Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Generování a vizualizace časové osy

Plzeň, 2014 David Merunko

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni den 8.6.2014

David Merunko

Abstract

Tool for creating and visualizing of timeline

The aim of this Master thesis was to create an application capable of storing and displaying of timelines for purpose of history learning. The main technologies used for implementation were Java and libraries Neo4j and GraphStrean, Application stores individual events, persons, places and other historical elements and also stores context connecting these elements. It is possible to create, delete and modify these elements and contexts. Output of this application is a graph representing timeline and it is used as main tool in working with timeline. The whole application is controlled by GUI, which allows direct changes to individual graph elements. Application is able to export and import data for creation and work with timeline and is able to show several degrees of zoom. Application was tested with large amount of data gather from historical databases.

Abstrakt

Tato diplomová práce měla za cíl vytvořit aplikaci pro udržování a zobrazování časových os pro potřeby výuky historie. Hlavními nástroji pro tvorbu byla java a knihovny Neo4j a GraphStream. Aplikace udržuje jednotlivé události, osoby, místa a jiné historické elementy a také navíc udržuje souvislosti, které tyto historické elementy propojují. V aplikaci je možné přidávat, ubírat a modifikovat elementy i souvislosti. Výstupem aplikace je graf který je hlavním nástrojem pro práci s časovou osou. Tato práce je možná skrze uživatelské rozhraní, na kterém je možné přímo pracovat s jednotlivými elementy grafu. Aplikace umí exportovat a importovat data pro vytváření a práci s časovými osami a umí zobrazovat několik stupňů přiblížení. Aplikace byla testována pro větší objem dat, která byla získána z historických databází.

Obsah

1 Úvod

Pro výuku historie a dějepisu se na školách používá několik pomůcek mezi něž patří i časové osy. Tyto osy jsou většinou vytištěné na papír a poměrně jednoduché. Jedná se o jednu osu, na které jsou vyznačené časové údaje a události. Časová osa se ale může díky digitálnímu zpracování stát složitější a poskytnout více než jen znázornění událostí v toku času.

Mohla by obsahovat několik úrovní, na nejnižším přiblížení by bylo možné zobrazit jen ty nejdůležitější události a s vyšším přiblížením i ty méně důležité. Události by mohly být propojeny souvislostmi, které by neříkali jen která událost následuje za kterou, ale také jak jsou spolu události svázány logicky. Bylo by možné události opatřovat klíčovými slovy a umožnit jejich vyhledávání. Stejně tak by bylo možné události seskupovat pod určité větší celky.

Cílem této diplomové práce je vytvořit aplikaci, která bude poskytovat výše zmíněné možnosti. Bylo pro to třeba nastudovat dostupné aplikace, které umožňují vytváření časových os a zjistit jejich možnosti. Také bylo třeba ozkoušet a najít nejvhodnější ukládání dat nad kterými byla časová osa vytvořená a najít nejvhodnější možnost vizualizace.

2 Databáze historických dat

Pro naplnění aplikace historickými daty bylo třeba projít dostupné zdroje. Většinou se jednalo o online databáze dostupné bez potřeby platit za přístup nebo aplikace poskytující data z historických aplikací. U každé databáze bylo třeba zjistit v jaké podobě a formátu jsou data dostupná a co za údaje obsahují. Tyto poznatky byli použity pro návrh formátu úložiště dat aplikace.

2.1 Wikipedie[1]

2.1.1 Popis

Wikipedia je komunitní internetová encyklopedie, která je otevřená pro editaci od jejích uživatelů. Má přes 30 miliónů záznamů a podporuje dohledání primárních zdrojů, ale není kontrolovaná. Články se zabývají celou škálou oborů, jako je historie, umění a věda. Wikipedie je přístupná v 287 jazycích a každý jazyk má separátní záznamy pro jednotlivé články. Protože editace je přístupná víceméně komukoli, je třeba ověřovat informace. Na to slouží systém referencí a citací, které jsou u článků. Každý článek také obsahuje odkazy v textu na související záznamy na Wikipedii a na konci článků je možné nalézt externí odkazy, které tématem zabývají. U mnoha článků je přístupná tabulka se základními informacemi.

2.1.2 Způsob přístupu

Data na Wikipedii jsou volně přístupná ve formátech jako je HTML a XML. Protože jsou standardizované, daly by se tedy extrahovat pomocí parseru, toto platí zejména pro tabulku základních informací, která je pro jednotlivé typy záznamů stejná. Data je také možné kompletně stáhnout

2.2 DBPedia[2]

2.2.1 Popis

DBPedie je systém pro extrahování strukturovaných informací z Wikipedie a zpřístupňování těchto informací na internetu. Umožňuje pokládat sofistikované dotazy nad daty Wikipedie a umožňuje linkování datových setů s daty na Wikipedii. Data se z DBPedie dají získávat z několika článků najednou, například pro získání informací o díle, autorovi a jeho práci. DBPedie tak umožňuje získávat ucelenější kusy informací.

2.2.2 Způsob přístupu

DBPedie nabízí několik možností jak přistupovat k jejich datům. Každý z nich je pokládá dotazy nad datasetem, který je zrovna nahraný pomocí jazyka SPARQL.

Je možné pokládat dotazy těmito způsoby:

- tvůrce Leipzig dotazů[19]
- OpenLink Interactive SPARQL Query Builder (iSPARQL)[20]
- SNORQL dotazový průzkumník[21]
- jakýkoli jiný SPARQL klient.

2.3 History World[3]

2.3.1 Popis

Internetová encyklopedie obsahující více jak 10 000 světových událostí. Každý článek je rozdělený do logických celků a jednotlivé události v něm je možné zobrazit na časové ose. Také obsahuje odkazy na příbuzná témata nebo témata, jimiž se zabývá. Jednotlivé články nemají vypsané zdroje, ale ty jsou poskytnuty pro celou encyklopedii. Je možné zobrazit časové osy pro události podle klíčových slov a let.

2.3.2 Způsob přístupu

Informace je možné zobrazovat jako články nebo jako záznamy na časové ose. Přímý přístup k datům je možné pouze jako k html dokumentu. Články jsou většinou rozděleny podle časových období na paragrafy. Při zobrazení časových os se dá opět přistupovat k datům ve formě html dokumentu.

2.4 Facts on File[4]

2.4.1 Popis

Historická databáze určená pro vyhledávání záznamů podle klíčových slov. K dispozici je několik databází, ze kterých lze vyhledávat, jako je například „Amercian History Online„ a „Ancient and Mediaval Histroy Online“. Tím je možné zužovat nebo zobecňovat záběr vyhledávání. Výsledky jsou rozděleny do několika kategorií, např. biografie a události. U každého článku se uvádí zdroje, ze kterých čerpá a možné další zdroje informací. Stejně tak jsou u každého článku uvedeny příslušné citace.

2.4.2 Způsob přístupu

Informace jsou dostupné jako prostý text v html dokumentu a obsahují odkazy na související informace. U některých druhů článků jsou základní informace uvedené před hlavním článkem.

2.5 Ancient History Encyclopedia[5]

2.5.1 Popis

Jedná se o online encyklopedii, zaměřenou převážně na starověk. Vyhledávání je možné přes indexy nebo jako plnotextové vyhledávání článků. U každého článku je napsaný autor a datum publikování. Některé články jsou založené na Wikipedii, ale jsou ověřené a editované. V takovém případě jsou uvedeny informace o tom, kdo a kdy daný článek editoval. Články obsahují odkazy na související témata a příbuzné texty. Některé

články obsahují časové linie s možností vizualizace. Každý článek je rozdělen do časových období. Encyklopedie obsahuje i interaktivní mapy ukazující území a pohyb obyvatelstva podle doby. Encyklopedie umožňuje zobrazování časových os podle kategorií i podle časového rozmezí a klíčových slov. Časová osa pak obsahuje události barevně odlišené pro různé kategorie.

2.5.2 Způsob přístupu

Články jsou přístupné v prostém textu v html dokumentech s odkazy a časové osy mají dvě části, jednu javascriptovou s lepší vizualizací obsahu a druhou jednodušší, kde jsou pouze seřazené události, bez rozlišení ke které kategorii patří.

2.6 Wolfram Aplha[6]

2.6.1 Popis

Online služba zodpovídající zadané dotazy, čerpající odpovědi z externích ověřovaných dat. Namísto poskytování seznamu odkazů, zobrazuje základní informace, časové osy a související témata. Externí data se získávají jak z akademických tak komerčních zdrojů. Zobrazené informace jsou rozdělené do jednotlivých částí podle zobrazené informace. Části se liší podle typu hledané informace, pro osobu se může zobrazit časová linie, vztahy, fakta a základní informace. Některé z pokročilejších možností manipulace s daty jsou placené.

2.6.2 Způsob přístupu

Neplacená verze Wolfram Alpha má data přístupná v plain textu po použití jejich UI. Placená verze poskytuje data ve formě tabulky pro stažení.

2.7 HyperHistory[7]

2.7.1 Popis

HyperHistory obsahuje interaktivní časové osy pro události, historii a osobnosti. Časové osy obsahují barevné rozlišení pro kontinenty, v případě historie, a země, v případě osobností. Jednotlivé části osy se dají rozkliknout pro zobrazení podrobnějšího textu na straně. Texty obsahují základní informace a odkaz na rozšířenou verzi, většinou na externích stránkách. HyperHistory také obsahují mapy pro různé období lidské historie.

2.7.2 Způsob přístupu

K datům jako takovým se nedá lehce přistoupit, stránky jsou ve framesetech a pro přístup by bylo nutné zjišťovat jednotlivé stránky použité ve framesetech a z těch až pak čerpat data.

