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

Bakalářská práce

Těžba grafových dat z Wikipedie

Plzeň 2016 Martin Mach

PROHLÁŠENÍ

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

V Plzni dne 4. května 2016 . . . . . .

Martin Mach

ABSTRAKT

Cílem této práce je vytvořit nástroj umožňující zpracování historických událostí z Wikipedie do podoby grafu. Vzhledem k povaze zdroje dat a požadavku na interakci s uživatelem byl za platformu zvolen webový prohlížeč Google Chrome a jeho systém rozšíření. Vytvořený nástroj umožňuje výběr preferovaných článků, na základě nichž jsou nabízeny další články dle odhadu jejich relevance. Obsah článků je zpracováván a jsou z něj získávány informace specifické pro typ dané historické události. Mezi jednotlivými články je hledán vzájemný vztah, který potom, spolu s typem tohoto vztahu, tvoří hrany získaného grafu. Vytvořené řešení poskytuje uživateli možnost zpracovat historické události do formy rešerše reprezentované grafem. Množství získávaných informací a jejich vzájemnou spojitost je možné ovlivnit pomocí systému modulů. Výsledek je poté možno zobrazit na časové ose, případně je možné upravit výstup aplikace tak, aby mohl být zobrazen nástrojem dle volby uživatele.

KLÍČOVÁ SLOVA

Wikipedie, těžba dat, graf, historická událost, časová osa, Google Chrome, rozšíření

ABSTRACT

The purpose of this thesis is to create a tool which would allow to process historical events from Wikipedia into the form of a graph. Due to the nature of the data source and the requirement for user interaction it was decided to choose Google Chrome and its extensions as a platfrom. The tool allows user to choose articles of his choice, based on which it will find and recommend other articles according to their estimated relevance. The content of the articles is processed and used as a source of informations specific for chosen historical event. The edges of the created graph are represented by found relationships between the articles and their types. The solution allows user to transform historical events into the form of a graph represented summary. The amount of retrieved informations and their mutual connection can be modified by the system of modules. The exported results can be displayed on a timeline. It is also possible to customize the output so it can be rendered by a tool of user's choice.

KEYWORDS

Wikipedia, data retrieval, graph, historical event, timeline, Google Chrome, extension

MACH, Martin *Těžba grafových dat z Wikipedie*: bakalářská práce. Plzeň: Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky, 2016. 58 s. Vedoucí práce byl Ing. Richard Lipka, Ph.D.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Richardu Lipkovi, Ph.D. za odborné vedení, vstřícný přístup, trpělivost a podnětné návrhy k práci.

OBSAH

1 ÚVOD

Internet je zdrojem nepřeberného množství veřejně dostupných dat. Tato data jsou často sice použitelná, z hlediska strojového zpracování jsou ale velmi špatně zpracovatelná a tedy nepřipravená pro další použití. Pro další využití těchto dat je nutné je určitým způsobem zpracovat – unifikovat jejich podobu a formát, zasadit je do kontextu ostatních dat, případně spojit více zdrojů.

Jedním z nejznámějších a nejrozsáhlejších veřejně přístupných zdrojů dat je v současné době Wikipedie. Tento projekt využívá odbornou veřejnost pro tvorbu obsahu, který je pak poskytován formou webové encyklopedie. Vzhledem k množství tohoto obsahu, který je navíc částečně strukturovaný a tedy lépe strojově čitelný, to činí Wikipedii vhodným zdrojem dat. Cílem této práce je zanalyzovat dostupné nástroje pro práci s Wikipedií a s jejich pomocí následně navrhnout a vytvořit aplikaci, která dokáže vhodným způsobem tato data získat. Preferovaná pak budou data o historických událostech, obecně tedy články mající časový rozsah, pomocí něhož mohou být následně zasazeny do kontextu s ostatními událostmi.

Význam takto získaných dat ale vždy záleží až na způsobu jejich využití. Pro hlubší a rychlejší pochopení problému je nutné získaná data uživateli určitým způsobem reprezentovat. Spíše než prostý výčet dat je vhodnější využít vizualizaci, která působí z pohledu uživatele přehledněji a přívětivěji. V případě historických událostí se na první pohled jeví jako vhodný způsob reprezentace časová osa. Je nutné vytvořit nástroj, který veřejně přístupná data dokáže využít a připravit je pro potřeby vizualizace.

Aplikace, která bude v rámci této práce vytvořena, tedy umožní snazší a rychlejší přípravu dat o historických událostech z Wikipedie. Uživateli dá možnost vybrat jím preferované články. Ty aplikace vhodně analyzuje a po zpracování příslušné části Wikipedie se pokusí ostatním článkům přiřadit ohodnocení a uživateli prezentovat články jevící se jako nejrelevantnější.

Výběr platformy, s kterou bude nástroj pracovat, bude záležet na výsledku analýzy, jejímž cílem bude především shrnout výhody a nevýhody jednotlivých přístupů k Wikipedii. Tento výběr bude učiněn s ohledem na jednoduchost implementace (z programovacího hlediska), stabilitu a dostupnost řešení. Platforma musí být také aktuální a aktualizovaná, aby umožnila dlouhodobou funkčnost vytvářeného nástroje.

Výstupem aplikace bude graf, jehož vrcholy budou představovat jednotlivé historické události, které byly do výběru zařazeny. Tyto vrcholy budou obsahovat dodatečné informace, jako název, stručný popis, obrázek a další položky. Hrany mezi nimi budou značit určitý vztah mezi články. Tento výstup bude následně zkontrolován a ohodnocen především s ohledem na přesnost a úplnost – u některých témat lze jednoduše předvídat, jaké články by měly být navrhovány a které jsou naprosto nevhodné, obecné či nepříbuzné.

2 WIKIPEDIE

Wikipedie[1] je webová encyklopedie tvořená dobrovolníky a odbornou veřejností. Vytváření obsahu je založeno na otevřenosti – nový článek či úpravu existujícího může publikovat kdokoliv. Wikipedie existuje v několika stovkách jazykových mutací, přičemž nejobsáhlejší je anglická verze [1].

2.1 Obsah

Při užívání Wikipedie a jí podobných projektů je nutné brát v úvahu původ veškerého obsahu, který nemusí být vždy zcela autentický. Podle práce publikované v magazínu Nature je však možné tvrdit, že obecně Wikipedie neobsahuje více nepřesností[2], než klasické encyklopedie (v případě tohoto porovnání *Encyclopædia Britannica*) [2]. Místo nepřesností se v článcích projevuje spíše autorův názor a informace, s kterými autor nesouhlasí, jsou záměrně opomíjeny [3]. Stránky jsou také často cílem vandalismu či úmyslného pozměňování některých informací. I přesto, že jsou tyto chyby poměrně rychle odhalovány a opraveny zpět [4], není možné se vždy spolehnout na autenticitu aktuálního obsahu.

Kromě samotného hlavního textu články na Wikipedii obsahují tzv. infoboxy, což jsou oddíly umístěné v pravé části stránky obsahující především heslovité informace o aktuálním článku. V případě infoboxů lze využít při zpracování vlastnosti, že tyto informace jsou strukturované.

2.2 Získávání dat z Wikipedie

Data z Wikipedie lze získávat v zásadě dvěma způsoby – offline a online. Oba z těchto přístupů mají své výhody a nevýhody. Po získání obsahu konkrétní stránky jej pak můžeme vhodným způsobem analyzovat.

2.2.1 Offline přístup

Programy umožňující prohlížení Wikipedie offline pracují s tzv. *dumpem* – vyexportovaným kompletním obsahem (většinou v rámci článků jednoho jazyka). Tento dump soubor[3] má řádově desítky GB (komprimovaný pak necelých 10 GB). Vzhledem k povaze a rozsahu staženého souboru je pro práci s dumpem nutné využít některý ze specializovaných programů.

[1] V originále Wikipedia – kombinace slov *wiki* a *encyclopedia*.

[2] Toto tvrzení se týká anglické verze Wikipedie. Důvěryhodná práce hodnotící spolehlivost verze české v současnosti neexistuje.

[3] Dostupný na `https://meta.wikimedia.org/wiki/Data_dump_torrents`.

Výhodou tohoto přístupu je především rychlost – práce s dumpem je rychlejší, než přístup k webovým stránkám, a celý soubor lze tak na dostatečně výkonném počítači zpracovat v rámci hodin. Dále pak odpadají problémy plynoucí z používání Wikipedie na serveru (zpravidla cizím) – tedy kontrola počtů přístupů, stažených článků a omezování v návaznosti na aktuální zátěž serveru. Nevýhodou je pak neaktuálnost, často také závislost na softwaru třetích stran a hlavně nedostupnost. Dané programy jsou většinou výhradně prohlížeče, poskytují tedy pouze přístup k obsahu a jakékoliv práce s ním je možné dosáhnout jen zásahem do zdrojového kódu. V případě, že chceme upravit existující program pro zpracování Wikipedie k určitému účelu, je nutné vzít v úvahu vlastnosti již existujícího zdrojového kódu. Díky tomu musíme výrazně omezit případnou volbu vhodného programovacího jazyka, což může mít negativní vliv na efektivitu a rychlost samotného procesu zpracovávání.

WikiTaxi

Pravděpodobně nejpoužívanější nástroj pro offline práci s Wikipedií. Jeho obrovskou výhodou je, že pracuje s původními neupravenými dumpy, uživatel tak není závislý na zpracování dumpu třetí stranou. Bohužel ale není open source[4] [5].

XOWA

Open source program velmi úspěšně kopírující živou verzi Wikipedie. Pracuje s vlastní verzí dumpu, minoritní jazyky (včetně češtiny) jsou tedy aktualizovány zřídka [6].

Kiwix

Multiplatformní open source aplikace dostupná i pro mobilní operační systémy Android a iOS. Pracuje s vlastním dumpem, který je aktualizován zhruba jednou za půl roku. Díky podpoře více platforem v rámci jednoho zdrojového kódu (C++) je ale velmi obtížné tuto aplikaci jakkoliv modifikovat [7].

WikiExtractor.py

Knihovna napsaná v Pythonu pracující s oficiálním dumpem, umožňující extrakci článků do jednoduššího formátu (s odstraněním často nepotřebného formátování). Nevýhodou tohoto přístupu je kompletní odstranění hypertextových odkazů a infoboxů – knihovnu je tedy vhodné použít při zájmu o skutečný text článků [8].

[4] Program s otevřeným, veřejně dostupným zdrojovým kódem.

2.2.2 Online přístup

Programy pro offline práci se zaměřují čistě na čtení Wikipedie, naproti tomu aplikace pro práci s online verzí Wikipedie jsou úzce specializované. K obsahu lze přistupovat pomocí zpracování klasické webové stránky nebo pomocí poskytovaného programovacího rozhraní (API[5]), případně kombinací těchto způsobů.

Vzhledem k obtížnosti zpracovávání HTML kódu je první ze způsobů vhodné zvolit pouze v případě, že uživatel pracuje s programem, který zobrazuje webovou stránku a v rámci tohoto prohlížení probíhá dané zpracování – typicky webový prohlížeč. Nejpoužívanější webové prohlížeče dnes umožňují tvorbu a instalaci rozšíření, která pak mohou pracovat s právě zobrazenou stránkou.

V ostatních případech je vhodnější a spolehlivější využít API Wikipedie. To umožňuje získat více informací o požadované stránce, vybírat pouze část obsahu a podobně. Drtivá většina aplikací pro práci s živou verzí Wikipedie pracuje právě s API.

Wikipedia–js

Knihovna napsaná v JavaScriptu (respektive v jeho mutaci pro Node.js) umožňující jednoduché dotazování na články. Dokáže získané články zobrazit v několika formátech, avšak pracuje pouze s hlavním textem (či volitelně jen s perexem) [9].

Wtf–wikipedia

JavaScriptová knihovna, která dokáže článek stáhnout pomocí API a zpracovat jej do formátu JSON, s nímž už JavaScript dokáže bez problémů pracovat. Dokáže automaticky od hlavního textu oddělit infobox a poskytnout jej programátorovi ve formě objektu, respektive asociativního pole [10].

2.2.3 Odvozené projekty

Za speciální kategorii mohou být považovány projekty od Wikipedie odvozené. Ty využívají Wikipedii jako zdroj informací, které zpracovávají a mohou k nim připojovat kontext či jiné souvislosti. Často také Wikipedii propojují s jinými zdroji a stávají se tak agregátory těchto informací.

DBpedia

DBpedia je projekt tvořený komunitou [11]. Jeho cílem je získávat z (nejen) Wikipedie strukturovaná data a zpřístupňovat je pomocí vlastního rozhraní. Je tak možné se dotazovat pomocí speciálního jazyka zvaného `SPARQL` (syntaxí je tento jazyk podobný SQL). Toto

[5] API označuje způsob, kterým aplikace či služba umožňuje pracovat ostatním se svými zdroji a funkcemi.

rozhraní, kromě Wikipedie, rovněž zahrnuje zdroje z jiných databází a otevřených zdrojů. Užitečnou vlastností může být podpora české verze Wikipedie [12].

BabelNet

Babelnet se snaží pomocí Wikipedie a dalších projektů o vytvoření lexikografické a sémantické sítě [13]. Ke každému pojmu se snaží přiřadit význam a propojit ho se synonymy a dalšími příbuznými tématy. Umožňuje propojování i napříč různými jazykovými verzemi. V současnosti podporuje 271 jazyků, včetně češtiny.

Semantic MediaWiki

Samentic MediaWiki se rovněž snaží k informacím v článcích na Wikipedii (obecně „wiki projektech“) přiřadit kontext a sémantický význam [14]. To umožňuje prohledávat a vyhodnocovat obsah článků s ohledem na jejich strojové zpracování.

Wikidata

Wikidata je součástí Wikimedia Foundation (a tedy rodiny projektů jako je Wikipedie) [15]. Od zbylých projektů se liší tím, že funguje i jako zdroj pro Wikipedii. Jedná se o databázi strukturovaných dat, která se snaží získávat z Wikipedie a dalších Wikimedia projektů. Ta jsou pak zpětně těmito projekty využívána a zdarma zveřejněna.

3 ZPRACOVÁNÍ TEXTU

Obecně lze text z pohledu programu rozdělit na dvě kategorie. První z nich je zpracovávání přirozeného jazyka (NLP), což je technika, která vyžaduje určitou formu porozumění významu a kontextu daného obsahu, navíc je nutné počítat s odlišnostmi jazyků a tak většinou nelze použít univerzální řešení. Druhou kategorií je strojově čitelný text, respektive text ve strojově čitelném formátu, který má jasně předdefinovaný význam. Takový obsah lze zpracovat pomocí jednodušších technik, případně lze kombinovat s některými metodami NLP, pokud se nelze spolehnout na to, že daný formát bude skutečně dodržen (typicky člověkem psaný text).

