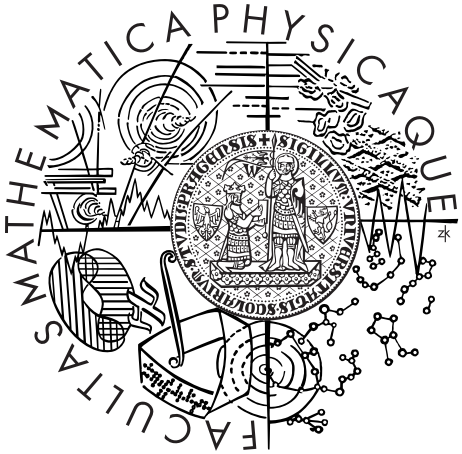
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

**BAKALÁŘSKÁ PRÁCE**




Adam Juraszek

# Webová čtečka a knihovna článků

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Martin Svoboda
Studijní program: Informatika
Studijní obor: Obecná informatika

Praha 2013

Rád bych poděkoval vedoucímu práce RNDR. Martinu Svobodovi za hodnotné poznámky a za veškerý čas, který věnoval vedení této práce.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne .....  Podpis autora

Název práce: Webová čtečka a knihovna článků

Autor: Adam Juraszek

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Martin Svoboda, Katedra softwarového inženýrství

Abstrakt: Cílem práce je navrhnout a implementovat čtečku jako webovou aplikaci, která umožňuje sledovat zdroje článků na Internetu prostřednictvím kanálů RSS a Atom. Od podobných aplikací se odlišuje integrací dalších služeb a možnostmi: sledování stránek, které tyto kanály neposkytují, zálohování stránek, ukládání článků do uživatelem spravované knihovny a systémem doporučování zdrojů. Aplikace je naprogramovaná v jazyku Java a skládá se ze tří částí: serverové části vytvořené na platformě Google AppEngine, klientské části využívající framework GWT a volitelného doplňku do běžných prohlížečů. Webová čtečka je navržená s ohledem na práci s velkým množstvím dat a lze ji snadno rozšířovat. Práce rovněž analyzuje charakteristiky dostupných čteček a rozebírá různé systémy doporučování.

Klíčová slova: webová aplikace, RSS čtečka, AppEngine, GWT

Title: Web Reader and Article Library

Author: Adam Juraszek

Department: Department of Software Engineering

Supervisor: RNDr. Martin Svoboda, Department of Software Engineering

Abstract: The aim of this thesis is to design and implement a reader web application, which can watch article sources on the Internet using RSS and Atom feeds. The difference from similar applications is in integration of other services and options: watching websites that do not provide news feeds, backing up web pages, saving articles into user-managed library and a source recommendation system. The application is programmed in Java and consists of three parts: the server part developed on Google AppEngine, the client part using GWT framework and an optional cross-browser extension. The Web reader is designed to work with large amounts of data and can be easily extended. The thesis also contains an analysis of the characteristics of available readers and various recommendation systems.

Keywords: web application, RSS reader, AppEngine, GWT

# Obsah

# Úvod

Bakalářská práce se zabývá vývojem aplikace – webové čtečky určené pro náročnější uživatele. Kromě běžné funkcionality, sledování nejrůznějších RSS[1] kanálů, v sobě bude integrovat i několik dalších služeb, které jsou dle našeho názoru čtečkám blízké. Myslíme si, že sjednocením vznikne celek, který bude plnit svoji úlohu lépe, než každá ze služeb odděleně.

## Historie projektu

Jelikož je tato práce silně inspirovaná naší zkušeností s dnes již neexistující čtečkou Google Reader[2], popíšeme v této části alespoň stručně její historii a osud.

### Používání čtečky Google Reader

V roce 2005 přišla společnost Google[3] se svojí čtečkou Google Reader. Zpočátku sice obsahovala jen nejnutnější funkcionalitu, nicméně v průběhu následujících několika let se její možnosti postupně rozšířily. Čtečku začaly používat miliony lidí a byly s ní spokojeny; pokrývala všechny jejich potřeby.

V roce 2011 byl nahrazen vnitřní mechanismus sdílení položek napojením na sociální síť Google+[4] od stejné společnosti. Tento krok z hlediska společnosti Google dává dobrý smysl. Propagují svoji sociální síť mezi uživateli čtečky a naopak, usnadňují používání čtečky uživatelům sociální sítě. Mnoho uživatelů ale sociální síť Google+ nechce používat. Důsledek změny způsobu sdílení položek se jich tedy citelně dotkl a omezil možnosti užití této čtečky.

Náročnější uživatelé mohli snadno narazit na hranici jejích možností; některé operace nešlo provést dostatečně pohodlně. Příkladem může být funkcionalita štítků; čtečka je sice nabízela, ale k jejich přiřazení bylo nutné vypsat celý jejich název. To je daleko od ideálu, možnosti nastavit štítkům klávesové zkratky a přiřazovat je stiskem jediné klávesy.

Stiskem klávesy bylo možné položce přidat či odebrat hvězdičku; stejně tak bylo možné si položku oblíbit. Nebylo ale možné přejít na seznam oblíbených položek klávesovou zkratkou, což tento mechanismus učinilo téměř nepoužitelným. Oblíbené položky byly úplně zrušeny po změně sdílení na Google+.

### Konec čtečky Google Reader

V březnu roku 2013 společnost Google vydala prohlášení, že některé služby budou za nedlouho ukončeny. Seznam obsahoval i oblíbenou čtečku Google Reader; její provoz byl zrušen k 1. červenci 2013 [8].

Po tomto prohlášení začaly vznikat petice dovolávající se zachování čtečky; přestože získaly tisíce podpisů, osud čtečky nezměnily. Zároveň začaly zaplavovat

[1] RSS – Rich Site Summary – rodina formátů, které popisují změny, novinky na webových stránkách

[2] Google Reader – `http://www.google.com/reader/about/`

[3] Google – `http://google.com/`

[4] Google+ – `http://plus.google.com/`

internet články obsahující seznam alternativních služeb, ke kterým se mohou aktuální uživatelé uchýlit.

### Vznik projektu

Počátkem roku 2012, krátce po změně mechanismu sdílení, jsme zaznamenali vznik projektu Hivemined[5]. Založil jej bývalý uživatel čtečky Google Reader Francis Cleary[6], jehož cílem bylo vyvinout plnohodnotnou náhradu se všemi vlastnosti původní verze čtečky od Googlu. Postup jeho práce na této náhradě jsme sledovali, nicméně v lednu roku 2012 se odmlčel a nebylo jasné, jestli vývoj dále pokračuje či své snahy zanechal. Na svém blogu se znovu ozval v listopadu 2012 [3]; vývoji čtečky se aktivně věnuje. Během následujícího půlroku, těsně před uzavřením naší práce, se mu podařilo čtečku dokončit pod názvem Hive[7] [4]. Poslal nám e-mailem pozvánku k uzavřenému beta testování, bohužel jsme zatím neměli čas čtečku vyzkoušet a zhodnotit.

Během jeho odmlky vzniklo částečně inspirací zmíněným projektem téma této bakalářské práce. V něm jsme se zaměřili na provázání několika služeb, které dle našeho názoru umožní efektivněji používat rozšířené funkcionality čtečky. Vycházeli jsme především ze zkušeností s několikaletým používám čtečky Google Reader. Chtěli jsme zachovat/rozšířit její dobré vlastnosti a naopak se vyvarovat těch, které nepovažujeme za šťastně řešené.

## Členění práce

V tomto textu se snažíme popsat většinu aspektů aplikace, kterou jsme navrhli a vytvořili. Aplikaci postupně rozebíráme z několika hledisek.

Nejprve v první kapitole rozčleníme čtečky podle zvolených kritérií, uvedeme několik dostupných čteček a porovnáme je v přehledové tabulce. Druhou kapitolu věnujeme vymezení problému, kterým se budeme dále zabývat, a jeho popisu z pohledu požadavků. Ve třetí kapitole představíme existující algoritmy doporučování a zvolíme takový, který bude v našem případě nejvhodnější. Následující, čtvrtou kapitolu zasvětíme představení technologií a knihoven, jež při implementaci využijeme.

Vlastní návrh aplikace podrobně rozebereme v páté kapitole: její členění z pohledu architektury, s jakými entitami pracuje a jaké procesy v ní probíhají. V šesté kapitole přiblížíme vlastní fungování aplikace, popíšeme, jaká důležitá rozhraní obsahuje a jak probíhá komunikace mezi jejími jednotlivými částmi; rovněž prodiskutujeme možná budoucí rozšíření aplikace a problémy, jimž jsme při vývoji čelili. Na závěr krátce shrneme výsledky naší práce.

[5]Hivemined – `http://hiveminedblog.tumblr.com/`
[6]Francis Cleary – apodysophilia@gmail.com
[7]Hive – `http://hivereader.com/`

# 1. Webové čtečky

Na internetu je dnes dostupné poměrně velké množství nejrůznějšího softwaru a nejrůznějších služeb, které poskytují svým uživatelům přehledný výtah z článků, které se objeví na několika webových portálech. Takové aplikace bývají často zaměřeny na různé cílové skupiny: některé se zaměřují na uživatele, kteří se chtějí o novém článku dozvědět okamžitě, jakmile je publikován, jiné zaujmou spíše uživatele, kteří mají příliš mnoho zdrojů na to, aby bylo v jejich silách je pravidelně navštěvovat; v neposlední řadě existují aplikace, které se snaží maximálně zvýšit komfort čtení článků svým uživatelům.

V této kapitole se pokusíme rozebrat charakteristiky jednotlivých typů aplikací.

## 1.1 Typy čteček a jejich uživatelů

Před vlastním dělením aplikací vyčleníme ty, které obsahují čtečku RSS jen jako doplněk nebo ji využívají jen jako informační kanál o svých změnách. Mezi takové aplikace můžeme zařadit například multimediální centrum XBMC[1], které obsahuje na úvodní obrazovce ve spodní části proužek s běžícím textem: tituly článků na domovské stránce projektu. Přestože XBMC má editor RSS kanálů a je možné nastavit vlastní kanál, nemůžeme aplikace takového typu považovat za plnohodnotné RSS čtečky.

Touto a podobnými aplikacemi se zabývat nebudeme; nebudou nás zajímat ani služby, které sice umožňují sledovat RSS, ale není to jejich primárním cílem. Příkladem takové služby může být titulní stránka vyhledávače Seznam.cz[2], kterou si uživatelé mohou upravit a přidat svoje RSS kanály, jejichž položky se na stránce budou zobrazovat. Další podobnou službou je iGoogle[3], který bude ukončený k 1. listopadu 2013.

V následujících podkapitolách rozdělíme čtečky podle několika kritérií:

- způsobu integrace do uživatelského prostředí,
- uživatele, na kterého jsou zaměřené.

Bude zřejmé, že výhody jednoho typu čteček jsou často nevýhodami jiného typu.

### 1.1.1 Nativní aplikace

Pro používání tohoto typu čteček je nutné nainstalovat nativní aplikaci do operačního systému. S tím souvisí několik omezení, se kterými musí uživatel předem počítat:

- výběr aplikací může být omezený operačním systémem, který uživatel používá,

[1] XBMC – `http://xbmc.org/`
[2] Seznam.cz – `http://seznam.cz/`
[3] iGoogle – `http://www.google.com/ig`

- v některém prostředí nemusí mít uživatel oprávnění instalovat aplikace (práce, škola),
- aplikace musí být trvale spuštěná, neboť periodicky kontroluje zdroje jednotlivých kanálů.

Mezi nativní aplikace budeme řadit i samotný internetový prohlížeč, pokud nabízí možnost sledování RSS kanálů.

#### Výhody

Mezi výhody nativní aplikace patří:

- rychlejší odezva. Aplikace může ukládat všechny položky na pevný disk, čímž odbourává nutnost neustálé komunikace a opakovaného stahování veškerých dat ze serveru poskytovatele služby.
- offline čtení. Aplikace může stáhnout kompletní webovou stránku, na kterou se uživatel může podívat a přečíst originální text i v době, kdy nemá přístup k internetu. Toho mohou využívat třeba uživatelé, kteří často cestují a chtějí si zkrátit čtením dlouhé čekání.
- nezávislost na jiném subjektu. Uživatel není závislý na provozovateli služby; webové služby mohou skončit nebo mít výpadek. U nativní aplikace maximálně hrozí zastavení jejího dalšího vývoje, což zpravidla nemívá zásadní vliv na její dostupnost a funkcionalitu.
- úplná kontrola nad daty. Někteří uživatelé nechtějí svěřit své údaje poskytovateli služby; informace o tom, které zdroje sledují či které články se jim líbí považuje za příliš důvěrné. Jiný druh uživatelů se může chtít aktivně zapojit do vývoje aplikace, přiblížit ji svým přáním.

### 1.1.2 Webová služba

Přesným opakem nativních aplikací jsou webové služby (webové aplikace); pro používání webové služby není vyžadována instalace aplikace, data jsou dostupná na kterémkoli zařízení s internetem. Největší slabinou těchto čteček je však nutnost neustálého připojení na internet.

#### Výhody

Mezi výhody aplikací postavených na využití webových služeb patří:

- synchronizace stavu mezi více počítači. Uživatel má přístup ke svým sledovaným zdrojům a odkazům na články odkudkoli; může se přesouvat mezi několika zařízeními a pokračovat v procházení seznamu článků.
- menší riziko přetížení zdroje. V případě, že jeden zdroj sleduje velké množství uživatelů, stačí jeden jediný dotaz pro stažení RSS dokumentu a uspokojení všech uživatelů najednou. S možností přetížení můžou mít problém nativní aplikace. Do této situace se dostal například poskytoval počasí, z jehož serverů si aplikace XBMC stahovala informace příliš často (každých 30 minut) [18].

- není nutné přepínat mezi aplikacemi. Pokud uživatel prochází seznam článků na stránce webové služby, nemusí se pro zobrazení originálního článku přepínat do internetového prohlížeče. Toto omezení neplatí pro samotné internetové prohlížeče a je redukované pro některé nativní aplikace, které umožňují zobrazit zjednodušenou webovou stránku přímo uvnitř sebe.

- uživatel má dostupné všechny položky bez ohledu na četnost spuštění počítače. Uživatel nemusí být stále online, webová služba kontroluje zdroj RSS bez ohledu na stav svých uživatelů. Pokud uživatel nativní aplikace nespustí čtečku po dobu několika dnů, může ztratit informace o některých článcích, zejména u frekventovaných zdrojů.

### 1.1.3 Běžný uživatel

Aby čtečku běžní uživatelé používali, musí jim být schopná nabídnout nějaký přínos, zvýšené pohodlí oproti procházení několika webových stránek. Čtečky, které se zaměřují na běžné uživatele spolu soupeří komfortem, který uživatelům nabízí. Výpis seznamu článků může vypadat třeba jako magazín nebo naopak, čtečka může prezentovat každý článek jen velkým obrázkem. Tyto čtečky většinou umožňují sledovat trendy, sledovat ostatní uživatele a kategorizovat obsah.

Obecně můžeme říci, že řeší potřebu uživatele, kterou bychom popsali následovně: „Mám chvilku volného času, tak se podívám, co je nového. Než abych procházel několik webových stránek, tak navštívím svoji čtečku a zjistím, co je nového zajímavého. Čtečka zajistí, že uvidím souhrn ze všech důležitých webů.“

Pokud má čtečka napojení na sociální sítě či sama je sociální sítí, může být uživatelova motivace ještě rozšířena: „Vím, že uvidím jen takové články, které se líbily ostatním, takže se mi pravděpodobně budou líbit také a nebudu se muset zabývat brakem.“

Tento přístup funguje uživatelům, kteří čtou články na internetu z důvodu, že se chtějí pobavit. Na druhou stranu, pokud by se chtěli dozvědět nepopularizované, originální informace třeba z oblasti vědy a výzkumu, tyto čtečky jim nejspíše nebudou sloužit dobře.

### 1.1.4 Náročný uživatel

Za náročného uživatele budeme považovat uživatele, který používá čtečku z důvodu, že sám již není schopný navštěvovat pravidelně stránky svých informačních zdrojů. Důvodem může být sledování příliš mnoha zdrojů, na kterých je publikováno mnoho článků každý den. „Mnoho“ v tomto kontextu znamená desítky zdrojů a stovky článků či zpráv denně.

Náročný uživatel klade na čtečku specifické nároky:

- vyžaduje kompaktní zobrazení, nesmí se plýtvat místem. Obrazovka musí být vyhrazená seznamu položek; každá položka musí zabírat minimální prostor, aby se jich vešlo na obrazovku co nejvíce.

- preferuje prostý text před grafickými efekty; nic nesmí rozptylovat jeho pozornost.

- vyžaduje aplikaci, která bude zcela či z velké většiny ovladatelná klávesovými zkratkami; pohyb a klikání myši je pomalé. Náročný uživatel ve čtečce tráví spoustu času, který musí využít efektivně.
- nepotřebuje zjednodušené prostředí, protože investice do podrobné konfigurace se mu časem vyplatí.

## 1.2 Přehled čteček

Jak jsme již na začátku kapitoly uvedli, na trhu je velké množství čteček. V této části krátce představíme některé, dle našeho názoru, zajímavé čtečky. U každé rozebereme nabízené vlastnosti a popíšeme ji z hlediska kritérií, které jsme si dříve definovali. Pokusíme se nalézt její klady a zápory a doporučit ji cílové skupině.

### 1.2.1 RSSOwl

RSSOwl[4] je zástupcem čteček, které si uživatel musí nainstalovat do svého operačního systému; mezi nimi RSSOwl vyniká nezávislostí na konkrétní platformě. K plnému využití jejích funkcí je však nutné mít tuto aplikaci neustále spuštěnou; aplikace to uživateli vynahradí upozorňováním na nové položky.

Vzhledem k délce vývoje (RSSOwl je velice stará čtečka, její vývoj začal v roce 2003) a způsobu vývoje (je open-source, kdokoli ji může vylepšit) je pochopitelné, že obsahuje všechny vlastnosti, které se od čtečky očekávají. Čtečka nabízí několik typů zobrazení seznamů položek a poskytuje bohaté možnosti přizpůsobení: vzhledu, štítků, klávesových zkratek, filtrování.

Při prvním spuštění čtečky jsme zvolili možnost importu seznamu zdrojů; všechny zdroje (přibližně sto) se úspěšně vložily. Okamžitě začala jejich kontrola, která vytvořila 2000 položek; při následné práce s nimi jsme nepozorovali žádné zpomalení. Po instalaci nám však nefungovaly klávesové zkratky a nepodařilo se nám je ani zprovoznit.

Čtečka poskytuje snad každou vlastnost, na kterou uživatel pomyslí. To je také důvod, proč ji nepovažujeme za vhodnou pro úplné začátečníky. Začátečník, který nemá s tímto typem softwaru větší zkušenost, se může lehce ztratit v záplavě možností, byť čtečka obsahuje průvodce, který je všechny krátce představí.

### 1.2.2 Opera

Opera[5] je webový prohlížeč, který nabízí netradiční integraci několika dalších komponent. Nás bude především zajímat Opera Mail, která umí sledovat RSS a Atom[6] kanály. Možnosti této čtečky jsou značně omezené, umožňuje pouze nastavit interval kontroly každého zdroje a rozložení obrazovky; jednotlivé položky je možné třídit přidáním štítků.