2.8 Srovnání historických databází

Jméno	Výhody	Nevýhody
Wikipedie	• Největší databáze • Semi-strukturovaná data	• Informace nemusí být exaktní • potřeba vytvořit parser pro data
DBpedia	• Strukturovaná data • Možno pokládat dotazy nad více články	• Závislé na datech ve Wikipedii • Závislé na načtených data-setech
History World	• Garantovaná přesnost • Formát dat sleduje časovou posloupnost	• Potřeba vytvořit parser pro data • Menší objem dat
Facts on File	• Přístup k více databázím • Výsledky rozděleny v kategoriích	• Potřeba vytvořit parser pro data
Ancient History Encyklopedia	• Články jsou rozděleny podle časového období • Možné čerpat data z časové osy	• Záměr na starobylou historii • Část dat je pouze v javascriptu • Potřeba vytvořit parser pro data
Wolfram Alpha	• Větší objem dat	• Nestandardizovaný formát databáze • Spoléhání na externí data • Použití UI
HyperHistory	• Rozdělení dat na události, osobnosti a historii	• Velmi obtížný přístup k datům

Tabulka 2.1: Srovnání historických databází

3 Software pro zobrazování časových os

Pro potřeby diplomové práce bylo nutné projít dostupný software pro vytváření a zobrazování časových os. Dostupný software se ukázal jako nedostatečný pro cíle, ale poskytl základní představu pro tvorbu časových os a přehled základních elementů, kterými takový software disponuje.

3.1 Tiki-Toki[8]

3.1.1 Popis

Web-based časová osa krom základní práce umožňuje tagování jednotlivých záznamů a jejich vyhledávání. Také umožňuje rozdělovat události do kategorií, které jsou zobrazovány jako sublinie. Časové osy lze stylizovat, například do 3D. Nad osou je možné zoomovat v rozsahu od měsíců do hodin. Rozsah osy je neomezený a lze zadávat data i před naším letopočtem. Bohužel, pouze základní verze je bezplatná, v té je možné vytvořit pouze jednu časovou osu. Nelze zaznamenávat souvislosti dat mezi sebou. Data se musí do osy zadávat ručně a nelze zaznamenávat souvislosti. V plánovaném rozšíření je importování dat skrze CSV.

Software vznikl v roce 2010 a poslední úprava byla 16.05.2014. Pro vytváření časových os využívá Tiki-Toki html a javascript. K dispozici je několik cenových balíčků, krom výše zmíněné verze zadarmo se cena pohybuje od 7,50 dolarů po 25 dolarů za měsíc. K dispozici je také speciální učitelský balíček s cenou 125 dolarů ročně.

3.2 Time Glider[9]

3.2.1 Popis

Web-based aplikace pro tvoření časové osy. Ovládání je vysoce interaktivní, umožňuje zoomovat a ukazovat tak časové úseky v různých délkách, od hodin po sto miliard let. Je možné zadávat události i úseky. Události se mohou překrývat. Každé události lze nastavovat důležitost. Rozsah osy je neomezený. Aplikace je zdarma pro studenty, ale i tak umožňuje vytvářet pouze 3 osy a omezuje počet zobrazení dané osy. Osy je možné vkládat do jiných stránek a lze ji exportovat jako CSV nebo JSON. V placených verzích aplikace je možné importovat data z CSV, ale verze zdarma toto neumožňuje.

Timeglider vznik v roce 2007, tehdy ještě pod názvem Mnemograph a poslední update byl 17.06.2013. Je postaven na technologiích html a javascript. K dispozici je výše zmíněná verze zdarma pro studenty nebo základní a skupinové balíčky. Základní balíčky jsou pro učitele, podnikatele a nadšence s cenou 5 dolarů měsíčně. Cena skupinových balíčků se liší podle velikosti skupiny od 24 dolarů měsíčně pro 5 lidí po 90 dolarů měsíčně pro 20 lidí.

3.3 Capzles[10]

3.3.1 Popis

Převážně multimediální časová osa pro zobrazování a ukládání obrázků a videí v časové posloupnosti. Nelze vytvářet záznamy, které by se skládaly pouze z textu. Do osy lze přidávat pouze soubory a to obrázky, videa, MP3, wordové, excelové a powerpointové soubory a PDF. Rozsah časové osy je od roku 1753 do roku 9999. Nelze importovat ani exportovat časovou osu. Umožňuje zobrazování obrázkových galerií a přehrávání videí. Zoomovat lze pouze vizuálně bez návaznosti na čas. Nelze zaznamenávat souvislosti událostí a neexistuje možnost zařazovat události do kategorií nebo sublinií. Je propojena se sociálními službami jako je twitter a facebook.

Bohužel Capzles nemá na svých stránkách žádné bližší informace o tom kdy byl

založen a jak je často je updatován. Nicméně má copyright od roku 2014. Celá webová stránka je vytvořena ve flashi a stejně tak i vytváření časových os používá flash. Capzles je zdarma bez prémiových účtů.

3.4 Read Write Think[11]

3.4.1 Popis

Umožňuje vytvářet základní formu časové osy a je převážně určena pro výuku. Umožňuje nahrávat obrázky a zobrazovat osu podle data, času nebo události. Časové osy lze ukládat v do souboru RWT formátu, což je binární soubor unikátní pro tuto časovou osu, a vkládat do webových stránek. Jednotlivé události nelze propojovat souvislostmi. Nad osou nelze zoomovat, ale rozsah není omezený. Datum událostí se vkládá ručně jako popisek u události. Nelze zadávat úseky událostí.

Na stránkách Read Write Think bohužel nejsou podrobnosti ohledně aplikace. Ta je poskytována společně s větším množstvím dalších aplikací jako prostředky pro rozvoj studentů. Celá aplikace je vybudovaná pomoci technologie flash a je zcela zdarma.

3.5 Timeline JS[12]

3.5.1 Popis

Vkládatelný tvůrce časových os. Časová osa se vytváří pomocí Google spreadsheetu, pro který má Timeline JS šablonu, jejímž vyplněním se vytváří události. Osu lze tedy exportovat a importovat jako soubor spreadsheety. K časovým osám je možné přidávat Wikipedia články, google mapy, youtube videa i náhledy webových stránek. Jednotlivé záznamy v časové ose lze propojit i s různými sociálními sítěmi jako je Twitter nebo Flickr. Text je možné formátovat skrze používání html tagů. Nad časovou osou je možné zoomova, nejnižší a nejvyšší hloubka závisí na přesnosti záznamů. Přesnost záznamů lze definovat na sekundy a lze zadávat data i před naším letopočtem, data nejsou omezena. Pro jednotlivé události nelze definovat souvislosti.

První verze byla vydaná v dubnu 2012 a poslední změna byla 04.06.2014. Zdrojový kód podléhá open-source licenci Mozilla Public License, v. 2.0 a je ho možné získat na GitHubu. Celá aplikace je programovaná v JavaScriptu.

3.6 TimeToast[13]

3.6.1 Popis

Jednoduchá webová interaktivní časová osa. Umožňuje jako elementy vlákat obrázky s textem. Umožňuje upravovat zoom pro lepší zobrazování časových posloupností a přepínání mezi časovou osou a listem čítajícím záznamy vytvořené v časové ose. Hloubka zoomu záleží na počtu záznamů v jednom místě a největším časové rozdílu mezi jednotlivými událostmi. Časovou osu lze vložit do jiné stránky. Verze zdarma umožňuje vytvořit pouze jednu časovou osu. Časová osa je omezena na data od roku 100 do roku 9999. Nelze vytvářet úseky událostí a nelze specifikovat souvislosti mezi jednotlivými událostmi.

TimeToast byl vytvořen v dubnu 2008 a poslední update nebylo možné dohledat. Aplikace využívá flash a javascript. Kromě výše zmíněné verze zdarma je možné platit 5,99 dolarů měsíčně za verzi basic a 8.99 dolarů měsíčně za verzi pro.

3.7 Dipity[14]

3.7.1 Popis

Webová aplikace pro vytváření časové osy. Události lze zadávat i před naším letopočtem. Umožňuje také přepínat zobrazení na seznam a prohlížeč jednotlivých událostí. Součástí aplikace je také mapa, kde se dají přiřazovat lokace k jednotlivým událostem. Nad zobrazením lze zoomovat a zobrazovat různé časové rozmezí od čtvrthodin po milénia. Události se pak budou zobrazovat podle důležitosti. Je možné zadávat události v časovém rozmezí od 9999 BC po 9999 AD. Také je možné vyhledávat události. Nelze propojovat události souvislostmi a lze zadávat časové úseky, ale ty jsou stejně zobrazeny pouze jako události. Verze zdarma neumožňuje importovat data a umožňuje vytvoření pouze tří os s maximálně 150 událostmi. Časovou osu lze vložit do webové stránky. Osy nelze jinak exportovat.

Dipity byla založena v dubnu roku 2007 a poslední update jsem nenašel, nicméně na stránkách je uvedeno, že aplikace je aktualizována týdně. Je vytvořena pomocí html a javascriptu. Krom verze zdarma jsou k dispozici placené verze od 4,95 dolarů měsíčně po 99,95 dolarů měsíčně.

3.8 ThinkPort Timeline[15]

3.8.1 Popis

Velmi jednoduchý tvůrce časových os, zvládá vytváření událostí a zadávání časů. K událostem lze přidávat obrázky. Nelze vytvářet časové období ani skupiny. Nad časovou osou lze měnit tři způsoby přiblížení. Nelze vytvářet souvislosti mezi jednotlivými událostmi. Data se zadávají ručně, takže lze vložit cokoli. Osy nelze exportovat a nelze importovat data.

ThinkPort Timeline je opět poskytovaná jako prostředek pro studenty, na stránkách není uvedeno datum vzniku, ani poslední datum změny. Aplikace je vytvořená ve flashi a je zdarma.

3.9 MyHistro[16]

3.9.1 Popis

Časová osa úzce propojená s mapou. Je možné zadávat události a využívat Google mapy pro zobrazování informací jako jsou přesuny, oblasti a umístění. Stejně tak je možné zadávat lidi, kteří jsou zapojeni do událostí na časové ose. Nad osou je také možné zoomovat od hodin po staletí. Časovou osu je možné exportovat do formátů CSV, PDF a KML. Exportovány jdou jednotlivé události, bez jakýchkoli vztahů, ale s mapovými informacemi. Události ani osu nelze importovat. Události lze zadávat i před naším letopočtem a je možné je zadávat jako úseky. Jednotlivé události se dají používat v různých osách.