3.1 Zpracování Wikipedie

Významnou část většiny stránek na Wikipedii tvoří právě text v přirozeném jazyce. Pro jednoduchou analýzu ale není potřeba zkoumat kontext a význam. Je možné využít přirozené součásti webových stránek – tj. hypertextových odkazů, společně s předpokladem, že na významné stránky povede velký počet zpětných odkazů. Každý odkaz ale není nutně příbuzný se současným tématem – text může obsahovat i odkaz na naprosto obecný výraz. Je tedy nutné těmto zdrojům přikládat menší váhu, případně využít jiná hodnotící kritéria.

Infoboxy jsou, vzhledem k jejich struktuře, strojově poměrně dobře čitelné. Wikipedie definuje jejich formát, který je ale pro každý typ stránky (osobnost, událost, místo apod.) jiný. Lze ale využít faktu, že informace a odkazy v infoboxech jsou tvořeny lidmi a jsou nepoměrně kratší, než hlavní text. Je tedy možné předpokládat, že odkazované stránky jsou s velkou pravděpodobností relevantní k danému tématu.

Jako další významné kritérium lze použít *tranzitivitu odkazů* – význam odkazu ze stránky A na stránku B můžeme zdůraznit, pokud existuje stránka C, na kterou je odkázáno z A a která odkazuje na stránku B.

3.1.1 Formát článků Wikipedie

Veškerý obsah Wikipedie je uložen ve speciálním formátu nazvaném *wikitext* (někdy wikimarkup). Díky definici tohoto jazyka lze poměrně efektivně získávat potřebné informace – odkazy a infoboxy. Ostatní části obsahu lze zanedbat, vzhledem k tomu, že žádná forma NLP nebude na textu prováděna.

Odkazy jsou zapsány v jednoduchém formátu `[[Název stránky]]`, čímž vznikne odkaz na stránku ve tvaru `<a href="/wiki/nazev_stranky">Název stránky</a>` při zachování jazykové verze. Odkazy mohou mít i složitější tvar, a to `[[Název stránky|Libovolný text]]`, což vytvoří odkaz na stejnou adresu, ale text při zobrazení odkazu bude

„Libovolný text“. Jak je tedy vidět, odkazy jsou zpracovatelné bez problému jak ve formátu wikitext, tak v HTML.

Syntaxe pro infoboxy je, vzhledem k jejich struktuře, komplikovanější. Existuje několik různých typů doporučených dle typu článku. Pro anglickou verzi Wikipedie existuje těchto typů infoboxů několik desítek. Obecná struktura infoboxu vypadá následovně:

```
{{Infobox typ
 | title      = Nazev stranky
 | subheader  = Podtitulek infoboxu
 ...
}}
```

Část za znakem =, tedy hodnota, samozřejmě nemusí obsahovat jen text, ale i odkazy, tabulky, obrázky a podobné struktury, což strojové zpracování infoboxů činí obtížnější. Část před tímto znakem je klíč hodnoty. Vzhledem k tomu, že existují předepsané hodnoty pro určité typy infoboxů[1], je možné této vlastnosti využít při zpracování – typicky například osoby spolehlivě budou obsahovat datum narození, které bude k nalezení vždy pod stejným klíčem. Tento fakt zpracování infoboxů významně ulehčuje a zpřesňuje odhad významu dat.

Kvůli vysokému počtu typů infoboxů je velice obtížné vytvořit nástroj tak, aby byl schopen je zpracovat. Je ale možné využít informací o počtech článků, které dané infoboxy využívají (tato informace je součástí stránky se seznamem šablon pro každý typ) a pokusit se maximálně zpracovat ty, které mají nejvyšší zastoupení.

[1] Dostupné z `https://en.wikipedia.org/wiki/Wikipedia:List_of_infoboxes`.

4 NÁSTROJE PRO PRÁCI S ČASOVOU OSOU

4.1 Knihovna časové osy

Knihovna *cz.kajda.timeline* je projekt realizovaný jako součást diplomové práce [16]. Tato knihovna dokáže zobrazovat vložená data na časové ose jako různé typy historických událostí a zobrazit vzájemný vztah mezi nimi (taktéž definovaný jako součást vstupu). Osa je rozdělena do několika horizontálních pruhů, z nichž každý obsahuje pouze určité typy událostí. Pokud se navíc udály ve stejném období, jsou zobrazeny souběžně bez toho, aby se překrývaly. Nástroj také umožňuje uživateli přibližovat či oddalovat požadovaná historická období – je tak možno zobrazit si přehled v rámci tisíciletí a postupným přibližováním se dostat až na rozlišení řádu minut. Díky této vlastnosti je ale možné zobrazit poměrně velký časový rozsah (např. právě tisíciletí), což může při vyšším počtu zobrazovaných událostí vést ke zhoršení přehlednosti a vypovídací hodnoty osy. Při překročení této meze tak dokáže knihovna vhodným způsobem vybrat důležité události a ty méně podstatné dočasně skrýt, což probíhá na základě ohodnocení událostí. Toto ohodnocení je dodáno společně s daty a o jeho výpočet se stará grafová databáze [17].

4.1.1 Formát vstupu

Nástroj dokáže zpracovávat vstup ve formátu JSON. Příloha A.1 obsahuje příklad a ukazuje možnosti tohoto způsobu definice grafu [16] [18].

- Položka *nodes* obsahuje definice jednotlivých událostí (uzlů grafu).
  - *id* – Každá událost má své (unikátní) celočíselné ID.
  - *stereotype* – Typ události, která se může rovnat jedné z hodnot *person, event, place, item*.
  - *name* – Název události či jméno osobnosti.
  - *description* – Volitelný slovní popis.
  - *begin, end* – Počátek a (volitelně) ukončení události ve formátu dle normy ISO 8601 [19].
  - *properties* – Další volitelné vlastnosti uzlu.
    - *imageSrc* – Adresa obrázku, který se má zobrazit společně s událostí.
    - *startPrecision, endPrecision* – Přesnost zobrazení počátku a ukončení události. Umožňuje definovat, s jakou přeností má být daný údaj reprezentován. Může nabývat jedné z následujících hodnot (nezávisle na sobě): *century, decade, year, month, day, none*.
- Položka *edges* určuje vztahy mezi událostmi (respektive hrany mezi uzly grafu).
  - *id* – Unikátní celočíselné ID vztahu.

- *sterepotype* – Typ vztahu mezi událostmi. Může nabývat jedné z následujících hodnot: *relationship, interaction, participation, creation, cause, part_of, takes_place*.
- *from, to* – ID událostí, mezi kterými je vztah definován.
- *name* – Slovní popis vztahu.
- *properties* – Volitelné vlastnosti vztahu.

4.2 Grafová databáze

Grafová databáze (volitelně) využívaná společně s časovou osou je výsledkem další diplomové práce [18]. Při použití samotné osy je nutné jí poskytnout data v daném formátu (viz výše). Nástroj, vytvořený v rámci této práce, především ale umožňuje uživateli načíst již existující databázi (nebo případně vytvořit novou) a v rámci ní pak přidávat, odebírat či různě opravovat vrcholy (tedy události) a hrany (vztahy mezi nimi). Toto je umožněno díky grafickému uživatelskému rozhraní (GUI), vytvořenému, stejně jako celý nástroj, v programovacím jazyce Java. Program umí při definici uzlu zadat všechny výše zmíněné údaje (pracuje se stejnými entitami), umí také navíc definovat nové vlastnosti a značky.

4.2.1 Implementace grafové databáze

Samotná databáze, s níž nástroj na pozadí manipuluje, využívá grafovou databázi Neo4j [20]. Typově se tato databáze velice liší od (dnes nejpoužívanějších) databází relačních. Ty lze v krajním případě využít jako určitou náhradu grafových databází, avšak pro reprezentaci dat založených na grafech se nehodí (každá hrana se tak stane cizím klíčem tabulky a jakýkoliv dotaz na druhý vrchol této hrany znamená užití operace JOIN[1], jejíž volání je poměrně drahé a náročné [21]). Naproti tomu, grafové databáze jsou na podobné operace přímo stavěny a optimalizovány. Další výhodou je, že lze k hranám připojit určitou informaci a jednotlivé hrany tak od sebe významově odlišit.

Mezi Neo4j databází na pozadí a GUI, s nímž komunikuje uživatel, je vytvořeno API. To slouží jako mezivrstva pro komunikaci mezi těmito částmi a navíc umožňuje ostatním nástrojům manipulovat s databází.

[1] Sloučení 2 a více tabulek

5 NÁVRH NÁSTROJE

Všechny dostupné open–source nástroje, analyzované v předchozích kapitolách, se jeví pro implementaci potřebných funkcí jako naprosto nevhodné. Kromě nedostupnosti a neaktuálnosti dumpu (interval mezi aktualizacemi může být, v případě, že nástroj používá svojí verzi dumpu, i přes rok) je největší překážkou pravděpodobně obtížná úprava již existujícího, obvykle velmi rozsáhlého kódu.

Nakonec byl tedy pro vytvoření nástroje zvolen jako platforma webový prohlížeč Google Chrome, a to díky jeho podpoře doplňků a také s ohledem na fakt, že je v současnosti (prosinec 2015) s podílem přes 50 % (přesná čísla se liší dle zdroje a způsobu měření) nejpoužívanějším prohlížečem [22] [23]. Díky použité technologii (viz dále) je tvorba doplňků poměrně jednoduchá a je navíc možné s minimálními úpravami přenést rozšíření i do jiných prohlížečů (již existující podpora prohlížeče Opera, v budoucnosti Mozilla Firefox a Microsoft Edge). Rozšíření bude pracovat přímo s webovými stránkami v kombinaci s poskytovaným API.

5.1 Tvorba rozšíření pro Google Chrome

Rozšíření jsou vlastně webové stránky a tomu odpovídají i používané technologie – tedy HTML a CSS pro tvorbu rozhraní (případně manipulaci se stránkou) a JavaScript. Chrome poskytuje rozšířením API, díky čemuž mohou interagovat se stránkami – měnit jejich obsah, zobrazovat okna s vlastním obsahem a podobně. Instalovaná rozšíření mají (volitelně) svou ikonu vedle adresního řádku, čímž je uživateli umožněno jej ovládat. Instalace probíhá z Internetového obchodu Chrome[1]. Pro vývojové účely (nebo pokud nechceme rozšíření poskytnout veřejnosti) je možné doplňky instalovat také z disku a to pomocí volby *Načíst rozbalené rozšíření* na adrese `chrome://extensions`[2] nebo přetažením instalačního souboru s koncovkou `crx`.

5.1.1 Manifest

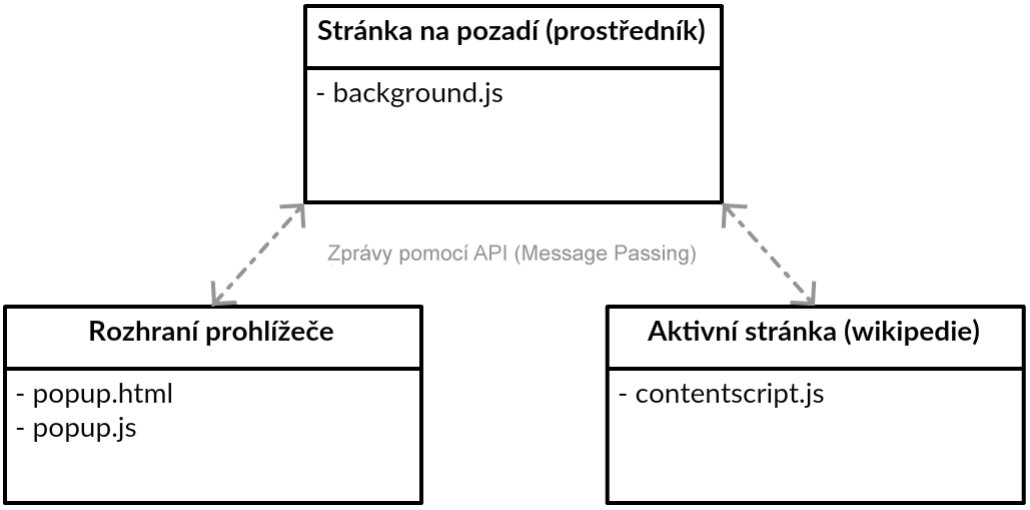
Manifest, tedy soubor s názvem manifest.json, je soubor ve formátu JSON, pomocí něhož lze k rozšíření připojit určité dodatečné informace. Obsahuje například zobrazované jméno a popis doplňku, verzi nebo ikonu. Především v něm ale lze definovat, k čemu rozšíření požaduje přístup – standardně totiž není povolen přístup k internetu nebo k zobrazovaným stránkám. Můžeme uvést skripty, které mají být vloženy přímo do kódu stránky a s jejich pomocí s ní potom manipulovat a podobně.

[1] Dostupný z `https://chrome.google.com/webstore`.

[2] Adresy začínající `chrome://` jsou dostupné pouze z prohlížeče Google Chrome a představují standardní způsob jeho konfigurace.

5.1.2 Struktura rozšíření

Kvůli bezpečnosti není možné přistupovat přímo ke stránce. Je nutné komunikovat pomocí „prostředníka“, tedy skript běžící na pozadí. Pomocí něho je možné zasílat zprávy mezi různými částmi rozhraní a především manipulovat se stránkou či zpracovat data na základě obsahu stránky či uživatelovy interakci s ní. Komunikace probíhá pomocí API, které poskytuje Chrome, nazývaného *Message Passing* [24]. Strukturu jednoduchého rozšíření popisuje obrázek 5.1 (názvy souborů jsou volitelné).



Obr. 5.1: Struktura rozšíření pro Google Chrome

Rozhraní prohlížeče je část, se kterou komunikuje uživatel a která mu dává možnost provádět poskytované akce. Za předpokladu, že v Manifestu patřičně vyplníme položku *browser_action*, bude rozhraní moci pracovat nezávisle na právě aktivní stránce – v opačném případě (*page_action*) bude možné pracovat pouze s jedinou stránkou (nezávisle na ostatních). Díky browser_action se také bude zobrazovat ikona rozšíření vedle menu a adresního řádku – po aktivaci této ikony se zobrazí formou vyskakovacího okna soubor *popup.html*, v němž můžeme nadefinovat potřebné uživatelské rozhraní. O funkcionalitu se stará příslušný skript, v tomto případě *popup.js*. Pomocí Message Passing pak publikujeme akce a změny do zbylých částí rozšíření.

Jako mezičlánek slouží skript na pozadí – *background.js*. Ten se stará o šíření zpráv do skriptů provádějících dané akce. Díky unikátnosti napříč všemi panely prohlížeče je také vhodným místem k persistenci či dočasnému uložení právě zpracovávaných dat.