Testování této čtečky pro nás bylo spíše zklamáním, chyběly nám některé z důležitých vlastností. Na druhou stranu oceňujeme integraci do prohlížeče, které nemáme co vytknout: kanály je možné do čtečky přidávat kliknutím na ikonku

[4] RSSOwl – `http://www.rssowl.org/`

[5] Opera – `http://www.opera.com/`

[6] Atom – formát určený k popisu změn webových stránek; je novější alternativou k RSS [14]

v adresním řádku, celý článek se zobrazuje v novém listu. Pokud je uživatel zvyklý na vzhled tohoto prohlížeče, bude spokojený i se vzhledem a chováním čtečky.

Tuto čtečku můžeme doporučit uživatelům, kteří prohlížeč Opera již používají; nemyslíme si, že samotná čtečka přiláká jiné uživatele. Zároveň se domníváme, že čtečku budou využívat především příležitostní uživatelé, kteří na ni nebudou klást příliš velké nároky. Jakmile začnou vyžadovat pokročilejší vlastnosti čtečky, budou se muset porozhlédnout po jiné. Je to prostě jen příjemné rozšíření prohlížeče, které potěší běžné uživatele.

### 1.2.3 Tiny Tiny RSS

Tiny Tiny RSS[7] je čtečka na pomezí webové služby a nativní aplikace. Neexistuje žádný oficiální poskytovatel služby, u kterého by se uživatelé mohli registrovat, a není doporučené ji instalovat na osobní počítač. Obvykle ji provozuje jednotlivec či skupina na vlastním serveru, často jím je domácí NAS[8], či pronajatý (virtualizovaný) server. Složitější způsob instalace slouží jako filtr potenciálních uživatelů.

Seznam položek je přehledný, lehce ovladatelný; všechny ovládací prvky se nacházejí na očekávaných místech. Čtečka překypuje mnoha drobnými vylepšeními, které uživatele potěší a zjednoduší mu její používání. Na druhou stranu, nefungovala nám taková základní funkcionalita, jakou je navigace v historii (tlačítka zpět a vpřed v prohlížeči). Několikrát nám čtečka přestala reagovat, pomohlo až obnovení stránky; nevíme, zda šlo o chybu aplikace nebo problém s konfigurací.

Tiny Tiny RSS můžeme doporučit pokročilým technicky zdatným uživatelům, kteří mají rádi svá data pod kontrolou. Pokud zvládnou složitější instalaci a konfiguraci, čtečka jim nabídne všechny vymoženosti, které poskytuje konkurence.

### 1.2.4 Netvibes

Netvibes[9] je webová služba, která je dostupná v základní verzi zdarma; pokročilejší verze jsou zpoplatněné. Netvibes se snaží nabídnout uživatelům pohodlné a graficky vyvedené rozhraní; nabízí dva různé pohledy na sledované kanály. První z nich má podobu nástěnky – graficky bohatých bloků na stránce, které zobrazují několik posledních položek z vybraných kanálů. Tento pohled je vhodný pro uživatele, kteří potřebují sledovat několik zdrojů najednou, paralelně vedle sebe. Druhý z nabízených pohledů je tvořen klasickým seznamem všech položek, lze ho filtrovat podle zdroje, ze kterého položky pochází. Tím však jeho možnosti nastavení končí.

Vyzkoušeli jsme do čtečky vložit seznam všech našich zdrojů, čtečka je během několika málo sekund načetla a začala poskytovat položky. Potěšila nás dobrá podpora klávesových zkratek a přítomnost kontextové nápovědy zobrazující všechny aktuálně dostupné operace. Zklamala nás ale téměř zcela chybějící podpora pro jakékoli ukládání a správu položek; čtečka umožňuje pouze označení položky k pozdějšímu přečtení.

[7] Tiny Tiny RSS – `http://tt-rss.org/`

[8] NAS – Network-attached storage – zařízení v lokální síti, které provozuje souborový server; obvykle je hardware a software takového zařízení uzpůsoben svému účelu

[9] Netvibes – `http://www.netvibes.com/`

Se čtečkou se nám nepracovalo příliš pohodlně, text přečtených položek je málo kontrastní, ikonky stavu aktuální položky jsou umístěné jinde než ikonky ostatních položek. Grafické rozhraní nám občas přišlo pomalé a její dvojakost (nástěnka – seznam) v nás zanechala silně rozpačité pocity. Čtečku můžeme doporučit uživatelům, kterým se líbí vzhled aplikace a nevyžadují od ní pokročilé vlastnosti.

### 1.2.5 Google Reader

Google Reader[10] je čtečka vyvinutá společností Google v roce 2005; postupem času získávala podporu jednotlivých vlastností. V březnu 2013 bylo oznámeno plánované ukončení provozu této čtečky; nyní již není dostupná [8]. Zmiňujeme ji především z důvodu, že inspirovala další tvůrce k vytvoření alternativ.

V době, kdy byla čtečka Google Reader ještě dostupná, se nám s ní pracovalo pohodlně; byla čtečkou, kterou jsme rutinně používali. Měla svá omezení, především v práci se štítky; chyběly nám klávesové zkratky pro některé méně časté operace. Musíme však ocenit její rychlost a přehledné rozhraní. Její výhodou bylo také velké množství aplikací třetích stran, které z ní přebíraly položky a umožňovaly synchronizaci jejich stavu.

### 1.2.6 The Old Reader

The Old Reader[11] je jednou z náhrad čtečky Google Reader; autoři ji začali vyvíjet v polovině roku 2012. Důvodem vzniku byl jejich nesouhlas se směrem, kterým se čtečka od Googlu začala ubírat. Po březnovém oznámení ukončení čtečky Google Reader začalo tuto čtečku používat mnohonásobně více lidí, to její další vývoj urychlilo.

Čtečku jsme vyzkoušeli a vložili do ní všechny naše zdroje. Samotný import trval několik minut; již během tohoto importu byla čtečka použitelná; postupně přibývaly všechny zdroje. Čtečka se nám z grafického hlediska líbí, její rozhraní je přehledné a ovládání logické. Jediným nedostatkem, který jsme zaznamenali, je nedostatečné odlišení přečtených a nepřečtených položek (nepřečtené se liší pouze zeleným proužkem po straně).

Čtečka nabízí tři akce, které je možné vykonat s každou položkou: sdílet ji ve svém RSS kanálu, ponechat nepřečtenou a oblíbit si ji. Pokud naši přátelé také používají tuto čtečku, je možné přímo sledovat položky, které označili ke sdílení. Oblíbené položky slouží k ukládání položek, jiný mechanismus aplikace neposkytuje; uložené položky není možné nijak třídit.

Čtečku považujeme za povedenou; je vhodnou volbou pro uživatele hledající nástupce zrušené čtečky Google Reader. Můžeme ji doporučit i uživatelům, kteří s čtečkami nemají velké zkušenosti; těm se může líbit možnost nalezení uživatelů, kteří mají v „oblíbených“ stejné položky. Naopak, čtečka nejspíše nebude postačovat náročným uživatelům, kteří si chtějí katalogizovat zajímavé články a budovat si z nich vlastní knihovnu.

[10] Google Reader – `http://www.google.com/reader/about/`
[11] The Old Reader – `http://theoldreader.com/`

### 1.2.7 Feedly

Feedly[12] je další z oblíbených náhrad ukončeného čtečky Google Reader; tato čtečka pravděpodobně zaznamenala příliv největší části jejích bývalých uživatelů. Média tuto čtečku často vyzdvihují a zmiňují; obzvláště zřetelné to bylo v předčervencových článcích informujících o blížícím se ukončení čtečky poskytované společností Google.

Stejně jako ostatní, zkusili jsme i tuto čtečku nějakou dobu používat. Rozhraní na nás jako celek působilo vlídně a přehledně; problém s používáním jsme zaznamenali u zobrazení položek. Příliš velký prostor je vynechaný kolem samotného textu položky; ovládací prvky jsou nelogicky rozházené do několika skupin. Velice nás obtěžovala přítomnost ikonek několika sociálních sítí, z nichž ani jednu nepoužíváme, nacházejí se totiž na místě, které by bylo vhodnější pro některé jiné, častější akce. Čtečka umožňuje označit položku k pozdějšímu přečtení nevýraznou ikonkou, kterou je snadné přehlédnout. Podporované je rovněž přiřazování štítků jednotlivým položkám, bohužel text štítku je špatně čitelný.

Feedly sice nabízí hodně možností, ale celkový dojem jí zhoršují detaily, které uživateli ztěžují běžnou práci. Cílovou skupinou této čtečky jsou uživatelé, kteří hledají náhradu za Google Reader. Čtečku můžeme doporučit jen těm, kteří nebyli spokojení s konkurenční čtečkou The Old Reader, od které se příliš neliší.

## 1.3 Srovnání čteček

V přehledné tabulce 1.1 stručně shrneme výsledky našeho průzkumu čteček, srovnáváme je v několika kritériích. Součástí tabulky je i námi návržená a naprogramovaná čtečka.

[12] Feedly – `http://www.feedly.com/`

Tabulka 1.1: Srovnání čteček

| **Čtečka** | **Typ** | **Cílový uživatel** | **Ukládání položek** | **Klávesové zkratky** |
|---|---|---|---|---|
| **RSSOwl** | nativní aplikace | náročný uživatel vyžadující čtečku bez kompromisů | přiřazení štítků, zvýraznění položky, přesun/kopírování položek do adresářů | kompletní, konfigurovatelné |
| **Opera** | nativní aplikace – internetový prohlížeč | uživatel Opery, který nevyžaduje pokročilé vlastnosti | přiřazení štítků, připnutí položky | základní, konfigurovatelné |
| **Tiny Tiny RSS** | webová služba, která vyžaduje vlastní server | náročný uživatel, který chce mít data pod kontrolou | přiřazení štítků, přidání hvězdičky | kompletní |
| **Netvibes** | webová služba | běžný uživatel, který potřebuje sledovat zároveň více zdrojů | přečíst později | kompletní |
| **Google Reader** | webová služba | ukončena; všechny typy uživatelů | přiřazení štítků, přidání hvězdičky | téměř kompletní |
| **The Old Reader** | webová služba | nepříliš náročný uživatel | oblíbené položky | kompletní |
| **Feedly** | webová služba | uživatelé, kterým nevyhovuje The Old Reader | přiřazení štítků, přečíst později | téměř kompletní |
| **Naše aplikace** | webová služba | náročný uživatel | přiřazení štítků | pokročilé, konfigurovatelné |

# 2. Analýza aplikace

V této kapitole nejprve identifikujeme vlastní problematiku čteček, jíž se budeme dále zabývat, a popíšeme požadavky na aplikaci, která bude dané problémy řešit.

## 2.1 Identifikace problému

Naší snahou je pokrýt jedinou aplikací několik příbuzných problémů, kterým čelí náročný uživatel. Náročný uživatel má podle nás především následující potřeby.

### 2.1.1 Přístup odkudkoli

Z vlastní zkušenosti víme, že náročný uživatel často střídá zařízení, na kterých prochází internet. Může to být domácí počítač, notebook, mobilní telefon, tablet; uživatel chce mít přístup ke čtečce odkudkoli. Vyžaduje, aby bylo možné přejít z jednoho zařízení na druhé a na něm pokračovat ve čtení článků.

Hledáme tedy webovou službu, která bude zajišťovat synchronizaci a dostupnost bez ohledu na zařízení, které uživatel aktuálně používá.

### 2.1.2 Kanály RSS, Atom

Náročný uživatel chce/musí sledovat několik desítek různých webových stránek, blogů, portálů. Naštěstí, většina z nich poskytuje informace o novinkách (nových článcích, které byly publikovány) pomocí kanálů RSS nebo Atom. Toto je také nejčastější využití hledané čtečky; vyhovuje mu téměř každá čtečka, kterou jsme popsali v minulé kapitole.

Čtečka musí podporovat získáváni informací z kanálů RSS a Atom.

### 2.1.3 Stránky bez RSS a Atomu

Mnoho webových stránek neposkytuje kanály RSS ani Atom; jedná se často o různé blogy, z akademického prostředí jsou to například stránky přednášek nebo cvičení. Existují služby, které umožňují sledovat webovou stránku a v případě detekování její změny nás informují e-mailem; takovou službou je například ChangeDetection[1]. Přestože tato služba může svoji úlohu vykonávat dobře, neumožňuje integraci s jinými systémy.

Zmíníme ještě snahu studentů, kteří si často píší jednoúčelové skripty, které periodicky stahují webovou stránku cvičení a porovnávají její obsah s minulou verzí. Jejich přístup ale trpí stejným problémem jako výše zmíněná služba.

Často nás nezajímá samotná změna stránky, ale důvod, proč k ní došlo. Pokud publikujeme nový článek na webu (vytvoříme tedy novou stránku), dáme o něm vědět (přidáme odkaz) většinou v seznamu nejnovějších článků na úvodní stránce. Pokud vyjdeme z tohoto předpokladu, nezajímá nás přítomnost nového odstavce na stránce, ale odkaz na samotný nový článek, který je v odstavci zmíněn.

Nejsme si vědomi, že by existovala služba, která by umožňovala detekci nově publikovaných článků pomocí rozboru samotných odkazů na webové stránce.

[1] ChangeDetection – `http://www.changedetection.com/`

### 2.1.4 Uložení adresy článku

Pokud si uživatel chce uložit nějaký článek, resp. jeho internetovou adresu, typicky má na výběr několik různých služeb.

Uživatel si může uložit adresu webové stránky do záložek v prohlížeči, nicméně jejich synchronizace je problematická. Výrobci prohlížečů sice často umožňují jejich synchronizaci, ale ta je podmíněná tím, že uživatel používá všude prohlížeč od stejného výrobce. Synchronizaci záložek umožňují také různé služby, které nabízejí doplňky do prohlížečů. Některé z nich fungují dokonce nezávisle na konkrétním prohlížeči, nicméně je třeba je mít nainstalované, jinak pohodlí používání těchto služeb dramaticky klesá. Příkladem je služba Delicious[2].

S těmito službami souvisí tlačítka, která najdeme všude na internetu obvykle pod názvy „share button“. Jejich příkladem může být plugin AddToAny[3] do redakčního systému Wordpress[4]. Použitím těchto tlačítek je možné odeslat články do některé z rozšířenějších služeb; často je jejich seznam omezený na dvě nejznámější sociální sítě. Počet webových portálů, které ukládání článků tímto způsobem umožňují, je značně omezený, určitě na tuto vlastnost nelze obecně spoléhat.

Čtečky RSS tuto problematiku také částečně řeší, umožňují k položkám přidávat hvězdičky a štítky. Problémem ale stále zůstává fakt, že štítek je možné přidat pouze k článku, který je ve čtečce dostupný. Obvykle není možné přidat libovolnou vlastní internetovou adresu do RSS čtečky tak, aby s ní čtečka pracovala stejným způsobem, jako s ostatními položkami.

Jelikož je naše aplikace povahou čtečka, budeme vyžadovat možnost jednoduchého přidání internetové adresy jako nové položky.

### 2.1.5 Přečíst později

Často si uživatelé nemohou přečíst zajímavý článek okamžitě, protože na to nemají čas, nebo se nenacházejí ve vhodném prostředí. Důvodem může být to, že článek je příliš dlouhý nebo obsahuje video. V takových případech si chtějí uživatelé článek uložit/poznamenat k pozdějšímu přečtení.

Jednou z možností je využít některou ze služeb zmíněných v minulém bodu. Další možností, které uživatelé využívají, jsou specializované služby. Tyto služby se často snaží přeformátovat článek do podoby vhodnější ke čtení (odstraňují rušivé elementy). Zástupcem je například Instapaper[5] či Pocket[6].

### 2.1.6 Efektivní ovládání

Aby náročný uživatel mohl čtečku efektivně používat, musí mu aplikace nabídnout adekvátní prostředí. Nároky, které na toto prostředí uživatel klade, jsme již uvedli v části 1.1.4. Zjistili jsme, že efektivního ovládání aplikace může uživatel docílit používáním klávesových zkratek.

[2]Delicious – `http://delicious.com/`
[3]AddToAny – `http://wordpress.org/plugins/add-to-any/`
[4]Wordpress – `http://wordpress.org/`
[5]Instapaper – `http://www.instapaper.com/`
[6]Pocket – `http://getpocket.com/`

Ke klávesovým zkratkám ještě zmíníme několik poznámek. Nejčastější operace musí být snadno vyvolatelné, ideálně stisknutím jediného tlačítka. Uživatel by měl mít možnost nastavit si své vlastní schéma klávesových zkratek.

Uvedeme ještě dvě další potřeby, které nejsou tak důležité, jako výše zmíněné, nicméně jsou jejich přirozeným rozšířením.

### 2.1.7 Zálohování článku

Občas se stane, že webový portál, na němž byl publikovaný zajímavý a pro uživatele důležitý článek, ukončí svůj provoz. V takové situaci je uživatel většinou rozladěn a následně hledá, zda je článek dostupný v internetových archivech.

Aby tomu předešel, mohl by si zazálohovat aktuální podobu článku pro případ, že originál nebude dostupný. Všechny prohlížeče umožňují uložit webovou stránku na disk, nicméně správa takto uložených stránek určitě není pohodlná.

Služba, kterou hledáme, by měla umožnit zazálohovat konkrétní webovou stránku a sama jí poskytovat v zazálohované podobě pro případ, že originál nebude dostupný.

### 2.1.8 Sdílení článků

Uživatelé mezi sebou nejčastěji sdílejí odkazy na zajímavé články pomocí sociálních sítí; ku příkladu Twitter[7] je na tomto principu prakticky založen. Sociální sítě umožňují sledovat několik zajímavých osob a zobrazovat obsah, který se jim líbí či jej doporučují.

Těmto službám nechceme konkurovat, chceme nabídnout odlišný přístup ke sdílení. Nabídnout uživatelům možnost nenásilně si vytvořit několik vlastních kanálů, třeba tematicky zaměřených, které mohou ostatní odebírat bez ohledu na aplikaci, kterou používají.

Přestože existuje mnoho aplikací, které naplňují jednotlivé potřeby, tak jak jsme je zde popsali, nepodařilo se nám najít žádnou takovou, která by komplexně řešila celou výše zmíněnou problematiku.

## 2.2 Zavedení pojmů

Před samotným rozborem problému zavedeme několik pojmů, které budeme používat v následujících částech dokumentu. Jednotlivým termínům bude později odpovídat jedna entita.

**Uživatel** Osoba, která používá naši aplikaci. Musíme být schopni rozlišit jednotlivé uživatele a poskytnout jim různý obsah.

**Zdroj** Zdroj je reprezentací jednoho dokumentu (HTML[8], XML[9]) v síti internet,

[7] Twitter – `http://twitter.com/`

[8] HTML – HyperText Markup Language – hlavní značkovací jazyk používaný při tvorbě webových stránek [17]

[9] XML – Extensible Markup Language – univerzální strojově i lidsky čitelný značkovací jazyk určený pro ukládání a výměnu dat [2]

který naše aplikace používá k získávání položek. Zdrojem může být webová stránka nebo některý ze systémů poskytování informací o novinkách.

