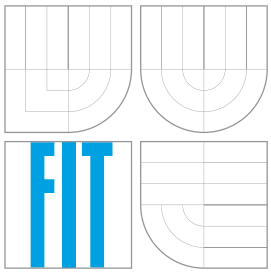
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

VIZUALIZACE SOCIÁLNÍCH KONTAKTŮ

PROJEKT DO PŘEDMĚTU GJA

AUTOR PRÁCE KAMIL DUDKA

BRNO 2008

Vizualizace sociálních kontaktů

Zadání

S využitím Social Graph API, případně knihovny Prefuse, implementujte vizualizátor sociálních kontaktů.

Obsah

Kapitola 1

Úvod

Tento dokument popisuje návrh a implementaci aplikace pro interaktivní vizualizaci sociálních kontaktů, kterou jsem nazval *SGVis*. Zdrojové kódy aplikace lze stáhnout z webu projektu http://dudka.cz/sgvis. Na tomto webu je rovněž umístěna vygenerovaná dokumentace API. V následující kapitole jsou uvedeny důležité pojmy, se kterými tento dokument pracuje. V kapitole 3 je popsán návrh aplikace a v kapitole 4 je popsána aplikace z implementačního hlediska včetně návodu k sestavení a spuštění.

Tento dokument byl vysázen systémem LaTeX.

Kapitola 2

Teoretická část

2.1 Social Graph API

Veřejný web je tvořen propojenými stránkami, které reprezentují jak dokumenty, tak lidi[5]. Vyhledávač Google se snaží, aby tyto informace byly více přístupné a užitečné. Pokud si odmyslíte dokumenty, zůstanou vám propojení mezi lidmi. Informace o veřejných propojeních mezi lidmi mohou být užitečné – jako uživatel budete chtít vědět, kdo je s vámi propojen a jako vývojář sociálních aplikací budete chtít poskytovat lepší služby pro uživatele, aby věděli, kdo jsou jejich přátelé. Dříve neexistoval rozumný způsob, jak se k těmto informacím dostat. *Social Graph API* jednoduše zpřístupňuje vývojářům sociálních aplikací informace o veřejně deklarovaných vztazích mezi lidmi.

2.2 Formát JSON a jeho zpracování

JSON (*JavaScript Object Notation*) je odlehčený formát pro výměnu dat[1]. Tento formát může být snadno čten/zapisován lidmi, ale zároveň jej lze jednoduše parsovat/generovat stroji. Je založen na podmnožině jazyka JavaScript (*Standard ECMA-262 3rd Edition – December 1999*). JSON je textový formát, který je zcela nezávislý na konkrétním jazyku a programovacích konvencí – proto je vhodný pro výměnu dat mezi různými platformami.

2.3 Vizualizační toolkit Prefuse

Prefuse je toolkit pro vytváření vysoce interaktivních vizualizací[2]. Originální Prefuse toolkit poskytuje vizualizační framework pro jazyk Java. *Prefuse Flare* toolkit poskytuje vizualizační a animační nástroje pro ActionScript a Adobe Flash Player.

Prefuse disponuje bohatým sortimentem komponent pro modelování, vizualizaci a interakci. Poskytuje optimalizované datové struktury pro tabulky, grafy a stromy. Podporuje různá grafická rozložení, vizuálně kódovací techniky, animace, dynamické dotazy, integrované vyhledávání a propojení s databází. Prefuse je napsaný v jazyce Java, používá *Java*

2D graphics library a je jednoduše integrovatelný do *Java Swing* aplikací nebo webových appletů. Prefuse je vydávána pod BSD licencí a může být volně používána pro komerční i nekomerční účely.

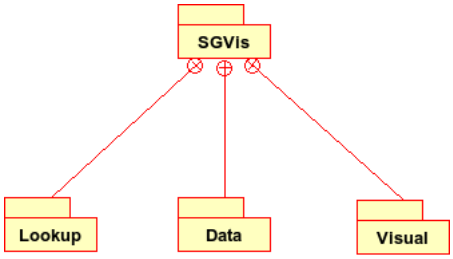
Kapitola 3

Návrh řešení

Při návrhu jsem vycházel z [9][4][8][3][7][6]. Aplikaci lze z hlediska návrhu rozdělit do několika částí:

- Komunikace se serverem Google
- Datové úložiště pro stažená data
- Interaktivní vizualizace stažených dat

Podle tohoto rozdělení jsou také děleny třídy do balíčků, jak znázorňuje obrázek 3.1. Celá



Obrázek 3.1: Hierarchie balíčků

aplikace je umístěna v balíčku `cz.vutbr.fit.dudka.SGVis`. V tomto balíčku jsou vnořeny ostatní balíčky. Kromě toho jsou na nejvyšší úrovni také třídy `Main` a `Config`. První z nich obsahuje vstupní bod aplikace – funkci `main` a druhá představuje společné úložiště pro konfiguraci aplikace. Ve zbytku této kapitoly bude postupně popsán návrh jednotlivých balíčků.