Posledním článkem jednoduchého rozšíření je skript *contentscript.js*. Ten je vkládán přímo do stránky a umožňuje přímou manipulaci s objektovým modelem stránky (DOM). Je možné využít klasických technologií známých z Javascriptu, například vyhledávání elementů podle atributů a podobně. Je také možnost připojit některé z užitečných knihoven – příkladem za všechny je jQuery. Skript může také pomocí zpráv (Message Passing)

komunikovat s ostatními částmi rozšíření. Vkládání tohoto skriptu je možné pomocí Manifestu omezit na stránky určité domény, subdomény nebo stránky splňující určité kritérium (na základě adresy) – typicky například **.server.com/** znamená, že skript má být vložen na všech stránkách webu server.com, včetně všech jeho subdomén. Kromě skriptů je také možné vkládat CSS soubory. Z bezpečnostních důvodů jsou však vkládané soubory spuštěny v omezeném, izolovaném prostředí. Mají tak přístup k DOM, avšak není jim umožněn přístup k JavaScriptovým souborům, které jsou spuštěny přímo stránkou, a k jejím funkcím. Podobné omezení samozřejmě platí i z druhé strany – původní skripty nemají možnost zjistit, že byly některé soubory přidány. To je, kromě již zmíněné bezpečnosti, výhodné zejména kvůli faktu, že kód skriptů nemůže kolidovat s jiným [25].

Často využívanou možností u obdobných doplňků je použití existující stránky jako součásti uživatelského rozhraní stránky. Pomocí výše popsaného skriptu je možné obsah stránky posunout či překrýt vlastní vrstvou a na ní zobrazit potřebné prvky rozhraní. To je výhodné zejména v případech, kdy se zobrazované informace přímo týkají obsahu stránky (jinak by bylo výhodnější si vytvořit nové okno či panel) a je tedy nutné zobrazovat obě položky najednou.

5.1.3 Omezení rozšíření

Rozšíření mají svá omezení a limity. Některá plynou z použité technologie (rozšíření jsou stále webové stránky a ne plnohodnotné desktopové aplikace), některá pak z bezpečnostních důvodů zavádí Chrome ve svém API.

Poměrně velkou překážkou, plynoucí z technologie, je jen omezená práce se soubory. V případě modulárních částí aplikace není možné dynamicky načíst všechny soubory složky, ale je potřeba soubory zaregistrovat v rozšíření a sdělit mi tak, které soubory má načíst. V případě, že by bylo možné vypsat obsah složky, by mohlo načítání modulů probíhat mnohem dynamičtěji, což by umožnilo rozšířit funkcionalitu aplikace.

5.2 Práce s API Wikipedie

Wikipedie poskytuje veřejné API, nejen pro práci s články, ale i ostatními částmi webu – uživateli, revizemi článků nebo obrázky. Díky tomu je možné pracovat (především) s obsahem bez toho, abychom jej museli zpracovávat jako HTML, ve kterém je přirozeně přístupný přes webové stránky (což je vzhledem k tomu, že ne všechny stránky mají stejné a korektní formátování, poměrně obtížné).

API je dostupné skrze klasické HTTP dotazy a odpovědi. Jednotlivé parametry zadáváme jako položky do GET[3] části adresy a následně odesíláme na webovou adresu

[3] Část adresy, v níž mohou být předány parametry ve formátu ?nazev=hodnota&nazev2=hodnota2...

https://en.wikipedia.org/w/api.php (pro ostatní jazyky a Wikimedia podstránky na stejné adrese, pouze s jinou doménou). Můžeme si však vybrat formát, ve kterém nám server bude odpovídat.

- *JSON* – Preferovaný způsob práce s API. Je přirozenou součástí JavaScriptu, ostatní populární jazyky pro něj mají velmi dobrou podporu ze strany knihoven. Tento formát je doporučený, použití dále zmíněných je sice podporované, avšak nedoporučené [26].
- *XML* – Formát, ze kterého vychází HTML. Na rozdíl od HTML parserů, které jsou k dodržování standardů značně benevolentní, je zpracovávání XML nekompromisní a standard je nutné dodržet (respektive je nutné dodržet příslušné schéma, např. XSD či DTD, což definuje podobu validního dokumentu). Je tak poměrně dobře strojově zpracovatelný.
- *PHP* – Výstup, ve kterém jazyk PHP ukládá své objekty. Je výhodný při práci v PHP – nemusí se tak nic zpracovávat a pomocí jediného příkazu je odpověď převedena do objektu. Pro ostatní jazyky nepoužitelný.

Dále je potřeba specifikovat akci. Nejpoužívanější a nejužitečnější je hodnota *query* [27].

Tím specifikujeme, odkud a v jakém formátu informace budeme čerpat. To můžeme učinit pomocí titulku (parametr titles), ID stránek (pageids) a podobně – zde už záleží na konkrétním druhu informace, kterou potřebujeme získat. Pro studijní a testovací účely je k dispozici sandbox – testovací sekce, sloužící k rychlému otestování základní funkčnosti [28].

5.3 Rozšíření JavaScriptu

I přesto, že JavaScript podporuje objektově orientované programování, neumožňuje definovat třídy (ve smyslu klíčového slova `class` z jiných objektově orientovaných jazyků)[4]. To ale není považováno za chybu či nedostatek, je to spíše vlastnost plynoucí z funkcionálního návrhu jazyku. V JavaScriptu jsou tak pro tento účel využívány právě funkce – třídou je někdy nazývána speciální „konstrukční funkce“. Je zde využíváno vlastnosti JavaScriptu, že vše, co není primitivní typ[5], je objektem – a tedy i funkce. A je tak možné vytvořit její instanci pomocí klíčového slova `new`, jak je vidět v ukázce 5.1.

Metody je do objektu možno přidávat několika způsoby. Na první pohled se jeví jako nejjednodušší možnost přiřadit funkci přímo do vlastnosti. Tento postup ale přináší několik problémů, z nichž nejvýznamnější je nemožnost přístupu k těmto metodám v případě

[4]Toto klíčové slovo (a tedy možnost definovat třídy) bylo přidáno ve standardu ECMAScript 6 [30] z roku 2015, jehož podpora ještě není napříč prohlížeči 100% [31].

[5]Primitivní typ jsou v JavaScriptu data, která nejsou objektem a nemají žádné metody. V současnosti jich existuje 6: `string`, `number`, `boolean`, `null`, `undefined` a `symbol`[29].

```
var MyClass = function(name) {
  this.name = name;
};

var created = new MyClass("Title");
```

Ukázka 5.1: Vytváření instance

dědění. V praxi se tedy používá definice třídních metod pomocí vlastnosti `prototype`. Tuto vlastnost má každá funkce, její výchozí hodnotou je prázdný objekt. Umožňuje volání rodičovských metod a často se také využívá vlastnosti, že po úpravě prototypu se tato úprava projeví i v rámci již existujících objektů. Příklad je vidět v ukázce 5.2.

```
MyClass.prototype = {
  getName: function() {
    return this.name;
  }
};

var name = created.getName();
```

Ukázka 5.2: Ukázka využití prototypu

Jako další požadavek na objektový návrh lze uvést dědičnost. K tomuto účelu je opět využíván prototyp. Dříve byly používány vlastní funkce, které ručně nastavily prototyp na instanci rodičovské funkce, v novějších verzích JavaScriptu (ECMAScript 5.1 a novější) je možné k tomuto účelu využít funkci `Object.create`.

Často zmiňovaným problémem v rámci objektového programování v JavaScriptu je klíčové slovo `this`. Typickým příkladem je použití anonymních metod v rámci objektů. Uvnitř anonymní metody totiž `this` odkazuje právě na tuto metodu a ne na rodičovský objekt. Častým řešením je proto následující konstrukce, kdy si programátor uvnitř metody objektu uloží hodnotu `this` do pomocné proměnné. Funkce `process` je tedy volána na rodičovském objektu, viz ukázka 5.3.

```
var that = this;
this.data.forEach(function(date) {
  that.process(date);
});
```

Ukázka 5.3: Nutnost ukládání proměnné this

Vzhledem k těmto vlastnostem JavaScriptu vznikla řada projektů, které se některé problematické části snaží vyřešit nebo přinejmenším ulehčit. Minimalizují riziko vzniku chyb a umožňují programátorovi snazší zápis kódu. V současné době jsou nejpoužívanější CoffeeScript a TypeScript. Oba jsou v podstatě samostatnými programovacími jazyky. Jejich

specifikem je, že jsou přeloženy („kompilovány“) do JavaScriptu. To umožňuje zavést konstrukce, které JavaScript přímo nepodporuje (typickým zástupcem je právě klíčové slovo `class`). Výhodou těchto jazyků je, že po zavedení nových vlastností do čistého JavaScriptu je možné kód pouze znovu přeložit a kompilátor tyto vlastnosti použije bez zásahu do kódu.

5.3.1 CoffeeScript

CoffeeScript přináší, kromě podpory pro třídy, celou řadu vylepšení [32]. Upravuje syntaxi již existujících vlastností a konstrukcí tak, aby jejich použití programátorovi maximálně ulehčil. Hned na první pohled je významným rozdílem fakt, že není nutné příkazy zakončovat středníkem. Složené závorky, v JavaScriptu používané pro oddělení bloků kódu, jsou zde nahrazeny odsazením mezerami či tabulátory. Další významnou odlišností je také volání funkcí – parametry nejsou ohraničeny závorkami, ale pouze odděleny mezerou.

Jak je vidět, oproti JavaScriptu je CoffeeScript poměrně benevolentní. Na druhou stranu je ale nutné se naučit řadu poměrně nestandarních jazykových konstrukcí, které zavádí, což může způsobovat začátečníkům problémy.

5.3.2 TypeScript

TypeScript je open source jazyk, jehož autorem je společnost Microsoft [33]. Na rozdíl od CoffeeScriptu, který syntaxi JavaScript výrazně pozměňuje, je TypeScript pouze jeho rozšířením. Důležitou vlastností tak je, že jakýkoliv validní kód v JavaScriptu je také validním TypeScriptem. To umožňuje transparentní používání již existujících knihoven. Pravděpodobně nejvýraznějším rozdílem je (volitelně) typová kontrola během kompilace. Díky tomu je možné zavést pokročilé konstrukce, které podporují zavedené typové jazyky (C#, Java...), například přetypování, dědičnost typů nebo generické typy. Umožňuje také definování rozhraní (`interface`), tříd (`class`) a modulů. Vzhledem k těmto vlastnostem a transparentímu používání JavaScriptu bude pro vývoj nástroje použit právě TypeScript.

Definice tříd

TypeScript podporuje definici tříd syntaxí podobnou například Javě, C# nebo standardu ECMAScript 6. Umožňuje definovat konstruktor, metody i třídní proměnné, včetně modifikátorů přístupu. Je však důležité si uvědomit, že všechny tyto vlastnosti – především typovost a modifikátory přístupu – jsou podporovány pouze v rámci TypeScriptu a v kódu přeložením do JavaScriptu lze již například k soukromým třídním proměnným bez problému přistupovat. Na zapouzdření se tedy po překladu není možné spolehnout.

Třídy v TypeScriptu jsou volitelné a jejich použití není, stejně jako v JavaScriptu, vyžadováno. Definice třídy je vidět na ukázce 5.4.

```
class MyClass {
    private name: string;

    constructor(name: string) {
        this.name = name;
    }

    getName(): string {
        return this.name;
    }
}
```

Ukázka 5.4: Definice třídy v TypeScriptu

Jak je vidět, lze definovat typy parametrů metod (v tomto případě konstruktoru), proměnných i návratových hodnot. Kód třídy je poté kompilátorem TypeScriptu přeložen na kód třídy podobný tomu, jaký byl uveden výše v příkladu definice pro JavaScript.

Dědičnost

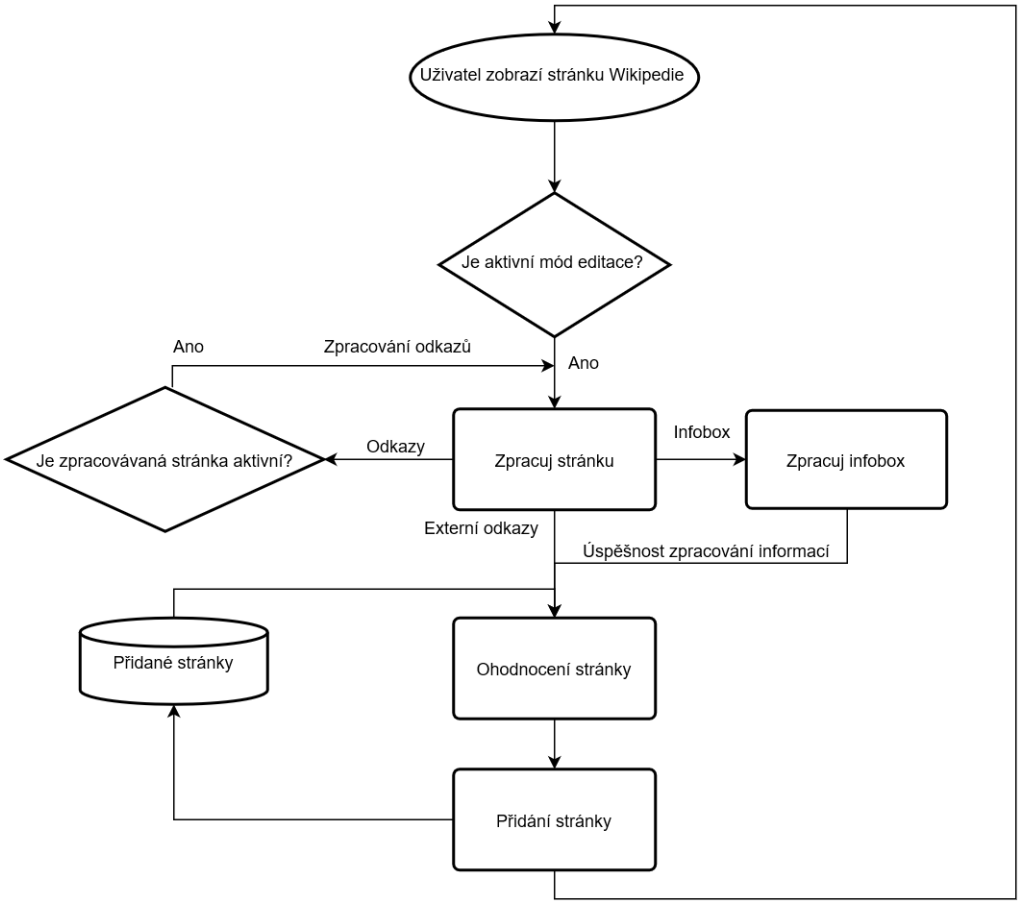
V TypeScriptu lze používat klíčové slovo `extends` tak, jak ho známe například z výše zmíněné Javy. Vzhledem k existenci rozhraní je možné také použít v rámci definice třídy `implements`, čímž deklarujeme implementaci daného rozhraní.

Moduly

Moduly v TypeScriptu jsou konstrukcí, jež ovlivňuje viditelnost struktur, a jsou ekvivalentem jmenných prostorů z jiných jazyků. Struktury (třídy, proměnné) definované uvnitř modulu nejsou viditelné mimo něj bez explicitního uvedení klíčovým slovem `export`. Na rozdíl od modifikátorů přístupu (`private`, public atd.) jsou definice modulů a exportovaných struktur přímo přeloženy do JavaScriptu.