MyHistro vzniklo v prosinci 2010 a poslední větší úprava byla 13.01.2014. Využívá technologií javascript a html. Apliace je zdarma.

3.10 TimeRime[17]

3.10.1 Popis

Multimediální časová osa, k událostem umožňuje připojovat youtube videa obrázky. Krom událostí je možné do časové osy přidávat období. V časové ose lze vyhledávat podle klíčových slov. Je možné zoomovat od sekund po maximální rozsah definovaný prvním a posledním záznamem. Také je možné definovat viditelnost jednotlivých událostí podle zoomu. Nelze definovat souvislosti mezi událostmi. Časovou osu lze vkládat na webové stránky. Nelze importovat nebo exportovat data.

TimeRime byl vydán v dubnu roku 2008 a poslední změna byla 05.12.2011. Aplikace je naprogramovaná ve flashi. Timerime je možné využívat zdarma nebo koupit, cena se pohybuje mezi 199 euro až 1999 euro. Také je možné místo koupě platit měsíčně od 12 euro po 125 euro podle verze.

3.11 Preceden[18]

3.11.1 Popis

Velmi základní web-based časová osa, určená pro školní výuku. Dobrý příklad jednoduchosti. Umožňuje do časové osy přidávat události a období. Událostem je možné přidávat popisy a poznámky. Rozsah událostí je omezen zleva rokem 9 999 999 999 před naším letopočtem. Události není možné propojovat souvislostmi. Osu lze rozdělit na vrstvy. Aplikace pak samostatně vypočítává věci jako dobu trvání. Je možné roztřídit jednotlivé události do různých oblastí. Nad časovou osou lze zoomovat od sekund po desítky miliard let. Osu lze vložit na webové stránky nebo stáhnout v PDF a CSV.

Preceden nemá uvedené datum vzniku, ani poslední úpravy. Pouze copyright pro rok 2014. Aplikace využívá html a javascript. K dispozici je verze pro studenty zdarma a Pro verze za 29 dolarů bez měsíčních poplatků.

3.12 Srovnání softwaru časových os

Časová osa	Datum vzniku	Datum poslední úpravy	Technologie	Cena
Tiki-Toki	Rok 2010	16.05.2014	Javascript, HTML	Základní verze zdarma Jinak od $7.50/měsíc do $25/měsíc Speciální učitelský balíček $125/měsíc
TimeGlider	Rok 2007	17.06.2013	Javascript, HTML	Verze zdarma bez exportu Placená verze pro jednoho člověka $5/měsíc Skupina 5 lidí $24/měsíc Skupina 20 lidí $90/měsíc
Capzles	Neuvedeno	Neuvedeno	Flash	Zdarma
ReadWriteThink	Neuvedeno	Neuvedeno	Flash	Zdarma
Timeline JS	Duben 2014	04.06.2014	Javascript, HTML	Licence Mozilla Public License, v2.0
TimeToast	Duben 2008	Neuvedeno	Flash, Javascript, HTML	Verze zdarma pro jednu časovou osu Verze Basic $5.99/měsíc Verze Pro $8.99/měsíc
Dipity	Duben 2007	Neuveden	HTML, Javascript	Verze zdarma bez importování dat Placené verze od $4.95/měsíc do $99.95/měsíc
ThinkPort Timeline	Neuvedeno	Neuvedeno	Flash	Zdarma
MyHistro	Prosinec 2010	13.01.2014	Javascript, HTML	Zdarma
TimeRime	Duben 2008	05.12.2011	Flash	Verze zdarma Jednorázová cena od €199 do €1999 Měsíční cena od €12 do €125
Preceden	Neuvedeno	Neuvedeno	Javascript, HTML	Verze pro studenty zdarma Jednorázová cena $29

Tabulka 3.1: Srovnání softwaru časových os

4 Graf

V této diplomové práci budou grafy využity pro udržování historických dat. Jednotlivé vrcholy udržují objekty jako jsou události, osoby nebo místa. Hrany pak propojují tyto objekty a odpovídají souvislostem a vztahům mezi objekty.

4.1 Definice

Graf G je dvojice G = (V, E), kde V je konečná množina a $E \subset \binom{V}{2}$, přičemž

(4.1)

$\binom{V}{2} = \{\{x, y\} : x, y \in V \; a \; x \neq y\}$ je množina všech dvouprvkových množin

(4.2)

(*neuspořádaných dvojic*) prvků množiny V . Prvky množiny V nazýváme *vrcholy* (často také *uzly*), prvky množiny *E* pak *hrany* grafu *G*. Vrcholy $x, y \in V$ jsou *sousední*,

(4.3)

pokud $\{x, y\} \in E$.[22]

(4.4)

4.2 Algoritmy

Při používání grafu pro reprezentaci databáze bude důležité správné využití grafových algoritmů. Zejména pak metrik grafů pro rychlý odhad náročnosti vyhledávání a přehledu o velikosti grafu. Dalšími použitými algoritmy bude vyhledávání v grafu a hledání cest mezi vrcholy. Jako poslední důležitou vlastnost grafu bych chtěl uvést stupeň vrcholu grafu, protože ten bude určovat počet souvisejících událostí daného vrcholu.

4.2.1 Metrika grafu

Metrika grafu je reprezentace vzdáleností v daném grafu. V této diplomové práci bude reprezentovat relevantnost vrcholů vůči sobě. Čím nižší vzdálenost mezi vrcholy, tím větší šance, že jsou relevantní vůči dané události.

Definice vzdálenosti je následující: Vzdálenost *d(x; y)* vrcholů *x; y* orientovaného grafu *G* je délka nejkratší cesty z *x* do *y*. Pokud taková cesta neexistuje, položíme *d(x; y) = 1*.

Pak pro metriku platí[22]:

Nechť *G* je souvislý neorientovaný graf. Pak funkce *d(x; y)* je metrikou na množině *V (G)*, tj. má následující vlastnosti:

1. $d(x,y) \geq 0$ *(4.5)* přičemž *d(x; y) = 0*, právě když x = y,
2. *d(x; y) = d(y; x)*;
3. $d(x,y) + d(y,z) \geq d(x,z)$ *(4.6)* („trojúhelníková nerovnost“):

4.2.2 Stupeň uzlu

Počet hran vedoucí do a z uzlu bude v této diplomové práci určovat důležitost daného uzlu, pokud nebude překryta uživatelským nastavením. Lze předpokládat, že uzly které mají více souvislostí s ostatními uzly jsou důležitější, nežli uzly s méně souvislostmi. Princip fungování je odvozen od technologie PageRanku použitého internetovým vyhledávačem Google.

Definice stupně uzlu je následující: Stupeň uzlu u v grafu G je počet hran grafu G, které obsahují uzel u.

Platí také věta[22]: Pro každý graf G platí $\sum_{u \in U(G)} d_G(u) = 2|H(G)|$. To znamená, že *(4.7)*

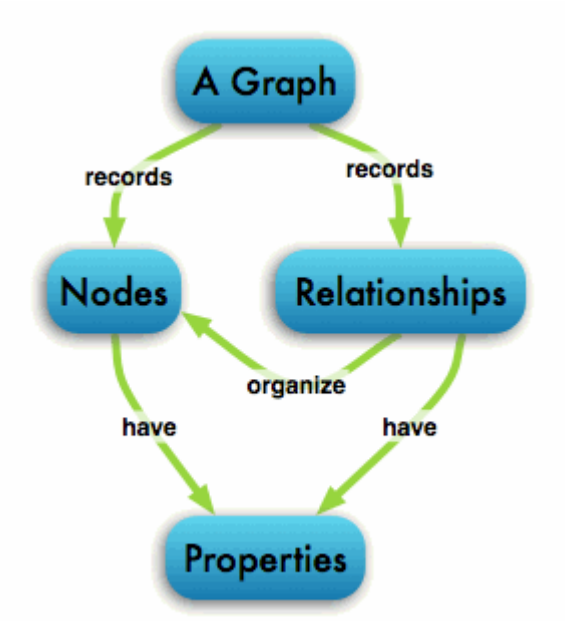
každý uzel může být propojen jen jednou hranu. Lze ale předpokládat, že některé uzly v této diplomové práci budou mít více souvislostí. Proto budou hrany navíc ohodnoceny, aby toto odrážely.

5 Grafová databáze

[25]V této kapitole se budeme věnovat grafovým databázím, co přesně jsou a jak se liší oproti klasickým relačním databázím. Jaké vlastnosti musí databáze mít, aby se mohla považovat za grafovou a jaké existují rozdíly mezi zpracováním jednotlivých grafových databázích.

5.1 Co je grafová databáze

Grafová databáze je systém řízení umožňující vytváření, čtení, měnění a mazání nad odhaleným datovým modelem grafu. Grafové databáze jsou obvykle postaveny na použití transakčních (OLTP) systémů. Zpravidla jsou optimalizovány na transakční výkon a vyvíjeny s důrazem na transakční integritu a operační dostupnost.



Obrázek 5.1: Architektura grafové databáze

5.1.1 Rozdíl grafové oproti relační databázi

Klasické relační tabulkové databáze jsou schopny propojovat jednotlivé tabulky skrze cizí klíče. Takové propojení ale postrádá jakékoli informace, funguje čistě jako kotva jedné tabulky v druhé. Nemají kvantifikovanou sílu nebo váhu.

Čím složitější je tabulková databáze, tím více je třeba přidávat join tabulek, které alespoň z části klarifikují vztahy mezi jednotlivými tabulkami. To samozřejmě snižuje výkon a zvyšuje nákladnost správy na aplikační úrovní. Shrnuto, grafová databáze se snaží vylepši tyto problémy relačních databází:

- Join tabulky, které přidávají složitost mícháním dat s metadaty cizích klíčů
- Cizí klíče, které zvyšují čas trvání vývoje a přidávají nutnou údržbu, aby relační databáze fungovala
- Řídce osídlené tabulky s nullable sloupci, které vyžadují kontrolu v aplikaci
- Nutnost použití několika joinů pro provedení relativně jednoduchých dotazů jako je například „jaký produkt si zákazník koupil?“
- Drahé použití rekurzivních dotazů, tj. „kteří zákazníci si koupili tento produkt, kteří také koupili tamten produkt?“

Na rozdíl od klasický relačních databází mají databáze grafové propojení představované vztahy mezi jednotlivými elementy. Díky vztahům tak máme na první pohled jasně definovaný obrázek toho jak jsou elementy propojeny v grafu a díky tomuto můžeme plně využít možností, které nám nabízejí grafové databáze.

