Czech University of Life Science Prague

Faculty of Economics and Management

Department of Information Engineering

Bachelor thesis

Design and implementation of web application using Ruby programming language

Author: Varvara Frolova

Supervisor: Ing. Jiří Brožek, Ph.D.

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Varvara Frolova

Informatics

Thesis title

Design and implementation of web application using Ruby programming language

---

Objectives of thesis

The thesis deals with using the Ruby programming language for creating web applications. The main objective of the thesis is to design and implement a website for sharing information about cultural events. The subgoal of the thesis is to describe the Ruby language in the context of web application development.

Methodology

The methodology of the thesis is based on analysis of technical and scientific sources related to web application development with the main focus on using the Ruby programming language for such purposes. Based on the synthesis of the gained knowledge, the Ruby language and its usage in web application design wil be described.

The practical part of the thesis will concern analysis, design and implementation of web application for sharing information about cultural events. The application will be deployed and tested. The application will be evaluated and possible changes and improvements will be proposed based on the evaluation.

The proposed extent of the thesis

35-40 pages

Keywords

Web application, website, programming language, Ruby

Recommended information sources

Bruce A. Tate. Seven Languages in Seven Weeks: A Pragmatic Guide to Learning Programming Languages. Pragmatic Bookshelf, 2010. ISBN 193435659X.

Dave T. – Fowler C. – Hunt A. Programming Ruby: The Pragmatic Programmers' Guide. Pragmatic Bookshelf, 2004. ISBN 0974514055.

Flanagan D. The Ruby Programming Language. Sebastopol, 2008. ISBN 0596516177.

Hartl M. Ruby on Rails Tutorial: Learn Web Development with Rails. Addison Wesley Professional, 2015. ISBN 0134077709.

Metz S. Practical Object-Oriented Design in Ruby: An Agile Primer. Addison-Wesley Professional, 2012. ISBN 0321721330.

Quick Start Guides. RUBY Beginner's Crash Course: Ruby for Beginner's Guide to Ruby Programming, Ruby On Rails & Rails Programming. CreateSpace Independent Publishing Platform, 2015. ISBN 1518721648

Expected date of thesis defence

2016/17 SS – FEM

The Bachelor Thesis Supervisor

Ing. Jiří Brožek, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 1. 11. 2016

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 1. 11. 2016

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 14. 03. 2017

Declaration

I hereby declare to have compiled this final thesis “Design and implementation of web application using Ruby programming language” entirely myself and in accordance with recommendations of my supervisor, that I indicate all the literature and other supporting materials used in the index of bibliography. Therefore I declare that I have not violated any right of third parties while writing my master’s thesis.

In Prague 15th of March, _____ Varvara Frolova

Acknowledgements

Therefore, I would like to thank my supervisor Ing. Jiří Brožek, Ph.D. for Your time, instructions and advice that was very helpful and essential during writing of this thesis.

Design and implementation of web application using Ruby programming language

---

Abstract

The aim of the thesis is to be familiar with Ruby programming language and other web application technologies, design, and implement the web application using Ruby programming language. The methodology used for running this research includes theoretical overview, case study, and deductive thinking.

There is no area that doesn't use computer technology. To get the desired information doesn't require going to the library. It is just enough to have a mobile phone or PC with Internet connection. Most often no one thinks about how really the Internet works and how the desired information is displayed on the screen. And who are these people that create all of these useful sites and services, which methods and tools they use for it.

Key words

Ruby language, programming language, framework, web application, Ruby on Rails.

Návrh a implementace webové aplikace s využitím programovacího jazyka Ruby

---

Souhrn

Cílem této bakalářské práce je seznámit s programovacím jazykem Ruby, dalšími technologiemi pro webové aplikace a navrhnout a implementovat webovou aplikaci s využitím jazyka Ruby. Metodika práce využívá teoretickou rešerši, případovou studii a deduktivní uvažování.

Neexistuje oblast, kde by se nevyužívaly počítačové technologie. K získání požadované informace již není nutné chodit do knihovny. Stačí mít mobilní telefon nebo PC s připojením k internetu. Nikdo většinou nepřemýšlí o tom, jak vlastně internet funguje a jakým způsobem se požadované informace zobrazí na obrazovce. A kdo jsou lidí, kteří tyto užitečné stránky a služby vytváří a jaké metody a nástroje k tomu používají.

Klíčová slova

Jazyk Ruby, programovací jazyk, framework, webová aplikace, Ruby on Rails.

Contents

1. Introduction

Connection between languages we are thinking and programming very close by each other. The language provides the programmer a set of conceptual tools and if they are not up to the task, programmer can simply ignores them. Linguistic resources cannot guarantee good programming skills and the absence of errors. Nowadays almost all programs created with programming languages. Important part of modern computers is software, which is continuation of the logical parts of a computer and extends the capabilities of the equipment and the scope of their use. The main purpose of the software is increasing the efficiency of user's work by reducing the time and costs in the preparation and implementation of programs.

At the present computer technologies development stage is impossible to imagine skilled specialist that has no idea about information technology. For free orientation in the information flow the modern specialist in IT sphere should be able to receive process and use information: especially using computers as well as telecommunications and other types of communication and of course deal with programming languages.

Nowadays, improving of computer's performance and change in the composition in software which we using in app creation becoming more and more important. Currently in the world, there are hundred used programming languages. Each has its own purpose.

In each of the areas of software development, there are usually at least two or three languages, able to cope with the task. However, the approach to these languages is usually somewhat different, and this raises the question of priorities in the development and individual preferences of the programmer. Ultimately, how you are thinking and understand of what needs to be done and how this then support is important. Therefore, it makes sense to choose the language in which will be easier and more convenient to write, which will give the opportunity to reach potential.

2. Objectives and Methodology

Objectives

The thesis deals with using the Ruby programming language for creating web applications. The main objective of the thesis is to design and implement a website for sharing information about cultural events. The sub goal of the thesis is to describe the Ruby language in the context of web application development.

Methodology

The thesis divided into two parts. The methodology of the thesis is based on analysis of technical and scientific sources related to programming languages and web application development with the main focus on using the Ruby programming language for such purposes. Based on the synthesis of the gained knowledge, the Ruby language and its usage in web application design will be described.

The practical part of the thesis will concern analysis, design and implementation of web application for sharing information about cultural events. The application will be deploy and tested. The application will evaluate and possible changes and improvements will proposed based on the evaluation.

3. Literature Overview

3.1 Programming languages

There is no universally accepted definition of "programming language". Programming languages, in my understanding, are artificial languages, which strictly formalized with special syntax of the language.

Programming language has a set of characters and rules for combining them that have the following characteristics:

1. There is unnecessary to know Machine code;
2. Must have option for conversion to other computers;
3. Instruction explosion (from one to many);
4. There is a notation, which is closer to the original problem than assembly language would be.[1]

In 1945, John Von Neumann was working at the Institute for Advanced Study.[2] He called the father of modern computers. John Von Neumann formulates the principles of organization of computer systems, which formed the basis of all modern computers:

1. Universal computing machine should include several key components such as arithmetic unit, memory, control unit and communication with the user.
2. Computing machine should be fully automatic, i.e. after the beginning of calculations; the machine should not depend on the user.
3. Computing machine should remember the intermediate results of calculations and commands which controls the operation of the machine.
4. Both the data and commands to the computing machine must translate to a numeric code, which must be stored in memory and be automatically recognized.
5. In the computing machine must be a control device that automatically executes commands, which are stored in memory.
6. Computing machine should have an arithmetic device that performs elementary arithmetic operations: addition, subtraction, multiplication, and division.
7. Computing machine should have input-output devices, which used for

[1] Jean E. Sammet. Programming Languages: History and Future. Communications of the ACM, 1972
[2] *History of computer programming languages* [online]. [cit. 2016-11-16]. Available from: https://cs.brown.edu/~adf/programming_languages.html

communication between users and computing machine.[3]

All the languages, which developed in this period, only FORTRAN (designed at IBM for scientific computing) and APT (the first language for a specialized application area). [4]

Most Prolific Years: 1958-1959. The most significant years in the history of programming languages are 1958 and 1959. The following events all occurred during that period:

1. John McCarthy of MIT created the LISt Processing language. It designed for Artificial Intelligence research.
2. Describing ALGOL language. This was the foundation for much of the theoretical work done in programming languages since then. Its major contribution is being the root of the tree that has led to such languages as Pascal, C, C++, and Java.[5]

Niklaus Wirth began Pascal in 1968. This language provides intuitive syntax. It named in honor of the French mathematician, physicist and philosopher Blaise Pascal. One of the purposes of the language Pascal is teaching students of structured programming. Pascal still considered as one of the best languages for primary education programming. Its modern modifications such as Object Pascal, is widely used in industrial programming -Delphi. [6]

Dennis Ritchie while working at Bell Labs in New Jersey developed C in 1972. Originally developed for the UNIX operating system, but later ported too many other platforms. According to the design of the C language, its design closely mapped to typical machine instructions, so that it used in projects, which was typical for Assembly language, including operating systems. [7]

In the late 1970's and early 1980's, a new programming method being developed. It was Object Oriented Programming. Bjarne Stroustroup liked this method and developed extensions to C known as "C with Classes." This set of extensions developed into the full-featured language C++, which released in 1983. [8]

What are the reasons that one language becomes widely used, while others remain for all practical purposes the property of a small group? The generally obvious answer is practicality;