5.4 Návrh zpracování

Funkcionalita nástroje je znázorněna na obrázku 5.2. Ve výchozím stavu nebude rozšíření vykonávat žádnou činnost – vzhledem k předpokládaným nárokům na zdroje (včetně připojení k internetu) je tato volba uživatelsky přívětivější a nebude zbytečně počítač zatěžovat. Po aktivaci *editačního módu* začně aplikace zpracovávat stránky – pro efektivnější zpracovávání lze využít možností API prohlížeče tak, že bude aktivováno pouze na stránkách, jejichž adresa odpovídá vzoru `http(s)://*.wikipedia.org/wiki/*`. K tomuto omezení dochází zcela automaticky prostřednictvím definice v `manifest.json`.



Obr. 5.2: Návrh „workflow“ uživatelské části rozšíření

Zpracování stránky je pravděpodobně funkčně nejobsáhlejší část aplikace. Musí zde dojít k několika akcím – prostřednictvím `content scriptu` bude do skriptu na pozadí odeslána informace o právě zobrazené stránce. Samotné zpracování musí proběhnout na pozadí, z důvodu persistence dat – především cachování článků[6]. Vzhledem k povaze výpočtů, jež je třeba vykonat, je bude pravděpodobně nutné spustit v rámci jiného vlákna nebo takovým způsobem (neblokujícím), aby neovlivňovaly plynulost uživatelského rozhraní a vykonávání jiných akcí.

Po stažení kompletního obsahu je možné z něj extrahovat odkazy článků, na něž daný článek odkazuje a které budou taktéž zpracovány. V textu se také nachází infobox, který je nutný k získání informací o článku.

Poslední získanou informací z API budou *zpětné odkazy*, respektive jejich počet. Z kombinace těchto informací a dalších faktorů bude poté přiřazeno zpracovávanému článku číselné ohodnocení, které následně bude zobrazeno v rámci uživatelského rozhraní (respektive přímo webové stránky) a články budou zvýrazněny dle jejich pravděpodobnosti.

[6]Ukládání již zpracovaných dat (náročných na výpočet či zdroje) tak, že při příštím použití budou rychle a levně k dispozici.

5.4.1 Zpracování infoboxů

Jak již bylo řečeno v kapitole 3.1.1, infoboxy nelze zpracovat univerzálním způsobem, ale je nutné se zaměřit na specifický typ infoboxu a získávat informace v rámci něj. Infoboxy jsou také lokalizované do různých jazyků. Bylo by tedy vhodné část nástroje starající se o zpracování infoboxů vyčlenit tak, aby nezasahovala přímo do kódu a bylo ji možno rozšiřovat o další jazyky a typy infoboxů.

5.5 Návrh metriky

Ze získaných informací uvedených výše je možné vypočítat přibližnou relevanci (ohodnocení) článku vůči článkům již přidaným. Tato metrika bude obsahovat několik složek s různými vahami. Systém výpočtu metrik bude taktéž navržen modulárně – každá metrika bude samostatným modulem a bude tedy možné je libovolně přidávat, odebírat či editovat bez ovlivnění ostatních. Metriky pravděpodobně nebudou závislé na jazyku či typu infoboxu.

Metrika článku bude číselná hodnota nabývající hodnoty $< 0; 1 >$, uživateli bude prezentována v procentech. Vzhledem k povaze metriky, kdy „odhadujeme“ uživatelovy preference a téma zájmu, je sice teoreticky možné dosáhnout 100% jistoty relevance, avšak v praxi se taková výše jistoty jeví jako jen velmi obtížně dosažitelná (uživatel může mít vždy na mysli mírně odlišné či konkrétnější téma, než nástroj odhadne). Je tedy vhodné jednotlivé moduly navrhnout tak, aby s touto mírou nejistoty počítaly a 100 % se svým ohodnocením blížily jen limitně nebo jich dosáhly jen ve velmi specifických případech.

Pro výpočet relevance článku a tedy platí následující vztah

$$r_{total}(a) = \sum_{i=0}^{n} r_i(a) \times w_i, \tag{5.1}$$

kde r_i jsou jednotlivé moduly metriky, w_i váhy těchto modulů a r_{total} je celkové ohodnocení. Hodnota celkového ohodnocení nesmí překročit hodnotu 1, proto je nutné zajistit, aby byla suma všech vah byla v součtu menší či rovna 1.

$$||w|| = \sum_{i=0}^{n} w_i \leq 1 \tag{5.2}$$

Každý modul metriky bude mít k dispozici kompletní získané údaje o daném článku (viz výše), zpracovaný infobox a již přidané články. Bude také definováno, zda modul potřebuje po změně stavu aplikace (typicky přidání či odebrání článku apod.) přepočítat svou hodnotu. Například počet externích článků (viz níže) se v krátkodobém horizontu prakticky nezmění, nebo jen velmi nevýznamně.

5.6 Uživatelské rozhraní

Vzhledem k prostřední, ve kterém je rozšíření spouštěno (rozhraní webového prohlížeče), je nutné zachovat jeho zvyklosti a doporučení. Protože bude rozšíření pracovat s aktivní webovou stránku, je možné maximálně využít i tento prostor – pomocí Chrome API je možné do stránky vkládat nové elementy a libovolně s nimi manipulovat.

Na uživatelské rozhraní (UI) jsou kladeny určité nároky. Část uživatelského rozhraní bude zobrazována nezávisle na stránkách, tj. bude společná pro všechny stránky. Bude jí ale umožněna funkčnost nezávisle na tom, zda jsou nějaké stránky zobrazeny či ne. V rámci této části bude umístěno tlačítko pro aktivaci či deaktivaci zpracovávání stránek. Součástí také bude seznam již přidaných článků – zde budou v seznamu či tabulce zobrazeny a bude umožněno jejich prohlížení (aktivní odkaz) a manipulace (mazání, v případě potřeby úprava).

Součástí této části rozhraní bude také tlačítko pro exportování vytvořeného grafu. Po aktivaci tlačítka bude zobrazen standardní dialog prohlížeče pro ukládání souborů. Bude zde také umístěno tlačítko aktivace vizualizace (viz dále). Vizualizace bude otevřena v novém okně prohlížeče (za využití API). Zobrazení vizualizace bude prosté, bez ovládacích prvků.

Je nutné uživateli poskytnout možnost přidávat články zobrazené („hlavní“) a odkazované (tj. odkazy uvnitř textu článku). V případě zobrazených článků je vhodné umístit informace (např. procentuální relevanci) a tlačítko pro přidání vedle nadpisu článku. Tím bude na tyto prvky kladena důležitost. Zároveň bude (vizuálně) očividné, že se prvky vztahují právě k zobrazenému článku.

Odkazy na články podobným způsobem vyřešit nelze – vzhledem k jejich počtu by přímá editace textu článku a zobrazení příslušných informací vedle odkazů způsobila problémy se zobrazením a výrazně by tím klesla přehlednost. Proto je vhodné tyto prvky zobrazit až při aktivaci odkazu, tedy pokud se nad ním nachází kurzor myši. V tomto nově zobrazeném panelu by byla možnost článek přidat (podobným tlačítkem jako u hlavního článku), zobrazit si základní zpracované informace a jeho relevanci (procentuálně). Odkazy na články by bylo vhodné obarvovat či jinak zvýraznit podle jejich relevance (výsledku metriky). Tím by měly relevantní články být zvýrazněny na úkor článků nepříbuzných či obecných. Toto zvýrazňování ale nesmí zhoršit čitelnost textu např. snížením kontrastu.

Po přidání článku by měl být zobrazen panel se seznamem doporučených článků – tedy takových, které byly doposud zpracovány a mají nejvyšší získané ohodnocení. Z tohoto seznamu bude možné články otevírat (odkazem) a také přímo přidávat.

I když Chrome umožňuje vytvářet stránku s nastavením[7], součástí nástroje nastavení nebude a konfigurace bude probíhat pomocí souboru `Config.ts` při překladu rozšíření.

[7] Popsáno na `https://developer.chrome.com/extensions/options`.

5.6.1 Návrh rozhraní

Obrázek 5.3 zobrazuje návrh uživatelského rozhraní. Jedná se o *wireframe*[8], jeho cílem není navrhnout rozhraní naprosto detailně (o to se případně stará až grafický návrh), ale naznačit rozvržení jednotlivých komponent.

Obr. 5.3: Návrh uživatelského rozhraní, tzv. wireframe

Obrázek znázorňuje prostředí webového prohlížeče se zobrazenou stránkou Wikipedie. Obsahuje několik bodů:

1. Tlačítko, které je do Chromu automaticky přidáno po načtení rozšíření. Po kliknutí levým tlačítkem myši je otevřena nabídka rozšíření, obsahující další ovládací prvky.
2. Tlačítko, které po aktivaci spustí zpracovávání stránek a aktuální stránku obnoví. Jelikož je tlačítko na první pohled jediným indikátorem aktivního rozšíření, musí být jasným způsobem rozlišen stav „Vypnuto“ a „Zapnuto“.
3. První element z prvků uživatelského rozhraní rozšíření, který je součástí stránky. Toto tlačítko bude vloženo v blízkosti nadpisu a po jeho použití bude článek přidán. Pokud už se článek v poli přidaných článků již nachází, má toto tlačítko popisek „Odebrat“ a slouží ke smazání článku.
4. Ohodnocení článku dle metriky. V případě přidaného článku jsou tato procenta skryta.

[8] Občas překládáno jako *drátěný model*

5. Zobrazený panel po najetí myší na (zvýrazněný) odkaz. Nadpis bude zobrazovat skutečný titulek článku (tj. ne jen obsah daného odkazu). Kromě něj bude součástí zobrazeného panelu tlačítko s identickou funkcionalitou, jako u předešlého. V případě článků, kde byly úspěšně získány informace z infoboxu, budou zde některé z nich zobrazeny. Pokud článek, na nějž článek odkazuje, ještě není zpracovaný, panel se nezobrazí.
6. Po přidání článku se zobrazí panel s navrženými články. Jeho účel byl popsán výše. Bude rovněž součástí stránky a po jeho zviditelnění bude obsah adekvátně zmenšen (zúžen).
7. Infobox zobrazené stránky. Obsahuje zpracovávané informace a odkazy na ostatní články.
8. Seznam přidaných článků. Články zde budou přehledně zobrazeny v tabulce či seznamu.

6 IMPLEMENTACE ROZŠÍŘENÍ

Výsledná aplikace odpovídá provedené analýze. Důraz byl především kladen na rozšiřitelnost. Moduly, u nichž se předpokládá, že budou upravovány (tj. parsery infoboxů a moduly metrik), byly navrženy tak, aby autorovi případných rozšíření bylo umožněno jednoduché zavedení nově vytvořených modulů do systému.

6.1 Architektura

V příloze A.2 se nachází objektový diagram UML, který znázorňuje návrh programu. Diagram je záměrně zjednodušený, některé vazby jsou vynechány tak, aby byl návrh přehlednější (např. ke třídě `Article`, kterou využívá většina aplikace). Uvedené metody a vlastnosti jsou nejdůležitější z hlediska návrhu (jsou ale uvedeny všechny veřejné metody, čímž může UML sloužit jako popis API), nepodstatné uvedené nejsou.

Jak je vidět, prostřednictvím Chrome API je aplikace rozdělena na 3 součásti, jak již bylo uvedeno v teoretické části. Z hlediska návrhu je nejjednodušší část `Popup`, která se stará o zobrazování a obsluhu hlavní nabídky. Část `ContentScript` je vkládána do každé stránky a zprostředkovává reprezentaci získaných dat ve stránce, případně zobrazuje prvky uživatelského rozhraní, které do stránky mají být vloženy. `Background` je skript na pozadí, kde jsou všechna data zpracována. Jak je na UML diagramu vidět, tento skript není k API připojen přímo, ale přes `BackgroundListener` a `BackgroundProxy`. Třída `BackgroundListener` umožňuje zaregistrovat se k odposlechu událostí určitého typu a při vyvolání této události je zaregistrovaná metoda automaticky spuštěna. Jako prostředník mezi těmito třídami slouží třída `BackgroundProxy`. Typy událostí jsou definovány ve třídě `Actions`, jejíž konstanty jsou využívány jako identifikátor událostí napříč aplikací.

6.2 Spuštění

Po načtení rozšíření do prohlížeče je rozšíření neaktivní, čímž šetří systémové prostředky tak, aby uživatelův prohlížeč zbytečně nezatěžovalo. Pro aktivaci zpracovávání stránek je nutné aktivovat režim editace prostřednictvím tlačítka v *popupu* (viz kapitola 5.6), čímž dojde (na stránkách Wikipedie) k vložení skriptu `contentscript.js` přímo do stránky. Je zkontrolována podpora jazyka, který je zjištěn z adresy. Podporované jazyky jsou definovány v souboru `config.ts`) a je nutné, aby pro podporovaný jazyk existovala definice infoboxů v adresáři `data`.

Po kontrole je adresa stránky odeslána do skriptu na pozadí – `Background.ts`. Tím je započato zpracování aktuální stránky.

6.3 Komunikace s API

Prostřednictvím API je aktuální stránka stažena. Z jejího obsahu je extrahován infobox a také odkazy. Ty jsou následně zpracovávány stejným způsobem, jako zobrazená stránka – s rozdílem, že jejich odkazy již dále zpracovávány nejsou. Všechny třídy starající se o komunikaci s Wikipedií a jejím API se nachází v adresáři `Api`, respektive modulu `Api`. Všechny části komunikující s API jsou pouze implementací rozhraní tak, aby mohly být v případě potřeby nahrazeny.

Části aplikace, které komunikují s API Wikipedie, nepokládají dotazy přímo, ale využívají dotazovou frontu. Vzhledem k omezení Wikipedie je potřeba, aby dotazy na její API probíhaly sériově – tzn. dotaz může být odeslán až po tom, co je obdržena odpověď (či chyba) z předchozího. V případě paralelního zasílání dotazů může dojít k přetěžování serveru a eventuálnímu zablokování přístupu [34]. V případě porušení tohoto pravidla (či dalších limitů, např. počtu článků na dotaz apod.) Wikipedie nejprve odpovídá chybou či prázdnou odpovědí – což se stalo i při vývoji aplikace. K zablokování je tedy přistoupeno až při vícenásobném, dlouhodobém porušování podmínek.

Dotazovou frontu, která toto chování zajišťuje, zastává třída `Processing.Default-Processor` (která se také stará o základní zpracování – viz dále kapitola 6.4). V této frontě (poli) jsou uloženy informace o článcích, co mají být načteny (konkrétněji jejich URL, titulek a jazyk). Důležité je, že společně s nimi je uložena i informace o tom, z které stránky přišel požadavek na jejich zpracování – v případě přechodu na jinou stránku či její opuštění jsou příslušné požadavky zrušeny a síť tak není zbytečně blokována. V případě, že by tato informace chyběla, rozšíření by neumožňovalo zpracovávání několika stránek naráz.