**Kontrola zdroje** Úkon provedený k zjištění, zda zdroj obsahuje nové položky. Obvykle je kontrola realizovaná stáhnutím dokumentu reprezentovaného zdrojem a porovnáním s jeho minulou verzí.

**Položka** Položka reprezentuje odkaz na jeden článek nebo stránku v síti internet. Každá položka je vázaná právě k jednomu zdroji.

**Štítek** Popisek, který může uživatel přidat k položce. V našem případě bude popisek krátký, většinou jednoslovný.

## 2.3 Analýza požadavků

Identifikované problémy popíšeme z různých pohledů vlastními požadavky, které bude naše aplikace splňovat.

### 2.3.1 Funkční požadavky

Mezi požadavky na funkcionalitu, které musí aplikace splňovat, patří následující:

- Každý uživatel aplikace si může zaregistrovat svoje zdroje, které bude sledovat.
- Existuje několik typů zdrojů: zdroje, které přebírají položky z kanálu RSS nebo Atom a zdroje, které parsují informace o novinkách přímo z webových stránek.
- Každý uživatel si bude moci do aplikace přidávat svoje položky.
- Seznam všech položek bude možné filtrovat podle několika kritérií.
- Uživatel může zazálohovat jakoukoli webovou stránku.
- Uživatel má možnost sdílet svoje články.

### 2.3.2 Požadavky na použitelnost

Požadavky na použitelnost popisují, jak má aplikace vypadat a jak se má chovat, aby se pohodlně používala a ovládala.

- Aplikace musí být ovladatelná klávesovými zkratkami.
- Největší část aplikace bude zaujímat seznam položek.
- Každá položka obsahuje odkaz na originální webovou stránku.
- Aplikace bude fungovat ve všech běžných prohlížečích.

### 2.3.3 Výkonnostní požadavky

Výkonnostní požadavky určují parametry zátěže, na kterou má být aplikace navržena.

- Aplikaci může používat několik stovek uživatelů.
- Každý uživatel může mít desítky až stovky zdrojů.
- Zdroj může poskytovat desítky položek denně.
- Každý zdroj se kontroluje přiměřeně často v závislosti na frekvenci změn.
- Aplikace bude dobře škálovatelná.

## 2.4 Moduly aplikace

Z požadavků, které na aplikaci klademe, vyplývají důsledky na volbu její architektury. Aplikace se bude skládat ze tří modulů, které spolu budou vzájemně komunikovat.

### 2.4.1 Serverová část

Serverová část aplikace bude zodpovědná za získávání nových položek z jednotlivých zdrojů, uložení všech dat včetně uživatelského nastavení a poskytování těchto dat ostatním modulům.

### 2.4.2 Klientská část

Klientská část bude zobrazovat data získaná ze serverové části a bude interagovat s uživatelem, který pomocí ní aplikaci ovládá. Půjde o standardní webové stránky fungující v běžných prohlížečích.

### 2.4.3 Doplněk do prohlížeče

Doplněk bude tvořit nepovinné rozšíření do nejběžnějších prohlížečů; bude mít za cíl zjednodušit práci s aplikací. Tento doplněk nebude poskytovat žádnou vyšší funkcionalitu, které by nešlo docílit klientskou částí. Bude umožňovat uživateli přidat novou položku do svého manuálního zdroje pouhými dvěma kliknutími myší.

# 3. Doporučování zdrojů

Abychom uživatelům nabídli možnost objevovat nové zdroje, které jsou podobné těm, které již sledují, bude aplikace obsahovat systém doporučování zdrojů na základě podobnosti. Na tomto místě popíšeme a rozebereme různé způsoby doporučování. Ve formulaci problematiky budeme používat pojmy uživatel – osoba, která interaguje se systémem a položka – objekt, který nabízíme a chceme doporučovat; v našem případě je položkou ve formulaci problematiky jakýkoli zdroj, který uživatelé sledují.

## 3.1 Globální doporučování

Nejjednodušším způsobem doporučování je využití globálních statistik o všech položkách. Vychází z předpokladu, že existují položky, které jsou objektivně kvalitní a vhodné pro většinu uživatelů. Kritérii, podle kterých jsou položky seřazeny, mohou být:

- počet uživatelů, kteří si zakoupili položku,
- počet zhlédnutí položky,
- průměr hodnocení všech uživatelů,
- počet recenzí položky a další.

Tento způsob používají některé menší internetové obchody z důvodu snadné implementace a nenáročnosti výpočtu.

Globální doporučování je v našem případě nevhodné, naším cílem není doporučovat velké zpravodajské servery, které zná každý uživatel. Naším cílem je naopak nabídnout uživatelům i menší zdroje, které třeba nemusí být mezi uživateli příliš známé. Takové zdroje mohou být úzce zaměřené, ale přesto vysoce kvalitní; příkladem jsou blogy různých vývojářů.

## 3.2 Doporučování na základě podobnosti

Všechny systémy doporučování vyžadují prvotní znalost o uživateli – modelu, který popisuje preference uživatele. Model uživatele může obsahovat nejrůznější informace:

- údaje, které si o sobě vyplní sám uživatel: oblíbený žánr (u hudby), oblíbený autor (u knih),
- hodnocení, které sám vyplnil: klikl na tlačítko „líbí/nelíbí“, ohodnotil položku na stupnici 1–10,
- záznam dosavadní interakce se systémem: písničky, jež si uživatel poslechl; knihy, které zakoupil,
- případně podrobněji: kategorie či produkty, které navštívil; čas, který strávil na jednotlivých stránkách.

Existují dva základní přístupy k nabízení položek na základě podobnosti. Buď doporučují položky na základě podobnosti obsahu – detailní znalosti atributů – položek, které se uživateli již líbily, nebo naopak porovnávají, jaký vztah k položce mají podobní uživatelé. Nároky na systémy doporučování ve velkých obchodech si vynucují konstrukci hybridních algoritmů, které se snaží získat dobré vlastnosti z obou skupin.

### 3.2.1 Doporučování založené na podobnosti obsahu

V případě, že položky v systému jsou takové povahy, že jsou o nich známé podrobné informace, může systém nabízet položky, které jsou podle nějakých kritérií blízké těm, které se uživateli již dříve líbily. Potřebnými podrobnými informacemi mohou být například [16]:

- zvukové charakteristiky hudební nahrávky, její tempo, melodie apod.,
- text jednotlivých knižních děl jako vektory ve vektorovém prostoru nad slovy.

Tento přístup však není v našem případě použitelný, nemáme dostatečné informace o jednotlivých zdrojích; jediné, co známe je typ zdroje a jeho internetová adresa.

### 3.2.2 Doporučování na základě podobnosti uživatelů

V dobách, kdy ještě neexistovaly počítače, si lidé mezi sebou doporučovali ku příkladu knihy, které se jim líbily. Jelikož se obvykle přátelí lidé, kteří mají podobné zájmy, je vysoce pravděpodobné, že kniha, která se jedné osobě líbila, se bude líbit i jejím přátelům.

Doporučování založené na podobnosti uživatelů je na tomto principu založené [19]. Pokud budeme mít informace o oblíbených položkách mnoha uživatelů, můžeme se uživateli pokusit doporučit takové, které se líbí jeho okolí. Okolím myslíme v tomto kontextu všechny uživatele, kteří jsou jemu podobní.

Všechny algoritmy tohoto typu pracují s maticí uživatelů a položek; v řádcích jsou všichni uživatelé, ve sloupcích jsou všechny položky. Prvek matice vyjadřuje hodnocení $i$-tého uživatele $j$-té položky. Pro nalezení vhodných položek k doporučení můžeme porovnávat vektory:

- uživatelů pro všechny položky: pokud jsou si dva vektory blízké, líbí se položka stejné skupině uživatelů,
- položek pro všechny uživatele: pokud jsou si dva vektory blízké, hodnotí uživatelé položky podobně – mají stejný vkus. Na základě omezené skupiny podobných uživatelů se v druhém kroku dopočítají konkrétní položky k doporučení.

Podobnost vektorů je možné porovnávat několika způsoby, například mírou jejich korelace či skalárním součinem (úhel mezi vektory).

## 3.3 Volba systému doporučování

Chování dat v naší aplikaci se liší od běžných modelů doporučování; mezi specifické vlastnosti patří:

- Uživatelů bude relativně málo; doporučení se vypočítává pouze uživatelům, kteří aplikaci používají (sledují alespoň jeden zdroj).
- Změna hodnocení položek (v našem případě: 0 – uživatel zdroj nesleduje, 1 – uživatel zdroj sleduje) neprobíhá příliš často. Vycházíme z pozorování, že uživatel stále sleduje stejné zdroje. Bude nám tedy stačit přepočítávat doporučení offline – například jednou za den, místo na prostředky náročného výpočtu při každém dotazu.
- Matice uživatelů a zdrojů je řídká; mohou sice existovat zdroje, které sleduje většina uživatelů, ale žádný uživatel nebude sledovat většinu zdrojů. Mnoho zdrojů bude sledováno právě jedním uživatelem, půjde nejčastěji o různé specificky zaměřené blogy.
- Nemáme téměř žádné využitelné informace o samotných zdrojích, jen jejich typ a internetovou adresu.

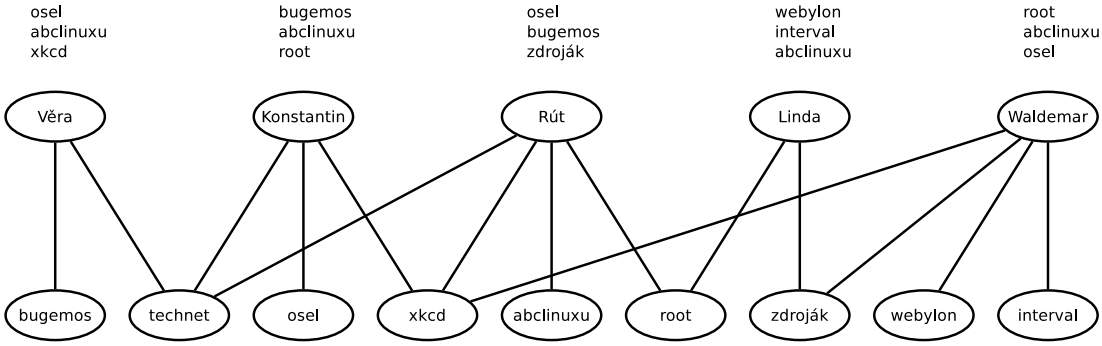
Z výše uvedených předpokladů vyplývá, že nejvhodnější volbou algoritmu bude některá z variant doporučování založená na podobnosti uživatelů. Jako největší problém při návrhu doporučování v naší aplikaci považujeme řídkost dat: obáváme se, že uživatelé nebudou dostatečně sdílet své zdroje, aby doporučení dávalo relevantní výsledky.

Zvolili jsme algoritmus, který navrhli Z. Huang, D. Zeng, H. Chen ve svém článku [11]. Tento algoritmus je do velké míry inspirovaný algoritmy známými z prostředí internetových vyhledávačů, jakým je např. algoritmus PageRank [15].

### 3.3.1 Popis algoritmu

Algoritmus si v průběhu výpočtu udržuje seznam zdrojů, které jsou vhodné k doporučení uživateli. Princip algoritmu je následující:

1. inicializuje seznam zdrojů našeho uživatele zdroji, které již sleduje,
2. najde všechny uživatele, kteří sledují některý ze zdrojů, které jsou v seznamu našeho uživatele: nazveme je sousedy,
3. vypočítá, jak jsou si náš uživatel a soused blízcí, podle toho přidělí každému sousedovi váhu,
4. přidá všechny zdroje všech sousedů do seznamu zdrojů našeho uživatele a přiřadí jim váhy podle toho, kteří sousedé je sledují,
5. iteruje několikrát kroky 2–4,
6. vrátí nový seznam zdrojů vhodných pro našeho uživatele seřazený podle jejich váhy.




Obrázek 3.1: Uživatelé sledující zdroje a jejich doporučení

Výhody a nevýhody tohoto algoritmu:

+ umožňuje vypočítat doporučení pro všechny uživatele najednou,

+ všechny operace lze implementovat efektivně jako operace s maticemi; kroky 2–4 spočívají pouze v násobení matic,

− algoritmus je iterativní – iterace by ideálně měla probíhat, dokud se výsledek neustabilizuje,

− je nutné udržovat v paměti celou matici, přestože nás může zajímat doporučení pro jediného uživatele.

### 3.3.2 Příklad doporučení

Naše aplikace bude vypočítávat doporučení pro všechny uživatele jednou za 24 hodin. Výsledky doporučení budou uživatelům dostupné v dialogu pro přidání nového zdroje. Mohou si vybrat, zda zadají vlastní internetovou adresu zdroje nebo chtějí sledovat některý z doporučených zdrojů.

Abychom přiblížili možnosti algoritmu, uvedeme příklad doporučení zdrojů na příbězích pěti lidí[1].

- Věra je běžná uživatelka, která sleduje zpravodajský magazín o technice (technet) a ráda se pobaví u vtipného komixu (bugemos).
- Konstantin se také zajímá o techniku, stejně jako Věra sleduje technet. O téma se však zajímá hlouběji, čte články na oslovi a baví se náročnějším komixem (xkcd).
- Rút je srdcem linuxačka; mezi její sledované weby patří především abclinuxu a root. Zajímá se okrajově o techniku (technet) a čte komix xkcd.
- Linda je Rútina kamarádka; začíná s tvorbou webu, sleduje tedy návody na zdroják. Ve čtečce má i web root, protože chce vědět, co je ten Linux zač.

[1]Popsané osoby jsou vymyšlené; uvedené zdroje skutečně existují, jsou jimi: `http://bugemos.com/`, `http://technet.idnes.cz/`, `http://osel.cz/`, `http://xkcd.com/`, `http://www.abclinuxu.cz/`, `http://www.root.cz/`, `http://www.zdrojak.cz/`, `http://webylon.info/` a `http://interval.cz/`.

```
Vera: ['bugemos', 'technet']
['osel (1.4)', 'abclinuxu (1.2)', 'xkcd (1.0)']
Konstantin: ['technet', 'osel', 'xkcd']
['bugemos (2.0)', 'abclinuxu (1.8)', 'root (1.5)']
Rut: ['technet', 'xkcd', 'root', 'abclinuxu']
['osel (2.4)', 'bugemos (2.3)', 'zdrojak (2.2)']
Linda: ['root', 'zdrojak']
['webylon (1.4)', 'interval (1.4)', 'abclinuxu (1.2)']
Waldemar: ['xkcd', 'zdrojak', 'webylon', 'interval']
['root (2.3)', 'abclinuxu (1.9)', 'osel (1.9)']
```

Výpis 3.2: Výstup algoritmu doporučení po 10 iteracích.

- Waldemar se věnuje tvorbě webových stránek, a proto sleduje všechny tématické weby (zdroják, webylon, interval). Jako mnoho ostatních má rád komixy na xkcd.

Na schématu 3.1 je znázorněno, který uživatel sleduje které zdroje. Nad každým uživatelem je seznam zdrojů, které mu byly doporučeny.

Doporučení jsme vypočítali skriptem, který je dostupný na přiloženém CD a přesně odpovídá implementaci, kterou používame ve vlastní aplikaci. Jeho výstupem (výpis 3.2) je dvojice řádků pro každého uživatele: jméno a sledované zdroje; doporučené zdroje (jejich váha).

- Věře jsou doporučeny zdroje, které sledují její sousedé (Konstantin a Rút). Přednost získaly zdroje, které sledují jen sousedé (osel, abclinuxu), protože se nerozkládá jejich váha mezi více uživatelů(xkcd a root sledují i jiní uživatel, kteří jsou Věře vzdálenější). Za dobré doporučení můžeme považovat zdroje osel a xkcd.

- Konstantinovi jsou doporučeny opět zdroje jeho sousedů (bugemos, abclinuxu) a root, u kterého se opět dělí jeho váha mezi více uživatelů. Dobrým doporučením je jen bugemos.

- Rútina záliba v komixech a technice jí způsobí doporučení osla (s Konstantinem, který jej sleduje má společnou polovinu zdrojů). Rút je Konstantinovi velice blízká, to je důvodem zvýšení váhy technetu (který sledují oba). Tedy systém doporučení preferuje Věru před Lindou a doporučí bugemos před zdrojákem. Bugemos a osel jsou dobrými doporučeními.

- Lindě jsou doporučeny zdroje jejích sousedů (Waldemara a Rút). Waldemarovi zdroje mají větší váhu, protože Waldemar získal doporučením dvou zdrojů větší váhu (projev iterace algoritmu). Všechny doporučené zdroje jsou dobré.

- Waldemarovi jsou doporučeny zdroje, které sledují sousedé (Linda, Rút, Konstantin). Abclinuxu a osel mají větší váhu než technet (nerozkládá se jejich váha; $1 > \frac{2}{3}$). Nejvhodnějším zdrojem pro Waldemara je ale root, který je sledovaný všemi jeho sousedy. Waldemarovi není doporučen žádný vhodný zdroj (bugemos, který by jej mohl zajímat, je v grafu příliš daleho).

# 4. Použité technologie

Jako implementační jazyk aplikace jsme zvolili jazyk Java [7]. Tento výběr ovlivňuje volbu jednotlivých knihoven a frameworků, které nám budou poskytovat a zprostředkovávat různé technologie. Z povahy aplikace vyplývá, že většina netriviálních úloh bude probíhat na serverové části, tam také použijeme nejvíce knihoven.

## 4.1 Poskytovatel platformy

Pro běh serverové části potřebujeme server(y), na kterých poběží naše aplikace. Na výběr jsme měli možnost pronájmu vlastního (virtuálního) serveru, nebo využít služby některého z poskytovatelů Cloudu[1].

Virtuální server sice poskytuje největší svobodu, ale za cenu nutné vlastní konfigurace systému. Tomu jsme se chtěli vyhnout a zaměřit své úsilí na samotný vývoj aplikace. Další jeho nevýhodou je nemožnost automatického škálování. Až na výjimky[2] se za pronájem účtuje paušální měsíční poplatek.

Naopak cloudoví poskytovatelé nabízí hotovou platformu, na níž je možné rovnou začít stavět aplikaci. Serverovou část aplikace tedy budeme provozovat v cloudu, to nám umožní efektivní správu prostředků, možnost trvalé expanze a škálovatelnost aplikace. Pro volbu konkrétního cloudového poskytovatele jsme měli (v době zahájení práce) na výběr z následujících možností:

- Amazon EC2[3]
- Google AppEngine[4]

Jelikož jsme již měli zkušenost s posledním zmíněným, a tedy jeho architektura nám byla známá, zvolili jsme jej.

## 4.2 Platforma Google AppEngine

Google AppEngine je platforma pro vývoj a hostování webových aplikací napsaných v jednom z několika podporovaných jazyků. Je poskytována společností Google; je dostupná široké veřejnosti; pro nenáročné aplikace je zdarma.

Platforma umožňuje automatické škálování výkonu podle aktuální potřeby aplikace, tzn. pokud je aplikace hodně vytížená, automaticky spustí další instance. Výběr tohoto poskytovatele nám umožňuje splnit výkonnostní požadavky. Vzhledem k principu fungování této platformy, totiž že výkon se přiděluje podle aktuálních požadavků a účtuje se spotřeba systémových prostředků (počet volání metod poskytovaného API), je relativně levná pro nenáročné aplikace.