---

[3] Fandom, *John von Neumann* [online]. [cit. 2016-11-18]. Available from: http://religion.wikia.com/wiki/John_von_Neumann
[4] Roundcrisis, *Past and Future* [online]. [cit. 2016-11-18]. Available from: http://www.roundcrisis.com/2016/01/03/Sammett-72/
[5] *History of computer programming languages* [online]. [cit. 2016-11-16]. Available from: https://cs.brown.edu/~adf/programming_languages.html
[6] WikiBooks, *Pascal Programming* [online]. [cit. 2016-11-19]. Available from: https://en.wikibooks.org/wiki/Pascal_Programming
[7] *Dennis M. Ritchie* [online]. [cit. 2016-11-19]. Available from: http://www.bell-labs.com/usr/dmr/www/
[8] *C++ and Java* [online]. [cit. 2016-11-19]. Available from: http://anhpopeye.blogspot.cz/

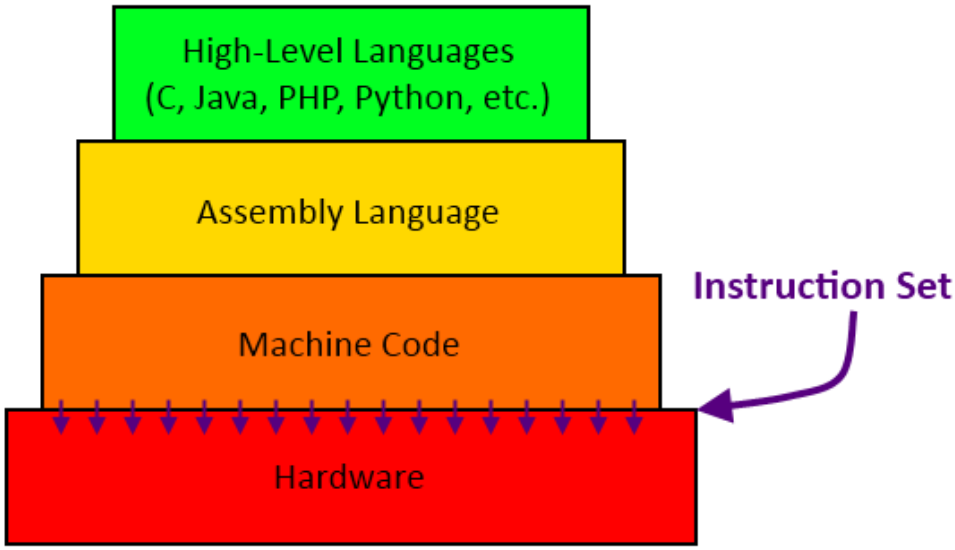
the language designed to solve a large class of problems and many good compilers written on it.[9]

In summary, if trace the history programming languages such as C and Pascal (as well as less popular Basic, FORTRAN, or Ada), it appears that they were created at the turn of the 60-ies and 70-ies. In other words, the age of modern programming languages, passed for a third decade that for the computer industry is an extreme term. [10]

3.2. Computer programming languages classification

The program is an algorithm written in a programming language. Program is sequence of statements of the language. [11]

Figure 1: computer programming languages classification



Source: *Computer programming languages classification* [online]. [cit. 2016-11-16]. Available from: http://wdc65xx.com/lessons/understanding-the-hierarchy-of-programming-languages/

3.2.1 Low-level programming languages

Low-level languages designed to operate and handle the entire hardware and instructions set architecture of a computer directly.[12]

---

[9] *History of computer programming languages* [online]. [cit. 2016-11-16]. Available from: https://cs.brown.edu/~adf/programming_languages.html

[10] Peter Grogono. The Evolution of Programming Languages. Department of Computer Science Concordia University, 2002.

[11] *History of computer programming languages* [online]. [cit. 2016-11-16]. Available from: https://cs.brown.edu/~adf/programming_languages.html

[12] Techlopedia, *Low-Level Language* [online]. [cit. 2016-11-20]. Available from: https://www.techopedia.com/definition/3933/low-level-language

Mnemonic designation typically used for description of machine instructions. This allows memorizing the commands not in sequence of binary zeros and ones, but in the form of human language meaningful abbreviations of words. In addition to machine instructions, low-level programming languages can provide additional functionality, such as macros. Often, these languages allow working with variables instead of specific memory locations. [13]

1. Machine language (the 40-50 years of XX century). A program in machine language – very long sequences of ones and zeros, was machine dependent, i.e. for each computer it was necessary to create own program.

   Main disadvantages are:

   - Really difficult to write a code, because binary system is not "user-friendly" for human;
   - Requires excellent memorizing power. [14]

2. Assembler (early 50-ies of XX century). Instead of 1 and 0 programmers could use the operators (MOV, ADD, SUB, etc.) that are similar to English words. Programs in Assembly language are machine-dependent. To convert to machine code was used compiler (the program-translator into machine code).

   Main disadvantages are:

   - Machine dependent;
   - Long programs;
   - Hard to learn and write programs. [15]

3.2.2 High-level programming languages

These languages were Machine independent (not created for a specific type of computer). However, for each language developed its own compilers. The programmer designed these programming languages for speed and ease of use. High-level programming languages developed for platform independence. Platform dependency shifted to tool programs - translators, which are, compile a code to elementary machine instructions. High-level languages tend not only to facilitate the solution of complex software problems, but also to simplify

[13] *Technology* [online]. [cit. 2016-11-20]. Available from: http://shilpikum.blogspot.cz/
[14] Computing Concepts, *Programming Languages* online]. [cit. 2016-11-20]. Available from: http://highered.mheducation.com/sites/0072834110/student_view0/chapter13/programming.html
[15] IMB knowledge center, *Assembler* [online]. [cit. 2016-11-20]. Available from: http://www.ibm.com/support/knowledgecenter/SSENW6_1.6.0/com.ibm.hlasm.v1r6.asma400/asmr102112.htm

software porting. Programs written in high-level languages are easier to understand by the programmer, but less efficient than their analogues that are created in low-level languages. [16]

Examples: C++, C#, Delphi, FORTRAN, Java, JavaScript, Pascal, PHP. The high-level languages have ability to work with complex data structures. Most of them have integrated support for string types, objects, file input/output, etc.

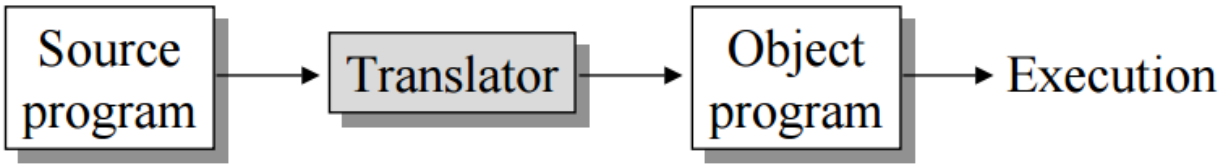
High-level languages divided further into interpreted and compiled:

- Compiled languages give a result of the executable module, such as EXE file, it obtained from a program's source code by compiling. Source code in high-level language automatically processed by the compiler and translated into machine code, which (with data) recorded inside the executable file.
- An interpreted language is not giving an executable file, it always remains in the source code, and in this case, the source code gets name "script". The script runs sequentially on a virtual machine. Therefore, to run the program, the computer must have a corresponding virtual machine, which will execute the script.

3.3 Translators

The text written in a programming language is unknown for the computer.[17] It should translate to machine code. This translation from programming language to machine code called "translation", and special programs -translators.

Figure 2: Program execution



Source: Program execution [online]. [cit. 2016-11-16]. Available from: http://www.sfcc.edu.hk/academic_subjects/cs/teaching_notes/CIT_theory/14_prg_lang.pdf

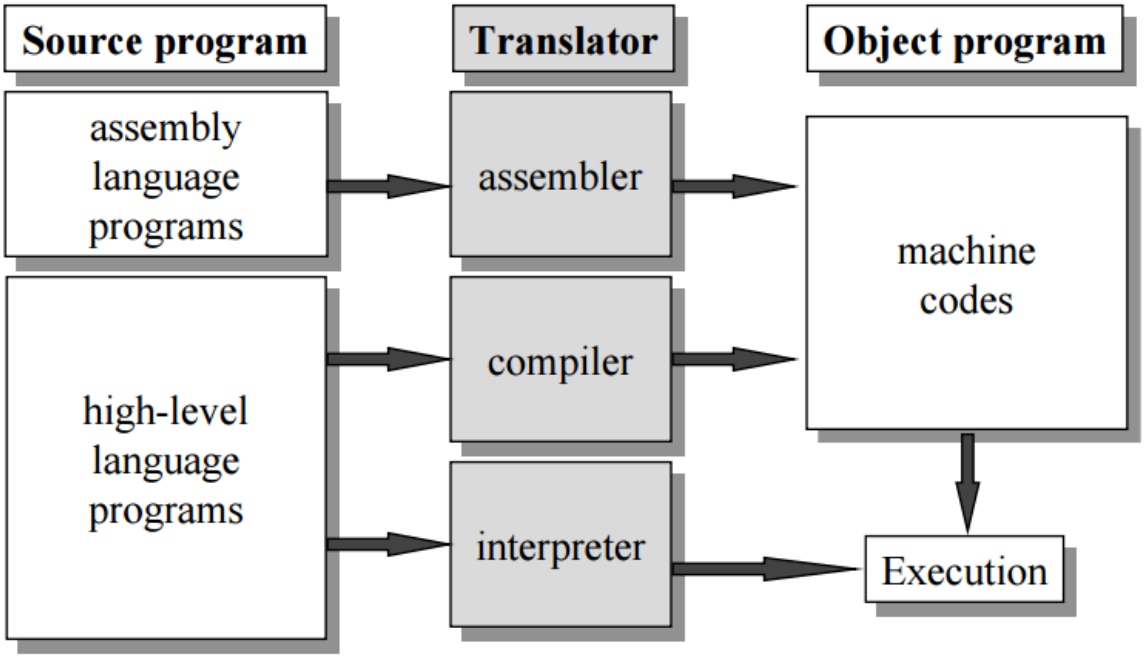
Roles of translator are:

[16] Webopedia, *What is translator* [online]. [cit. 2016-11-18]. Available from: http://www.webopedia.com/TERM/H/high_level_language.html
[17] *What is translator* [online]. [cit. 2016-11-18]. Available from: https://www.reference.com/technology/computer-language-translator-1c3fb8a9ee90cfde#

- Translating input of the the high-level language program into an equivalent machine language program.
- Providing diagnostic messages if programmer violates specification of the high-level language program. [18]

Currently, the translators divided into three main groups: assemblers, compilers and interpreters.

