nesprávne používateľské meno a ponúkne administrátorovi možnosti:
   (a) Zmeniť používateľské meno
   (b) Ukončiť
2. Ak administrátor zvolí Zmeniť používateľské meno, prípad použitia pokračuje krokom 3 hlavného toku
3. Ak administrátor zvolí možnosť Ukončiť, prípad použitia končí

4.4.16 Resetovanie účtu

Autor: Zuzana Jalcová
Identifikátor: UC17
Krátky opis: Administrátor resetuje heslo používateľa.
Predpoklady: účet používateľa existuje
Dôsledky: nové heslo pre používateľa
Účastníci: administrátor
Priorita: 2 = stredná
Hlavný tok:

1. Administrátor zadá Resetovanie účtu
2. Systém požiada o zadanie používateľského mena, ktorému má byť resetované heslo
3. Administrátor zadá používateľské meno
4. Systém overí správnosť mena a vyžiada nové heslo
5. Administrátor zadá nové heslo
6. Systém zmení heslo používateľa
7. Prípad použitia končí

Alternatívny tok: Neexistujúce používateľské meno
Aktivuje sa po kroku 4 hlavného toku, ak administrátor zadá neexistujúce používateľské meno

1. Systém informuje administrátora, že zadal nesprávne používateľské meno a ponúkne administrátorovi možnosti:
   (a) Zmeniť používateľské meno
   (b) Ukončiť
2. Ak administrátor zvolí Zmeniť používateľské meno, prípad použitia pokračuje krokom 3 hlavného toku
3. Ak administrátor zvolí možnosť Ukončiť, prípad použitia končí

4.4.17 Zrušenie používateľského účtu

Autor: Zuzana Jalcová
Identifikátor: UC18
Krátky opis: Administrátor zruší používateľský účet. Odstráni všetky objekty používateľa a ich metadáta zo systému
Predpoklady: existujúci používateľský účet
Dôsledky: nové heslo pre používateľa
Účastníci: administrátor
Priorita: 2 = stredná
Hlavný tok:

1. Používateľ zvolí Zrušenie používateľského účtu
2. Systém vyžiada meno používateľského účtu, ktorý má byť zrušený
3. Používateľ zadá meno účtu
4. Systém vyžiada potvrdenie
5. Ak používateľ potvrdí zrušenie účtu, systém zruší používateľský účet a odstráni všetky objekty používateľa a ich metadáta
6. Prípad použitia končí

Alternatívny tok: Neexistujúce používateľské meno
Aktivuje sa po kroku 3 hlavného toku, ak administrátor zadá neexistujúce používateľské meno

1. Systém informuje administrátora, že zadal nesprávne používateľské meno a ponúkne administrátorovi možnosti:
   (a) Zmeniť používateľské meno
   (b) Ukončiť

2. Ak administrátor zvolí Zmeniť používateľské meno, prípad použitia pokračuje krokom 3 hlavného toku

3. Ak administrátor zvolí možnosť Ukončiť, prípad použitia končí

4.5 Nefunkcionálne požiadavky

Autor: Katarína Valaliková
V tejto časti dokumnetu sú uvedené nefunkcionálne požiadavky na vyvíjanú aplikáciu.

- *Dokumentácia* - úlohou je vytvoriť knižnicu. Táto knižnica bude mať definované rozhranie, ktoré bude dobre zdokumentované. Pomocou dobrej dokumentácie tak bude zaručené jednoduché použitie klientskej knižnice pre vytváranie vlastného používateľského rozhrania.

- *Rozšíriteľnosť* - systém bude navrhnutý tak, aby bol ľahko rozšíriteľný. Na základe deklaratívneho prístupu bude možné dodefinovať nové komponenty s pridanou funkcionalitou.

- *Softvér s otvoreným zdrojovým kódom* - vyvíjaný systém bude predstavovať softvér s otvoreným zdrojovým kódom, vydaný pod všeobecne verejnou licenciou.

- *Podpora operačných systémov* - systém bude navrhnutý a vyvíjaný tak, aby sa dosiahla jeho multiplatformovosť.

- *Prenositeľnosť* - prenositeľnosť systému bude zaručená dobrým zdokumentovaním poskytovaného rozhrania. Knižnica, ktorá bude vytváraná ako výsledok tímovej práce, bude použiteľná aplikáciami tretích strán.

- *Robustnosť* - aplikácia bude ošetrovať neštandardné situácie s cieľom zabrániť útokom na databázu a s cieľom predísť nefunkčnosti aplikácie pri zadaní neštandardných vstupov.

- *Škálovateľnosť* - aplikácia je navrhovaná tak, aby podporovala distribuovanosť, je bezstavová, čím sa zaručí jej škálovateľnosť "po celom internete".

Kapitola 5

Návrh riešenia

Vyvíjaná aplikácia má slúžiť ako úložisko rôznych druhov objektov. Požiadavky, ktoré má aplikácia spĺňať boli spomenuté už skôr, v kapitole 4. V tejto časti dokumentu sa nachádza návrh aplikácie a opis navrhovaných technológií potrebných pre jej tvorbu. Pri navrhovaní aplikácie sa vychádzalo z funckionálnych a nefunkcionálnych požiadaviek a zo stanovených cieľov.

5.1 Architektúra aplikácie

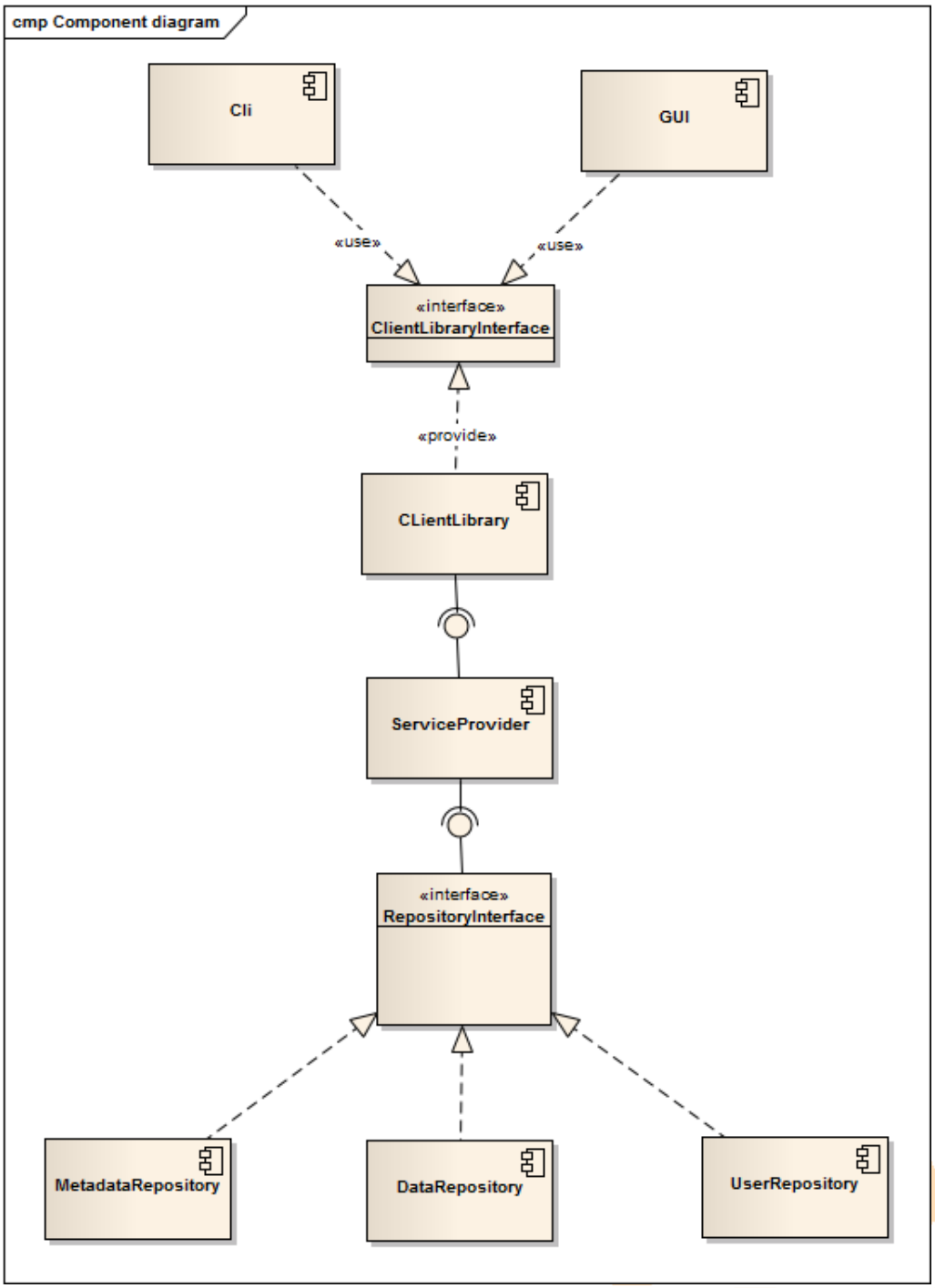
Autor: Katarína Valaliková

Na tvorbu aplikácie bola zvolená klient-server architektúra. Na strane servera sa bude nachádzať webová aplikácia poskytujúca služby. Stranu klienta bude predstavovať knižnica s verejným rozhraním využívajúcim webové služby. Na strane klienta bude táto knižnica využívaná prostredníctvom príkazového riadku. Komunikácia medzi klientom a serverom bude prostredníctvom HTTP protokolu a bude využité REST rozhranie. Na obrázku 5.1 sa nachádza diagram komponentov, z ktorých bude aplikácia pozostávať.

Aplikácia je rozdelená na šesť komponentov (obr. 5.1). Komponenty sú slabo previazané, čím bude zaručená vysoká modularita aplikácie. Ďalej sa nachádza opis jednotlivých komponentov.

User Repository

Komponent User Repository bude zodpovedný za funkcie týkajúce sa správy používateľov. Úlohou komponentu bude prihlasovanie existujúcich používateľov. V prípade,



Obr. 5.1: Diagram komponentov

že používatelia v databáze nebudú existovať, majú možnosť zaregistrovať sa. Po prihlásení sa používateľovi zobrazia funkcie, na ktoré má právo. Rovnako na základe údajov o prihlásenej osobe budú sprístupnené objekty, na ktoré má používateľ právo. Ako riešenie pre databázu používateľov navrhujeme použiť LDAP.

Data Repository

Komponent Data Repostiory zahŕňa funkcie potrebné na prácu s dátami, ktoré chceme v úložisku ukladať. Typy podporovaných dát (objektov) boli spomenuté v kapitole 4.3.1. Komponent Data Repository má zaručiť najmä ukladanie, vymazávanie a vyhľadávanie dát (objektov) vo vzdialenom úložisku. Ako úložisko navrhujeme použiť súborový systém. Ak súborový systém nebude stačiť, bude vďaka modulárnosti nahraditeľný databázou.

Ukladanie dát bude riešené serializáciou objektu. Funkcia zastrešujúca pridávanie objektov bude mať ako vstupný parameter objekt, ktorý má byť uložený na s[borovom systéme. Výstupom bude miesto uloženia objektu. Toto miesto bude identifikátorom objektu. Na miesta uloženia objektu (URI) môžu byť k objektu priradené metadáta.

Pri vyhľadávaní dát bude potrebné vedieť miesto uloženia konkrétneho objektu, ktorý má byť nájdený. Toto miesto uloženia sa získa z metadát. Vstupným parametrom teda je miesto uloženia objektu a výstupom vyhľadávania sú dopytované objekty.

Vymazávať objekty bude možné na základe ich identifikátorov. Ak k vymazávanému objektu existujú aj metadáta alebo iné previazanie, budú vymazané aj tie. Vstupným parametrom funkcie bude identifikátor objektu, ktorý má byť vymazaný. Výsledkom je vymazanie objektu a všetkých súvisiacich metadát. Pri neúspešnom vyhľadávaní bude vyhlásená výnimka.

Metadata Repository

Komponent Metadata Repository obsahuje funkcie, potrebné na prácu s metadátami. Bude obsahovať funkcie ako pridanie, vymazanie, upravovanie a vyhľadávanie metadát. Funkcionálne požiadavky opisujúce tento komponent sa nachádzajú v kapitole 4.

Metadáta navrhujeme ukladať vo formáte RDF (Resource Description Framework). Tento formát umožňuje ukladanie metadát vo forme objekt-predpoklad-subjekt, nazývanom aj ako tzv. triplet. „Objekt“ reprezentuje to, čo bude fyzicky uložené v databáze (fotka, video, dokument, ...). „Subjekt“ môže to byť objekt dostupný cez data repository (fotka, video, dokument), ale môže to byť takisto hodnota vyjadrujúca dátum, čas, geografickú polohu, meno, priezvisko atď. „Predpoklad“ vyjadruje vzťah medzi „Objekt“-om a „Subjekt“-om. „Predpoklady“ budú v istej miere vopred našpecifikované. Pri vkladaní nových tagov k objektu si používateľ zvolí vzťah z existujúcich. Metadáta vo forme tripletov budú ukladané v databáze podporujúcej ukladanie tripletov.

Repository interface

Repository interface definuje rozhranie pre komponenty User Repository, Data Repository a Metadata Repository. Rozhranie, ktoré bude poskytovať Repository Interface bude rovnaké ako rozhranie, ktoré bude poskytovať klientská knižnica. Zaručí sa tým

vyššia modulárnosť a rovnako bude jednoduché nahradiť klientsku časť aplikácie za webovú aplikáciu.

Service Provider
Komponent Service Provider predstavuje webovú aplikáciu. Úlohou komponentu Service Provider je zabezpečiť prepojenie s komponentami User Repository, Metadata Repository a Data Repository. Toto prepojenie bude zabezpečené deklaratívnym spôsobom. Komponent Service Provider je dôležitý aj z ďalšieho dôvodu, a to poskytnutie webových služieb.

Webové služby zhŕňajú funkcionalitu úložiska dát a metadát a slúžia na komunikáciu medzi klientom a serverom. Pre webové služby navrhujeme použiť rozhranie REST a na komunikáciu medzi klientom a serverom použitie protokolu HTTP. REST rozhranie poskytuje rozhranie pre štyri základné typy operácii (CRUD proces), ktoré môžu byť použité na posielanie dát a metadát medzi klientom a serverom.

Client Library

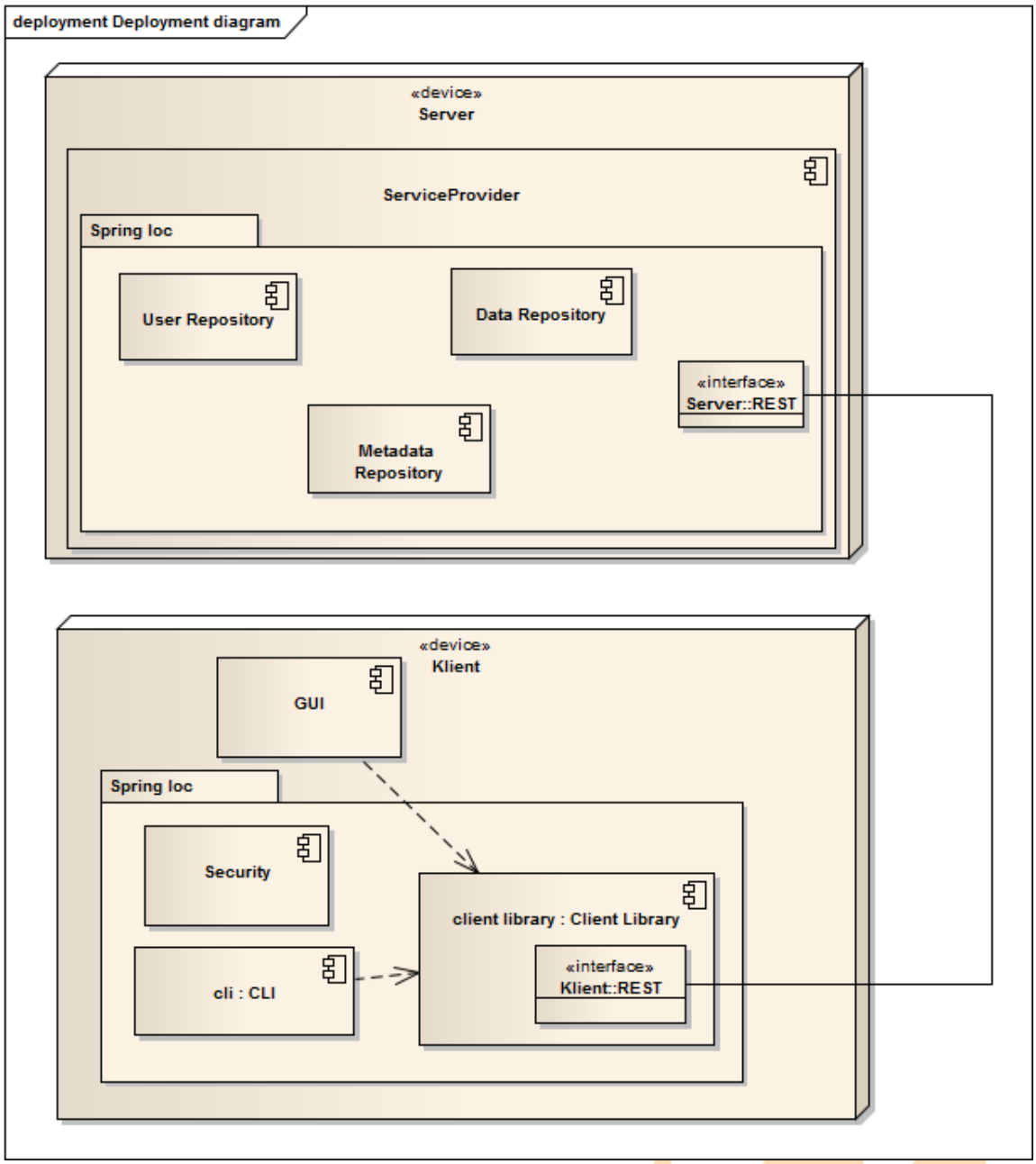
Komponent Client Library opisuje klientskú časť aplikácie. Tento komponent bude poskytovať verejné rozhranie, ktoré budú využívať klientské aplikácie. Úlohou tohto komponentu je zabezpečiť komunikáciu so stranou servera (komponent Service Provider). Komunikácia bude prebiehať prostredníctvom služieb s RESTful rozhraním. Na základe tejto komunikácie bude prebiehať práca s objektami a metadátami. Client Library bude mať rovnaké rozhranie ako API.

CLI

Komponent CLI predstavuje testovaciu aplikáciu. Bude reprezentovať používateľské rozhranie prostredníctvom príkazového riadku.

Komponenty sú podľa funkcie, ktorú plnia, zaradené buď na stranu klienta alebo servera. Rozmiestnenie jednotlivých komponentov zobrazuje diagram nasadenia (obr. 5.1).

Na strane servera sa nachádzajú komponenty Service Provider, Metadata Repository a Data Reposiotry. Aby sme zaručili vysokú modulárnosť medzi jednotlivými komponentami, navrhujeme prepojiť ich deklaratívnym spôsobom. Na strane klienta sa nachádzajú dva komponenty, a to Client Library a CLI. Na konštrukciu klientskej strany sa bude používať tiež deklaratívny spôsob.



Obr. 5.2: Diagram nasadenia

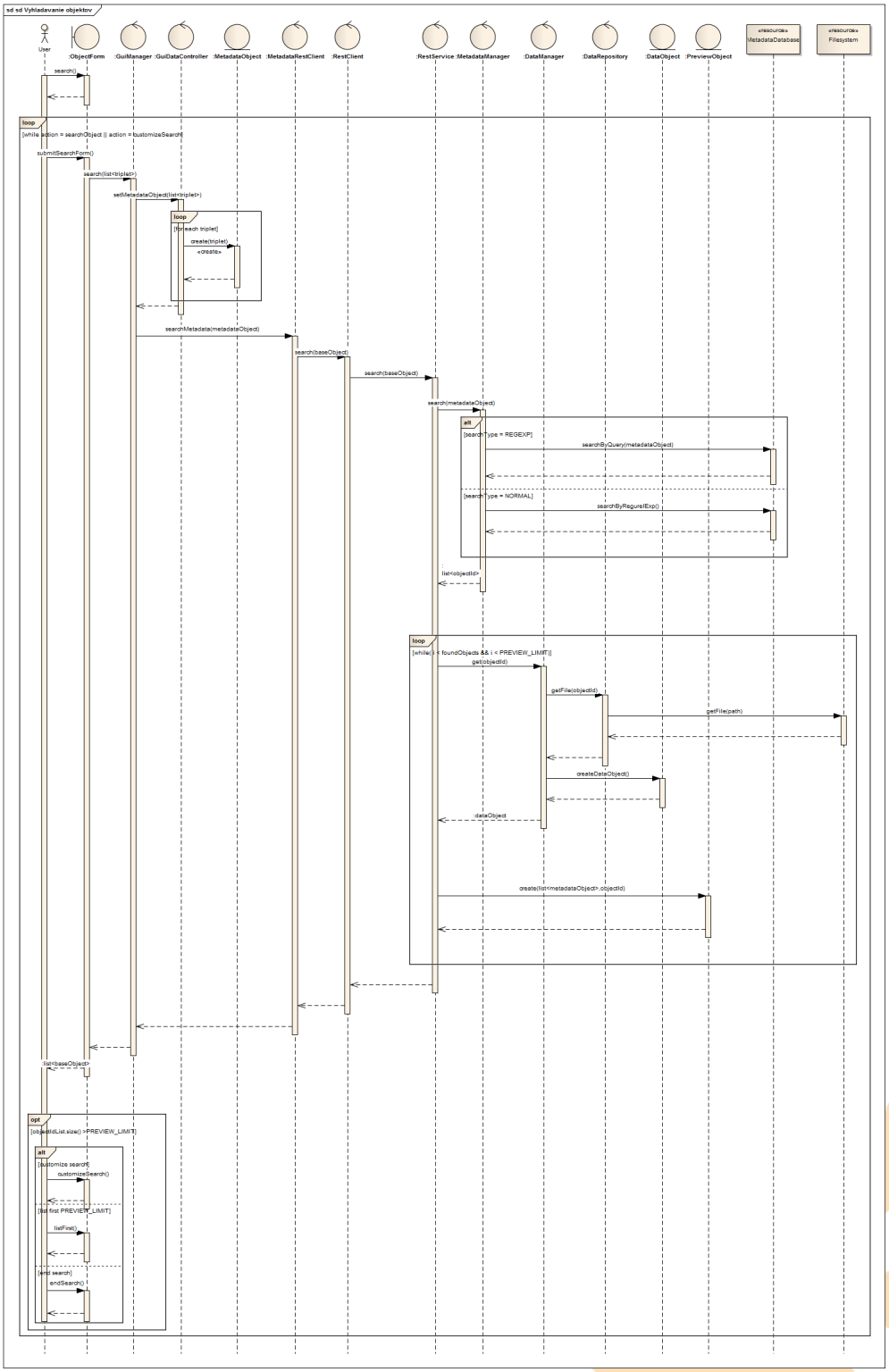
5.2 Podrobný návrh systému

V tejto kapitole sa nachádza podrobný návrh systému. Kapitola má nasledovnú štruktúru. V časti 5.2.1 boli vybrané sekvenčné diagramy na úrovni komponentov upravené na úroveň tried. Z nich boli vytvorené diagramy tried. V časti 5.2.2 sa nachádza diagram tried s vyznačenými balíkmi a v časti 5.2.3 s vyznačenými vzormi. Ďalej je opísaný stavový priestor vybraných tried (kapitola 5.2.4) a nakoniec je uvedený návrh grafického používateľského rozhrania a CLI aplikácie.