Z API jsou získávány 2 druhy dat:

Obsah článků Pro získání odkazů, které článek obsahuje, a především infoboxu je nutné stáhnout kompletní obsah článku ve formátu `wikitext`. Díky tomu, že v případě tohoto dotazu Wikipedie umožňuje získat až 50 článků najednou, je tato část velice efektivní (z hlediska odezvy). Na druhou stranu stahuje poměrně velké množství dat (při plném využití 50 článků řádově jednotky MB – záleží samozřejmě na délce dílčích článků).

Počet externích odkazů Pro potřeby metriky je získávána informace o počtu zpětných odkazů na jednotlivé články. Zde je Wikipedie slabým článkem, jelikož neumožňuje dotázat se pouze na počet odkazů samotných, ale jen získat jejich seznam. Tento seznam je navíc omezen shora maximálně 500 odkazy na dotaz, což by při získávání kompletního počtu odkazů činilo tuto část úzkým hrdlem celé aplikace[1]. Proto je z důvodu zvýšení rychlosti zpracování tento počet dotazů omezen na 2, tj. na tisíc

[1] Například anglická verze článku o 2. světové válce má v současné době zhruba 150 000 zpětných odkazů, tudíž získání odkazů jen tohoto článku by trvalo $(150000 \div 500) \times p = 300p$, kde p je průměrná odezva.

odkazů. Tento limit je možné pozměnit úpravou hodnoty v souboru `Config.ts`. Změnou hodnoty je dosaženo změny přesnosti příslušné metriky, ale za cenu velkého dopadu na výkon aplikace.

V rámci třídy `Background` je zavedena cache, která je prohledávána před jakýmkoliv dotazem na API. Dotaz na každý článek je tedy položen pouze jednou.

6.4 Zpracování článků

Je evidentní, že zpracovávání článků je pevně provázáno s předchozí částí – přesto, že se jedná o stejné třídy, jsou tyto části funkčně poměrně odlišné. Po stažení potřebných dat je vytvořena entita představující článek, což je rovněž prováděno v třídě `Processing.DefaultProcessor`, a dojde ke zpracování odkazů článku a také infoboxu (viz dále). Vzhledem k velkým nárokům této části na výkon způsobovalo toto zpracování v prvních verzích rozšíření vysoké zatížení procesoru, což mělo negativní dopad na plynulost uživatelského rozhraní. Při zpracovávání informací o článcích je tedy velmi kladen důraz na složitost algoritmu. Především ale došlo k přesunu výkonné části do jiného vlákna na pozadí, díky čemuž jakákoliv činnost tohoto skriptu nemůže ovlivnit uživatelské rozhraní a jeho odezvu. Ve vlákně je spuštěna instance třídy `ProcessingWorker`.

Web Workers

JavaScript nepodporuje tradiční pojetí vláken (standardně představováno třídou `Thread` apod.), ale společně s HTML5 (avšak v odlišném standardu) byl zaveden koncept `Web Workers`. Ten umožňuje spouštění kódu JavaScriptu na pozadí tak, aby neovlivňoval uživatelské rozhraní, tedy skripty běžící v hlavním vlákně webu [35].

Web Workers umožňují načíst soubor a jeho kód spustit na pozadí při jeho vytvoření.

```
1  var worker = new Worker("script.js");
```

S vláknem lze komunikovat prostřednictvím zpráv (událostí), pomocí nichž lze vláknu zprávy zasílat či je od něj přijímat. Zde se nachází drobná překážka – při předání objektu zprávou je objekt pouze zkopírován, tj. není zachována jeho instance. Pokud se ale jedná o objekt specifického typu, např. entitu článku, dojde k jeho přetypování na standardní objekt JavaScriptu (`Object`). Přestože jsou zachována všechna data, tento objekt již nemá žádné metody původního objektu. Je tedy nutné ručně vytvořit objekt nový a data do něj zkopírovat. Toto chování je, pochopitelně, prováděno z důvodu zachování vláknové bezpečnosti (`thread-safe`), může ale způsobit komplikace. Web Workers rovněž, na rozdíl od standardního pojetí vláken, nesdílí (globální) proměnné s hlavním vláknem a není tedy možný přístup např. k objektu `window` zobrazené stránky.

6.5 Zpracování infoboxů

Infoboxy je potřeba vyextrahovat z kompletního textu. To lze poměrně spolehlivě provést následujícím postupem:

1. Nalézt začátek infoboxu. Úkoly podobného typu se nejlépe řeší pomocí regulárních výrazů. Zde lze využít faktu, že všechny infoboxy začínají sekvencí `{{Infobox`. Bohužel, vzhledem k tomu, že i tento text je zadáván ručně autorem článku, není možné se spolehnout na přesný tvar této sekvence. Regulární výraz tak musí počítat s existencí velkých i malých písmen, případně bílých znaků (mezera, tab). Odpovídající regulární výraz je tedy následovný: `/{{(\s*)infobox/gi`. Symbol `\s` zastupuje libovolný bílý znak, `gi` jsou pak modifikátory – text odpovídající výrazu je tedy hledán v celém řetězci nezávisle na velikosti písma. Infobox se také v textu vůbec nacházet nemusí.
2. Získat typ infoboxu. Ten je prakticky vždy umístěn na stejném řádku jako výše nalezená počáteční sekvence, pro jeho získání tedy stačí pouze načíst zbytek příslušného řádku. Validním typem infoboxu je i případ, kdy je zbytek řádku prázdný – jedná se o obecný infobox bez konkrétního typu.
3. Nalézt konec infoboxu. To bohužel nelze provést prostým nalezením posloupnosti znaků `}}`, protože se v infoboxu mohou vyskytovat další boxy a konstrukce, které jsou uvozeny a ukončeny stejnými znaky. Pro úlohy podobného typu lze tedy využít následující algoritmus:
   (a) S prvním znakem `{`, který jsme nalezli v prvním bodě algoritmu, nastavíme kontrolní proměnnou na hodnotu 0.
   (b) Procházíme od počáteční značky text znak po znaku. V případě, že je znak rovný `{`, zvýšíme hodnotu kontrolní proměnné o 1. V případě `}` hodnotu kontrolní proměnné snížíme.
   (c) Takto text prohledáváme do doby, než bude mít kontrolní proměnná znovu hodnotu 0, čímž jsme našli konec infoboxu.

Tím jsme získali kompletní infobox a můžeme tak pokračovat ve zpracování jeho obsahu. Ten je, jak už bylo popsáno v kapitole 3.1.1, ve formátu `klíč = hodnota`, přičemž tato dvojice je navíc uvozena znakem `|`. Cílem zpracování je uložit hodnoty do asociativního pole pod příslušným klíčem tak, aby mohly být efektivně zpracovávány.

Vzhledem k tomu, že může být hodnota několikařádková a může obsahovat i formátovaný text, je výhodné provést celé zpracování ve dvou krocích. Prvním krokem je projít každou řádku infoboxu a zjistit, zda obsahuje znak `=`. Pokud ano, zkopírujeme řádku na nový řádek vytvořeného řetězce. Pokud ne, pouze daný řádek k vytvořenému řetězci připojíme (k posledně vytvořenému řádku). Tímto postupem dosáhneme toho, že po zpracování celého infoboxu dostaneme ve vytvořeném řetězci infobox ve tvaru `klíč = hodnota` tak, že je na každém řádku tato dvojice právě jednou.

Tím můžeme vytvořit z tohoto zpracovaného řetězce asociativní pole pod příslušným klíčem a toto pole poskytnout modulům starajícím se o zpracování infoboxů (tzv. parserům).

6.5.1 Parsery infoboxů

Parsery jsou závislé na jazyce, protože klíče a typ infoboxu je lokalizované. Z toho důvodu jsou umístěny v adresáři `data` a dále v adresáři pro danou jazykovou verzi – např. adresář `data/en` pro anglický jazyk. Je nutné, aby parsery a definice pro libovolný jazyk byly umístěny ve složce podle příslušného klíče jazyka tak, jak jej definuje Wikipedie [1].

Definice typů infoboxů pro jazyk je umístěna v souboru `InfoboxTypes.json`. Tento soubor obsahuje především seznam typů infoboxů pro daný jazyk společně s entitami, na které budou namapovány. Kromě toho obsahuje také klíč `definitions`, což je seznam souborů obsahujících parsery. Bez této deklarace nebude při zpracování článků příslušného jazyka neuvedený parser použit ani načten.

Parser infoboxu je třída, která dědí od `Processing.InfoboxDefinition`. Při zpracování článku je seznam těchto parserů postupně procházen a na každém z nich je zavolána metoda `matches`. Té jsou předány informace o článku a uvnitř této metody je ověřeno, zda daný parser „rozumí“ infoboxu příslušného typu – v takovém případě vrací návratovou hodnotu `true`. Díky výchozí implementaci této metody můžou nejjednodušší parsery přepsat pouze třídní proměnnou `matchingRegex` regulárním výrazem odpovídajícího typu a není nutné definovat metodu `matches`.

V případě, že je ověření úspěšné, je zavolána metoda `process` (kterou už každý parser musí implementovat povinně sám). Této metodě jsou předána všechna data o infoboxu a pouze na ní je, jaké informace z něj dokáže zpracovat. Zde je možné využít pomocné metody rodičovské třídy, především pro práci s daty.

6.5.2 Vytvoření nového parseru

Při vytváření nového parseru (či přímo jazyka) je vhodné využít již existujících příkladů a některý parser zkopírovat (v případě vytváření jazyka rovnou celou složku, např. `en`). Pokud se parser dokáže rozhodnout o své způsobilosti zpracovávat informace jen na základě hlavičky infoboxu (tj. úvodní sekvence a typu), stačí přepsat pouze proměnnou `matchingRegex`, jak bylo uvedeno výše. V opačném případě je nutné vytvořit vlastní implementaci třídy `matches`. V metodě `process` pak můžeme přistoupit k samotnému zpracování.

Parser je rovněž nutné následně zaregistrovat v souboru `InfoboxTypes.json`. Je důležité si uvědomit, že parsery jsou vyhodnocovány v uvedeném pořadí a v případě shody je procházení ukončeno. Parsery nejsou překládány se zbytkem aplikace, ale je nutné je přeložit do JavaScriptu ručně (viz kapitola 6.10).

6.5.3 Provazování článků

Články a jejich infoboxy jsou zpracovány už při samotném načtení. Zde ale nastává problém: pokud se některému z parserů podaří v infoboxu získat informace, respektive rozpoznat vztah s jiným článkem, nemusí být tento článek k dispozici (zatím nebyl načten a zpracován). Proto byl do aplikace zaveden koncept, který tento problém řeší pomocí třídy `Processing.LinkPromise`. Tato třída umožní odkázat na jiný článek (pouze pomocí jeho názvu, nemusí tedy v době zpracování existovat) a tomuto odkazu přiřadí kontext, tedy vztah s právě zpracovaným článkem. Při zpracování (typicky exportování) grafu jsou poté tyto odkazy procházeny a články jsou pomocí nich provazovány vztahem příslušného typu.

6.6 Ohodnocení článků

Každý článek je po vytvoření a zpracování infoboxu ohodnocen. Implementace metriky odpovídá návrhu v analytické části. Moduly, jež zastávají jednotlivé části metriky, jsou umístěny v adresáři `Processing.Rating`. Vzhledem k tomu, že tyto metriky nejsou závislé na jazykové verzi, nemusejí být umístěny v rámci jazykového modulu (avšak mohou být – aplikace jejich umístění nevynucuje).

Modul metriky musí implementovat rozhraní `Processing.Rating.Rating`. Toto rozhraní definuje několik vlastností:

rate(article, added[]) Metoda, která slouží k samotnému ohodnocení článku `article`. Měla by vracet číslo (a neměla by jej nastavovat přímo článku), a to v intervalu $< 0; 1 >$. Pro hodnocení může metoda využít i druhý argument, který je do funkce předán – pole článků, které již byly uživatelem přidány a článek `article` by k nim tedy měl být určitým způsobem relevantní.

needsRecalculation Vlastnost, která naznačuje, zda je nutné, aby byla hodnota metriky přepočítána při změně stavu aplikace, tj. při změně pole přidaných článků (pole `added[]`).

uid Unikátní identifikátor metriky (vlastnost). Kromě celkového ohodnocení článku je také udržováno pole s hodnotami jednotlivých metrik a tento unikátní identifikátor je použit jako klíč tohoto asociativního pole. To je využíváno při přepočtu metriky – pokud je vlastnost `needsRecalculation` nastavena na `false`, je použita uložená hodnota z tohoto pole, v opačném případě je znovu použita metoda `rate`.

Metrika, která má být použita, je definována v souboru `Config.ts`, konkrétně konstantou `DEFAULT_RATER`. Výchozí hodnotou je zde `BaseRating`, což je implementace zajišťující všechny uvedené vlastnosti metrik. Hodnotou jejího konstruktoru je pole metrik, které mají být použity pro výpočet celkové metriky, a příslušných priorit – číselných hodnot, kterými mají být výsledky jednotlivých metrik vynásobeny.

Tato metrika se zavoláním její metody `rate` projde všechny metriky předané v konstruktoru, zavolá jejich metodu `rate` a výsledek vynásobí její prioritou. Protože je metrika `BaseRating` nadřazenou ostatním, ukládá metriku přímo do entity článku. Do pole `ratings` ukládá výsledky jednotlivých metrik pod jejich `uid` způsobem uvedeným výše. Do vlastnosti `rating` výslednou číselnou hodnotu (výchozí hodnota je -1) a do `ratingPercentage` výslednou hodnotu v textové podobě, včetně znaku %, zaokrouhlenou na určitý počet desetinných míst (výchozí řetězec `N/A`, tj. nedostupné). Toto zaokrouhlení lze ovlivnit další konstantou souboru `Config.ts` – `RATING_ROUNDING`.

6.6.1 Vytvoření nové metriky

Každá metrika musí implementovat výše zmíněné rozhraní, tj. metodu `rate` a 2 vlastnosti. Soubor s její třídou může být umístěn v jakémkoliv adresáři, je ale vhodné využít výchozí umístění, aby byl její kód překládán společně se zbytkem aplikace. Pokud je využívána jako rodičovský modul metrika `BaseRating`, je pro její funkčnost nutné přidat definici do pole konstruktoru v souboru `Config.ts` a podle uvážení zvolit váhu (prioritu) dané metriky, společně s příslušným upravením ostatních hodnot. Je nutné, aby suma těchto vah nepřesahovala hodnotu 1.

Pro funkčnost v prohlížeči je také nutné přidat cestu k přeloženému souboru (tj. k souboru `.js`, nikoliv zdrojovému `.ts`) do klíče `background.scripts` v manifestu rozšíření (viz kapitola 5.1.1). Tím dojde k zavedení JavaScriptového souboru do prohlížeče.[2]

6.7 Doporučené články

Panel s doporučenými články se objeví po každém přidání článku. Panel obsahuje deset článků s nejlepším ohodnocením (počet lze upravit v souboru `Config.ts`), které byly zatím zpracovány. Doporučené články lze pomocí tlačítka na panelu obnovit a články je možné procházet či rovnou přidávat přímo z panelu.