Figure 3: Types of translator



Source: Program execution [online]. [cit. 2016-11-16]. Available from: http://www.sfcc.edu.hk/academic_subjects/cs/teaching_notes/CIT_theory/14_prg_lang.pdf

3.3.1 Assembler

Assembler is a system service program that converts a symbolic design commands into machine language. A specific feature of assemblers is providing a verbatim translation of one symbolic command into one machine command. Thus, Assembly language designed to facilitate the perception of the computer system programs and speed up programming in this system. For the programmer is much easier to remember the mnemonic machine instructions than their

[18] *Roles of translator* [online]. [cit. 2016-11-18]. Available from: http://ecomputernotes.com/compiler-design/translators-and-its-type

binary code. Assembly language also contains additional directives that facilitate the computer resource management, the writing of repetitive pieces, the construction of multi-module programs. Therefore, the expressiveness of the language is much richer than just symbolic coding, which significantly increases the efficiency of programming. [19]

Table 1: Example of a short program in assembly language

```
MOV EAX,1
SHL EAX,5
MOV ECX,17
SUB EAX,ECX
....
```

Short program in assembly language [online]. [cit. 2016-11-28]. Available from: http://www.friedspace.com/assembly/intro.php

3.3.2 Compiler

The compiler is a service program that perform the translation into machine code program which written in a source programming language. Like the assembler, the compiler converts a program from one language to another. The generated machine code can later executed many times against different data each time. However, commands of the source language differ significantly in the organization and capacity of commands in machine language. There are languages in which one command of the source language translated into seven-ten machine instructions. However, there are languages in which every command can fit a hundred or more machine commands (e.g., Prolog). In addition, in the source languages is strongly typed data is often used which carried out through their preliminary description. Programming can rely not on the algorithm coding, but on careful consideration of data structures or classes. The process of translating from such languages usually called compilation, and the source languages generally refer to languages high-level programming (or high-level languages). The programming language abstraction from a system of commands of the computer led to the independent creation of a wide variety of languages focused on specific tasks. Languages for

[19] Fried space, *Assembly* [online]. [cit. 2016-11-28]. Available from: http://www.friedspace.com/assembly/intro.php

scientific calculations, economic calculations, access databases and others appeared.[20]

3.3.3 Interpreters

Interpreter is a program or device that operating the stream and the source code implementation. It directly executes the operations specified in the source program when the user gives the input. Unlike the compiler, the interpreter does not generate the output program in machine language. Recognizing the command of the source language, it immediately performs it. Compilers and interpreters are used the same methods of analysis of the source program text. However, the interpreter allows starting processing the data after writing even a single command-line code. This makes the process of developing and debugging programs more flexible. The interpreter can be adapted to any machine architecture, developing it only once on a widely used programming language. The disadvantage of interpreters is the low speed of program execution. Usually interpreted programs run 50-100 times slower than programs written in machine code. [21]

3.3.4 Difference between compiler and interpreter

The difference between the functioning of compiler and interpreter will be clear from the table of comparison given below:

Table 2: Difference between compiler and interpreter

	Compilers	**Interpreters**
Translation of source program	The whole program before execution	One line at a time when it runs
Frequency of translation	Each line is translated once	Has to be translated every time it is executed - slower
Object program	Can be saved for future execution without the source	No object program is generated, source program

[20] Lambda, *What is compiler* [online]. [cit. 2016-12-10]. Available from: https://lambda.uta.edu/cse5317/notes/node3.html
[21] Computer Notes, Interpreters [online]. [cit. 2016-12-10]. Available from: http://ecomputernotes.com/compiler-design/translators-and-its-type

	program	must be present for execution

Source: Difference between compiler and interpreter [online] [cit. 2016-12-14]. Available from: http://www.sfcc.edu.hk/academic_subjects/cs/teaching_notes/CIT_theory/14_prg_lang.pdf

A compiled runs faster because it converted into machine code all at once. Computer can easily understand machine code and therefore can run the whole program quickly. Nevertheless, this also means that the entire compiled code has to reside in the memory and thereby making the compiled programs more memory guzzling. [22]

[22] Techwelkin, *Difference between compiler and interpreters* [online] [cit. 2016-12-14]. Available from: http://techwelkin.com/compiler-vs-interpreter

4. Ruby programming language

Ruby is a dynamic, open source programming language with a focus on simplicity and productivity. It has an elegant syntax that is natural to read and easy to write. Its creator Yukihiro "Matz" Matsumoto, blended parts of his favorite languages (Perl, Smalltalk, Eiffel, Ada, and Lisp) to form a new language that balanced functional programming with imperative programming. [23]

For every one who wishes to follow the path of a programmer, sooner or later will be the question of the choice of the sphere of software development (web, gaming, mobile..) and related tools (programming languages, frameworks...). Moreover, here, an important role is tools that can use. In fact, they largely define what ultimately will build, how quickly, what properties it will have, etc.

Matsumoto wanted it to be truly object-oriented and easy to use high-level language. Thus, the main purpose of Ruby is creation of a powerful and in the same time clear programs where is speed of the program not so important, how short time of development, clearness and simplicity of syntax. The language follows the principle of "least surprise": the program should behave as the programmer expects. He also inherited the ideology of the Perl language, providing the programmer the option to achieve the same result in several ways. [24]

After few years, Ruby spread around the world, helped by the emergence of English-language books, "Programming Ruby: The Pragmatic Programmers' Guide" and "Why's (Poignant) Guide to Ruby". Until 2004, the year Ruby was not widely known in Europe and the United States. The real interest in Ruby has provoked the emergence of Ruby-On-Rails (RoR) is a framework for developing web applications. [25]

To version 1.8 languages has evolved, while remaining compatible with previous versions, then later the developers of Ruby, led by Akihiro Matsumoto decided that to move forward it should abandon the 100% compatibility. Therefore, the development of the Ruby divided into two branches: support of versions 1.8.* and the new versions 1.9.*, which are the forerunner of the next version of the language, Ruby 2. [26]

---

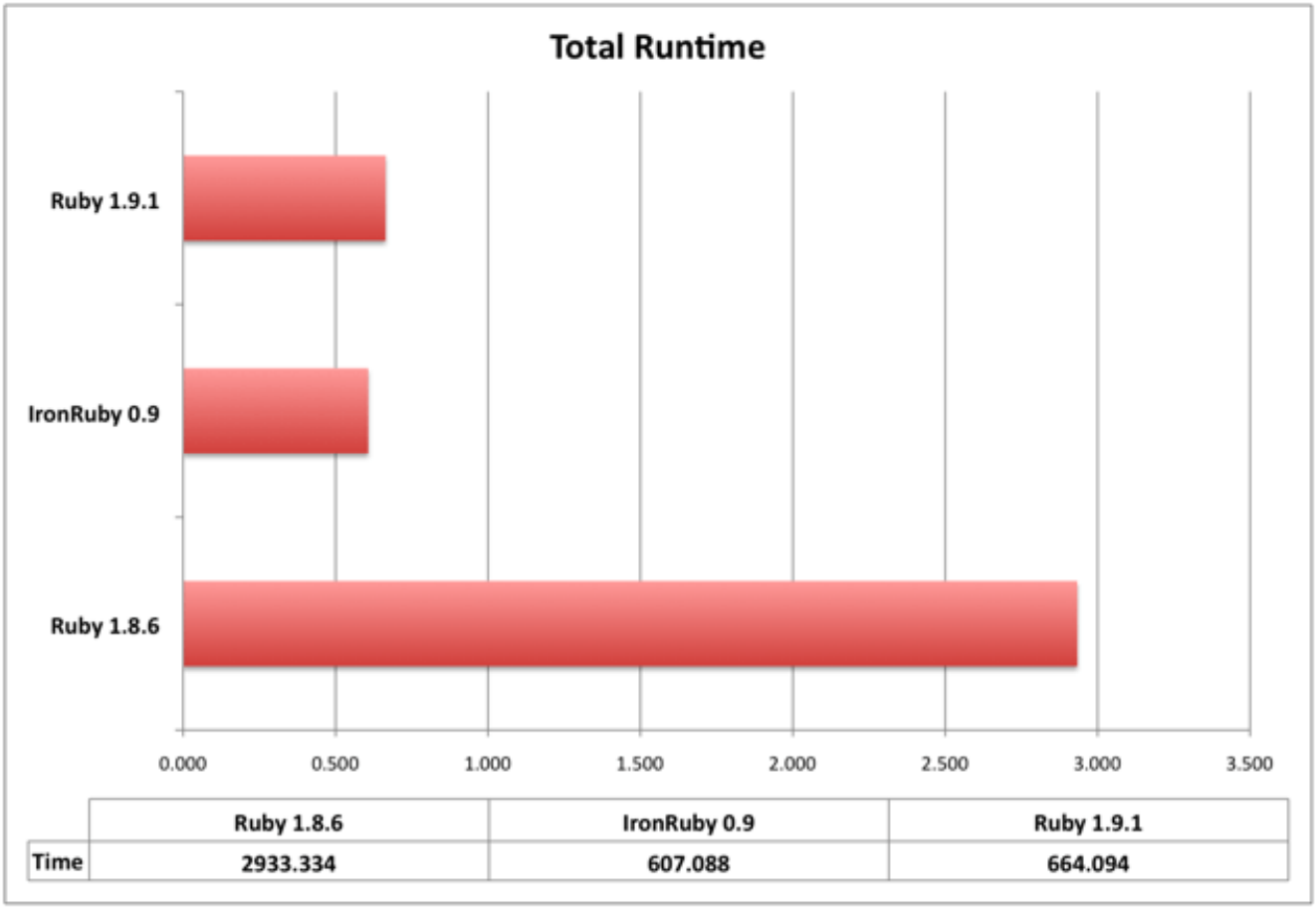
[23] Ruby, *What is Ruby* [online] [cit. 2016-12-14]. Available from: https://www.ruby-lang.org/en/

