Masarykova univerzita
Fakulta informatiky

Technologické řešení uživatelského rozhraní systému WayBack webového archivu

Diplomová práce

Filip Kusalík

Brno, 2011

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: Mgr. Adam Brokeš

Poděkování

Rád bych poděkoval svému vedoucímu práce Mgr. Adamu Brokešovi za cenné připomínky a podněty k práci při zachování přiměřené volnosti, v rámci které rád pracuji. Dále bych rád poděkoval Sofii Karpowiczové za konzultace ohledně jazykové korektury a rodině za podporu. Děkuji.

Shrnutí

Cílem práce je uživatelské rozhraní pro vyhledávání a procházení archivovaných záznamů webového archivu WebArchiv s využitím informací o kolekcích systému WA Admin. Práce analyzuje existující řešení na poli webových archivů, hodnotí jejich klady a zápory s přihlédnutím na eventuální využití získaných poznatků pro samotný návrh a implementaci uživatelského rozhraní. Práce pojednává o stávajících možnostech vyhledávání nad bázemi metadat v projektu WebArchiv. Jedná se o fulltextové vyhledávání nad indexem NutchWAX, základní vyhledávání WayBacku a procházení a vyhledávání v databázi kolekcí systému WA Admin. Uživatelské rozhraní je implementováno jako modul webové aplikace WayBack v jazyce Java běžící na serveru Tomcat formou nástrojové lišty. Její obsluha probíha interaktivně pomocí technologie AJAX. Výsledkem je univerzální rozhraní se základní funkcionalitou procházení kolekcí, klíčových slov, fulltextové vyhledávání jak v indexu NutchWAX, tak v databázi kolekcí. Do tohoto rozhraní je možno v budoucnu implementovat moduly dle vyvíjejících se požadavků projektu WebArchiv.

Práce byla vysázená v LaTeX

Klíčová slova

Archivace webu, WebArchiv, Internet, WA Admin, WBToolbar, JSP, webová aplikace, WayBack.

Motivace

V práci se budu pohybovat v oblasti archivace webu na mezinárodní úrovni i na úrovni České republiky, jejíž současný stav výstižně popisuje Mgr. Adam Brokeš ve své práci *Integrace a automatizace systemů v pracovních procesech projektu WebArchiv* [3] v kapitole 1 *Archivace webových dokumentů* a kapitole 2 *Projekt WebArchiv*. Proto pro hlubší uvedení do historie a problematiky archivace webu, a zejména specifika projektu WebArchiv, doporučuji čtenáři zmiňované kapitoly k prečtení. Další publikace lze naleznout v knihovnických systémech pod heslem „webarchiv“.

Obsah

Úvod

Internetové archivy webových stránek jsou vzhledem k samotnému stáří Internetu relativně mladé. Když bych to vyjádřil jinak, archivy jsou mladé, protože je mladými vidíme. Opak je ale pravdou – nejstarší archiv *Internet Archive* uchovává kulturní dědictví různých států již čtrnáct let, což není v poměru k rozšířenosti Internetu zanedbatelná doba. Svým celosvětovým působením je brán jako autorita na poli archivace digitálních médií jednak z pohledu *množství archivovovaných* – nebo též „sklízených" – dat, jednak z pohledu *technologického zázemí* v podobě různých aplikací na jejich sklízení, indexaci a prezentování.

Otázkou je, jaký regionálně nebo také tematicky laděný přínos má Internet Archive. Politika jeho sklízení se opírá o princip *kvantitativního* sběru dat, kdy je kladen větší důraz na široký záběr sklizně a důraz omezený na sběr dat do hloubky – *kvalitativně*. Z této situace vyplynul vznik lokálně (i tematicky nebo formátově) zaměřených archivů, které si kladou za cíl přesný opak než Internet Archive – sklízení menších jednotek dat s důrazem na hlubší archivaci. Pro Českou republiku se tohoto úkolu zhostila Národní knihovna ve spolupráci s její dceřinou Moravskou Zemskou knihovnou založením projektu WebArchiv. Není překvapující, že tento projekt vznikl právě na půdě knihoven. Za éry neelektronických zdrojů znalostí – knih, novin, časopisů, magazínů, věděckých publikací, sborníků a dalších, byla jediná možnost dohledávat data ručně procházením katalogizovaných papírových záznamů. Zde hrály důležitou roli právě knihovny jako ústřední centra. S nástupem a následným rozmachem elektronických publikací (a technických zařízení pro jejich čtení) by se mohlo zdát, že knihovny ustupují do pozadí před snadnou dostupností zdrojů přes Internet.

Jistě, internetové encyklopedie a obrovské databanky vytlačily knihovny z pozice prostředků pro amatérské vyhledávání dat a výzkum, ale realita je poněkud jiná. Současný zájem čtenářů o knihovny, co se týče neelektronických publikací, oslabuje. To ale není nic špatného. Knihovny by se měly více situovat do role knihoven nové generace – míst, kde se lidé budou spolu scházet, komunikovat a spolupracovat na projektech. Knihovna by také měla sloužit jako znalostní báze a autorita v hodnocení kvality

na širokém poli elektronických publikací, u nichž častokrát není dodržena míra kvality a odbornosti jako u klasických neelektronických médií. Projekt WebArchiv je jedním z kroků ke knihovně nové generace. Z pohledu této práce v něm probíhají paralelně dva procesy [1, sekce 3.2], proces *celoplošné sklizně* české národní domény a proces *výběrové sklizně* ručně ověřovaných zdrojů.

Právě kvůli druhému procesu řadím projekt WebArchiv mezi projekty posouvající knihovnu k titulu nové generace. Pro prezentaci sklízených dat je využíván nástroj WayBack vyvíjený pod záštitou Internet Archive, nicméně tento nástroj je schopen pracovat pouze s procesem prvním. Proto vznikla idea doimplementování modulu, který by byl schopen v nástroji spolupracovat s procesem druhým.

V kapitole 1 rozeberu uživatelské rozhraní existujících nástrojů pro prezentaci záznamů ve webových archivech, jejich výhody a nevýhody, následně v kapitole 2 popíši existující možnosti vyhledávání v systému WayBack. V kapitole 3 tyto poznatky spojím do návrhu a realizace vzhledu a rozmístění komponent uživatelského rozhraní. Finální logiku implementace vysvětlím v kapitole 4.

Kapitola 1

Existující rozhraní archivů a jejich charakteristika

Archivací webu se zaobírá více institucí[1], u nichž není triviální sjednotit požadavky kladené na prezentaci archivovaných dat unifikovaným způsobem vzhledem k rozdílnému přístupu ke sklízení dat. Zejména se jedná o **hloubku** a **vyhraněný rozsah archivace**, ať už důkladné sklízení malé, anebo povrchní sklízení rozměrné množiny stránek. Dále záleží také na charakteru sklízených dat, protože některé projekty archivace se zaměřují na vyčleněné skupiny mediálních typů dat[2], oproti projektům s klasickým přístupem sklízení primárně textového obsahu. Proto je vhodné si ozřejmit, jaké možnosti dané prezentační mechanizmy poskytují, aby bylo možno porovnat přínos pro uživatele, a z těchto poznatků se poučit.

Pro tento účel jsem vybral z výše popsaného souboru internetových archivů čtyři konkrétní. Kritérium výběru bylo jediné: archiv jistým způsobem modifikuje stávající uživatelské rozhraní WayBacku, nebo používá své vlastní, případně poskytuje zajímavé řešení pro zobrazování kolekcí.

Vybrané archivy jsou:

Icelandic Web Archive – islandský webový archiv pod záštitou *National and University Library of Iceland*[3];

Archive It – katalogizovaný archiv kolekcí spadající pod *Internet Archive*[4];

Web Archiving Service – soubor archivů z různých oblastí vyvíjený *California Digital Library*[5];

UK Web Archive – archiv národní domény `.uk` Spojeného království Velké Británie a Severního Irska pod záštitou *British Library*[6].

---

1. Tyto instituce se sdružují v konzorciu IIPC (International Internet Preservation Consortium) a jejich seznam je dostupný na adrese `http://netpreserve.org/about/archiveList.php`
2. Např. video, obrázky, hudba atp.
3. `http://landsbokasafn.is/`
4. `http://www.archive.org/`
5. `http://www.cdlib.org/`
6. `http://www.bl.uk/`

1.1 Icelandic Web Archive

Archiv dostupný na `http://wayback.vefsafn.is/` mírně modifikuje rozhraní WayBacku. Úvodní stránku a systémové podstránky (přehled archivovaných verzí, chybová stránka, rozšířené vyhledávání, atd.) ponechává v nezměněné podobě, jedinou upravenou komponentou je uživatelská lišta. Originální verze WayBacku vkládá do HTML výstupu archivované stránky HTML kód uživatelské lišty hned za tag `<body>`. Vývojáři se tak potýkají s problémem u nestandardních archivovaných stránek, ať už se jedná o flashové prezentace, absolutní pozicování obsahu, chyby v JavaScriptu a kaskádových stylech a jiné [16], kvůli kterým může docházet ke kolizím ve vzhledu a funkcionalitě lišty.

Vývojáři z *Icelandic Web Archive* vyřešili tuto situaci naprosto odlišně, samotná lišta je sice umístěna hned za tag `<body>`, ale archivovaná stránka nepokračuje dál v toku výstupu, nýbrž je umístěna ve vnořeném rámu `<iframe>`[7], jehož velikost se přepočítává JavaScriptem podle velikosti okna prohlížeče (i při změně jeho velikosti). Efektivně se tak vyhnuli problému s kolizí obsahu a tímto řešením vynikají nad ostatními archivy.

Ostatní funkcionalita je vesměs standardní, samotná lišta poskytuje informace o datu sklizení, u kterého se po najetí kurzorem zobrazí dodatečné metainformace, zejména z hlavičky *ARC* [4] souboru. Pak se zde nachází základní ovládací prvky WayBacku (viz také 2.1.3), jako je překlikávání časových verzí, formulář pro zadání URL a časová osa. Časová osa je řešena přehledným, ale na ovládání nepříliš šťastným způsobem. Pro každou archivaci stránky je zde umístěn malý čtvereček, kvůli kterému je u vysokého počtu archivací lišta příliš široká, a protože je rolovací v horizontálním směru, je neobratně ovladatelná.

1.2 Archive It

Archive It sídlící na `http://www.archive-it.org/` je katalogizovaný archiv kolekcí archivovaných stránek. Archiv je rozdělen na dvě hlavní sekce: *partneři* a *kolekce*. Partneři jsou rozděleni do šesti kategorií podle typu (knihovny, univerzity, instituce, atp.) a u každého partnera je pak seznam k němu se vztahujících kolekcí. Partneři jsou rozděleni do devíti kategorií a u každé kolekce v seznamu je pak uveden partner, ke kterému se daná kolekce vztahuje, čímž vzniká oboustranně propojený systém. Prohlížení

7. HTML element, který se na stránce vykresluje v definované obdélníkové oblasti, do které načítá obsah z externí URL.

jednotlivých záznamů je pak realizováno v standardním rozhraní nástroje WayBack. Největším plusem archivu je právě jeho rozsáhlá databáze kolekcí.

1.3 California Digital Library

Soubor archivů s propracovaným využitím metadat sídlí na adrese `http://webarchives.cdlib.org/`. Z úvodní stránky se lze dostat na přehled jednotlivých archivů, na které lze nahlédnout jako na tematické kategorie z kalifornské oblasti[8]. Každy archiv má pak u sebe zaveden popis, seznam archivovaných stránek a obrázkové záhlaví, které uživatele provází po celou dobu prohlížení konkrétního archivu. U detailu archivované stránky jsou uvedena následující metadata v záložkách – název, URL v archivu, URL *živé* verze, datum a čas sklizení, abstrakt, popis, velikost a *mimetype*[9]. Záložky nejsou interaktivní, tudíž je potřeba vždy vyčkat pro obnovení celé stránky. Technologické řešení je provedeno přes rámy, což je z hlediska kompatibility prohlížečů nepříliš vhodné řešení. Vhodnější by bylo využití tagu `<iframe>` podobně jako u Iceland Web Archive v 1.1.

1.4 UK Web Archive

Národní doména Spojeného království Velké Británie a Severního Irska je archivována v archivu dostupném na `http://www.webarchive.org.uk/ukwa/`. Z pohledu uživatele jde o velice pokročilý archiv, jehož nabízené možnosti jsou následující:

Kolekce – Stránky jsou kategorizovány do tří typů kolekcí[10]: *kolekce událostí* (např. Olympijské hry 2012), *tematické kolekce* (např. hospodářská krize) a *předmětově orientované kolekce* (např. britský venkov). Dále je stránkám přiřazen obor, pod který spadají.

Vyhledávání – V kolekcích je možné vyhledávat na základě kolekce, oboru, data archivace a *fulltextového* dotazu.

Náhledy – Každá časová verze archivované stránky má svůj náhled, nicméně poslední přidané záznamy se zobrazují dočasně bez náhledu.

8. Kupříkladu archivy kalifornských okresů, přírodních památek, společenských událostí atp.
9. Formát souboru v prostředí Internetu.
10. `http://www.webarchive.org.uk/ukwa/collection/`

Procházení kolekcí – Přehled kolekcí může být zobrazen v režimu náhledů, kdy se zobrazuje název kolekce a vybraný náhled kolekce, nebo v režimu textovém, kdy se místo náhledu zobrazuje slovní popis kolekce. Po rozkliknutí kolekce archiv nabídne seznam archivovaných stránek se stránkováním.

Vizualizace – Alternativní zobrazení archivu nabízí dvě možnosti – mrak klíčových slov a 3D stěnu. Mrak klíčových slov zobrazuje nejvýraznější klíčová slova z dané kolekce. V archivu lze tímto způsobem prohlížet pouze jednu kolekci a to „Volby 2005“. Technické zázemí zabezpečuje technologie od společnosti IBM *BigSheets*[11]. Na 3D steně jsou zobrazeny všechny archivované stránky ve třech řádcích a pomocí posuvníku je možné „prolítat“ celým archivem. Význam této funkce je spíš estetický než praktický.

1.5 Porovnání archivů

Shrnutí porovnávaných archivů je prezentováno v tabulce 1.1.

Archiv	**Plusy**	**Mínusy**
IWA	Nekonfliktní řešení nástrojové lišty.	Nepřehledná časová osa.
AI	Rozsáhlé provázání kolekcí a partnerů.	Lišta bez funkcionality.
WAS	Propracovaný systém metadat.	Rámy, neinteraktivní lišta.
UKWA	Kategorizace záznamů, náhledy, vizualizace.	Základní lišta WayBacku.

Tabulka 1.1: Porovnání referenčních archivů.

11. `http://www-01.ibm.com/software/ebusiness/jstart/bigsheets/index.html`

Kapitola 2

Báze pro vyhledávání

Aby bylo možné smysluplně využít potenciál záznamů webového archivu a dodatečných metadat k těmto záznamům se vztahujícím, je potřeba se orientovat v možnostech, které nástroje projektu WebArchiv pracující nad bází dat archivu poskytují, jakým způsobem data získávají, uchovávají a hlavně jakým způsobem jsou schopny je poskytnout k prezentaci. V projektu WebArchiv existují takovéto nástroje tři: *základní vyhledávání WayBacku*, *fulltextové vyhledávání NutchWAX* a *vyhledávání nad databází metadat nástroje WA Admin*.

2.1 Základní vyhledávání WayBacku

Momentálně existují dvě zásadní verze WayBacku[1] a to – verze stabilní `1.4.2` a verze experimentální `1.6.0`. V podkapitolách 2.1.1 a 2.1.2 ukážu sklízení a indexaci webu a v následující podkapitole 2.1.3 vysvětlím principy nativního vyhledávání ve WayBacku. Pokud to bude možné a smysluplné, tak ukážu posun funkcionality a jeho význam od verze `1.4.2` k verzi `1.6.0`.

2.1.1 Sklízení obsahu webu pomocí nástroje Heritrix [12]

Internet Archive v roce 2004 vyvíjel vlastní open source řešení pro sklízení obsahu webových stránek. Výsledkem bylo vyvinutí samostatné *standalone* aplikace *Heritrix*. Heritrix dostane na vstupu počáteční semínka[2] tzv. *seeds*, která podle administrátorem nastavených pravidel prochází a archivuje. Sklizený obsah ukládá sekvenčně spolu s metadaty do formátu *ARC*. V experimentálním módu je možno zvolit pro ukládání archivovaného obsahu nástupní formát *WARC* [11], který, jak je vidno z následujícího přehledu, poskytuje mnohem širší možnosti uložení a využití metadat.

---

1. `http://archive-access.sourceforge.net/projects/wayback/release_notes.html`
2. Základní domény libovolného řádu.

Ukázka metadat formátu *ARC* [8, str. 4]. Formát ARC umožňuje uchování pouze informací hlavičky HTTP odpovědi sklizeného záznamu a neexistuje tu možnost do něj jednoduše uložit další data. Pro bližší popis formátu viz [9, str. 12–14].

```
http://www.oac.cdlib.org/ 128.48.120.68
   20050727235250 text/html 11182
HTTP/1.1 200 OK
Date: Wed, 27 Jul 2005 23:52:49 GMT
Server: Apache/1.3.27 (Unix) mod_perl/1.27
Last-Modified: Thu, 02 Jun 2005 00:04:46 GMT
ETag: "3cb67-2aa6-429e4d1e"
Accept-Ranges: bytes
Content-Length: 10918
Connection: close
Content-Type: text/html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
    Transitional//EN">
<html>
. . .
</html>
```

Ukázka metadat formátu *WARC* [8, str. 13]. Formát WARC poskytuje možnost uložení doplňkových metadat pomocí XML sekce `harvestmetadata`, do které lze při sklízení webu uložit libovolná data pomocí XML tagů. Pro bližší popis formátu viz [9, str. 14–15].

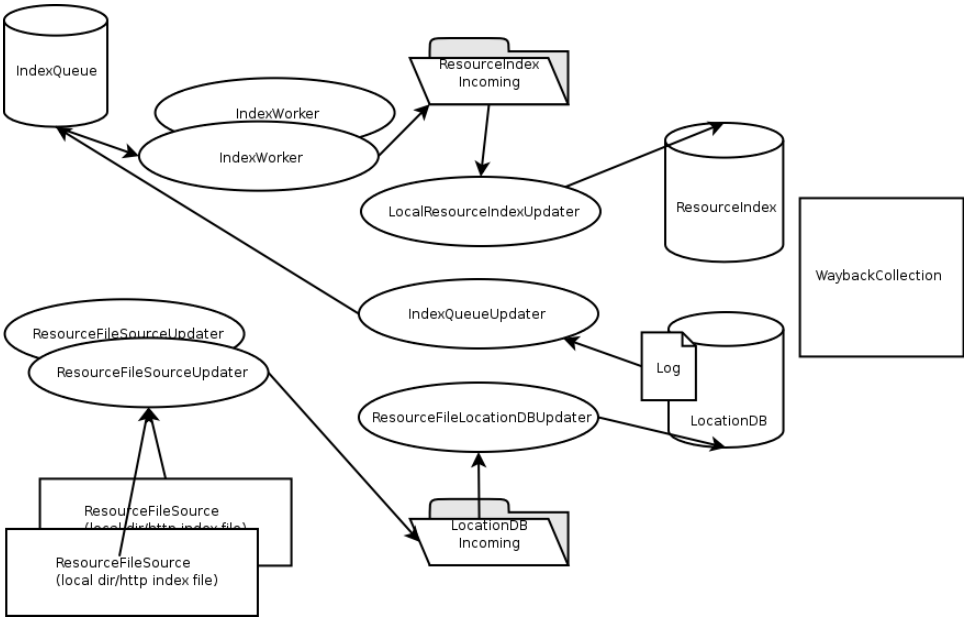
```
warc/0.8 395 metadata http://www.archive.org/logo.jpg
20050708010101 text/xml
http://ark.cdlib.org/ark:/13030/xt12rk835gm/_s
Related-Record-ID: http://ark.cdlib.org/ark:/13030/
xt12rk835gm
<?xml version="1.0"?>
<harvestmetadata
xmlns="http://archive.org/harvest/0.8/">
<discovered-via>http://www.archive.org<discovered-via>
<download-time-ms>565</download-time-ms>
</harvestmetadata>
```

2.1.2 Automatická indexace sklizených dat [14]

Po vytvoření archivních souborů ARC nebo WARC přichází na řadu indexace obsahu. Tyto soubory jsou umístěny do adresářů nakonfigurovaných ve WayBacku jako `ResourceFileSource`, které v pravidelných intervalech kontroluje na nové soubory komponenta `ResourceFileUpdater`. Tato komponenta vytvoří z nových souborů tzv. *manifest*, který vloží do adresáře nakonfigurovaného ve WayBacku jako `LocationDB incoming`. Kontrolu tohoto adresáře na příchozí manifesty má na starosti komponenta

`ResourceFileLocationDBUpdater`, která nové manifesty spojí se stávajícími záznamy v `LocationDB`[3]. Tato databáze provedené změny zapíše do svého *logu* sledovaného komponentou `IndexQueueUpdater` a ta všechny nově objevené archivní soubory vloží do fronty `IndexQueue`.

Komponenta `IndexQueueWorker` je zodpovědná za vytvoření indexu ze všech zdrojů ve frontě a tento index posune na vstup komponentě `LocalResourceIndexUpdater`, která sloučí nový index se stávajícím kmenovým indexem. Celý proces je znázorněn na obrázku 2.1[4].



Obrázek 2.1: Ukázka průběhu indexování nových archivních souborů.

2.1.3 Dotazování a prezentace výsledků

Z pohledu uživatele existují pouze omezené možnosti, kterými by mohl být systém WayBack dotazován na archivované stránky. Ve verzi standardního vyhledávání WayBacku `1.4.2` uživatel pouze zadává URL adresu webu, následně mu systém nabídne seznam časových verzí v archivu dělených

3. Implementována v *Berkeley DB Java Edition*, viz `http://www.oracle.com/technetwork/database/berkeleydb/overview/index-093405.html`
4. Originál obrázku je dostupný na WWW: `http://archive-access.sourceforge.net/projects/wayback/resource_store.html`

podle měsíců a let. Uživatel se následně překlikem dostane na konkrétní detail záznamu. Ve verzi rozšířeného vyhledávání může uživatel specifikovat další požadavky a to:

Časové rozmezí od–do – pokud chce uživatel vyfiltrovat pouze výsledky z určitého období;

Přepínač URL Matching – zda chce uživatel vypsat stránku, která nejblíže odpovídá dotazu, nebo chce vypsat všechny vyhovující stránky;

Přepínač Aliases – umožňuje výsledky seskupit podle duplicity, kupříkladu záznamy `www.webarchiv.cz`, `webarchiv.cz` a `www.webarchiv.cz/index.php` obsahují všechny ten samý obsah a WayBack podle zvolené možnosti zobrazí tyto tři možnosti seskupené v jednom výsledku, anebo je zobrazí zvlášť, anebo zobrazí pouze variantu, která vyhovuje podmínce zadání uživatele. Přehled této funkcionality je v tabulce 2.1;

Zvolená možnost	Zobrazení
Merge aliases	www.webarchiv.cz, webarchiv.cz, www.webarchiv.cz/index.php
Show aliases separately	1. www.webarchiv.cz 2. webarchiv.cz 3. www.webarchiv.cz/index.php
Don't show aliases	1. www.webarchiv.cz

Tabulka 2.1: Různé zobrazení výsledků pro konfigurace aliasů.

Omezení typu souborů – na určitou mediální povahu (například obrázky, videa, PDF apod.);

Další – z hlediska použitelnosti systému irelevantní možnosti.

Vyhledávání dále podporuje hvězdičkovou notaci, tzv. *wildcard*. URL konkrétní archivované stránky je ve tvaru:

```
http://wayback.webarchiv.cz/
    {prefix}/YYYYMMDDHHMMSS/http://domena.tld/
```

Skládá se tedy ze tří segmentů:

1. `http://wayback.webarchiv.cz/{prefix}/` – host a prefix[5]
2. `YYYYMMDDHHMMSS` – časový údaj
3. `http://domena.tld` – URL archivovaného záznamu[6]

Lze vypozorovat, že zobrazený záznam je otisk domény `domena.tld` v čase `HH:MM:SS DD.MM.YYYY`. Existuje možnost zadat místo přesného data a času *hvězdičku* (hvězdička se musí nacházet vždy na konci řetězce), která zabezpečí vypsání všech časových otisků dané URL ve všech časech, které splňují zadání hvězičkové (*wildcard*) notace. Obdobně lze zadat hvězdičku i na libovolně ořezaný konec URL, kdy se vypíšou všechna URL v daném čase začínající na řetězec umístěný před hvězdičkou. Také lze tyto možnosti kombinovat, například výběr všech stránek z podadresáře `fajnestromy.cz` v červenci 2010 je v následujícím tvaru:

```
http://wayback.webarchiv.cz/
    wayback/201007*/http://fajnestromy.cz/listnate*
```

Uživatel pozoruje u konkrétního záznamu (*replay*) v systému WayBack `1.4.2` obdélníkovou lištu umístěnou v horní části obrazovky, na které jsou umístěny informace o čísle časové verze, datu a času sklizení. Dále pozoruje jednoduchou časovou osu, pomocí které se dá přepínat mezi časovými verzemi. Neexistuje možnost lištu schovat, ani jiným způsobem odsunout, tudíž oblasti v horní části archivované stránky nejsou pro uživatele přístupné vizuálně ani funkčně. Ve verzi `1.6.0` je pokročilejší, na liště je vyhledávací pole, odkaz na všechny časové verze dané stránky a také odkaz na hlavní stranu. Lišta je poloprůhledná, takže je možné rozpoznat objekty schované za ní a také je možné ji zavřít, což má ovšem za následek její úplné zmizení, pro její opětovné zobrazení je nutné obnovit stránku.

2.2 Fulltextové vyhledávání NutchWAX [13]

NutchWAX[7] slouží na fulltextové vyhledávání ve webových archivech. To je realizováno modifikací *crawleru Nutch*[8], který místo procházení „živého“

5. WayBack je možno nakonfigurovat pro spuštění i v režimu *defaultní* webové aplikace, nahraje se tedy v kořenovém adresáři do podadresáře /ROOT, čímž vznikne tvar URL bez prefixu.
6. `http://classic-web.archive.org/collections/web/advanced.html`
7. Nutch + Web Archive eXtensions
8. `http://nutch.apache.org/` – Open-source nástroj Nutch je vyvíjen organizací Apache jako vyhledávač ve webových stránkách. Je postaven nad *engine* Lucene a Solr, které

Internetu prochází archivní soubory ARC, a dalšími pluginy, které přidávají další data do indexu[9]. Fulltextové vyhledávání ve webovém archivu má jisté odlišnosti oproti fulltextovému vyhledávání nad plošnými daty:

Algoritmus vyhledávání – protože obsahem archivu jsou periodicky sklízené internetové stránky, nachází se zde velká míra duplicity (i částečné) záznamů. Vyhledávání probíhá tedy nad databází duplicitních dat a algoritmus musí zohlednit duplicitu (i částečnou) a výsledky seskupit do jednoho výsledku. U tohoto seskupeného výsledku je třeba zohlednit specifický charakter prezentace.

Specifický charakter prezentace výsledků – U výsledku musí být možné zjistit, z kolika podvýsledků se skládá, je potřeba rozhodnout, jaký titulek, náhled, případně popis se použije.

2.3 WA Admin a jeho kolekce

Národní knihovna vytváří ručně spravovanou databázi obsahově přínosných internetových stránek – *zdrojů*. Tyto zdroje jsou vybírány tzv. *kurátory*, kteří procházejí Internet a ručně hodnotí vybrané domény jako přínosné. Pokud je tento zdroj schválen všemi kurátory, je osloven[10] *vydavatel* zdroje s nabídkou jeho archivace. Pokud souhlasí, je podepsána *smlouva*. Tento proces v minulosti probíhal ručně a komplikovaně, až se dostal do neúnosného stádia, a vznikla potřeba centrálního poloautomatizovaného nástroje. Byla vytvořena zadávací dokumentace, na základě které byl specifikován, navržen a implementován systém WA Admin, který má na starosti správu zdrojů, vydavatelů a smluv v projektu WebArchiv. Tento systém je realizován formou webové aplikace umožňující kurátorům efektivně spravovat jednotlivé položky. Mezi jeho stěžejní funkce patří [2, str. 15]:

Vložení zdroje – kurátor vloží do systému základní informace o zdroji a ten mu ke zdroji okamžitě nabídne, zda už stejný záznam v databázi neexistuje. Pokud ne, kurátor vyplní ke zdroji klíčová slova, kategorie Konspektu[11] a uloží jej do databáze.

Hodnocení zdroje – umožňuje kurátorovi u záznamu nastavit míru přínosnosti zdroje. Možné hodnoty jsou ANO, NE, MOŽNÁ (kontrola

---

rozšiřuje o funkcionalitu webu – crawler, databáze orientovaných grafů, HTML parsery, další formáty dokumentů apod.

9. Pozici záznamu v ARC souboru, kolekce atp.
10. Také existuje možnost pro vydavatele podat návrh na prozkoumání jejich webů.
11. Konspekt je kategorizace archivu do kolekcí.

za půl roku), TECHNICKÉ NE (zdroj není možno technologicky sklidit v požadované kvalitě). V nástroji WBToolbar se zobrazují pouze zdroje s hodnocením ANO a s vyplněnou a aktivní smlouvou (viz další bod).

Přiřazení smlouvy – po dohodě s vydavatelem je sepsána smlouva, kterou kurátor zanese do systému. Zároveň jsou vytvořeny záznamy o vydavateli, kontaktu na něj a doménach, které se mají podle smlouvy sklízet.

Kapitola 3

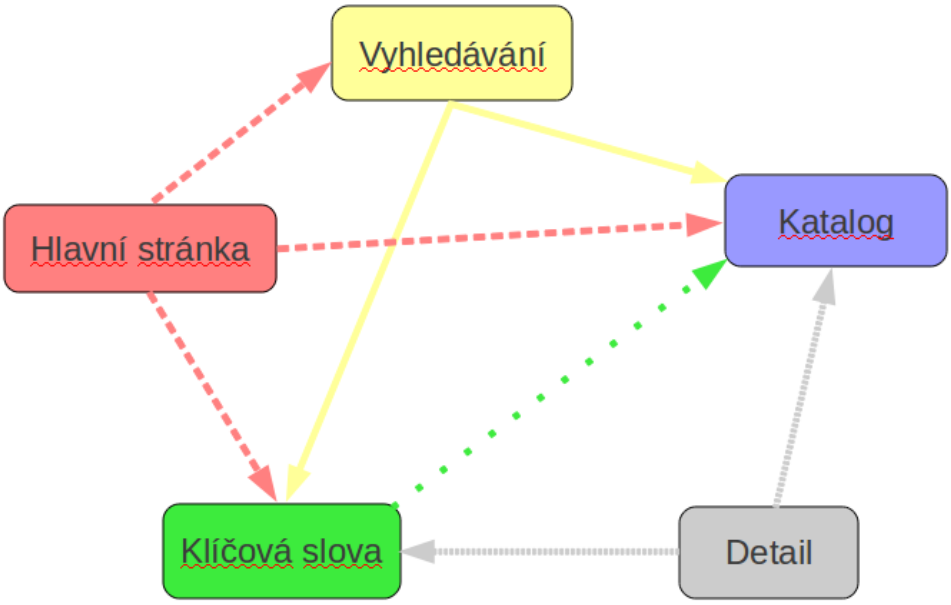
Uživatelské rozhraní

3.1 Funkční prvky v teoretické rovině

Systém se snaží uživateli zjednodušit provádění jednotlivých úloh, proto je uživatelský panel rozdělen vizuálně do záložek podobných jako u California Digital Library v 1.3 na stránce 7, ale mezi kterými lze interaktivně překlikávat. Panel je situovaný na střed stránky nahoru a v závislosti na zvoleném obsahu se mu přizpůsobuje výška. Panel lze zúžit na minimální výšku kliknutím na tlačítko umístěné vlevo dole. Tudíž uživatel v případě, že má zájem pouze o prohlédnutí stránky, nikoli o funkcionalitu panelu, panel schová minimalizačním tlačítkem při zachování obsahu celé prohlížené stránky. Systém nabídne uživateli formulář pro zadání klíčového slova, resp. URL, na základě kterého se zobrazí fulltextové výsledky vyhledávání, resp. detail záznamu. Fulltextové vyhledávání je dvojí: nad indexem NutchWAX viz 2.2, nebo nad databází metadat WA Admin, viz 2.3. Dále uživateli nabídne možnost procházet kategorie a podkategorie kolekcí, klíčová slova a zobrazovat detailní informace o záznamu. Na obrázku 3.1 je vyobrazeno propojení jednotlivých komponent (tj. z které komponenty se lze dostat na jinou komponentu).

3.2 Grafický návrh a stříhání šablony

Grafický návrh i šablonu jsem vypracoval sám. První fází bylo načrtnutí drátěných modelů (*wireframes*) panelu, aby bylo zřejmé, kam umístit při tvorbě ostrého grafického návrhu jednotlivé komponenty rozhraní. Při určování pozice, rozložení komponent a jejich velikosti jsem vycházel částečně z poznatků shrnutých v [7] a částečně dle vlastní intuice. Jako další následovala fáze třetí, nákres konkrétních elementů ztvárněných ve finální grafické podobě. Na obrázku 3.2 lze vidět rozdíl a posun od drátěného modelu k finální grafické podobě. Grafický návrh jsem vypracoval v programu



Obrázek 3.1: Propojení jednotlivých komponent.

Adobe Photoshop[1] s využitím skupin a vrstev (*groups and layers*), které dovolují spolu v kombinaci se soustavou řezů (*slices*) jednoduchou změnu a záměnu grafických prvků.

Exportování grafických elementů pro použití pro webové stránky se v Adobe Photoshop nejdříve připraví pomocí výše zmiňovaných slices, na které lze nahlížet jako na obdélníky s definovanými rozměry (výškou, šířkou), umístěním (vzdálenost v pixelech od levého horního rohu) a úrovní zanoření (jednotlivé slices je možné dávat na sebe a za sebe). Také je u nich praktické vyplnit jméno, které se při exportu použije jako název obrázku, a také formát souboru, případně průhlednost a kvalitu (požadovanou velikost souboru).

Při exportu program rozdělí celý prostor grafického návrhu na předem definované obdélníky a pro každý vytvoří samostatný obrázek v definovaném formátu, kvalitě a případné průhlednosti. Ten se dále použije při tvorbě šablony přímým vložením do HTML, anebo nepřímo vložením na pozadí objektu pomocí kaskádových stylů CSS. Při definici slices – obdél-

1. http://www.photoshop.com/

Obrázek 3.2: Posun od drátěného modelu ke grafickému návrhu.

níků je důležité mít představu, jakým způsobem se obrázky po exportu zapíšou v CSS[2]. Proto je potřeba vypracovat tzv. *střih* (od toho „stříhání šablony“), dostačující formou je nákres tužkou na papír, jehož výsledkem je grafická reprezentace HTML elementů s popisem základních CSS vlastností. Podle tohoto střihu se nadefinují slices a vyexportují obrázky.

Na základě střihu se dále vytvoří HTML soubor, do kterého se současně zanesou popisy CSS tříd a identifikátorů. Vzhledem k faktu, že WayBack vkládá nástrojovou lištu do archivovaného záznamu přímo za HTML tag `<body>` při zachování veškerých CSS stylů a JavaScriptů záznamu, existuje možnost kolize technologie záznamu a nástrojové lišty. Vývojáři WayBacku řeší tento problém vkládáním výhradně in-line stylů a hojným využíváním klauzule `!important`, kvůli čemuž ovšem ztrácí možnost jednoduše a přehledně vyvíjet nové funkce a měnit vzhled podle potřeby. V nástroji WBToolbar jsem tuto situaci řešil extrahováním stylů do externího souboru a použitím nesmyslného prefixu `w1b4_` v názvu identifikátorů, tříd kaskádových stylů a javascriptových funkcí. V celé práci dále použiji, pokud budu uvádět název identifikátoru, třídy kaskádového stylu, javascriptové funkce, nebo jiného parametru, nebo objektu, vždy pro přehlednost verzi bez prefixu. V aplikaci jsou ale všechny názvy s prefixem.

Panel je rozdělen na záložky *Domů*, *Vyhledávání*, *Katalog*, *Detail* a *Klíčová slova*. V záložce *Vyhledávání* uživatel vlastním dotazem zadaným do formuláře ovlivňuje prezentované výsledky a může si pomocí systému podzáložek vybrat výsledky buď z fulltextu, nebo z kolekcí. V případě, že se nachází na detailu záznamu, je zvýrazněna záložka *Detail*. Výsledky vyhledávání z fulltextu, kolekcí, procházení strukturou kolekcí a klíčových slov mají stejný vzhled, kromě výsledků z kolekcí a klíčových slov, které mají

2. Kaskádové styly slouží pro separovanou definici vzhledu HTML elementů od sémantického vyjádření.

rozšíření v podobě dodatečných metadat[3]. V záložce *Kolekce* je uživateli nabídnut seznam kategorií a podkategorií kolekcí. Po výběru kategorie nebo podkategorie se uživateli zobrazí všechny záznamy dané kategorie nebo podkategorie. Jedná se tedy o standardní prohlížení stromové struktury.

Přepínání mezi záložkami je realizováno pomocí javascriptové knihovny *jQuery* [5]. Původním plánem bylo použít podknihovnu jQuery UI[4], ale kvůli své chybějící možnosti vložit do těla přepínače záložky formulář s textovým polem jsem byl nucen tuto podknihovnu vyloučit. Podobně dopadl i velice flexibilní plugin pro jQuery *idTabs*[5], který sice nabízel širší funkcionalitu, ale nezvládl bez komplexnější modifikace možnost přepínání pomocí jiných elementů než odkazů `<a>`. Výsledkem tedy byly dvě několika řádkové funkce.

3.3 Začlenění šablony do systému WayBack

Organizace a kompilování systému WayBack je realizováno přes nástroj řízení projektů *Apache Maven* [15], který se automaticky stará o správu *reportů*, dokumentace a řízení kompilace projektu na základě zdrojového centrálního modelu POM (Project Object Model). Koncept POM využívá jako ústřední zdroj informací XML soubor, kde je definována struktura projektu a závislosti na externích balících a knihovnách. Na rozdíl od klasické Java aplikace odpadá potřeba individuálně přikládat knihovny do seznamu knihoven využívaných balíky. Architektura projektu řízeného nástrojem Maven je stromová a modulární, tedy v kořenovém POM souboru jsou definovány dílčí podprojekty a každý z těchto podprojektů má svůj vlastní POM soubor. Výsledkem je možnost celý projekt přeložit a spustit jediným příkazem nad kořenovým POM souborem. Maven automaticky stáhne přes Internet závislé balíky, přeloží dílčí podprojekty a následně celý projekt vystaví, například v případě webové aplikace na server, a spustí. Celý proces je tak velmi zjednodušen a v kombinaci s nástroji správy verzí[6] umožňuje mezi různé členy vývojového týmu jednoduše distribuovat plně funkční verzi systému. Maven také vygeneruje HTML stránky i s dokumentací, kterou vystaví na server.

Nástroj WayBack je v systému Maven rozdělen na podprojekty (moduly):

---

3. Seznam klíčových slov, příslušnost do kategorie, náhled, apod.
4. `http://jqueryui.com/`
5. `http://www.sunsean.com/idTabs/`
6. V projektu WebArchiv se využívá SVN, více na WWW: `http://subversion.tigris.org/`

Třídy jádra systému – balíky, které provádějí zpracování dat a zabezpečují logiku systému WayBack;

Třídy Hadoop – nástroj pro škálovatelné a distribuované výpočty nad rozsáhlým indexem archivu;

Distribuce Hadoop jako JAR knihovny;

Distribuce kompletního systému WayBack;

Webová aplikace WayBacku – slouží k prezentaci archivovaných stránek pro koncového uživatele.

Při spuštění Mavenu nad systémem WayBack se automaticky přeloží třídy jádra systému, třídy Hadoop a webová aplikace, vygenerují se z nich příslušné knihovny JAR a WAR, Maven stáhne balíky, na kterých je WayBack závislý, a vystaví webovou aplikaci na server. Webová aplikace WayBacku, jejíž modifikaci v této práci popíšu, má následovnou adresářovou strukturu:

```
\META-INF       -- soubory manifestu
\WEB-INF        -- hlavní adresář aplikace
    \classes    -- Java třídy
    \exception  -- Java výjimky
    \query      -- JSP pro dotazování WayBacku
    \replay     -- JSP pro zobrazování záznamů WayBacku
    \template   -- univerzální soubory šablony
    \css        -- kaskádové styly
    \images     -- obrázky
    \jflot      -- knihovna pro generování časové osy
    \js         -- ostatní javascriptové knihovny
    \jsp        -- ostatní JSP stránky
```

Abych minimalizoval nutnost radikálního zásahu do systému, pozměnil jsem pouze soubor `/WEB-INF/replay/Toolbar.jsp` a přidal jsem do struktury podadresář `/wbtoolbar`, ve kterém se nacházejí všechny obrázky, CSS a JS soubory pro šablonu a JSP soubory[7].

V souboru `/WEB-INF/replay/Toolbar.jsp` byl původně skript pro generování uživatelské lišty. Tento skript jsem nahradil vlastní direktivou pro vložení upravené lišty v značkovacím jazyce JSP:

7. Java Server Pages, více viz `http://www.oracle.com/technetwork/java/javaee/jsp/index.html`

```
<jsp:include page='<%= "/wbtoolbar/toolbar.jsp" %>'>
    <jsp:param name="active_tab" value="metadata" />
</jsp:include>
```

Tato direktiva v místě jejího umístnění vloží do toku stránky obsah souboru `/wbtoolbar/toolbar.jsp` s defaultním parametrem pro zobrazení záložky detailu záznamu. Lišta obsahuje následující záložky:

Katalog (collections) – Katalog kolekcí generován ze systému WA Admin, zobrazeny jsou zde primární kategorie a sekundární kategorie;

Domů (home) – Úvodní stránka WayBacku poskytující základní informace o archivu a katalogu kolekcí;

Klíčová slova (keywords) – Seznam klíčových slov ve formátu mraku klíčových slov, kterými jsou záznamy označeny;

Detail (metadata) – Detail záznamu archivu, který zobrazuje k aktuálně zobrazovanému archivu relevantní data, a pokud se záznam nachází také v katalogu kolekcí, relevantní data z tohoto katalogu;

Vyhledávání (results) – Výsledky vyhledávání ve fulltextu, nebo v katalogu na základě vyhledávacího dotazu uživatele.

V záhlaví lišty se také zobrazuje formulářové textové pole sloužící jak pro vložení vyhledávacího dotazu, tak pro zobrazení URL aktuálně prohlíženého záznamu.

Kapitola 4

Popis implementace systému

Systém, kromě zmiňovaných souborů šablony a JSP, také obsahuje balíky[1]:

models – logická reprezentace databázového modelu

util – pomocné třídy

4.1 Balíky

4.1.1 Models

Katalog kolekcí je vytvořen v databázovém systému MySQL[2] s využitím relačních vztahů *enginu* InnoDB [10]. Autor databázového modelu měl na výběr mezi enginy MyISAM a InnoDB a zvolil systém InnoDB [2, str. 24] kvůli jeho schopnosti výjadřit relaci mezi jednotlivými entitami pomocí cizích klíču, čímž je zaručena lepší integrita databáze než u enginu MyISAM, který cizí klíče nepodporuje. Úskalí tohoto výběru je zásadní v tom, že InnoDB zase neumožňuje vytvářet fulltextové indexy nad atributy entit[3]. Absence této možnosti se projevuje nejvíce ve funkcionalitě fulltextového vyhledávání nad bází dat katalogu kolekcí, kde místo relevantního a sofistikovaného vyhledávácího algoritmu, který je podporován systémem MyISAM, je nutno použít pouze základní omezené SQL příkazy. Engine MyISAM podporuje SQL syntax[4]:

```
SELECT * FROM articles
    WHERE MATCH (title, body)
    AGAINST ('keyword' IN NATURAL LANGUAGE MODE)
    ORDER BY score DESC;
```

1. Java packages
2. http://www.mysql.com/
3. http://dev.mysql.com/doc/refman/5.0/en/innodb-restrictions.html
4. Natural language mode, http://dev.mysql.com/doc/refman/5.5/en/fulltext-natural-language.html

Tento příkaz vrátí seznam výsledků vyhledávání klíčového slova nad atributy `title` a `body` entity `articles` obohacený o identifikátor `score`, který určuje míru relevance záznamu vůči klíčovému slovu (klíčovým slovům). Funkce pro práci s fulltextem poskytují dále jazykový mód IN BOOLEAN LANGUAGE MODE (umožnění přikázání nebo vyloučení některých slov z dotazu), expanzi dotazu (umožnění vyhledání i záznamu, který neobsahuje všechna dotazovaná slova), stop-slova, omezení na dotaz (např. minimální délka dotazu) a další. Pro veškerý výčet možností a funkcí lze čtenáře odkázat na dokumentaci[5]. O tyto možnosti je systém InnoDB ochuzen. Existují různé možnosti jak tento problém řešit, ale nacházejí se za hranicí rozsahu této práce. Pro motivaci čtenáře pouze uvedu některé z nich:

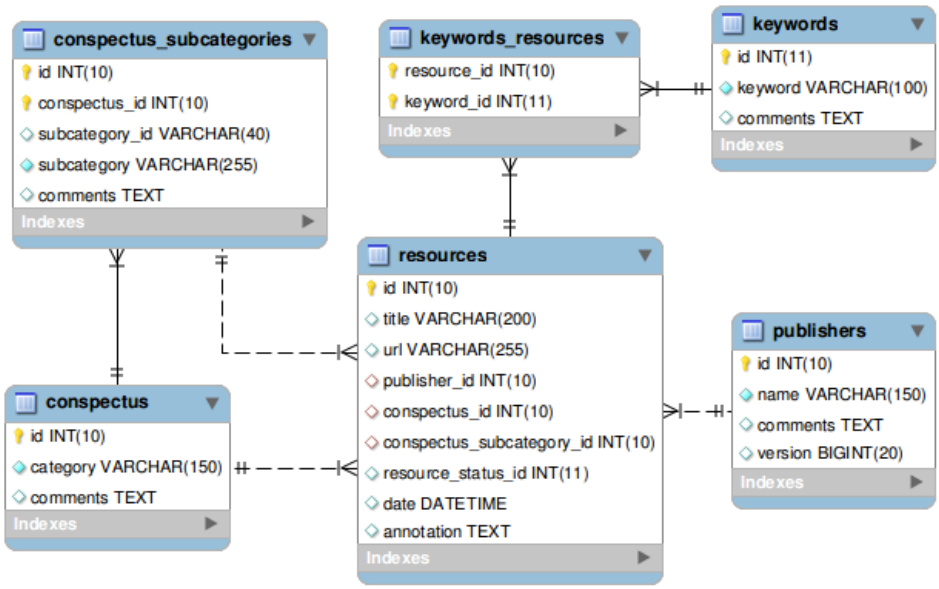
- Externí vyhledávací knihovna, například Apache Lucene[6], Sphinx[7] nebo Xapian[8], kde je potřeba z databáze MySQL InnoDB načíst textová data a vyexportovat do vyhledávací knihovny, která nad daty postaví index, nad kterým vyhledává a vrací výsledky. Komunikace může probíhat např. přes protokol XML RPC[9]
- Vytvoření dočasné (*temporary*) tabulky v enginu MyISAM[10], s fulltextovými indexy nad určitými atributy, do které se vloží data a následně se provede vyhledávací dotaz nad touto tabulkou. Toto řešení je možno skloubit se sadou *triggerů*[11], které budou upravovat dočasnou tabulku podle právě prováděné akce. Nástin řešení v jazyce SQL:

```
CREATE TEMPORARY TABLE temporary_myisam_table
    (FULLTEXT INDEX (attributeA,attributeB,...))
        Engine=MyISAM (SELECT * FROM
            original_innodb_table);

CREATE TRIGGER fulltext_refresher AFTER INSERT
    ON original_innodb_table
        INSERT temporary_myisam_table
            (SELECT * FROM original_innodb_table
```

5. http://dev.mysql.com/doc/refman/5.1/en/fulltext-search.html
6. http://lucene.apache.org/java/docs/index.html
7. http://sphinxsearch.com/
8. http://xapian.org/
9. http://www.techpresentations.org/Lucene_with_MySQL
10. MySQL umožňuje i v databázi na engine InnoDB vytvářet MyISAM tabulky.
11. Spouštěč definované akce při změně, vytvoření, nebo smazání záznamu.

```
                    WHERE id=last_insert_id()
                        ON DUPLICATE KEY UPDATE);

SELECT * FROM temporary_myisam_table WHERE
    MATCH(attributeA, attributeB, ...)
        AGAINST ('search string');
```

Řešení vyhledávání v systému WBToolbar popisuji podrobněji v kapitole 4.3.3 na stránce 41. Databázový model katalogu kolekcí v systému WA Admin zjednodušený pro potřeby mnou vyvíjeného systému vypadá následovně:



Obrázek 4.1: Zjednodušený databázový model systému WA Admin.

Lze pozorovat, že ústřední tabulkou celého databázového modelu je entita `resources` se svými `resources_status`, k níž jsou přes relační vztahy napojeny entity `keywords` (přes `keywords_resources`), dále pak `publishers`, `conspectus` a jeho podkategorie `conspectus_subcategories`. Zbývající entity modelu slouží pro interní fungování systému WA Admin, pro systém WBToolbar jsou nerelevantní a tudíž je na obrázku 4.1 neuvádím[12].

12. V případě zájmu může čtenář naleznout kompletní model v [2, str. 21, obr. 4.1].

Balíky systému WBToolbar respektují konvenci pro pojmenovávání balíků využívanou celým týmem WebArchivu[13], název balíku je tedy reverzní doména projektu WebArchiv následovaná hierarchií balíku v projektu WBToolbar:

- `cz.webarchiv.wbtoolbar.models`
- `cz.webarchiv.wbtoolbar.util`

Systém operuje s třídami modelů:

ConspectusModel – primární kategorie zdrojů, které vycházejí z metod knihovnického kategorizování. V čase psaní této práce se zde nachází 26 záznamů.

ConspectusSubcategoryModel – sekundární kategorie zdrojů (podkategorie). V čase psaní této práce se zde nachází 482 záznamů. Nejvíc jich patří do kategorie *Technika, technologie, inženýrství* v počtu 63 a nejméně do kategorie *Literatura pro děti a mládež*. Průměrný počet podkategorií na kategorii je 18.35 záznamu.

KeywordModel – klíčová slova jednotlivých zdrojů. V čase psaní této práce se zde nachází 5718 záznamů.

PublisherModel – vydavatel příslušného zdroje. V čase psaní této práce se zde nachází 4993 záznamů.

ResourceModel – zdroje. V čase psaní této práce se zde nachází 6399 záznamů, ale schválených kurátory je jich pouze 2713.

Základní představa o modelech je následující:

1. Model musí podporovat možnost vytvoření přes primární klíč záznamu a také přes objekt ResultSet[14], tedy přes řádek ve výsledku dotazu.
2. Z modelu se musí zadáním jednoho dotazu dát zjistit počet záznamů podle určité podmínky. Tato možnost se využívá pro inicializaci stránkovače (viz 4.1.2).

---

13. `http://download.oracle.com/javase/tutorial/java/package/namingpkgs.html`
14. `http://download.oracle.com/javase/1.4.2/docs/api/java/sql/ResultSet.html`

3. Podobným dotazem musí model vrátit seznam výsledků sebe sama podle určité podmínky.

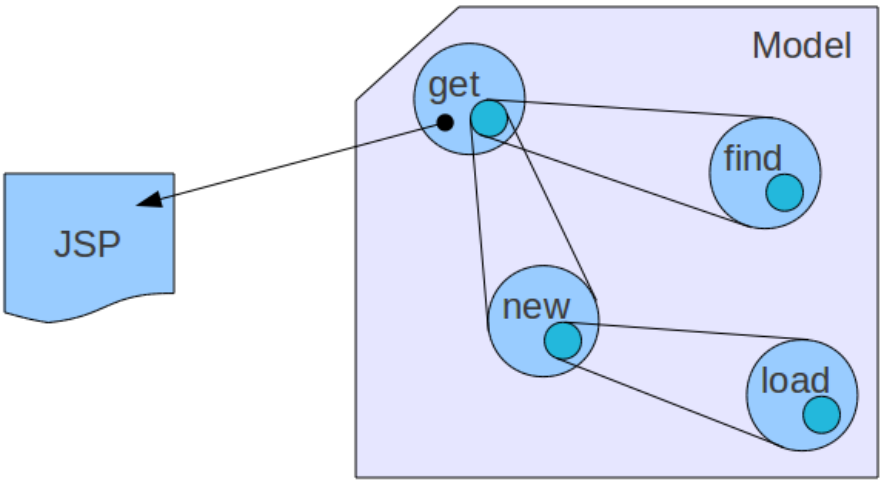
4. Pokud to dává smysl, model musí být schopen vrátit URL na sebe sama a také URL pro AJAXový dotaz.

Vzhledem k nesourodosti jednotlivých modelů nemělo smysl využívat dědičnost a společné metody. Typický model tedy vypadá následovně (pseudo-kód):

```
třída MyModel {
    privátní atributy, většinou atributy entity modelu

    constructor MyModel(ResultSet rs) {
        load(rs);
    }

    constructor KeywordModel(int id) {
        load(id);
    }

    metoda void load(int id) {
        rs = získej ResultSet z db
        load(rs)
    }

    metoda ResultSet find(Podmínka,
            Limit, Offset) {
        rs = získej ResultSet z db
        return rs
    }

    metoda LinkedList get(Podmínka,
            Limit, Offset) {
        list = new LinkedList ()
        pro každý rs = ResultSet
            z this.find(Podmínka, Limit, Offset) {
                list.add(new MyModel(rs))
        }
        return list
    }
```

```
    metoda int count() {
        počet = zjisti z db počet
        return počet
    }

    metoda String getURL() {
        return "?section="
           + WBRequest.instance().getUrl(MyModel.class)
           + "&keyword_id=" + this.id;
    }

    metoda String getAJAXURL() {
        return "AJAXový prefix" + this.getURL
    }
}
```

Grafické znázornění spolupráce metod uvnitř modelu:



Obrázek 4.2: Spolupráce metod uvnitř modelu.

Modely vesměs poskytují metody pro výběr záznamů na základě určitých kritérií, vrácení jednotlivých atributů a další metody, které budu popisovat v průběhu práce na místech k nim se vztahujících.

4.1.2 Util

Balík Util obsahuje podpůrné třídy pro chod aplikace. Implementovány jsou zde:

Pager – stránkovač, třída poskytující univerzální metody pro stránkování záznamů. Při jeho inicializaci je objektu předána proměnná rozhraní `HttpServletRequest`[15] a celé číslo *totalItems* (celkový počet záznamů ke stránkování). Informace o aktuální stránce, na které se uživatel nachází, se předávají metodou *GET* v URL dotazu formou parametru `page={číslo stránky}`. Číslo stránky si objekt Pager zjistí sám[16]. Dále má předdefinován počet záznamů na jednu stránku (5 záznamů) a počet stránek, které má zobrazit ve stránkovači (10 stránek). Algoritmem si spočítá, která je stránka následující, předchozí (a jestli vůbec existuje) a vygeneruje SQL hodnoty pro `limit` a `offset`. Soustavou vracejících metod (tzv. *getterů*) pak tyto hodnoty a také URL jednotlivých stránek poskytuje JSP stránce vyhrazené pro vygenerování Pageru.

WBDB – zapouzdření připojení k databázi, z kterého lze statickou metodou `WBDB.conn()` získat aktuální připojení k databázi.

WBRequest – poskytuje zejména metody pro generování URL, které propojují systém WayBack se systémem WBToolbar.

4.2 Control flow systému

Systém WBToolbar se začleňuje do WayBacku ve výše zmiňované direktivě (viz 3.3 na stránce 20), která nahrazuje původní lištu. Tato direktiva vkládá do těla výstupu obsah JSP stránky `/wbtoolbar/toolbar.jsp`, konkrétně vyhledávácí formulář, primární záložky a pokud se uživatel nachází na detailu záznamu v archivu, také informace o něm. Záložky jsou řešeny modulárně a každá z nich disponuje dvěma JSP soubory – `tab.jsp` a `content.jsp`. JSP soubory jsou uspořádány ve struktuře:

```
\wbtoolbar
    \jsp
        \tabs
```

15. Proměnná poskytující hodnoty předávaných parametrů při požadavku o změnu stránky, více na `http://download.oracle.com/javaee/5/api/javax/servlet/http/HttpServletRequest.html`
16. Metodou `HTTPServletRequest.getParameter("page")`.

```
\{identifikátor záložky}
    \tab.jsp
    \content.jsp
```

Soubor `tab.jsp` obsahuje HTML kód pro obsah přepínače záložky a soubor `content.jsp` obsahuje tělo záložky.

Vzhledem k formátu URL konkrétní archivované stránky ve WayBacku není možné smysluplně předat přes URL jakýkoliv parametr. URL je totiž tvaru:

```
http://localhost:8080/wayback
    /20110508193616/http://fajnestromy.cz/
```

Formálně zapsáno:

```
{protokol}://{server}:{port}/{prefix}
    /{timestamp}/{url_záznamu}
```

Na první pohled se může zdát, že by bylo možné předávat parametry přes *query string*[17], ale `{url_záznamu}` může obsahovat jakoukoliv URL, včetně query stringu. Bylo by tedy pravděpodobné, že by docházelo ke konfliktům, kupříkladu úplně triviální duplicita otazníků jakožto oddělovače URL a query stringu. Všechny URL jsou ve WayBacku směřovány přes tzv. *AccessPoint*[18], tudíž by jednou z možností mohla být modifikace směřování URL v třídě AccessPoint na formát např.:

```
{protokol}://{server}:{port}/{prefix}
    /{timestamp}/{parametry}/{url_záznamu}
```

Na tuto úpravu navazuje další komplikace – WayBack u prezentovaného záznamu modifikuje všechny `href` atributy odkazů a přidává jim prefix své URL, bylo by tedy nutné modifikovat i odpovídající třídu přepisování odkazů. Tyto zásahy do systému jsem zhodnotil jako neodpovídající představě o míře maximalizace vyhranění WBToolbaru jakožto pluginu do WayBacku.

Při tomto zhodnocení jsem také bral v úvahu možnost, která je z programátorského hlediska čistší a přináší s sebou i další výhody – zejména pro komfort uživatele – dynamická změna obsahu bez obnovení stránky přes AJAX. URL archivovaného záznamu jsem tedy ponechal v původním stavu

17. Řetězec parametrů, nacházející se v URL za znakem „?“.
18. Ústřední přístupový bod pro směrování parametrů v URL.

a rozhodnul se pro dynamické načítávání obsahu AJAXem. Využil jsem metody javascriptové knihovny jQuery, konkrétně tedy metody `$.ajax()` a `.load()`. Metoda `$.ajax()` je v žebříčku komplexnosti metod pro práci s AJAXem v knihovně jQuery na prvním místě, tedy nejkomplexnější [17], a metoda `.load()` na posledním, tedy nejjednodušší. Další metody jsou: `$.getJSON()`,`$.getScript()`,`$.get()`,`$.post()`. Přehled možností jednotlivých metod je znázorněn v tabulce 4.1 (Pokud je v sloupci Data uvedeno „všechny“, jedná se o formáty: XML, HTML, JSON, JavaScript, JSONP, text):

Název	GET	POST	Data
`.load(url,data,callback)`	ano	ano	html
`$.getJSON(url,data,callback)`	ano	ne	JSON
`$.getScript(url,callback)`	ne	ne	JavaScript
`$.get(url,data,callback,type)`	ano	ne	všechny
`$.post(url,data,callback,type)`	ne	ano	všechny
`$.ajax(options)`	ano	ano	všechny

Tabulka 4.1: Přehled možností jednotlivých metod knihovny jQuery pro práci s AJAXem.

Nejdříve jsem v aplikaci naimplementoval zvlášť AJAXové příkazy pro každý aktivní prvek (tedy prvek, po kliknutí na nějž by mělo dojít k změně obsahu). Vznikl tak poměrně velký objem JavaScriptového kódu vůči funkcionalitě, která byla vesměs pro všechny položky stejná – potřebné bylo pouze zaktivnit, nebo zneaktivnit ovladač záložky a změnit obsah v ní načtený. Každá záložka proto dostala unikátní identifikátor, resp. HTML atribut `id`. Každý aktivní prvek dostal CSS třídu `.ajaxer`, aby jQuery funkce poznala, že se jedná o aktivní prvek. Dále dostal HTML atribut `href`, ve kterém se nachází URL, z níž se má načíst nový obsah pro záložku. A nakonec dostal HTML atribut `rel`, jehož hodnotu tvoří identifikátor záložky, která se má zaktivnit.

Pak byla potřeba vytvořit univerzální funkci, která vezme dva parametry, identifikátor záložky a URL nového obsahu, zaktivní ovladač záložky a načte její obsah. Výsledkem je implementace funkce relink:

```
function relink(id, url) {
    $(".tab").removeClass("active");
```

```
    $("#content").fadeOut();
    $("#content").load(url);
    $("#content").fadeIn();

    $("#tab-" + id).addClass("active");
}
```

Tato funkce v posloupnosti:

1. Odstraní přepínač aktivnosti ze všech záložek.
2. Skryje aktuální obsah pomocí efektu postupného skrytí.
3. Načte nový obsah.
4. Zobrazí nový obsah pomocí efektu postupného zobrazení.
5. Nastaví přepínač aktivnosti na aktuální záložku dle parametru.

Posledním krokem pro dynamickou změnu obsahu bylo vytvoření řídící funkce jQuery:

```
$(".ajaxer").live("click", function() {
    relink($(this).attr("rel"), $(this).attr("href"));

    return false;
});
```

Tato funkce má na starosti zaregistrování handleru `click` pro všechny elementy s třídou `ajaxer` s parametry `rel` (identifikátor záložky) a `href` (URL nového obsahu). Zde bylo nutno použít pro registraci handleru nikoliv standardní funkci `.bind("click", ...)`, nebo také zkráceným zápisem `.click()`, ale funkci `.live("click", ...)` právě z důvodu změny obsahu stránky i po jejím úplném načtení. Ještě připadala do úvahy funkce `.delegate()`, ale tu jsem kvůli její nadbytečné robustnosti zamítl. Nevylučuji ale její využití při případných úpravách aplikace, když by bylo nutno využít sofistikovanější řízení událostí. Řídící funkce jQuery nakonec vrací logickou hodnotu `false`, což má smysl, pokud je třída `.ajaxer` (a také atribut `rel` a `href`) aplikována na HTML tag `<a>`, který by jinak pokračoval na hodnotu uvedenou v atributu `href` a došlo by tak k přesměrování stránky pouze na obsah stránky bez možnosti návratu zpět. Z předchozí věty čtenář lehce vydedukuje, že v aplikaci používám atributy `rel` a

`href` i na jiné HTML elementy než na odkazy, což není v souladu se standardem HTML a už vůbec ne se standardem XHTML 1.0, který WayBack deklaruje na začátku svého HTML výstupu. K tomuto nestandardnímu použití mě vedly následující důvody:

Přehlednost a rychlost zpracování – bylo by možné vytvořit subelement elementu s třídou `.ajaxer`, který by měl jako HTML obsah hodnotu parametru `rel` nebo `href`. Příklad:

```
<div class="ajaxer">
    <span class="rel">collections<\span>
    <span class="href">http:\\...<\span>
    ... obsah elementu ...
<\div>
```

a JavaScript

```
$(".ajaxer").live("click", function() {
    relink($(this).find('.rel')[0].html(),
        $(this).find('.href')[0].html());
    return false;
});
```

Metody `.find()` a `.html()` jsou obě výpočetně náročné. Metoda `.find()` prochází celou vnitřní strukturou elementu a vrací seznam všech subelementů se zvoleným selektorem. Naproti tomu metoda `.attr()` vrátí první atribut, který vyhovuje podmínce selekce. Dále metoda `.html()` nemusí být podporovaná všemi prohlížeči (zejména Internet Explorer[19]) a také se vyznačuje nekonzistentním výstupem napříč ostatními prohlížeči kvůli internímu využití javascriptové funkce `innerHTML()`, která může způsobit následující problémy [6]:

- Některé atributy bez uvozovek;
- HTML tagy velkými písmeny;
- Koncový tag `</li>` pouze na posledním elementu;
- a další.

19. `http://windows.microsoft.com/en-US/internet-explorer/products/ie/home`

Řešením by mohla být knihovna *innerXHTML*[20], která využívá pro čtení HTML obsahu manipulaci čistými *DOM*[21] funkcemi, ale na druhou stranu přidává k náročnosti na zpracování dotazu, jehož rychlá odezva je hlavně u AJAXu důležitá z důvodu očekávání uživatele okamžité odpovědi stránky, která se neobnovuje.

Navíc by byla potřeba přes CSS styly tyto subelementy skrýt.

WayBack jako takový na více místech porušuje normu ještě závažnějším způsobem – nesprávné, nebo žádné ukončení tagů, zastaralé (*deprecated*) atributy atp.

Tato funkcionalita je využita u prohlížení klíčových slov a listování katalogem. Pro její zaručení postačuje již zmiňovaná metoda `.load()`, která jednoduše načte a nastaví obsah elementu. Náročnější požadavky na možnosti AJAXu má vyhledávání – tedy odeslání formuláře, rozhodnutí, zda je dotaz URL, nebo klíčové slovo. A dále adekvátně k tomu zobrazit buď detail archivovaného záznamu, nebo výsledky vyhledávání. Implementace řídící funkce jQuery tedy využívá metodu `$.ajax()`:

```
$("#searchform .submit").click(function() {
    var is_url = false;
    var q_val = $("input#searchform_q").val();
    $.ajax({
        type: "GET",
        url: "<%= WBRequest.PREFIX
            + "/wbtoolbar/jsp/urlchecker.jsp" %>",
        dataType: "json",
        data: {
            searchform_q: q_val
        },
        success: function(data) {
            is_url = data.is_url;

            if (is_url) {
                window.location.href =
                    "<%=WBRequest.
                    getWaybackSubmitUrl("")%>" + q_val;
            } else {
```

20. http://www.stevetucker.co.uk/page-innerxhtml.php
21. Document Object Model.

```
                var dataString = "section="
                    + $("input#section").val()
                    + "&searchform_q="
                    + $("input#searchform_q").val();
                $.ajax({
                    type: "GET",
                    url: "<%= WBRequest.PREFIX
                       + "/wbtoolbar/jsp/tabs/results/"
                       + "content.jsp" %>",
                    dataType: "html",
                    data: dataString,
                    success: function(data) {
                       $("#results").fadeOut();
                       $("#results").html(data);
                       $("#results").fadeIn();
                    }
                });
            }
        },
    });
    return false;
});
```

Funkce je složená ze dvou bloků. V prvním se skript dotáže servletu `URLChecker`, zda se jedná o řetězec ve formátu URL, nebo jestli uživatel zadal klíčová slova. Komunikace probíha přes formát JSON[22], který umožňuje jednoduché posílání informací ve formátu `{klíč:hodnota}`, kde klíč i hodnota jsou řetězce, nebo rekurzivně, kde klíč je řetězec a hodnota je další JSON objekt, například `{klíčA:{klíčB:hodnota}}`. Servlet vrací klíč `is_url`. Pokud je `is_url` pravdivé, proběhne přesměrování na dotazový servlet WayBacku `query`, kde se jako parametr `url` uvede uživatelem zadaná hodnota. V opačném případě se provede další AJAXový dotaz na výsledky vyhledávání, které se následně zobrazí popsaným postupem v 4.2 na straně 30.

4.3 Jednotlivé komponenty systému – záložky

V aplikaci se na více místech využívá stejná funkcionalita – prezentace záznamů a stránkování. Prezentace záznamů je stejná jak pro výsledky z full-

22. `http://www.json.org/`

textového vyhledávání v indexu NutchWAX, tak pro výsledky z vyhledávání a listování v katalogu kolekcí. Také se použije při prohlížení záznamů odpovídajících klíčovým slovům. Stránkování je stejné pro všechny výše uvedené prezentace záznamů a také pro zobrazení odpovídajících hodnot z WA Adminu pro detail záznamu v archivu (viz 4.3.4 na stránce 42). Zavedl jsem proto pro každé použití univerzální JSP soubory – `results.jsp` a `pager.jsp`. Control flow probíha následovně (zjednodušený pseudokód):

```
String identifier = "identifikátor_použití";
int početZáznamů = ResourceModel.count(Podmínka);

if (početZáznamů > 0) {
    pager = Pager.instance(request, početZáznamů);

    session.setAttribute("result_contents-"+identifier,
        ResourceModel.get(Podmínka, pager.getLimit(),
            pager.getOffset()));
    session.setAttribute("pager-" + identifier, pager);
}

<jsp:include page="/wbtoolbar/jsp/results.jsp">
    <jsp:param name="identifier"
       value="<%= identifier %>" />
</jsp:include>
```

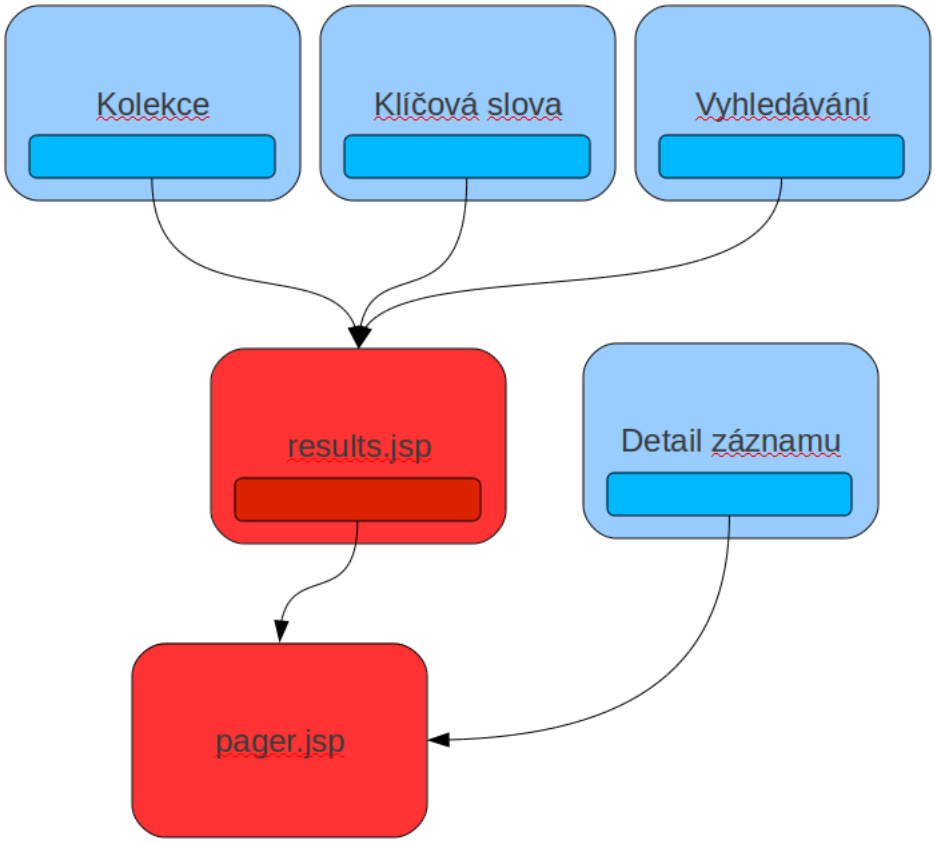
Pro zdařilý průběh je tedy potřeba do JSP souboru `results.jsp` předat identifikátor, o které použití se jedná[23], a zároveň nastavit do proměnné prostředí `session` hodnotu parametru `"result_contents-"+ identifier` a hodnotu parametru `"pager" + identifier`. První dvě hodnoty se načtou na začátku zpracování JSP skriptu `results.jsp`, třetí až na začátku načtení JSP skriptu `pager.jsp`. JSP `results.jsp` vypadá (zjednodušeně) následovně:

```
String id = request.getParameter("identifier");

LinkedList<ResourceModel> results_contents =
    (LinkedList<ResourceModel>)
        session.getAttribute("result_contents" + id);
```

23. Fulltextové vyhledávání v indexu, v katalogu, listování katalogem, klíčová slova atp.

```
for (ResourceModel result : result_contents) {
    // vypiš informace o záznamu
}
<jsp:include page="/wbtoolbar/jsp/pager.jsp">
    <jsp:param name="identifier" value="<%= id %>" />
</jsp:include>
```

Zde je vidět, že jsem se snažil kód vytvořit strukturovaně s ohledem na efektivní znovupoužitelnost kódu. Na obrázku 4.3 je znázorněné spoluvyužívání jednotlivých JSP souborů. JSP `pager.jsp` pouze vypisuje HTML reprezentaci stránkovače na základě objektu Pager ze session.



Obrázek 4.3: Spoluvyužívání jednotlivých JSP souborů.

4.3.1 Katalog

Listování katalogem je z pohledu změny obsahu lišty pro uživatele složeno prakticky z dvou kroků:

1. Výběr primární kategorie a volitelně sekundární kategorie (podkategorie);
2. Procházení seznamu odpovídajících záznamů.

V levém sloupci se nachází seznam primárních kategorií, u kterých se po najetí kurzorem změní seznam sekundárních kategorií v pravém sloupci pomocí jQuery kódu:

```
// Mouseover results
$(".result").live("mouseover", function () {
    $(".result-preview-switchable").hide();
    $("#"+$(this).attr('rel')).show();
    $(".result").removeClass("active");
    $(this).addClass("active");
});
```

Opět je zde použita funkce `.live()` místo funkce `.bind()` z důvodů, které jsem rozebíral v 4.2 na stránce 31. Pro každou kategorii je vygenerován blok `<div>` třídy `.result-preview-switchable` a zaktivňován pomocí identifikátoru dostupného z atributu `rel`. Při načtení stránky se všechny bloky skryjí pomocí funkce `.hide()` a pouze první záznam se zaktivní pomocí funkce `.show()`. Vzhledem k nízkému počtu kategorií (26 kategorií) není nutno obsah načítavat pomocí AJAXu. V horním oddílu obsahové části se nachází drobečková navigace, která uživateli zajišťujě představu o své pozici v struktuře během přechodu z přehledů kategorií (Krok 1.) na výpis položek konkrétní kategorie (Krok 2.).

Stěžejní část logiky komponenty je zapouzdřena v modelech. Pro výpis primárních kategorií se využívá metoda modelu `ConspectusModel`:

```
public static LinkedList<ConspectusModel>
    getAll(int limit, int offset);
```

Pro výpis sekundárních kategorií se využívá metoda modelu `ConspectusSubcategoryModel`:

```
public static LinkedList<ConspectusSubcategoryModel>
    getAll(int conspectus_id, int limit, int offset);
```

Zde lze pozorovat určité podobnosti a zároveň odlišnosti. Jak popisuji v 4.1.1 na stránce 25, modely zapouzdrují vesměs podobnou funkcionalitu, ale i tak jsem se rozhodl je naimplementovat bez dědičnosti a společných metod. Zdůvodňuji to snížením flexibility kódu a nutností volat místo elegantního volání jedné metody, volání neobratné a těžko čitelné (a v neposlední řadě také výpočetně náročné) univerzální metody. U velkých projektů tento přístup smysl má, u aplikace formátu WBToolbar je výhodnější se orientovat na srozumitelnost a rychlost kódu.

V druhém kroku, při prezentaci jednotlivých záznamů, se využívá metoda modelu `ResourceModel`:

```
public static LinkedList<ResourceModel>
    getAll(int categoryId, int subcategoryId,
        int limit, int offset);
```

Vzhledem k informacím zobrazovaným při prezentaci, jako jsou klíčová slova, vydavatel, a odpovídající kategorie, je potřeba vybrat SQL dotazem co nejvíc dat. Pro výběr seznamu zdrojů se tedy v této metodě využívá SQL dotaz:

```
String query = ""
    + "SELECT * FROM resources r "
    + "LEFT JOIN publishers p ON r.publisher_id=p.id "
    + "LEFT JOIN conspectus c ON r.conspectus_id=c.id "
    + "LEFT JOIN conspectus_subcategories cs "
    + "ON r.conspectus_subcategory_id=cs.id "
    + "WHERE ";
if (subcategoryId != 0) {
    query += "resources.conspectus_subcategory_id = '"
        + subcategoryId + "' ";
}
if (isBoth) { query += "AND "; }
if (categoryId != 0) {
    query += "resources.conspectus_id = '"
        + categoryId + "' ";
}
query += "ORDER BY resources.title ASC ";
if (limit > 0 && offset > -1) {
    query += "LIMIT " + offset + ", " + limit + "";
}
```

Dotaz spojí tabulky zdrojů, kategorií, podkategorií a vydavatelů dohromady. Podle potřeby případně omezí vybrané zdroje podle kategorie a (nebo) podkategorie, případně je také omezí početně a určí `offset`, od kterého se má začít číst pro potřeby stránkování. Pro tento účel je vhodné záznamy seřadit. Zde je použito řazení abecední, protože systém neposkytuje žádnou informaci o míře relevance nebo kvalitě záznamů.

Přechod mezi krokem 1 a 2 je realizován po kliknutí na příslušnou kategorii, nebo podkategorii AJAXovým načtením stránky, způsobem již popsaným v 4.2 na stránce 30.

Při procházení záznamů jsou uživateli zobrazeny následující informace:

Název zdroje – odpovídající atributu entity `resources.title`.

URL zdroje – odpovídající atributu `resources.url`. Směřuje na „živý“ web, tedy aktuální online verzi.

Archivovaná URL zdroje – směřuje na lokaci v rámci archivu.

Popis zdroje – odpovídající atributu `resources.annotation`.

Klíčová slova – množina klíčových slov uložených v entitě `keywords`. Relační vztah mezi klíčovými slovy a zdroji je `m:n` realizován přes vazební entitu `keywords_resources`. Dotaz na výběr odpovídajících klíčových slov ke zdroji je prováděn metodou `ResourcesModel.getKeywords()`, která interně načítá klíčová slova pomocí SQL dotazu:

```
SELECT * FROM keywords_resources kr
    LEFT JOIN keywords k~ON kr.keyword_id=k.id
    WHERE kr.resource_id={id};
```

Tento dotaz se provádí pro každý prezentovaný záznam. Hodnoty jsou aktivní, lze se tedy přes ně překliknout na jejich detail.

Katalogizace – odpovídající kategorie a (nebo) podkategorie. Výběr hodnot probíhá ve fázi výběru zdrojů spojením entit `resources` s entitami `conspectus` a `conspectus_subcategories`. Hodnoty jsou aktivní, lze se tedy přes ně překliknout na jejich detail.

Vydavatel – statická hodnota odpovídající záznamu `publishers.title` sloužící pouze jako doplňková informace pro uživatele.

Náhled – pokud existuje, zobrazí uživateli grafický náhled stránky.

4.3.2 Klíčová slova

Obdobně jako v katalogu je uživatelská prezentace složena z dvou kroků:

1. Výběr klíčového slova;
2. Procházení seznamem odpovídajících záznamů.

Klíčových slov je uložených v systému několik tisíc (5718 záznamů), proto by HTML výstup byl zdlouhavý a mohl by způsobovat přílišné paměťové nároky na klientský prohlížeč. Z tohoto důvodu jsem zvolil zobrazování pouze klíčových slov, které jsou použity u čtyř a více zdrojů[24], čímž jsem dosáhl zobrazení pouze 562 klíčových slov. Dotaz probíha pomocí SQL výrazu:

```
SELECT keywords.id, keywords.keyword,
    COUNT(keywords_resources.keyword_id)
    AS keyword_count FROM keywords
    LEFT OUTER JOIN keywords_resources
    ON keywords.id=keywords_resources.keyword_id
    GROUP BY keywords.id
    HAVING keyword_count>3 ORDER BY keyword ASC
```

Klíčová slova jsou prezentována pomocí abecedně seřazeného mraku (*tag cloud*) klíčových slov. Princip tkví v tom, že čím víc použití u zdrojů klíčové slovo má, tím je větším písmem, a naopak. Nastavil jsem zde navíc z důvodu čitelnosti omezení na maximální i minimální velikost písma. Algoritmus je následující:

```
// Maximální zvětšení písma v procentech
double maxPercent = 800;

// Omezení velikosti písma kvůli čitelnosti. Zde by mě-
// la postačovat hodnota maxPercent, ale rozprostření
// četnosti klíčových slov je vysoce nesourodé: první
// místo obsazuje klíčové slovo "česko" s četností 482,
// ale druhé místo jsou "turistické zajímavosti" s če-
// tností pouze 81. Klíčové slovo "česko" tak velmi vy-
// niká nad ostatními klíčovými slovy. Z druhé strany,
// pokud bych zvolil hodnotu maxPercent=300, grafické
```

24. Tuto hodnotu lze v systému změnit v konfiguračním souboru `properties`.

```
// rozlišení významnosti slova pomocí velikosti by bylo
// minimální.

// Maximálni velikost písma
double maxFontSize = 300;

// Maximální zmenšení písma v procentech
double minPercent = 80;

// Počet užití nejčetnějšího klíčového slova, ted' 481
double max = bestKeywordCount;

// Počet užití nejméně četného klíčového slova
double min = 3;

// Koeficient
double multiplier = (maxPercent-minPercent)/(max-min);

while (rs.next()) {
    double size = Math.min(
        minPercent + (
            (max - (max - (
                rs.getInt("keyword_count") - min))
            ) * multiplier
        ), maxFontSize
    );
}
```

Po kliknutí na klíčové slovo se AJAXovým dotazem, obdobně jako u katalogu (viz 4.2 na stránce 30), načte seznam odpovídajích zdrojů přes JSP soubor *results.jsp*.

4.3.3 Vyhledávání

Uživatel může vyhledávat nad dvěma bázemi dat, nad fulltextovým indexem NutchWAX, který je postaven nad celým archivem, nebo nad bází kolekcí systému WA Admin. V horní části uživatelské lišty je umístěno pole pro zadání vyhledávaného dotazu. Po zadání klíčových slov dotazu jsou uživateli zobrazeny dvě podzáložky – vyhledávání v kolekcích a vyhledávání ve fulltextu formou dvou separátních množin výsledků. Primárně je

zobrazena záložka vyhledávání v kolekcích a uživatel může po kliknutí na záložku „fulltext“ zobrazit výsledky vyhledávání ve fulltextu. Obě záložky sdílejí stejný vzhled, ale technologické pozadí pro získání výsledků je odlišné:

Fulltextové vyhledávání – Zdrojem dat je fulltextový index NutchWAX, který se dotazuje metodou GET vracející XML soubor s výsledky. Dotazovací URL je ve tvaru:

```
http://war.webarchiv.cz:8080/nutch/opensearch
    ?query={klíčová slova}&start={offset}
    &count={limit}
```

Vrácený XML soubor má formát uvedený v příloze C. V tomto výstupu jsou zajímavá následující data: *totalResults*, *startIndex*, *itemsPerPage* pro potřeby stránkování výsledků; *title*, *description*, *link*, *content*, *date*, *type* pro smysluplnou prezentaci výsledku pro uživatele; a *site* pro potřeby ověření příslušnosti záznamu do kolekce.

Vyhledávání v kolekcích – Vyhledávání v kolekcích probíha na základě SQL dotazu do databáze WA Admin, který je prováděn nad sloupci entity `Resources.title` a `Resources.annotation`.

4.3.4 Detail záznamu

U detailu záznamu je zobrazená časová osa z WayBacku `1.6.0`, stránkovaný seznam kolekcí na dané doméně a dodatečná metadata. Seznam kolekcí má svůj důvod – v databázi WA Admin jsou zdroje identifikovány přes URL startovní stránky zdroje, nikoliv přes výčet stránek, domény, nebo regulárního výrazu pro URL. Proto je možno porovnat záznam o kolekci s aktuální zobrazovanou stránkou pouze přes domény, protože v opačném případě by kolekce k zobrazovanému záznamu identifikována nebyla. Tuto situaci bude vhodné ukázat na příkladu. Je zobrazovaná stránka:

`http://www.geology.upol.cz/ostatni/odkazy.html`

Tato stránka je součástí kolekce (ve stejném kontextu jako kolekce):

`http://www.geology.upol.cz/paleo/paleografie.html`

Otázkou je, jak technologicky rozhodnout, jestli daná stránka do kolekce spadá, nebo ne. Možností by bylo „uřezat“ z URL název souboru `paleografie.html` a `odkazy.html` a porovnat, zda stránky mají shodný začátek:

```
http://www.geology.upol.cz/ostatni/ se nerovná
http://www.geology.upol.cz/paleo/
```

V tomto případě by zmíněný přístup nebyl vyhovující. Rozhodnul jsem se tedy porovnávat pouze domény (případně subdomény) a vypisovat u archivované stránky všechny k ní se vztahující kolekce. Na příkladě by to tedy vypadalo takto:

```
www.geology.upol.cz se rovná
www.geology.upol.cz
```

Vypsána by byla jedna kolekce, ale v případě, že by se další kolekce nacházela např. na:

```
http://www.geology.upol.cz/geo/magazin.html
```

Pak by byla vypsána i ta.

Závěr

Bohužel v současné době legislativa neumožňuje zobrazení archivu pro širokou veřejnost, proto jsou jí přístupné pouze sklízené zdroje s podepsanou smlouvou, a kompletní archiv je přístupný pouze na klientských stanicích poboček Národní knihovny. Do budoucna ovšem existuje předpoklad pro širší využití archivu, pokud by byl veřejně přístupný. Sem lze zahrnout využití pro novináře jako zdroj informací, pro právníky jako důkazní materiál, pro studenty a výzkumníky jako citační opora atp.

