Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

WebGephi - Webové rozhraní pro Gephi

Bc. Václav Čokrt

Vedoucí práce: Ing. Jaroslav Kuchař

4. května 2014

Poděkování

Doplňte, máte-li komu a za co děkovat. V opačném případě úplně odstraňte tento příkaz.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či spracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 4. května 2014 .....

Odkaz na tuto práci

Čokrt, Václav. *WebGephi - Webové rozhraní pro Gephi.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2014.

Abstrakt

Cílem této práce je vytvoření aplikace *WebGephi*, která zpřístupní funkcionalitu aplikace *Gephi*[1] pomocí standardních webových technologií. *Gephi* je opensource desktopová aplikace sloužící k vizualizaci a manipulaci s grafy. WebGephi bude poskytovat jednotné REST[1] rozhraní k funkcionalitě *Gephi*. Aplikace bude obsahovat také řízení přístupu - správu uživatelů a klientských aplikací. Součástí řešení bude i ukázková klientská aplikace - grafická nadstavba demonstrující funkcionalitu *WebGephi*.

Klíčová slova Gephi, Gephi Toolkit, graf, REST, webové služby, webová aplikace

Abstract

TODO the same in english

Keywords Gephi, Gephi Toolkit, graph, REST, web services, web application

[1]Representational State Transfer - architektura webového rozhraní (webových služeb)

Obsah

Seznam ukázek zdroj. kódu **xv**

Úvod **1**

1 Cíl práce **3**

2 Analýza **5**
2.1 Gephi 5
2.2 Komponenty relevantní pro WebGephi 12
2.3 Gephi Toolkit 13
2.4 Možnosti autentizace a autorizace 19

3 Návrh **25**
3.1 REST rozhraní 25
3.2 Autorizace 25
3.3 Správa uživatelů a klientských aplikací 25
3.4 Struktura aplikace 25
3.5 GUI 25

4 Realizace **27**
4.1 Požité technologie 27
4.2 Server TODO popis struktury a zajímavých částí 27
4.3 Klient (konektor) TODO popis struktury a zajímavých částí . . 27
4.4 Klientská aplikace (konektor) TODO popis struktury a zajímavých částí 27

Závěr **29**

Literatura **31**

A Gephi **33**

B Seznam použitých zkratek **35**

C Obsah přiloženého CD **37**

Seznam obrázků

2.1 Záložka „Přehled“ 6
2.2 Záložka „Laboratoř dat“ 7
2.3 Záložka „Náhled“ 8
2.4 Změna rozložení podle funkce `Foce Atlas 2` 9
2.5 Report po aplikaci funkce `PageRank` 9
2.6 Hodnocení barvou a velikostí podle funkce `PageRank` 10
2.7 Rozdělení na oddíly podle hodnoty atributu `Modularity Class` . . 10
2.8 Aplikace filtru `Rozsah` na základě hodnoty funkce `PageRank` 11
2.9 Struktura tříd pro výpočet funkcí „Rozložení“ 15
2.10 Struktura tříd pro výpočet „Statistik a metrik“ 16
2.11 Struktura tříd pro výpočet „Hodnocení“ 17
2.12 Průběh Basic autorizace 20
2.13 Ukázkový příklad využití OAuth protokolu 22
2.14 Průběh OAuth autorizace 22

A.1 Porovnání formátů vhodných pro ukládání grafů 33

Seznam ukázek zdroj. kódu

2.1 Ukázka použití Lookup API 14
2.2 Aplikace funkce rozložení `YifanHu` 15
2.3 Aplikace statistické funkce `GraphDistance` 15
2.4 Změna velikosti uzlu podle hodnoty atributu „betweenesscentrality“ 16
2.5 Import grafu do Workspace 18
2.6 Výřez z implementace třídy `DefaultProcessor` 18

Úvod

„Graf je základním objektem teorie grafů. Jedná se o reprezentaci množiny objektů, u které chceme znázornit, že některé prvky jsou propojeny. Objektům se přiřadí vrcholy a jejich propojení značí hrany mezi nimi. Grafy slouží jako abstrakce mnoha různých problémů. Často se jedná o zjednodušený model nějaké skutečné sítě (například dopravní), který zdůrazňuje topologické vlastnosti objektů (vrcholů) a zanedbává geometrické vlastnosti, například přesnou polohu.[2]“

Graf tedy můžeme chápat jako zjednodušený obraz nějaké skutečnosti - seznam lidí a vztahů mezi nimi, množina webových stránek a hypertextových odkazů, cokoli, co lze znázornit jako množinu uzlů a hran (vztahů mezi nimi).

Gephi[1] je desktopová platforma sloužící k analýze grafů. Umožňuje interaktivně měnit rozložení grafů na základě jejich struktury, vypočítávat metriky (PageRank, shlukovací koeficient, ...), filtrovat uzly podle jejich vlastností a mnoho dalšího. To vše slouží k tomu, aby uživatel byl schopný v grafu najít skryté závislosti a mohl lépe pochopit (a vizualizovat) strukturu grafu.

Jednou z hlavních výhod *Gephi* je jeho rozšiřitelnost. Kdokoli může vytvořit svou vlastní funkci k manipulaci s grafem a ve formě pluginu ji přidat do aplikace.

Aplikace *WebGephi* by měla zachovat tyto vlastnosti a navíc přidat výhody plynoucí ze standardizovaného rozhraní webové aplikace.

Kapitola 1

Cíl práce

Cílem této práce je vytvořit webovou aplikaci poskytující funkcionalitu *Gephi*. Hlavní rozhraní této aplikace bude založeno na architektuře REST.

„ REST (Representational State Transfer) – je architektura rozhraní, navržená pro distribuované prostředí. REST navrhnul a popsal v roce 2000 Roy Fielding (jeden ze spoluautorů protokolu http) v rámci disertační práce Architectural Styles and the Design of Network-based Software Architectures. V kontextu práce je nejzajímavější kapitola 5, ve které Fielding odvozuje principy RESTu na základě známých přístupů k architektuře. Rozhraní REST je použitelné pro jednotný a snadný přístup ke zdrojům (resources). Zdrojem mohou být data, stejně jako stavy aplikace (pokud je lze popsat konkrétními daty). REST je tedy na rozdíl od známějších XML-RPC či SOAP, orientován datově, nikoli procedurálně. Všechny zdroje mají vlastní identifikátor URI a REST definuje čtyři základní metody pro přístup k nim.“[3]

WebGephi bude sloužit jen jako poskytovatel služeb pro jiné (klientské) aplikace. Tyto aplikace budou pomocí webových služeb přistupovat k *WebGephi* (nahrávat grafy, aplikovat na ně funkce, exportovat výsledky, ...) a komunikovat s koncovým uživatelem přes vlastní grafické rozhraní. Klientské aplikace mohou být webové, desktopové i mobilní aplikace. Přístup klientských aplikací samozřejmě bude také třeba řídit. *WebGephi* tedy kromě REST rozhraní bude poskytovat také grafické rozhraní pro registraci a správu uživatelů a klientských aplikací.

Hlavní výhody *WebGephi* budou jednoduché, standardizované, deklarativní rozhraní. *Gephi* sice poskytuje knihovnu pro práci s grafy, *Gephi Toolkit*[4], ta je však dosti složitá na použití. Klientské aplikace budou moci jednoduše přistupovat k tomuto rozhraní pomocí HTTP protokolu, na kterém je v naprosté většině založena REST architektura. To umožní jak jednoduché prozkoumávání rozhraní (např. pomocí webového prohlížeče), tak strojové zpracování koncovými klientskými aplikacemi.

Pro koncového uživatele je hlavní výhodou možnost použití bez nutnosti instalace (ve spolupráci s klientskými aplikacemi), použití jako centrálního

úložiště grafů a možnost sdílení s jinými uživateli.

WebGephi bude postaveno nad již zmíněnou knihovnou *Gephi Toolkit*. Ta obsahuje základní moduly *Gephi* (bez GUI[2] funkcionality) ve formě jednoduché Java knihovny. V prvním kroku bude třeba určit, jaká funkcionalita bude implementována ve *WebGephi*. Následně bude třeba analyzovat strukturu knihovnu *Gephi Toolkit* a s její pomocí tuto funkcionalitu implementovat. Dále bude potřeba navrhnout a implementovat strukturu REST rozhraní a způsob autentizace uživatelů.

Součástí práce je i implementace ukázkové klientské aplikace. Ta bude využívat naprostou většinu funkcionality WebGephi a demonstrovat její funkčnost.