[24] Wikipedia, *Ruby Programming Language* [online] [cit. 2016-12-14]. Available from: https://en.wikipedia.org/wiki/Ruby_(programming_language)

[25] Read and Write, *Ruby books* [online] [cit. 2016-12-14]. Available from: http://readwrite.com/2011/04/08/10-free-e-books-on-ruby-for-be/

[26] Wikipedia, *Ruby Programming Language* [online] [cit. 2016-12-14]. Available from:

Figure 4: Total Ruby runtime



Source: *Ruby Runtime* [online] [cit. 2016-12-14]. Available from: http://programmingzen.com/performance-of-ironruby-ruby-on-windows/

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index updated once a month. The ratings based on the number of third party vendors, skilled engineers, courses. Popular search engines such as Google, Yahoo!, Wiki, and YouTube helped to calculate the ratings. It is important to note that the TIOBE index is not about the best programming language or the language in which most lines of code been written. [27]

According TIOBE Programming Community index Ruby takes the 11th place, and this is 1.5% of the total market and for a global scale, it is quite well. The paces of Ruby development are impressive: over the last 2.5 years, they have increased in 2,5 times! This gives us hope that Ruby has every chance soon to find its niche to continue to keep quite large part of the market of web development.

---

https://en.wikipedia.org/wiki/Ruby_(programming_language)

[27] Tiobe, *Whai is Tiobe index* [online] [cit. 2016-12-16]. Available from: http://www.tiobe.com/tiobe-index//

Figure 5: TIOBE Programming Community Index

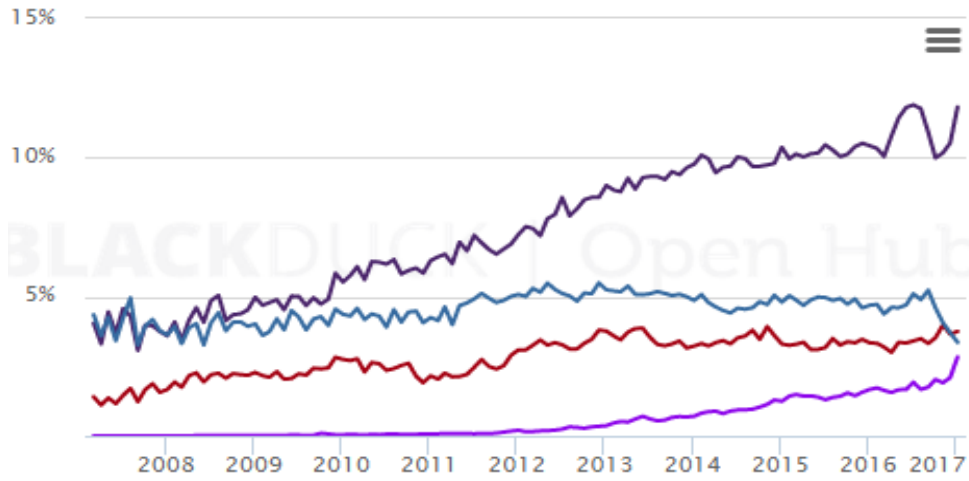
Feb 2017	Feb 2016	Change	Programming Language	Ratings	Change
1	1		Java	16.676%	-4.47%
2	2		C	8.445%	-7.15%
3	3		C++	5.429%	-1.48%
4	4		C#	4.902%	+0.50%
5	5		Python	4.043%	-0.14%
6	6		PHP	3.072%	+0.30%
7	9	^	JavaScript	2.872%	+0.67%
8	7	v	Visual Basic .NET	2.824%	+0.37%
9	10	^	Delphi/Object Pascal	2.479%	+0.32%
10	8	v	Perl	2.171%	-0.08%
11	11		Ruby	2.153%	+0.10%

Source: *TIOBE* [online] [cit. 2016-12-16]. Available from: http://www.tiobe.com/tiobe-index//

Statistics from Black Duck Software with the graph. The graph compares the languages picked by the user. The height of each point on the graph is the sum of all commits in that month that included at least one line of change for that language. A commit that changed two languages will count for each language. The lines show the count of monthly commits made by source code developers. Commits including multiple languages counted once for each language.[28]

[28] BlackDuck, *Statistic from BlackDuck* [online] [cit. 2016-12-16]. Available from: http://blog.openhub.net/compare_languages/

Figure 6: Statistics from Black Duck Software



Source: Black Duck Software statistics [online] [cit. 2016-12-16]. Available from: https://www.openhub.net/languages/compare?utf8=%E2%9C%93&measure=commits&language_name%5B%5D=golang&language_name%5B%5D=php&language_name%5B%5D=python&language_name%5B%5D=ruby&language_name%5B%5D=-1&commit=Update

However, it should understand that learning languages and learning new technologies are two different things. Always learn new technologies! The study of technology provides knowledge and experience (as long as your programming language supports of this technology).

4.1 Frameworks

Frameworks are software products that simplify the creation and maintenance of technically complex or loaded projects. A framework usually contains only the basic software modules and the developer based on them implements all project-specific components. This achieves not only high speed of development, but also great performance and reliability solutions. This platform is suitable for creating websites, business applications and web services.[29]

A framework differs from a library that the library can be used in the software product as a collection of subsystems close to the functionality without affecting the architecture of a major software product and not imposing on it restriction. The framework dictates the rules to build the

[29] Code Project, *What is framework* [online] [cit. 2016-12-19]. Available from: https://www.codeproject.com/Articles/5381/What-Is-A-Framework

architecture of the application, establishing in the initial stage of development the default behavior, forming a frame, which will need to expanded and changed according to specified requirements. The framework may include support programs, code libraries, scripting language and other software that facilitates the development and integration of different components in a big software project.[30]

One of the main advantages when using frameworks is that web applications often use a standardized organizational structure of the components. At some point, a framework becomes all three simply out of necessity. You cannot implement a methodology without implementing wrappers and an architecture.

Frameworks advantages:

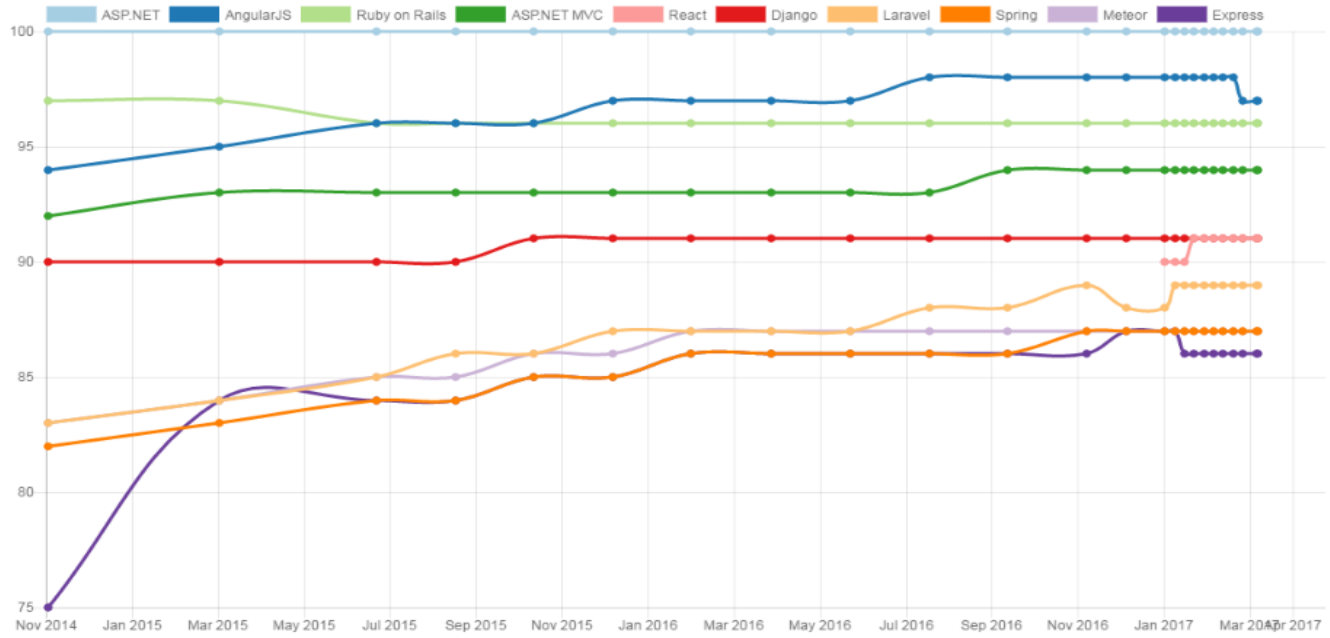
- Development on the framework allows achieving ease of maintainability of the project;
- It is possible implementation of any business process, not just those who originally laid out the system;
- Solutions on frameworks tend to work much faster and withstand more loads.