AppEngine je platforma, která poskytuje několik spolu vzájemně provázaných rozhraní. Čtyři z nich, které jsou pro naši aplikaci důležité, nyní krátce popíšeme.

[1] Cloud computing – model poskytování prostředků založený na internetu; obvykle je zpoplatněno využití služby namísto klasické platby za software

[2] Jedinou nám známou výjimkou je poskytovatel `http://4smart.cz/`

[3] Amazon EC2 – `http://aws.amazon.com/ec2/`

[4] Google AppEngine – `http://appengine.google.com/`

### 4.2.1 Datastore

Datastore[5] je databáze navržená společností Google s cílem výborné škálovatelnosti a extrémní rychlosti. Těchto dvou požadavků se jim podařilo docílit, nicméně za cenu jistých omezení, která mohou být svazující. Nejprve popíšeme, jak databáze funguje, a následně, jaké důsledky to obnáší.

Databáze nemá pevné schéma; obdobou tabulek z relačních databází je `kind` – druh. Každý záznam v databázi je nějakého druhu; všechny záznamy stejného druhu jsou seskupené a lze nad nimi provádět dotazy. Každý záznam může mít jiné atributy, příp. jiné datové typy atributů. Takže je teoreticky možné mít u stejného druhu odlišné typy záznamů; například záznam s atributem pojmenovaným `a`, který má řetězcovou hodnotu `"aaa"` a jiný záznam, který tento atribut vůbec nemá nebo se jedná o číslo `1` nebo se dokonce jedná o seznam dat. Datastore podporuje hodnotu atributů mnoha datových typů, některé z nich jsou neobvyklé, například geografický bod pro určení polohy na Zemi, či hodnocení v rozsahu 0–100. To jsou možnosti uložení dat, které databáze poskytuje.

Požadavek na vysoký výkon omezuje možnosti manipulací s daty a dotazování. Databáze umožňuje jen následující operace:

- získat záznam – podle klíče najde záznam v databázi a vrátí jeho aktuální hodnotu,
- uložit záznam – uloží záznam do databáze; přepíše původní hodnotu, pokud již v databázi takový záznam existoval,
- odstranit záznam – odstraní záznam z databáze; záznam je určen svým klíčem,
- vyhledat záznamy – vyhledá záznamy daného druhu, které vyhovují filtru seřazené podle některého atributu.

Popsané operace jsou obdobami operací známých z objektových databází. Nejvíce omezené jsou možnosti vyhledávání záznamů; uvedeme některá z nich:

- neexistují spojení tabulek,
- neexistují žádné funkce; agregační funkce musí uživatel draze emulovat,
- hodnoty atributů lze porovnávat jen s konstantami,
- dotaz smí obsahovat maximálně jednu nerovnost,
- prvním kritériem pro řazení musí být atribut s nerovností, pokud existuje,
- výsledky dotazu nemusí vždy vyhovovat filtru.

Všechna z popsaných omezení souvisejí se způsobem provádění dotazů. Každý dotaz, který se provádí, neprobíhá proti vlastním záznamům, ale proti jednomu či více indexům. To vlastně znamená, že výsledky všech možných dotazů, které kdy může uživatel databáze provést, jsou předpočítané. S tím také souvisí neaktuálnost dat v indexech, mohou být o několik sekund zpožděné oproti hodnotám vlastních záznamů. Samotný dotaz probíhá v několika krocích:

[5] Datastore – `http://developers.google.com/appengine/docs/java/datastore/`

1. nalezení nejlepšího indexu či nejlepší kombinace indexů. Index(y) musí pokrývat všechny atributy uvedené ve filtru; neexistence vhodného indexu způsobí běhovou chybu.
2. nalezení klíče v indexu prvního záznamu, který vyhovuje filtru,
3. lineární průchod následujících záznamů, dokud vyhovují filtru,
4. vrácení záznamů získaných podle klíče.

Jak je vidět, není v tomto algoritmu místo pro funkcionalitu, která by řešila výše uvedená omezení.

### 4.2.2 Task Queues

Task Queues[6] realizuje frontu úloh, které jsou postupně vykonávány. Uživatel může do fronty vložit najednou mnoho úloh a fronta se postará o to, že budou spuštěny se stejnou frekvencí, například spustí maximálně pět úloh za sekundu. Toto záměrné zpoždění je vhodné z důvodu snížení aktuální zátěže systému. Další výhodou jsou samostatné limity pro každou úlohu; nestane se, že by série úloh selhala z důvodu, že zpracování všech dohromady přesáhlo maximální časový limit.

### 4.2.3 Users

Platforma AppEngine poskytuje službu Users[7], která umožňuje přebírat informace o aktuálně přihlášeném uživateli; deleguje tak na sebe celou infrastrukturu okolo registrace, přihlašování, odhlašování, zapomenutého hesla. Pokud uživatel odsouhlasí, že aplikace má přístup k datům uživatele, může aplikace zjistit, jaký je aktuálně přihlášený uživatel pro každý webový dotaz.

Obvykle, pokud aplikace detekuje nepřihlášeného uživatele, reaguje zobrazením odkazu, který vede na přihlašovací formulář Google účtu. Jakmile jej uživatel vyplní, odešle a úspěšně se přihlásí, je přesměrován zpět do naší aplikace. Proces odhlášení probíhá analogicky, jen opačně.

### 4.2.4 Google Cloud Storage a Blobstore

Jelikož AppEngine poskytuje pouze platformu, na níž běží naše aplikace, není možný přístup k tradičnímu souborovému systému. Není to možné ani technicky: není jasné, na kolika, natož na kterých serverech na celém světě je aplikace právě spuštěná. Aby toto omezení platforma obešla, nabízí své rozhraní a abstrakci souborů.

Google Cloud Storage[8] umožňuje jak klasické vytváření a mazání souborů, tak poskytuje i rozhraní k přímému uploadu souborů uživateli aplikace. Doplňkem tohoto rozhraní je Blobstore[9], který poskytuje přístup k těmto souborům, umožňuje přímé streamování dat z úložiště.

[6] Task Queues – `http://developers.google.com/appengine/docs/java/taskqueue/`

[7] Users – `http://developers.google.com/appengine/docs/java/users/`

[8] Google Cloud Storage – `http://developers.google.com/appengine/docs/java/googlecloudstorageclient/`

[9] Blobstore – `http://developers.google.com/appengine/docs/java/blobstore/`

## 4.3 Slim3

Slim3[10] je framework, který zajišťuje mapování databázových entit postrádajících pevné schéma na reálné javové objekty. Je vyvíjen japonským vývojářem Yasuem Higem[11] a je poskytován pod open-sourcovou licencí Apache 2.0 [20].

Na rozdíl od jiných ORM[12] frameworků typu Hibernate[13], které mapování provádějí za běhu aplikace rozborem atributů objektu za pomoci javové reflexe, Slim3 předem vygeneruje meta třídy ke všem třídám modelu. Tyto třídy slouží k převádění reprezentace dat mezi objekty modelových tříd a Datastore entitami, poskytují metody `modelToEntity` a `entityToModel` a konstanty, které reprezentují atributy používané během dotazování.

## 4.4 ROME

ROME[14] je knihovna, která poskytuje sadu parserů a generátorů pro různé typy formátů zpravujících o novinkách na webovém zdroji. Jedná se o starší knihovnu, která se nezdá být aktivně vyvíjena, nicméně je běžně používaná a pro nás dostatečně robustní a stabilní; je dostupná pod open-sourcovou licencí Apache 2.0 [20].

Knihovna umí pracovat s většinou variant RSS a Atom dokumentů, umožňuje jejich stahování po síti, rozparsování i generování. Poskytuje jednotné rozhraní, přes které je možné dokumenty číst a vytvářet bez ohledu na jejich původní či výsledný formát.

## 4.5 Apache HttpComponents

Apache HttpComponents[15] je široce rozšířená knihovna určená pro zajištění komunikace přes HTTP[16]. Knihovna je vyvíjena Apache Software Foundation a je zdarma poskytovaná pod open-sourcovou licencí Apache 2.0 [20].

Knihovna poskytuje znovupoužitelné komponenty pro komunikaci klienta se serverem, řeší nejčastější požadavky, jako jsou autentikace, stavový management (Cookies), řízení spojení. Je široce využívaná a oblíbená na jedné straně pro jednoduchost svého použití a na druhé pro široké možnosti, které nabízí.

[10] Slim3 – `http://sites.google.com/site/slim3appengine/`

[11] Yasuo Higa – higayasuo@gmail.com

[12] ORM – Object-relational mapping – slouží k převádění dat mezi dvěma nekompatibilními systémy uložení; tradičně mezi objekty tříd a záznamy v relační databázi

[13] Hibernate – `http://www.hibernate.org/`

[14] ROME – `http://rometools.jira.com/wiki/display/ROME/Home`

[15] Apache HttpComponents – `http://hc.apache.org/`

[16] HTTP – Hypertext Transfer Protocol – bezestavový protokol určený ke komunikaci klienta se serverem při poskytování webových stránek

## 4.6 jsoup[17]

Knihovna jsoup[18] umožňuje rozparsovat HTML stránku do DOM[19] struktury. Vyvíjí ji Jonathan Hedley[20] a poskytuje ji zdarma pod licencí MIT.

Knihovna zvládá zpracování veškerých HTML stránek, i takových stránek, které nevyhovující standardům. V možnostech této knihovny je:

- stáhnout a rozparsovat HTML stránku z URL adresy, souboru či řetězce,
- najít a získat data pomocí metod DOM či CSS[21] selektorů,
- manipulace s HTML elementy, atributy a textem prostřednictvím metod, které poskytuje DOM, či CSS selektorů,
- ošetření dokumentu před nebezpečným obsahem,
- vyčištění HTML dokumentu – náprava chyb

Seznam vlastností jsme převzali ze stránek projektu [10].

## 4.7 JScience

Knihovna JScience[22] poskytuje implementaci různých výpočetních algoritmů používaných v matematice a fyzice. Knihovna je vyvíjena pod vedením vývojáře Jean-Marie Dautellem[23] a je poskytována zdarma pod vlastní open-sourcovou licencí.

Knihovna se skládá z několika modulů, které se zabývají efektivními algoritmy v následujících oblastech:

- výpočty s jednotkami a měnami,
- rigorózní zpracování algebraických objektů v javě,
- lineární algebra, výpočty s vektory a maticemi,
- symbolická matematická analýza: integrování, derivování funkcí, řešení rovnic, výpočet výrazů,
- a dalších.

Seznam vlastností jsme převzali ze stránek projektu [6].

[17] Název knihovny jsoup je tvořen malými písmeny, toto rozhodnutí autora respektujeme.

[18] jsoup – `http://jsoup.org/`

[19] DOM – Document Object Model – popisuje obsah HTML nebo XML dokumentů pomocí stromové struktury elementů; specifikace [13] definuje i metody pro přístup k elementům a manipulaci

[20] Jonathan Hedley – jonathan@hedley.net

[21] CSS – Cascading Style Sheets – jazyk určený pro popis vzhledu a formátování dokumentů [1]

[22] JScience – `http://jscience.org/`

[23] Jean-Marie Deutelle – jean-marie@dautelle.com

## 4.8 Diff Match Patch

Diff Match Patch[24] je malá knihovna, která nabízí robustní algoritmy pro synchronizaci textu. Knihovnu vyvíjí Neil Fraser[25] a nabízí ji zdarma pod licencí Apache 2.0 [20].

Knihovna poskytuje tři základní algoritmy:

- porovnání dvou textů – nalezne rozdíly mezi dvěma vstupními texty; umožňuje zadat jako parametr snahu, kterou má vykonat pro nalezení minimální změny. Výstupem algoritmu může být i HTML dokument, ve kterém jsou zvýrazněné přidané a odebrané části textu.
- nalezení nejpodobnější části textu – ve vstupním textu nalezne hledaný podřetězec, který se od vstupního textu nejméně liší. Výstupem je pozice – znak, kde hledaný podřetězec začíná.
- aplikace změn – snaží se aplikovat sadu změn na vstupní text. Výstupem je změněný text a informace o změnách, které se v pořádku provedly a které naopak selhaly.

## 4.9 Google Web Toolkit

GWT[26] je nástroj, který umožňuje vytvářet klientskou část aplikace ve stejném jazyku jako serverovou část. Knihovnu vyvíjí společnost Google, poskytuje ji zdarma pod open-sourcovou licencí Apache 2.0 [20].

GWT se skládá z několika komponent:

- kompilátor z Javy do JavaScriptu[27],
- zjednodušená implementace standardní knihovny Javy,
- vlastní knihovna, která umožňuje tvorbu síťových služeb, lokalizace a správu prostředků,
- knihovna grafických komponent: od jednoduchých obalů prvků HTML po komplexní panely rozložení.

Tyto komponenty dohromady umožňují naprogramovat téměř libovolnou klientskou část aplikace v Javě. Pokud by nabízené prostředky Javy nestačily, je možné části kódu naprogramovat v JavaScriptu.

Mezi výhody použití GWT v projektu patří:

- serverová i klientská část je psána ve stejném jazyku.
- společný kód pro serverovou a klientskou část se napíše jen jednou (a přeloží dvakrát),

[24] Diff Match Patch – `http://code.google.com/p/google-diff-match-patch/`

[25] Neil Fraser – root@neil.fraser.name

[26] GWT – `http://www.gwtproject.org/`

[27] JavaScript – formálně ECMAScript [12], interpretovaný jazyk běžící v internetovém prohlížeči, který umožňuje tvůrci webové stránky interaktivně manipulovat s jejím obsahem

- poskytuje základ pro komunikaci mezi klientskou a serverovou částí, stará se o serializaci a deserializaci požadavků,
- stará se o minifikaci kódu odeslaného klientovi,
- poskytuje hotové základní komponenty pro tvorbu uživatelského rozhraní.

### 4.9.1 Vaadin

GWT není jedinou knihovnou zaměřenou na prezentační vrstvu aplikace v prohlížeči. Nejbližší alternativou se nám jeví Vaadin[28] [9], který jsme také z počátku zvažovali. Je taktéž dostupný zdarma pod stejnou licencí: Apache 2.0 [20].

Nakonec jsme vybrali GWT, které nám přišlo jednodušší a pro naše účely vhodnější.

## 4.10 GWT Eye Candy

GWT Eye Candy[29] je drobná knihovna, která nabízí kolekci ovládacích prvků do uživatelského prostředí aplikace. Knihovna je určena jako rozšíření GWT, samostatně není použitelná. Vyvíjí ji Gabriel Axel[30] a poskytuje ji pod open-sourcovou licencí Apache 2.0 [20].

Knihovna obsahuje následující grafické prvky: tlačítka, přepínací tlačítka, nástrojovou lištu, titulek, výběr barev.

## 4.11 Crossrider

Platforma Crossrider[31] umožňuje vytvořit jediný doplněk, který bude fungovat shodně v nejrozšířenějších prohlížečích (Internet Explorer 7 a novější, Chrome, Firefox 3.6 a novější) [5]. Platforma nabízí vlastní API, které poskytuje následující vlastnosti:

- trvalou databázi klíč – hodnota,
- posílat libovolné webové dotazy bez omezení na původ,
- procházet a vytvářet záložky (bookmarks),
- umístit do prohlížeče na lištu tlačítko s ikonou,
- reagovat na klávesové zkratky,
- obsahuje knihovnu jQuery[32].

Tvorba rozšíření probíhá v online editoru na stránkách projektu; výstupem práce jsou doplňky instalovatelné do zmíněných prohlížečů.

[28] Vaadin – `http://vaadin.com/home`
[29] GWT Eye Candy – `http://code.google.com/p/gwt-eye-candy/`
[30] Gabriel Axel – guznik@gmail.com
[31] Crossrider – `http://crossrider.com/`
[32] jQuery – `http://jquery.com/`

## 4.12 TextExt

TextExt[33] je plugin do knihovny jQuery, poskytuje textovou komponentu, kterou lze přizpůsobit mnoha způsoby. TextExt vyvíjí Alexem Gorbatchevem[34] a je poskytován pod open-sourcovou licencí MIT.

Knihovna obsahuje jako jádro standardní HTML textové pole, které je obalené vrstvou JavaScriptu, která je zodpovědná za rozšiřující možnosti. Textové pole lze rozšířit několika funkcionalitami, například:

- vkládání štítků – umožňuje vkládat jednoslovné popisky, které jsou graficky zvýrazněné; ve výsledku budou oddělené čárkami,
- automatické doplňování – umožňuje nabízení vhodných doplnění textu na základě předem známého seznamu možností,
- zobrazení výzvy – zobrazí text na pozadí pole, který upozorňuje uživatele na možnost akce,
- tlačítko rozbalit – zobrazí všechny nabízené možnosti k doplnění.

[33] TextExt – `http://textextjs.com/`

[34] Alex Gorbatchev – alex.gorbatchev@gmail.com

# 5. Návrh

V této kapitole popíšeme návrh realizace požadavků, které jsme uvedli v kapitole Analýza.

## 5.1 Architektura aplikace

Aplikace se bude skládat ze tří částí; dvou těsně provázaných (serverová, klientská) a jedné volitelné (doplněk do prohlížeče).

Celek tvořený serverovou a klientskou části bude rozdělen do několika vrstev podle konceptu MVP[1].

### 5.1.1 Serverová část

Serverovou část postavíme na platformě Google AppEngine; využijeme mnoho možností, které nabízí. Pro realizaci některých specifických částí použijeme volně dostupné knihovny.

Serverová část bude obsahovat vrstvy: model a presenter.

#### Úloha serverové části

Serverová část bude mít v aplikaci následující úlohy:

- uložení všech dat,
- poskytování dat klientům,
- periodický sběr nových položek kontrolou zdrojů,
- výpočet doporučení zdrojů klientům na základě podobnosti.

#### Uložení dat

Všechna data, jak sdílená, tak i dedikovaná jednotlivým uživatelům, budeme ukládat do databáze Datastore, kterou poskytuje platforma AppEngine. Data budou modelována s pomocí frameworku Slim3, který použijeme jako obálku kolem samotné databáze. K vlastním datům v databázi budeme přistupovat pomocí sady služeb zaměřených podle typu objektů, se kterými pracují.

#### Rozhraní služeb

Pro komunikaci s klientskou částí navrhneme sadu rozhraní/služeb, přes která bude probíhat veškerá komunikace. Realizaci vlastní komunikace přenecháme frameworku GWT za účasti knihovny Slim3. Tyto služby budou mít za úkol kontrolovat požadavky klientů a volat příslušné služby komunikující přímo s databází.

[1]MVP – Model-view-presenter – členění aplikace na tři části, kde každá je zodpovědná za jednu úlohu: model reprezentuje data, view zobrazuje data, presenter předává požadavky uživatele a vrací výsledná data

#### Služby pro realizaci procesů

Naše aplikace bude vykonávat několik náročnějších procesů; mezi ně patří kontrola zdrojů, zálohování a výpočet doporučení.

**Kontrola zdroje** Při kontrole zdrojů využijeme frontu úloh – pro každý požadavek na kontrolu zdroje vytvoříme úlohu. Úloha kontroly zdrojů vyžaduje stažení dokumentu z webu: buď HTML, nebo XML. Pro stažení použijeme knihovnu HttpComponents.