[2]Graphical User Interface - grafické uživatelské rozhraní

Kapitola 2

Analýza

Cílem této kapitoly je hlubší náhled do problematiky a analýza dostupných nástrojů. Na konci kapitoly bychom měli mít jasno v tom, zda a jak je tento úkol řešitelný.

Nejdříve si představíme aplikaci *Gephi* a její možnosti. Určíme, jakou funkcionalitu bude možné a vhodné přenést do *WebGephi*. Dále představíme strukturu knihovny *Gephi Toolkit* a nastíníme, zda a jakým způsobem bude možné požadovanou funkcionalitu implementovat. Aplikace bude také muset řešit autentizaci a autorizaci přístupu. V poslední části si tedy také představíme standardní řešení tohoto problému.

2.1 Gephi

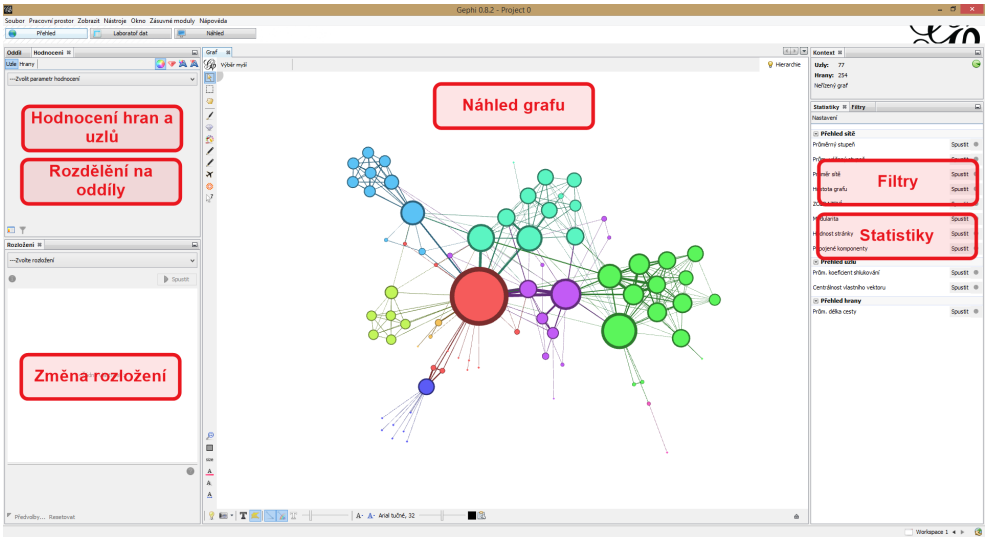
„Gephi je platforma pro vizualizaci a prozkoumávání všech druhů sítí a komplexních systémů, dynamických a hierarchických grafů. Je dostupný pro Windows, Linux a Mac OS x. Gephi je open-source a zdarma.“[3]

Jedná se tedy o volně dostupný, bezplatný nástroj pro práci s grafy. První veřejně dostupná verze měla číslo 0.6alpha1 a vyšla v roce 2008. Původně se jednalo o studentskou práci francouzských studentů[4]. V současnosti se o jeho vývoj stará „Gephi Consortium“[5].

Přesto, že i současná verze (0.8.2-beta) je stále betaverze, jedná se o široce používaný software s rozsáhlou funkcionalitou. Budoucí verze číslo 0.9 by měla být výrazným krokem v evoluci *Gephi*, bude obsahovat změny v samotném jádru aplikace. Jedná se o velmi živý projekt, který je v neustálém vývoji.

[3]Volný překlad z oficiálních stránek *Gephi*[1]. Orig.: „Gephi is an interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs. Runs on Windows, Linux and Mac OS X. *Gephi* is open-source and free.“

[4]The University of Technology of Compiègne (Université de Technologie de Compiègne or UTC)



Obrázek 2.1: Záložka „Přehled“

Gephi je napsáno v programovacím jazyku Java. Je založeno na *NetBeans*[5] platformě, grafické rozhraní využívá framework Swing.

2.1.1 Struktura aplikace

Zde si představíme strukturu aplikace *Gephi* a její uživatelské rozhraní.

Základní jednotkou při práci s *Gephi* je „Projekt“. Aplikace může pracovat vždy jen s jedním projektem. Projekt je nejvyšší jednotkou v *Gephi*, může být ukládán a načítán z disku a může obsahovat jeden nebo více „Pracovních prostorů“ (Workspace). V aplikaci je vždy aktivní jen jeden Workspace. Workspace představuje pracovní plochu, na které se mohou vytvářet a editovat grafy.

Uživatelské rozhraní se skládá ze tří záložek - Přehled, Laboratoř dat a Náhled.

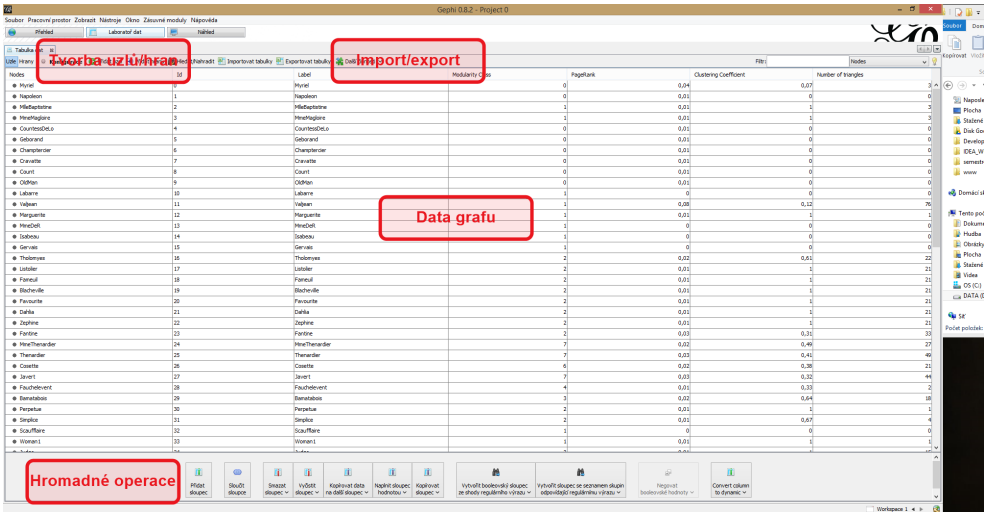
2.1.1.1 Záložka Přehled

Jedná se o nejdůležitější záložku (obr. 2.1). Na středu je vidět náhled grafu a po stranách jsou dostupné funkce, které lze na graf aplikovat. Po aplikaci funkce je graf okamžitě překreslen. S grafem lze manipulovat i ručně pomocí nástrojů na svislé liště na levé straně náhledu grafu.

2.1.1.2 Záložka Laboratoř dat

Tato záložka (obr. 2.2) slouží k úpravě zdrojových dat grafu. Uzly a hrany lze vkládat ručně nebo importovat z různých zdrojů (soubor, databáze, ...). Na

[5] https://*NetBeans*.org/



Obrázek 2.2: Záložka „Laboratoř dat“

středu je vidět seznam uzlů a hran aktuálního grafu. S daty lze provádět po sloupcích (atributech) i hromadné operace.

2.1.1.3 Záložka Náhled

Poslední záložka (obr. 2.3) slouží k vizuální úpravě grafu. Typicky se používá před exportem výsledku do obrázku. Lze nastavovat velikost hran, uzlů, nastavení popisků, průhlednost, . . . Výsledný graf lze exportovat ve formátu png, svg a pdf.

2.1.2 Funkcionalita

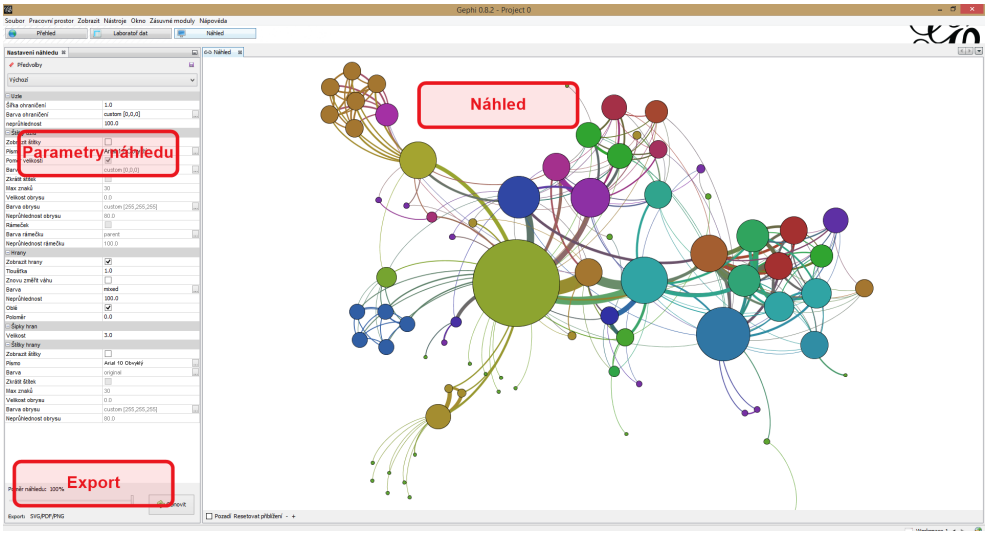
Zde podrobně popíšeme funkcionalitu *Gephi*.