Frameworks disadvantages:

- Frameworks contain only the basic components of the business logic in the application-level, and many functions must be implemented individually;
- To develop on the framework, requires an understanding of the business processes which needed to implement.[31]

[30] Code Project, *What is framework* [online] [cit. 2016-12-19]. Available from: https://www.codeproject.com/Articles/5381/What-Is-A-Framework
[31] Infinity Web, *Framework advantages and disadvantages* [online] [cit. 2016-12-19]. Available from: https://nagbhushan.wordpress.com/2010/10/03/framework-advantages-and-disadvantages/

Figure 7: Most popular web frameworks



Source: *Popular Web frameworks* [online] [cit. 2016-12-20]. Available from: https://hotframeworks.com/

4.1.1 Ruby on Rails

Rails is a web application development framework written in the Ruby language. It designed to make programming web applications easier by making assumptions about what every developer needs to get started.[32]

Mostly professionals are working in the programming language Ruby: barrier to entry is quite high, so the programmers in Ruby usually come after a few years of experience in any other programming languages (most often from the world of PHP). Therefore, even a novice Ruby programmer is an experienced web developer with plenty of knowledge and experience. For the Ruby language the most popular web framework is Rails, over 90% of web apps written in Ruby, use Rails. Experienced Rails developers also report that it makes web application development more fun.[33]

The Rails philosophy includes two major guiding principles:

- **Do not Repeat Yourself**: the framework provides mechanisms to reuse the code. This allows not only minimizing code duplication and increasing the speed of

[32] Rails Guides, *Ruby on Rails* [online] [cit. 2016-12-20]. Available from: http://guides.rubyonrails.org/getting_started.html
[33] Wikipedia, *Ruby on Rails* [online] [cit. 2016-12-20]. Available from: https://en.wikipedia.org/wiki/Ruby_on_Rails

development.

- **Convention Over Configuration**: by default, the framework uses numerous agreements on configuration, typical for most applications. This greatly simplifies the creation of applications, as explicit configuration specification is required only in unusual cases. [34]

Ruby on Rails framework extensibility: Around Ruby on Rails has developed a large ecosystem of plug-ins open-source ("jams", gems) that implement the most requested features. "Jams" are very different: from low-level responsible for some aspect of the inner working in the applications to higher-level, which represent the individual modules to address a range of business tasks. Using the system plug-ins served as the reason for the popularity of the framework - the ability to connect separate components and libraries greatly speeds up the development. [35]

Ruby on Rails may not work with all servers that support FastCGI: Apache, Lighttpd, and SCGI. During development, often the easiest way is to use the WEBrick Web server that comes with Ruby. As a database server, Ruby on Rails supports MySQL, Firebird, PostgreSQL, IBM DB2, Oracle, Microsoft SQL Server and embedded database - SQLite. [36]

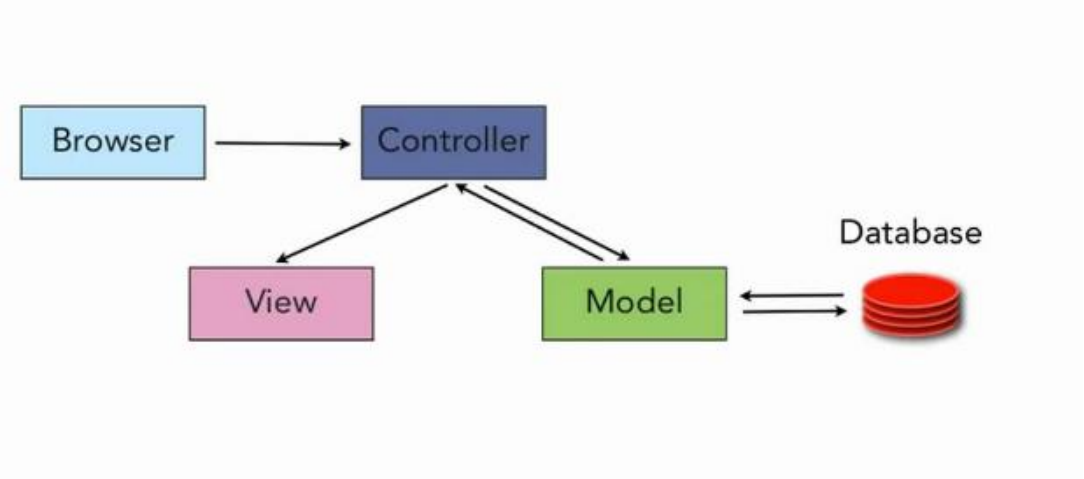
The Model-view-controller Architecture.

MVC is a pattern for the architecture of a software application. It separates an application into the following three components:

- Models for handling data and business logic. The application logic in the model performs two important functions: returns information about the state of the application and changes the application state.
- Controllers, for handling user interface and application logic.
- Views, for handling graphical user interface objects and presentation logic[37]

[34] Nascenia, Ruby on Rails principles [online] [cit. 2016-12-21]. Available from: http://www.nascenia.com/ruby-on-rails-development-principles/
[35] Rails, The Rails Doctrine [online] [cit. 2016-12-21]. Available from: http://rubyonrails.org/doctrine/
[36] Rails Guides, *Ruby on Rails* [online] [cit. 2016-12-20]. Available from: http://guides.rubyonrails.org/getting_started.html
[37] Sitepoint, Rails architecture [online] [cit. 2016-12-20]. Available from: https://www.sitepoint.com/learn-ruby-on-rails-8/

Figure 8: Processing a page request in an MVC architecture



Source: *Ruby on Rails architecture* [online] [cit. 2016-12-21]. Available from: https://www.lynda.com/Ruby-Rails-tutorials/Welcome/55960/75149-4.html?utm_medium=viral&utm_source=youtube&utm_campaign=videoupload-55960-0103

Rails consists of two logical components—ActiveRecord and ActionPack, the first implements the presentation layer i.e. model, ActionPack combines the other two levels of MVC architecture controller and view.

ActiveRecord provides the ability to view the database as objects. This creates a domain model, which merges the data itself and ways to manage them. Following the standards of the ORM–model (Object-relational mapping), the table is represented by a class, row is object, field is attributes of an object. With the dynamic typing in Ruby language developers can inherit model class from the base class ActiveRecord::Base. [38]

ActionPack contains two parts: ApplicationController responsible for the implementation of the business logic, and ActionView—the view engine. ActionController is the Manager of the controllers in a Rails application. He manages the application logic of the program, acting as a bridge between presentation layer and a Web browser. A controller in Ruby on Rails is a class inherited from ActionController::Base. Public methods of a controller called actions. [39]

Action Controller is responsible for the following tasks:

[38] Sitepoint, Rails architecture [online] [cit. 2016-12-20]. Available from: https://www.sitepoint.com/learn-ruby-on-rails-8/
[39] Sitepoint, Rails architecture [online] [cit. 2016-12-20]. Available from: https://www.sitepoint.com/learn-ruby-on-rails-8/

- The decision about how you should handle a particular request;
- Fetching data from the model to pass it to the view;
- Receiving information from the user queries and uses it to create or modify data in the model. [40]

ActionView (View). In Rails the view handles everything that sent to the browser. The views are the page templates that use RHTML (HTML with embedded Ruby) or RXML (XML, generated by Ruby). Action View manages rendering of nested and partial templates, and includes AJAX support.[41]

Ruby on Rails provides useful features:

- Schema Migration. In Rails, there is a possibility of any change in the database structure with using migrations. In addition, migration can used to return the database structure to a previous version.
- The build tool Rake. Rake - a tool for automating build of software code. It has a library of basic tasks, such as functions for manipulating files and a library to remove compiled files.
- A custom URL. Rails clearly separate the URL from the file names, method names and other sensitive internal details of your application. URL in Rails is simple and clear, not long and cryptic. [42]

Myths about Ruby language and Ruby on Rails framework:

- "No developers". Myth. There are developers. Of course, they are fewer than in PHP.
- "Very expensive". Myth. Good web programmers in General are expensive, regardless of language and platform development.
- "Slow" and "Unscaled". Myths. GitHub, Groupon, Basecamp, Twitter, Lenta.ru and many more projects with thousands of attendance used Rails.

Why Ruby on Rails can choose for developing web application or site?