6.8 Exportování grafu

Po zpracování popsaném v předešlých kapitolách může uživatel tyto články přidat do svého výběru. Pro znázornění svého výběru může uživatel využít tlačítko *Visualisation* v hlavní nabídce rozšíření. Pro vizualizaci je využívána knihovna *d3.js* a její modul *bihisankey* (viz dále). Dojde k otevření nového okna s vizualizací jednotlivých uzlů (článků). Hrany mezi nimi představují odkazy mezi danými články, tloušťka těchto hran určuje pak počet těchto

[2]Při definici nového parseru infoboxů není nutné cestu ke skriptu v manifestu uvádět, protože parsery jsou načítány odlišným způsobem.

odkazů. Vizualizaci je možné využít jako znázornění vztahů mezi články, tématických okruhů nebo ji lze využít k odhalení nerelevantních článků.

Když je uživatel se svým výběrem spokojen, může přistoupit k exportování přidaných článků ve formě grafu pro časovou osu. Pro tuto akci slouží tlačítko *Export* v nabídce, po jehož aktivaci je zobrazen standardní dialog pro soubor ke stažení a graf je tak vyexportován do souboru. Toto tlačítko je aktivní pouze v případě, že jsou některé články přidány.

Všechny soubory využívané ke stahování se nachází ve jmenném prostoru `Export`. Hlavní třídou je `Download`, která se stará o samotné stažení souboru. Jako parametr v konstruktoru přijímá výchozí název stahovaného souboru a především instanci rozhraní `Formatter`. Toto rozhraní má jedinou metodu `format`, které lze předat seznam článků, které mají být uloženy. Takové abstraktní řešení bylo zvoleno záměrně – při změně formátu exportovaného souboru nebo úpravě způsobu zpracování lze pouze zaměnit jednu třídu `Formatter` za jinou.

Výchozí implementací je třída `JsFormatter`, která dokáže články naformátovat podle požadavku časové osy. Využívá dvě další třídy – `ArticleFormatter` pro články (události osy) a `EdgeFormatter` pro jejich propojení. Při ukládání článků jsou pouze přemapovány vlastnosti objektu tak, aby odpovídaly formátu časové osy, případně jsou nastaveny výchozí hodnoty, které osa vyžaduje (začátek nebo konec události je při neúspěšném odhadu nastaven na současný datum). Provazování článků je už komplexnější, jelikož jsou procházeny všechny objekty `LinkPromise` (viz kapitola 6.5.3) a pomocí nich jsou články provázány.

6.9 Importování grafu

Články reprezentované grafem, které jsou z nástroje vyexportovány, je možné zpětně naimportovat pomocí tlačítka *Import*. Toto tlačítko je aktivní, pouze pokud je vypnutý mód editace. Po kliknutí na toto tlačítko je zobrazen standardní dialog načítání souboru a je možné vybrat dříve exportovaný soubor. Po potvrzení dialogu jsou články postupně naimportovány, přičemž jsou zachovány případné úpravy data počátku a konce události a její typ, pokud byly před exportováním provedeny.

Pro načítání je využit skript, který vytvoří v hlavní nabídce skrytý formulář pro soubor a simuluje kliknutí na tlačítko formuláře. Protože je manipulováno přímo se zdrojovým kódem stránky, je nutné, aby byl takový skript spuštěn v hlavní nabídce, nikoliv na pozadí (`Background`). Hlavní třídou starající se o vyvolání tohoto dialogu je `Upload`, která je součástí jmenného prostoru `Import`. Zde se také nachází rozhraní `Import` a jeho implementace `JsLoader`, což jsou součásti starající se o interpretaci načteného obsahu a jeho převedení na informace o článcích, které mají být zpracovány.

6.10 Překlad zdrojových kódů a spuštění

Pro stažení veškerých závislostí je nutná instalace balíčkovacího nástroje Node Package Manager (NPM). Pomocí tohoto nástroje je možné příkazem `npm install` nainstalovat všechny závislosti zcela automaticky. Jmenovitě pak pro vývoj nutný `TypeScript` a balíček `Typings`, který zajišťuje stažení definic typů pro TypeScript (jeho konfigurace se nachází v souboru `typings.json`. Po instalaci všech závislostí je možné přeložit TypeScript do JavaScriptu, se kterým Chrome pracuje. To lze provést příkazem `tsc` – je využit kompilátor TypeScriptu, jež je automaticky stažen nástrojem NPM. Po přeložení lze načíst celou složku (což je vhodné pouze pro vývoj) volbou *Načíst rozbalené rozšíření* nebo rozšíření zabalit pomocí volby *Zabalit rozšíření* na adrese `chrome://extensions`. Tím vznikne soubor přípony `crx`, který je možné do stejné stránky přesunout a tím dojde k instalaci. Tento soubor je možné samostatně publikovat.

Eventuálně lze pak přeložit parsery pro jednotlivé jazyky, a to spuštěním příkazu `tsc` v adresáři `data/<jazyk>`.

6.11 Použité knihovny

Všechny použité knihovny, jež nejsou součástí zdrojového kódu, lze nainstalovat pomocí NPM.

6.11.1 TypeScript, Typings

Za knihovny lze pouvažovat i programovací jazyk, ve kterém je rozšíření vytvořeno – TypeScript a balíček Typings.

6.11.2 jQuery

Tato knihovna se dá považovat za nepsaný standard ve světě JavaScriptu. Nanízí nepřeberné množství funkcí a možností. Často se používá pro manipulaci s DOM, kde nabízí příhodné zkratky. Dnes už poměrně velkou část funkcí této knihovny dokáže zastat novější verze JavaScriptu spolu s HTML5, avšak jQuery se stará o kompatibilitu a jednotné rozhraní napříč prohlížeči [36].

6.11.3 Bootstrap

Bootstrap je HTML, CSS a JavaScriptový framework [37]. Umožňuje rychlý návrh uživatelského rozhraní, které je ale zároveň vizuálně uspokojivé a tak je v řadě projektů použit jako základ pro design. K tomu poskytuje mnoho komponent, které lze v projektu použít.

V rozšíření jsou v zásadě použity 2 verze tohoto frameworku – plná instalace, která je stažena pomocí NPM a „redukovaná verze“, která obsahuje ručně vybrané části a komponenty. Plná verze je použita v ovládacím panelu, především pro vzhled. Vytváří také přehlednou pruhovanou tabulku, která je použita pro zobrazení seznamu přidaných článků. Knihovna také obsahuje ikony, které jsou použity v tlačítkách a odkazech, což působí přehledněji a zvyšuje použitelnost rozhraní [38].

6.11.4 qTip2

JavaScriptová knihovna, která je pluginem pro knihovnu jQuery. Umožňuje vytváření „bublin“ připojených k HTML elementům. Má široké možnosti konfigurace, včetně podpory pro lazy-loading, která je využívána v případě zobrazování panelů nad zpracovanými odkazy.

6.11.5 d3.js

Knihovna umožňující vizualizaci dat a manipulaci s nimi [40]. Není ale monolitickým nástrojem, umožňuje spíše knihovnu využívat pro definici vlastních komponent a pluginů. Pro vizualizaci spojitosti přidaných článků je použit plugin využívající knihovnu s názvem *bihisankey* [41].

6.11.6 Implementace algoritmu MD5

JavaScript neobsahuje implementaci hashovacího algoritmu MD5, který je potřeba při vytváření adresy obrázku, proto je používána knihovna přidávající podporu [42] [43]. Jelikož je v textu uveden obrázek pouze pomocí názvu souboru, jeho celou adresu je nutné vypočítat. K základní adrese `https://upload.wikimedia.org/wikipedia/commons/` je připojen první znak MD5 hashe názvu souboru, za lomítkem pak první 2 znaky a za dalším lomítkem název souboru. Někdy je slovo `commons` v adrese nahrazeno jazykovou verzí (např. `en`). Způsob detekce ale není nikde veřejně popsán a tento fakt je důvodem, proč se u některých článků načtení obrázku nezdaří.

7 TESTOVÁNÍ IMPLEMENTACE

Při testování implementace je potřeba počítat s omezeným počtem definic infoboxů a zvolit pro testování takovou tématiku článků, aby bylo možné relevantně posoudit správnost výsledků. Jinak řečeno, za chybu je nutné považovat nesprávné či chybějící zpracování, ale nikoliv chybějící informace díky nepodporované tématice. Články byly samozřejmě zpracovány v angličtině, nicméně jejich názvy jsou zde pro přehlednost uvedeny v češtině. Testována byla dvě kritéria – úspěšnost odhadu příbuzných témat (tedy relevance dalších článků) a úspěšnost získávání informací. Testování relevance bylo prováděno na menším vzorku dat tak, abychom mohli posoudit ohodnocení konkrétního okruhu článků. Jako téma testování byla zvolena česká panovnická linie Lucemburků.

Při testování uměl nástroj rozpoznávat následující informace:

Obrázky Hlavní obrázek infoboxu (nemusí být uveden).

Osoby Obecné informace, které lze zpracovat z infoboxů všech osob bez ohledu na jejich konkrétní typ.

- Datum narození.
- Datum úmrtí.
- Místo narození.
- Místo úmrtí.
- Manželé, manželky.
- Děti.

Místa Města, vesnice či obecně místa, usedlosti.

- Datum založení.
- Hlavní představitel – typicky starosta, primátor.

Vojenský konflikt Války či bitvy.

- Velitelé, vůdci. Může jich být v rámci konfliktu více. V tom případě dokáže nástroj rozpoznat jejich spojenectví.
- Datum počátku konfliktu.
- Datum konce konfliktu.
- Nadřazený konflikt – typicky bitvy jsou součástí některé války.

Šlechta Panovníci, obecně šlechta (anglicky *royalty*).

- Rod, ke kterému daný šlechtic patří.
- Následovník. Nástroj úmyslně nerozeznává předchůdce, protože by mezi předchůdcem a následovníkem existovaly vždy 2 vztahy.

Šlechtický rod Šlechtická linie.

- Datum počátku rodu.
- Datum vymření rodu.

Pro odhad relevance byly využity následující moduly (včetně jejich priorit):

Rozpoznané vztahy Byl rozpoznán vztah s některým z přidaných článků (30 %).

Odkaz z infoboxu Na článek vede odkaz z infoboxu z přidaných článků (25 %).

Externí odkazy Počet externích odkazů na článek (20 %).

Je událost Byl rozpoznán počátek a konec události (10 %).

Odkaz z článku Na článek vede odkaz z textu některého z přidaných článků (10 %).

7.1 Odhad relevance

Odhady relevance témat byly testovány pouze na samotné linii panovníků a jejich následovníků. Jako první byl přidán článek o rodu samotném a poté bylo pokračováno v pořadí Jan Lucemburský, Karel IV., Václav IV. a konečně Zikmund. Před každým přidáním byl zaznamenán odhad nástroje pro daný článek.

Článek	Lucemburkové	Jan Lucemburský	Karel IV.	Václav IV.	Zikmund
Ohodnocení	N/A	30 %	95 %	76 %	89 %

Tab. 7.1: Odhad relevance článků

Jak je v tabulce 7.1 vidět, odhad relevance je v tomto konkrétním případě poměrně spolehlivý a přináší uspokojivé výsledky. Před přidáním prvního článku je relevance *N/A* (Not Available, nedostupné), jelikož nástroj nemá možnost článek porovnat s jiným. Po přidání prvního článku vyroste relevance prvního z panovníků na *30 %*. Toto ohodnocení je nízké zejména z důvodu zmínění tohoto panovníka pouze v textu samotném, v infoboxu například uveden není. Po jeho přidání rapidně vzroste relevance Karla IV., z důvodu velkého provázání s Janem Lucemburským – je jeho synem a nástupcem, což článek vezme v úvahu. Pro další panovníky jsou výsledky rovněž uspokojivé.

Samotné hodnoty bez relativního srovnání s ostatními odkazy každé stránky jsou samozřejmě bezpředmětné. Na obrázku 7.1 je vidět část článku o Václavu IV. ve stavu před přidáním posledního testovaného článku. Obsahuje několik odkazů, z nichž jeden směřuje právě na požadovaný článek o Zikmundovi. Jak je na obrázku vidět, je vizuálně velice dobře odlišený od ostatních článků, jejichž relevance se pohybuje v průměru okolo 30 %.

7.2 Rozpoznávání údajů

Pro úspěšnost získávání údajů a rozpoznávání vztahů byl zvolen širší okruh článků. Kromě rodu a panovnické linie byly přidány všechny manželky, a pokud byla uvedena, tak i místa narození a úmrtí. Úspěšnost získaných údajů byla posuzována zvlášť.

King of Bohemia [edit]

During his long reign, Wenceslaus' grip on power was tenuous at best, as he came into repeated conflicts with the Bohemian nobility led by the House of Rosenberg. On two occasions he was even imprisoned for lengthy spells by rebellious nobles.

But Wenceslaus' greatest liability proved to be his own family. Charles IV had divided his holdings among his sons and other relatives. Although Wenceslaus upon his father's death retained Bohemia, his younger half-brother Sigismund inherited Brandenburg, while John received the newly established Duchy of Görlitz in Upper Lusatia. The March of Moravia was divided between his cousins Jobst and Procopius, and his uncle Wenceslaus I was made Duke of Luxembourg. Hence the young king was left without the resources his father had enjoyed. In 1386, Sigismund became king of Hungary and became involved in affairs further east.

Wenceslaus also faced serious opposition from the Bohemian nobles and even from his chancellor, the Prague archbishop Jan of Jenštejn. In a conflict surrounding the investiture of the abbot of Kladruby, the torture and murder of the archbishop's vicar-general John of Nepomuk by royal officials in 1393 sparked a noble rebellion. In 1394 Wenceslaus' cousin Jobst of Moravia was named regent, while Wenceslaus was arrested at Králův Dvůr. King Sigismund of Hungary arranged a truce in 1396, and for his efforts was recognized as Wenceslaus' heir.

In the Papal Schism, Wenceslaus had supported Pope Urban VI. As Bohemian king he sought to protect the religious reformer Jan Hus and his followers against the demands of the Roman Catholic Church for their suppression as heretics. This caused many Germans to withdraw from the University of Prague, and set up their own university at Leipzig. Hus was executed in Konstanz in 1415, and the rest of Wenceslaus's reign in Bohemia featured precursors of the Hussite Wars that would follow his death during the Defenestrations of Prague.