2.1.2.1 Import a export

Gephi ukládá projekty do svého interního formátu `gephi`. Ten kromě samotného grafu ukládá i strukturu projektu a jeho nastavení. Pro výměnu dat mezi aplikacemi jsou důležitější standardizované formáty. *Gephi* umí načítat téměř jakýkoli formát vhodný pro ukládání grafu, včetně `csv`, `graphML` a `gexf`. Do těchto formátů umí samozřejmě grafy i exportovat.

Nejvhodnější z těchto formátů je jednoznačně formát `gexf`[6]. Ten je schopen uložit všechny důležité informace včetně hierarchických a dynamických grafů a je prakticky standardem to tomto oboru. Jedná se o xml formát s relativně jednoduchou strukturou[6]. Základem je seznam uzlů a hran (včetně

[6]Přesná struktura je definována standardně pomocí xml schématu: http://www.gexf.net/1.2draft/gexf.xsd



Obrázek 2.3: Záložka „Náhled“

jejich atributů).

V příloze A.1 můžete vidět porovnání dostupných formátů.

Gephi je schopno načítat grafy také SQL databáze (stačí specifikovat potřebný SQL dotaz).

Kromě datového exportu je dostupný taká export vizuální reprezentace grafu. Dostupné jsou formáty `pdf`, `svg` a `png`.

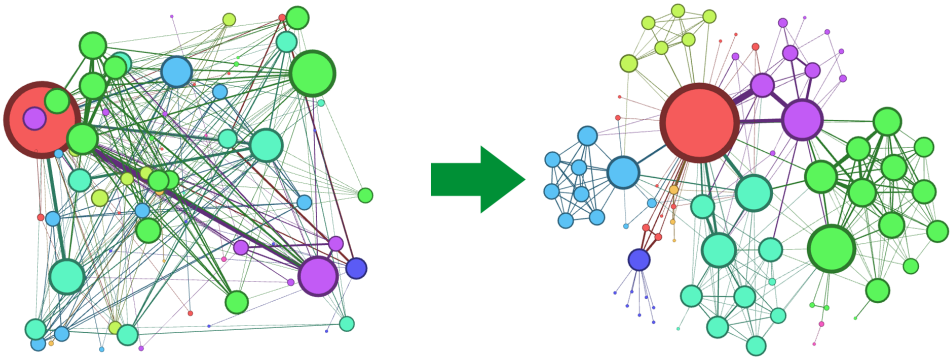
2.1.2.2 Grafové funkce

Hlavní silou *Gephi* je manipulace s grafem jako s celkem. *Gephi* už v základu obsahuje množství takovýchto funkcí a je možné je dále rozšířit pomocí pluginů. Funkce jsou podle zaměření rozděleny do několika skupin.

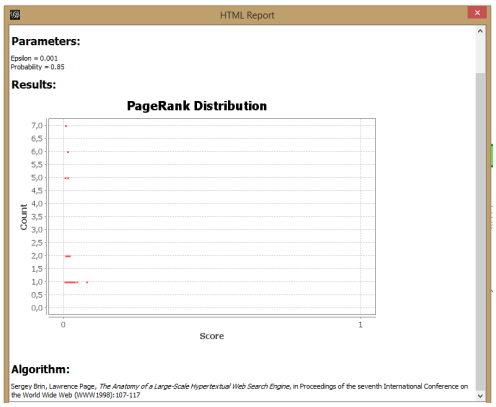
Rozložení Funkce pro změnu rozložení slouží k systematickému přeskupení uzlů. Může se jednat o velmi jednoduché funkce, např. `Otočení podle směru hodinových ručiček`, která jen otočí graf o definovaný úhel, ale i o velmi specifikované funkce, jako je `Force Atlas`, který přeskupuje uzly podle jejich vazeb k okolí. Obecně se jedná o jakékoli funkce, které mění poziční atributy uzlů.

Některé funkce podporují interaktivní režim, kdy se funkce spustí v nekonečné smyčce, takže okamžitě vidíme, jaký vliv na rozložení grafu má změna atributů. Když jsme s rozložením spokojeni, stačí funkci zastavit. Ukázku změny rozložení můžete vidět na obr. 2.4.

Statistiky a metriky Tako kategorie funkcí je určená k výpočtu metrik nad grafem. Výstupem funkce je přidání jednoho nebo více atributů k uzlům



Obrázek 2.4: Změna rozložení podle funkce `Foce Atlas 2`.



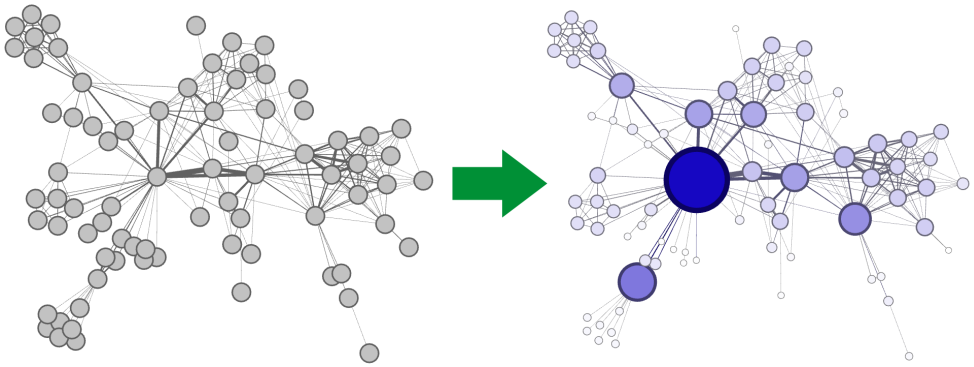
Obrázek 2.5: Report po aplikaci funkce `PageRank`PageRank

nebo hranám. Navíc je zobrazena shrnující zpráva (obr. 2.5), která typicky prezentuje rozložení hodnot atributů v grafu. Jako příklad statistické funkce lze uvést např. `Hodnost stránky` (PageRank[7]).

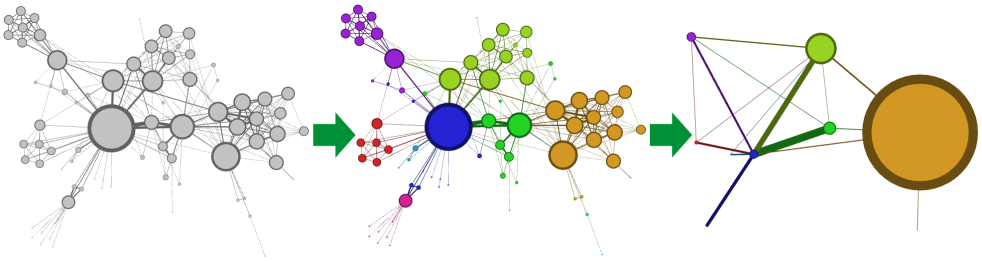
Hodnocení Hodnocení slouží k lepší vizuální reprezentaci hodnot atributů (vypočtených např. pomocí již zmíněných statistických funkcí). Podle hodnoty atributu lze měnit barvu nebo velikost uzlů a hran. Aplikaci „hodnocení“ můžeme vidět na obr. 2.6.

Rozdělení na oddíly Rozdělení na oddíly má podobnou funkci jako „hodnocení“. V tomto případě se obarvují uzly se stejnou hodnotou atributu na stejnou barvu (lze použít jen na celočíselné atributy). Tyto uzly lze také seskupit do

[7] Algoritmus pro ohodnocení důležitosti webových stránek na základě struktury hypertextových odkazů.



Obrázek 2.6: Hodnocení barvou a velikostí podle funkce `PageRank`.



Obrázek 2.7: Rozdělení na oddíly podle hodnoty atributu `Modularity Class`.

jednoho uzlu, jehož velikost je závislá na počtu uzlů, které obsahuje. Ukázku můžete vidět na obr. 2.7.

