VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

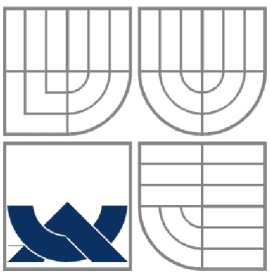
# ZPRACOVÁVÁNÍ WEBOVÝCH INZERÁTŮ V RUBY ON RAILS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

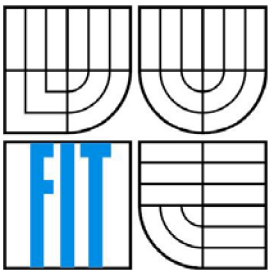
AUTOR PRÁCE
AUTHOR

MARIAN MRÓZEK

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# ZPRACOVÁVÁNÍ WEBOVÝCH INZERÁTŮ V RUBY ON RAILS

PROCESSING OF WEB ADVERTISEMENTS IN RUBY ON RAILS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARIAN MRÓZEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ZDENĚK LETKO

BRNO 2009

# Abstrakt

Cílem bakalářské práce bylo vytvořit aplikaci, která agreguje pracovní inzeráty z inzertních serverů na internetu. V určitých časových intervalech stahuje inzeráty, zpracovává je a ukládá do databáze. Následně tyto inzeráty jsou připraveny k zobrazení uživateli přes webové rozhraní. Aplikace je implementována pomocí frameworku Ruby on Rails.



# Abstract

The goal of this thesis was to create an application that aggregates job ads from the ad servers on the Internet. In certain time intervals adverts are downloaded , then processed and stored in a database. Subsequently, these ads are ready to display through a web interface. The application is implemented using Ruby on Rails framework.



# Klíčová slova

Ruby on Rails, databáze, MySQL, webová aplikace



# Keywords

Ruby on Rails, database, MySQL, web application



# Citace

Marian Mrózek: Zpracovávání webových inzerátů v Ruby on Rails, bakalářská práce, Brno, FIT VUT v Brně, 2009

# Zpracovávání webových inzerátů v Ruby on Rails

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Zdeňka Letka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Marian Mrózek
10. května 2009

## Poděkování

Rád bych touto cestou chtěl poděkovat Ing. Zdeňku Letkovi za odbornou pomoc při vytváření této bakalářské práce. Také bych chtěl poděkovat firmě Ataxo Czech s.r.o. a v neposlední řadě všem blízkým, kteří mi pomáhali a podporovali mně.

# Obsah

# 1 Úvod

V dnešní době, kdy světem vládne ekonomická krize se zvyšuje poptávka po práci a jedním z nejlepších zdrojů hledání práce je určitě internet, na kterém se nachází nespočet pracovních inzertních serverů. Aby uživatel nemusel vyhledávat a procházet všechny tyto servery jednotlivě, byla navržena tato webová aplikace, která agreguje inzeráty z těchto serverů do jednoho zdroje. Uživatel jednoduše zadá parametry hledání práce a aplikace zobrazí inzeráty ze všech agregovaných inzertních serverů nabízejících práci.

V následující kapitole se čtenář seznámí s použitými internetovými technologiemi, databázovými technologiemi, unifikovaným modelovacím jazykem, programovacím jazykem Ruby, šablonovacím systémem eRuby, návrhovými vzory, jazykem XPath a některými používanými aplikacemi při provozu webových aplikací. Celá třetí kapitola je věnována frameworku Ruby on Rails, kde se čtenář dozví jak zprovoznit základní aplikaci napsanou v tomto frameworku, jak používat generátory kódu a také se seznámí s některými rozšířeními pro tento framework. Čtvrtá kapitola se věnuje analýzou požadavků, která se dále dělí na funkční a nefunkční požadavky a graf případů užití. V páté kapitole je popsán návrh databáze a zobrazen Entitě-relační diagram. Šestá kapitola se věnuje implementaci aplikace. Tato kapitola se dále dělí do podkapitol, a to vytvoření řadiče a jeho metod, modelů aplikace, prezentační vrstvy aplikace, zprovoznění aplikace, přidání dalších inzertních serverů nabízející inzeráty práce a ukázky z aplikace. V závěrečné kapitole je popsáno zhodnocení výsledků a možnosti dalšího rozšíření aplikace.

# 2 Použité technologie

V této kapitole jsou popsány s některé technologie, které se používají při provozu webových aplikací.

## 2.1 Internetové technologie

### 2.1.1 Protokol HTTP

HTTP (HyperText Transport Protocol) [2] je protokol aplikační vrstvy, který slouží ke komunikaci mezi klientem a webovým serverem. Klientem je nejčastěji internetový prohlížeč, ale může jím být třeba vyhledávací robot nebo i jiný program. Protokol definuje tvar dat, která jsou přenášena, a pravidla dotazů a odpovědí komunikujících stran. Obvykle je spouštěn na portu 80.

### 2.1.2 Adresa URL

URL (Uniform Resource Locator) [2] je jednoznačné určení zdroje, způsob jak jednoznačně zapsat umístění souboru na internetu nebo intranetu. URL je synonymem pro internetové adresy jako například http://www.seznam.cz, která určuje protokol a umístění.

### 2.1.3 Značkovací jazyk HTML

HTML (HyperText Markup Language) [2] je značkovací jazyk pro hypertext, jedním z jazyků pro vytváření stránek v systému www, který umožňuje publikaci dokumentů na internetu. Používá se ve třech hlavních verzích a to: HTML 2, HTML 3.2 a HTML 4. HTML 4 existuje ve třech typech: přechodová (transitional), striktní (strict) a s rámy (frameset), přičemž nejpoužívanější je přechodová. V HTML se používá URL pro zacílení odkazů, načítaní obrázků a podpůrných souborů, které se nejčastěji přenášejí HTTP protokolem.

### 2.1.4 Značkovací jazyk XML

XML (Extensible Markup Language) [2, 11] je formou standardizovaného značkovacího jazyka, pomocí kterého je umožněno vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat. Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, nezabývá se sám o sobě vzhledem dokumentu nebo jeho částí. Jazyk XML nemá žádné předdefinované značky a jeho syntaxe je přísnější, než syntaxe HTML. V XML může mít každý element libovolné množství atributů. Atributy se většinou používají pro přidání různých metainformací k elementům. Deklarace jednotlivých atributů se skládá ze tří části. První část je jméno

atributu, za jménem následuje typ atributu a poslední část určuje standardní hodnotu atributu, popřípadě, zda je používaní atributu u daného elementu povinné. Deklarace atributu se opakuje pro každý atribut, který má být u elementu k dispozici.

### 2.1.5 Jazyk XPath

XPath (XML Path Language) [4, 11] je jazyk, pomocí kterého lze adresovat části XML dokumentu, vybírat jednotlivé elementy a pracovat s jejich hodnotami a atributy. Používá se v mnoha aplikacích XML. Základní součástí jazyka je výraz popisující cestu (path expression). Taková cesta se zapisuje oddělenými lomítky jako posloupnost přechodů mezi jednotlivými sadami uzlů.

Každý přechod je určen pomocí tří složek: osa (axis), test (node test) a predikát (predicate). Pokud některé složky mají implicitní hodnotu, nemusí být uvedeny. Nejjednodušší zápis, který se podobá zápisu cesty k souboru v souborovém systému, má tvar například `/A/B/C a` používá pouze položky test, přičemž označuje množinu elementů `C`, které jsou uvnitř elementů `B` a ty jsou uvnitř elementu `A`. U složitějších dotazů se místo implicitní osy typu potomek (`child`) může zapsat jiná osa, oddělená dvěma dvojtečkami, případně predikáty, uvedené jako seznam podmínek v hranatých závorkách. Příkladem takového dotazu může být `/A/B/following-sibling::*[1]`, který vybere všechny elementy, které jsou prvním elementem následujícím po elementu `B`, který je uvnitř kořenového elementu `A`. Specifikace osy popisuje směr pohybu po stromové reprezentaci XML dokumentu. Je definováno celkem 13 os, mezi které patří také osa `attribute` (atribut). Pro některé často používané osy existuje zkratka zápisu, například místo `attribute::id` lze zapsat `@id`. Kromě výrazů popisující cestu jsou v XPath definovány také běžné číselné a logické výrazy, které se používají nejčastěji v rámci predikátů. Element `div` s atributem `name` rovným `main` se tedy zapíše:

```
div[@name="main"]
```

### 2.1.6 Formátovací sady CSS

Formátovací sady CSS (Cascading Style Sheets) [2] je možno použít pro určení vzhledu webové stránky a určují pravidla, která definují prezentaci typu určitého vzoru, skupiny, či přesněji třídy značek, případně jedné značky. Pravidla formátovacích sad mohou být použita k definování různých vizuálních aspektů objektů stránky, včetně barvy, velikosti a umístění. Tato různá pravidla formátovacích sad je možné kombinovat v závislosti na použití značek.

### 2.1.7 Technologie RSS

RSS [4] je rodina XML formátů určených pro přenos a zpracování metadat, která má mnoho využití, jako například čtení novinek na webových stránkách. Samotná zkratka má více výkladů. RSS verze 0.9 od firmy Netscape znamená Rich Site Summary, RSS verze 2.0 Really Simple Syndication.
Aby uživatelé internetu nemuseli procházet všechny své oblíbené stránky s novinkami, zpravodajstvím nebo stránky, které se často aktualizují, lze použít RSS kanál, pokud tímto daný web disponuje. Umožňuje tedy uživateli z jednoho místa sledovat informace zveřejňované na různých webech. Ke čtení těchto zdrojů potřebujeme tzv. RSS čtečku. Jedná se o specializovaný program nebo rozšíření pro prácí s těmito zdroji. Uživatel zadá do čtečky URL příslušného zdroje, čtečka pak pravidelně kontroluje toto URL a zobrazuje nové položky.