Obr. 7.1: Zvýraznění relevance článků

Kritérium	Počet rozpoznání	Úspěšnost
Manželky	10/10	100 %
Následovník	3/3	100 %
Datum narození (či počátek)	12/19	63 %
Datum úmrtí (či konec)	10/15	67 %
Místo narození, úmrtí	4/6	67 %
Obrázek	9/11	82 %
Stereotyp	12/19	63 %
Průměr		77 %

Tab. 7.2: Úspěšnost získaných údajů

Jak je v tabulce 7.2 možné vidět, na testovacím vzorku dat se celková úspěšnost rozpoznání dat pohybuje průměrně na hranici 77 %. 100% úspěšnosti dosahuje v případě rozpoznávání vztahů mezi ostatními osobami – to je způsobeno neměnnou syntaxí těchto parametrů. Naopak nejnižší úspěšnost dosáhla data narození (či obecně ukončení události) společně se stereotypem, protože v případě těchto hodnot není striktně vyžadováno dodržování syntaxe. Z tohoto důvodu je možné v hlavní nabídce editovat právě tyto hodnoty. Vygenerovaný graf zobrazený v časové ose je možné vidět v příloze A.3, na obrázku je zobrazen detail Karla IV.

8 ZÁVĚR

Na základě provedené analýzy byl pro řešení problému použit systém rozšíření webového prohlížeče Google Chrome, který se pro tento typ úlohy jevil jako nejvhodnější. Pro implementaci nástroje byl zvolen programovací jazyk TypeScript, který programátorovi poskytuje pohodlnější prostředí než JavaScript, jazyk běžně používaný v prohlížeči Google Chrome. Tento jazyk umožňuje používat konstrukce běžné v jiných programovacích jazycích, jako je Java, ale na druhou stranu není jeho použití vyžadováno a je stále možné využít JavaScript. V případě pokračování ve vývoji nástroje tak může příští autor zvolit mezi těmito dvěma jazyky dle jeho preference.

Z hlediska použitelnosti a dostupnosti byla jako zdroj dat vybrána online verze Wikipedie, díky čemuž může nástroj pracovat pouze v rámci prohlížeče bez nutnosti instalovat jakýkoliv další software. Data jsou také vždy aktuální a díky existujícímu API je možné dynamicky získávat informace, které prosté zobrazení obsahu článku neposkytne. Tyto dodatečné informace byly využity také při tvorbě systému ohodnocení, který je vytvořen takovým způsobem, aby jeho jednotlivé složky bylo možné měnit, přidávat či upravovat, případně měnit jejich prioritu.

Možnost dalšího pokračování v práci je především v oblasti modulů pro zpracování infoboxů. Nebylo účelem práce ani záměrem pokrýt velké množství typů článků, ale spíše poskytnout jednoduchý prostředek pro jejich tvorbu. Toho bylo docíleno právě modulární koncepcí, čímž je umožněno jednoduché přidání dalších typů bez nutnosti zasáhnout do zbytku aplikace. Rovněž je stejným způsobem možné přidávat další jazyky a není tedy nutné se omezit na články v angličtině, na kterou byla práce zaměřena. Na menším vzorku předpřipravených modulů byly ukázány jejich možnosti.

Vytvořený nástroj v současném stavu splňuje podmínky a nároky kladené v zadání a v úvodu. Cílem bylo vytvořit prostředek pro zpracovávání historických událostí tak, aby mohly být rychle a jednoduše uvedeny do kontextu s ostatními prostřednictvím časové osy – což bylo dle testování splněno a poskytuje uspokojivé a očekávatelné výsledky. Testování rovněž ukázalo, že v případě článků, jejichž zpracování je v modulech infoboxů podporováno, dokáže spolehlivě rozeznat důležitá data a spojení mezi jednotlivými články. V některých případech rozeznávání sice selže, což ale není nutně chyba nástroje, ale spíše vlastnost Wikipedie plynoucí z její podstaty – původu obsahu. Řešením pro tato selhání by bylo zavést při zpracovávání některé techniky z oblasti NLP, případně striktnější dodržování syntaxe na straně Wikipedie.

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

NLP Natural Language Processing – zpracování textu v přirozeném jazyce

API Application Programming Interface – programovací rozhraní programu

GUI Graphical User Interface – grafické rozhraní

DOM Document Object Model – objektový model dokumentu

NPM Node Package Manager – balíčkovací nástroj pro JavaScript

UML Unified Modeling Language – jazyk pro vizualizaci diagramů návrhu programu či systému

LITERATURA

[1] List of Wikipedias. Meta-Wiki [online]. San Francisco: Wikimedia Foundation, 2001-, 2015-09-27 [cit. 2015-12-17]. Dostupné z: https://meta.wikimedia.org/wiki/List_of_Wikipedias

[2] GILES, Jim. Internet encyclopaedias go head to head. Nature. 2005, 438(7070): 900-901. DOI: 10.1038/438900a. ISSN 0028-0836. Dostupné také z: http://www.nature.com/doifinder/10.1038/438900a

[3] BROWN, Adam R. Wikipedia as a Data Source for Political Scientists: Accuracy and Completeness of Coverage. PS: Political Science. 2011, 44(02): 339-343. DOI: 10.1017/S1049096511000199. ISSN 1049-0965. Dostupné také z: http://www.journals.cambridge.org/abstract_S1049096511000199

[4] History flow: results. History flow [online]. New York: IBM, 2004, 204-01-01 [cit. 2015-12-17]. Dostupné z: https://www.research.ibm.com/visual/projects/history_flow/results.htm

[5] WikiTaxi [online]. WikiTaxi, 2015, 2015-06-26 [cit. 2015-12-17]. Dostupné z: http://www.wikitaxi.org/

[6] XOWA [online]. XOWA, 2015, 2015-12-07 [cit. 2015-12-17]. Dostupné z: http://gnosygnu.github.io/xowa/

[7] Kiwix [online]. Kiwix, 2015, 2015-08-17 [cit. 2015-12-17]. Dostupné z: http://www.kiwix.org/

[8] Wikiextractor. GitHub [online]. San Francisco, California: GitHub, 2015, 2015-12-17 [cit. 2015-12-17]. Dostupné z: https://github.com/attardi/wikiextractor

[9] Wikipedia.js. GitHub [online]. San Francisco, California: GitHub, 2015, 2015-12-17 [cit. 2015-12-17]. Dostupné z: https://github.com/kenshiro-o/wikipedia-js

[10] WtfWikipedia. GitHub [online]. San Francisco, California: GitHub, 2015, 2015-12-17 [cit. 2015-12-17]. Dostupné z: https://github.com/spencermountain/wtf_wikipedia

[11] DBpedia [online]. 2015 [cit. 2016-04-22]. Dostupné z: http://wiki.dbpedia.org

[12] Czech DBpedia: Česká DBpedia = sémantická verze české Wikipedie [online]. Praha: Vysoké škola ekonomická v Praze, 2015 [cit. 2016-04-22]. Dostupné z: http://cs.dbpedia.org/wiki

[13] BabelNet: The largest multilingual encyclopedic dictionary and semantic network [online]. Rome: Sapienza University of Rome, 2016 [cit. 2016-04-22]. Dostupné z: http://babelnet.org/

[14] Semantic MediaWiki [online]. 2016 [cit. 2016-04-22]. Dostupné z: https://www.semantic-mediawiki.org/wiki/Semantic_MediaWiki

[15] Wikidata [online]. San Francisco: The Wikimedia Foundation, 2015 [cit. 2016-04-22]. Dostupné z: https://www.wikidata.org/wiki/Wikidata:Main_Page

[16] KACEROVSKÝ, Michal. Vizuální reprezentace precedenčního grafu. Plzeň, 2015. Diplomová práce. Západočeská univerzita v Plzni.

[17] HRBÁČEK, David. Zpracování časových údajů pro jejich vizualizaci. Plzeň, 2015. Diplomová práce. Západočeská univerzita v Plzni.

[18] MERUNKO, David. Generování a vizualizace časové osy. Plzeň, 2014. Diplomová práce. Západočeská univerzita v Plzni.

[19] ISO 8601:2004. Data elements and interchange formats. 2004. Ženeva, Švýcarsko: ISO, 2004.

[20] Neo4j: The World's Leading Graph Database [online]. San Francisco, California: Neo4j, 2015, 2015-12-17 [cit. 2015-12-17]. Dostupné z: http://neo4j.com/

[21] ROBINSON, Ian, James WEBBER a Emil EIFREM. Graph databases. First edition. Sebastopol, CA: O'Reilly, 2013, xii, 208 pages. ISBN 1449356265.

[22] Browser Statistics. W3Schools Online [online]. Refsnes Data, 2015, 2015-12-17 [cit. 2015-12-17]. Dostupné z: http://www.w3schools.com/browsers/browsers_stats.asp

[23] StatCounter - GlobalStats [online]. 2015, 2015-12-17 [cit. 2015-12-17]. Dostupné z: http://gs.statcounter.com/

[24] Message Passing. Google Chrome Extensions [online]. Mountain View, USA: Google Inc., 2010, 2015-12-02 [cit. 2015-12-17]. Dostupné z: https://developer.chrome.com/extensions/messaging

[25] Content Scripts. Google Chrome Extensions [online]. Mountain View, USA: Google Inc., 2010, 2015-12-02 [cit. 2015-12-17]. Dostupné z: https://developer.chrome.com/extensions/content_scripts

[26] API:Data formats - MediaWiki. MediaWiki [online]. San Francisco: Wikimedia Foundation, 2015 [cit. 2016-04-22]. Dostupné z: https://www.mediawiki.org/wiki/API:Data_formats

[27] API:Main page. MediaWiki [online]. San Francisco: Wikimedia Foundation, 2001-, 2015-11-27 [cit. 2015-12-17]. Dostupné z: https://www.mediawiki.org/wiki/API:Main_page

[28] API sandbox. Wikipedia: the free encyclopedia [online]. San Francisco: Wikimedia Foundation, 2001-, 2015-07-19 [cit. 2015-12-17]. Dostupné z: https://en.wikipedia.org/wiki/Special:ApiSandbox

[29] Primitive - Glossary: MDN. Mozilla Developer Network [online]. Mountain View, California: Mozilla Foundation, 2016 [cit. 2016-04-22]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/Primitive

[30] Classes - JavaScript: MDN. Mozilla Development Network [online]. Mountain View, California: Mozilla Foundation, 2016 [cit. 2016-04-22]. Dostupné z: https://developer.mozilla.org/cs/docs/Web/JavaScript/Reference/Classes

[31] ECMAScript 6 compatibility table. Github.io [online]. 2016 [cit. 2016-04-22]. Dostupné z: http://kangax.github.io/compat-table/es6

[32] CoffeeScript [online]. 2015 [cit. 2016-04-22]. Dostupné z: http://coffeescript.org

[33] TypeScript: JavaScript that scales [online]. Redmont: Microsoft, 2016 [cit. 2016-04-22]. Dostupné z: http://www.typescriptlang.org

[34] API:Tutorial - MediaWiki. MediaWiki [online]. San Francisco: Wikimedia Foundation, 2016 [cit. 2016-04-22]. Dostupné z: https://www.mediawiki.org/wiki/API:Tutorial

[35] Web Workers API - Web APIs: MDN. Mozilla Developer Network [online]. Mountain View, California: Mozilla Foundation, 2016 [cit. 2016-04-22]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API

[36] You Might Not Need jQuery [online]. HubSpot, 2016 [cit. 2016-04-22]. Dostupné z: http://youmightnotneedjquery.com

[37] Bootstrap: The world's most popular mobile-first and responsive front-end framework. [online]. Bootstrap, 2015 [cit. 2016-04-22]. Dostupné z: http://getbootstrap.com

[38] BEDFORD, Aurora: Icon Usability. Nielsen Norman Group [online]. Fremont: Nielsen Norman Group, 2016 [cit. 2016-04-22]. Dostupné z: https://www.nngroup.com/articles/icon-usability

[39] QTip2: Pretty powerful tooltips [online]. Newcastle Upon Tyne, 2013 [cit. 2016-04-22]. Dostupné z: http://qtip2.com

[40] D3.js: Data-Driven Documents [online]. 2015 [cit. 2016-04-22]. Dostupné z: https://d3js.org

[41] Neilos/bihisankey: A d3 javascript library/plugin for drawing bi-directional hierarchical sankey diagrams. Github [online]. San Francisco, California, 2015 [cit. 2016-04-22]. Dostupné z: https://github.com/Neilos/bihisankey

[42] MD5 Algorithm. MyersDaily [online]. 2008 [cit. 2016-04-27]. Dostupné z: http://www.myersdaily.org/joseph/javascript/md5-text.html

[43] Commons:FAQ - Wikimedia Commons. MediaWiki [online]. San Francisco: Wikimedia Foundation, 2016 [cit. 2016-04-23]. Dostupné z: https://commons.wikimedia.org/wiki/Commons:FAQ#What_are_the_strangely_named_components_in_file_paths.3F