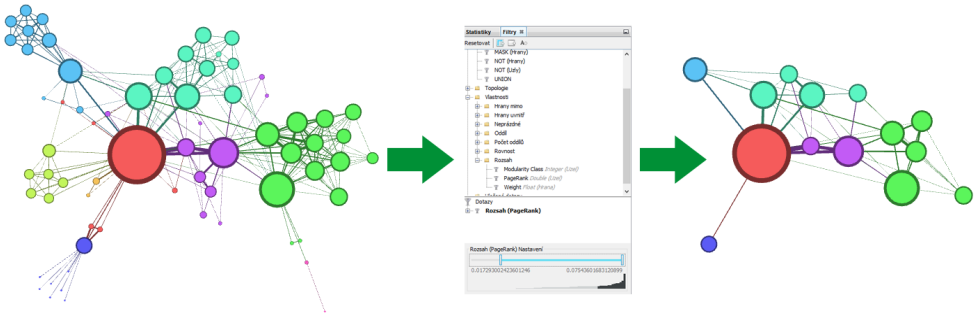
Filtry Ve velkých grafech je pro uživatele těžké se orientovat, je zahlcen velkým množstvím uzlů a hran. Pro tento účel obsahuje *Gephi* filtry. Na základě nejrůznějších podmínek, většinou závislých na hodnotě atributů, lze uzly filtrovat (odstranit) z grafu. Jednoduché filtry lze navíc řetězit pomocí logických operátorů do složitějších výrazů.

Příklad aplikace jednoduchého filtru můžete vidět na obr. 2.8.

Generování grafu Pokud nemáme žádný vhodný graf (např. pro testování vyvíjeného pluginu), *Gephi* poskytuje možnost vygenerování grafu. V základu je možnost generování jen náhodného grafu, kde můžeme nastavit pouze požadovaný počet hran a uzlů. Existují však pluginy, které možnosti *Gephi* v tomto směru dále rozšiřují.

2.1.2.3 Dynamické grafy

Gephi podporuje také dynamické grafy. Jedná se o běžný graf, kde každý uzel a hrana má nastavený interval platnosti. Většinou se jedná o časový rozsah.



Obrázek 2.8: Aplikace filtru `Rozsah` na základě hodnoty funkce `PageRank`.

Tímto způsobem lze zachytit vývoj skutečnosti, kterou graf představuje, v čase.

Gephi, kromě běžných operací, umí pomocí časové osy takovýto graf přehrát. Jedná se vlastně o specifický filtr na základě hodnoty intervalu.

2.1.2.4 Manuální editace

Gephi poskytuje také možnosti manuální editace, a to jak grafického znázornění grafu, tak datového modelu. V okně „Graf“ je v levém svislém panelu množství funkcí, které umožňují ruční, interaktivní editaci grafu - změnu velikosti uzlu, barvy, pozice, popisku, ... V „Laboratoři dat“ je zase kromě importu a exportu také možné ručně přidávat a odebírat uzly a hrany, nastavovat ručně hodnoty atributů a popisků.

2.1.2.5 Instalace pluginů

Jak už bylo několikrát zmíněno, *Gephi* poskytuje možnost snadného rozšíření funkcionality pomocí pluginů. Využívá přitom možností platformy *NetBeans*, která je od základu přizpůsobena modulární architektuře. *Gephi* plugin je *NetBeans* plugin[8] využívající API *Gephi*. *Gephi* při startu automaticky načte všechny pluginy a tím rozšíří svou funkcionalitu. Využívá k tomu *NetBeans* specifické řešení, tzv. *Lookup API*[9]. Přesný způsob implementace není pro tuto práci podstatný, stačí vědět, že pouhou implementací nějakého rozhraní lze rozšířit funkcionalitu *Gephi*.

[8] Jedná se o zip archive obsahující deskriptor modulu a jednu nebo více Java knihoven, které implementují vlastní funkcionalitu.

[9] API pro volnou vazbu mezi moduly (pluginy). Modul deklaruje veřejné rozhraní SPI (*Service Provider Interface*). Jiné pluginy pak mohou toto rozhraní implementovat a deklarovat jako *Service Provider* a tím rozšířit funkcionalitu. Implementace *Lookup API* poté dokáže tyto implementace nalézt i během runtime a použít.

2.2 Komponenty relevantní pro WebGephi

Gephi poskytuje opravdu rozsáhlé možnosti práce s grafy, od manuální editace až po aplikaci komplexních funkcí a práci s dynamickými grafy. Ne všechna funkcionalita je však vhodná pro *WebGephi*. Zde si určíme, jakou funkcionalitu je vhodné (a možné) implementovat ve *WebGephi*.

2.2.1 Cílový uživatel

Nejdříve si musíme shrnout, jak vlastně bude vypadat cílový uživatel, k jakému účelu bude *WebGephi* primárně používáno.

2.2.1.1 Student, vývojář aplikací

Jedním z typických uživatelů bude pravděpodobně student vysoké školy, který v rámci nějakého projektu (např. semestrální práce) bude mít za úkol analýzu grafu. Většinou to nebude jediný smysl aplikace, pouze jedna z jejích částí. Místo vlastní implementace pomocí *Gephi Toolkit* využije API *WebGephi*. Nebude se tedy muset zabývat strukturou knihovny ani způsobem implementace. Navíc není omezen cílovou platformou (např. není nucen psát aplikaci v jazyku Java).

Nemusí se samozřejmě jednat pouze o studenty. Stejně tak se může jednat o jakéhokoli vývojáře, který chce v rámci své aplikace využít možností *Gephi*. *WebGephi* ho odstíní od detailů implementace *Gephi Toolkit* a poskytne platformovou nezávislost.

2.2.1.2 Tvůrce pluginů

Dalším typickým uživatelem bude tvůrce algoritmů pro práci s grafy. *Gephi* poskytuje možnost snadného rozšíření pomocí pluginů. Pokud někdo takovýto plugin vytvoří, pomocí *WebGephi* (a její klientské aplikace) může vytvořit demo pro prezentaci jeho funkcionality. Případní uživatelé pluginu tak nebudou nuceni instalovat plugin jen pro jeho vyzkoušení.

2.2.2 Požadovaná funkcionalita WebGephi

Je tedy vidět, že pro *WebGephi* není nutné ani vhodné pokrýt celou funkcionalitu *Gephi*. *WebGephi* bude zaměřeno na práci s již existujícími grafy, není nutné řešit manuální editaci grafu. Stejně tak není nutné podporovat všechny formáty pro import a export. Naopak je nutné zachovat většinu grafových funkcí. Klíčovou vlastností *Gephi* je jeho rozšiřitelnost, je nezbytné aby *WebGephi* tuto rozšiřitelnost zachovalo.

2.2.2.1 Import a export

WebGephi bude podporovat import grafu ve formátu `gexf`. Tento formát je ve svém oboru standard a z dostupných formátů má největší možnosti (viz srovnání A.1). Jedná se relativně jednoduchý formát založený na `xml`, je tedy snadno strojově zpracovatelný a díky definici ve formátu `xsd` je jednoznačný. Ve stejném formátu bude dostupný také export grafu.

Vizualizace grafu bude zodpovědností především klientských aplikací. Přesto je vhodné, aby i *WebGephi* obsahovalo základní export v grafickém formátu. Nejvhodnější z podporovaných formátů[10] je `svg`. Jedná se o vektorový formát založený na `xml`, který je pro tento účel vhodnější než rastrový formát `png`. A oproti `pdf` je možná jeho další editace. Formát `svg` je podporován většinou grafických editorů a všechny hlavní webové prohlížeče obsahují nativní podporu pro jeho vizualizaci.

2.2.2.2 Grafové funkce

Grafové funkce budou nejdůležitější částí aplikace. WebGephi se nebude zabývat tvorbou grafu ve většině případů ani jeho vizualizací. Typické workflow bude *načtení grafu - aplikace funkcí - export ve formátu gexf.*

Nejdůležitější bude podpora pro funkce „Rozložení“ a „Statistiky“, které jsou nejpoužívanější. Pro lepší vizualizaci výsledků je také potřeba podpora pro funkce „Hodnocení“. Podpora pro „Filtry“, „Generování grafů“ a „Rozdělení na oddíly“ nemusí být součástí této práce, jedná se o méně využívané funkce. Struktura *WebGephi* však musí umožnit snadné doplnění této funkcionality v budoucnu.

2.3 Gephi Toolkit

Gephi je aplikace určená pro koncového uživatele. Kromě modulů obsahujících logiku pro práci s grafy obsahuje také moduly související s uživatelským rozhraním, nápovědou... Není tedy vhodné pro další použití v jiných aplikacích. Naštěstí *Gephi* pro tyto účely poskytuje knihovnu *Gephi Toolkit*. Tato knihovna obsahuje jádro aplikace ve formátu jedné Java knihovny.