3.1 Komunikace se serverem Google

Zdrojová data načítá aplikace za běhu ze serveru Google pomocí *Social Graph API* – viz. kapitola 2.1. Rozhraní vyšší úrovně pro komunikaci se serverem představuje třída `Lookup` a její metoda `lookup`. Tato operace je blokující – to znamená, že volající vlákno je pozastaveno, dokud nejsou data stažena a zpracována. Komunikace je zahájena zasláním

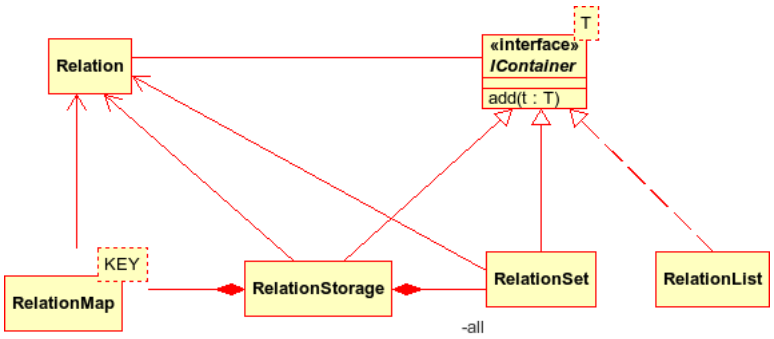
požadavku ve vhodném tvaru na server. Následně je přečtena celá odpověď serveru a teprve potom zpracována.

Odpověď serveru je ve formátu *JSON*, který byl představen v kapitole 2.2. Pro jednoduché zpracování toho formátu, používá aplikace stejnojmennou knihovnu, kterou lze stáhnout z http://www.json.org/java/ ve formě zdrojových kódů. Tato knihovna je volně šiřitelná a byla přidána přímo ke zdrojovým kódům aplikace.

3.2 Datové úložiště

Data načtená ze serveru je potřeba někam uložit. Vzhledem k interaktivní povaze aplikace byly zvoleny datové struktury s přímým přístupem jako jsou množina a mapa ze standardní knihovny. Jedinný vztah mezi dvěma umístěními je reprezentován třídou `Relation`. Jedná se o třídu hodnotového typu, která má přetížené metody `equals` a `hashCode`, aby její instance bylo možné umísťovat do zmíněných kontejnerů.

Pro tento typ jsou nadefinovány různé typy kontejnerů s přímým i sekvenčním přístupem. Komplexní úložiště pro stažená data představuje třída `RelationStorage`, jak znázorňuje náčrt na obrázku 3.2. Nad daty umístěné v tomto úložiště je potom možné se efektivně



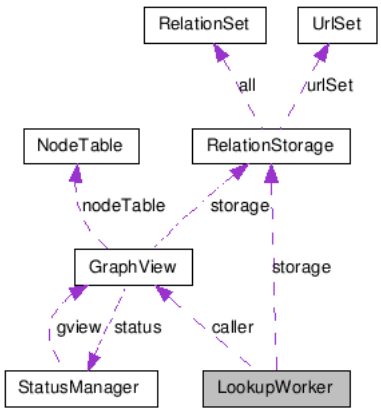
Obrázek 3.2: Datové úložiště - náčrt

dotazovat – např. je možné vyhledat všechny vztahy mezi weby z jednoho serveru nebo vyhledat všechny vztahy určitého typu apod.

3.3 Vizualizace

Interaktivní vizualizaci stažených dat zajišťuje knihovna *Prefuse*, která byla uvedena v kapitole 2.3. Na úrovni aplikace je to pak balíček `Visual`. Třída `GraphView` tvoří abstrakci nad právě zobrazovanou vizualizací a obsahuje vizualizační data – ty se obecně mohou lišit od dat globálních, která jsou uložena v objektu třídy RelationStorage. Třída `GraphDisplay` představuje komponentu grafického uživatelského rozhraní, na které je pak vizualizace zobrazována.

Protože je operace stažení dat ze serveru blokující, není možné ji volat přímo z vlákna, které obsluhuje smyčku zpráv – uživatelské rozhraní aplikace by se tak mohlo zcela zablokovat na delší dobu. Je tedy potřeba spouštět komunikaci v odděleném vlákně. Stažená data je následně potřeba zobrazit v kontextu GUI vlákna tak, aby nedošlo k problémům typu *race condition*. Stahování na pozadí včetně synchronizace zajišťuje třída `LookupWorker` – její graf spolupráce je zachycen na obrázku 3.3. Třída je odvozena ze třídy `SwingWorker`, která



Obrázek 3.3: Digram spolupráce třídy `LookupWorker`

byla uvedena v Java 1.6 – tím je bohužel vynucena minimální požadovaná verze běhového prostředí pro spuštění aplikace.

Kapitola 4

Implementace

Po spuštění aplikace se objeví její hlavní okno a v pozadí se začne provádět první komunikace se serverem. Výchozí bod vizualizace po spuštění aplikace lze nastavit v třídě `Config` stejně jako adresu serveru, který na dotazy odpovídá. V současné verzi aplikace je možné konfiguraci měnit pouze uvnitř zdrojového kódu třídy `Config`, ale tato třída je připravena pro rozšíření o (de)serializaci nastavení z/do souboru.

Za běhu aplikace je potom možné vytvořit novou vizualizaci z nového výchozího bodu pomocí položky menu *Visualization* ⇒ *New visualization*. Se zobrazenou vizualizací je možné interaktivně pracovat – přehled nejdůležitějších operací je shrnut v rychlé nápovědě, kterou lze vyvolat položkou menu *Help* ⇒ *Quick help*. V tabulce 4.1 jsou shrnuty základní operace nad vizualizací.

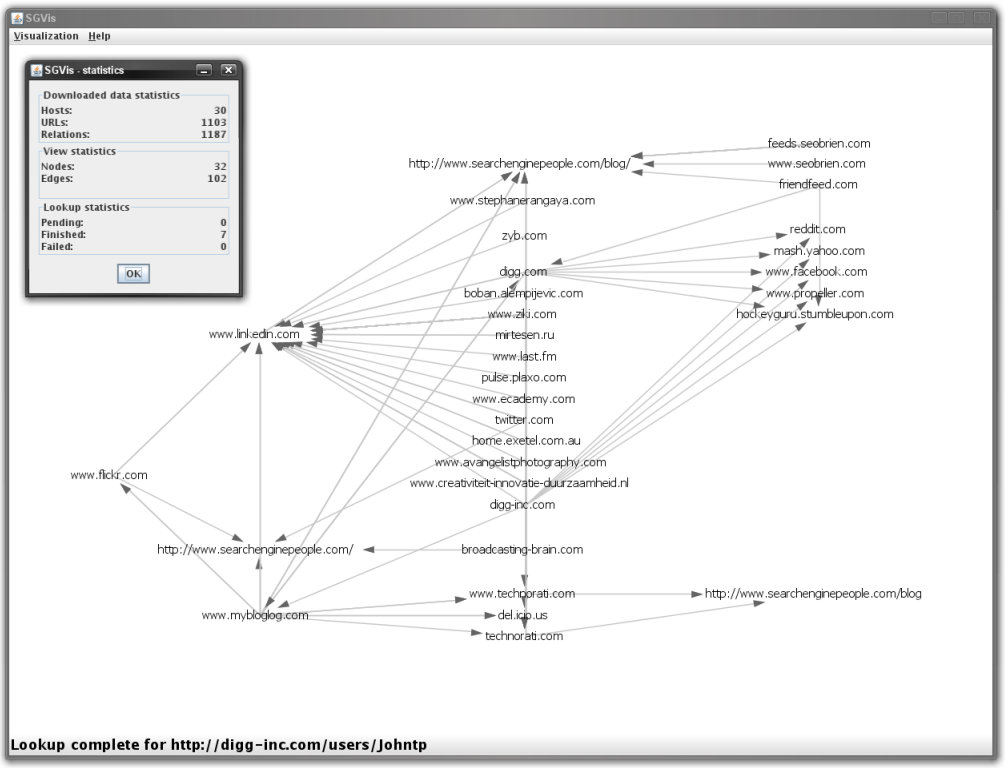
Uspořádání vizualizace	levý klik myši na uzel
Přesunutí uzlu	drag&drop levým tlačítkem myši
Operace s uzlem – kontextové menu	pravý klik myši na uzel
Přesunutí vizualizace	drag&drop levým tlačítkem myši mimo uzel
Lupa	kolečko myši
Přizpůsobení velikosti/umístění	pravý klik myši mimo uzel

Tabulka 4.1: Operace nad vizualizací

Výchozí chování je takové, že jsou různé adresy v rámci jednoho serveru reprezentovány jedním uzlem. V grafu jsou pak vidět pouze jména serverů. Kterýkoliv takovýto uzel lze *rozbalit* pomocí položky v kontextovém menu. Později lze uzly opět *sbalit* pod jeden uzel – to se hodí zejména, když je grafu hodně uzlů a stává se tak nepřehledný.