SEZNAM PŘÍLOH

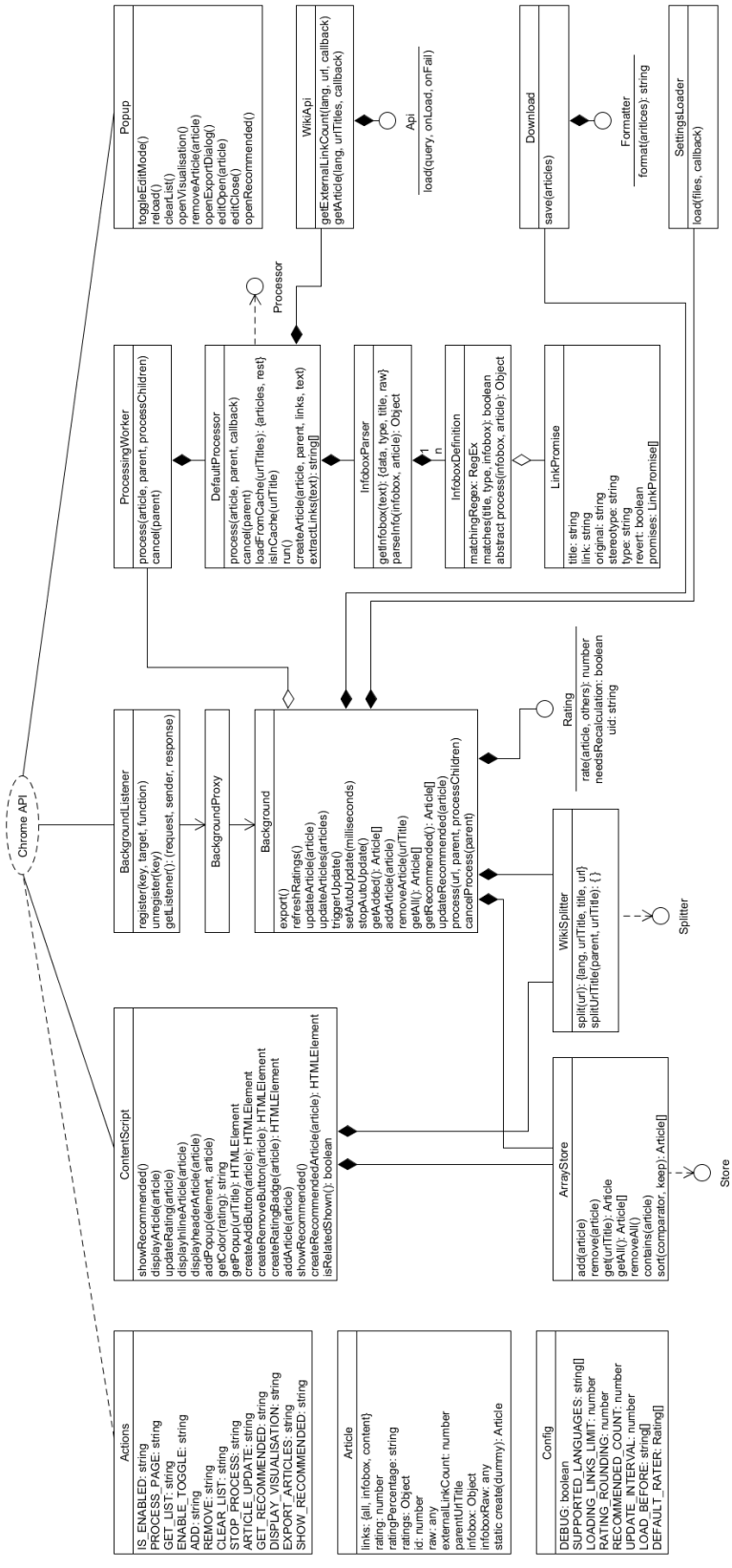
A PŘÍLOHY

A.1 Příklad vstupu pro časovou osu

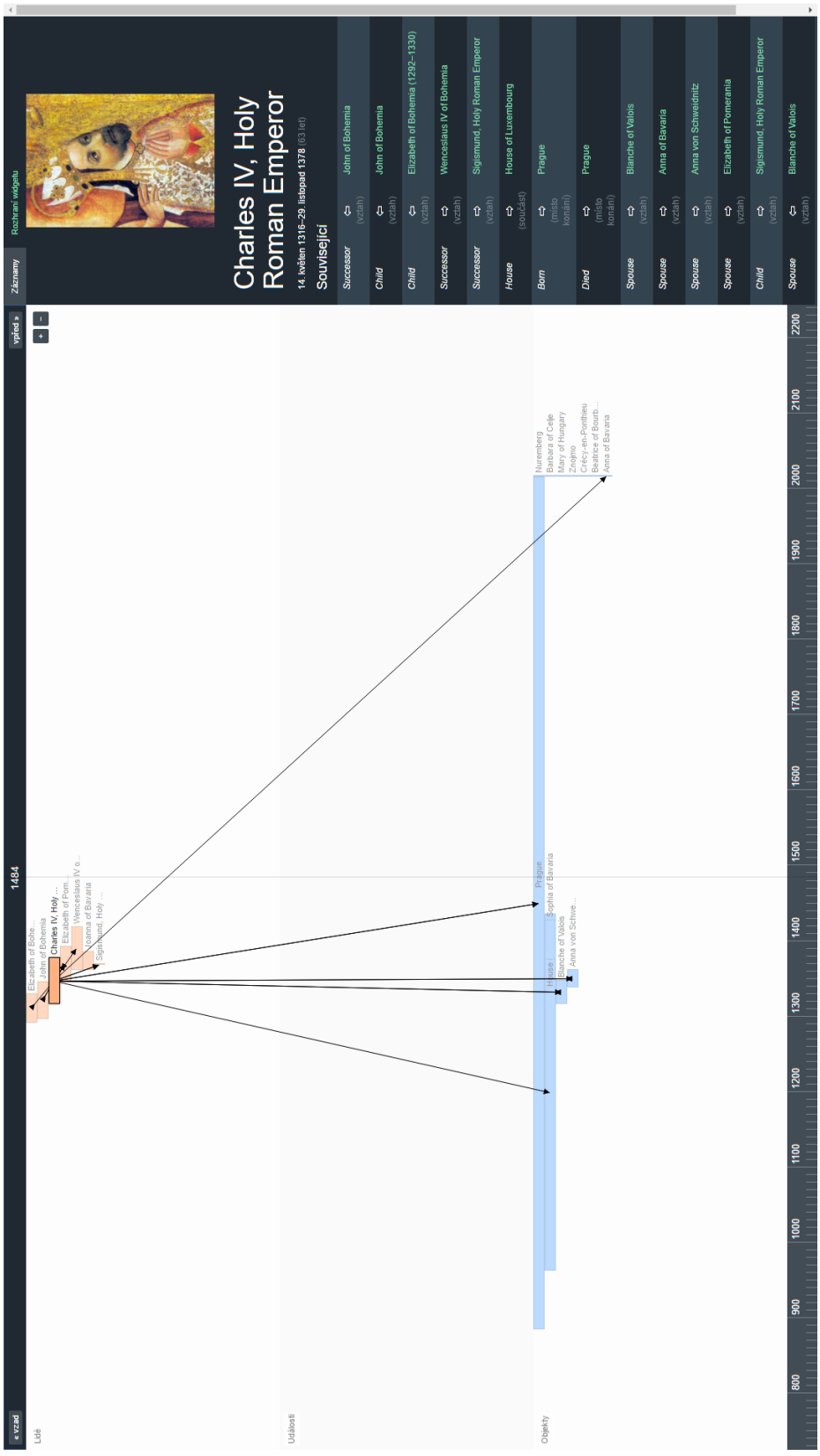
```
{ "list": { nodes": [
          {
             "id":1,
             "stereotype":"person",
             "name":"Samo",
             "description":"Fransky kupec, vladce
                slovanskeho kmenoveho svazu, tzv.
                Samovy rise. ",
             "begin":"0601-01-01T00:00:00",
             "end":"0659-12-31T23:59:59",
             "properties":{
                "imageSrc":"Samo.jpg",
                "startPrecision":"decade",
                "endPrecision":"year"
             }
          }, {
             "id":2,
             "stereotype":"item",
             "name":"Zlata bula sicilska",
             "description":"Soubor tri navzajem
                provazanych listin...",
             "begin":"1212-09-26T00:00:00",
             "properties":{
                "imageSrc":"Golden_Bull_of_Sicily.jpg",
                "startPrecision":"day"
             }
          }, {
             "id":3,
             "stereotype":"event",
             "name":"Vydana Zlata bula sicilska",
             "begin":"1212-09-26T00:00:00",
             "properties":{
                "startPrecision":"day"
             }
```

```
        }
    ], "edges":[
        {
            "id":1,
            "stereotype":"relationship",
            "from":3,
            "to":2,
            "name":"synovec"
        }, {
            "id":2,
            "stereotype":"interaction",
            "from":1,
            "to":3,
            "name":"synovec"
        }, {
            "id":3,
            "stereotype":"participation",
            "from":1,
            "to":3,
            "name":"potomek"
        }
    ] }}
```

A.2 Zjednodušený UML diagram



A.3 Vygenerovaný graf



A.4 Uživatelská dokumentace

WikiGraph je nástroj pro vytváření grafu z článků Wikipedie. Primární oblastí zpracování jsou historické události, které následně mohou být zobrazeny na časové ose (součástí diplomové práce *Vizuální reprezentace precedenčního grafu*, Kacerovský Michal, Plzeň 2015). *WikiGraph* umožňuje rychlou a efektivní přípravu dat pro tuto osu, čímž je možné uvést historické události do kontextu jiných, případně vizualizovat vzájemné souvislosti.

Nástroj je vytvořen formou rozšíření pro prohlížeč Google Chrome. Pracuje s živou verzí Wikipedie (na adrese `wikipedia.org`), pro využití nástroje je tedy nutný přístup k internetu.

A.4.1 Softwarové požadavky

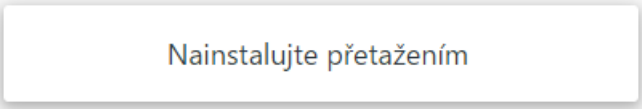
Pro instalaci rozšíření je nutné mít nainstalovaný prohlížeč Google Chrome minimálně verze 31, ideálně však aktuální.

A.4.2 Hardwarové požadavky

Pro plynulý chod rozšíření je doporučeno jej spouštět na počítači s minimálně 4 GB RAM a dvoujádrovým procesorem (Dual Core) s frekvencí 1,5 GHz a vyšší. Vzhledem k intenzivnímu využití internetového připojení není doporučeno rozšíření používat při připojení přes mobilní síť a další připojení limitovaná FUP limitem (tedy omezením množství stažených dat). Během činnosti rozšíření běžně dojde ke stažení několika jednotek až desítek MB v rámci kompletního zpracování jediné stránky. Je doporučeno využít připojení s rychlostí stahování (download) minimálně 10 Mb/s. Pro běh nástroje je velmi důležitá stabilní nízká odezva (latence), především na web Wikipedie (konkrétně domény dle jazyka). Tyto parametry jsou doporučené, nástroj bude plnohodnotně funkční i při nižších hodnotách, ale s odpovídajícím snížením výkonu.

A.4.3 Instalace

Nástroj je standardně dodáván ve formě souboru typu `crx`, což je balíček rozšíření pro Google Chrome. Pro instalaci je nutné otevřít nabídku `Menu - Další nástroje - Rozšíření` (či zadat adresu `chrome://extensions`). Zde se nachází seznam nainstalovaných rozšíření. Pro instalaci pomocí souboru `.crx` je nutné tento soubor pomocí myši vložit do stránky – při přesunu souboru do oblasti okna se objeví dialog, jak je vidět na obrázku A.1. Po zobrazení dialogu můžete tlačítko myši uvolnit. Nyní se, maximálně po pár sekundách, objeví okno s potvrzením požadavků nutných k běhu rozšíření. Pokud s požadavky souhlasíte, tento souhlas potvrďte a instalace je tím dokončena. Na hlavním panelu napravo od adresního řádku by se nyní mělo objevit tlačítko s logem Wikipedie.



Obr. A.1: Dialog pro instalaci rozšíření

A.4.4 Spuštění nástroje

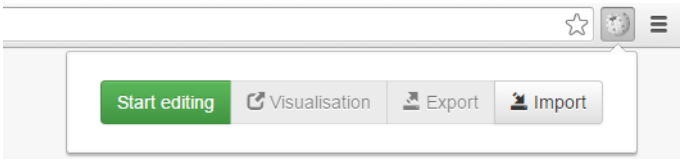
Na obrázku A.2 je vidět hlavní nabídka nástroje, která se objeví po kliknutí myší na logo rozšíření. Obsahuje tlačítka, jejichž funkcionalita je následující:

Start editing Zelené tlačítko spouští proces zpracovávání a aktivuje uživatelské rozhraní uvnitř stránek (viz dále). V případě, že je toto zpracování aktivní, je tlačítko červené s textem *Stop editing*.

Visualisation Tlačítko pro zapnutí vizualizace. V případě, že nejsou přidány žádné články (jako na obrázku), není tlačítko aktivní.

Export Tlačítko, které vyexportuje přidané články z nástroje ve formátu časové osy. Rovněž je neaktivní v případě, že je seznam přidaných článků prázdný.

Import Pro zpětné naimportování exportovaných článků slouží toto tlačítko. Je aktivní pouze v případě, že je vypnutý proces zpracovávání (viz první bod).



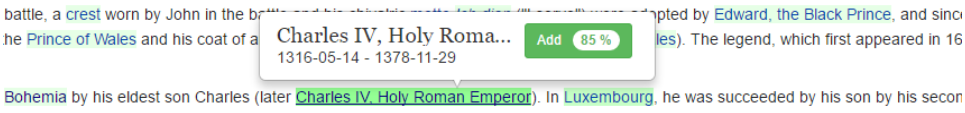
Obr. A.2: Hlavní nabídka

A.4.5 Výběr článků

Po zapnutí zpracování pomocí tlačítka *Start editing* dojde k aktivaci uživatelského prostředí stránky. Po otevření stránky Wikipedie se u nadpisu (pokud je jazyk podporován) objeví tlačítko pro výběr článku, jak ukazuje obrázek A.3. Pomocí tlačítka je možné zobrazenou stránku zařadit do seznamu přidaných článků. V případě, že žádný článek ještě není přidán, je hodnocení nedostupné (*N/A*). V opačném případě se v tlačítku nachází odhad relevance k již přidaným článkům (v procentech). Pokud je článek již přidán, místo tlačítka pro přidání se na jeho místě objeví tlačítko pro odstranění. Po přidání článku se v levé části stránky objeví panel s potvrzením a s články, které mají nejvyšší ohodnocení a které jsou tedy doporučené.

Obr. A.3: Přidání zobrazeného článku

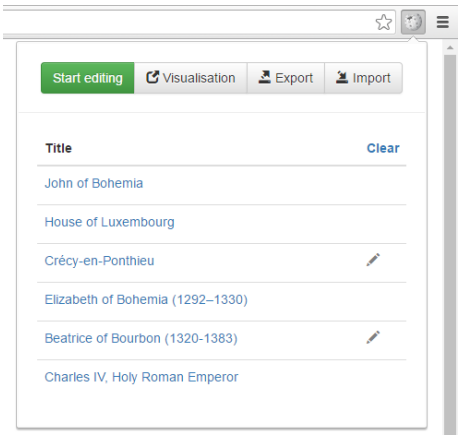
Odkazy stránky jsou postupně zpracovávány a zpracované články je možné rovněž přidávat. Pokud je přidán alespoň jeden článek, odkazy v textu článků jsou postupně zbarvovány podle relevance odkazovaných článků (tmavší, výraznější barva znamená vyšší relevanci). Po najetí kurzorem myši na odkaz se objeví nabídka s možností přidání článku bez nutnosti jeho otevření. V případě rozpoznání jeho trvání (začátek a konec události), je zde rovněž zobrazen – viz obrázek A.4.



Obr. A.4: Přidání článku pomocí odkazu

A.4.6 Správa přidaných článků

V hlavní nabídce se po přidání článků nachází jejich seznam (obrázek A.5. Jednotlivé odkazy na články je možné otevřít. Pokud se nepodařilo získat kompletní rozsah události (začátek a konec), ve stejném řádku je zobrazena „tužka“ a daný článek je tedy s nejvyšší pravděpodobností nutné upravit, k čemuž právě tato ikona slouží. V editačním dialogu je rovněž možné upravit stereotyp události. Vedle ní se také po umístění kurzoru nad řádek objeví tlačítko pro smazání článku. Nad seznamem se nachází tlačítko pro kompletní vymazání celého seznamu.



Obr. A.5: Seznam přidaných článků

S přidáním článku se zaktivují ovládací tlačítka pro export a vizualizaci. Vizualizace zobrazuje propojení (odkazy) článků a může posloužit pro snazší orientaci ve vybraných článcích, nalézt málo propojené (a často tedy

málo relevantní) články a podobně. Exportování článků slouží pro ukládání. Ukládaný formát lze použít jako zdroj pro časovou osu. Pro import slouží tlačítko *Import*, které je aktivní pouze v případě vypnutého módu zpracování. Pomocí něj lze naimportovat dříve vyexportovaný soubor s vybranými články. Případné úpravy dat a stereotypu zůstanou zachovány.

Po exportování vznikne soubor s (výchozím) názvem `data.js`. Tento soubor je možné nahrát do složky `data` nástroje časové osy, čímž jsou exportovaná data použita jako zdroj této osy.