„Gephi Toolkit obsahuje základní moduly (Graf, Rozložení, Filtry, IO...) ve formě standardní Java knihovny, kterou může jakýkoli projekt použít pro své potřeby. Toolkit je jeden JAR, který může kdokoli znovupoužít v jiné Java aplikaci a například z příkazové řádky dosáhnout automatizovaně toho samého jako v Gephi. Možnost využít takto schopností Gephi v jiných Java aplikací rozšiřuje jeho možnosti a zdá se být velmi užitečné.“[11]

[10] `png`, `pdf` a `svg`

[11] Volný překlad z oficiálních stránek *Gephi Toolkit*[4]. Orig.: „The Gephi Toolkit project package essential modules (Graph, Layout, Filters, IO...) in a standard Java library, which any Java project can use for getting things done. The toolkit is just a single JAR that

Gephi Toolkit je součástí *Gephi* a je distribuováno pod stejnou licencí - duální licencí CDDL 1.0 a GNU General Public License v3. Může tedy být použit bezplatně i v rámci komerčního software.

2.3.1 Struktura

Gephi je založeno na platformě *NetBeans* a tudíž využívá jeho techniky pro práci s modulárním software. Ta je založena na `Lookup API`. Veřejné API poskytuje přístup pouze k rozhraním a jeho implementace může být získána pomocí třídy `Lookup`.

```
1 ProjectController pc = Lookup.getDefault().lookup(
    ProjectController.class);
```

Ukázka zdroj. kódu 2.1: Ukázka použití Lookup API

Práce s Gephi je prováděna s pomocí kontrolerů. Pro každou oblast funkcionality (modul) existuje kontroler, což je singleton poskytující příslušné metody. Základní kontrolery jsou:

ProjectController
: Vytváření a správa projektů a workspace.

ImportController
: Import grafu ze souboru, databáze, ...

ExportController
: Export grafu do obrázku, v `gexf` formátu, `pdf`

GraphController
: Manipulace s grafem, vytváření uzlů a hran.

AttributeController
: Přidávání, mazání a editace atributů u uzlů a hran.

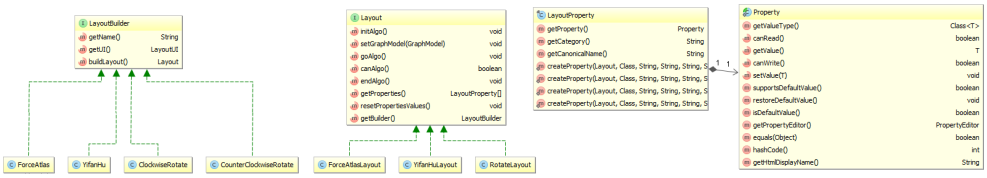
LayoutController, StatisticsController, RankingController, ...
: Práce s příslušnými grafovými funkcemi.

Klíčovým požadavkem pro implementaci *WebGephi* bude možnost vylistování všech dostupných funkcí. To bohužel není možné udělat standardní cestou, jak je to řešeno v *NetBeans* platformě[12]. *Gephi Toolkit* je už jen standardní knihovna bez deskriptorů, které jsou nezbytné pro *NetBeans* moduly.

Nejdříve analyzujeme strukturu potřebných grafových funkcí a poté popíšeme, zda a jak bude možné řešit nutnost výčtu dostupných funkcí.

anyone could reuse in new Java applications and achieve tasks that can be done in Gephi automatically, from a command-line program for instance. The ability to use Gephi features like this in other Java applications boost possibilities and promise to be very useful.“

[12]Pomocí `Service Provider Interface` a `Service Provider`, viz 2.1.2.5



Obrázek 2.9: Struktura tříd pro výpočet funkcí „Rozložení“.

2.3.1.1 Rozložení

Základní třídou pro výpočet „Rozložení“ je třída `LayoutBuilder` (strukturu tříd je znázorněna na obr. 2.9). Ta poskytuje přístup ke jménu funkce, grafické komponentě pro zadání parametrů a třídě `Layout`, která již provádí samotný výpočet nad grafem. Parametry rozložení jsou přístupné ve formě pole objektů `LayoutProperty`, což umožňuje jejich snadné vylistování a nastavení. Objekt `LayoutProperty` obaluje instanční proměnné a přistupuje k nim promocí reflexe[13].

```
YifanHu yifanHu = new YifanHu(); // Factory
YifanHuLayout layout = yifanHu.buildLayout(); // Layout function
layout.setGraphModel(graphModel); // Set the graph to work with
layout.initAlgo();
layout.resetPropertiesValues(); // Set default values
layout.getProperties()[0].getProperty().setValue(200f); // Change
    first param (OptimalDistance)
for (int i = 0; i < 100 && layout.canAlgo(); i++) {
    layout.goAlgo(); // Apply layout function
}
layout.endAlgo();
```

Ukázka zdroj. kódu 2.2: Aplikace funkce rozložení `YifanHu`

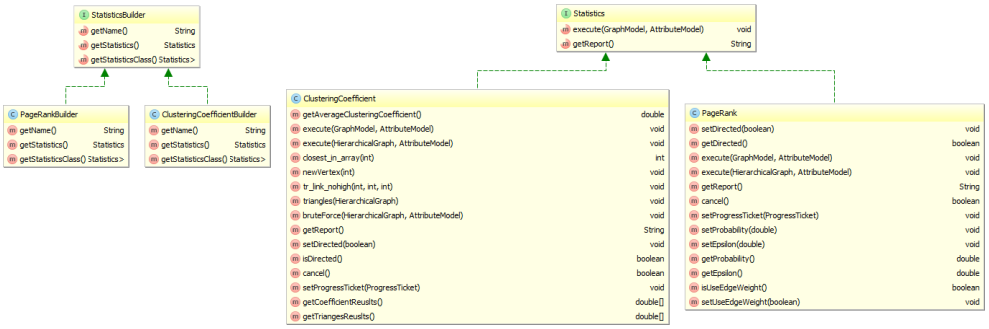
2.3.1.2 Statistiku a metriky

Struktura tříd (viz obr. 2.10) je obdobná jako u „Rozložení“. V tomto případě ale třída `Statistics` neobsahuje žádný seznam možných parametrů. Ty jsou nastavovány přímo z uživatelského rozhraní (implementace rozhraní `StatisticsUI`) pomocí getterů a settrů. Seznam parametrů bude tedy nutné získat na základě metod třídy za běhu programu pomocí reflexe.

Výstupem „Statistik“ je kromě pozměněného grafu také html zpráva obsahující souhrnné informace o rozložení metrik. Ta je dostupná po aplikaci metriky pomocí metody `Statistics::getReport()`.

```
// Create statistics builder
StatisticsBuilder builder = new GraphDistanceBuilder();
```

[13] Možnost za běhu programu získat od objektu kompletní informace o jeho typu a dále s nimi pracovat.



Obrázek 2.10: Struktura tříd pro výpočet „Statistik a metrik“.

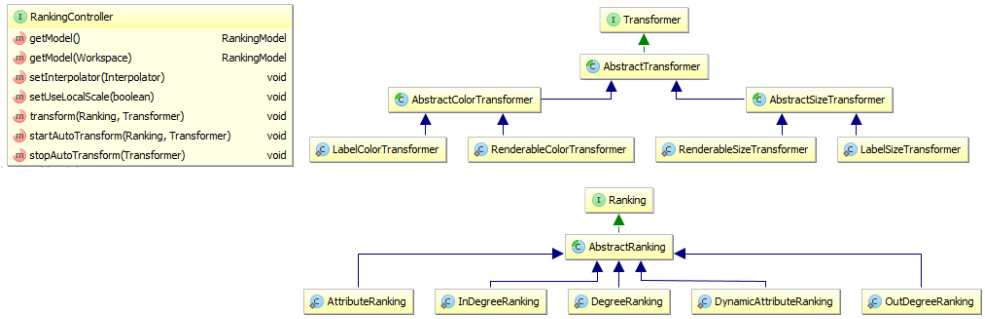
```
// Get statistics function implementation
GraphDistance gd = (GraphDistance) builder.getStatistics();
gd.setDirected(true); // Set parameter directly using setter
gd.execute(graphModel, attributeModel); // apply function
String report = gd.getReport(); // Get html report

// We can iterate over metric values of nodes too
AttributeColumn col = attributeModel.getNodeTable().getColumn(
    GraphDistance.BETWEENNESS);
for (Node n : graphModel.getGraph().getNodes()) {
    Double centrality = (Double) n.getNodeData().getAttributes().
        getValue(col.getIndex());
}
```