- High-speed development - projects on Rails developed really faster than in PHP or Java. The technical features of the framework allow it (for example simplifying the configuration) and development tools (command line utilities and generators, ready

[40] RailsGuides, *Action controller* [online] [cit. 2016-12-22]. Available from: http://guides.rubyonrails.org/action_controller_overview.html
[41] RailsGuides, *Action view* [online] [cit. 2016-12-22]. Available from: http://guides.rubyonrails.org/action_view_overview.html
[42] Linux Journal, *Rails features*[online] [cit. 2016-12-22]. Available from: http://www.linuxjournal.com/content/ruby-rails-features-railsonrubycom

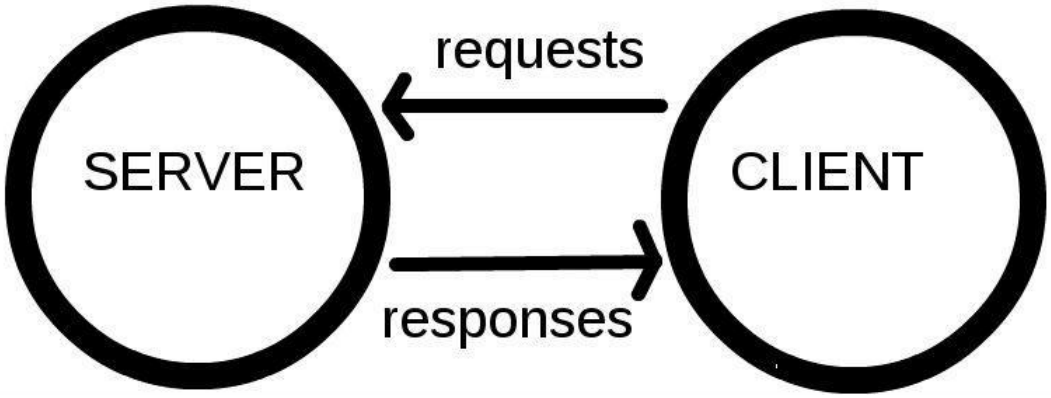
libraries, extensions and modules). Development time is Customer's money, the more time it takes to develop - the more expensive it is.

- Complex business logic is easier. Coding on the base of the Rails allow you to write clear code that can easily be modified in timing. Program code could be followed not just the original developer, but also by any other developer or any other team.
- Scalability, performance and high load. Rails application deployed and working fine in server clusters or in the "clouds".
- Competent developers - the proportion of good programmers in Ruby is much higher than in PHP.[43]

4.2 Web services

Computers connected to the network called clients and servers. A simplified scheme of how they interact may look like:

Figure 9: Client–server model



Source: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works

A browser is the program with which people use the web: browsing websites or using web applications like Google Mail, Twitter, Facebook. To display the user website or web application, the browser will obviously need some way to receive information from the Internet what it is going to display. To do this, it makes the web request and gets the response. The request in most cases occurs on the Protocol http (hypertext transfer Protocol). The response, of course, comes with the same Protocol and usually contain, in addition to any proprietary

[43] RailsCarma, *Myths about Rails* [online] [cit. 2016-12-22]. Available from: http://www.railscarma.com/blog/technical-articles/busting-ruby-on-rails-myths/

information, desired resource presented in HTML format (HyperText Markup Language). All this is happening over a TCP connection, which established by the browser and a web server using DNS (Domain Name System). [44]

The browser uses DNS to determine the Internet address of the server on which the desired resource is located. The browser picks received from the system DNS address, and connects to the computer that located in this address. Using the established TCP connection, browser requests the server of needed resource. The server sends the browser a response that contains the requested resource (usually).[45]

[44] Jeff Knupp, *What is Web* [online] [cit. 2017-01-15]. Available from: https://jeffknupp.com/blog/2014/03/03/what-is-a-web-framework/
[45] *How web works* [online] [cit. 2017-01-15]. Available from: http://www.garshol.priv.no/download/text/http-tut.html/

5. Practical Part

5.1 Main idea

The main idea for the practical part of bachelor thesis is creating the web application, which provides users information about cultural events in Prague. Intended audience is tourist and English-speaking community in Prague, which does not understand Czech language, but wanted to find something interesting in Prague. While creating the application I tried to make it in user-friendly way, provide some pages for help and create intuitively clear design.

Thinking about my future auditory, I stared to plan the application, which will have next functionality:

- Web application will give users possibility to create their own account
- Web application will give users possibility to log in and log out, and manage their own accounts;
- Web application will give users possibility to read, create and comment articles;
- Web application will protect articles from unregistered users;
- Web application will provide some information about itself in special Help page.

For the tool, I chose Ruby on Rails framework. I was not familiar before with frameworks and Ruby language, but I wanted to try something new and modern. When I started with Ruby on Rails, I was not experienced with Linux operation system, and for the platform, I chose Windows 7 OS, which not recommended. However, my goal is to learn basics and create small web application. Looking ahead, I had some difficulties because of platform, but I was always able to found the solutions on forums or from the book.

For the study material, I used book "Ruby on Rails tutorial: Learn Web Developments with Rails"[46]

For the installing Ruby on Rails I used console, and write down command ***gem install rails***. After finishing, I created needed directory and framework automatically generated application and installed all Gems, which was in **Gemfile**. These files allow specifying which dependencies on the gems needed for Rails. Ruby on Rails created main folders for the application:

[46] Michael Hartl. Ruby on Rails tutorial: Learn Web Developments with Rails. Addison-Wesley, 2013. ISBN: 0321832051

- App – includes all controllers, resources, views, models for the app.
- Public – there will be materials, which available for everyone. For example pictures.
- Tmp – temporary files.
- Db- database.
- Etc.

5.2 Prototype and main page creation

First, I started with creating prototype of the page. For this purpose, I used free online service **gomockingbird.com**. The service allows creating web and mobile prototypes, sharing it with friends and working on the project with friends at the same time!

Figure 10: Main page prototype

In the top of the page, I placed navigation bar with links to another pages and same bar for the footer. Moreover, I placed registration button on the middle of main page. I will give users link to registration in login page.

Ruby on Rails already includes special design instrument – sass. I used it for the style my

sheets with my own css file.

First, I create controller for the main page with ***ruby bin/rails generate controller*** command.

Figure 11: Controller code

```
def home
end

def contact
  end

def articles
end

 def about
end


def help
end
```

Main page will be static with the links to other pages, so I just defined them inside of controller and that is all.

Table 3: View for the main page

```
<img class="prague_pic" src ="prague.jpg" alt="Prague">
                                <div class="center jumbotron">
                                <h1>News about cultural events in Prague</h1>
                                <br>
        <h3 class="indent">Text for the main page</h3> <br>
                                <h3 class="indent">Text for the main page </h3>


                                <br><br>
            <h4>   <%= link_to "Sign up now!", signup_path, class: "btn btn-lg btn-primary" %></h4>
          </div>
```

With the view, I provide design and content for the main page. I places image in the top

of the page right after navigation bar, which is inside of public folder. In addition, I provide some css classes like “indent” for <h3> tag. Sass tool takes care about “center jumbotron” style class, which I will use in all pages of my application. <h4> tag includes link to the registration page, which I placed inside of the button.

Ruby on Rails has layouts, which allows defining structure of the pages. Add header/body and footer for the all pages.

Table 4 Layouts for the web application (based on Ruby on Rails tutorial book)

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <%= stylesheet_link_tag "application", media: "all",
                                    "data-turbolinks-track" => true %>
    <%= javascript_include_tag "application", "data-turbolinks-track" => true %>
    <%= csrf_meta_tags %>
    <%= render 'layouts/shim' %>
  </head>
  <body>
     <%= render 'layouts/header' %>
     <%= render 'layouts/footer' %>
  </body>
</html>
```

Layouts code has HTML code structure, started and ended with <html> tag. In the header, there are links to the JavaScript and css files. Moreover, header includes paths needed for the sass tool. <%= render 'layouts/footer' %> in the browser page calls layout for the footer for the all pages in the application. Moreover, it not needed to write it all the time. The code in the footer looks like:

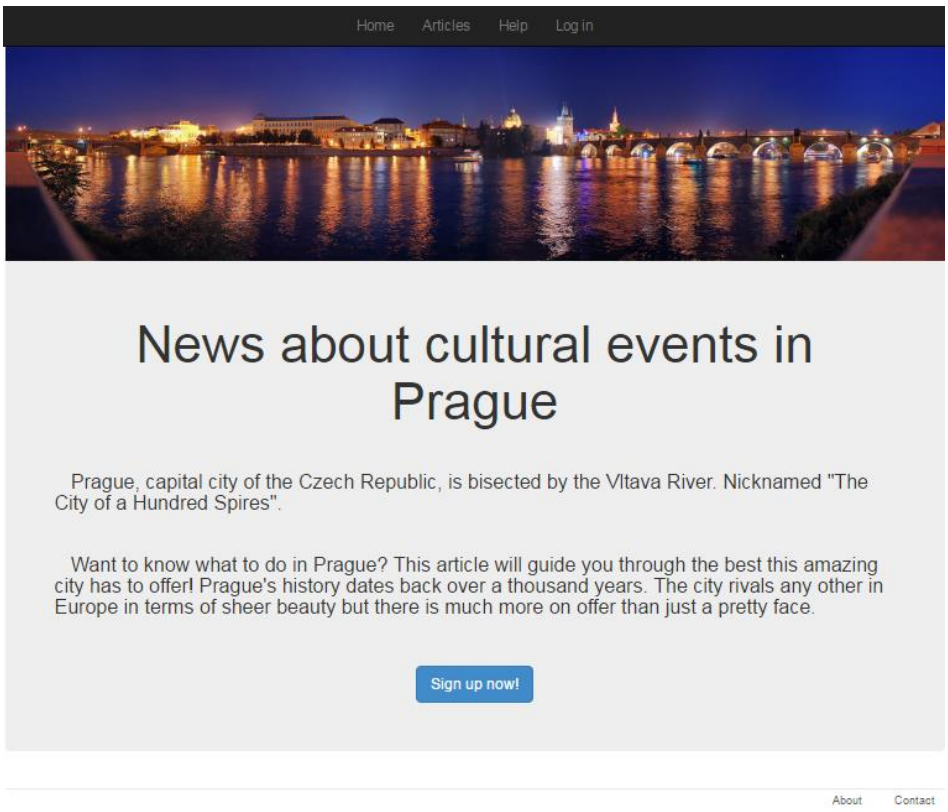
Table 5: Footer layout

```
<footer class="footer">
  <nav>
    <ul>
      <li><%= link_to "About", about_path%></li>
      <li><%= link_to "Contact", contact_path%></li>
    </ul>
  </nav>
</footer>
```

Sass tool has special class for the footed design, which includes <nav> tag inside and ordered list with <ul> and <li> tags, so I just used it for my code. Otherwise, it could solve with usual css way, with the class and some parameters for the ordered list.

After creation another static pages – “Help”, “About”, “Contact” where I just used sass class for the style and write down some information, my main page starts to look like:

Figure 12: Main page



5.3 Articles

Page with articles will provide users information about the events. Page should include list with events and give possibility to create own event.

Figure 13: Article page prototype

For the article page header and footer will be the same, I changed only body part. I added button for the new article creation, and placed already existed articles on the page.

Article page will have some functionality inside like calls form for creation article and comment. For this page, I will need to use database, which also require creation some models for connection database and controller.

As usual, for starting any logic in Ruby on Rails first needed to generate controller. After creation I define their needed methods, and during the coding, of course I make some changes, added more functionality. In the end controller for this page had a lot of code inside; let's have a look on small piece of it.

Table 6: Article controller (based on Ruby on Rails tutorial book)

```
class ArticlesController < ApplicationController
before_action :logged_in_user, only: [:new, :edit, :update, :create, :destroy]
  def destroy
    @article = Article.find(params[:id])
    @article.destroy
    redirect_to articles_path
  end
def logged_in_user
      unless logged_in?
        flash[:danger] = "Please log in."
        redirect_to login_url
      end
    end
end
```

Where in code line ***before_action :logged_in_user, only: [:new, :edit, :update, :create, :destroy]*** means that actions with articles like creation, edition and destroy is available only for registered users. Line ***unless logged_in?*** is the check is user is login or not, and if not

flash[:danger] = "Please log in." will show red message to the user with redirection to the login page.

Method ***destroys*** in the articles, will search in the database article with the current Id, destroy it and redirect user to the list with articles page.

In Ruby in Rails, there is special layer, which represents business data and logic, which calls models. Usually it is uses for all objects which data needs to be store into the database. The model creates connection between object and database automatically, and save itself information about how this data should behave and dependencies of the data. In addition, models allow using **form_for** method for creation form. I used this method for comment form and article form creation. It's small piece of code. The form creation is not mandatory, but In Ruby on Rails forms makes programmer's work much easier. It allowed calling them in any time when they needed with fully work functionality.

Table 7: Form for comment creation (based on Ruby on Rails tutorial book)

```
<%= form_for([@article, @article.comments.build]) do |f| %>
  <%= f.label :commenter %><br>
  <%= f.text_field :commenter %>
  <%= f.label :body %><br>
  <%= f.text_area :body %>
  <%= f.submit "Submit", class: "btn btn-primary" %>
<% end %>
```

It's look-like Windows form creation, for example in Visual Studio. Nevertheless, in VS programmer usually doing it manually - placing needed element on the form. In Ruby on Rails, I was doing it with the words. For example, ***<%= f.label :commenter %>
*** is placing label (just line of next) with name "commenter". ***<%= f.text_field :commenter %>*** line will place field, where user of web application will add his name. This field will save in the database.

Coming back to the models, in my application there is good example of the model, where I describe behavior articles and comments.

Table 8: Article model

```
class Article < ActiveRecord::Base
  belongs_to :user
      has_many :comments, dependent: :destroy
      validates :title, presence: true,
          length: { minimum: 4 }
end
```

This model describes, that all articles, which was created, belongs to the users - ***belongs_to :user***. And each article can have many comments ***- has_many :comments***. And each article must have a title - ***validates :title, presence: true***, with minimum length if 4 characters - ***length: { minimum: 4 }.***

For the comments, I created controller with the same way. I did not create view for the controller, because it not need - comments will be inside of the show_view for the article.

In the automatically generated by Ruby on Rails database, there is schema with table comments and articles. This contains fields, which I placed inside of comment and article forms (title and body) and automatically generated fields like date of creation and date of last modify.

Table 9: Database schema. Comments table.

```
create_table "comments", force: :cascade do |t|
  t.string   "commenter"
  t.text     "body"
  t.integer  "article_id"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
end
```

As table shows, a table ***comments*** includes title and body fields and foreign key – ***article_id***. Because model was describes relationship between article and comments. As in my form for the comments, there is the same name for the “column”. For example, ***body*** and

commenter. created_at and ***updated_at*** automatically generated by the Ruby on Rails.

I used ***created_at*** field in my view to represent date of comment.

According my prototype, page must contain only list of articles with links for update/delete or open it, and visible button for creation new article.

For this purpose, I created new view for show article, where will be possibility to open article on big screen, read comments and write own comment.

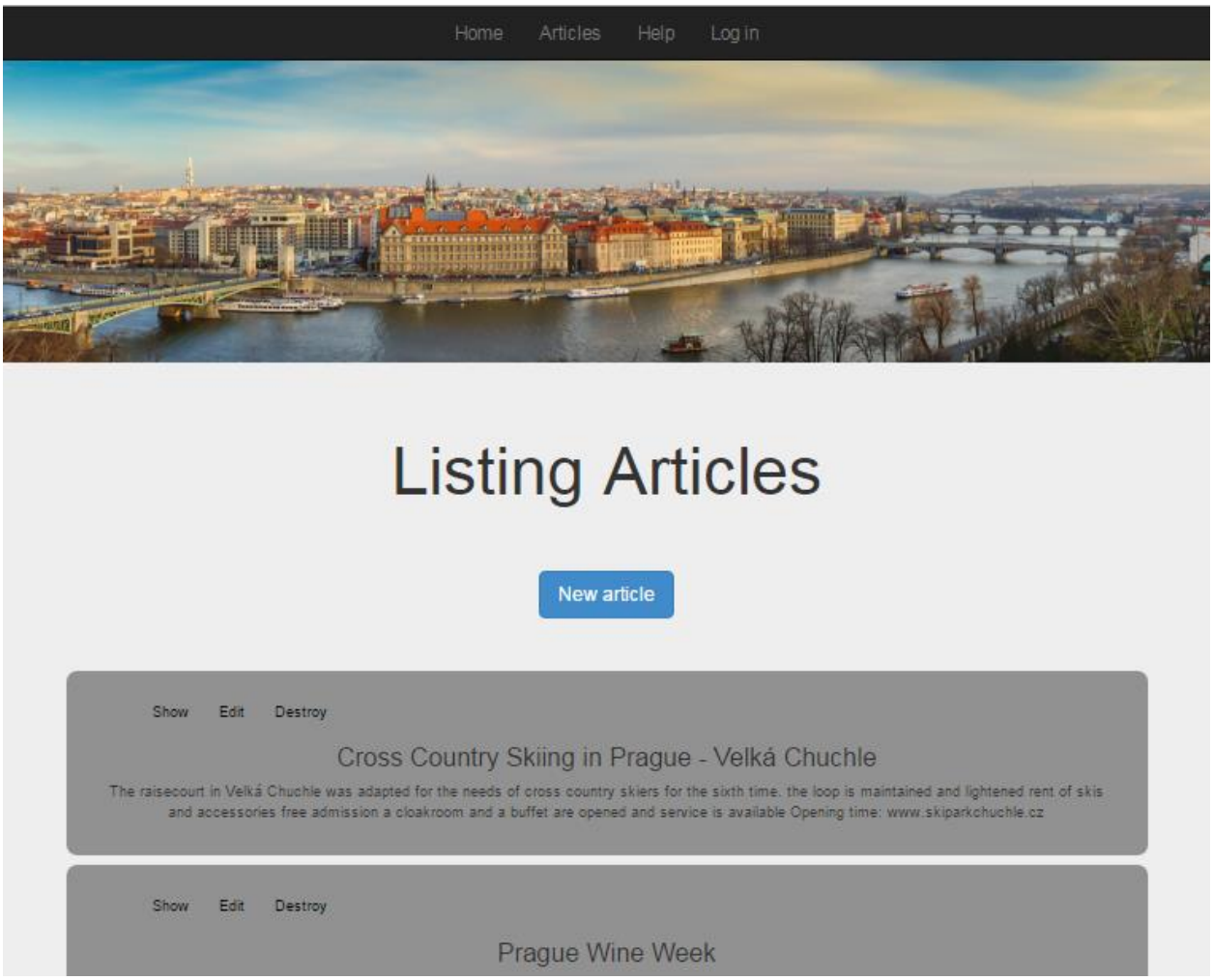
I decided to show article with reverse loop (to have new articles on top) and added some css style for them

Table 10: Article view

```
<h1>Listing Articles</h1>

        <br><br>

 <h4> <%= link_to 'New article', new_article_path, class: "btn btn-lg btn-primary" %> </h4>