Jednotlivá umístění je možné zobrazit ve webovém prohlížeči, nebo je použít pro další vyhledání vztahů. Takhle lze inkrementálně doplňovat vizualizaci o nové vztahy, přičemž jednotlivé dotazy jsou zpracovávány na pozadí. Pokud je volba *lookup* v kontextovém menu nepřístupná, znamená to, že se zpracovává příliš mnoho dotazů současně – je tedy nutné počkat na dokončení nejméně jednoho z nich. Konkrétní hodnotu omezení současně probíhajících dotazů je možné opět nastavit ve třídě `Config`.

Pomocí položky menu *Visualization* ⇒ *Statistics* je možné zobrazit celkové statistiky, a to jak z hlediska globálních dat, tak z hlediska vizualizace a dotazování. Na obrázku 4.1 je screenshot aplikace – jejího hlavního okna a okna statistik.



Obrázek 4.1: Screenshot aplikace – originální obrázek naleznete na `http://dudka.cz/sgvis`

4.1 Sestavení ze zdrojových kódů

Aplikace byla původně vyvíjena v integrovaném vývojovém prostředí *Eclipse* (http://www.-eclipse.org/). Pro sestavení a spuštění však není potřeba mít tohle vývojové prostředí k dispozici. Součástí zdrojových kódů je soubor `build.xml`, který obsahuje potřebná metadata pro automatický build-systém *ANT* (http://ant.apache.org/). Součástí archivu jsou také volně šiřitelné knihovny, na kterých je aplikace závislá. V tabulce 4.2 je přehled základních příkazů pro práci se zdrojovými kódy pomocí ANT.

Příkaz	Význam
`ant run`	Sestaví a spustí aplikaci.
`ant jar`	Sestaví aplikaci a vytvoří `JAR` archiv.
`ant jar-all`	Sestaví aplikaci a vytvoří `JAR` archiv zahrnující kód závislých knihoven.

Tabulka 4.2: Práce se zdrojovými kódy pomocí ANT

K sestavení a spuštění aplikace je tedy nezbytné mít k dispozici ANT a vývojové/běhové prostředí Java 1.6. Pro linuxovou distribuci *Gentoo Linux* byl vyvinut *ebuild*, který automaticky stáhne zdrojové kódy z webu projektu, sestaví aplikaci a nainstaluje do systému včetně všech závislostí. Tento ebuild lze stáhnout rovněž z webu projektu http://dudka.cz/sgvis.

Kapitola 5

Závěr

Díky vizualizačnímu toolkitu Prefuse bylo možné jednoduše vytvořit vysoce interaktivní aplikaci pro vizualizaci sociálních kontaktů. Hlavní problém při používání této aplikace je v absenci veřejně deklarovaných vztahů na některých částech internetu. Další překážkou byla občasná delší odezva dotazovacího serveru Google.

Budování sociálních sítí je však teprve v počátcích a lze předpokládat, že zmíněné problémy budou v blízké budoucnosti vyřešeny. Užitečnost této aplikace by se tedy měla zvyšovat spolu s očekávaným postupným růstem sociální pavučiny. Nezanedbatelný je však můj osobní přínos – seznámení se s vizualizačním toolkitem Prefuse.

Literatura

[1] Introducing JSON. http://www.json.org/, 2008.

[2] The prefuse visualization toolkit. http://prefuse.org/, 2008.

[3] Arnold, K.; Gosling, J.; Holmes, D.: *The Java Programming Language*. Addison-Wesley, fourth edition vydání, 2005, ISBN 0321349806.

[4] Gamma, E.; Helm, R.; Johnson, R.; aj.: *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, 1997, ISBN 0-201-63361-2.

[5] Google: About the Social Graph. http://code.google.com/apis/socialgraph/docs/, 2008.

[6] Pecinovský, R.: *Návrhové vzory*. 2007, ISBN 978-80-251-1582-4.

[7] Peringer, P.: Seminář C++. https://www.fit.vutbr.cz/study/courses/ICP/public/Prednasky/ICP.pdf, 2004.

[8] Stroustrup, B.: *The C++ Programming Language*. Addison-Wesley, special edition vydání, 1997, ISBN 0-201-88954-4.

[9] Sutter, H.; Alexandrescu, A.: *C++ 101 programovacích technik*. Zoner Press, 2005, ISBN 80-86815-28-5.