Žilinská Univerzita v Žiline
Fakulta riadenia a informatiky

Bakalárska práca

Študijný odbor: **Informatika**

Adam Gavlák

Webová aplikácia pre podporu tvorby katedrových rozvrhov

Vedúci: **Mgr. Michal Kaukič, PhD.**

Reg.č. 23/2016 Máj 2017

Abstrakt

Gavlák Adam: *Webová aplikácia pre podporu tvorby katedrových rozvrhov* [Bakalárska práca]
Žilinská Univerzita v Žiline, Fakulta riadenia a informatiky, Katedra matematických metód a operačnej analýzy.
Vedúci: Mgr. Michal Kaukič, PhD.
FRI ŽU v Žiline, 2017 — ? s.

Obsahom práce je analýza, návrh a implementácia webovej aplikácie pre podporu tvorby katedrových rozvrhov s využitím moderných webových technológii, ktorá umožní urýchlenie a automatizáciu niektorých interných procesov Katedry matematických metód.

Abstract

GAVLÁK ADAM: *Web application for support of development of departmental schedules* [Bachelor thesis]
University of Žilina, Faculty of Management Science and Informatics, Department of mathematical methods and operational analysis.
Tutor: Mgr. Michal Kaukič, PhD.
FRI ŽU in Žilina, 2017 — ? p.

The content of bachelor thesis describes analysis, design and implementation of web application for support of development of departmental schedules using modern web technologies, which will aid some of the internal processes of Department of Mathematical Methods and Operations Research.

Prehlásenie

Prehlasujem, že som túto prácu napísal samostatne a že som uviedol všetky použité pramene a literatúru, z ktorých som čerpal.

V Žiline, dňa 21.4.2017 Adam Gavlák

Obsah

Úvod

Na počiatku bol Internet používaný hlavne pre akademické účely, ale za roky prevádzky sa rozvinul do dnešnej gigantickej podoby, keď už každý z nás môže mať prístup k Internetu aj vo vrecku, pričom sa stále rozrastá každým dňom.

Podobnou rýchlosťou sa vyvíjajú aj nástroje na vývoj webových aplikácii a v posledných rokoch sa začínajú deliť na oddelené disciplíny. Jednou z mojich úloh bolo porovnať a zvoliť si technológiu, na ktorej postavím internú webovú aplikáciu pre Katedru matematických metód a operačnej analýzy.

Keďže celý vývoj musím zvládnuť sám rozhodol zvoliť z obrovského množstva dnes dostupných technológii jazyk Ruby a použiť framework Ruby on Rails, pretože už s ním mám skúsenosti, poskytne mi všetky potrebné nástroje na splnenie daných cieľov a verím že mi uľahčí efektívny vývoj webovej aplikácie, kde sa môžem lepšie sústrediť na spracovanie požiadaviek koncových používateľov.

V tejto práci analyzujem vytvorenie kompletnej webovej aplikácie pre podporu tvorby katedrových rozvrhov, od analýzy a návrhu až po samotné nasadenie aplikácie na produkčný server. Konečným cieľom tejto práce je webová aplikácia, ktorá dokáže vypočítať úväzky pre jednotlivých vyučujúcich katedry, generovať prehľady do PDF súborov a následne ich aj odoslať danému vyučujúcemu cez e-mail.

Kapitola 1

Analýza a prehľad technológii

1.1 Požiadavky aplikácie

Na základe zadania a nasledovnými konzultáciami s koncovými používateľmi aplikácie bol vytvorený nasledujúci prehľad požiadaviek aplikácie:

- Vytvánie, úprava a zmazanie nasledujúcich dát:
  - Predmety katedry
  - Študijné skupiny
  - Vyučujúci
- Vzťahy medzi jednotlivými údajmi
- Prehľady predmetov, študijných skupín a vyučujúcich
- Prepočet úväzkov vyučujúcich podľa stanovených pravidiel
- Generovanie prehľadových dokumentov pre vyučujúcich
- Odosielanie emailov vyučujúcim s vygenerovanými prehľadmi

1.2 Back-end technológie

PHP

Python

Ruby

1.3 Databázové riešenia

MySQL

PostgreSQL

SQLite

Kapitola 2

Návrh aplikácie

2.1 Prípady použitia

2.2 Ruby

2.2.1 RubyGems

2.3 Framework – Ruby on Rails

2.3.1 Model, view a controller

2.3.2 ActiveRecord

2.3.3 Migrácie

2.4 Databáza a entitno-relačný diagram

Kapitola 3

Implementácia

3.1 Systém pre správu verzií – Git

3.2 Vývoj s pomocou príkazového riadku

3.3 Routing

3.4 Models

3.5 Controllers

3.6 Views

3.6.1 HTML a ERB

3.6.2 CSS

3.6.3 JavaScript

3.7 Spracovanie úloh na pozadí

3.8 Generovanie PDF prehľadov

3.9 Odosielanie e-mailov

3.12 Problémy pri implementácii a ich riešenie

3.12.1 N+1 queries

Pri načítaní niektorých prehľadov, ako napríklad prehľad vyučujúcich sa začali v logoch servera objavovať dlhé zoznamy SQL dotazov. Ako prvé som si všimol, že je to spôsobené iteráciou cez vzťahy kolekcie modelov. Ako napríklad:

```
@techers = Teacher.all()


@teachers.courses.each do |course|
  puts course.title
end
```

To spôsobí, že pre každý model vo vzťahu je načítaný samostatným SQL selectom, čo ale nechceme, pretože to spôsobuje nadmerné zaťaženie databázy. Po zamyslení nad týmto problémom je jasné, že musíme načítať modely aj všetky ich vzťahy cez ktoré chceme iterovať naraz s použitím JOIN-u. Našťastie ale vývojári ActiveRecord-u na toto správanie mysleli a nezabudli ho implementovať. Riešenie je veľmi jednoduché:

```
@techers = Teacher.includes(:courses).all()


@teachers.courses.each do |course|
  puts course.title
end
```

Pri načítaní modelov použijeme funkciu *includes()* do ktorej môžeme napísať symboly, ktoré reprezentujú jednotlivé vzťahy. Po skontrolovaní logov teraz vidíme, že pri načítaní náhľadu sa teraz spustí iba 1 SQL query.

3.12.2 Dlhé časy požiadaviek

Toto správanie je spôsobené tým, že funkcia v controlleri trvá dlhšie ako je obvyklé a tým zdržiava navrátatenie odpovede užívateľovi. Problém sa vyskytol najmä pri odosielaní e-mailov a generovaní PDF prehľadov. Ide o jednoduchý problém, ktorý sa dá vyriešiť niekoľkými spôsobmi.

V aplikácii je tento problém vyriešený použitím balíčka (*gem 'delayed_job'*), ktorý vykonáva dlho-trvajúce funkcie na pozadí. Balíček má v databáze svoju tabuľku, do ktorej ukladá všetky funkcie, ktoré má vykonať.

Avšak, tento balíček sa musí spustiť externe cez príkazový riadok, preto je múdre pridať aj balíček (*gem 'daemons'*) aby sme mohli tento proces daemonizovať. Spustíme ho zo zložky aplikácie príkazom:

```
$ bin/delayed_job start
```

Neskôr treba na produkčnom serveri tento skript nalinkovať aby sa automaticky spúšťal pri štarte/reštarte servera.

Kapitola 4

Nasadenie a príručka administrátora

Táto príručka je napísaná pre inštaláciu na Unix-like operačných systémoch a je napísaná priamo pre distribúciu *Debian 8 Jessie*. Keďže požadujeme od nášho stroja – kde bude aplikácia nasadená – flexibilitu budeme jazyk Ruby inštalovať s pomocou manažéra prostredia jazyku Ruby nazvaného rbenv. K úspešnému nainštalovaniu manažéra prostredia jazyku Ruby a chodu našej aplikácie potrebujeme najskôr nainštalovať nasledujúce balíčky. Všetky kroky príručky sú vykonávané ako *sudo* užívateľ.

```
$ apt update
$ apt install git-core curl zlib1g-dev build-essential libssl-dev
  libreadline-dev libyaml-dev libsqlite3-dev sqlite3 libxml2-dev
  libxslt1-dev libcurl4-openssl-dev python-software-properties
  libffi-dev nodejs
```

4.1 Inštalácia manažéra prostredia Ruby – rbenv

Rbenv nám umožní inštalovať a meniť verziu jazyka Ruby bez zložitého inštalovania cez aplikáciu v príkazovom riadku, čo výrazne zrýchli a zjednoduší prechod medzi verziami.

Aby sme rbenv mohli používať potrebujeme nainštalovať dve aplikácie z Git repozitárov rbenv aplikácie a to samotný rbenv a ruby-build, ktorý nám umožní inštalovať rôzne verzie Ruby.

```
$ cd
$ git clone https://github.com/rbenv/rbenv.git ~/.rbenv
$ echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
$ echo 'eval "$(rbenv init -)"' >> ~/.bashrc
$ exec $SHELL

$ git clone https://github.com/rbenv/ruby-build.git ~/.rbenv/plugins/ruby-build
$ echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
$ exec $SHELL

$ rbenv install 2.4.0
$ rbenv global 2.4.0
```

Teraz môžeme overiť, či sa Ruby správne nainštaloval pomocou príkazu

```
$ ruby -v
```

Po úspešnom nainštalovaní rbenv a Ruby môžeme nainštalovať balíček Bundler, ktorý nám bude manažovať závislosti v aplikáciach pomocou súboru *Gemfile*.

```
$ gem install bundler
$ rbenv rehash
```

4.2 Príprava aplikácie na serveri

Po nainštalovaní Ruby a Bundler-a môžeme stiahnuť našu aplikáciu na server (odporúčam naklonovať Git repozitár). Aplikáciu umiestnime jednoducho do */home/uzivatel/aplikacia*. Potom prejdeme do adresára kde sme aplikáciu umiestnili a spustíme príkaz:

```
$ bundle install
```

Bundler automaticky nainštaluje všetky potrebné závislosti a aplikácia je pripravená na spustenie. Predtým ale ešte musíme vytvoriť databázu. Keďže, používame iba SQLite, ktoré už bolo nainštalované v kroku vyššie stačí nám spustiť iba príkazy:

```
$ rake db:create
$ rake db:migrate
```

Príkaz (*rake db:create*) nám databázu vytvorí v zložke db v adresári aplikácie a *rake db:migrate* vytvorí tabuľky podľa migrácii.

Posledná časť prípravy aplikácie je už len spustenie balíčka *delayed_job*, aby sme mohli spúšťať úlohy na pozadí aplikácie. To spravíme spustením príkazu z adresára aplikácie:

```
$ bin/delayed_job start
```

Po týchto krokoch je aplikácia pripravená a môžeme prejisť na inštaláciu webového servera.

4.3 Inštalácia webového servera Nginx

Naša aplikácia bude využívať webový server Nginx s plug-inom *Passenger* od spoločnosti *Phusion*, pretože celý proces inštalácie a konfigurácie je veľmi jednoduchý.

Ako prvý krok nainštalujeme PGP kľúč a pridáme HTTPS podporu pre APT repozitára

```
$ apt-key adv --keyserver hkp://keyserver.ubuntu.com:80
  --recv-keys 561F9B9CAC40B2F7
$ apt install -y apt-transport-https ca-certificates
```

V ďalšom kroku pridáme samotný repozitár a aktualizujeme list repozitárov

```
$ sh -c 'echo deb https://oss-binaries.phusionpassenger.com/apt/passenger
  jessie main > /etc/apt/sources.list.d/passenger.list'
$ apt update
```

V poslednom kroku nainštalujeme samotný web server Nginx aj spolu s Passengerom

```
$ apt install -y nginx-extras passenger
$ service nginx start
```

Keď si otvoríme IP stroja vo webovom prehliadači, mala by sa zobraziť štandardná stránka nainštalovaného serveru Nginx. [1]

4.4 Konfigurácia webového servera Nginx

Súbory možno editovať v akomkoľvek textovom editore, ja použijem v príklade editor *Joe*.

Nájdeme nasledujúci riadok v súbore */etc/nginx/nginx.conf* a odkomentujeme ho:

```
include /etc/nginx/passenger.conf;
```

Po úprave súbor uložíme a zavrieme. Teraz otvoríme súbor */etc/nginx/passenger.conf* a zmeníme/pridáme riadok ktorý začína *passenger_ruby* na pričom zameníme UZIVATEL za užívateľa ktorý aplikáciu inštaluje:

```
passenger_ruby /home/UZIVATEL/.rbenv/shims/ruby;
```

Teraz je Nginx nastavený aby pri Ruby aplikáciach využíval aktuálne nainštalovanú verziu Ruby z rbenv. Posledným krokom je nakonfigurovať prepojenie Nginx-u na našu aplikáciu v súbore */etc/nginx/sites-enabled/default*:

```
server {
        listen 80;

        server_name domena.sk;
        passenger_enabled on;
        rails_env    production;
        root         /home/UZIVATEL/APLIKACIA/public;

        # presmerovanie errorov na statickú stránku /50x.html
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
}
```

Samozrejme, treba zameniť UZIVATEL za meno užívateľa, pod ktorým aplikácia beží a APLIKACIA za adresár aplikácie. Posledný krok konfigurácie je reštartovať Nginx, aby sa načítala nová konfigurácia.

```
$ service nginx restart
```

Po tomto kroku by mala aplikácia bežať na zadanej doméne, prípadne IP servera.

Záver

Literatúra

[1] Phusion Holding B.V. Installing passenger + nginx on debian 8 (with apt) - passenger library. `https://www.phusionpassenger.com/library/install/nginx/install/oss/jessie/`. [Online, 20.3.2017].