        <br><br>


        <% @articles.reverse.each do |article| %>

  <div class="articles_list">

        <div class="articles_links">

        <ul>

        <li><%= link_to 'Show', article_path(article) %></li>

        <li><%= link_to 'Edit', edit_article_path(article) %> </li>

        <li><%= link_to 'Destroy', article_path(article),

method: :delete, data: { confirm: 'Are you sure?' } %> </li>

        </ul>

</div>

        <h3><%= article.title %></h3>

    <%= article.text %>
```

```
        </div>

<% end %>
```

Figure 14: Final look of the articles page



For the next web application update, I'm going to add possibility to add some pictures of the event and make it more formatting way.

5.4 Sessions

My next step of the application creation was make users possibility to register. I created controller, models (because I saved user information in the database) and views for registration and login pages.

I used example from the book, and connect my web service to the **Gravatar,** because I wanted to have avatars on my web application, and I still not so familiar with Ruby language to

code my own way of avatar loading.

So I already had in my application database with few users, but I wanted to teach to system that my user is online and using the service. Finally, I start to work with a sessions.

I asked Ruby on Rails generate Session controller, and with this session controller Rails generated helper module, which will help a controller in the future. Sometimes helpers are automatically included into the views, when programmer includes module into base class of all controllers.

I included calling my helper into mail controller with the line ***include SessionsHelper***.

Table 11: Sessions helper (based on Ruby on Rails tutorial book)

```
module SessionsHelper
def log_in(user)
session[:user_id] = user.id
  end
  def current_user
  end
def log_out
end
```

I defined some methods inside of helper, and code log_in method for the user. So, ***session[:user_id] = user.id*** will use user's browser and save there cookie with encrypted version of user's id. The cookie will be temporary and created with session's method, and information there will encrypt, it will save it from the hacker attack.

Table 12: Part of the code from session_controller file (based on Ruby on Rails tutorial book)

```
user = User.find_by(email: params[:session][:email].downcase)
   if user && user.authenticate(params[:session][:password])
     log_in user
     redirect_to user
```

```
else
  flash.now[:danger] = 'Invalid email/password'
```

After helper file created, I added lines ***log_in user*** and ***redirect_to user*** into session controller file. This small piece of code represent search in the database user, using email address and password. After my helper line of code and warning if user wrote incorrect data.

I make some changes into my navigation bar, with using sass tool, to have dropdown menu. This is easy, because sass provides special class in css for it - ***dropdown-menu***. Moreover, allow users to change their data into my web application, via creating view with form for updating their data into database.

5.5 Future development

In this part of the thesis, I want to mention some problems, which I improve into my web application:

- Remove Gravatar service from my project and allow users to add their avatars without any third services;
- Add some categories of the events and sort them by the category;
- Protect user post and comments to the way that only author and moderator can delete and update it;
- New design for the articles: more style, more structured information and pictures;
- Allow users to share articles with each other.

6. Conclusion

The invention of high-level programming languages, and their constant improvement and development, has allowed people to not only communicate with the machine and understand it, but to use computers for complex calculations in different areas: aircraft construction, medicine and economics.

Today, many average and big companies, has a teams of programmers with knowledge of various programming languages, to edit, change and modify programs, which used by the employees of the company. Moreover, I may suppose that labor market has demand with the knowledge and experience with various programming languages.

Every 256th day of the year is celebrated unofficial holiday - The programmer Day. This number (two to the eighth power) chosen because it is the number of integers that can be expressed using one byte.

According theoretical part of my thesis, I learned how fast computers start to be invisible part of our lives. Sometimes we do not even realize, about computations and features, which modern computers and programs provides to us. It was not always like this. Before, only high-educated people were able to work with computers and write first programs. Modern programming languages provide huge benefits compared to previous languages. They are more structured and provided an integrated development environment.

In the practical part, I programmed and implemented web application, using Ruby language and Ruby on Rails framework. Web application provides users information about cultural events that will be occurs in Prague, and possibility to create their own account and events. In addition, user can comment articles.

I achieved main goals and became familiar with Ruby language and framework, learned Ruby on Rails architecture and structure. Moreover, I used some design tools for style my pages with css.

I will continue to learn Ruby and make changes in my Web application, because right now there are a lot of things, which what I should work and learn how to do it, but the only way to learn a new programming language is to write programs on it.

7. Bibliography

1. Jean E. Sammet. Programming Languages: History and Future. Communications of the ACM, 1972.
2. *History of computer programming languages* [online]. [cit. 2016-11-16]. Available from: https://cs.brown.edu/~adf/programming_languages.html
3. Fandom, *John von Neumann* [online]. [cit. 2016-11-18]. Available from: http://religion.wikia.com/wiki/John_von_Neumann
4. Roundcrisis, *Past and Future* [online]. [cit. 2016-11-18]. Available from: http://www.roundcrisis.com/2016/01/03/Sammett-72/
5. WikiBooks, *Pascal Programming* [online]. [cit. 2016-11-19]. Available from: https://en.wikibooks.org/wiki/Pascal_Programming
6. *Dennis M. Ritchie* [online]. [cit. 2016-11-19]. Available from: http://www.bell-labs.com/usr/dmr/www/
7. *C++ and Java* [online]. [cit. 2016-11-19]. Available from: http://anhpopeye.blogspot.cz/
8. Peter Grogono. The Evolution of Programming Languages. Department of Computer Science Concordia University, 2002.
9. Techlopedia, *Low-Level Language* [online]. [cit. 2016-11-20]. Available from: https://www.techopedia.com/definition/3933/low-level-language
10. *Technology* [online]. [cit. 2016-11-20]. Available from: http://shilpikum.blogspot.cz/
11. Computing Concepts, *Programming Languages* online]. [cit. 2016-11-20]. Available from: http://highered.mheducation.com/sites/0072834110/student_view0/chapter13/programming.html
12. IMB knowledge center, *Assembler* [online]. [cit. 2016-11-20]. Available from: http://www.ibm.com/support/knowledgecenter/SSENW6_1.6.0/com.ibm.hlasm.v1r6.asma400/asmr102112.htm
13. Webopedia, *What is translator* [online]. [cit. 2016-11-18]. Available from: http://www.webopedia.com/TERM/H/high_level_language.html
14. *What is translator* [online]. [cit. 2016-11-18]. Available from:

https://www.reference.com/technology/computer-language-translator-1c3fb8a9ee90cfde#

15. *Roles of translator* [online]. [cit. 2016-11-18]. Available from: http://ecomputernotes.com/compiler-design/translators-and-its-type
16. Fried space, *Assembly* [online]. [cit. 2016-11-28]. Available from: http://www.friedspace.com/assembly/intro.php
17. Lambda, *What is compiler* [online]. [cit. 2016-12-10]. Available from: https://lambda.uta.edu/cse5317/notes/node3.html
18. Computer Notes, Interpreters [online]. [cit. 2016-12-10]. Available from: http://ecomputernotes.com/compiler-design/translators-and-its-type
19. Techwelkin, *Difference between compiler and interpreters* [online] [cit. 2016-12-14]. Available from: http://techwelkin.com/compiler-vs-interpreter
20. Ruby, *What is Ruby* [online] [cit. 2016-12-14]. Available from: https://www.ruby-lang.org/en/
21. Wikipedia, *Ruby Programming Language* [online] [cit. 2016-12-14]. Available from: https://en.wikipedia.org/wiki/Ruby_(programming_language)
22. Read and Write, *Ruby books* [online] [cit. 2016-12-14]. Available from: http://readwrite.com/2011/04/08/10-free-e-books-on-ruby-for-be/
23. Tiobe, *Whai is Tiobe index* [online] [cit. 2016-12-16]. Available from: http://www.tiobe.com/tiobe-index//
24. BlackDuck, *Statistic from BlackDuck* [online] [cit. 2016-12-16]. Available from: http://blog.openhub.net/compare_languages/
25. Code Project, *What is framework* [online] [cit. 2016-12-19]. Available from: https://www.codeproject.com/Articles/5381/What-Is-A-Framework
26. Infinity Web, *Framework advantages and disadvantages* [online] [cit. 2016-12-19]. Available from: https://nagbhushan.wordpress.com/2010/10/03/framework-advantages-and-disadvantages/
27. Rails Guides, *Ruby on Rails* [online] [cit. 2016-12-20]. Available from: http://guides.rubyonrails.org/getting_started.html

28. Wikipedia, *Ruby on Rails* [online] [cit. 2016-12-20]. Available from: https://en.wikipedia.org/wiki/Ruby_on_Rails

29. Nascenia, *Ruby on Rails principles* [online] [cit. 2016-12-21]. Available from: http://www.nascenia.com/ruby-on-rails-development-principles/

30. Rails, *The Rails Doctrine* [online] [cit. 2016-12-21]. Available from: http://rubyonrails.org/doctrine/

31. Sitepoint, *Rails architecture* [online] [cit. 2016-12-20]. Available from: https://www.sitepoint.com/learn-ruby-on-rails-8/

32. RailsGuides, *Action controller* [online] [cit. 2016-12-22]. Available from: http://guides.rubyonrails.org/action_controller_overview.html

33. RailsGuides, *Action view* [online] [cit. 2016-12-22]. Available from: http://guides.rubyonrails.org/action_view_overview.html

34. Linux Journal, *Rails features*[online] [cit. 2016-12-22]. Available from: http://www.linuxjournal.com/content/ruby-rails-features-railsonrubycom

35. RailsCarma, *Myths about Rails* [online] [cit. 2016-12-22]. Available from: http://www.railscarma.com/blog/technical-articles/busting-ruby-on-rails-myths/

36. *How web works* [online] [cit. 2017-01-15]. Available from: http://www.garshol.priv.no/download/text/http-tut.html/

37. Michael Hartl. Ruby on Rails tutorial: Learn Web Developments with Rails. Addison-Wesley, 2013. ISBN: 0321832051

8. List of figures

9. List of tables