Ukázka zdroj. kódu 2.3: Aplikace statistické funkce `GraphDistance`

2.3.1.3 Hodnocení

K aplikaci hodnocení slouží třída `RankingController` a její metoda `transform(Ranking, Transformer)`. Stačí jen specifikovat podle čeho hodnotit a co bude výstupem. K tomu slouží implementace rozhraní `Ranking` (na základě čeho hodnotit) a `Transformer`. Jejich implementace můžete vidět na obr. 2.11. Hodnotit lze na základě jakéhokoli atributu uzlu nebo hrany (případně podle stupně uzlu). Výstupem hodnocení může být změna barvy (uzlu, hrany, popisku) nebo velikosti (uzlu, hrany).

```
// Column with centrality attribute, which we wat to use for
    ranking
AttributeColumn centralityColumn = attributeModel.getNodeTable().
    getColumn(GraphDistance.BETWEENNESS);
// Create ranking based on centrality attribute
Ranking centralityRanking = rankingController.getModel().
    getRanking(Ranking.NODE_ELEMENT, centralityColumn.getId());
// Create transformer - defines what we want to change (node size)
AbstractSizeTransformer sizeTransformer = (AbstractSizeTransformer
    ) rankingModel.getTransformer(Ranking.NODE_ELEMENT,
    Transformer.RENDERABLE_SIZE);
```



Obrázek 2.11: Struktura tříd pro výpočet „Hodnocení“.

```
// Set transformer parameters
sizeTransformer.setMinSize(3);
sizeTransformer.setMaxSize(20);
// Apply ranking, node size is updated
rankingController.transform(centralityRanking, sizeTransformer);
```

Ukázka zdroj. kódu 2.4: Změna velikosti uzlu podle hodnoty atributu „betweenesscentrality“

2.3.2 Seznam dostupných funkcí

Jak už jsme zmínili, klíčovým požadavkem pro *WebGephi* je schopnost najít všechny dostupné grafové funkce, tzn. všechny implementace rozhraní `LayoutBuilder` a `StatisticsBuilder`. Tento list nemůže být statický. Musíme totiž zároveň zachovat rozšiřitelnost, pokud k aplikaci přidáme plugin obsahující např. funkci pro výpočet „Rozložení“, aplikace ho musí sama přidat do svého (REST) rozhraní.

Jelikož nemůžeme využít možností *NetBeans* platformy, musíme využít možností samotného jazyku Java. Pomocí reflexe nalezneme všechny implementace daných rozhraní (`LayoutBuilder` a `StatisticsBuilder`) a s jejich pomocí už máme dostupnou potřebnou funkcionalitu. Takto nalezneme všechny implementace dostupné na classpath aplikace, je tím tedy zajištěna požadovaná rozšiřitelnost. Z bezpečnostních důvodů nebude možné přidávat pluginy za běhu aplikace, proces nalezení všech dostupných funkcí tedy bude stačit provést pouze jednou po startu aplikace.

V případě „Hodnocení“ se nepočítá s dalším rozšiřováním, tudíž není třeba zjišťovat dostupné možnosti dynamicky. Bude stačit implementovat několik základních možností.

2.3.3 Víceuživatelský přístup

Architektura *Gephi* je od počátku navrhována pro potřeby desktopové aplikace. Je z velké části založena na návrhovém vzoru Singleton[14]. Např. všechny kontrolery jsou singletony. To není v zásadě problém, protože kontrolery nejsou stavové. Problémy nastávají v momentě, kdy chceme pracovat s více Workspace najednou (ve vícevláknovém prostředí). Tento problém je ostatně zmíněn (ne příliš výrazně) i na stránkách s příklady použití *Gephi Toolkit*.

„(. . .) many modules just do their job on this current workspace and doesn't allow to specify which workspace to work on from the API. Therefore it is often required to set current workspace, as showed in export here.“[4]

Architektura *Gephi* je totiž navržena tak, že vždy právě jeden Projekt a Workspace je aktivní. A přestože některé moduly (třídy, metody) se tváří tak, že umí pracovat s Workspace předaným v parametru, v těle metody se používá „current Workspace“. Tento problém ilustruji na příkladu 2.5 třídy `DefaultProcessor`, která se využívá při importu grafu.

„currentWorkspace“

```
// Set current workspace to 'currentWorkspace'
projectController.openWorkspace(currentWorkspace);
DefaultProcessor defaultProcessor = new DefaultProcessor();
// Set processor to use different workspace than current
defaultProcessor.setWorkspace(notCurrentWorkspace);
// Import graph to Workspace 'currentWorkspace'
importController.process(container, new DefaultProcessor(),
    currentWorkspace);
```

Ukázka zdroj. kódu 2.5: Import grafu do Workspace

Vše vypadá na první pohled v pořádku, graf se zdánlivě načte do Workspace `notCurrentWorkspace`. Pokud se však podíváme do implementace třídy `DefaultProcessor` (viz ukázka kódu 2.6) zjistíme, že se ze skutečnosti používá Workspace nastavený v Singletonu `ProjectController` jako „currentWorkspace“.

```
@ServiceProvider(service = Processor.class, position = 10)
class DefaultProcessor extends AbstractProcessor implements
    Processor {
  // ...
  // Method called during import
  public void process() {
    // ...
    // Get GraphModel from current workspace!!!
    GraphModel graphModel = Lookup.getDefault().lookup(
        GraphController.class).getModel();
    // It should be:
```

[14] *„Singleton (česky jedináček nebo také unikát) je název pro návrhový vzor, používaný při programování. Využijeme ho při řešení problému, kdy je potřeba, aby v celém programu běžela pouze jedna instance třídy. Tento návrhový vzor zabezpečí, že třída bude mít jedinou instanci a poskytne k ní globální přístupový bod.“*[7]

```
        // Lookup.getDefault().lookup(GraphController.class).getModel(
            workspace)
11      // ...
    }
13  // ...
}
```

Ukázka zdroj. kódu 2.6: Výřez z implementace třídy `DefaultProcessor`

Tento problém lze vyřešit (jak ostatně zní doporučení v dokumentaci *Gephi Toolkit*) tím, že vždy nastavíme jeden workspace jako „current“ a pracujeme pouze s ním. To může fungovat u desktopové aplikace, v serverovém prostředí se však pohybujeme ve vícevláknovém prostředí. Museli bychom prakticky veškerou práci s *Gephi Toolkit* provádět serializovaně, (v `synchronized` bloku nebo nějakou pokročilejší technikou zajišťující, že k danému úseku kódu přistupuje vždy jen jedno vlákno). To by samozřejmě způsobovalo úzké hrdlo aplikace a výrazně degradovalo výkon. Prakticky by mohl být v jednu chvíli zpracováván vždy jen jeden požadavek. V kombinaci s tím, že operace nad grafy mohou trvat relativně dlouho, by se aplikace stala naprosto nepoužitelnou už pro relativně malý počet uživatelů.

Druhé možné řešení je odstranění těchto chyb tím, že upravíme původní třídy tak, abychom místo s „currentWorkspace“ pracovali s konkrétním Workspace předaným komponentě. Jedná se o relativně malé množství komponent, tudíž bude toto řešení vhodnější. Tyto třídy budou součástí *WebGephi*, nebude se jednat o úpravy samotné knihovny.

2.4 Možnosti autentizace a autorizace

WebGephi bude multiuživatelská aplikace. Kdokoli se bude moci založit svůj účet, nahrávat a upravovat své grafy. Z toho přirozeně plyne nutnost autentizace (ověření identity uživatele) a autorizace (určení, zda má uživatel práva pro danou operaci).

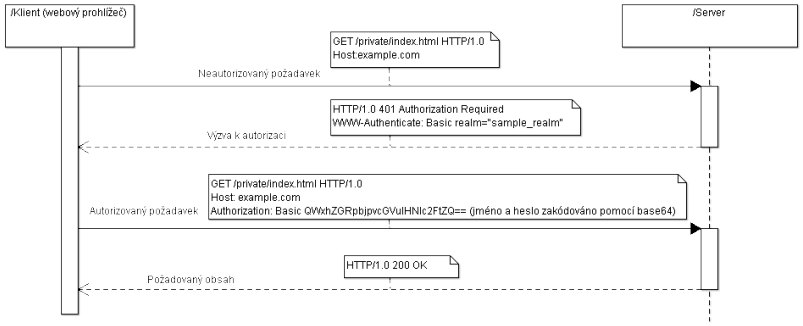
WebGephi však bude poskytovat pouze REST rozhraní, které málokdo bude využívat přímo. Místo toho autorizuje nějakou klientskou aplikaci, aby mohla přistupovat k jeho účtu. Tato aplikace pak skrze vhodné grafické rozhraní umožní uživateli pracovat s jeho grafy.