Pokud je zdroj typu RSS/Atom, použijeme knihovnu ROME pro zpracování obsahu dokumentu. Pokud naopak zdroj pracuje s webovými stránkami, vytvoříme pomocí knihovny jsoup její stromovou reprezentaci, se kterou budeme dále pracovat. V případě, že zdroj je typu Změna stránky, použijeme knihovnu Diff Match Patch pro výpočet, určení změny.

**Zálohování položky** Při zálohování položky musíme nejprve stáhnout webovou stránku, kterou reprezentuje; využijeme opět knihovnu HttpComponents. Stránku následně upravíme (změníme všechny relativní odkazy a cesty k obrázkům) převedením do reprezentace DOM pomocí knihovny jsoup a provedením vhodných operací s elementy. Upravenou stránku uložíme do „souboru“ pomocí rozhraní poskytovaného službou Google Cloud Storage.

Zálohovanou stránku budeme poskytovat pomocí metod služby Blobstore, které nám umožní její přímé streamování.

**Exportování položek** Knihovnu ROME použijeme i pro opačný směr konverze, využijeme její generátory pro tvorbu RSS a Atom dokumentů při sdílení (exportu) položek.

**Výpočet doporučení** Algoritmus pro výpočet doporučení je realizovatelný posloupností operací provedených nad maticemi. Použijeme modul lineární algebry z knihovny JScience k výpočtům s vektory a maticemi.

### 5.1.2 Klientská část

Klientskou část uvidí uživatel při každé návštěvě webové stránky aplikace. Je tvořena drobnou kostrou napsanou v HTML, která je masivně rozšířená JavaScriptovým frameworkem, který zajišťuje dynamičnost a interaktivitu celé stránky. Vzhled stránky je definovaný v jazyku CSS.

Klientská část bude obsahovat vrstvu prezentéra a zobrazení.

#### Úloha klientské části

Uživatel ovládá celou aplikaci klientskou částí pomocí prvků grafického prostředí webové stránky. Přes ní manipuluje se všemi svými daty uloženými v serverové části aplikace. Mezi nejdůležitější úkony prováděné uživatelem patří:

- správa zdrojů: přidání a úprava zdroje, zastavení a obnovení sledování zdroje

- procházení seznamu položek, zobrazení originálu položky, přidání/odebrání štítku, zazálohování položky
- filtrování seznamu položek, změna kritérií filtru, správa filtrů
- nastavení ostatních vlastností aplikace

Jelikož serverová část aplikace bude vytvořena v jazyku Java, zvolili jsme pro vývoj klientské části nástroj, který umožňuje zkompilovat kód v jazyku Java do jazyku JavaScript. Tímto nástrojem je GWT.

Samotné knihovny GWT obsahují mnoho užitečných grafických prvků, nicméně neobsahují například dialog pro výběr barev. GWT Eye Candy je knihovna, která nám rozšíří nabídku grafických komponent. My z ní využijeme jediný prvek: ColorPicker pro výběr barvy štítků.

### 5.1.3 Doplněk do prohlížeče

Nepovinný doplněk má za úkol zvýšit komfort používání aplikace těm uživatelům, kteří mají možnost instalace doplňků do prohlížeče.

#### Funkce doplňku

Doplněk slouží k přidávání nové manuální položky do aplikace. Doplněk bude mít podobu tlačítka na liště v prohlížeči; kliknutím na tlačítko se objeví vyskakovací okno s předvyplněnou adresou a titulkem aktuální stránky. V případě, že uživatel vybral část textu na stránce, bude jím předvyplněn popis položky. Kliknutím na tlačítko „Odeslat“ se vytvoří v aplikaci nová manuální položka podle vyplněných údajů.

Doplněk vytvoříme na platformě Crossrider, která zajistí produkci výsledných doplňků do nejběžnějších prohlížečů. Z možností platformy využijeme jen zobrazení tlačítka v liště prohlížeče, komunikaci po síti, databázi k uložení hodnot a posílání zpráv mezi jednotlivými částmi doplňku.

Nejdůležitější částí doplňku je popup okno, zobrazené po kliknutí na tlačítko; obsah tohoto okna je běžná webová stránka, která bude obsahovat formulář s údaji o položce. Součástí formuláře je i seznam štítků, který budeme realizovat knihovnou TexExt, která bude poskytovat uživatelsky přívětivé rozhraní.

## 5.2 Entity

Entity jsou obecné struktury, se kterými aplikace pracuje. Cílem této kapitoly je entity popsat a vysvětlit úlohu, kterou mají v kontextu celé aplikace. Pro přehlednost jsme entity sdružili do oblastí, jichž se týkají.

### 5.2.1 Uživatelé a konfigurace

Oblasti uživatele a konfigurace obsahuje následující entity.

#### Uživatel

Uživatel je reprezentace osoby, která systém používá. Entitu uživatele přebíráme od poskytovatele serveru pomocí Users API.

#### Konfigurační položka

Konfigurační položka nemá nic společného s termínem položka ani entitou položka.

Jedna konfigurační položka definuje jeden parametr, který má vliv na vzhled nebo chování aplikace; lze rozlišit dva typy konfiguračních položek:

- konfigurace serverové části – závisí na ní chování serverové části,
- konfigurace klientské části – závisí na ní vzhled nebo chování klientské části. Hodnoty položek tohoto typu mohou záviset na konkrétním uživateli, který aplikaci používá.

### 5.2.2 Zdroje

Tato část seskupuje všechny entity, které souvisejí se zdroji položek, jejich reprezentací a zpracováním. Nejdůležitější dvojicí entit je zdroj a uživatelský zdroj: zdroj má význam pouze pro pravidelné kontroly existence nových položek, uživatelský zdroj popisuje zdroj z pohledu uživatele. Oblast zdrojů obsahuje tyto entity:

#### Zdroj

Zdroj reprezentuje každou webovou stránku nebo službu, kterou aplikace používá k získávání nových položek. Aplikace pravidelně kontroluje změny na zdrojích; pokud zaregistruje změnu, aplikace přidá odpovídající položky. Zdroj je abstraktní entita, od které jsou odvozeny jednotlivé typy zdrojů; obsahuje všechny informace nutné k tomu, aby zdroj mohl být periodicky kontrolován. Typy zdrojů jsou:

**Manuální zdroj** Manuální zdroj patří vždy jednomu konkrétnímu uživateli. Nereprezentuje žádnou internetovou adresu a sám o sobě neposkytuje žádné položky. Všechny položky, které tomuto zdroji patří, přidal sám uživatel pomocí některé ze služeb, které má k dispozici. Jde tedy o technický prostředek, jak naplnit funkcionalitu přidávání vlastních položek.

**RSS/Atom** RSS a Atom jsou formáty XML dokumentů, které popisují novinky, ke kterým došlo za poslední dobu na webovém portálu. Tento zdroj typicky obsahuje odkazy na články, které byly publikovány v nedávné minulosti.

**Webový rozcestník** Nahrazuje funkcionalitu RSS či Atom pro webové portály, které tyto kanály neposkytují. Vychází z jednoduchého pozorování, že úvodní stránka takového portálu obsahuje seznam nejnovějších článků a že položky převzaté z RSS či Atomu by odpovídaly odkazům na této stránce.

**Změna stránky** Reprezentuje webovou stránku, o jejíž změně obsahu chce být uživatel informován. Každá změna stránky způsobí vytvoření položky informující o změně.

#### Region

Region definuje pro typ zdroje Webový rozcestník a Změna stránky oblast stránky, která je z pohledu uživatele zajímavá. Region je určen výběrem oblastí stránky, které se mají kontrolovat a oblastí, které se mají při kontrole ignorovat. Motivací k pozitivnímu a negativnímu přístupu k jednotlivým částem je fakt, že stránky obsahují prvky, který se mění při každém dotazu (reklamy, informace o času generování stránky) nebo není zajímavý (hlavička a patička stránky, menu).

#### Uživatelský zdroj

Zavedeme entitu uživatelský zdroj, jež realizuje vztah mezi uživateli a zdroji. Několik uživatelů může sledovat stejný zdroj a každý uživatel většinou sleduje více zdrojů. Tato entita uchovává vlastní nastavení každého zdroje zvlášť pro každého uživatele. Počet aktivních uživatelských zdrojů určuje, politiku podle které bude probíhat jeho kontrola. Uživatel vždy pracuje s uživatelskými zdroji, od zdrojů, které jsou pod nimi, je odstíněn. Pokud je v uživatelském rozhraní zmíněn pojem zdroj, je jím myšlena tato entita; entita zdroje nemá pro uživatele praktický význam.

Uživatelský zdroj obsahuje entitu štítek, která reprezentuje informaci o příslušnosti položky ke zdroji. Této nepřímé vazby využíváme v dotazech do databáze, při kterých klademe na uživatelské položky kritérium, že musejí pocházet z konkrétního uživatelského zdroje. Více o tomto štítku a jeho funkci popíšeme v kapitole 5.2.4.

### 5.2.3 Položky

Oblast položek obsahuje následující entity.

#### Položka

Položkou v aplikaci myslíme jednu adresu webové stránky, která pochází z některého ze zdrojů nebo byla přidána uživatelem manuálně. Položka je závislá jen na zdroji, ze kterého pochází, vlastní personalizaci (přizpůsobení jednotlivým uživatelům) se zabývá entita uživatelská položka. Každý typ zdroje poskytuje položky jiného typu:

**Manuální položka** Manuální položka formálně patří k manuálnímu zdroji; jediným důvodem je zajištění konzistence rozhraní všech typů položek. Manuální položka reprezentuje libovolnou internetovou adresu spolu s titulkem a popisem. Manuální položka vzniká jen a pouze činností uživatele, který si ji sám přidá do aplikace. Motivací zavést manuální položku je umožnění uživateli:

- přidat vlastní informaci k libovolné stránce,
- zazálohovat libovolnou stránku,

- označit si libovolnou stránku k pozdějšímu přečtení.

**Položka typu článek** Položka typu článek je produkována při kontrole zdrojem typu RSS/Atom. Každému záznamu v XML dokumentu odpovídá jedna položka vytvořená v aplikaci při kontrole zdroje. Titulek a popis přebíráme spolu s dalšími informacemi přímo z RSS/Atom dokumentu. Položka typu článek může poskytovat více informací než jiné typy položek: jedná se především o jméno autora článku a datum publikace. Obecně tyto informace nejsou jednoduše zjistitelné v případě položek jiných typů.

**Položka typu odkaz** Položka typu odkaz pochází ze zdroje typu webový rozcestník. Každému nalezenému odkazu na webové stránce (uvnitř regionem omezené oblasti) odpovídá právě jediná položka. Titulek resp. popis přebíráme z textu odkazu resp. textu nejbližšího okolí odkazu.

**Položka typu změna stránky** Položka typu změna stránky pochází ze zdroje typu změna stránky. Při kontrole stránky (na kterou se odkazuje zdroj) se porovnává její textový obsah s verzí, kterou jsme si uložili při předchozí kontrole. Pokud je nalezena změna, aplikace vytvoří novou položku, která bude obsahovat popis změn stránky.

Položka je závislá jen na zdroji, ze kterého pochází; stejně jako v oblasti zdrojů existuje dvojice: zdroj – uživatelský zdroj, existuje i entita uživatelská položka.

#### Uživatelská položka

Entita položka obsahuje jen obecné – na uživateli nezávislé atributy, které jsou sdíleny všemi uživateli. Jsou jimi například: titulek, obsah, zdroj, ze kterého pochází, datum a čas přidání. Aby mohl každý uživatel mít položku ve vlastním stavu, s vlastními štítky, zavedeme entitu uživatelská položka. Tato entita uchovává nastavení vlastní každému uživateli, včetně zálohy, pokud byla vytvořena.

Pokud je v uživatelském rozhraní zmíněn pojem položka, je jím myšlena tato entita; entita položka nemá pro uživatele praktický význam.

Na obrázku 5.1 je zobrazen vztah nejdůležitějších entit z oblastí zdrojů a položek.

### 5.2.4 Štítky

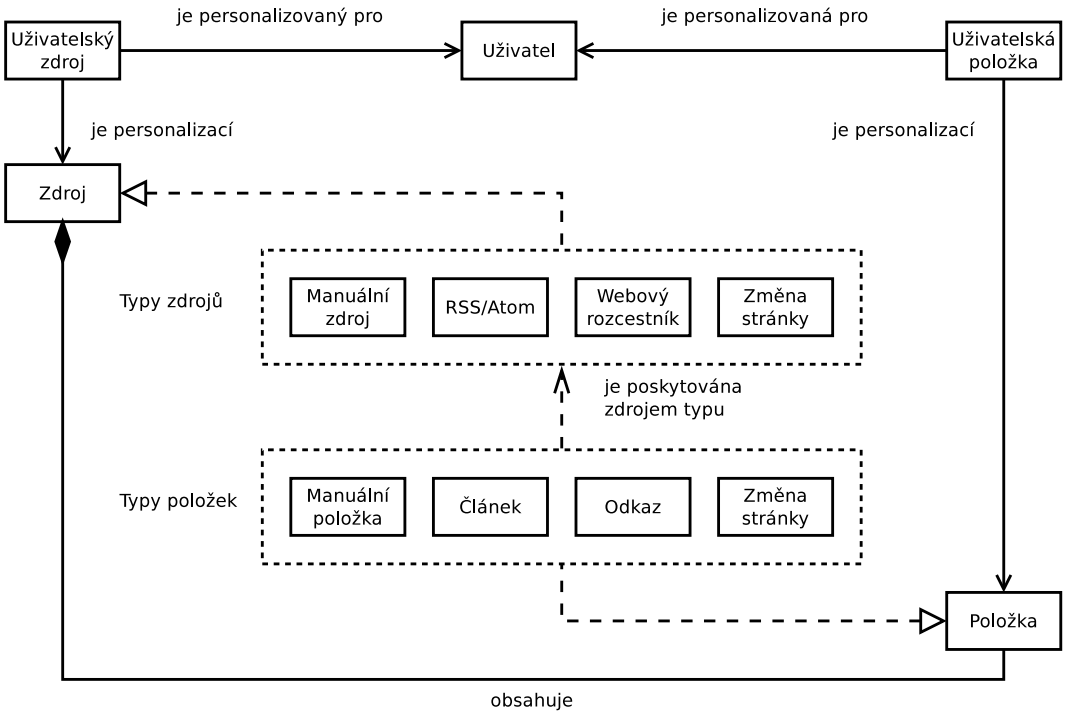
#### Seznam položek

Než zavedeme entity této oblasti, zadefinujeme pojem seznam položek.

Seznam (uživatelských) položek je podle času vzestupně či sestupně uspořádaný seznam všech uživatelských položek, které vyhovují všem kritériím, které jsou uživatelem na seznam kladeny. Seznam položek je výsledkem dotazu, neexistuje dokud není dotaz proveden.

Termín seznam položek používáme také pro komponentu grafického prostředí, která zobrazuje uživatelské položky, které jsou součástí seznamu položek, výsledku dotazu.

Oblast štítků obsahuje následující entity.




Obrázek 5.1: Vztahy mezi uživatelem, zdroji a položkami

### Štítek

Štítek reprezentuje informaci, kterou může uživatel přidat ke své uživatelské položce. Štítek má svého vlastníka, uživatele, který jej vytvořil; tento štítek není dostupný jiným uživatelům. Štítek má několik úloh:

- tvoří poznámku u uživatelské položky,
- lze filtrovat uživatelské položky mající přiřazený konkrétní štítek,
- přiřazením vhodného štítku k uživatelské položce se provede záloha webové stránky položky.

Abychom zajistili jednoduché používání štítků, zavedli jsme následující dvě omezení:

- Název štítku nesmí obsahovat bílé znaky a znak $. Bílé znaky na okrajích názvu se oříznou, bílé znaky uvnitř a znak $ způsobí chybu při vytváření/přiřazování štítku. Toto omezení vynucuje výstižná jednoslovná pojmenování štítků.
- Název štítku nelze později změnit. Toto omezení umožňuje využít název štítku jako část jeho klíče.

V aplikaci existují dva typy štítků:

1. uživatelský štítek – štítek, který vytvořil uživatel, je přiřazovaný uživatelem a je zobrazovaný v seznamu položek,

2. zdrojový štítek – štítek reprezentující uživatelský zdroj – tento typ štítku není spravován uživatelem, nemůže jej ani vytvořit ani přiřadit, nezobrazuje se v seznamu položek. Jde o technickou záležitost, která nám umožňuje zjednodušit některé dotazy do databáze.

**Seznam štítků u uživatelských zdrojů** Uživatel má možnost u svého uživatelského zdroje uvést seznam štítků, které mají být automaticky přidány ke každé uživatelské položce, která patří danému uživatelskému zdroji. V případě, že zdroj je typu manuální zdroj, nebudou štítky přidány k uživatelské položce automaticky, ale budou jen uživateli nabídnuty přednostně.

Každý uživatelský zdroj obsahuje právě jeden zdrojový štítek; tento štítek zároveň patří právě jednomu uživatelskému zdroji a je vždy součástí zmíněného seznamu štítků.

Můžeme říct, že existuje dvojí vazba mezi uživatelským zdrojem a jeho uživatelskými položkami:

1. přímá vazba uživatelské položky na uživatelský zdroj;
2. nepřímá vazba přes společný zdrojový štítek. Tuto vazbu využijeme dále při popisu entity seznamový filtr.

### Seznamový filtr

Seznamový filtr popisuje kritéria a způsob řazení seznamu položek. Řazení je vždy buď sestupně či vzestupně podle času přidání uživatelské položky. Možná kritéria jsou následujících typů:

- přečtená/nepřečtená uživatelská položka,
- nejstarší/nejnovější datum přidání uživatelské položky,
- filtr tvořený predikáty: „Uživatelská položka $X$ má přiřazený štítek $Y$“

Poslední uvedené kritérium je zásadní pro funkčnost celé aplikace a všech seznamů položek:

- seznam všech položek – filtr je prázdný
- položky mající konkrétní štítek – filtr obsahuje právě ten štítek,
- položky konkrétního uživatelského zdroje – filtr obsahuje štítek patřící uživatelskému zdroji,
- složitější filtry – předchozí dvě možnosti spojené pomocí logických operátorů `AND` a `OR`[2].

[2]Databáze má stanovené omezení na složitost dotazu: dotaz může obsahovat maximálně 30 větví disjunkce.

**Typy filtrů** V rámci aplikace rozlišujeme tři typy seznamových filtrů.

- obyčejný seznamový filtr – seznamový filtr, který si uživatel uložil a kdykoli si může zobrazit seznam položek, které mu vyhovují,
- exportovaný seznamový filtr – chová se shodně jako obyčejný seznamový filtr; umožňuje navíc zobrazit seznam položek jako RSS či Atom dokument komukoli mimo naši aplikaci,
- ad-hoc seznamový filtr – slouží pro okamžitou navigaci mezi různými seznamy položek v aplikaci, není trvale uložený. Ad-hoc seznamovým filtrem je například realizován seznam položek zobrazený po kliknutí na libovolný štítek.

### 5.2.5 Klávesové zkratky

V aplikaci existuje jediná entita, která popisuje klávesové zkratky.

#### Klávesová zkratka

Klávesová zkratka definuje chování aplikace, které nastane při stisku klávesy či kláves. Každý uživatel si může definovat svoji vlastní sadu klávesových zkratek. Rozlišujeme tři různé typy klávesových zkratek:

- zkratky, které přiřazují či odebírají štítky uživatelským položkám. Klávesová zkratka má vazbu na štítek, který se má k uživatelské položce přidat, pokud přiřazen není, či odebrat, pokud je již přiřazen.
- zkratky, které zobrazují seznam položek. Klávesová zkratka má vazbu na seznamový filtr, který se má použít pro zobrazení seznamu položek.
- zkratky, které vykonávají jednu z mnoha podporovaných akcí. Takovou akcí může být například přechod na následující položku, zrušení aktuálního filtru či zobrazení originální stránky položky.

## 5.3 Procesy

Na tomto místě popíšeme procesy, které manipulují s entitami. Z popsaných procesů je zřejmé, jak jsou jimi realizovány jednotlivé funkční požadavky na aplikaci.

### 5.3.1 Zdroje

Po přihlášení si uživatel může vytvořit své vlastní zdroje:

1. zjistí se, zda takový zdroj již existuje (URL a typ zdroje),
2. pokud neexistuje, vytvoří se nový zdroj příslušného typu,
3. vytvoří se uživatelský zdroj, který propojí zdroj s uživatelem.

Uživatel nebude moci vytvořit zdroj typu manuální zdroj; manuální zdroj se vytvoří automaticky při prvním přihlášení. Při každém přihlášení se zkontroluje, zda existuje; pokud neexistuje, vytvoří se výše uvedeným postupem.

Odstranění zdroje není možné z důvodu, že mohou existovat položky, které jsou na tento zdroj navázané. Uživatel může zrušit sledování zdroje změnou vlastnosti uživatelského zdroje; sledování zdroje může uživatel obnovit. Pokud bylo u zdroje zrušeno sledování, nebudou se nadále vytvářet uživatelské položky pro tento uživatelský zdroj. V případě, že neexistuje aktivní uživatelský zdroj pro zdroj, nebude se provádět jeho kontrola. K tomu může dojít dvěma způsoby:

- buď uživatel zrušil sledování zdroje,
- nebo během vytváření uživatelského zdroje byl objeven chybně zadaný údaj.

### 5.3.2 Položky

Aplikace pravidelně provádí kontrolu zdrojů; stáhne z internetu dokument odpovídající adrese zdroje a zpracuje jej. Pokud se při kontrole zdroje zjistí, že záznam ještě v aplikaci neexistuje, bude vytvořen. Postup vytvoření položky při kontrole:

1. vytvoří se položka podle typu zdroje,
2. zjistí se z databáze seznam všech aktivních uživatelských zdrojů pro daný zdroj,
3. vytvoří se nová uživatelská položka pro každý aktivní uživatelský zdroj.

Uživatel může do aplikace přidat webovou stránku jako novou položku. V případě, že uživatel přidává položku manuálně:

1. nalezne se uživatelský zdroj odpovídající uživatelovu manuálnímu zdroji,
2. vytvoří se manuální položka pro manuální zdroj nalezeného uživatelského zdroje,
3. vytvoří se uživatelská položka pro nalezený uživatelský zdroj.

### 5.3.3 Seznamy položek

Seznam položek je definován seznamovým filtrem – několika kritérii, kterým musí každá položka vyhovět. Ad-hoc seznamový filtr vzniká v aplikaci přirozeně akcemi uživatele, je jím realizován například výpis položek uživatelského zdroje. Uživatel může zobrazit seznam položek na základě svého ad-hoc seznamového filtru; takový seznamový filtr může dále pojmenovat a uložit. Uživatel si může zadefinovat libovolné množství vlastních seznamových filtrů, přeneseně i seznamů položek. Seznamový filtr může dále měnit, případně odstranit.

### 5.3.4 Štítky

Uživatel má možnost přidat k položce dodatečnou informaci prostřednictvím štítku; přidání štítku může probíhat i poloautomaticky (klávesovou zkratkou) či plně automaticky (uvedením štítku do seznamu automaticky přidávaných štítku v uživatelském zdroji).

Uživatel může na několika místech vybrat štítek ze seznamu existujících štítků, případně vytvořit nový, pokud žádaný štítek ještě neexistuje. Štítek lze odstranit, za předpokladu, že není nikde v aplikaci použit: položkou, seznamovým filtrem, či klávesovou zkratkou.

Pokud má štítek nastavenou vlastnost zálohování, jeho přiřazení uživatelské položce způsobí zazálohování webové stránky položky:

1. stáhne se webová stránka položky,
2. veškeré relativní odkazy na stránce se nahradí za absolutní,
3. uloží se soubor stránky do úložiště pomocí služby Google Cloud Storage a zapíše klíč do uživatelské položky.

### 5.3.5 Klávesové zkratky

Uživatel si může přiřadit klávesové zkratky k nejběžnějším úkonům. Klávesovou zkratku vytvoří uvedením klávesy či kombinace kláves, které spustí akci, ke které je klávesová zkratka přidána. Uživatel může klávesovou zkratku odebrat smazáním kombinace kláves, které ji spouští.

Jakmile je klávesová zkratka definována a uživatel stiskne příslušnou kombinaci kláves, spustí se příslušná akce. Některé akce mohou být závislé na kontextu, ve kterém se uživatel nachází; takovými zkratkami jsou především klávesové zkratky manipulující s aktuálně vybranou uživatelskou položkou v seznamu položek.

# 6. Dokumentace

V této části práce popíšeme funkci/implementaci nejdůležitějších částí aplikace. Tento popis je určen programátorům; má za cíl poskytnout dostatečný přehled o všech částech aplikace před tím, než se setká se zdrojovým kódem.

## 6.1 Členění aplikace

Jak jsme již popsali v předchozích kapitolách, aplikace se člení na tři hlavní části:

1. serverovou část,
2. klientskou část,
3. nepovinný doplněk do prohlížeče.

První dvě části jsou naprogramovány v jazyku Java a sdílejí některé části kódu – balík `cz.artique.shared`. Na obrázku 6.1 jsou zobrazeny jednotlivé balíky, postup kompilace a její výsledný produkt.

Serverová část závisí na všech balících; můžeme však říci, že závislost na balíku `cz.artique.client` je slabá a vynucená pouze použitým frameworkem GWT. Vyžaduje, aby rozhraní služby a jeho asynchronní varianta byly ve stejném balíku. Samotná rozhraní jsou sice implementována jen v balíku `cz.artique.server`, nicméně je nutné je mít přístupné v klientské části.

Sdílená část zdrojového kódu obsahuje především definici modelu – tříd reprezentujících strukturu dat v databázi podle návrhu entit. Jednotlivým entitám odpovídá jedna nebo více tříd.

Kromě modelu obsahuje sdílená část rozhraní implementovaná třídami modelu a pomocné třídy používané při komunikaci mezi serverovou a klientskou částí.

Doplněk do prohlížeče je vytvořen zcela odděleně od zbytku aplikace v jazyku JavaScript.

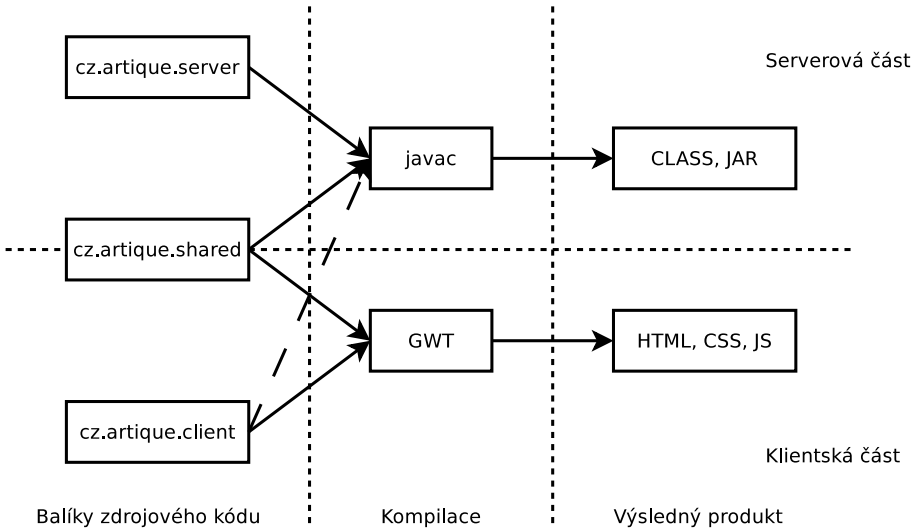
## 6.2 Rozhraní serverové části

Serverová část je zodpovědná za abstrakci přístupu k databázi, poskytování služeb klientům a vykonávání periodických procesů.

### 6.2.1 Obsluha databáze

Jelikož platformou poskytovaná databáze neumožňuje složitější operace, navrhli jsme kolekci služeb, které obalují přístup k databázi a provádějí jednoduché manipulace. Přístup k databázi ze všech částí aplikace směřujeme přes tyto služby. To nám umožňuje mít přehled o typech prováděných operací a zároveň eliminuje skryté závislosti. Nejčastějšími operacemi jsou:

- získání dalšího objektu podle klíče, například k uživatelskému zdroji doplní region,




Obrázek 6.1: Vztahy mezi zdrojovým kódem a výsledným produktem kompilace

- získání objektu, příp. jeho vytvoření, pokud neexistuje,
- sestavení složitějšího dotazu podle kritérií.

### 6.2.2 Rozhraní pro komunikaci s klientskou částí

Klientská část komunikuje po síti internet se serverovou částí pomocí několika služeb. Tyto služby lze rozpoznat podle jejich názvu začínajícího slovem `Client`.

#### Validace vstupu

Všechny služby, které jsou veřejně dostupné – mezi ně patří i služby volané z klientské části – vždy, bez výjimky, provádějí ošetření hodnot vstupů. Některé typy validace jsou prováděné z důvodu bezpečnosti, jiné z důvodu zajištění integrity dat či omezení danými použitými knihovnami a databází. Prováděné úpravy a kontroly hodnot jsou následující:

- vyplnění vypočitatelných hodnot; patří sem například:
  - výpočet databázového klíče,
  - dosazení typu z výčtu, pokud existuje jediná možnost,
  - dosazení aktuálního data do datumových atributů,
  - nahrazení vlastníka za aktuálně přihlášeného uživatele.
- kontrola, zda povinný atribut je vyplněný,
- kontrola, zda vlastník všech odkázaných objektů je aktuálně přihlášený uživatel,
- oříznutí bílých znaků z řetězců a následná kontrola délky (databáze povoluje řetězce pouze o maximální délce 500 znaků),
- oříznutí bílých znaků z delších textů,

- kontrola, zda URL je validní,
- kontrola, zda URL je validní a webová stránka jí určená je dostupná,
- kontrola, zda CSS selektor je validní,
- oříznutí bílých znaků z názvu štítku a kontrola, zda název štítku neobsahuje bílé znaky a znak $.

Pokud nebudeme uvažovat validaci, která samotná často zabírá několik desítek řádků kódu, zjistíme, že mnoho z dále popsaných služeb je triviálních. Tyto služby většinou spočívají jen v zavolání metody poskytnuté obsluhou databáze a vrácení jejího výsledku.

Dále popíšeme rozhraní všech služeb.

### Přihlášení, odhlášení

Služba `ClientUserService` poskytuje klientské části informace o aktuálně přihlášeném uživateli pomocí jediné metody – `login`. Pokud je uživatel přihlášen, vrací metoda objekt uživatele a internetovou adresu určenou k jeho odhlášení. Součástí objektu uživatele je i klientský token, který se používá při komunikaci s klientským doplňkem. Zároveň zkontroluje, jestli existuje manuální zdroj pro daného uživatele: pokud neexistuje (uživatel se přihlásil poprvé), vytvoří se. Naopak, pokud není uživatel přihlášen, vrací jen adresu určenou k přihlášení.

Informaci o aktuálně přihlášeném uživateli přebíráme ze služby `UserService`, kterou nám poskytuje platforma Google AppEngine.

### Kontrola spojení

Služba `ClientPingService` nemá při běžném chodu aplikace žádný význam, ten získá až v okamžiku, kdy dojde k chybě. Jakákoli služba pro komunikaci mezi klientskou a serverovou částí může selhat z nejrůznějších důvodů; jedním z důvodů je vypršení času určeného k vyčkání na odpověď. Pokud dojde k výjimce `RequestTimeoutException`, zkusí se, zda je serverová část vůbec dostupná pomocí metody `ping` této služby. Její implementace je záměrně triviální z důvodu vyloučení jiných možných chyb.

### Konfigurace

Služba `ClientConfigService` poskytuje klientské části metody pro získání seznamu všech konfiguračních položek pro aktuálního uživatele. V případě, že konfigurační položka v databázi neexistuje, vrátí se klientovi s výchozí hodnotou.

Dále tato služba umožňuje dávkovou aktualizaci hodnot několika konfiguračních položek. Novou hodnotu konfigurační položky uloží služba do databáze pro aktuálně přihlášeného uživatele.

Tato služba v současné době není plně využívána; navrhli jsme ji obecněji, abychom si zajistili možnost pozdějšího rozšíření aplikace.

### Zdroje

Veškeré manipulace jak se zdroji, tak i s uživatelskými zdroji probíhají přes službu `ClientSourceService`. Zvolili jsme integraci manipulace s několika entitami do jediné služby z důvodu, že klientská část nepracuje se zdroji téměř nikdy přímo, ale výhradně prostřednictvím uživatelských zdrojů. Tato služba poskytuje následující metody:

- přidat zdroj – vytvoří v databázi zdroj daného typu, pokud ještě neexistoval a vrátí jej. Tato metoda se volá v situaci, kdy uživatel vytváří svůj nový uživatelský zdroj: při vytváření uživatelského zdroje je nutná znalost zdroje, který je personalizován.
- přidat uživatelský zdroj – vytvoří v databázi uživatelský zdroj, který je personalizací existujícího zdroje. Tato metoda se volá při vytváření uživatelského zdroje po metodě přidat zdroj.
- aktualizovat uživatelský zdroj – aktualizuje v databázi existující uživatelský zdroj.
- získat regiony – vrátí seznam existujících regionů ke zdroji; zdroj musí být typu, který podporuje regiony. Tato metoda se volá v okamžiku načtení detailu uživatelského zdroje: buď po jeho vytvoření, nebo během jeho úpravy.
- zkontrolovat region – zkontroluje, zda region je platný. Tato metoda může být volitelně zavolána před aktualizací uživatelského zdroje; stejná kontrola se provádí během samotné aktualizace.
- naplánovat kontrolu zdroje – naplánuje okamžitou kontrolu zdroje. Mezi naplánováním a spuštěním kontroly může uběhnout časový interval definovaný frekvencí periodické kontroly zdrojů. Tato metoda je určena uživatelům pro případ, že se dozví jiným způsobem, že existují nové položky na zdroji dříve, než proběhne jeho řádná naplánovaná kontrola.
- získat doporučení – vrátí předpočítaný seznam doporučených zdrojů pro aktuálního uživatele. V situaci, kdy uživatel vytváří nový zdroj, si může vybrat, zda přidá zdroj podle URL adresy či si vybere některý z jemu doporučených.

### Štítky

Veškeré manipulace se štítky probíhají pomocí služby `ClientLabelService`. Jelikož jsme zavedli silná omezení na štítky, je tato služba spíše jednodušší:

- získat všechny štítky – vrátí všechny štítky z databáze, které patří aktuálně přihlášenému uživateli, bez ohledu na jejich typ.
- přidat štítek – vytvoří nový štítek v databázi, pokud štítek se stejným názvem ještě neexistuje a vrátí jej. Tuto metodu volá klientská část v situacích, kdy uživatel má možnost vybrat štítek a uživatel zadá název neexistujícího štítku.

- aktualizovat štítky – provede dávkovou aktualizaci v databázi několika štítků; nastavením atributu `toBeDeleted` lze štítek odstranit. Odstranění štítku je podmíněné tím, že není nikde použitý; v opačném případě odstranění selže.

### Seznamové filtry

Služba `ClientListFilterService` provádí veškeré manipulace se seznamovými filtry. Nabízí obvyklé metody:

- získat všechny seznamové filtry – vrátí z databáze seznam všech seznamových filtrů, které patří aktuálně přihlášenému uživateli.
- přidat seznamový filtr – uloží seznamový filtr do databáze.
- aktualizovat seznamový filtr – aktualizuje v databázi seznamový filtr.
- odstranit seznamový filtr – odstraní z databáze seznamový filtr.

### Položky

Nejkomplexnější a nejdůležitější službou, kterou serverová část poskytuje klientské části je `ClientItemsService`. Tato služba poskytuje 3 metody, které manipulují s položkami:

- Přidat manuální položku – vytvoří v databázi novou manuální položku a odpovídající uživatelskou položku k manuálnímu zdroji aktuálně přihlášeného uživatele.
- Získat položky – vrátí seznam uživatelských položek, které vyhovují seznamovému filtru. Pokud nejde o první dotaz, bude v dotazu uvedený i klíč poslední uživatelské položky, kterou klientská část zná; vráceny budou jen novější/starší uživatelské položky (závisí na způsobu řazení). V případě, že seznam má být řazený sestupně, bude výsledek obsahovat i seznam uživatelských položek, které jsou novější než klientské části známá nejnovější uživatelská položka. Výsledek obsahuje i informaci o tom, zda mohou existovat ještě další uživatelské položky – jestli existuje šance, že následující dotaz vrátí neprázdný výsledek.
- Aktualizovat položky – aktualizuje v databázi uživatelské položky na základě souboru změn, které se na ně mají aplikovat. Soubor změn udává, které štítky se mají uživatelské položce přidat či odebrat a jaký má být její nový stav (přečtená – nepřečtená). Metoda vrací seznam uživatelských položek, které byly změněny.

### Klávesové zkratky

Klávesová zkratka je jednoduchá entita, na níž není kladen požadavek složitější manipulace; vystačíme si s možností vytvoření a odstranění klávesové zkratky. Tato služba poskytuje následující metody:

- získat všechny klávesové zkratky – vrátí z databáze seznam všech klávesových zkratek aktuálně přihlášeného uživatele.
- vytvořit klávesovou zkratku – vytvoří v databázi klávesovou zkratku.
- odstranit klávesovou zkratku – odstraní klávesovou zkratku z databáze.

Změna klávesové zkratky je možná posloupností odstranění staré zkratky a následného vytvoření nové zkratky.

### 6.2.3 Rozhraní pro komunikaci s klientským doplňkem

Služba pro komunikaci s klientským doplňkem – `ClientServlet` vyžaduje přihlášení uživatele. Rozhodli jsme se přidělit každému uživateli náhodný jednoznačný identifikátor, který bude doplněk v prohlížeči používat při komunikovat se serverovou částí aplikace ke své identifikaci. Tento identifikátor – token – je určen jen a pouze pro komunikaci s doplňkem. Myslíme si, že jde o rozumný kompromis mezi zajištěním bezpečí a jednoduchostí implementace.

Rozhraní pro komunikaci s doplňkem nabízí dvě metody:

- získat štítky – vrátí z databáze seznam názvů všech uživatelských štítků. Funguje velice podobně jako metoda získat všechny štítky služby štítků pro komunikaci s klientskou částí. Liší se jen tím, že tato metoda filtruje štítky podle typu a vrací ještě seznam výchozích štítků pro manuální zdroj.
- přidat položku – přidá položku popsanou JSON[1] objektem k manuálnímu zdroji uživatele. Provádí podobné kontroly a úpravy jako metoda přidat manuální položky služby položek pro komunikaci s klientskou částí. Jelikož toto rozhraní je textové (nepřenáší se klíče štítků, ale jejich názvy), může uživatel přidělit položce i neexistující štítek. V takovém případě bude v okamžiku, kdy je zřejmé, že objekt položky prošel všemi kontrolami, vytvořen nový štítek a následně k uživatelské položce přiřazen.

Při použití tohoto klientského rozhraní musí klient poskytnout svůj klientský token a uvést akci (metodu), která se má provést.

### 6.2.4 Rozhraní pro plánování a plánované úlohy

Do této skupiny jsme zařadili služby, které jsou spouštěny:

- buď periodicky plánovačem Cron[2] (kontrola zdrojů, výpočet doporučení),
- nebo jsou realizovány pomocí úloh, které se postupně zpracovávají (kontrola zdrojů, zálohování položky)

[1] JSON – JavaScript Object Notation – textový formát určený pro přenos strukturovaných dat založený na syntaxi jazyku JavaScript

[2] Cron – plánovač, který spouští úlohy periodicky v předem určený čas

### Kontrola zdrojů

Naše aplikace vyžaduje pro získávání nového obsahu periodické spouštění kontroly zdrojů. Plánovačem Cron spouštěná služba nejprve zjistí seznam zdrojů, které je třeba zkontrolovat: to jsou takové, jejichž plánovaný čas kontroly je menší než aktuální čas, nenastalo u nich příliš mnoho chyb během poslední kontroly a zároveň nejsou právě kontrolovány. Pro každý nalezený zdroj se vytvoří úloha, která je následně vložena do fronty. Fronta úloh se stará o to, aby byly jednotlivé úlohy spouštěny postupně s určitou frekvencí, což snižuje okamžitou zátěž. Dalšími výhodami naplánování úloh oproti okamžitému spuštění kontroly zdroje jsou:

- možnost kontroly několika zdrojů zároveň – každá kontrola probíhá ve vlastním vlákně,
- maximální doba běhu skriptu se týká každé kontroly zvlášť, nehrozí tak vypršení časového limitu.
- možnost opakování kontroly zdroje v případě, že dojde k chybě.

Aby se aplikace lépe vypořádala s chybnými zdroji (nekontrolovala je nepřiměřeně často), po několika chybách při kontrole zastavíme jeho další kontroly v normálním režimu. Takový zdroj bude kontrolován v chybovém režimu, méně často, jen jednou za 12 hodin; v případě, že se následná kontrola zdaří, vrátí se zdroj do normálního režimu.

### Výpočet doporučení

Seznam zdrojů doporučovaných jednotlivým uživatelům přepočítáváme z důvodů, které jsme zmínili v kapitole 3.3, jen jednou za den. O pravidelné spouštění procesu se stará plánovač Cron poskytovaný platformou Google AppEngine.

Výpočet probíhá v několika krocích:

1. Načtení všech uživatelských zdrojů z databáze a vytvoření matice uživatelů a zdrojů. Matice má v řádcích uživatele, ve sloupcích zdroje. Prvek matice vyplníme:
   - buď jedničkou, pokud existuje uživatelský zdroj – uživatel sleduje zdroj,
   - nebo nulou v opačném případě.

   Tvorbou matice z uživatelských zdrojů si zajistíme přítomnost alespoň jedné jedničky v každém řádku i sloupci. Matici nazveme $M$.

2. Matici normalizujeme zvlášť v řádcích a sloupcích: každý řádek, resp. sloupec vydělíme jeho součtem. Tato normalizace nám zajistí rovnost vah všech uživatelů, resp. zdrojů (jinak by měl větší váhu uživatel s mnoha zdroji resp. zdroj sledovaný mnoha uživateli). Matici s normalizovanými zdroji pro uživatele nazveme $M_u$. Matici s normalizovanými uživateli pro zdroje nazveme $M_z$

3. Připravíme si čtvercovou matici popisující vhodnost uživatelů: jak jsou si uživatelé blízcí. Před začátkem výpočtu je tato matice jednotková; zatím neznáme blízké uživatele (reflexivita zajišťuje stabilitu výpočtu). Matici označíme $U_0$; $U = U_0$ bude matice používaná během výpočtu.

4. Provedeme krok výpočtu: aktualizujeme nejprve váhy zdrojům pro všechny uživatele: $Z = U \cdot M_u$ a následně aktualizujeme blízkost každých dvou uživatelů: $U = Z \cdot M_z + U_0$.

5. Několikrát provedeme iteraci 4. kroku výpočtu.

6. Matice $Z$ obsahuje váhy popisující vhodnost zdroje pro každého uživatele. Matici budeme procházet po řádcích (pro uživatele); setřídíme zdroje podle vah sestupně a vyfiltrujeme ty, které uživatel již sleduje (jsou v matici $M$). Ze seznamu vezmeme prvních $k$ doporučených zdrojů a výsledný seznam uložíme pro každého uživatele do databáze.

#### Zálohování položky

Kdykoli je nějaké uživatelské položce přiřazen štítek, zkontroluje se, zda štítek má nastavenou vlastnost zálohování. Pokud ji má, vytvoří se nová zálohovací úloha, která se zařadí do fronty. Stejně jako v případě kontroly zdrojů, i zde se fronta úloh stará o jejich postupné spouštění. Postup zálohování jsme popsali v části 5.3.4

### 6.2.5 Veřejné rozhraní k obsahu

Některá rozhraní aplikace jsou dostupná i neregistrovaným uživatelům

#### Exportování položek

Toto rozhraní poskytuje seznam položek vyhovujících seznamovému filtru, u kterého jeho vlastník povolil možnost exportování. Export je identifikovaný kombinací aliasu a uživatelského jména vlastníka seznamového filtru. Služba zodpovědná za výběr položek z databáze je stejná jako v případě poskytování seznamu položek klientské položky se všemi svými důsledky. Seznam položek je vrácen buď ve formě RSS, nebo Atom (záleží na parametru poskytnutého v URL).

#### Zálohované položky

Připomeňme, že během zálohování stáhneme webovou stránku, mírně ji upravíme a uložíme; záloze je přidělen klíč, který ji identifikuje. Odkaz na zálohovanou stránku (přístupnou přes toto rozhraní) je dostupný uživateli, který zálohu provedl ze seznamu položek.

## 6.3 Členění klientské části

Doposud jsme se příliš klientskou části nezabývali z důvodu, že klientská část přímo neprovádí žádné manipulace s daty; využívá pouze prostředků, které jí

poskytuje serverová část. Přesto můžeme klientskou část rozčlenit do několika částí (spíše podle funkce než podle balíků):

- realizace hierarchií a stromových komponent,
- správci, kteří komunikují se serverovou částí a ostatní správci,
- komponenta nekonečného seznamu,
- dialogy, pomocí kterých se vytvářejí a upravují všechny objekty.

### 6.3.1 Hierarchie

U objektů, které jsou součástí hierarchie, ukládáme v databázi cestu ke kořeni. Z toho důvodu je nutné pro zobrazení stromu primárně vlastní stromovou strukturu rekonstruovat. Všichni správci, kteří poskytují hierarchii, musejí každý objekt, který spravují, zpracovat a vložit do uměle vytvořené hierarchie. Postup vypadá následovně:

1. vytvoření kořenového uzlu,
2. pro všechny spravované objekty:
   2.1. rozdělení cesty podle lomítek (oddělovače jednotlivých úrovní),
   2.2. vytvoření uzlů pro jednotlivé úrovně, pokud ještě neexistují,
   2.3. vytvoření listu v nejnižší úrovni; jeho hodnotou bude příslušný objekt,
3. vrácení kořenového uzlu.

Uzly se stejným rodičem jsou řazeny podle abecedy.

Z důvodu propagace změn, všechny uzly umožňují zaregistrování posluchače událostí změn v hierarchii: uzel byl přidán, odebrán, název uzlu se změnil.

#### Stromové komponenty

Klientská část aplikace obsahuje tři komponenty, které umožňují zobrazit hierarchie v podobě stromu; jsou to:

- strom zdrojů – hierarchii poskytuje správce zdrojů,
- strom seznamových filtrů – hierarchii poskytuje správce seznamových filtrů,
- strom zpráv – hierarchii poskytuje správce zpráv.

Zprávy ve skutečnosti netvoří strom, ale prostý seznam. Využili jsme stromovou komponentu pro jejich zobrazení z několika důvodů:

- poskytuje všechny možnosti, které vyžadujeme,
- vzhled seznamu zpráv je blízký vzhledu stromů zdrojů a stromu seznamových filtrů,
- zjednodušuje výslednou implementaci.

### 6.3.2 Správci

Téměř každé službě, kterou jsme popsali z pohledu serverové části, odpovídá správce v klientské části. Správce má za úkol zprostředkovávat veškerou komunikaci se serverovou částí a poskytovat nakešovaná data. Každý správce poskytuje klientské části rozhraní k metodám služby, kterou obaluje. Tento obal většinou spočívá v zavolání příslušné metody služby a ve většině případů zobrazení zprávy o výsledku. Informaci o úspěchu zobrazuje v případě, že volání metody bylo vyvoláno akcí uživatele. V případě neúspěchu zobrazíme zprávu vždy; ta obsahuje informaci, proč volání selhalo, například z důvodu nevyplnění některého z atributů. Navíc se zavolá služba kontroly spojení pro zjištění, zda je vůbec dostupné spojení se serverovou částí.

V popisu jednotlivých správců budeme zmiňovat pouze metody, které provádějí netriviální další akce.

#### Správce konfigurace

Správce konfigurace tvoří pouze obálku kolem vlastní služby. V okamžiku spuštění si stáhne veškerou konfiguraci relevantní pro přihlášeného uživatele. Jednotlivé dotazy na konfigurační položky pak může zodpovídat okamžitě bez nutnosti komunikace se serverovou částí.

Správce konfigurace umožňuje rovněž i aktualizovat hodnoty konfiguračních položek, nicméně tuto možnost naše aplikace v současné době nevyužívá.

#### Správce zdrojů

Správce zdrojů načte ze serveru seznam všech uživatelských zdrojů přihlášeného uživatele. Jednotlivé uživatelské zdroje uloží do několika slovníků, aby byly rychle vyhledatelné: podle klíče uživatelského zdroje a podle zdrojového štítku. Ze všech uživatelských zdrojů vybuduje hierarchii. Správce zdrojů si pamatuje, který zdroj je manuální (vždy existuje právě jeden).

Jelikož správce zdrojů poskytuje přístup k hierarchii uživatelských zdrojů, musí reagovat na určitě události. Jakékoli úspěšné volání metody přidat uživatelský zdroj či aktualizovat uživatelský zdroj může způsobit změnu hierarchie. Správce zdrojů ve zmíněných metodách případně upravuje hierarchii uživatelských zdrojů.

#### Správce štítků

Správce štítků obsahuje seznam všech štítků rozdělených podle typů: uživatelský štítek, zdrojový štítek, systémový štítek. Existují dva systémové štítky, oba reprezentují operátory: AND – operátor *a*, OR – operátor *nebo*. Zbylé dva seznamy štítků (uživatelských a zdrojových) jsou naplněny dotazem na serverovou část při prvním spuštění správce. Každému štítku je nastaven zobrazovaný název: u uživatelských štítků je to jejich vlastní název, u zdrojových je to název zdroje, ke kterému štítek patří.

Správce štítků poskytuje metodu pro vyhledávání štítků podle části jejich názvu. Tuto metodu používá komponenta, která nabízí štítky v situaci, kdy uživatel může vložit název štítku (například, pokud se přidává štítek uživatelské položce).

V případě, že uživatel přidá novou položku doplňkem v prohlížeči a přiřadí jí dosud neexistující štítek, dostane se klientská část do zvláštní situace. Seznam uživatelských položek by měl obsahovat uživatelskou položku se štítkem, o kterém nemá žádnou informaci. Samotná uživatelská položka zná pouze klíč štítku a správce štítků by jí měl objekt štítku dodat. V takovém případě se aktualizace seznamu položek pozastaví a bude pokračovat, až si správce štítků obnoví seznam všech existujících štítků dotazem na serverovou část.

### Správce seznamových filtrů

Správce seznamových filtrů poskytuje hierarchii všech seznamových filtrů, kterou si vytvoří po svém spuštění. Dále poskytuje metody pro manipulaci se seznamovými filtry, které v případě úspěchu upraví hierarchii. Těmito metodami jsou: přidat seznamový filtr, aktualizovat seznamový filtr a odstranit seznamový filtr.

### Správce položek

Správce položek poskytuje transparentní rozhraní k metodám: získat položky a přidat manuální položku. Poslední metoda příslušné služby – aktualizovat položky – není uživateli přímo dostupná.

Správce položek si udržuje seznam změn jednotlivých uživatelských položek, které se mají provést v databázi na straně serveru. Uživatel tento seznam změn vytváří a upravuje metodami:

- štítek přidán – přidá štítek (jeho klíč) do seznamu štítků k přidání v sadě změn příslušející uživatelské položce,
- štítek odebrán – přidá štítek (jeho klíč) do seznamu štítků k odebrání v sadě změn příslušející uživatelské položce,
- přečtena – nastaví stav přečtení v sadě změn příslušející uživatelské položce.

Změna v grafickém rozhraní se provede okamžitě, odeslání změny na server se odloží. V případě posloupnosti přidání a odebrání stejného štítku od uživatelské položky se obě změny vyruší. Periodicky, každých několik sekund probíhá dotaz obsahující seznam všech sad změn mezi správcem položek a službou serverové části. Toto „zpoždění“ provedení změny v databázi nám umožňuje snížit počet dotazů po síti a zvýšit subjektivní rychlost odezvy aplikace.

### Správce klávesových zkratek

Správce klávesových zkratek si při svém spuštění načte pomocí serverové služby seznam všech klávesových zkratek, které si uživatel sám definoval. Kombinace kláves klávesových zkratek rozparsuje a klávesové zkratky vloží do slovníku. Dále zaregistruje posluchače událostí stisknutí klávesy: události `KeyPress` a `KeyDown`. Pokud je detekováno stisknutí kláves odpovídající nějaké zkratce, vyvolá správce událost `ShortcutEvent`, pomocí které informuje o klávesové zkratce své posluchače. Sám správce klávesových zkratek zpracovává některé obecné typy zkratek: navigace mezi seznamy položek, jejich obnovu a podobné.

Správce poskytuje metody vytvořit klávesovou zkratku a odstranit klávesovou zkratku. Tyto metody v případě úspěchu upraví slovník klávesových zkratek.

Některé typy klávesových zkratek se odkazují na jiné objekty (štítky a seznamové filtry). Tyto klávesové zkratky je možné vyhledat na základě klíče objektu, na který se odkazují. Využíváme toho v dialozích s nastavením seznamových filtrů a dialogu s nastavením štítků.

Zbylí dva správci nejsou závislí na službách serverové části; spravují objekty, které mají význam pouze v kontextu klientské části.

#### Správce zpráv

Zpráva je krátký text o různé důležitosti, který slouží k informování uživatele o úspěchu či selhání některé akce.

Správce zpráv je centrálním bodem v naší aplikaci pro zpracování zpráv. Poskytuje rozhraní klientské části aplikace, přes které může kterákoli komponenta zobrazit zprávu. Správce zpráv také poskytuje seznam posledních zpráv v podobě hierarchie.

Správce sám zprávy nezobrazuje, ale umožňuje zaregistrovat posluchače, kteří mají být zpraveni o nových zprávách. Jediným posluchačem je vlastní grafická komponenta zodpovědná za zobrazování zpráv.

#### Správce historie

Framework GWT obsahuje podporu pro sledování historie v rámci aplikace. Jednou položkou v historii je token – řetězec reprezentující aktuální stav aplikace. V našem případě stav aplikace odpovídá zobrazenému seznamovému filtru.

Náš správce historie rozšiřuje tento mechanismus o několik důležitých vlastností. Pro nás je užitečnější znalost samotného seznamového filtru než jeho serializované varianty; z toho důvodu si ukládáme do seznamu stavů dvojici token – seznamový filtr a nižší vrstvě (GWT) předáváme token. Při reakci na událost změna historie, kterou vyvolá GWT reagujeme změnou aktuálního seznamového filtru a vyvoláním vlastní události `HistoryEvent`, pomocí které informujeme příslušnou část aplikace o změně.

### 6.3.3 Nekonečný seznam

V této části si pod pojmem položka představíme jeden záznam v seznamu, který je reprezentován uživatelskou položkou.

Nejdůležitější grafickou komponentou, která tvoří naši aplikaci, je nekonečný seznam položek. Nazýváme ho nekonečným, neboť nemá konec – jakmile uživatel odroluje seznam ke konci, načtou se další položky; takto jich může postupně zobrazit stovky. Samozřejmě, že v případě omezeného množství položek, vlastní konec seznamu existuje, avšak může být prakticky nedosažitelný. Výhodou tohoto přístupu je možnost stahovat ze serverové části pouze data, o která má uživatel právě zájem.

Položky, které jsou v seznamu zobrazené, jsou dodávané poskytovatelem, jenž reaguje na událost odrolování blízko konce seznamu zavoláním metody získat položky správce položek. Kdykoli se změní aktuální seznamový filtr, zruší se původní poskytovatel a nahradí se novým.

Nekonečný seznam je posluchačem událostí `ShortcutEvent` správce zkratek; reaguje na zkratky typu: přesun na předchozí/další položku, zobrazit originální webovou stránku, přidat/odebrat štítek a podobné.

Každý řádek reprezentuje jednu uživatelskou položku se všemi jejími informacemi: zdroj ze kterého pochází, datum a čas přidání položky, seznam přiřazených štítků a hlavně titulek. Obsah položky (text nebo HTML) se zobrazí pouze v případě, že položka je vybraná a rozbalená.

### 6.3.4 Dialogy

Manipulace s daty v aplikaci probíhá v rámci dvou oblastí: v nekonečném seznamu a přes dialogy. V nekonečném seznamu může uživatel provádět následující operace:

- označení uživatelské položky jako přečtené,
- přidání/odebrání štítku uživatelské položce, příp. vytvoření nového štítku, pokud zadá neexistující název.

Všechny ostatní operace provádí uživatel prostřednictvím dialogů – oken, která dočasně překryjí stránku aplikace. Každá oblast, kterou jsme dříve zmínili, s výjimkou položek a konfigurace, nabízí jeden či více dialogů pro svoji konfiguraci:

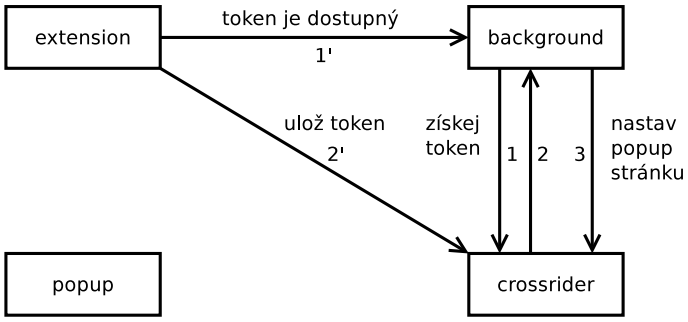
- vytvoření a úpravu zdroje,
- vložení manuální položky,
- vytvoření a úpravu seznamového filtru, úpravu ad-hoc seznamového filtru,
- úpravu a odstranění klávesových zkratek,
- tvorbu klávesové zkratky typu akce,
- úpravu a odstranění štítku.