Vztahy v grafu přirozeně vytvářejí cesty. Traverzování těchto cest nahrazuje klasické dotazování na databázi. Toto dotazování je vysoce efektivní, protože většina operací v grafové databázi je těsně propojena s tím jak jsou data v databázi uložena.

Jako příklad zde poslouží tabulka uvedená v knize Neo4j in action[26], která srovnává dobu provádění dotazu hledajícího cestu skrze přátele dvou lidí na sociální síti, která bude maximálně o velikosti pěti vztahů. Dotaz je prováděný nad tabulkou s jedním milionem lidí, každý s zhruba 50 přáteli. Použitá grafová databáze je Neo4j.

Hloubka	RDBMS doba provádění	Neo4j doba provádění	Vráceno záznamů
2	0,016	0,01	~2500
3	30,267	0,168	~110 000
4	1543,505	1,359	~600 000
5	nedokončeno	2,132	~800 000

Tabulka 5.1: Srovnání RDBMS a grafových databází

Jak je možno v tabulce vidět, grafové databáze vyžadují zlomek času, který vyžadují klasické relační databáze pro vyhledání v propojených datech.

5.2 Vlastnosti grafové databáze

Grafová databáze má dvě důležité vlastnosti. První je jakým způsobem ukládá data a druhá je, jakým způsobem k nim přistupuje a zpracovává je – procesní engine.

5.2.1 Ukládání dat

Některé grafové databáze používají *nativní grafové úložiště*, které je optimalizované a designované pro ukládání a řízení grafů. Jiné grafové databáze serializují data do relačních databází, objektově orientovaných databázi nebo jiných univerzálních datových úložišť

5.2.2 Procesní engine

Některé grafové databáze používají indexy volné sousednosti (index-free adjacency), které zajišťují, že propojené uzly na sebe vzájemně fyzicky v databázi „ukazují“. Je možné vytvořit grafové databáze bez indexů volné sousednosti, pokud je zajištěno, že CRUD operace vytváří perspektivu grafové databáze. Nicméně, indexy volné sousednosti poskytují signifikantní výhodu ve výkonu grafových databází a Ian Robinson používá termín *nativní grafové zpracování* pro popis grafových databází jež jej poskytují[25].

5.2.3 Rozdíl nativních zpravcování oproti univerzálním

Nativní zpracování nemusí být nutně výhodnější oproti univerzálním a je třeba mít na paměti, že ač nabízejí lepší výkonnost a škálovatelnost díky tomu, že byly vytvořeny přesně pro tuto práci, jsou oproti univerzálním méně časté a tedy složitější na naučení oproti systémům jako je MySQL. N*ativní grafové zpracování* poskytuje lepší traverzovací výkon, ale za cenu obtížnějších ne-traverzovacích dotazů a zvýšení paměťové náročnosti.

6 Návrh aplikace

V této kapitole se budeme věnovat návrhu aplikace, tak aby splňovala požadavky na ni kladené. První krok pro navržení aplikace bude správně zanalyzovat požadavky a podle nich zvolit vhodné nástroje pro implementaci aplikace tak, aby bylo možné dosáhnout všech cílů.

6.1 Specifikace požadavků

Na aplikaci jsou kladeny následující požadavky:

- Aplikace musí být schopná uchovávat historické události a objekty.
- Tyto události a objekty budou provázány souvislostmi
- Události a objekty budou mít časové údaje pro jejich zobrazení na časové ose, mohou být buď bodem v čase nebo mít trvání od - do
- Data také musí mít hodnotu, která bude určovat jejich viditelnost
- Data v databázi budou poskytována aplikaci, která je bude zobrazovat jako časovou osu, včetně souvislostí mezi událostmi a objekty
- Aplikace musí mít GUI, skrze které bude možné editovat data v úschovně. Bude možné vytvářet nové, editovat a mazat existují události a objekty. Také bude možné mezi nimi vytvářet, editovat a mazat souvislosti.
- Primární zaměření aplikace bude na skladování a práci s daty. GUI bude sloužit hlavně pro kontrolu správné funkčnosti aplikace a korektnost dat.

6.2 Analýza požadavků

Aplikace musí být schopná ukládat a nahrávat jednotlivé časové osy, takže bude potřebovat úložiště do kterého se budou ukládat jednotlivé prvky časové osy. Bude třeba ukládat části grafu tak aby se udrželo jejich propojení a atributy. Vzhledem k objemu datům bude potřeba použít úložiště s dostatečnou kapacitou a schopností rychlého vyhledávání a procházení.

Také bude třeba zvolit způsob, jakým bude časová osa zobrazována pro uživatele. Neměl by to být složitý způsob, protože to není hlavní zaměření této práce, ale musí být dostatečně robustní pro splnění požadovaných cílů a demonstrace funkčnosti aplikace.

6.3 Volba implementačních prostředků

Java 7 byla zvolena jako programovací jazyk pro aplikaci, protože poskytuje mnoho různých knihoven, které je možné použít pro práci s grafy. Také umožňuje přenositelnost mezi různými operačními systémy. Protože GUI bude použita hlavně pro ozkoušení funkčnosti aplikace, je výběr Javy optimální i pro jednoduché vytváření a práci s GUI.

S ohledem na výběr Javy je možné se při výběru datového úložiště zaměřit na různé knihovny, které můžou poskytnout použitelné technologie. Data která budeme ukládat budou rozsáhlá, ale budeme je potřebovat mít k dispozici a prohledávat je. Proto bylo jako úložiště zvolena databáze. Protože data, která budou ukládána obsahují vrcholy a vztahy, představují funkčně graf. Je tedy nejvhodnější použít grafovou databázi, která poskytne efektivní přístup k datům. Pro ukládání dat jsem vybral grafovou databázi Neo4j, která má i všechny potřebné vlastnosti, které jsou potřebné pro splnění cíle diplomové práce. Specifické důvody jsou rozebrány v kapitole 7.3.1.

Pro zobrazení grafu v GUI pro uživatele byla zvolena knihovna GraphStream. Ta umožňuje nejen zobrazování grafu jako takového, ale i interakci s ním. To umožní vytváření, editaci a likvidaci prvků grafu v GUI. Bližší informace o GraphStreamu jsou v kapitole 8.1.2.

Přestože v Javě existují třídy pro reprezentaci dat a časů, nelze je v našem případě použít. Hlavním problémem je práce s daty před rokem 1970. S ohledem na tyto problémy byla zvolena knihovna Joda-Time pro reprezentaci dat a času v aplikaci.

6.4 Návrh architektury

Protože aplikace bude využívat databázi pro ukládání dat a GUI pro jejich zobrazování, typ architektury byl zvolen jako klasická třívrstvá. To umožní samostatný vývoj jednotlivých modulů a jejich případnou náhradu, pokud v budoucnu bude třeba část aplikace změnit nebo nahradit. Stejně tak to zjednoduší možné napojení nebo uplatnění části aplikace v jiné.

7 Návrh databáze

Protože databáze bude obsahovat pouze vrcholy a hrany, je třeba zvolit druhy vrcholů a hran, které bude obsahovat. Druhy vrcholů budou rozhodovat o tom, jaký typ historických dat budou udržovat a typ hrany bude rozhodovat jaký vztah představuje.

7.1 Návrh vrcholů

Pro dobrý návrh databáze je nejdříve třeba zvolit druh vrcholů, ten by měl být co nejvíce vysoko úrovňový, protože dodatečné informace budou doplněny v jeho vlastnostech. Zvolil jsem čtyři hlavní typy vrcholů: Person, Event, Place, Item. U každého vrcholu jsou doplněny základní vlastnosti vrcholu, název klíče pod kterým budou uschovány a typ objektu, který budou používat v aplikaci.

7.1.1 Person

Tento typ vrcholu představuje důležitou historickou osobnost a udržuje vlastnosti spjaté s jejím životem. Příkladem může být „Karet IV Luxebmburský“ nebo „Albert Einstein“.

- Jméno – firstname String
    - Tato vlastnost udržuje křestní jméno osoby, z hlediska historie a časové osy je důležité pro odlišení členů jednotlivých rodů. Součástí jména vládců se bere i jejich pořadí v rodu. Tato vlastnost se bude udržovat v řetězci. Příkladem je „Karel IV“ nebo „Albert“.

- Příjmení – lastname String
    - Vlastnost udržující rodové jméno nebo příjmení. Umožňuje vyhledávání jednotlivých rodin nebo rodových linií. Příjmení bude reprezentováno řetězcem. Jeho příkladem je „Luxemburský“ nebo „Einstein“

- Titul – title String[]
    - Titul může být buď akademický, vladařský, profesní nebo se může jednat o oficiální pozici. Těchto titulů může být u jedné osoby více a proto budou udržovány v poli řetězců. {„Císař“, „Král“}, {Docent, Profesor}

- Specifické datum narození – specificBirthDate DateTime
    - Určuje specifické datum narození dané osobnosti v tradičním formátu dne, měsíce a roku. Bude použito pro osobnosti, které se narodily nedávno nebo jejichž datum narození je přesně známo. Velmi důležité pro správné umístění osobnosti na časové ose. Bude udržováno java typem Date. Příkladem je „26.12.1904“

- Specifické datum úmrtí – deathDate DateTime
    - Určuje specifické datum úmrtí dané osobnosti obdobně jako specifické datum narození. Bude udržováno java typem DateTime z knihovny Joda-Time. Příkladem je „03.05.1953“.

- Přibližný rok narození – approximateBirthYear Long
    - Obsahuje přibližné narození dané osobnosti, které bude použité v případě neznalosti přesného data narození osobnosti nebo při narození před naším letopočtem. V tom případě je značeno jako záporné. Přibližné rok narození bude použitý pouze tehdy, pokud není použité specifické datum narození. Bude udržováno v primitivní Java typu Long. Příkladem je „-30000“ nebo „546“.