Zde si představíme standardizovaná řešení této problematiky.

2.4.1 Basic autentizace[8]

Basic autentizace je jednou z nejstarších metod autentizace. Princip, jak už název napovídá, je velmi jednoduchý. Klient (většinou webový prohlížeč) zašle v hlavičce HTTP požadavku své jméno a heslo[15]. Server ověří správnost údajů a na jejich základě poskytne nebo odmítne poskytnout požadovaný obsah.

[15] ve formě `uživatelskéJméno:heslo`, zakódováno pomocí base64



Obrázek 2.12: Průběh Basic autorizace.

Pokud požadované přihlašovací údaje chybí, vrátí server odpověď s požadavkem na Basic autentizaci.

Výhodou této metody je její jednoduchost. Snad každý HTTP klient obsahuje podporu pro Basic autentizaci. Všechny rozšířené webové prohlížeče si navíc pamatují zadané přihlašovací údaje a není tedy nutné je zadávat vždy znovu.

Nevýhodou je, že je nutné zasílat přímo přihlašovací údaje, navíc v nezašifrované podobě a tudíž kdokoli může údaje po cestě odposlechnout. Při použití HTTPS protokolu se však jedná o bezpečnou metodu autentizace.

2.4.1.1 Použití pro WebGephi

Jedná se o vhodnou metodu pro přímé procházení a prozkoumávání REST rozhraní. Není však vhodné pro poskytování přístupu aplikacím třetích stran (klientské aplikace). V tomto případě by bylo nutné předat klientské aplikaci přímo své uživatelské jméno a heslo. Tím by uživatel poskytl plný přístup ke svému účtu a nevěrohodná klientská aplikace by toho mohla zneužít (v krajním případě změnit heslo a tím „ukrást“ uživatelův účet). Uživatel by navíc nemohl odebrat aplikaci již jednou udělené oprávnění. Pro tento účel by jedině musel změnit své heslo.

Tento způsob autentizace je tedy vhodný jedině pro přímé prozkoumávání REST rozhraní aplikace. Mohl by být použit také v případě, kdy klientská aplikace využívá jen svůj vlastní účet (může případně řešit správu uživatelů na své straně).

2.4.2 Digest autentizace[9]

Jedná se o mírnou evoluci Basic autentizace. Princip je úplně stejný, v tomto případě se však neposílá heslo přímo, ale jeho hash[16]. Z hlediska bezpečnosti se jedná o mírné zlepšení. Při odposlechnutí požadavku nelze (pokud je autentizace správně implementována) požadavek zopakovat. Nevýhodou je složitější implementace jak na straně serveru, tak na straně klienta.

2.4.2.1 Použití pro WebGephi

Tato metoda, za předpokladu že použijeme HTTPS, nepřináší prakticky žádné výhody oproti Basic autorizaci. Naopak je složitější na implementaci.

2.4.3 OAuth v1.0a

„Protokol OAuth umožňuje webovým stránkám nebo aplikacím (Konzumentům) přistupovat k chráněným zdrojům webových služeb (Poskytovatele služeb) skrz API bez nutnosti, aby uživatel prozradil své přihlašovací údaje k Poskytovateli služeb Konzumentům. Obecněji, OAuth vytváří volně implementovatelnou a obecnou metodiku pro autentizaci k API“ [17]

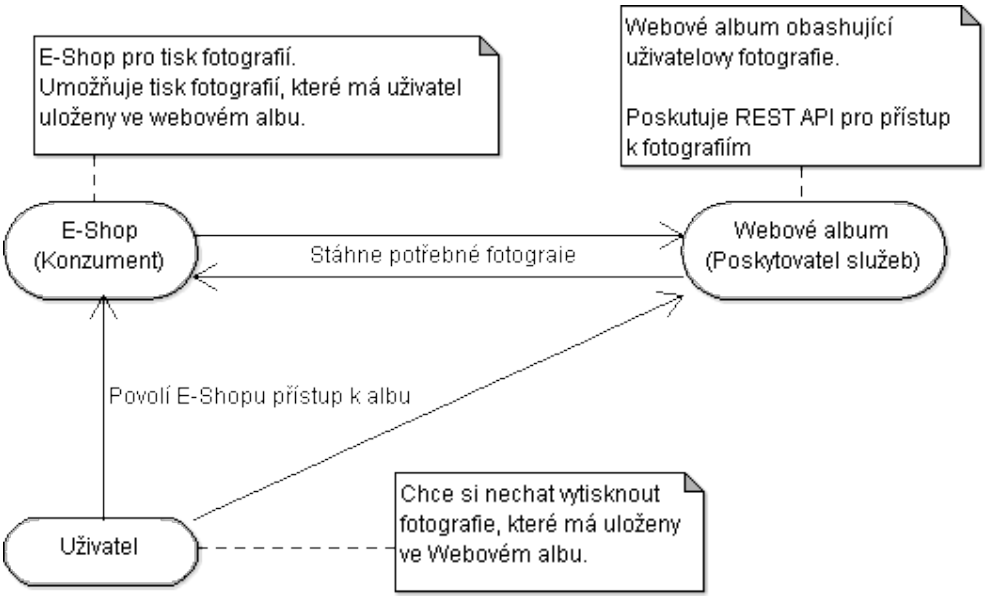
OAuth je standardizovaný způsob, jak uživatel může poskytnout přístup ke svému účtu (chráněným zdrojům) u nějaké aplikace (*Poskytovatel služeb*) jiné aplikaci (*Konzument*). A to bez toho, aby této aplikaci (*Konzumentovi*) musel prozradit své přihlašovací údaje. Ukázkový příklad pro využití OAuth protokolu je znázorněn na obr. 2.13.

Konzument musí být zaregistrován u *Poskytovatele služeb* a má svůj jednoznačný identifikátor (`consumer key`) a heslo (`consumer secret`). Pokud požaduje přístup k uživatelovým datům u *Poskytovatele služeb*, zažádá nejdříve o `Request token`. Ten specifikuje, k jakým zdrojům požaduje aplikace přístup. Poté přesměruje uživatele na stránky *Poskytovatele služeb*. Uživatel se zde přihlásí a autorizuje požadavek *Konzumenta*, následně je přesměrován zpět na stránky *Konzumenta*. Odpověď obsahuje `verifier`, který *Konzument* použije k výměně `Request tokenu` za `Access token`. `Access token` už může být použit přímo k přístupu k požadovaným zdrojům. Podrobný popis průběhu OAuth autorizace je znázorněn na obr. 2.14.

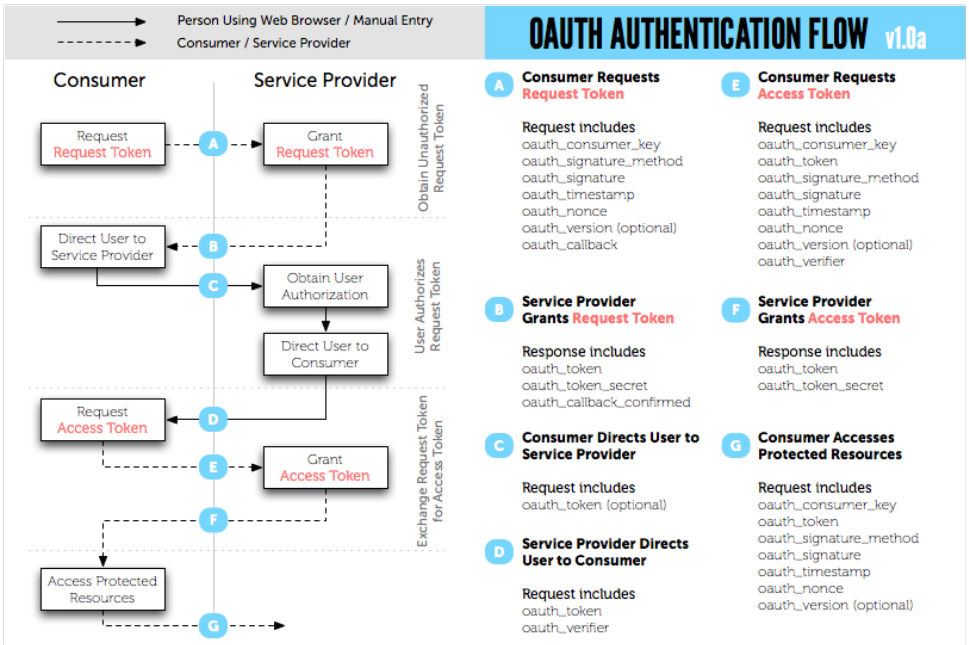
Autentizace a autorizace pomocí OAuth v1.0a protokolu je velmi bezpečná. Veškerá komunikace s API *Poskytovatele služeb* je podepisována pomocí informací z `Access tokenu`. Navíc všechny citlivé informace jsou při přenosu

[16] Používá se `MD5` hash kombinace hesla, cílové uri, metody (GET, POST, ...), nonce (serverem poskytnutý náhodný řetězec, cnonce (klientův náhodný řetězec) a pořadí požadavku. Více viz [9])

[17] Volný překlad. Orig.: *„The OAuth protocol enables websites or applications (Consumers) to access Protected Resources from a web service (Service Provider) via an API, without requiring Users to disclose their Service Provider credentials to the Consumers. More generally, OAuth creates a freely-implementable and generic methodology for API authentication.“*[10]



Obrázek 2.13: Ukázkový příklad využití OAuth protokolu.