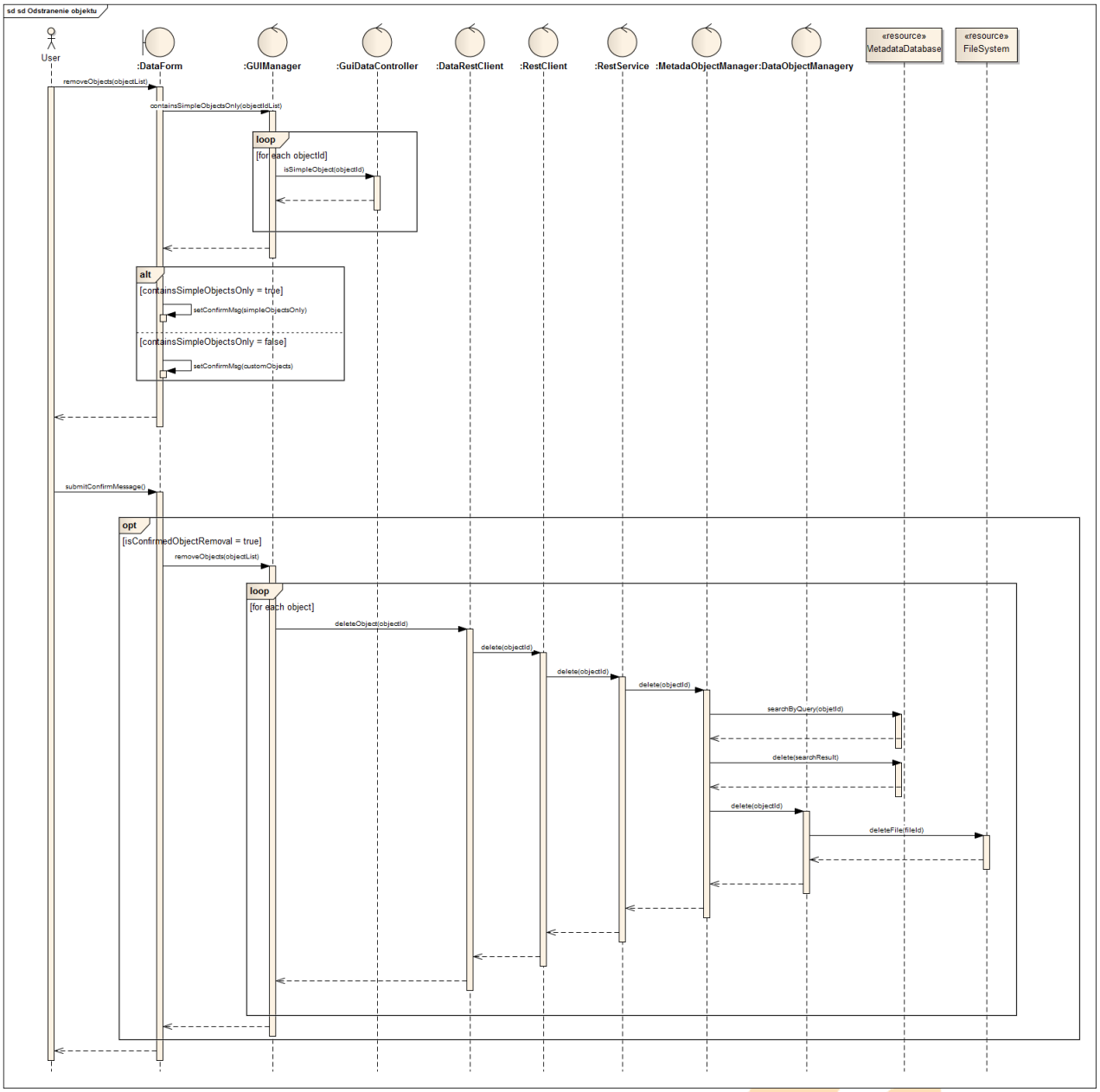
5.2.1 Sekvenčné diagramy na úrovni tried pre vybrané prípady použitia

V tejto časti sa nachádzajú sekvenčné diagramy pre prípady použitia

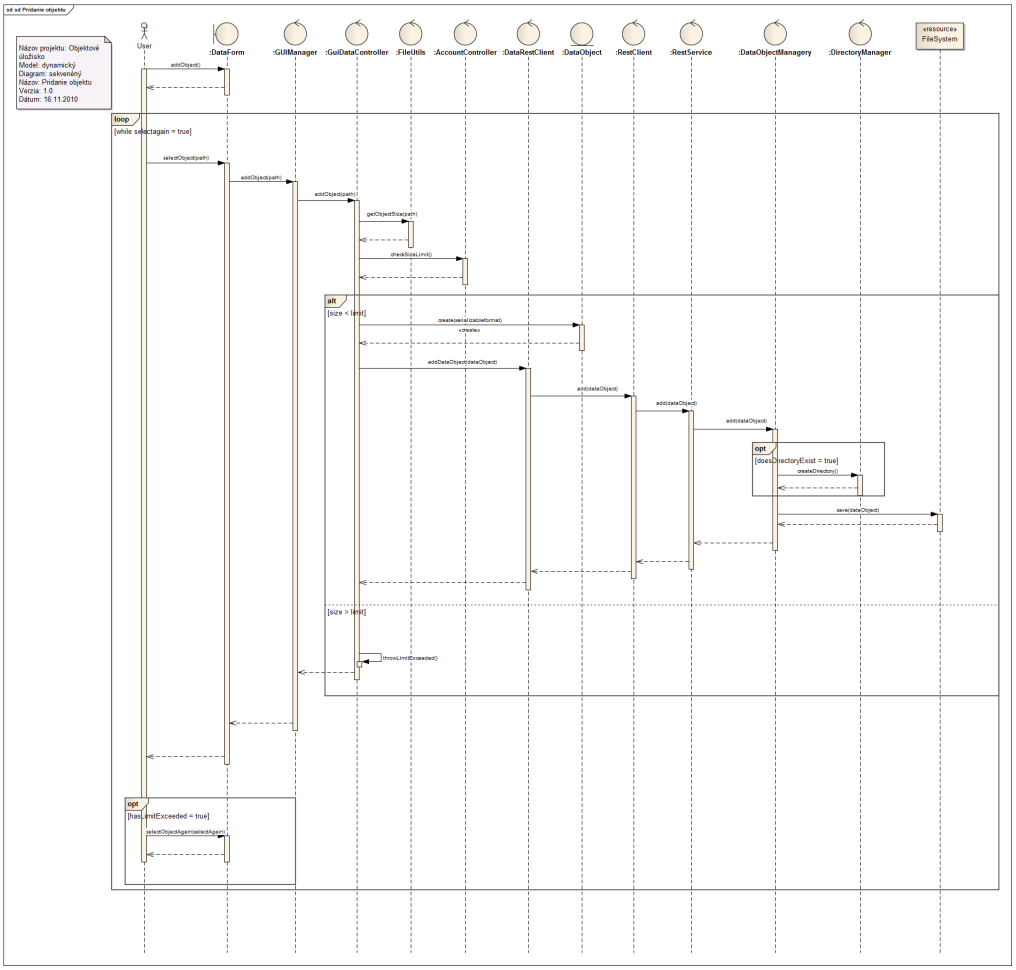
- UC01 Vyhľadanie objektov (obr. 5.3),
- UC04 Odstránenie objektov (obr. 5.2.1),
- UC07 Pridanie objektu (obr. 5.2.1) a
- UC16 Vytvorenie účtu (obr. 5.2.1).



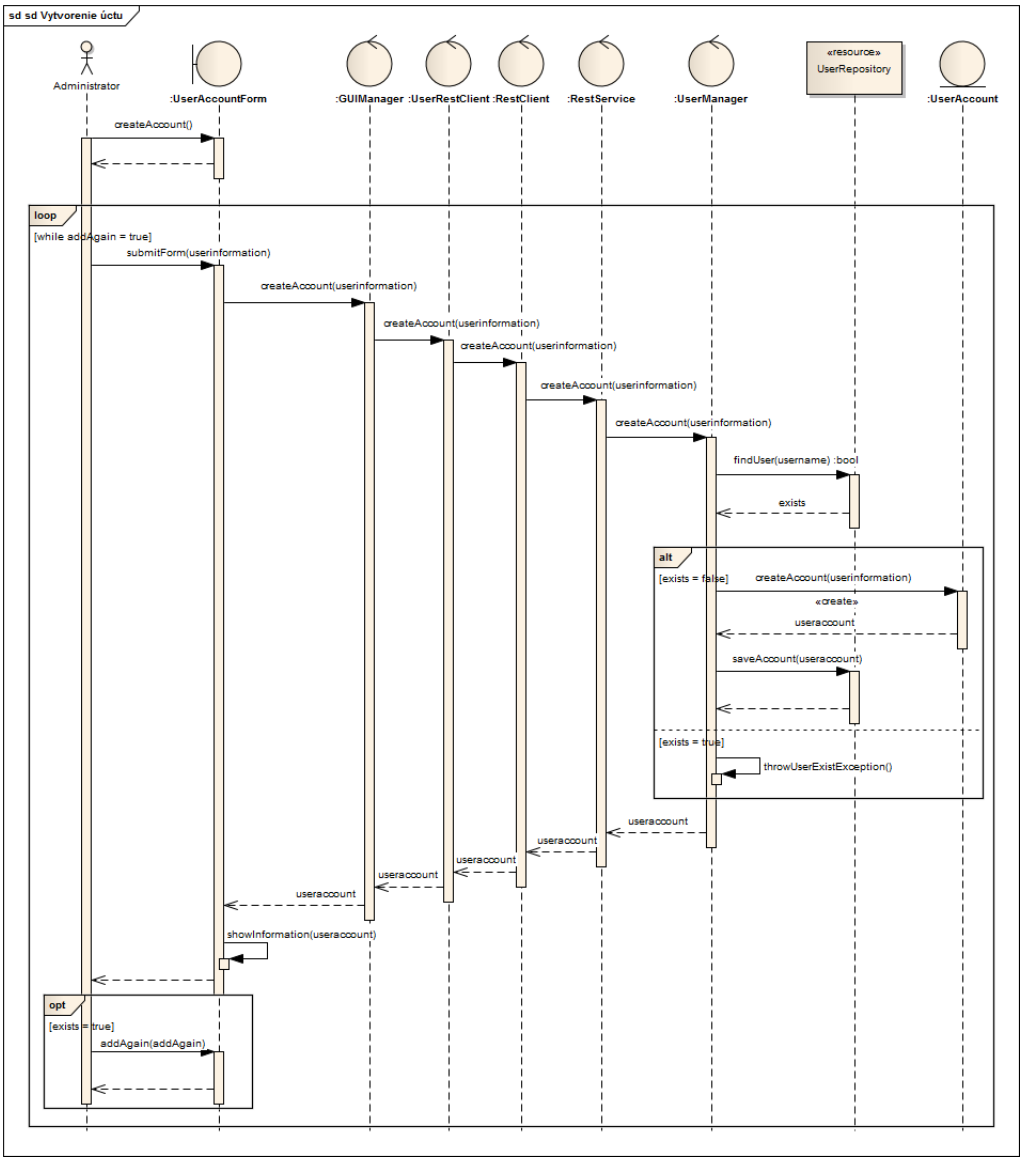
Obr. 5.3: Podrobný sekvenčný diagram k UC01 Vyhľadanie objektov



Obr. 5.4: Podrobný sekvenčný diagram k UC04 Odstranenie objektu



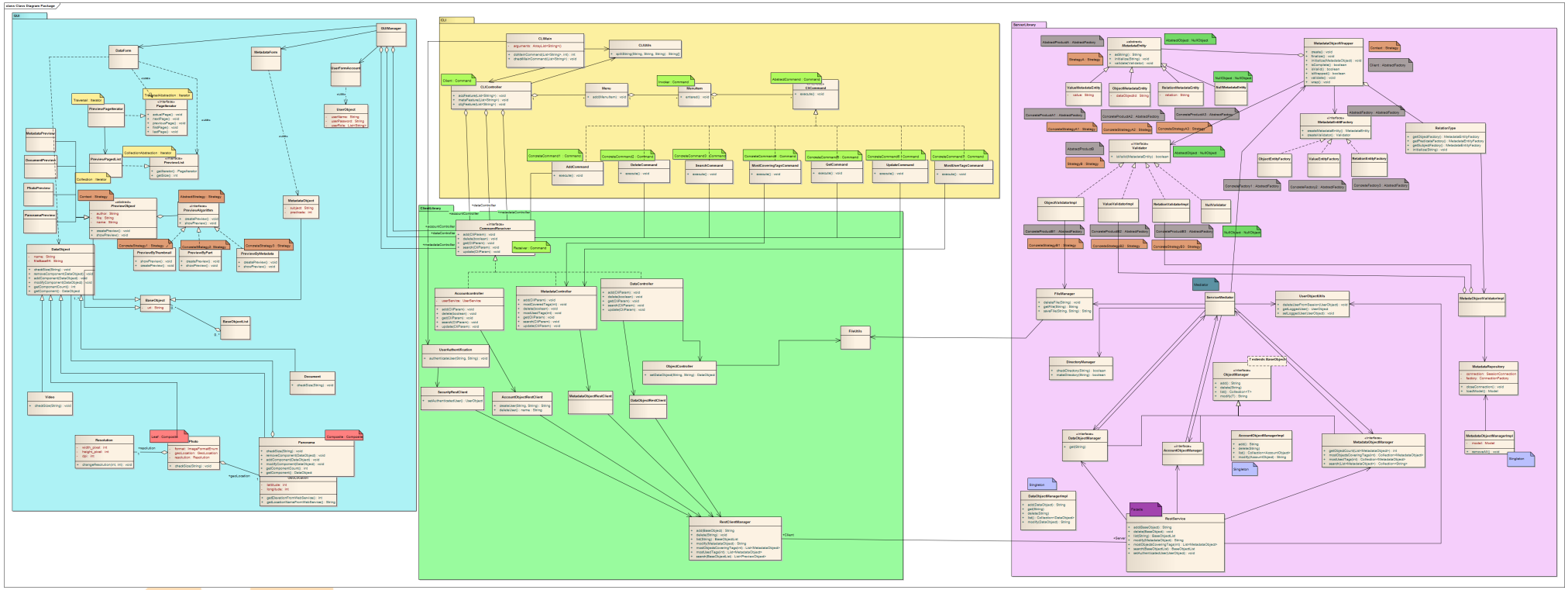
Obr. 5.5: Podrobný sekvenčný diagram k UC07 Pridanie objektu



Obr. 5.6: Podrobný sekvenčný diagram k UC16 Vytvorenie účtu

5.2.2 Diagram tried s vyznačenými balíkmi

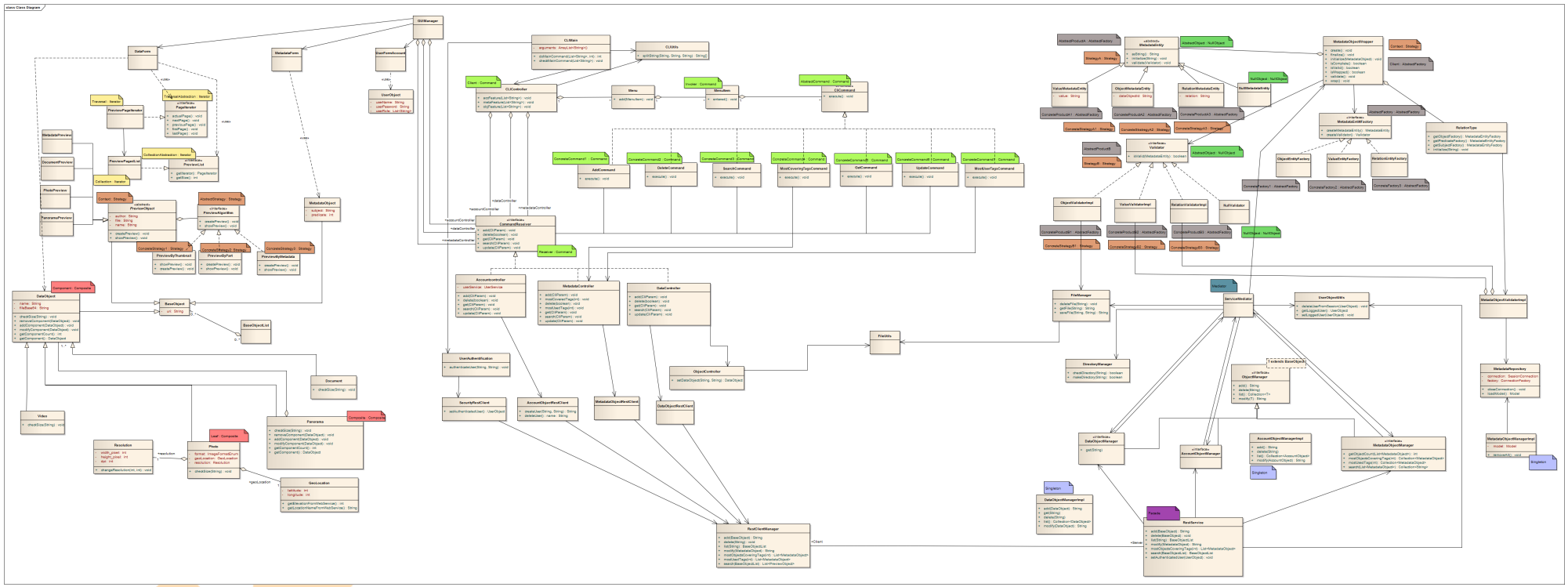
Na nasledovnom obrázku (obr. 5.2.2) je zobrazený diagram tried s vyznačenými balíkmi. Balíky vyplývajú z komponentov špecifikovaných v časti 5.1.



Obr. 5.7: Diagram tried s vyznačenými balíkmi

5.2.3 Diagram tried s vyznačenými návrhovými vzormi

V tejto časti je znázornená architektúra aplikácie pomocou diagramu tried (viď obr. 5.2.3). V diagrame tried sú vyznačené jednotlivé návrhové vzory. Ich podrobnejší opis sa nachádza v prílohe G.

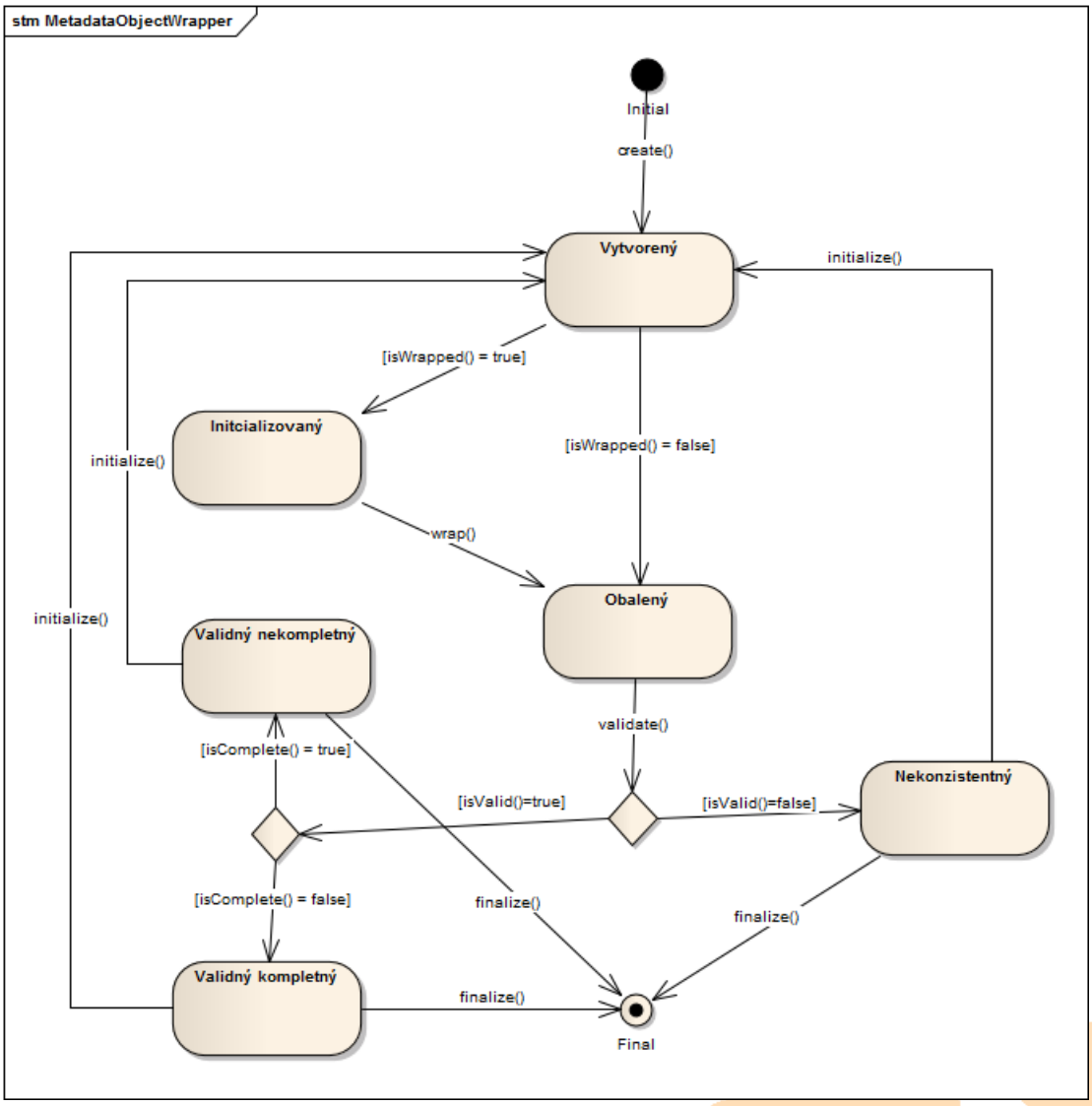


Obr. 5.8: Diagram tried

5.2.4 Stavový diagram pre triedu MetadataObjectWrapper

Autor: Peter Kajan

MetadataObjectWrapper je trieda, ktorá slúži na spracovanie objektov metadát reprezentovaných vo forme tripletov. Používa sa predovšetkým na ich validáciu a konverziu na tvar, ktorý je ukladaný do databázy. Jej stavový priestor je znázornený na obr. 5.9 pomocou stavového diagramu.



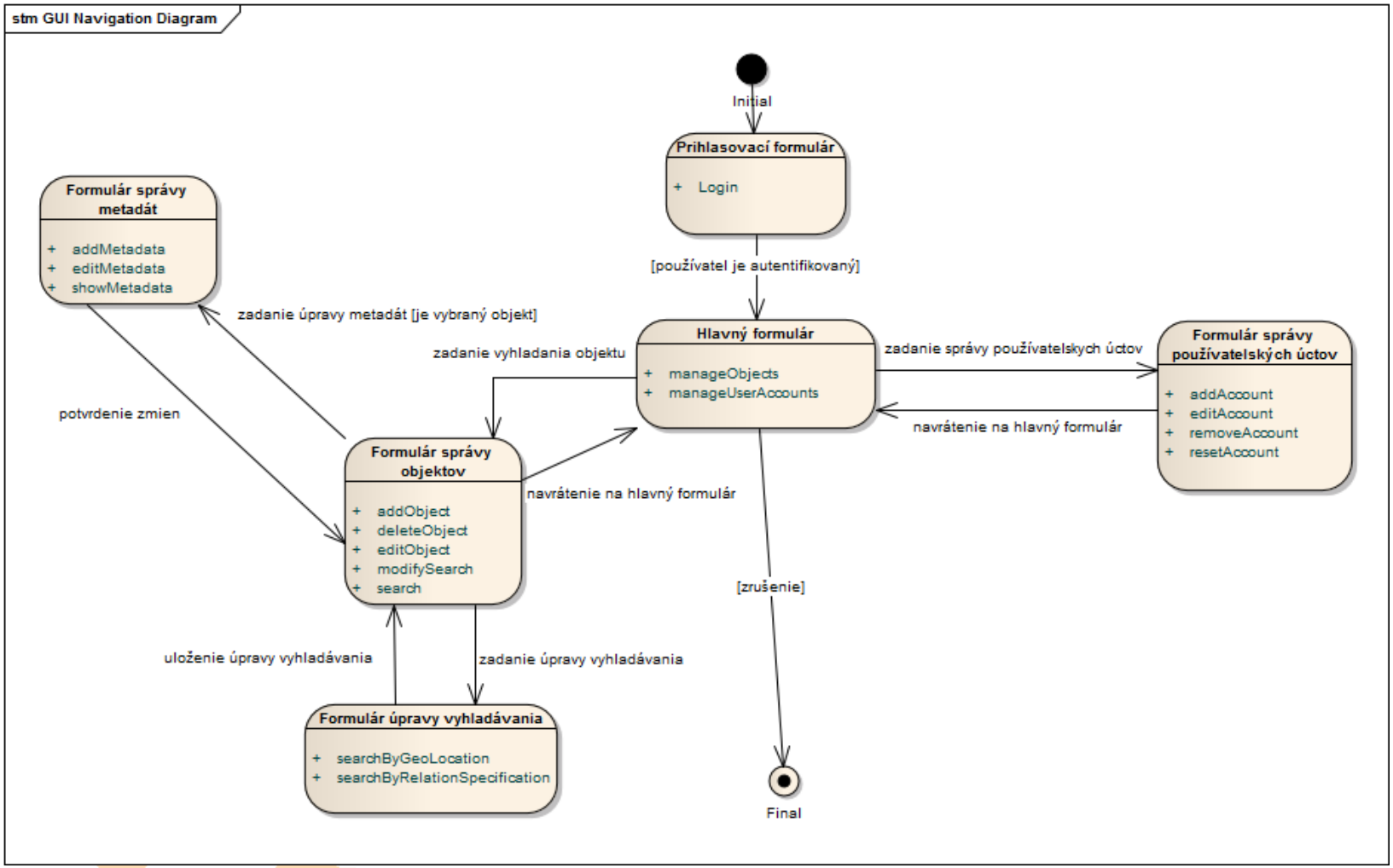
Obr. 5.9: Stavový diagram pre triedu MetadataObjectWrapper

Stav *Vytvorený* predstavuje pseudostav, v ktorom sa objekt (inštancia triedy *MetadataObjectWrapper*) nebude nikdy nachádzať. Pri vstupe do tohto stavu sa vykonajú určené akcie. Potom objekt podľa podmienky prechádza do ďalšieho stavu. Ďalej

metódy začínajúce na *check* (napr. *checkWrapped()*) predstavujú zložitejšie operácie zisťujúce stav objektu, metódy začínajúce na *is* (napr. *isWrapped()*) prestavujú jednoduché zistenie hodnoty daného atribútu.

5.3 Návrh Gui

Grafické používateľské rozhranie aplikácie pozostáva z formulárov. V navigačnom diagrame (obr 5.3) sú zobrazené podstatné formuláre, prechody medzi nimi a akcie, ktoré jednotlivé formuláre poskytujú.



Obr. 5.10: Navigačný diagram pre GUI

5.4 Návrh CLI aplikácie

Autor: Lenka Baková
Návrh CLI aplikácie vychádza z prípadov použitia a pokrýva celú funkcionalitu, ktorú prípady použitia opisujú. Obmedzenia funkčnosti pre jednotlivých používateľov pristupujúcich k aplikácii, tiež vychádzajú z prípadov použitia a z hráčov, pre ktorých sú prípady použidia určené.

Príkazy v CLI aplikácii sa budú zadávať formou príkaz –prepínač –prepínač ... Prepínače môžu byť povinné a nepovinné. Povinný je prepínač vtedy, keď sa jeho neuvedenie za príkazom považuje za chybu. V príkladoch sú povinné prepínače sú znázornené v [] nepovinné v {}. Prepínače môžu mať aj atribúty, ktoré ich konkretizujú. Každý prepínač má presne definované koľko atribútov obsahuje a čo znamenajú. Atribúty bude možné zadávať aj formou regulárneho výrazu aj formou intervalu nasledovne - *dolná_hranica ..horná_hranica* . Atribúty sú v nasledujúcom texte písané kurzívou.

Aplikácia bude funkčná v dvoch módoch – v interaktívnom a neinteraktívnom.

5.4.1 Opis interaktívneho módu

Interaktívny mód bude umožnovať prihlásenie, interaktívne zadávanie ľubovoľných príkazov a následné odhlásenie.

Spustenie aplikácie a prihlásenie sa

`STFU [-l` *`login`* `] [-pw` *`heslo`*`]`
Týmto príkazom sa bude aplikácia spúšťať a zároveň aj prihlási užívateľa. Ak nebude existovať na serveri konto so zadanými údajmi, aplikácia sa nespustí. STFU je skratka z názvu nášho úložiska - Super Truper Fajnové Úložisko.
súvisiace UC: UC14 Prihlásenie používateľa

Odhlásenie sa

`logout`
Napísaním a potvrdením príkazu bude realizované odhlásenie zo systému a zároveň ukončenie aplikácie.

Operácie s účtom

Všetky operácie v aplikácii súvisiace s účtom sa budú realizovať príkazom acc. Jednotlivé funkcie sa budú volať na základe rôznych prepínačov tohto príkazu.

Vytvorenie účtu

`acc [-new] [-nl` *nový_login* `] [-npw` *nové_heslo*`]`
-new vytvorí nový účet s menom *nový_login* a heslom *nové_heslo*
súvisiace UC: UC16 Vytvorenie používateľského účtu

Výpis informácií o účte

`acc`
Príkaz bez prepínačov vypíše informácie o účte – meno a email.

Upravenie účtu

`acc [-ch] [-nl` *nový_login* `|-npw` *nové_heslo* `|-e` *nový_email* `]`
-ch bude slúžiť na zmenu parametrov používateľského účtu
-nl zmení login na *nový_login*
-npw zmení heslo na *nové_heslo*
-e zmení email na *nový_email*
súvisiace UC: UC15 Upravenie účtu

Resetovanie účtu

`acc [-re` *meno1 meno2...*`]`
-re resetuje účet s menom *meno1 meno2...*
súvisiace UC: UC17 Resetovanie účtu

Zrušenie účtu

`acc [-rm` *meno1 meno2...*`]`
-rm vymaže účty s menom *meno1 meno2...*
súvisiace UC: UC18 Zrušenie používateľského účtu

Práca s objektami

Všetky operácie v aplikácii súvisiace s objektami sa budú realizovať príkazom obj. Jednotlivé funkcie sa budú volať na základe rôznych prepínačov tohto príkazu.

Vyhľadávanie objektov

`obj [-search] {-t` *tag1 tag2...*`} {-r` *vztah-hodnota* `| -ri` *vztah-interval_hodnot* `} {-g [ [-s` *suradnice* `| -si` *interval_suradnic*`] | [-z` *zem_nazov* `] ]} {-size` *pocet*`} {-prev}`

-search nájde objekty a zobrazí ich ID
-t vyhľadá objekty obsahujúce tagy *tag1 tag2...*
-r vyhľadá objekty obsahujúce vzťah *vztah* k hodnote *hodnota*. Podrobný rozpis vzťahov sa nachádza v kapitole 4.3.2 Metadáta.
-ri vyhľadá objekty obsahujúce vzťah *vztah* k hodnote patriacej do interavlu *interval_hodnot* . Podrobný rozpis vzťahov sa nachádza v kapitole 4.3.2 Metadáta.
-g vyhľadá objekty podľa geologickej lokácie
-s vyhľadáva podľa súradníc *suradnice*
-si vyhľadáva podľa súradníc patricich do interval *interval_suradnic*
-z vyhľadá podľa zemepisného názvu *zem_nazov*
-size zabezpečí, že sa z nájdených objektov vypíše prvých *veľkosť* ID, ak ich je viac, upozorní, že ich je viac
-prev uloží do defaultnej zložky náhľady nájdených objektov
súvisiace UC: UC01 Vyhľadanie objektu, UC02 Upravenie vyhľadávania špocifikovaním vzťahu, UC03 Upravenie vyhľadávania zadaním geografickej pozície

Poskytnutie objektov
`obj [-give IDobjektu1 IDobjektu2 ... ] {-p path} {-m} {-prev}`
-give skopíruje do počítača vybrané objekty *IDobjektu1 IDobjektu2 ...*
-p skopíruje objekty do počítača do priečinku udaného cestou *path*, ak nie je prepínač použitý, objekty sa skopírujú do defaultnej zložky
-m skopíruje s objektami aj ich metadáta
-prev uloží do defaultnej zložky náhľady nájdených objektov
súvisiace UC: UC05 Poskytnutie objektov a metadát

Odstránenie objektu
`obj [-rm IDobjektu1 IDobjektu2...]`
-rm odstráni objekty *IDobjektu1 IDobjektu2...*
súvisiace UC: UC04 Odstránenie objektov

Pridanie objektu
`obj [-add meno1 meno2...] {-p path}`
-add bude pridávať na server objekty *meno1 meno2...*
-p vezme objekty z počítača z priečinka udaného cestou *path*, ak nie je prepínač použitý, objekty vezmú z defaultnej zložky
súvisiace UC: UC07 Pridanie objektu

Výpis uložených objektov
`obj [-list]`
-list : vypíše všetky objekty, ktoré má používateľ uložené na serveri

súvisiace UC: UC01 Vyhľadanie objektov

Práca s metadátami

Metadáta sú reprezentované trpletmi hodnota1-vzťah-hodnota2, pričom hodnoty sú spojené vzťahom, hodnota1 môže byť aj IDobjektu, ku ktorému sa vzťahom viaže hodnota2, táto tiež môže byť IDobjektu.
Všetky operácie v aplikácii súvisiace s metadátami sa budú realizovať príkazom meta. Jednotlivé funkcie sa budú volať na základe rôznych prepínačov tohto príkazu.

Odstránenie metadát

`meta [-rm` *`hodnota1-vztah1-hodnota1 hodnota2-vztah2-hodnota2 ...`*`]`
-rm odstráni metatáta určené tripletmi *hodnota1-vztah1-hodnota1 hodnota2-vztah2-hodnota2 ...* Podrobný rozpis vzťahov sa nachádza v kapitole 4.3.2 Metadáta.
súvisiace UC: UC10 Editovanie a odstránenie metadát, UC12 Editovanie a odstránenie vzťahov medzi hodnotami

Pridanie metadát

`meta [-add` *`hodnota1-vztah1-hodnota1 hodnota2-vztah2-hodnota2 ...`*`]`
-add pridá metatáta určené tripletmi *hodnota1-vztah1-hodnota1 hodnota2-vztah2-hodnota2 ...* Podrobný rozpis vzťahov sa nachádza v kapitole 4.3.2 Metadáta.
súvisiace UC: UC09 Pridanie metadát, UC12 Editovanie a odstránenie vzťahov medzi hodnotami, UC13 Pridanie vzťahov medzi tagmi

Vyhľadávnie metadát

`meta [-search] {-id` *`IDobjektu1 IDobjektu2...`*`}`
-search vyhľadá metadáta a vypíše ich na terminál, samotný search bez prepínačov bude vypisovať metadáta vzťahujúce sa k používateľovi vo forme hodnota-vztah-hodnota
-id vyhľadá metadáta vzťahujúce sa k objektom *IDobjektu1 IDobjektu2...* vo forme *IDobjektu1*-vztah-hodnota , *IDobjektu2*-vztah-hodnota ...
súvisiace UC: UC08 Zobrazenie metadát, UC11 Zobrazenie vzťahov medzi hodnotami

Výpis metadát

`meta [-list]`
-list : vypíše všetky metadáta, čo má používateľ uložené pri svojich objektoch
súvisiace UC: UC08 Zobrazenie metadát

5.4.2 Opis neinteraktívneho módu

Mód bude umožnovať prihlásenie, zadanie ľubovoľných príkazov a následné odhlásenie v jednom príkaze.
`STFU [-l login ] [-pw heslo] [-command "príkaz s prepínačmi"]`
-*"príkaz s prepínačmi"* – je ľubovoľný príkaz z uvedených v kapitole 5.4.1 Opis interaktívneho módu
súvisiace UC: všetky – podľa povahy príkazu a jeho prepínačov

Kapitola 6

Opis riešenia

Autor: Katarína Valaliková

6.1 Prostriedky použité na vývoj aplikácie

6.1.1 Maven

Na buildovanie aplikácie je použitý nástroj Maven. Je založený na projektovom objektovom modeli (POM), okrem buildovanie aplikácie môže manažovať aj reportovanie a dokumentovanie. Ciele Maven-u sú:

- zjednodušiť proces buildovania,
- zabezpečiť jednotný build systém,
- zabezpečiť kvalitné informácie o projekte,
- zabezpečiť usmernenia pre osvedčené postupy vývoja
- dovoliť transparentnú migráciu nových funkcií.

6.1.2 Java + Netbeans

Ako programovací jazyk bol zvolený jazyk Java. Tento jazyk bol zvolený z dôvodu, že väčšina členov tímu má s ním najväčšie skúsenosti a vyvíjaná aplikácia má byť multiplatformová, čo Java zaručí. Za vývojové prostredie bol zvolený nástroj Netbeans, ktorý napomáha pri vývoji webových aplikácií využívajúcich platformu Java, ale aj JavaFX, Ajax, Javasripc a pod.

6.1.3 Apache Tomcat

Vyvíjaná aplikácia je testovaná na webovom kontajneri Apache Tomcat, ktorý implementuje špecifikácie Servlet 2.5 a JavaServer Pages 2.1 z Java Community Process a zahŕňa množstvo prídavných funkcií, ktoré ho robia použiteľným pre vývoj a nasadenie webových aplikácií a webových služieb.

6.2 Technológie použité pri vývoji aplikácie

6.2.1 Args4j

Pre vyvíjanú knižnicu je vytváraná aplikácia spúšťaná pomocou príkazového riadku. Na parsovanie argumentov zadaných prostredníctvom tejto aplikácie je použitá knižnica ars4j, ktorá parsovanie uľahčuje. Použitím anotácií sa stáva parsovanie argumentov z príkazového riadku veľmi jednoduché, je možné ľahko a rýchlo generovať príklad použitia argumentov, generovať HTML/XML dokument opisujúci všetky dostupné argumenty do dokumentácie a plne podporuje lokalizáciu.

6.2.2 Spring Security

Technológia Spring Security je použitá na zabezpečenie aplikácie. Pri tvorbe webových aplikácií je využitie Spring Security na zabezpečenie štandardom. Pomocou neho jeho vykonaná autentifikácia a autorizácia používateľov a rovnako aj zabezpečenie na úrovni metód a prístupu k URL adresám. Spring Security ponúka množstvo implementácií pre overovanie používateľov, napríklad voči databáze či LDAP-u a takisto poskytuje implementácie filtrov, ktoré chránia prístupy k URL adresám podľa používateľských rolí a ich práv.

6.2.3 Spring REST

Na komunikáciu medzi klientskou knižnicou a serverom je použité rozhranie REST a na implementáciu komunikácie je zvolená knižnica Spring. Na strane klienta je použítá technológia Spring MVC, kde pomocou anotácií sú namapované služby využívané klientskou knižnicou. V klientskej knižnici je na pripojenie sa k službám využitá RestTemplate implementovaná v knižnici Spring.
Spring „REST“ je pomerne nová technológia, ktorá poskytuje jednoduchú prácu so službami využívajúcimi REST rozhranie. Klientka časť je dodaná prostredníctvom RestTemplate a jej výstupy môžu byť ďalej ľahsko spracované.

6.2.4 Jena

Metadáta sú ukladané vo formáte RDF. Na prácu s metadátami uloženými vo formáte RDF je použitá knižnice Jena. Jena zahŕňa:

- RDF API
- OWL API
- čítanie a zapisovanie RDF do RDF/XML, N3 a N-Triplets
- SPARQL dopytovací jazyk.

Jena je pomerne starý projekt (stálosť, vyzretosť) s podporou od HP. Na oficiálnej stránke projektu je možné nájsť množstvo dokumentácií a návodov pre prácu s Jenou narozdiel od knižnice JRDF, ktorej dokumentácia pokrýva málo poskytovanej funkcionality. Narozdiel od Seasamu zasa Jena poskytuje lepšiu podporu pre prácu s RDF súbormi, napríklad na stránke projektu Seasame nie je zmienka o podpore SPARQL.

6.3 Porovnanie RDF databáz

Autor: Peter Kajan

V nasledovnom texte sú analyzované databázy na uchovávanie RDF. Uvažované boli databázy:

- Jena (http://jena.hpl.hp.com/)
- Sesame (http://www.openrdf.org/)
- Virtuoso (http://www.openlinksw.com/virtuoso/)
- Mulgara (http://www.mulgara.org/)
- 4Store (http://4store.org/)

Jena

Je to súbor knižníc pre pácu s RDF (podrobnosti v kapitole Opis použitých technológii). Obsahuje dve databázy: SDB a TDB. SDB je postavané na SQL databáze. TDB je natívna RDF databáza, bez podpory tranzakčnosti. Najväčšou výhodou knižníc Jena je poskytované API pre prácu s RDF, hlavnou nevýhodou je zlý výkon databáz.

Databáza	Výhody	Nevýhody
Jena	Poskytovaná funkcionalita, čistá Java	Výkon podporovaných databáz
Sesame	Podpora a dokumentácia, čistá Java	Škálovateľnosť
Virtuoso	Škálovateľnosť, prepojenie so Sesame a Jena API	Poskytovaná funkcionalita
Mulgara	Tranzakčnosť, prepojenie so Sesame a Jena API, čistá Java	Poskytovaná funkcionalita, podpora a dokumentácia

Sesame

Zahŕňa API pre prácu s RDF, ktoré je v porovaní s knižnicami Jena menej rozsiahle. Rovnako obsahuje databázu založenú na SQL, no ponúka aj výkonnú natívnu databázu Sesame Native. Jej nevýhoda je škálovateľnost len do 70 miliónov tripletov. Virtuoso Virtuoso je natívna databáza pre ukladanie RDF tripletov. Je škálovateľná do viac ako 1 miliardy tripletov. Podporuje prepojenie s Jena a Sesame API. Nevýhodou je jej v porovaní s predošlími databázami chudobnejšie API a obmedzená podpora OWL.

Mulgara

Mulgara je natívna RDF databáza s podporou SPARQL. Umožňuje prepojenie s Jena API. Jej hlavnou výhodou je podpora tranzakčnosti. Je napísaná v jazyku Java. Nevýhodou je slabšia podpora a dokumentácia.

4 store

Je natívna RDF databáza s obmedzenou podporou Java. Z toho dôvodu je pre nás neprijateľná.

V tabuľke 6.1 sa nachádza zhrnutie výhod a nevýhod uvažovaných databáz.

V projekte sme sa rozhodli využiť pre prácu s RDF súbor knižníc Jena z dôvodu poskytovanej funkcionality a podpory jazyku Java. Databázy podporované knižnicami Jena sme sa rozhodli nahradiť databázov Mulgara. V porovaní s databázou Virtusio podporuje tranzakčnosť. Podobná kombinácia je odporúčana aj v článku [1].

Zdroje: [1] http://www.bioontology.org/wiki/images/6/6a/Triple_Stores.pdf

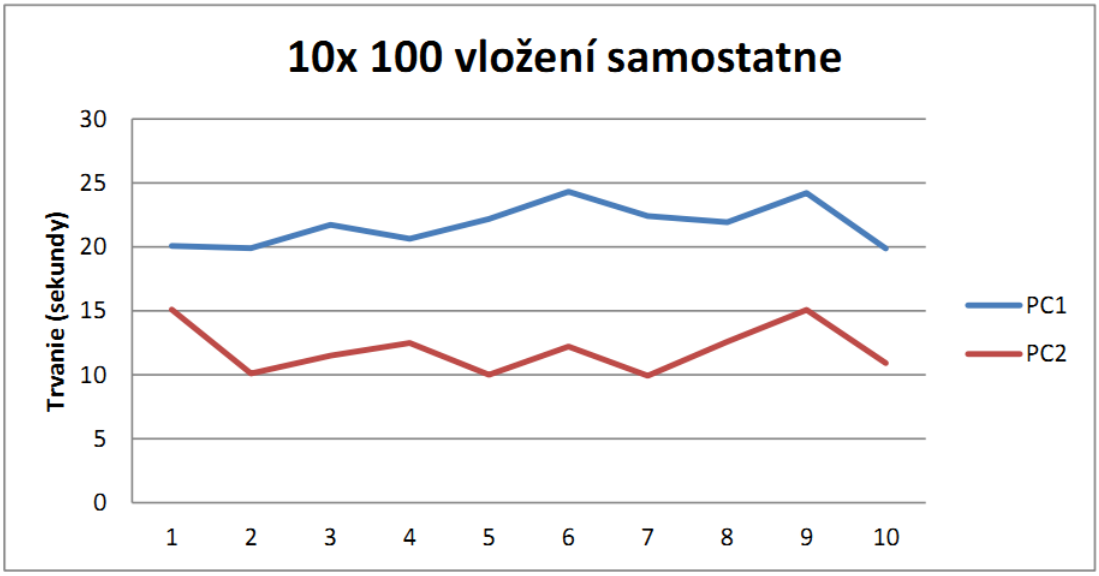
6.4 Mulgara

Autor: Marek Uhlár

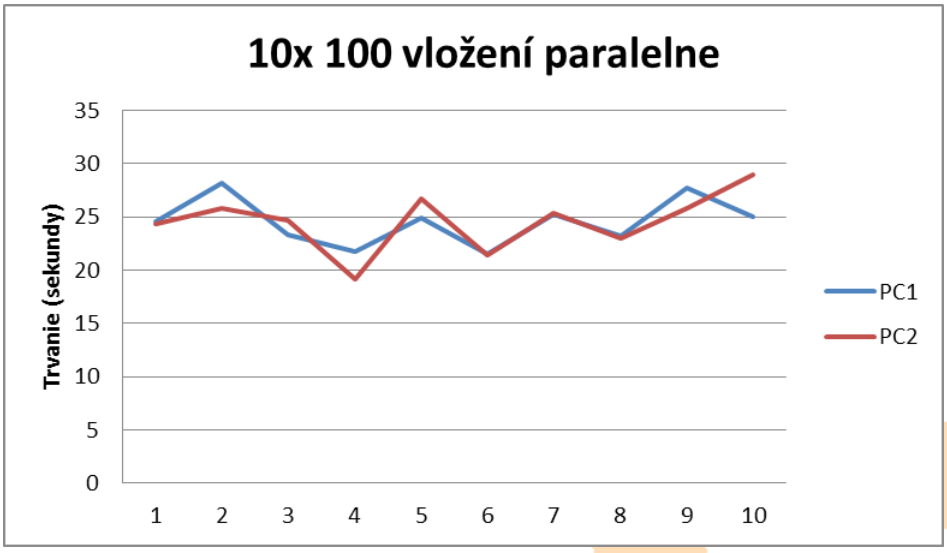
Na uchovávanie tripletov sme výberovým procesom na základe viacerých parametrov zvolili databázu Mulgara. Keďže naša aplikácia je určená pre uchovávanie veľkého množstva dát v tomto formáte a zároveň schopnosť vykonávať svoju činnosť pod záťažou, bolo potrebné overiť viaceré vlastnosti tohto úložiska. Vykonali sme preto viaceré záťažové testy.

Virtuoso (evaluation)[1]	Free text search, SPARQL, subClassOf, subPropertyOf, JDBC driver, Cross-platform, good documentation, GPL	limited OWL support
Jena (evaluation) [2]	Pure Java, BSD license, OWL API, SPARQL API, can be persisted using MySQL	Lack of documentation
Sesame [3]	Open source, BSD license, pure Java, query support using SeRQL	Lack of documentation / feature list
4 store [4]	SPARQL support, backup support, cluster support	imposes certain hardware, lack of documentation, lack of Java drivers
Semantic server [5]		built on top of MS SQL server, proprietary software, requires licensing per machine
SIREn [6]	Full-text indexing (Lucene)	built on top of Lucene search engine, lack of SPARQL support
AllegroGraph RDFStore [7]	SPARQL, Free text search, implemented in Java	Proprietary software. Free version has restricted heap space (60MB)
Kowari [8]		this project has been forked and no longer supported. See mulgara project
Mulgara [9]	SPARQL, can be embedded, pure Java, transaction support	Lack of documentation

Na **??** vidíme v grafe časy vykonania každého dopytu. A na 6.2 vidíme situáciu pri vykonávaní paralelne.



Obr. 6.1: Časy vloženia 100 tripletov samostatne.



Obr. 6.2: 3 Časy vloženia 100 tripletov paralelne.

Pri paralelnom vkladaní narástli časy priemerne o 13% pri PC1 a 23,4%. Zaujímavý je ale celkový nárast času testu ktorý je približne 28 s pre oboch používateľov. Z toho a aj z grafu na vidíme vyrovnané škálovanie výkonu.

6.4.2 Test vkladania cez aplikáciu

Vkladali sme 9000 tagov prostredníctvom skriptu. Trvanie bolo 75 minút. Pri vkladaní na priamo bol čas trvania 20 minút. Overhead je približne 73%. To je pozitívny výsledok vzhľadom na množstvo vykonávanej vnútornej logiky pri vkladaní cez aplikáciu.

6.4.3 Test naplnením veľkým počtom tagov

Do databázy sme vložili vyše 10 578 450 tagov. Nárast veľkosti bol o 2105 MB. Naplnenie nemalo merateľný dopad na výkon. Dopyt na vybranie všetkých tagov trval 101 sekúnd.

Dodatok A

Tútoriál ku knižnici

A.1 Rozbehanie knižnice

Knižnica ja dostupná na:
http://labss2.fiit.stuba.sk/TeamProject/2010/team17is-si/ine.html

1. stiahnuť knižnicu zo stránky uvedenej vyššie (.zip súbor obsahujúci aj ostatné knižnice, ktoré je potrebné importovať)

2. importovať knižnicu a ostatné knižnice, ktoré sú potrebné do projektu vo vývojovom prostredí

3. v .zip súbore sa nachádza konfiguračný súbor (classes/host-configuration.xml), kde je potrebné nastaviť adresu, kam sa knižnica pripája. Štruktúra konfiguračného súboru (host-configuration.xml) je nasledovná:

   ```
   <server>
           <hostname>http://localhost:8080/provider/</hostname>
   </server>
   ```

   Podľa servera, na ktorý sa má knižnica pripojiť, je potrebné zmeniť hostname nasledovne: http://localhost:8080/provider/ nahradiť aktuálnou adresou (kde je nasadená serverovská časť)

4. Konfiguračný súbor treba vložiť na miesto, kde sa nachádza spustiteľná aplikácia (napr. jar), ktorá využíva knižnicu (pri vývoji je potrebné umiestniť tento súbor na classpath).

5. v .zip súbore sa nachádza aj konfiguračný súbor *application-context.xml*, ktorý je potrebný na správne zinicializovanie jednotlivých tried. Súbor treba umiestniť na classpath.

6. v .zip súbore sa nachádza aj konfiguračný súbor *objectStorage.properties* kde si používateľ nastaví cestu, kam sa budú ukladať sťahované objekty. Cestu je potrebné nastaviť do premennej *PATH*.

A.2 Používanie knižnice

Knižnica obsahuje triedy : DataObjectRestClient, MetadataObjectRestClient, RestClientManager, ObjectController a UserRestClient.
Štruktúra balíkov je nasledovná:

```
sk.fiit.team17cf.stfu.client.controller
        ObjectController.java
sk.fiit.team17cf.stfu.client.impl
        DataObjectRestClient.java
        MetadataObjectRestClient.java
        UserRestClient.java
sk.fiit.team17cf.stfu.client
        RestClientManager.java
```

A.2.1 Vytváranie DataObjectov a MetadataObjectov

```
DataObject dataObject = ObjectController.createDataObject(String fileName, String objectName);
MetadataObject metaObject = ObjectController.createMetadataObject(String object, String predicat
MetadataObject metaObject = ObjectController.createMetadataObject(String predicate, String subje
MetadataObject metaObject = ObjectController.createMetadataObject(String subject);
```

A.2.2 Inicializácia DataObjectRestClient, MetadataObjectRestClient a UserRestClient

Objekty stačí inicializovať raz na začiatku, ďalej využívame referencie na ne.

```
ApplicationContext context = new ClassPathXmlApplicationContext("application-context.xml");
DataObjectManager dataRestClient = (DataObjectManager) context.getBean("dataRestClient");
MetadataObjectManager metaRestClient = (MetadataObjectManager) context.getBean("metaRestClient")
UserRestClient userRestClient = (UserRestClient) context.getBean("userRestClient");
```

A.2.3 Práca s objektami (fotky, videá, . . .)

Na prácu s objektami slúži trieda DataObjectRestClient, ktorá implementuje rozhranie DataObjectManager.

A.2.4 Pridanie objektu

```
DataObject dataObject = ObjectController.createDataObject(fileName, objectName);
DataObjectRestClient dataRestClient;
String result = dataRestClient.add(dataObject);
```

Funkcia add vráti id objektu (kde je objekt fyzicky uložený).

Získanie objektu

```
DataObjectRestClient dataRestClient;
DataObject dataObject = dataRestClient.get(idObjektu);
```

Funkcia vráti objekt typu DataObject a príslušný súbor uloží na disk na používateľom definované miesto (v objectStorage.properties, viď úvod).

Vymazanie objektu

```
DataObjectRestClient dataRestClient;
dataRestClient.delete(objectId);
```

Vymaže objekt a všetky jeho metadáta.

Získanie zoznamu všetkých objektov, ktoré má používateľ uložené

```
DataObjectRestClient dataRestClient;
Collection<DataObject> resultList = dataRestClient.list();
```

Vráti zoznam Idčiek objektu.

A.2.5 Práca s metadátami

Na prácu s metadátami slúži trieda MetadataObjectResrClient, ktorá implementuje rozhranie MetadataObejctManager.

Pridanie metadát k objektu

```
MetadataObject metaObject = ObjectController.createMetadataObject(IDobject, predicate, subject);
MetadataObjectRestClient metaRestClient;
String result = metaRestClient.add(metaObj);
```

Funkcia vráti Idobjektu, ku ktorému boli pridané metadáta.

Vymazanie metadát

```
MetadataObject metaObject = ObjectController.createMetadataObject(IDobject, predicate, subject);
MetadataObjectRestClient metaRestClient;
metaRestClient.delete(metadataObject);
```

Vymažme zadaný triplet.

Získanie zoznamu všetkých metadát, ktoré má používateľ uložené

```
MetadataObjectRestClient metaRestClient;
Collection<MetadataObject> metaCollection = metaRestClient.list();
```

Získanie zoznamu najviac používaných tágov

Funkcia získa tágy, ktoré sú najviac používané. Vstupným parametrom je počet, koľko najviac používaných tágov chce používateľ získať.

```
MetadataObjectRestClient metaRestClient = new MetadataRestClient();
int count;
Collection<TagCount> tagCounts = metaRestClient.mostUsedTags(count);
```

Výstupom je zoznam tagov a ich početnosti (trieda TagCount opisuje dvojicu tag - počet výskytov).

Získanie zoznamu tágov, ktoré pokrývajú najviac objektov

Funkcia, ktorá získa zoznam tágov, ktoré pokrývajú najviac objektov. Vstupným parametrom je počet, koľko najviac pokrývajúcich tágov chce používateľ získať.

```
MetadataObjectRestClient metaRestClient;
int count;
Collection<TagCount> tagCounts = metaRestClient.mostCoveringTags(count);
```

Výstupom funkcie sú najviac pokrývajúce tágy a počet objektov, ktoré tág pokrýva (trieda TagCount opisuje dvojicu tag - počet objektov, ktoré pokrýva).

A.3 Vyhľadanie objektu (fotky, videá) podľa metadát

- je zadaný iba subject (tag, ktorým je fotka otagovaná). Napríklad triplet vyzerá nasledovne:

- object: http://localhost:8080/katka/fotka1.jpg
- predicat: tag
- subject: slnko

Chceme vyhľadať všetky fotky, ktoré obsahujú slnko. Na to slúži nasledujúci príklad:

```
MetadataObject metaObj = ObjectController.createMetadataObject(subject);
MetadataObjectRestClient metaRestClient;
List<MetadataObject> metaObjList = new ArrayList<MetadataObejct>();
metaObjList.add(metaObj);
Collection<PreviewObject> previewCollection = metaRestClient.search(metaObjList);
```

Funkcia vráti zoznam, ktorý obsahuje PreviewObject-y (náhľady).

- je zadaný predikát a subject ("vzťah" a "tag"). Napríklad triplet vyzerá nasledovne:

- object: http://localhost:8080/katka/fotka1.jpg
- predicat: je
- subject: fotka

Chceme vyhľadať všetky objekty, ktoré sú fotky. Na to slúži nasledujúci príklad:

```
MetadataObject metaObj = ObjectController.createMetadataObject(predicat, subject);
List<MetadataObject> metaObjList = new ArrayList<MetadataObejct>();
metaObjList.add(metaObj);
Collection<PreviewObject> previewCollection = metaRestClient.search(metaObjList);
```

Funkcia vráti zoznam, ktorý obsahuje PreviewObject-y (náhľady).

Dodatok B

Používateľská príručka k webovému rozhraniu

Obsah

B.1 Úvod

Príručka opisuje prácu s webovou aplikáciou. V kapitole 2 je priblížený postum pre login a registráciu. Kapitola 3 slúži na zorienotovanie sa v aplikácii. V prvej časti vysvetľuje rozloženie panelov a v druhej časti rozoberá funkcionalitu aplikácie a postup pre jednotlivé funkcie.

B.2 Login / Registrácia

Ak účet na serveri nemáme, vyplníme polia meno a heslo. Zvolíme možnosť "registracia". Týmto spôsobom si vytvoríme konto s menom a heslom aké sme zadali. Po registrácii nás automaticky aplikácia odkáže na rovnakú obrazovku. Ak chceme otvoriť aplikáciu, prihlásime sa údajmi nášho konta.
Pre prihlásenie do aplikácie zadáme meno a heslo do prislúchajúcich políčok a zvolíme tlačítko "login". Systém nám zobrazí hlavnú obrazovku aplikácie, ktorá je opísaná v kapitole 3.

B.3 Aplikácia a jej možnosti

B.3.1 Zorientujem sa v hlavnom okne aplikácie

Okno pozostáva z troch hlavných panelov. Podľa farieb si ich môžeme rozdeliť na žltý – ľavý, biely – stredný a oranžový – pravý.


Super Truper Fajnové Úložisko
Prihlásený užívateľ
admin
odhlásiť
Usporiadanie plochy:
Vitajte na stránke STFU - Super Truper Fajnového Úložiska
CAT fight
Objekty
Metadáta
Help
Správa účtu
Admin
Nahranie súboru
Choose File
no file selected
Upload
Vyhľadávanie
Rozšírené vyhľadávanie
Tag :
Hladaj
Vylistuj všetky
Odstránenie objektu
ID :
Odstráň
Stiahnutie objektu
Stiahnuť s metadátami
ID :
Over

Žltý – ľavý

- na ňom vidíme zobrazenné meno prihláseného používateľa
- ponúka možnosť odhlásiť
- nachádzajú sa na ňom tri tlačítka umožňujúce meniť rozloženie troch hlavných panelov

Biely – stredný panel

- je to panel, kam sa nám zobrazujú uložené objekty

Oranžový – pravý panel

- obsahuje niekoľko záložiek, do ktorých je rozdelená celá funkcionalita aplikácie (popísané v kapitole 3.2)

B.3.2 Možnosti aplikácie

Funkcionalita celej aplikácie je rozdelená v taboch v oranžovom - pravom paneli.

B.3.3 Objekt

Nahranie súboru na server

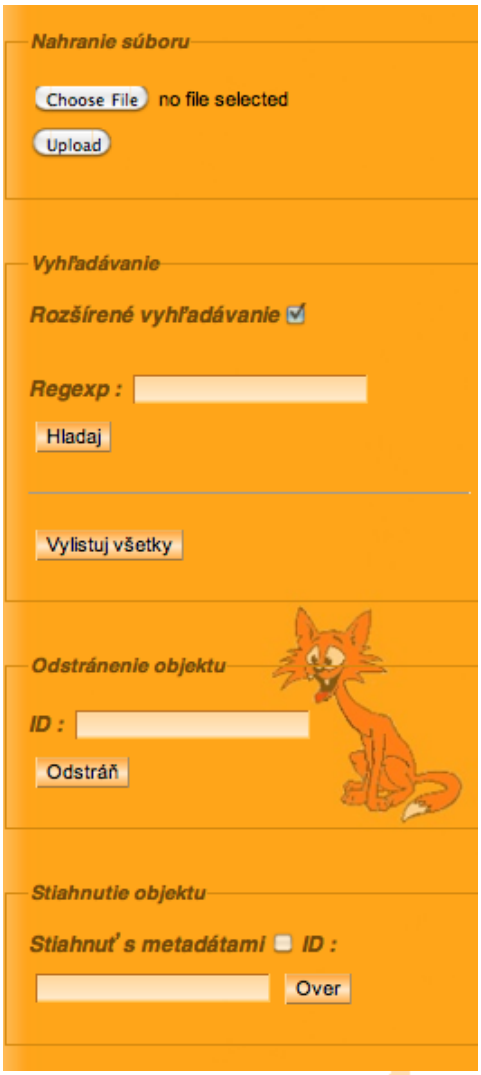
Cez tlačítko vybrať súbor si vyberieme súbor, ktorý chceme nahrať na server, potvrdíme tlačidlom upload

Vyhľadávanie

Pokiaľ chceme vyhľadávať jednoducho, len na základe tagu, napíšeme želaný tag do pola a potvrdíme tlačidlom hľadaj
Ak chceme vyhľadávať pomocou rozšíreného vyhľadávania, zaškrtneme si možnosť Rozšírené vyhľadávanie a do pola napíšeme regulárny výraz.

Vylistuj všetky

Zvolením tlačítka Vylistuj všetky sa nám zobrzia všetky naše objekty v bielom – strednom panely



Odstránenie objektu zo serveru

Ak chceme odstrániť objekt, stačí napísať jeho ID do pola a potvrdiť tlačidlom Odstráň

Ak nevieme ID objektu, stačí ak si ho vylistujeme, alebo nájdeme pomocou vyhľadávania. Po zobrazení v bielom – strednom panely naň klikneme a jeho ID sa automaticky uloží do pola.

Stiahnutie objektu do PC

Ak chceme stiahnuť objekt do nášho PC, stačí napísať jeho ID do pola a potvrdiť tlačidlom. Volíme si, či chceme objekt stiahnuť aj s metadátami, alebo bez nich, ak chceme s metadátami zaškrtneme voľbu Stiahnuť s metadátami.
Ak nevieme ID objektu, stačí ak si ho vylistujeme, alebo nájdeme pomocou vyhľadávania. Po zobrazení v bielom – strednom panely naň klikneme a jeho ID sa automaticky uloží do pola.

B.3.4 Metadáta

Pridanie metadát

V tomto prípade zadávame tri parametere: Objekt – Subjekt – Predikát

- Subjekt je vzťah, ten si zvolíme z ponúkaných
- Na základe vybraného vzťahu vyplníme Objekt a Predikát
  - V prípade vzťahu panoráma sú Objekt aj Predikát IDobjektov
  - V prípade vzťahu tag je Objekt IDobjektu a Predikát hodnota, čiže obyčajné slovo
  - V prípade vzťahu synonymum sú Objekt aj Predikát hodnoty, čiže obyčajné slová

Odstránenie metadát

Rovnako ako v predošlom prípade zadávame tri parametere, ktoré definujú metadáta, ktoré chceme odstrániť.

Tagcloudy

Najpočetnejšie tagy

Volíme si počet. Tento počet definuje počet tagov, ktoré sa vyskytujú najčastejsie. Po zadaní počtu a potvrdení sa nám vypíšu tagy a k nim počty, koľkokrát sú použité.

Najviac pokrývajúce tagy

Volíme si rovnako počet. Počet označuje počet tagov, ktoré pokrývajú najväčšiu množinu objektov. Po zadaní počtu a potvrdení sa nám vypíšu tagy a k nim počty, v koľkých objektoch sú použité.

B.3.5 Správa účtu

Zmena hesla

Ak si chceme zmeniť heslo, stačí napísať do pola nové heslo a potvrdiť tlačítkom Zmeň

Odstránenie účtu

S tlačidlom pre odstránenie účtu odporúčame zaobchádzať veľmi opatrne. Vymažeme ním aktuálny účet.

B.3.6 Administrátor

Pridanie účtu

Po zadaní mena a hesla do pola a následnom potvrdení sa vytvorí na server účet s danými parametrami.

Odstránenie účtu

Po zadaní mena a potvrdení sa vymaže účet s používateľským menom zadaným do pola.

Dodatok C

Používateľská príručka k Super Truper Fajnovému Úložisku

C.1 Úvod

Príručka opisuje prácu s CLI aplikáciou. Na začiatku sú uvedené základné informácie potrebné pre celkové pochopenie príručky a správne používanie aplikácie. Neskôr je uvenedný návod ako spustiť aplikáciu v interaktívnom
a v neinteraktívnom režime, ako zadávať príkazy a ako aplikáciu ukončiť.

C.2 Základy

Názov

- Super Truper Fajnové Úložisko má skratku **STFU**
  Zakázané znaky
- pri názvoch objektov sú zakázané tieto znaky: [medzera], [.], [|], [“], [-]
  Forma zadávania príkazov
- príkaz –prepínač –prepínač . . .
- súčasťou prepínača môže byť aj atribút alebo atribúty (podľa definície prepínača)
- povinné prepínače sú vpísané v [] (musia byť uvedené)
- nepoviné prepínače sú vpísané v {} (nemusia byť uvedené)
- znak [|] zanmená **alebo** a **a**
- interval sa zadáva formou ľavá_hlanica..pravá_hranica

- pri práci s metadátami sa používa formát
'ID_objektu/hodnota'-vztah-'ID_objektu/hodnota'

C.3 Interaktívny režim

C.3.1 Spustenie aplikácie

Príkaz
názov [–l login] [–pw heslo]

Práva
každý používateľ, ktorý má účet na serveri

Prepínače (ukážkové prihlasovaci údaje sú v kapitole 5. Prílohy)
-l za prepínačom uvediete váš login
-pw za prepínačom uvediete vaše heslo

Príklad
STFU -l both -pw secret

C.3.2 Práca s používateľským účtom

Funkcie týkajúce sa používateľského účtu spúšťame príkazom **acc**.

Vytvorenie nového používateľskéhu účtu

Príkaz
acc [-new] [-nl novy_login] [-npw novy_password]

Práva
administrátor

Prepínače
-new prepínačom vytvoríte nový používateľský účet
-nl za prepínačom uvediete login k novému účtu
-npw za prepínačom uvediete heslo k novému účtu

Príklad
acc -new -nl meno –npw sec12

Výpis informácií o účte

Príkaz
acc

Práva
každý používateľ, ktorý má účet na serveri
Pri tomto príkaze nepovažujem za nutné uvádzať príklad.

Zmena používateľského účtu

Príkaz
acc [-ch] [-npw novy_password]

Práva
každý používateľ, ktorý má účet na serveri

Prepínače
-ch prepínačom zmeníte vybrané atribúty účtu
-npw za prepínačom uvediete nové heslo

Príklad
acc -ch -nl meno

Resetovanie používateľského účtu

Príkaz
acc [-re meno1 meno2...]

Práva
administrátor

Prepínače
-re prepínačom resetujete účty s uvedenými menami

Príklad
acc -re both user

Odstránenie používateľského účtu

Príkaz
acc [-rm meno1 meno2...]

Práva
administrátor

Prepínače
-rm prepínačom odstránite účty s uvedenými menami

Príklad
acc -rm both user

Help k používateľskému účtu

Príkaz
acc [-help]

C.3.3 Práca s objektami

Funkcie týkajúce sa objektov spúšťate príkazom **obj**.

Pridanie objektov na server

Príkaz
obj [-add IDobjektu1 IDobjektu2...] {-p cesta}

Práva
každý používateľ, ktorý má účet na serveri

Prepínače
-add prepínač pridá na server objekty so zadanými ID z defaultnej zložky
-p prepínačom pridáte objekty zo zadanej cesty

Príklad

```
COMMAND: obj -add zamocek.jpg tekvice.jpg -p C:/Users/Zuzanka/Desktop/CLIatestov
anie/obrazky/
Added object with ID: http://192.168.59.1:8081/admin/zamocek.jpg

Added object with ID: http://192.168.59.1:8081/admin/tekvice1.jpg
```

Stiahnutie objektov do počítača

Príkaz
obj [-give IDobjektu1 IDobjektu2...] {-m}

Práva
každý používateľ, ktorý má účet na serveri

Prepínače
-give uloží do počítača objekty so zadanými ID do defaultnej zložky
-m prepínačom zvolíte uloženie objektov aj s metadátami

Príklad

```
COMMAND: obj -give http://192.168.59.1:8081/admin/tekvice.jpg -m
Object saved to: C:\Users\Zuzanka\Desktop\CLIatestovanie\download\tekvice.jpg
```

Vyhľadávanie objektov

Príkaz
obj [-search] { tag1 tag2... }

Práva
každý používateľ, ktorý má účet na serveri

Prepínače
-search prepínačom vyhľadáte objekty, ID vyhľadaných objektov sa vypíšu do konzoly

Príklad
obj -search slnko pes

Pokročilé vyhľadávanie

Príkaz
obj [-asearch] {-x AND OR ()}{-t tag1 tag2... } { -r vztah-hodnota | -ri vztah-interval_hodnot } { -g [[-s suradnice | -si interval_suradnica] } {-size pocet } {-prev }{-o nazov}

Práva
každý používateľ, ktorý má účet na serveri

Prepínače
-t prepínačom si zvolíte vyhľadávanie podľa tagov, ktoré uviediete ako tag1, tag2....
-r prepínačom si zvolíte vyhľadávanie podľa vzťahu, atribút sa zadáva vo forme vzťah-hodnota viď. Poznámka 1
-ri prepínačom si zvolíte vyhľadávanie podľa vzťahu, atribút sa zadáva vo forme vzťah hodnota, pričom hodnota je zadaná intervalom viď. Poznámka 1
-g prepínačom zvolíte vyhľadávanie podľa geolokácie
-s prepínačom zvolíte vyhľadávanie podľa súradníc
-si prepínačom zvolíte vyhľadávanie podľa súradníc, pričom súradnice sú zadané intervalom
-size prepínačom zvolíte výpis obmedzeného počtu vyhľadaných objektov
-prev prepínačom zvolíte uloženie náhľadov vyhľadaných objektov do defaultnej zložky
-o prepínačom zvolíte vyhľadávanie podľa reálneho názvu objektu
-x prepínačom zapnete používanie rehulárnych výrazov
AND operátor použijete, ak chcete vyhľadať podľa dvoch a viacerých tagov naraz
OR operátor použijete ak chcete vyhľadať podľa splnenia jedného alebo viacerých tagov
() operátory AND a OR, môžete používať naraz v jednom príkaze, podľa potreby použite () ako v matematickej logike

Príklad
obj -asearch (-o .*txt OR -t jano -ri width-1..5) AND -x -t jaho..da

Odstránenie objektov zo servera

Príkaz
obj [-rm IDobjektu1 IDobjektu2...]

Práva
každý používateľ, ktorý má účet na serveri

Prepínače
-rm prepínač odstráni zo servera objekty so zadanými ID

Príklad

```
COMMAND: obj -rm http://192.168.59.1:8081/admin/veverka.jpg
Object deleted successfully
```

Vylistovanie všetkých objektov

Príkaz
obj [-list]

Práva
každý používateľ, ktorý má účet na serveri

Príklad

```
COMMAND: obj -list
You have stored these files:
http://192.168.59.1:8081/admin/oznamko.bmp
http://192.168.59.1:8081/admin/oznamko1.bmp
http://192.168.59.1:8081/admin/oznamko2.bmp
http://192.168.59.1:8081/admin/oznamko3.bmp
http://192.168.59.1:8081/admin/tekvice.jpg
http://192.168.59.1:8081/admin/tekvice1.jpg
http://192.168.59.1:8081/admin/tekvice2.jpg
http://192.168.59.1:8081/admin/tekvice3.jpg
http://192.168.59.1:8081/admin/veverka.jpg
http://192.168.59.1:8081/admin/zamocek.jpg
http://192.168.59.1:8081/admin/zamocek1.jpg
http://192.168.59.1:8081/admin/zamocek2.jpg
http://192.168.59.1:8081/admin/zamocek3.jpg
```

Help k práci s objektami

Príkaz
obj [-help]

Poznámka 1
Podrobný zoznam vzťahov, podľa ktorého je možné zadávať vzťah v dvojci vzťah-hodnota, na ktorú sa odvolávam v kapitole 3.3.1 Vyhľadávanie objektov pri prepínačoch –r a –ri nájdete v kapitole 5. Prílohy v Tabuľke 1 v časti Vzťahy medzi objektami a hodnotami.
Napríklad:
autor-Jano zanmená, že Jano je autorom

C.4 Práca s metadátami

Funkcie týkajúce sa metadát spúšťate príkazom meta.

Pridávanie metadát

Príkaz
meta [-add hodnota1-vztah1-hodnota1 hodnota2-vztah2-hodnota2...]

Práva
každý používateľ, ktorý má účet na serveri

Prepínače
-add pridá zvolené metadáta zadané v tripletoch hodnota-vztah-hodnota viď. Poznámka 2

Príklad

```
COMMAND: meta -add 'http://192.168.59.1:8081/admin/oznamko.bmp'-dateTimeCreated-
'0001-01-01T00:00:00'
Added metadata to object with id: http://192.168.59.1:8081/admin/oznamko.bmp
```

Odstraňovanie metadát

Príkaz
meta [-rm hodnota1-vztah1-hodnota1 hodnota2-vztah2-hodnota2...]

Práva
každý používateľ, ktorý má účet na serveri

Prepínače
-rm odstráni zvolené metadáta zadané v tripletoch hodnota-vztah-hodnota viď. Poznámka 2

Príklad

```
COMMAND: meta -rm 'http://192.168.59.1:8081/admin/oznamko.bmp'-dateTimeCreated-'
0001-01-01T00:00:00'
Metadata deleted sucessfully
```

Vyhľadanie najviac pokrývajúcich tagov

Príkaz
meta [-mc pocet tagov]

Práva
každý používateľ, ktorý má učet na serveri

Prepínače
-mc vypíše zadaný počet najviac pokrývajúcich tagov

Príklad

```
COMMAND: meta -mc 4
Hladam najviac pokryvajuce tagy, pocet je 4
You have stored these most covering tags:
Tag: .jpg

Tag: Karolko

Tag: Canon

Tag: animacia
```

Vyhľadanie najviac používaných tagov

Príkaz
meta [-mu pocet tagov]

Práva
každý používateľ, ktorý má učet na serveri

Prepínače
-mu vypíše zadaný počet najviac používaných tagov

Príklad

```
COMMAND: meta -mu 6
Hladam najviac pouzivane tagy, pocet je 6
You have stored these most used tags:
Tag: autor

Tag: .jpg

Tag: blablaobjektiv

Tag: Karolko

Tag: Canon

Tag: animacia
```

Vylistovanie všetkých metadát

Príkaz
meta [-list]

Práva
každý používateľ, ktorý má účet na serveri

Príklad

```
COMMAND: meta -list
You have stored these metadata:
Object: http://Zuzanka-PC:8081/admin/oznamko.bmp
        tag http://Zuzanka-PC:8081/admin/values/obrazok

Object: http://Zuzanka-PC:8081/admin/oznamko.bmp
        dateTimeCreated 0001-01-01T00:00:00

Object: http://Zuzanka-PC:8081/admin/oznamko.bmp
        dateTimeChanged 0001-01-01T00:00:00

Object: http://Zuzanka-PC:8081/admin/oznamko.bmp
        autor http://Zuzanka-PC:8081/admin/values/Karolko

Object: http://Zuzanka-PC:8081/admin/oznamko.bmp
        ISO 300

Object: http://Zuzanka-PC:8081/admin/oznamko.bmp
        focus http://Zuzanka-PC:8081/admin/values/blablaobjektiv

Object: http://Zuzanka-PC:8081/admin/oznamko.bmp
        cameraModel http://Zuzanka-PC:8081/admin/values/Canon

Object: http://Zuzanka-PC:8081/admin/oznamko.bmp
```

Help k práci s metadátami

Príkaz
meta [-help]

C.4.1 Odhlásenie z interaktívneho režimu CLI aplikácie

Príkaz
logout

Poznámka 2
Podrobný zoznam vzťahov, podľa ktorého je možné zadávať vzťah v triplete hodnota-vzťah-hodnota, na ktorú sa odvolávam v kapitole 3.4.2 Odstraňovanie metadát a 3.4.3 Pridávanie metadát pri prepínačoch –**add** a –**rm** nájdete v kapitole 5. Prílohy v Tabuľke 1 v častiach Vzťahy medzi objektami a hodnotami a Vzťahy medzi objektami navzájom. Hodnota v triplete znamená hodnota prislúchajúca vzťahu, ale môže to byť aj ID objektu, ktorý je daným vzťahom spojený s hodnotou alebo s iným ID objektu.
Napríklad:
'http://fero-pc/fero/lavicka.jpg'-tag-'Marienka' zanmená, že na fotke je označená Marienka
'http://fero-pc/fero/esej.odt'-autor-'Fero' znamená, že autorom objektu http://fero-

pc/fero/esej.odt je Fero

C.5 Neinteraktívny režim

Neinteraktívny režim znamená, že všetko, čo potrebujete, vybavíte jedným príkazom, vrátane spustenia aplikácie, prihlásenia na server, zadania potrebných príkazov a odhlásenia. Pri nesprávnom použití príkazu sa zobrazí help. **Príkaz** (ukážkové prihlasovaci údaje sú v kapitole 5. Prílohy)
názov [–l login] [–pw heslo] [-command “ prikaz1 -prepinace1 | prikaz2 -prepiace2 |...”]

Práva
zavisí od príkazov, ktoré zadáte ako argumenty prepínaču command, jednotlivé práva sú uvedené pri opisoch príkazov v kapitole 3. Interaktívny režim

Prepínače
-l za prepínačom uvediete váš login
-pw za prepínačom uvediete vaše heslo
-command prepínaču dáte do úvodzoviek ako argumenty tie príkazy, ktoré chcete vykonať, môžete zadať akýkoľvek príkaz alebo sekvenciu príkazov z uvedených v kapitole 3. Interaktívny režim, jednotlivé príkazy v sekvencii sú oddelené znakom [|]

Príklad
STFU -l both -pw secret -command “ obj -search -t banan | meta -search”

C.6 Prílohy

Ukážkové prihlasovacie údaje:

- meno – admin , heslo – secret
- meno – user , heslo – secret
- meno – both , heslo – secret

Dodatok D

Štatistiky kvality kódu

Autor: Bc. Juraj Bahno
Pri písaní kódu sme sa riadili týmito pravidlami:

- Počet metód na triedu by nemal presiahnuť 10.
- Počet logických riadkov na metódu by mal byť najviac 20. (Fyzických riadkov by malo byť maximálne 40)
- Počet logických riadkov na triedu by nemal byť viac ako 200. (Ak počítame fyzické riadky, nemal by počet presiahnuť dvojnásobok, teda maximum 400)
- Javadoc komentár musí mať každá metóda a trieda

D.1 Kompletné výsledky

	Počet Riadkov	Počet Prázdnych riadkov	Počet Tried	Počet Metód	Počet Importov
4.12.2010	5370	853	65	239	396
14.12.2010	6347	991	69	334	419
4.5.2011	15283	2947	126	881	962

LCOM 1-4 (Lack of Cohesion of Methods) – metriky, ktoré určujú previazanosť medzi komponentami.

LCOM1 (priemer)	LCOM2 (priemer)	LCOM3 (priemer)	LCOM4 (priemer)	Cyklometrická zložitosť (priemer)	Neokomentované metódy
3	0.1967	0.2512	2	1.53138	81%
25	0.2415	0.3043	2	1.425	81%
32	0.3274	0.3982	3	1.5993	

Cyklometrická zložitosť

Priemerná cyklometrická zložitosť je 1.5993189557321226

Dátum	4.12.2010	14.12.2010
Priemer	1.53	1.43
Triedy s najväčšou cyklometrickou zložitosťou	DataController::searchObject: 13 CliMain::doMain: 8 CliControler::accFeature: 6 CliControler::objFeature: 6 UriAdressManager::getUri: 6	DataController::searchObject: 11 CliMain::doMain: 8 CliControler::objFeature: 7 UriAdressManager::getUri: 6 CliControler::metaFeature: 6
Dátum	4.5.2011	
Priemer	1.6	
Triedy s najväčšou cyklometrickou zložitosťou	SearchExpParser::parse: 11 PreviewObject::createThumbnail: 11 MetadataObject::equals: 9 SearchQueryUtils::getFilterObject: 9 CliMain::doMain: 9	

Počet metód

Počet všetkých metód v programe je 334.

Dátum	4.12.2010	14.12.2010
Počet metód	239	334
Triedy s najväčším počtom metód	MetadataObjectManagerImpl: 15 RestClientManager: 12 MetadataObjectRestClient: 10 RestService: 10 CliControler: 9	CliParam: 56 MetadataObjectManagerImpl: 16 RestService: 15 RestClientManager: 13 DataController: 12
Dátum	4.5.2011	
Počet metód	881	
Triedy s najväčším počtom metód	CliParam: 66 MetadataObjectController: 59 DataObjectController: 34 MetadataObjectManagerImplTest: 33 MetadataObjectManageStressTest: 32	

Počet tried

Počet všetkých tried v programe je 69.

Dátum	4.12.2010	14.12.2010
Počet tried	65	69
Triedy s najväčším počtom fyzických riadkov	MetadataObjectManagerImpl: 366 CliMain: 325 DataController: 288 RestService: 261 MetadataRepositoryImpl: 228	MetadataRestServiceTest: 376 MetadataObjectManagerImpl: 374 CliParam: 305 DataController: 301 RestService: 290
Dátum	4.5.2011	
Počet tried	126	
Triedy s najväčším počtom fyzických riadkov	MetadataObjectManagerImpl: 1113 MetadataObjectManagerImplTest: 753 MetadataObjectManageStressTest: 676 MetadataObjectController: 432 RestService: 388	

Počet neokomentovaných metód
Celkový počet metód v programe je 239 z toho 193 neokomentovaných.

Dátum	4.12.2010	14.12.2010
Počet neokomentovaných metód	239	271
Najviac neokomentované metódy	RestClientManager: 13 MetadataObject: RestService: 14 MockServiceScope: 9 CliController: 9 RestService: 8	CliParam: 56 MetadataObject: 10 RestClientManager: 8 MetadataObjectManagerImplMock 8

D.2 Výsledky v porovnaní so skutočnosťou

Maximálny počet logických riadkov na triedu sme si na začiatku určili na 200 a počet fyzických riadkov 400. Kedže táto štatistika uvádza počty fyzických riadkov, vidíme, že tento limit sa nám podarilo dodržať.
Počty metód na triedu sme mali stanovené na 10, a boli prekročené v dvoch prípadoch pri metódach MetadataObjectManagerImpl a RestClientManager.
Na otestovanie kvality kódu sme použili plugin do NetBeansu s názvom SimpleCodeMetrics. Po označení koreňového uzla programu a spustení pluginu sme dostali nasledovné výsledky.

Ako vidíme, javadoc komentár nemá až 81% metód. Najideálnejšie je, ak by boli okomentované všetky metódy.
Na otestovanie komentárov javadoc sme použili Analyze javadoc z Netbeans, kde boli zobrazené všetky neokonemtované metódy. Tie boli následne manuálne zrátané.

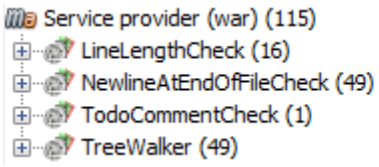
D.3 Statická analýza

Pri určovaní kvality kódu sme použili nástroje statickej analýzy. Jedná sa o CheckStyle, PMD a FindBugs.

D.3.1 CheckStyle

Nástroj, ktorý pomáha vývojárom písať kód, ktorý dodržiava definovaný štandard zdrojového kódu. Poskytuje možnosť konfigurácie, takže môže podporovať v podstate ľubovoľný štandard pre písanie zdrojového kódu. V novších verziách už poskytuje aj

kontroly, ktoré sa týkajú problémov pri návrhu tried, hľadanie duplicitného kódu a databázu známych chýb. V zozname kontrol na web stránke nástroja sa vždy nachádza aj dôvod, prečo sa takáto kontrola robí. Na obrázku **??** je znázornený modul Service Provider. Vidíme, že sa celkovo našlo 115 chýb, ktoré nespĺňajú pravidlá písania kódu podľa CheckSytle.



Obr. D.1: CheckStyle

D.3.2 PMD

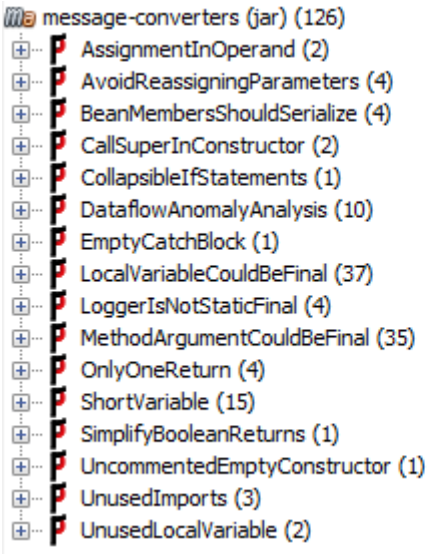
Prehľadáva zdrojový kód a hľadá potenciálne problémy, ako:

- *možné chyby* – napr. prázdne try/catch/finally/switch príkazy
- *nepoužívaný kód* – premenné, parametre, metódy
- *neoptimálny kód* – zbytočné spájanie reťazcov cez operátor „+“
- *zbytočne zložité výrazy* – zbytočné if príkazy, for príkazy, ktoré môžu byť zmenené na while príkaz
- *duplicitný kód* – znamená aj kopírovanie chýb

Pracuje s tzv. sadami pravidiel (angl. ruleset), ktoré združujú isté kontroly dokopy. Podľa stránky sú implicitne vytvorené tieto sady pravidiel:

- *unusedcode* – nepoužitý kód
- *basic* – základne postupy, ktoré by mal každý vývojár dodržiavať
- *design* – možné návrhové problémy
- *controversial* – kontroverzné pravidlá, pri ktorých sa vývojári nezhodujú na tom, či je to užitočné alebo nie.

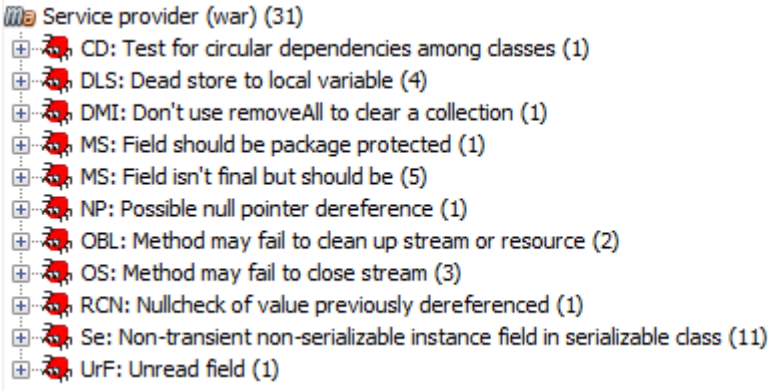
Každý si môže vytvoriť vlastnú sadu z existujúcich pravidiel, a tiež je možnosť zadefinovať si vlastné pravidlá. Na obrázku D.2 je zobrazený výsledok pre modul message-converters. Našlo sa 126 chýb.



Obr. D.2: PMD

D.3.3 FindBugs

Nástroj na statickú analýzu Java kódu za účelom nájdenia chýb. Je založený na koncepte vzoru chyby (bug pattern). Pracuje so skompilovaným kódom, teda nie je potrebný originálny zdrojový kód. Na stránke upozorňujú, že nájdené chyby nemusia byť aj chyby reálne. Na obázku D.3 vidíme výsledky pre nástroj FindBugs aplikovaný na modul Service provider s 31 chybami.



Obr. D.3: FindBugs

D.3.4 Zhodnotenie

Chyby, ktoré boli nájdene spomýnanými nástrojmi, sme sa snažili odstrániť.

Dodatok E

Technická dokumentácia

V tejto časti sa nachádzajú návody na inštaláciu a úpravu nastavení potrebných nástrojov pre našu tímovú prácu.

E.1 Maven Instalation

Autor: Bc. Katarína Valaliková

1. download maven from http://www.apache.org/dyn/closer.cgi/maven/binaries/apache-maven-2.2.1-bin.zip
2. install (extract) maven into directiory e.g `C:\ProgramFiles\Maven`
3. set `M2_HOME` enviromental variable pointing to Maven instalation path `(e.g C:\ProgramFiles\Maven)`
4. add `M2_HOME/bin` to Path `(\%M2_HOME\%\bin)`
5. set path to Maven in Netbeans (Tools->Options->Miscellaneous->Maven->External Maven Home)

Development Tools

I recommend NetBeans IDE 6.7.1, that shoud be dowloaded from https://open-esb.dev.java.net/Downloads.html Install Java 1.6, lates update (21)

Open, Compile and Run maven project

Open

1. checkout project from svn repository

2. open prototype-parent project in netbeans
3. in prototype-parent maven project, there are modules -> open them, so you can modify each project

Compile

1. right click on the prototype-parent project
2. choose clean and build

Run

1. right click on the prototype-library project
2. choose run project
3. it should display following text "result is rdfrepo search obj repo search"

E.2 Inštalácia Latex

Autor: Bc. Peter Kajan

Na naištalovanie Latexu na váš počítač, postupujte podľa nasledovných krokov:

1. Stiahnite si a nainštalujte: http://ftp.cstug.cz/pub/tex/CTAN/systems/texlive/tlnet/install-tl.zip http://www.xm1math.net/texmaker/
2. Priložený súbor rozbaľte do Textmaker priečinku (napr. C:
   Program Files
   Texmaker)
3. Spustite Textmaker
4. Options -> Configure Texmaker -> Editor:
   - **Editor Font Encoding** zmeňte na Utf-8
   - do **Spelling dictionary** napíšte C:/Program Files/Texmaker/hunspell-sk-20091213/sk_SK.dic
5. Otvorte požadovaný súbor a dajte Quick build - ak sa zobrazí vygenerované PDF, tak je všetko v poriadku

E.3 Návod na inštaláciu TortoiseSVN

Autor: Bc. Marek Uhlár

V nasledujúcich krokoch je opísaný postup ako si na vašom počítači rozbeháte SVN :

1. Stiahnite si tortoiseSVN http://tortoisesvn.net/downloads
2. Naištalujte. Next, next, next.
3. Vytvorte si v systéme priečinok.
4. Kliknite na neho pravým tlačítkom -> SVN checkout
5. Do URL of repository zadajte svn+ssh://labss2.fiit.stuba.sk/mnt/sdb1/home/users/team1si/team17is-si/svnrep
6. OK
7. Pravým kliknite na priečinok -> TortoiseSVN -> Settings -> Network a do SSH Client zadajte C:
   Program Files
   TortoiseSVN
   bin
   TortoisePlink.exe -l VASE_MENO -pw VASE_HESLO
8. Prípadne poupraviť tú cestu k TortoisePlink.exe ak ste si to nainštalovali na iné miesto
9. Skúsite update commit nad priečinkom, pridajte súbor atď.

E.4 Nastavenie mailového klienta Gmail pre osobitné ukladanie tímovej pošty

Autor: Bc. Juraj Bahno

1. V poštovej schránke zvolíte nastavenia -> menovky
2. Dole máte políčko kde napíšete názov menovky napr. Tímak a kliknete na *Vytvorit*. Naľavo niekde pod doručenými správami sa vám vytvorí menovka.
3. Ďalej v nastaveniach prejdete na záložku filtre a zvolíte *Vytvoriť nový filter*

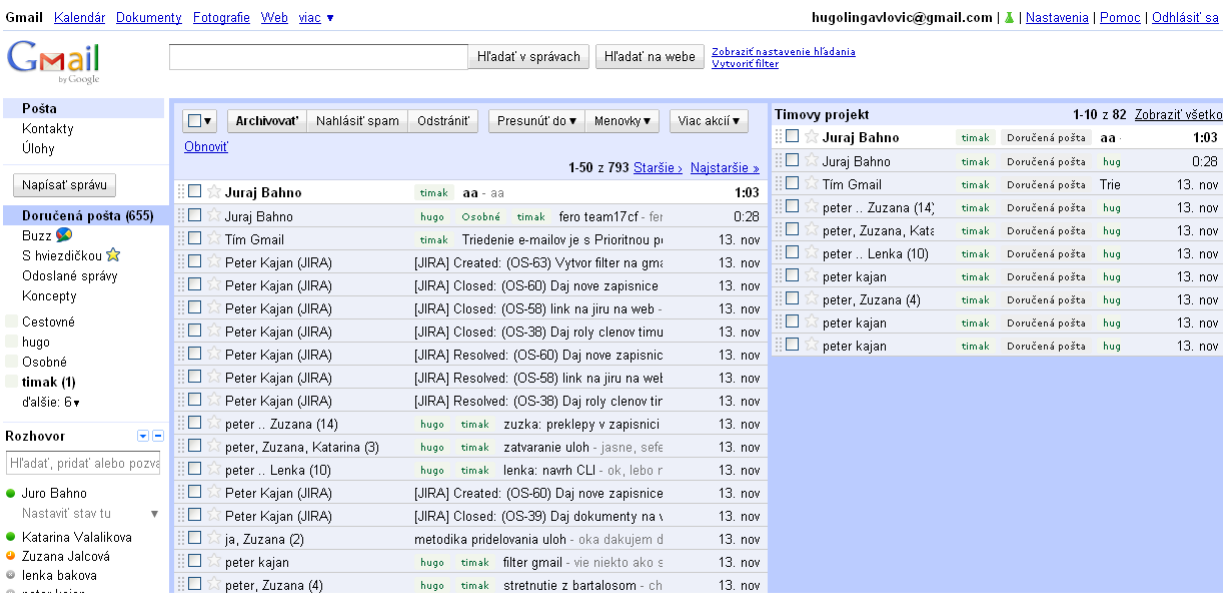
4. Do kolónky *Obsahuje slová* napíšete “team17cf” a stlačíte *Ďalší krok*

5. Zaškrtnete možnosť *Použiť označenie* a vyberiete názov menovky, ktorú ste si pred tým vytvorili pre tímový projekt

6. Ak chcete, aby sa vám tímova pošta nezobrazovala v normálnej doručenej pošte, ale iba vo vytvorenej záložke, zvoľte aj možnosť *Preskočiť doručenú poštu.*

7. Ak chcete mať v tejto záložke aj predchádzajúce správy tímaku, zaškrtnite možnosť *Použiť filter aj na X konverzácií nižšie.*

8. Stlačte *Vytvorit filter* a je to.



Obr. E.1: Obrazovka Gmail

Teraz sa vám budú do tejto záložky ukladať správy, ktoré sú určené pre všetkých členov tímu teda pre adresu team17cf@googlegroups.com . Ak chcete aby sa vám do záložky ukladali aj správy od jednotlivých členov tímu, musíte si vytvoriť nové filtre, kde do kolonky “Od” zadáte postupne adresy členov(každého treba asi zvlášť) a postupujete rovnako ako pri prvom návode. Ak chcete aby sa vám správy tímoveho projektu zobrazovali do zvlášť panela ako na obrázku, možete si to nastaviť podľa tohto návodu:

1. V poštovéj schránke zvolíte nastavenia -> Labs.

2. Nájdite funkciu *Viaceré priečinky doručenej pošty*(je niekde na konci zoznamu), pri nej zvoľte možnosť povoliť a stlačte *Uložiť zmeny*

3. V nastaveniach budete mať novú funkciu *Viaceré priečinky doručenej pošty*, zvoľte ju.

4. Do kolónky *Vyhľadávací dotaz pre okno 0* napíšte "is:timak" bez úvodzoviek. Namiesto slova timak napíšte názov vašej menovky, ktorú ste si zvolili pred tým pre tímové správy. Do kolónky *Názov panela* (voliteľné) napíšete ako sa má váš nový panel volať.

5. Pri možnosti *Maximálna veľkosť stránky* ešte možete nastavit, koľko najnovších správ sa má v novom panely zobrazovať.

6. Nastavíte, kde má byť nový panel umiestnený a zvolíte *Uložiť zmeny*.

E.5 Inštalácia databázy Mulgara

Autor: Peter Kajan(úprava Zuzana Jalcová)

V nasledujúcej časti je opísaný postup inštalácie databázy Mulgara.

1. Zo stránky http://www.mulgara.org/download.html si stiahnite a rozbaľte **mulgara-2.1.9-bin.zip**

2. V svn/navody/mulgara sa nachádzajú dva .bat súbory:
   mulgaraStart.bat obsahuje príkaz
   `java -jar C:\mulgara-2.1.9-bin\mulgara-2.1.9\dist\mulgara-2.1.9.jar -p 8060`
   mulgaraShutdown.bat obsahuje príkaz
   `java -jar C:\mulgara-2.1.9-bin\mulgara-2.1.9\dist\mulgara-2.1.9.jar -x`

3. Nastavte si v nich cestu k **mulgara-2.1.9.jar** (v ceste nemôže byť medzera!!!)

4. Mulgaru spustíte pomocu *mulgaraStart.bat*, okno počas práce s Mulgarou nezatvárajte!!!

5. Vo webovom prehliadači zadajte adresu http://localhost:8060

6. Po ukončení práce zatvorte Mulgaru pomocou *mulgaraShutdown.bat*

E.6 Návod na priradenie podúloh v JIRA

Autor: Bc. Juraj Bahno

Tento návod slúži na pridenanie podúloh v JIRA , kvôli vytvoreniu Ganntového diagramu. Tieto dve časti môžete urobiť aj v opačnom poradí, teda najskôr nastaviť čas a potom priradť podúlohu. Možno to bude ľahšie, ak si otvoríte viac okien s JIROU, kvôli porovnaniu časov.

PRIRADENIE PODÚLOHY

1. Otvorte si úlohu, ktorú chcete priradiť pod väčší task.
2. Zvoľte možnosť More Actions->Convert to Sub-Task.
3. Keď zvolíte [select issue], možete si vybrať, pod ktorý task chcete úlohu priradiť.
4. Ak sa tam nenachádza task, ktorý hľadáte, treba namiesto [select issue], začať písať do prázdneho políčka buď názov hľadaného tasku alebo jeho označenie "OS- ...".
5. Pri písani sa vám budú zobrazovať všetky zodpovedajúce tasky.
6. Keď nájdete ten správny, zvoľte ho a dávajte stále "next" a nakoniec "finish".
7. Podúloha je priradená. Keď zvolíte väčší task, budete vidieť, ktoré podúlohy k nemu patria.

NASTAVENIE ČASU

Vložte dátum začiatku a dátum konca tak, ze otvoríte úlohu a zvolíte "edit". Vyplníte políčka *PlannedEnd* a *PlannedStart* tak, aby tento čas sedel s plánovaným časom väčšieho tasku, ale aj s dátumom, ktorý je uvedený v políčku Due Date.

E.7 Návod pre vytvorenie štatistík stats SVN

Autor: Bc. Marek Uhlár

Nasledujúci návod ukazuje, ako si vytvoríte štatistiky stats SVN.
Predpoklady: lokálne nainštalované svn a Java.

1. Pripojiť sa na labss2.fiit.stuba.sk cez WinSCP.
2. Zmazať /team17is-si/svnrep.dump.
3. Pripojiť sa na labss2.fiit.stuba.sk cez putty.

4. Spustiť príkaz svnadmin dump /home/users/team17is-si/team17is-si/svnrep > svn.dump.
5. Vytvoriť adresár `c:\svn\dumps\` na lokálnom PC.
6. V ňom vytvoriť adresáre svnrp, svnwc a reports.
7. Skopírovať svn.dump do `c:\svn\dumps`.
8. Otvoriť cmd.
9. Spustiť príkaz svnadmin create `c:\svn\dumps\svnrp`.
10. Spustiť príkaz svnadmin `c:\svn\dumps\\svnwc < svn.dump`
11. Spustiť príkaz cd `c:\svn\dumps\svnwc`.
12. Spustiť príkaz svn log -v –xml > logfile.log
13. Spustiť príkaz cd `C:\svn\dumps\reports`.
14. Spustiť príkaz java -jar `c:\svn\dumps\statsvn.jar c:\svn\dumps\svnwc\projekt\logfile.log c:\svn\dumps\svnwc\projekt\`.
15. Zmazať obsah /team17is-si/public_html /statsvn.
16. Nakopírovať sem obsah `C:\svn\dumps\reports`.
17. Štatistiky sú dostupné na http://labss2.fiit.stuba.sk/TeamProject/2010/team17is-si/statsvn/.

E.8 Nastavenie Tomcatu pre NetBeans

Autor: Bc. Katka Valaliková

V nasledujúcej kapitole sa nachádza návod na rozbehanie Tomcatu v NetBeans-e.

1. Zo stránky http://tomcat.apache.org/download-60.cgi treba stiahnuť tomcat. Odporúčame Core zip pre windows.
2. Zip súbor rozbaliť do nejakého priečinka, napr. `C:\ProgramFiles\apache-tomcat-6.0.32`

3. Nastaviť cestu v environmental variables (pravy klik na my computer-> properties->advanced properties->environmental variables->system variables->new a tam zadat name:CATALINA_HOME, value="cesta kde je rozbaleny tomcat", napr.:C:/ProgramFiles/apache-tomcat-6.0.32).

4. Otvoriť NetBeans.

5. Hore v menu Tools->Servers, dole vlavo "add server" -> vybrať Tomcat 6.0 -> kliknúť na next -> nastavťt server location (tam kde je ten apache tomcat rozbalený, teda napr: `C:\ProgramFiles\apache-tomcat-6.0.32`.

6. Nastaviť user name a password.

7. Klik na finish.

8. Pravy klik na Service Provider -> properties -> run -> vybrať server Tomcat 6.

E.9 MySQL

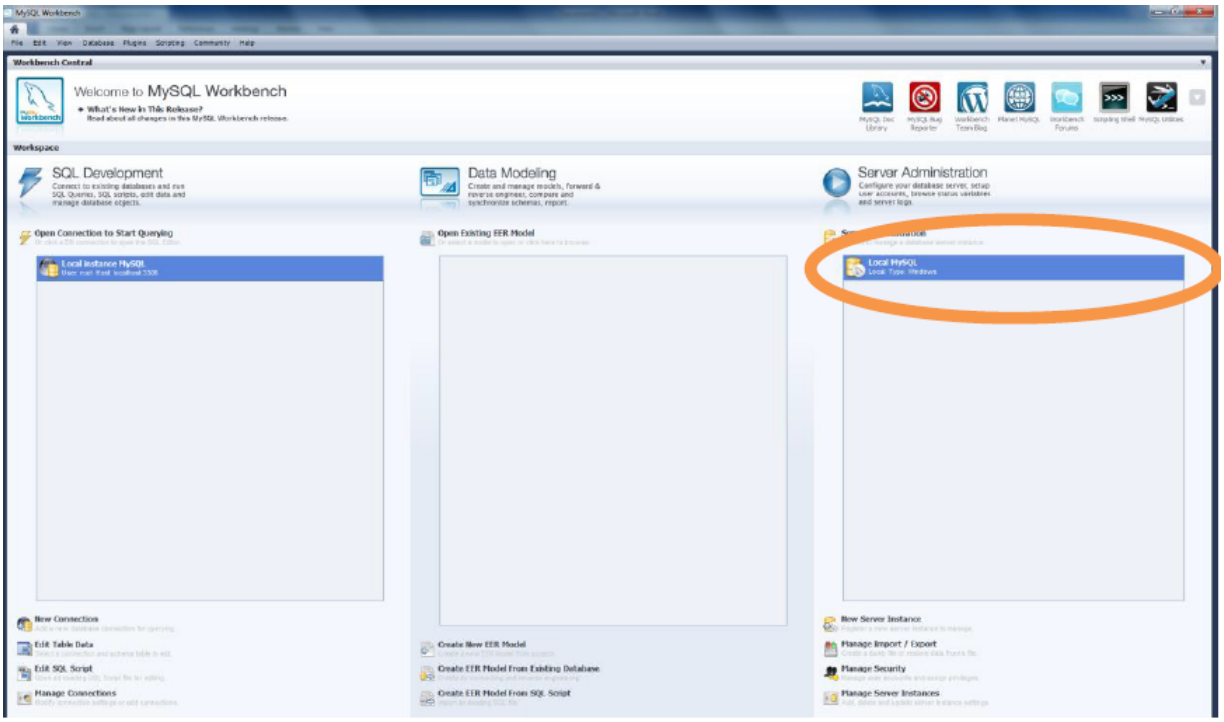
Autor: Marek Uhlár V nasledujúcej kapitole je opísaný postup naištalovania MySQL na počítač. Túto databázu potrebujeme pre správu používateľov.

1. Stiahnite MySQL z http://dev.mysql.com/downloads/mysql/ a nainštalujte. Ponechajte default konfiguráciu.

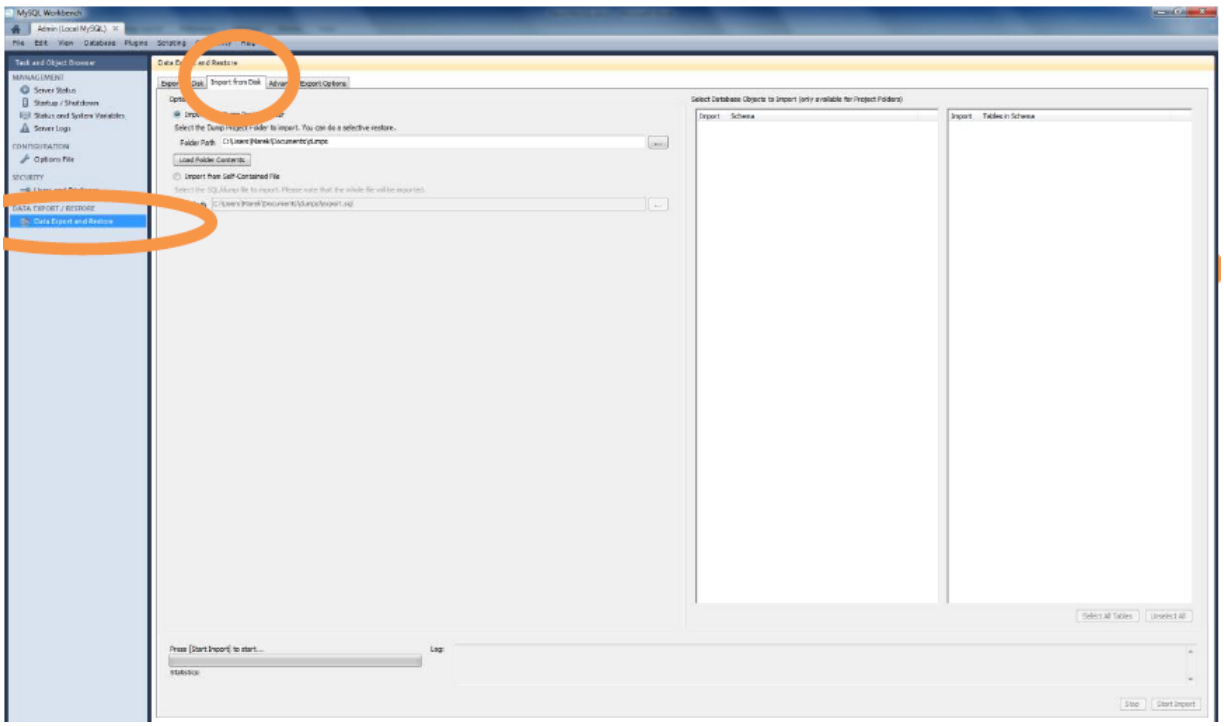
2. Stiahnite workBench z http://dev.mysql.com/downloads/workbench/ a nainštalujte.

3. Spustite MySQL Workbench.

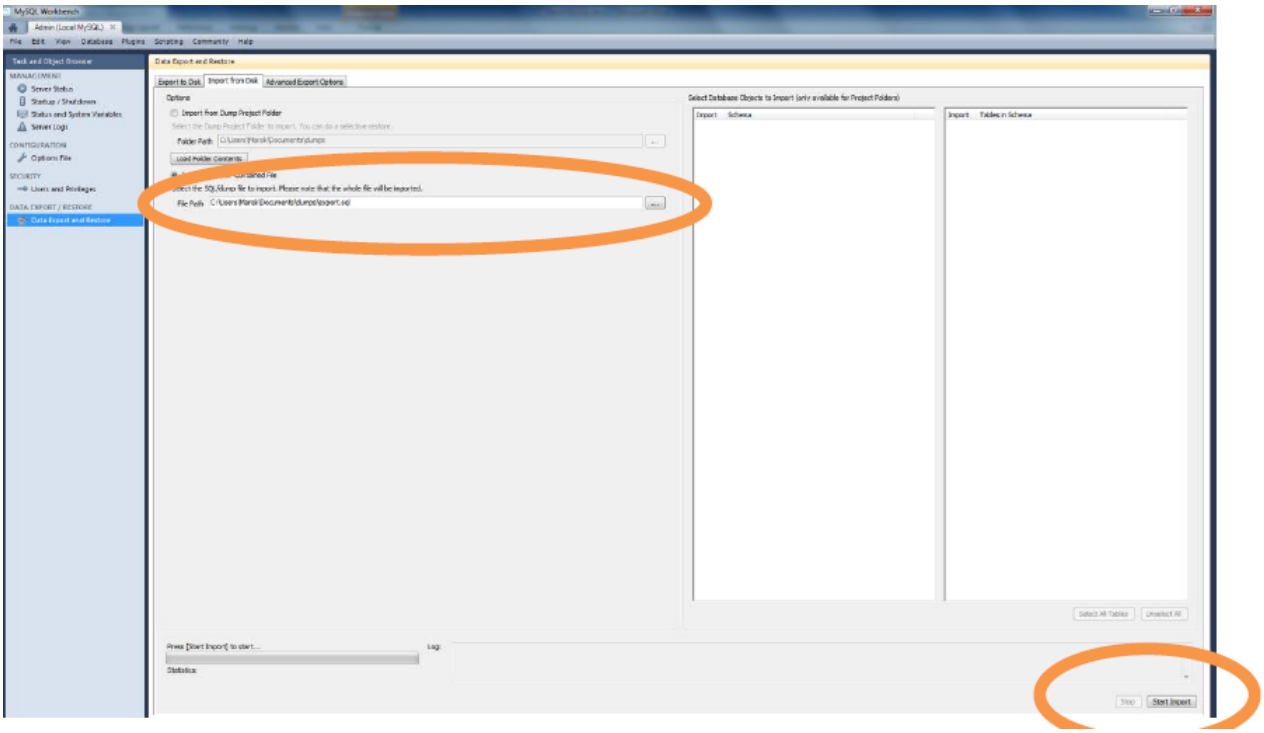
4. V časti Server administration otvorte LocalMySQL.



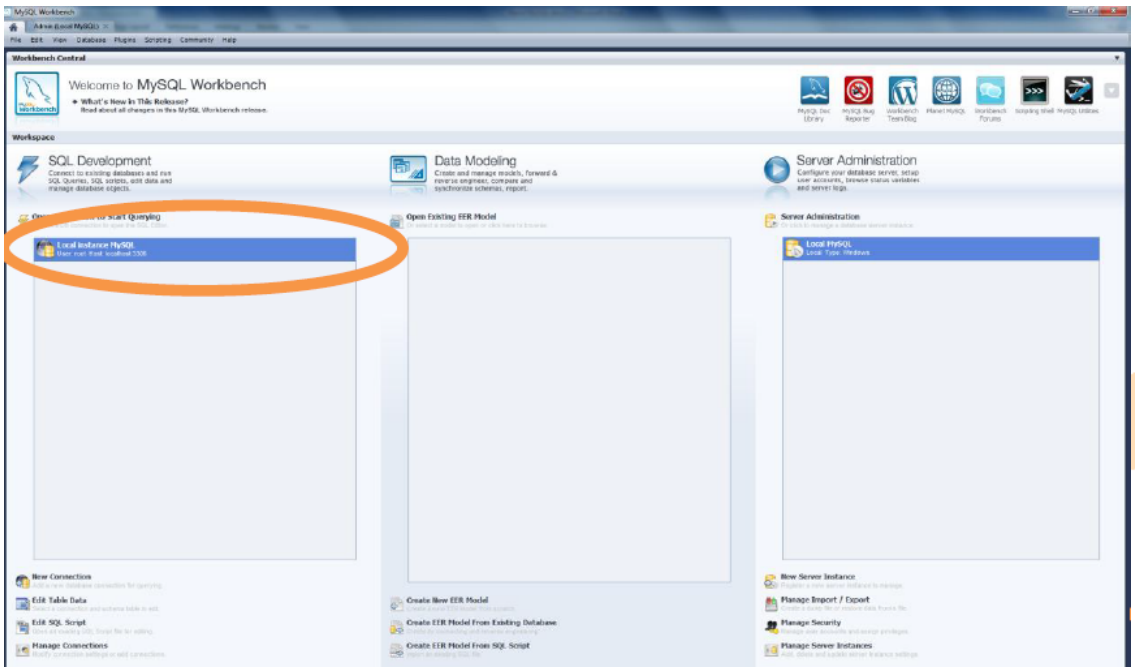
5. Zvoľte Data Export and Restore a tu import from disk.



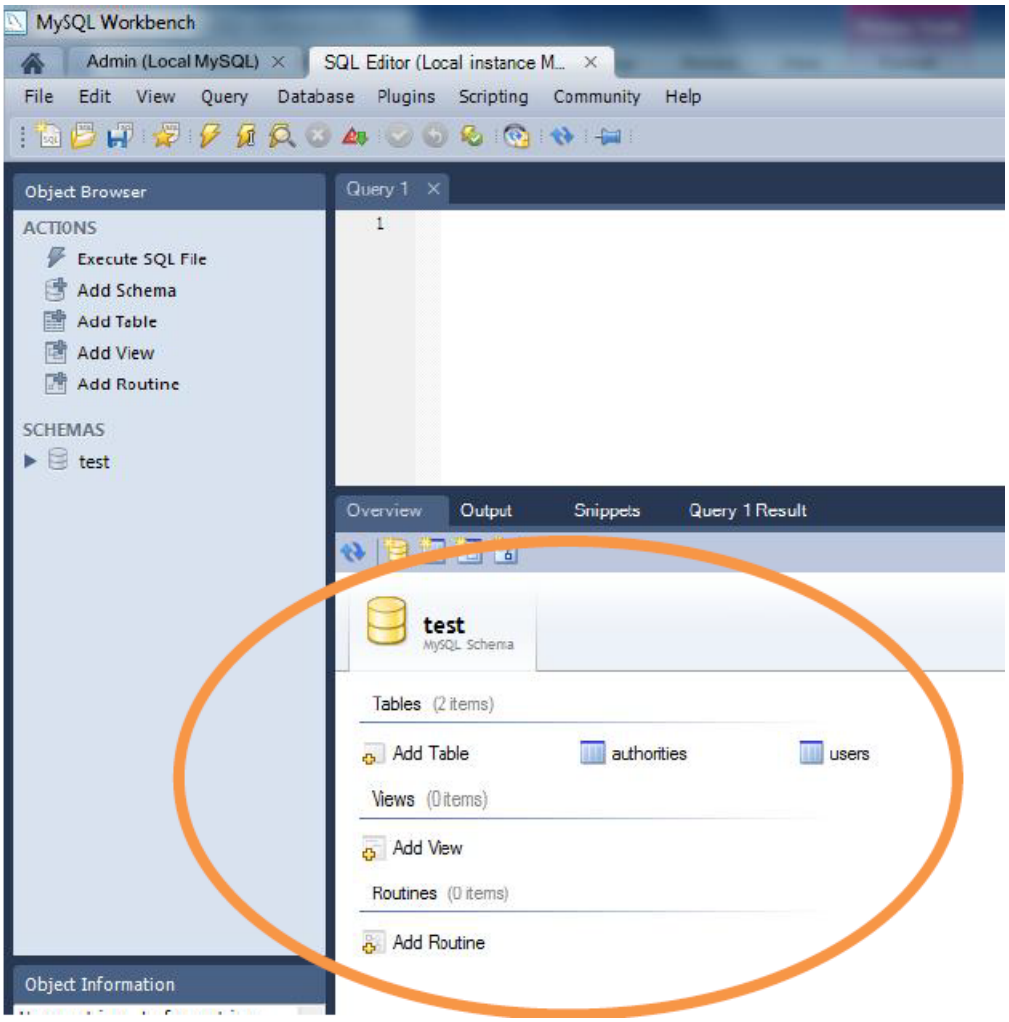
6. A tu zvoľte import from self-contained file a cestu podpora/navody/mysql_backup/Dump20110318.sql. Nasledne dajte Start import.



7. Výsledok skontrolujete tu.



po otvorení by tu mala byť databáza test s tabuľkami users a autorities.



Dodatok F

Vzťahy špecifické pre základné typy objektov

V tabuľke F.1 sú uvedené vzťahy špecifické pre základné typy objektov.

Tabuľka F.1: Špecifické vzťahy pre základné typy objektov

Názov vzťahu	**Prináležiaci typ hodnoty**	**Popis**	**Kedy je hodnota platná**
Fotografia			
Názov fotografie	text	Názov pod akým sa fotka ukladá	Ak obsahuje aspoň jeden znak (nie biely)
Názov modela fotoaparátu	text	Názov fotoaparátu, ktorý bola fotka vytvorená	Ak obsahuje aspoň jeden znak (nie biely)
Dátum vytvorenia	dátum	Dátum vytvorenia fotografie	Ak je správny formát dátumu
Dátum úpravy	dátum	Dátum poslednej úpravy fotografie	Ak je správny formát dátumu
Softvér fotoaparátu	text	Názov a číslo verzie softvéru, ktorý používa daný fotoaparát	Ak obsahuje aspoň jeden znak (nie biely)
Objektív	číslo	Veľkosť použitého objektívu	Ak je to kladná hodnota
Pokračovanie z predchádzajúcej strany			

Tabuľka F.1 – pokračovanie na nasledujúcej strane

Názov vzťahu	**Prináležiaci typ hodnoty**	**Popis**	**Kedy je hodnota platná**
Kvalita fotogarfie	text	Kvalita vytvorenej fotografie	Ak obsahuje aspoň jeden znak (nie biely
Blesk	text	Informácia o použití blesku pri vytvorení fotografie	Ak obsahuje aspoň jeden znak (nie biely
Vyváženie bielej	text	Spôsob vyváženia bielkej farby	Ak obsahuje aspoň jeden znak (nie biely
Veľkosť súboru	číslo	Veľkosť fotky	Ak je to kladné číslo
Formát	formát fotografie	Formát fotografie	Ak je to jedna z hodnôt definovaných v tabuľke 4.1
Šírka	číslo	Šírka fotografie v pixeloch	Ak je to kladná hodnota
Výška	číslo	Výška fotografie v pixeloch	Ak je to kladná hodnota
Vertikálne rozlíšenie	číslo	Rozlíšenie fotografie vo formáte dpi	Ak je to kladná hodnota
Horizontálne rozlíšenie	číslo	Rozlíšenie fotografie vo formáte dpi	Ak je to kladná hodnota
ISO	číslo	Citlivosť snímača	Ak je to kladné číslo
Film			
Názov filmu	text	Názov pod akým sa film ukladá	Ak obsahuje aspoň jeden znak (nie biely)
Pokračovanie z predchádzajúcej strany			

Tabuľka F.1 – pokračovanie na nasledujúcej strane

Názov vzťahu	**Prináležiaci typ hodnoty**	**Popis**	**Kedy je hodnota platná**
Formát	formát filmu	Formát filmu	Ak je to jedna z hodnôt definovaných v tabuľke 4.1
Dátum vytvorenia	dátum	Dátum vytvorenia fotografie	Ak je správny formát dátumu
Dátum úpravy	dátum	Dátum poslednej úpravy filmu	Ak je správny formát dátumu
Šírka	číslo	Šírka filmu v pixeloch	Ak je to kladná hodnota
Výška	číslo	Výška filmu v pixeloch	Ak je to kladná hodnota
Veľkosť súboru	číslo	Veľkosť videa	Ak je to kladné číslo
Prenosová rýchlosť zvuku	číslo	Prenosová rýchlosť zvuku v bitoch	Ak je to kladné číslo
Obnovovací kmitočet	číslo	Obnovovací kmitočet v okno ze sekundu	Ak je to kladné číslo
kodeky	text	Kodeky filmu	Ak obsahuje aspoň jeden znak (nie biely)
Celková prenosová rýchlosť	číslo	Rýchlosť prenosu	Ak je to kladná hodnota
Dĺžka trvania	číslo	Dĺžka trvania filmu v minútach	Ak je to kladná hodnota
Hudobná skladba			
Formát	formát hudobnej skladby	Formát hudobnej skladby	Ak je to jedna z hodnôt definovaných v tabuľke 4.1
Celková prenosová rýchlosť	číslo	Rýchlosť prenosu	Ak je to kladná hodnota
Pokračovanie z predchádzajúcej strany			

Tabuľka F.1 – pokračovanie na nasledujúcej strane

Názov vzťahu	**Prináležiaci typ hodnoty**	**Popis**	**Kedy je hodnota platná**
Dĺžka trvania	číslo	Dĺžka trvania skladby v minútach	Ak je to kladná hodnota
Interpret	meno	označuje interpreta skladby	Ak obsahuje aspoň jeden znak (nie biely)
album	text	označuje album, do ktorého skladba patrí	Ak obsahuje aspoň jeden znak (nie biely)
Dokument			
Názov súboru	text	Názov pod akým sa súbor ukladá	Ak obsahuje aspoň jeden znak (nie biely)
Formát	formát dokumentu	Formát dokumentu	Ak je to jedna z hodnôt definovaných v tabuľke 4.1
Autor	meno	Označuje autora dokumentu	Ak obsahuje aspoň jeden znak (nie biely)
Dátum vytvorenia	dátum	Dátum vytvorenia súboru	Ak je správny formát dátumu
Dátum úpravy	dátum	Dátum poslednej úpravy súboru	Ak je správny formát dátumu
Veľkosť súboru	číslo	Veľkosť súboru	Ak je to kladné číslo
Počet slov	číslo	Označuje počet slov v dokumente	Ak je to kladná hodnota
Url	link	Odkaz na originál súboru	Ak link funguje

Dodatok G

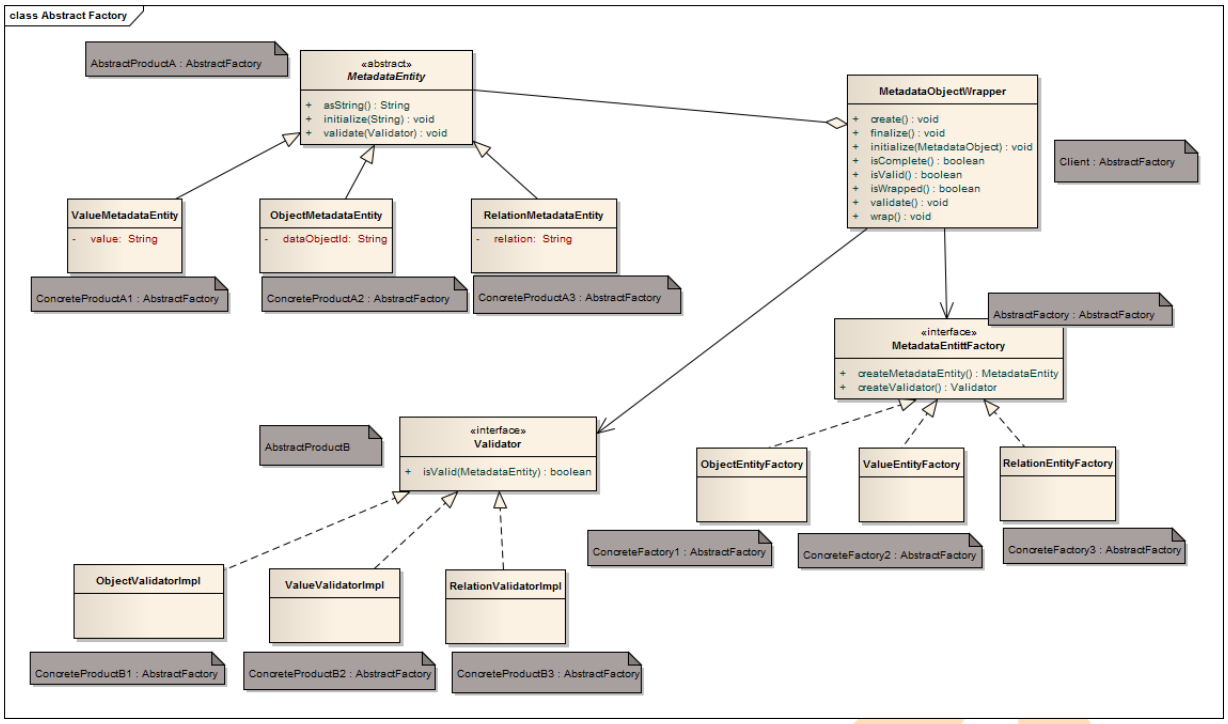
Popis identifikovaných návrhových vzorov

V tejto prílohe sa nachádza popis návrhových vzorov využitých v aplikácii.

G.1 Abstract factory

Autor: Peter Kajan
Na obr. G.1 je znázornený vzor *Abstract factory*. Využíva ho trieda *MetadataObject-Wrapper*, ktorá slúži na uchovávanie a spracovanie metadát. Tie sú vo forme tripletov v tvare objekt, predikát, subjekt. Každá položka tripletu môže byť nasledovná (pozri kapitola 4.3 Opis základných dátových entít):

- objekt,
- vzťah,
- hodnota.



Obr. G.1: Abstract factory

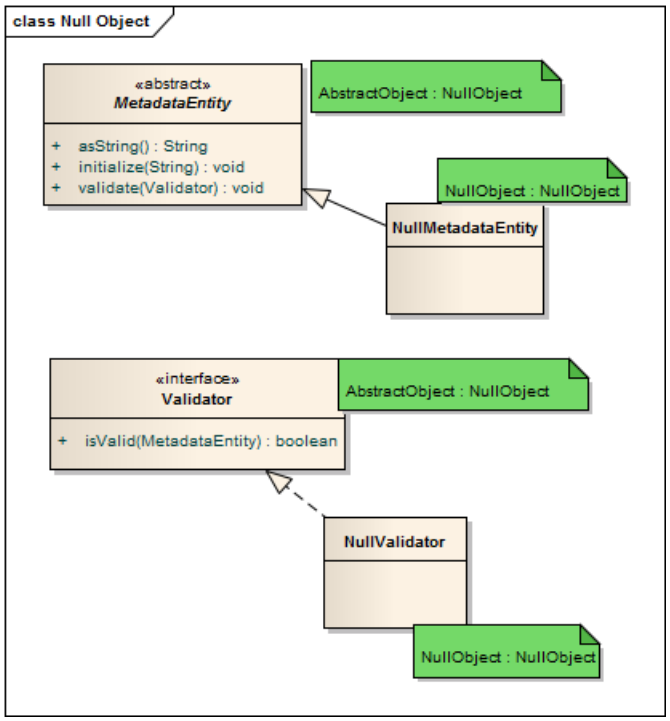
Pre každý typ dátovej entity je potrebný iný proces validácie, preto boli navrhnuté rôzne validátory.

Aplikovanie vzoru *Abstract factory* má nasledovné výhody:

- s každou položkou môžeme pracovať rovnako bez ohľadu na jej typ,
- každá položka tripletu bude mať priradený správny validátor.

G.2 Null object

Autor: Peter Kajan
Vzor null object je použitý na špeciálnu implementáciu rozhraní *MetadataEntity* a *Validator* (pozri obr. G.2).



Obr. G.2: Null object

Null object pre MetadataEntity

Trieda *MetadataObjectWrapper*, popísaná v predchádzajúcej časti, nemusí mať vyplnený celý triplet, ale iba jeho časť. Je to využívané napríklad na reprezentáciu filtrov pri vyhľadávaní. Vtedy sa do jednotlivých položiek priradí nulový objekt, konkrétne *NullMetadataEntity*.

Zavedenie vzoru *Null Object* má nasledovné výhody:

- práca s nekompletným objektom *MetadataObjectWrapper* je rovnaká, ako keby bol kompletný.
- dochádza k šetreniu pamäte, keďže trieda *NullMetadataEntity* je *Singleton*.

Null object pre Validator

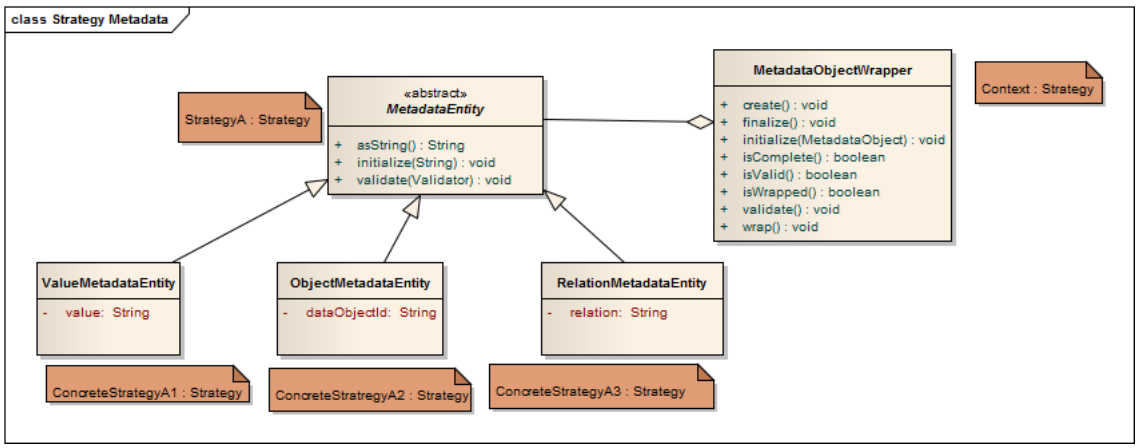
V niektorých prípadoch, napríklad ak inštancia položky triedy *MetadataObjectWrapper* je vyššie spomínaný nulový objekt, nie je potrebné položku validovať. Vtedy sa jej priradí validátor typu *NullValidator* navrhnutý podľa vzoru *Null object*.

Vzorom sme dosiahli nasledovné výhody:

- špeciálne prípady, kedy nemá byť položka validovaná, nie je potrebné riešiť vetvením na viacerých miestach, ale iba vytvorením a priradením null object validátora na jednom mieste. Táto výhoda súvisí so vzorom *Strategy* opísanom nižšie.
- S null object validátorom sa pracuje rovnako ako s ostatnými.
- Dochádza k šetreniu pamäte, keďže trieda *NullValidator* je *Singleton*.

G.3 Strategy 1

Autor: Peter Kajan
Rozhranie *MetadataEntity* je implementované rôznymi triedami. Každá má rozdielnu stratégiu inicializácie, validácie a konverzie na reťazec. Na implementáciu tohto správania sme využili vzor *Strategy* (pozri G.3).



Obr. G.3: Strategy 1

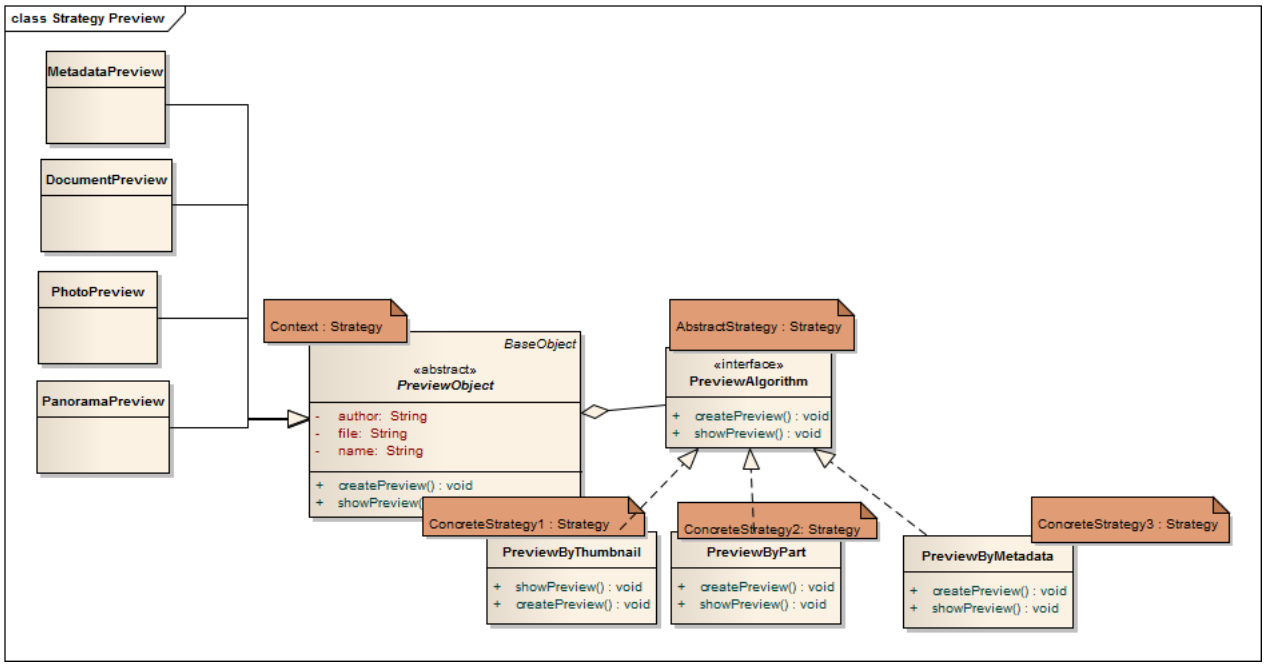
Aplikácia vzoru *Strategy* v kontexte triedy *MetadataObjectWrapper* má nasledovné výhody:

- jednotlivé konkrétne stratégie nie je potrebné riešiť vetvením na viacerých miestach, ale iba vytvorením a priradením špeciálnej inštancie na jednom mieste.

G.4 Strategy 2

Autor: Marek Uhlár

Aplikácia vzoru stratégia v kontexte triedy PreviewObject (pozri G.4) nám prináša možnosť jednoduchšej editácie algoritmov zobrazenia a vytvorenia náhľadu pri objektoch, ktoré dedia od objektu *PreviewObject*. Viaceré z týchto objektov potom môžu mať priradený rovnaký algoritmus (*PreviewByThumbnail*, *PreviewByPart* atď.), ktorý stačí editovať v jeho objekte.



Obr. G.4: Strategy 2

G.5 Command

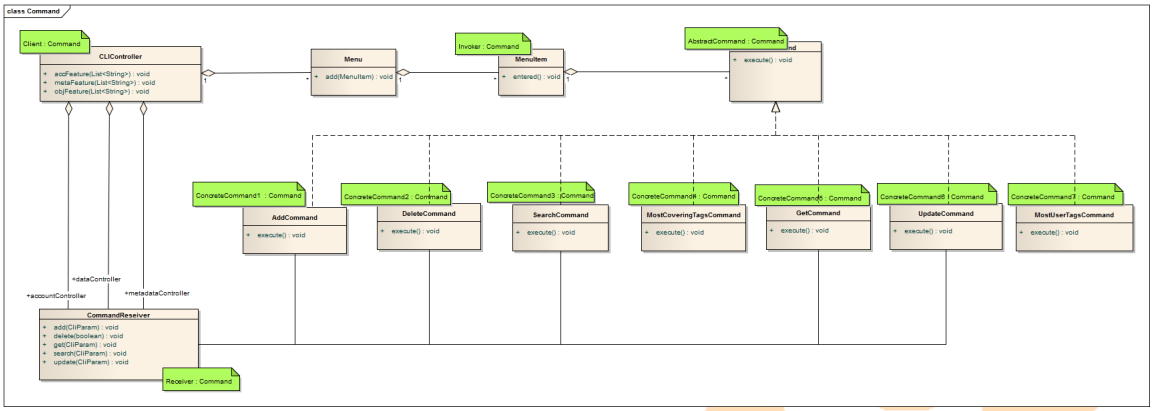
Autor: Katarína Valaliková
Vzor *Command* použijeme pri implementácii používateľského rozhrania. V role *Receiver* budú tri kontroleri:

- DataController,
- MetadataController,
- AccountController

so spoločným rozhraním. Klient, v našom prípade *CliController* môže využívať jeden zo siedmych konkrétnych príkazov, napr.

- AddCommand,
- DeleteCommand,
- SearchCommand.

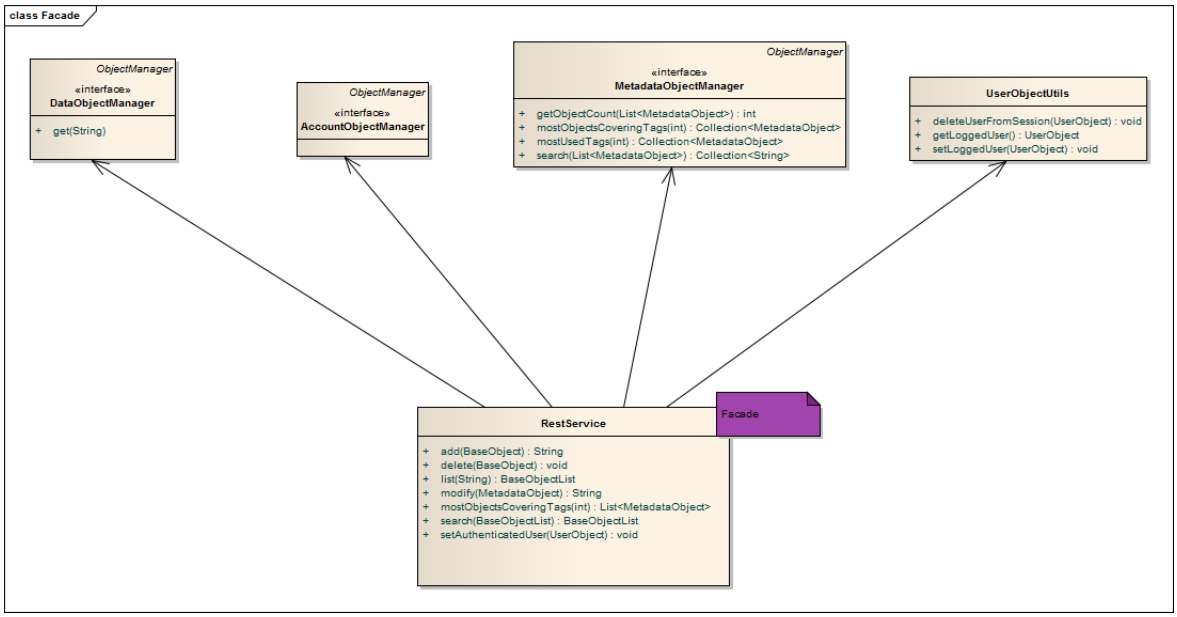
Každý z príkazov zabezpečuje vykonanie funkcie(execute) podľa svojho názvu. Ďalej každý z príkazov obsahuje znovu vykonanie(redo) a vrátenie späť(undo). Diagram znázorňujúci vzor *Command* sa nachádza na obr. G.5.



Obr. G.5: Command

G.6 Facade

Autor: Katarína Valaliková
Vzor *Facade* je zobrazený na obr. G.6.



Obr. G.6: Facade

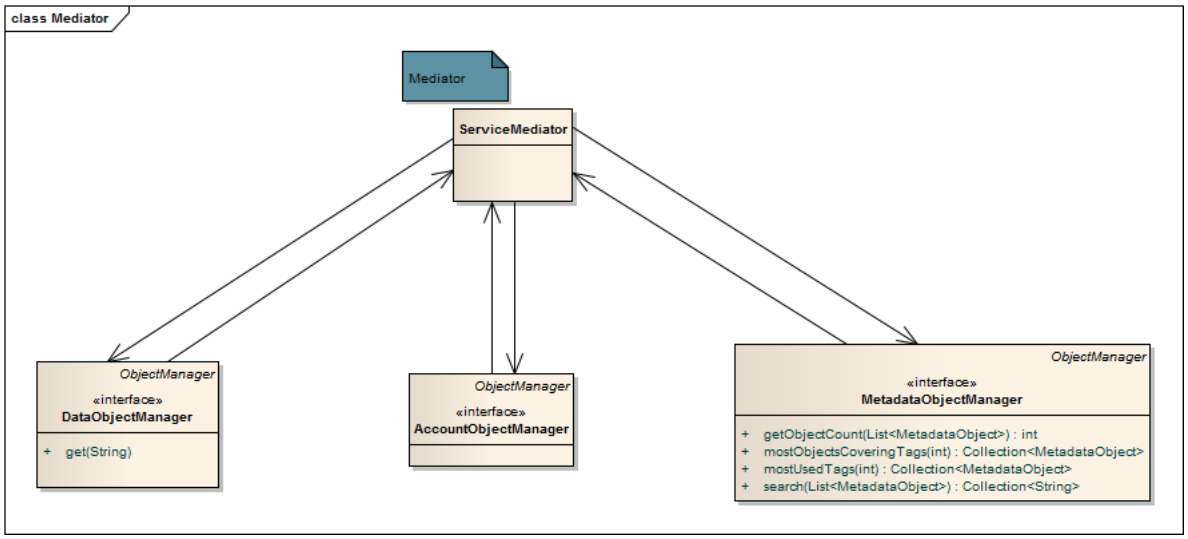
Tento vzor sme sa rozhodli použiť pre triedu *RestService*, ktorá spája rozhrania *DataObjectManager*, *MetadataObjectManager* a *AccountObjectManager*. Využitím návrhového vzoru *Facade* poskytuje *RestService* jednotné rozhranie, čo uľahčuje aj komunikáciu s klientskou časťou.

Výhody:

- jednotný interface
- stačí jedna REST služba
- nepotrebujeme zložitú komunikáciu medzi klientom a serverom

G.7 Mediator

Autor: Katarína Valaliková
Vzor *Mediátor* je zobrazený na obr. G.7.



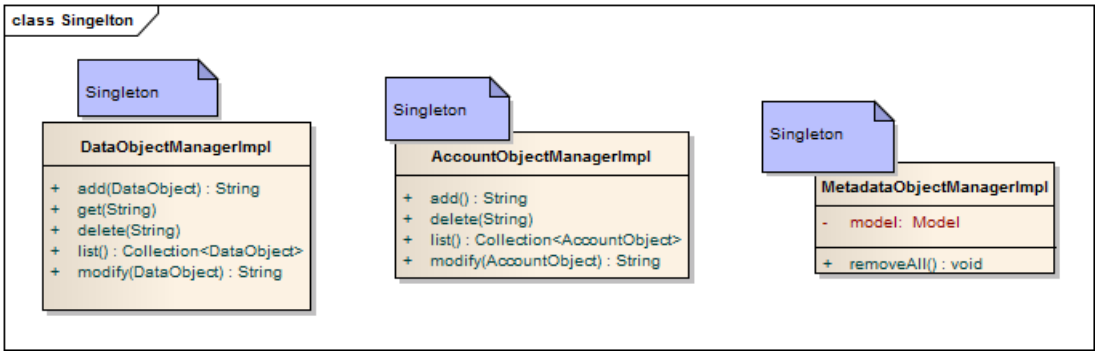
Obr. G.7: Mediator

Mediátor navrhujeme použiť na strane servera. Jeho hlavným prínosom je, že sa vyhneme vzájomnému previazaniu komponentov *Metadata Repository*, *Data Repository* a *User Respository* a komunikáciu medzi nimi bude sprostredkovávať mediátor. Napr. ak používateľ pridáva metadáta k objektu, musíme overiť, či taký objekt existuje. Aby sme sa vyhli tomu, že komponenty *Metadata Repository* a *Data Repository* budú medzi sebou navzájom prepojené, overenie či objekt existuje sprostredkuje mediátor.

G.8 Singelton

Autor: Katarína Valaliková
Vzor *Singleton* využívame v nasledujúcich triedach:

- *MetadataObjectManagerImpl* – trieda, ktorá pracuje s metadátami uloženými v databáze Mulgara. Táto trieda vytvára spojenie s databázou a toto spojenie si uchováva, preto je používaná ako *Singleton*.
- *DataObjectManagerImpl* – trieda pracujúca s objektami (napr. fotka, video, dokument).
- *AccountObjectManagerImpl* – trieda pracujúca s používateľskými účtami. Podobne ako *MetadataObjectManagerImpl* vytvára spojenie a uchováva si referenciu na spojenie s databázou, preto sme sa aj v tomto prípade rozhodli použiť návrhový vzor *Singleton*.



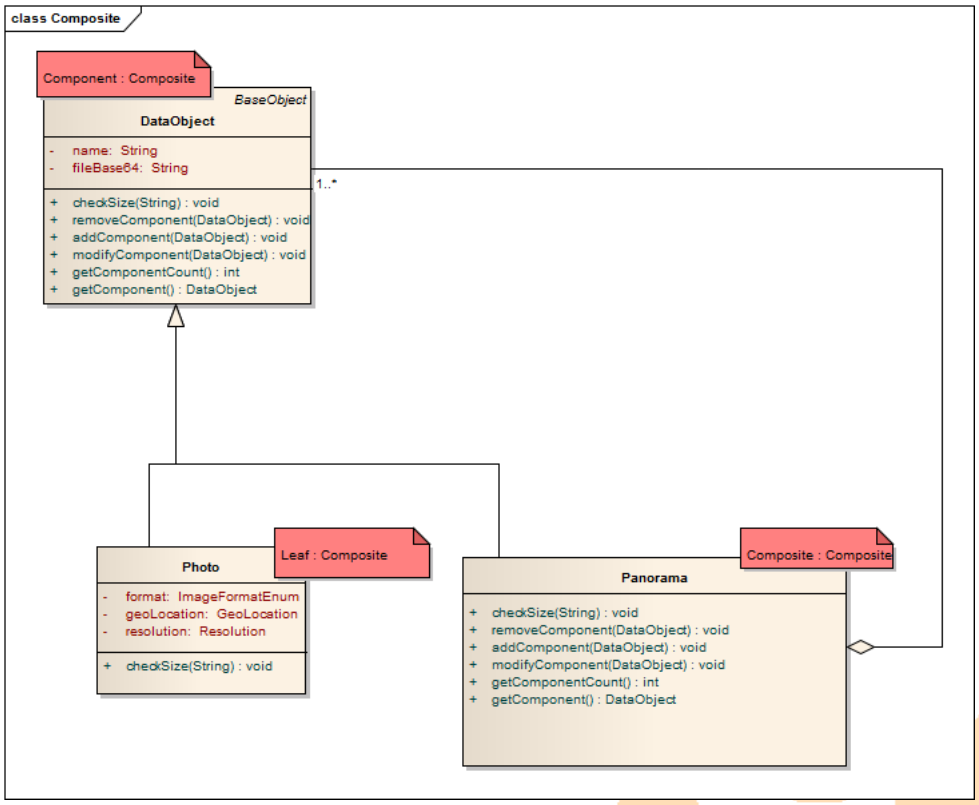
Obr. G.8: Singelton

G.9 Composite

Autor: Marek Uhlár

V rámci aplikácie sme identifikovali hierarchickú štruktúru. Vyskytuje sa pri vytváraní koláží. Koláž (objekt Collage) môže obsahovať obrázky (objekt *Photo*), videá (objekt *Video*) atď., ale aj ďalšie koláže.

Hierarchia je vytvorená prostredníctvom vzoru *Composite* (pozri G.9). Composite tu predstavuje objekt Collage a Leaf objekt Photo, Video atď. Preťažujú metódy objektu DataObject. Do objektu Collage sa cez metódu addComponent() dajú pridávať objekty typu Collage (Composite), ale aj typu Photo, Video atď. (Leaf). Nad objektmi v rámci koláže je možné iterovať bez ohľadu nato, či ide o Composite, alebo Leaf.

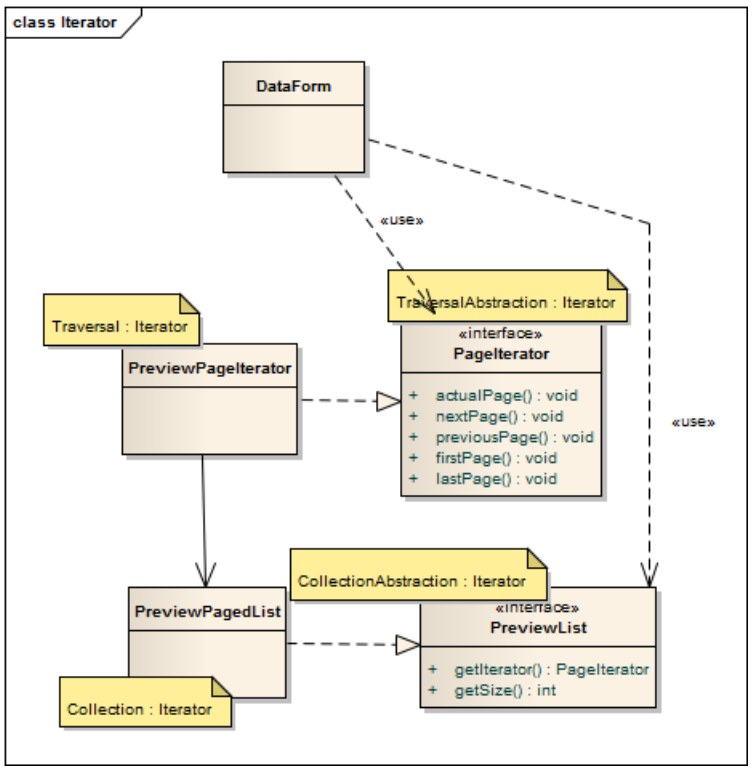


Obr. G.9: Composite

G.10 Iterator

Autor: Marek Uhlár

Prostredníctvom vzoru Iterator v aplikácií vytvárame špecializovaný iterátor (pozri G.10). Iteruje nad kolekciou objektov *PreviewObject*. Je ním umožnené iterovať po stránkach ukážok. Stránka obsahuje istý počet *PreviewObject* (napr. 50). Vždy je zobrazená len jedna stránka. Vďaka iterátoru môžeme s týmito stránkami jednoduchšie pracovať. Iterátor PageIterator prechádza nad kolekciou, konkrétne u nás ide o PreviewPagedList.



Obr. G.10: Iterator

Dodatok H

Ukážka procesu refactoring zdrojového kódu

V nasledovnej časti je opísaný proces refactoring na vybraných častiach kódu. Pre každý príklad jeho úpravy je identifikovaný pach a pravidlo odstránenia pachu.

H.1 Pach - Dátové zhluky

Autor: Peter Kajan

V nasledovnom kóde bol identifikovaný pach s názvom *Dátové zhluky*. Tento pach nastane, keď sa skupina atribútov vyskytuje na viacerých miestach pohromade.

```
public void addMetadata(List<String> v_r_v) {
        MetadataObjectRestClient metaRestClient = new MetadataObjectRestClient();
        String springContextFile = "application-context.xml";
        System.out.println(springContextFile);
        ApplicationContext context = new ClassPathXmlApplicationContext(springContextFile);
        RestTemplate restTemplate = (RestTemplate) context.getBean("restTemplate");
        metaRestClient.getRestClientManager().setRestTemplate(restTemplate);


        String[] valRelVal = null;
        if (v_r_v.size() > 0) {
            System.out.println("Pridavam nasledovne trojce metadatt : ");
            for (String es : v_r_v) {
                valRelVal = cliControler.splitString(es, "-", "[-]+");


                String object = valRelVal[0];
                String predicate = valRelVal[1];
                String subject = valRelVal[2];

                metaRestClient.add(object,predicate,subject);

            }
        }
    }
```

Pravidlo odstránenia – Vyňať triedu

Pach bol odstránený pravidlom s názvom *Vyňať triedu*, kedy sa pre skupinu atribútov definuje nová trieda. V tomto prípade sa položky RDF tripletu zlúčili do triedy *MetadataObject*. Ukážka upraveného kódu sa nachádza nižšie.

```
public void addMetadata(List<String> v_r_v) {
        MetadataObjectRestClient metaRestClient = new MetadataObjectRestClient();
        String springContextFile = "application-context.xml";
        System.out.println(springContextFile);
        ApplicationContext context = new ClassPathXmlApplicationContext(springContextFile);
        RestTemplate restTemplate = (RestTemplate) context.getBean("restTemplate");
        metaRestClient.getRestClientManager().setRestTemplate(restTemplate);

        String[] valRelVal = null;
        if (v_r_v.size() > 0) {
            System.out.println("Pridavam nasledovne trojce metadatt : ");
            for (String es : v_r_v) {
                valRelVal = cliControler.splitString(es, "-", "[-]+");

              MetadataObject metaObj = new MetadataObject();
              metaObj.setUri(valRelVal[0]);
              metaObj.setPredicate(valRelVal[1]);
              metaObj.setSubject(valRelVal[2]);

              metaRestClient.add(metaObj);

            }
        }
    }
```

H.2 Pach - Duplicitný kód

Autor: Peter Kajan
V nasledovnom kóde bol identifikovaný pach s názvom *Duplicitný kód*, ktorý nastáva, keď sa na viacerých miestach opakuje ten istý (resp. podobný) kód. V tomto prípade malo niekoľko metód rovnaké spracovanie výnimiek.

```
public void searchMetadata(Boolean id, List<String> IDobject) {
        //...

        try {

            List<MetadataObject> moList = new ArrayList<MetadataObject>(metaRestClient.list());
            System.out.println("Listing metada:");

            //...

        } catch (HttpServerErrorException ex) {
            logger.error("Server error", ex);
            System.out.println("Internal server error");
        } catch (HttpClientErrorException ex) {
            logger.error("Client error", ex);
            System.out.println("Client library error");
            //TODO: retry?
        }

}


public void addMetadata(List<String> v_r_v) {
        //...

       try {

            String result = metaRestClient.add(metaObj);
            logger.info("result " + result);

       } catch (HttpServerErrorException ex) {
            logger.error("Server error", ex);
            System.out.println("Internal server error");
       } catch (HttpClientErrorException ex) {
            logger.error("Client error", ex);
            System.out.println("Client error: " + ex.getMessage());
            //TODO: retry?
       }

}

//...
```

Riešenie - Vyňať kód do aspektu

Pach bol odstránený vyňatím duplicitného kódu do aspektu. Ukážka upravených častí kódu sa nachádza nižšie.

```
@Aspect
public class EhCliAspect {

    org.slf4j.Logger logger = org.slf4j.LoggerFactory.getLogger(EhCliAspect.class);

    @Around("execution(public * MetadataController.*(..)) || execution(public *
        DataController.*(..))")
    private Object anyOldTransfer(ProceedingJoinPoint pjp) throws Throwable {
        logger.info("eh aspect start");
        try {
            return pjp.proceed(pjp.getArgs());
        } catch (HttpServerErrorException ex) {
            logger.error("Server error", ex);
            System.out.println("Internal server error");
        } catch (HttpClientErrorException ex) {
            logger.error("Client error", ex);
            System.out.println("Client library error");
            //TODO: retry?
        }
        return new Object();
    }
}

public void searchMetadata(Boolean id, List<String> IDobject) {
    //...

    List<MetadataObject> moList = new ArrayList<MetadataObject>(metaRestClient.list());
    System.out.println("Listing metada:");

    //...
}

public void addMetadata(List<String> v_r_v) {
    //...

    String result = metaRestClient.add(metaObj);
    logger.info("result " + result);
}

//...
```

H.3 Pach - Duplicitný kód

Autor: Katarína Valaliková

Pri implementácií autentifikácie a autorizácie vznikol pach s názvom *Duplicitný kód*, ktorý možno vidieť v nasledovnom zdrojovom kóde. Pri spustení ktorejkoľvek funkcie bolo potrebné overiť či je používateľ prihlásený. Len prihlásený a autentifikovaný používateľ môže pracovať s aplikáciou.

```
public class DataController{

     public void addObject(String path, List<String> nameObject) {

          Authentication authentication =
          SecurityContextHolder.getContext().getAuthentication();
             if (authentication == null) {
                 System.out.println("Sorry, but you must be logged in. Log in using -l
                 (username) -pw   (password)");
                 return;

             }
           .........
      }
     ........
}

public class AccountController {

    void newAccount(String nl, String npw) {

          Authentication authentication =
          SecurityContextHolder.getContext().getAuthentication();
             if (authentication == null) {
                 System.out.println("Sorry, but you must be logged in. Log in using -l
                 (username) -pw   (password)");
                 return;

             }
        .........
      }
     ........
}

public class MetadataController {

    public void addMetadata(List<String> v_r_v) {

          Authentication authentication =
          SecurityContextHolder.getContext().getAuthentication();
             if (authentication == null) {
                 System.out.println("Sorry, but you must be logged in. Log in using -l
                 (username) -pw   (password)");
                 return;

          }
            .........
      }
     ........
}
```