Výsledkem této práce je webová aplikace WBToolbar, která umožňuje potencionálnímu uživateli využít archivovaná data ve webovém archivu smysluplněji, než poskytuje stávající řešení. Stěžejní funkcionalita aplikace je založena na využití ručně vytvářeného katalogu kolekcí – dovoluje uživateli přehledně a interaktivně prohlížet seznam kategorií, podkategorií a zobrazení k nim odpovídajících záznamů. Dále aplikace poskytuje vyhledávací službu nad tímto katalogem kolekcí a také nad fulltextovým indexem napříč celým archivem. Důraz je kladen na propojenost jednotlivých komponent aplikace, uživatel se tedy lehce dostane z výsledků vyhledávání do konkrétní kategorie záznamu. A odtud například přes klíčové slovo na seznam všech záznamů se stejným klíčovým slovem.

Možností rozšíření je mnoho. K nejvýznamnějším bych řadil zmiňovanou úpravu fulltextového vyhledávacího algoritmu, generátor citací na archivované zdroje, který by umožňoval výběr formátu citace (BibTEX, DocBook, Word, apod.), prezentaci archivu jako portálu, kde bych doporučoval se inspirovat archivem United Kingdom Web Archive prezentujícího data opravdu přehlednou formou. Dále by byla možnost zapojit uživatele do procesu zkvalitnění obsahu archivu, kupříkladu možnost nahlásit závadný nebo technicky špatně sklizený obsah, případně ohodnotit kvalitu archivovaného záznamu.

Při psaní této práce jsem získal spoustu informací, zejména o asynchronním JavaScriptu a architektuře webové aplikace. Také jsem shledal nástroj Maven velice užitečným jako pomůcku při vývoji aplikace.

Literatura

[1] A. Brokeš. Projekt webarchiv – archiv českého webu, 2008. Dostupné z WWW: http://www.ics.muni.cz/zpravodaj/articles/578.html.

[2] A. Brokeš. Systém pro správu procesu archivace webových informačních zdrojů. Bakalářská práce, Masarykova Univerzita, Fakulta Informatiky, 2009.

[3] A. Brokeš. Integrace a automatizace systému v pracovních procesech projektu WebArchiv. Diplomová práce, Masarykova Univerzita, Fakulta Informatiky, 2011.

[4] M. Burner and B. Kahle. Internet archive arc files. [online], 1996. [cit. 14. 5. 2011]. Dostupné z WWW: http://www.archive.org/web/researcher/ArcFileFormat.php.

[5] J. Chaffer and K. Swedberg. *Learning jQuery*. Packt Publishing, Birmingham, 2009. 444 s.

[6] T. Hooper. jquery.html() vs. innerxhtml(). [online], 2007. [cit. 3. 4. 2011]. Dostupné z WWW: http://www.stainlessvision.com/jquery-html-vs-innerxhtml.

[7] S. Krug. *Web design: Nenuťte uživatele přemýšlet!* Computer Press, Brno, 2003. 152 s.

[8] J. Kunze. WARC: an Archiving Format for the Web. [online], 2005. 14 s. Dostupné z WWW: http://www.iwaw.net/05/kunze.pdf.

[9] L. Matějka. Zpřístupnění archivu českého webu. Diplomová práce, Masarykova Univerzita, Fakulta Informatiky, 2006.

[10] Oracle Corporation. Innodb storage engine. [online]. [cit. 14. 5. 2011]. Dostupné z WWW: http://dev.mysql.com/doc/refman/5.0/en/innodb-storage-engine.html.

[11] Secretariat New Zealand. Information and documentation of the warc file format. [online], 2008. [cit. 14. 5. 2011]. Dostupné z WWW: http://archive-access.sourceforge.net/warc/WARC_ISO_28500_final_draft%20v018%20Zentveld%20080618.doc.

[12] The Internet Archive. Heritrix crawler. [online]. [cit. 3. 4. 2011]. Dostupné z WWW: http://crawler.archive.org/.

[13] The Internet Archive. Nutchwax. [online]. [cit. 3. 4. 2011]. Dostupné z WWW: http://archive-access.sourceforge.net/projects/nutchwax/.

[14] The Internet Archive. Resourcestore configuration options. [online]. [cit. 3. 4. 2011]. Dostupné z WWW: http://archive-access.sourceforge.net/projects/wayback/resource_store.html.

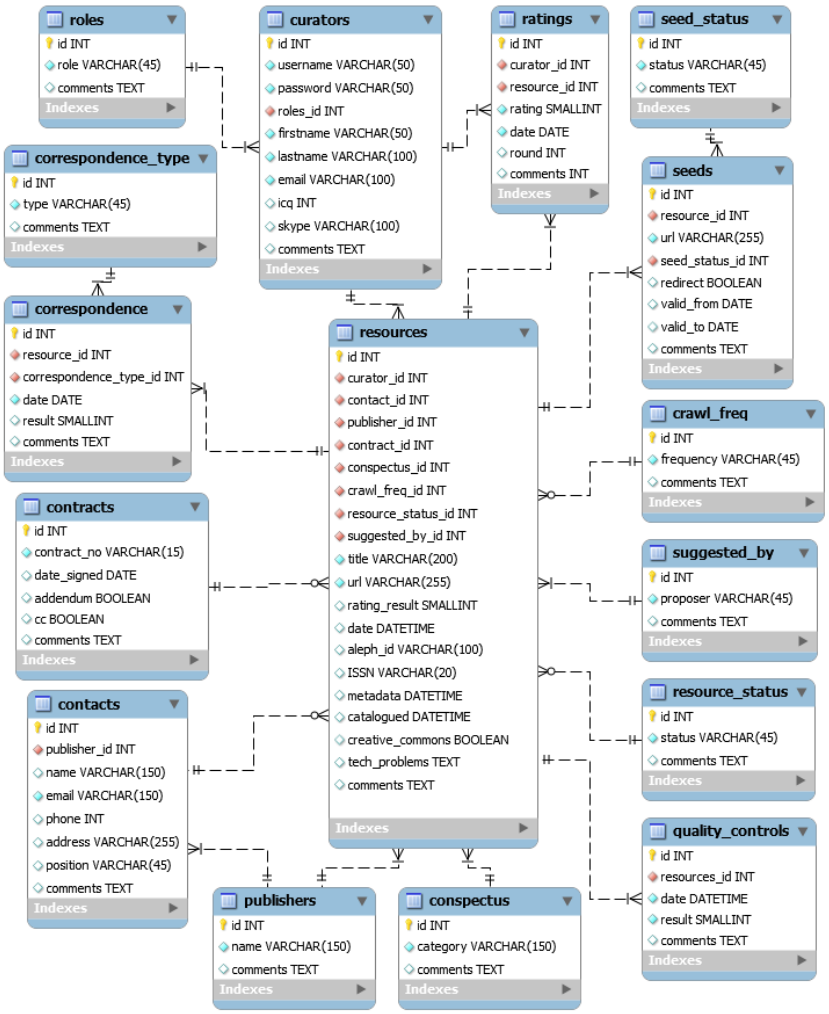
[15] J. van Zyl, B. Fox, J. Casey, B. Snyder, T. O'Brien, and E. Redmond. *Maven: The Definitive Guide*. O'Reilly Media, Silver Spring, 2008. 480 s.

[16] R. Voorburg. *Improve Quality Assurance for Selective Harvesting*. Nationale bibliotheek van Nederland, Vídeň, 2010.

[17] P. Yishi. 5 Ways to Make AJAX Calls with jQuery. [online], 2009. [cit. 1. 5. 2011]. Dostupné z WWW: http://net.tutsplus.com/tutorials/javascript-ajax/5-ways-to-make-ajax-calls-with-jquery/.

Příloha A

Kompletní model WA Adminu

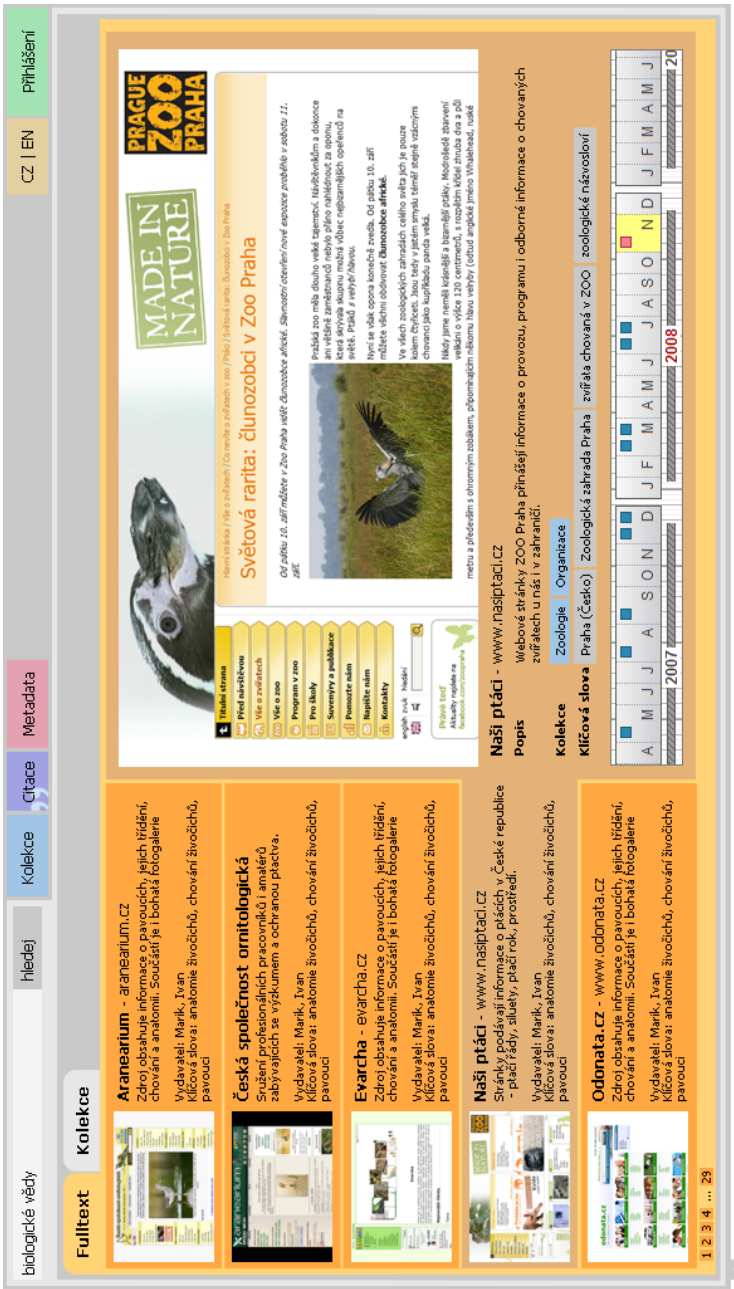


Obrázek A.1: Kompletní model WA Adminu.

Příloha B

Ukázka grafického rozhraní

Obrázek B.1: Grafické rozhraní.



Obrázek B.2: Drátěný model.

Příloha C

Formát XML výstupu z vyhledávání v NutchWAXu

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rss xmlns:nutch="..." xmlns:opensearch="...">
  <channel>
    <title>Nutch: {klíčová slova}</title>
    <description>
        Results for query: {klíčová slova}
    </description>
    <opensearch:totalResults>
        1877665
    </opensearch:totalResults>
    <opensearch:startIndex>0</opensearch:startIndex>
    <opensearch:itemsPerPage>
        10
    </opensearch:itemsPerPage>
    <nutch:query>{klíčová slova}</nutch:query>
    <nutch:responseTime>0.84</nutch:responseTime>
    <nutch:urlParams>
      <nutch:param name="query" value="dotaz"/>
    </nutch:urlParams>
    <item>
      <title>{Titulek výsledku}</title>
      <description>{Popis výsledku}</description>
      <link>{Adresa výsledku}</link>
      <nutch:site>{HOST výsledku}</nutch:site>
      <nutch:segment>20100408143726</nutch:segment>
      <nutch:content>{Obsah výsledku}</nutch:content>
      <nutch:digest>sha1:...</nutch:digest>
      <nutch:date>20090712223418</nutch:date>
      <nutch:type>text/html</nutch:type>
      <nutch:length>17881</nutch:length>
      <nutch:boost>3.5965972</nutch:boost>
    </item>
  </channel>
</rss>
```

Příloha D

Obsah přiloženého CD

wbtoolbar.zip – zdrojové soubory aplikace.

wbtoolbar.war – instalační WAR soubor.

thesis.pdf – vysázená práce v LaTeX.

thesis/ – adresář se zdrojovými soubory pro vysázení práce.