Obrázek 2.14: Průběh OAuth autorizace. Převzato z [10]

šifrovány. To umožňuje jeho bezpečné použití i bez použití SSL. Na druhou stranu je kvůli tomu OAuth náročný na implementaci jak na straně serveru, tak na straně klienta. Při použití HTTPS je navíc šifrování v podstatě duplikátní. Na druhou stranu díky přesné specifikaci existují hotové knihovny pro jeho použití.

Hlavní výhodou OAuth protokolu je, že nemusíme aplikacím třetích stran vyzradit své přihlašovací údaje jen kvůli tomu, abychom jim poskytli přístup k části svých chráněných zdrojů. Toto povolení můžeme navíc kdykoliv odebrat. Protokol také umožňuje poskytnout přístup pouze k části zdrojů (tzv. `scopes`).

2.4.3.1 Použití pro WebGephi

OAuth je ideální protokol pro *WebGephi*. Jeho součástí je částečně i způsob registrace klientských aplikací. Umožňuje uživatelům udělovat oprávnění klientským aplikacím (*Konzumentům*), aby mohli přistupovat k jejich účtům na *WebGephi* (*Poskytovatel služeb*). A to bez toho, aby museli prozradit své přihlašovací údaje. Navíc nemusí udělovat plný přístup ke svému účtu, některé aplikace mohou např. požadovat pouze práva pro čtení. Typicky žádná aplikace nebude mít práva pro úpravu uživatelova profilu (jména, hesla).

Jedinou nevýhodou je složitější implementace protokolu na straně serveru i klienta.

2.4.4 OAuth v2.0[11]

Jedná se o novou verzi, která vyšla v říjnu 2012 (RFC 6749). Již dlouho předtím však byla ve stadiu návrhu. Její cílem bylo zjednodušení protokolu a větší použitelnost pro jiné než webové aplikace (desktopové, mobilní, ...). Hlavní změnou je, že komunikace již není šifrována, plně se spoléhá na SSL šifrování (nutnost použití společně s HTTPS). Zavádí také více různých způsobů autorizace (kromě klasického „three-legged“ OAuth), které jsou vhodné např. pro mobilní aplikace.

Jedná se o volnější specifikaci než je OAuth 1.0a. Jednotlivé implementace se výrazně liší a většinou nejsou spolu kompatibilní.

2.4.4.1 Použití pro WebGephi

OAuth v2.0 nepřináší oproti první verzi z pohledu *WebGephi* žádné výrazné výhody. Volnější specifikace a nekompatibilita jednotlivých implementací ztěžuje jeho použití různými *Konzumenty*. Výhodou by byla lepší podpora desktopových a mobilních klientů.

Kapitola 3

Návrh

3.1 REST rozhraní

3.2 Autorizace

3.3 Správa uživatelů a klientských aplikací

3.4 Struktura aplikace

3.4.1 Server

3.4.2 Klient (Java konektor)

3.4.3 Klientská aplikace

3.5 GUI

Kapitola 4

Realizace

4.1 Požité technologie

4.1.1 Wildfly 8

4.1.2 RestEasy

4.1.3 Enterprise Java Beans

4.1.4 JPA

4.1.5 Errai a GWT

4.1.6 Vaadin

4.2 Server TODO popis struktury a zajímavých částí

4.3 Klient (konektor) TODO popis struktury a zajímavých částí

4.4 Klientská aplikace (konektor) TODO popis struktury a zajímavých částí

Závěr

Literatura

[1] Gephi website. 2014. Dostupné z: http://gephi.org

[2] Wikipedie: Graf (teorie grafů) — Wikipedie: Otevřená encyklopedie. 2013, [Online; navštíveno 21. 04. 2014]. Dostupné z: http://cs.wikipedia.org/w/index.php?title=Graf_(teorie_grafu)&oldid=10977342

[3] Wikipedie: Representational State Transfer — Wikipedie: Otevřená encyklopedie. 2013, [Online; navštíveno 22. 04. 2014]. Dostupné z: http://cs.wikipedia.org/w/index.php?title=Representational_State_Transfer&oldid=10902570

[4] Gephi toolit website. 2014. Dostupné z: https://gephi.org/toolkit/

[5] Gephi Consortium. 2014. Dostupné z: https://consortium.gephi.org/

[6] GEXF formát. 2014. Dostupné z: http://gexf.net/format/

[7] Wikipedie: Singleton — Wikipedie: Otevřená encyklopedie. 2014, [Online; navštíveno 1. 05. 2014]. Dostupné z: http://cs.wikipedia.org/w/index.php?title=Singleton&oldid=11083315

[8] Wikipedie: Basic access authentication — Wikipedie: Otevřená encyklopedie. 2013, [Online; navštíveno 3. 05. 2014]. Dostupné z: http://cs.wikipedia.org/w/index.php?title=Basic_access_authentication&oldid=9997988

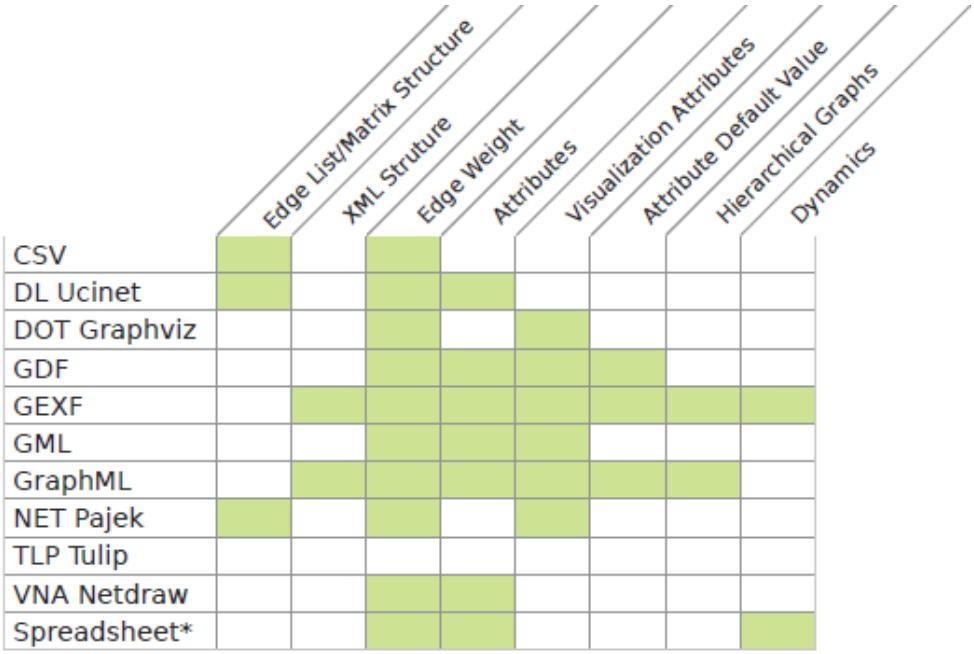
[9] Wikipedia: Digest access authentication — Wikipedia, The Free Encyclopedia. 2013, [Online; accessed 3-May-2014]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Digest_access_authentication&oldid=588566752

[10] OAuth: OAuth Core 1.0. 2014. Dostupné z: http://oauth.net/core/1.0/

[11] OAuth. 2014. Dostupné z: http://oauth.net

Příloha A

Gephi



Obrázek A.1: Porovnání formátů vhodných pro ukládání grafů. Převzato z [1]

.

Příloha B

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS