Warehouse Management System in Ruby on Rails Framework on Cloud Computing Architecture

Kamil Durski, Jan Murlewski, Dariusz Makowski, Bartosz Sakowicz

Abstract – **This article describes internet-based application for warehouse management written in Ruby with use of Ruby on Rails framework and distributed as Software as a Service (SaaS) platform. This approach allows for great compatibility between operating systems as well as makes it possible to access application from all kind of devices – from standalone desktop computers up to tables and all kind of mobile devices. Example application was developed in Ruby on Rails version 2.3.**

Keywords –**Warehouse Management, Ruby, Rails, SaaS**

I. Introduction

These days Internet is the faster developing medium in the whole World. It is not only available via landline, but also mobile phone networks and still growing Wi-Fi hotspots. Along with the global network accessibility, the amount of mobile devices grows. More and more popular are telephones with screens exceeding 3″ as well as netbooks – small laptops with screen size of 10″.

This development in mobile technologies makes present business solutions obsolete and outdated. Most of current Enterprise Resource Planning (ERP) software is published and licensed per seat – meaning they are tied to specific computers, usually stationed in offices. Another disadvantage is the need of installation the software itself as well as additional, required packages and libraries (like frameworks or relational databases).

From the end user point of view, using ERP software via Internet browser should make their work considerably easier and shorten the time required to complete tasks. Applications like Internet Explorer, Firefox or Safari are pre-installed on almost every device that can access Internet. The need of software installation is removed completely, which guarantees that user can access application from virtually everywhere: in business trip via mobile or netbook, in the office and, if needed, from home.

This kind of software is not tied to specific hardware or software platforms. For example, apart from creating orders by phone or in the e-commerce store, company agent can check product availability and place new order directly at client's office.

Manuscript received November 09, 2011.
Katedra Mikroelektroniki I Technik Informatycznych
ul. Wolczanska 221/223 budynek B18, 90-924 Lodz, POLSKA
al. Politechniki 11, 90-924 Lodz, POLSKA
NIP 727-002-18-95
tel. +48 (42) 631 26 45 faks +48 (42) 636 03 27

During past few years a lot of web frameworks [1] were created, most of them in PHP language – Zend Framework, Kohana, CakePHP to name a few. They are very useful at what they do, however they are all limited by language itself, which, like most of popular tools, is struggling with backward compatibility and therefore does not even support all of Object Oriented Programming paradigm. To overcome this, some people dedicated to create web frameworks in different programming languages. Out of those two become vastly popular – Django [2] in Python and Ruby on Rails [3] in Ruby.

Ruby in Rails (short: Rails) is a framework with three basic principles:

1. Convection over Configuration guarantees short (if at all) configuration needed by application.
2. Do not Repeat Yourself (DRY) ensures that no piece of code should be repeated more then once.
3. Model-View-Controller as a main architectural pattern that helps to separate data from logic and templates [20].

This approach allows reducing time needed to create application but has some disadvantages as well. Things like connections between database and class names are created automatically at the cost of reserved names for objects, methods and attributes. Lack of attention might cause conflicts and unpredictable behavior as a result.

Rails application works on top of a webserver. By default Webrick is used, however it is possible to use other solutions like Mongrel or even Apache using Passenger module [4]. The full data flow between end-user and application is shown on Figure 1.

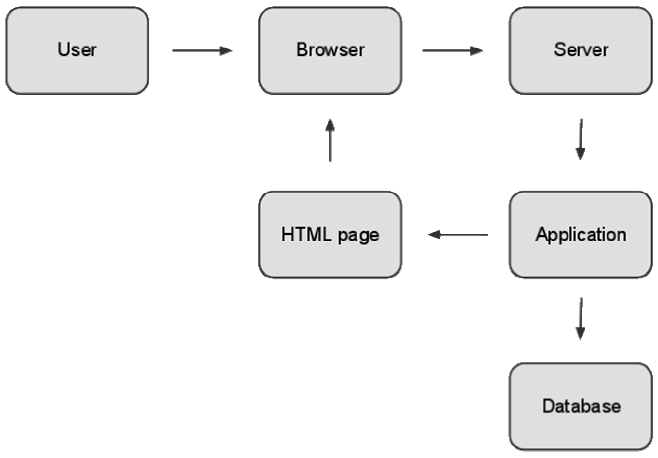

Fig. 1. Client – Server data flow

II. Application Goals

Developed software allows for managing product, orders and clients. Because it is a web application, it will be possible to access it with all popular web browsers. When visiting page, user will need to sign in with three credentials – besides the usual username and password there will also be Account ID. This text-type field will be used to identify user's company within our platform.

As stated before, application will be distributes as SaaS [5]. In this model more than one client is handled by single application instance, and in order to identify client uniqueness we will need additional field. SaaS has a lot of advantages – it offers cost reduction for users and simplified software distribution for developers. Because all of the code is kept at one place, upgrading process is much faster and instantly applies to all of customers.

It is also possible to integrate developed software with third party applications using API. Two methods were created for that purpose – one for checking item's availability and one for adding new orders.

III. Architecture

Described application was created using Model-View-Controller pattern used to separate data from logic and templates. Representational State Transfer (REST) [6] was used as well. This particular architecture was designed for stateless protocols (like HTTP) and defines sets of methods that should be used when creating web services.

TABLE 1
HTTP method used in web services

Method	Is safe?	Is idempotent?
GET	YES	YES
POST	NO	NO
PUT	NO	YES
DELETE	NO	YES

HTTP specification, as described in RFC 2616 [7], describes 8 methods, each one being at the same time English verb. Four of those methods are used for diagnostic and informational purposes and are not used by our application. The other four methods are used to create, read, update and delete resources and their short comparison can be seen in Table 1.

IV. Active Record

Ruby on Rails by default uses Active Record [8] design pattern. It is used in Object-relational_mapping (ORM) and allows to access database fields via class attributes and methods [21]. All information about database's tables and its fields is automatically gathered when object is initialized and adequate methods are created.

As an example we can use simple table called *products* with two columns – *name* (string) and *price* (decimal). In such a case Active Record class name should be *Product*. In order to create new record in database, which executing query below would typically do [23]:

```
INSERT INTO products (name, price) VALUES
('Item 1', 99.99);
```

we could just run the following Ruby code:

```
product = Product.new
product.name = "Item 1"
product.price = 99.99
product.save!
```

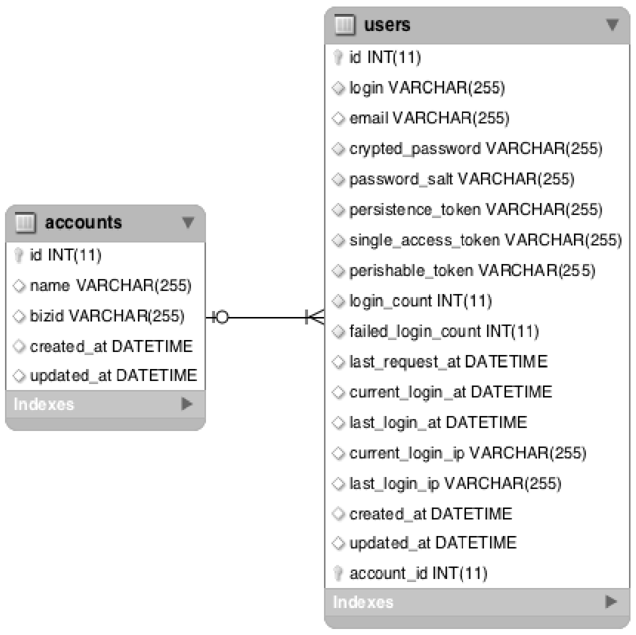

Fig. 2. *Users* and *accounts* table schema

In order to store data within application the MySQL 5.1 database is used [9, 10, 19]. It is free software with open source code, fully compatible with ANSI SQL standards. It supports relations and transactions (if InnoDB engine is used) and supports virtually all modern operating systems.

To configure MySQL a special file needs to be created under *config* directory of our application called *database.yml*. That file will keep authentication details. Example is shown on Figure 3.

```
development:
host: localhost
adapter: mysql
encoding: utf8
database: example_db
username: root
password: ****
```

Fig. 3. Example database.yml file

On Figure 2 a schema of two main tables is shown – *users* and *accounts*. Fields *users.account_id* is connected to *accounts.id* with foreign key and severs as a base for SaaS model.

V. Front End

Front end of our application was created in HTML 5 [11] - fairly new standard, which is slowly replacing its predecessors (HTML 4.01 and XHTML 1.1) [18]. It is still under development, however current browsers already support most of its features. HTML 5 is much more elastic and implements a lot of new tags and attributes but most important – makes browser independent from third party plugins used to play audio or video (required codecs are now built-in).

In order to simplify HTML creation process a different markup language was used – Haml [12]. It is an abstract description of (X)HTML along with some helpers that allow to create dynamic content. Haml greatly simplifies the process of HTML writing and the created code is even up to 50% smaller, as shown on Figure 4 and Figure 5.

```
#box
  .title
    %h1
      = link_to @title, page_url
  .content
      %p= render :partial => 'box_content'
```

Fig. 4. Example Haml code

```
<div id="box">
  <div class="title">
    <h1><%= link_to @title, page_url
%></h1>
  </div>
  <div class="content">
    <p><%= render :partial =>
'box_content' %></p>
  </div>
</div>
```

Fig. 5 Example HTML code

It is very easy to notice that Haml code takes less space and uses very few special characters.

Along with HTML we will also use Cascading Style Sheet (CSS) to separate page content from presentation details. On top of it we will use jQuery - JavaScript library that will allow to add some dynamic effects to website, like dropdown menus and simple AJAX features [22].

VI. Back End

One of the biggest features of Rails is support for external plugins. Their main role is to extend functionality with sets of features that did not make it into the core of framework.

And so our application uses a few of them listed below:

– **MySQL** – database support.

– **Haml** – enables support for markup language with the same name.

– **Authlogic** – small and easy extension that allows to quickly implement web session and users authentication. Also supports Facebook, Twitter and OpenID integration.

– **Searchlogic** – extensively use metaprogramming feature in Ruby by creating set of methods for Rails model that allows finding records in database with easy.

– **JRails** – drops Prototype JavaScript library support in favor of jQuery [12,13].

– **Formtastic** – creates helper methods for HTML forms and automatically generates necessary fields that are semantically valid. Also supports model relationships and implements advanced internationalization methods.

– **Inherited Resource** – easy REST support for Rails application. This plugin extend Rails with module that automatically adds methods to controller class that create, read, update and delete resources – no additional code is required.

– **Responders** – small extension that add necessary headers to sessions and HTTP headers when adding, updating or deleting database records. Required by Inherited Resources.

In order to better understand the full capability of rails plugins in the Figure 6 a simple controller class is shown.

```
class CustomersController <
InheritedResources::Base
      before_filter :check_account, :only =>
[:show, :edit, :update, :destroy]
      before_filter :require_user
      respond_to :js, :only => [:index]
end
```

Fig. 6 Source code of customer's controller

This small piece of code is responsible for all operations run on customer records – create, read, update and delete – no coding is needed thanks to its parent class from Inherited Resources plugin. Besides that it also call methods used to validate user and its access before running record-based methods and makes sure that index action will respond to JavaScript requests (used in AJAX - based record filters).

VII. Summary

The main purpose of created application was to show that Ruby on Rails is a real competition for currently most popular PHP language and its frameworks. Even though less then 5% of sites use Ruby it is stable and supported enough to be capable of running big commercial projects thanks to one of its main advantages - speed. Even though in benchmarks the efficiently of different Ruby implementations is substantial [15], its still a lot faster then PHP. A few of most popular Ruby project include Twitter or Dig social networks or widely used business solutions offered by 37 Signals – Basecamp and Campfire.

Although Ruby and Rails are available for Microsoft Windows it is still best supported in Unix operating systems like Linux or MacOS X thanks to command-line utility called “gem”. Created application was developed with Firefox, Safari and Internet Explorer in mind and works seamlessly under desktop computers as well as mobile systems like iOS and Android.

Example integration can be easily created using built-in API support in designed application. E-commerce stores or any other application that allows for XML integration can be synchronized. This approach did not create much of additional work. In fact it was a matter of adding a few new template files and some additional logic to controllers. Everything else was handled directly by Rails framework core components.

Ruby language might be hard for people that are used to imperative programming languages (like C/C++ or Java), as its core language constructions are a bit different and so is approach to some problems. Nonetheless after reading some popular books and tutorials [16,17] most of the people will appreciate what it has to offer and how much easier and faster programming can be.

ACKNOWLEDGMENT

This research was supported by the Technical University of Lodz.

REFERENCES

[1] PHP Frameworks, http://www.phpframeworks.com/
[2] Django project, http://www.djangoproject.com/
[3] Ruby on Rails, http://rubyonrails.org/
[4] Phusion Passenger, http://www.modrails.com/
[5] NIST – Computer Security Division – Cloud Computing, http://csrc.nist.gov/groups/SNS/cloud-computing/
[6] Roy Fieldings Dissertation, Chapter 5, http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
[7] RFC 2616 – HTTP/1.1, http://tools.ietf.org/html/rfc2616
[8] ActiveRecord,http://api.rubyonrails.org/classes/ActiveRecord/Base.html
[9] MySQL 5 Storage Engines http://dev.mysql.com/doc/refman/5.0/en/storage-engines.html
[10] MySQL 5.1 Numeric Types http://dev.mysql.com/doc/refman/5.1/en/numeric-type-overview.html
[11] W3C HTML5 Specification, http://www.w3.org/TR/html5/
[12] HAML Reference, http://haml-lang.com/docs/yardoc/file.HAML_REFERENCE.html
[13] Prototype JavaScript framework, http://www.prototypejs.org/
[14] jQuery, http://www.jquery.com/
[15] The Great Ruby Shootout (July 2010), http://programmingzen.com/2010/07/19/the-great-ruby-shootout-july-2010/
[16] Sam Ruby, Dave Thomas, David Heinemeier Hannson, *Agile Web Development With Rails 3rd Edition*, The Pragmatic Bookshelf, June 2008
[17] Ruby in Twenty Minutes, http://www.ruby-lang.org/en/documentation/quickstart/4/
[18] Sakowicz B., Wójtowski M., Zalewski P., Napieralski A., „ Problems of Standardization in Web Technologies” XI Konferencja „ Sieci i Systemy Informatyczne, Łódź, październik 2003, pp. 111-114, ISBN 83-88742-91-4
[19] Wilk S., Sakowicz B., Napieralski A.,” Wdrażanie aplikacji J2EE w oparciu o serwer Tomcat 5.0 i bazę danych MySQL” SIS XII Konferencja, Łódź paździerink 2004, pp.379-386, ISBN 83-7415-042-4
[20] Wojciechowski J., Sakowicz B., Dura K., Napieralski A.,” MVC model struts framework and file upload issues in web applications based on J2EE platform” TCSET’2004, 24-28 Feb. 2004, Lviv, Ukraine, pp., 342-345 , ISBN 966-553-380-0
[21] Ziemniak P., Sakowicz B., Napieralski A.: "Object oriented application cooperation methods with relational database (ORM) based on J2EE Technology"; CADSM'2007; ISBN 978-966-553-587-4
[22] Cisz, M.; Zabierowski, W.: "Community services portal based on the campus of Technical University of Lodz. Using Ajax technology".; TCSET 2010 , Page(s): 183 - 184; ISBN 978-966-553-875-2
[23] Murlewski J., Kowalski T., Adamus R., Sakowicz B., Napieralski A., “Query Optimization in Grid Databases”, 14th International Conference Mixed Design of Integrated Circuits and Systems MIXDES 2007, 21-23, str. 707-710, ISBN 83-922632-4-3