RSS formát poskytuje obsah celého článku, případně odkaz na daný článek a další informace o článku jako autora nebo datum. Jedná se o soubor v XML formátu, který se nazývá RSS zdroj. Lze jej vytvořit ručně a nebo využít nástrojů určených pro generování XML souboru přímo z databáze podle specifikace RSS. Starší verze 0.9 může obsahovat pouze název příspěvku, popis a URL, kde příspěvek nalezneme. Novější verze 2.0 nabízí možnost zaslání dalších informací jakými jsou například autor příspěvku a datum jeho vydání.

### 2.1.8 Provázaní vlastností

Provázaní vlastností (Data Binding) [5] slouží k synchronizaci vlastností zdroje s provázanou vlastností. Jakmile se změní hodnota zdrojové vlastnosti, projeví se i v provázaných vlastnostech a naopak, pokud v provázaných vlastnostech dojde ke změně, tato změna se projeví zároveň ve zdroji.

Existují tři druhy svázaní podle způsobu synchronizace [4]. Buď se ovlivňuje zdroj s provázanou vlastností navzájem, tedy hodnoty jsou synchronizovány obousměrně nebo jsou hodnoty synchronizovány v jednom směru od zdroje a změny v provázané vlastnosti jsou zdrojem ignorovány a nebo hodnoty jsou synchronizovány pouze po startu, poté jsou všechny změny ignorovány.

Jakmile je požadována synchronizace zdroje s jiným objektem, je třeba zdroj se synchronizovaným objektem provázat. Příkladem může být RSS zdroj a RSS čtečka, která je se zdrojem svázaná pomocí URL v jednom směru od zdroje.

## 2.2 Databázové technologie

### 2.2.1 Dotazovací jazyk SQL

SQL (Structured Query Language) [4] je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. SQL příkazy se dělí do čtyř základních skupin: příkazy pro manipulaci s daty, příkazy pro definici dat, příkazy pro řízení přístupových práv a příkazy pro řízení transakcí.

### 2.2.2 Relační databázový systém MySQL

Jeden z nejrozšířenějších databázových systémů je MySQL (My Structured Query Language) [4], který byl vytvořen švédskou firmou MySQL AB. Jedná se o multiplatformní databázi s kterou komunikujeme pomocí SQL jazyka. MySQL bylo od počátku optimalizováno především na rychlost. V současné době se používá verze 5 a připravuje se verze 6.

Verze 5 podporuje cizí klíče, transakce, různé znakové sady a různá časová pásma v datech, poddotazy, uložené procedury, triggery, pohledy a práci s metadaty. MySQL nabízí několik typů databázových tabulek, které se liší svými možnostmi, použitím a způsobem ukládání dat do souborů. Nejpoužívanější typy jsou MyISAM, který nepodporuje transakce, a InnoDB s podporou transakcí.

## 2.3 Unifikovaný modelovací jazyk

Jazyk UML (Unified Modeling Language, unifikovaný modelovací jazyk) [3] je univerzální jazyk pro vizuální modelování systémů, který byl navržen proto, aby spojil nejlepší existující postupy modelovacích technik a softwarového inženýrství. Jazyk UML nenabízí žádný druh metodiky modelování, poskytuje pouze vizuální syntaxi, kterou můžeme využít při sestavování svých modelů.

Nejlépe adaptovaná metodika pro jazyk UML je UP (Unified Process) [3]. Metodika UP specifikuje pět základních pracovních postupů: požadavky, analýza, návrh, implementace a testování.

## 2.4 Programovací jazyk Ruby

Jazyk Ruby [1, 4] je plně objektový, interpretovaný skriptovací jazyk. Základní myšlenkou jazyka Ruby je co nejkomfortnější a zábavné programování, které by se přiblížilo přirozenému jazyku. Má integrované regulární výrazy, snadnou čitelnost a úspornou syntax.
Následující příklad vypíše tři krát slovo „Ahoj!“:

```
3.times { print "Ahoj! "} # Tři krát napiš ahoj
=> Ahoj! Ahoj! Ahoj!
```

Tedy v angličtině řečeno: „Three times print Ahoj!“, neboli „Tři krát napiš Ahoj!“. Číslo 3 označuje počet opakování a iterátor `times` se používá, je-li znám přesný počet požadovaných průchodů. Mezi složenými závorkami je blok programu, který se má opakovat. Na konci řádku se nepíše žádný oddělovač. V jazyku Ruby lze psát i komentáře a to za symbolem #. Protože jazyk Ruby je hodně čitelný a z kódu je většinou hned zřejmé k čemu daný úsek kódu slouží, není třeba psát tolik komentářů.

Pro vytvoření třídy v jazyce Ruby se použije příkaz `class`, za kterým následuje název třídy, který se zapisuje s velkým písmenem na začátku. Konstruktor třídy, který se používá ke konstrukci nových objektů, je metoda `initialize`. Volá se automaticky, pokud použijeme danou třídu pro vytvoření nového objektu. Před názvem metod se píše příkaz `def` a proměnné instancí se označují znakem `@` na začátku názvu proměnné, který oznamuje, že má hodnotu uchovávat, a nikoli opakovaně inicializovat při každém zavolání odpovídající metody. Dědění třídy se provádí zápisem: `class B < A`, tzn. že třída B dědí vše z třídy A.

Mezi výhody tohoto jazyka lze jistě zařadit přenositelnost zdrojového kódu mezi platformami. Oproti kompilovaným jazykům je pomalejší, protože interpretovaný jazyk je překládán až za běhu programu.

## 2.5 Šablonovací systém eRuby

Procesor označovaný eRuby [1], kde e značí Embedded neboli vestavěný Ruby se používá tam, kde potřebujeme vykonávat kód jazyka Ruby v HTML kódu. Jedná se o „normální“ Ruby, které se zapisuje mezi značky `<% %>` a nebo `<%= %>`. První z nich vykonává Ruby kód, ale nic nezobrazuje. Druhá vloží na své místo hodnotu posledního uvnitř provedeného výrazu.

## 2.6 Nástroje pro jazyk Ruby

Rake [7] je jednoduchý program napsaný v jazyce Ruby pro sestavování programů s podobnými schopnostmi jako má program make. Rakefile, tedy rake varianta Makefile, je kompletně definovaný standardní syntaxí jazyka Ruby. Spouští se pomocí příkazu `rake`, za kterým následuje název úlohy, kterou má provést.

Pomocí něj lze vytvářet například tzv. gem balíčky [7], což jsou knihovny pro jazyk Ruby, které se instalují pomocí baličkovacího nástroje RubyGems [7]. Zdroj odkud se může balíček stáhnout se přidává pomocí příkazu `gem sources`, za kterým následuje URL zdroje. Balíček se poté instaluje pomocí příkazu `gem install`, za kterým následuje název instalovaného balíčku.

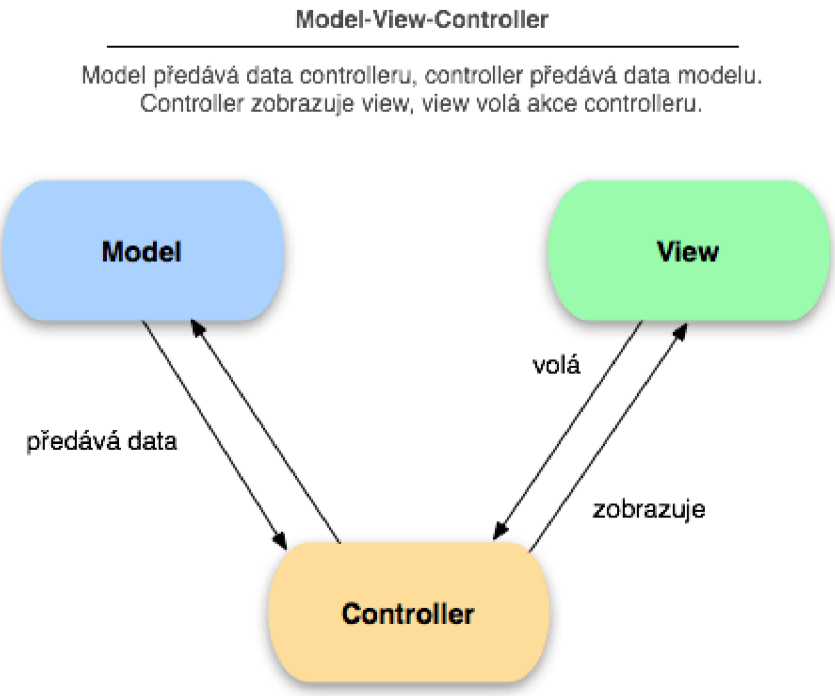
## 2.7 Návrhové vzory

Návrhový vzor představuje obecné řešení problému, které se využívá při návrhu programů.

### 2.7.1 Návrhový vzor MVC

MVC (Model-View-Controller, Model-Pohled-Řadič) [1, 4] je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent, kterými jsou model, pohled a řadič.

Model představuje doménově specifickou reprezentaci informací, s nimiž aplikace pracuje. Pohled (view) zodpovídá za zobrazení výsledku akce, převádí data reprezentovaná modelem do podoby vhodné k interaktivní prezentaci uživateli. Obsahuje grafické rozhraní aplikace. Řadič (controller) zpracovává požadavky, reaguje na události, typicky pocházející od uživatele, a zajišťuje změny v modelu nebo pohledu. Návrhový vzor Model-Pohled-Řadič je na následujícím obrázku, viz. Obrázek 2.1.




**Obrázek 2.1: Model-Pohled-Řadič. Převzato z [6].**

### 2.7.2 Návrhový vzor Active Record

Active Record [6] je návrhový vzor, který mapuje databázové tabulky na třídy a převádí řádky na objekty a sloupce na jejich atributy.

Například při výběru dat z tabulky, nemusí vývojáři psát databázové dotazy jako:

```
SELECT * FROM animals WHERE id = 5
```

Ale stačí vytvořit třídu `Animal`, která je zděděna z třídy `ActiveRecord::Base`:

```
class Animal < ActiveRecord::Base
end
```

Pak jednoduchým zápisem `Animal.find(5)` příkaz vrátí objekt s `id = 5` a s příslušnými atributy.

Active Record nabízí metody nejen pro vyhledání, ale i pro změnu, vytvoření a mazání záznamů. Změna atributu konkrétního objektu se provede vyhledáním objektu, změnou atributu a uložením objektu:

```
animal = Animal.find(1)
animal.name = 'Kelly'
animal.save
```

## 2.8 Použité aplikace

V této kapitole jsou popsány aplikace, které se můžou používat při provozu webových aplikací.

### 2.8.1 Mongrel server

Mongrel [4] je HTTP knihovna a webový server pro webové aplikace napsané v jazyku Ruby. Instaluje se jako RubyGems balíček příkazem `gem install mongrel`. A spouští se příkazem: `mongrel_rails start`.

### 2.8.2 Cron démon

Cron [10] je softwarový démon, který v operačních systémech na bázi Unixu slouží jako plánovač úloh, jenž umožňuje automatizovaně spouštět v určitý čas skript nebo program.

Pro nastavení intervalů spouštění skriptů a aplikací slouží utilita `crontab`. Příkazem `crontab -e` se otevře editor, kde lze psát úlohy pro spuštění démonem Cron podle těchto pravidel. Každý jeden řádek znamená jedna úloha, na kterém je zapsáno šest parametrů oddělených tabelátorem nebo mezerami: 1. minuta, 2. hodina, 3. den v měsíci, 4. měsíc, 5. den v týdnu (0-neděle, 1-pondělí, 6-sobota), 6. cesta k programu, který má démon Cron spustit.

Následující příklad spustí skript umístěný v `/home/marian/skript.sh` každý den ve čtyři ráno:

```
0 4 * * * /home/marian/skript.sh
```

# 3 Framework Ruby on Rails

Ruby je samostatný programovací jazyk a Rails [1] je rámec webových aplikací. Ruby on Rails („rubín na kolejích“) [1] je tedy framework pro vývoj webových aplikací, který je založen na jazyce Ruby. Autorem Rails je dánský programátor David Heinemeier Hansson.

Jeden z klíčových principů Rails je upřednostnění konvence před konfigurací. To znamená, že Rails vytvoří nejen základní aplikace, ale také je nastaví výchozími hodnotami tak, aby nebylo třeba vše konfigurovat od začátku. Jsou postaveny na bázi návrhovém vzoru model-pohled-řadič. Automaticky mapují URL na vnitřní řídicí prvky aplikace tzv. směrování. Abstrahují přístup k datům v databázi pomocí mapování záznamů z relační databáze na objekty prostřednictvím návrhového vzoru Active Record a obsahují pomocné knihovny pro snadné generování HTML.

## 3.1 Vložení kódu Ruby on Rails

Pro vytvoření aplikace pomocí frameworku Ruby on Rails je třeba vytvořit aplikační rámec pomocí příkazu `rails názevAplikace`. Rails tímto vytvoří potřebné soubory. Adresářová struktura pak vypadá následovně:

```
názevAplikace
|____ README
|____ app
|      |____ controllers
|      |____ models
|      |____ views
|      |____ helpers
|____ config
|____ components
|____ db
|____ doc
|____ lib
|____ public
|____ script
|____ test
|____ tmp
|____ vendor
```

Adresář `app` obsauje veškerý kód specifický pro tuto konkrétní aplikaci. V adresáři `app/controllers` se nalézají řadiče, které jsou potomky od `ApplicationController`, což je třída zděděna z `ActionController::Base`. V adresáři `app/models` jsou umístěny modely, které jsou většinou potomky `ActiveRecord::Base`. Šablonové soubory pro pohledy akce jsou

uloženy v adresáři `app/views`. Všechny pohledy využívají syntaxi eRuby. Adresář `app/views/layout` obsahuje šablonové soubory pro rozvržení používaná pohledy. Pomocné pohledy se ukládají do adresáře `app/helpers`. Pomocné prvky lze použít k zabalení funkčnosti pohledů do metod. V adresáři `config` jsou umístěny konfigurační soubory prostředí Rails. Adresář `components` obsahuje sebeobsažné miniaplikace, které mohou vázat řadiče, modely a pohledy. Soubory týkající se databáze jsou umístěny v adresáři `db`. Dokumentace se ukládá do adresáře `doc`. Jakýkoliv vlastní kód, který nepatří pod řadiče, modely, ani pomocné metody se ukládá do adresáře `lib`. Adresář `public` je k dispozici pro webový server, obsahuje podadresáře obrázků a šablon stylů. V adresáři `script` lze najít pomocné skripty pro automatizaci a generování. Externí knihovny na nichž aplikace závisí, se ukládají do adresáře `vendor`.

Spuštění aplikace se provede příkazem `ruby script/server`. Tímto se spustí webový server WEBrick na portu 3000 lokálního hostitele, který je součástí Rails. K aplikaci pak lze přistupovat na adrese `http://localhost:3000`. Po zadání této URL adresy do webového prohlížeče se objeví na webové stránce přivítání potvrzující, že uživatel je na Rails.

## 3.2 Propojení řadiče s pohledem

V Rails se kód Ruby v pohledech vykonává procesorem označovaným eRuby. Vytvořená proměnná instancí v určité akci určitého řadiče odkazuje na stejnou proměnnou instancí v šabloně pohledu připojenou k této akci. To znamená, že lze vykonat výpočty v akci a automaticky odeslat do pohledu, aniž by bylo potřeba nějaká zvláštní volání metod či exportování. V adresáři `app/views` se nachází podadresář s názvem řadiče a v něm jsou uloženy pohledy se stejným názvem a příponou `html.erb`, jako se jmenuje daná akce tohoto řadiče. Tímto je zajištěno, že Rails ví, který pohled patří jaké akci.

Následující příklad zobrazuje řadič `Hele` s akcí `tady`, která vytváří proměnnou instancí `@aktualni_cas` pomocí metody `Time.now`:

```
class HeleController < AplicationController
      def tady
            @aktualni_cas = Time.now
      end
end
```

Vytvořená proměnná instancí `@aktualni_cas` v akci `tady` se pak v šabloně pohledu připojené k této akci odkazuje na stejnou proměnnou instancí:

```
<html>
      <head>
            <title>Použití pohledů</title>
      </head>
      <body>
            Aktuální čas: <%= @aktualni_cas %>
      </body>
</html>
```

## 3.3 Směrování

Směrování (routing) [1] zajišťuje, že aplikace ví na jaké prvky má směrovat konkrétní požadavky. Směrovacím požadavkem pro webové aplikace může být URL. V současné době se lze setkat s přátelskými URL (tzv. pretty URL), které jsou čitelné nejen pro vyhledávače, ale i pro člověka. Jsou tedy srozumitelné a lépe zapamatovatelné. Taková přátelská URL může například vypadat:

```
http://localhost:3000/blog/clanek/2009/04/15
```

Podle adresy je odhadnutelné, že odkaz zobrazí článek publikovaný 15. dubna 2009. Aby i aplikace Rails věděla co má udělat, jsou direktivy řídicí směrování uloženy v konfiguračním souboru `config/routes.rb`. Pro tuto URL by vypadal směrovací záznam ve směrovacím konfiguračním souboru následovně:

```
map.connect 'blog/clanek/:rok/:mesic/:den',
        :controller => 'blog', :action => 'clanek'
```

Tímto se nastavuje jedinečné směrování umožňující zpracovávat kalendářní data v podobných adresách: `http://localhost:3000/blog/clanek/2009/04/15`. Každé směrování zpracovává jedinečný vzor URL, takže rámec Rails ví, jak přiřadit směrovací vzory skutečným adresám.

V akci `clanek` řadiče `Blog` je možné převzít atributy den, měsíc a rok prostřednictvím asociace `params`, které se ukládají do proměnných instancí `@den`, `@mesic` a `@rok`. Asociace `params` obsahuje v hranatých závorkách názvy atributů, které se píšou s dvojtečkou na začátku. Řadič `Blog` s akcí `clanek`, který je potomkem řadiče `AplicationController` pak vypadá následovně:

```
class BlogController < AplicationController
    def clanek
          @den = params[:den]
          @mesic = params[:mesic]
          @rok = params[:rok]
    end
end
```

Následně v šabloně pohledu akce `clanek`, lze zobrazit údaje `@den`, `@mesic` a `@rok` tímto způsobem:

```
<html>
    <head>
          <title>Trasovací požadavky</title>
    </head>
    <body>
    Den: <%= @den %> Měsíc: <%= @mesic %> Rok: <%= @rok %>
    </body>
</html>
```

## 3.4 Automatizované generátory kódu

Generátory kódu slouží k vytváření modelů, řadičů a jiných součástí aplikace, aby nebylo třeba například stále dokola vytvářet nové soubory s modely v příslušných složkách s náležitými jmény a vepisovat do nich kód dědění z třídy `ActiveRecord::Base`. Rails obsahují několik základních generátorů kódu, ke kterým lze přidat další generátory pomocí rozšíření. V adresáři `script` Rails aplikace se nachází skript `generate`, pomocí kterého lze generovat kód. Jedná se o skript napsaný v jazyku Ruby. Jako parametr skriptu `generate` se uvádí jaký kód generátor bude vytvářet:

```
ruby script/generate název
```

Některé základní generátory kódu jako `scaffold` nebo `migration` budou uvedeny v následujících podkapitolách.

### 3.4.1 Migrační generátor

Nástroj `migration` [6] slouží k vytvoření migračních souborů pro automatizované vytváření, změnu nebo mazání tabulek databáze, aniž bychom museli psát SQL příkazy. Každá provedená

migrace má svou vlastní číslovanou verzi. Po provedení migrace se navýší číslo verze migrace a odebráním migrace se provedené změny vrátí zpět do původního stavu, jsou tedy přírůstkové a odvolatelné. Migrační soubory jsou umístěny v adresáři Rails aplikace `db/migrate`. Příklad vygenerování migračního souboru může vypadat například následovně:

```
ruby script/generate migration TabulkaAuta

exists  db/migrate
create  db/migrate/001_tabulka_auta.rb
```

Generátor `migration` vygeneruje soubor `001_tabulka_auta.rb`, který obsahuje metodu `up` pro provedení migrace a metodu `down` pro odvolání migrace. Číslo `001` na začátku souboru označuje verzi migrace. Soubor `001_tabulka_auta.rb` může obsahovat následující kód, který je částečně vygenerovány generátorem kódu a částečně ručně doplněný. Ručně doplněný kód je ten kód, který se nachází uvnitř metody `up` a uvnitř metody `down`:

```
class TablukaAuta < ActiveRecord::Migration
      def self.up
            create_table :auta do |polozka|
                  polozka.string barva
                  polozka.string znacka
            end
      end
      def self.down
            drop_table :auta
      end
end
```

Soubor obsahuje třídu `TabulkaAuta`, která je potomkem třídy `ActiveRecord::Migration`. Metoda `up` provede migraci, která vytvoří tabulku `auta` s položkami `barva` a `znacka` typu `string`. Metoda `down` vymaže tabulku `auta` z databáze.

Příkazem `rake db:migrate` se provedou všechny migrace, podle migračních souborů uložených v adresáři `db/migration`. Pokud není uvedeno, zda se má provést metoda `up` nebo `down`, implicitně se volá metoda `up`. V tomto případě se provede migrace, tedy vytvoří tabulka `auta`, podle jediného migračního souboru `001_tabulka_auta.rb`, který se nachází v adresáři `db/migration`. Vrátit se k určité verzi migrace, například k verzi 001 lze provedením příkazu: `rake db:migrate:down VERSION=001`.

### 3.4.2 Generování modelu a řadiče

Generovat lze také modely a řadiče [1]. Model se generuje příkazem:

```
ruby script/generate model MujModel
```

Generátor kódu vytvoří v adresáři `app/models` příslušný soubor `muj_model.rb`, podle názvu modelu, který byl zadán jako druhý parametr generátoru. Soubor obsahuje třídu `MujModel`, která je potomkem třídy `ActiveRecord::Base`.

Obdobně generátor umí vytvářet i řadiče. Navíc za název řadiče lze napsat názvy jeho metod. Uvedený příklad vygeneruje řadič `Main` s metodami `index`, `new` a `list`:

```
ruby script/generate Main index new list
```

Generátor vytvoří soubor řadiče `app/controllers/main_controller.rb`, který obsahuje třídu `MainController` zděděná ze třídy `ApplicationController` a tato třída obsahuje předpřipravené prázdné metody `index`, `new` a `list`. Dále vygeneruje prázdné pohledy pro tyto metody.

### 3.4.3 Generátor Scaffold

Pomocí nástroje `scaffold` (lešení) [1] lze vygenerovat migrace, model, řadič a pohledy najednou. Nástroj `scaffold` vytvoří pro aplikaci lešení neboli rámec, který je možno, ale není nutno vyplnit.

Následující příklad ukazuje možné vygenerování kódu pomocí nástroje `scaffold`:

```
ruby script/generate scaffold Car color:string
max_speed:integer description:text
```

Uvedený příklad generuje model `Car`, řadič `Cars` s určitými metodami, které jsou popsány níže, a jejich pohledy. Mimo to ještě migrační soubor, který obsahuje předpřipravený kód pro vytvoření tabulky `cars` s třemi položkami: `color` typu string, `max_speed` typu integer a `description` typu text. Jako první argument nástroje `scaffold` se uvádí název modelu, který musí být v jednotném čísle. Název řadiče je pak totožný s názvem modelu, ale v množném čísle. Dále jsou uvedeny položky tabulky, kde název položky a datový typ položky je oddělený dvojtečkou.

Jakmile aplikace obsahuje nějakou tabulku, je třeba ji naplnit daty z formuláře, mazat je, editovat nebo zobrazovat. Pro všechny tyto funkce nástroj `scaffold` vygeneruje metody v řadiči a automaticky vygeneruje v pohledech těchto metod formuláře pro plnění a editaci dat podle typu

sloupců v tabulce. Například pro sloupec `varchar` vytvoří input prvek typu text, pro `boolean` checkbox. Není třeba tedy všechno pracně psát, ale případně jen upravit již vygenerované.

## 3.5 Rozšíření

Pomocí externích knihoven lze webové aplikace vytvořené ve frameworku Ruby on Rails rozšiřovat a měnit. Jelikož existuje hodně vývojářů vytvářejících webové aplikace, existuje spousta rozšíření, které vytvořil již někdo jiný, který potřeboval obdobnou funkcionalitu. Rozšíření se instalují buď pomocí balíčkovacího nástroje pro jazyk Ruby RubyGems a nebo do samotné Rails aplikace do adresáře `vendor/plugins`.

### 3.5.1 Syntaktický analyzátor Hpricot

Hpricot [8] je velmi rychlý XML a HTML parser pro jazyk Ruby napsaný v jazyce C. Lze jej instalovat jako RubyGems balíček a to příkazem: `gem install hpricot`. Do vlastního kódu se pak tento nástroj přidá na začátek souboru zapsáním: `require 'hpricot'`. V Rails aplikaci jej lze takhle přidat buď do určitého modelu, nebo do celé aplikace zapsáním do konfiguračního souboru `config/environment.rb`.

Analyzovat jde řetězce uložené v proměnné, soubory uložené na disku a nebo webové stránky stažené z internetu. V případě analýzy souboru uloženého na disku nebo webových stránek stažených z internetu, je třeba nejprve tyto stránky otevřít pomocí metody `open`. Metoda `open` je obsažena ve standardní knihovně jazyka Ruby `open-uri`, kterou je třeba připojit k našemu kódu napsáním příkazu `require 'open-uri'` na začátek souboru s kódem nebo do konfiguračního souboru `config/environment.rb`. Následující příklad otevře webovou stránku na URL adrese http://www.seznam.cz a uloží ji do proměnné `doc` připravenou pro analýzu pomocí nástroje Hpricot:

```
doc = Hpricot(open("http://www.seznam.cz"))
```

Následně lze v tomto otevřeném dokumentu vyhledávat nebo měnit HTML elementy a jejich atributy. Vyhledávání jde provést dvěma způsoby. První způsob se zapisuje pomocí metody `search`, následují závorky, do kterých se zapisuje v uvozovkách cesta hledaného elementu stejně jako pomocí jazyka XPath. Zkrácený zápis pro hledání potomka do libovolné hloubky se zapíše pomocí dvou lomítek, v hranatých závorkách je uveden predikát uzlu. Znak `@` se používá pro zkrácený zápis testování atributu uzlu:

```
result = doc.search("//p[@class='main']")
```

Příklad hledá v otevřeném dokumentu `doc` všechny `p` HTML elementy s atributem `class` pojmenovaným `main`. Druhý způsob má úspornější zápis:

```
result = (doc/"p.main")
```

Lomítko nahrazuje metodu `search` a tečka nahrazuje hledání podle atributu `class`.

### 3.5.2 Stránkovací nástroj Mislav Will Paginate

Mislav Will Paginate [9] je rozšíření, které vývojáři pomáhá vyřešit otázku ohledně toho, jak jednoduše stránkovat záznamy z databáze do pohledů, jakmile je třeba vypsat velké množství záznamů. Instalaci rozšíření je možno provést více způsoby, například stáhnutím rozšíření a následným rozbalením archívu do adresáře `vendor/plugins` a nebo přidáním zdroje a instalací pomocí gem balíčku. Jelikož implicitní gem zdroj obsahuje v době psaní této práce ještě starou verzi tohoto nástroje, je třeba přidat zdroj ke stažení novější verze. Přidání zdroje a instalace nástroje je uvedeno v následujícím příkladu:

```
gem sources -a http://gems.github.com
gem install mislav-will_paginate
```

Programátor nesmí pak ještě zapomenout přidat kód `require "will_paginate"` na konec konfiguračního souboru `config/environment.rb`.

Použití je pak opravdu jednoduché. V daném řadiči určité metody, pomocí které se pak budou záznamy stránkovat, je třeba napsat například následující kód:

```
  @comps = Comp.paginate :page => params[:page], :order => 'name',
:per_page => 60
```

Metoda `paginate` stránkuje záznamy z modelu `Comp`, tedy tabulky `comps`. Do argumentu `:page` se přiřazuje číslo stránky, které se má zobrazit. Argument `:order` je nastaven, aby záznamy byly řazeny podle jména a argument `:per_page` znamená kolik záznamů se má zobrazit na jedné stránce. Do pohledu je třeba napsat kód, pomocí kterého se budou stránkovat vypsané záznamy:

```
<%= will_paginate @comps %>
```

# 4 Analýza požadavků

Jednou z fází životního cyklu vývoje software je analýza požadavků a specifikací uživatele tzn. zákazníka, pro kterého aplikaci nebo software vyvíjíme. I přesto, že požadavky na naši aplikaci do značné míry vycházejí ze zadání, musíme je určit a analyzovat blíže. Proto naše požadavky na aplikaci rozdělíme na funkční, tedy výpis služeb systému a nefunkční, které se týkají například bezpečnosti nebo vzhledu aplikace. V poslední podkapitole se seznámíme s grafem případu užití, z kterého budeme vycházet při tvorbě návrhu řešení.

## 4.1 Funkční požadavky

Automat bude v určitých časových intervalech stahovat inzeráty z inzertních serverů a ukládat je v databázi. Po novém naplnění budou staré inzeráty určitou dobu uchovány a po uplynutí této doby smazány. Automat bude muset v určitých intervalech aktualizovat statistiky nejžádanějších profesí a poměry počtu nabízených inzerátů ve všech krajích.

Správce této aplikace má možnost přidávat nové inzertní servery, mazat inzertní servery nebo jen deaktivovat inzertní server, například v případě poruchy zdroje. Taktéž může ručně spustit aktualizaci statistik a inzerátů nebo mazat inzeráty pomocí příkazové řádky.

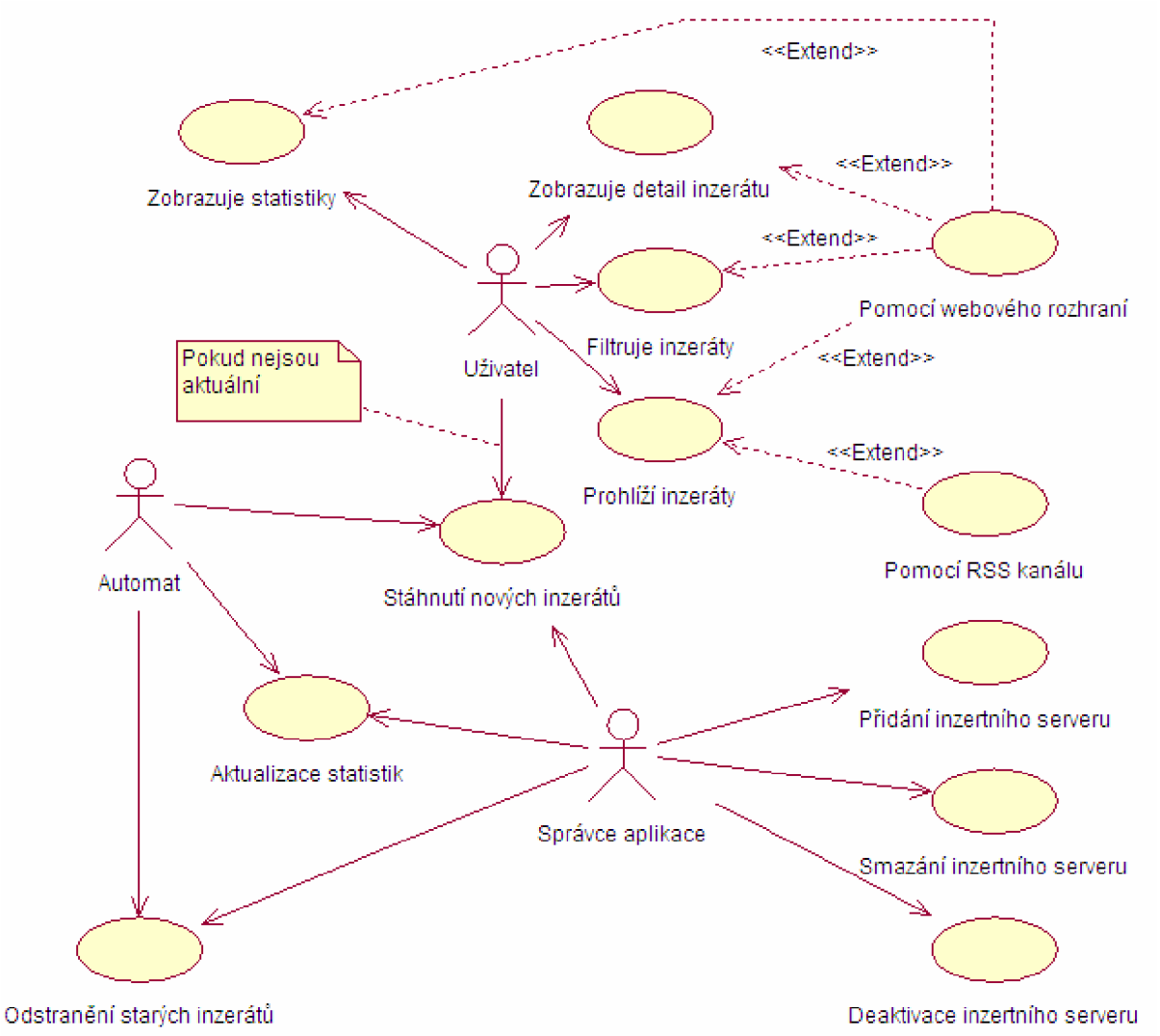
Uživatel může vyhledávat inzeráty podle kraje, oboru a pracovního poměru. Navíc tuto kombinaci lze filtrovat pomocí klíčového slova. Zobrazenými inzeráty může stránkovat a zobrazit detail inzerátu. Po zobrazení výsledku hledaných inzerátů, aplikace nabídne uživateli adresu RSS kanálu, pomocí kterého může odebírat inzeráty, které uživatel právě hledal podle určitého kraje, oboru a pracovního poměru. Nejsou-li zobrazené inzeráty aktuální, aplikace zobrazí odkaz, kterým lze zobrazené inzeráty aktualizovat. Uživatel může prohlížet statistiky nejhledanějších oborů, krajů a pracovních poměrů, a statistiky nejžádanějších profesí a poměry počtu nabízených inzerátů ve všech krajích.

## 4.2 Nefunkční požadavky

Aplikace bude multiplatformní, napsána v jazyce Ruby ve frameworku Ruby on Rails, který bude napojen na databázový systém MySQL a poběží na webovém serveru Mongrel. Zdrojem budou HTML stránky umístěny na inzertních webových serverech poskytující práci. Automat bude stahovat, ukládat, mazat inzeráty a aktualizovat statistiky v časových intervalech nastavených v Cron démonu. U každého inzerátu musí být uveden zdroj odkud byl inzerát stažen, aby nedocházelo k porušování autorských práv. Před nasazením do ostrého provozu, musí být aplikace řádně otestována. Musí být zajištěno jednoduché přidávání nových inzertních serverů pomocí modulů přes příkazový řádek.

## 4.3 Graf případů užití

Na základě funkčních a nefunkčních požadavků byl vytvořen graf případů užití, který je zobrazen na následujícím obrázku, viz. Obrázek 4.1.

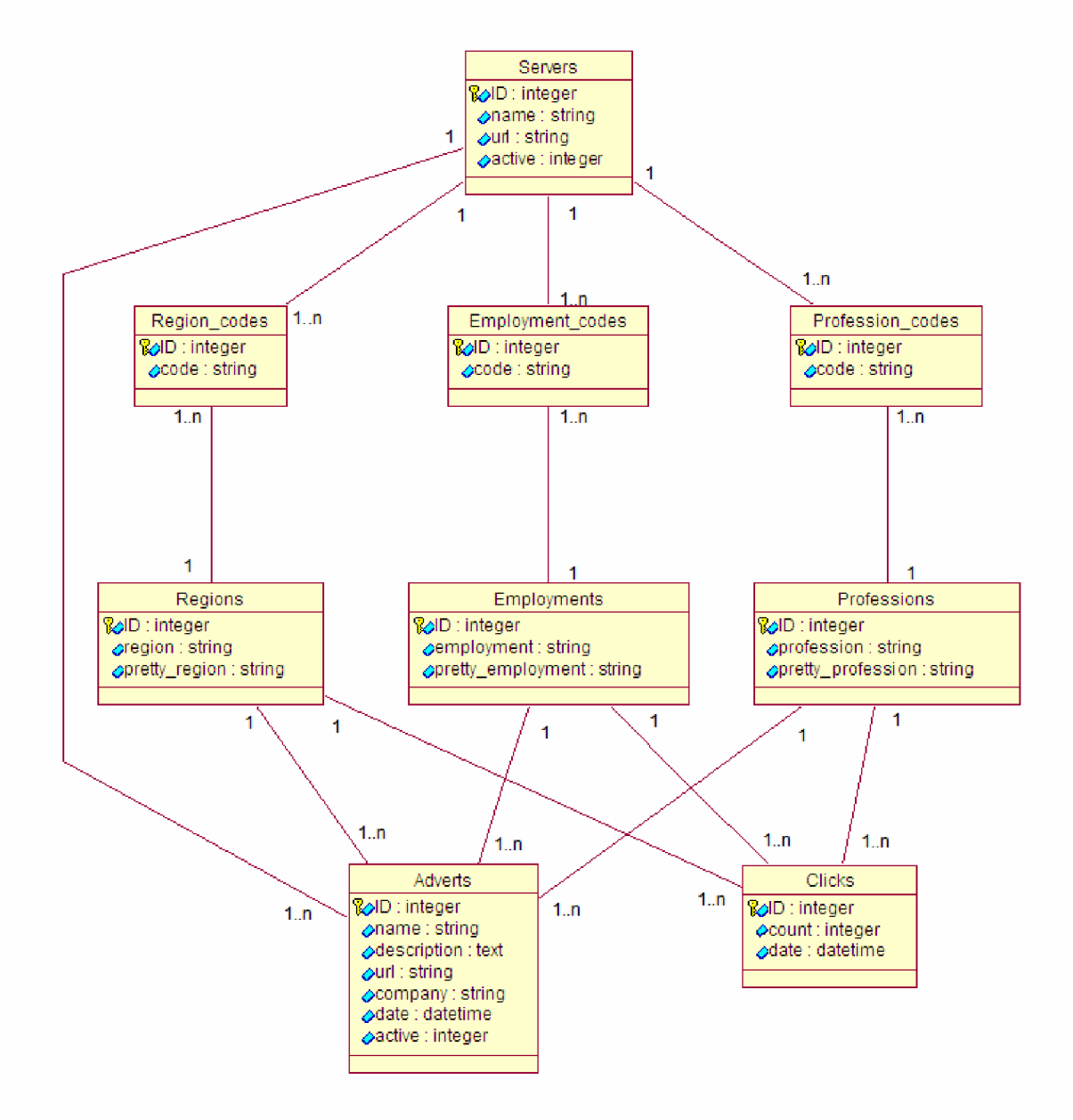



**Obrázek 4.1: Graf případů užití.**

Graf případů užití obsahuje tři účastníky a to uživatele aplikace, správce aplikace a automat. Automat má za úkol stahovat, mazat inzeráty a aktualizovat statistiky. Správce aplikace může provést totéž co automat, navíc má možnost přidat, smazat nebo deaktivovat inzertní server. Uživatel aplikace může pomocí webového rozhraní zobrazovat statistiky, hledat inzeráty, prohlížet inzeráty a zobrazovat detail inzerátu. Navíc má oprávnění aktualizovat statistiky, pokud nebudou aktuální a odebírat inzeráty pomocí RSS kanálu.

# 5 Návrh databáze

Před vytvořením návrhu datového modelu, bylo třeba analyzovat několik webových serverů nabízejících inzeráty práce. Výsledkem analýzy je Entitně-relační diagram zobrazený na následujícím obrázku, viz. Obrázek 5.1.




**Obrázek 5.1: Entitě-relační diagram aplikace.**

Po důkladné analýze bylo zjištěno, že se musí navrhnout tři entity, které budou obsahovat vyhledávací „kódy“, podle kterých bude aplikace komunikovat s jednotlivými webovými servery nabízejících inzeráty práce. Jedná se o entity `region_codes`, `employment_codes` a `profession_codes`. Všechny tyto entity jsou v relaci s entitou `servers`, aby aplikace mohla

odpovídající vyhledávací kód přiřadit příslušnému serveru a také je každá tato entita spojena se svou „bratrskou“ entitou, tzn. `region_codes` s `regions`, `employment_codes` s `employments` a `profession_codes` s `professions`. Tímto se zajistí, že aplikace bude vědět, jaký kraj, profesi a pracovní úvazek určitého serveru má přiřadit k příslušnému vyhledávacímu kódu.

Entita `servers` obsahuje atributy jméno (`name`) a URL adresu (`url`) webových serverů nabízejících inzeráty práce. Navíc obsahuje ještě atribut `active`, která bude sloužit ke globální aktivaci nebo deaktivaci aktualizace inzerátů práce určitého webového serveru.

Entity `regions`, `employments` a `professions` slouží k nadefinování existujících krajů, pracovních úvazků a profesí. Každá z nich obsahuje ještě navíc svůj atribut `pretty_region`, `pretty_employment` a `pretty_profession` označující kraj, pracovní úvazek a profesi v předpřipravené podobě pro použití ohledně přátelských URL (pretty URL).

Uložené inzeráty se nacházejí v entitě `adverts`, která je v relaci s entitami `servers`, `regions`, `employments` a `professions`, aby aplikace věděla jaké inzeráty přiřadit k určitému kraji, pracovnímu úvazku a profesi a věděla, odkud byl inzerát stažen. Dále obsahuje atribut pro název pracovního inzerátu (`name`), krátký popis pracovního inzerátu (`description`), URL pro zobrazení detailu inzerátu (`url`), název firmy zadávající inzerát (`company`), datum stažení inzerátu (`date`) a položku `active` pro aktivaci nebo deaktivaci inzerátu.

Statistiky vyhledávaní pracovních inzerátů pomocí uživatelů aplikace se ukládají do entity `clicks`. Obsahuje atribut `date`, v které je uložen čas, kdy uživatel vyhledával nějaké inzeráty a atribut `count` obsahující počet hledání za poslední hodinu, kdy uživatel vyhledával inzeráty nacházející se v určitém kraji, profesi a pracovním úvazku. Pro rozeznání v jakém kraji, profesi a pracovním úvazku je entita `clicks` ještě v relaci s entitami `regions`, `professions` a `employments`.

# 6 Implementace

Popis implementace aplikace jsem rozdělil do podkapitol podle model-pohled-řadič návrhového vzoru. Tímto je vidět, že tyto tři části návrhového vzoru jsou opravdu odděleně. V dalších podkapitolách popisuji, co všechno třeba nastavit a nainstalovat, aby tato aplikace byla provozuschopná na serveru, a jak přidat do této aplikace další servery nabízející inzeráty práce. V poslední podkapitole jsou ukázky z aplikace.

## 6.1 Vytvoření řadiče aplikace a jeho metod

Řadič aplikace `Main`, který je jediným řadičem v této aplikaci, jsem vygeneroval pomocí generátoru kódu bez jakékoliv jeho metody. Metody `index`, `list`, `rss`, `redirect_to_list`, `update`, `detail`, `contact`, `stats`, `error` a `noadvert`, které obsahuje, jsem vytvořil ručně.

Metoda `index` slouží k zobrazení úvodní domovské stránky aplikace. Jeho pohled je uložen v adresáři `app/views/main/index.html.erb`.

Po vyplnění a odeslání formuláře pro vyhledání pracovních inzerátů, aplikace volá metodu `redirect_to_list`, která přebírá parametry z pohledu nebo ze směrovací tabulky, pomocí nichž vyhledá nejprve odpovídající `id` kraje, pracovního úvazku a profese z tabulek `regions`, `employments` a `professions`. Pokud by v této části nastal případ, že by metoda některé z `id` nenašla, například uživatel zadá neodpovídající URL, aplikace se přesměruje do metody `error` a tímto zobrazí chybové hlášení. V případě naleznutí odpovídajících údajů, metoda uloží do tabulky `clicks` záznam s aktuálním časem o vyhledávaní inzerátů, pro zobrazení ve statistikách a dále přesměruje program do metody `list`.

Metoda `list` má za úkol zobrazit uživatelem vyhledané inzeráty. Metoda přebírá parametry z metody `redirect_to_list` a vyhledá v tabulce `adverts` všechny inzeráty podle zadaných parametrů. Do proměnné instancí `@count` uloží počet nalezených inzerátů a do proměnné instancí `@companies` uloží pomocí metody `paginate` odpovídající počet inzerátů, které se mají zobrazit na jedné stránce. Počet zobrazených inzerátů na jedné stránce jsem nastavil na 50 pracovních inzerátů. Nakonec vyhledá, kdy byly inzeráty naposled aktualizovány, a podle toho pak zobrazí v pohledu možnost aktualizace vyhledaných pracovních inzerátů. Nastavil jsem, že se tato možnost zobrazí pro inzeráty starší 12 hodin. Hodnoty počtu inzerátů na stránku a stáří inzerátů pro zobrazení možnosti aktualizace inzerátů uživatelem, lze změnit v konfiguračním souboru `config/environment.rb`.

Jakmile jsou inzeráty práce zobrazeny a uživatel klikne na nadpis jednoho z inzerátů, pro zobrazení detailu inzerátu, aplikace volá metodu `detail`, která přebírá z pohledu jediný parametr, a to `id` inzerátu. Metoda `detail` vyhledá podle tohoto `id` v tabulce `adverts` odpovídající inzerát a předá pohledu metody `detail` URL pro zobrazení detailu inzerátu. Pokud by požadované `id` inzerátu neexistovalo, program se přesměruje do metody `noadvert` a zobrazí chybové hlášení. Pro zobrazení detailu inzerátu metoda `detail` používá jiné rozvržení stránky než ostatní metody řadiče `Main`, a to proto, že při zobrazení detailu inzerátu nebude zobrazeno menu aplikace, ale pouze hlavička stránky a detail inzerátu umístěn v HTML prvku `iframe` přes celou webovou stránku v originální podobě.

Po vyhledání inzerátů, aplikace taktéž nabídne uživateli odebírání vyhledávaných inzerátů pomocí RSS kanálu. Odkaz pro odebírání je umístěn na stránce nad nalezenými inzeráty. Kliknutím na odkaz RSS kanálu řadič volá metodu `rss`. Metoda `rss`, tak jak metoda `redirect_to_list` vyhledává nejprve odpovídající `id` kraje, pracovního úvazku a profese z tabulek `regions`, `employments` a `professions`. V případě neuspěšeného vyhledání `id`, aplikace se přesměruje do metody `error` a zobrazí chybové hlášení. V případě bezchybného průběhu, metoda vyhledá z tabulky `adverts` pracovní inzeráty odpovídající nastaveným parametrům vyhledávaní a pokud prohlížeč internetových stránek podporuje RSS zdroje, zobrazí je. Klíčové slovo v tomto případě nehraje žádnou roli. Metoda nepoužívá žádné rozvržení stránky, tedy je generována příkazem `render :layout => false`. Aplikace taktéž zobrazuje ikonku v adresovém řádku internetového prohlížeče stránek, označující možnost odebírání nových inzerátů pomocí RSS kanálu.

Pokud uživatel klikne na možnost aktualizace právě vyhledaných inzerátů, řadič aplikace volá metodu `update`, která tak jak metoda `redirect_to_list` a `rss` nejprve vyhledá odpovídající `id` kraje, pracovního úvazku a profese. V případě nenalezení nějakého z `id`, metoda se přesměruje do metody `error`, v opačném případě metoda volá metodu `get_advs` třídy `Advert`, které předává `id` kraje, pracovního úvazku a profese, pro aktualizaci pracovních inzerátů podle zadaných parametrů. Třída `Advert`, která je modelem pro tabulku `adverts` bude popsána níže. Metoda `get_advs` třídy `Advert` slouží k stahování nebo aktualizaci pracovních inzerátů. Po aktualizaci inzerátů se obnoví stránka se zobrazenými inzeráty práce.

Metoda `contact` a `stats` slouží k zobrazení kontaktu a statistik. Pohled metody `stats` zobrazuje koláčové grafy nejžádanějších profesí a poměr nabízených inzerátů ve všech krajích pomocí obrázků, které jsou generovány po celkové aktualizaci inzerátů práce a to kvůli tomu, že získání požadovaných údajů z tabulky `adverts` pro vygenerování těchto grafů trvá moc dlouhou dobu na to, aby se tyto grafy generovaly při každém zobrazení statistik uživatelem. Generování grafů obstarává metoda `regenerate_graphs` třídy `Advert`, která bude popsána v kapitole 6.2.

Směrování pro tyto akce jsem nastavil v konfiguračním souboru `config/routes.rb`.

## 6.2 Vytvoření modelů aplikace

Aplikace obsahuje v adresáři `app/models` modely `region.rb`, `employment.rb`, `profession.rb`, `region_code.rb`, `employment_code.rb`, `profession_code.rb` a `server.rb`, které slouží pro práci s databázovými tabulkami `regions`, `employments`, `prefessions`, `region_codes`, `employment_codes`, `profession_codes` a `servers`. Obsahují pouze kód pro nastavení závislosti k udržení referenční integrity databáze, to znamená nastavení cizích klíčů pro tyto tabulky.

V modelu `advert.rb` se nacházejí metody pro práci s pracovními inzeráty. Metoda `get_advs`, která vyžaduje 4 parametry, slouží k stahování pracovních inzerátů. Potřebné parametry je třeba zadávat v tomto pořadí: `id` serveru z tabulky `servers`, `id` kraje z tabulky `regions`, `id` pracovního úvazku z tabulky `employments` a poslední parametr `id` profese z tabulky `professions`. Podle zadaných `id` serveru, kraje, pracovního úvazku a profese metoda stáhne, zpracuje a uloží pracovní inzeráty do tabulky `adverts`. Metoda `get_advs` umí také stáhnout všechny možné inzeráty najednou ze všech serverů, krajů, pracovních úvazků nebo profesí zadáním do parametru místo odpovídajícího `id` hodnotu 0. Tabulka `adverts` obsahuje položku `active`, která má následující význam. Nastavená hodnota 0 znamená, že inzerát byl nově stažen, ale ještě nebyl aktivován pro zobrazení uživatelům. Inzeráty nastavené na hodnotu 1 jsou zobrazovány

uživatelům a hodnota 2 slouží k označení inzerátu jako už neplatného, nachystaného k smazání. Metoda nastavuje na hodnotu 0 právě stahující inzeráty a jakmile dokončí stahování pracovních inzerátů z určitého kraje, pracovního úvazku, profese a serveru inzeráty nastavené na hodnotu 1 změní na hodnotu 2 a nově stáhnuté inzeráty, které mají hodnotu 0 nastaví na hodnotu 1. Pro toto „přepnutí" inzerátů metoda `get_advs` používá pomocnou metodu `switch_items`. Předím, než metoda `get_advs` začne stahovat nové inzeráty, zavolá ještě pomocnou metodu `delete_temp_items`, která zkontroluje jestli náhodou už neexistují nějaké inzeráty nastavené na hodnotu 0 stáhnuté z konkrétního serveru, kraje, pracovního úvazku a profese, například pokud v minulém stahování došlo k nějaké chybě a tyto dočasné inzeráty tam zůstaly.

Zajištění jednoduchého přidávání dalších serverů nabízejících práci jsem vyřešil tak, že každý pracovní server má vytvořenou vlastní třídu ve vlastních souborech a aplikace ví o jaké soubory se jedná, podle uložených záznamů v tabulce `servers`. V tuto chvíli jsou přidány do aplikace 4 servery. Jedná se o soubory `jobs.rb` pro server nabízející práci na http://www.jobs.cz, `sprace.rb` pro server http://www.sprace.cz, `nabidkyprace.rb` pro server http://www.nabidky-práce.cz a `dobraprace.rb` pro server http://www.dobraprace.cz uložené v adresáři `app/models`. Dále v tabulce `servers` je pod položkou `name` uložen název konkrétního serveru, tedy i název použité třídy pro daný server, který je totožný z názvem třídy ve výše vyjmenovaných souborech. Metoda `get_advs` pak vezme z tabulky `servers` název třídy pro konkrétní server, vytvoří objekt pro stahování z konkrétního serveru a zavolá metodu `parse_server` třídy vytvořeného objektu pro stáhnutí inzerátu. Metoda `parse_server` analyzuje konkrétní webový server nabízející inzeráty práce pomocí nástroje Hpricot a vrátí uložené inzeráty v poli. V metodě `get_advs` se uloží inzeráty do tabulky `adverts`. Návod jak přidat další servery bude popsán v kapitole 6.5.

Třída `Advert` obsahuje také metodu pro obnovení grafů statistik, kterou jsem pojmenoval `regenerate_graphs`. Tato metoda volá postupně pomocné metody `stat_offer_profession` a `stat_offer_region`, které vracejí pole profesí a krajů s celkovými počty záznamů z tabulky `adverts` pro určité profese a kraje. Metoda `regenerate_graphs` pak pomocí rozšíření `Gruff` vygeneruje dva koláčové grafy a uloží je do oddělených souborů do adresáře `public/images`. Tyto grafy se pak zobrazují v pohledu metody `stats` řadiče `Main`.

Třída `Click` obsahuje metody pro práci s tabulkou `clicks`. Tabulka `clicks` slouží k ukládání v jaký čas uživatelé hledali jaký kraj, jaký pracovní úvazek a jakou profesi. Pomocí metody `save_click` aplikace uloží záznam do tabulky `clicks`, a to tak, že metoda nejprve vyhledá v tabulce `clicks` záznam s hledaným krajem, pracovním úvazkem a profesí uložený za poslední hodinu. Pokud takový záznam existuje navýší počítadlo tohoto záznamu (položka `count`) a uloží do tabulky. Pokud takový záznam za poslední hodinu ještě není, vytvoří nový záznam s položkou `count` nastavenou na počáteční hodnotu 1. Metody `stat_search_region`, `stat_search_employment` a `stat_search_profession` vracejí uspořádaný seznam hledaných krajů, pracovních úvazků a profesí za posledních tolik hodin, kolik je zadáno v parametru těchto metod.

## 6.3 Vytvoření prezentační vrstvy aplikace

Prezentační vrstva aplikace se skládá z rozvržení stránky (layout) a pohledů metod řadiče `Main`. Pro metodu `detail` řadiče `Main` je rozvržení v souboru `detail.html.erb` a uloženo v adresáři

app/views/layouts, kde se mají ukládat rozvržení. Pro ostatní metody řadiče Main, mimo metody rss, která nepoužívá žádné rozvržení, aplikace používá rozvržení stránky uložené v souboru main.html.erb.

Hlavní rozvržení stránky main.html.erb obsahuje po levé straně menu, kde si uživatel může zobrazit kontakty na správce stránek nebo statistiky nejžádanějších profesí a počty inzerátů v krajích v procentech. Pod tímto menu je formulář pro hledání inzerátů práce, který obsahuje tři HTML prvky select pro výběr kraje, pracovního úvazku a profese a jeden HTML prvek input pro možnost toto hledání filtrovat ještě pomocí klíčového slova. Statistiky, které zobrazují nejhledanější kraje, nejhledanější pracovní úvazky a profese za posledních 24 hodin a za posledních 7 dní jsou zobrazeny pod formulářem hledání. Tyto statistiky pohled bere pomocí metod třídy Click z tabulky clicks.

Obrázky použité pro vzhled aplikace jsou uloženy v adresáři public/images a CSS soubor default.css v adresáři public/stylesheets.

## 6.4 Zprovoznění aplikace

Aplikace potřebuje pro svůj provoz následující rozšíření: rubygems, hpricot, open-uri, will_paginate, gruff a localization_simplified. Tyto rozšíření jsou zapsány na pozici nejníže v konfiguračním souboru config/environment.rb.

Dále je třeba nastavit propojení s databází, tzn. jméno a heslo uživatele a název databáze, které se nastavuje v konfiguračním souboru config/database.yml. Aplikace pro svůj provoz vyžaduje MySQL databázový systém verze 5. V adresáři db/migrate je uložen migrační soubor 001_initial.rb, který obsahuje vše potřebné pro vytvoření tabulek aplikace. Adresář db/migrate/data obsahuje soubory, kterými je třeba pro provoz aplikace naplnit tabulky regions, employments, professions, region_codes, employment_codes, profession_codes a servers. O tabulky regions, employments a professions se není třeba starat, ty budou naplněny zároveň v rámci migrace. Pokud je konfigurační soubor databáze database.yml již nastaven, příkazem rake db:create:all se vytvoří databáze a příkazem rake db:migrate se vytvoří databázové schéma a zároveň automaticky naplní tabulky regions, employments a professions požadujícími záznamy. Data pro naplnění zbylých čtyř zmíněných tabulek jsou uložena v souboru servers_and_codes.sql.

Pro naplnění těchto tabulek se použije příkaz v příkazovém řádku mysql -u user < db/migrate/data/servers_and_codes.sql, s tím, že uživatel databáze je nastaven na jméno user a databáze této aplikace má název webadvert. Následně příkazový řádek zažádá o napsání hesla pro tohoto uživatele.

Automat, který má na starost spouštět skripty v nastavených časových intervalech používá skript aplikace v adresáři scripts, který má název runner. Je tedy třeba nastavit v konfiguračním souboru Cron démonu spuštění automatického obnovení všech inzerátů práce ze všech serverů, krajů, pracovních úvazků a profesí pomocí metody get_advs třídy Advert každy den ve tři ráno pomocí příkazu: ruby webadvert/script/runner "Adverts.get_advs(0,0,0,0)". Předpokládá se, že aplikace je uložena v adresáři webadvert domovského adresáře uživatele.

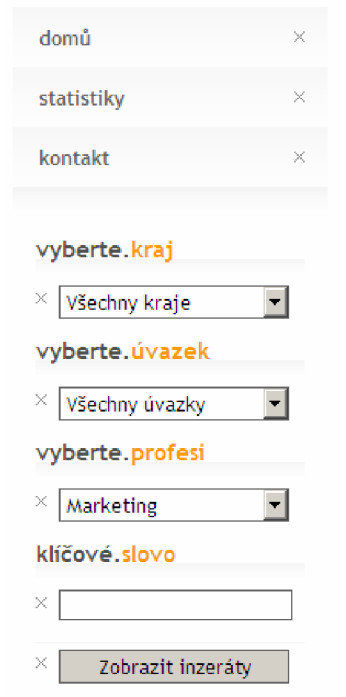
Mongrel server se spustí příkazem mongrel_rails start -p 3000, kde parametr -p znamená na jakém portu se má aplikace spustit. V případě problémů se spuštěním nebo instalací Mongrel serveru lze použít a spustit WEBRick server, který je součástí každé Rails aplikace a to příkazem: ruby script/server.

## 6.5 Přidání dalších serverů do aplikace

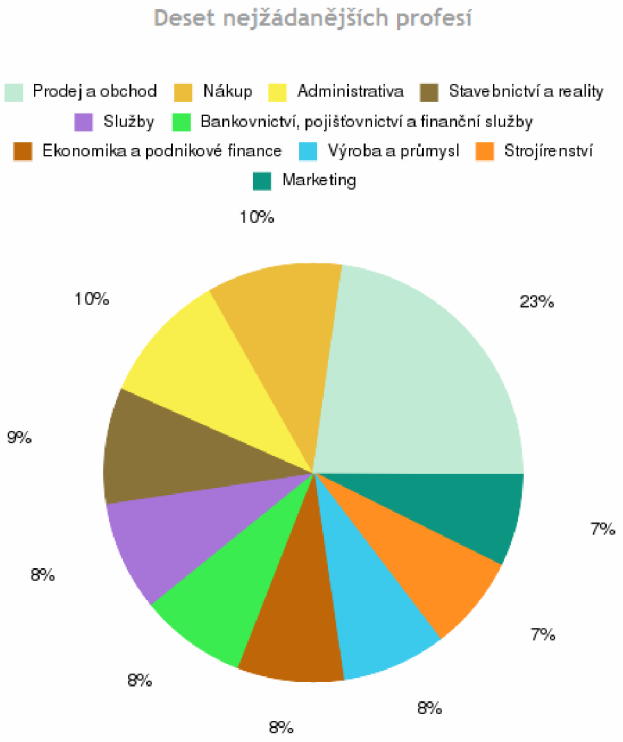
Do aplikace lze přidávat další servery nabízející inzeráty práce a to jednoduše přidáním pouze jednoho souboru do adresáře `app/models` s vytvořenou třídou, v které bude naprogramováno stahování inzerátů z tohoto požadovaného serveru, obdobně jak například pro server `http://www.jobs.cz` je vytvořena třída v souboru `jobs.rb` uloženého v adresáři `app/models`. Dále je třeba do databáze nahrát „komunikační kódy" pro požadovaný nový server pomocí již zmíněného příkazu v předchozí podkapitole `mysql -u user webadvert < db/migrate/data/new_server.sql` a pak už je možno přidaný server používat a stahovat z něj pracovní inzeráty.

## 6.6 Ukázky z aplikace

V této kapitole je zobrazeno několik ukázek ze samotné aplikace. Na obrázku, viz. Obrázek 6.1 je zobrazeno jednoduché menu aplikace, pomocí kterého se lze vrátit na úvodní stránku aplikace, zobrazit statistiky nejžádanějších profesí a poměr počtu inzerátů v jednotlivých krajích a zobrazit kontakt. Pod tímto menu se nachází formulář pro vyhledávaní inzerátů. Inzeráty lze vyhledat vybráním hledaného kraje, pracovního úvazku a požadované profese. Navíc vyhledané inzeráty lze filtrovat pomocí klíčového slova. Na následujícím obrázku je zobrazen koláčový graf deseti nejžádanějších profesí, viz. Obrázek 6.2.




**Obrázek 6.1: Menu aplikace.**




**Obrázek 6.2: Koláčový graf deseti nejžádanějších profesí.**

Na posledním obrázku je zobrazen výsledek hledání, viz. Obrázek 6.3. Aplikace zobrazí počet nalezených inzerátů a informaci před jakou dobou byly inzeráty naposled aktualizovány. Pod

těmito údaji je pak zobrazen odkaz pro odebíraní inzerátů pomocí RSS kanálu. Následně je zobrazena tabulka s nalezenými inzeráty.


**Obrázek 6.3: Ukázka vyhledaných inzerátů.**

# 7 Závěr

Cílem této bakalářské práce bylo vytvořit aplikaci, která agreguje inzeráty z inzertních serverů nabízejících práci do jednoho zdroje. V první části této práce se čtenář seznámil s použitými technologiemi pro provoz této aplikace, druhá část byla věnována samotnému frameworku Ruby on Rails, ve kterém je tato aplikace implementována. Další kapitoly byly věnovány již konkrétnímu popisu vývoje této aplikace.

Aplikace byla vyvíjena v operačním systému Gentoo, který je postaven na bázi Unix, ale jelikož framework Ruby on Rails je multiplatformní bude provozuschopný i na jiných operačních systémech. Implementací této aplikace jsem si prohloubil znalosti ve skriptovacím jazyce Ruby a s prací ve frameworku Ruby on Rails, seznámil se s technologií RSS, jazykem XPath, a zopakoval si znalosti z databází a modelování systémů. Aplikace agreguje inzeráty ze čtyř serverů nabízejících inzeráty práce a to: http://jobs.cz, http://sprace.cz, http://dobraprace.cz a http://nabidky-prace.cz. Servery jsem vybíral podle počtu nabízených inzerátů a podle rozdílnosti nabízených inzerátů, aby se do jisté míry zamezilo agregování serverů nabízející totožné inzeráty práce. Jelikož servery http://dobraprace.cz a http://nabidky-prace.cz nemají rozdělené inzeráty podle pracovního úvazku, jsou v této aplikaci umístěny pouze pod úvazkem „Všechny úvazky“.

Tato aplikace je umístěna na serveru poskytnutým firmou Ataxo Czech s.r.o. na adrese `http://webadverts.rstage.ataxo.cz/`, ke které se přistupuje pod jménem `webadverts` a heslem `CertovaSkala`. Je naplněna reálnými daty ze čtyř výše uvedených serverů nabízejících práci. Naplnění aplikace inzeráty nebo obnovení všech inzerátů nabízejících práci, tedy stáhnutí inzerátů a uložení do databáze trvalo 9 hodin. Obnovování všech inzerátů bylo plánováno na interval jednou denně, ale jelikož stahování trvá moc dlouhou dobu, bude vhodnější inzeráty stahovat obden nebo v jiný časový interval, aby nedocházelo k většímu zatěžování těchto serverů nabízejících práci. Protože aplikace obsahuje statistiku nejhledanějších krajů, profesí a pracovních úvazků, lze podle této statistiky více žádané inzeráty aktualizovat častěji a méně žádané inzeráty aktualizovat ve větších časových intervalech. Bohužel až při testování na serveru http://webadverts.rstage.ataxo.cz/ jsem zjistil, že jakmile uživatel aktualizuje inzeráty v určitém kraji, požadovaného pracovního úvazku a profese, a inzerátů je hodně (víc jak tisíc inzerátů), celý tento proces pak trvá moc dlouhou dobu a server zahlásí chybové hlášení. Stránku lze pak obnovit, poté co server dokončí požadavek pro stáhnutí inzerátů. Aplikaci jsem původně testoval na svém počítači a na jiném serveru, na kterým to fungovalo bez problémů, i když prováděný požadavek trval delší dobu. Webové rozhraní aplikace jsem testoval pod operačním systémem MS Windows XP v prohlížečích Internet Explorer verze 8 a Opera verze 9.64, a pod operačním systémem Gentoo v internetovém prohlížeči Mozzila verze 3.0.10.

Dle mého názoru je aplikace užitečná a plánuji ji dále rozšířit o další servery nabízející inzeráty práce. Detaily inzerátů zobrazuji v původní podobě, protože po zobrazení detailu inzerátů na některých serverech nabízející práci, má hodně inzerátů rozdílné rozvržení stránky. Jako další rozšíření aplikace, by mohly být metody pro různé skupiny rozvržení detailu inzerátů, které místo zobrazení detailu inzerátu v HTML prvku `iframe`, inzerát analyzuje a zobrazí ve stylu samotné aplikace. Uživatel by také mohl přidávat vlastní inzeráty. Muselo by se vytvořit uživatelské rozhraní, pomocí kterého by se uživatel mohl registrovat, přihlásit do systému a přidávat vlastní inzeráty. A rozhraní pro administrátora, který by mohl přidávat, blokovat a mazat uživatele, schvalovat inzeráty, mazat inzeráty nebo aktualizovat inzeráty.

# Literatura


[1] Holzner, S.: *Začínáme programovat v Ruby on Rails*, Computer Press 2007, ISBN: 978-80-251-1630-2

[2] Thomas A. Powell: *Web design: kompletní průvodce*, Computer Press 2004, ISBN: 80-7226-949-6

[3] Arlow J., Neustadt I.: *UML a unifikovaný proces vývoje a aplikací*, Computer Press 2003, ISBN: 80-7226-947-X

[4] *Wikipedie, otevřená encyklopedie* [online], [cit. 2009-04-29], Dostupné na URL: <http://cs.wikipedia.org>

[5] Šturala A.: *WPF - Data Binding* [online] 2007, [cit. 2009-04-24], Dostupné na URL: <http://www.netstudent.cz/ >

[6] Minařík K.: *Karmi is on Rails* [online] 2007-2008, [cit. 2009-04-16], Dostupné na URL: <http://blog.karmi.cz/>

[7] *Root.cz* [online], [cit. 2009-04-16], Dostupné na URL: <http://www.root.cz/>

[8] *Hpricot* [online], [cit. 2009-04-23], Dostupné na URL: <http://wiki.github.com/why/hpricot>

[9] *Mislav Will Paginate* [online], [cit. 2009-04-24], Dostupné na URL: <http://wiki.github.com/mislav/will_paginate>

[10] *Jak na démona Cron* [online], [cit. 2009-04-26], Dostupné na URL: <http://interval.cz/clanky/jak-na-demona-cron/>

[11] *XML pro každého* [online], [cit. 2009-05-17], Dostupné na URL: < http://www.kosek.cz/xml/>

Seznam příloh

# Příloha 1.

Návod k instalaci aplikace:

1. Zkopírovat z přiloženého CD adresář webadverts na disk počítače. Adresář webadverts obsahuje celou aplikaci.
2. Nastavit v konfiguračním souboru config/database.yml jméno a heslo uživatele, a název databáze.
3. Nainstalovat: jazyk Ruby, ruby-18gems, iconv, mongrel.
4. Pomocí příkazu „gem install“ je třeba nainstalovat balíčky: rails2, activerecord, rake, hpricot, open-uri, rubygems a gruff. Ostatní rozšíření není třeba instalovat, jsou obsaženy v samotné aplikaci v adresáři vendor/plugins.
6. Provést příkazy rake db:create a rake db:migrate
7. Nahrát do databáze soubor db/migrate/data/servers_and_codes.sql
8. Spustit server mongrel_rails start -p 3000
9. Příkazem: ruby script/runner „Advert.get_advs(0,0,0,0)“ se začne plnit databáze inzeráty. (Pozor: celá databáze se může plnit až 10 hodin!) Parametry metody get_advs jsou popsány v dokumentaci v kapitole 6.2.

Aplikace vyžaduje:

- MySQL server verze 5
- Ruby on Rails verze 2.2.2
- Nejnovější gem balíčky