Každý z těchto dialogů má jinou grafickou podobu závislou na atributech spravované třídy. V případě, že uživatel změnu provedenou v dialogu potvrdí, zavolají se metody příslušných správců, které změnu delegují na služby serverové části.

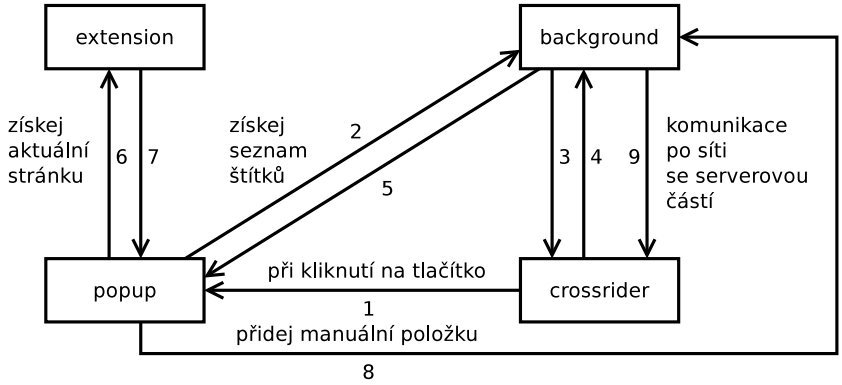
## 6.4 Doplněk do prohlížečů

Doplněk do prohlížeče je nepovinné rozšíření aplikace, které má za cíl zjednodušit její ovládání uživatelům. Implementace doplňku spoléhá na API poskytované platformou Crossrider, na níž je postaven. Doplněk můžeme rozdělit do tří částí, které spolu komunikují; pro jednoduchost je budeme pojmenovávat terminologií platformy:

- background – skript, který běží v prohlížeči na pozadí; je nezávislý na otevřených stránkách; existuje vždy v jediné instanci,




Obrázek 6.2: Postup inicializace doplňku




Obrázek 6.3: Postup zobrazení popupu a vytvoření manuální položky

- extension – skript, který se vykonává zvlášť v rámci každé otevřené stránky
- popup – html stránka, která se zobrazí při kliknutí na ikonku doplňku v liště prohlížeče; mezi dvěma zobrazeními se stav nezachovává.

### 6.4.1 Inicializace doplňku

Doplněk se inicializuje při spuštění prohlížeče vykonáním background skriptu; postup je znázorněn na obrázku 6.2. Nejprve zkontrolujeme, zda známe klientský token (1, 2); pokud ho známe, nastavíme akci zobrazení popup stránky při kliknutí na ikonku (3). V případě, že token neznáme, budeme čekat na zprávu o dostupnosti tokenu (1', 2'). Až token budeme znát, nastavíme popup stránku (kroky 1, 2, 3).

Skript extension zjišťuje podle URL adresy aktuální stránku; pokud se jedná o stránku naší aplikace, vyhledá v hlavičce stránky elementu meta s názvem `clientToken`. Tento meta element do stránky přidá klientská část aplikace při své inicializaci.

### 6.4.2 Přidání manuální položky

Postup je znázorněn na obrázku 6.3. Uživatel vyvolá akci vložení aktuální stránky do naší aplikace kliknutím na ikonu doplňku v liště prohlížeče (1). Zobrazí se vyskakovací (popup) okno s formulářem, který bude předvyplněný informacemi o aktuální stránce: titulek, URL, vybraná část textu. Informace se získají zasláním

zprávy (6) skriptu běžícímu v aktuálně otevřené stránce; skript zašle zpět (7) požadované údaje.

Formulář bude dále obsahovat předvyplněný seznam štítků a umožní přidat další štítky. Popup zašle zprávu (2) background skriptu, který odešle dotaz (3) na serverovou část pomocí metody `post` poskytované platformou Crossrider. Background skript odpověď (4) zašle zpět (5) do popup okna.

Pokud uživatel odešle formulář, odchytí skript událost odeslání a zašle background skriptu zprávu (8) s vyplněnými údaji o manuální položce, kterou chce uživatel přidat. Background skript odešle dotaz (9) serverové části pomocí metody `post`. Využití této metody je nutné z důvodu obejití některých omezení prohlížeče: skript může kontaktovat jen stránky ze stejné domény.

## 6.5 Diskuse

### 6.5.1 Možná rozšíření

Vzhledem k šíři problematiky, kterou se naše aplikace snaží pokrýt, je zřejmé, že nemůže v této verzi obsahovat veškeré vlastnosti a funkce, které by nás mohly dále napadnout. Aplikace srovnatelného rozsahu vytvářejí celé týmy zkušených vývojářů. Snažili jsme se proto především vytvořit pevné základy, na kterých bude možné dále pokračovat.

Uvedeme zde některá témata rozšíření, která by možnosti naší aplikace mohly výrazně posunout dopředu.

#### Import a export zdrojů

Import by umožnil snazší příchod nových uživatelů. Již existuje standardizovaný formát OPML[3], který používají všechny běžné čtečky k usnadnění příchodu nového uživatele či odchodu stávajícího.

#### Pravidla přiřazování štítků

Jde o rozšíření současného systému; nyní může uživatel nastavit štítky přiřazované ke každé položce některého zdroje. Uživatel by měl možnost vytvoření pravidel, která by rozhodovala o automatickém přiřazení štítku při vytváření položky.

#### Přizpůsobení uživatelského rozhraní

Umožnilo by to používat naší aplikaci na mobilních telefonech a tabletech. Podle velikosti a typu zařízení se zvolí vhodné rozložení uživatelského rozhraní aplikace.

#### Nastavení vzhledu a chování štítku

Pokud umožníme nastavit štítek tak, aby byl zobrazený, i když není přiřazený (odlišným stylem) a přidáme možnost nahrazení textu štítku ikonkou, bylo by možné docílit funkcionality známé z konkurenčních čteček: přepínání mezi prázdnou a plnou hvězdičkou.

[3]OPML – Outline Processor Markup Language – XML formát popisující stromové struktury [21]

#### Klávesová makra

Pokud by bylo možné provést více akcí se štítky najednou, šlo by například přepínat mezi několika štítky jedinou klávesou. Mohli bychom tím simulovat prioritu nebo důležitost položky.

#### Stav přečtení

Během návrhu aplikace jsme zvolili reprezentaci stavu přečtení uživatelské položky jako zvláštního atributu. Vzhledem k obecnosti dotazů, které do databáze pokládáme, je možné tento stav reprezentovat lépe dvojicí štítků „přečtená“ a „nepřečtená“. Výhody plynoucí ze změny by byly dvě: samotné zjednodušení modelu, poskytnutí více možností při definici seznamových filtrů.

### 6.5.2 Problémy při implementaci

Během implementace jsme se potýkali s několika problémy, které bychom zde chtěli krátce zmínit.

#### Kurzory dotazů do databáze

Kurzor je ukazatel na poslední záznam seznamu výsledků databázového dotazu. Chtěli jsme jej využít při realizaci nekonečného seznamu v situaci, když se uživatel přiblíží k jeho aktuálnímu konci. Tehdy se dotazujeme na uživatelské položky v databázi, které následují po položkách získaných minulým dotazem.

Použitá databáze sice umožňuje kurzory při dotazování, ale jejich omezení vylučuje použití v naší aplikaci. Fungují jen v případě, že dotaz neobsahuje logický operátor `OR`, ani operátor výčtu `IN`; dotazy, jež pokládáme však tyto operátory obsahují.

Naším řešením je využít uspořádání na klíčích položek: novější uživatelské položce přiřadíme vyšší hodnotu klíče, resp. přiřadí ji databáze z rostoucí sekvence. Každý náš dotaz bude obsahovat podmínku ostré nerovnosti klíče v databázi a klíče poslední uživatelské položky získané minulým dotazem.

#### Nekonečný seznam

Nekonečného seznamu se týká i druhý problém, se kterým jsme se potýkali. Podle přehledu grafických komponent, které knihovna GWT poskytuje, jsme usoudili, že potenciálně nekonečný seznam můžeme realizovat komponentou `CellList`. Neuvědomili jsme si však plně důsledky její implementace – buňka seznamu není widget, není možné uvnitř buňky mít aktivní prvky, může obsahovat jen prostý kus HTML kódu. Náš návrh ale vyžaduje možnost rozbalení a skrytí obsahu uživatelské položky, zobrazení dynamického seznamu štítků.

Z důvodu tohoto omezení jsme museli naprogramovat svoji vlastní komponentu prakticky zcela od začátku. Obdobně jsme museli vyvinout několik dalších komponent; jednou z nich byla například komponenta nabízející výběr štítku ze seznamu.

GWT nás svojí nabídkou grafických komponent zklamalo, očekávali jsme jich více a kvalitnějších. Několikrát jsme objevili chyby a byli jsme nuceni hledat řešení na fórech.

# Závěr

Výsledkem naší práce na tomto projektu je aplikace, která je, dle našeho názoru, schopná nabídnout uživatelům vlastnosti, které se od čtečky očekávají. Aplikace v sobě navíc kombinuje několik dalších služeb, díky nimž je unikátní; nejedná se „jen o další čtečku“. Z tohoto pohledu považujeme náš cíl za splněný.

Čtečku jsme navrhli a vytvořili především tak, aby odpovídala našim požadavkům. Aplikace nabízí následující možnosti:

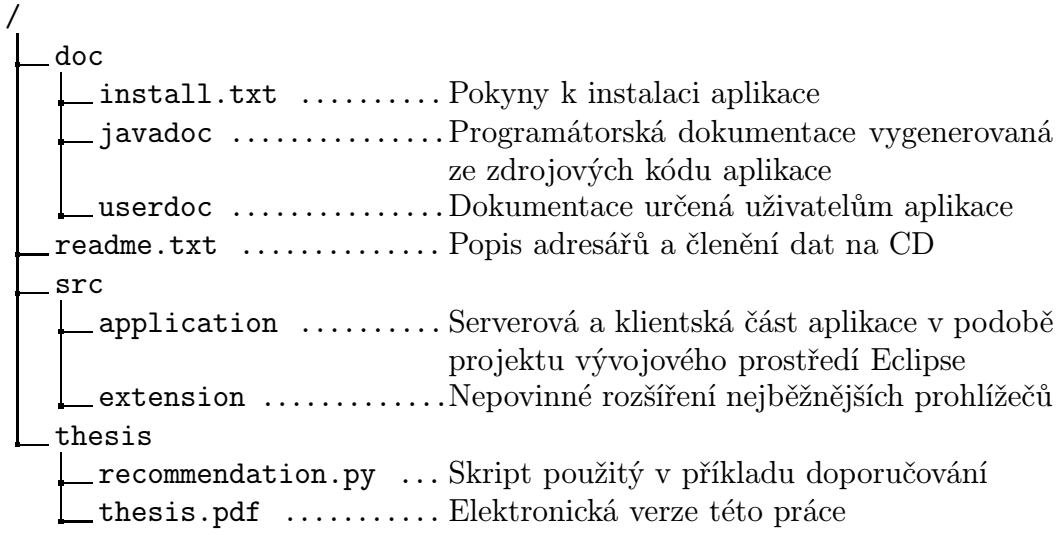
- sledovat libovolný kanál RSS nebo Atom,
- získávat informace o změnách i ze stránek, které zmíněné kanály neposkytují,
- doporučovat zdroje, které sledují podobní uživatele,
- přidat vlastní internetovou adresu jako novou položku (použitím rozhraní aplikace nebo prostřednictvím volitelného doplňku),
- přiřadit položkám štítky,
- filtrovat položky podle zdrojů či štítků,
- ukládat své filtry a používat je jako seznamy položek,
- zálohovat aktuální stav webové stránky reprezentované položkou,
- exportovat seznam položek jako RSS nebo Atom kanál,
- ovládat většinu aplikace jen pomocí konfigurovatelných klávesových zkratek.

Podařilo se nám vyhovět všem důležitým požadavkům, a tím i vyřešit identifikované nevýhody jiných čteček. Věříme proto, že aplikace má potenciál být využívána.

# Obsah přiloženého CD

Na přiloženém kompaktním disku je k dispozici v jednotlivých adresářích jak uživatelská a programátorská dokumentace, tak i samotné zdrojové kódy aplikace a doplňku do prohlížeče.

Seznam adresářů a popis jejich obsahu:



# Seznam použité literatury


[1] Bos, B.; Çelik, T.; Hickson, I.; aj.: Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. Recommendation, W3C, Červen 2011, http://www.w3.org/TR/2011/REC-CSS2-20110607. Latest version available at http://www.w3.org/TR/CSS2.

[2] Bray, T.; Paoli, J.; Sperberg-McQueen, M.: XML 1.0 Recommendation. Recommendation, W3C, Únor 1998, http://www.w3.org/TR/1998/REC-xml-19980210. Latest version available at http://www.w3.org/TR/REC-xml.

[3] Cleary, F.: 1 year ago. (OK, it's a bit more then that). Listopad 2012.
URL http://hiveminedblog.tumblr.com/post/35346314078/1-year-ago-ok-its-a-bit-more-then-that

[4] Cleary, F.: Hive Launch! Květen 2013.
URL http://hiveminedblog.tumblr.com/post/50026064514/hive-launch

[5] Crossrider: FAQ - Crossrider Extensions.
URL http://crossrider.com/pages/faq

[6] Deutelle, J.-M.: JScience. Říjen 2011.
URL http://jscience.org/

[7] Gosling, J.; Joy, B.; Steele, G.; aj.: *The Java Language Specification, Third Edition*. Prentice Hall, June 2005.

[8] Green, A.: Powering Down Google Reader. Březen 2013.
URL http://googlereader.blogspot.cz/2013/03/powering-down-google-reader.html

[9] Grönroos, M.: *Book of Vaadin*. Oy IT Mill Ltd, 2009, ISBN 9789529260454.
URL http://vaadin.com/book

[10] Hedley, J.: jsoup. Leden 2013.
URL http://jsoup.org/

[11] Huang, Z.; Zeng, D.; Chen, H.: A Link Analysis Approach to Recommendation under Sparse Data. In *AMCIS*, 2004, str. 239.

[12] International, E. C. M. A.: *ECMA-262: ECMAScript Language Specification*. Geneva, Switzerland: ECMA (European Association for Standardizing Information and Communication Systems), třetí vydání, Prosinec 1999.
URL http://www.ecma-international.org/publications/standards/Ecma-327.htm

[13] Isaacson, S.; Byrne, S. B.; Champion, M.; aj.: Document Object Model (DOM) Level 1. Recommendation, W3C, Listopad 1998, http://www.w3.

org/TR/1998/REC-DOM-Level-1-19981001. Latest version available at http://www.w3.org/TR/REC-DOM-Level-1.

[14] Nottingham, M.; Sayre, R.: The Atom Syndication Format. RFC 4287 (Proposed Standard), Prosinec 2005, updated by RFC 5988.
URL http://www.ietf.org/rfc/rfc4287.txt

[15] Page, L.; Brin, S.; Motwani, R.; aj.: The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, Listopad 1999, previous number = SIDL-WP-1999-0120.
URL http://ilpubs.stanford.edu:8090/422/

[16] Pazzani, M. J.; Billsus, D.: Content-Based Recommendation Systems. In *The Adaptive Web*, *Lecture Notes in Computer Science*, svazek 4321, editace P. Brusilovsky; A. Kobsa; W. Nejdl, Springer Berlin Heidelberg, 2007, ISBN 978-3-540-72078-2, s. 325–341, doi:10.1007/978-3-540-72079-9˙10.
URL http://dx.doi.org/10.1007/978-3-540-72079-9_10

[17] Raggett, D.; Hors, A. L.; Jacobs, I.: HTML 4.01 Specification. Technická zpráva, 1999, http://www.w3.org/TR/1999/REC-html401-19991224. Latest version available at http://www.w3.org/TR/html401.

[18] ronie: Fetch weather once every hour max. Únor 2013.
URL https://github.com/xbmc/xbmc/pull/2177

[19] Schafer, J.; Frankowski, D.; Herlocker, J.; aj.: Collaborative Filtering Recommender Systems. In *The Adaptive Web*, *Lecture Notes in Computer Science*, svazek 4321, editace P. Brusilovsky; A. Kobsa; W. Nejdl, Springer Berlin Heidelberg, 2007, ISBN 978-3-540-72078-2, s. 291–324, doi:10.1007/978-3-540-72079-9˙9.
URL http://dx.doi.org/10.1007/978-3-540-72079-9_9

[20] The Apache Software Foundation: Apache License, Version 2.0. Leden 2004.
URL http://www.apache.org/licenses/LICENSE-2.0

[21] Winer, D.: OPML 2.0. Červenec 2007.
URL http://dev.opml.org/spec2.html

# Seznam použitých zkratek

**Atom** formát určený k popisu změn webových stránek; je novější alternativou k RSS [14]

**Cloud computing** model poskytování prostředků založený na internetu; obvykle je zpoplatněno využití služby namísto klasické platby za software

**Cron** plánovač, který spouští úlohy periodicky v předem určený čas

**CSS** Cascading Style Sheets – jazyk určený pro popis vzhledu a formátování dokumentů [1]

**DOM** Document Object Model – popisuje obsah HTML nebo XML dokumentů pomocí stromové struktury elementů; specifikace [13] definuje i metody pro přístup k elementům a manipulaci

**GWT** Google Web Toolkit – knihovna a nástroj pro vývoj klientské části webových stránek v jazyku Java

**HTML** HyperText Markup Language – hlavní značkovací jazyk používaný při tvorbě webových stránek [17]

**HTTP** Hypertext Transfer Protocol – bezestavový protokol určený ke komunikaci klienta se serverem při poskytování webových stránek

**JavaScript** formálně ECMAScript [12], interpretovaný jazyk běžící v internetovém prohlížeči, který umožňuje tvůrci webové stránky interaktivně manipulovat s jejím obsahem

**JSON** JavaScript Object Notation – textový formát určený pro přenos strukturovaných dat založený na syntaxi jazyku JavaScript

**MVP** Model-view-presenter – členění aplikace na tři části, kde každá je zodpovědná za jednu úlohu: model reprezentuje data, view zobrazuje data, presenter předává požadavky uživatele a vrací výsledná data

**NAS** Network-attached storage – zařízení v lokální síti, které provozuje souborový server; obvykle je hardware a software takového zařízení uzpůsoben svému účelu

**OPML** Outline Processor Markup Language – XML formát popisující stromové struktury [21]

**ORM** Object-relational mapping – slouží k převádění dat mezi dvěma nekompatibilními systémy uložení; tradičně mezi objekty tříd a záznamy v relační databázi

**RSS** Rich Site Summary – rodina formátů, které popisují změny, novinky na webových stránkách

**XML** Extensible Markup Language – univerzální strojově i lidsky čitelný značkovací jazyk určený pro ukládání a výměnu dat [2]