- Přibližný rok smrti – approximateDeathYear Long
    - Přibližný rok smrti je vlastnost obdobná přibližnému roku narození. Používá se při neznalosti specifického data smrti nebo když je rok smrti před naším letopočtem. Pak je značeno jako záporné. Bude použito pouze při nevyplnění specifického data smrti. Java typ Long bude udržovat tuto vlastnost. Příkladem je „-120“ nebo „217“.

- Oblast historického významu – fieldImportance String[]
    - Tato vlastnost určuje do které historické oblasti daná osobnost spadá. Může se jednat o výzkum, politiku, válku nebo jiné oblasti lidské historie. Protože jedna osobnost může spadat do více kategorií, bude se tato vlastnost udržovat v poli řetězců.

- Podrobné informace – detailedInfo String
    - Tato vlastnost bude udržovat text popisující dopodrobna život osobnosti. Bude uložen v řetězci.

7.1.2 Event

Event bude typ vrcholu udržující důležité historické události. Tento vrchol bude mít vlastnosti, které specifikují o jakou událost se jedná. Mezi důležité události například patří „Druhá světová válka“, „Vynález kola“.

- Jméno – eventName String
    - Řetezec představující jméno pod kterým je událost tradičně známá. Příkladem může být „Druhá světová válka“ nebo „Vynálze kola“.

-

- Specifické datum začátku události – eventSpecificStartDate DateTime
    - Tato vlastnost bude představovat začátek události, pokud je známé přesné datum a datum začátku je v tradičním formátu dne, měsíce a roku. Pro události, které nemají trvání nebo nemají konec, tak představuje bod v čase, kdy událost nastala. V aplikaci je představován Java třídou DateTime z knihovny Joda-Time. Jako příklad je „1.1.1993“

- Specifické datum konce události – eventSpecificEndDate DateTime
    - Vlastnost udržující specifický datum konce události obdobně jako specifické datum začátku události. Může být prázdné pro události bez trvání nebo konce. Udržováno v Java třídě DateTime z knihovny Joda-Time. Příklad je například „29.12.1989“

- Přibližný rok začátku – eventApproximateEndDate Long
    - V této vlastnosti se bude udržovat rok začátku události, pokud událost nemá specifické datum nebo začala před naším letopočtem. V případě začátku před naším letopočtem je hodnota záporná. Bude se používat pouze tehdy, když nebude vyplněný specifické datum začátku události. Stejně jako specifické datum začátku, pokud je událost bez trvání nebo nemá konec, bude představovat bod v čase kdy událost nastala. Tato vlastnost bude udržována Java primitivním typem Long. Příkladem je například „-3000000“ nebo „0“.

- Přibližný rok konce – eventApproximateStartDate Long
    - Obdobně jako přibližný rok začátku, tato vlastnost udržuje přibližný rok konce události. Může být prázdný pro události bez trvání nebo konce. Vlastnost bude mít Java typ Long.

- Oblast historického významu – fieldImportance String[]
    - Tato vlastnost je obdobná jako u osoby, ale obsahuje oblasti specifické pro události.

- Podrobné informace – detailedInfo String
    - V této vlastnosti budou uloženy veškeré ostatní informace o událost, včetně souvislého textu o události. Tento text bude uložen v řetězci.

7.1.3 Place

Vrchol typu Place bude udržovat historická místa, lokace a politické útvary. Může se jednat o státy, země. Tento vrchol nebude zobrazován na časové ose, ale poskytuje umístění pro události, osoby a věci. Bude použito pro vyhledávání spojených vrcholů.

- Jméno – placeName String
    - Obsahuje jméno místa uloženého v řetězci. Příkladem je „Anglie“ nebo „Evropa“.

- Podrobné informace – detailedInfo String
    - Tento řetězec obsahuje podrobné informace o místu. Může být použit pro fulltextové vyhledávání.

7.1.4 Item

Tento vrchol představuje věci, může se jednat o vynálezy, dokumenty nebo technologii.

- Jméno – itemName String
    - Tato vlastnost obsahuje jméno věci v řetězci. Příkladem je „Zlatá Bula Sicilská“ nebo „Atomová bomba“.

- Specifické datum vytvoření – specificCreationDate DateTime
    - V této vlastnosti je uloženo specifické datum vytvoření věci v tradičním formátu dne, měsíce a roku. Pro tuto vlastnost je použit Java typ DateTime z knihovny Joda-Time. Příkladem je „16.05.1945“

- Přibližný rok vytvoření – approximateCreationYear Long
    - Pokud není známé přesné datum vytvoření nebo je před naším letopočtem, uloží se do této vlastnosti přibližný rok vytvoření věci. Pokud je před naším letopočtem, udá se v záporné hodnotě. Použije se pro ni Java typ Long. Příkladem je „-2600“.

- Oblast historického významu – fieldImportance String[]
    - Udržuje pole řetězců určující oblasti do kterých věc spadá. Může se jednat o politické oblasti, výzkumné nebo technologické. Příkladem je například „Fyzika“, „Biologie“ nebo „Povolení“.

- Podrobné informace – detailedInfo String
    - Tato vlastnost obsahuje podrobný text o objektu. Je uložena v řetězci.

7.1.5 Vlastnosti shodné pro všechny vrcholy

- Označení – tags String[]
    - Vlastnost udržující uživatelem definované označení vrcholu, které se nemusí týkat pouze historických dat jako takových. Může se jednat o označení důležitých vrcholů pro danou časovou osu bez ohledu na důležitost historických dat. Příkladem může být: „Důležité“, „Nedostatečně informace“ nebo „Rozpor“

- Viditelnost – visibility Int
    - Viditelnost označuje uživatelem specifikovanou oblast přiblížení na které je vrchol vidět. Pokud není vyplněná bude se viditelnost počítat podle stupně vrcholu. Tato vlastnost bude uložena jako primitivní typ Integer s rozsahem od 1 do 5. Příkladem je „1“ nebo „4“.

7.2 Návrh vztahů

Teď když jsou navrhnuty vrcholy, tak je třeba zjisti jaké vztahy mezi nimi mohou existovat. Postupným procházením dvojic vrcholů určíme existující vztahy a vytvoříme celkovou množinu vztahů a určíme jejich vlastnosti. Tímto způsobem jsem určil následující vztahy: Relationship, Interaction, Participation, Creation, Cause, Part_of, Takes_Place

7.2.1 Relationship

Tento typ vztahu bude určovat vztahy mezi dvěma osobami nebo dvěma místy. Může se jednat například o vztah mezi rodičem a dítětem nebo manželem a manželkou. U míst může jít o přerod jedné země do druhé, příkladem může být vztah Československé Republiky a Česka a Slovenska.

7.2.2 Interaction

Typ vztahu, který poukazuje na spolupráci nebo jednání mezi dvěma osobami. Může se jednat o dva vědce na spolupracující na vynálezu, dva státníky jednající o míru nebo válce. Příkladem může být Bill Hewlett a David Packard.

7.2.3 Participation

Tento vztah ukazuje na účast osoby nebo místa na určité události. Může se jednat o účast země ve válce nebo státníka na konferenci. Například účast Velké Británie v Druhé světové válce.

7.2.4 Creation

Vztah, který ukazuje ukazuje na vrchol zodpovědný za tvorbu jiného. Může se jednat o vznik místa, věci, osoby nebo události. Stejně tak může být osoba, událost, věc nebo místo zodpovědná za vznik. Příkladem je Albert Einstein a Teorie Relativity.

7.2.5 Cause

Tento typ vztahu odráží příčinou souvislost. Může nastat mezi dvěma událostmi, věcí a událostí nebo věcí a věcí. Jako příklad je možné uvést Sarajevský atentát a První světovou válku.

7.2.6 Part_of

Vztah poukazující na vrchol jež je součástí jiného. Většinou se jedná o dva stejné typy vrcholů. Příkladem může být Skotsko a Velká Británie nebo Bitva o Midway a Druhá světová válka.

7.2.7 Takes_Place

Tento vztah určuje, kde nastala určitá událost. Propojuje tedy místo a událost. Jednat se může například o Bitvu o Midway a ostrov Midway.

7.2.8 Vlastnosti všech vztahů

Každý vztah bude mít několik vlastností, které budou určovat přesnější podrobnosti o vztahu mezi dvěma vrcholy. Přestože existuje několik typů vztahů, všechny vztahy budou mít stejné vlastnosti.

- Jméno – name String
    - Vlastnost udržující jméno vztahu v řetězci. Použito pro jednodušší lokalizaci vztahu v grafu. Příkladem může být „Záminka První světové války“

- Podrobnosti o vztahu – detailedInfo String
    - Tato vlastnost udržuje podrobnosti o vztahu v souvislém textu. Může se například jednat o popis vztahu mezi Skotskem a Velkou Británií.

- Označení – tags String[]
    - V tomto poli řetězců bude udržováno uživatelem specifické označení, pro snazší vyhledávání vztahů a traversování grafů. Příkladem je „Důležité“, „Válka“, „Teorie“.

- Viditelnost – visibility Int
    - Stejně jako u vrcholů, viditelnost určuje při jakém přiblížení bude hrana vidět. Pro viditelnost bude použit ty Integer v rozahu od 1 do 5.

7.3 Grafová databáze Neo4j

Tato databáze umožňuje ukládat data v grafech místo v tabulkách a nabízí plnou perzistenci. To umožňuje vynechat prostředníka při používání a ukládání grafů vytvořených v aplikaci. Grafové databáze ukládají data ve vrcholech, které mají vlastnosti.

Vrcholy jsou organizovány podle vztahů, které mají také vlastnosti. Také je možné seskupovat vrcholy a přiřazovat k nim popisy. Tyto vlastnosti Neo4j splňují požadavky, které tato aplikace má pro ukládání dat a může nahradit část logické části aplikace.

7.3.1 Důvod volby Neo4j

Na rozdíl od většiny ostatních grafových databázích má Neo4J veřejnou verzi, která je open-source s licencí GPL. Databáze splňuje ACID podmínky a je vytvořena v Javě. Umožňuje používat databázi přímo vloženou do Java aplikace, bez nutnosti použití databázového serveru. Má stabilní verzi a je stále vyvíjena takže se dá očekávat její další použitelnost v budoucnosti. K datům je přístup přes CypherQuery a vlastní JavaApi.

7.3.2 Vytváření vkládané databáze

Vkládanou databázi lze vytvářet v jakémkoli umístění, které neobsahuje jinou neo4j grafovou databázi. Databáze se vytvoří následujícím příkazem:

```
new GraphDatabaseFactory().newEmbeddedDatabase(databasePath);
```

Kde `databasePath` je cesta ke složce, kde bude databáze umístěna. Tento příkza vrací objekt `GraphDatabaseService`, který umožňuje s databází pracovat.

7.3.3 Přístup do existující databáze

Pokud již na disku databáze existuje a chceme s ní pouze pracovat, použijeme příkaz

```
new GraphDatabaseFactory().newEmbeddedDatabase(databasePath);
```

Kde `databasePath` je umístění složky obsahující databázi. Příkaz opět vrací objekt `GraphDatabaseService`, který umožňuje s databází pracovat.

7.3.4 Vyhledávání v databázi

Z grafové databáze lze vyhledávat uzly obdobně jak v databázi tabulkové. Pomocí příkazu `new ExecutionEngine(graphDb);` lze vytvořit objekt, který umožňuje provádět Cypher dotazy nad databází. Ty pak vrací výsledek v objektu `ExecutionResult`, kde jsou uloženy výsledky v sloupcích a řádcích.

Obdobně jako výsledky z tabulkové databáze. Na rozdíl od tabulkové databáze je však možné ve výsledku získat přímo objekt Node, který obsahuje atributy vrcholu a jeho hrany. Řádky výsledku je možné iterovat a vybírat z nich příslušné hodnoty.

7.3.5 Procházení databáze

Pokud máme k dispozici výchozí uzel, je možné neo4j databázi procháze pomocí objektu `TraversalDescription`. Chování procházení je možné dospecifikovávat pomocí následujících metod:

- `depthFirst()`
  - Zařizuje použití procházení do hloubky.
- `breadthFirst()`
  - Zařizuje použití procházení do šířky.

- `relationships(RelationType relType, Directions dir)`
  - Definuje druh hran, které smí při procházení použít. Druhy hran jsou definovány parametrem `relType` jako enumerace RelationType a její implementace. Je možné definovat jestli hrana může být příchozí nebo odchozí, což se zadává v parametr `dir`.

- `uniqueness(Uniqueness uniq)`
  - Tato metoda definuje zda lze uzel/hranu projít znovu. Je třeba brát v úvahu zvýšené využití paměti při zakázání procházení uzlu/hran v důsledku ukládání uzlů do paměti pro kontrolu průchodu. Typ unikátnosti uzlu/hrany se definuje parametrem `uniq`

- `evaluator(Evaluator eval)`
  - Umožňuje určit, zadáním parametru `eval`, jestli bude uzel zahrnut ve výsledku a jestli bude procházení grafu pokračovat nebo ne. Mezi podmínky patří například je-li uzel startovací, je-li uzel v určité vzdálenosti od startovacího nebo měl-li poslední uzel hranu určitého typu.

- `expand(RelationshipExpander relExpand)`
  - Tato metoda umožňuje rozšířit procházení o uživatelem definované podmínky. Další podmínky je možné definovat implementací třídy `RelationshipExpander` a použitím její instance jako parametr `relExpand.`

- `order(BranchOrderingPolicy bop)`
  - Umožňuje obecnější přístup k definici procházení grafu než

`depthFirst()` a `breadthFirst()`. Je například možné vytvořit různé kombinace postorder, preorder a breadth, depth procházení. Je možné zadat toto procházení jako parametr `bop`.

- `reverse()`
  - U daného procházení prohodí příchozí a odchozí podmínky pro hrany.

- `sort(Comparator comp)`
  - Umožňuje použít `Comparator` pro třídění výsledků.

7.3.6 Konec práce s databází

Práci s databází lze ukončit použitím metody `shutdown()` nad objektem databáze. Po použití této metody nelze pracovat s API neo4j.

7.3.7 Použité metody v databázové části aplikace

Databázová část aplikace je zodpovědná za tvoření a přístup k databázi. Také musí být schopna ukládat, mazat a načítat vrcholy a hrany. Ukládání hran a vrcholů je poměrně přímočaré, ale k načítání se dá přistoupit dvěma způsoby. Buď se v databázi plošně vyhledají uzly a hrany je spojující podle specifických parametrů nebo se určí začáteční uzel a parametry pro procházení po hranách a uzlech.

První způsob je výhodný pro počáteční načtení grafu nebo pro vyhledávání specifických uzlů podle parametrů určených uživatelem. Druhý způsob je vhodný pro vyhledávání souvislostí mezi uzly a pro tvorbu podgrafů založených na na startovacím uzlu a prohledávání do určité hloubky.

Nedjůležitější použité metody budou následující:

- `void createDatabase(String databasePath)`
  - Metoda zodpovědná za vytváření vložené databáze Neo4j. Parametr `databasePath` určuje složku, ve které bude tato databáze uložena.
  -
- `void loadDatabase(String databasePath)`
  - Tato metoda načte databázi uloženou v složce, která je určená parametrem `databasePath`.
- `void shutDown()`
  - Účel této metody je přerušit připojení do databáze, použije se při vypnutí aplikace nebo při načtení jiné databáze.
- `GraphData getNodeAndEdges(Map<String, Object> properties)`
  - Tato metoda prohledá graf a najde uzly podle vlastností určených v parametrů `properties`. Vrací grafové data uložená v třídě GraphData.

- `GraphData traverseGraph(Map<String, Object> properties, NodeData startingNode)`
  - Metoda, která projde graf z uzlu určeného parametrem `startingNode` skrze hrany a uzly určené vlastnostmi v parametru `properties`. Vrací grafové data uložené ve třídě GraphData.

- `void SaveNode(NodeData node)`
  - Skrze tuto metodu se dá uložit nový uzel do databáze. Data uzlu se předají metodě v parametru `node`.

- `void editNode(NodeData node)`
  - Tato metoda změní data uzlu uloženého v databázi. Z parametru `node` se načtou nová data a zároveň i identifikátor pro nalezení uzlu v databázi.

- `void deleteNode(NodeData node)`
  - Metoda pro smazání grafu z databáze. Identifikátor mazaného uzlu se načte z parametru `node`.

8 Návrh logiky

Celá aplikace bude založena na GUI, které bude zobrazovat graf a bude umožňovat uživateli provádět nad grafem požadované změny. V této kapitole se budeme věnovat návrhu zobrazování grafu a návrhu GUI, tak aby splňovalo požadavky kladené na aplikaci.

8.1 Práce s časy a daty

Jak již bylo zmíněno ve podkapitole 6.3, přestože Java má vlastní třídy pro práci s daty a časem, tyto třídy nejsou vyhovující pro reprezentaci dat a časů a to z těchto důvodů:

- Třída Date ve skutečnosti nereprezentuje datum, ale timestamp. To je problematické při práci s daty před rokem 1970
- Neexistuje jednoduchý práce a konverze s komponenty data (den, měsíc a rok), pro to by bylo potřeba použít třídy Calendar, která ale vyžaduje více práce a je zbytečně složitá pro jednoduché uchovávání data

Pro lepší reprezentaci dat byla vybrána knihovna Joda-Time, která poskytuje třídy s rozšířenou funkčností pro práci s daty a časem. Tato knihovna je upravena tak, aby výše zmíněné problémy opravila. Je časem prověřená a existuje k ní dostatek dokumentace, což s ní usnadňuje práci.

8.2 Vizualizace grafu

Vizualizace grafu bude využita pro zobrazování hotové časové osy. Vrcholy grafu budou představovat objekty osy jako jsou události a lidé, hrany mezi osami budou představovat souvislosti. Pro vytvoření vizualizace grafu byla vybrána java knihovna GraphStream.

8.2.1 Knihovna GraphStream

Tato javovská knihovna se zaměřuje nejen na zobrazování grafů, ale i na dynamickou práci s nimi. Na grafových elementech zajišťuje ukládání jakýchkoli dat a umožňuje přidávat a odstraňovat jak vrcholy, tak hrany. Také umožňuje editaci, přidávání a odebírání atributů jednotlivých vrcholů a hran. To zajistí interaktivitu časové osy.

8.2.2 Důvod volby GraphStream

GraphStream nabízí stabilní verzi a je stále ve vývojem na rozdíl od většiny ostatních populární Java knihoven pro vizualizaci grafů. GraphStream je také zdarma pod licencí GNU General Public Licence. K dispozici je pro ni jasná dokumentace a návody k použití.

Mezi její schopnosti patří: Vytváření grafické i logické podoby grafu, vytváření, mazaní a propojování vrcholů a hran, změna vlastností hran a uzlů, změna pozic hran a uzlů, odchytávání událostí nad grafem, nastavování vzhledu grafu a umožňuje kreslení do grafu.

8.2.3 Vytvoření grafu jako pohledu

Pokud je potřeba vytvářet graf jako součást existujícího GUI, vytvoří se takový graf jako pohled(View) a vloží se do příslušného JFrame. Knihovna GraphStream umožňuje vytváření několika druhů grafů. Graf použitý v diplomové práci je GraphicGraph, protože umožňuje odchytávat GUI události nad pohledem. Graf se vytváří pomocí příkazu `new GraphicGraph(graphName);` kde parametr `graphName` je jméno grafu.

Pro vytvoření pohledu je třeba nastavit objekt `Viewer` následujícím příkazem:

```
new Viewer(graph, Viewer.ThreadingModel.GRAPH_IN_SWING_THREAD);
```

Parametry je předem vytvořený graf a ThreadingModel, který upřesňuje v jakém vlákně bude graf použitý. To je důležité, protože pro odchytávání GUI událostí je potřeba vlastního vlákna.

Objektu `Viewer` je možné nastavit další atributy, například `enableAutoLayout()`, který nastavuje automatické rozvržení grafu. `Viewer` pak může například metodou `addDefaultView(createNewWindow)` přidat View, který se zakomponuje do JFrame. Parametr `createNewWindow` určuje zda se bude vytvářet zvlášť okno pro graph nebo se použije existující.

8.2.4 Vytváření uzlů a hran

Pro vytváření uzlů je potřeba volat na grafem metodu `addNode(nodeName)`, kde parametr `nodeName` je jméno uzlu. Obdobně, hrana se tvoří použitím metody `addEdge(edgeName, firstNodeName, secondNodeName)`, kde `edgeName` je jméno hrany, `firstNodeName` je jméno prvního uzlu, který bude hranou propojen a `secondNodeName` je jméno druhého uzlu, který bude touto hranou propojen.

8.2.5 Atributy uzlů a hran

Po vytvoření uzlů a hran je možné přidávat jednotlivým objektům atributy pomocí metody `addAttribute(attributeName, attributeValue)`, kde parametr `attributeName` je jméno atributu a `attributeValue` je jeho hodnota.

Je samozřejmě možné si vytvářet vlastní atributy, ale je několik atributů, které jsou předdefinované. Patři mezi ně například „`ui.label`“, který vypíše hodnotu tohoto atributu nad příslušný element v grafu. Mezi další předdefinované atributy patří „`x`“ a „`y`“, což jsou koordináty elementu.

8.2.6 Nastavení GUI událostí grafu

Pro odchytávání GUI událostí na grafu je třeba vytvořit třídu implementující `MouseManager`. V tom je možné překrýt metody zpracovávající GUI události jako je například `mouseClicked(MouseEvent event)` nebo `mouseDragged(MouseEvent`

`event)`. Při inicializace objektu této třídy je potřeba předat mu graf a view, pro odchytávání GUI událostí. Vytvořený objekt `MouseManager` je pak nutno předat metodou `addMouseListener(mouseManager)` volanou nad objektem View.

8.2.7 Metody použité v aplikaci

V aplikaci se používají metody hlavně propojující GUI a databázi. Metody logiky jsou shodné s databázovými metodami. Krom těchto metod existuje základní metoda, které vytvářejí GUI. Logika obsahuje datové třídy `GraphData`, `EdgeData`, `NodeData`. Ty jsou použity pro ukládání dat grafu.

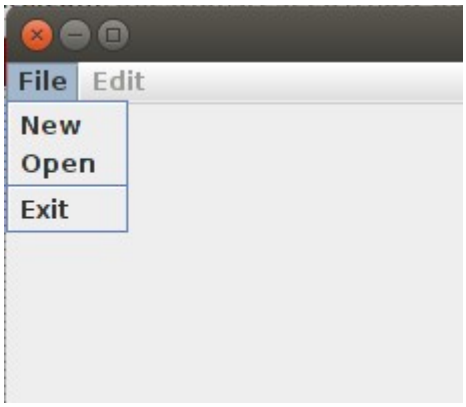
Třída `GraphData` obsahuje množiny hran a uzlů uložených ve listech tříd `EdgeData` a `NodeData`. Ve třídě `NodeData` jsou obsaženy vlastnosti uzlu, který tato třída představuje a jeho identifikáor. Třída `EdgeData` obsahuje stejné vlastnosti pro hranu a zároveň i identifikátory uzlů, které spojuje.

9 Návrh GUI

V této kapitole bude třeba navrhnout GUI tak, aby mělo všechny potřebné prvky pro interakci s grafem a pro práci s databází.

9.1.1 GUI pro práci s databází

Práce s databází vyžaduje pouze dva prvky interakce. Vytváření nové databáze a přístup k již vytvořené databázi. Tyto možnosti budou k dispozici v klasickém horním menu aplikace. V kategorii „File“ budou tlačítka „New“ a „Open“, které budou vytvářet novou nebo zpřístupňovat existující databázi. Také bude v této kategorii obsaženo tlačítko „Exit“, které ukončí aplikaci.

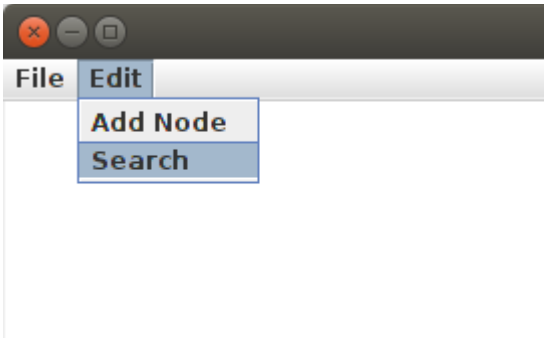


Obrázek 9.1: File menu

9.1.2 GUI pro práci s grafem

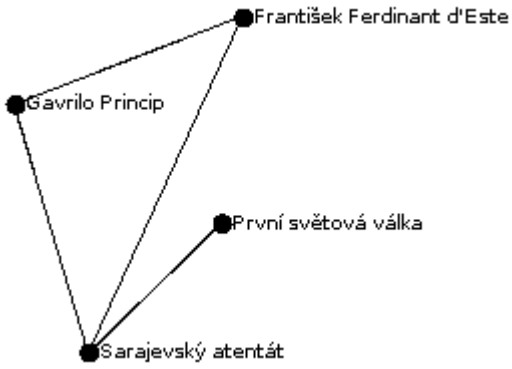
Základní vkládání vrcholů a hran se provádí přes kategorii menu „Edit“. Toto menu je přístupné pouze pokud je vytvořena databáze. Tato kategorie bude obsahovat dvě tlačítka. „Add Node“ otevře dialog pro přidání nového vrcholu. V něm bude možno zadat podrobnosti vrcholu, které jsou zmíněné v kapitole 7.1.

Tlačítko „Search“ bude otvírat dialog, který umožní specifikovat parametry vyhledávání a procházení grafu, které je nastíněno v kapitole 7.3.5.



Obrázek 9.2: Edit menu

Zbylé akce zařizuje interakce myši s grafem. Dvojitým poklepáním na vrchol nebo hranu se otevře editační dialog. Táhnutím myši z vrcholu na vrchol se tyto vrcholy propojí hranou. Stejně tak kliknutí pravým tlačítkem myši odstraňuje vrcholy a hrany.



Obrázek 9.3: Ukázka grafu

V levém horním rohu aplikace je posuvník, který ovládá přiblížení a s tím i viditelnost vrcholů a hran. Pokud se přiblížení změní tak, že uzel přestane být viditelný automaticky se skryjí i všechny hrany, které vedou z nebo do daného uzlu.

Obrázek 9.4: Posuvník

9.1.3 Jaká data bude možné filtrovat

Důležitá vlastnost aplikace bude filtrování a vyhledávání v grafu, podle parametrů obsažených ve vrcholech a hranách. Vyhledávání se bude provádět pomocí výše zmíněného formuláře a bude umožňovat vyhledávání nejen jednotlivých uzlů, ale i jeho vlivy do určité hloubky. Například při vyhledávání souvislostí pro uzel František Ferdinant d'Este v Obrázku 8.3 do hloubky 1, je možné najít událost Sarajevský atentát a Gavrilo Principa. Při vyhledávání do hloubky 2 najdeme i První světovou válku.

9.1.4 Metody použité v GUI

V GUI jsou metody zodpovědné hlavně za vytváření oken pro vkládání dat, vytváření menu, jejich tlačítek a odchytávačů akcí nad nimi provedených. Mezi méně obecné metody patří ty zodpovědné za vykreslování grafu skrze knihovnu GraphStream a pro práci s grafem. Mezi tyto důležité metody patří:

- `createNewGraph()`
    - Tato metoda vytváří základní pohled s grafem v něm. Tento pohled se umisťuje do aktivního okna aplikace.

- `createGraph()`
  - Metoda, vytvářející graf z dat v tříde GraphData.
- `setVisibility(int visibility)`
  - Tato metoda nastavuje viditelnost nad grafem, ta určuje přiblížení a viditelnost uzlů a hran. Parametr `visibility` určuje nastavení viditelnosti u grafu.

- `addNewNode(NodeData newNode)`
  - Metoda přidávající uzel do grafu s daty uloženými v parametru `newNode`.

- `editNode(NodeData newNode)`
  - Metoda editující uzel v grafu podle dat uložených v parametru `newNode`.

- `removeNode(NodeData newNode)`
  - Metoda odstraňující uzel z grafu podle identifikátoru uloženém v parametru `newNode`.

10 Možná rozšíření

Aplikace má základní funkcionalitu specifikovanou v cílech této diplomové práce. Je ale možné některé části rozšířit či zefektivnit. Tato kapitola má za cíl popsat možná rozšíření a jejich výhody, které byli zjištěny v průběhu práce na aplikaci.

10.1 Rozšíření práce s databází

Hlavní vylepšení práce v databázi by bylo v rychlejší a efektivnějším přístupu k datům. Protože objekty získané přístupem k databázi jsou pouze aktivní během dané transakce, je třeba je předávat do vlastní datové třídy. To není problém při nižším počtu vrcholů a hran, ale s rostoucí rozlehlostí grafu a propojením uzlů to zpomaluje přístup k databázi. Ukládání grafových dat do negrafových tříd také komplikuje přístup k těmto datům. Pokud by se podařilo toto nějakým způsobem obejít v dalších verzích aplikace, přineslo by to rychlejší a efektivnější přístup k datům.

10.2 Rozšíření GUI

Protože hlavní důraz byl kladen na práci s daty, GUI je místo nabízející nejvíce možností pro rozšíření. Knihovna GraphStream nabízí mnoho možností pro úpravu grafů, které tvoří. Nejdůležitější část grafu je jeho schopnost odchytávat události, které se nad ním můžou odehrát. Skrze tyto události by bylo možné přidat další možnosti práce s časovou osou. Příkladem může být možnost přesouvat uzly a hrany, aby uživatel mohl přesněji definovat pozici na časové ose. To by mohlo být propojené s časovým údajem vrcholu.

Další možné rozšíření by bylo přesunutí časové osy z 2D do 3D. To by mohlo být umožněno budoucím update knihovny GraphStream, která by měla umožnit modelování grafů ve 3D

11 Testování

Testování aplikace bylo provedeno hlavně z hlediska uživatelské přehlednosti a použitelnosti. Protože má být aplikace využita pro výuku, je důležité vědět, jak jsou uživatelé schopni pracovat s aplikací. Které části jsou nejasně vysvětleny nebo nepřehledné. Pro otestování GUI byla vytvořen případ použití, který ozkoušel vzorek uživatelů. Pak vyplnil formulář obsahující otázky ohledně přehlednosti a použitelnosti GUI.

11.1 Forma testování

Základní testovací scénář byl tento:

1. Vytvoření databáze
2. Nahrání databáze
3. Vytvoření uzlů a jejich naplnění daty
4. Editace existujících informací v uzlech
5. Smazání několika uzlů.
6. Propojení uzlů hranami a zadaní informací o hranách
7. Zobrazení podgrafů pomocí vyhledávání

11.2 Testovací formulář

Testovací formulář obsahoval otázky zaměřené na jednotlivá kroky v případu použití. Tyto otázky byly shrnuty do třech oblastí: Práce s databází, vytváření uzlů a hran, vyhledávání v grafu.

11.2.1 Práce s databází

Otázky v této oblasti byly zaměřeny na přehlednost vytváření a nahrávání databáze. Každá otázka byla hodnocena na škále od 1 do 10, kde 1 je nejhorší a 10 nejlepší. Otázky obsažené v této sekci byly:

- Snadnost a přímočarost vytvoření databáze
- Snadnost a přímočarost nahrání databáze
- Přehlednost oken pro výběr lokace databáze

11.2.2 Vytváření uzlů a hran

V této oblasti jsou otázky zaměřené na práci s uzly a hranami v načteném grafu. Opět byly hodnoceny jako předchozí otázky. Tato sekce obsahovala následující otázky

- Snadnost a přímočarost vytvoření uzlu
- Snadnost a přímočarost editace uzlu
- Snadnost a přímočarost vytvoření hrany mezi uzly
- Snadnost a přímočarost smazání uzlu
- přehlednost formulářů pro zadávání informací o uzlech
- Přehlednost grafu jako časové osy

11.2.3 Vyhledávání v grafu

Tato oblast obsahovala otázky se zaměřením na vyhledávání grafu. Opět byly hodnoceny stejně jako předchozí otázky. Sekce obsahoval tyto otázky:

- Snadnost a přímočarost zadávání informací pro vyhledávání
- Přehlednost formulářů pro zadávání informací pro vyhledávání
- Přehlednost výsledného podgrafu

11.3 Výsledky testování

Tato sekce obsahuje výsledky testování. Ty jsou uvedeny v tabulce 11.1 jako aritmetický průměr všech dotazníků poskytnutých testovacími uživateli. U každé otázky jsou uvedeny i nejvyšší a nejnižší hodnoty a rozptyl.

Otázka	**Průměr**	**Nejvyšší hodnota**	**Nejnižší hodnota**	**Odchylka**
Vytvoření databáze	8	7	9	
Nahrání databáze	9	6	10	
Přehlednost oken	7	5	8	
Vytvoření uzlu	6	5	9	
Editace uzlu	7	4	8	
Smazání uzlu	7	6	9	
Vytvoření hran	8	6	10	
Přehlednost formulářů	9	8	10	
Přehlednost grafu	7	6	8	
Zadání informace pro vyhledávání	6	4	7	
Přehlednost formulářů	9	7	10	
Přehlednost podgrafu	8	5	10	

Tabulka 11.1: Přehled výsledků testování

12 Závěr

Hlavním cílem této diplomové práce bylo vytvořit funkční datovou základnu pro aplikaci tvoření časové osy. Druhotným cílem bylo vytvoření GUI, které by umožnilo kontrolu dat a položilo základ pro ovládání aplikace.

Vytvořená aplikace je schopná vytvářet a nahrávat grafovou databázi a v této databázi vytvářet grafy, modifikovat je a mazat. Dále je schopná v těch grafech vyhledávat určité elementy podle vlastností. Dále pak může vytvářet podgrafy procházením databáze podle kritérií určených uživatelem.

Aplikace byla testována pro jednoduchost použití, přehlednost GUI a jasné použití jednotlivých částí. Testování bylo provedeno na vzorku uživatelů od kterých byla shromážděna zpětná vazba k jednotlivým oblastem použití aplikace. Z této zpětné vazby byl vytvořen jednoduchý přehled oblastí aplikace z hlediska testovaných hledisek.

Výsledkem této diplomové práce je aplikace poskytující funkční nástroj pro výuku historie. Zároveň umožňuje rozšíření GUI podle dalších požadavků.

Použité zkratky

XML - Extensible Markup Language

HTML - HyperText Markup Language

SPARQL - SPARQL Protocol and RDF Query Language

CSV – Comma-Separated Values

JSON - JavaScript Object Notation

PDF - Portable Document Format

AD - Anno Domini

BC - Before Christ

KML - Keyhole Markup Language

OLTP - Online Transaction Processing

CRUD - Create, Read, Update and Delete

SQL - Structured Query Language

GUI – Graphic User Interface

GPL - General Public License

ACID - Atomicity, Consistency, Isolation, Durability

API - Application Programming Interface

Seznam použitých zdrojů

[1] *Wikipedia*, online [14.04.2014], http://www.wikipedia.org

[2] *DBPedia*, online [14.04.2014], http://www.dbpedia.org

[3] *HistoryWorld*, online [14.04.2014], http://www.historyworld.net

[4] *Facts on File*, online [14.04.2014], http://www.fofweb.com/History/Reference.asp?ID=17241

[5] *Ancient History, online [14.04.2014]*, http://www.ancient.eu.com

[6] *Wolfram Alpha, online [14.04.2014]*, https://www.wolframalpha.com

[7] *HyperHistory, online [14.04.2014]*, http://www.hyperhistory.com/online_n2/History_n2/a.html

[8] *Tiki-Toki, online [20.04.2014]*, http://www.tiki-toki.com

[9] *Time Gilder, online [20.04.2014]*, http://timeglider.com

[10] *Capzles, online [20.04.2014]*, http://www.capzles.com

[11] *Read Write Think Timeline, online [20.04.2014]*, http://www.readwritethink.org/files/resources/interactives/timeline_2

[12] *Timeline JS, online [20.04.2014]*, http://timeline.knightlab.com

[13] *Timetoast, online [20.04.2014]*, http://www.timetoast.com

[14] *Dipity, online [20.04.2014]*, http://www.dipity.com

[15] *Thinkport: My Timeline, online [20.04.2014]*, http://timeline.thinkport.org

[16] *MyHistro, online [20.04.2014]*, http://www.myhistro.com

[17] *Timerime, online [20.04.2014]*, http://timerime.com

[18] *Preceden, online [20.04.2014]*, http://www.preceden.com

[19] *DBPedia QueryBuilder, online [14.04.2014]*, http://querybuilder.dbpedia.org

[20] *OpenLink iSPARQL*, *online [14.04.2014]*, http://dbpedia.org/isparql

[21] *SPARQL Explorer ,online [14.04.2014]*, http://dbpedia.org/snorql

[22] Roman Čada, Tomáš Kaiser, Zdeněk Ryjáček, *Diskrétní matematika*, 2004, Katedra matematiky FAV, Západočeská univerzita v Plzni, Plzeň

[23] Jiří Demel, *Grafy a jejich aplikace*, 2002, Academia

[24] Ján Plesník, *Grafové algoritmy*, 1983, Veda, Bratislava

[25] Ian Robinson, Jim Webber, Emil Eifrem, *Graph Databases*, 2013, O'Reilly Media

[26] Jonas Partner, Aleksa Vukotic, Nicki Watt, *Neo4j in action*, 2013, Manning

Uživatelská příručka

Tvorba databáze:

Vytvoření databáze se provede vybráním tlačítka „New“ z menu „File“. Pak se vybere adresář do kterého se nově vytvořená databáze umístí. Po vytvoření databáze se zobrazí prázdný graf se kterým lze pracovat

Nahrání existující databáze:

Nahrání databáze se provede vybráním tlačítka „Open“ z menu „File“. Pak je možné vybrat adresář ve kterém je uložena databáze. Po vybrání se zobrazí graf uložený v databázi. S tím lze pak dále pracovat

Přidání vrcholu:

Přidání vrcholu se provede dvojklikem levým tlačítkem myši do prostoru grafu. Po dvojkliku se otevře formulář pro přidávání nového vrcholu ve kterém lze specifikovat bližší podrobnosti a vlastnosti.

Editace vrcholu:

Provádí se dvojklikem na existující uzel grafu. Po dvojkliku se otevře formulář pro editaci uzlu. Je obdobný jako formulář pro přidání vrcholu, ale již s předvyplněnými informacemi.

Smazání vrcholu:

Existující vrchol lze smazat, pokud se na něj klikne pravým tlačítkem myši. To automaticky smaže i všechny jeho příchozí a odchozí hrany

Přidání hrany:

Přidání hrany se provede stisknutím levého tlačítka myši nad jedním vrcholem, přetažením na cílový vrchol a puštěním levého tlačítka myši. Poté se otevře formulář přidání hrany, kde se specifikují jednotlivé vlastnosti hrany.

Editace hrany:

Pro editaci hrany je potřeba dvojkliknout levým tlačítkem myši na označení hrany, které se nachází v jejím prostředku. To otevře formulář editace hrany, který je obdobný jako formulář přidání hrany, ale již obsahuje předvyplněné informace.

Smazání hrany:

Hranu je možné smazat pravým kliknutím na označení hrany.

Vyhledávání:

Vyhledávání se provádí kliknutím na tlačítko „Search“ v menu „Edit“. To otevře formulář pro vyhledávání, ve kterém je možné specifikovat typ vyhledávání a jeho parametry. Po potvrzení vyhledávání se zobrazí příslušný podgraf.