Riešenie: Vyňať metódu do aspektu

V prípade autentifikácie používateľa ide o pretínajúce sa záležitosti, ktoré spôsobujú duplicitný kód. Pach bol odstránený vyňatím duplicitného kódu do aspektu. Ukážka upravených častí sa nachádza nižšie.

```
@Aspect
public class AuthenticationAspect {

    org.slf4j.Logger logger = org.slf4j.LoggerFactory.getLogger(AuthenticationAspect.class);

    @Pointcut("within(sk.fiit.team17cf.stfu.cli.MetadataController )")
    public void metadataPointcut() {
    }

    @Pointcut("within(sk.fiit.team17cf.stfu.cli.DataController)")
    public void dataPointcut() {
    }

    @Pointcut("within(sk.fiit.team17cf.stfu.cli.AccountController)")
    public void accountPointcut(){
    }

    @Around("metadataPointcut() || dataPointcut() || accountPointcut() ")
    public void tryUserAuthenticated(ProceedingJoinPoint pjp) throws Throwable {
        logger.info("User verification start");
        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
        if (authentication == null) {
            System.out.println("Sorry, but you must be logged in. Log in using -l
             (username) -pw (password)");
            return;
        } else {
            pjp.proceed();
            return;
        }
    }

}
```

H.4 Pach - Duplicitný kód

Autor: Marek Uhlár
Vo viacerých metódach sa nachádzalo ukladanie súboru. Ide o zbytočný duplicitný kód, ktorý spôsobuje neprehľadnosť.

```
public void giveObject(String path, Boolean metadata, List<String> IDObject, Boolean preview) {
.....

            File f = null;

            if (!path.equals("(default)")) {
                f = new File(path + dataObject.getName());
                logger.info("Objekty pridavam do zlozky " + path);
            } else {
                f = new File(dataObject.getName());
            }

            FileOutputStream fos = null;
            try {
                fos = new FileOutputStream(f);
                byte[] fileByte = (new BASE64Decoder()).decodeBuffer(dataObject.getFile());
                fos.write(fileByte);
                fos.close();
            } catch (FileNotFoundException ex) {
                logger.error("Exception {}", ex);
            } catch (IOException ex) {
                logger.error("Exception {}", ex);
            }

.....

}

public void searchObject(Boolean tag, Boolean geolocation, String relationship, ...) {
.....
       for (PreviewObject po : previewSet) {


             File f = new File(path + previewObject.getName());

             FileOutputStream fos = null;
             try {
                  fos = new FileOutputStream(f);
                  byte[] fileByte = (new BASE64Decoder()).decodeBuffer(previewObject.getFile());
                  fos.write(fileByte);
                  fos.close();
             } catch (FileNotFoundException ex) {
                       logger.error("Exception {}", ex);
             } catch (IOException ex) {
                       logger.error("Exception {}", ex);
             }

       }
.....
}
```

Riešenie - vyňať metódu

Pach sme vyriešili vyňatím duplicitných častí kódu do metódy *saveFile*. Dlhé neprehľadné časti sú teda nahradené zavolaním metódy.

```
private void saveFile(String path) {

        File f = new File(path);

        FileOutputStream fos = null;
        try {
            fos = new FileOutputStream(f);
            byte[] fileByte = (new    BASE64Decoder()).decodeBuffer(previewObject.getFile());
            fos.write(fileByte);
            fos.close();
        } catch (FileNotFoundException ex) {
            logger.error("Exception {}", ex);
        } catch (IOException ex) {
            logger.error("Exception {}", ex);
        }
}

public void giveObject(String path, Boolean metadata, List<String> IDObject, Boolean preview) {


saveFile(path + previewObject.getName());

}


public void searchObject(Boolean tag, Boolean geolocation, String relationship, ...) {

saveFile(path + dataObject.getName());

}
```

H.5 Pach - Duplicitný kód

Autor: Peter Kajan, Marek Uhlár
Vo viacerých metódach triedy *DataController* programátor zbytočne vytváral a inicializoval niektoré premenné. Tato inicializácia bola vyňatá do metódy.

```
public class DataController {

public void giveObject(String path, Boolean metadata, List<String>IDObject, Boolean preview) {

    String springContextFile = "application-context.xml";
    ApplicationContext context = new ClassPathXmlApplicationContext(springContextFile);
    RestTemplate restTemplate = (RestTemplate) context.getBean("restTemplate");
    DataObjectRestClient dataRestClient = context.getBean("dataObjectRestClient");

.....
public void rmObject(List<String> IDObject) {

    String springContextFile = "application-context.xml";
    ApplicationContext context = new ClassPathXmlApplicationContext(springContextFile);
    RestTemplate restTemplate = (RestTemplate) context.getBean("restTemplate");
    DataObjectRestClient dataRestClient = context.getBean("dataObjectRestClient");

.....
public void searchObject(Boolean tag, Boolean geolocation, String relationship,...) {

    String springContextFile = "application-context.xml";
    ApplicationContext context = new ClassPathXmlApplicationContext(springContextFile);
    RestTemplate restTemplate = (RestTemplate) context.getBean("restTemplate");

.....
}
```

Riešenie - Vyňať metódu a presunúť metódu

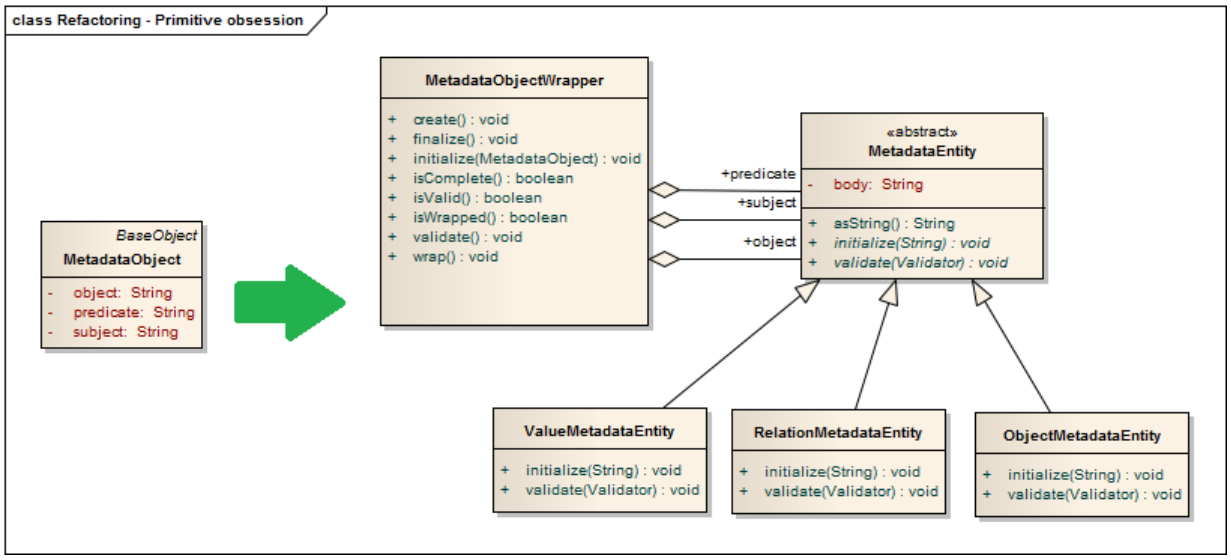
```
public class DataController {


DataRestClient dataRestClient;

public loadContext() {
    String springContextFile = "application-context.xml";
    ApplicationContext context = new ClassPathXmlApplicationContext(springContextFile);
    RestTemplate restTemplate = (RestTemplate) context.getBean("restTemplate");

    dataRestClient = context.getBean("dataObjectRestClient");

        //...
}


public initialize() {
    //..

    loadContext();

    //...
}

}
```

H.6 Pach - Primitívna obsesia

Autor: Peter Kajan

Pach primitívna obsesia je identifikovaný na miestach, kde sa používajú primitívne typy namiesto vhodnejších a komplexnejších tried. Tento pach sa vo vyvíjanej aplikácii vyskytol napríklad v triede *MetadataObject*, v ktorej atribúty *object*, *predicate* a *subject* boli typu reťazec. Na obr. H.1 je uvedené odstránenie tohto pachu, kedy spomínané atribúty boli nahradené za triedy implementujúce rozhranie *MetadataEntity*.

Pravidlo riešenia: Nahradiť dátovú položku objektom



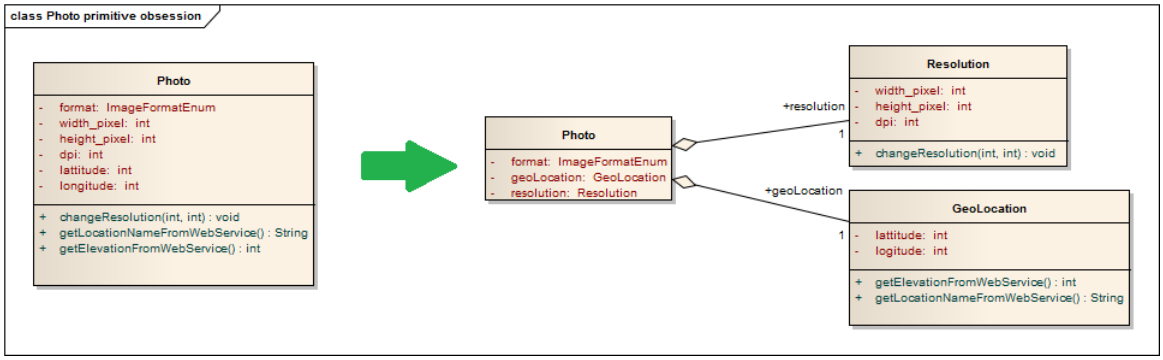
Obr. H.1: Refactoring - Primitive obsession

H.7 Pach - Primitívna obsesia

Autor: Marek Uhlár

V triede *Photo* sa používalo veľké množstvo atribútov primitívnych typov. Ide o pach primitívna obsesia. Vyriešili sme ho zabalením niektorých atribútov do samostatných objektov. Takisto sme do týchto novovzniknutých objektov presunuli metódy, ktoré sem logicky viac zapadali ako do triedy *Photo*.

Pravidlo riešenia: Nahradiť dátovú položku objektom



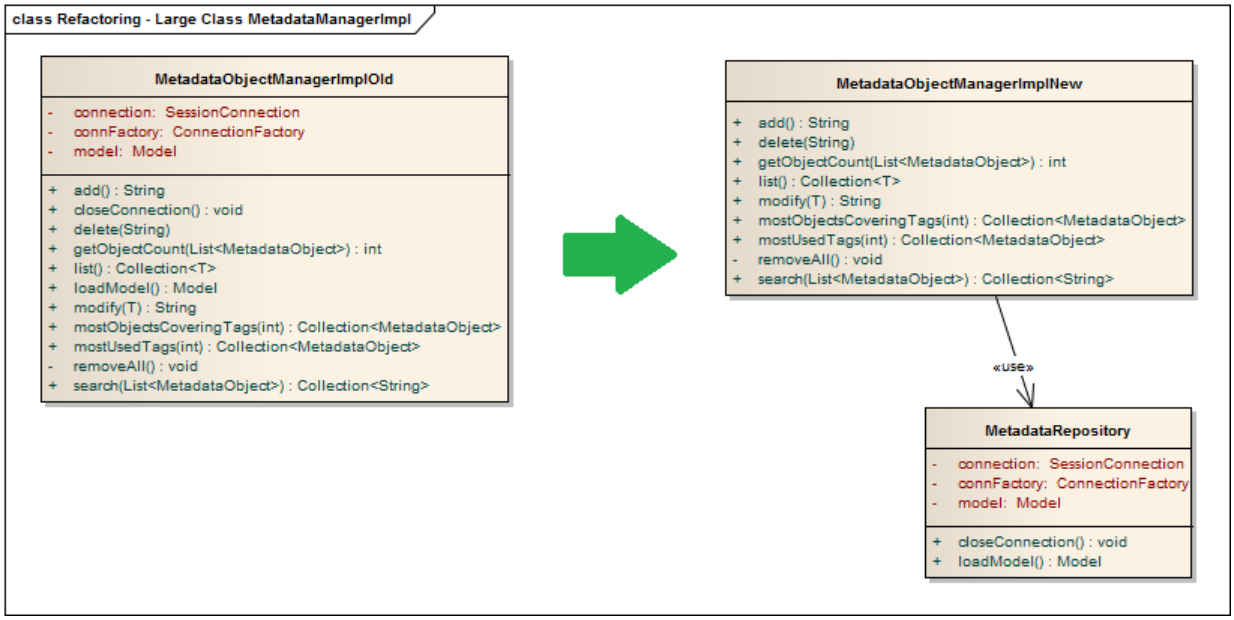
Obr. H.2: Photo primitive obsession

H.8 Pach - Veľká trieda

Autor: Peter Kajan
Trieda *MetadataManagerImpl* zahŕňala v sebe príliš veľa funkcionality. Pripájala sa do databázy a takisto manipulovala s metadátami. V tejto triede bol identifikovaný pach Veľká trieda.

Pravidlo riešenia: Vyňať triedu

Pach bol odstránený vyňatím triedy MetadataRepository, v ktorej bolo implementované spomínané pripájanie do databázy. Teraz v prípade zmeny databázy bude postačujúce implementovať triedu zabezpečujúcu pripájanie, pričom trieda MetadataManagerImpl ostane nezmenená.



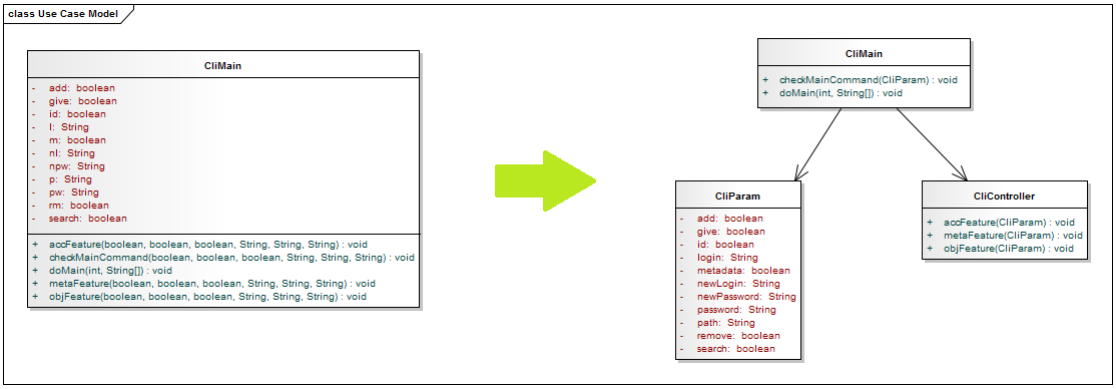
Obr. H.3: Refactoring - Large Class MetadataManagerImpl

H.9 Pachy - Veľká trieda, Dátové zhluky

Autor: Katarína Valaliková

Pach - Veľká trieda V triede zobrazenej na obrázku nižšie bol identifikovaný pach s názvom *Veľká trieda*. Ide o hlavnú triedu, pomocou ktorej je aplikácia spúšťaná. Trieda obsahuje nevhodné metódy.

Pravidlo riešenia: Vyňať metódu Po vyňatí triedy sa trieda CliMain rozdelila na CliMain a CliController



Obr. H.4: Refactoring - Veľká trieda, dátové zhluky

Pach – Dátové zhluky

Druhý pach, identifikovaný v tejto triede je pach s názvom *Dátové zhluky*. Ide o skupinu parametrov potrebných na načítavanie používateľových vstupov z konzoly.

Pravidlo riešenia: Zaviesť objekt pre parametre

Zavedením nového objektu pre parametre dostávame ďalšiu triedu. Odstránenie oboch pachov je znázornené na obrázku H.4.

H.10 Pach - Dlhý zoznam parametrov

Autor: Katarína Valaliková

Nižšie je zobrazený zdrojový kód metódy, ktorá obsahuje veľa vstupných parametrov.

V tomto zdrojovom kóde bol identifikovaný pach *Dlhý zoznam parametrov*.

```
public void checkMainCommand(List<String> arguments, Boolean add, String p, Boolean search, Bo
    Boolean rm, Boolean m, Boolean t, Boolean g, String r, String ri, String s,
    String si, String z, int size, Boolean id, Boolean news, Boolean ch, Boolean re,
    String l, String pw, String e, String nl, String npw, Boolean prev, Boolean help,
    Boolean list){

      String mainCommand = "mainCommand";
      if (arguments.size() > 0) {
          mainCommand = arguments.get(0);
          arguments.remove(0);
      }
      if (mainCommand.equals("obj")) {

          cliControler.objFeature( search, add, give, rm, t, g, r, ri, s, si, z, arguments,
                size, p, m, prev, help, list);

      } else if (mainCommand.equals("meta")) {

          cliControler.metaFeature( rm, add, search, id, arguments, help );

      } else if (mainCommand.equals("acc")) {

          cliControler.accFeature( news, ch, re, l, pw, rm, e, nl, npw, arguments, help );

      }

}
```

Riešenie: Zaviesť objekt pre parametre

Pach sme vyriešili pravidlom zaviesť objekt pre parametre. Ukážka zdrojového kódu po odstránení pachu sa nachádza nižšie.

```
public void checkMainCommand(CliParam cliParam){

        String mainCommand = "mainCommand";
        if (arguments.size() > 0) {
            mainCommand = arguments.get(0);
            arguments.remove(0);
        }

        if (mainCommand.equals("obj")) {

            cliControler.objFeature(cliParam);

        }

        else if (mainCommand.equals("meta")) {

            cliControler.metaFeature(cliParam);

        }

        else if (mainCommand.equals("acc")) {

            cliControler.accFeature(cliParam);

        }
    }
```

H.11 Pach - Komentár

Autor: Marek Uhlár

Kvôli väčšej dĺžke metódy *add* a zrozumiteľnosti bola časť kódu okomentovaná. Ide o pach, ktorý nám indikuje zbytočne dlhú metódu. Komentár nám napovedá, že časť kódu ktorej sa týka, je určitým spôsobom ucelená a mohla by tvoriť samostatnú metódu.

```
public String add(DataObject baseObject) throws OperationFailedException {

        String userName = userObjectController.getUserObject().getUserName();
        DataObject dataObject = (DataObject) baseObject;

        try {
            //check if exist directory for user, creates unique name for file, saves it a
            returns path to file
            if (checkDirectory(userName)) {

                String fpath = getUniqueName(userName + File.separator + dataObject.getName());
                File f = new File(fpath);
                FileOutputStream fos = new FileOutputStream(f);

                fos.write(FileUtils.decodeBase64StringToByteArray(dataObject.getFile()));

                fos.close();

                return f.getAbsolutePath();

            } else {
                throw new FileNotFoundException("Error creating " + userName + " directory.");
            }

        } catch (FileNotFoundException ex) {
            ...
        } catch (IOException ex) {
            ...
        }

    }
```

Riešenie - Vyňať metódu

Riešenie spočíva vo vyňatí časti kódu, ktorú opisuje komentár do samostatnej metódy. Následne nahradíme komentár s príslušnou časťou kódu volaním metódy.

```
public String add(DataObject baseObject) throws OperationFailedException {

        String userName = userObjectController.getUserObject().getUserName();
        DataObject dataObject = (DataObject) baseObject;

        try {
            if (checkDirectory(userName)) {

                return fileManager.saveFile(dataObject.getFile(), userName + File.separator +
                        dataObject.getName());

            }
            else {
                throw new FileNotFoundException("Error creating " + userName + " directory.");
            }

        } catch (FileNotFoundException ex) {
            ...
        } catch (IOException ex) {
            ...
        }

}

public String saveFile(String fileToWrite, String filePath) throws ... {

         String fpath = getUniqueName(userName + File.separator + dataObject.getName());
         File f = new File(fpath);
         FileOutputStream fos = new FileOutputStream(f);

         fos.write(FileUtils.decodeBase64StringToByteArray(dataObject.getFile()));

         fos.close();

         return f.getAbsolutePath();

}
```

H.12 Pach - Zreťazené správy

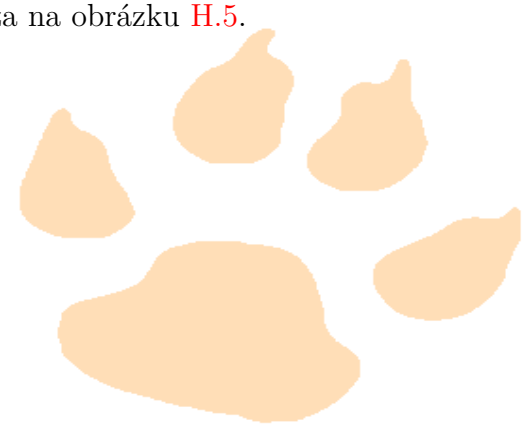
Autor: Katarína Valaliková
Pach zreťazené správy nastáva, keď sú triedy navzájom prepletené. Jedna trieda využíva druhú a naopak, druhá prvú. V tomto prípade došlo k zreťazeným správam z tohto dôvodu:

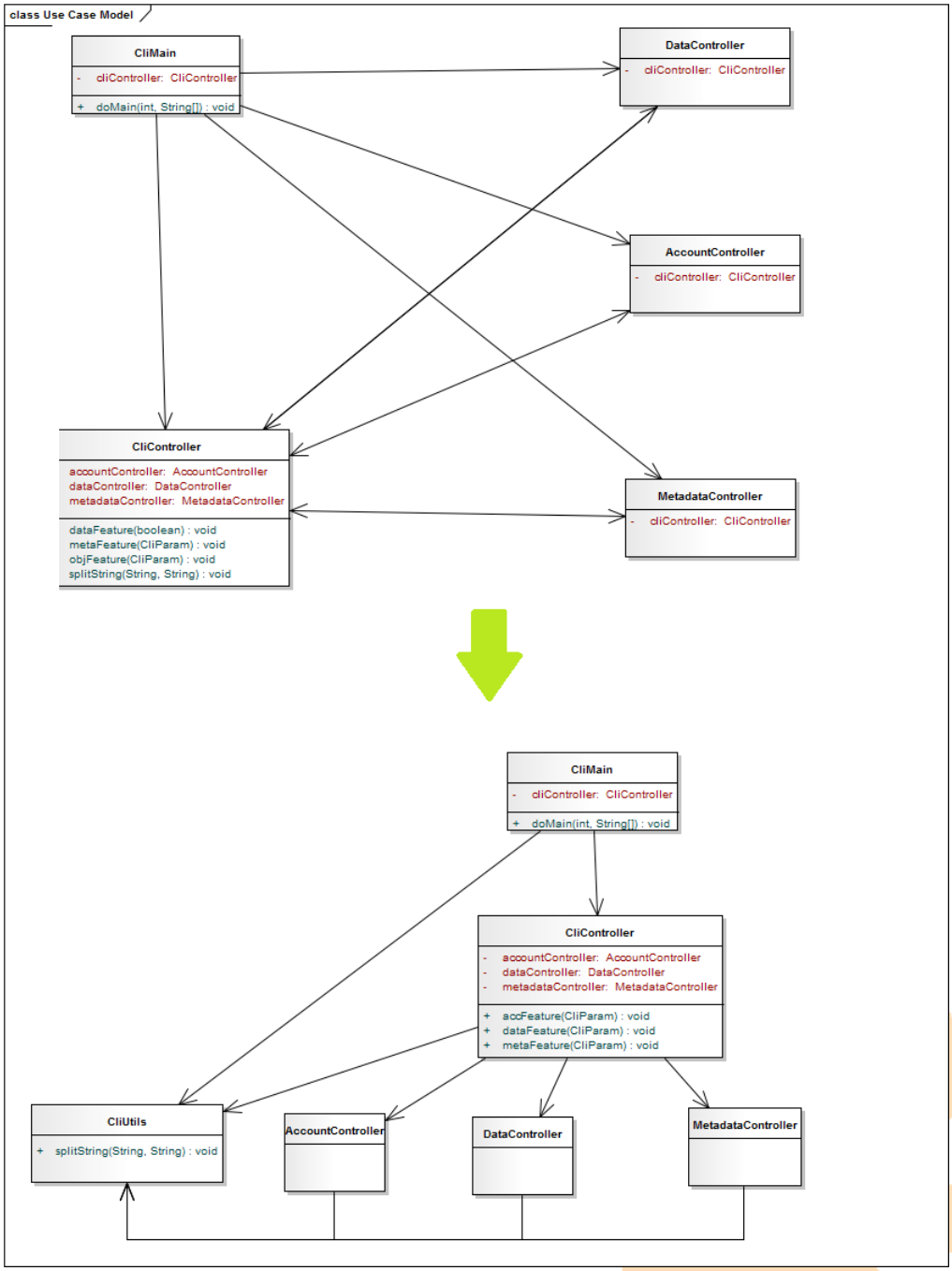
- *CliController* obsahoval metódu *splitString()*, ktorú využívali *AccountController, DataController, MetadataController a CliMain,*
- *DataController* obsahoval metódu *dataFeature()*, ktorú využíval *CliController*,
- *MetadataController* obsahoval metódu *metaFeature()*, ktorú využíval *CliController* a
- *AccountController* obsahoval metódu *accFeature()*, ktorú využíval *CliController*

Riešenie – vyňať metódu a presunúť metódu
Pach zreťazené správy je možné vyriešiť vyňatím metódy a presunutím metódy. V tomto prípade sa jedná o metódu *splitString()*, ktorú využívali triedy

- CliMain,
- AccountController,
- DataController a
- MetadataController.

Túto metódu sme vyňali a presunuli z triedy *CliController* do novovzniknutej triedy *CliUtils*. Pach a jeho vyriešenie sa nachádza na obrázku H.5.





Obr. H.5: Zreťazné správy